Introduction to Computational Science, Practical Work

Report must be send by mail (guillaume.fuhr@univ-amu.fr)
before February 15, 2016 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 :

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

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

Assuming a domain of size : $[L_x \times L_y] = [2\pi \times 2\pi]$, choosen Δt and Δx must allow a simulation representing $\simeq$ **10** times the typical time scale of the considered dynamic.
Diffusion time scale : $\tau_D = L^2/C$
Advection time scale : $\tau_A = L/V$

Values of parameters/grid sizes are not imposed/given, as starting point, you should use mesh size like $[8 \times 8, 16 \times 16, 32 \times 32, \cdots, 256 \times 256]$ and values of C, V in range like $\left[10^{-2}, 10\right]$

1 1D equation

1.1 Diffusion

Considering only diffusive part in 1D :

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

1. Write down discrete equation in matrix form for implicit and explicit Euler method.
$$AU^{t+1} = BU^t + S$$
2. Derivate stability criterion in both cases.
3. Assuming that $u(0,t) = u(L_x,t) = 0$ and $u(X,0) = u_0(x)$ derivate the analytic solution of the equation.
4. Simplify the obtained solution if $u_0(x) = A\sin\left(k\frac{\pi}{L_x}x\right)$
5. Resolve numerically previous equation using explicit Euler scheme. Using the plots of output data, what kind of boundary conditions are used? How is it implemented in the code?
6. How does system evolves when $C\Delta t/\Delta x^2$ is $< 1/2, = 1/2-\epsilon, = 1/2, = 1/2+\epsilon, > 1/2$
7. Calculate numerical growth rate and compare it with the analytic one.
8. For a given Δx, do you observe a limit on time step for numerical stability if you use RK4 scheme? Compare this value to Δt corresponding to explicit Euler scheme.
9. Theoretically, there is no constraint on time step with implicit schemes, is it verified numerically? Comments?

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 method
   - using backward finite difference in space
   - using centered finite difference in space
2. Calculate corresponding stability criterion in both cases.
3. Which implementation of derivative is coded?
4. What is the physical definition of an advection ? Is radial structure conserved in simulations?
   - if a sin function is used for $u_0(x)$
   - if an *heavyside* function is used for $u_0(x)$
5. Compare Euler and Runge-Kutta time schemes. In particular, stability limit and possibility for space derivative discretisation.
6. How does the system evolves when $V\Delta t/\Delta x$ is $< 1, \simeq 1, = 1, > 1$

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 on the system
2. Calculate the local truncature error of the scheme?
3. What kind of derivate is implemented for advection with implemented 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. Is 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 criterion exists for diffusion and advection, what happens if we choose parameters which verifies only one of these criterions?

2 Heat equation in 2D

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

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

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,

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

   write the corresponding equation.
3. From a physical point of view what changes in that case?
4. Supposing $S(x,y)=0$, Supposing $u_0(x,y) = A\sin(k_x x)\sin(k_y y)$ calculate the corresponding growth rate. Can we do the same analysis if we consider the Fourier form previously used?

2.2 Explicit scheme

We will consider only explicit schemes here,

- Explicit Euler
- Runge-Kutta 4

1. Give discret form of the equation
2. Calculate the numerical stability condition when laplacian is expressed in real space. Same question if we consider DFT 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? Comment.
4. Does the initial condition $u_0(x,y)$ modifies previous results? Why?
5. Estimate error with respect to exact solution in numerical solution with parameters you have used.
6. For one set on inputs parameters, represent 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 number of points?
7. What are the boundary conditions implemented? Implement 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. How does these new B.C. affects the results.

2.3 Implicit and Cranck-Nicholson schemes

Now we consider only implicit/semi-explicit schemes,

- Implicit Euler
- Cranck-Nicholson

1. From the stability analysis, 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. In the source code, considering *euli* function, how U^{t+1} is computed from U^t, in particular is it a direct or iterative method?
10. 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.
11. Is it proportional to N^3 ? conclusion?
12. Same question if Fourier transformation is considered with a relaxation method.
13. Using the same initial condition as in 2.2.5, compare both relative errors, which method give the best results.

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

$$
\begin{aligned}
\partial_t u(x,t) &= C\partial_x^2 u(x,t) + Du(x,t) + S(x) && (3)\\
\text{Domaine :} \quad & 0 \le x \le Lx && (4)\\
u(x,0) &= u_0(x) && (5)\\
u(0,t) &= 0 && (6)\\
u(0,Lx) &= 0 && (7)\\
u(x,t) &= \int_0^{Lx} u_0(x)G(x,\xi,t)\mathrm{d}\xi + \int_0^t \int_0^{Lx} S(\xi)G(x,\xi,t-\tau)\mathrm{d}\xi\mathrm{d}\tau && (8)\\
G(x,\xi,t) &= \frac{2}{Lx}e^{(Dt)}\sum_{n=1}^{\infty}\sin\left(\frac{n\pi x}{Lx}\right)\sin\left(\frac{n\pi\xi}{Lx}\right)\exp\left(-\frac{Cn^2\pi^2 t}{Lx^2}\right) && (9)\\
& && (10)
\end{aligned}
$$

2D case

$$
\begin{aligned}
\partial_t u(x,y,t) &= C\partial_x^2 u(x,t) + C\partial_y^2 u(x,y,t) + S(x,y) && (11)\\
\text{Domaine :} \quad & 0 \le x \le Lx && (12)\\
& -\infty \le y \le \infty && (13)\\
u(x,y,0) &= u_0(x,y) && (14)\\
u(0,y,t) &= 0 && (15)\\
u(Lx,y,t) &= 0 && (16)\\
u(x,t) &= \int_{-\infty}^{\infty}\int_0^{Lx} u_0(\xi,\eta)G(x,y,\xi,\eta,t)\mathrm{d}\xi\mathrm{d}\eta && (17)\\
& + \int_0^t \int_0^{Lx} \int_{-\infty}^{\infty} S(\xi,\eta)G(x,y,\xi,\eta,t-\tau)\mathrm{d}\xi\mathrm{d}\eta\mathrm{d}\tau && (18)\\
G(x,\xi,t) &= G_1(x,\xi,t)G_2(y,\eta,t) && (19)\\
G_1(x,\xi,t) &= \frac{2}{Lx}\sum_{n=1}^{\infty}\sin\left(\frac{n\pi x}{Lx}\right)\sin\left(\frac{n\pi\xi}{Lx}\right)\exp\left(-\frac{an^2\pi^2 t}{Lx^2}\right) && (20)\\
G_2(y,\eta,t) &= \frac{1}{2\sqrt{\pi Ct}}\left[\exp\left(-\frac{(y-\eta)^2}{4Ct}\right) - \exp\left(-\frac{(y+\eta)^2}{4Ct}\right)\right] && (21)
\end{aligned}
$$

3.2 Numerical schemes

Supposing this convention for discretisation :

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

3.2.1 Derivates

And considering an equation expressed as

$$\frac{\partial f(x,t)}{\partial t} = L(t, f(x,t)) \qquad (22)$$

$$\begin{array}{lcll} \text{Forward difference :} & \partial_t f(t) & = & \frac{f^{t+1}-f^t}{\Delta t} \\ \text{Backward difference :} & \partial_t f(t) & = & \frac{f^{t}-f^{t-1}}{\Delta t} \\ \text{Centered difference :} & \partial_t f(t) & = & \frac{f^{t+1}-f^{t-1}}{2\Delta t} \end{array} \tag{23}$$

3.2.2 explicit Euler

$$\frac{f^{t+1}-f^t}{\Delta t} = L(t, f^t) \tag{24}$$
$$f^{t+1} = f^t + \Delta t L(t, f^t) \tag{25}$$

3.2.3 implicit Euler

$$\frac{f^{t+1}-f^t}{\Delta t} = L(t+\Delta t, f^{t+1}) \tag{26}$$
$$f^{t+1} = f^t + \Delta t L(t+\Delta t, f^{t+1}) \tag{27}$$

3.2.4 Cranck-Nicholson

$$\frac{f^{t+1}-f^t}{\Delta t} = \frac{1}{2}L(t, f^t) + \frac{1}{2}L(t+\Delta t, f^{t+1}) \tag{28}$$
$$f^{t+1} = f^t + \Delta t \left(\frac{1}{2}L(t, f^t) + \frac{1}{2}L(t+\Delta t, f^{t+1})\right) \tag{29}$$

3.2.5 Runge-Kutta 4

$$f^{t+1} = f^t + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{30}$$
$$\text{with} \tag{31}$$
$$k_1 = \Delta t L(t, f^t) \tag{32}$$
$$k_2 = \Delta t L(t + \frac{\Delta t}{2}, f^t + \frac{1}{2}k_1) \tag{33}$$
$$k_3 = \Delta t L(t + \frac{\Delta t}{2}, f^t + \frac{1}{2}k_2) \tag{34}$$
$$k_4 = \Delta t L(t + \Delta t, f^t + k_3) \tag{35}$$
$$\tag{36}$$

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.
- To run it on linux,after compilation step through **make**, type **./advdiff.exe**
- 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 :
Everytime you run the program, $Nout$ output files will be generated **out_XXXX.dat**. Please notice that files will not be erased between runs. Each file is coded in text format (ASCII) and contains $(Nx - 2)$ points with values of u_i^j at time $t = j * Nout$.

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,
H2D code is in folder 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 :
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 time 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 and 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* :

$$\text{Nx*Ny lines, 3 columns}\left\{\begin{array}{ccc} x_0 & y_0 & u_{i,j} \\ x_1 & y_0 & u_{i,j} \\ \vdots & \vdots & \vdots \\ x_{Nx-1} & y_0 & u_{Nx-1,Ny-1} \\ x_0 & y_1 & u_{0,1} \\ \vdots & \vdots & \vdots \\ x_{Nx-1} & y_{Ny-1} & u_{Nx-1,Ny-1} \end{array}\right. \tag{37}$$

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

$$\text{2*Nx*Ny lines, 3 columns}\left\{\begin{array}{ccc} x_0 & m_0 & \Re(u_{i,j}) \\ x_1 & m_0 & \Re(u_{i,j}) \\ \vdots & \vdots & \vdots \\ x_{Nx-1} & m_0 & \Re(u_{Nx-1,Ny-1}) \\ x_0 & m_1 & \Re(u_{0,1} \\ \vdots & \vdots & \vdots \\ x_{Nx-1} & m_{Ny-1} & \Re(u_{Nx-1,Ny-1}) \\ x_0 & m_0 & \Im(u_{i,j}) \\ x_1 & m_0 & \Im(u_{i,j}) \\ \vdots & \vdots & \vdots \\ x_{Nx-1} & m_0 & \Im(u_{Nx-1,Ny-1}) \\ x_0 & m_1 & \Im(u_{0,1}) \\ \vdots & \vdots & \vdots \\ x_{Nx-1} & m_{Ny-1} & \Im(u_{Nx-1,Ny-1}) \end{array}\right. \tag{38}$$

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

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

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

$$\text{2*Ny lines, Nx columns}\left\{\begin{array}{cccc} \Re(u_{0,0}) & \Re(u_{1,0}) & \cdots & \Re(u_{Nx-1,0}) \\ \Re(u_{0,1}) & \Re(u_{1,1}) & \cdots & \Re(u_{Nx-1,1}) \\ \vdots & \vdots & \vdots & \\ \Re(u_{0,Ny-1}) & \Re(u_{1,Ny-1}) & \cdots & \Re(u_{Nx-1,Ny-1}) \\ \Im(u_{0,0}) & \Im(u_{1,0}) & \cdots & \Im(u_{Nx-1,0}) \\ \Im(u_{0,1}) & \Im(u_{1,1}) & \cdots & \Im(u_{Nx-1,1}) \\ \vdots & \vdots & \vdots & \\ \Im(u_{0,Ny-1}) & \Im(u_{1,Ny-1}) & \cdots & \Im(u_{Nx-1,Ny-1}) \end{array}\right. \tag{40}$$

3.5 usefull 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 greater than 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
```

- How to plot 1D data with 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 ./