1 Utilisation avec python

[10]:
```
%matplotlib inline

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

1.1 Répliement du spectre

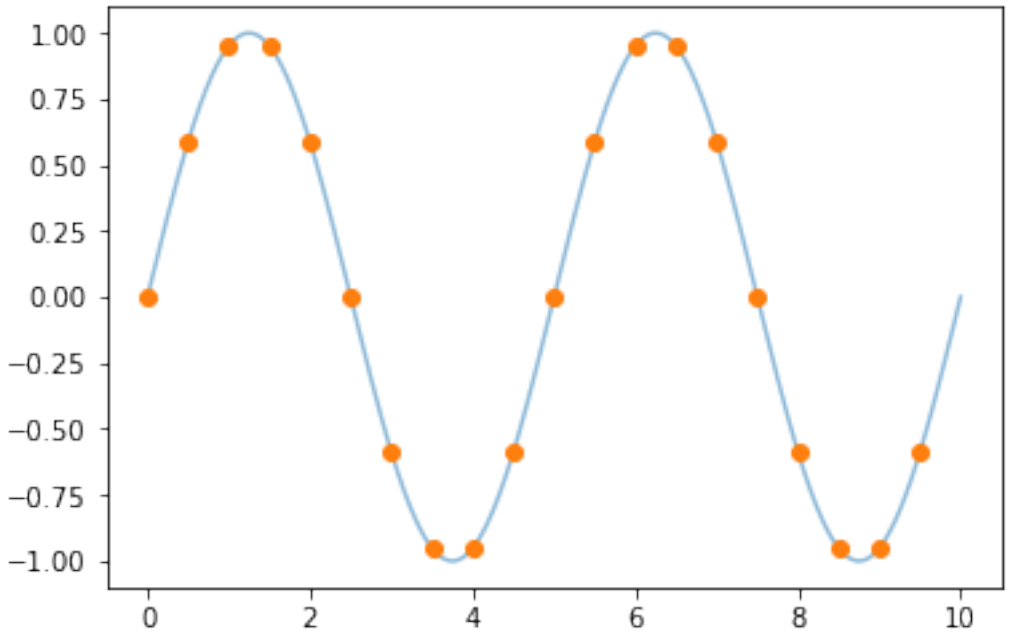
[30]:
```
T = 10
N = 20
dt = T/N
t = np.arange(N)*dt
print("Sample rate", 1/dt)

t_big = np.linspace(0, T, 10000)
```

```
Sample rate 2.0
```

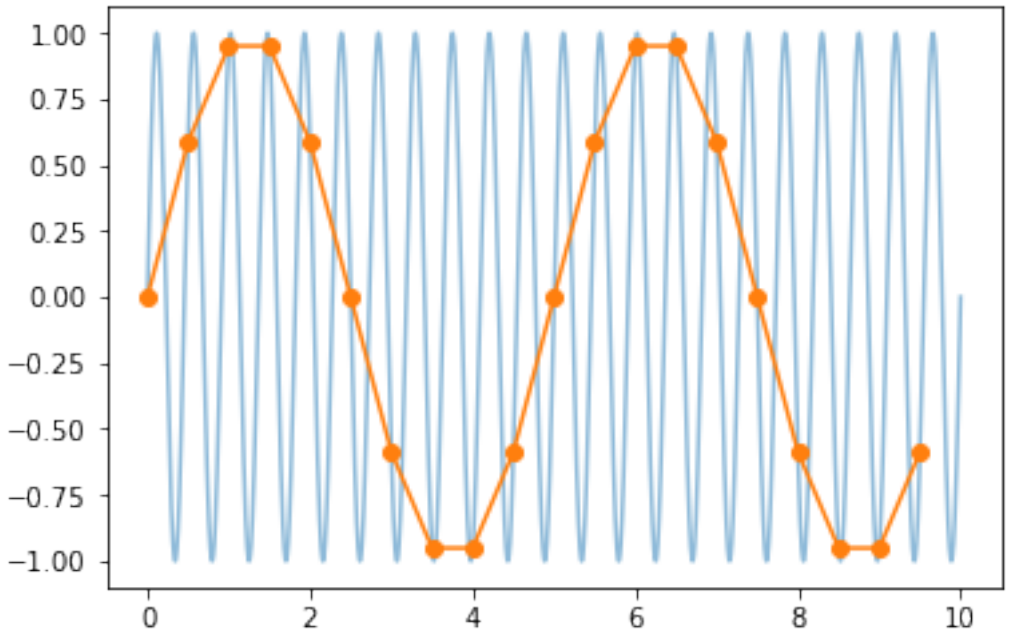
[31]:
```
freq = .2
signal = np.sin(2*pi*freq*t)
signal_big = np.sin(2*pi*freq*t_big)
plt.plot(t_big, signal_big, alpha=.5)
plt.plot(t, signal, 'o')
```

[31]:
```
[<matplotlib.lines.Line2D at 0x7f7b9f127110>]
```



[33]:
```
freq = 2.2
signal = np.sin(2*pi*freq*t)
signal_big = np.sin(2*pi*freq*t_big)
plt.plot(t_big, signal_big, alpha=.5)
plt.plot(t, signal, '-o')
```

[33]: [<matplotlib.lines.Line2D at 0x7f7b9f138f10>]



1.2 Filtres

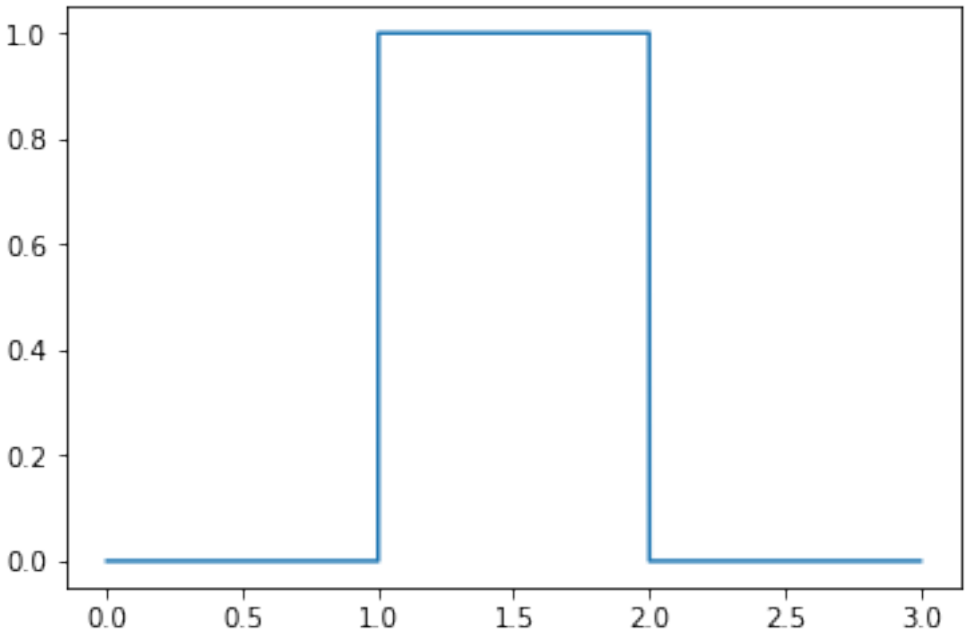
[ ]:
```
from numpy.fft import fft, ifft
```

[6]:
```
samplerate = 10000
delta_t = 1/samplerate

T = 3
N = T*samplerate


signal = np.zeros(N)
signal[10000:20000] = 1
t = np.arange(len(signal))*delta_t
plt.plot(t, signal)
```

[6]:
```
[<matplotlib.lines.Line2D at 0x7f7b9fa0b810>]
```

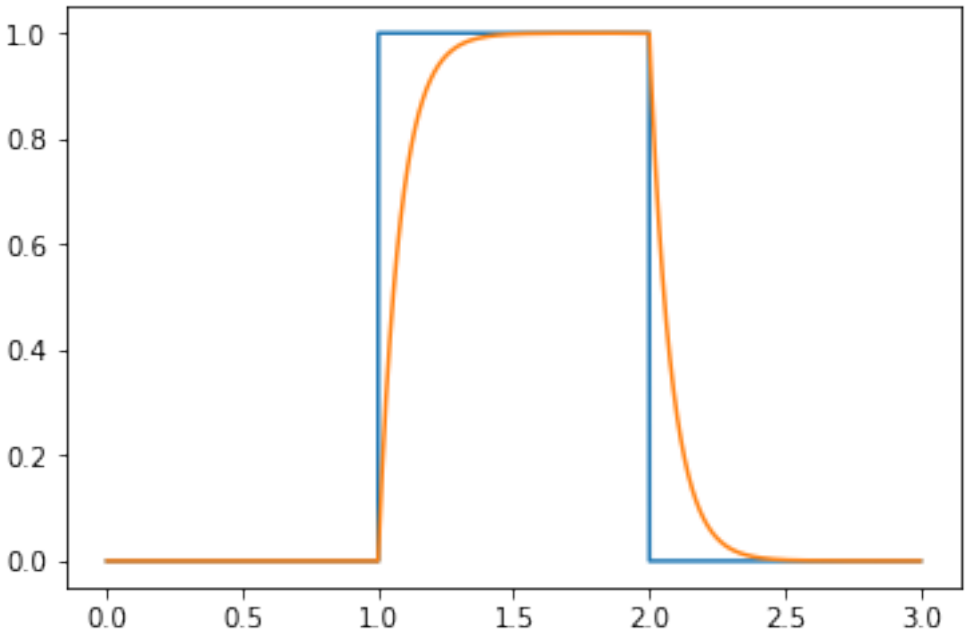


[7]:
```
# Filtre passe bas
tau = .5

signal_tilde = np.fft.rfft(signal)
freqs = np.fft.rfftfreq(len(signal), 1/samplerate)
H = 1/(1+1J*(freqs*tau))
signal_2 = np.fft.irfft(H*signal_tilde)


plt.plot(t, signal)
plt.plot(t, signal_2)
```

[7]:
```
[<matplotlib.lines.Line2D at 0x7f7b9f974490>]
```

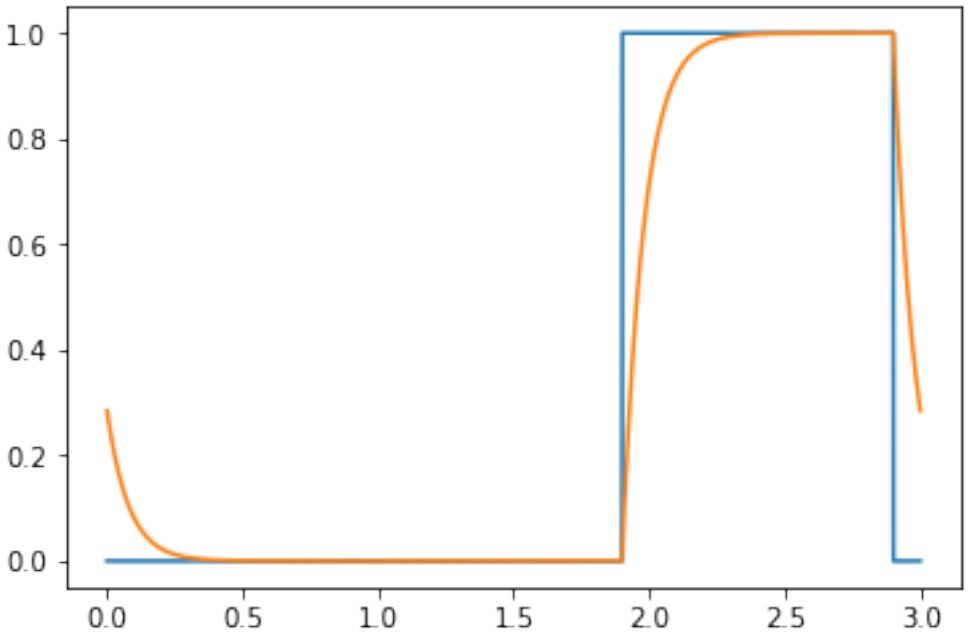


[8]:
```
signal = np.zeros(N)
signal[19000:29000] = 1


signal_tilde = np.fft.rfft(signal)
freqs = np.fft.rfftfreq(len(signal), 1/samplerate)
H = 1/(1+1J*(freqs*tau))
signal_2 = np.fft.irfft(H*signal_tilde)


plt.plot(t, signal)
plt.plot(t, signal_2)
```

[8]:
```
[<matplotlib.lines.Line2D at 0x7f7b9f95a4d0>]
```



1.3 Filtre sur une image

[40]:
```
from matplotlib.image import imread

image = imread('marguerite.jpg')
plt.imshow(image, cmap='gray')
```

[40]:
```
<matplotlib.image.AxesImage at 0x7f7bb03d7d90>
```

[44]:
```
from numpy.fft import fft2, ifft2, fftfreq
```

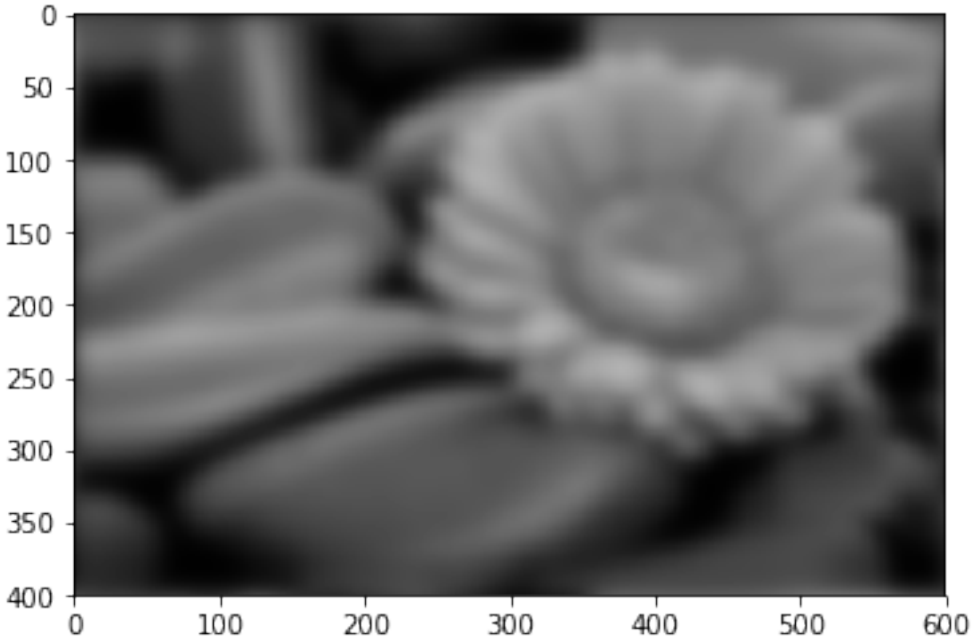
[59]:
```
image_tilde = fft2(image)

Delta_x = 50

freq_x = fftfreq(image.shape[0])
freq_y = fftfreq(image.shape[1])
freq_square = freq_x[:, np.newaxis]**2 + freq_y[np.newaxis,:]**2
filtre = np.exp(-freq_square*Delta_x**2/2)

image2_tilde = image_tilde*filtre
image2 = np.real(ifft2(image2_tilde))
plt.imshow(image2, cmap='gray', vmin=0, vmax=256)
```

[59]:
```
<matplotlib.image.AxesImage at 0x7f7b9c6e3150>
```


0
50
100
150
200
250
300
350
400
0
100
200
300
400
500
600

[ ]: