Méthodes numériques, travaux pratiques

le rapport devra être envoyé par mail (guillaume.fuhr@univ-amu.fr)
avant le 06 mars 2016 au format pdf

Contents

Introduction

Le but du sujet proposé est d'étudier une équation d'advection-diffusion du point de vue des méthodes numériques de résolution. L'équation de base étudiée est la suivante :

$$\frac{\partial u(x,t)}{\partial t} + V\nabla u(x,t) = C\nabla^2 u(x,t) \tag{1}$$

$$\nabla := \begin{cases} \partial_x \\ \partial_y \\ \partial_z \end{cases}$$

Le domaine radial sera de taille $[L_x \times L_y] = [2\pi \times 2\pi]$, pour les diverses méthodes étudiées, il faudra choisir des valeurs pour Δt et Δx permettant d'atteindre un temps dans la simulation de l'ordre de $\simeq \mathbf{10}$ fois le temps caractéristique de la dynamique considérée.
Diffusion : $\tau_D = L^2/C$
Advection : $\tau_A = L/V$

Les coefficients et paramètres de grille ne sont pas imposés mais peuvent-être choisi librement. Un point de départ pour ces diverses valeurs vous pouvez utiliser une grille de taille $[8 \times 8, 16 \times 16, 32 \times 32, \cdots, 256 \times 256]$ et des valeurs pour C, V de l'ordre de $\left[10^{-2}, 10\right]$

1 1D equation

1.1 Diffusion

Dans cette partie, seul le terme de diffusion sera considéré :

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

1. Utilisez le programme advdiff pour résoudre l'équation utilisant la méthode d'Euler explicite. Déduisez du graphe de la solution aux divers temps les conditions de bords utilisées. Comment cela est-il implémenté dans le code?

2. Comparez les solutions obtenues quand $C\Delta t/\Delta x^2$ est $< 1/2, = 1/2 - \epsilon, = 1/2, = 1/2 + \epsilon, > 1/2$

3. Calculez le taux de croissance obtenu numérique et comparez le au taux analytique.

4. Pour Δx donné, observez vous une limite de stabilité pour le schéma RK4 en fonction du pas Δt? Estimez cette valeur à ± 0.01 prêt et comparez là au Δt correspondant à la méthode d'Euler explicite.

5. Théoriquement, il n'y a pas de limitation sur le pas de temps lors de l'utilisation d'un schéma implicite, est-ce vérifié numériquement? Commentez.

1.2 Advection

On s'interesse maintenant uniquement à l'équation d'advection :

$$\frac{\partial u(x,t)}{\partial t} + V\frac{\partial u(x,t)}{\partial x} = 0$$

1. Comment sont codées les dérivées spatiales dans le code? Est-ce un schéma avant, arrière ou centré?

2. Au sens physique, que fait une advection? en particulier, est-ce que la structure radiale de la solution est modifiée?

3. Qu'en est il dans les résultats obtenus avec les simulations numériques dans les deux cas suivant :

   - Quand on prend une fonction sin pour $u_0(x)$
   - Quand on prend une fonction de heavyside pour $u_0(x)$

4. Vérifiez si l'utilisation d'une méthode de RK4 permet d'utiliser un terme de dérivé centré en espace.

5. Comment évolue le système quand $V\Delta t/\Delta x$ est $< 1, \simeq 1, = 1, > 1$ pour les méthodes d'Euler et RK4.

6. Estimez la valeur maximale pour $V\Delta t/\Delta x$ si vous considérez un schéma RK4

1.3 Advection-Diffusion

Regardons maintenant l'équation complète :

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

1. Quel type de dérivée est utilisée pour le terme d'advection dans le schéma d'Euler implicite implémenté?

2. Comme dans le cas avec uniquement la diffusion, calculez le coefficient de diffusion à partir des données obtenues, ce résultat correspond-il à la valeur attendue?

3. Calculez à partir de vos résultats, le taux de croissance suivant qu'une dérivée centrée ou arrière est utilisée en espace pour le terme d'advection? Ce résultat dépend-il de la valeur du coefficient d'advection?

4. Dans les parties précédentes, deux critères de stabilités ont pu être mis en évidence, que se passe-t-il si l'on choisit des paramètres ne vérifiant que l'un des 2?

2 Equation de la chaleur en 2D

L'étude précédente va être comparée/étendue au cas à 2 dimensions.

$$\frac{\partial u(x,y,t)}{\partial t} = \frac{\partial^2 u(x,y,t)}{\partial x^2} + C\frac{\partial^2 u(x,y,t)}{\partial y^2}, \tag{3}$$

Nous ne regarderons que les méthodes d'Euler implicite et de Runge-Kutta 4 dans cette partie en assumant que la grille spatiale sera carré.

2.1 Discrétisation dans l'espace réel

1. Utilisant l'analyse de stabilité réalisée dans le cas $1D$, comment peut-on l'étendre simplement à ce cas $2D$?

2. Calculez l'erreur comparée à la solution analytique pour 3 résolutions différentes. Est-ce que l'erreur évolue en temps?

3. Pour un cas uniquement, estimez le temps de calcul nécessaire pour passer de t à $t+$ pour les résolutions suivantes : 8×8,16×16,32×32,64×64,128×128, 256×256. Is it linear with respect to number of points? proportional to N^2? N^3? Conclusion

4. En utilisant le code source de la fonction *euli*, expliquez comment U^{t+1} est calculé depuis U^t, en particulier, quel est le choix de méthode utilisée pour la résolution matricielle?

5. Quelles sont les différentes conditions de bord possible présentes dans le code? Ajoutez le code de la fonction null_bc du fichier boundary.c afin d'implementer les conditions suivantes suivant y:

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

6. Par rapport aux conditions actuellement présentes suivant la direction y, la solution obtenue avec ces nouvelles conditions est-elle différente? pourquoi? Vérifiez si le critère de stabilité est modifié.

2.2 Décomposition en modes de Fourier

Un autre choix de décomposition possible pour résoudre cette équation est d'utiliser la décomposition en modes de Fourier. Supposant que le domaine est périodique en y, une décomposition en modes de Fourier peut-être effectuée :

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

Nous avons alors pour chaque mode m l'équation suivante à résoudre :

$$\frac{\partial u_m(x,t)}{\partial t} = C \frac{\partial^2 u_m(x,t)}{\partial x^2} - C m^2 \kappa u_m(x,t), \tag{4}$$

1. La limité de stabilité est-elle modifiée suivant la direction x et/ou y ?

2. Plot computational time from t to $t + 1$ for the following resolutions : 8×8,16×16,32×32,64×64,128×128, 256×256.

3. Utilisant les mêmes conditions que en 2.2.2, estimez et tracez l'erreur relative obtenue et comparez là au cas précédent.

4. Cette fois-ci, estimez de nouveau le temps de calcul pour une itération en fonction de la résolution. Est-il plus rapide que dans le cas précédent?

5. Lors de l'utilisation des modes de Fourier, quelle méthode est employée pour l'inversion matricielle?

2.3 Conclusions

1. En utilisant les résultats précédents, si vous ne pouviez choisir qu'une seule méthode pour résoudre l'équation de la chaleur laquelle choisiriez vous? pourquoi?

3 Appendices

3.1 Résultats analytiques

3.1.1 Solution générale de l'équation de la chaleur dans un domaine de taille finie

cas 1D

$$\begin{aligned}
\partial_t u(x,t) &= C\partial_x^2 u(x,t) + Du(x,t) + S(x) && (5)\\
\text{Domaine :} \quad & 0 \leq x \leq Lx && (6)\\
u(x,0) &= u_0(x) && (7)\\
u(0,t) &= 0 && (8)\\
u(0,Lx) &= 0 && (9)\\
u(x,t) &= \int_0^{Lx} u_0(x)G(x,\xi,t)\mathrm{d}\xi + \int_0^t \int_0^{Lx} S(\xi)G(x,\xi,t-\tau)\mathrm{d}\xi\mathrm{d}\tau && (10)\\
G(x,\xi,t) &= \frac{2}{Lx}e^{(Dt)} \sum_{n=1}^{\infty} \sin\left(\frac{n\pi x}{Lx}\right)\sin\left(\frac{n\pi\xi}{Lx}\right)\exp\left(-\frac{Cn^2\pi^2 t}{Lx^2}\right) && (11)\\
& && (12)
\end{aligned}$$

cas 2D

$$\begin{aligned}
\partial_t u(x,y,t) &= C\partial_x^2 u(x,t) + C\partial_y^2 u(x,y,t) + S(x,y) && (13)\\
\text{Domaine :} \quad & 0 \leq x \leq Lx && (14)\\
& -\infty \leq y \leq \infty && (15)\\
u(x,y,0) &= u_0(x,y) && (16)\\
u(0,y,t) &= 0 && (17)\\
u(Lx,y,t) &= 0 && (18)\\
u(x,t) &= \int_{-\infty}^{\infty}\int_0^{Lx} u_0(\xi,\eta)G(x,y,\xi,\eta,t)\mathrm{d}\xi\mathrm{d}\eta && (19)\\
& + \int_0^t \int_0^{Lx}\int_{-\infty}^{\infty} S(\xi,\eta)G(x,y,\xi,\eta,t-\tau)\mathrm{d}\xi\mathrm{d}\eta\mathrm{d}\tau && (20)\\
G(x,\xi,t) &= G_1(x,\xi,t)G_2(y,\eta,t) && (21)\\
G_1(x,\xi,t) &= \frac{2}{Lx}\sum_{n=1}^{\infty}\sin\left(\frac{n\pi x}{Lx}\right)\sin\left(\frac{n\pi\xi}{Lx}\right)\exp\left(-\frac{an^2\pi^2 t}{Lx^2}\right) && (22)\\
G_2(y,\eta,t) &= \frac{1}{2\sqrt{\pi Ct}}\left[\exp\left(-\frac{(y-\eta)^2}{4Ct}\right) - \exp\left(-\frac{(y+\eta)^2}{4Ct}\right)\right] && (23)
\end{aligned}$$

3.2 Méthodes numériques

La convention suivante est utilisée dans la suite :

$$f(x,t) \to f(x_i,t_j) \to f(x+i\Delta x, t+j\Delta t) \to f_{t+i}^{x+j}$$

3.2.1 Dérivés

Partant d'une équation type,

$$\frac{\partial f(x,t)}{\partial t} = L(t, f(x,t)) \tag{24}$$

$$\begin{array}{lrcl} \text{dérivé avant :} & \partial_t f(t) & = & \frac{f^{t+1}-f^t}{\Delta t} \\ \text{dérivé arrière :} & \partial_t f(t) & = & \frac{f^t-f^{t-1}}{\Delta t} \\ \text{dérivé centrée :} & \partial_t f(t) & = & \frac{f^{t+1}-f^{t-1}}{2\Delta t} \end{array} \tag{25}$$

3.2.2 Euler explicite

$$\frac{f^{t+1}-f^t}{\Delta t} = L(t, f^t) \tag{26}$$

$$f^{t+1} = f^t + \Delta t L(t, f^t) \tag{27}$$

3.2.3 Euler implicite

$$\frac{f^{t+1}-f^t}{\Delta t} = L(t+\Delta t, f^{t+1}) \tag{28}$$

$$f^{t+1} = f^t + \Delta t L(t+\Delta t, f^{t+1}) \tag{29}$$

3.2.4 Cranck-Nicholson

$$\frac{f^{t+1}-f^t}{\Delta t} = \frac{1}{2}L(t, f^t) + \frac{1}{2}L(t+\Delta t, f^{t+1}) \tag{30}$$

$$f^{t+1} = f^t + \Delta t \left(\frac{1}{2}L(t, f^t) + \frac{1}{2}L(t+\Delta t, f^{t+1})\right) \tag{31}$$

3.2.5 Runge-Kutta 4

$$f^{t+1} = f^t + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{32}$$

$$\text{with} \tag{33}$$

$$k_1 = \Delta t L(t, f^t) \tag{34}$$

$$k_2 = \Delta t L(t + \frac{\Delta t}{2}, f^t + \frac{1}{2}k_1) \tag{35}$$

$$k_3 = \Delta t L(t + \frac{\Delta t}{2}, f^t + \frac{1}{2}k_2) \tag{36}$$

$$k_4 = \Delta t L(t + \Delta t, f^t + k_3) \tag{37}$$

$$\tag{38}$$

3.3 Code AdvDiff1D

3.3.1 téléchargement et compilation

adresse de téléchargement : **https://github.com/GFuhr/MF_FCM6/zipball/master**
Code source se trouve dans le dossier TP1.

Comment compiler

- Sous linux (Mac?) , la compilation se fait via la commande **make**.
- Pour éxécuter le programme généré sous linux, tapez **./advdiff.exe**
- Sous Windows, dans un éditeur type VisualStudio ou Code::Blocks, chargez les fichiers *advdiff.c* et *advdiff.h*.

3.3.2 Conditions initiales

temps

	$advection$	$diffusion$
$t = 0$	$u^0(x)$	$u^0(x)$

espace

	$advection$	$diffusion$
$x = 0$	$u(t, 0)$	$u(t, 0)$
$x = L_x$		$u(t, L_x)$

Champs initiaux

$u^0(x)$	$A \sin(\sigma \frac{\pi}{L_x}(x - x_0))$
$u^0(x)$	$A \exp(-(x - x_0)^2/\sigma^2)$
$u^0(x)$	$\begin{cases} A \text{ if } x \in [x_0 - \sigma/2,\ x_0 + \sigma/2] \\ 0 \text{ else} \end{cases}$

I/O Inputs :
Signification des entrées initiales.

- "Nx=?" : nombre de points en espace
- "Npas=?" : nombre d'itérations en temps
- "Nout=?" : chaque $Nout$ iterations, $u(t, x)$ sera exporté dans un fichier
- "C=?" : coefficient de diffusion
- "V=?" : coefficient d'advection
- "A=?" : Amplitude initiale, comme décrit en 2.3.2
- "x0=?": Paramètre x_0 défini dans 2.3.2
- "sigma=?": Paramètre σ défini dans 2.3.2
- "Dx=?" : valeur du pas Δx
- "Dt=?" : pas de temps Δt

Sorties :
À chaque éxecution du programme, *Nout* fichiers de sortie sont générés **out_XXXX.dat**. Remarque, ces fichiers ne sont pas écrasés entre les runs. Chaque fichier est codé en format texte (ASCII) et contient $(Nx - 2)$ points avec les valeurs de u_i^j au temps $t = j * Nout$.

3.4 H2D code

3.4.1 Download

Code is in the same archive as previous one :
https://github.com/GFuhr/MF_FCM6/zipball/master,
Le code H2D est dans le dossier TP2.

3.4.2 Compilation

- on linux, with make command.
- To run it on linux,after compilation step through **make**, type **./bin/h2d_gcc.exe**
- on windows, a "project" file usable with *Code::Blocks* and a "solution" file for *Visual Studio* can be found in folder *TP2/H2D/*

3.4.3 I/O

Inputs :
Les paramètres de la simulation sont à préciser dans le fichier params/params.h. Following parameters can be modified :

- "C" : diffusion coefficient
- "NX" : nombre de points dans la direction x
- "NY" : nombre de points dans la direction y
- "DT" : pas de temps
- "ITER" : nombre d'itérations en temps
- "LX" : Taille de la boîte suivant x
- "LY" : Taille de la boîte suivant y
- "discret" : discretisation spatiale, peut-être "real" ou "fourier"
- "scheme" : schéma numérique
  "eule" : explicit Euler
  "euli" : implicit Euler
  "eulis" : implicit Euler with relaxations
  "rk4" : Runge-Kutta 4
  "cn" : Cranck-Nicholson

Le profil initial ainsi que la source peuvent être modifiés dans le fichier params/functions.c

Outputs :
2 fichiers sont générés à chaque fois : **H2D_GPLOT_XXXX.dat** and **H2D_OCT_XXXX.dat**. La seule différence est que les fichiers ($\_OCT\_$) peuvent être utilisés avec octave et les fichiers ($\_GPLOT\_$) avec gnuplot.

Files $\_OCT\_$, contain Ny lines and Nx columns with values of $u(x, y, t)$. Files $\_GPLOT\_$ contain $Nx * Ny$ lines and 3 columns with values of $u(x, y, t)$.

Output format $u(x_i, y_j) \rightarrow u_{i,j}$**:**

Case where "discret=real", files $GPLOT$:

$$\text{Nx*Ny lines, 3 columns} \begin{cases} x_0 & y_0 & u_{i,j} \\ x_1 & y_0 & u_{i,j} \\ \vdots & \vdots & \vdots \\ x_{Nx-1} & y_0 & u_{Nx-1,Ny-1} \\ x_0 & y_1 & u_{0,1} \\ \vdots & \vdots & \vdots \\ x_{Nx-1} & y_{Ny-1} & u_{Nx-1,Ny-1} \end{cases} \tag{39}$$

Case where "discret=fourier", files $GPLOT$:

$$\text{2*Nx*Ny lines, 3 columns} \begin{cases} x_0 & m_0 & \Re(u_{i,j}) \\ x_1 & m_0 & \Re(u_{i,j}) \\ \vdots & \vdots & \vdots \\ x_{Nx-1} & m_0 & \Re(u_{Nx-1,Ny-1}) \\ x_0 & m_1 & \Re(u_{0,1} \\ \vdots & \vdots & \vdots \\ x_{Nx-1} & m_{Ny-1} & \Re(u_{Nx-1,Ny-1}) \\ x_0 & m_0 & \Im(u_{i,j}) \\ x_1 & m_0 & \Im(u_{i,j}) \\ \vdots & \vdots & \vdots \\ x_{Nx-1} & m_0 & \Im(u_{Nx-1,Ny-1}) \\ x_0 & m_1 & \Im(u_{0,1}) \\ \vdots & \vdots & \vdots \\ x_{Nx-1} & m_{Ny-1} & \Im(u_{Nx-1,Ny-1}) \end{cases} \tag{40}$$

Case where "discret=real", files OCT :

$$\text{Ny lines, Nx columns} \begin{cases} u_{0,0} & u_{1,0} & \cdots & u_{Nx-1,0} \\ u_{0,1} & u_{1,1} & \cdots & u_{Nx-1,1} \\ \vdots & \vdots & \vdots & \\ u_{0,Ny-1} & u_{1,Ny-1} & \cdots & u_{Nx-1,Ny-1} \end{cases} \tag{41}$$

Case where "discret=fourier", files OCT :

$$\text{2*Ny lines, Nx columns} \begin{cases} \Re(u_{0,0}) & \Re(u_{1,0}) & \cdots & \Re(u_{Nx-1,0}) \\ \Re(u_{0,1}) & \Re(u_{1,1}) & \cdots & \Re(u_{Nx-1,1}) \\ \vdots & \vdots & \vdots & \\ \Re(u_{0,Ny-1}) & \Re(u_{1,Ny-1}) & \cdots & \Re(u_{Nx-1,Ny-1}) \\ \Im(u_{0,0}) & \Im(u_{1,0}) & \cdots & \Im(u_{Nx-1,0}) \\ \Im(u_{0,1}) & \Im(u_{1,1}) & \cdots & \Im(u_{Nx-1,1}) \\ \vdots & \vdots & \vdots & \\ \Im(u_{0,Ny-1}) & \Im(u_{1,Ny-1}) & \cdots & \Im(u_{Nx-1,Ny-1}) \end{cases} \tag{42}$$

3.5 commandes utiles

- pour mesurer le temps d'éxécution, la commande **time** est utilisée :
  time ./bin/h2d_gcc.exe
  Remarque : une mesure ne peut être considérée comme fiable que si elle est supérieure à 10 secondes.

- How to plot 3D data with gnuplot. In following, we suppose NX=64 and NY=64

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

- tracer des données 1D avec gnuplot.

```
gnuplot> plot "out_0000.dat" with lines
```

- with Octave
  - to read datas use function *load*
  - to plot datas $u(x, y)$, use function *surf*

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

- download code in a terminal :
  wget https://github.com/GFuhr/MF_FCM6/zipball/master

- unzip archive :
  unzip master

- put unzipped files in a directory with a "friendly name" :
  mv GFuhr-MF_FCM6* *newname*

- change directory :
  cd

- create directory mkdir
  mkdir directory_name

- delete file :
  rm filename

- list file in a directory :
  ls directory_name

- list file in current directory :
  ls ./

3.6 premiers pas avec python

Attention en python les espaces en début de ligne sont très importants, ils permettent de séparer les niveaux des boucles/fonctions

- pour avoir les fonctions mathématiques

```
import numpy as np
import matplotlib.pyplot as plt
from os import chdir, getcwd
import sys
```

- changer de dossier (pour se mettre dans le dossier où sont les résultats des simulations) chdir('path')
- tracer les données d'un fichier et enregistrer le résultat sous forme de figure

```
plt.figure();
data = np.loadtxt('out__0000.dat')
plt.plot(data)
plt.xlabel('x')
plt.ylabel('u(x)')
plt.title('title')
plt.savefigs('plot.png')
plt.show()
```

- tracer une série de courbes pour des fichiers numérotés

```
plt.figure();
for idx in xrange(0,10):
    data = np.loadtxt('out__{0:04}.dat'.format(idx))
    print(max(data))
    plt.plot(data, label = idx)
plt.legend()
plt.xlabel('x')
plt.ylabel('u(x)')
plt.title('diffusion')
plt.show()
```

- Tracer les données du programme H2D :

```
from h2d_loader import  h2Dloader
data = h2Dloader('H2D_OCT_0430.dat')
data.convertreal()
data.plot()
data.plotmodes()
print data.max()
print data.min()
#data in real format
data = h2Dloader('H2D_OCT_0190.dat')
data.plot()
plt.show()
```