TP0 : Solution analytique de l'équation de Helmholtz

Clément Baillet

September 21, 2025

Le but de ce TP était de commencer à apprivoiser la mise en oeuvre d'un solveur BEM dans le cadre de l'équation d'Helmholtz en 2D. Nous sommes dans le cas de la diffraction d'une onde incidente plane, i.e. $u^{\text{inc}} = \mathrm{e}^{-i\mathbf{k}\cdot\mathbf{x}}$, par un disque centré en 0 et de rayon a. Pour ce faire, j'ai commencer par coder un Python un générateur de maillage pour la frontière de ce disque, que nous noterons Γ et qui correspond au cercle de centre 0 et de rayon a. Comme je ne savais pas par où commencer, je me suis servi de ChatGPT pour me générer le maillage, en lui fournissant le prompt suivant :

Prompt

Comment répondre à la question : "générez un maillage du bord du disque du centre 0 et de rayon a : d'abord les noeuds du maillage puis les extrémités de chaque segment. Quel est le nombre de segments (en fonction du nombre de noeuds du maillage) ?", en sachant que j'aimerais le coder en Python.

Voici le code obtenu en retour :

```python
def generate_nodes(N, a):
    angles = np.linspace(0, 2*np.pi, N, endpoint=False)
# angles uniformes
    x = a * np.cos(angles)
    y = a * np.sin(angles)
    return np.column_stack((x, y))

def generate_segments_indices(N):
    return [(k, (k+1) mod N) for k in range(N)]
```

Afin de vérifier le code, je l'ai simplement relu ligne par ligne, afin de voir s'il faisait sens, puis je l'ai exécuté (il y avait également une partie de code généré permettant d'afficher le maillage, mais je ne l'ai pas inclus ici). Le code ayant été relu et le résultat final étant probant, je suis passé à la suite.

L'étape suivante a été de calculer la dérivée normale de la fonction $-u^+ - u^{\text{inc}}$, que l'on notera p. Pour ce faire, il faut rappeler la formule du gradient en coordonnées polaires :

$$\nabla = \frac{\partial}{\partial r}\vec{u_r} + \frac{1}{r}\frac{\partial}{\partial \theta}\vec{u_\theta}$$

où $\vec{u_r}$ et $\vec{u_\theta}$ sont les vecteurs de la base mobile. Une fois ceci en tête, le calcul de la dérivée normale de p sur Γ devient direct, puisque $\nabla_{\vec{n}} = \nabla \cdot \vec{u_r} = \frac{\partial}{\partial r}$. On trouve donc, en dérivant sous le signe somme (possible en se plaçant sur un anneau compact autour de Γ) :

$$\nabla_{\vec{n}} p(a,\theta) = \sum_{n\in\mathbb{Z}} \left((-i)^n \frac{J_n(ka)}{H_n^{(1)}(ka)} \underbrace{\frac{\mathrm{d}}{\mathrm{d}r}(H_n^{(1)})(ka)}_{=\frac{k}{2}(H_{n-1}^{(1)}(ka) - H_{n+1}^{(1)}(ka))} \mathrm{e}^{in\theta} \right) + ik\cos\theta \mathrm{e}^{-ika\cos\theta}.$$

On la code en Python en se servant du module scipy et en important les fonctions de Bessel et de Hankel du premier ordre, puis en effectuant une boucle for pour calculer les termes successifs de la somme (j'ai démarré la somme à $n = -99$ et l'ai arrêté à $n = 99$). Voici le graphe du module de p sur Γ en fonction de θ :

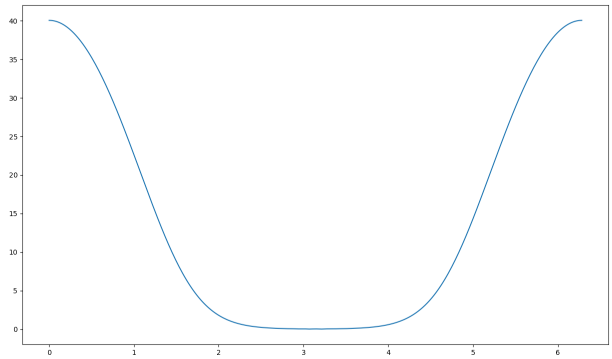

Figure 1: Module of p