Résolution Numérique de quelques Équations aux Dérivées Partielles en utilisant la Méthode des Différences Finies et la Méthode des Volumes Finis

Réalisé par : ADIB Mehdi - HANI Moad
Filière : GMI3
Module :Volumes finis
Sous l'encadrement du professeur : Monsieur Ahmed TAIK
Faculté de Sciences et Techniques de Mohammedia

10 janvier 2018

Sommaire

Introduction **3**

1 Rappel sur la méthode des différences finies (DF) **4**
1.1 Cas 1D : 4
1.1.1 Discrétisation du domaine [A,B] 5
1.1.2 Schéma numérique(DF) 5
1.2 Code : 6
1.2.1 Résultats graphiques 8

2 Résolution de l'équation de Laplace 2D par la méthode des Différences Finies **9**
2.1 Définitions 9
2.2 Discrétisation de l'EDP 11
2.3 Approximation de l'EDP 11
2.4 Calcul numérique 12
2.4.1 Principe 12
2.4.2 Algorithme1 13
2.4.3 Résultats graphiques 16

3 Résolution de l'équation de Laplace 2D avec des conditions aux limites mixtes **17**
3.1 Approximation de l'EDP : 17
3.2 Code Matlab 19
3.3 Résultats graphiques 21

4 Résolution de l'équation de Laplace 3D par la méthode des Différences Finies **22**
4.1 Discrétisation du domaine $]0,1[^3$ 22
4.2 Schéma numérique(DF) 23
4.3 Calcul numérique 23
4.3.1 Algorithme 23
4.3.2 Résultats graphiques 32

5 Résolution de l'équation de Laplace 2D par la méthode RBF **33**
5.1 Généralités . . . 33
5.2 Approximation de l'EDP : . . . 34
5.3 Code Matlab . . . 35
5.4 Résultats graphiques . . . 38

6 Conduction de chaleur par maillage structuré **39**
6.1 Position du problème . . . 39
6.2 Maillage . . . 39
6.3 Construction du schéma . . . 41
6.4 Calcul des flux numériques . . . 42
6.4.1 Flux interne : σ est une arête interne . . . 42
6.4.2 Flux au bord de Γ_2 . . . 42
6.4.3 Flux au bord de Γ_4 . . . 43
6.4.4 Flux au bord de $\Gamma_1 \cup \Gamma_3$ . . . 43
6.4.5 Flux sur l'interface I . . . 43
6.5 Calcul numérique . . . 44
6.5.1 Principe . . . 44
6.5.2 Code Matlab . . . 46
6.5.3 Résultats graphiques . . . 52

7 Conduction de chaleur par maillage non structuré **53**
7.1 Maillage . . . 53
7.2 Construction du schéma . . . 54
7.3 Calcul des flux numériques . . . 55
7.3.1 Flux interne : σ est une arête interne . . . 55
7.3.2 Flux au bord de $\sigma \subset \Gamma 1 \bigcup \Gamma 3$ . . . 55
7.3.3 Flux au bord de $\sigma \subset \Gamma 2$ . . . 56
7.3.4 Flux au bord de $\sigma \subset \Gamma 4$ . . . 56
7.3.5 Flux sur l'interface I . . . 56
7.4 Calcul numérique . . . 57
7.4.1 Principe . . . 57
7.4.2 Algorithme . . . 58
7.4.3 Résultats graphiques . . . 63

Conclusion **64**

Introduction

Le métier d'ingénieur ne se conçoit guère sans l'utilisation de logiciels de modélisation. Ces logiciels résolvent des équations telles que l'équation de continuité, de conservation de la quantité de mouvement, de conservation de l'énergie ou de conservation de la masse.

Du fait de la complexité de la géométrie, ainsi que de la variation dans le temps ou dans l'espace des conditions aux limites, ces équations différentielles ne peuvent en général pas être résolues de façon exacte. Elles sont résolues de façon approchée, à l'aide des méthodes numériques.

Les méthodes numériques ne donnent pas la solution véritable du problème que l'on cherche à résoudre. Des méthodes numériques mal employées peuvent conduire à des résultats totalement faux, allant à l'encontre de la réalité physique (exemples typiques : concentrations négatives, création ou disparition artificielle de la masse d'eau ou de soluté dans un modèle).

L'apparition d'ordinateurs extrêmement puissants permet néanmoins aujourd'hui d'obtenir des solutions approchées pour des équations aux dérivées partielles, même très compliquées.

Chapitre 1

Rappel sur la méthode des différences finies (DF)

1.1 Cas 1D :

Différences finies

La méthode des différences finies est une méthode de discrétisation d'une équation différentielle ou d'un ensemble de relations aux dérivées partielles en un ensemble discret de 'N' équations à 'N' inconnues.

Cette méthode s'inscrit dans le processus général de discrétisation d'un problème physique au niveau de l'étape 3 :

1. Modélisation de la physique du problème * choix des équations régissant le problème AVEC conditions limites et initiales, * choix du domaine physique, * loi de comportement, * propriétés physiques, * autres hypothèses ...

2. Représentation discrète de la géométrie : génération d'un maillage

3. Obtention de l'équation d'équilibre discrète Cette étape consiste à remplacer tous les termes de dérivées sous une forme discrètes à l'aide d'une technique basée sur les développements limités.

4. Assemblage du système discret Application de l'équation discrète sur l'ensemble des noeuds autorisés pour obtenir le système suivant : [A]X = B. (1)

5. Mise en place des conditions limites de type Dirichlet sur le système d'équations

6. Résolution Pour un problème linéaire, il s'agit de trouver : X = $[A]^{-1}B$. (2)

7. Post-traitement * Affichage déformée, * Calcul du champ de gradients pour affichage des champs de contrainte et de déformation * ...

8. Analyse des résultats

On considère l'équation différentielle ordinaire :

$$\begin{cases} a(x)y"(x) + b(x)y'(x) + c(x)y(x) = f(x) \text{ pour tout } x \in]A, B[\\ y(A) = \alpha \\ y(B) = \beta \text{ avec } A, B \in \mathbb{R},\ \alpha, \beta \in \mathbb{R} \end{cases}$$

Généralement la résolution analytique de ce type d'équation différentielle ordinaire (EDO) est difficile :

Numériquement, c'est toujours possible, en effet :

1.1.1 Discrétisation du domaine [A,B]

$$\frac{B-A}{n} \text{ avec } n : \text{le pas de discrétisation}$$
$$x(i) = x_i = A + ih,\ i = 0, ..., n$$

1.1.2 Schéma numérique(DF)

$$y'(x_i) \simeq \frac{y(x_{i+1}) - y(x_i)}{h} = \frac{y_{i+1} - y_i}{h}$$
$$y"(x_i) = (y(x_i)')' \simeq \frac{y'(x_{i+1}) - y'(x_i)}{h}$$
$$= \frac{\frac{y(x_{i+2}) - y(x_{i+1})}{h} - \frac{y(x_{i+1}) - y(x_i)}{h}}{h}$$
$$\simeq \frac{y_{i+2} - 2y_{i+1} + y_i}{h^2}$$
$$y"(x_i) \simeq \frac{y_{i+2} - 2y_{i+1} + y_i}{h^2}$$

(EDO) continue écrite pour tout $x_i \in]A, B[,\ i = 1, ..., n-1$:

$$a_i \frac{y_{i+2} - 2y_{i+1} + y_i}{h^2} + b_i \frac{y_{i+1} - y_i}{h} + c_i y_i = f_i, \text{ pour tout } i = 1, ..., n-1$$

$$y(x_0) = y(A) = y_0 = \alpha$$
$$y(x_n) = y(B) = y_n = \beta$$

$$(\frac{a_i}{h^2} + \frac{b_i}{h})y_{i+1} + (-\frac{2a_i}{h^2} - \frac{b_i}{h} + c_i)y_i + \frac{a_i}{h^2}y_{i-1} = f_i \ i = 1, ..., n-1$$

$$\iff r_i y_{i+1} + s_i y_i + t_i y_{i-1} = f_i \ \ i = 1, ..., n-1$$

$$pour\ i = 1 :\ r_1 y_2 + s_1 y_1 = f_1 - t_1 y_0$$
$$pour\ i = 2 :\ r_2 y_3 + s_2 y_2 + t_2 y_1 = f_2$$
$$pour\ i = 3 :\ r_3 y_4 + s_3 y_3 + t_3 y_2 = f_3$$

$$pour\ i = n-1 :\ s_{n-1} y_{n-1} + t_{n-1} y_{n-2} = f_{n-1} - r_{n-1}\beta$$

En utilisant les DF, la résolution numérique de l'EDO revient à résoudre le système linéaire de (n-1) inconnues $y_1, y_2, ..., y_{n-1}$ et (n-1) équations, chose équivalente à résoudre le système matriciel suivant :

$$A_h Y_h = F_h$$
$$\Rightarrow Y_h = A_h^{-1} F_h$$

$$\begin{pmatrix} s_1 & r_1 & 0 & \dots & 0 & 0 \\ t_2 & s_2 & r_2 & \dots & \dots & \dots \\ 0 & t_3 & s_3 & r_3 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & s_{n-2} & r_{n-2} \\ 0 & \dots & \dots & \dots & t_{n-1} & s_{n-1} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ \vdots \\ y_{n-1} \end{pmatrix} = \begin{pmatrix} f_1 - t_1\alpha \\ f_2 \\ f_3 \\ \vdots \\ \vdots \\ f_{n-1} - r_{n-1}\beta \end{pmatrix}$$

Une fois Y_h est déterminé on s'occupera de la représentation graphique de y en fonction de x.

1.2 Code :

Code matlab

```
%Ce programme resoud EDO 2eme ordre sur [A, B] en utilisant la MDF
%-------------------------------------------------------------------
clear;
clc;
%-------------------------------------------------------------------
%Input
A = 0;
B = 1;
alpha = 2;
beta = exp(1)+exp(2);
%Pre-processeur
```

```
n = 10;
h= (B-A)/n;
X = A:h:B;
%Schema numerique
r = zeros(1,n-1);
s = zeros(1,n-1);
t = zeros(1,n-1);
Ah = zeros(n-1);
Fh = zeros(1,n-1);
for i=1:(n-1)
    r(i) = ((X(i+1))^2+1)/(h^2)+X(i+1)/h;
    s(i) = -2*(((X(i+1))^2+1)/(h^2))-X(i+1)/h+1;
    t(i) = ((X(i+1))^2+1)/(h^2);
end
Ah(1,1)=s(1); Ah(1,2)=r(1); Ah(n-1,n-2)=t(n-1); Ah(n-1,n-1)=s(n-1);
Fh(1)=(X(2)^2+X(2)+2)*exp(X(2))+(4*(X(2)^2)+2*X(2)+5)*exp(2*X(2))-t(1)*alpha;
Fh(n-1)=(X(n)^2+X(n)+2)*exp(X(n))+(4*(X(n)^2)+2*X(n)+5)*exp(2*X(n))-r(n-1)*beta;
for i=2:(n-2)
    Ah(i,i)=s(i);
    Ah(i,i-1)=t(i);
    Ah(i,i+1)=r(i);
    Fh(i)=(X(i+1)^2+X(i+1)+2)*exp(X(i+1))+(4*(X(i+1)^2)+2*X(i+1)+5)*exp(2*X(i+1));
end
%Resolution
Yh = Ah\(Fh');
%Post-processeur
Ynum = zeros(1,n+1);
Ynum(1)=alpha; Ynum(length(X))=beta;
for i=2:(length(X)-1)
    Ynum(i)=Yh(i-1);
end
%Validation
%Solution exacte
X1 = A:0.01:B;
Yex = zeros(1,length(X1));
for i=1:length(X1)
    Yex(i)=exp(X1(i))+exp(2*X1(i));
end

a=figure(1);
plot(X,Ynum,'r'),title('Solution approchee')
saveas(a,'Solapprochee.eps','epsc');
hold off;
c=figure(2);
plot(X1,Yex,'b'),title('Solution exacte')
saveas(c,'Solexacte.eps','epsc');
```

1.2.1 Résultats graphiques

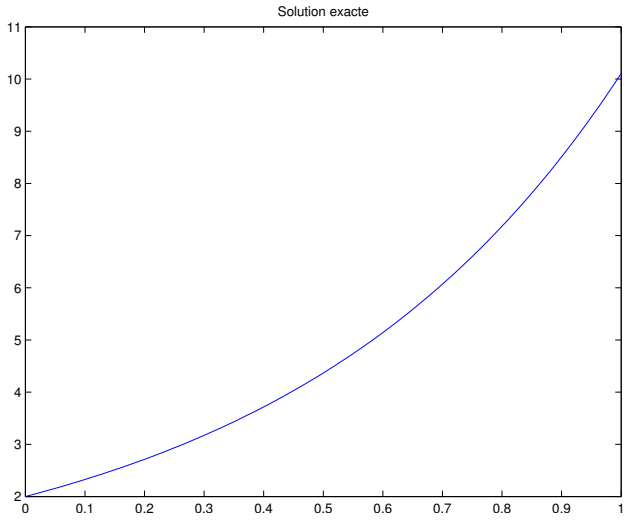


FIGURE 1.1 – Solution exacte

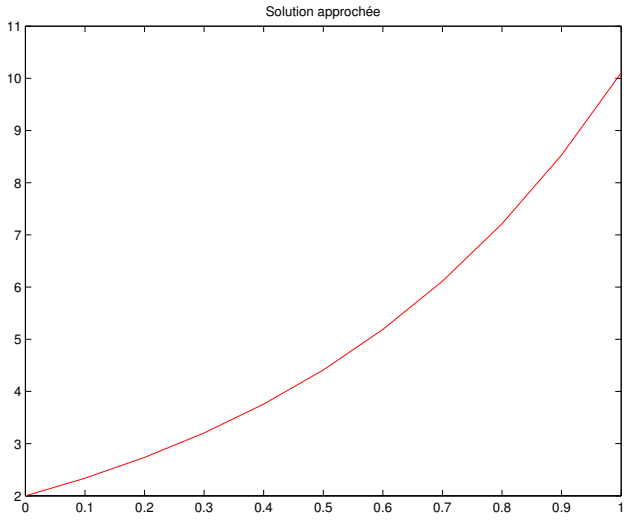


FIGURE 1.2 – Solution approchee

Chapitre 2

Résolution de l'équation de Laplace 2D par la méthode des Différences Finies

2.1 Définitions

Système global

Le principe de toute méthode numérique de discrétisation repose sur l'écriture d'un système comportant autant d'équations que d'inconnues. Ce système est dépendant du découpage du domaine (maillage) de calcul. En toute généralité, ce système ne peut être construit directement mais résulte d'un assemblage de systèmes d'équations locaux : * en un noeud et ses voisins pour les différences finies, * en une cellule et ses voisines en volumes finis, * sur un élèment en élèments finis.

Vecteur sollicitation

Le vecteur des sollicitations résulte de l'intégration de toutes les sollicitations externes à un systéme (mécanique, thermique ...) et représente le second membre du système d'équations global à résoudre pour obtenir la solution du problème traité.

Schéma

La notion de schéma est ici abordée au sens des méthodes numériques, à savoir en termes de schèmas de discrètisation.

1. les schémas de discrétisation spatiale pour l'étude des problèmes stationnaires (ou statique pour le mécanicien des structures). 2. les schémas de discrétisation temporelle d'ordre 1 ou 2 pour l'étude respective des problèmes transitoire (thermique,

fluide) et dynamique (mécanique des structures). Au sein même de ces schémas, on peut distinguer les schémas explicites, implicites, semi-implicites ... Il peut s'écrire sous la forme récurrente ou les indices n-1, n et n+1 désignent des instants successifs de calculs.

3. les schémas spatio-temporels, c'est-à-dire regroupant un un seul schéma, les caractèristiques des deux précedents.

Les schémas résultent de techniques de discrétisation telles les méthodes des différences finies, des volumes finis ou encore des éléments finis.

Discret

On parle d'une équation discrète ou d'un domaine discret pour spécifier des entités en nombres finis. La notion de 'discret' s'oppose à la notion de 'continu' qui elle reprèsente aussi bien un domaine géomètrique ou un système d'équations définis en tout point du domaine (infinité de points géométriques et de valeurs).

Nœud

Les noeuds sont des points de coordonnées du domaine étudié en lesquels sont définies les valeurs d'une fonction, utilisées lors d'une l'approximation nodale par exemple. On les nomme noeuds d'approximation.

Des noeuds peuvent egalement servir de support à la definition des elements dans le cadre de la méthode des elements finis. Ce sont des noeuds géométriques .

Les noeuds d'approximation et les noeuds géométriques peuvent etre distincts ou confondus.

La figure ci-dessous illustre un maillage simple comportant 8 noeuds.

Coordonnées

Les coordonnées d'un point ou d'un noeud définissent sa localisation spatiale dans un espace donné (cartésien, cylindrique, polaire ...).

On souhaite résoudre l'EDP suivante :

$$\begin{cases} \Delta u = f(x,y) \\ (x,y) \in [0,20],[0,10] \end{cases}$$

Avec les conditions aux limites suivantes :

$$\begin{cases} u(0,y) = y^2 = f_1(y) \\ u(20,y) = 400 + y^2 = f_2(y) \\ u(x,0) = x^2 = g_1(x) \\ u(x,10) = x^2 + 100 = g_2(x) \end{cases}$$

Pour résoudre cette équation avec la méthode des différences finies, il faut suivre les étapes suivantes :

2.2 Discrétisation de l'EDP

Soit :

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x_i, y_j), \forall (x, y) \in [0, 20] \times [0, 10]$$

On prendra h_x et h_y les pas de discrétisation des intervalles [0,20] et [0,10]

1. Discrétisation de l'intervalle [0,20]
   $h_x = \frac{b-a}{n_x}$ (n_x étant le nombre d'intervalles dans [0,20])
   $\Longrightarrow x(i) = x_i = 0 + i \times h_x, i = 0, 1, ..., n_x$
2. Discrétisation de l'intervalle [0,10]
   $h_y = \frac{d-c}{n_x}$ (n_y étant le nombre d'intervalles dans [0,10])
   $\Longrightarrow y(j) = y_j = 0 + j \times h_y, j = 0, 1, ..., n_y$

Remarque1. : Constatons que $x_{i+1} = 0 + (i+1)h_x = (0 + ih_x) + h_x = x_i + h_x$. Dans la suite, nous remplacerons chaque fois $x_i + h_x, x_i - h_x, y_j + h_y, y_j - h_y$ par $x_{i+1}, x_{i-1}, y_{i+1}, y_{i-1}$.

2.3 Approximation de l'EDP

Notre équation de départ est :

$$\Delta u = f(x, y) \Leftrightarrow \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y)$$

Posons $u(x_i, y_j) = u_{i,j}$ (en notation indicielle).On va maintenant approximer les dérivées secondes de u par rapport à x et y avec la formule de Taylor, on commence avec :

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{x_{i+1}, y_j} - 2u_{x_i, y_j} + u_{x_{i-1}, y_j}}{h_x^2}$$

Puisque x_i et y_j jouent un rôle symétrique dans l'équation de Laplace, un raisonnement analogue à celui de l'approximation de $u"_x$ nous donne :

$$\frac{\partial^2 u}{\partial y^2} \approx \frac{u_{x_i, y_{j+1}} - 2u_{x_i, y_j} + u_{x_i, y_{j-1}}}{h_y^2}$$

rapportons ces approximations dans notre EDP de départ, on obtient :

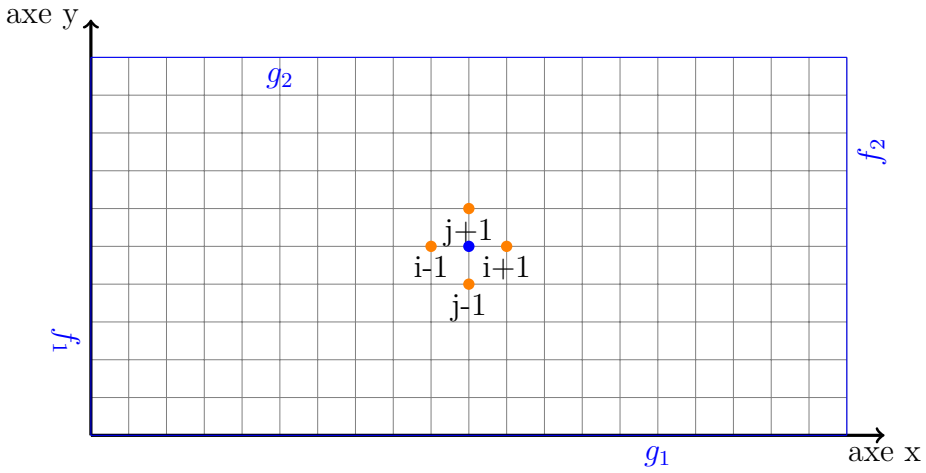
$$\Leftrightarrow \Delta u \approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h_x^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h_y^2} = f(x_i, y_j)$$

Dans ce cas particulier ou $h_x = h_y = h$, donc nous avons finalement :

$$\begin{cases} \Delta u = f(x_i, y_j) \Leftrightarrow \frac{u_{i+1,j}+u_{i,j+1}-4u_{i,j}+u_{i-1,j}+u_{i,j-1}}{h^2} = f(x_i, y_j) \\ i = 0, 1, ..., n_x \text{ et } j = 0, 1, ..., n_y \end{cases}$$

Remarque2. : Pour la numérotation il s'avère l'égitime d'utiliser cette relation qui nous aide à associer à chaque noeud de coordonnées (i,j) un numéro noté N tel que : $(i, j) \rightarrow N = i + (j - 1)(n_x - 1)$

Remarque3. :A chaque étape, nous remarquons que pour calculer la valeur de $u_{i,j}$ au point (x_i, y_j) nous avons besoin de connaitre les points $u_{i-1,j}$, $u_{i,j-1}$, $u_{i+1,j}$ et $u_{i,j+1}$ comme l'indique le dessin suivant :



la formule à 5 points qui peut être représentée comme suit :

$$\Delta u = f(x_i, y_j) \Rightarrow \frac{1}{h^2} \begin{Bmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{Bmatrix} u_{i,j} = f(x_i, y_j)$$

2.4 Calcul numérique

2.4.1 Principe

Nous obtenons un système linéaire à $(n_x - 1).(n_y - 1)$ équations et $(n_x - 1).(n_y - 1)$ inconnues de la forme $A_h.U_h = B_h$, que nous pouvons résoudre numériquement en inversant la matrice carré A_h :

$$U_h = A_h^{-1}.B_h$$

2.4.2 Algorithme1

Code matlab

```
%Ce programme resoud une EDP 2eme ordre sur [A,B]x[C,D] en utilisant la MDF
%-------------------------------------------------------------------------
clear
clc
%-------------------------------------------------------------------------
%Input
A = 0;
B = 20;
C = 0;
D = 10;
%Pre-processeur
nx = 40;
ny = nx/2;
h = (B-A)/nx;
X = A:h:B;
Y = C:h:D;
f1 = @(y) y^2;
f2 = @(y) 400+y^2;
g1 = @(x) x^2;
g2 = @(x) 100+x^2;
%Schema numerique
Ah = zeros((nx-1)*(ny-1));
F = zeros(1,(nx-1)*(ny-1));
m = nx-1;
%1er cas
for i=2:(nx-2)
    for j=2:(ny-2)
        N=i+(j-1)*m;
        Ah(N,N)=-4; Ah(N,N-1)=1; Ah(N,N+1)=1;
        Ah(N,N+m)=1; Ah(N,N-m)=1; F(N)=4*(h^2);
    end
end
%2eme cas
i=1; j=1;
N=i+(j-1)*m;
Ah(N,N)=-4; Ah(N,N+1)=1;
Ah(N,N+m)=1; F(N)=4*(h^2)-f1(Y(j+1))-g1(X(j+1));
%3eme cas
j=ny-1;
for i=2:(nx-2)
    N=i+(j-1)*m;
    Ah(N,N)=-4; Ah(N,N-1)=1; Ah(N,N+1)=1;
    Ah(N,N-m)=1; F(N)=4*(h^2)-g2(X(i+1));
end
%4eme cas
i=1;
for j=2:(ny-2)
    N=i+(j-1)*m;
    Ah(N,N)=-4; Ah(N,N+1)=1;
    Ah(N,N+m)=1; Ah(N,N-m)=1; F(N)=4*(h^2)-f1(Y(j+1));
end
%5eme cas
```

```
i=1; j=ny-1;
N=i+(j-1)*m;
Ah(N,N)=-4; Ah(N,N+1)=1;
Ah(N,N-m)=1; F(N)=4*(h^2)-f1(Y(j+1))-g2(X(i+1));
%6eme cas
j=1;
for i=2:(nx-2)
    N=i+(j-1)*m;
    Ah(N,N)=-4; Ah(N,N-1)=1; Ah(N,N+1)=1;
    Ah(N,N+m)=1; F(N)=4*(h^2)-g1(X(i+1));
end
%7eme cas
i=nx-1; j=1;
N=i+(j-1)*m;
Ah(N,N)=-4; Ah(N,N-1)=1;
Ah(N,N+m)=1; F(N)=4*(h^2)-g1(X(i+1))-f2(Y(j+1));
%8eme cas
i=nx-1;
for j=2:(ny-2)
    N=i+(j-1)*m;
    Ah(N,N)=-4; Ah(N,N-1)=1;
    Ah(N,N+m)=1; Ah(N,N-m)=1; F(N)=4*(h^2)-f2(Y(j+1));
end
%9eme cas
i=nx-1; j=ny-1;
N=i+(j-1)*m;
Ah(N,N)=-4; Ah(N,N-1)=1;
Ah(N,N-m)=1; F(N)=4*(h^2)-g2(X(i+1))-f2(Y(j+1));
%Resolution
Ysol=Ah\(F');
Yh=transpose(reshape(Ysol,nx-1,ny-1));
%Post-processeur
Ynum=zeros(ny+1,nx+1);
Ynum(1,nx+1)=400; Ynum(ny+1,1)=100; Ynum(ny+1,nx+1)=500;
for j=2:(length(X)-1)
    Ynum(1,j)=X(j)^2;
end
for j=2:(length(X)-1)
    Ynum(ny+1,j)=100+X(j)^2;
end
for i=2:(length(Y)-1)
    Ynum(i,1)=Y(i)^2;
end
for i=2:(length(Y)-1)
    Ynum(i,nx+1)=400+Y(i)^2;
end
for i=2:(length(Y)-1)
    for j=2:(length(X)-1)
        Ynum(i,j)=Yh(i-1,j-1);
    end
end
%validation
%Solution exacte
[U,V]=meshgrid(X,Y);
Yex=U.^2+V.^2;
```

```
108 subplot(1,2,1), mesh(U,V,Ynum), title('Solution approchee')
109 subplot(1,2,2), mesh(U,V,Yex), title('Solution exacte')
```

2.4.3 Résultats graphiques

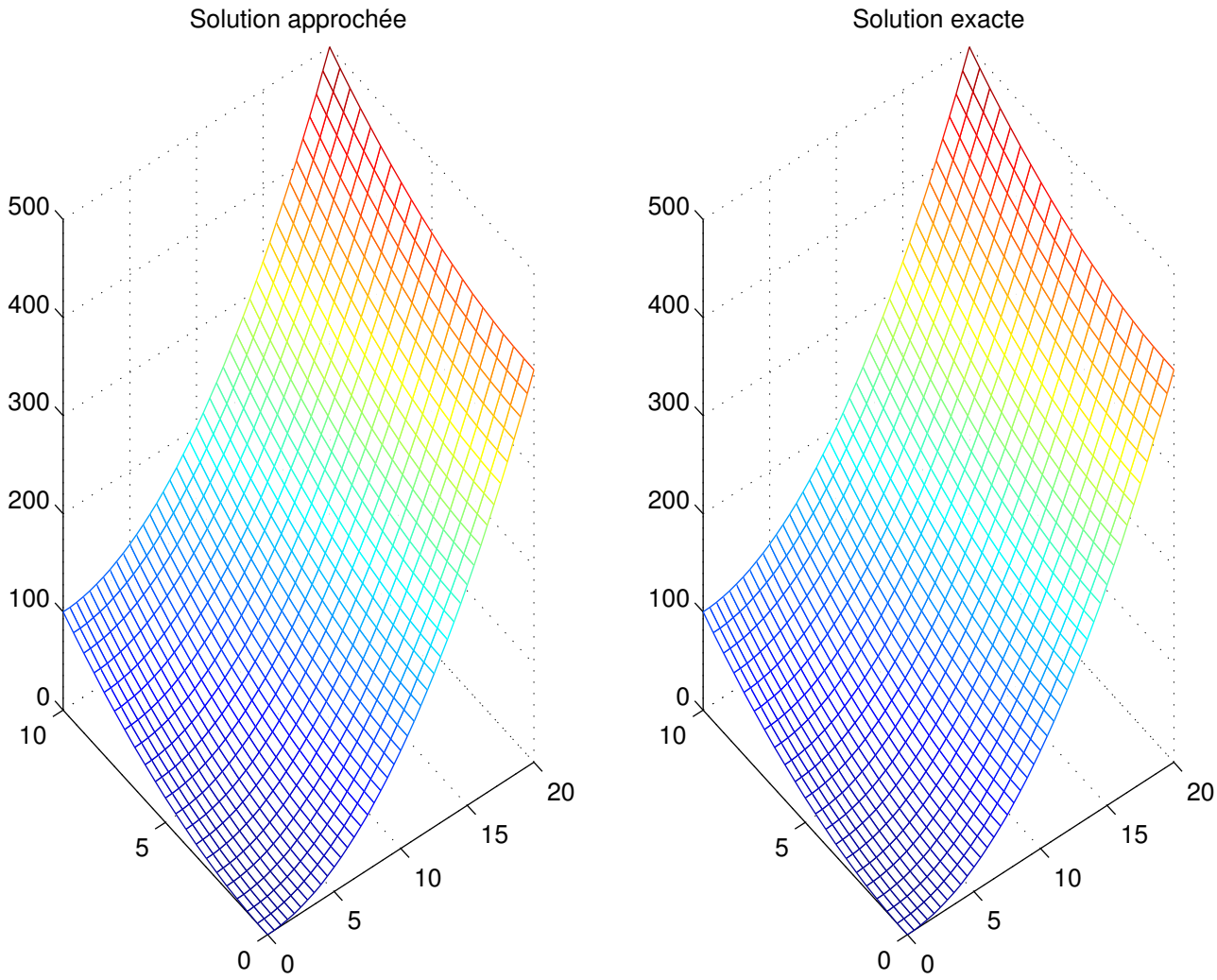


FIGURE 2.1 – Solution Exacte VS Solution Numérique

Chapitre 3

Résolution de l'équation de Laplace 2D avec des conditions aux limites mixtes

On souhaite résoudre l'équation aux dérivées partielles suivante :

$$\begin{cases} \Delta u(x,y) = f(x,y) \\ (x,y) \in]A,B[\times]C,D[\end{cases}$$

Avec les conditions aux limites suivantes :

$$\begin{cases} u_{\Gamma_1} = f_1(x) \\ u_{\Gamma_3} = f_2(x) \\ \dfrac{\partial u}{\partial \eta}_{/\Gamma_2} = g_1(x) \\ \dfrac{\partial u}{\partial \eta}_{/\Gamma_4} = g_2(x) \end{cases}$$

Avec $A = C = 0$, et $B = 20$, $D = 10$, $f(x,y) = 4$.
$f_1(x) = x^2$, $f_2(x) = x^2 + 100$, $g_1(x) = 2x$, $g_2(x) = -2x$

3.1 Approximation de l'EDP :

Soit :

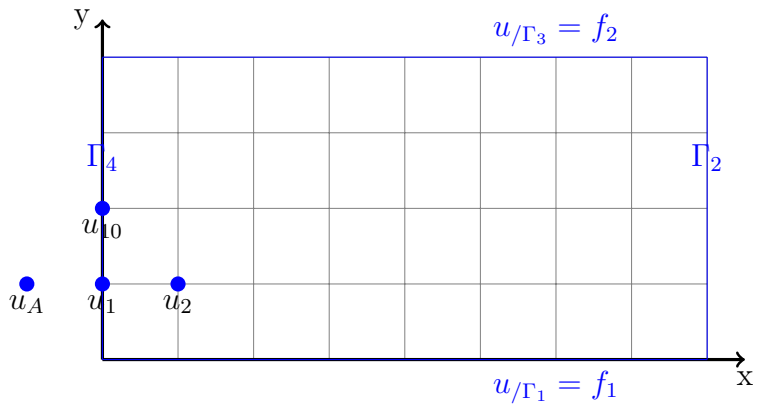
$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x,y), \forall (x,y) \in [0,20] \times [0,10]$$

Alors on a :

$$\begin{cases} \Delta u(x_i, y_j) = f(x_i, y_j) \Leftrightarrow \dfrac{1}{h^2}(u_{i+1,j} + u_{i,j+1} - 4u_{i,j} + u_{i-1,j} + u_{i,j-1}) = f(x_i, y_j) \\ i = 0, 1, ..., n_x \text{ et } j = 0, 1, ..., n_y \end{cases}$$

A chaque étape, le calcule de la valeur de $u_{i,j}$ au point (x_i, y_j) avec $1 \leq i \leq n_x - 1$ et $1 \leq j \leq n_y - 1$ se fait de manière identique au chapitre précédent.

Pour le calcul des valeurs de u sur les bord Γ_2 et Γ_4 on ajoute des nœuds fictifs comme l'indique la figure suivante :



Par exemple pour le premier nœud on a l'équation :

$$\frac{1}{h^2}(-4u_1 + u_2 + u_{10} + u_A) = f_{0,1}$$

$$\begin{aligned} \frac{\partial u}{\partial \eta}(A) &= -\frac{\partial u}{\partial x}(A) \\ &= -\frac{u_1 - u_A}{h} \\ &= -2x_0 \end{aligned}$$

donc

$$u_A = u_1 - 2x_0 h$$

d'où

$$-3u_1 + u_2 + u_{10} = h^2 f_{0,1} + 2x_0 h$$

On fait de même pour les autres nœud et nous obtenons un système linéaire à $(n_x + 1).(n_y - 1)$ équations et $(n_x + 1).(n_y - 1)$ inconnues de la forme $A_h.U_h = F_h$, que nous pouvons résoudre numériquement en inversant la matrice carré A_h :

$$U_h = A_h^{-1}.F_h$$

.

3.2 Code Matlab

Conditions aux limites

```
%Ce programme resoud une EDP 2eme ordre sur [A,B]x[C,D] en utilisant la MDF
clear
clc
%Input
A = 0;
B = 20;
C = 0;
D = 10;
%Pre-processeur
nx = 40;
ny = nx/2;
h = (B-A)/nx;
X = A:h:B;
Y = C:h:D;
g1 = @(x) x^2;
g2 = @(x) 100+x^2;
f1 = @(y) y^2;
f2 = @(y) 400+y^2;
%Schema numerique
Ah = zeros((nx+1)*(ny-1));
F = zeros(1,(nx+1)*(ny-1));
m = nx+1;
%1er cas
for i=2:nx
    for j=2:(ny-2)
        N=i+(j-1)*m;
        Ah(N,N)=-4; Ah(N,N-1)=1; Ah(N,N+1)=1;
        Ah(N,N+m)=1; Ah(N,N-m)=1; F(N)=4*(h^2);
    end
end
%2eme cas
i=1; j=1;
N=i+(j-1)*m;
Ah(N,N)=-3; Ah(N,N+1)=1;
Ah(N,N+m)=1; F(N)=4*(h^2)-g1(X(i))+2*X(i)*h;
%3eme cas
j=ny-1;
for i=2:nx
    N=i+(j-1)*m;
    Ah(N,N)=-4; Ah(N,N-1)=1; Ah(N,N+1)=1;
    Ah(N,N-m)=1; F(N)=4*(h^2)-g2(X(i));
end
%4eme cas
i=1;
for j=2:(ny-2)
    N=i+(j-1)*m;
    Ah(N,N)=-3; Ah(N,N+1)=1;
    Ah(N,N+m)=1; Ah(N,N-m)=1; F(N)=4*(h^2)+2*X(i)*h;
end
%5eme cas
i=1; j=ny-1;
N=i+(j-1)*m;
```

```
Ah(N,N)=-3; Ah(N,N+1)=1;
Ah(N,N-m)=1; F(N)=4*(h^2)-g2(X(i))+2*X(i)*h;
%6eme cas
j=1;
for i=2:nx
    N=i+(j-1)*m;
    Ah(N,N)=-4; Ah(N,N-1)=1; Ah(N,N+1)=1;
    Ah(N,N+m)=1; F(N)=4*(h^2)-g1(X(i));
end
%7eme cas
i=nx+1; j=1;
N=i+(j-1)*m;
Ah(N,N)=-3; Ah(N,N-1)=1;
Ah(N,N+m)=1; F(N)=4*(h^2)-g1(X(i))-2*X(i)*h;
%8eme cas
i=nx+1;
for j=2:(ny-2)
    N=i+(j-1)*m;
    Ah(N,N)=-3; Ah(N,N-1)=1;
    Ah(N,N+m)=1; Ah(N,N-m)=1; F(N)=4*(h^2)-2*X(i)*h;
end
%9eme cas
i=nx+1; j=ny-1;
N=i+(j-1)*m;
Ah(N,N)=-3; Ah(N,N-1)=1;
Ah(N,N-m)=1; F(N)=4*(h^2)-g2(X(i))-2*X(i)*h;
%Resolution
Ysol=Ah\(F');
Yh=transpose(reshape(Ysol,nx+1,ny-1));
%Post-processeur
Ynum=zeros(ny+1,nx+1);
Ynum(2:ny,1:nx+1)=Yh;
for j=1:length(X)
    Ynum(1,j)=g1(X(j));
end
for j=1:length(X)
    Ynum(ny+1,j)=g2(X(j));
end
%validation
%Solution exacte
[U,V]=meshgrid(X,Y);
Yex=U.^2+V.^2;
subplot(1,2,1), mesh(U,V,Ynum), title('Solution approchee')
subplot(1,2,2), mesh(U,V,Yex), title('Solution exacte')
```

3.3 Résultats graphiques

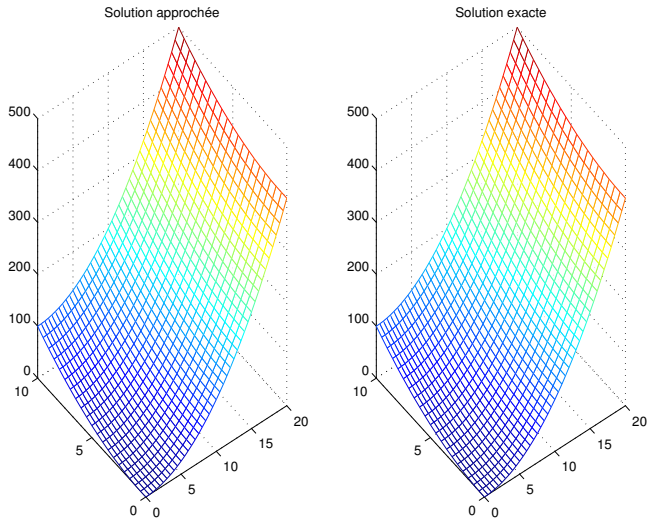


FIGURE 3.1 – Solutions obtenues

Chapitre 4

Résolution de l'équation de Laplace 3D par la méthode des Différences Finies

On souhaite résoudre l'EDP suivante :

$$\begin{cases} \Delta u = f(x,y,z) \\ (x,y,z) \in [0,1],[0,1],[0,1] \end{cases}$$

Avec les conditions aux limites suivantes :

$$\begin{cases} u(0,y,z) = y^2 + z^2 = f_1(y,z) \\ u(1,y,z) = 1 + y^2 + z^2 = f_2(y,z) \\ u(x,0,z) = x^2 + z^2 = g_1(x,z) \\ u(x,1,z) = 1 + x^2 + z^2 = g_2(x,z) \\ u(x,y,0) = x^2 + y^2 = h_1(x,y) \\ u(x,y,1) = 1 + x^2 + y^2 = h_2(x,y) \end{cases}$$

Généralement la résolution analytique de ce type d'EDP est difficile :
Numériquement, c'est toujours possible, en effet :

4.1 Discrétisation du domaine $]0,1[^3$

On considère

$$n = 4 \Rightarrow h = \frac{1}{n} = 0.25$$

Ce qui nous donne un système de $N = (n-1)^3 = 27$ équations de 27 inconnues.
Pour éclaircir la technique si on représente ci-dessous les plans de discrétisation de u

après avoir diviser le domaine de définition des variables selon le h obtenu. On a donc :

$$x(i) = x_i = 0 + i.h,\ i = 0, ..., 4$$
$$y(j) = y_j = 0 + j.h,\ j = 0, ..., 4$$
$$z(k) = z_k = 0 + k.h,\ k = 0, ..., 4$$

4.2 Schéma numérique(DF)

Pour i, j et k on a :

$$N = (k-1)(n-1)^2 + (j-1)(n-1) + i$$
$$A(N,N) = -6,\ A(N,N+1) = 1,\ A(N,N-1) = 1$$
$$A(N,N+m) = 1,\ A(N,N-m) = 1\ \ avecm = n-1$$
$$A(N,N+m^2) = 1,\ A(N,N-m^2) = 1\ \ avecm = n-1$$
$$F(N) = f(x_i, y_j, z_k)$$

4.3 Calcul numérique

4.3.1 Algorithme

Code matlab

```
%Ce programme resoud une EDP 3eme ordre sur [A,B]^3 en utilisant la MDF
%-------------------------------------------------------------------------
clear
clc
%-------------------------------------------------------------------------
%Input
A = 0;
B = 1;
%Pre-processeur
n = 20;
h = (B-A)/n;
x = A:h:B;
y = A:h:B;
z = A:h:B;
%Schema numerique
Ah = zeros((n-1)^3);
F = zeros(1,(n-1)^3);
%1ere cas
i=1; j=1; k=1;
N=i+(j-1)*(n-1)+(k-1)*((n-1)^2);
Ah(N,N)=-6;
Ah(N,N+1)=1;
```

```
Ah(N,N+(n-1))=1;
Ah(N,N+(n-1)^2) =1;
F(N)=6*(h^2)-f1(y(j+1),z(k+1))-g1(x(i+1),z(k+1))-r1(x(i+1),y(j+1));
%2eme cas
i=n-1; j=1; k=1;
N=i+(j-1)*(n-1)+(k-1)*((n-1)^2);
Ah(N,N)=-6;
Ah(N,N-1)=1;
Ah(N,N+(n-1))=1;
Ah(N,N+(n-1)^2) =1;
F(N)=6*(h^2)-f2(y(j+1),z(k+1))-g1(x(i+1),z(k+1))-r1(x(i+1),y(j+1));
%3eme cas
i=1; k=1; j=n-1;
N=i+(j-1)*(n-1)+(k-1)*((n-1)^2);
Ah(N,N)=-6;
Ah(N,N+1)=1;
Ah(N,N-(n-1))=1;
Ah(N,N+(n-1)^2)=1;
F(N)=6*(h^2)-f1(y(j+1),z(k+1))-g2(x(i+1),z(k+1))-r1(x(i+1),y(j+1));
%4eme cas
i=1; j=1; k=n-1;
N=i+(j-1)*(n-1)+(k-1)*((n-1)^2);
Ah(N,N)=-6;
Ah(N,N+1)=1;
Ah(N,N+(n-1))=1;
Ah(N,N-(n-1)^2)=1;
F(N)=6*(h^2)-f1(y(j+1),z(k+1))-g1(x(i +1),z(k+1))-r2(x(i+1),y(j+1));
%5eme cas
for i=2:n-2
 for j=2:n-2
  for k=2:n-2
      N=i+(j-1)*(n-1)+(k-1)*((n-1)^2);
      Ah(N,N)=-6;
      Ah(N,N+1)=1;
      Ah(N,N-1)=1;
      Ah(N,N+(n-1))=1;
      Ah(N,N-(n-1))=1;
      Ah(N,N+(n-1)^2)=1;
      Ah(N,N-(n-1)^2)=1;
      F(N)=6*(h^2);
  end
 end
end
%6eme cas
i=n-1; j=n-1; k=1;
N=i+(j-1)*(n-1)+(k-1)*((n-1)^2);
Ah(N,N)= -6;
Ah(N,N-1)=1;
Ah(N,N-(n-1))=1;
Ah(N,N+(n-1)^2)=1;
F(N)=6*(h^2)-f2(y(j+1),z(k+1))-g2(x(i+1),z(k+1))-r1(x(i+1),y(j+1));
%7eme cas
i=n-1; j=1; k=n-1;
N=i+(j-1)*(n-1)+(k-1)*((n-1)^2);
Ah(N,N)=-6;
```

```
Ah(N,N-1)=1;
Ah(N,N+(n-1))=1;
Ah(N,N-(n -1)^2)=1;
F(N)=6*(h^2)-f2(y(j+1),z(k+1))-g1(x(i+1),z(k+1))-r2(x(i+1),y(j+1));
%8eme cas
i=1; j=n-1; k=n-1;
N=i+(j-1)*(n-1)+(k-1)*((n-1)^2);
Ah(N,N)=-6;
Ah(N,N+1)=1;
Ah(N,N-(n-1))=1;
Ah(N,N-(n-1)^2)=1;
F(N)=6*(h^2)-f1(y(j+1),z(k+1))-g2(x(i+1),z(k+1))-r2(x(i+1),y(j+1));
%9eme cas
i=n-1; j=n-1; k=n-1;
N=i+(j-1)*(n-1)+(k-1)*((n-1)^2);
Ah(N,N)=-6;
Ah(N,N-1)=1;
Ah(N,N-(n-1))=1;
Ah(N,N-(n-1)^2)=1;
F(N)=6*(h^2)-f2(y(j+1),z(k+1))-g2(x(i+1),z(k+1))-r2(x(i+1),y(j+1));
%10eme cas
i=1;
for j=2:n-2
  for k=2:n-2
    N=i+(j-1)*(n-1)+(k-1)*((n-1)^2);
    Ah(N,N)=-6;
    Ah(N,N+1)=1;
    Ah(N,N+(n-1))=1;
    Ah(N,N-(n-1))=1;
    Ah(N,N+(n-1)^2)=1;
    Ah(N,N-(n-1)^2)=1;
    F(N)=6*(h^2)-f1(y(j+1),z(k+1));
  end
end
%11eme cas
i=n-1;
for j=2:n-2
  for k=2:n -2
    N=i+(j-1)*(n-1)+(k-1)*((n-1)^2);
    Ah(N,N)=-6;
    Ah(N,N-1)=1;
    Ah(N,N+(n-1))=1;
    Ah(N,N-(n-1))=1;
    Ah(N,N+(n-1).^2)=1;
    Ah(N,N-(n-1).^2)=1;
    F(N)=6*(h^2)-f2(y(j+1),z(k+1));
  end
end
%12eme cas
i=1; j=1;
for k=2:n-2
  N=i+(j-1)*(n-1)+(k-1)*((n-1)^2);
  Ah(N,N)=-6;
  Ah(N,N+1)=1;
  Ah(N,N+(n-1))=1;
```

```
    Ah(N,N+(n-1)^2)=1;
    Ah(N,N-(n-1)^2)=1;
    F(N)=6*(h^2)-f1(y(j+1),z(k+1))-g1(x(i+1),z(k+1));
  end
  %13eme cas
  i=n-1; j=1;
  for k=2:n-2
    N=i+(j-1)*(n-1)+(k-1)*((n-1)^2);
    Ah(N,N)=-6;
    Ah(N,N-1)=1;
    Ah(N,N+(n-1))=1;
    Ah(N,N+(n-1)^2)=1;
    Ah(N,N-(n-1)^2)=1;
    F(N)=6*(h^2)-f2(y(j+1),z(k+1))-g1(x(i+1),z(k+1));
  end
  %14eme cas
  j=1;
  for i=2:n-2
   for k=2:n-2
     N=i+(j-1)*(n-1)+(k-1)*((n-1)^2);
     Ah(N,N)=-6;
     Ah(N,N+1)=1;
     Ah(N,N-1)=1;
     Ah(N,N+(n-1))=1;
     Ah(N,N+(n-1)^2)=1;
     Ah(N,N-(n-1)^2) =1;
     F(N)=6*(h^2)-g1(x(i+1),z(k+1));
   end
  end
  %15eme cas
  j=n-1;
  for i=2:n-2
   for k=2:n-2
     N=i+(j-1)*(n-1)+(k-1)*((n-1)^2);
     Ah(N,N)=-6;
     Ah(N,N+1)=1;
     Ah(N,N-1)=1;
     Ah(N,N-(n-1))=1;
     Ah(N,N+(n-1)^2)=1;
     Ah(N,N-(n-1)^2)=1;
     F(N)=6*(h^2)-g2(x(i+1),z(k+1));
   end
  end
  %16eme cas
  i=1; j=n-1;
  for k=2:n-2
    N=i+(j-1)*(n-1)+(k-1)*((n-1)^2);
    Ah(N,N)=-6;
    Ah(N,N+1)=1;
    Ah(N,N-(n-1))=1;
    Ah(N,N+(n-1)^2)=1;
    Ah(N,N-(n-1)^2)=1;
    F(N)=6*(h^2)-f1(y(j+1),z(k+1))-g2(x(i+1),z(k+1));
  end
  %17eme cas
```

```
i=n-1; j=n-1;
for k=2:n-2
  N=i+(j-1)*(n-1)+(k-1)*((n-1)^2);
  Ah(N,N)=-6;
  Ah(N,N-1)=1;
  Ah(N,N-(n-1))=1;
  Ah(N,N+(n-1)^2)=1;
  Ah(N,N-(n-1)^2)=1;
  F(N)=6*(h^2)-f2(y(j+1),z(k+1))-g2(x(i+1),z(k+1));
end
%18eme cas
j=1; k=1;
for i=2:n-2
  N=i+(j-1)*(n-1)+(k-1)*((n-1)^2);
  Ah(N,N)=-6;
  Ah(N,N-1)=1;
  Ah(N,N+1)=1;
  Ah(N,N+(n-1))=1;
  Ah(N,N+(n-1)^2)=1;
  F(N)=6*(h^2)-g1(x(i+1),z(k+1))-r1(x(i+1),y(j+1));
end
%19eme cas
j=n-1; k=1;
for i=2:n-2
  N=i+(j-1)*(n-1)+(k-1)*((n-1)^2);
  Ah(N,N)=-6;
  Ah(N,N-1)=1;
  Ah(N,N+1)=1;
  Ah(N,N-(n-1))=1;
  Ah(N,N+(n-1)^2)=1;
  F(N)=6*(h^2)- g2(x(i+1),z(k+1))-r1(x(i+1),y(j+1));
end
%20eme cas
i=1; k=1;
for j=2:n-2
  N=i+(j-1)*(n-1)+(k-1)*((n-1)^2);
  Ah(N,N)=-6;
  Ah(N,N+1)=1;
  Ah(N,N-(n-1))=1;
  Ah(N,N+(n-1))=1;
  Ah(N,N+(n-1)^2)=1;
  F(N)=6*(h^2)-f1(y(j+1),z(k+1))-r1(x(i+1),y(j+1));
end
%21eme cas
i=n-1; k=1;
for j=2:n-2
  N=i+(j-1)*(n-1)+(k-1)*((n-1)^2);
  Ah(N,N)=-6;
  Ah(N,N-1)=1;
  Ah(N,N-(n-1))=1;
  Ah(N,N+(n-1))=1;
  Ah(N,N+(n-1)^2)=1;
  F(N)=6*(h^2)-f2(y(j+1),z(k+1))-r1(x(i+1),y(j+1));
end
%22eme cas
```

```
k=1;
for i=2:n-2
 for j=2:n-2
   N=i+(j-1)*(n-1)+(k-1)*((n-1)^2);
   Ah(N,N)=-6;
   Ah(N,N+1)=1;
   Ah(N,N-1)=1;
   Ah(N,N-(n-1))=1;
   Ah(N,N+(n-1))=1;
   Ah(N,N+(n-1)^2) =1;
   F(N)=6*(h^2)-r1(x(i+1),y(j+1));
 end
end
%23eme cas
k=n-1;
for i=2:n-2
 for j=2:n-2
   N=i+(j-1)*(n-1)+(k-1)*((n-1)^2);
   Ah(N,N)=-6;
   Ah(N,N+1)=1;
   Ah(N,N-1)=1;
   Ah(N,N-(n-1))=1;
   Ah(N,N+(n-1))=1;
   Ah(N,N-(n-1)^2)=1;
   F(N)=6*(h^2)-r2(x(i+1),y(j+1));
 end
end
%24eme cas
k=n-1; j=1;
for i=2:n-2
  N=i+(j-1)*(n-1)+(k-1)*((n-1)^2);
  Ah(N,N)=-6;
  Ah(N,N+1)=1;
  Ah(N,N-1)=1;
  Ah(N,N+(n-1))=1;
  Ah(N,N-(n-1)^2)=1;
  F(N)=6*(h^2)-r2(x(i+1),y(j+1))-g1(x(i+1),z(k+1));
end
%25eme cas
k=n-1; j=n-1;
for i=2:n-2
  N=i+(j-1)*(n-1)+(k-1)*((n-1)^2);
  Ah(N,N)=-6;
  Ah(N,N+1)=1;
  Ah(N,N-1) =1;
  Ah(N,N-(n-1))=1;
  Ah(N,N-(n-1)^2)=1;
  F(N)=6*(h^2)-r2(x(i+1),y(j+1))-g2(x(i+1),z(k+1));
end
%26eme cas
i=1;k=n-1;
for j=2:n-2
  N=i+(j-1)*(n-1)+(k-1)*((n-1)^2);
  Ah(N,N)=-6;
  Ah(N,N+1)=1;
```

```
298     Ah(N,N-(n-1))=1;
299     Ah(N,N+(n-1))=1;
300     Ah(N,N-(n-1)^2)=1;
301     F(N)=6*(h^2)-r2(x(i+1),y(j+1))-f1(y(j+1),z(k+1));
302   end
303   %27eme cas
304   i=n-1; k=n-1;
305   for j=2:n-2
306     N=i+(j-1)*(n-1)+(k-1)*((n-1)^2);
307     Ah(N,N)=-6;
308     Ah(N,N-1)=1;
309     Ah(N,N-(n-1))=1;
310     Ah(N,N+(n-1))=1;
311     Ah(N,N-(n-1)^2) =1;
312     F(N)=6*(h^2)-r2(x(i+1),y(j+1))-f2(y(j+1),z(k+1));
313   end
314   %Resolution
315   Y1 = Ah\(F');
316   Y2 = reshape(Y1,n-1,n-1,n-1);
317   Yh = zeros(n-1,n-1,n-1);
318   for k=1:(n-1)
319       Yh(:,:,k) = (Y2(:,:,k))';
320   end
321   clear Y1 Y2
322   %Post-processeur
323   Ynum = zeros(n+1,n+1,n+1);
324   for k=2:n
325       for i=2:(length(y)-1)
326           for j=2:(length(x)-1)
327               Ynum(i,j,k)=Yh(i-1,j-1,k-1);
328           end
329       end
330       for j=1:length(x)
331           Ynum(1,j,k)=g1(x(j),z(k));
332           Ynum(n+1,j,k)=g2(x(j),z(k));
333       end
334       for i=2:(length(y)-1)
335           Ynum(i,1,k)=f1(y(i),z(k));
336           Ynum(i,n+1,k)=f2(y(i),z(k));
337       end
338   end
339   for i=1:length(y)
340       for j=1:length(x)
341           Ynum(i,j,1)=r1(x(j),y(i));
342           Ynum(i,j,n+1)=r2(x(j),y(i));
343       end
344   end
345   %validation
346   %Solution exacte
347   Yex = zeros(n+1,n+1,n+1);
348   for k=1:(n+1)
349       for i=1:(n+1)
350           for j=1:(n+1)
351               Yex(i,j,k) = x(j)^2+y(i)^2+z(k)^2;
352           end
```

```
353         end
354     end
355     %Graphe de la solution exacte et de la solution approchee
356     plan = round(n/2);
357     %c=figure(1);
358     subplot(1,2,1), mesh(x,y,Ynum(:,:,plan)), title('Solution approchee z=0.5')
359     subplot(1,2,2), mesh(x,y,Yex(:,:,plan)), title('Solution exacte z=0.5')
360     %saveas(c,'DF3D.eps','epsc');
```

Condition aux limites pour x=0, la fonction f_1 :

Code matlab

```
1  function z=f1(y,z)
2  z=y.^2+z.^2;
```

Condition aux limites pour x=1, la fonction f_2 :

Code matlab

```
1  function z=f2(y,z)
2  z=1+y.^2+z.^2;
```

Condition aux limites pour y=0, la fonction g_1 :

Code matlab

```
1  function z=g1(x,z)
2  z=x.^2+ z.^2;
```

Condition aux limites pour y=1, la fonction g_2 :

Code matlab

```
1  function z=g2(x,z)
2  z=1+x.^2+ z.^2;
```

Condition aux limites pour z=0, la fonction r_1 :

Code matlab

```
1  function z=r1(x,y)
2  z=x.^2+y.^2;
```

Condition aux limites pour z=1, la fonction r_2 :

Code matlab

```
function z=r2(x,y)
z=1+x.^2+ y.^2;
```

4.3.2 Résultats graphiques

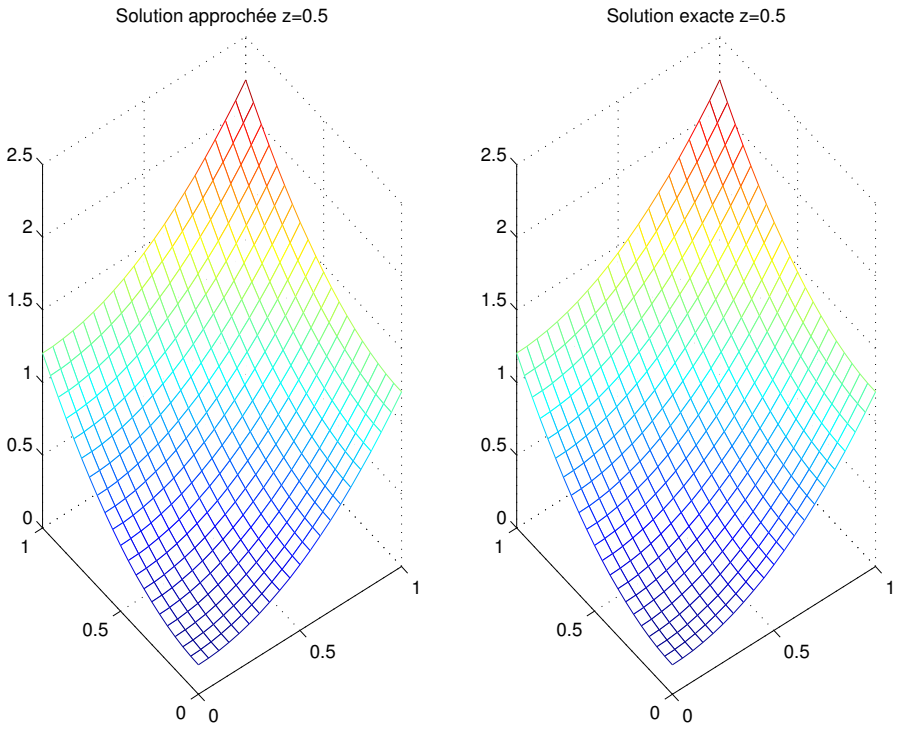


FIGURE 4.1 – Solutions obtenues

Chapitre 5

Résolution de l'équation de Laplace 2D par la méthode RBF

5.1 Généralités

Les fonctions de base radiale (RBF) sont une série de techniques d'interpolation exacte qui impliquent le passage de la surface par chaque valeur d'échantillon mesuré. Il existe cinq fonctions de base différentes :
-Spline de plaque fine
-Spline avec tension
-Spline entièrement régularisée
-Fonction multiquadratrique
-Fonction multiquadratrique inverse
Chaque fonction de base présente une forme différente et produit une surface d'interpolation différente. Les méthodes RBF représentent un cas particulier de splines.

Le concept des fonctions de base radiale revient à l'ajustement d'une membrane en caoutchouc à travers les valeurs d'échantillons mesurés tout en réduisant au maximum la courbure totale de la surface. La fonction de base sélectionnée vous permet de déterminer comment la membrane en caoutchouc va être ajustée entre les valeurs. Le diagramme suivant explique en théorie comment une surface RBF est ajustée en passant par une série de valeurs d'échantillons d'altitude.

En tant qu'interpolateurs exacts, les méthodes RBF diffèrent des interpolateurs polynomiaux globaux et locaux qui sont tous deux inexacts et n'exigent pas que la surface passe par les points mesurés.

On souhaite résoudre l'EDP suivante :

$$\begin{cases} \Delta u(x,y) = f(x,y) & (x,y) \in \Omega =]0,20[\times]0,10[\\ u_{/\partial\Omega} = g \end{cases}$$

Avec les conditions aux limites suivantes :

$$\begin{cases} u(0,y) = y^2 = f_1(y) \\ u(20,y) = 400 + y^2 = f_2(y) \\ u(x,0) = x^2 = g_1(x) \\ u(x,10) = x^2 + 100 = g_2(x) \end{cases}$$

5.2 Approximation de l'EDP :

Soient $X_1, X_2, \ldots, X_N$ des points intérieurs à Ω.
Et $X_{N+1}, X_{N+2}, \ldots, X_{N+n_f}$ des points sur $\partial\Omega$.

La solution du problème s'écrit de la manière suivante :

$$U(X) = \sum_{i=1}^{N+n_f} \alpha_i \varphi_i(X) \qquad X = (x,y)$$

Avec $\varphi_i(X)$ une fonction radiale de base qui s'écrit :

$$\begin{aligned} \varphi_i(X) &= \sqrt{||X - X_i||^2 + c^2} \\ &= \sqrt{(x - x_i)^2 + (y - y_i)^2 + c^2} \end{aligned}$$

Alors d'après l'équation aux dérivés partielle on obtient :

$$\text{pour } j = 1, \ldots, N : \qquad \Delta U(X_j) = \sum_{i=1}^{N+n_f} \alpha_i \Delta\varphi_i(X_j) = f(X_j)$$

$$\text{pour } j = N+1, \ldots, N+n_f : \qquad U(X_j) = \sum_{i=1}^{N+n_f} \alpha_i \varphi_i(X_j) = g(X_j)$$

Donc on obtient le système linéaire suivant :

$$\begin{cases} \alpha_1\Delta\varphi_1(X_1) + \alpha_2\Delta\varphi_2(X_1) + \ldots + \alpha_{N+n_f}\Delta\varphi_{N+n_f}(X_1) = f(X_1) \\ \quad \vdots \qquad\qquad \vdots \\ \quad \vdots \qquad\qquad \vdots \\ \alpha_1\Delta\varphi_1(X_N) + \alpha_2\Delta\varphi_2(X_N) + \ldots + \alpha_{N+n_f}\Delta\varphi_{N+n_f}(X_N) = f(X_N) \\ \alpha_1\varphi_1(X_{N+1}) + \alpha_2\varphi_2(X_{N+1}) + \ldots + \alpha_{N+n_f}\varphi_{N+n_f}(X_{N+1}) = g(X_{N+1}) \\ \quad \vdots \qquad\qquad \vdots \\ \quad \vdots \qquad\qquad \vdots \\ \alpha_1\varphi_1(X_{N+n_f}) + \alpha_2\varphi_2(X_{N+n_f}) + \ldots + \alpha_{N+n_f}\varphi_{N+n_f}(X_{N+n_f}) = g(X_{N+n_f}) \end{cases}$$

Ce système s'écrit sous la forme matricielle :

$$A\alpha = F$$

que nous pouvons résoudre numériquement en inversant la matrice carré A :

$$\alpha = A^{-1}F$$

.

Code matlab de φ

```
function z = phi(x,y,xi,yi,c)
z = sqrt((x-xi)^2+(y-yi)^2+c^2);
end
```

Code matlab du laplacien de φ

```
function z = laplacien(x,y,xi,yi,c)
z = 2/phi(x,y,xi,yi,c)- ((x-xi)^2+(y-yi)^2)/(phi(x,y,xi,yi,c)^3);
end
```

5.3 Code Matlab

```
%Ce programme resoud une EDP 2eme ordre sur [A,B]x[C,D] en utilisant la RBF
%-------------------------------------------------------------------
clear
clc
%-------------------------------------------------------------------
%Input
A = 0;
B = 20;
C = 0;
D = 10;
c=100;
%Pre-processeur
nx = 20;
ny = 10;
h = (B-A)/nx;
x = A:h:B;
y = C:h:D;
%Schema numerique
Ah = zeros((nx+1)*(ny+1));
F = zeros(1,(nx+1)*(ny+1));
S = zeros((nx+1)*(ny+1),2);
k = 1;
%Remplissage de S par les sommets internes
for i=1:(nx-1)
    for j=1:(ny-1)
        S(k,1) = x(i+1);
        S(k,2) = y(j+1);
```

```
28          k = k+1;
29      end
30  end
31  %Remplissage de S par les somments du bord Gamma1
32  for i=1:(nx+1)
33      S(k,1) = x(i);
34      S(k,2) = y(1);
35      k = k+1;
36  end
37  %Remplissage de S par les somments du bord Gamma2
38  for j=2:ny
39      S(k,1) = x(nx+1);
40      S(k,2) = y(j);
41      k = k+1;
42  end
43  %Remplissage de S par les somments du bord Gamma3
44  for i=1:(nx+1)
45      S(k,1) = x(i);
46      S(k,2) = y(ny+1);
47      k = k+1;
48  end
49  %Remplissage de S par les somments du bord Gamma4
50  for j=2:ny
51      S(k,1) = x(1);
52      S(k,2) = y(j);
53      k = k+1;
54  end
55
56  for i=1:(nx-1)*(ny-1)
57      for j=1:(nx+1)*(ny+1)
58          Ah(i,j) = laplacien(S(i,1),S(i,2),S(j,1),S(j,2),c);
59      end
60  end
61
62  for i=((nx-1)*(ny-1)+1):(nx+1)*(ny+1)
63      for j=1:(nx+1)*(ny+1)
64          Ah(i,j) = phi(S(i,1),S(i,2),S(j,1),S(j,2),c);
65      end
66  end
67
68  for i=1:(nx-1)*(ny-1)
69      F(i) = 4;
70  end
71
72  for i=((nx-1)*(ny-1)+1):(nx*ny-ny+2)
73      F(i) = S(i,1)^2;
74  end
75
76  for i=(nx*ny-ny+3):(nx*ny+1)
77      F(i) = 400+S(i,2)^2;
78  end
79
80  for i=(nx*ny+2):(nx*ny+nx+2)
81      F(i) = 100+S(i,1)^2;
82  end
```

```
83
84  for i=(nx*ny+nx+3):(nx+1)*(ny+1)
85      F(i) = S(i,2)^2;
86  end
87  alpha = Ah\(F');
88  %Post-processeur
89  Sol = zeros(ny+1,nx+1);
90  for i=1:(ny+1)
91      for j=1:(nx+1)
92          for k=1:(nx+1)*(ny+1)
93              Sol(i,j) = Sol(i,j)+alpha(k)*phi(x(j),y(i),S(k,1),S(k,2),c);
94          end
95      end
96  end
97  %Validation
98  %Solution exacte
99  [X,Y]=meshgrid(x,y);
100 Yex=X.^2+Y.^2;
101 %Representation graphique
102 subplot(1,2,1), mesh(x,y,Sol), title('Solution approchee')
103 subplot(1,2,2), mesh(x,y,Yex), title('Solution exacte')
```

5.4 Résultats graphiques

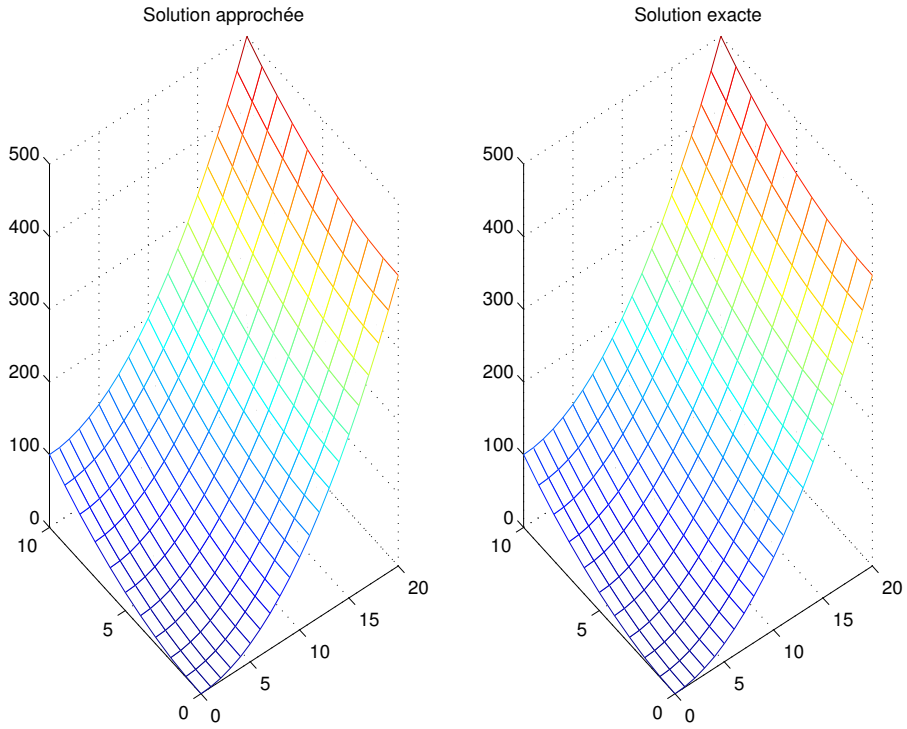


FIGURE 5.1 – Solutions obtenues

Chapitre 6

Conduction de chaleur par maillage structuré

6.1 Position du problème

On considère le problème modèle suivant (conduction de la chaleur) :

$$-div(\lambda_i \nabla u(x)) = f(x), \;\; x \in \Omega_i, \;\; i = 1, 2 \tag{6.1}$$

où $\lambda_1 > 0$, $\lambda_2 > 0$ sont les conductivités thermiques dans les domaine Ω_1 et Ω_2, avec $\Omega_1 =]0,1[\times]0,1[$ et $\Omega_2 =]0,1[\times]0,2[$. On appelle $\Gamma_1 =]0,1[\times\{0\}$, $\Gamma_2 = \{1\}\times]0,2[$, $\Gamma_3 =]0,1[\times\{2\}$, $\Gamma_4 = \{0\}\times]0,2[$ les frontières extérieures de Ω, et on note $I =]0,1[\times\{1\}$ l'interface entre Ω_1 et Ω_2 (voir figure). Dans la suite on notera λ la conductivité thermique sur Ω, $\lambda/\Omega_i = \lambda_i, i = 1, 2$

6.2 Maillage

On se donne un maillage admissible $\mathbb{M}$ de Ω, $\Omega = \bigcup_{K\in\mathbb{M}} K$ qu'on suppose uniforme. On suppose qu'on a $n \times 2p$ mailles. On note $h_x = \frac{1}{n}$ (resp. $h_y = \frac{1}{p}$). Comme le maillage est cartésien, il est commode de numéroter les mailles dans l'ordre lexicographique, c'est à dire que le k^{ieme} maille a comme centre le point $x_{i,j} = (ih_x, ih_y)$ avec $k = n(j-1)+i$. On peut donc déterminer le numéro de la maille de l'inconnue associée k à partir de la numérotation cartésienne (i, j).

On se donne ensuite des inconnues discrètes $(u_K)_{K\in M}$ associée aux mailles et $(u_\sigma)_{\sigma\in\epsilon}$ associées aux arêtes. ϵ est l'ensemble des arêtes.

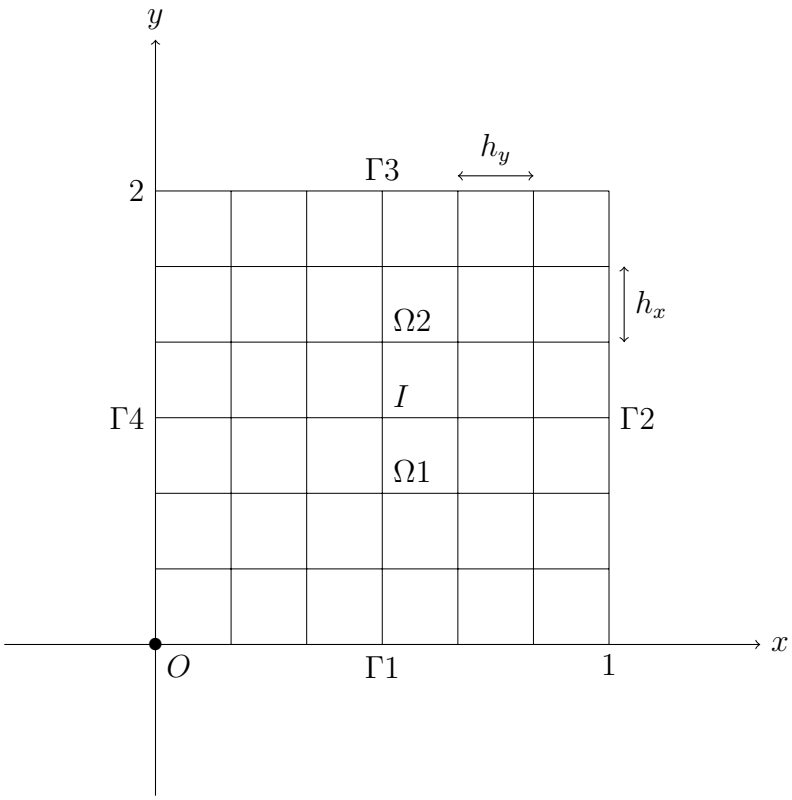


FIGURE 6.1 – Maillage du domaine $[0,1] \times [0,2]$

6.3 Construction du schéma

Pour construire le schéma numérique, on intègre l'équation (1.1) sur un élément de volume K :

$$-div(\lambda_i \nabla u(x)) = f(x)$$

$$\iff -\int_K div(\lambda_i \nabla u(x)) = \int_K f(x)\,dx$$

$$\iff -\int_K div(\lambda_i \nabla u(x)) = m(K)\frac{1}{m(K)}\int_K f(x)\,dx$$

Rappel 6.3.1 (Important)**.** *[Formule de Stokes] Soit $\Omega \subset \mathbb{R}^2$ ou $\mathbb{R}^3$:*

$$\int_\Omega div(\omega) = \int_{\partial\Omega} \omega.\vec{n}\,d\sigma$$

On a :

$$\partial K = \bigcup_{\sigma \in \epsilon_k}$$

Et :

$$\sum_{\sigma \in \epsilon_K} \int_\sigma (\lambda_i \nabla u(x))dx = \int_{\bigcup\limits_{\sigma \in \epsilon_k}} n(x)ds$$

On a :

$$\nabla u(x)n(x) \approx \frac{u_\sigma - u_k}{d_{K,\sigma}}$$

$$F_{k\ \sigma} = \int_\sigma (\lambda_i \nabla u(x))dx$$

$$\iff F_{k\ \sigma} \approx \int_\sigma \lambda_i \frac{u_\sigma - u_k}{d_{K,\sigma}} ds$$

$$\iff F_{k\ \sigma} \approx \lambda_i \frac{u_\sigma - u_k}{d_{K,\sigma}} \int_\sigma ds$$

$$\iff F_{k\ \sigma} \approx \lambda_i \frac{u_\sigma - u_k}{d_{K,\sigma}} m(\sigma)$$

$F_{K,\sigma}$ représente le flux numérique à travers l'arête σ. Finalement, on obtient le schéma numérique suivante :

$$\sum_{\sigma \in \epsilon_K} F_{K,\sigma} = m(K)\, f_K$$

6.4 Calcul des flux numériques

On considère l'élément de volume K délimité par les arêtes $\partial K = \sigma_1 \cup \sigma_2 \cup \sigma_3 \cup \sigma_4$. Nous allons exprimer dans cette paragraphe le flux numérique relatif à une arête σ quelconque. Le flux numérique total est la somme des flux partiels relatifs à chaque arête.

6.4.1 Flux interne : σ est une arête interne

Soit $\sigma = K/L$ où $K, L \in \mathbb{M}$. On a vu au paragraphe précédente que le flux numérique $F_{K,\sigma}$ s'écrit :

$$F_{K,\sigma} = -\lambda_i \frac{u_\sigma - u_K}{d_{K,\sigma}} m(\sigma) \tag{6.2}$$

En utilisant la conservation du flux ($F_{K,\sigma} = -F_{L,\sigma}$), on a :

$$-\lambda_i \frac{u_\sigma - u_K}{d_{K,\sigma}} m(\sigma) = \lambda_i \frac{u_\sigma - u_L}{d_{L,\sigma}} m(\sigma)$$

Alors :

$$u_\sigma \left(\frac{1}{d_{K,\sigma}} + \frac{1}{d_{L,\sigma}}\right) = \frac{u_k}{d_{K,\sigma}} + \frac{u_L}{d_{L,\sigma}}$$

On a donc :

$$u_\sigma = \frac{d_{K,\sigma} d_{L,\sigma}}{d_{K,L}} \left(\frac{u_K}{d_{K,\sigma}} + \frac{u_L}{d_{L,\sigma}}\right)$$

On remplaçons u_σ dans (1.1), on obtient :

$$F_{K,\sigma} = -\lambda_i \left(\frac{d_{L,\sigma}}{d_{K,L}} \left(\frac{u_K}{d_{K,\sigma}} + \frac{u_L}{d_{L,\sigma}}\right) - \frac{u_K}{d_{K,\sigma}}\right) m(\sigma)$$

Finalement :

$$F_{K,\sigma} = -\lambda_i \frac{m(\sigma)}{d_{K,L}} (u_K - u_L).$$

6.4.2 Flux au bord de Γ_2

Sur Γ_2, la condition au limite se traduit par l'équation suivante :

$$-\lambda_i \nabla u \, n = 0 \tag{6.3}$$

Par intégration de la condition (1.3), on a :

$$F_{K,\sigma} = 0.$$

6.4.3 Flux au bord de Γ_4

Sur Γ_4, la condition au limite se traduit par l'équation suivante :

$$u(x) = g(x) \tag{6.4}$$

Selon (1.4) on a : $u_\sigma = g_\sigma$ qu'on va remplacer dans l'expression du flux numérique :

$$F_{K,\sigma} = -\lambda_i \frac{m(\sigma)}{d_{K,\sigma}} (g_\sigma - u_K)$$

où :

$$g_\sigma = \frac{1}{m(\sigma)} \int_\sigma g(y)dy$$

6.4.4 Flux au bord de $\Gamma_1 \cup \Gamma_3$

Sur $\Gamma_1 \cup \Gamma_3$, on a l'équation suivante :

$$-\lambda_i \nabla u \, n = \alpha \, (u_\sigma - u_{ext}) \tag{6.5}$$

Si on intègre l'équation (1.5) sur σ, on obtient :

$$F_{K,\sigma} = \alpha \, m(\sigma) \, (u_\sigma - u_{ext}) \tag{6.6}$$

Et par identification de (1.1) et (1.6) , on a :

$$-\lambda_i \frac{u_\sigma - u_K}{d_{K,\sigma}} = \alpha \, (u_\sigma - u_{ext})$$

Alors :

$$u_\sigma = \frac{\alpha \, d_{K,\sigma} \, u_{ext} + \lambda_i \, u_K}{\alpha \, d_{K,\sigma} + \lambda_i}$$

Finalement :

$$F_{K,\sigma} = \alpha \, m(\sigma) \, (\frac{\alpha \, d_{K,\sigma} \, u_{ext} + \lambda_i \, u_K}{\alpha \, d_{K,\sigma} + \lambda_i} - u_{ext})$$

$$F_{K,\sigma} = \frac{\alpha \, m(\sigma) \, \lambda_i}{\alpha \, d_{K,\sigma} + \lambda_i} \, (u_K - u_{ext})$$

6.4.5 Flux sur l'interface I

$K \in \Omega_1$ et $L \in \Omega_2$

Sur l'interface I, on a un saut de flux de chaleur. Cette condition de saut se traduit par l'équation suivante :

$$-\lambda_1 \nabla u_1(x) \, n_1(x) - \lambda_2 \nabla u_2(x) \, n_2(x) = \Theta(x) \qquad \forall x \in I \tag{6.7}$$

On intègre sur σ, on obtient :

$$\int_\sigma \nabla u_1(x)\, n_1(x) dx - \int_\sigma \nabla u_2(x)\, n_2(x) dx = \int_\sigma \Theta(x) dx$$

Ce qui donne :

$$F_{K,\sigma} + F_{L,\sigma} = m(\sigma)\, \Theta_\sigma$$

Avec : $\Theta_\sigma = \frac{1}{m(\sigma)} \int_\sigma \Theta(x)\, dx$

En remplaçant le flux numérique par son expression :

$$-\lambda_1 \frac{u_\sigma - u_K}{d_{K,\sigma}} m(\sigma) - \lambda_2 \frac{u_\sigma - u_L}{d_{L,\sigma}} m(\sigma) = m(\sigma)\, \Theta_\sigma$$

On en déduit l'expression de u_σ :

$$u_\sigma = \frac{\lambda_1\, d_{L,\sigma}\, u_K + \lambda_2\, d_{K,\sigma}\, u_L - d_{K,\sigma}\, d_{L,\sigma}\, \Theta_\sigma}{\lambda_1\, d_{L,\sigma} + \lambda_2\, d_{K,\sigma}}$$

Qu'on remplace dans le flux numérique :

$$F_{K,\sigma} = -\lambda_1\, m(\sigma)\, \frac{\frac{\lambda_1\, d_{L,\sigma}\, u_K + \lambda_2\, d_{K,\sigma}\, u_L - d_{K,\sigma}\, d_{L,\sigma}\, \Theta_\sigma}{\lambda_1\, d_{L,\sigma} + \lambda_2\, d_{K,\sigma}} - u_K}{d_{K,\sigma}}$$

En simplifiant, on obtient :

$$F_{K,\sigma} = \frac{\lambda_1\, m(\sigma)}{\lambda_1\, d_{L,\sigma} + \lambda_2\, d_{K,\sigma}} \left(\lambda_2\, (u_K - u_L) + d_{L,\sigma}\, \Theta_\sigma\right)$$

$K \in \Omega_2$ et $L \in \Omega_1$

Par le même raisonnement :

$$F_{K,\sigma} = \frac{\lambda_2\, m(\sigma)}{\lambda_1\, d_{K,\sigma} + \lambda_2\, d_{L,\sigma}} \left(\lambda_1\, (u_K - u_L) + d_{L,\sigma}\, \Theta_\sigma\right)$$

6.5 Calcul numérique

6.5.1 Principe

On veut écrire le système linéaire $AU = b$ que satisfait les inconnues $U = (u_K)_{K \in \mathbb{M}}$. On remarque que $\forall K \in \mathbb{M}$,$m(K) = h_x\, h_y$, pour toute arête intérieure verticale σ : $d_\sigma = h_x$, $m(\sigma) = h_y$ et pour tout arête horizontal : $d_\sigma = h_y$, $m(\sigma) = h_x$. Pour chaque numéro de maille, nous allons maintenant construire l'équation correspondante.

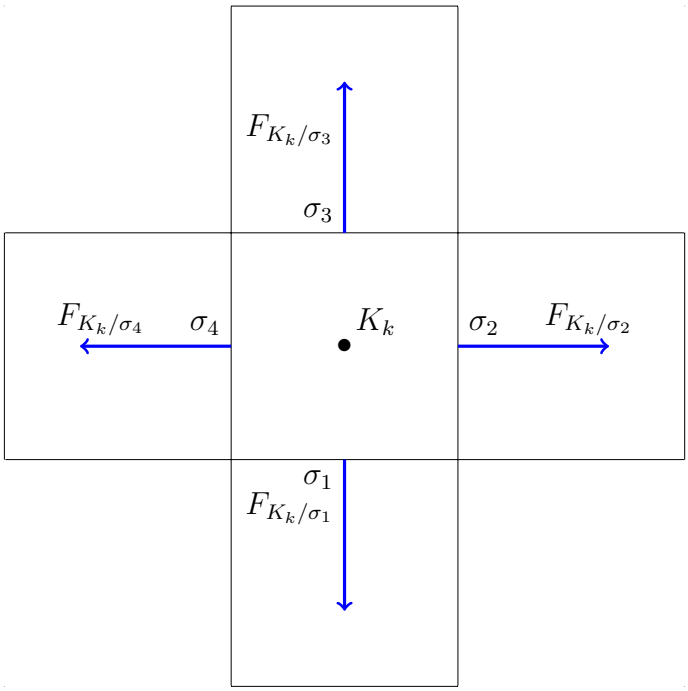


FIGURE 6.2 – Flux sur les arrêtes d'une maille

Exemple 6.5.1. *Exemple[Mailles internes]*

$$i = 2, ..., n-1$$

$$j = 2, ..., n-1$$

$$k = i + (j-1)n$$

$$F_{K_k\ \sigma 1} + F_{K_k\ \sigma 2} + F_{K_k\ \sigma 3} + F_{K_k\ \sigma 4} = m(K) f_K$$

$$\iff F_{K_k\ \sigma 1} = \lambda_i \frac{h_x}{h_y}(U_k - U_{k-n})$$

$$F_{K_k\ \sigma 2} = \lambda_i \frac{h_y}{h_x}(U_k - U_{k+1})$$

$$F_{K_k\ \sigma 1} = \lambda_i \frac{h_x}{h_y}(U_k - U_{k+n})$$

$$F_{K_k\ \sigma 1} = \lambda_i \frac{h_y}{h_x}(U_k - U_{k-1})$$

Cela implique :

$$A_k(k,k) = 2\lambda_i(\frac{h_x}{h_y} + \frac{h_y}{h_x});$$

$$A_k(k,k-1) = -\lambda_i\frac{h_y}{h_x};$$

$$A_k(k,k+1) = -\lambda_i\frac{h_y}{h_x};$$

$$A_k(k,k-n) = -\lambda_i\frac{h_x}{h_y};$$

$$A_k(k,k+n) = -\lambda_i\frac{h_x}{h_y};$$

$$F_h(k) = h_x h_y f_k$$

6.5.2 Code Matlab

```
%Ce programme resoud une EDP 2eme ordre sur [A,B]x[C,D] en utilisant la MVF

clear
clc

%Input
A = 0;
B = 1;
C = 0;
D = 2;
%Pre-processeur
n = 20;
p = 20;
hx = (B-A)/n;
hy = (D-C)/(2*p);
x = A:hx:B;
y = C:hy:D;
lamda1 = 2;
lamda2 = 1;
alpha = 1;
f1 = @(y) -lamda1*(y+2);
f2 = @(y) -lamda2*(y+2);
g = @(y) y.^2;
uext1 = @(x) -(x.^2)+2*x;
uext2 = @(x) 3*((x.^2)/2-x)+8;
teta = @(x) -((x.^2)/2-x+2);
%Schema numerique
Ah = zeros(2*n*p);
F = zeros(1,2*n*p);
%Aretes internes
for i=2:n-1
    for j=2:p-1
```

```
33              k=i+(j-1)*(n);
34      Ah(k,k)=lamda1*((2*hx)/hy+(2*hy)/hx);
35      Ah(k,k-1)=-lamda1*(hy/hx);
36      Ah(k,k+1)=-lamda1*(hy/hx);
37      Ah(k,k-n)=-lamda1*(hx/hy);
38      Ah(k,k+n)=-lamda1*(hx/hy);
39      F(k)=hx*integral(f1,y(j),y(j+1));
40        end
41  end
42
43  for  i=2:n-1
44      for  j=p+2:2*p-1
45        k=i+(j-1)*(n);
46      Ah(k,k)=lamda2*((2*hx)/hy+(2*hy)/hx);
47      Ah(k,k-1)=-lamda2*(hy/hx);
48      Ah(k,k+1)=-lamda2*(hy/hx);
49      Ah(k,k-n)=-lamda2*(hx/hy);
50      Ah(k,k+n)=-lamda2*(hx/hy);
51      F(k)=hx*integral(f2,y(j),y(j+1));
52        end
53  end
54  %Cas Gamma2
55  i=n;
56  for  j=2:p-1
57        k=i+(j-1)*(n);
58        Ah(k,k)=lamda1*((2*hx)/hy+(hy)/hx);
59        Ah(k,k-1)=-lamda1*(hy/hx);
60        Ah(k,k-n)=-lamda1*(hx/hy);
61        Ah(k,k+n)=-lamda1*(hx/hy);
62        F(k)=hx*integral(f1,y(j),y(j+1));
63  end
64
65  for  j=p+2:2*p-1
66      k=i+(j-1)*(n);
67      Ah(k,k)=lamda2*((2*hx)/hy+(hy)/hx);
68      Ah(k,k-1)=-lamda2*(hy/hx);
69      Ah(k,k-n)=-lamda2*(hx/hy);
70      Ah(k,k+n)=-lamda2*(hx/hy);
71      F(k)=hx*integral(f2,y(j),y(j+1));
72  end
73  %Cas Gamma 4
74  i=1;
75  for  j=2:p-1
76      k=i+(j-1)*(n);
77      Ah(k,k)=lamda1*((2*hx)/hy+(3*hy)/hx);
78      Ah(k,k+1)=-lamda1*(hy/hx);
79      Ah(k,k-n)=-lamda1*(hx/hy);
80      Ah(k,k+n)=-lamda1*(hx/hy);
81      F(k)=hx*integral(f1,y(j),y(j+1))+2*lamda1*(((1/hy)*integral(g,y(j),y(j+1)))*hy/hx);
82  end
83
84  for  j=p+2:2*p-1
85        k=i+(j-1)*(n);
86        Ah(k,k)=lamda2*((2*hx)/hy+(3*hy)/hx);
87        Ah(k,k+1)=-lamda2*(hy/hx);
```

```
88        Ah(k,k-n)=-lamda2*(hx/hy);
89        Ah(k,k+n)=-lamda2*(hx/hy);
90        F(k)=hx*integral(f2,y(j),y(j+1))+2*lamda2*(((1/hy)*integral(g,y(j),y(j+1)))*hy/hx);
91    end
92    %Cas Gamma1 union Gamma3
93    j=1;
94    for  i=2:n-1
95        k=i+(j-1)*(n);
96        Ah(k,k)=lamda1*((hx)/hy+(2*hy)/hx)+
97    ((alpha*lamda1*hx)/(lamda1+alpha*(hy/2)));
98        Ah(k,k-1)=-lamda1*(hy/hx);
99        Ah(k,k+1)=-lamda1*(hy/hx);
100       Ah(k,k+n)=-lamda1*(hx/hy);
101       F(k)=hx*integral(f1,y(j),y(j+1))+
102   ((alpha*lamda1*hx)/(lamda1+alpha*(hy/2)))
103   *uext1(x(i));
104   end
105   j=2*p;
106   for  i=2:n-1
107       k=i+(j-1)*(n);
108       Ah(k,k)=lamda2*((hx)/hy+(2*hy)/hx)+
109   ((alpha*lamda2*hx)/(lamda2+alpha*(hy/2)));
110       Ah(k,k-1)=-lamda2*(hy/hx);
111       Ah(k,k+1)=-lamda2*(hy/hx);
112       Ah(k,k-n)=-lamda2*(hx/hy);
113       F(k)=hx*integral(f2,y(j),y(j+1))+
114   ((alpha*lamda2*hx)/(lamda2+alpha*(hy/2)))
115   *uext2(x(i));
116   end
117   %Mailles des coins exterieurs
118   %Coin Sud-Ouest
119   i=1;
120   j=1;
121   k=i+(j-1)*(n);
122   Ah(k,k)=lamda1*((hx)/hy+(3*hy)/hx)+
123   ((alpha*lamda1*hx)/(lamda1+alpha*(hy/2)));
124   Ah(k,k+1)=-lamda1*(hy/hx);
125   Ah(k,k+n)=-lamda1*(hx/hy);
126   F(k)=hx*integral(f1,y(j),y(j+1))+
127   ((alpha*lamda1*hx)/(lamda1+alpha*(hy/2)))
128   *uext1(x(i))+2*lamda1*(((1/hy)*
129   integral(g,y(j),y(j+1)))*hy/hx);
130   %Coin Sud Est
131   i=n;
132   j=1;
133   k=i+(j-1)*(n);
134   Ah(k,k)=lamda1*((hx)/hy+(hy)/hx)+
135   ((alpha*lamda1*hx)/(lamda1+alpha*(hy/2)));
136   Ah(k,k-1)=-lamda1*(hy/hx);
137   Ah(k,k+n)=-lamda1*(hx/hy);
138   F(k)=hx*integral(f1,y(j),y(j+1))+
139   ((alpha*lamda1*hx)/(lamda1+alpha*(hy/2)))
140   *uext1(x(i));
141   %Coin Nord Ouest
142   i=1;
```

```
j=2*p;
k=i+(j-1)*(n);
Ah(k,k)=lamda2*((hx)/hy+(3*hy)/hx)+
((alpha*lamda2*hx)/(lamda2+alpha*(hy/2)));
Ah(k,k+1)=-lamda2*(hy/hx);
Ah(k,k-n)=-lamda2*(hx/hy);
F(k)=hx*integral(f2,y(j),y(j+1))+
((alpha*lamda2*hx)/(lamda2+alpha*(hy/2)))
*uext2(x(i))+2*lamda2*(((1/hy)
*integral(g,y(j),y(j+1)))*hy/hx);
%Coin Nord Est
i=n;
j=2*p;
k=i+(j-1)*(n);
Ah(k,k)=lamda2*((hx)/hy+(hy)/hx)+
((alpha*lamda2*hx)/(lamda2+alpha*(hy/2)));
Ah(k,k-1)=-lamda2*(hy/hx);
Ah(k,k-n)=-lamda2*(hx/hy);
F(k)=hx*integral(f2,y(j),y(j+1))+
((alpha*lamda2*hx)/(lamda2+alpha*(hy/2)))
*uext2(x(i));
%Interface I (Coins exclus)
j=p;
for i=2:n-1
    k=i+(j-1)*(n);
    Ah(k,k)=lamda1*((hx)/hy+(2*hy)/hx)+
(2*(lamda1*lamda2*hx)/(hy*(lamda1+lamda2)));
    Ah(k,k-1)=-lamda1*(hy/hx);
    Ah(k,k+1)=-lamda1*(hy/hx);
    Ah(k,k-n)=-lamda1*(hx/hy);
    Ah(k,k+n)=-(2*(lamda1*lamda2*hx)/
(hy*(lamda1+lamda2)));
    F(k)=hx*integral(f1,y(j),y(j+1))-\\
((lamda1*hx)/((lamda1+lamda2)))*\\
(1/hx)*integral(teta,x(i),x(i+1));\\
end

j=p+1;
for i=2:n-1
    k=i+(j-1)*(n);
    Ah(k,k)=lamda2*((hx)/hy+(2*hy)/hx)+
(2*(lamda1*lamda2*hx)/(hy*(lamda1+lamda2)));
    Ah(k,k-1)=-lamda2*(hy/hx);
    Ah(k,k+1)=-lamda2*(hy/hx);
    Ah(k,k-n)=-(2*(lamda1*lamda2*hx)/
(hy*(lamda1+lamda2)));
    Ah(k,k+n)=-lamda2*(hx/hy);
    F(k)=hx*integral(f2,y(j),y(j+1))-
((lamda2*hx)/((lamda1+lamda2)))*(1/hx)*
integral(teta,x(i),x(i+1));
end
%Interface I les coins
j=p;
i=1;
k=i+(j-1)*(n);
```

```
Ah(k,k)=lamda1*((hx)/hy+(3*hy)/hx)+(2*
(lamda1*lamda2*hx)/(hy*(lamda1+lamda2)));
Ah(k,k+1)=-lamda1*(hy/hx);
Ah(k,k-n)=-lamda1*(hx/hy);
Ah(k,k+n)=-(2*(lamda1*lamda2*hx)/
(hy*(lamda1+lamda2)));
F(k)=hx*integral(f1,y(j),y(j+1))-
((lamda1*hx)/((lamda1+lamda2)))*
(1/hx)*integral(teta,x(i),x(i+1))+2*lamda1*(1/hy)*integral(g,y(j),y(j+1))*(hy/hx);

j=p+1;
i=1;
k=i+(j-1)*(n);
Ah(k,k)=lamda2*((hx)/hy+(3*hy)/hx)+
(2*(lamda1*lamda2*hx)/(hy*(lamda1+lamda2)));
Ah(k,k+1)=-lamda2*(hy/hx);
Ah(k,k-n)=-(2*(lamda1*lamda2*hx)/(hy*(lamda1+lamda2)));
Ah(k,k+n)=-lamda2*(hx/hy);
F(k)=hx*integral(f2,y(j),y(j+1))-((lamda2*hx)/((lamda1+lamda2)))*(1/hx)*integral(teta,x(i),x(

j=p;
i=n;
k=i+(j-1)*(n);
Ah(k,k)=lamda1*((hx)/hy+(hy)/hx)+
(2*(lamda1*lamda2*hx)/(hy*(lamda1+lamda2)));
Ah(k,k-1)=-lamda1*(hy/hx);
Ah(k,k-n)=-lamda1*(hx/hy);
Ah(k,k+n)=-(2*(lamda1*lamda2*hx)/
(hy*(lamda1+lamda2)));
F(k)=hx*integral(f1,y(j),y(j+1))-
((lamda1*hx)/((lamda1+lamda2)))*
(1/hx)*integral(teta,x(i),x(i+1));

j=p+1;
i=n;
k=i+(j-1)*(n);
Ah(k,k)=lamda2*((hx)/hy+(hy)/hx)+
(2*(lamda1*lamda2*hx)/(hy*(lamda1+lamda2)));
Ah(k,k-1)=-lamda2*(hy/hx);
Ah(k,k-n)=-(2*(lamda1*lamda2*hx)/
(hy*(lamda1+lamda2)));
Ah(k,k+n)=-lamda2*(hx/hy);
F(k)=hx*integral(f2,y(j),y(j+1))\\
-((lamda2*hx)/((lamda1+lamda2)))\\
*(1/hx)*integral(teta,x(i),x(i+1));
%Resolution
Yh=Ah\(F');
%Post-processeur
Ynum=zeros(2*p+1,n+1);
  for j=1:(2*p -1);
    for i=1:n-1
        k=i+(j-1)*n;
        Ynum(j+1,i+1)=
(Yh(k)+Yh(k+1)+Yh(k+n)+Yh(k+n+1))/4;

```

```
253       end
254     end
255     %Le bord Gamma1
256     j=1;
257     for i=2:n
258         k=i;
259         Ynum(j,i)=(Yh(k)+Yh(k-1))/2;
260     end
261     Ynum(1,1)=Yh(1);
262     Ynum(1,n+1)=Yh(n);
263   %Le bord Gamma2
264   i=n+1;
265     for j=2:2*p
266         k=(j-1)*n;
267         Ynum(j,i)=(Yh(k)+Yh(k+n))/2;
268     end
269     Ynum(2*p+1,n+1)=Yh(2*n*p);
270     %Le bord Gamma3
271     j=2*p+1;
272     for i=2:n
273         k= (i-1)+(2*p-1)*n;
274         Ynum(j,i)=(Yh(k)+Yh(k+1))/2;
275     end
276     %Le bord Gamma4
277     i=1;
278     for j=2:2*p
279         k=i+(j-1)*n;
280         Ynum(j,i)=(Yh(k)+Yh(k-n))/2;
281     end
282     Ynum(2*p+1,1)=g(y(2*p+1));
283   %Validation
284   %Solution exacte
285   [X,Y] = meshgrid(x,y);
286   Z = Y.*((X.^2)/2-X)+Y.^2;
287   subplot(1,2,1), mesh(x,y,Ynum),
288   title('Solution approchee')
289   subplot(1,2,2), mesh(x,y,Z),
290   title('Solution exacte')
```

6.5.3 Résultats graphiques

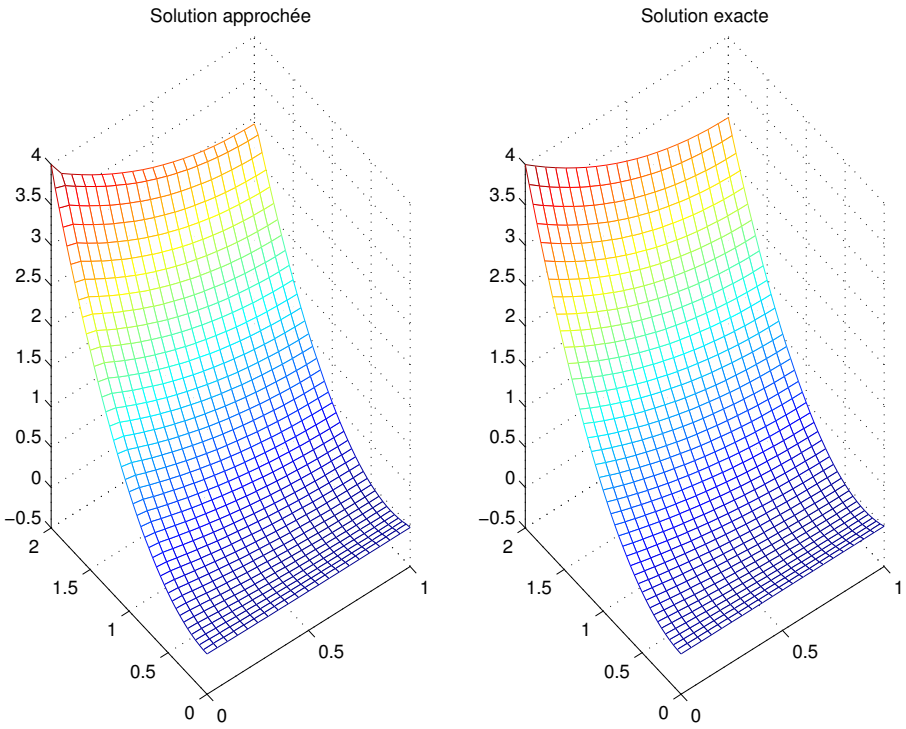


FIGURE 6.3 – Solutions obtenues

Chapitre 7

Conduction de chaleur par maillage non structuré

En 2D, c'est un maillage dont les éléments sont des triangles ou des quadrilatères assemblés de manière quelconque. La topologie est complètement arbitraire. Pour une géomètrie élèmentaire donnée, il est possible de considrer différents nombres de noeuds par élément. Par exemple, on peut considérer l'élément triangle :

* trois noeuds : T3 (ordre 1), * six noeuds : T6 (ordre 2), * ...

En 3D, si l'on considère des élèments d'ordre 1, il s'agit de tétraèdres, de prismes, d'hexaèdres et de pyramides. (+) (-)

* Création de maillages triangulaires ou tétraédriques dans des géomètries quelconques et complexes. * Economie de points par rapport aux maillages structurés notamment dans les zones de raffinement. * Possibilités d'associer différentes topologie d'élements.

* Difficultés pour contrôler la densité des points dans une zone précise. * Contrainte de stockage (tableau de connectivites des éléments). * Plus longs à générer.

Ces maillages sont essentientiellement utilisés par la méthode des éléments finis et parfois en volumes finis 'non-structurés'.

7.1 Maillage

Nous allons utilisé un maillage non-structuré triangulaire M avec $\Omega = \bigcup_{K \in M} K$. Nous allons utiliser le mailleur fourni pour générer le maillage du domaine Ω tout en foréant des points fixes tout au long de l'interface I. Les points fixes seront donc une discrétisation du segment $[0, 1] \times 1$ avec un pas qui doit être inférieur au pas du maillage afin que le mailleur ne génére aucun point trop proche du segment :

On remarque que du fait que le pas des points fixes soit inférieur au pas du maillage assure qu'aucune maille ne traverse l'interface I. Nous avons donc deux domaines distincts Ω_1 et Ω_2. De plus, pour un triangle donné K, il est possible

de déterminer à quel domaine il appartient : il suffit de comparer le minimum des ordonnées de ses sommets avec la valeur 1. Si il est strictement inférieur alors $K \in \Omega_1$, sinon $K \in \Omega_2$.

Connectivité

La connectivité, encore appelée table des connectivités, définit la 'connexion' entre les numéros des noeuds d'un maillage sous forme d'une liste d'élèments. Un élèment se voit alors défini par la liste des numéros des noeuds qui le composent.

Quelques remarques :

* un meme sens de lecture est établi pour permettre un calcul des surfaces au moyen d'un produit vectoriel et obtenir ainsi des surfaces positives (composantes selon 'Z' du produit vectoriel) ;

* une permutation dans l'ordre de lecture des noeuds d'un élèment n'a aucune influence sur le déroulement ultérieur du calcul.

7.2 Construction du schéma

Le schéma se construit par la même méthode : On intègre l'équation (1.1) sur un élément de volume K :

$$\int_K -div(\lambda_m \nabla u(x)) = \int_K f(x)dx$$

$$-\lambda_m \int_K div(\nabla u(x)) = \int_K f(x)dx$$

En appliquant le théorème de flux-divergence :

$$-\lambda_m \int_{\partial K} \nabla u(x)n(x) = \int_K f(x)dx$$

Sur une arête $\sigma \in K$, $n(x)$ reste constante $n(x) = n_\sigma$. Soit :

$$-\lambda_m \sum_{\sigma \in \epsilon_K} \int_\sigma \nabla u(x)n_\sigma(x) = \int_K f(x)dx$$

En considérant l'approximation $\nabla u(x)n_\sigma(x) \approx \frac{u_\sigma - u_K}{d_{K,\sigma}}$ et en posant f_K la moyenne de f sur K, on a :

$$-\lambda_m \sum_{\sigma \in \epsilon_K} \int_\sigma \frac{u_\sigma - u_K}{d_{K,\sigma}} = mes(K)f_K$$

$$\sum_{\sigma \in \epsilon_K} -\lambda_m mes(\sigma)\frac{u_\sigma - u_K}{d_{K,\sigma}} = mes(K)f_K$$

7.3 Calcul des flux numériques

On considère l'élément de volume K délimité par les arêtes $\partial K = \sigma_1 \cup \sigma_2 \cup \sigma_3$. Le raisonnement étant symétrique, nous allons exprimer le flux numérique relatif à une arête σ quelconque. Le flux numerique total étant la somme des flux partiels relatifs à chaque arête.

7.3.1 Flux interne : σ est une arête interne

Le raisonnement reste similaire que celui expliqué dans le cas d'un maillage structuré :

$$F_{K_k,\sigma} = -\lambda_m mes(\sigma)\frac{u_{K_l} - u_{K_k}}{d_{k,l}}$$

avec u_{K_l} et u_{K_k} sont respectivement les valeurs de u au barycentre des mailles K_k et K_l et $d_{k,l}$ la distance entre ces deux barycentres.

7.3.2 Flux au bord de $\sigma \subset \Gamma 1 \bigcup \Gamma 3$

Si $\sigma \subset \Gamma_1 \bigcup \Gamma_3$, on a l'équation suivante :

$$-\lambda_m \vec{\nabla} u.\vec{n_\sigma} = \alpha(u - u_{ext})$$

soit :

$$F_{K_{k,\sigma}} = \int_\sigma -\lambda_m \vec{\nabla} u.\vec{n_\sigma}$$

$$= \alpha((\int_\sigma -\lambda_m u) - mes(\sigma)u_{ext})$$

$$= \alpha(mes(\sigma)u_\sigma - mes(\sigma)u_{ext})$$

$$= \alpha mes(\sigma)(u_\sigma - u_{ext})$$

De plus on sait que :

$$F_{K_{k,\sigma}} = -\lambda_m mes(\sigma)\frac{u_{K_l} - u_{K_k}}{d_{K_k,\sigma}}$$

Par identification :

$$-\lambda_m \frac{u_{K_l} - u_{K_k}}{d_{k,l}} = \alpha(u - u_{ext})$$

Soit :

$$(\alpha + \frac{\lambda_m}{d_{K_k,\sigma}})u_\sigma = \alpha u_{ext} + \frac{\lambda_m}{d_{K_k,\sigma}}u_{K_k}$$

$$u_\sigma = \frac{\alpha d_{K_k,\sigma} u_{ext} + \lambda_m u_{K_k}}{\alpha d_{K_k,\sigma} + \lambda_m}$$

Finalement :

$$F_{K_k,\sigma} = \alpha\, mes(\sigma)(\frac{\alpha d_{K_k,\sigma} u_{ext} + \lambda_m u_{K_k}}{\alpha d_{K_k,\sigma} + \lambda_m} - u_{ext})$$

$$F_{K_k,\sigma} = \frac{\alpha\, mes(\sigma)\lambda_m}{\alpha d_{K_k,\sigma} + \lambda_m}(u_{K_k} - u_{ext})$$

7.3.3 Flux au bord de $\sigma \subset \Gamma 2$

Par intégration de l'équation qui modélise $\sigma \subset \Gamma_2$, on a :

$$F_{K_k,\sigma} = 0$$

7.3.4 Flux au bord de $\sigma \subset \Gamma 4$

On a la CL si $\sigma \subset \Gamma_4$: $u_\sigma = g_\sigma$ qu'on remplace dans l'expression du flux numérique :

7.3.5 Flux sur l'interface I

σ représente la frontière entre K_k et K_l. O a deux cas :

$K_k \in \Omega_1$ et $L_l \in \Omega_2$

Sur l'interface I, on a un saut de flux de chaleur. Cette condition de saut se traduit par l'équation suivante :

$$-\lambda_1 \nabla u_1(x)\, n_1(x) - \lambda_2 \nabla u_2(x)\, n_2(x) = \Theta(x) \qquad \forall x \in I$$

On intègre sur σ, on obtient :

$$\int_\sigma \nabla u_1(x)\, n_1(x) dx - \int_\sigma \nabla u_2(x)\, n_2(x) dx = \int_\sigma \Theta(x) dx$$

Ce qui donne :

$$F_{K_k,\sigma} + F_{K_l,\sigma} = mes(\sigma)\, \Theta_\sigma$$

Avec : $\Theta_\sigma = \frac{1}{mes(\sigma)} \int_\sigma \Theta(x)\, dx$

En remplaçant le flux numérique par son expression :

$$-\lambda_1 \frac{u_\sigma - u_{K_k}}{d_{K_k,\sigma}} mes(\sigma) - \lambda_2 \frac{u_\sigma - u_{K_l}}{d_{K_l,\sigma}} mes(\sigma) = mes(\sigma)\,\Theta_\sigma$$

On en déduit l'expression de u_σ :

$$u_\sigma = \frac{\lambda_1\, d_{K_l,\sigma}\, u_{K_k} + \lambda_2\, d_{K_k,\sigma}\, u_{K_l} - d_{K_k,\sigma}\, d_{K_l,\sigma}\, \Theta_\sigma}{\lambda_1\, d_{K_l,\sigma} + \lambda_2\, d_{K_k,\sigma}}$$

Qu'on remplace dans le flux numérique :

$$F_{K_k,\sigma} = -\lambda_1\, mes(\sigma)\, \frac{\frac{\lambda_1\, d_{K_l,\sigma}\, u_{K_k} + \lambda_2\, d_{K_k,\sigma}\, u_{K_l} - d_{K_k,\sigma}\, d_{K_l,\sigma}\, \Theta_\sigma}{\lambda_1\, d_{K_l,\sigma} + \lambda_2\, d_{K_k,\sigma}} - u_{K_k}}{d_{K_k,\sigma}}$$

En simplifiant, on obtient :

$$F_{K_k,\sigma} = \frac{\lambda_1\, mes(\sigma)}{\lambda_1\, d_{K_l,\sigma} + \lambda_2\, d_{K_k,\sigma}} \left(\lambda_2 \left(u_{K_k} - u_{K_l}\right) + d_{K_l,\sigma}\, \Theta_\sigma\right)$$

$K_k \in \Omega_2$ et $K_l \in \Omega_1$

Par le même raisonnement :

$$F_{K_k,\sigma} = \frac{\lambda_2\, mes(\sigma)}{\lambda_1\, d_{K_k,\sigma} + \lambda_2\, d_{K_l,\sigma}} \left(\lambda_1 \left(u_{K_k} - u_{K_l}\right) + d_{K_l,\sigma}\, \Theta_\sigma\right)$$

7.4 Calcul numérique

7.4.1 Principe

On a le systeme lineaire :

$$F_{K_k} = B_k, \qquad k \in [(1, 2np)]$$

et chaque F_{K_k} s'exprime en fonction des inconnues numériques $(u_i)_{i\in[(1,2np)]}$ et éventuellement de constantes. En déplacent le seconde terme vers le second membre de l'équation, nous obtenons un système linéaire et $N = 2np$ équations et N inconnues de la forme $A.U = B$, que nous pouvons résoudre numériquement en inversant la matrice carré A :

$$U = A^{-1}.B$$

7.4.2 Algorithme

Solution exacte

Code matlab

```
function z=solExacte(x,y)
z=y.*((x.^2)/2-x)+y.^2;
end
```

Programme

Code matlab

```
function $[u,U]$=MailleurNS(a,L1,L2)
    $[ T,S,Fr]$ = lecture;
    $[u,XK]$=solution(T,S,Fr,a,L1,L2);
    v=tracer(u,T,S);
end

%Dans Notre cas : a=1, L1=2 et L2=1

function $[T,S,Fr]$ = lecture
%Lecture des fichiers triangles.txt, sommets.txt et arretes.txt
% Fonction qui lit les fichiers texte et retourne les tableaux
% correspondants

    %Lecture des triangles
    fid1=fopen('triangles.txt');
    nt=fscanf(fid1,'%d',1);
    T=zeros(nt,3);
    for i=1:nt,
        for j=1:3,
            T(i,j)=fscanf(fid1,'%d',1);
        end
    end
    fclose(fid1);

    %Lecture des sommets
    fid2=fopen('noeuds.txt');
    ns=fscanf(fid2,'%d',1);
    S=zeros(ns,2);
    for i=1:ns
        for j=1:2
            S(i,j)=fscanf(fid2,'%f',1);
        end
    end
    fclose(fid2);

    %Lecture de la frontiere
    fid3=fopen('arretes.txt');
    nf=fscanf(fid3,'%d',1);
    Fr=zeros(nf,3);
    for i=1:nf
```

```
41          for j=1:2
42              Fr(i,j)=fscanf(fid3,'%d',1);
43          end
44          if S(Fr(i,1),1)==0 && S(Fr(i,2),1)==0 %l'arete appartient a la frontiere 4
45              Fr(i,3)=4;
46          elseif S(Fr(i,1),1)==1 && S(Fr(i,2),1)==1 %l'arete appartient a la frontiere 2
47              Fr(i,3)=2;
48          elseif S(Fr(i,1),2)==0 && S(Fr(i,2),2)==0 %l'arete appartient a la frontiere 1
49              Fr(i,3)=1;
50          elseif S(Fr(i,1),2)==2 && S(Fr(i,2),2)==2 %l'arete appartient ala frontiere 3
51              Fr(i,3)=3;
52          end
53      end
54      fclose(fid3);
55  end
56
57  function $[u,XK]$= solution(T,S,Fr,a,L1,L2)
58  %Fonction qui retourne la solution numerique u aux points XK
59
60      nt=length(T);
61      A=zeros(nt,nt);
62      b=zeros(nt,1);
63      XK=zeros(nt,2);
64      for k=1:nt
65          if min(S(T(k,1),2),S(T(k,2),2))<1 || min(S(T(k,2),2),S(T(k,3),2))<1
66          b(k)=int2d(@f1,$[S(T(k,1),:); S(T(k,2),:); S(T(k,3),:)]$,100);
67          elseif max(S(T(k,1),2),S(T(k,2),2))>1 || max(S(T(k,2),2),S(T(k,3),2))>1
68              b(k)=int2d(@f2,$[S(T(k,1),:); S(T(k,2),:); S(T(k,3),:)]$,100);
69          end
70
71          XK(k,1:2)=1/3*(S(T(k,1),1:2)+S(T(k,2),1:2)+S(T(k,3),1:2));%Barycentre du
    triangle k
72          for i=1:2
73              for j=i+1:3
74              %On raisonne sur l'arrete [T(k,i)T(k,j)] du triangle T(k)
75                  flag=RechercheFrontiere($[T(k,i) T(k,j)]$,Fr);
76                  mS=((S(T(k,i),1)-S(T(k,j),1))^2 + (S(T(k,i),2)-S(T(k,j),2))^2)^(1/2);
77                                  %mesure de l'arrete
78                  if flag==0
79                  %Si l'arrete n'est pas sur la frontiere
80                      L=RechercheTriangle($[T(k,i) T(k,j)]$,T);
81                                          %Recherche des N des triangles qui ont l'arrete
    donnee
82                      %L'un de ces N sera K (notre triangle), l'autre est donc l (triangle voisin)
83                      if L(1)==k
84                          l=L(2);
85                      else
86                          l=L(1);
87                      end
88                      XL=1/3*(S(T(l,1),1:2)+S(T(l,2),1:2)+S(T(l,3),1:2));
89                                          %Barycentre du triangle L
90                      dKL=((XK(k,1)-XL(1))^2 + (XK(k,2)-XL(2))^2)^(1/2);
91                                          %Distance entre XK et XL
92                      %Ajout du flux passant par notre arrete
93                      if min(S(T(k,i),2),S(T(k,j),2))<1 %Sur O1
```

```
94                              A(k,k)=A(k,k)+L1*mS/dKL;
95                              A(k,l)=A(k,l)-L1*mS/dKL;
96                          elseif max(S(T(k,i),2),S(T(k,j),2))>1  %Sur O2
97                              A(k,k)=A(k,k)+L2*mS/dKL;
98                              A(k,l)=A(k,l)-L2*mS/dKL;
99                          else %Sur l'interface I
100                             dKS=((XK(k,1)-(S(T(k,i),1)+S(T(k,j),1))/2)^2 +
101                                                 (XK(k,2)-(S(T(k,i),2)...
102                                 +S(T(k,j),2))/2)^2)^(1/2);
103                                                     %Distance entre le barycentre
    et le milieu de l'arrete
104                             dLS=((XL(1)-(S(T(k,i),1)+S(T(k,j),1))/2)^2 +
105                                                 (XL(2)-(S(T(k,i),2)...
106                                   +S(T(k,j),2))/2)^2)^(1/2);%Distance entre le barycentre
    ...
107                                      % et le milieu de l'arrete
108                             if XK(2)<XL(2)  %Le triangle k est dans O1
109                                 A(k,k)=A(k,k)+L1*L2*mS/(L1*dLS + L2*dKS);
110                                 A(k,l)=A(k,l)-L1*L2*mS/(L1*dLS + L2*dKS);
111                                 b(k)=b(k)-L1*mS*dLS*teta((S(T(k,i),1)+S(T(k,j),1))/2)/
112                                                     (L1*dLS +L2*dKS);
113                             else %Le triangle k est dans O2
114                                 A(k,k)=A(k,k)+L1*L2*mS/(L2*dLS + L1*dKS);
115                                 A(k,l)=A(k,l)-L1*L2*mS/(L2*dLS + L1*dKS);
116                                 b(k)=b(k)-L2*mS*dLS*teta((S(T(k,i),1)+S(T(k,j),1))/2)/
117                                                     (L2*dLS + L1*dKS);
118                             end
119                         end
120                     else
121                     % Si l'arrete est sur une frontiere
122                         dKS=((XK(k,1)-(S(T(k,i),1)+S(T(k,j),1))/2)^2 +
123                                             (XK(k,2)-(S(T(k,i),2)+...
124                         S(T(k,j),2))/2)^2)^(1/2);%Distance entre
125                                                 %le barycentre et le milieu de l'arrete
126                         if Fr(flag,3)==1
127                         %Si l'arrete est sur la frontiere 1
128                             A(k,k)=A(k,k)+(a*L1*mS)/(L1+a*dKS);
129                             b(k)=b(k)+(a*L1*mS)/(L1+a*dKS)
130                                                 *uext1((S(T(k,i),1)+S(T(k,j),1))/2);
131                         elseif Fr(flag,3)==3
132                         %Si l'arrete est sur la frontiere 3
133                             A(k,k)=A(k,k)+(a*L2*mS)/(L2+a*dKS);
134                             b(k)=b(k)+(a*L2*mS)/(L2+a*dKS)*
135                                                 uext2((S(T(k,i),1)+S(T(k,j),1))/2);
136                         elseif Fr(flag,3)==4
137                             if min(S(T(k,i),2),S(T(k,j),2))<1  %Sur O1
138                                 A(k,k)=A(k,k)+L1*mS/dKS;
139                                 b(k)=b(k)+L1*mS*g((S(T(k,i),2)+
140                                                     S(T(k,j),2))/2)/dKS;
141                             elseif max(S(T(k,i),2),S(T(k,j),2))>1  %Sur O2
142                                 A(k,k)=A(k,k)+L2*mS/dKS;
143                                 b(k)=b(k)+L2*mS*g((S(T(k,i),2)+
144                                                     S(T(k,j),2))/2)/dKS;
145                              end
146                         end
```

```
147                     end
148                 end
149             end
150         end
151         u=A\b;
152     end
153
154     function v = tracer(u,T,S)
155     %Tracer de la fonction u ainsi que la solution exacte
156
157         nt=length(T);
158         ns=length(S);
159         v=zeros(ns,1);
160         vex=zeros(ns,1);
161         for i=1:ns
162            k=0;
163            for j=1:nt
164                if T(j,1)==i || T(j,2)==i || T(j,3)==i
165                    k=k+1;
166                    v(i)=v(i)+u(j);
167                end
168            end
169             v(i)=v(i)/k;
170              vex(i)=solExacte(S(i,1),S(i,2));
171         end
172
173          subplot(1,2,1)
174          trimesh(T, S(:,1),S(:,2), v)
175          xlabel('x');
176          ylabel('y');
177          zlabel('u');
178          title('Solution apprchee')
179          subplot(1,2,2)
180          trimesh(T, S(:,1),S(:,2),vex)
181          xlabel('x');
182          ylabel('y');
183          zlabel('u');
184          title('Solution exacte')
185     end
186
187
188
189
190     function flag = RechercheFrontiere(Ar, Fr )
191     %Fonction qui cherche l'arrete Ar dans le tableau Fr et retourne l'indice de l'arrete ou 0 sinon
192         flag=0;
193         for i=1:length(Fr)
194             count=0;
195             for j=1:2
196                 for k=1:2
197                     if Ar(j)==Fr(i,k)
198                         count=count+1;
199                     end
200                 end
201             end
```

```
202             if count==2
203                 flag=i;
204             end
205         end
206     end
207
208     function L = RechercheTriangle (A, T)
209     %Fonction qui parcoure le tableau des triangles et retourne
210     %les indices des deux triangles que l'arrete A appartient
211         L=zeros(2);%Tableau des deux indices
212         l=1;
213         for i=1:length(T)
214             s=0;
215             for j=1:3
216                 if T(i,j)==A(1) || T(i,j)==A(2)
217                     %Le triangle T(i) a un point commun avec l'arrete
218                     s=s+1;
219                 end
220             end
221             if s==2
222                 %Le triangle T(i) a deux points communs avec l'arrete
223                 L(l)=i;
224                 l=l+1;
225             end
226         end
227     end
```

Second Membre f

Code matlab

```
1  function z=f1(y,z)
2  z=y.^2+z.^2;
```

Code matlab

```
1  function z=f2(y,z)
2  z=1+y.^2+z.^2;
```

Les condition aux limites,fonction g :

Code matlab

```
1  function z=g(y)
2  z=y.^2;
3  end
```

Les condition aux limites, la fonction θ

Code matlab

```
function z=teta(x)
z=-((x.^2)/2-x+2);
end
```

Les condition aux limites,la fonction U_{ext1}

Code matlab

```
function z=uext1(x)
z=-(x.^2)+2*x;
end
```

Les conditions aux limites, la fonction U_{ext2}

Code matlab

```
function z=uext2(x)
z=3*((x.^2)/2-x)+8;
end
```

7.4.3 Résultats graphiques

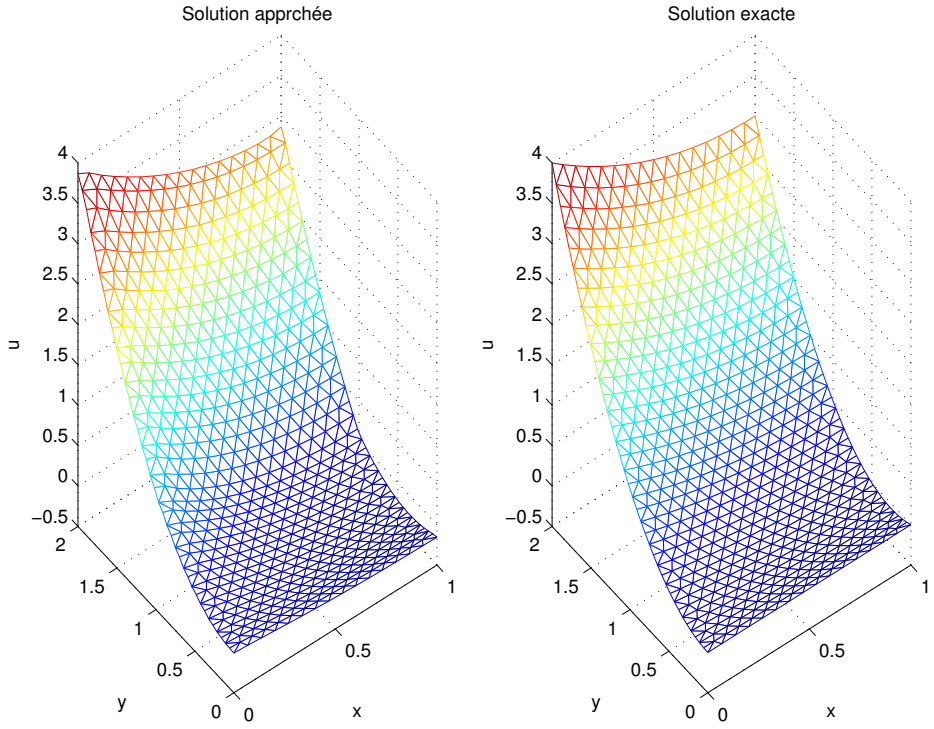


FIGURE 7.1 – Solutions obtenues

Conclusion

Les difficultés des méthodes numériques pour la résolution des équations aux dérivées partielles ont des origines diverses : elles proviennent soit des subtilités liées au choix de l'algorithme, soit des problèmes de stockage en mémoire. Des difficultés supplémentaires apparaissent pour des équations aux dérivées partielles non-linéaires où il est nécessaire d'introduire des nouveaux critères pour l'analyse de la stabilité des algorithmes. Le domaine de la résolution des équations aux dérivées partielles reste un domaine très actif de recherche en analyse numérique, soulignant à la fois l'intérêt de nouvelles méthodes et la nécessite d'obtenir des méthodes de plus en plus performantes.