Introduction to Computational Science, Practical Work

Report must be send by mail (guillaume.fuhr@univ-amu.fr)
before March 4th, 2013 in pdf format

Contents

Introduction

The aim of this work will be to study numerical methods and results obtained for an advection-diffusion equation. This equation can be expressed as :

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

with

$$\nabla \rightarrow \begin{cases} \partial_x \\ \partial_y \\ \partial_z \end{cases}$$

In this work, you will have to compare different schemes,

- Explicit Euler scheme, 1st order
- Implicit Euler scheme, 1st order
- Runge-Kutta 4, 4th order explicit scheme

For each scheme you will have to compare impact of discretization used, steps... In the first section, you will have to study the full equation in a reduced 1D space, and in the second part you will look more closely to diffusion in a 2D space. Details on the numerical code which will be used in this practical work can be found in appendices 3.3 for section 1 and 3.4 for section 2.

1 1D equation

1.1 Diffusion

We consider only diffusive part :

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

1. Substitute centered finite difference in the equation
2. Write down equation in matrix form for implicit and explicit Euler method.
3. Derivate stability criterion in both cases.
4. Using output data plots, what kind of boundary conditions are used? How is it implemented in the code?
5. How does system evolves when $C\Delta t/\Delta x^2$ is $< 1/2, \simeq 1/2, = 1/2, > 1/2$
6. When $C\Delta t/\Delta x^2 < 1/2$, is the radial structure of $u(x,t)$ modified? How to demonstrate the result analytically?
7. Calculate the growth rate obtained with simulations, is it equal to C? If you find a difference, where can it come from?
8. Run same simulations using RK4 scheme.
9. If you fix spacial step Δx, do you observe a limit on time step (Δx) for numerical stability? Compare this value to Δt corresponding to explicit Euler.
10. Theoretically, there is no constraint on time step with implicit schemes, is it verified numerically? Why?

1.2 Advection

This time we consider only advection :

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

1. Write down equation in matrix form for implicit and explicit Euler methods
   - using backward finite difference
   - using centered finite difference
2. Which one is implemented in the code?
3. Modify the code to use the other scheme.
4. What is the physical definition of an advection ? Are your simulations in agreement?
5. Are your results modified if you use another time scheme ?
   - if we consider sin for $u_0(x)$
   - if we consider a gate function for $u_0(x)$
6. Is there any benefit to use Runge-Kutta scheme?
7. How does the system evolves when $C\Delta t/\Delta x$ is $< 1, \simeq 1, = 1, > 1$
8. For a fixed Δx is there a limit on time step with RK4? Compare this value to the one corresponding to explicit Euler scheme.

1.3 Advection-Diffusion

Now we consider the full equation :

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

1. Substitute centered finite difference in the system
2. What is the equivalent differential equation to this scheme?
3. What kind of derivative is implemented for advection with implicit Euler scheme?
4. Write down equation in matrix form. Can you expressed the result as a sum of 2 matrix, one for advection and one for diffusion?
5. Calculate numerically diffusive growth rate from simulations. Are your result in agreement with expected value?
6. Runs simulations with centered or backward difference for advection. Calculate growth rate, is it linked to diffusion coefficient?
7. In previous sections, it has been shown that stability criteria exist for diffusion and advection, what do you obtain if we choose parameters which verifies only one of them?

2 Heat equation in 2D

In this part, we consider only diffusion term in $2D$ case with a source term.

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

2.1 Preliminary work

1. If we want to consider an infinite domain in y direction, how can this be done numerically (give 2 possibilities)?
2. Using Discrete Fourier Transform (DFT) in y,

$$u(x,y,t) \rightarrow \sum_{m=0}^{M} u_m(x,t)\exp(imk_y y)$$

   rewrite equation.
3. From a physical point of view what changes in that case?
4. Supposing $S(x,y) = 0$, choose a function $u_0(x,y)$ such that growth rate can be calculated analytically (avoid trivial cases like $u_0(x,y) = Constant$) and calculate it.
5. Can we do the same analysis if we consider the Fourier form previously used? What is the growth rate in that case?

2.2 Explicit scheme

We will consider only explicit schemes here,

- Explicit Euler
- Runge-Kutta 4

1. Derive discrete form of the equation
2. Calculate the numerical stability condition when laplacian is expressed in real space. Same question if we consider Fourier transformation in y direction.
3. If we increase number of points/modes, does the two methods give the same results? Which one is the most precised? why?
4. If we change $u_0(x,y)$, are the previous results modified? Why?
5. Estimate error on numerical solution in your case.
6. For one set of inputs parameters, plot computation time spent from t to $t+1$ for the following resolutions : 8×8, 16×16, 32×32, 64×64, 128×128, 256×256.Is it linear with respect to resolution?
7. What are boundary conditions implemented? Add in the code a function null_bc in file boundary.c which correspond to following condition in y:

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

8. Does these new boundary conditions modify your final solution? Why?

2.3 Implicit and Cranck-Nicholson schemes

Now we consider only implicit/semi-explicit schemes,

- Implicit Euler
- Cranck-Nicholson

1. Express amplification factor for 1D diffusion equation for implicit Euler and Cranck-Nicholson schemes.
2. Compared to explicit Euler , is there a numerical stability condition?
3. What changes between implicit euler and Cranck-Nicholson schemes?
4. Give matrix form of the discrete equation for both 1D and 2D cases with implicit Euler. Result should be expressed like :
$$AU^{t+1} = S + BU^t$$
5. What are the differences concerning matrix obtained in the 2 cases?
6. Does boundary conditions modify the coefficients ? If yes, which ones?
7. Which coefficients should be modified if we want to use Von Neumann like boundaries?
8. And if we consider Fourier modes in y direction?
9. Give some methods used to resolve system of linear equations.
10. In the source code, considering *euli* function, how U^{t+1} is obtained from U^t? What is the name of this method ?
11. Typically, matrix inversion complexity is proportional to N^3. Plot computational time from t to $t+1$ for the following resolutions : 8×8,16×16,32×32,64×64,128×128, 256×256.
12. Is it proportional to N^3 in that case? conclusion?
13. Same question if Fourier transformation is considered with a relaxation method.
14. Using a well choose initial condition, compare relative error between numerical and theoretical solution in function of time.

2.4 Conclusions

1. Using previous results, if you can choose only one method to resolve 2D heat equation, which one will you choose and why.

3 Appendices

3.1 Analytical results

3.1.1 General solution of heat equation on a finite domain

1D case

$$\begin{aligned}
\partial_t u(x,t) &= C\partial_x^2 u(x,t) + Du(x,t) + S(x) && (2)\\
\text{Domain :} & \quad 0 \le x \le Lx && (3)\\
u(x,0) &= u_0(x) && (4)\\
u(0,t) &= 0 && (5)\\
u(0,Lx) &= 0 && (6)\\
u(x,t) &= \int_0^{Lx} u_0(x)G(x,\xi,t)\mathrm{d}\xi + \int_0^t \int_0^{Lx} S(\xi)G(x,\xi,t-\tau)\mathrm{d}\xi\mathrm{d}\tau && (7)\\
G(x,\xi,t) &= \frac{2}{Lx}e^{(Dt)}\sum_{n=1}^{\infty} \sin\left(\frac{n\pi x}{Lx}\right)\sin\left(\frac{n\pi\xi}{Lx}\right)\exp\left(-\frac{Cn^2\pi^2 t}{Lx^2}\right) && (8)\\
& && (9)
\end{aligned}$$

2D case

$$\begin{aligned}
\partial_t u(x,y,t) &= C\partial_x^2 u(x,t) + C\partial_y^2 u(x,y,t) + S(x,y) && (10)\\
\text{Domain :} & \quad 0 \le x \le Lx && (11)\\
& \quad -\infty \le y \le \infty && (12)\\
u(x,y,0) &= u_0(x,y) && (13)\\
u(0,y,t) &= 0 && (14)\\
u(Lx,y,t) &= 0 && (15)\\
u(x,t) &= \int_{-\infty}^{\infty}\int_0^{Lx} u_0(\xi,\eta)G(x,y,\xi,\eta,t)\mathrm{d}\xi\mathrm{d}\eta && (16)\\
& \quad + \int_0^t\int_0^{Lx}\int_{-\infty}^{\infty} S(\xi,\eta)G(x,y,\xi,\eta,t-\tau)\mathrm{d}\xi\mathrm{d}\eta\mathrm{d}\tau && (17)\\
G(x,\xi,t) &= G_1(x,\xi,t)G_2(y,\eta,t) && (18)\\
G_1(x,\xi,t) &= \frac{2}{Lx}\sum_{n=1}^{\infty} \sin\left(\frac{n\pi x}{Lx}\right)\sin\left(\frac{n\pi\xi}{Lx}\right)\exp\left(-\frac{an^2\pi^2 t}{Lx^2}\right) && (19)\\
G_2(y,\eta,t) &= \frac{1}{2\sqrt{\pi Ct}}\left[\exp\left(-\frac{(y-\eta)^2}{4Ct}\right) - \exp\left(-\frac{(y+\eta)^2}{4Ct}\right)\right] && (20)
\end{aligned}$$

3.2 Numerical schemes

Supposing this convention for discretization :

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

3.2.1 Derivatives

And considering an equation expressed as

$$\frac{\partial f(x,t)}{\partial t} = L(t, f(x,t)) \qquad (21)$$

$$\begin{array}{lrcl} \text{Forward difference :} & \partial_t f(t) & = & \frac{f^{t+1}-f^t}{\Delta t} \\ \text{Backward difference :} & \partial_t f(t) & = & \frac{f^{t}-f^{t-1}}{\Delta t} \\ \text{Centered difference :} & \partial_t f(t) & = & \frac{f^{t+1}-f^{t-1}}{2\Delta t} \end{array} \quad (22)$$

3.2.2 explicit Euler

$$\frac{f^{t+1}-f^t}{\Delta t} = L(t, f^t) \quad (23)$$
$$f^{t+1} = f^t + \Delta t L(t, f^t) \quad (24)$$

3.2.3 implicit Euler

$$\frac{f^{t+1}-f^t}{\Delta t} = L(t+\Delta t, f^{t+1}) \quad (25)$$
$$f^{t+1} = f^t + \Delta t L(t+\Delta t, f^{t+1}) \quad (26)$$

3.2.4 Cranck-Nicholson

$$\frac{f^{t+1}-f^t}{\Delta t} = \frac{1}{2}L(t, f^t) + \frac{1}{2}L(t+\Delta t, f^{t+1}) \quad (27)$$
$$f^{t+1} = f^t + \Delta t\left(\frac{1}{2}L(t, f^t) + \frac{1}{2}L(t+\Delta t, f^{t+1})\right) \quad (28)$$

3.2.5 Runge-Kutta 4

$$f^{t+1} = f^t + \frac{\Delta t}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) \quad (29)$$
$$\text{with} \quad (30)$$
$$k_1 = \Delta t L(t, f^t) \quad (31)$$
$$k_2 = \Delta t L(t+\frac{\Delta t}{2}, f^t + \frac{1}{2}k_1) \quad (32)$$
$$k_3 = \Delta t L(t+\frac{\Delta t}{2}, f^t + \frac{1}{2}k_2) \quad (33)$$
$$k_4 = \Delta t L(t+\Delta t, f^t + k_3) \quad (34)$$
$$(35)$$

3.3 Code AdvDiff1D

3.3.1 download source and compile

Source code can be download at : **https://github.com/GFuhr/MF_FCM6/zipball/master** Source code is in folder TP1.

how to compile

- On linux , this program can be compiled via the **make** command.
- On Windows, with any IDE editor such as VisualStudio or Code::Blocks, loading files *advdiff.c* and *advdiff.h*.

3.3.2 Initial Conditions

Time

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

Space

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

Initial Fields

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

I/O Inputs :
Initial parameters must be entered when you run the program.

- "Nx=?" : number of points in radial direction
- "Npas=?" : number of iterations
- "Nout=?" : every $Nout$ iterations, $u(t, x)$ will be write in a file
- "C=?" : diffusion coefficient
- "V=?" : advection coefficient
- "A=?" : Amplitude of the initial field described in 3.3.2
- "x0=?": parameter x_0 defined in 3.3.2
- "sigma=?" parameter σ defined in 3.3.2
- "Dx=?" : radial step Δx
- "Dt=?" : time step Δt

Outputs :
For each run of the program, another output file will be generated **out_XXXX.dat**. This file is coded in text (ASCII) contains $Npas/Nout$ lines and Nx columns with values of $u(t_i, x_j)$.

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 And H2D code is in folder TP2.

3.4.2 Compilation

- on linux, with make command
- 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 :
Initial parameters must be put in file params/params.h. Following parameters can be modified :

- "C" : diffusion coefficient
- "NX" : number of points in x direction
- "NY" : number of points in y direction
- "DT" : time step
- "ITER" : number of iterations
- "LX" : Box size in x direction
- "LY" : Box size in y direction
- "discret" : spacial discretization used, can be "real" or "fourier"
- "scheme" : numerical scheme used
  "eule" : explicit Euler
  "euli" : implicit Euler
  "eulis" : implicit Euler with relaxations
  "rk4" : Runge-Kutta 4
  "cn" : Cranck-Nicholson

Initial fields/source can be modified in file params/functions.c
Outputs :
Two files will be generated at each runs : **H2D_GPLOT_XXXX.dat** and **H2D_OCT_XXXX.dat**. The only difference is that (files *_OCT_*) can be used with octave and (files *_GPLOT_*) with gnuplot.
Files *_OCT_*, contain Ny lines de Nx columns with values of $u(x, y, t)$. Files *_GPLOT_* contain $Nx * Ny$ lines and Nx 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* **:**

$$\text{Nx*Ny lines, 3 columns}\left\{\begin{array}{ccc} x_0 & y_0 & u_{i,j} \\ x_1 & y_0 & u_{i,j} \\ \vdots & \vdots & \vdots \\ x_{Nx-1} & y_0 & u_{Nx-1,Ny-1} \\ x_0 & y_1 & u_{0,1} \\ \vdots & \vdots & \vdots \\ x_{Nx-1} & y_{Ny-1} & u_{Nx-1,Ny-1} \end{array}\right. \tag{36}$$

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

$$\text{2*Nx*Ny lines, 3 columns}\left\{\begin{array}{ccc} x_0 & m_0 & \Re(u_{i,j}) \\ x_1 & m_0 & \Re(u_{i,j}) \\ \vdots & \vdots & \vdots \\ x_{Nx-1} & m_0 & \Re(u_{Nx-1,Ny-1}) \\ x_0 & m_1 & \Re(u_{0,1} \\ \vdots & \vdots & \vdots \\ x_{Nx-1} & m_{Ny-1} & \Re(u_{Nx-1,Ny-1}) \\ x_0 & m_0 & \Im(u_{i,j}) \\ x_1 & m_0 & \Im(u_{i,j}) \\ \vdots & \vdots & \vdots \\ x_{Nx-1} & m_0 & \Im(u_{Nx-1,Ny-1}) \\ x_0 & m_1 & \Im(u_{0,1}) \\ \vdots & \vdots & \vdots \\ x_{Nx-1} & m_{Ny-1} & \Im(u_{Nx-1,Ny-1}) \end{array}\right. \tag{37}$$

Case where "discret=real", files *OCT* **:**

$$\text{Ny lines, Nx columns}\left\{\begin{array}{cccc} u_{0,0} & u_{1,0} & \cdots & u_{Nx-1,0} \\ u_{0,1} & u_{1,1} & \cdots & u_{Nx-1,1} \\ \vdots & \vdots & \vdots & \\ u_{0,Ny-1} & u_{1,Ny-1} & \cdots & u_{Nx-1,Ny-1} \end{array}\right. \tag{38}$$

Case where "discret=fourier", files *OCT* **:**

$$\text{2*Ny lines, Nx columns}\left\{\begin{array}{cccc} \Re(u_{0,0}) & \Re(u_{1,0}) & \cdots & \Re(u_{Nx-1,0}) \\ \Re(u_{0,1}) & \Re(u_{1,1}) & \cdots & \Re(u_{Nx-1,1}) \\ \vdots & \vdots & \vdots & \\ \Re(u_{0,Ny-1}) & \Re(u_{1,Ny-1}) & \cdots & \Re(u_{Nx-1,Ny-1}) \\ \Im(u_{0,0}) & \Im(u_{1,0}) & \cdots & \Im(u_{Nx-1,0}) \\ \Im(u_{0,1}) & \Im(u_{1,1}) & \cdots & \Im(u_{Nx-1,1}) \\ \vdots & \vdots & \vdots & \\ \Im(u_{0,Ny-1}) & \Im(u_{1,Ny-1}) & \cdots & \Im(u_{Nx-1,Ny-1}) \end{array}\right. \tag{39}$$

3.5 useful commands

- to measure computation time, command **time** is used :
  time ./bin/h2d_gcc.exe
  Remark : time measure can be considered reliable only if it's at least 10 seconds.

- 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
```

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

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