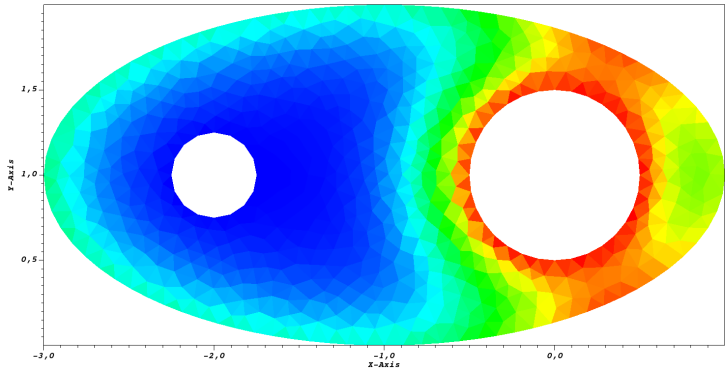
Matmeca 2ème année - TP de MNPI1
année 2022-2023

TP 2 : Approximation volumes finis de l'équation de la chaleur en dimension 2 sur maillage non structuré



1 Le modèle

Le but de ce TP est de programmer un schéma pour résoudre le problème suivant :

$$\partial_t T = \nabla \cdot (D \nabla T), \qquad t > 0, \quad X \in \Omega \in \mathbb{R}^2$$
$$CI : T(0, X) = T_{init}(X)$$
$$CL : T(t, X) = T_b(t, X) \text{ ou } -D(X)\nabla T(t, X) \cdot n(X) = \phi_b(t, X) \qquad X \in \partial\Omega,$$

où l'inconnue est $T(t, X)$. Le coefficient de diffusion D est une fonction positive de X. Des conditions aux limites différentes peuvent être appliquées sur différentes parties du bord de Ω.

Nous étudierons plusieurs problèmes qui diffèrent par leur condition initiale et leurs conditions aux limites.

Pour la suite, prenez le temps de réfléchir en organisant votre programme sur le papier : même si l'essentiel est donné dans ce document, ne foncez pas tête baissée dans l'écriture de votre programme. En particulier, il faut soigneusement traduire le schéma numérique en utilisant les structures de données proposées.

2 Le schéma

On considère le schéma explicite sur maillage non structuré vu en cours, dans lequel on rajoute le traitement du terme source.

Le maillage. Le maillage constitué de triangles, est généré avec le logiciel `gmsh`, et sauvegardé sous format medit (avec l'extension .mesh), comme vu en TP de C++.

Toutes les subroutines et fonctions pour lire le maillage et en tirer les informations nécessaires au schéma volumes finis sont stockées dans le module `mod_maillage`, à télécharger sur `http://www.math.u-bordeaux.fr/~lmieusse/PAGE_WEB/enseignement.html`.

Dans le programme principal, il suffit d'appeler la subroutine `maillage` pour calculer toutes les informations nécessaires (voir ci-dessous pour les arguments de la subroutine).

Le schéma. À l'initialisation, on pose pour chaque maille Ω_i

$$T_i^0 = T_{init}(X_i),$$

où X_i est le centre de la maille Ω_i.

Ensuite, on pose pour $n = 0$ à n_{max} et $i = 1$ à i_{max},

$$T_i^{n+1} = T_i^n - \frac{\Delta t}{|\Omega_i|} \sum_{e \in \mathcal{E}_i^{int}} |e| F_{i,j}^n - \frac{\Delta t}{|\Omega_i|} \sum_{e \in \mathcal{E}_i^{bord}} |e| F_i^{n,b} + \Delta t S(t_n, X_i),$$

où l'on note $\mathcal{E}_i^{int}$ et $\mathcal{E}_i^{bord}$ l'ensemble des arêtes intérieures et de bord de Ω_i. Les flux aux arêtes intérieures sont définis par

$$F_{i,j}^n = -D(X_{ij}) \frac{T_j^n - T_i^n}{d_{ij}},$$

où j est l'indice de la maille Ω_j qui partage l'arête e avec Ω_i, $d_{ij} = \|X_i X_j\|$ est la distance joignant les centres des cercles circonscrits aux mailles adjacentes Ω_i et Ω_j, et X_{ij} le milieu de l'arête e.

Les flux sur les arêtes de bord sont définis en utilisant les conditions aux limites. Pour une condition de Neumann, on pose

$$F_i^{n,b} = \phi_b(t_n, X_i^b).$$

Pour une condition de Dirichlet, on pose

$$F_i^{n,b} = -D(X_i^b) \frac{T_b(t_n, X_i^b) - T_i^n}{d_i^b},$$

où X_i^b est le milieu de l'arête e, et $d_i^b = \|X_i X_i^b\|$ est la distance joignant le centre du cercle circonscrit à Ω_i au milieu de l'arête de bord.

On rappelle enfin que pour des CL de Neumann, ce schéma est L^∞ stable sous la condition CFL

$$\Delta t \leq \min_i \frac{|\Omega_i|}{\sum_{e \in \mathcal{E}_i^{int}} |e| D(X_{ij})/d_{ij}}.$$

3 Le code

Un certain nombre de modules sont fournis sur

`http://www.math.u-bordeaux.fr/~lmieusse/PAGE_WEB/enseignement.html`

pour la lecture et le traitement du maillage ainsi que pour les sorties fichier au format `vtk` permettant de visualiser les résultats du code.

Il vous reste à écrire le programme principal qui contient toutes les structures de données nécessaires et les instructions correspondant au schéma, et vous pouvez aussi éventuellement créer quelques modules pour définir les fonctions utilisées dans le problème (donnée initiale, coefficient de diffusion, solutions exactes, etc.).

Pour compiler votre code, utilisez un makefile. Durant la phase de développement, utilisez les options de détection des erreurs

```
-g -ffpe-trap=invalid,zero,overflow -fbounds-check -fcheck=all -Wall
```

Quand le code fonctionne correctement, vous pouvez accélérer son exécution en remplaçant toutes les options de compilation par `-O2`. Pour cela, le plus simple est d'utiliser une macro dans votre makefile.

3.1 Structures de données

On suggère fortement d'utiliser les structures de données (tableaux) suivantes :

Solution numérique

- `T(i=1:nb_mailles)` tel que `T(i)` $= T_i^n$
- `Tnp1(i=1:nb_mailles)}` tel que `Tnp1(i)` $= T_i^{n+1}$

Quantités géométriques

- `aire_maille(1:nb_mailles)}` tel que `aire_maille(i)` $= |\Omega_i|$
- `l_arete(1:nb_aretes)` tel que `l_arete(k)` soit la longueur de l'arête n°k
- `d_arete(1:nb_aretes)` tel que `d_arete(k)` soit la longueur d_{ij} si l'arête n°k est celle commune aux mailles Ω_i et Ω_j
- `coord_noeud(1:nb_noeuds)` tel que `coord_noeud(i,1)` et `coord_noeud(i,2)` soient l'abscisse et l'ordonnée du noeud n°i
- `milieu_arete(1:nb_aretes,1:2)` tel que `milieu_arete(k,1)` et `milieur_arete(k,2)` soient l'abscisse et l'ordonnée du milieu de l'arête n°k (c.- à-d. les coordonnées du point X_{ij} si l'arête n°k est celle commune aux mailles Ω_i et Ω_j)

Conditions aux limites

- `cl_arete(1:nb_aretes)` tel que `cl_arete(k)` soit un entier qui indique la condition aux limites à appliquer à l'arête n°k, si celle-ci est une arête de bord. Pour les arêtes intérieures, cette valeur n'est pas utilisée.

Le tableau `cl_arete` est bien plus grand que nécessaire, car il suffirait de le définir pour les arêtes de bord uniquement, mais il permet une implémentation plus simple. Le logiciel de maillage `gmsh` permet d'attribuer un numéro entier à toute les arêtes d'une partie du bord, et ainsi de tenir compte de plusieurs conditions aux limites différentes. C'est à vous de décider quel numéro correspond à telle ou telle condition aux limites, et de maintenir une cohérence entre le numéro donné dans `gmsh` et celui utilisé dans votre programme.

Donnons un exemple : vous voulez appliquer 2 conditions de Dirichlet différentes et une condition de Neumann sur 3 parties du bord de Ω. Dans `gmsh`, vous pouvez attribuer les numéro 10, 11, et 20 à ces trois parties. Dans votre code, sur une arête k telle que `cl_arete(k)` vaut 10, on appliquera la première condition aux limites de Dirichlet. Si elle vaut 11, on appliquera la seconde, et si elle vaut 20, on appliquera la condition aux limites de Neumann.

Informations de connectivité

- `arete_maille(1:nb_mailles,1:3)` tel que `arete_maille(i,1:3)` contienne les numéros des 3 arêtes du bord de la maille Ω_i : ces numéros sont stockés en parcourant le bord dans le sens trigonométrique
- `noeud_maille(1:nb_mailles,1:3)` : comme le tableau précédent, pour les noeuds des mailles
- `maille_arete(1:nb_aretes,1:2)` tel que `maille_arete(k,1)` et `maille_arete(k,2)` soient les deux mailles adjacentes à l'arête n°k.

Ce dernier tableau possède la propriété suivante : si l'arête n°k est sur le bord, il n'y a qu'une maille adjacente, et le code génère `maille_arete(k,2)=0`. Cela donne donc le moyen de repérer si une arête est intérieure ou sur le bord et permet de traiter les conditions aux limites facilement.

3.2 Structure du code

On propose de programmer le schéma avec une structure *par arêtes*, détaillée ci-dessous :

1. lecture des données dans un fichier de données (donnée initiale, conditions aux limites, temps maximum, cfl, nom du fichier de maillage)
2. lecture du maillage
3. allocation et initialisation des tableaux pour T^n et T^{n+1}
4. calcul du pas de temps
5. boucle en temps
   - boucle sur les arêtes
     - calcul du flux sur l'arête (intérieure ou de bord)
     - ajout de la contribution du flux à la température dans les deux ou l'unique maille(s) adjacente(s)
   - mise à jour de T^n
   - sortie fichier

L'étape 1 se fait en appelant la subroutine `maillage` de la façon suivante :

```
call maillage(fichier_maillage, nb_mailles, nb_aretes                    &
     &        , coord_noeud, noeud_maille, aire_maille, l_arete, d_arete &
     &        , milieu_arete, arete_maille, maille_arete, cl_arete)
```

Cette subroutine lite le fichier de maillage, en déduit le nombre de mailles et d'arêtes, puis alloue tous les tableaux donnés en arguments et les remplit correctement. La variable `fichier_maillage` est une chaîne de 30 caractères contenant le nom du fichier de maillage (donné par l'utilisateur dans le fichier de données). Cette subroutine se trouve dans le module `mod_maillage`. Si vous voulez avoir une idée de la façon dont on peut calculer les informations de connectivité, vous pouvez lire cette subroutine et les sous-programmes appelés.

À l'étape 5, à la suite du calcul du flux : attention, le flux a deux contributions *opposées* aux deux mailles adjacentes. Selon la façon dont vous calculez votre flux, il doit apparaître avec un signe $-$ dans une maille, et avec un signe $+$ dans l'autre maille.

En fin d'étape 5, l'appel à la subroutine `sortie` de la façon suivante :

```
call sortie(iter, T, coord_noeud, noeud_maille)
```

Cette subroutine (contenue dans le module `mod_sortie`) écrit dans un fichier au format vtk le maillage et la valeur de T dans chaque maille. La variable `iter` est un compteur entier contenant le numéro de l'itération en temps. Il doit être incrémenté à chaque itération. Ce compteur est utilisé pour créer le nom du fichier de sortie, qui est du type `sortie_iter.vtk`.

Pour éviter de stocker de trop nombreux fichiers, on pourra se contenter de 10 sorties par simulation, par exemple en définissant `nplot = nmax / 10` et en ne faisant appel à la subroutine de sortie fichier que si `n` est divisible par `nplot` (voir le TP n°1). Vous pouvez aussi vous contenter d'une seule sortie finale, en particulier pour les maillages les plus fins, car les fichiers de sorties sont en format ascii, et prennent donc beaucoup de place.

4 La visualisation

Elle sera faite avec le logiciel paraview, en visualisant le maillage et le champ de température sous forme de *contours*, voir le TP1 de C++. Avec paraview, ouvrez la suite de fichiers `T_*.vtk` générée

par le code : à chaque appel par le programme principal, la subroutine `sorties_fichier` génère un fichier ascii au format VTK. Paraview est alors capable d'afficher le champ de température corresponsant à chacun de ces fichiers, et aussi de permettre une visualisation de l'évolution du champ par une animation.

5 Les cas tests

Cas test 1 : solution 1D dans un carré. Téléchargez le fichier de maillage `cas_1d.mesh`. Ce fichier contient le maillage du carré $[0,1] \times [0,1]$, avec le même code de conditions aux limites sur les bords ouest et est (code 10) et le même code de conditions aux limites sur les bords sud et nord (code 20).

Votre code et ce fichier doivent vous permettre de résoudre le problème défini par des conditions de Dirichlet homogènes sur les bords ouest ($T_b = T_G = 100$) et est ($T_b = T_D = 300$), des conditions de Neumann homogènes sur les bords sud et nord, la donnée initiale $T_{init} = T_G$, $t_{max} = 1$, $D = 1$. La cfl sera prise égale à 1.

Vérifiez avec paraview que la solution est bien 1D. Comparez votre solution numérique à la solution approchée 1D à 100 modes (calculée avec la méthode de séparation des variables et une décomposition en série de sinus)

$$\theta(t,x) = T_G + (T_D - T_G)\left(x + \sum_{n=1}^{100} \frac{2}{n\pi}(-1)^n \exp(-D(n\pi)^2 t)\sin(n\pi x)\right)$$

Pour cette comparaison, le plus simple est de calculer une erreur absolue dans chaque maille (différence entre la solution numérique et la solution exacte évaluée au centre de la maille), puis une erreur relative globale (avec une norme quelconque).

Si cela ne marche pas, modifiez votre code pour tester une solution constante (voir le TP1 et l'exercice 11).

Ensuite, faites une analyse de convergence en maillage. Pour cela, il vous faut raffiner le maillage : lisez la marche à suivre donnée en annexe A. Pour chaque maillage, rentrez à la main dans un fichier le nombre de mailles et l'erreur globale. Que constatez-vous ? Voir en annexe B pour une explication.

À la fin de ce cas-test, essayez de refaire vous même le maillage en totalité : construisez d'abord le fichier de géométrie (que vous nommerez différemment), en suivant la marche à suivre donné en annexe C. Comparez votre fichier à `cas_1d.geo` (avec un éditeur de texte). Générez ensuite le maillage. Relancez alors votre programme avec ce nouveau maillage.

Cas test 2 : solution axisymétrique dans un disque. Créez un maillage du disque de rayon $R = 2$ centré en $(0,0)$ pour le problème défini par les données suivantes : $T_{init} = 100$, condition de Dirichlet $T_b = 300$ sur le bord du disque, $D = 1$, $t_{max} = 0.1$.

Modifiez votre code pour qu'il résolve ce problème. Vous pouvez aussi paramétrer le code pour qu'il fonctionne pour encore pour le problème précédent : il suffit d'indiquer dans le fichier de données combien il y a de bords avec des CL différentes, et quelle valeur de T_b ou de ϕ_b est appliquée sur chacun de ces bords. Le code doit alors être capable de traiter ces données automatiquement.

Vérifiez que votre solution est bien axisymétrique. Vous pouvez ensuite comparer votre solution numérique à la solution exacte de l'équation 1D axisymétrique correspondante. Celle-ci peut se calculer avec une décomposition en série de fonctions de Bessel (voir le cours *Phénomènes de transfert* au S8) et l'on trouve :

$$T(t,r) = T_b + \sum_{n\geq 1} c_n e^{-\frac{z_n^2}{R^2}Dt} J_0(\frac{r}{R}z_n),$$

où les c_n sont des coefficients de la décomposition, et les z_n sont les zéros de la fonction de Bessel de première espèce J_0. Vous trouverez le code fortran correspondant dans la fonction `T1d_axi` et que vous pouvez intégrer à l'un de vos modules.

Une autre possibilité est de calculer une solution numérique de cette équation 1D axisymétrique via un petit programme utilisant un schéma numérique volumes finis pour cette équation 1D (le code n'est pas fourni).

Cas test 3 : solution axisymétrique dans une couronne. Créez un maillage de la couronne comprise entre un disque de rayon $R_{int} = 1$ centré en $(0, 0)$ et d'un disque de rayon $R_{ext} = 2$ centré au même point, pour problème défini par les données suivantes : $T_{init} = 1$, conditions de Dirichlet avec $T_b = 1$ sur le bord du disque intérieur et $T_b = 2$ sur le bord du disque extérieur, et $t_{max} = 1$.

Calculez la solution numérique de ce problème avec votre code. Là encore, il est possible de calculer une solution exacte. Il est cependant plus simple de calculer la solution exacte en régime stationnaire. Vous pouvez alors comparer à la solution numérique donnée par votre code : il vous faut vérifier que le régime stationnaire est bien atteint, par exemple en mesurant l'écart relatif entre T^n et T^{n+1}.

Cas test 4 : problème de thermique dans un domaine quelconque. Essayez de créer un domaine de type elliptique, avec deux trous de rayons différents. Faites en sorte de pouvoir utiliser une condition aux limites différentes sur chacun des 3 bords du domaine. Définissez une donnée initiale et des conditions aux limites, et testez votre code sur ce problème. Vous pouvez éventuellement utiliser aussi une solution manufacturée.

A Raffinement d'un maillage

On indique ici la marche suivre pour créer plusieurs maillages obtenus par raffinements successifs, pour le premier cas-test. NB : sur centospedago, l'appel à `gmsh` se fait avec la commande : `/opt/gmsh/3.0.2/bin/gmsh`.

Création du maillage

- lancez `gmsh`, puis sélectionnez le fichier `cas_1d.geo` avec `File / Open`
- cliquez sur `Mesh / 2D` : un premier maillage assez grossier, est automatiquement créé
- cliquez sur `Mesh / Refine by splitting` deux fois pour avoir un maillage plus fin
- si les triangles vous semblent peu admissibles, cliquez sur `Mesh / Smooth 2D` puis à nouveau sur `Mesh / 2D`. N'hésitez pas à itérer ce processus plusieurs fois.
- pour générer le fichier de maillage, cliquez sur `File / export / Format: Mesh-INRIA Medit (*.mesh)` puis donnez un nom à votre fichier de maillage
- dans la fenêtre de dialogue qui s'ouvre, sélectionnez `Mesh options : Physical entity` (si vous sélectionnez Elementary entity, vous n'aurez pas les codes de CL sur les bords)
- gardez la session `gmsh` ouverte

Raffinement du maillage

- Cliquez sur `Mesh / Refine by splitting`. Chaque triangle est alors divisé en 4 sous triangles semblables.
- Sauvegardez dans un fichier avec un nouveau nom pour ne pas perdre le maillage plus grossier (cf ci-dessus)
- Répétez cette opération pour chaque niveau de raffinement supplémentaire.

— Attention : si vous fermez `gmsh`, vous serez obligé de recommencer au début. Vous pourrez réouvrir le fichier de maillage précédent pour le raffiner, mais il semble que cela fasse perdre les codes de CL pour les arêtes de bord.

B Convergence du schéma

Vous devez constater que l'erreur est divisée par 2 à chaque nouveau raffinement de maillage. Cela est dû au fait que le schéma est d'ordre un : on peut montrer que l'erreur est majorée par une constante multipliée par une longueur caractéristique du maillage (par exemple la plus petite hauteur de tous les triangles). Quand on raffine une fois le maillage, l'aire des triangles est divisée par 4, et la hauteur est donc divisée par 2 (l'aire est proportionnelle au carré de la hauteur), et donc l'erreur aussi.

En traçant la courbe de convergence avec `gnuplot`, vous devez aussi constater qu'en échelle log, vous obtenez une droite de pente 1.

C Création d'une géométrie et d'un maillage

On décrit ici la création du maillage pour la couronne. Il vous sera facile de mailler ensuite n'importe quelle géométrie plus complexe. La marche à suivre est la suivante :

Création de la géométrie

— tapez `gmsh` dans un terminal
— cliquez sur `File / New` et donnez le nom `couronne.geo` à votre nouveau fichier (toutes les opérations que vous réalisez ensuite sont automatiquement sauvegardées dans le fichier)
— cliquez sur `Module / Geometry / Elementary entities / Add / circle` pour définir le premier cercle. Idem pour le second.
— cliquez sur `Module / Geometry / Elementary entities / Add / Plane Surface` puis sélectionnez le cercle extérieur, puis le cercle intérieur (c'est le "trou" mentionné par gmsh). Quittez ce mode en tapant « q »
— cliquez sur `Module / Physical groups / Add / Line` pour attribuer le code 10 au cercle intérieur : pour cela, donnez un nom au bord considéré (par exemple, "cercle_int", ce nom ne sera pas utilisé par la suite), décochez la case `Automatic Numbering`, et donnez le numéro que vous voulez attribuer à ce bord (ici, on suggère 10). N'oubliez pas de valider en tapant "e". Même procédure pour attribuer le n° 11 au cercle extérieur (pour pouvoir utiliser deux CL différentes)
— cliquez sur `Module / Physical groupes / Add / Surface` pour sélectionner le domaine (ici, la couronne) : il est ici possible de donner un code qui permette de choisir différentes CI pour différentes parties du domaine, mais cela ne sera pas utilisé dans notre cas.

Création du maillage

— cliquez sur `Mesh / 2D` : un premier maillage assez grossier, est automatiquement créé
— cliquez sur `Mesh / Refine by splitting` une ou deux fois pour avoir un maillage plus fin
— si les triangles vous semblent peu admissibles, cliquez sur `Mesh / Smooth 2D` puis à nouveau sur `Mesh / 2D`. N'hésitez pas à itérer ce processus plusieurs fois.
— pour générer le fichier de maillage, cliquez sur `File / export / Format: Mesh-INRIA Medit (*.mesh)` puis donnez un nom à votre fichier de maillage

— dans la fenêtre de dialogue qui s'ouvre, sélectionnez `Mesh options : Physical entity` (si vous sélectionnez Elementary entity, vous n'aurez pas les codes de CL sur les bords)