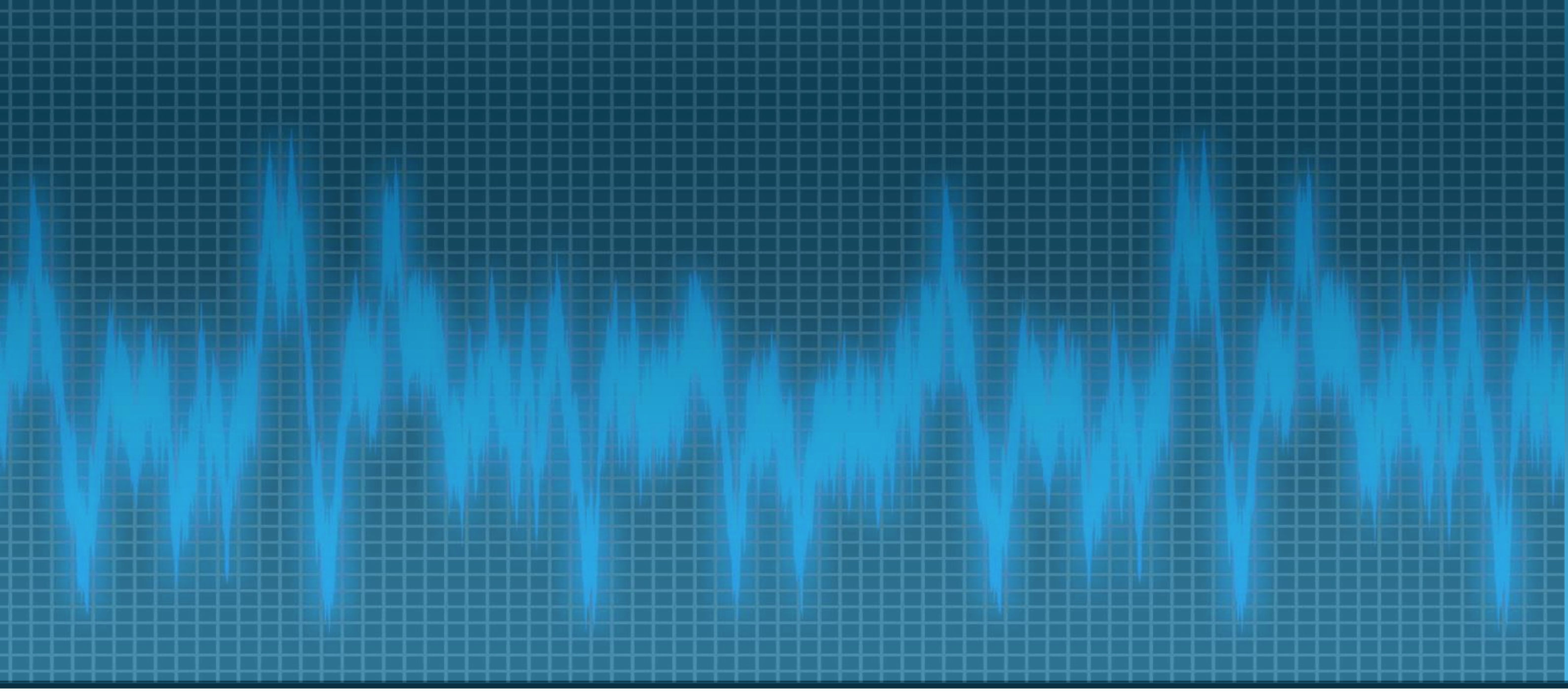


Compte Rendu TP1

Filtrage numérique d’un signal d’entrée

Réalisé par : Hachem Squalli ElHoussaini N°29

Dirigé par : Pr. H. TOUZANI

LAB 1 :

Exercice :

I.

1. Calculer, sur papier, la réponse impulsionnelle (RI) en fonction de a, sachant que le système est causal. Les conditions initiales sont nulles.

La réponse impulsionnelle est donc :

$$h[n] = a^n \cdot u[n]$$

2. Sous python, consulter l'aide de la fonction ***lfilter*** par *help(lfilter)*, et tâcher d'en comprendre le fonctionnement.

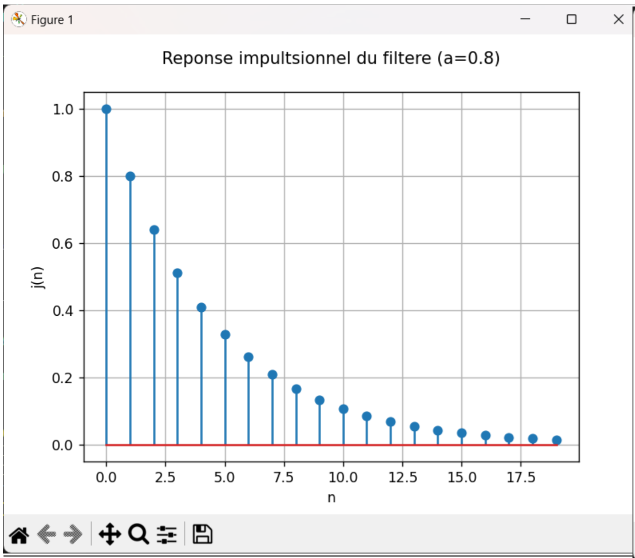
```
>>> import scipy.signal as ss
>>>
>>> help(ss.lfilter)
Help on function lfilter in module scipy.signal._signaltools:

lfilter(b, a, x, axis=-1, zi=None)
    Filter data along one-dimension with an IIR or FIR filter.

    Filter a data sequence, `x`, using a digital filter.  This works for many
    fundamental data types (including Object type).  The filter is a direct
    form II transposed implementation of the standard difference equation
    (see Notes).

    The function `sosfilt` (and filter design using ``output='sos'``) should be
    preferred over `lfilter` for most filtering tasks, as second-order sections
    have fewer numerical problems.

    Parameters
    ----------
    b : array_like
-- Suite  --
```

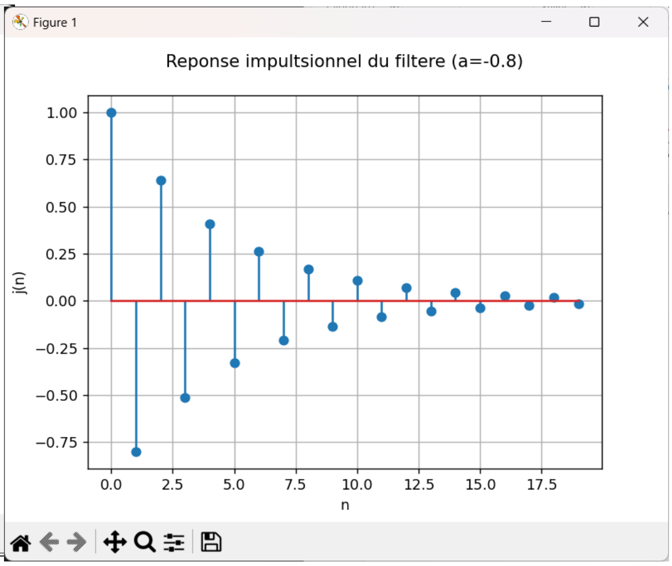
3. Calculer et visualiser la réponse impulsionnelle pour a = -0.8 a = 0.99 et a = 1.01. Il pour être utile de définir une fonction qui rend directement la réponse impulsionnelle. Conclusions.

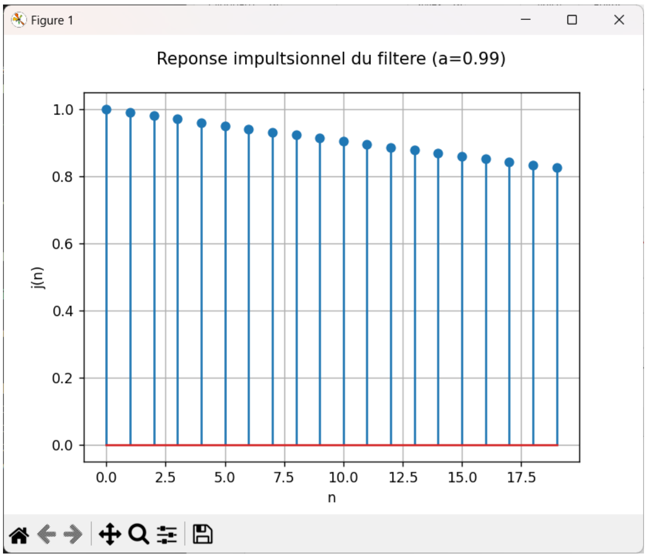
=> Quand a>1, il faut éviter ce filtre car il va amplifier le signal d'entré. => a doit être entre 0.8 et 0.1

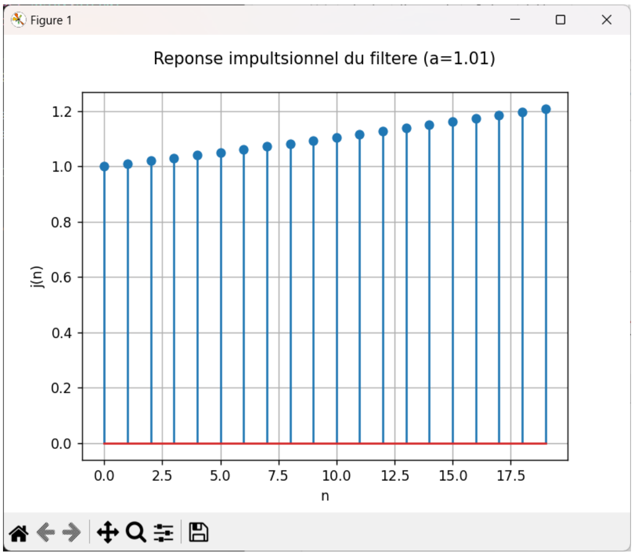
```python
import numpy as np
import scipy.signal
import matplotlib.pyplot as plt

#-----------------etude temporel----------------------
N=20

dirac = np.zeros(N)
dirac[0] = 1

b=[1] #numerator


for i in [0.8,-0.8,0.99,1.01]:
    a = i

    den = [1,-a]#denominator

    h =  scipy.signal.lfilter(b,den,dirac)

    plt.stem(range(N),h)


    plt.title(f"Reponse impultsionnel du filtere (a={i})", pad=20)
    plt.xlabel('n')
    plt.ylabel('j(n)')
    plt.grid(True)
    plt.show()

```

Résultat









II. Etude fréquentielle

1. Donner l'expression de la fonction de transfert en z correspondant à cette équation aux différences.

$$H(z) = \frac{1}{1 - ae^{-j2\pi f}}$$

2. Donner l'expression de la fonction de transfert H(f), puis de |H(f)| pour 'a' quelconque.

$$|H(f)| = \frac{1}{\sqrt{1 - 2a\cos(2\pi f) + a^2}}$$

3. Préciser les amplitudes théoriques en f=0 et f=1/2

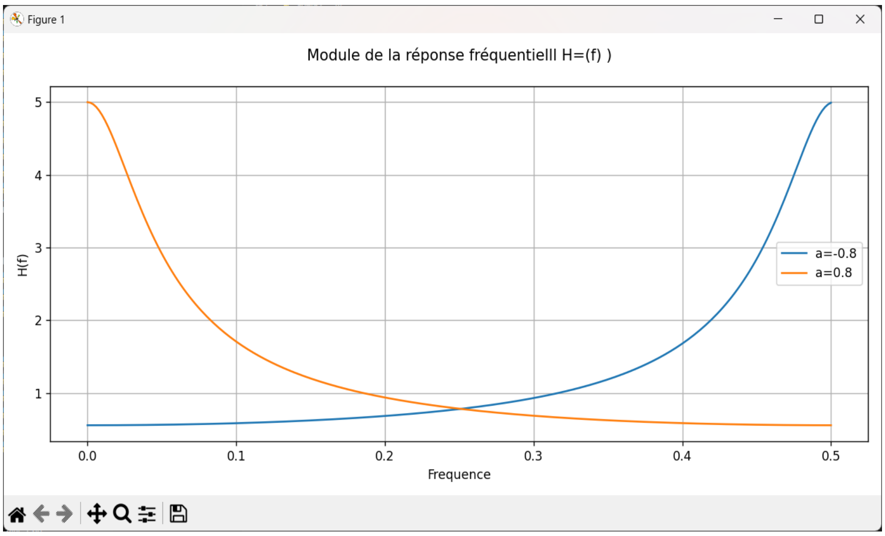
Pour f = 0 :

$$|H(0)| = \frac{1}{|1 - a|}$$

Pour f = 1/2 :

$$|H(1/2)| = \frac{1}{|1 + a|}$$

```python
import numpy as np
import scipy.signal
import matplotlib.pyplot as plt


fe = 32
Te = 1/fe
a_values = [-0.8,0.8]
N=512

plt.figure(figsize=(10,5))
for a in a_values:
    #impulsion de dirac
    dirac = np.zeros(N)
    dirac[0] =1

    #Réponse impulsionnellle h(n)
    b =[1]
    den = [1,-a]#denominator
    h =  scipy.signal.lfilter(b,den,dirac)

    #Transformee de fourir
    H =np.fft.fft(h)
    H = np.abs(H[:N//2]) #module + demi spectre (fréquence réelle)
    f =np.linspace(0,0.5,N//2) #axe des fréquences


    plt.plot(f,H,label=f'a={a}')



plt.title(f"Module de la réponse fréquentielll H=(f) )", pad=20)
plt.xlabel('Frequence')
plt.ylabel('H(f)')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```



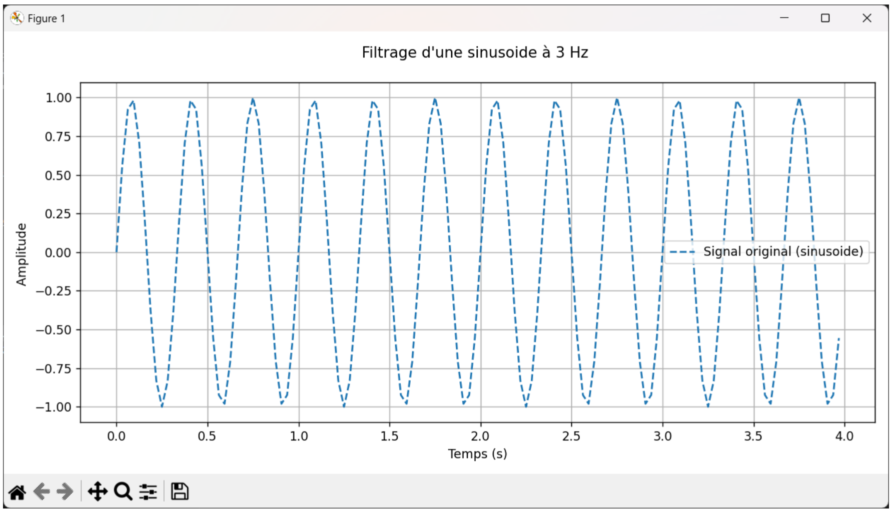
4. Le coefficient 'a' n'influence pas non seulement la pente de la courbe, mais encore le type de filtrage !

III. Filtrage :

$$y(n) = a.y(n-1) + x(n)$$

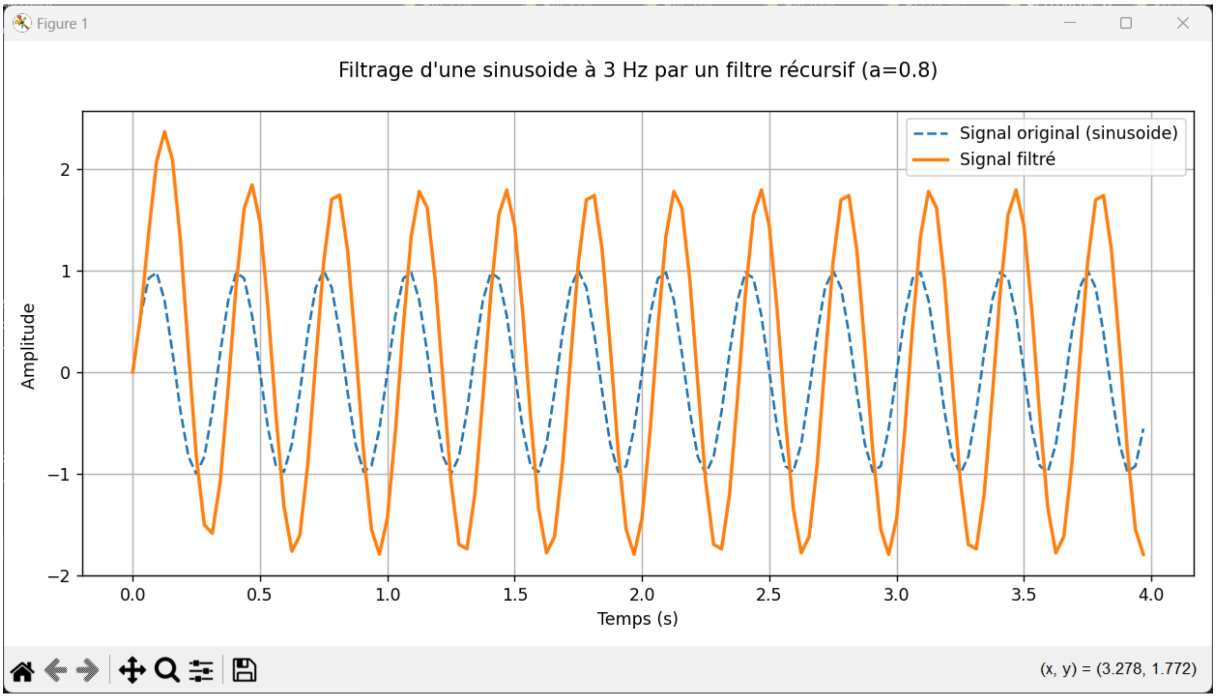
1. Créer une sinusoïde x, à la fréquence fo = 3, échantillonnée à Fe = 32, sur N = 128 points

```python
import numpy as np
import scipy.signal
import matplotlib.pyplot as plt

N=128
fo=3
Fe=32
a=0.8

t= np.arange(N)/Fe


x = np.sin(2*np.pi*fo*t)




#application de filtre récursif
y1 = scipy.signal.lfilter([1],[1,-a],x)



#affichage
plt.figure(figsize=(10,5))
plt.plot(t,x,label="Signal original (sinusoide)",linestyle="--")

plt.title(f"Filtrage d'une sinusoide à 3 Hz ", pad=20)
plt.xlabel('Temps (s)')
plt.ylabel('Amplitude ')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



2. Filtrer cette sinusoïde par le filtre précédent en utilisant la fonction *lfilter :* **=>** le filtre IIR représente un retard car il calcul les y(n-1) et présente une amplification de signal car il fait une accumulation des entrées passées !

```python
import numpy as np
import scipy.signal
import matplotlib.pyplot as plt

N=128
fo=3
Fe=32
a=0.8

t= np.arange(N)/Fe


x = np.sin(2*np.pi*fo*t)




#application de filtre récursif
y1 = scipy.signal.lfilter([1],[1,-a],x)



#affichage
plt.figure(figsize=(10,5))
plt.plot(t,x,label="Signal original (sinusoide)",linestyle="--")
plt.plot(t,y1,label="Signal filtré",linewidth=2)

plt.title(f"Filtrage d'une sinusoide à 3 Hz par un filtre récursif (a=0.8)", pad=20)
plt.xlabel('Temps (s)')
plt.ylabel('Amplitude ')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



3. Filtrer cette sinusoïde par le filtre précédent :

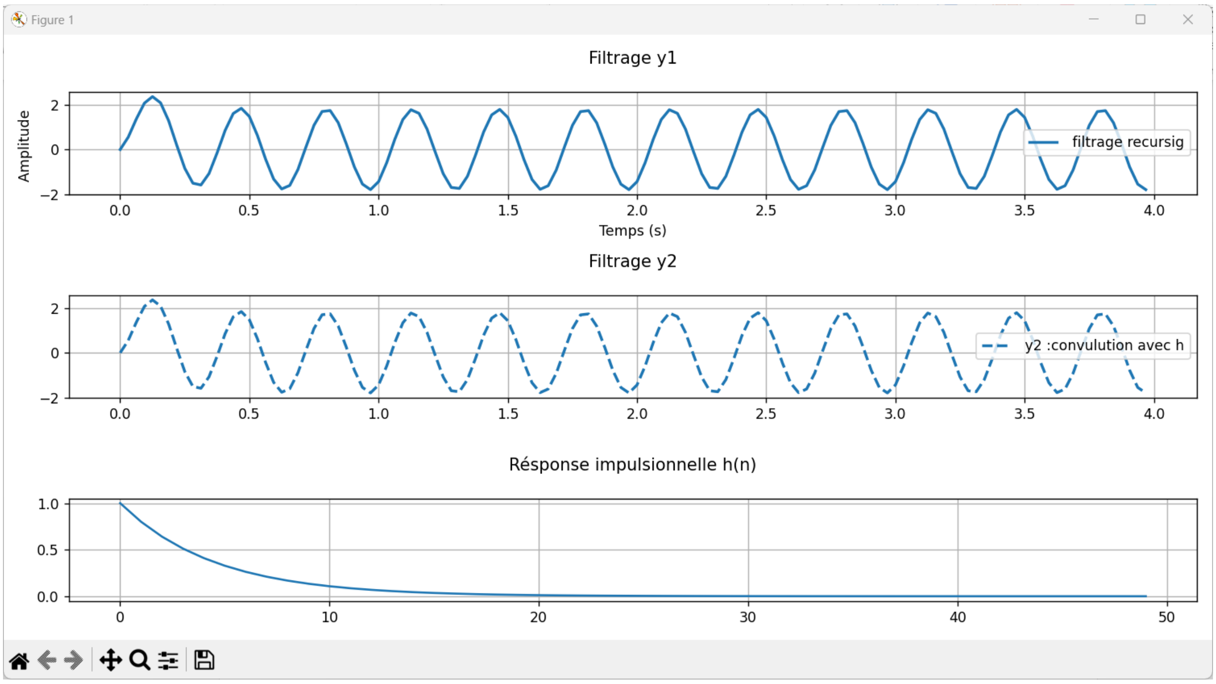
- En utilisant une convolution : y2 = lfilter(h, [1], x)

- Expliquer pourquoi ce dernier calcul correspond effectivement à une convolution.

Comparer graphiquement ces deux résultats. Afficher les deux courbes, voire la différence des courbes

```
#Filtrage par convulution (en utilisatn lfilter comme FTR avec h)
import numpy as np
import scipy.signal
import matplotlib.pyplot as plt
N=128
fo=3
Fe=32
a=0.8
L=50
t= np.arange(N)/Fe

h = a**np.arange(L)
x = np.sin(2*np.pi*fo*t)

#application de filtre récursif
y1 = scipy.signal.lfilter([1],[1,-a],x)
y2= scipy.signal.lfilter(h,[1],x)

#affichage des resultat

```

```python
#affichage
plt.figure(figsize=(12,6))

plt.subplot(3,1,1)
plt.plot(t,y1,label=" filtrage recursig",linewidth=2)
plt.title(f"Filtrage y1", pad=20)
plt.xlabel('Temps (s)')
plt.ylabel('Amplitude ')
plt.legend()
plt.grid(True)

plt.subplot(3,1,2)
plt.plot(t,y2[:N],"--",label=" y2 :convulution avec h",linewidth=2)
plt.title(f"Filtrage y2", pad=20)
plt.legend()
plt.grid(True)

plt.subplot(3,1,3)
plt.plot(np.arange(L),h)
plt.title(f"Résponse impulsionnelle h(n)", pad=20)
plt.grid(True)

plt.tight_layout()
plt.show()
```



Question

Les modules des transformées de Fourier du signal x et de la réponse impulsionnelle h :

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


N = 128
Fe = 32


t = np.arange(N)/Fe
x = np.sin(2*np.pi*3*t)


a = 0.8
h = a ** np.arange(N)


X_f = np.fft.fft(x)
H_f = np.fft.fft(h)
f = np.fft.fftfreq(N, d=1/Fe)


plt.figure(figsize=(12, 5))


plt.subplot(1, 2, 1)
plt.plot(f, np.abs(X_f))
plt.title('Module de X(f)')
plt.xlabel('Fréquence (Hz)')
plt.ylabel('|X(f)|')
plt.grid()


plt.subplot(1, 2, 2)
plt.plot(f, np.abs(H_f))
plt.title('Module de H(f)')
plt.xlabel('Fréquence (Hz)')
plt.ylabel('|H(f)|')
plt.grid()

plt.tight_layout()
plt.show()
```

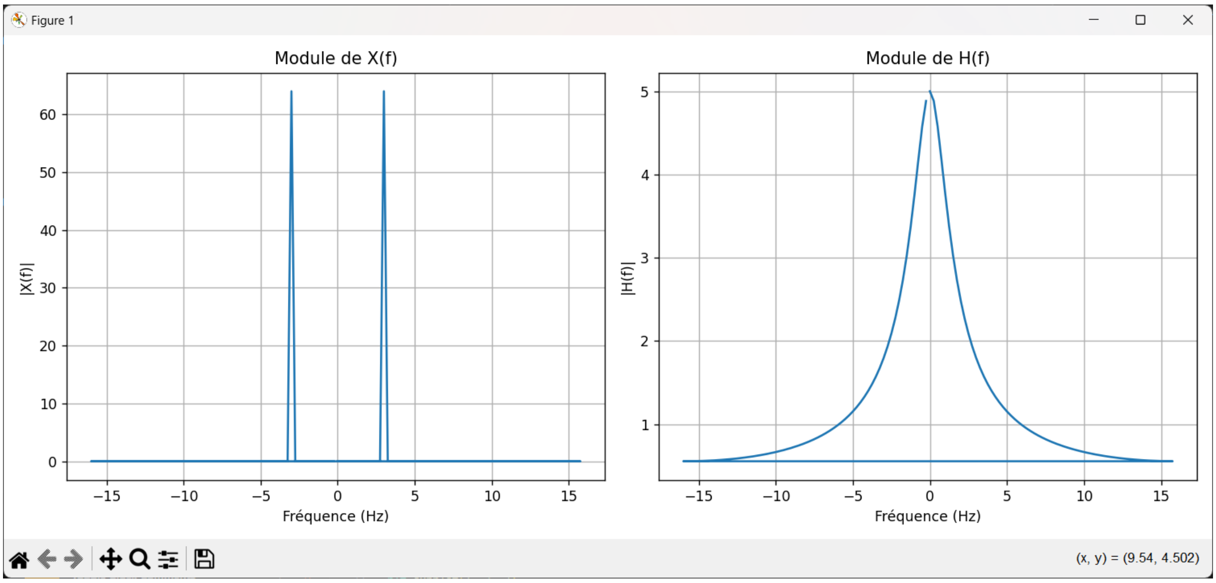


Figure 1
Module de X(f)
|X(f)|
Fréquence (Hz)
Module de H(f)
|H(f)|
Fréquence (Hz)
(x, y) = (9.54, 4.502)