Cours: Equations aux Dérivées Partielles
Méthodes des Différences Finies

Préface. *Ce cours est une enquête de pratique des techniques de résolution numérique des différentes classes d'équations aux dérivées partielles (elliptiques, paraboliques et hyperboliques). L'accent sera mis sur la programmation de schémas numériques à des problèmes pratiques dans l'ingénierie et les sciences physiques. La résolution comprendra la méthode des différences finies et la méthode des caractéristique pour les EDP hyperboliques. Pratiquement, elles seront pleinement utilisées en MATLAB et ses fonctionnalités de programmation.*

Contents

List of Figures

1. Introduction

Les équations aux dérivées partielles (EDP) sont omniprésentes dans toutes les sciences, puisqu'elles apparaissent aussi bien en dynamique des structures, mécanique des fluides que dans les théories de la gravitation ou de l'électromagnétisme (Exemple: les équations de Maxwell). Elles sont primordiales dans des domaines tels que la simulation aéronautique, la synthèse d'images, la prévision météorologique, la démographie, ou les finances. Enfin, les équations les plus importantes de la relativité générale et de la mécanique quantique sont également des EDP. Ce sont des équations indispensables pour la résolution de presque la totalité des problèmes dans ces domaines.
Nous pouvons citer par exemple:

1. l'équation de Schrdinger indispensable à la mécanique quantique: $\frac{\hbar^2}{2m}\frac{\partial^2 u}{\partial x^2}+\imath\hbar\frac{\partial u}{\partial t}-U(x)u=0$
2. l'équation d'advection qui décrit comment une quantité est transportée dans un courant (par exemple un poluant dans de l'eau): $\frac{\partial u}{\partial t}(x,t)+c\frac{\partial u}{\partial x}(x,t)=f(x,t)$, c étant la vitèsse du milieu qui est souvent une constante.
3. l'équation de Black-Scholes utilisée en finances:
   $\frac{\partial c}{\partial t}+S\frac{\sigma^2}{2}\frac{\partial^2 c}{\partial S^2}+rS\frac{\partial c}{\partial S}-rc=0$ où $c=c(t,S)$ est un prix et $\sigma,\ r$ des constantes.
4. L'équation d'ondes decrivant les phénomènes de propagation des ondes sonores et des ondes électromagnétiques comme la lumière dans des milieux comme l'air ou le vide physique: $\Delta u-\frac{1}{c^2}\frac{\partial^2 u}{\partial t^2}=0\Rightarrow\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}=\frac{1}{c^2}\frac{\partial^2 u}{\partial t^2}$. Le nombre c représente la célérité pour le cas de la lumière ou la vitesse de propagation de l'onde u.
5. L'équation de Fourrier ou équation de la chaleur qui decrit l'évolution de la température en fonction du temps et de l'espace: $\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}=\frac{1}{\alpha}\frac{\partial u}{\partial t}$. Le nombre α est appelé diffusivité thermique du milieu.

Certaines de ces EDP ont été résolues analytiquement et leurs solutions sont connues. Toutefois, un nombre important d'autres existent sans solutions analytiques. C'est dans cette optique que les recherches se sont penchées sur les méthodes numériques pour arriver à approximer les solutions de ces équations.

Notons que malgré ces efforts indéniables, il n'existe pas de méthodes universelles pour la résolution numérqiue des EDP. L'algorithme de résolution dépend très étroitement du type de problème posé. C'est pour cela que nous allons restreindre notre champs d'étude. On exigera que l'équation satisfasse quelques propriètés comme la *linéarité* pour que la résolution soit possible.

2. Définition

En mathématiques, plus précisément en calcul différentiel, une équation aux dérivées partielles ou équation différentielle partielle (EDP) est une équation dont les solutions sont les fonctions inconnues vérifiant certaines conditions concernant leurs dérivées partielles. C'est une équation

mathématique contenant en plus de la variable dépendante (u dans les cas suivants) des variables indépendantes $(x, y, ...) \in \mathbb{R}^n$ et une ou plusieurs dérivées partielles qu'on peut écrire sous la forme:

$$F(x, y, ..., u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, \frac{\partial^2 u}{\partial x^2}, \frac{\partial^2 u}{\partial y^2}, ...) = 0,$$

Exemple

- l'équation aux dérivées partielles $\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = 0$ qui admet comme solutions: $u(x, y) = (x + y)^3$, $u(x, y) = sin(x - y)$,...
- l'équation de la Laplace $\Delta u = 0$, en dimensions 2 2D, qui admet aussi au moins deux solutions dont $u(x, y) = x^2 - y^2$ et $v(x, y) = e^x sin(y)$

les conditions étant moins strictes que dans le cas d'une équation différentielle ordinaire; les problèmes incluent souvent des conditions aux limites qui restreignent l'ensemble des solutions. Pour assurer donc l'unicité de la solution, comme on le fait avec les équations différentielles ordinaires, *EDO*, on tiendra compte des conditions prédonnées comme les ***conditions aux limites*** et les ***conditions initiales***.

Classification des EDP linéaires du second ordre

Comme il est dit haut, il n'existe pas de méthodes universelles pour la résolution des EDP, nous allons nous contenter de celles qui sont linéaires et du second ordre.

Quand on pose $X = (x_1, x_2, ..., x_n) \in \mathbb{R}^n$, une équation aux dérivées partielles du second ordre sera de la forme:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} A_{i,j}(X) \frac{\partial^2 u}{\partial x_i \partial x_j}(X) + \sum_{i=1}^{n} B_i(X) \frac{\partial u}{\partial x_i}(X) + Cu = G(X)$$

avec $A_{i,j}, B_i, C, G$ des fonctions indépendantes de u ne s'annulant pas toutes simultanément dans $\mathbb{R}^n$. Si nous nous limitons dans $\mathbb{R}^2$, c'est à dire $X = (x, y) \in \mathbb{R}^2$ l'égalité précédemment posée prend la forme de:

$$A \frac{\partial^2 u}{\partial x^2} + B \frac{\partial^2 u}{\partial x \partial y} + C \frac{\partial^2 u}{\partial y^2} + D \frac{\partial u}{\partial x} + E \frac{\partial u}{\partial y} + Fu = G(x, y) \tag{2.1}$$

La classe d'une telle équation est détérminée par le calcul de
$\Delta = B^2(x_0, y_0) - 4A(x_0, y_0)C(x_0, y_0)$

- Si $\Delta < 0$, on parle d'une équation elliptique,

- Si $\Delta = 0$, l'EDP est dite parabolique,
- Si $\Delta > 0$, on a une équation hyperbolique.

Résoudre numériquement un exemple pour chaque type d'équation aux dérivées partielles susmentionnés constituera nos projets dans ce rapport.

Remarque:

Notons que les méthodes numériques passent toujours par des discrétisations des problèmes analytiques en des problèmes numériques et qu'il existe une infinité des méthodes de discrétisation d'une équation. Nous ne pouvons jamais les énumerer toutes mais les plus couramment utilisées pour la résolution des équations aux dérivées partielles sont:

1. La méthode des différences finies,
2. La méthode des éléments finis,
3. la méthode des volumes finis,
4. la méthode des caractéristiques.

Mais sachez que nous n'utiliserons ici que la méthode des *différences finies*.
La méthode consiste à remplacer les dérivées partielles par des différences divisées ou combinaisons de valeurs ponctuelles de la fonction en un nombre fini de points discrets ou noeuds du maillage. L'avantage de cette méthode est qu'il y a une grande simplicité d'écriture et un faible cot de calcul. Elle est couramment pratique et facile d'accès. Elle repose sur deux notions : la discrétisation des opérateurs de dérivation ou différentiation et la convergence du schéma numérique ainsi obtenu. Son inconvénient est qu'on se limite à des géométries simples, et qu'il y a des difficultés de prise en compte des conditions aux limites de type Neumann.

Maillage:

Puisqu'on a évoqué le mot *maillage* dans le paragraphe précédent et qu'on en aura tout le temps besoin, définissons-le ici.
On appelle maillage un ensemble de points du domaine de définition sur lequel on va appliquer la méthode des différences finies. Pour une application définie sur un segment de $\mathbb{R}$, on ajoutera en général les deux extrémités du segment; pour un maillage en dimension supérieure, on sera amené à choisir, éventuellement, des points du contours du domaine de définition. On appelle le pas du maillage la distance entre deux points successifs du maillage voisins. En dimension 1, cela se simplifie en différence des abscisses. Ce pas n'est pas nécessairement constant, il peut même être judicieux de ne pas le fixer comme tel. Le pas (global) de l'approximation peut être défini comme le plus grand pas du maillage. Ainsi, si ce pas global tend vers 0, cela veut dire

que la répartition des points du maillage dans l'intervalle choisi tend à se faire sur tout le domaine d'étude par densité.
Exemple: Pour un intervalle de validité $[0, 2]$, avec n le nombre des pas, on aura $n + 1$ points qui sont donnés par la relation $x_i = i \times h$ avec $h = \frac{2}{n}$ constant, $0 \leq i \leq n$.

Notation indicielle:

Durant ces projets nous utiliserons souvent la *notation indicielle*. C'est pourquoi nous voulons en rappeler le principe. si x est un des vecteurs de base du repère (quadrillage) discrétisé, nous noterons le point $x(i)$, qui est la i^{eme} abscisse par x_i et de même la j^{eme} ordonnée $y(j)$ sera noté y_j et si u est maintenant la fonction, ici la solution de l'équation aux dérivées partielles dépendant seulement des variables de l'espace, on remplacera $u(x_i, y_j)$ par $u_{i,j}$. Si, en plus des variables de l'espace, il existe une variable temprelle $t(k) = t_k$, alors la fonction $u(x_i, y_j, t_k)$ sera notée $u^k_{i,j}$.
En résumé, les indices des variables spatiales resteront en indices et celui du temps sera en exposant. C'est ce qu'on appelera la *notation indicielle*.

Chapitre 1

Résolution d'une EDP elliptique

1. Discrétisation de l'EDP:

Soit :

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2}{\partial x^2} = 0,\ \forall (x, y) \in [a, b] \times [c, d]$$

On prendra h_x et h_y les pas de discrétisation des intervalles $[a, b]$ et $[c, d]$

1. Discrétisation de l'intervalle $[a, b]$
   $h_x = \frac{b-a}{n_x}$ (n_x *étant le nombre d'intervalles dans* $[a, b]$)
   $\Rightarrow x(i) = x_i = a + i \times h_x,\ i = 0, 1, \cdots, n_x$

2. Discrétisation de l'intervalle $[c, d]$
   $h_y = \frac{d-c}{n_y}$ (n_y *étant le nombre d'intervalles dans* $[c, d]$)
   $\Rightarrow y(j) = y_j = c + j \times h_y,\ j = 0, 1, \cdots, n_y$

Remark 1. *: Constatons que* $x_{i+1} = a + (\imath + 1)h_x = (a + \imath h_x) + h_x = x_i + h_x$*. Dans la suite, nous remplacerons chaque fois* $x_i + h_x$*,* $x_i - h_x$*,* $y_j + h_y$*,* $y_j - h_y$ *succéssivement par* x_{i+1}*,* x_{i-1}*,* y_{i+1}*,* y_{i-1}*.*

1.1. Méthode des différences finies:

Cette méthode consiste à approximer les dérivées partielles d'une équation au moyen des developpemets de Taylor et ceci se déduit directement de la définition de la dérivée.
Soit $f(x, y)$ une fonction continue et dérivable de classe C^∞, alors la dérivée partielle prémière de f par rapport à x est calculée par la formule:

$$f'_x(x, y) = \lim_{h_x \to 0} \frac{f(x + h_x, y) - f(x, y)}{h_x}$$

Si $h_x <<< 1$, le développement de Taylor au voisinage de 0 de $f(x+h_x, y)$ donne:
$f(x+h_x, y) = f(x,y) + h_x \frac{\partial f}{\partial x} + \theta(h_x) \simeq f(x,y) + h_x \frac{\partial f}{\partial x}$ avec une erreur de l'ordre de h_x.

$$\Rightarrow \frac{\partial f(x,y)}{\partial x} \simeq \frac{f(x+h_x, y) - f(x,y)}{h_x}$$

Ceci est appelé le *schéma avant*.
De la même manière, nous pouvons aussi donner le *schéma arrière* qui est de la forme:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{h_x \to 0} \frac{f(x,y) - f(x-h_x, y)}{h_x}$$

Avec la formule de Taylor, ceci nous donne:
$f(x,y) = f(x-h_x, y) + h \frac{\partial f(x,y)}{\partial x} + \theta(h_x) \simeq f(x-h_x, y) + h \frac{\partial f(x,y)}{\partial x}$ avec une erreur de h_x.

$$\Rightarrow \frac{\partial f(x,y)}{\partial x} \simeq \frac{f(x,y) - f(x-h_x, y)}{h_x}$$

La somme de ces deux schémas nous donne le *schéma centré* suivant:

$$\frac{\partial f(x,y)}{\partial x} \simeq \frac{f(x+h_x, y) - f(x-h_x, y)}{2h_x}.$$

En résumé, on a les trois approximations suivantes pour la dérivée partielle première de $f(x,y)$ par rapport à x avec la formule de Taylor:

$$f'_x(x,y) = \lim_{h_x \to 0} \frac{f(x+h_x, y) - f(x,y)}{h_x} \approx \begin{cases} \frac{f(x+h_x,y)-f(x,y)}{h_x} & \text{schéma avant} \\ \frac{f(x,y)-f(x-h_x,y)}{h_x} & \text{schéma arrière} \\ \frac{f(x+h_x,y)-f(x-h_x,y)}{2h_x} & \text{schéma centré} \end{cases}$$

La dérivée séconde f''_x de $f(x,y)$ sera alors de la forme:

$$\frac{\partial^2 f}{\partial x^2} \approx \frac{\frac{f(x_{i+1},y_j)-f(x_i,y_j)}{h_x} - \frac{f(x_i,y_j)-f(x_{i-1},y_j)}{h_x}}{h_x}$$

$$\frac{\partial^2 f}{\partial x^2} \simeq \frac{f(x_{i+1}, y_j) - 2f(x_i, y_j) + f(x_{i-1}, y_j)}{h_x^2} \tag{1.1}$$

Nous utiliserons tour à tour ces égalités dans la suite pour approximer les dérivées partielles.

1.2. Approximation de l'équation différentielle partielle

Soit l'équation de Laplace:

$$\Delta u = 0 \Leftrightarrow \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \tag{1.2}$$

Posons $u(x_i, y_j) = u_{i,j}$ (*en notation indicielle*). Compte ténu de la relation 1.1 du paragraphe précécdent,

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

Puisque x_i et y_j jouent un rle symétrique dans l'équation du potentiel (*de Laplace*), un raisonnement analogue à celui de l'approximation de f_x'' nous donne:

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

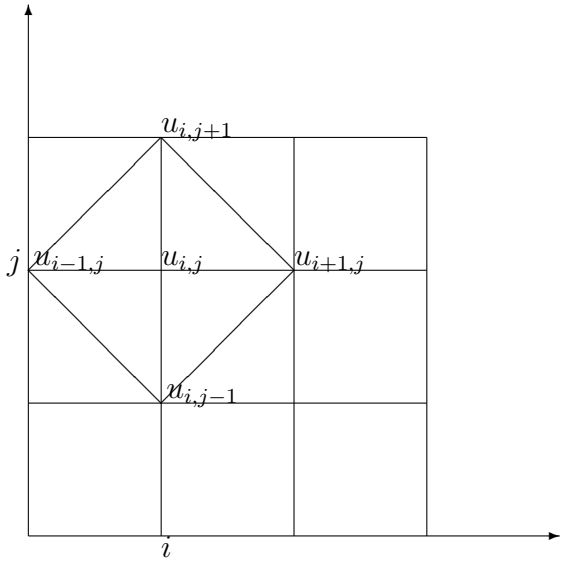
rapportons ces aproximations dans l'EDP 1.2:

$$\Leftrightarrow \Delta u \approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h_x^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h_y^2} = 0$$

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

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

Remark 2. *: A chaque étape, nous rémarquons que pour calculer la valeur de $u_{i,j}$ au point (x_i, y_j) nous avons besoin de connaître les points $u_{i-1,j}$, $u_{i,j-1}$, $u_{i+1,j}$ et $u_{i+1,j}$ comme l'indique le dessin suivant:*



C'est pour cela que nous appelons cette formule ***la formule à 5 points*** *qui peut être representée comme suit:*

$$\Delta u = 0 \Longrightarrow \frac{1}{h^2} \begin{Bmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{Bmatrix} u_{i,j} = 0$$

2. Résolution de l'EDP par la méthode directe:

Exemple:

Soit à résoudre l'équation de Laplace

$$\begin{cases} \Delta u = 0 \text{ dans le domaine } (x,y) \in [0,20] \times [0,10] \\ u(x,0) = u(x,10) = u(0,y) = 0 \text{ et } u(20,y) = 100 \\ h_x = h_y = h \in \{5, 2.5, 1.25, 0.625, 0.3125\} \end{cases}$$

Tout en variant h, résoudre cette EDP:

1. En utilisant la méthode directe.
2. En utilisant la méthode de relaxation de Liebmann.
3. Conclure.

2.1. Cas où $h = 5$

On a: $h_x = \frac{b-a}{n_x} \Rightarrow n_x = \frac{b-a}{h_x} = \frac{20-0}{5} = 4$ et $n_y = \frac{d-c}{h_y} = \frac{10-0}{5} = 2$ La grille maillée contient alors $(n_x+1)\times(n_y+1)$ mailles vu que nous avons à rajouter les points où $x_i = 0$ et ceux où $y_j = 0$ c'est à dire les points intersection de la courbe avec les axes. Mais comme les conditions aux limites nous donnent les images sur les bords, alors les points inconnus restent seulement ceux de l'intérieur du cadrillage. Ce qui fait donc que le nombre d'inconnues est alors $(n_x - 1) \times (n_y - 1) = 3 \times 1 = 3$ Nous obtenons le système de trois équations à trois inconnues suivant:

$$\begin{cases} -4u_{1,1} + u_{2,1} + 0u_{3,1} = 0 \\ u_{1,1} - 4u_{2,1} + u_{3,1} = 0 \\ 0u_{1,1} + u_{2,1} - 4u_{3,1} = -100 \end{cases}$$

Il nous reste maintenant à résoudre le système matriciel: $A \times U = B$ Avec:

$$A = \begin{pmatrix} -4 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & -4 \end{pmatrix}, B = \begin{pmatrix} 0 \\ 0 \\ -100 \end{pmatrix} \text{ et } U = \begin{pmatrix} u_{1,1} \\ u_{2,1} \\ u_{3,1} \end{pmatrix}$$

Avec une des méthodes vues en Analyse Numérique II (*résolution des systèmes linéaires*), nous obténons la solution:

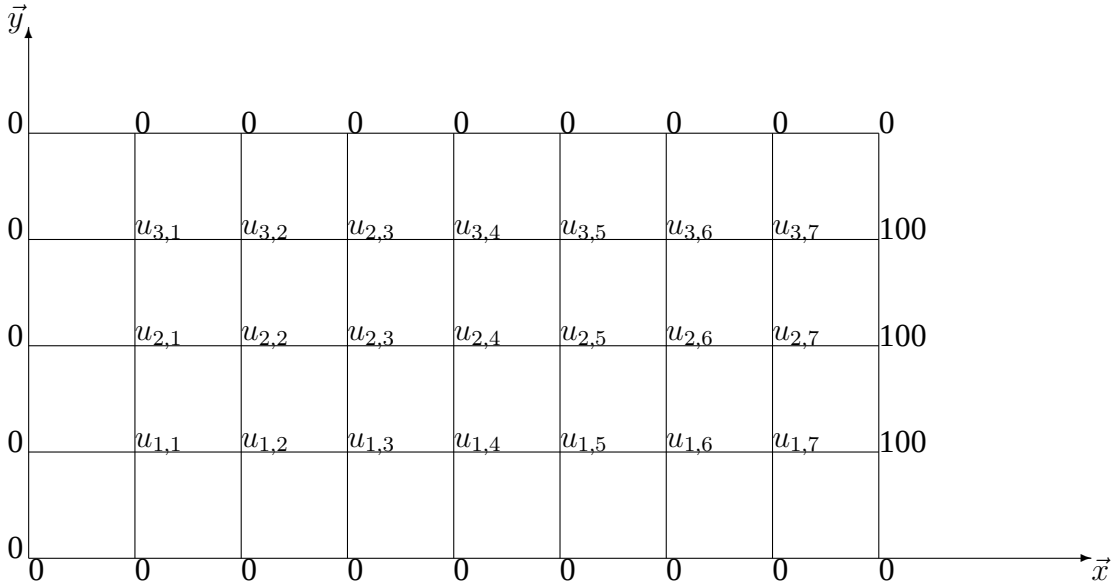
$$U = \begin{pmatrix} 1.786 \\ 7.143 \\ 26.786 \end{pmatrix}$$

2.2. Cas où $h = 2.5$

Nous avons aussi: $n_x = \frac{b-a}{h} = \frac{20-0}{2.5} = 8$ et $n_y = \frac{d-c}{h} = \frac{10-0}{2.5} = 4$, ce qui nous donne un système à $n = (n_x - 1) \times (n_y - 1) = 7 \times 3 = 21$ équations à 21 inconnues de la forme:

$$\begin{cases} -4u_{1,1} + u_{2,1} + \cdots + u_{1,2} + \cdots\cdots = 0 \\ u_{1,1} - 4u_{2,1} + u_{3,1} + \cdots + u_{2,2} + \cdots = 0 \\ \vdots \\ \cdots\cdots\cdots + u_{6,1} - 4u_{7,1} + \cdots\cdots = -100 \\ u_{1,1} + \cdots\cdots - 4u_{2,1} + u_{2,2} + \cdots\cdots = 0 \\ \vdots \end{cases}$$

Les conditions aux limites nous ont ramené à avoir la grille suivante dans laquelle nous allons chercher les inconnus de l'équation:



Avec un petit programme sur Matlab, nous transformons la matrcie U en un vecteur $\vec{v}$ pour pouvoir bien résoudre le système sans erreur puisque la résolution du système $AU = B$ éxige que U soit un vecteur. Voici le programme qui a assuré la transformation:

****

for j=1:ny-1
for i=1:nx-1
v(k)=u(i,j);
k=k+1;
end
end

****

Ce qui nous donne après, le système:

$$\begin{cases} -4v_1 + v_2 + \cdots + v_8 + \cdots\cdots = 0 \\ v_1 - 4v_2 + v_3 + \cdots + v_9 + \cdots = 0 \\ \vdots \\ \cdots\cdots\cdots + v_6 - 4v_7 + \cdots\cdots = -100 \\ v_1 + \cdots\cdots - 4v_8 + v_9 + \cdots\cdots = 0 \\ \vdots \end{cases}$$

Il nous reste maintenant à résoudre le système matriciel suivant: $A \times \vec{v} = B$, avec

$$A = \begin{pmatrix} -4 & 1 & & & 1 & & & & & \\ 1 & -4 & 1 & & & 1 & & & & \\ 0 & 1 & -4 & 1 & & & 1 & & & \\ & & \ddots & & \ddots & & & \ddots & & \\ & & & 1 & -4 & 0 & & & 1 & \\ 1 & & & & 0 & -4 & 1 & & & 1 \\ 0 & \ddots & & \ddots & & 1 \ddots & \ddots & \ddots & & \\ \vdots & & & & & & & & & \\ 0 & & & & 1 & & & & 1 & -4 \end{pmatrix} \text{ et } B = \begin{pmatrix} 0 \\ \vdots \\ \vdots \\ -100 \\ \vdots \\ \vdots \\ -100 \\ \vdots \\ \vdots \\ -100 \end{pmatrix}$$

L'utilisation de la méthode de **Jacobi** ou **Gauss-Seidel** pourrait nous donner les solutions du système. Toutefois, nous allons adapter la résolution directe comme le titre l'indique pour résoudre ce système. voici le programme complet, expliqué, saisi en Matlab qui nous a permis d'avoir la matrice U à partir des calculs des élements du vecteur $\vec{v}$.

****

```
clc;clear
h=2.5;
a=0;
b=20;
c=0; d=10;
nx=(b-a)/h;
ny=(d-c)/h;
n=(nx-1)*(ny-1);
%%% (remplissage des éléments de la matrice A)
A=zeros(n);
for i=1:(n-1)
A(i,i)=-4;
A(i+1,i)=1;
A(i,i+1)=1;
```

```
if (mod(i,(nx-1))==0)
A(i+1,i)=0;
A(i,i+1)=0;
end
end
for i=1:n-nx+1
A(nx-1+i,i)=1;
A(i,nx-1+i)=1;
end
A(n,n)=-4;
%%% (remplissage des éléments de la matrice B)
for i=1:n
B(i)=0;
if (mod(i,nx-1)==0)
B(i)=-100;
end
end
%%% (résolution du système Av=B et transformation du vecteur v en la matrice U).
V = A\B';
k=1;
for j=1:ny-1
for i=1:nx-1
u(j,i)=V(k);
k=k+1;
end
end
%%% (décallage des élements pour insérer les conditions aux limites)
for j=ny:-1:2
for i=nx:-1:2
u(j,i)=u((j-1),i-1);
end
end
for i=1:nx
for j=1:ny
u(1,i)=0;
u(j,1)=0;
u(ny+1,i)=0;
u(j,nx+1)=100;
end
end
u(1,nx+1)=0;
%%% les vecteurs x et y
```

x=0:h:b; y=0:h:d;
%%% (affichage la courbe en tenant compte des vecteurs x ,y et de la matrice U)
mesh(x,y,u)
****

Remark 3. *: Pour évaluer le système avec les autres valeurs de h, il suffit de remplacer 2.5 par les autres valeurs et compiler le programme, cela donnera la courbe correspondante pour chaque valeur de h.*

Voici en résumé, les courbes qu'on obtient en variant h:

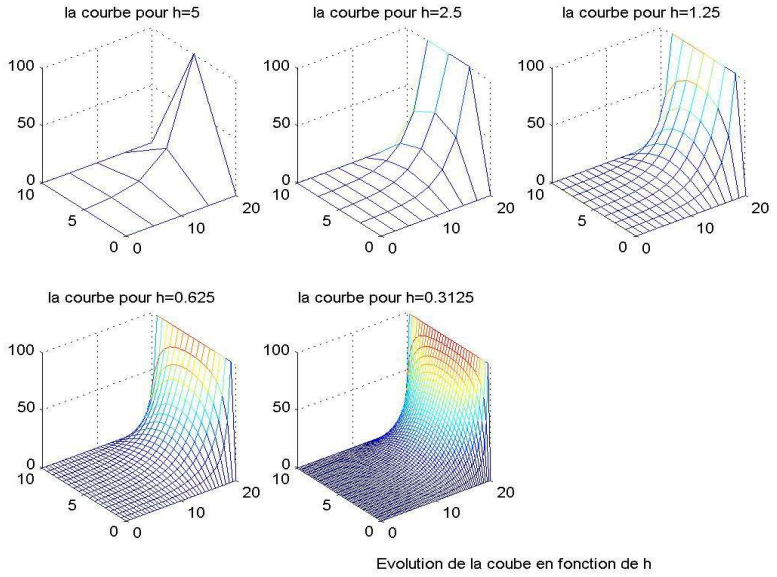


Figure 1.1: Evolution de la courbe $\Delta u = 0$ en fonction de h avec la méthode directe

Malgré que cette méthode ait pu nous donner une solution approchée à l'EDP $\Delta u = 0$, celle-ci est obténue par une approximation des $\frac{\partial^2 u}{\partial x^2}$ et $\frac{\partial^2 u}{\partial y^2}$ par l'utilisation de la formule de Taylor tronquée à l'ordre 2. Nous avons donc commis une erreur de l'ordre de h^2.
Non seulement cette erreur commise est considérable mais le calcul des éléments du vecteur $\vec{v}$ est de plus en plus couteux en mémoire. Par exemple pour $h_x = h_y = h = 1.25$, la matrice $A \in \mathfrak{M}(105)$. La résolution de ce système matriciel avec A une matrice carrée $(105, 105)$ nécéssite une mémoire de 14000 bytes (octets) dans la machine. Or, pour affiner beaucoup plus la solution numérique vers la solution analytique, nous dévons faire tendre h vers zero pour qu'il soit très proche de la limite approximée par la méthode de Taylor. Ce qui augmentera encore la nécéssité

de mémoire et qui risquera de planter la machine sinon nous devons avoir des machines à mémoires gigantesques (qui ne sont pas à la portée de tout le monde!).

3. Méthode de relaxation de Liebmann

Pour rémèdier le problème précédement posé, Liebmann a pensé à la méthode itérative de **Gauss-Seidel** puisque la matrice A est à diagonales dominantes, c'est à dire:

$$\mid A_{i,i} \mid > \sum_{j=1, j\neq i}^{n} \mid A_{i,j} \mid, \forall 1 \leq i \leq n$$

3.1. Méthode:

Le schéma numérique obtenu dans la partie 1 de ce projet nous donne:

$$\begin{aligned} & u_{i-1,j} + u_{i,j-1} - 4u_{i,j} + u_{i+1,j} + u_{i,j+1} = 0 \\ \Leftrightarrow \; & u_{i,j} = \frac{u_{i-1,j} + u_{i,j-1} + u_{i+1,j} + u_{i,j+1}}{4} \\ \Leftrightarrow \; & u_{i,j}^{k+1} = \frac{u_{i-1,j}^{k+1} + u_{i,j-1}^{k+1} + u_{i+1,j}^{k} + u_{i,j+1}^{k}}{4} \end{aligned} \tag{3.1}$$

Cette méthode est appelée *méthode itérative de **Liebmann***. Une condition initiale est éxigée, comme pour toute autre méthode itérative, pour pouvoir commencer le procéssus.
Pour cette méthode, on a besoin de seulement 6000 bytes (octets) pour résoudre le système au lieu de 14000 bytes pour la méthode directe, pour $h = 1.25$.
On peut aussi écrire:

$$u_{i,j}^{k+1} = u_{i,j}^{k} + \left[\frac{u_{i-1,j}^{k+1} + u_{i,j-1}^{k+1} + u_{i+1,j}^{k} + u_{i,j+1}^{k}}{4} \right]$$

le terme entre crochets est appelé *résidus* conu comme un ajustement de la valeur précédente $u_{i,j}^{k}$ pour avoir la valeur $u_{i,j}^{k+1}$

3.2. Idée de Relaxation:

Au lieu d'ajouter éxactement le résidus, on ajoute un terme un peu plus grand en introduisant un *facteur de relaxation* w compris entre 1 et 2 pour avoir la nouvelle relation :

$$u_{i,j}^{k+1} = u_{i,j}^{k} + w \times \left[\frac{u_{i-1,j}^{k+1} + u_{i,j-1}^{k+1} + u_{i+1,j}^{k} + u_{i,j+1}^{k}}{4} \right]$$

w est appelé un **optimum**, et il est obtenu, dans les conditions de Dirichlet, par une estimation plus raisonnable comme étant la plus petite valeur des solutions de l'équation

$$(\cos\frac{\pi}{n_x}+\cos\frac{\pi}{n_y})^2w^2-16w+16=0$$

La résolution de cette équation du sécond dégré nous donne:

$$w_{opt}=\frac{4}{2+\sqrt{4-(\cos\frac{\pi}{n_x}+\cos\frac{\pi}{n_y})^2}}$$

Nous reprenons ici le même exercice énoncé au début du projet et le résoudre en utilisant cette nouvelle relation. voici, en Matlab, le programme à compiler avec $h=0.625$. Pour avoir les autres courbes, il suffit de changer la valeur de h par les autres valeurs.

****

```
clear; clc;
a=0; b=20;
c=0; d=10;
h=0.625 ;
%%% les vecteurs ox, oy x=0:h:b; y=0:h:d;
nx=(b-a)/h; ny=(d-c)/h;
c=cos(pi/nx)+cos(pi/xy);
w=4/(2+sqrt(4-C*C));
u=zeros(ny+1,nx+1);
v=zeros(ny+1,nx+1);
for j=2:ny
u(j,nx+1)=100;
end
while (abs(norm(u-v) 1e-6))
v=u;
for j=2:nx
for i=2:ny
u(i,j)= u(i,j)+w/4*(u(i+1,j)+u(i-1,j)+u(i,j+1)+u(i,j-1)-4*u(i,j));
end
end
end
mesh(x,y,u)
```

****

Voici les courbes répresentatives pour chaque valeur de h.

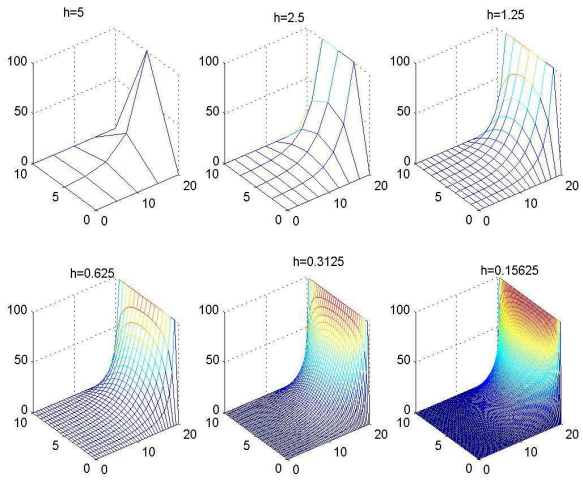


Figure 1.2: Evolution de la courbe $\Delta u = 0$ en fonction de h pour la méthode de relaxation

Conclusion

Nous remarquons que les deux méthodes convergent et tendent à former une vraie courbe spatiale quand h tend vers zero. Ce qui est en commun accord avec les approximations car si $h <<< 1$, les approximations de Taylor sont presque égales aux dérivées partielles respectives. Il faut donc tenir compte de la valeur de h et ne lui attribuer une valeur à son gré. Il est aussi à noter qu'avec la méthode de relaxation le problème de mémoire qui a été un fléau dans la méthode explicite est résolu puisqu'on est passé d'un besoin en mémoire de 14000 bytes pour la méthode directe à 6000 bytes pour la méthode de Liebmann avec $h = 1.25$. Cette même méthode de relaxation de Liebmann converge vite que la méthode directe, ceci se voit par comparaison simple des figures.
Le choix de la méthode compte aussi plus pour mieux mener la résolution numérique d'une équation aux dérivées partielles.

Chapitre 2

Résolution d'une EDP parabolique

Enoncé:

Soit à résoudre l'équation de la chaleur $\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}$ avec $\alpha \in \mathbb{R}$ fixe. Considérons le système:

$$\begin{cases} \frac{\partial u}{\partial t} = \frac{\kappa}{c\rho} \frac{\partial^2 u}{\partial x^2} \\ u(x,0) = \begin{cases} 100x \text{ si } x \in [0,1] \\ 100(2-x) \text{ si } x \in [1,2] \end{cases} \\ u(0,t) = u(2,t) = 0 \end{cases}$$

Prendre $\kappa = 0.13$, $c = 0.11$, $\rho = 7.8g/cm^3$, $\Delta x = 0.25$. Δt sera donné par les conditions de stabilité définies par l'inéquation: $r = \frac{\kappa}{c\rho} \frac{\Delta t}{\Delta x^2} < \frac{1}{2}$. La solution analytique est donnée par:

$$u_{exact}(x,t) = 800 \times \sum_{n=0}^{\infty} \frac{1}{\pi^2(2n+1)^2} \times \cos \frac{\pi(2n+1)(x-1)}{2} \times e^{-0.3738(2n+1)^2 t}$$

1. Utiliser la méthode directe (méthode explicite) en supposant $r > \frac{1}{2}$ puis $r < \frac{1}{2}$. Que constatez-vous?
2. Utiliser maintenant la méthode de Crank- Nicolson et faites varier le r. Que constatez-vous?
3. Comparaisons des méthodes:
   (a) Comparer, en fonction de r, graphiquement $u_{exact}(0.25,t)$ et $u_{app}(0.25,t)$ puis $u_{exact}(0.75,t)$ et $u_{app}(0.75,t)$
   (b) Même question pour $u_{exact}(x,0.99)$ et $u_{app}(x,0.99)$ puis $u_{exact}(x,1.98)$ et $u_{app}(x,1.98)$
4. Conclure.

1. Discrétisation de l'espace et du temps:

1.1. Discrétisation de l'espace:

Nous devons résoudre notre équation aux dérivées partielles dans l'espace compris entre 0 et 2 avec un pas Δx. Pour passer donc trouver le point x_i à partir du point x_{i-1}, nous devons ajouter à ce dernier le terme Δx, de sorte que nous avons:

$$\Rightarrow \begin{cases} x_i = x_{i-1} + \Delta x \\ x_{i-1} = x_{i-2} + \Delta x \\ \vdots \\ x_2 = x_1 + \Delta x \\ x_1 = x_0 + \Delta x \end{cases}$$

En sommant membres à membres les équations ainsi obtenues et après simplifications, nous trouvons:
$x_i = x_0 + (i - 1 + 1)\Delta x \ = \ x_0 + i\Delta x$ or x_0 n'est autre que le point a de l'intervalle $[a, b]$, donc, finalement:

$$x_i = a + i\Delta x, \ 1 \leq i \leq n_x$$

avec $n_x = \frac{b-a}{\Delta x} = \frac{2}{\Delta x}$

1.2. Discrétisation du temps:

Notre équation, c'est à dire le procéssus de résolution doit se refaire dans un intervalle de temps égal à Δt jusqu'à arriver à T_{max}. Pour passer de t_j à t_{j+1} nous devons à chanque fois ajouter Δt à t_j. Par un raisonnement par récurence comme précédemment, nous tirons: $t_j = j\Delta t$ car $t_0 = 0$ et correspond à l'instan $t = 0$.
$t_j = j\Delta t, \ 1 \leq j \leq n_t$ avec $n_t = \frac{T_{max}}{\Delta t}$

2. Programmation de la solution analytique:

Soit:

$$u_{exact}(x, t) = 800 \times \sum_{n=0}^{\infty} \frac{1}{\pi^2(2n+1)^2} \times \cos \frac{\pi(2n+1)(x-1)}{2} \times e^{-0.3738(2n+1)^2 t}$$

Puisque nous sommes en analyse numérique et non en analyse fondamentale nous devons savoir que nous ne travaillons qu'avec des points d'un maillage et non dans tout le domaine comme on aurait cru le faire. Pour évaluer la valeur numérique de u_{exact} pour tout point appartenant dans la grille, nous devons faire une boucle qui, pour chaque point (x_i, t_j), calcule $u_{exact}(x_i, t_j)$.
Posons $\infty = 100$ acceptable en analyse numérique, nous avons donc le programme suivant qui calcule les éléments de la matrice v représentant éxactement les valeurs de $u(x_i, t_j)$ dans le cadrillage.

****
clc; clear;
k=0.13;c=0.11; p=7.8;dx=0.25;
r=1/4
*dt=dx*dx*c*p*r/k;*
*Tmax=100*dt;%%(Nous avons fait Tmax un multiple de dt pour qu'en le divisant par dt, nt soit toujours un entier, car sinon le programme ne compilera pas)%*
cla=0; clb=0;
a=0;b=2;
nx=(b-a)/dx;
nt=Tmax/dt;
x=0:dx:b; t=0:dt:Tmax;
*% ********** la solution analytique **********
v=zeros(nx+1,nt+1);*
n=0;
while($n \leq 100$*)*
for i=1:nx+1
for j=1:nt+1
$u(i,j) = v(i,j) + 800 * 1/(pi^2 * (2*n+1)^2) * cos(pi * (2*n+1) * (x(i)-1)/2) * exp(-0.3738 * (2*n+1)^2 * t(j))$*;*
v(i,j)=u(i,j);
end
end
n=n+1;
end
mesh(t,x,v)
****

Voici la courbe que nous avons obténue pour $r = \frac{1}{4}$:

3. Discrétisation de l'équation différentielle:

3.1. Rappel de la notation spatiale et temporelle:

Si la fonction u prend comme variables le temps t_j et de l'esapce x_i, $y_j, \cdots$, par commodité du langage, on notera $u(x_i, t_j)$ par u_i^j et en dimension $2D$ de l'espace et $1D$ en temps, $u(x_i, y_j, t_k)$ sera notée par $u_{i,j}^k$

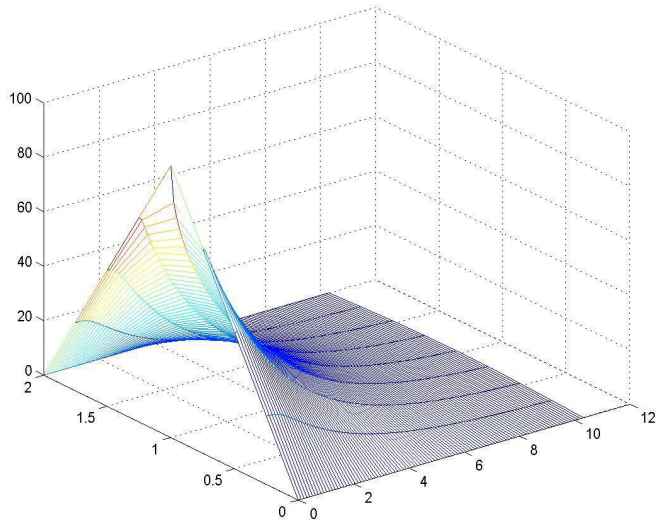


Figure 2.1: Représentation graphique de la solution analytique $u_{exact}(x_i, t_j)$

3.2. Discrétisation des dérivées partielles:

D'après les approximations vues dans le projet 1, pargraphe 1.1.1, nous pouvons approximer $\frac{\partial^2 u}{\partial x^2}$ par:

$$\frac{\partial^2 u}{\partial x^2}(x_i, t_j) \approx \frac{u_{i+1}^j - 2u_i^j + u_{i+1}^j}{(\Delta x)^2} \tag{3.1}$$

Puisque $\frac{\partial u}{\partial t}$ est une dérivée simple c'est à dire une dérivée prémière, nous utilisons ici l'une des trois formules vues en projet 1, paragraphe 1.1.1. Prenons la formule avale (droite):

$$\frac{\partial u}{\partial t}(x_i, t_j) \approx \frac{u_i^{j+1} - u_i^j}{\Delta t} \tag{3.2}$$

Nous avons remplacé h_x par Δx et h_t par Δt.

3.3. Equation aux dérivées partielles discrètisée:

Remplaons 3.1 et 3.2 dans l'EDP donnée en énoncé. On aura: $\frac{u_i^{j+1} - u_i^j}{\Delta t} \approx \frac{\kappa}{c\rho} \times \frac{u_{i+1}^j - 2u_i^j + u_{i-1}^j}{(\Delta x)^2}$
$\Rightarrow u_i^{j+1} - u_i^j = \frac{\Delta t}{(\Delta x)^2} \times \frac{\kappa}{c\rho} \times (u_{i+1}^j - 2u_i^j + u_{i-1}^j)$
Posons $r = \frac{\Delta t}{(\Delta x)^2} \times \frac{\kappa}{c\rho}$, l'équation devient:

$$u_i^{j+1} = ru_{i+1}^j + (1 - 2r)u_i^j + ru_{i-1}^j$$

Ceci est l'équation discrétisée de l'équation de la chaleur avec la méthode des différences finies. On l'appelle *équation de la méthode directe*.

Remark 4. *:*

1. r *regroupe les constantes physiques du problème et celles de la discrétisation de l'espace* $[a,b]$ *et du temps* $[0,T_{max}]$*. Donc les pas de discrétisations influencent l'équation discrétisée.*

2. *Pour* $j=0$*,* $t=0$ *et* $u_i^1 = ru_{i+1}^0+(1-2r)u_i^0+ru_{i-1}^0$ $\Rightarrow$ *l'équation nécéssite la connaissance de la condition initiale pour démarrer le procéssus.Cette condition n'est autre que* $f(x,0) = f(x_i)$ *donnée par l'énoncé du problème.*

3.4. Résolution numérique du système:

Quand on fixe j et on varie i de 1 à n_x-1, on obtient le système linéaire de n_x-1 équations à n_x-1 inconnues suivant:

$$\begin{cases} u_1^{j+1} = ru_2^j + (1-2r)u_1^j + ru_0^j \\ u_2^{j+1} = ru_3^j + (1-2r)u_2^j + ru_1^j \\ \vdots \\ \\ \vdots \\ u_{n_x-1}^{j+1} = ru_{n_x}^j + (1-2r)u_{n_x-1}^j + ru_{n_x-2}^j \end{cases} \iff \begin{cases} U^{j+1} = M\times U^j + N \\ 0 \le j \le n_t - 1 \end{cases}$$

Avec: u_0^j la condition aux limites en a, on la notera dans la programmation $u_0^j = cla$ et $u_{n_x}^j$ la condition aux limites en b qui sera notée $u_{n_x}^j = clb$. Nous avons donc:

$$M = \begin{pmatrix} 1-2r & r & 0 & \cdots\cdots & \cdots & 0 \\ r & 1-2r & r & \ddots & & \vdots \\ 0 & r & 1-2r & r & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & r & 1-2r & r \\ 0 & \cdots & \cdots & 0 & r & 1-2r \end{pmatrix} \text{ et } N = \begin{pmatrix} rU_0^j \\ 0 \\ \vdots \\ \vdots \\ \vdots \\ 0 \\ rU_{n_x}^j \end{pmatrix}$$

La matrice M est une matrice *tridiagonale* ou *trigonale* et la résolution est d'autant plus simple à programmer avec une des méthodes itératives ou directes vues en analyse numérique II. Mais, encore une fois, nous allons nous suffire de la résolution directe par la division matricielle. C'est à dire si on a à résoudre le système $Au = B$ alors on écrira en Matlab $x = A \setminus B$ et nous aurons automatiquement la réponse.
***Attention*:** B doit être un vecteur colonne de même nombre d'éléments que le nombre des lignes de A et le vecteur $\vec{u}$ sera donné en un vecteur colonne. Il ne nous est pas nouveau de donner la transposée d'une matrice pour avoir tout ce que nous voulons.

Nous avons pris $r = \frac{1}{4}$ pour le cas où $r < \frac{1}{2}$ et $r = 0.625$ pour le cas où $r > \frac{1}{2}$, on a le programme suivant qui nous calcule les valeurs du vecteur $\vec{u}$, remplacé par le vecteur $\vec{h}$ dans le programme pour pouvoir lier le programme avec le premier calculant les éléments de la matrice v pour la solution éxacte, à chaque instant j et qui les stocke dans une matrice w, en commanant par les valeurs initiales qui ne sont d'autres que les conditions initiales. Pour avoir la courbe du deuxième cas, il suffit de remplacer r par 0.625

****

```
clc; clear;
k=0.13;c=0.11;
p=7.8;dx=0.25;
r=1/4; dt=dx*dx*c*p*r/k;
Tmax=100*dt;
a=0;b=2;
cla=0;clb=0;
nx=(b-a)/dx;
nt=Tmax/dt;
x=0:dx:b; t=0:dt:Tmax;
for i=1:nx-1
N(i)=0;
end
N(1)=r*cla;
N(nx-1)=r*clb;
for i=1:nx-2
M(i,i)=1-2*r;
M(i,i+1)=r;
M(i+1,i)=r;
end
M(nx-1,nx-1)=1-2*r;
for i=1:nx+1
if x(i) < 1
Ci(i)=100*x(i);
else
Ci(i)=100*(2-x(i));
end
end
for i=1:nx-1
h(i)=Ci(i+1);
end
j=1;
```

```
h=h';
while(j < nt + 2)
for i=1:nx-1
w(i,j)=h(i);
end
h=M*h+N';
j=j+1;
end
for i=nx:-1:2
for j=nt+1:-1:1
w(i,j)=w(i-1,j);
end
end
for j=1:nt+1
w(1,j)=0;
w(nx+1,j)=0;
end
mesh(t,x,w);
**********
```

Voici les courbes obtenues après compilation pour les deux cas:

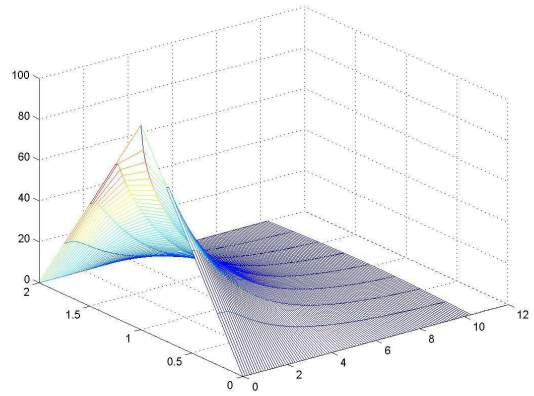


Figure 2.2: Courbe de la solution numérique, méthode explicite pour $r < \frac{1}{2}$

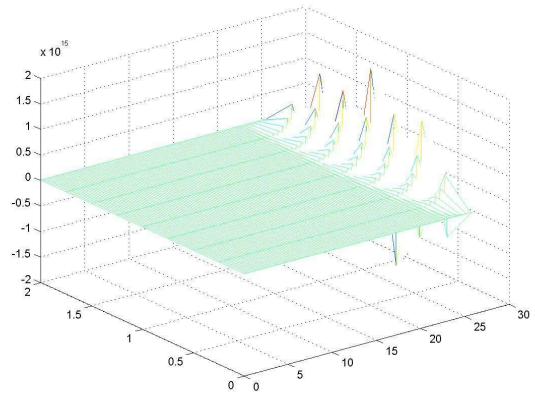


Figure 2.3: Courbe de la solution numérique, méthode explicite pour $r > \frac{1}{2}$

Constatations:

Nous avons rémarqué, d'après les deux graphes corresondantes à $r < \frac{1}{2}$ et $r > \frac{1}{2}$, que la méthode présente deux régions de stabilité. Si $r < \frac{1}{2}$ la méthode est stable et converge vers la solution

analytique. Nous avons une courbe semblable à celle de la solution analytiquement établie. Plus j s'accroit, plus les valeurs du vecteur $\vec{u}$ deviennent nulles. Ce qui est en entier accord avec la solution éxacte qui est une série des fonctions convergente, alors son terme général doit tendre forcement vers 0.
Mais si $r > \frac{1}{2}$, la méthode présente des failles et la courbe ne converge même pas. L'allure obténue dans ce cas est totalement en désaccord avec celle de la solution éxacte.
Pour avoir un bon résultat dans cette méthode, il faut donc ténir compte de la valeur de r car elle influencera le résultat.

4. Méthode de Crank - Nicolson ou méthode implicite:

En mathématiques, en analyse numérique, la méthode de **Crank-Nicolson** est un algorithme simple permettant de résoudre des systèmes d'équations aux dérivées partielles. Cette méthode utilise les différences finies pour approcher une solution du problème : elle est numériquement stable, et quadratique pour le temps. On peut facilement la généraliser à des problèmes à deux ou trois dimensions.
Cette méthode, publiée en 1947, est le résultat des travaux de la mathématicienne britannique **Phyllis Nicolson**$(1917-1968)$ et du physicien **John Crank** $(1916-2006)$. Ils l'utilisèrent dans la résolution de l'équation de la chaleur. Mais, peu après la méthode est devenue usuelle dans plusieurs problèmes numériques.

4.1. Méthode:

Pour assurer une résolution numérique stable et ce dans toute la région de l'étude, Crank et Nicolson ont proposé de prendre la moyenne des derivées partielles spatiales de u entre les instants t_j et t_{j+1} au lieu de considérer seulement la dérivée spatiale à l'instant t_j, c'est à dire:

$$\frac{\partial u}{\partial t}(x_i,t_j)=\frac{\alpha}{2}\left[\frac{\partial^2 u}{\partial x^2}(x_i,t_j)+\frac{\partial^2 u}{\partial x^2}(x_i,t_{j+1})\right]$$

4.2. Résolution numérique de l'EDP

Nous remplaons les dérivées partielles par leurs approximations déjà établies pour avoir l'équation numérique suivante:
$\frac{1}{2}\left(\frac{u_{i+1}^j-2u_i^j+u_{i-1}^j}{(\Delta x)^2}+\frac{u_{i+1}^{j+1}-2u_i^{j+1}+u_{i-1}^{j+1}}{(\Delta x)^2}\right)=\frac{c\rho}{\kappa}\times\frac{u_i^{j+1}-u_i^j}{\Delta t}$
$\Rightarrow 2\left(u_i^{j+1}-u_i^j\right)=\frac{\Delta t}{(\Delta x)^2}\frac{\kappa}{c\rho}\left(u_{i+1}^j-2u_i^j+u_{i-1}^j+u_{i+1}^{j+1}-2u_i^{j+1}+u_{i+11}^{j+1}\right)$.
Posons $r=\frac{\Delta t}{(\Delta x)^2}\frac{\kappa}{c\rho}$, on trouve l'équation:

$$-ru_{i-1}^{j+1}+(2+2r)u_i^{j+1}-ru_{i+1}^{j+1}=ru_{i-1}^j+(2-2r)u_i^j+ru_{i+1}^j$$

Ceci est le **schéma numérique** de Crank - Nicolson.
Lorsque nous fixons j et faisons varier i, nous obténons le système linéaire suivant:

$$\begin{cases} -ru_0^{j+1} + (2+2r)u_1^{j+1} - ru_2^{j+1} = ru_0^j + (2-2r)u_1^j + ru_2^j \\ -ru_1^{j+1} + (2+2r)u_2^{j+1} - ru_3^{j+1} = ru_1^j + (2-2r)u_i^2 + ru_3^j \\ -ru_2^{j+1} + (2+2r)u_3^{j+1} - ru_4^{j+1} = ru_2^j + (2-2r)u_3^j + ru_4^j \\ \vdots \\ \vdots \\ -ru_{n_x-2}^{j+1} + (2+2r)u_{n_x-1}^{j+1} - ru_{n_x}^{j+1} = ru_{n_x-1}^j + (2-2r)u_{n_x}^j + ru_{n_x}^j \end{cases}$$

$$\Longrightarrow M_1 U^{j+1} + N_1 = M_2 U^j + N_2$$

Avec:

$$M_1 = \begin{pmatrix} 2+2r & -r & 0 & \cdots\cdots & \cdots & 0 \\ -r & 2+2r & -r & \ddots & & \vdots \\ 0 & -r & 2+2r & -r & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & -r & 2+2r & -r \\ 0 & \cdots & \cdots & 0 & -r & 2+2r \end{pmatrix}, \quad N_1 = \begin{pmatrix} -rU_0^{j+1} \\ 0 \\ \vdots \\ \vdots \\ \vdots \\ 0 \\ -rU_{n_x}^{j+1} \end{pmatrix}$$

$$M_2 = \begin{pmatrix} 2-2r & r & 0 & \cdots\cdots & \cdots & 0 \\ r & 2-2r & r & \ddots & & \vdots \\ 0 & r & 2-2r & r & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & r & 2-2r & r \\ 0 & \cdots & \cdots & 0 & r & 2-2r \end{pmatrix}, \quad N_2 = \begin{pmatrix} rU_0^j \\ 0 \\ \vdots \\ \vdots \\ \vdots \\ 0 \\ rU_{n_x}^j \end{pmatrix}$$

$$\Longrightarrow U^{j+1} = M_1^{-1}[M_2 U^j + (N_2 - N_1)]$$

Ce système matriciel est linéaire d'ordre $n_x - 1$.

4.3. Programmation de la solution

Voici le programme qui sert à donner la courbe de la solution numérique de la méthode de Crank - Nicolson. Pour avoir la solution (la courbe) pour le deuxième cas, il suffit de remplacer r par 0.25:

****

clc; clear;
k=0.13;c=0.11;

```
p=7.8;dx=0.25;
r=0.625;
dt=dx*dx*c*p*r/k;
Tmax=100*dt;
a=0;b=2;
cla=0;clb=0;
nx=(b-a)/dx; nt=Tmax/dt;
x=0:dx:b; t=0:dt:Tmax;
for i=1:nx-1
N1(i)=0;
N2(i)=0;
end
N2(1)=-r*cla;
N2(nx-1)=-r*clb;
for i=1:nx-2
M1(i,i)=2+2*r;
M1(i,i+1)=-r;
M1(i+1,i)=-r;
M2(i,i)=2-2*r;
M2(i,i+1)=r;
M2(i+1,i)=r;
end
M1(nx-1,nx-1)=2+2*r;
M2(nx-1,nx-1)=2-2*r;
for i=1:nx+1
if x(i)<1
Ci(i)=100*x(i);
else
Ci(i)=100*(2-x(i));
end
end
for i=1:nx-1
h(i)=Ci(i+1);
end
j=1;
h=h';
I=eye(nx-1)
while(j<nt+2)
for i=1:nx-1
w(i,j)=h(i);
end
h=(M1\I)*(M2*h+(N2'-N1'));
```

```
j=j+1;
end
for i=nx:-1:2
for j=nt+1:-1:1
w(i,j)=w(i-1,j);
end
end
for j=1:nt+1
w(1,j)=0;
w(nx+1,j)=0;
end
mesh(t,x,w);
*****************************************
```

Voici les courbes répresentatives de w dans les deux situations:

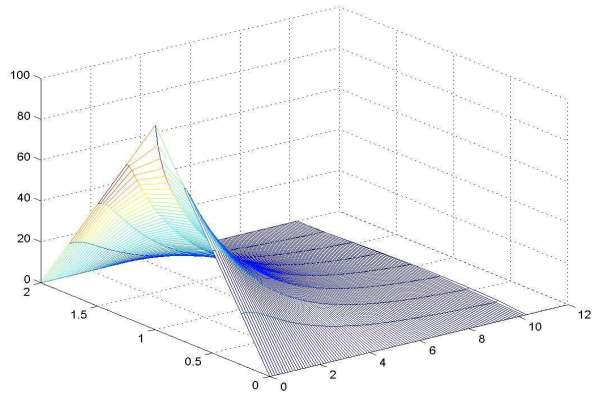


Figure 2.4: Courbe de la solution numérique, méthode implicite pour $r < \frac{1}{2}$

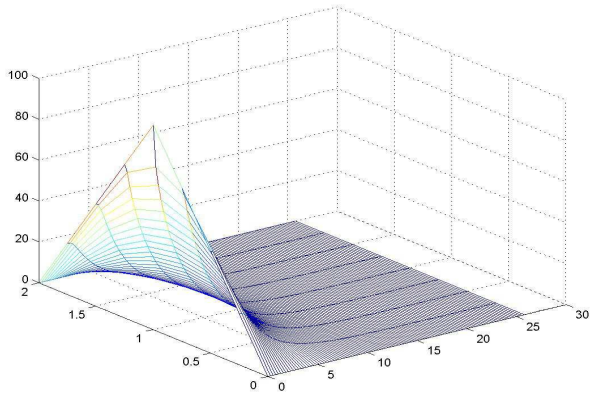


Figure 2.5: Courbe de la solution numérique, méthode implicite pour $r > \frac{1}{2}$

Constatations

Nous remarquons ici que la méthode de Crank - Nicolson est inconditionnellement stable (c'est à dire quel que soit $r \in \mathbb{R}$). Toutefois, quand $r > \frac{1}{2}$ la méthode converge plus vite vers 0 que si c'est le cas contraire. Il faut aussi savoir qu'on paie cette stabilité par la résolution, à chaque instant t_j, d'un système linéaire qui nécéssite une inversion de la matrice M_1 tridiagonale, soit:

$$M_1 U^{j+1} + N_1 = M_2 U^j + N_2$$

$$\Longrightarrow U^{j+1} = M_1^{-1}[M_2 U^j + (N_2 - N_1)]$$

5. Comparaison des méthodes:

En fonction de r, nous voulons comparer graphiquement la méthode numérique explicite (directe) avec la solution analytique pour voir comment la solution approchée se rapproche à la solution éxacte.

5.1. Comparaison de $u_{exact}(0.25,t)$ et $u_{app}(0.25,t)$, $u_{exact}(0.75,t)$ et $u_{app}(0.75,t)$

Nous combinons le programme de la solution analytique avec celui de la solution numérique directe pour pouvoir faire la comparaison, puis on ajoute à la fin du nouveau programme, la partie suivante:
Rémarquons que $x_i = 0.25$ si $i = 2$ et $x_i = 0.75$ pour $i = 4$.

****
Sa1=v(2,:);
Sn1=w(2,:);
plot(t,Sa1,'r',t,Sn1,'g');
Sa2=v(4,:);
Sn2=w(4,:)
figure(2)
plot(t,Sa2,'r',t,Sn2,'g');
****

Voici donc les courbes comparatives de ces deux méthodes aux points 0.25 et 0.75:

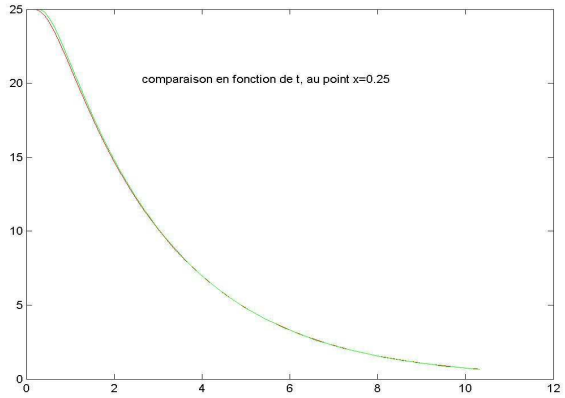


Figure 2.6: Comparaison graphique de $u_{exact}(0.25,t)$ et $u_{app}(0.25,t)$

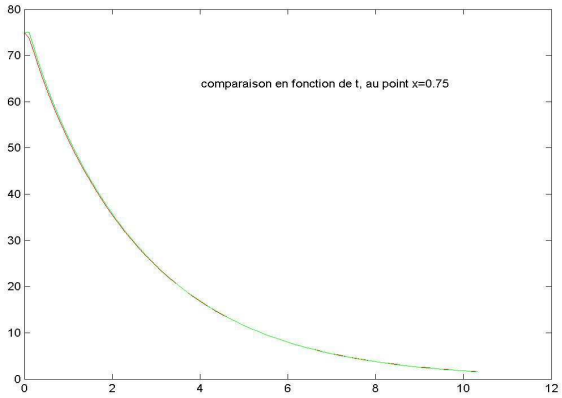


Figure 2.7: Comparaison graphique de $u_{exact}(0.75,t)$ et $u_{app}(0.75,t)$

5.2. Comparaison de $u_{exact}(x, 0.99)$ et $u_{app}(x, 0.99)$, $u_{exact}(x, 1.98)$ et $u_{app}(x, 1.98)$

Remarquons qu'avec $r = \frac{1}{4}$, on n'a pas un point j tel que $t_j = 0.99$ ni $t_j = 1.98$ mais on voit que pour $j = 10,\ t_j = 0.9281 \approx 0.99$ et pour $j = 20,\ t_j = 1.9594 \approx 1.98$. On refait ensuite la même chose mais on ajoute à présent la partie suivate à la fin de la combinaison:

```
%%*********************************
Sa1=v(:,10);
Sn1=w(:,10);
plot(t,Sa1,'r',t,Sn1,'g');
Sa2=v(:,20);
Sn2=w(:,20)
figure(2)  plot(t,Sa2,'r',t,Sn2,'g');
*********************************
```

Voici donc les courbes comparatives des solutions pour les deux méthodes à $t_j = 0.9281$ et $t_j = 1.9594$:

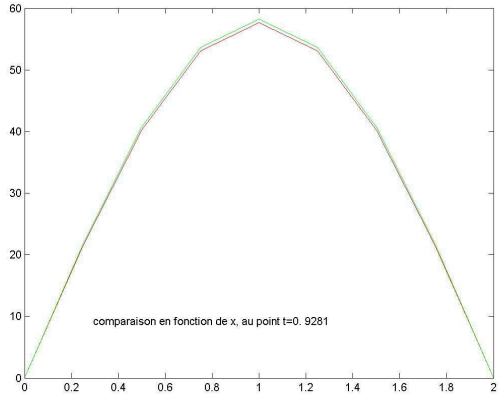


Figure 2.8: Comparaison graphique de $u_{exact}(x, 0.9281)$ et $u_{app}(x, 0.9281)$

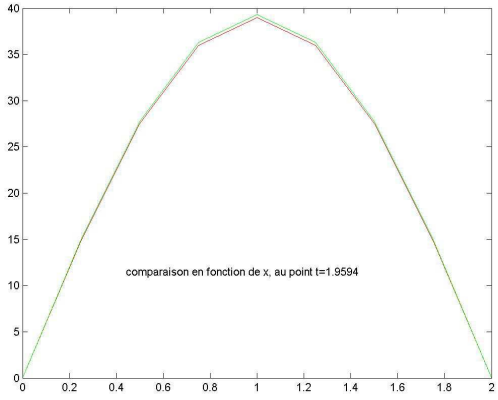


Figure 2.9: Comparaison graphique de $u_{exact}(x, 1.9594)$ et $u_{app}(x, 1.9594)$

6. Etudes théoriques de la méthode explicite

6.1. Convergence numérique

La notion de *Convergence* signifie que la solution numérique (donnée par le schema numérique) approche la solution éxacte (analytique) quand $\Delta x \to 0$ et $\Delta t \to 0$, ce qui signifie que la maille devient très petite.

Considerons u_i^j la solution numérique approchée et U_i^j la solution éxacte de l'équation:

$$\frac{\partial U}{\partial t} = \frac{k}{c\rho}\frac{\partial^2 U}{\partial x^2}$$

Au point $x = x_i$ et $t = t_j$, notons: $e_i^j = U_i^j - u_i^j$ Pour $r = \frac{k}{c\rho} \times \frac{\Delta t}{(\Delta x)^2}$, $u_i^{j+1} = r(u_{i+1}^j + u_{i-1}^j) + (1-2r)u_i^j$.
$e_i^j = U_i^j - u_i^j \Rightarrow u_i^j = U_i^j - e_i^j$ que l'on remplace dans le schema numérique précédente:

$$U_i^{j+1} - e_i^{j+1} = r(U_{i+1}^j - e_{i+1}^j + U_{i-1}^j - e_{i-1}^j) + (1-2r)(U_i^j - e_i^j)$$
$$\Rightarrow e_i^{j+1} = \left[r(e_{i+1}^j + e_{i-1}^j) + (1-2r)e_i^j\right] + U_i^{j+1} - r(U_{i+1}^j + U_{i-1}^j) - (1-2r)U_i^j.$$

Avec la méthode de Taylor, nous trouvons:

$$\begin{cases} U_i^{j+1} = U_i^j + \Delta t \times \frac{\partial U}{\partial t}(x_i, \xi),\ t_j \le \xi \le t_{j+1} \\ U_{i+1}^j = U_i^j + \Delta x \times \frac{\partial U}{\partial x}(x_i, t_j) + \frac{(\Delta x)^2}{2} \times \frac{\partial^2 U}{\partial x^2}(\eta_1, t_j),\ x_i \le \eta_1 \le x_{i+1} \\ U_{i-1}^j = U_i^j - \Delta x \times \frac{\partial U}{\partial x}(x_i, t_j) + \frac{(\Delta x)^2}{2} \times \frac{\partial^2 U}{\partial x^2}(\eta_2, t_j),\ x_{i-1} \le \eta_2 \le x_i \end{cases}$$

que l'on remplace dans l'équation précédemment trouvée pour avoir:

$$e_i^{j+1} = \left[r(e_{i+1}^j + e_{i-1}^j) + (1-2r)e_i^j\right] + U_i^j + \Delta t \times \frac{\partial U}{\partial t}(x_i, \xi) - r(U_i^j + \Delta x \times \frac{\partial U}{\partial x}(x_i, t_j)$$

$$+\frac{(\Delta x)^2}{2} \times \frac{\partial^2 U}{\partial x^2}(\eta_1, t_j)) - r(U_i^j - \Delta x \times \frac{\partial U}{\partial x}(x_i, t_j) + \frac{(\Delta x)^2}{2} \times \frac{\partial^2 U}{\partial x^2}(\eta_2, t_j)) - (1-2r)U_i^j.$$

Après simplification des quantités en couleurs, l'équation devient:

$$e_i^{j+1} = \left[r(e_{i+1}^j + e_{i-1}^j) + (1-2r)e_i^j\right] + \Delta t \times \frac{\partial U}{\partial t}(x_i, \xi) - r(\frac{(\Delta x)^2}{2} \times \frac{\partial^2 U}{\partial x^2}(\eta_1, t_j) + \frac{(\Delta x)^2}{2} \times \frac{\partial^2 U}{\partial x^2}(\eta_2, t_j))$$

Dans un maillage très fin, les points de la discrétisation sont très proches les uns des autres. Nous avons donc $x_i \approx \eta_1 \approx x_{i+1}$, $x_{i-1} \approx \eta_2 \approx x_i$ ainsi que $t_j \approx \xi \approx t_{j+1}$. Pour ce, nous allons remplacer η_1 et η_2 par x_i et ξ par t_j puis r par $\frac{k}{c\rho}\frac{\Delta t}{(\Delta x)^2}$ pour avoir l'équation sous la forme:

$$e_i^{j+1} = \left[r(e_{i+1}^j + e_{i-1}^j) + (1-2r)e_i^j\right] + \Delta t \times \frac{\partial U}{\partial t}(x_i, t_j) - \frac{k}{c\rho}\frac{\Delta t}{(\Delta x)^2}(\frac{(\Delta x)^2}{2} \times \frac{\partial^2 U}{\partial x^2}(x_i, t_j)$$

$$+\frac{(\Delta x)^2}{2} \times \frac{\partial^2 U}{\partial x^2}(x_i, t_j)).$$

Après simplification de la quatité $(\Delta x)^2$, factorisons Δt:

$$e_i^{j+1} = \left[r(e_{i+1}^j + e_{i-1}^j) + (1-2r)e_i^j\right] + \Delta t\left[\frac{\partial U}{\partial t}(x_i, t_j) - \frac{k}{c\rho}\frac{\partial^2 U}{\partial x^2}(x_i, t_j)\right].$$

Le deuxième crochet, qui n'est autre que l'équation de la chaleur initialement posée dans l'énoncé, est nul. Ce qui nous done l'égalité finale:

$$e_i^{j+1} = re_{i+1}^j + re_{i-1}^j + (1-2r)e_i^j$$

$$\Rightarrow \left|e^{j+1}\right| = \left|re_{i+1}^j + re_{i-1}^j + (1-2r)e_i^j\right|$$

Si $r < 1/2$, on a: $1 - 2r > 0 \Rightarrow |1-2r| = 1-2r$, alors:

$$\left|e^{j+1}\right| \leq r\left|e_{i+1}^j\right| + r\left|e_{i-1}^j\right| + (1-2r)\left|e_i^j\right|$$

Posons

$$E^j = \max_{1\leq i\leq n_x-1} \left|\, e_i^j \,\right|$$

$$\Rightarrow \left|e^{j+1}\right| \leq rE^j + rE^j + (1-2r)E^j \Rightarrow \left|e^{j+1}\right| \leq E^j,\ \forall\ 1 \leq j \leq n_t - 1$$

Par recurence,

$$\Rightarrow \left|e^{j+1}\right| \leq E^j \leq \cdots \leq E^1 \leq E^0$$

$E^0 = U^0 - u^0$ à $t = 0$, or, d'après les conditions initiales, $U^0 - u^0 = 0$, donc $E^0 = 0$.

$$\left|e^{j+1}\right| \leq 0 \Rightarrow e^j = 0,\ \forall j \text{ et } r < \frac{1}{2}$$

Donc, la méthode explicite converge vers zéro quand $r < 1/2$.

6.2. Stabilité numérique

On dit qu'un schema numérique est stable si et seulement si au cours des calculs, l'erreur commise d'une itération à l'autre (d'une maille à l'autre) n'infecte pas le calcul suivant. Dans le cas contraire, le calcul peut exploser et on n'aura pas une bonne convergence. soit l'équation de la chaleur:

$$\frac{\partial U}{\partial t} = \frac{k}{c\rho}\frac{\partial^2 U}{\partial x^2}$$

La numérisation de cette équation nous donne la relation matricielle:

$$U^{j+1} = AU^j \Rightarrow U^1 = AU^0$$

U^0 étant donnée par les conditions initiales.

$$\begin{cases} U^1 = AU^0 \\ U^2 = AU^1 = A^2U^0 \\ U^3 = AU^2 = A^3U^0 \\ \vdots \\ U^j = AU^{j-1} = A^jU^0 \end{cases}$$

Supposons qu'on introduit U^0 par une certaine erreur et on donne $\overline{U}^0$

$$\begin{cases} \overline{U}^1 = A\overline{U}^0 \\ \overline{U}^2 = A\overline{U}^1 = A^2\overline{U}^0 \\ \overline{U}^3 = A\overline{U}^2 = A^3\overline{U}^0 \\ \vdots \\ \overline{U}^j = A\overline{U}^{j-1} = A^j\overline{U}^0 \end{cases}$$

Avec $e^0 = \overline{U}^0 - U^0 \Rightarrow e^j = \overline{U}^j - U^j = A^j\overline{U}^0 - A^jU^0 = A^j(\overline{U}^0 - U^0)$

$$\Rightarrow e^j = A^j e^0$$

A admet N valeurs propores distinctes (Ref: cours sur les valeurs propres), donc ses vecteurs propres associés forment une basse (Cf Algèbre 1):

$$\begin{cases} AX_1 = \lambda_1 X_1 \\ AX_2 = \lambda_2 X_2 \\ AX_3 = \lambda_3 X_3 \\ \vdots \\ AX_n = \lambda_n X_n \end{cases}$$

λ_i est la valeur propore associéé au vecteur propre X_i. Il existe alors $c_1,\ c_2\ \cdots,\ c_n$ tels que: $e^0 = c_1X_1 + c_2X_2 + \cdots + c_nX_n$

$$\Rightarrow e^j = A^j(\sum_{i=1}^{N} c_iX_i) = \sum_{i=1}^{N} A^jc_iX_i = \sum_{i=1}^{N} c_iA^jX_i = \sum_{i=1}^{N} c_i\lambda_i^jX_i$$

$$e^j = \sum_{i=1}^{N} c_i\lambda_i^jX_i$$

$$\Rightarrow \left|e^j\right| = \left|\sum_{i=1}^{N} c_i\lambda_i^jX_i\right| \leq \sum_{i=1}^{j} |c_i| \times \left|\lambda_i^j\right| \times |X_i|$$

L'erreur est maîtrisable si $\forall$, $|\lambda_i| < 1$. Ceci est vérifié si et seulement si:

$$\max_{1\leq i\leq N} |\lambda_i| < 1$$

Pour la matrice A de cet exemple, les valeurs propres sont données par (voir le cours sur les valeurs et vecteurs propres):

$$\lambda_i = 1 - 4r\sin^2\frac{i\pi}{2(N+1)},\ i = 1, \cdots,\ N$$

Alors:

$$|\lambda_i| < 1 \Rightarrow |1 - 4r\frac{\sin^2 i\pi}{2(N+1)}| < 1, \text{ avec } r = \frac{k}{c\rho} \times \frac{\Delta t}{\Delta x}$$

$$\Rightarrow -1 < 1 - 4r\frac{\sin^2 i\pi}{2(N+1)} < 1$$

pour:

$$-1 < 1 - 4r\frac{\sin^2 i\pi}{2(N+1)} \Rightarrow r < \frac{1}{2} \times \left[\frac{\sin^2 i\pi}{2(N+1)}\right]^{-1}, \forall\, i = 1, 2, \cdots, N$$

$$\Rightarrow r < \frac{1}{2} \times \min_{i=1,\cdots,N} \left[\frac{\sin^2 i\pi}{2(N+1)}\right]^{-1}$$

Or $\forall x,\ \sin^2 x \leq 1 \Rightarrow \frac{\sin^2 x}{a} \leq 1,\ \forall a > 1 \Rightarrow \left[\frac{\sin^2 x}{a}\right]^{-1} \geq 1$, donc:

$$\min_{i=1,\cdots,N} \left[\frac{\sin^2 i\pi}{2(N+1)}\right]^{-1} \leq 1 \Rightarrow r < \frac{1}{2}$$

On tire encore une fois que la méthode directe (explicite) est stable si $r < \frac{1}{2}$. Ce qui est en parfait commun accord avec l'expérience réalisée.

Conclusion

Il est intéressant de remarquer que l'équation de la chaleur, introduite initialement pour décrire la conduction thermique, apparaît également dans d'autres branches de la physique théorique. Elle permet par exemple de décrire :

1. Le phénomène de diffusion ;
2. Certains aspects probabilistes du mouvement brownien

Les méthodes de résolution de cette équation sont multiples et diversifiées. Toutefois, il faut faire attention à la stabilité et la convergence de la méthode choisie car, simple et facile à manipuler qu'elle soit, elle pourrait causer des instabilités inattendues et si on n'a pas la solution analytique pour faire la comparaison, des confusions pourraient s'engendrer. Il serait sage de payer le cot de résolution et avoir une méthode stable et convergente inconditionnellement que de se contenter d'une résolution aléatoire (valable pour quelques valeurs particulières) vue qu'elle soit maniable facilement.

Chapitre 3

Résolution d'une EDP hyperbolique

1. Enoncé:

Soit à résoudre une équation hyperbolique donnée par:

$$A\frac{\partial^2 u}{\partial t^2} + B\frac{\partial^2 u}{\partial x \partial t} + C\frac{\partial^2 u}{\partial x^2} + E = 0 \tag{1.1}$$

1.1. Résolution par la méthode des différences finies:

Schéma numérique simplifié

Posons $a = c_1$ (la célérité), $b = 0$, $c = -1$ et $e = 0$, ce qui nous donne:
$\Delta = b^2 - 4ac = 0 - 4(c_1)(-1) = 4c_1 > 0$ car $c_1 > 0$. C'est bien donc une équation hyperbolique qui n'est autre que l'équation d'ondes:

$$\frac{\partial^2 u}{\partial t^2} - c_1\frac{\partial^2 u}{\partial x^2} = 0.$$

Comme les deux dérivées sont des dérivées secondes, nous utilisons les approximations de Taylor trouvées dans les équations 1.1 pour établir le schéma numérique de cette EDP. Nous avons donc:

$$\frac{u_i^{j+1} - 2u_i^j + u_i^{j-1}}{(\Delta t)^2} = c_1\frac{u_{i+1}^j - 2u_i^j + u_{i-1}^j}{(\Delta x)^2}$$

$$\Rightarrow u_i^{j+1} = c_1\frac{(\Delta t)^2}{(\Delta x)^2}(u_{i+1}^j + u_{i-1}^j) - 2(1 - c_1\frac{(\Delta t)^2}{(\Delta x)^2})u_i^j - u_i^{j-1}.$$

Ceci est l'équation numérique de l'équation d'ondes. Pour simplifier le schéma, posons:

$$c_1\frac{(\Delta t)^2}{(\Delta x)^2} = 1 \ \Rightarrow \Delta t = \frac{\Delta x}{\sqrt{c_1}}.$$

$$\Rightarrow u_i^{j+1} = u_{i+1}^j + u_{i-1}^j - u_i^{j-1}. \tag{1.2}$$

Remark 5. *Avec les différences finies, l'équation* 1.2 *est le schéma numérique utilisée pour la résolution de l'équation d'ondes* $\frac{\partial^2 u}{\partial t^2} = c_1 \frac{\partial^2 u}{\partial x^2}$.
Mais, il se pose un problème de mise en oeuvre de ce schéma vu que u *est connu à* $t = t_0 = 0$ *qui est la condition initiale. Or, pour calculer* u *à* $t = \Delta t = t_1$*, nous devons connaitre la valeur de* u *à* $t = t_{-1} = -\Delta t$.

Conditions mixtes:

Connaître les valeurs de u à $t = -\Delta t$ n'est plus un problème quand nous avons des conditions de Newmann, de type dérivée (une condition sur la vitèsse de propagation): $\frac{\partial u}{\partial t}(x, 0) = g(x),$ à $t = 0$

$$\frac{\partial u(x_i, 0)}{\partial t} = \frac{u(x_i, \Delta t) - u(x_i, -\Delta t)}{2\Delta t} = g(x)$$

$$\Rightarrow \frac{u_i^1 - u_i^{-1}}{2\Delta t} = g(x_i) \ \Rightarrow u_i^{-1} = u_i^1 - 2\Delta t \times g(x_i).$$

Remplaons-le alors dans l'équation 1.2 il vient:

$$u_i^1 = u_{i+1}^0 + u_{i-1}^0 - u_i^1 + 2\Delta t \times g(x_i) \Rightarrow u_i^1 = \frac{1}{2}(u_{i+1}^0 + u_{i-1}^0) + \Delta t \times g(x_i).$$

Limite de la méthode de différences finies

C'est avec une condition sur la vitèsse, une approximation de quelques variables comme $b = 0$ et $e = 0$, une simplification de l'expression $c_1 \frac{\Delta t}{\Delta x}$ par 1, que nous avons pu résoudre cette équation hyperbolique. Toutefois et malheureusement, ce n'est pas tout le temps pareil et la donnée d'une telle condition de type Newmann n'est pas usuelle. On sera donc souvent bloqué sur la résolution d'une éqaution hyperbolique si nous nous inspirons seulement de la méthode des différences finies.

2. Méthode des caractéristiques:

Il existe une classe d'équations aux derivées partielles dont on peut obtenir les solutions en utilisant une méthode du type géometrique, **la méthode des caractéristique**.
C'est une des plus grandes méthodes énumerées en introduction utilisées pour la résolution des équations différentielles partielles.
Cette méthode des caractéristiques est une technique qui est particulièrement adaptée aux problèmes de transport, elle est utilisée dans de nombreux domaines tels que la mécanique des fluides ou encore le transport de particules.
Dans certains cas particuliers, cette méthode peut permettre la résolution purement analytique de l'EDP. Dans des cas plus complexes comme en modélisation des systèmes physiques, la méthode des caractéristiques peut être utilisée comme une méthode de résolution numérique du problème.
la méthode de résolution que nous présentons s'applique aux équations du premier et deuxième

ordres avec un nombre des variables supérieur à deux, mais pour la clarté du rapport nous nous limitons à une équation aux derivées partielles de deux variables.

2.1. Pourquoi la méthode des caractéristiques?

la méthode des caractéristique est simple à appliquer, et démande peu de calculs et le résultat obtenu est rémarquablement précis, contrairement à ceux des methodes classiques des différences finies et de Galerkin.

2.2. Principe

En mathématique, la méthode des caractéristique est une technique pour résoudre les equations aux derivées partielles, plus genéralement la méthode des caractéristiques est valable pour toutes les équations les équations aux derivées partielles hyperboliques.
La méthode consiste à réduire une équation aux dérivées partielles à une famille d'équations differntielles ordinaires, le long de laquelle la solution peut être integrée à partir des données initiales .
Elle cherche des courbes appelées les *courbes caractéristiques* ou tout simplement *les caractéristiques* le long desquelles l'équation aux dérivées partielles se réduit en une équation simple à résoudre.
La résolution de cette simple équation sur les caractéristiques nous permet de retrouver la solution globale du problème originale sur tout le maillage. comment pouvons-nous trouver les courbes caractéristiques ?

2.3. Méthode

En posant $\frac{\partial^2 u}{\partial x^2} = u_{xx}$, $\frac{\partial^2 u}{\partial t^2} = u_{tt}$ et $\frac{\partial^2 u}{\partial x \partial t} = u_{xt}$ et en considérant que $u_{xt} = u_{tx}$, l'équation 1.1 prend la forme:

$$au_{xx} + bu_{xt} + cu_{tt} + e = 0. \tag{2.1}$$

Où a, b, c, e sont des fonctions ne dependant pas de u et non toutes nulles. Posons:

$$p = \frac{\partial u}{\partial x} = u_x, \ et \ q = \frac{\partial u}{\partial t} = u_t.$$

$$\Longrightarrow dp = \frac{\partial p}{\partial x}dx + \frac{\partial p}{\partial t}dt = u_{xx}dx + u_{xt}dt. \tag{2.2}$$

$$\text{et } dq = \frac{\partial q}{\partial x}dx + \frac{\partial q}{\partial t}dt = u_{tx}dx + u_{tt}dt. \tag{2.3}$$

Ce qui nous donne, après addition et soustraction:

$$(2.2) \ \Rightarrow \ u_{xx} = \frac{dp}{dx} - u_{xt}\frac{dt}{dx},$$

$$(2.3) \Rightarrow u_{tt} = \frac{dq}{dt} - u_{xt}\frac{dx}{dt}.$$

Substituons ces égalités dans l'équations 2.1 il vient:

$$\Rightarrow -au_{xt}\frac{dt}{dx} + bu_{xt} - cu_{xt}\frac{dx}{dt} + a\frac{dp}{dx} + c\frac{dq}{dt} + e = 0.$$

En multipliant cette équation par $-\frac{dt}{dx}$, elle devient:

$$u_{xt}\left[a(\frac{dt}{dx})^2 - b(\frac{dt}{dx}) + c\right] - \left[a\frac{dp}{dx} \times \frac{dt}{dx} + c\frac{dq}{dx} + e\frac{dt}{dx}\right] = 0. \tag{2.4}$$

Grce aux transformations précédentes, la résolution de l'équation 2.1 revient à ne résoudre 2.4.
On suppose que, dans le plan x, t, on définit les courbes de sorte que le premier crochet soit **nul**. Sur ces courbes 2.1, il est équivalent à prendre le deuxième crochet aussi égal à 0.
Posons $m = \frac{dt}{dx}$, donc si $am^2 - bm + c = 0$, alors la solution de l'EDP 2.1 peut-être trouvée en résolvant:

$$amdp + cdq + edt = 0.$$

Les courbes données par $am^2 - bm + c = 0$ sont appelées *les caractéristiques* de l'équation 2.1.
On se limite dans le cas où $b^2 - 4ac > 0$, ce qui fait donc que l'équation susmentionnée admet deux racines distinctes. Chaque point possédera deux courbes caractéristiques dont les pentes sont les racines de l'équation précédente.

Comment résoudre une EDP hyperbolique par la méthode des caractéristiques ?.

Stratégie générale

Nous allons maintenant donner les grandes lignes pour résoudre l'EDP 2.1 par une intégration numérique le long des caractéristiques. On considère deux points P et Q (voir figure1), l'équation aux dérivées partielles 2.1 étant hyperbolique, il y a deux caractéristiques en chaque point. La courbe caractéristique à droite de P, de pente m_+, intersecte celle de gauche de Q, de pente m_- au point R.
La solution du problème 2.1 peut-être trouvée en résolvant $amdp + cdq + edt = 0$ le long de ces caractéristiques.

Exemple

Soit à résoudre l'équation:

$$u_{tt} = c_1^2 u_{xx}$$

On a: $a = -c_1^2$, $b = e = 0$ et $c = 1 \Rightarrow b^2 - 4ac = 4c_1^2 > 0$.

$am^2 - bm + c = -c_1^2 + 1 = 0 \Rightarrow m = \pm\frac{1}{c_1} \Rightarrow m_1 = \frac{1}{c_1}$ et $m_2 = -\frac{1}{c_1}$.

On a posé $m = \frac{dt}{dx} = \pm\frac{1}{c_1} \Rightarrow dt = \pm\frac{1}{c_1}dx$

$$\Longrightarrow \Delta t = \pm\frac{1}{c_1}\Delta x$$

à $t = \Delta t$, on a alors $t - 0 = \Delta t = \pm\frac{1}{c_1}(x - x_i)$

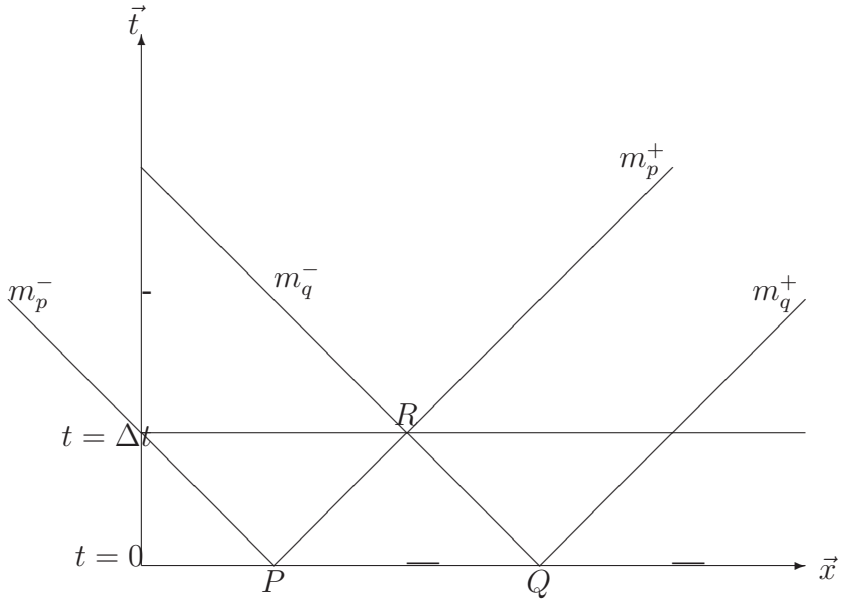


Figure 3.1: Représentation graphique des caractéristiques

La premiere étape sera de trouver les coordonnées du point R, pour cela, nous résolvons l'équation: $\Delta t = (\frac{dt}{dx})_{av} \quad \Delta x = m_{av}\Delta x$ sur les arcs PR et QR.
Ecrivons l'équation $amdp + cdq + edt = 0$ sous la forme:

$$a_{av}m_{av}\Delta p + c_{av}\Delta q + e_{av}\Delta t = 0. \tag{2.5}$$

Commenons par le point P et puis par Q en utilisant les valeurs de m appropriées. Ceci va estimer p et q au point R.
Finalement, on évaluera u en R de $du = \frac{\partial u}{\partial x}dx + \frac{\partial u}{\partial t}dt$, en utilisant la forme:

$$\Delta u = p_{av}\Delta x + q_{av}\Delta t. \tag{2.6}$$

3. Exercices d'application

3.1. Exercice 1:

Soit à résoudre:

$$\begin{cases} \frac{\partial^2 u}{\partial t^2} = 2\frac{\partial^2 u}{\partial x^2} - 4 \\ \\ \frac{\partial u}{\partial t} = 0,\ 0 \leq x \leq 1 \\ \\ u(x,0) = \begin{cases} 12x,\ 0 \leq x \leq 0.25 \\ 4 - 4x,\ 0.25 \leq x \leq 1 \end{cases} \\ \\ u(0,t) = u(1,t) = 0 \end{cases}$$

Prendre $\Delta x = 0.25$ et $\Delta t = 0.1768$

Corrigé

On a $a = -2$, $b = 0$, $c = 1$ et $e = 4$,

Donc: $b^2 - 4ac = 0 - 4(-2)(1) = 8 > 0$, $m_1 = \frac{+2\sqrt{2}}{4} = \frac{\sqrt{2}}{2}$ et $m_2 = -\frac{\sqrt{2}}{2}$.

Pour chaque point, nous avons deux caractéristiques:

$\Delta t = \frac{\sqrt{2}}{2}\Delta x$ et $\Delta t = -\frac{\sqrt{2}}{2}\Delta x$ (voir figure 2).

Calculons u au point $R_2(0.5, 0.1768)$: On a l'équation

$$am\Delta p + c\Delta q + e\Delta t = 0,$$

suivant la droite PR_2 puis la droite QR_2 on trouve:

$$\begin{cases} -2\frac{\sqrt{2}}{2}(p_{R_2} - p_P) + (q_{R_2} - q_P) + 4\Delta t = 0, \\ \\ 2\frac{\sqrt{2}}{2}(p_{R_2} - p_Q) + (q_{R_2} - q_Q) + 4\Delta t = 0. \end{cases}$$

On prend: $p_P = (\frac{\partial u}{\partial x})_P = -4$ (car on est à droite du point P), $p_Q = (\frac{\partial u}{\partial x})_Q = -4$, et $q_P = (\frac{\partial u}{\partial t})_P = q_Q = (\frac{\partial u}{\partial t})_Q = 0$ et on les remplace dans le système pour avoir:

$$\begin{cases} -\sqrt{2}(p_{R_2} + 4) + q_{R_2} + 4\Delta t = 0, \\ \sqrt{2}(p_{R_2} + 4) + q_{R_2} + 4\Delta t = 0. \end{cases}$$

Après résolution de ce système, on obtient: $p_{R_2} = -4$ et $q_{R_2} = -\frac{\sqrt{2}}{2}$

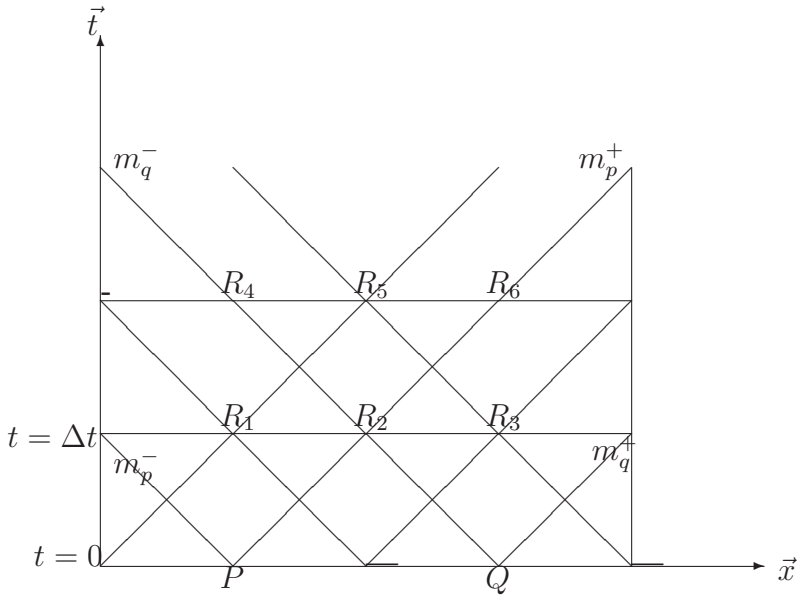


Figure 3.2:

Maintenant, on évalue u au point R_2 à travers ces variations le long de de $P \to R_2$.

$$\Delta u = p_{av}\Delta x + q_{av}\Delta t,$$

$$\Rightarrow u_{R_2} - u_P = (\frac{p_P + p_{R_2}}{2}) \times 0.25 + (\frac{q_P + q_{R_2}}{2}) \times 0.1768,$$

$$\Rightarrow u_{R_2} = (\frac{-4-4}{2}) \times 0.25 + (\frac{0 - \frac{\sqrt{2}}{2}}{2}) \times 0.1768 + 3.$$

$$\Rightarrow u_{R_2} = 1.9375$$

En continuant de la sorte à calculer les autres valeurs de u, on remplit le tableau suivant:

x	0	0.25	0.5	0.75	1
$u(t=0)$	0.0	3.0	2.0	1.0	0.0
$u(t=0.1768)$	0.0	0.9375	1.9375	0.9375	0.0
$u(t=0.3535)$	0.0	-1.1875	-0.2500	0.8125	0.0
$u(t=0.5303)$	0.0	-1.3125	-2.4375	-1.3125	0.0

Table 3.1: Tableau 1

Implementation numérique

code:

```
clear all;clc;
syms a b c e x u;
a=input('donner la valeur de a:');
b=input('donner la valeur de b: ');
c=input('donner la valeur de c: ');
e=input('donner la valeur de e: ');
dx=input('entrer le pas de la discretisation:  ');
delta=b*b-4*a*c; m=(b-sqrt(delta))/(2*a); n=1/dx; N=n;
d=zeros(N+1,n+1); t=zeros(N+1,n+1); p=zeros(N+1,n+1);
q=zeros(N+1,n+1); u=12*x; f=diff(u);
u=4-4*x; g=diff(u); y=0:dx:1;
i=1;
for j=1:n+1
    d(i,j)=y(i,j);
    if(0.25<=d(i,j))
        u(i,j)=4-4*d(i,j);
        p(i,j)=subs(g,x,d(i,j));
        q(i,j)=0;
    else
        if(d(i,j)==0)
            u(i,j)=12*d(i,j);
            p(i,j)=subs(f,x,d(i,j));
            q(i,j)=0;
        end
    end
end for i=2:N+1
    for j=2:n
        d(i,j)=(d(i-1,j-1)+d(i-1,j+1))/(2);
        t(i,j)=t(i-1,j-1)+m*(d(i,j)-d(i-1,j-1));
        A=[a*m c ; -a*m c];
        B=[a*m*p(i-1,j-1)+c*q(i-1,j-1)-e*(t(i,j)-t(i-1,j-1)) ;
        -a*m*p(i-1,j+1)+c*q(i-1,j+1)-e*(t(i,j)-t(i-1,j+1))];
        V=A\B;
        p(i,j)=V(1);
        q(i,j)=V(2);
        u(i,j)=((p(i,j)+p(i-1,j-1))/2)*(d(i,j)-d(i-1,j-1))
        +((q(i,j)+q(i-1,j-1))/2)*(t(i,j)-t(i-1,j-1))+u(i-1,j-1);
    end
```

```
    for j=1:n+1
        if(j==1)
            d(i,j)=0;
            t(i,j)=t(i-1,j+1)-m*(d(i,j)-d(i-1,j+1));
            q(i,j)=0;
            A=[-a*m];
            B=[-a*m*p(i-1,j+1)+c*q(i-1,j+1)-e*(t(i,j)-t(i-1,j+1))];
            V=A\B;
            p(i,j)=V(1);
        else
            if(j==n+1)
                d(i,j)=1;
                t(i,j)=t(i-1,j-1)+m*(d(i,j)-d(i-1,j-1));
                A=[a*m];
                B=[a*m*p(i-1,j-1)+c*q(i-1,j-1)-e*(t(i,j)-t(i-1,j-1))];
                V=A\B;
                p(i,j)=V(1);
            end
        end
    end
end mesh(t,d,double(u))

Remarque: B est une matrice qui s'\'ecrit sous la forme B=[v;w]
suite \'a la longueur du code nous avons tronqu\'es l'expression
de B de m\^eme pour l'expression de u
```

Execution du code:

x	0	0.25	0.5	0.75	1
$u(t=0)$	0.0	3.0	2.0	1.0	0.0
$u(t=0.1768)$	0.0	0.9375	1.9375	0.9375	0.0
$u(t=0.3535)$	0.0	-1.1875	-0.2500	0.8750	0.0
$u(t=0.5303)$	0.0	-1.3125	-2.3750	-1.3125	0.0

Table 3.2: tableau 1

Constat

Dans ce cas les caractéristiques données par l'equation $am^2 - bm + c$ sont des droites.ici nous avons pris un pas de discrétisation egal à 0.25.

3.2. Exercice 2:

Soit à résoudre:

$$\begin{cases} \frac{\partial^2 u}{\partial t^2} = (1+2x)\frac{\partial^2 u}{\partial x^2},\ x \in [0,1] \\ u(x,0) = 0 \\ \frac{\partial u}{\partial t}(x,0) = x(1-x) \\ u(0,t) = u(1,t) = 0 \end{cases}$$

Corrigé

Pour ce problème, $a = -(1+2x),\ b = 0,\ c = 1,\ e = 0$. La résolution de l'équation $am^2+bm+c = 0$ donne:

$$m = \pm\sqrt{\frac{1}{(1+2x)}}$$

Les courbes caractéristiques sont obtenues en résolvant les équations différentielles:

$$\begin{cases} \frac{dt}{dx} = \sqrt{\frac{1}{1+2x}}, \\ \frac{dt}{dx} = -\sqrt{\frac{1}{1+2x}}. \end{cases}$$

En intègrant à partir du point initial de coordonnées (x_0, t_0), on trouve:

$$\begin{cases} t = t_0 + \sqrt{1+2x} - \sqrt{1+2x_0},\ \text{pour } m_+, \\ t = t_0 - \sqrt{1+2x} + \sqrt{1+2x_0},\ \text{pour } m_-. \end{cases}$$

La figure suivante nous montre les différentes courbes caractéristiques que nous obtenons avec cette équation.

On choisit deux points $P(0, 0.25)$ et $Q(0, 0.75)$ et on se propose de chercher la valeur de u dans leur intérsection R qui, après résolution du système, donne les coordonnées $R(0.4841, 0.1782)$.

En suite , on résout l'équation 2.5 pour avoir $p = \frac{\partial u}{\partial x}$ et $q = \frac{\partial u}{\partial t}$.

Après utilisation de deux valeurs de m en chaque point, l'équation 2.5 donne le système suivant:

$$\begin{cases} -1.77341(0.45015)(p_R - 0) + (1)(q_R - 0.1875) = 0;\ P \to R, \\ -2.22341(-0.6726)(p_R - 0) + (1)(q_R - 0.1875) = 0;\ Q \to R. \end{cases}$$

Après la résolution de ce système, on obtient $p_R = 0$ et $q_R = 0.1875$.

Au point P: $x = 0.25$, $t = 0$, $u = 0$; $p = (\frac{\partial u}{\partial x})_P = 0$;

$q = (\frac{\partial u}{\partial t})_P = x - x^2 = 0.8175$;

$m = \sqrt{1/(1+2x)} = 0.18765$;

$a = -(1+2x) = -1.5$, $b = 0,\ c = 1,\ e = 0$

Au point Q: $x = 0.75$, $t = 0$, $u = 0$; $p = 0$;

$q = (\frac{\partial u}{\partial t})_P = x - x^2 = 0.1875$;

$m = -\sqrt{1/(1+2x)} = -0.6325$;

$a = -(1+2x) = -2.5$, $b = 0,\ c = 1,\ e = 0$

Au point R: $x = 0.4841$, $t = 0.1783$;

$m_+ = \sqrt{1/(1+2x)} = 0.7128$;

$m_- = -\sqrt{1/(1+2x)} = -0.7128$

$a = -(1+2x) = -1.9682$, $b = 0,\ c = 1,\ e = 0$

Pour trouver sa valeur au point R, on calcule la variation de u le long de deux caractéristiques en utilisant l'équation 2.6.

$$P \to R:\ \Delta u = 0(0.2341) + 0.1875(0.1783) = 0.0334$$

$$Q \to R:\ \Delta u = 0 + 0.0334 = 0.0334$$

$$\Rightarrow u_R = 0.0334 + u_P = 0.0334 + u_Q = 0.0334$$

Avec $\Delta u = u_R - u_P$.

La représentation graphique précédente résume les autres valeurs de u dans le maillage.

Implémentation numérique

```
clear all; clc;
 syms a b c e x u z h;
a=input('donner la valeur de a: ');
%
b=input('donner la valeur de b: ');
```

```
%
c=input('donner la valeur de c: ');
%
e=input('donner la valeur de e: ');
%
dx=input('entrer le pas de la discretisation:  ');
%
n=1/dx; N=n; d=zeros(N+1,n+1); t=zeros(N+1,n+1);
%
w=x.^((1:n+1)'*(1:n+1)); v=x.^((1:n+1)'*(1:n+1));
p=zeros(N+1,n+1); q=zeros(N+1,n+1); u=zeros(N+1,n+1);
delta=b*b-4*a*c; k=(b-sqrt(delta))/(2*a); g=simplify(k);
f=int(g,x); i=1; y=0:dx:1;
%
for j=1:n+1
    d(i,j)=y(i,j);
    w(i,j)=z-t(i,j);
    v(i,j)=int(g,d(i,j),h);
    q(i,j)=d(i,j)*(1-d(i,j));
    p(i,j)=0;
    r(i,j)=subs(a,x,d(i,j));
    m(i,j)=subs(g,x,d(i,j));
    u(i,j)=0;
end
%
j=1;
%
for i=2:N+1
    u(i,j)=0;
    q(i,j)=0;
    u(i,n+1)=0;
    q(i,n+1)=0;
end
%
for i=2:N+1
    for j=2:n
        M=w(i-1,j-1)-v(i-1,j-1);
        T=w(i-1,j+1)+v(i-1,j+1);
        [d(i,j),t(i,j)]=solve(M,T);
        v(i,j)=int(g,d(i,j),h);
        w(i,j)=z-t(i,j);
        r(i,j)=subs(a,x,d(i,j));
```

```
        m(i,j)=subs(g,x,d(i,j));
        S1=((r(i,j)+r(i-1,j-1))/2)*((m(i,j)+m(i-1,j-1))/2);
        S2=((r(i,j)+r(i-1,j+1))/2)*((-m(i,j)-m(i-1,j+1))/2);
        A=[S1 c ; S2 c];
        B=[S1*p(i-1,j-1)+c*q(i-1,j-1)-e*(t(i,j)-t(i-1,j-1)) ;
           S2*p(i-1,j+1)+c*q(i-1,j+1)-e*(t(i,j)-t(i-1,j+1))];
        V=A\B;
        p(i,j)=V(1);
        q(i,j)=V(2);
        u(i,j)=((p(i,j)+p(i-1,j-1))/2)*(d(i,j)-d(i-1,j-1))
        +((q(i,j)+q(i-1,j-1))/2)*(t(i,j)-t(i-1,j-1))+u(i-1,j-1);
    end
%
    for j=1:n+1
        if(j==1)
            d(i,j)=0;
            t(i,j)=t(i-1,j+1)-subs(sqrt(1+2*x),d(i,j))...
            +subs(sqrt(1+2*x),d(i-1,j+1));
            r(i,j)=subs(a,x,d(i,j));
            m(i,j)=subs(g,x,d(i,j));
            v(i,j)=int(g,d(i,j),h);
            w(i,j)=z-t(i,j);
            S2=((r(i,j)+r(i-1,j+1))/2)*((-m(i,j)-m(i-1,j+1))/2);
            A=[S2];
            B=[S2*p(i-1,j+1)+c*q(i-1,j+1)-e*(t(i,j)-t(i-1,j+1))];
            V=A\B;
            p(i,j)=V(1);
        else
            if(j==n+1)
                d(i,j)=1;
                t(i,j)=t(i-1,j-1)+subs(sqrt(1+2*x),d(i,j))...
                  -subs(sqrt(1+2*x),d(i-1,j-1));
                r(i,j)=subs(a,x,d(i,j));
                m(i,j)=subs(g,x,d(i,j));
                v(i,j)=int(g,d(i,j),h);
                w(i,j)=z-t(i,j);
                S1=((r(i,j)+r(i-1,j-1))/2)*((m(i,j)+m(i-1,j-1))/2);
                A=[S1];
                B=[S1*p(i-1,j-1)+c*q(i-1,j-1)-e*(t(i,j)-t(i-1,j-1))];
                V=A\B;
                p(i,j)=V(1);
            end
```

```
        end
    end
end
```

Execution du code:

0.0	0.0	0.0	0.0	0.0
0.0	0.00235	0.0334	0.0209	0.0
0.0	0.0297	-0.0205	0.0017	0.0
0.0	-0.0010	-0.0187	-0.0329	0.0

Table 3.3: tableau de u

Constat:

Dans ce cas les caractéristiques ne sont pas des droites ,la pente m est une fonction en x. Nous avons pris un pas égal à 0.25.

3.3. Exercice 3:

Soit à résoudre le système d'équations:

$$\begin{cases} \frac{\partial^2 u}{\partial x^2} - u\frac{\partial^2 u}{\partial t^2} + (1-x^2) = 0,\ x \in\ [0,\ 1] \\ \\ u(x,0) = x(1-x),\ u_t(x,0) = 0, \\ \\ u(0,t) = u(1,t) = 0. \end{cases}$$

Corrigé

On considère le point R donné par l'intersection de la caractéristique de pente positive du point $P(0.2, 0)$ et la caractéristique de pente négative du point $Q(0.4, 0)$. Comparons cette équation avec la forme standard:

$$au_{xx} + bu_{xt} + cu_{tt} + e = 0,$$

On trouve $a = 1$, $b = 0$, $c = -u$, $e = 1 - x^2$. On calcule, avant tout, la valeur numérique de u, p et q au point P et Q:
avec les conditions initiales,

$$u = x(1-x),$$

donc,

$$u_P = 0.2(1-0.2) = 0.16,$$
$$u_Q = 0.4(1-0.4) = 0.24;$$

De plus, avec les différentiation des conditions initiales,

$$p = \frac{\partial u}{\partial x} = 1 - 2x,$$

ce qui nous donne

$$p_P = 1 - 2(0.2) = 0.6,$$
$$p_Q = 1 - 2(0.4) = 0.2;$$

et enfin,

$$q = \frac{\partial u}{\partial t} = 0.$$

Alors:

$$q_P = 0 \text{ et } q_Q = 0$$

pour avoir les coordonnées du point R, nous devons connaître la pente m des caractéristiques.
En utilisant l'équation : $am^2 - bm + c = 0$, on obtient:

$$m = \frac{b \pm \sqrt{b^2 - ac}}{2a} = \frac{\pm\sqrt{4u}}{2} = \pm\sqrt{u}.$$

Puisque m dépend de la solution u, nous aurons besoin de trouver le point R par des approximations. Pour un premier essai, on utilise des valeurs initiales le long de tout l'arc; i.e, on prend $m_+ = +m_P$ et $m_- = -m_Q$:

$$m_+ = \sqrt{u_P} = \sqrt{0.16} = 0.4,$$
$$m_- = \sqrt{u_q} = -\sqrt{0.24} = -0.490.$$

A présent, on peut estimer les coordonnées de R en résolvant le système:

$$\begin{cases} t_R = m_+(x_R - x_P) = 0.4(x_R - 0.2) \\ \\ t_R = m_-(x_R - x_Q) = -0.49(x_R - 0.4). \end{cases}$$

Ce qui nous donne $x_R = 0.310$ et $t_R = 0.044$.
Puisqu'on ne connait pas m au point R, on utilise sa valeur initiale le long de chanque caractéristique pour approximer p et q au point R en utilisant l'équation 2.5:

$$am\Delta p + c\Delta q + e\Delta t = 0,$$

Nous remplacons les contantes connues par leurs valeurs pour chaque caractéristique pour avoir le système suivant:

$$(1)(0.4)(p_R - 0.6) + (-0.16)(q_R - 0) + \left(1 - \frac{0.04 + 0.096}{2}\right)(0.044) = 0$$

$$(1)(0.490)(p_R - 0.2) + (-0.24)(q_R - 0) + \left(1 - \frac{0.16 + 0.096}{2}\right)(0.044) = 0$$

Après résolution, on obtient $p_R = 0.399$ et $q_R = -0.246$.
Pour une première approximation de u au point R, on a:

$$\Delta u = p\Delta x + q\Delta t,$$

$$\Rightarrow u_R - 0.16 = \frac{0.6 + 0.399}{2}(0.310 - 0.2) + \frac{0 - 0.246}{2}(0.044 - 0),$$

$$u_R = 0.2095$$

Cette approximation a été établie en tenant compte de la courbe PR, en utilisant les différentes valeurs de p et q. Toutefois, on pourrait alternativement procéder par la caractéristique QR. Ce qui nous donnerait:

$$u_R - 0.24 = \frac{0.2 + 0.399}{2}(0.310) + \frac{0 - 0.246}{2}(0.044 - 0),$$

$$\Rightarrow u_R = 0.2076 \approx 0.2095$$

Implémentation numérique

```
clear all; syms a b c e u x;
a=input('donner la valeur de a: ');
b=input('donner la valeur de b: ');
u=input('donner la valeur de u:');
c=input('donner la valeur de c: ');
e=input('donner la valeur de e: ');
dx=input('entrer le pas de la discretisation: ');
n=1/dx; N=n; delta=b*b-4*a*c; m=(b+sqrt(delta))/(2*a);
d=zeros(N+1,n+1); t=zeros(N+1,n+1); q=zeros(N+1,n+1);
p=zeros(N+1,n+1); f=diff(u); i=1; y=0:dx:1; for j=1:n+1
    d(i,j)=y(i,j);
    u(i,j)=d(i,j)*(1-d(i,j));
    m(i,j)=sqrt(u(i,j));
    p(i,j)=subs(f,x,d(i,j));
    q(i,j)=0;
    h(i,j)=subs(e,x,d(i,j));
    r(i,j)=subs(c,x,d(i,j));
end
j=1;
for i=2:N+1
    u(i,j)=0;
    u(i,n+1)=0;
    if(mod(i,2)==0)
        for j=2:n+1
            A=[1 -m(i-1,j-1) ; 1 m(i-1,j)];
```

```
        B=[t(i-1,j-1)-m(i-1,j-1)*d(i-1,j-1)...
        + t(i-1,j)+m(i-1,j)*d(i-1,j)];
        K=A\B;
        t(i,j)=K(1);
        d(i,j)=K(2);
        u(i,j)=d(i,j)*(1-d(i,j));
        m(i,j)=sqrt(u(i,j));
        h(i,j)=subs(e,d(i,j));
        r(i,j)=subs(c,x,d(i,j));
        S1=(h(i,j)+h(i-1,j-1))/(2);
        S2=(h(i,j)+h(i-1,j))/(2);
        A=[a*m(i-1,j-1) r(i-1,j-1)-a*m(i-1,j) r(i-1,j)];
        B=[a*m(i-1,j-1)*p(i-1,j-1)+r(i-1,j-1)*q(i-1,j-1)...
        -S1*(t(i,j)-t(i-1,j-1))-a*m(i-1,j)*p(i-1,j)...
        +r(i-1,j)*q(i-1,j)-S2*(t(i,j)-t(i-1,j))];
        K=A\B;
        p(i,j)=K(1);
        q(i,j)=K(2);
        u(i,j)=((p(i,j)+p(i-1,j-1))/2)*(d(i,j)-d(i-1,j-1)) ...
        +((q(i,j)-q(i-1,j-1))/2)*(t(i,j)-t(i-1,j-1))+u(i-1,j-1);
    end
    for j=1:n+1
        if(j==1)
            d(i+1,j)=0;
            t(i+1,j)=t(i,j+1)-m(i,j+1)*(d(i+1,j)-d(i,j+1));
            m(i+1,j)=0;
            r(i+1,j)=subs(c,x,d(i+1,j));
            h(i+1,j)=subs(e,x,d(i+1,j));
            S2=((h(i+1,j)+h(i,j+1))/2);
            A=[-a*m(i,j+1)];
            B=[-a*m(i,j+1)*p(i,j+1)+r(i,j+1)*q(i,j+1)...
            -S2*(t(i+1,j)-t(i,j+1))];
            K=A\B;
            p(i,j)=K(1);
        else
            if(j==n+1)
                d(i+1,j)=1;
                t(i+1,j)=t(i,j)+m(i,j)*(d(i+1,j)-d(i,j));
                m(i+1,j)=0;
                r(i+1,j)=subs(c,x,d(i+1,j));
                h(i+1,j)=subs(e,x,d(i+1,j));
                S1=((h(i+1,j)+h(i,j))/2);
```

```
                    A=[a*m(i,j)];
                    B=[a*m(i,j)*p(i,j)+r(i,j)*q(i,j)...
                    -S1*(t(i+1,j)-t(i,j))];
                    K=A\B;
                    p(i,j)=K(1);
                end
            end
        end
    else
        if(mod(i,2)~=0)
            for j=2:n
                A=[1 -m(i-1,j) ; 1 m(i-1,j+1)];
                B=[t(i-1,j)-m(i-1,j)*(d(i-1,j))...
                + t(i-1,j+1)+m(i-1,j+1)*(d(i-1,j+1))];
                K=A\B;
                t(i,j)=K(1);
                d(i,j)=K(2);
                u(i,j)=d(i,j)*(1-d(i,j));
                m(i,j)=sqrt(u(i,j));
                h(i,j)=subs(e,x,d(i,j));
                r(i,j)=subs(c,x,d(i,j));
                S1=(h(i,j)+h(i-1,j))/2;
                s2=(h(i,j)+h(i-1,j+1))/2;
                A=[a*m(i-1,j) r(i-1,j) -a*m(i-1,j+1) r(i-1,j+1)];
                B=[a*m(i-1,j)*p(i-1,j)+r(i-1,j)*q(i-1,j)...
                -S1*(t(i,j)-t(i-1,j))-a*m(i-1,j+1)*p(i-1,j+1)...
                +r(i-1,j+1)*q(i-1,j+1)-S2*(t(i,j)-t(i-1,j+1))];
                K=A\B;
                p(i,j)=K(1);
                q(i,j)=K(2);
                u(i,j)=((p(i,j)+p(i-1,j))/2)*(d(i,j)-d(i-1,j))...
                +((q(i,j)+q(i-1,j))/2)*(t(i,j)-t(i-1,j))+u(i-1,j);
            end
        end
    end
end
```

Execution du code:

Constat:

On constate que m est une fonction qui dépend de la solution u. Nous avons pris un pas de discrétisation égal à 0.2.

0.0	0.1600	0.2400	0.2400	0.1600	0.0
0.0	0.1600	0.2096	0.2435	0.2049	0.1600
0.0	0.2096	0.2228	0.2166	0.2049	0.0
0.0	0.0415	0.02346	0.2233	0.2166	0.1831
0.0	0.0636	0.2143	0.2233	0.1897	0.0
0.0	0.149	0.0594	0.2143	0.2289	0.1880

Table 3.4: valeurs de u

3.4. Exercice 4

Soit à résoudre le système d'équation :

$$\begin{cases} \frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}, \ x \in [0,2], \\ u(0,t) = u(2,t) = 0, \\ \frac{\partial u}{\partial t}(x,0) = x(1-x), \quad 0 < x < 2, \\ u(x,0) = 2x \quad 0 < x < 1, \\ u(x,0) = 2(2-x) \quad 1 < x < 2, \end{cases}$$

La solution éxacte est donnée par :

$$u(x,t) = \frac{16}{\pi^2} \sum_{n=1}^{\infty} \frac{(-1)^{n-1} sin((2n-1)\frac{\pi x}{2}) cos((2n-1)\frac{\pi ct}{2})}{(2n-1)^2}.$$

Implémentation numérique

Code:

```
clear all; clc;
syms a b c e x k; cla=0;clb=2;
k=input('donner lavaleur de k: ');
a=input('donner la valeur de a: ');
b=input('donner la valeur de b: ');
c=input('donner la valeur de c: ');
e=input('donner la valeur de e: ');
dx=input('entrer le pas de discretisation: ');
n=(clb-cla)/dx N=n; delta=b*b-4*a*c; m=(b-sqrt(delta))/(2*a);
d=zeros(N+1,n+1); t=zeros(N+1,n+1); p=zeros(N+1,n+1);
q=zeros(N+1,n+1); y=0:dx:2; u=2*x; f=diff(u); u=2*(2-x);
g=diff(u); i=1; for j=1:n+1
```

```
    d(i,j)=y(i,j);
    if(1<=d(i,j))
        p(i,j)=subs(g,x,d(i,j));
        u(i,j)=2*(2-d(i,j));
        q(i,j)=0;
    else
        if(d(i,j)<1)
            p(i,j)=subs(f,x,d(i,j));
            u(i,j)=2*d(i,j);
            q(i,j)=0;
        end
    end
end j=1;
for i=2:N+1
    u(i,j)=0;
    q(i,j)=0;
    u(i,n+1)=0;
    q(i,n+1)=0;
end
for i=2:N+1
    for j=2:n
        d(i,j)=(d(i-1,j-1)+d(i-1,j+1))/(2);
        t(i,j)=t(i-1,j-1)+m*(d(i,j)-d(i-1,j-1));
        A=[a*m c ; -a*m c];
        B=[a*m*p(i-1,j-1)+c*q(i-1,j-1)-e*(t(i,j)-t(i-1,j-1)) ;
          -a*m*p(i-1,j+1)+c*q(i-1,j+1)-e*(t(i,j)-t(i-1,j+1))];
        V=A\B;
        p(i,j)=V(1);
        q(i,j)=V(2);
        u(i,j)=(((p(i,j)+p(i-1,j-1))*(d(i,j)-d(i-1,j-1)))/2)...
       +(((q(i,j)+q(i-1,j-1))*(t(i,j)-t(i-1,j-1)))/2)+u(i-1,j-1);
    end
    for j=1:n+1
        if(j==1)
            d(i,j)=0;
            t(i,j)=t(i-1,j+1)-m*(d(i,j)-d(i-1,j+1));
            A=[-a*m];
            B=[-a*m*p(i-1,j+1)+c*q(i-1,j+1)-e*(t(i,j)-t(i-1,j+1))];
            p(i,j)=A\B;
        else
            if(j==n+1)
                d(i,j)=2;
```

```
                t(i,j)=t(i-1,j-1)+m*(d(i,j)-d(i-1,j-1));
                A=[a*m];
                B=[a*m*p(i-1,j-1)+c*q(i-1,j-1)-e*(t(i,j)-t(i-1,j-1))];
                p(i,j)=A\B;
            end
        end
    end
end
 mesh(t,d,double(u))
```

Execution du code:

0.0	0.5000	1.0000	1.5000	2.0000	1.5000	1.0000	0.5000	0.0
0.0	0.5000	1.0000	1.0000	1.5000	1.5000	1.0000	0.5000	0.0
0.0	0.5000	0.5000	1.0000	0.5000	1.0000	1.0000	0.5000	0.0
0.0	0.0	0.5000	0.0	0.5000	0.0	0.5000	0.5000	0.0
0.0	0.0	-0.5000	0	-0.5000	0.0	-0.5000	0.0	0.0
0.0	-0.5000	-0.5000	-1.0000	-0.500	-1.0000	-0.5000	-1.0000	0.0
0.0	-0.5000	-1.0000	-1.0000	-1.5000	-1.0000	-1.5000	-0.5000	0.0
0.0	-0.5000	-1.0000	-1.5000	-1.5000	-2.0000	-1.0000	-1.5000	0.0
0.0	-0.5000	-1.0000	-1.5000	-2.0000	-1.5000	-2.0000	-0.5000	0.0

Table 3.5: valeurs de u

4. Conclusion

La résolution numérique des équations aux dérivées partielles restent un défit à relever, il existe des méthodes numériques qui permettent la résolution des EDPs telles que la méthode des différences finies et la méthode des caractéristiques, nous avons remarqué que pour quelques équations aux dérivées partielles l'équations d'onde par exemple, la méthode des différences finies posait problème, il fallait simplifier certains termes et ajouter une condition supplémentaire (condition mixte ou de Neumann) et la donnée d'une telle condition n'est pas usuelle, on sera souvent bloquer sur la résolution d'une équations hyperbolique si nous nous inspirons seulement de la méthode de différences finies; il fallait trouver une autre méthode entre autre la méthode des caractéristiques, cette dernière est simple à appliquer et donne une bonne approximation contrairement à la méthode des différences finies. Nos résultats obtenus correspondaient à ceux des manuels utilisés tandis que le graphe de la solution éxacte n'était pas identique à celui des valeurs approchées, ce qui met en doute nos résultats numériques. Mis à part ces désagréments, nous avons trouvé ce travail très intéressant et fructueux, par ce travail nous avons maitrisé l'outil Matlab, nous avons su que les EDPs hyperboliques sont utilisées en acoustique, océanographie, météorologie. En plus de cela ce travail nous a permis de connaitre qu'il existe plusieurs méthodes numériques

qui permettent d'approcher une équation aux dérivée partielles mais qu'elles sont limitées, ce qui nous conduit à affirmer qu'il n'existe pas une méthode numérique universelle pour approcher la solution éxacte.

Référence

[1] **Curtis F. Gerald, Patrick O. Wheatley**, "*Applied Numerical Analysis*". Third Edition, Addison-Wesley Publishing Company.