PROJET EDP & approximation numérique

2022

Le code à été fait sur le l'interface Spyder , on utilise seulement les bibliothèques Numpy , Scipy et plt.plot comme demandé dans l'énoncé , aussi des fonctions faites dans les TP précédent ont été recopié mais pas de manière systématique, la raison principale étant qu'il était nécessaire de les modifier pour les faire fonctionner dans le contexte du projet.En effet, LoadVTX et LoadELT ne sont pas explicitement demandés mais permettent de facilement lire les fichier config1 et config2 données.

Ce document est un résumé pour chaque question qui explique ce que fait le code associé à chaque question et indique le type de retour ,Les réponses des TP seront aussi indiqués sur ce document accompagnés d'illustration.

Question 1 : LOADELT

Ici j'ai repris la fonction qu'on avait codé en TP **LoadELT** , simplement j'ai rajouté qui permet de récupérer le Label et sépare correctement le tableau des nœuds (qui prend en maintenant compte 4 arguments)

J'ai aussi rajouté pour simplifier les test en aval la Fonction L**oadVTX** , pas de modification sur celle-ci

Les arguments des deux fonctions sont : un fichier type config.1.msh ou config2.msg

Les arguments de retour sont :

Pour **LoadVtx :** Une liste ordonnée de tuples (dans l'ordre du fichier) qui contiennent les coordonnées x et y de chaque sommet

Pour **LoadELT** : Un tuple, **le premier élémen**t du tuple est une liste ordonnée (dans l'ordre du fichier) de chaque triangles (représenté par un triplet de sommet), **le deuxième élément** est une liste (dans l'ordre du fichier encore) qui représente le label correspondant du triangle, c'est-à-dire s'il se trouve dans Ω1ou Ω2

Question 2&3 : Boundary + Plotsubdomain

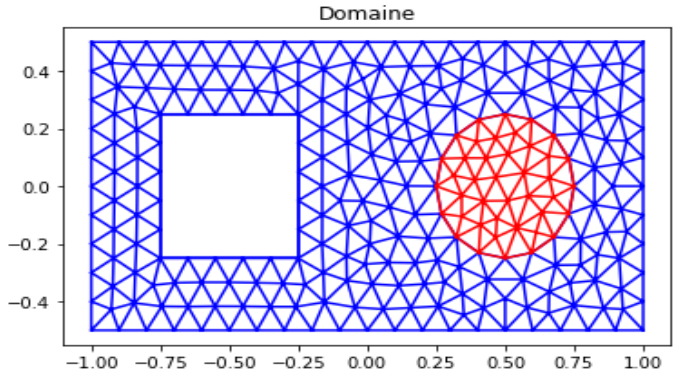
Pour cette question j'ai importé une question faite dans les précédent TP, il s'agit de la fonction **Boundary :** elle a deux arguments : elle prend un type de sortie de type **LoadELT,** et un type de sortie de type **LoadVTX ,** Contrairement à celle fait en TP elle ne renvoie pas les bord , mais seulement le bord intérieur de Ω0 , il s'agit donc de la même fonction mais un peu modifié

Elle récupère dans un premier temp toute les arrêtes de bord, puis ensuite trie seulement celles qui les intéresses avec une méthode bien particulière qui sera décrite à l'oral, son code en revanche n'est probablement en revanche pas le plus optimisé.

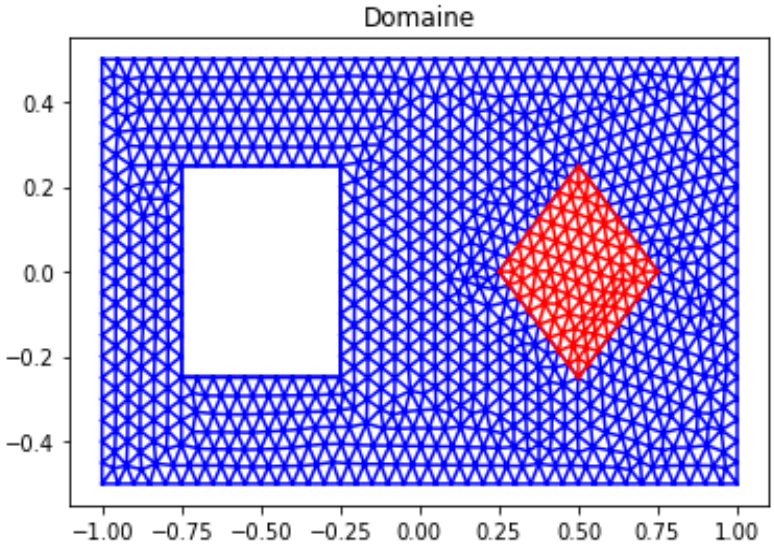
PlotSubDomain(vtx,elt,label): les deux premiers argument de cette fonction sont les même que pour **boundary** , en revanche elle prend en dernier argument la liste des Labels (qui est fourni avec LoadElT[1])

Voici son résultat :

Sur config 1



Sur config 2



<u>Question 4 ,5,6</u>

La méthode variationelle faite a la main prouve l'existence et l'unicité du problème au limite , on trouve 1 comme solution évidente du problème

Implementer ce résultat on code plusieurs fonctions :

<u>Mloc(vtx, e):</u> Prend en argument un tableau de type sortie de **LoadVTX** , un triplet ou tuple quelquonque qui represente les **coordonées d'un triangle quelquonque** , son type de sortie est une matrice a coefficiement reels de taille 3*3 si e est un triangle ou une matrice de taille 2*2 si e est une arête , Ici on effectue le calcul d'air par la formule de héron

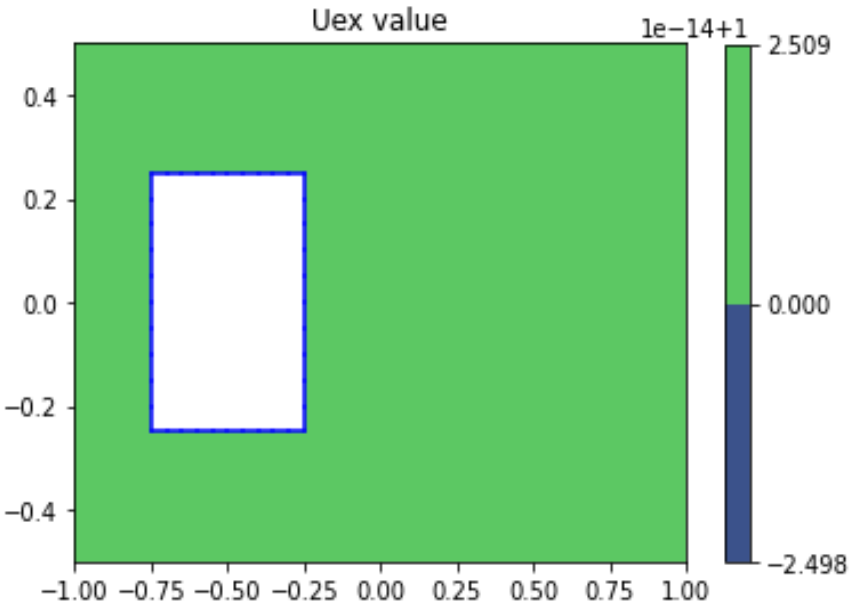
<u>rigiditeloc(vtx,e):</u> Même arguments que Mloc , renvoit une matrice de taille 3*3 si e est de taille 3 (donc un triangle) qui utilise encore la formule de héron pour l'air , sinon si e est de taille 2 (donc une arrête) il renvoit une matrice de taille 2*2

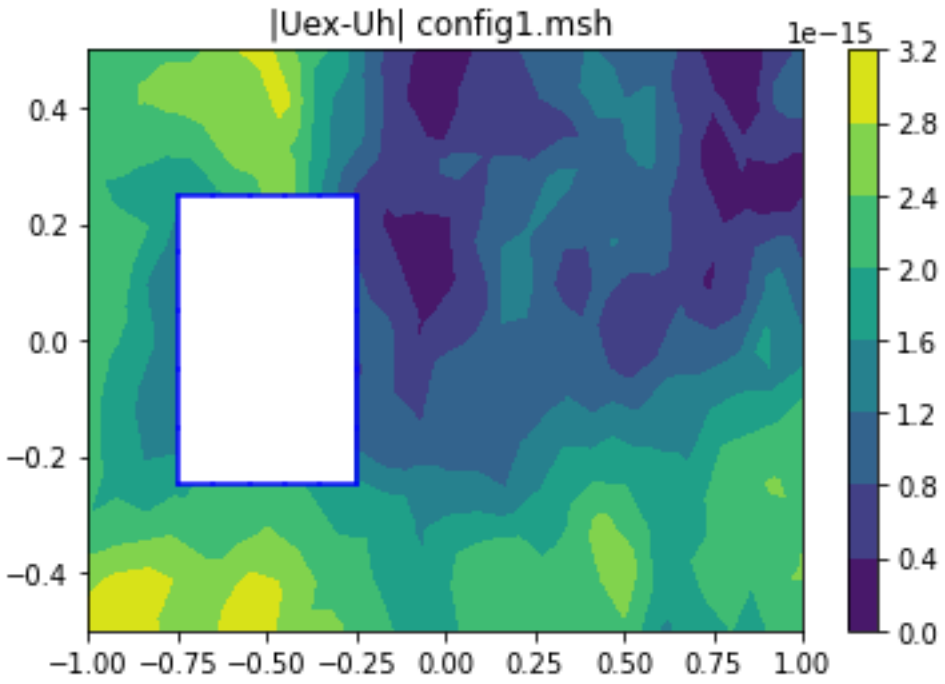
Mass(vtx,elt) : Prend en argument le type de sortie de **LoadVtx**,**Loadelt** , renvoit la Matrice **Masse + la matrice de ridigité** globale en utilisant **Mloc** et **Rigiditeloc** (taille dépendant du fichier considéré)

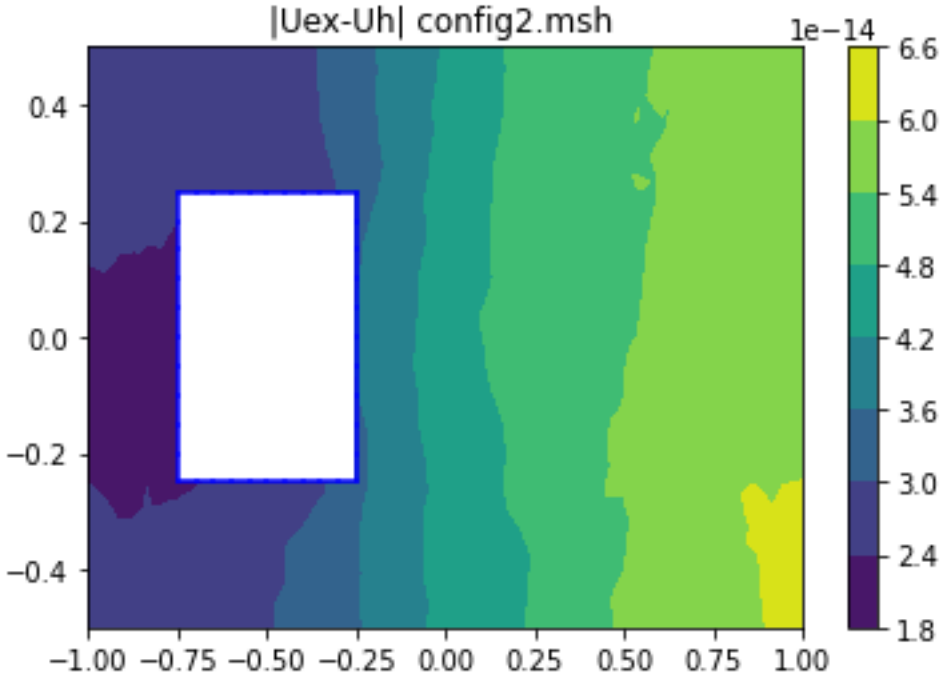
F(vtx,elt) : Prend en argument le type de sortie de **LoadVtx**,**Loadelt ,** Renvoit le Second membre de la formulation variationnel

On aura pris soin dans le calcul de **Mass(vtx,elt) ,** D'effectuer les test du cours pour la conformité de la matrice de masse.

On represente directement la solution sur **Plotsubdomain** : voici le résultat pour chaque fichier







On calcule ensuite la norme deux de l'erreur on obtient une norme L2 de l'ordre l a 2.78e-14 pour config 1 et 1.e-12 pour config 2"

Déjà l'erreur est très faible car on approxime une fonction linéaire par des fonctions linéaires (Méthode P1) ;

Etonnament l'erreur est plus grande en utilisant config 2 ce qui est contre intuitif car a priori un maillage plus detaillé devrait rendre l'erreur sur la solution sur faible , On peut expliquer ça par l'approximation en elle-même , on approxime une fonction linéaire par des fonctions linéaires elles-mêmes (On utilise la méthode P1) , a priori ajouter des mailles a un fichier ne devrait pas améliorer

la consistance de la méthode , un maillage très simple devrait donner une solution quasi exacte , par contre augmenter la taille des matrice comme on le fait avec Config 2 , augmente les erreurs , commise erreurs mais aussi le conditionnement de la matrice de masse M car on fait des approximations avec des racines carrés sur des flottants ce qui pourrait expliquer pourquoi le niveau d'erreur augmente même avec un fichier plus précis (même si elle reste très faible !)

Question 7

On recode les fonctions Mass2 et F2 , qui sont des copies quasi exactes des fonctions précédente , juste en ajoutant ou supprimant des conditions nécéssaires qui ont disparu dans la méthode variationelle , ici aussi l'existence et l'unicité de la solution peut se verifier facilement à la main ; on trouve :