Python simulation of super-resolved acoustic Doppler imaging:

Table of content

- Abstract.
- Principles and theory.
- Modelization.
- Interpretation.
- Conclusion.
- Bibliography.

Abstract

We have designed and performed an acoustic experiment to create a new nonlinear imaging technique. Since the contributions of structured illumination and nonlinear fluorescent imaging, it has been demonstrated that it is possible to exceed the diffraction limit in the far field.

Here we implement a new method, based on the Doppler effect, to get rid of the contrast agents present in fluorescent imaging methods. In this article, we summarize the results of the Numpy modelization, we have also tried other python modelization using scipy and scikit learn who aren't discussed in this paper.

Using rotating sources and receivers, we have shown that this system is efficient, the spectral richness of the signal allows us to further increase the resolving power of our system, and thus to take a step towards super-resolution by obtaining spectral information that we would not have without the Doppler effect.

Principles and theory

The proposed idea is directly inspired by the STED method, which presents a depletion beam with a singularity at its center. Our concept consists in creating a Doppler effect with a singularity at a point, which we will try to isolate and thus observe with a finer resolution than the diffraction limit would allow. A sample of scatterers that we wish to image is rotated at the pulse Ω. The velocity of the various objects on the stage depends on their distance from the axis of rotation. Any object at a distance R from the center would have a tangential velocity $V = \Omega R$. The field scattered by this moving object should therefore be frequency shifted by the Doppler effect. A scatterer exactly at the center would have a zero velocity: the field scattered by this object would not be shifted in frequency, this point would then be singular, and would seem differentiated from the rest of the object to be imaged. By moving the center of rotation at each point, we hope to form a complete image. The speed of rotation would then be central in the power of resolution of this system: the faster the rotation, the steeper the singularity at the center. If two scatterers are too close to be discernible in the static regime, the rotation should induce a change in frequency of the responses between these two scatterers. If we can spectrally resolve these two waves. It is here the Doppler effect that would mark the space, and allow the super-resolution by singularizing the axis of rotation. By selectively filtering the received frequency, only the central point should be detected. It is therefore the ability to filter spectrally that would seem to limit our ability to separate two objects too close.

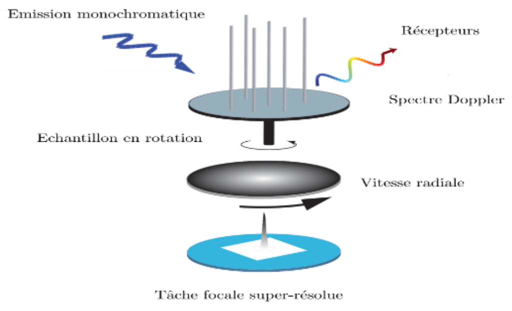


FIGURE IV.5 – Schéma de l'imagerie super-résolue basée sur l'effet Doppler proposée.

Modelization:

Some reminders of signal processing:

The idea was to reproduce the experiment as realistically as possible, we will see that we however had to deal with sampling problems, a problem that can be avoided by adopting an interpolation approach.

Continuous cosines:

Before moving on to rotating cosines and thus the introduction of the Doppler effect, we are interested in continuous cosines, i.e. the case where all microphones emit the same thing at the same time during the experiment. The goal is to make sure that the code works on an easier manipulation before introducing the Doppler effect.

The idea is the following:

We start by introducing the parameters of the experiment and the different useful libraries:

```
import numpy as np
import math
from math import cos
from math import sin
from math import sqrt
from math import pi
import matplotlib.pyplot as plt #condition implicite pour la suite: x<=L, et y<=l
c = 340 * 100 #Light speed in cm/s
t = 1 #duration of study of sinusoidal signals
fe = 5000 #sampling frequency
```

We define the space-time which is the list of the number of points to be studied during the experiment. The experiment lasts 1 second, this number of points is thus equal to the sampling frequency:

```
#Creation of the time scale which will be used for the sinusoidal inputs according to the desired period of analysis. it is necessary
def echantillonnagecos(t1 ,f):

T1=np.linspace(0, t1, int(t1 * f)) #Time scale to be used for sinusoidal inputs
return T1

T = echantillonnagecos(1, fe)
```

T is then our temporal list.

We will also need the distance of each loudspeaker from any point of the focal plane, so we implement the distance function to do this, which takes the x and y coordinates of a point of the focal plane as a variable and gives us a list containing the 32 distances of each loudspeaker from this point.

```
def distance(L, 1, r, x, y, F):
  X = np.zeros(32)
  Y = np.zeros(32)
  R = np.zeros(32)
  for i in range(32):
    X[i] = L/2 + r * cos(i * 2 * pi/32)
    Y[i] = 1/2 + r * sin(i * 2 * pi/32)
    R[i] = sqrt(((x - X[i])**2)+((y - Y[i])**2) + F**2)
return R
```

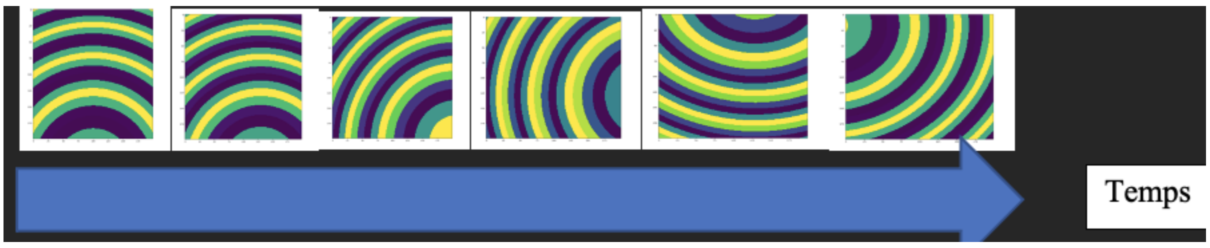
Rotating cosines:

We just have to change the emission matrix by putting zeros at the right places in order to simulate a rotation.

Here are the main helper functions for the modelization. The full script relative to this Numpy approach can be found on GitHub (https://github.com/Na00s/Python-simulation-of-super-resolved-acoustic-Doppler-imaging).

```
L = 200 # Length of the anechoïc chamber (cm)
l = 200 # Width of the anechoïc chamber (cm)
r = 150 # Radius of the sound rotation (cm)
x = 90 # Absciss of the measurement point on the focal plane (cm)
y = 120 # Ordinate of the measuremant point on the focal plane (cm)
F = 200 # Height of the focal plane (cm)
A = 5 # Amplitude of the sound signal (cm)
W = 6280 # Signal pulse

# Creation of the signal matrix, containing the signal of the 32 microphones
def createemissioncos(L, l, r, x, y, F, A, w):
  E=np.zeros((32,5000)) #list of functions, groups the signals sent by the microphones or the inputs
  for i in range(32):
    for j in range(5000):
      E[i][j]=(A * cos(w * T[j]))

  return E

W = createemissioncos(L,l,r,x,y,F,A,w)

# Measurement of the signal received in M (x, y, F) at tm
def signalcos(L, l, F, r, x, y, A, w, tm,  W): #tm is the measurement time between 0 and 1
  S=np.zeros(32)
  D=distance(L, l, r, x, y, F)
  for i in range(32):
    if ((tm - (D[i] / c)) * 5000) >= 0 and ((tm - (D[i] / c)) * 5000) < 5000:
      S[i] = W[i][int((tm - D[i] / c) * 5000)]
    else:
      S[i]=(0) #the signal has still not reached the point of interest
  return sum(S)

# Measurement of the signal received in the line x = 110 cm of the focal plane at tm
def cartographieligne(L, l, r, F, A, w, tm, W): #we choose to make the image of the line in the middle of the focal plane by traversin
  K = np.zeros(200)
  for j in range(200):
    K[j] = signalcos(L, l, F, r, 110, j, A, w, tm, W)
  return K

# Measurement of the signal received in the line x = 110 cm of the focal plane in function of time
def cartographielignetemporelle(L, l, r, F, A, w, W):
  G = np.zeros((5000,200))
  for i in range(len(T)):
    G[i] = cartographieligne(L, l, r, F, A, w, T[i], W)
  return G

# Measurement of the signal received on the whole focal plane at tm
def cartographie2Dinstant(L, l, r, F, A, w, tm, W):
  K=np.zeros((200, 200))
  for i in range(200):
    for j in range(200):
      K[i][j]=signalcos(L, l, F, r, i, j, A, w, tm, W)
  return K

# Measurement of the signal received on the whole focal plane in function of time
def cartographie2Dtemporelle(L, l, r, F, A, w, W):
  N=np.zeros((200,200,5000))
  for k in range(5000):
    for i in range(200):
      for j in range(200):
        N[i][j][k]=signalcos(L, l, F, r, i, j,  A, w, T[k], W)
  return N
```

We launched the temporal mapping (Temps means time in french) (matrix M of size 5000x200x200):



We then performed the Fourier transform of M in the following way:

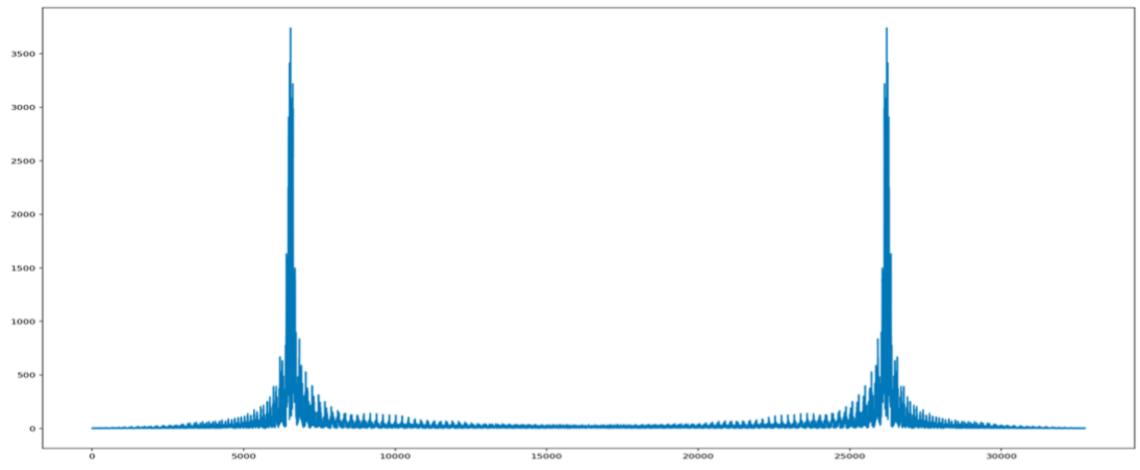
```
Mf = np.fft.fft(M, n=4096*8, axis=0)
```

Note that the FFT is done on a larger number of points to gain in resolution, the choice of the number of points is also due to the fact that the FFT programs are designed to work better with powers of 2.

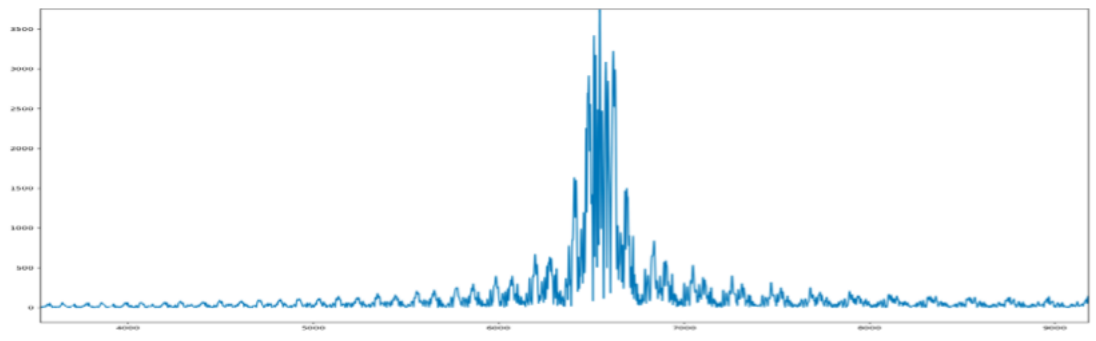
Interpretation

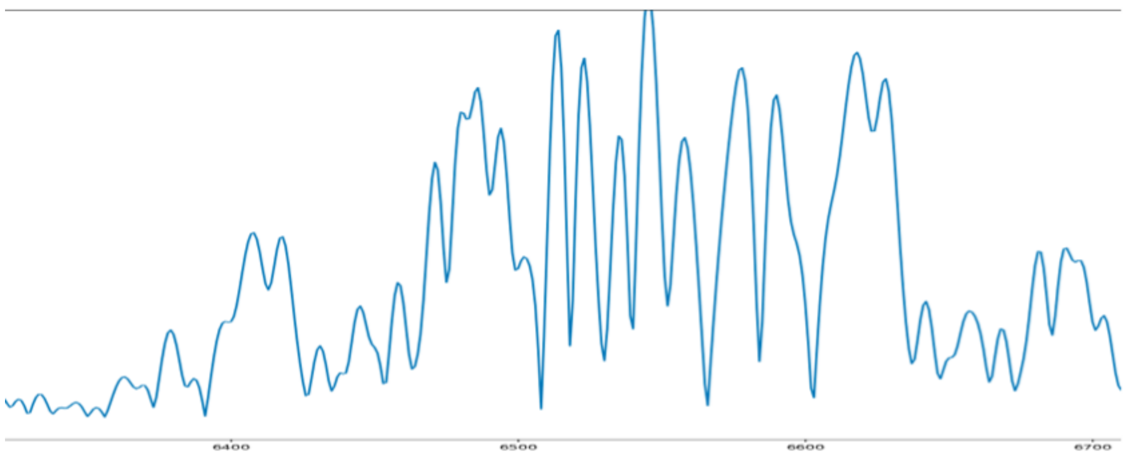
We then isolate a point of the focal plane (here the point of coordinates 20,20) and we trace this temporal FFT in the following way:

```
plt.plot(np.abs(Mf[:, 20, 20]))
plt.show()
```



We zoom on the first peak:





We do have a semblance of harmonic centered on the fundamental frequency :

f0≈1000Hz

f1≈1001Hz

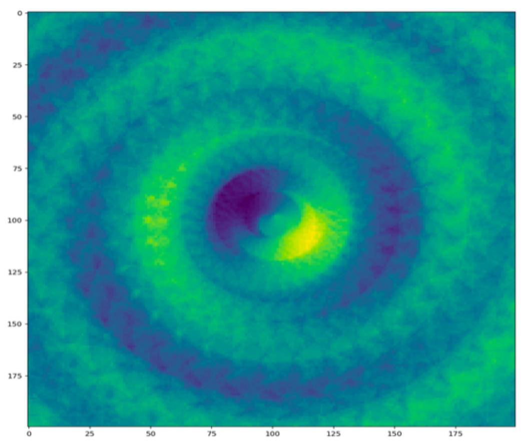
f2≈1004Hz

f3≈1006Hz

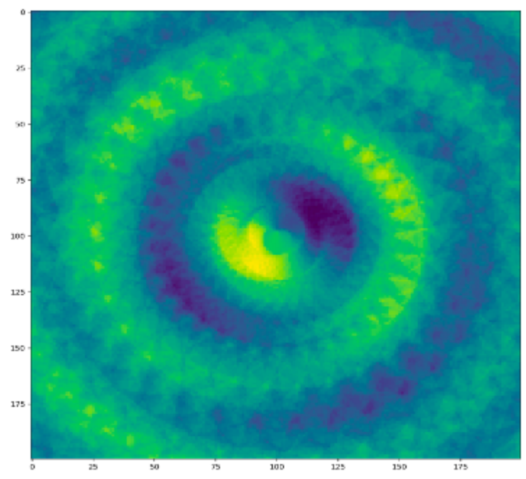
f4≈1010Hz

We are interested in the signal at these frequencies in the following way (projection onto frequency space to harvest the spectral information of interest):

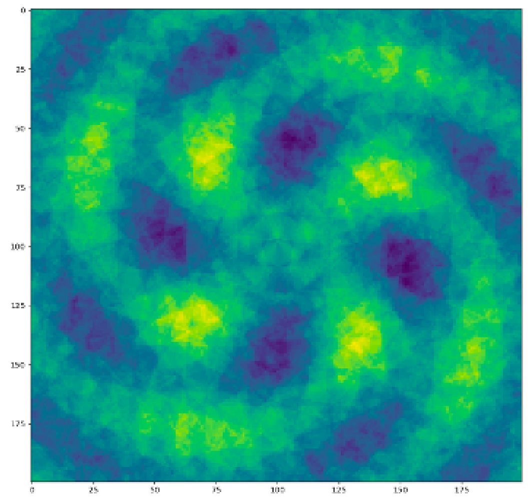
```
plt.imshow(np.real(Mf[6546, :, :])) #Projection of f0 on the fourier space
```



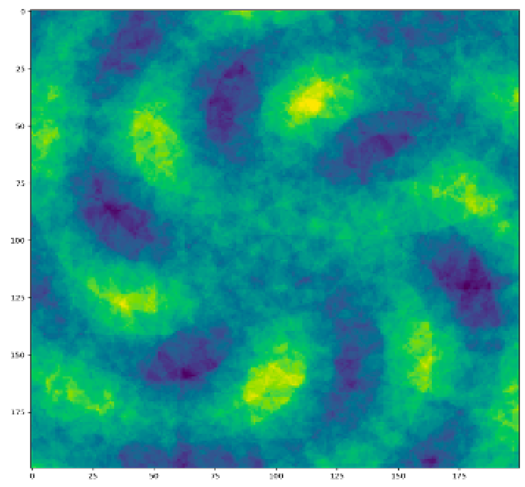
```
plt.imshow(np.real(Mf[6558, :, :])) #Projection of f1 on the fourier space
```



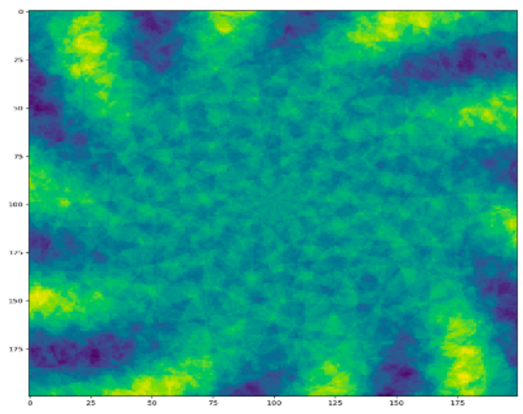
```
plt.imshow(np.real(Mf[6578, :, :])) #Projection of f2 on the fourier space
```



```
plt.imshow(np.real(Mf[6590, :, :])) #Projection of f3 on the fourier space
```

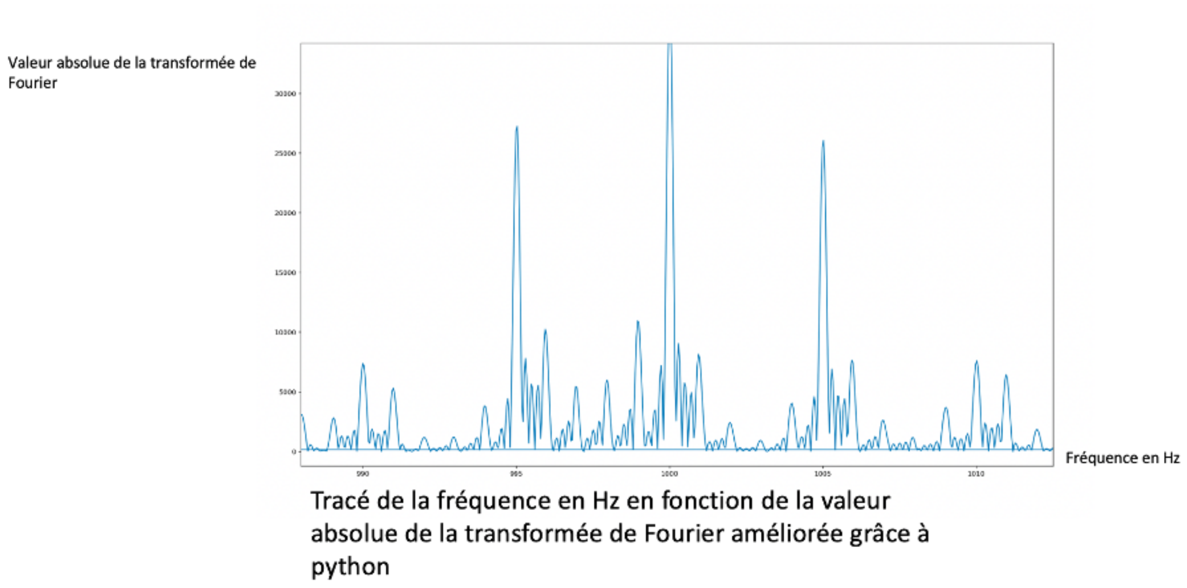


```
plt.imshow(np.real(Mf[6618, :, :])) #Projection of f4 on the fourier space
```



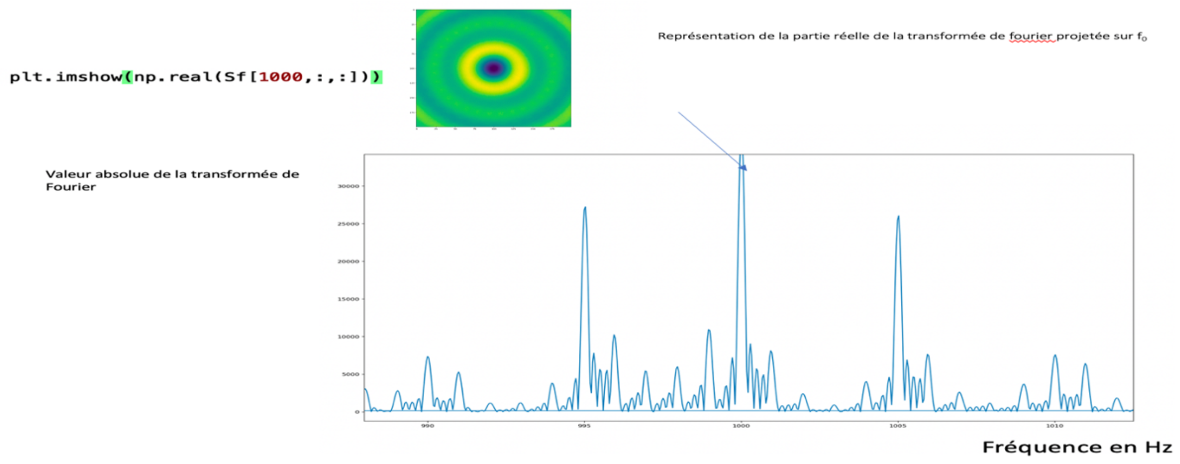
We have a clear undersampling: two solutions are possible, spinning faster to better separate the peaks and interpolating the received signal to get rid of the integer part.

We start by rotating at 5 revolutions per second while keeping the integer part:



Plot of the frequency in Hertz as a function of the absolute value of the Fourier transform improved with python.

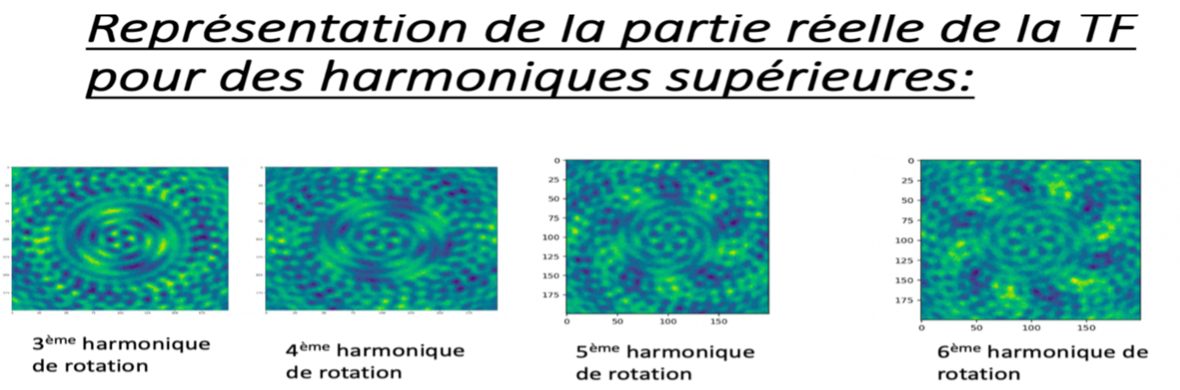
The new emission matrix is created and the previous procedure is repeated, the results obtained are as follows:



```
plt.imshow(np.real(Sf[1000,:,:]))
```

Plot of the real part of the Fourier transformate, projected on f0.

We focus on the harmonics:



Représentation de la partie réelle de la TF pour des harmoniques supérieures:

Plot of the real part of the Fourier transformate, projected on f3, f4, f5 and f6.

Conclusion

We have succeeded in doing two things: separating the peaks and observing a more coherent spectral projection, especially at the fundamental where we have no sign change at the center but a perfect neutrality, proof that the center of the anechoic chamber is not affected by the rotation. We therefore have additional spectral information that we did not have without the rotation that introduces the Doppler effect, information that we can use later in the context of the answer to our initial problem.

Thus the additional spectral information collected via the projection on the Fourier space in the case of the introduction of the Doppler effect was the proof that the idea of Samuel Métais to introduce this non-linearity was the right one, at least from a computer simulation point of view, it must now be confirmed in the anechoic chamber with rotating microphones for example.

- ***Bibliography:***

Effet Doppler Futura-sciences: https://www.futura-sciences.com/sciences/definitions/physique-effet-doppler-2470/

Transformée de Fourier Wikipédia: https://fr.wikipedia.org/wiki/Transformation_de_Fourier

Samuel Métais Thesis: http://theses.md.univ-paris-diderot.fr/METAIS_Samuel_2_complete_20190307.pdf

Imagerie Acoustique Data ifremer: https://data.ifremer.fr/Tout-savoir-sur-les-donnees/Thematiques/Geophysique/Imagerie-acoustique

Limite de la diffraction Wikipédia: https://fr.wikipedia.org/wiki/Pouvoir_de_r%C3%A9solution

Interface Antelope Orion 32+ site web:

https://www.bax-shop.fr/convertisseur-audio/antelope-audio-orion-32-gen-3-convertisseur-an-na-usb-thunderbolt?
gclid=CjwKCAjwyvaJBhBpEiwA8d38vAhhTYMQD67FMZWFA60kO1tQ2ss3PmQIdqMn87wvx8Bx0ztUvgVYRhoC8lMQAvD_Bw