SeaTech École d'ingénieurs

Modélisation et Calculs Fluides et Structures

Compte Rendu de TP 2: Elements finis

Résolution de l'équation de la chaleur de 2D

Enseignant: M.Schneider

Etudiants: Dupont Ronan - Croguennec Guillaume

Année Universitaire 2019-2020

I) Introduction

Dans ce TP, nous allons nous intéresser à la résolution de l'équation de la chaleur en 2D. Pour se faire, nous utiliserons la méthode des éléments finies en utilisant des matrices de rigidité et élémentaires.
Pour se faire, nous coderons cette méthode en fortran 90.

II) Modèle Physique

Notre projet dans ce travail est d'obtenir la répartition de la chaleur dans une plaque dont les bords en haut et en bas sont maintenus à 100C et dont les bords sur les côtés sont maintenus à 0C.
On pose u la fonction dont la valeur est la température de la plaque en un point à un moment.
Dans cette plaque, la propagation de la chaleur est fixée par $-\Delta u = 0$ sur l'ensemble de la plaque.

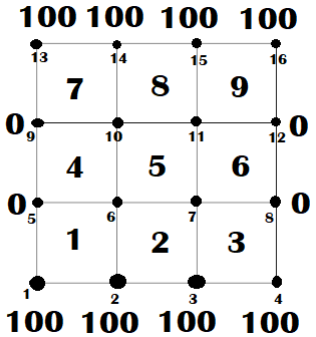
III) Modèle Numérique

Donc pour ce problème, nous devons résoudre l'équation suivante:

$$\begin{cases} -\Delta u = 0 \ sur \ \Omega = [0,10]^2 \\ u = y \ sur \ \partial\Omega \end{cases} \tag{1}$$

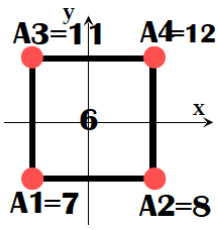
1) Discretisation

Pour obtenir les valeurs de u, nous procédons donc à une discrétisation du problème pour ensuite utiliser la méthode des éléments finis.
On discrétise donc ce problème en considérant n^2 points de la plaque, soit n lignes de n points sur la plaque. En quadrillant cette plaque avec les points, on se retrouve avec $(n-1)^2$ cases, que l'on appelle mailles. Ci-dessous la grille correspondant à n=4.



Ensuite, on peut entamer la méthode des éléments finis.
Pour cela, la première étape est d'associer à chaque maille les points qui la forme (ses quatre coins). Nous cherchons donc la **connectivité** de chaque maille.

2) Connectivité



Par exemple, en prenant la 6ème maille, on a les sommets respectifs de valeurs (7,8,11,12).
On pose k: l'élément de la maille, n: le nombre de noeuds dans la maille: ici 4.

On peut en déduire facilement la formule:
A1=q(k-1,n-1)+k
A2=q(k-1,n-1)+k+1
A3=q(k-1,n-1)+k+n
A4=q(k-1,n-1)+k+n+1
où q est le quotient.

Vérifions pour k=6:
A1=q(5,3)+6=7
A2=q(5,3)+6+1=8
A3=q(5,3)+6+4=11
A4=q(5,3)+6+4+1=12
Nous pouvons donc valider notre formule que nous utilisons dans un vecteur *lun* de taille 4 comme on peut voir figure 5.

3) Matrice élémentaire

Ensuite, il est nécessaire de créer la matrice élémentaire, qui va nous servir à obtenir la matrice de rigidité.
Pour trouver les $\tilde{F}_i$, nous posons par exemple $\tilde{F}_1 = a + bx + cy + dxy$. Puis à l'aide des conditions : $\tilde{F}_1(\tilde{A}1) = 1 \quad \tilde{F}_1(\tilde{A}2) = 0 \quad \tilde{F}_2(\tilde{A}2) = 0 \quad \tilde{F}_4(\tilde{A}2) = 0$
On trouve: $\tilde{F}_1$=1-x+y-xy
De même on trouve les autres $\tilde{F}_i$ et on peut en déduire la matrice suivante que nous écrivons dans la figure 5:

$$(\int \nabla\tilde{F}_i.\nabla\tilde{F}_j)_{i,j} = \frac{1}{6}\begin{pmatrix} 4 & -1 & -2 & -1 \\ -1 & 4 & -1 & -2 \\ -2 & -1 & 4 & -1 \\ -1 & -2 & -1 & 4 \end{pmatrix}$$

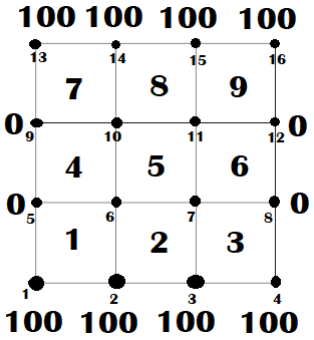
Nous créons cette matrice sur le programme en figure 5.Il ne nous reste donc plus qu'à implémenter la matrice de rigidité M.

4) Matrice de rigidité

Pour faire la matrice de rigidité, on parcourt les mailles, puis sur chaque maille on parcourt les doublets de sommets possibles, puis pour chaque doublet on ajoute quelque chose dans la matrice de rigidité.
Pour savoir où on ajoute quel terme dans la matrice, il faut savoir distinguer deux appellations. Le numéro d'un point, est son numéro dans la discrétisation de la plaque, tandis que son numéro de sommet par rapport à une maille est la combientième sommet il est pour la maille (en partant de en bas à gauche et en allant dans le sens trigonométrique).

donc pour chaque doublet de sommets de chaque maille, on au terme de la matrice de rigidité qui a pour coordonnée les numéros des deux points, le terme de la matrice élémentaire qui a pour coordonnée les les numéros de sommet des deux points par rapport à la maille.

Par exemple, dans le cas où on prend n=4, on va discrétiser la plaque en 16 points et on aura donc 9 mailles comme on peut voir sur la figure ci-dessous:



On va donc parcourir les mailles, en commençant par la première. Dans chacune de ces mailles, on va parcourir tous les doublets de sommets possibles comme on peut voir sur la figure 5.
C'est-à-dire :

- le sommet (1,1), donc les points 1 et 1
- le sommet (1,2), donc les points 1 et 2
- le sommet (1,3), donc les points 1 et 6
- le sommet (1,4), donc les points 1 et 5
- le sommet (2,1), donc les points 2 et 1
- le sommet (2,2), donc les points 2 et 2
- le sommet (2,3), donc les points 2 et 6
- le sommet (2,4), donc les points 2 et 5
- le sommet (3,1), donc les points 6 et 1
- le sommet (3,2), donc les points 6 et 2
- le sommet (3,3), donc les points 6 et 6
- le sommet (3,4), donc les points 6 et 5
- le sommet (4,1), donc les points 5 et 1
- le sommet (4,2), donc les points 5 et 2
- le sommet (4,3), donc les points 5 et 6
- le sommet (4,4), donc les points 5 et 5

Puis pour chacun de ces doublets on va ajouter un terme à la matrice de rigidité. Par exemple, pour le troisième de ces cas, on va ajouter le terme (1,3) de la matrice élémentaire au terme (1,6) de la matrice de rigidité.

Un fois que cela est fait pour chaque doublet de chaque maille, la matrice de rigidité est faite. Mais elle ne tient pas encore compte des conditions limites, donc dans ce cas les températures sur les bords.
Pour pallier cela, une des méthodes possibles est la pénalisation.

5) Pénalisation

Cette méthode permet d'avoir une solution convergente, elle consiste à ajouter, sur le terme de la diagonale de la ligne de la matrice de rigidité correspondant au point avec la condition limite, un nombre assez grand pour que les autres termes sur cette ligne de la matrice soient négligeables. Puis à ajouter sur la même ligne du second membre ce même nombre multiplié par la valeur que l'on veut imposer en ce point.
Par exemple, si on veut imposer 100 degrés à u_1 , et en choisissant 10^6 comme nombre de pénalisation, on aura comme pour la première ligne l'équation $10^6 u_1 = 100.10^6$, u_1 sera donc égal à 100 degrés.
On peut voir cette pénalisation dans notre programme figure 6.

IV) Résultats

1) Diffusion en image

Une fois notre programme fonctionnel, nous avons donc tracé à l'aide du module gnuplot et sa fonction "plot with matrix" les différentes diffusions pour des maillages de tailles $n = 10, 20, 100$.

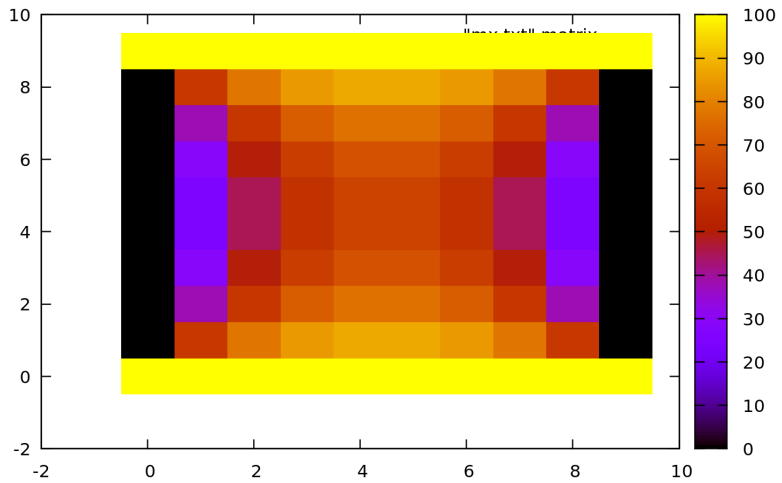


Figure 1: Diffusion pour n=10

On remarque que pour $n = 10$, la diffusion est pas très précise car nous avons des sauts d'environ 10 degrés d'un élément du maillage à un autre. Sinon on voit bien que les conditions de bords sont bien respectés.

Le temps d'exécution pour obtenir ces valeurs est de t=7.75e-3s.

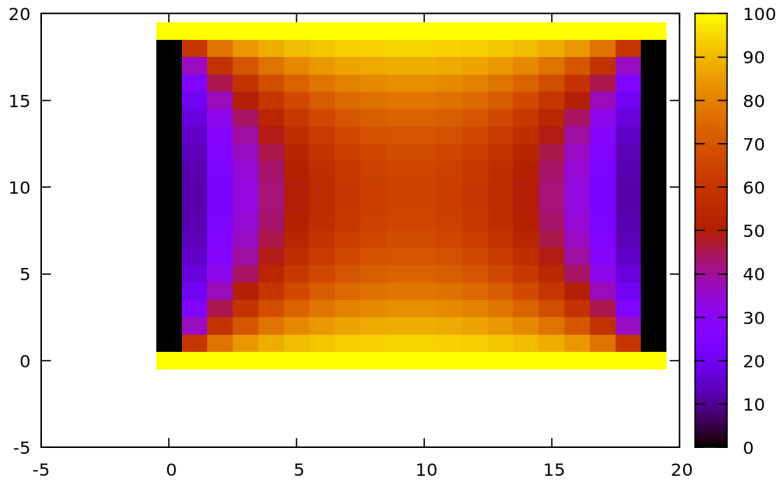


Figure 2: Diffusion pour n=20

De même pour $n = 20$, la diffusion est bien plus précise car nous avons des sauts de plus en plus petit d'un élément du maillage à un autre. De même, les conditions de bord sont bien respectés.

Le temps d'exécution pour obtenir ces valeurs est de t=0.10s.

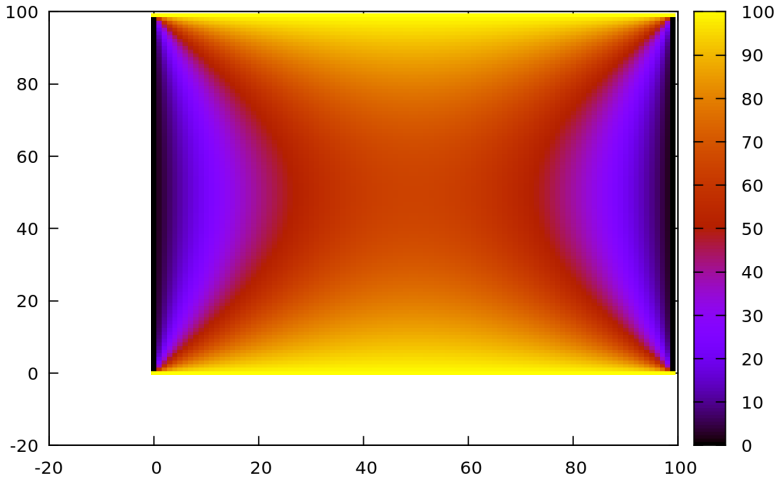


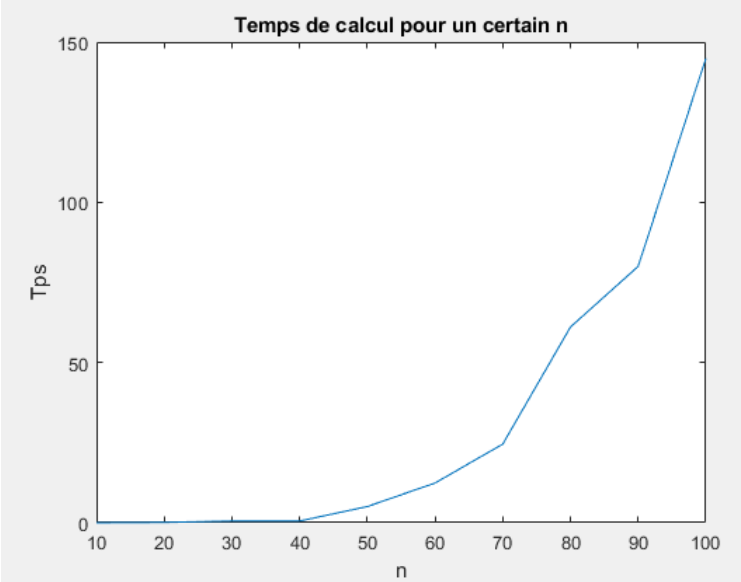
Figure 3: Diffusion pour n=100

De même pour $n = 100$, la diffusion est très précise. Les conditions de bord sont toujours respectés.

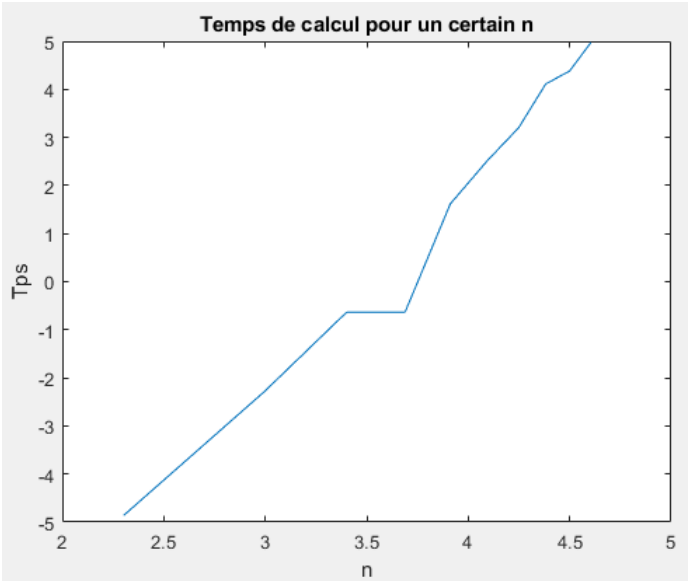
Le temps d'éxecution pour obtenir ces valeurs est ici bien plus long que les deux précédents, il est de t=145s.

2) Complexité temporelle

Nous avons ensuite cherché à déterminer la complexité temporelle de notre programme. Pour se faire, nous avons tracé le temps de calcul en fonction des différents n:



On observe une croissante de manière très rapide, nous allons donc tracer en échelle logarithmique:



On observe graphiquement que le coefficient directeur est de 4. Nous avons donc une complexité temporelle en o(n^4). Ceci vient du programme de résolution SOLP qui a une complexité en o(n^3). En outre, nous l'utilisons avec une matrice de taille (n^2, n^2), nous devrions donc avoir une complexité temporelle en o($(n^2)^3$) soit ((n^6). On peut expliquer ceci par le fait que la matrice contient beaucoup de zéros et donc la complexité temporelle de SOLP peut sûrement se ramener à du o(n^2) d'où la complexité en o(n^4).

V) Conclusion

Dans ce TP, nous avons résolu l'équation de la chaleur en 2D. Nous avons utilisé la méthode des éléments finis qui est une méthode très puissante avec des algorithmes à la pointe car ils permettent de résoudre des problèmes 2D avec un simple système matriciel. L'approche théorique peut être assez complexe au premier abord mais elle permet une fois la théorie posée de résoudre des problèmes beaucoup plus complexes.

Annexe

```
 !-- Declarations
  implicit none
  integer                 :: i,j,k,ii,jj
  integer                 :: n
  real(kind=kind(0.d0)) :: h
  real(kind=kind(0.d0)), dimension(:,:),allocatable :: A,M
  real(kind=kind(0.d0)), dimension(:,:),allocatable :: Aaux,E
  real(kind=kind(0.d0)), dimension(:),allocatable :: b,x
  real(kind=kind(0.d0)), dimension(:),allocatable :: lun

n=10
h=1.d0/(n-1)

allocate(A(4,4))
allocate(M(n*n,n*n))
allocate(E(n,n))
allocate(Aaux(n*n,n*n))
allocate(b(n*n))
allocate(x(n*n))
allocate(lun(n))
```

Figure 4: Declaration des variables

```
 ! Matrice elementaire A
do i=1,4
  A(i,i)=4.d0/6
  A(i,i+1)=-1.d0/6
  A(i,i-1)=-1.d0/6
  A(i,i+2)=-2.d0/6
  A(i,i-2)=-2.d0/6
  A(i,i+3)=-1.d0/6
  A(i,i-3)=-1.d0/6
enddo


! on fait la matrice de rigidite M
do k=1,(n-1)*(n-1)
  lun(1)=(k-1)/(n-1)+k
  lun(2)=(k-1)/(n-1)+k+1
  lun(3)=(k-1)/(n-1)+k+n
  lun(4)=(k-1)/(n-1)+k+1+n
    do i=1,4
      do j=1,4
        M(int(lun(i)),int(lun(j)))=M(int(lun(i)),int(lun(j)))+A(i,j)
      enddo
    enddo
enddo
```

Figure 5: Matrice elementaire et de rigidité

```
 ! Penalisation
do i=1,n !Bord horizontal
  M(i,i)=M(i,i)+10**6
  M(n*n-i+1,n*n-i+1)=M(n*n-i+1,n*n-i+1)+10**6
enddo
do i=2,n-1 ! Bord vertical
  M(1+n*(i-1),1+n*(i-1))=M(1+n*(i-1),1+n*(i-1))+10**6
  M(n+n*(i-1),1+n*(i-1))=M(n+n*(i-1),1+n*(i-1))+10**6
enddo

! b avec penalisation
b(1:n)=100*10**6
b(n*n-n+1:n*n)=100*10**6
! On resout le systeme
CALL SOLP(1,M,b,1,1,n*n,x)
do i=1,n*n
  print*,x(i)
enddo
```

Figure 6: Penalisation et resolution du système

```
 ! Ecriture de la matrice de temp rature
DO I=1,n
  DO J=1,n
    E(i,j)=x((i-1)*n+j)
  ENDDO
ENDDO
! Ecriture de xk dans un fichier sous forme de matrice de temperature
OPEN(UNIT=48,FILE='mx.txt')
DO i=1,n
  WRITE(48,*) (E(i,j),j=1,n)
ENDDO
CLOSE(48)
```

Figure 7: Ecriture de la matrice de température

```
 SUBROUTINE SOLP   (NSYM,VKG,VFG,IKG,IFG,NEQ,VU)
!============================ DEBUT DES DECLARATIONS ====================
implicit none
real(kind=8)::vzero=0.d0,cl,rpiv
real(kind=8),allocatable,dimension(:,:)::vkg
real(kind=8),allocatable,dimension(:)::VFG,VU
integer :: NSYM,IKG,IFG,NEQ, i,j,ij,n1,ij1,is,is1,ii
!-------  Triangularisation
      IF(IKG.EQ.0.AND.IFG.EQ.0) GOTO 9999
      N1=NEQ-1
      DO 50 IS=1,N1
      RPIV=VKG(IS,IS)
      IF (RPIV) 10,800,10
10    IS1=IS+1
      DO 50 II=IS1,NEQ
      IF(NSYM.EQ.1) CL=VKG(II,IS)
      IF(NSYM.NE.1) CL=VKG(IS,II)
      IF(CL.EQ.VZERO) GOTO 50
      CL=CL/RPIV
      IF(IFG.EQ.1) VFG(II)=VFG(II)-CL*VFG(IS)
      IF(IKG.EQ.0) GOTO 50
      IF(NSYM.EQ.1) GOTO 32
      DO 30 IJ=II,NEQ
      VKG(II,IJ)=VKG(II,IJ)-CL*VKG(IS,IJ)
30    VKG(IJ,II)=VKG(II,IJ)
      GOTO 50
32    DO 40 IJ=IS1,NEQ
40    VKG(II,IJ)=VKG(II,IJ)-CL*VKG(IS,IJ)
50    CONTINUE
!-------  RESOLUTION DU SYSTEME TRIANGULAIRE
      IF(VKG(NEQ,NEQ).NE.VZERO) GOTO 55
      IS=NEQ
      GOTO 800
55    IF(IFG.EQ.0) GOTO 9999
      VU(NEQ)=VFG(NEQ)/VKG(NEQ,NEQ)
      DO 70 II=1,N1
      IS1=IS1-1
      CL=VZERO
      IJ1=IS1+1
      DO 60 IJ=IJ1,NEQ
60    CL=CL+VKG(IS1,IJ)*VU(IJ)
70    VU(IS1)=(VFG(IS1)-CL)/VKG(IS1,IS1)
!-------  Fin SUBROUTINE SOLP
      GOTO 9999
!-------  Erreurs
800    WRITE(*,8000) IS
8000  FORMAT('␣*␣SOLP␣␣-E-RPIVNUL␣Pivot␣nul␣equation',I5)
      GOTO 9999
!-------  Fin
9999  CONTINUE
      RETURN
!========   FIN DU MODULE SOLP      ====
      END SUBROUTINE SOLP
```


Figure 8: Subroutine Résolution par Pivot de Gauss