**Introduction to Computational Science, Practical Work**

**Report must be sent by mail ( guillaume.fuhr@univ-amu.fr )**
**before Febuary, 17 2019 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 $\Delta t$ and $\Delta x$ must allow a simulation representing $\simeq \mathbf{10}$ 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. Resolve numerically discrete equation using advdiff.exe program and using explicit Euler scheme. Using the plots of output data, what kind of boundary conditions are used? How is it implemented in the code?

2. How does system evolves when $C\Delta t/\Delta x^2$ is $< 1/2, = 1/2 - \epsilon, = 1/2, = 1/2 + \epsilon, > 1/2$

3. Calculate numerical growth rate and compare it with the analytic one.

4. For a given $\Delta x$, do you observe a limit on time step for numerical stability if you use RK4 scheme? Estimate this with a precision of $10^{-2}$ and compare this value with the one corresponding to an explicit Euler scheme.

5. 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. Which implementation of spatial derivative are implemented in the code? is it backward, centered or forward scheme?
2. What is the physical definition of an advection ? Is radial structure conserved in simulations?
3. Is it the case in the simulations for the two following cases :
   - if a sin function is used for $u_0(x)$
   - if an *heavyside* function is used for $u_0(x)$
4. Verify if solution is convergent in the case where you use an RK4 time scheme and centered derivatives in space.
5. How does the system evolves when $V\Delta t/\Delta x$ is $< 1, \simeq 1, = 1, > 1$
6. As in previous section, estimate the stability limit when you consider the RK4 time 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. What kind of derivate is implemented for advection with implicit Euler scheme ?
2. Calculate numerically diffusive growth rate from simulations. Is your result in agreement with expected value?
3. Runs simulations with centered or backward difference for advection. Calculate growth rate,is it linked to advection coefficient?
4. 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

Previous work will be extended to a $2D$ heat equation with a source.

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

Only Implicit Euler (IE) and Rung-Kutta 4 (RK4) time discretizations on a squared spatial grid will be studied in this part ($N_x = N_y$).

### 2.1 discretization in real space

1. Using stability analysis made for the $1D$ case, how can you extend the stability limit in this case?
2. Estimate error with respect to exact solution in the numerical solution for 3 resolutions. Does the error remains the same in time?

3. For one set on inputs parameters, represent computation time spent from $t$ to $t+1$ for the following resolutions : $8 \times 8$,$16 \times 16$,$32 \times 32$,$64 \times 64$,$128 \times 128$, $256 \times 256$. Is it linear with respect to number of points? proportional to $N^2$? $N^3$? Conclusion

4. 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?

5. What are the boundary conditions implemented? Implement in the code a function null_bc in file boundary.c which correspond to following condition in $y$:

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

6. How does these new B.C. affects the results. Check numerically if the stability criterion is affected by these new conditions.

### 2.2 Fourier discretization

Finite difference is not the only spatial discretization which can be used in that case. Assuming periodicity in the $y$ direction, Fourier modes decomposition can also be used :

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

As consequence, Eq. 3 can be splitted for each mode $m$ :

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

1. Is the stability modified in $x$ and/or $y$ direction?

2. Estimate and plot the relative error and compare it to the previous case using the following resolutions ($N_x times N_m$) : $8 \times 8$,$16 \times 16$,$32 \times 32$,$64 \times 64$,$128 \times 128$, $256 \times 256$.

3. Plot computational time from $t$ to $t+1$ for the same resolutions.

4. Compare with your results obtained in section 2.1

5.

### 2.3 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**

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

**2D case**

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

## 3.2 Numerical schemes

Supposing this convention for discretisation :

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

### 3.2.1 Derivates

And considering an equation expressed as

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

$$\begin{array}{lrcl} \text{Forward difference :} & \partial_t f(t) & = & \frac{f^{t+1}-f^t}{\Delta t} \\ \text{Backward difference :} & \partial_t f(t) & = & \frac{f^t-f^{t-1}}{\Delta t} \\ \text{Centered difference :} & \partial_t f(t) & = & \frac{f^{t+1}-f^{t-1}}{2\Delta t} \end{array} \tag{25}$$

#### 3.2.2 explicit Euler

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

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

#### 3.2.3 implicit Euler

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

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

#### 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 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 $\sigma$ defined in 3.3.2
- "Dx=?" : radial step $\Delta x$
- "Dt=?" : time step $\Delta t$

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 (or number of modes associated to the $y$ direction in Fourier space)
- "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 $GPLOT$ :**

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

**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 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 ./

### 3.6 Some python commands

- modules to load at startup to be able to use mathematical functions

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

- change folder chdir('path')
- Load data in a text file, plot output and save result as a png file

```
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()
```

- Plot data series correspondingto numbered files

```
plt.figure();
for idx in np.arange(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()
```

- Load and plot H2D data :

```
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()
```