Analyse Numérique, Applications 2

Introduction

Dans cette seconde partie, nous allons continuer à nous interesser au phénomène de diffusion en présence d'un terme de source, en 2 dimensions. L'équation à résoudre s'écrit alors

$$\frac{\partial u(x,y,t)}{\partial t} = S(x,y) + C\frac{\partial^2 u(x,y,t)}{\partial x^2} + C\frac{\partial^2 u(x,y,t)}{\partial y^2}$$

Le but de ce TP va être de s'intéresser à différentes méthodes de résolutions de cette équation, aussi bien au niveau stabilité que de l'exactitude des solutions fournies.

1 Considérations analytiques

1.1 Solutions générales de l'équation de diffusion sur un domaine fini

Cas 1D

$$\begin{aligned}
\partial_t u(x,t) &= C\partial_x^2 u(x,t) + Du(x,t) + S(x) && (1)\\
\text{Domaine :} \quad & 0 \leq x \leq Lx && (2)\\
u(x,0) &= u_0(x) && (3)\\
u(0,t) &= 0 && (4)\\
u(0,Lx) &= 0 && (5)\\
u(x,t) &= \int_0^{Lx} u_0(x)G(x,\xi,t)\mathrm{d}\xi + \int_0^t \int_0^{Lx} S(\xi)G(x,\xi,t-\tau)\mathrm{d}\xi\mathrm{d}\tau && (6)\\
G(x,\xi,t) &= \frac{2}{Lx}e^{(Dt)} \sum_{n=1}^{\infty} \sin\left(\frac{n\pi x}{Lx}\right) \sin\left(\frac{n\pi \xi}{Lx}\right) \exp\left(-\frac{Cn^2\pi^2 t}{Lx^2}\right) && (7)\\
& && (8)
\end{aligned}$$

Cas 2D

$$\begin{aligned}
\partial_t u(x,y,t) &= C\partial_x^2 u(x,t) + C\partial_y^2 u(x,y,t) + S(x,y) && (9)\\
\text{Domaine :} \quad & 0 \leq x \leq Lx && (10)\\
& -\infty \leq y \leq \infty && (11)\\
u(x,y,0) &= u_0(x,y) && (12)\\
u(0,y,t) &= 0 && (13)\\
u(Lx,y,t) &= 0 && (14)\\
u(x,t) &= \int_{-\infty}^{\infty}\int_{0}^{Lx} u_0(\xi,\eta)G(x,y,\xi,\eta,t)\mathrm{d}\xi\mathrm{d}\eta && (15)\\
& + \int_0^t \int_0^{Lx}\int_{-\infty}^{\infty} S(\xi,\eta)G(x,y,\xi,\eta,t-\tau)\mathrm{d}\xi\mathrm{d}\eta\mathrm{d}\tau && (16)\\
G(x,\xi,t) &= G_1(x,\xi,t)G_2(y,\eta,t) && (17)\\
G_1(x,\xi,t) &= \frac{2}{Lx}\sum_{n=1}^{\infty}\sin\left(\frac{n\pi x}{Lx}\right)\sin\left(\frac{n\pi\xi}{Lx}\right)\exp\left(-\frac{an^2\pi^2 t}{Lx^2}\right) && (18)\\
G_2(y,\eta,t) &= \frac{1}{2\sqrt{\pi C t}}\left[\exp\left(-\frac{(y-\eta)^2}{4Ct}\right) - \exp\left(-\frac{(y+\eta)^2}{4Ct}\right)\right] && (19)
\end{aligned}$$

2 Travail préparatoire analytique

Nous allons regarder comment nous pouvons modéliser un tel système au niveau numérique.

- Nous voulons simuler un domaine de taille infinie suivant la direction y, quelles méthodes pouvons nous utiliser ?
- Supposons que nous calculions la série de Fourier suivant la direction y de notre champs :

$$u(x,y,t) \to \sum_{m=0}^{M} u_m(x,t)\exp(\mathrm{i}mk_y y)$$

  Comment s'écrit alors notre équation ?
- Quelles hypothèses supplémentaires cela entraine sur la physique associée à notre problème ?
- Si nous supposons que $S(x,y) = 0$, quelle fonction pouvons nous choisir pour $u_0(x,y)$ afin de calculer analytiquement le taux de diffusion du système (hors cas triviaux pour $u_0(x,y)$ tels que $u_0(x,y) = Constante$?Peut on faire la même chose si nous utilisons la série de Fourier définie precedemment ?
- Calculez le taux de diffusion obtenu dans les deux cas.

3 Traitement explicite

Dans cette partie nous allons nous intéresser à la résolution de notre équation avec 2 méthodes différentes,

- méthode d'Euler explicite
- méthode de Runge-Kutta 4
- Écrire la formulation discrète de l'équation de la chaleur considérée ici
  Nous avions vu que, dans le cas de la méthode d'Euler, la condition C.F.L. (ou condition de stabilité du schéma) pour l'équation de diffusion $1D$ s'exprimait par la formule :

$$2C\frac{\Delta t}{\Delta x^2} \leq 1$$

- Quelle est la nouvelle expression dans le cas 2D réel ? si nous appliquons une transformée de Fourier suivant la direction y ?
- Si nous augmentons le nombre de points/nombre de modes, les 2 méthodes donnent-elles les mêmes résultats ? Sont-elles aussi précises ?

- Si nous changeons l'expression de $u_0(x, y)$ cela influence-t-il les résultats précédents, en particularité la stabilité ? Pourquoi ?
- Estimez l'erreur commise pour les différents schémas en fonction des paramètres que vous avez choisis.
- Pour un même jeu de paramètres, tracez le temps de calcul nécessaire au passage de t à $t+1$ pour les résolutions suivantes : 8×8,16×16,32×32,64×64,128×128, 256×256.Quel est le rapport entre les temps d'éxécution ?
- Quelles sont les différentes conditions de bord possible présentes dans le code ? Écrire le code de la fonction null_bc du fichier boundary.c afin d'implementer les conditions suivantes suivant y :

$$\partial_y u(x, 0, t) = 0, \quad \partial_y u(x, Ly, t) = 0$$

- Par rapport aux conditions actuellement présentes suivant la direction y, la solution obtenue avec ces nouvelles conditions est-elle différente ? pourquoi ?

4 Traitement implicite et semi-implicite

Dans cette partie nous allons nous intéresser à la résolution de notre équation avec 2 méthodes différentes,

- méthode d'Euler implicite
- méthode de Cranck-Nicholson
- Retrouvez l'expression du facteur d'amplification dans le cas d'une équation de diffusion $1D$ avec les méthodes d'Euler implicite et de Cranck-Nicholson.
- Avons nous encore une condition de stabilité comme dans le cadre des méthodes explicites ?
- Pouvons nous quand même choisir une valeur quelconque pour le pas de temps ?
- Quelles sont les différences entre la méthode d'Euler implicite et la méthode de Cranck-Nicholson ?
- Écrire l'équation de la chaleur sans source, sous forme matricielle, pour une utilisation de la la méthode d'Euler implicite dans les cas 1D et 2D. Le résultat doit se présenter sous la forme :

$$AU^{t+1} = S + BU^t$$

- Quelles différences a-t-on sur le type de matrice obtenue entre les cas 1D et 2D ?
- Les conditions de bords entrent-elles en ligne de compte sur la formulation matricielle ? Que devons nous changer si nous voulons passer de conditions de type Dirichlet à des conditions de type Von Neumann
- Quels changements a-t-on si nous utilisons une décomposition en série de Fourier suivant la direction y ?
- Quelles sont les méthodes que vous connaissez pour résoudre ce type de systèmes linéaires ?
- Dans le code, dans la fonction *euli*, comment se calcule U^{t+1} en fonction de U^t ?Comment appele-t-on ce type de méthode de résolution ?
- Typiquement, une inversion matricielle a une complexité proportionnelle à N^3 Tracez le temps de calcul nécessaire au passage de t à $t+1$ pour les résolutions suivantes : 8×8,16×16,32×32,64×64,128×128, 256×256.
- Est-ce que le temps de calcul est proportionnel à N^3 dans ce cas ? conclusion ?
- Même question concernant la méthode de relaxation et l'utilisation d'une discretisation utilisant les séries de Fourier.
- Á partir d'une condition initiale judicieusement choisie, comparez l'erreur obtenue avec la solution théorique en fonction du temps.

5 Conclusions

- Sur les différentes méthodes proposées, si vous ne pouviez choisir qu'une seule méthode de résolution pour résoudre l'équation de la chaleur 2D, laquelle choisiriez vous ? pourquoi ?

6 Annexes

6.1 commandes diverses

- Pour mesurer le temps d'éxécution d'un programme, on utilise sous linux la commande **time** : time ./bin/h2d_gcc.exe
  Remarque importante : une mesure de temps ne peut être considérée comme fiable que si elle est d'au minimum 10 secondes.
- Pour tracer des données en 3D avec gnuplot. Dans cet exemple, nous supposons que les champs sont de taille NX=64, NY=64

```
gnuplot> set dgrid3d 64,64
gnuplot> set hidden3d
gnuplot> splot "H2D_0000.dat" u 1:2:3 with lines
```

- Commandes Octave
  - Pour charger un fichier de données, utilisez la fonction *load*
  - Pour tracer ensuite les valeurs de $u(x, y)$, utilisez la fonction *surf*

```
octave> data=load('H2D_0000.dat');
octave> surf(data)
```

6.2 Utilisation du code H2D

6.2.1 Téléchargement

Le code peut-être récupéré à l'adresse suivante :
https ://github.com/GFuhr/MF_FCM6/zipball/master

6.2.2 Compilation

- sous linux la compilation peut se faire via la commande make. Contrairement au programme précédent, en particulier à cause du nombre de fichier, il n'est pas possible de faire une compilation directe en ligne de commande via gcc.
- sous windows, un fichier "projet" compatible avec la suite de développement *Code : :Blocks* est disponible dans le dossier *TP2/H2D/* ainsi qu'un fichier "solution" pour exécuter le code à partir de *Visual Studio*

6.2.3 Entrées/Sorties

Entrées :
Les paramètres initiaux sont entrées via une série de saisie à faire avant compilation dans le fichier params/params.h. Les paramètres pouvant être modifiés sont les suivants :

- "C" : valeur du coefficient de diffusion
- "NX" : nombre de points pour la discrétisation suivant x
- "NY" : nombre de points pour la discrétisation suivant y
- "DT" : valeur du pas de temps
- "ITER" : nombre d'itérations en temps
- "LX" : taille de la boite suivant la direction x
- "LY" : taille de la boite suivant la direction y
- "discret" : type de discretisation utilisée, peut valoir "real" ou "fourier"
- "scheme" : methode de résolution utilisée, peut valoir
  "eule" : algorithme d'euler explicite
  "euli" : algorithme d'euler implicite
  "eulis" : algorithme d'euler implicite avec méthode de relaxation
  "rk4" : méthode de Runge-Kutta 4
  "cn" : méthode de Cranck-Nicholson

Il est aussi possible de modifier les conditions initiales de la simulation ainsi que la valeur de la source en modifiant les expresssions se trouvant dans le fichier params/functions.c

Sorties :
Chaque exécution du programme crée deux nouveaux fichiers **H2D_GPLOT_XXXX.dat** et **H2D_OCT_XXXX.dat**. La différence entre ces 2 fichiers concerne juste le format utilisé pour tracer des données en 3D sous octave (fichiers $\_OCT\_$) ou sous gnuplot (fichiers $\_GPLOT\_$)

Les fichiers $\_OCT\_$, codé en ASCII, contiennent, Ny lignes de Nx colonnes avec les valeurs de $u(x, y, t)$. Les fichiers $\_GPLOT\_$, codé en ASCII, contiennent, $Nx * Ny$ lignes de Nx colonnes avec les valeurs de $u(x, y, t)$.

Format des sorties pour $u(x_i, y_j) \rightarrow u_{i,j}$:

Cas où "discret=real", fichiers $GPLOT$:

$$\text{Nx*Ny lines, 3 columns} \begin{cases} x_0 & y_0 & u_{i,j} \\ x_1 & y_0 & u_{i,j} \\ \vdots & \vdots & \vdots \\ x_{Nx-1} & y_0 & u_{Nx-1,Ny-1} \\ x_0 & y_1 & u_{0,1} \\ \vdots & \vdots & \vdots \\ x_{Nx-1} & y_{Ny-1} & u_{Nx-1,Ny-1} \end{cases} \tag{20}$$

Cas où "discret=fourier", fichiers $GPLOT$:

$$\text{2*Nx*Ny lines, 3 columns} \begin{cases} x_0 & m_0 & \Re(u_{i,j}) \\ x_1 & m_0 & \Re(u_{i,j}) \\ \vdots & \vdots & \vdots \\ x_{Nx-1} & m_0 & \Re(u_{Nx-1,Ny-1}) \\ x_0 & m_1 & \Re(u_{0,1} \\ \vdots & \vdots & \vdots \\ x_{Nx-1} & m_{Ny-1} & \Re(u_{Nx-1,Ny-1}) \\ x_0 & m_0 & \Im(u_{i,j}) \\ x_1 & m_0 & \Im(u_{i,j}) \\ \vdots & \vdots & \vdots \\ x_{Nx-1} & m_0 & \Im(u_{Nx-1,Ny-1}) \\ x_0 & m_1 & \Im(u_{0,1}) \\ \vdots & \vdots & \vdots \\ x_{Nx-1} & m_{Ny-1} & \Im(u_{Nx-1,Ny-1}) \end{cases} \tag{21}$$

Cas où "discret=real", fichiers OCT :

$$\text{Ny lines, Nx columns} \begin{cases} u_{0,0} & u_{1,0} & \cdots & u_{Nx-1,0} \\ u_{0,1} & u_{1,1} & \cdots & u_{Nx-1,1} \\ \vdots & \vdots & \vdots & \\ u_{0,Ny-1} & u_{1,Ny-1} & \cdots & u_{Nx-1,Ny-1} \end{cases} \tag{22}$$

Cas où "discret=fourier", fichiers OCT :

$$\text{2*Ny lines, Nx columns} \left\{ \begin{array}{cccc} \Re(u_{0,0}) & \Re(u_{1,0}) & \cdots & \Re(u_{Nx-1,0}) \\ \Re(u_{0,1}) & \Re(u_{1,1}) & \cdots & \Re(u_{Nx-1,1}) \\ \vdots & \vdots & \vdots & \\ \Re(u_{0,Ny-1}) & \Re(u_{1,Ny-1}) & \cdots & \Re(u_{Nx-1,Ny-1}) \\ \Im(u_{0,0}) & \Im(u_{1,0}) & \cdots & \Im(u_{Nx-1,0}) \\ \Im(u_{0,1}) & \Im(u_{1,1}) & \cdots & \Im(u_{Nx-1,1}) \\ \vdots & \vdots & \vdots & \\ \Im(u_{0,Ny-1}) & \Im(u_{1,Ny-1}) & \cdots & \Im(u_{Nx-1,Ny-1}) \end{array} \right. \tag{23}$$