MOUSTAMSIK Rémi CHOUPAS Jérémy

Récapitulatif projet de fin d'Enseignement d'Intégration

(Dans cette partie, les figures représentent toutes la valeur de l'estimateur pour un grille 2D de sources.)

Notions intéressantes :

- Meshgrid (passage à la 2D) : Afin de balayer l'intégralité du plan (profondeur/largeur), nous avions besoin d'un maillage -> utilisation de np.meshgrid(x,y) afin d'explorer toutes les coordonnées sans la notion de hauteur.

- 2D verticale : cela ne fait aucune différence.

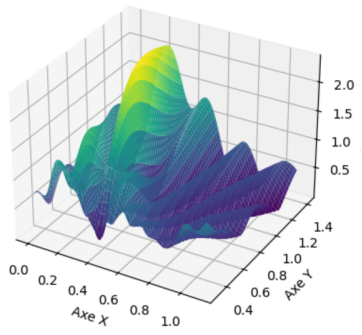
```python
def G2(x, f, c):
    return(np.exp(-1j*(2*pi*f/c)*x)/(4*pi*x))
```

✓ 0.0s

```python
xs = np.arange(nb_points) * deltaXs
ys = np.arange(nb_points) * deltaXs + 0.3
xs, ys = np.meshgrid(xs,ys)
M = fft[indice_m]
deltaX = 0.16
y1 = np.arange(8)*deltaX
y2 = np.zeros([8,1])
y = np.column_stack((y1, y2))
xm = y[:,0]
ym = y[:,1]
deltax = xs.flatten() - xm[:,np.newaxis]
deltay = ys.flatten() - ym[:,np.newaxis]
```

```python
N = np.sqrt(deltax**2 + deltay**2)
g = G2(N, 2000, c)
print(g.shape)
```

```python
ng = np.sqrt(np.sum(np.abs(g)**2,axis = 0))
gn = g/ng

Z = (np.conj(gn).T) @ M
print(Z)
Z = np.reshape(Z, xs.shape)
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(xs, ys, np.abs(Z), cmap='viridis')
# Ajout des titres aux axes
ax.set_xlabel('Axe X')
ax.set_ylabel('Axe Y')
ax.set_zlabel('vraissemblance')
plt.show()
```



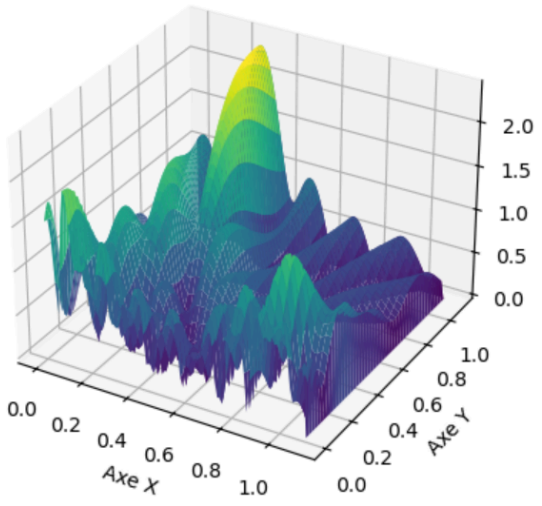
- Passage à la 3D : Superposition des solutions 2D verticales et horizontales. Il est important de bien placer les micros par rapport à l'origine dans le code (cf schéma dans le protocole expérimental). L'ajout de la localisation 2D vertical permettra

d’accéder à z la hauteur, et d’avoir deux valeurs de y la profondeur calculées à partir de données expérimentales (plus robuste !).

- Approche naïve (boucles) :

```
X = np.load(r"C:\Users\Rémi\Desktop\CentraleSupelec\Cours\traitement du signal audio\jour1\source\source2000.npy")
fft = np.fft.fft(X, axis=0)
a = 1
nb_points = 100
deltaXs = 1.12/nb_points
deltaX = 0.16
y1 = np.arange(8)*deltaX
y2 = np.zeros([8,1])
y = np.column_stack((y1, y2))
fe = 20000
c = 340
frequencies = np.fft.fftfreq(len(X[:,0]), 1/fe)
indice_m = np.argmax(fft[:50000,0])
f = frequencies[indice_m]
M = fft[indice_m]
f = 2000
#g = G(y-[1,3], f, c)
Xsmv = np.zeros([nb_points,nb_points])
Y = [i*deltaXs for i in range(nb_points)]
x = [i*deltaXs for i in range(nb_points)]
x, Y = np.meshgrid(x, Y)
for i in range(nb_points-1):
    for j in range(nb_points-1):
        g = G(y-[x[i,j], Y[i,j]], f, c)
        norme_g = np.linalg.norm(M@np.conj(g.T)/np.linalg.norm(g))
        Xsmv[i,j] = norme_g
```

```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x, Y, Xsmv, cmap='viridis')
# Ajout des titres aux axes
ax.set_xlabel('Axe X')
ax.set_ylabel('Axe Y')
ax.set_zlabel('vraissemblance')
plt.show()
```



Cette approche prend 0,9 secondes pour calculer ceci sur un plan

- Traitement matriciel : Calcul des différents éléments vus en cours via des array numpy

```
xcadrillage = np.arange(nb_points) * deltaXs - 0.6
ycadrillage = np.arange(nb_points) * deltaXs + 0.3
zcadrillage = np.arange(nb_points) * deltaXs + 1
xcadrillage, ycadrillage = np.meshgrid(xcadrillage,ycadrillage)
ycadrillage = np.arange(nb_points) * deltaXs + 0.3
zcadrillage, ycadrillage = np.meshgrid(zcadrillage,ycadrillage)
M = fft[indice_m]

deltax_croix_h = xcadrillage.flatten() - Coordonné_micro_h[:,np.newaxis]
deltay_croix_h = ycadrillage.flatten() - y2[:,np.newaxis]

deltaz_croix_v = zcadrillage.flatten() - Coordonné_micro_v[:,np.newaxis]
deltay_croix_v = ycadrillage.flatten() - y2[:,np.newaxis]
```

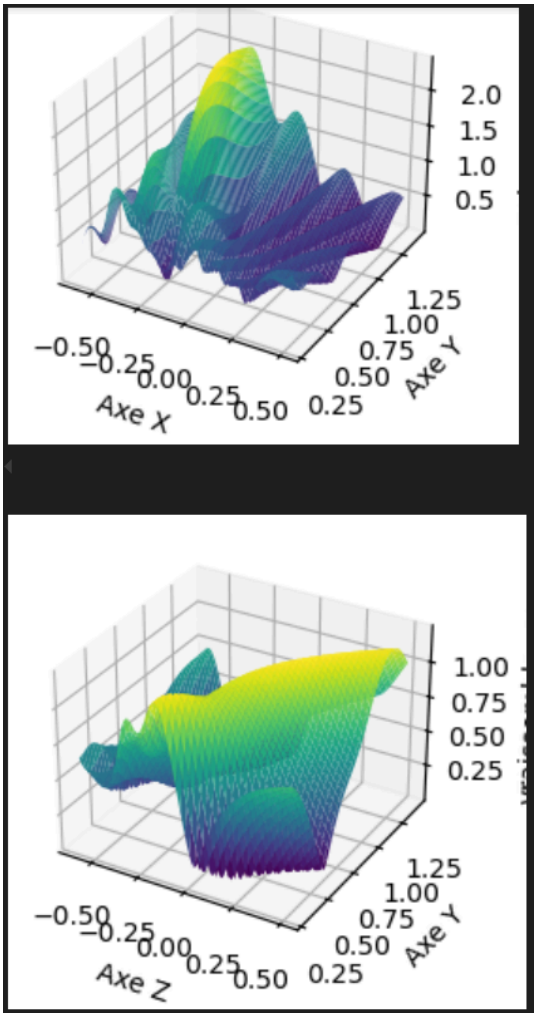
```
N_croix_h = np.sqrt(deltax_croix_h**2 + deltay_croix_h**2)
g_croix_h = G2(N_croix_h, 2000, c)
N_croix_v = np.sqrt(deltaz_croix_v**2 + deltay_croix_v**2)
g_croix_v = G2(N_croix_v, 2000, c)
```

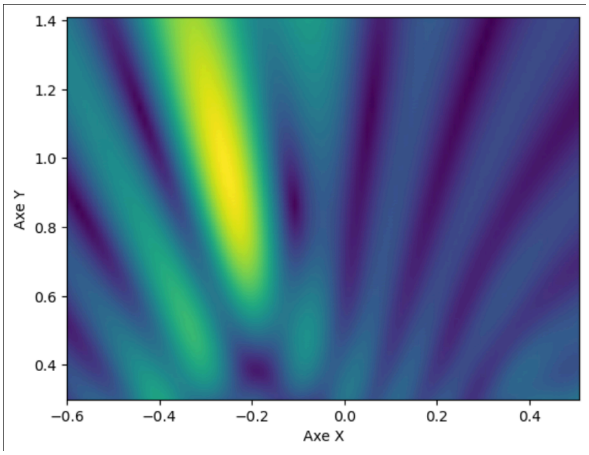
```
ng_croix_h = np.sqrt(np.sum(np.abs(g_croix_h)**2,axis = 0))
gn_croix_h = g_croix_h/ng_croix_h

ng_croix_v = np.sqrt(np.sum(np.abs(g_croix_v)**2,axis = 0))
gn_croix_v = g_croix_v/ng_croix_v

Z_croix_h = (np.conj(gn_croix_h).T) @ M
Z_croix_h = np.reshape(Z_croix_h, xs.shape)
coord_h = np.argmax(Z_croix_h)

Z_croix_v = (np.conj(gn_croix_v).T) @ M
Z_croix_v = np.reshape(Z_croix_v, xs.shape)

coord_v = np.argmax(Z_croix_v)
coord_h = np.argmax(Z_croix_h)
coord_v = [coord_v//nb_points,coord_v%nb_points]
coord_h = [coord_h//nb_points,coord_h%nb_points]
```

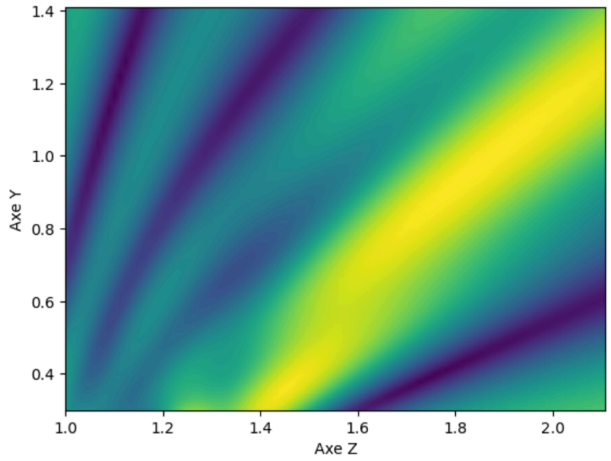
```
print("les coordonnées de la source sont:", xcadrillage[coord_h[0],0], ycadrillage[coord_v[1],0], zcadrillage[coord_v[0],0])

fig = plt.figure()
ax = fig.add_subplot(122, projection='3d')
ax.plot_surface(xcadrillage, ycadrillage, np.abs(Z_croix_h), cmap='viridis')
ax.set_xlabel('Axe X')
ax.set_ylabel('Axe Y')
ax.set_zlabel('vraissemblance')


fig = plt.figure()
ax = fig.add_subplot(121, projection='3d')
ax.plot_surface(xcadrillage, ycadrillage, np.abs(Z_croix_v), cmap='viridis')
ax.set_xlabel('Axe Z')
ax.set_ylabel('Axe Y')
ax.set_zlabel('vraissemblance')

# Affichage de la figure
plt.show()
```


```
print("les coordonnées de la source sont:", xcadrillage[coord_h[0],0], ycadrillage[coord_v[1],0], zcadrillage[coord_v[0],0])
```

```
les coordonnées de la source sont: -0.6 1.1624 1.0
```





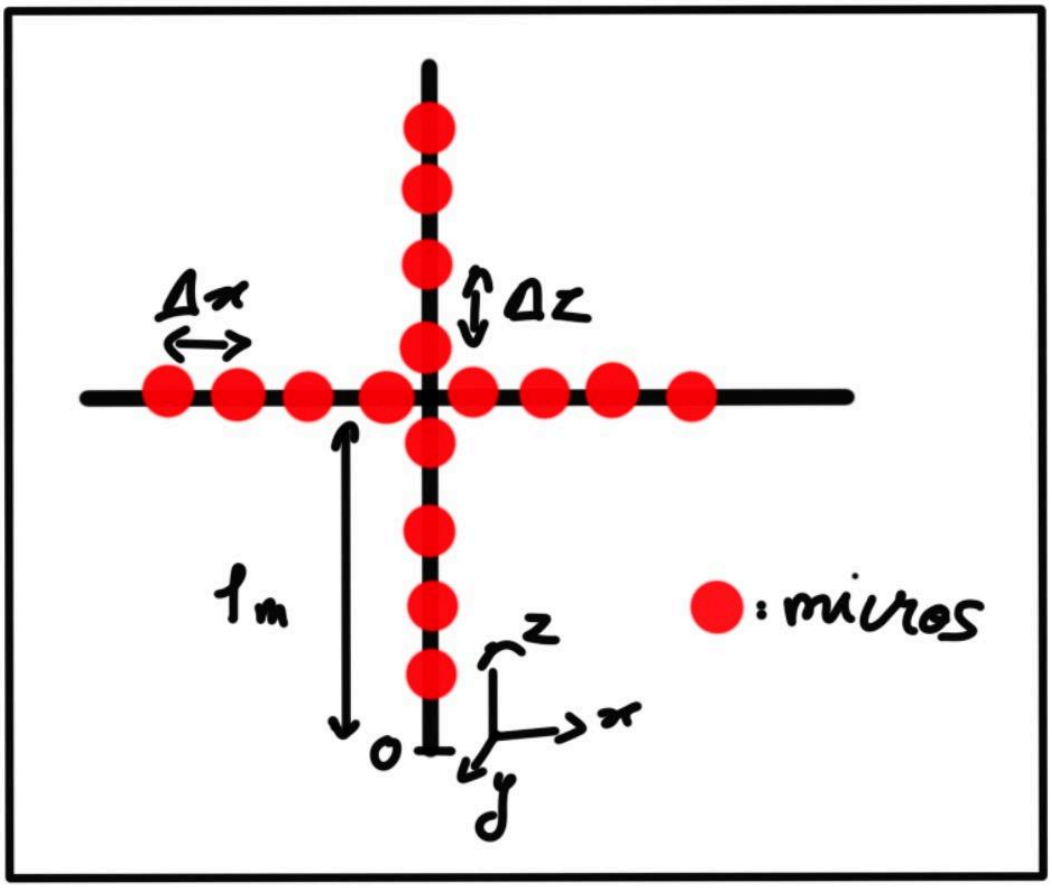


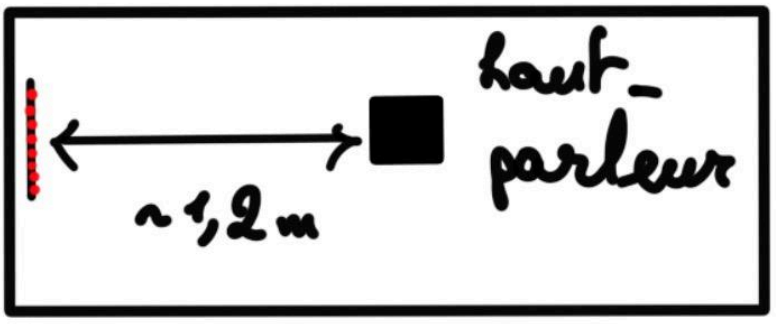
Cette approche nécessite 0.5 secondes pour calculer ceci sur un plan. Elle est donc plus efficace que la précédente.

Protocole expérimental :

- **Source du signal :** Enceinte placée en [-0.6,1.2,0.9](± 5 cm).
- **Réception du signal:** Micros en croix, l'origine est placée au pied de l'intersection des deux barres, ce qui donne les coordonnées des micros suivantes (mesurées sur le système) : micros horizontaux [0.55 -0.16*i, i de 0 à 7] sur x (coordonnées latérales) et micros horizontaux [0.27 + 0.16*i, i de 0 à 7] (coordonnées verticales).

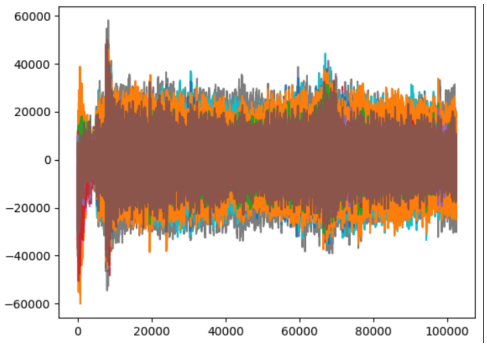
  Les micros sont tous sur y=0 et (seulement) les verticaux sur x=0 comme dans le schéma suivant :





- **Signal émis :** sinusoïdal, de fréquence freq = 2 kHz, échantillonné à un taux de 10 kHz.
- **Signal reçu :** échantillonné à un taux de 20 kHz (et exporté en .wav)

A la réception : le rendu des 16 micros (matrice de 102400 * 16):



Résultats et interprétation :

Les résultats coïncident avec la réalité et les graphes 2D montrent des signes distinctifs intéressants (comme la formation (sur l'estimateur de position) caractéristique de maximums locaux périodiques).
L'écart en z de 10 cm pourrait se justifier par l'inclinaison des enceintes vers le haut (1.20 * sin(pi/6) = 11 cm), mais ce n'est pas le cas (un autre essai avec les données ci-dessus provenant d'un enregistrement avec une enceinte non inclinée donnant le même résultat).

Manquement :

Nous aurions pu tester l'algorithme sur un signal de parole (il aurait été beaucoup plus compliqué de le localiser notamment à cause de l'acquisition du signal émis, qui n'est plus "parfaite" comme pour un signal simulé, ainsi il aurait notamment fallu agir sur la durée de l'enregistrement dans le protocole).