INP Toulouse - ENSEEIHT

Bureau d'étude

UE Méthodes Numériques
Programmation pour le calcul scientifique

PROJET FORTRAN

Élèves :
Mohemed Reda Yacoubi

Année 2021-2022

Table des matières

1 Introduction

L'objectif de ce BE est d'implémenter la méthode des éléments finie pour la simulation numérique d'un guide d'onde rectangulaire métallique creux en mode TE et TM et de retrouver les résultats du simulateur matlab de la pdetool pour enfin faire un interfaçage python et comparer les résultats.

On s'intéresse donc au problème de Dirichlet suivant :

$$\begin{cases} \Delta u(x,y) + k_c^2 u(x,y) = 0 & dans \quad \Omega \\ u(x,y) = 0 & sur \quad \partial\Omega \end{cases}$$

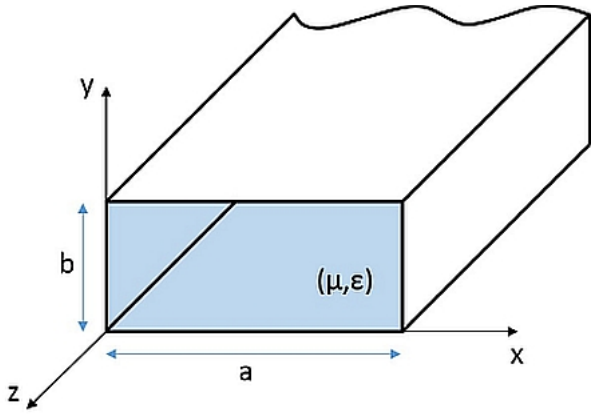

Figure 1 – Guide d'onde rectangulaire avec a=b=1m

On rappelle que la constante de propagation pour un mode TEM est donnée par l'expression suivante : $k_c^2 = (\frac{n\pi}{a})^2 + (\frac{m\pi}{b})^2$.
Le premier mode TM qui se propage est le mode TM_{11} correspond à une constante de propagation $k_c^2 = 2\pi^2$. Alors que le premier mode de propagation pour les modes TE est TE_{10} de constante de propagation $k_c^2 = \pi^2$.

2 Partie 1

Dans la première partie du projet, on est censé à coder et modifier sur Fortran le programme principal ainsi que d'autres différents sous programmes nécessaires pour la lecture du maillage exporté depuis Matlab et la réalisation de l'assemblage des matrices de rigidité et de masse : **GetSize, ReadData, Surface, Jacobian, Diag, Assemblage.**
Une fois le programme test complété permettant de valider tout les sous-programmes, on modifie notre propre programme principal pour prendre en compte n'importe quel maillage. On validera alors ce programme en utilisant un nouveau maillage à l'aide de la procédure décrite dans le document : **BE partie contextuelle**.

3 Partie 2 : interfacage python

Une fois les 2 matrices assemblées, la solution s'obtient en résolvant un problème aux valeurs propres généralisé .Implémenter cette résolution étant trop lourd elle sera réalisée en interfaçant le code fortran avec un code python. On résout donc notre problème aux valeurs propres, pour le mode TE_{10}, comme on a indiqué auparavant la constante de propagation (valeur propre) est π^2. On voit bien que ce résultat est similaire à celui trouvé avec la simulation **Matlab** en utilisant l'outil pdetool, ainsi qu'avec la simulation sur **Fortran** et **Python**.

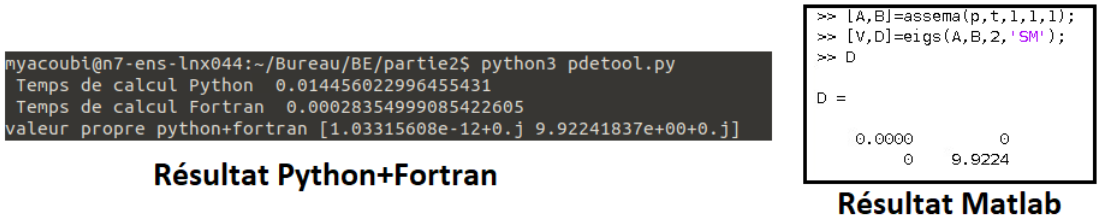

FIGURE 2 – Résultats avec maillage non raffiné

On trouve donc lun résultat proche de la théorie soit pour la simulation Matlab soit pour la simulation Fortran et Python et pour plus de précision, on raffine notre maillage et on fait la simulation une autre fois, on trouve des résultats plus proches que la théorie, presque la même valeur :

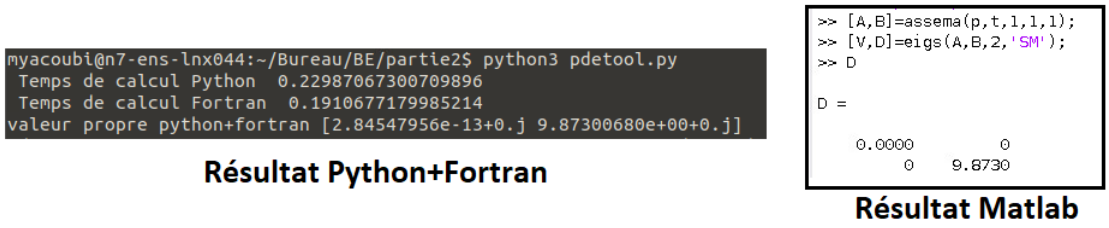

FIGURE 3 – Résultats avec maillage raffiné

On voit bien que le résultat après raffinement du maillage devient plus proche de la valeur théorique $\pi^2 = 9.8696$.
On peut aussi visualiser les tracés des solutions obtenues avec Python/Fortran et Matlab (pdetool) avec un maillage non raffiné ainsi qu'avec un maillage raffiné.

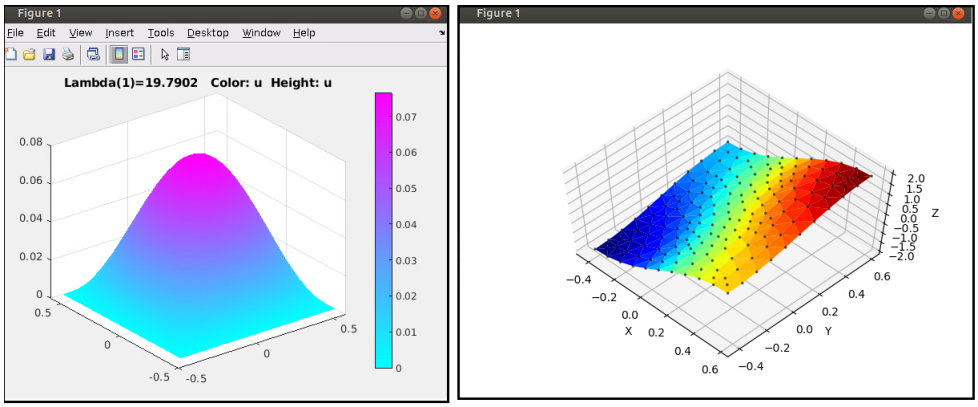

FIGURE 4 – Résultats avec maillage non raffiné

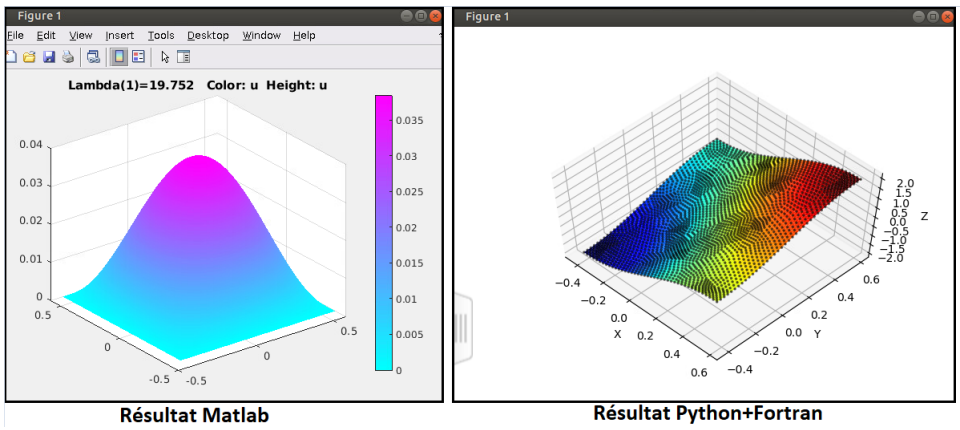

FIGURE 5 – Résultats avec maillage raffiné

On trouve toujours le même résultat avec plus de précision lorsque le maillage est raffiné. En fait Matlab et Python traite les données avec le même principe mathématique ; principe de la résolution des problèmes aux valeurs propres.

3.1 Comparaison Python/Fortran

Python et Fortran sont tous deux des langages relativement faciles à apprendre. Il est probablement plus facile de trouver de bons supports d'apprentissage Python que de bons supports d'apprentissage Fortran, car Python est utilisé plus largement et Fortran est actuellement considéré comme un langage "spécialisé" pour le calcul numérique.
En utilisant Python (ou MATLAB ou n'importe quel langage interprété), vous renoncez aux performances au profit de la productivité. Par rapport à Fortran (ou tout autre langage compilé), vous écrirez moins de lignes de code pour accomplir la même tâche, ce qui signifie généralement qu'il vous faudra moins de temps pour obtenir une solution fonctionnelle. Et on voit bien ça dans les résultats de simulation :

FIGURE 6 – Le temps de calcul Python vs Fortran

4 Conclusion

Grâce à ce projet, j'ai pu découvrir un nouveau langage de programmation Fortran. En fait, Python est utilisé plus largement dans l'ensemble en tant que langage à usage général, alors que Fortran est largement limité au calcul numérique et scientifique.
Finalement on peut dire que la résolution des problèmes basés sur le calcul numérique sur fortran est bien meilleure que sur un autre langage de programmation interprété, même si le codage sur Fortran est un peu plus difficile.
Une stratégie courante consiste à mélanger Python et Fortran (interfaçage Python), et n'utilisez le langage compilé(Fortran) que pour les parties du code les plus sensibles aux performances ; le coût de développement est, bien sûr, qu'il est plus difficile d'écrire et de déboguer un programme en deux langues qu'un programme en une seule langue.