BE : Méthode des différences finies

Par groupe de 4 étudiants maximum, vous devez rédiger un rapport détaillé dans lequel vous répondrez aux questions du sujet et plus globalement au problème demandé. La qualité de rédaction ainsi que tous les compléments que vous apporterez (réflexions, analyses des réponses, etc...) seront fortement pris en compte. Toutes les fonctions programmées le seront en Python et seront abondamment commentées. Le rendu du projet se fera sous la forme d'un fichier .zip contenant le rapport ainsi que l'ensemble des programmes.

Partie 1 : L'équation de la chaleur sur une barre (le retour)...

Le but de cet exercice est de modéliser le phénomène de conduction thermique dans une barre à partir d'une discrétisation de l'équation de la chaleur. Soient L et T deux réels strictement positifs. On fixe $f \in \mathcal{C}^\infty([0,L]\times[0,T],\mathbb{R})$, $u_0 \in \mathcal{C}^\infty([0,L],\mathbb{R})$ et $\alpha,\beta \in \mathcal{C}^\infty([0,T],\mathbb{R})$ et γ un réel fixé strictement positif.

On considère le problème aux limites suivant : trouver une fonction $u = u(x,t)$ représentant la température au point (x,t) avec $x \in [0,L]$ et $t \in [0,T]$, telle que :

$$(1)\begin{cases} \frac{\partial u}{\partial t} - \gamma \frac{\partial^2 u}{\partial x^2} = f & \text{dans } [0,L]\times[0,T] \\ u(0,t) = \alpha(t) & \forall t \in [0,T] \\ u(L,t) = \beta(t) & \forall t \in [0,T] \\ u(x,0) = u_0(x) & \forall x \in [0,L] \end{cases}$$

On fixe M et N deux entiers et on discrétise $[0,L]\times[0,T]$ en introduisant les points $x_j = j\triangle x$ pour $j \in [\![0, M+1]\!]$ et les instants $t^n = n\triangle t$ pour $n \in [\![0,N]\!]$. On cherche alors $u_j^n \simeq u(x_j,t^n)$ pour tout $(j,n) \in [\![1,M]\!]\times[\![1,N]\!]$.

1. Nous allons appliquer le schéma numérique suivant :

$$\frac{1}{\triangle t}\left(\frac{3}{2}u_j^{n+1} - 2u_j^n + \frac{1}{2}u_j^{n-1}\right) - \frac{\gamma}{(\Delta x)^2}\left(u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}\right) = f_j^{n+1}$$

   (a) De quel type est cette EDP ?

   (b) Étudier l'ordre du schéma en temps et en espace.

   (c) Montrer que le schéma est inconditionnellement stable au sens de Von Neumann-Fourier i.e. que l'on a toujours un facteur d'amplification de module inférieur à 1 .

   (d) Montrer que le facteur d'amplification ξ temps vers 0 quand la quantité $\lambda = \gamma\dfrac{\Delta t}{(\Delta x)^2}$ tend vers $+\infty$.

   (e) Que peut-on en conclure sur la convergence de ce schéma ?

2. En posant $u^n = (u_1^n, u_2^n, \cdots, u_M^n)^T$, écrire sous la forme d'une suite vectorielle le schéma numérique précédent.

3. En notant u le vecteur colonne suivant :

$$u = \begin{pmatrix} u^1 \\ \vdots \\ u^N \end{pmatrix}$$

   montrer que le schéma numérique s'écrit matriciellement sous la forme :

$$Au = b$$

   où A est une matrice carrée de taille $M\times N$ et $b \in \mathbb{R}^{M\times N}$, tous deux composés de données connues.

4. Pour des mêmes conditions initiales comparer l'approche matricielle de la question précédente et l'approche vectorielles. Quelle est la principale limite de l'implémentation de l'approche matricielle ? Quelle est la principale limite de l'implémentation vectorielle ici ?

5. Programmation sous Python :
   Réaliser un programme permettant d'implémenter la suite vectorielle $(u^n)_{n\in[\![0,N]\!]}$ précédente pour les données suivantes :

— 1) $\begin{cases} \gamma = 1 \\ \alpha = \beta = f = 0 \\ L = 10 \\ T = 100 \\ u_0(x) = x(L-x) \end{cases}$

— 2) $\begin{cases} \gamma = 1 \\ \alpha(t) = 10(1 - \cos(15\pi t/T)) \\ \beta(t) = \alpha(t) \\ f = 0 \\ L = 10 \\ T = 100 \\ u_0(x) = 0 \end{cases}$

— 3) Des conditions de votre choix où α, β et f sont des fonctions non constantes.

Trouver numériquement u . Comparer les deux méthodes et tracer les différentes solutions. Votre programme se présentera sous la forme d'une fonction dépendant des paramètres de discrétisation.

6. En modifiant les conditions initiales interpréter le terme γ et l'influence de f sur l'évolution de la température dans la barre.

Exercice 2 : L'équation de la chaleur sur une surface

Le but de cet exercice est de modéliser le phénomène de conduction thermique dans une surface carrée à partir d'une discrétisation de l'équation de la chaleur. On conservera les notations de l'exercice précédent. On considère le problème aux limites suivant : trouver une fonction $u = u(x, y, t)$ représentant la température au point (x, y, t) avec $(x, y) \in [0, L]^2$ et $t \in [0, T]$, telle que :

$$(1) \begin{cases} \frac{\partial u}{\partial t} - \gamma \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) = f & \text{dans } [0, L]^2 \times [0, T] \\ u = 0 & \text{sur } \partial [0, L]^2 \\ u(x, y, 0) = u_0(x, y) & \forall (x, y) \in [0, L]^2 \end{cases}$$

On fixe M et N deux entiers et on discrétise $[0, L]^2 \times [0, T]$ en introduisant les points $x_i = i\triangle x$ et $y_j = j\Delta y$ pour $(i, j) \in [\![0, M+1]\!]^2$ et les instants $t^n = n\triangle t$ pour $n \in [\![0, N]\!]$. On cherche alors $u_{ij}^n \simeq u(x_i, y_j, t^n)$ pour tout $(i, j, n) \in [\![1, M]\!]^2 \times [\![1, N]\!]$.

Nous allons appliquer le schéma suivant :

$$\frac{1}{\triangle t} \left(\frac{3}{2} u_{ij}^{n+1} - 2u_{ij}^n + \frac{1}{2} u_{ij}^{n-1} \right) - \frac{\gamma}{(\Delta x)^2} \left(\delta_x^2 u_{ij}^{n+1} + \delta_y^2 u_{ij}^{n+1} \right) = f_{ij}^{n+1}$$

avec $\begin{cases} \delta_x^2 \phi_{ij}^n := \phi_{i+1j}^n - 2\phi_{ij}^n + \phi_{i-1j}^n \\ \delta_y^2 \phi_{ij}^n := \phi_{ij+1}^n - 2\phi_{ij}^n + \phi_{ij-1}^n \end{cases}$.

1. Étudier l'ordre du schéma en temps et en espace.
2. En posant u^n la matrice carrée dont les termes sont donnés par : u_{ij}^n , écrire sous la forme d'une suite matricielle le schéma numérique précédent.
3. Programmation sous Python :
   (a) Réaliser un programme permettant d'implémenter cette suite pour des données initiales et que vous choisirez. Votre programme se présentera sous la forme d'une fonction dépendant des paramètres de discrétisation.
   (b) En utilisant la fonction *np.linalg.inv* de Numpy, trouver numériquement les valeurs des u^n. Vous pouvez tracer vos résultats avec les commandes suivantes pour un instant donné n (à adapter si besoin) :

```
Zn = Un.reshape(M, M)
plt.contourf(x,y,Zn)
```

   (c) Réaliser des vidéos des sorties (= valeurs des u^n) que vous obtenez en fonction des conditions initiales choisies. On pourra utiliser les commandes d'OpenCV vu en Ma313.

En fonction des possibilités de votre ordinateur on pourra supposer que $u^0 = u^{-1}$.