CALCOLO SCIENTIFICO

Project 4

Student
Pranav KASELA
Roll no. 866261

Student
Roberto CAVASSI
Roll no. 868504

June 24, 2018

Contents

Exercise 1 **1**
- Numerical Implementation 5
  - UpWind 5
  - Lax- Friederichs 8

Esercizio 2 **11**

MATLAB Codes **16**
- Exercise 1 Code 16
  - Esericizio 1 (A) 16
  - Esericizio 1 (B) 18
  - Esericizio 1 (C) 20
- Exercise 2 Code 23
  - K1 23
  - Average 23
  - Esercizio 2 24

Exercise 1

We study the following three problems for $t \geq 0, x \in (-\infty, +\infty)$:

$$\begin{aligned}(A) : u_t + 2u_x = 0\\ (B) : u_t + xu_x = 0\\ (C) : u_t + (xu)_x = 0\end{aligned}$$

with the initial condition $u_0(x) = e^{-x^2}$. There three are linear transport equations
We solve them using the method of characteristics and find the exact solution starting by (A) :
We have that $\dot{x} = 2$ thus $x = x_0 + 2t \Rightarrow x_0 = x - 2t$ and u is constant on this characteristics since $\dot{u} = u_t + \dot{x}u = 0$ so

$$u(x,t) = u_0(x_0) = e^{-(x-2t)^2}$$

is the exact solution. We draw the characteristics in Figure 1 to give an idea how the data is transported.

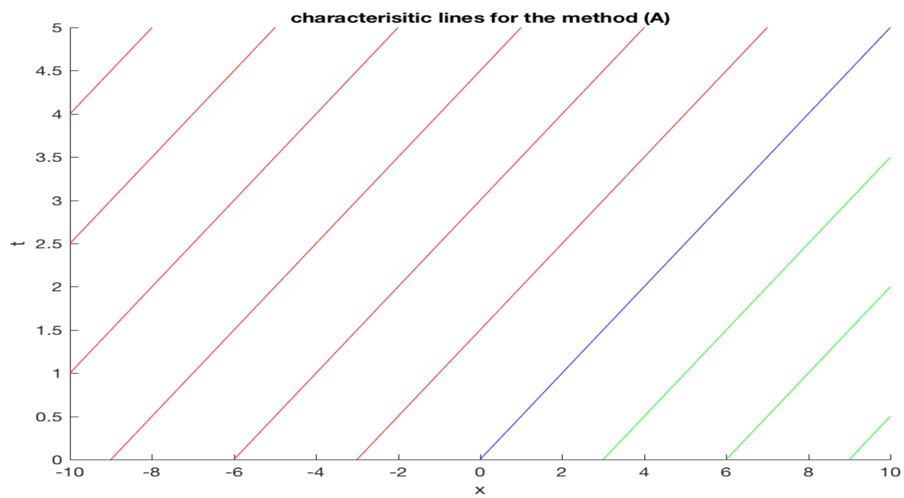


Figure 1: Plot of the characteristics for (A)

We plot the exact solution for the equation (A), we observe that the transport of the the initial wave is on the characteristic lines.

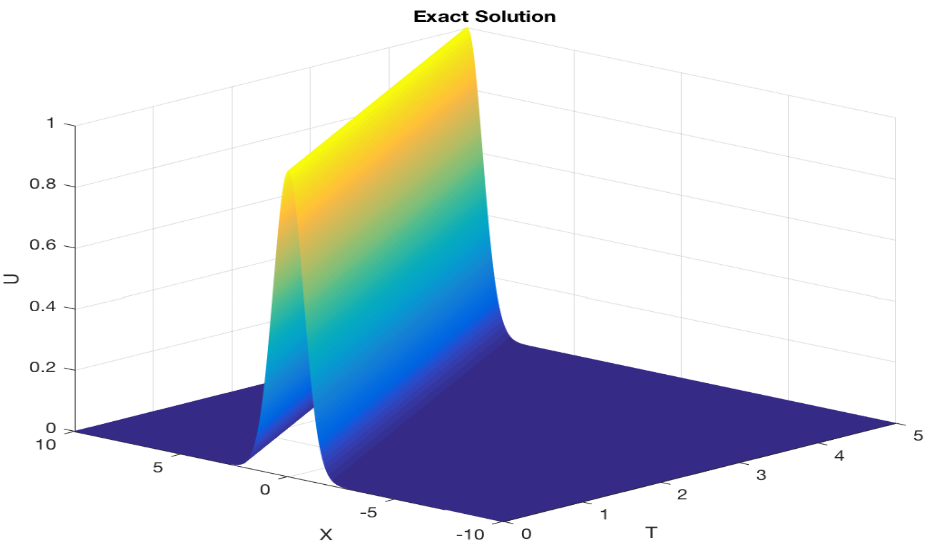


Figure 2: Plot of the exact solution for (A)

In the case of (B) we have that

$$\dot{x} = x \Rightarrow x = x_0 e^t \Rightarrow x_0 = xe^{-t}$$

so the characteristics on which u is constants is xe^{-t} so with the same procedure as above we have that the exact solution is :

$$u(x,t) = e^{-(xe^{-t})^2}$$

In the last case (C) we re-write it as $u_t + xu_x = -u$ obtained by doing the derivative of xu in x. We can see that the difference between (B) and (C) is that (B) is a homogenous equation but (C) has the external source (sink in this case). Let's compute the solution of (C) and see the difference in the solution between (B) and (C): in this case the characteristics are the same of (B) since

$$\dot{x} = x \Rightarrow x_0 = xe^{-t}$$

but u is not constant on this characteristics because $\dot{u} = -u$ so

$$u = u_0 e^{-t}$$

in the same fashion as before $u_0 = e^{-(xe^{-t})^2}$ so the exact solution is:

$$u(x,t) = e^{-t} e^{-(xe^{-t})^2}$$

We have the same characteristics on method both methods now we draw them :

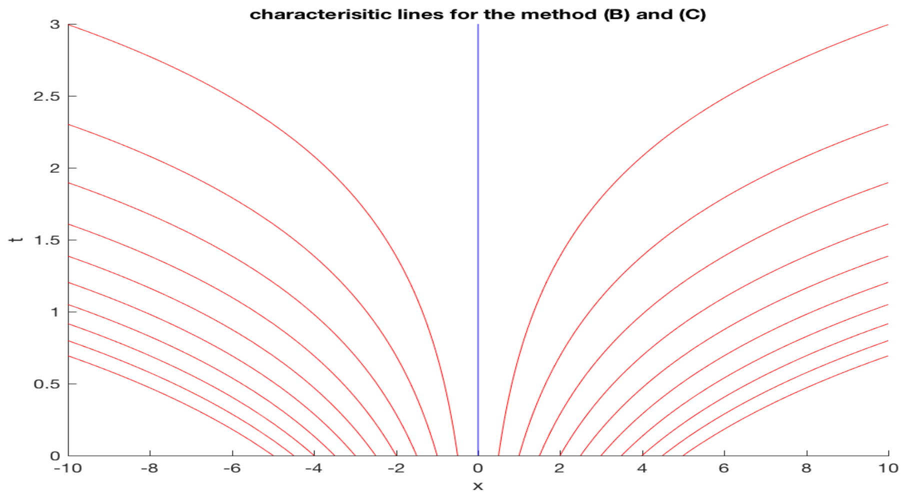
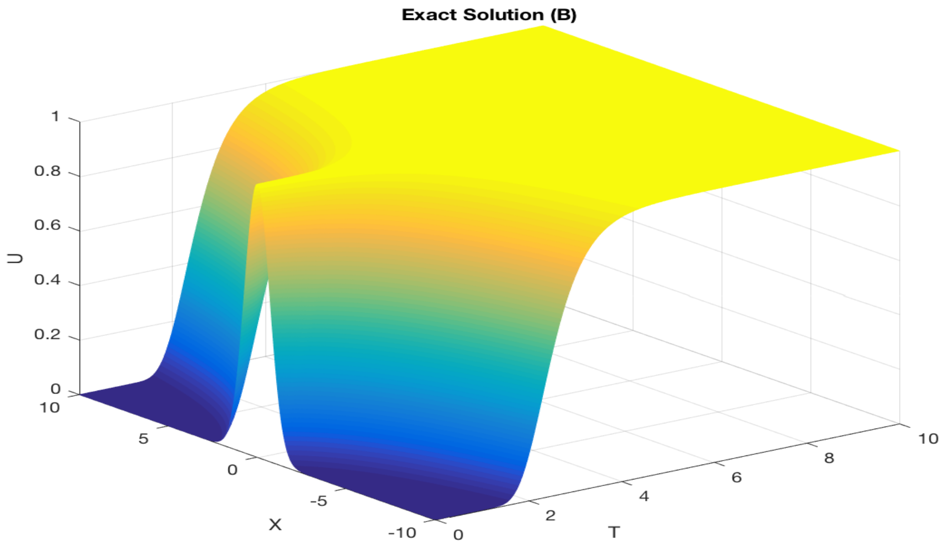


Figure 3: Plot of the characteristics for (B) and (C)

We can see that, in the case of (B), the values near 0 are all spread all over the space and the information on $x = 0$ is maintained over time always on x=0 so we expect that after $t = \tau$ for some $\tau > 0$ we can approximately say that the solution is constant $= u(0,0)$ since our initial data is continuous, this is because the characteristics near $x = 0$ spreads out really fast and if we are near enough to $x = 0$ the value of the initial function is $u(0,t) - \varepsilon = u(0,0) - \varepsilon$ for some $\varepsilon > 0$ due to the continuity of the ICs and since ε is arbitrary, follows our observation.
On the other hand the information transported on the characteristics is not constant for (C), it exponentially decrease in time, so we will see the solution loosing it's information and we can see that $u(x,t) \to 0$ for $t \to +\infty$. So in (B) the solution spreads out maintaining it's information but in (C) it spreads out loosing it's information due to the external sink.
We observe all we said above in a 3D plot Figure 4. In the solution of (C) we see a small bump near $t = 2$, it's due to the wave transported, near $x = 10$ initially the solution is very near to zero but near $t = 2$ information has reached this point and is not completely vanished so it creates a small bump. In the solution of (B) we can clearly see how the information is spreading over the space.



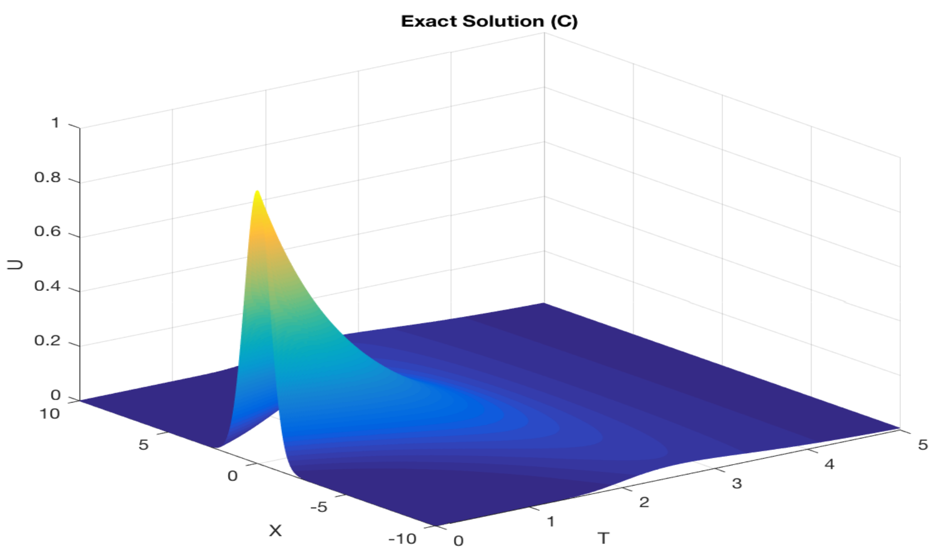


Figure 4: Plot of the exact solution for (B) and (C)

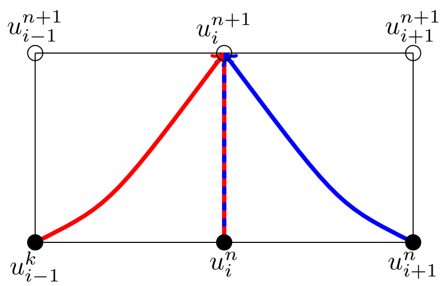
Numerical Implementation

UpWind

We start by approximating the solution with the Up-Wind scheme, since we cannot work on the whole space we will use $x \in [-5, 5]$ because with our ICs the function is near zero near the border so approximable with the whole plane for the initial condition, since our interval is limited we will be needing some boundary conditions to solve the problem, we already have the exact solution so we can impose the boundary condition with our solution. We could have also used boundary conditions and as soon as the wave leaves our domain it will enter from the other side and we might study it as a translation of our initial interval, another idea could be using null Neumann conditions. The Up-Wind Scheme we developed for a generic equation $u_t + f(x)u_x = 0$ is the following :

$$u_i^{n+1} = u_i^n - \frac{\Delta x}{\Delta y}\frac{f(x_i)}{2}(u_{i+1}^n - u_{i-1}^n) + \frac{\Delta x}{\Delta y}\frac{|f(x_i)|}{2}(u_{i+1}^n - 2u_i^n + u_{i-1}^n)$$

where $u_i^n = u(x_i, t_n)$. We wrote it this way so it may adjust itself in the basis of the sign of $f(x)$ (if $f(x) < 0$ the wind blows in the opposite direction of the case with $f(x) > 0$). We will modify it a little for (C) since it has a non homogenous part. The computation molecule is :



The red one is when $f(x) > 0$ and blue is when $f(x) < 0$, and we already know that this method has a convergence order of $O(\Delta x)$.

We start with case (A) :
We prefer not using the matrix because explicitly writing the scheme makes clear how are we imposing the BCs and how is the scheme working. For the convergence of the method we meed to impose some restriction on the choice of Δt, we will impose the CFL condition $\Delta t \leq \frac{\Delta x}{|a|}$ where a is the coefficient in front of u_x. We will be using $\Delta t = \frac{\Delta x}{|a|}$ with $a = 2$. The approximation with UpWind is very good in this case, as we can see using a $\Delta x = 0.02$ that the error is of 0.0172. We report various errors for different Δx :

Table 1: Error table for (A) with UpWind

Δx	0.05	0.04	0.02	0.01	0.005
Error	0.0428	0.0343	0.0172	0.0086	0.0043

We can see that the convergence order is $O(\Delta x)$ as we said before, we don't even need the log plot in this case as we can see that dividing Δx by two the error is also divided by two.

We plot the approximation for $t = 0, 0.5, 1$ and 2 in Figure 5:

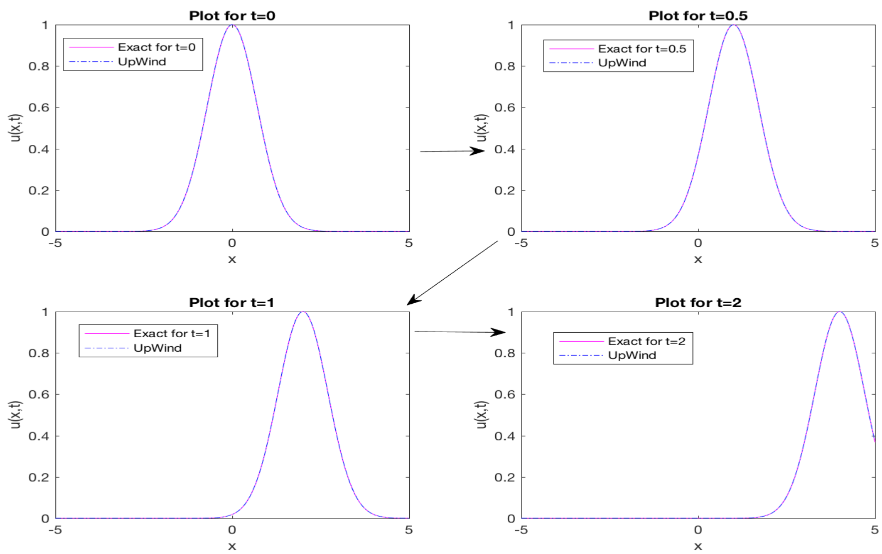


Figure 5: Plot of the exact and numerical solution for (A) with t=0, 0.5, 1 and 2, $\Delta x = 0.02$

Case (B):
Let's see what happens for (B), the global CFL condition for a non constant velocity term is

$$\Delta t \leq \frac{\Delta x}{\max_{x\in[-5,5]} |f(x)|}$$

since $f(x) = x$ and $\max_{x\in[-5,5]} x = 5$, so the CFL condition is $\Delta t \leq \Delta x/5$.
Let's write down the error table :

Table 2: Error table for (B) with UpWind

Δx	0.05	0.04	0.02	0.01	0.005
Error	0.0088	0.0071	0.0036	0.0018	$9.2\cdot 10^{-4}$

It is clear that the convergence order is also experimentally $O(\Delta x)$. One might think that somehow (B) is being approximated a lot better than (A) but it's not exactly the truth because of the change in CFL condition, now when we fix a Δx, the Δt in (A) is bigger than the one in (B), so the scheme is more accurate in (B). In the Figure 6 we plot the solution for some fixed t and the arrows indicate how the time is flowing in our plot. We can see the behaviour mentioned before of the wave which is "expanding" in time due the different velocity of every point but the peak is being maintained (information is not leaking).

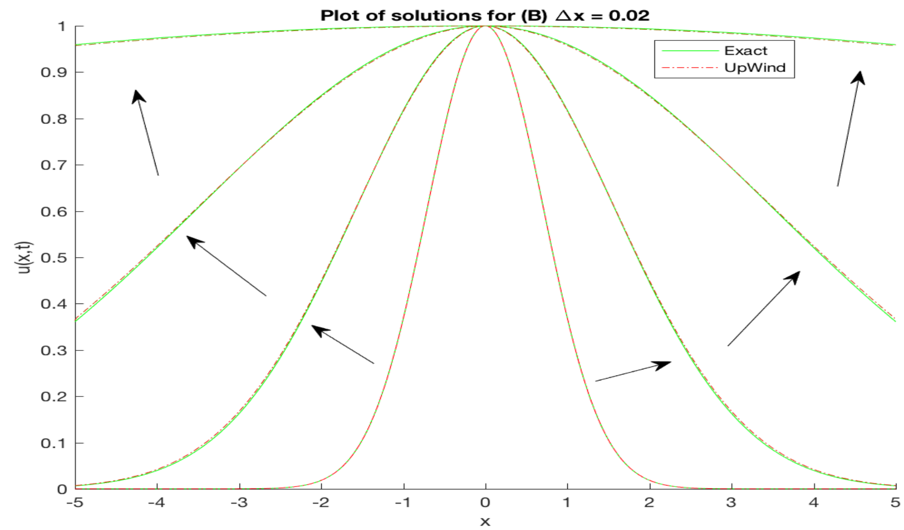


Figure 6: Plot of the exact and numerical solution for (B) with t=0, 0.8, 1.6 and 3.2, $\Delta x = 0.02$

Case (C):
The CFL condition is the same as case (B) but now we choose interval [-10,10], we choose x high enough to observe the solution on the "whole line", now the CFL condition is $\Delta t \leq \Delta x/10$, for the external sink with a naive approach we wrote it like $(u_{i+1}^n + u_i^n)/2$ which is the upwind way writing it. We plot in Figure 7 the numerical and exact solution:

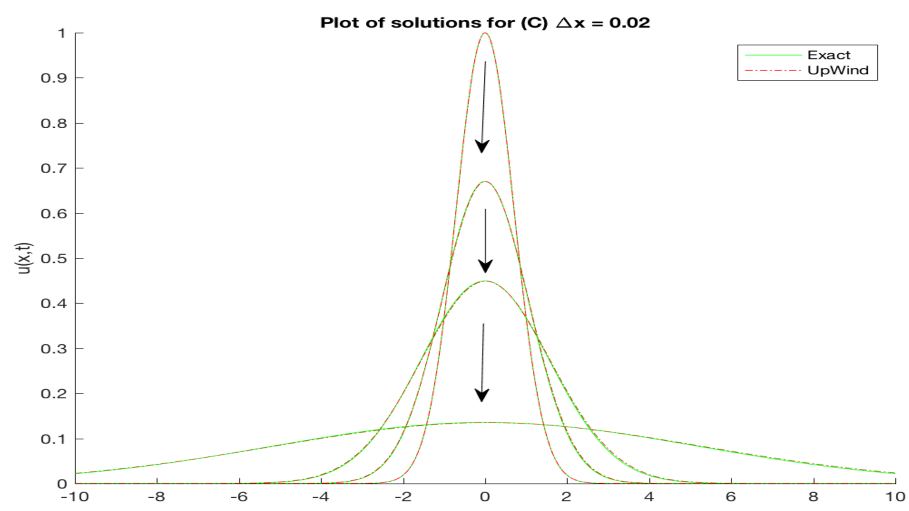


Figure 7: Plot of the exact and numerical solution for (C) with t=0, 0.4, 0.8 and 2, $\Delta x = 0.02$

We observe that numerically that the order of convergence is still $O(\Delta x)$ (can be seen in the table), here is the table of error:

Table 3: Error table for (C) with UpWind

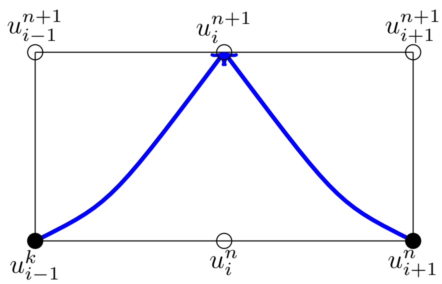
Δx	0.05	0.04	0.02	0.01	0.005
Error	0.0093	0.0075	0.0038	0.0019	$9.6 \cdot 10^{-4}$

Lax- Friederichs

We start by writing down the scheme, which we will call LF from now on, for $u_t + f(x)u_x = 0$:

$$u_i^{n+1} = \frac{1}{2}(u_{i+1}^n + u_{i-1}^n) - \frac{\Delta t}{2\Delta x} f(x_i)(u_{i+1}^n - u_{i-1}^n)$$

The computational molecule is:



The BCs and the interval will be the same as in UpWind so we can confront the two scheme easily. As already seen in lesson the convergence order is $O(\Delta x^2)$ but the scheme is very dissipative, so choosing a small Δt increases the dissipation (this fact can be seen on the locus plot we did in class). Let's start studying the three cases.
Case (A):
We plot the approximation with $\Delta x = 0.02$ in Figure 8. Here we write down the table of errors:

Table 4: Error table for (A) with LF

Δx	0.05	0.04	0.02	0.01	0.005
Error	0.0428	0.0343	0.017	0.0086	0.0043

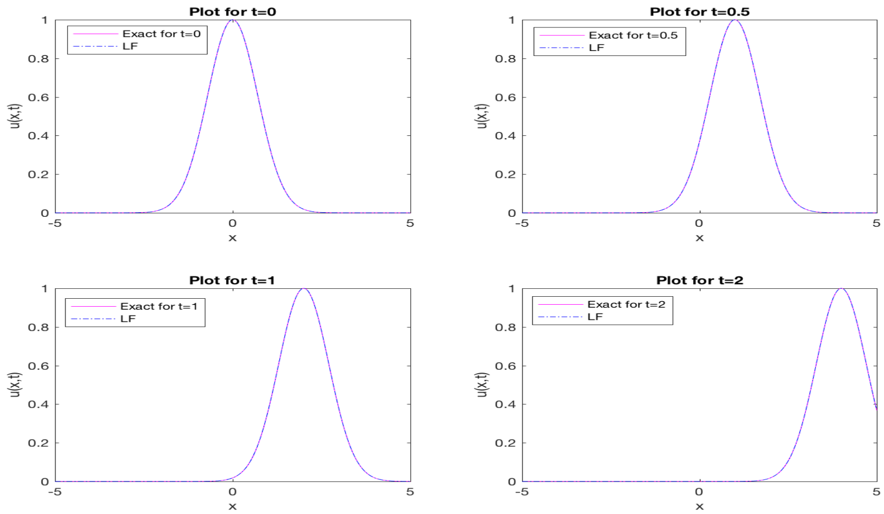


Figure 8: Plot of the exact and numerical solution for (A) with t=0, 0.8, 1.6 and 3.2, $\Delta x = 0.02$

Case (B):
For the plot we choose $\Delta x = 0.005$ from the plot (Figure 9) we can see the dissipation effect we mentioned before, observe the peak that is slowly falling down. We chose such a low Δx, this way the dissipative property was lower.

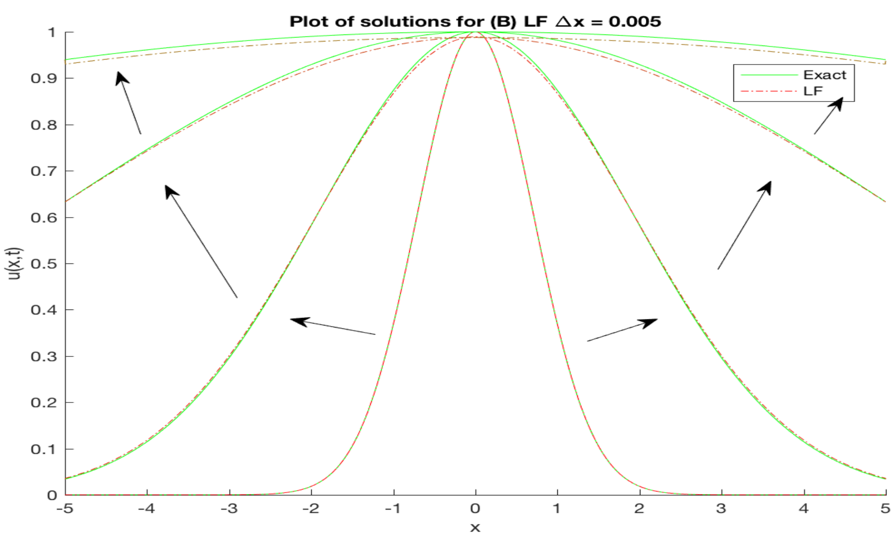


Figure 9: Plot of the exact and numerical solution for (B) with t=0, 1, 2 and 3, $\Delta x = 0.005$

We write down the error table:

Table 5: Error table for (B) with LF

Δx	0.05	0.04	0.02	0.01	0.005
Error	0.11	0.0872	0.0465	0.0241	0.0123

Case (C):
To plot we choose $\Delta x = 0.01$, even here the dissipative property can be seen(Figure 10):

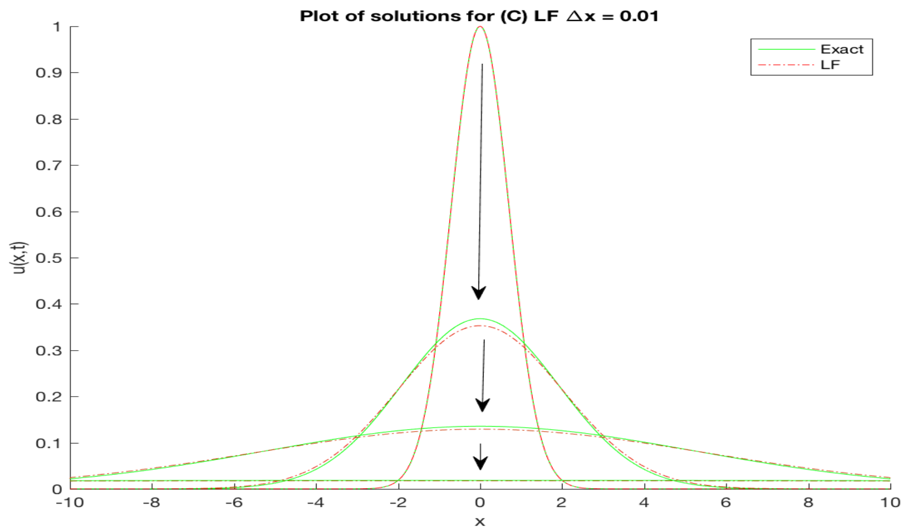


Figure 10: Plot of the exact and numerical solution for (C) with t=0, 1, 2 and 4, $\Delta x = 0.01$

Table 6: Error table for (C) with LF

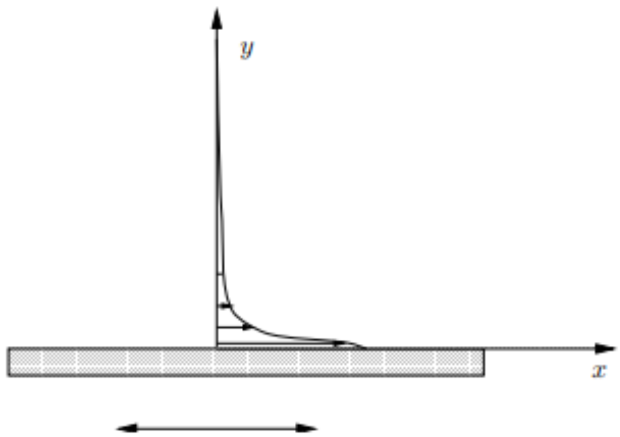
Δx	0.05	0.04	0.02	0.01	0.005
Error	0.076	0.0628	0.0342	0.018	0.009

We conclude this exercise saying that in these particular equations the UpWind methods seem better than LF, first of all because of it's non dissipative property and also we can see from the error tables that with UpWind we have less error than LF, except for the exercise (A) where they are practically the same. Another thing we observe is that numerically the error of LF has order $O(\Delta x)$ instead if $O(\Delta x^2)$, because we are changing Δt along with Δx and theoretically the error is of order $O(\Delta t + \Delta x^2)$(not only$O(\Delta x^2)$), and Δt is changing linearly when we change Δx so the error order is $O(\Delta t)$ which is linear.

Esercizio 2

Soluzione esatta

Prendiamo ora in considerazione il flusso generato sul piano (x, y) da una superficie oscillante, come in figura:



Il flusso generato presenta un campo vettoriale per la velocità della forma $\mathrm{u} = (\mathrm{u}(\mathrm{y};\ \mathrm{t});\ 0;\ 0)^t$. Lungo l'asse x ha la forma $\mathrm{u}(0;\ \mathrm{t}) = \mathrm{U}\cos(\omega\ \mathrm{t})$, con ω, U costanti. Inoltre supponiamo che $u \to 0$, per $y \to\ +\infty$.

Per ricavare analiticamente il vettore velocità utilizzo le equazioni di Navier Stokes per i fluidi incomprimibili. Prima però è possibile fare varie supposizioni teoriche per semplificare la risoluzione.

Innanzitutto, dato che per y elevate la velocità tende a zero, possiamo supporre che il gradiente della pressione lungo la x valga zero. Inoltre possiamo supporre $\mathrm{u}(\mathrm{y};\ \mathrm{t})\ =\ \mathrm{Re}(\mathrm{f}(\mathrm{y})\mathrm{e}^{i\omega t})$, con f funzione complessa.

Calcolo ora la soluzione esatta, per farlo innanzitutto scrivo le equazioni di Navier Stokes:

$$\begin{cases} \nabla \cdot \boldsymbol{u} = 0 \\ \rho\left(\dfrac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u}\right) = -\nabla p + \mu \Delta \boldsymbol{u} \end{cases}.$$

Aggiungendo anche le condizioni iniziali e al bordo.

Facendo tutte le semplificazioni scritte in precedenza si ottiene un sistema di questo tipo:

$$\begin{cases} \partial \mathrm{p}/\partial y = 0 \\ \rho\, \partial \mathrm{u}/\partial t = \mu\, \partial^2 \mathrm{u}/\partial y^2 \end{cases}.$$

Infatti, dato che $\mathrm{u} = (\mathrm{u}(\mathrm{y};\ \mathrm{t});\ 0;\ 0)^t$, allora il termine $\boldsymbol{u} \cdot \nabla \boldsymbol{u}$ è nullo per ogni tempo in tutti i punti del piano. Analogamente la divergenza è sempre nulla, così come anche tutti i termini del laplaciano non considerati.

Poniamo ora $\nu = \mu/\rho$ e analizziamo solo la seconda equazione:

$$\partial \mathrm{u}/\partial t = \nu\, \partial^2 \mathrm{u}/\partial y^2$$

Sostituiamo ora u con $\mathrm{Re}(\mathrm{f}(\mathrm{y})\mathrm{e}^{i\omega t})$. Secondo questa supposizione f è una funzione complessa, che possiamo scrivere come $f(y) = h(y) + ig(y)$. Ricaviamo: $\mathrm{Re}\big(\mathrm{f}(\mathrm{y})\mathrm{e}^{i\omega t}\big) = \mathrm{h}(\mathrm{y})\cos(\omega t) - \mathrm{g}(y) sin(\omega t)$. Ottengo:

$$\omega \mathrm{h}(\mathrm{y}) \sin(\omega t) + \omega \mathrm{g}(y) cos(\omega t) + \nu h''(y) \cos(\omega t) - \nu g''(y) \sin(\omega t) = 0$$

$$(\omega \mathrm{h}(\mathrm{y}) - \nu g''(y)) \sin(\omega t) + (\omega \mathrm{g}(y) + \nu h''(y)\,) \cos(\omega t) = 0$$

Affinché l'uguaglianza valga per tutti i tempi è necessario che i due coefficienti dipendenti da y valgano 0.

$$\begin{cases} \omega \mathrm{h}(y) - \nu g''(y) = 0 \\ \omega \mathrm{g}(y) + \nu h''(y) = 0 \end{cases}$$

Derivo due volte la prima equazione e ottengo:

$$\begin{cases} \omega \mathrm{h}''(y) = \nu g''''(y) \\ \omega \mathrm{g}(y) = -\nu h''(y) \end{cases}$$

Definisco ora $k = \sqrt{\omega/(2\nu)}$ e sostituisco la seconda equazione nella prima.

$$\begin{cases} -4\mathrm{k}^4 \, \mathrm{g}(y) = \nu \mathrm{g}''''(y) \\ 2\mathrm{k}^2 \, \mathrm{g}(y) = -h''(y) \end{cases}$$

Poniamo ora le seguenti condizioni al bordo: $g(0) = 0, g(+\infty) = 0$. Queste condizioni sono ricavate dalle condizioni al bordo di u, infatti $u(0,t) = Ucos(\omega t)$, quindi dato che g è moltiplicata per $\sin(\omega t)$, ricaviamo la condizione al bordo in 0. Mentre la condizione $u \to 0$, per $y \to +\infty$ impone quella a ∞.

Ottengo quindi la soluzione: $g(y) = Ae^{-ky}sin(ky) + Be^{-ky}cos(ky) + Ce^{ky}sin(ky) + De^{ky}cos(ky)$, dato che g deve essere una funzione reale. Data la condizione a infinito ricavo che C=D=0. Data la condizione a zero, ricavo B=0. Quindi:

$$g(y) = Ae^{-ky}sin(ky)$$

$$g'(y) = -kAe^{-ky}sin(ky) + kAe^{-ky}cos(ky)$$

$$g''(y) = -2k^2Ae^{-ky}cos(ky)$$

Sostituendo g'' in quella che era la prima equazione, prima di essere derivata due volte, ottengo:

$$h(y) = \frac{g''(y)}{2k^2} = -Ae^{-ky}cos(ky)$$

Perciò:

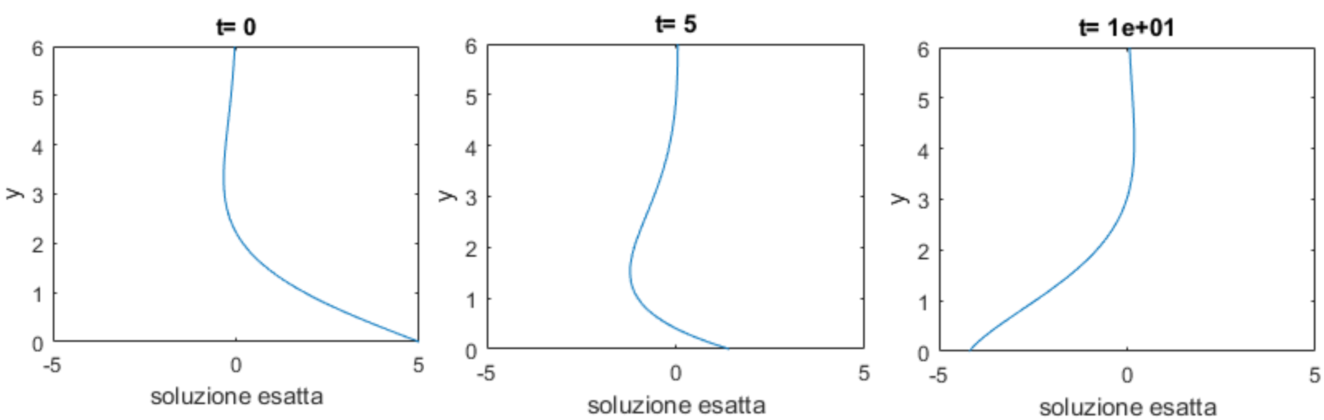
$$f(y) = Ae^{-ky}(-\cos(ky) + isin(ky))$$

Dato che $\mathrm{f}(0) = \mathrm{U}$, si ricava che A=-U. Si ottiene quindi la soluzione esatta al problema:

$$u(y,t) = Ue^{-ky}\cos(ky)\cos(\omega t) + Ue^{-ky}\,sin(ky)sin(\omega t)$$

$$u(y,t) = Ue^{-ky}\cos(\omega t - ky)$$

Pongo ora $\omega = 1, U = 5$ e mostro come si presenta la soluzione esatta, in vari tempi, nel piano $(u(y), y)$.



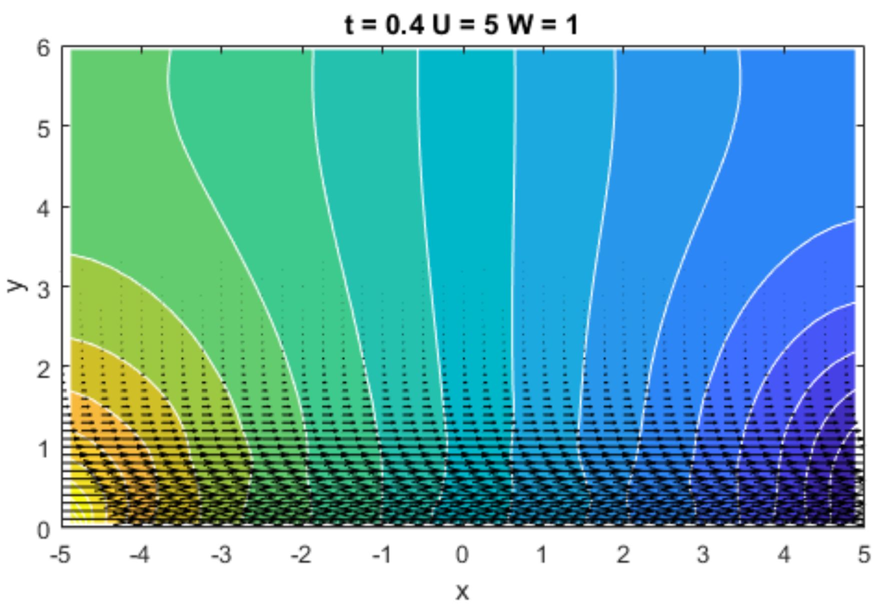
Soluzione numerica

Per risolvere numericamente il problema vado a modificare il codice analizzato durante il corso per la risoluzione numerica delle equazioni di Navier Stokes.

Il dominio sarà $\Omega = (-5,5)\mathrm{X}(0,6)$. Le condizioni al bordo che avremo saranno di Dirichlet per la velocità lungo l'asse y. Infatti sarà 0 in ogni istante, su tutti i lati. Mentre per la velocità lungo l'asse x avremo condizioni di Dirichlet sul bordo sud e nord (date dal movimento del piatto e dal valore limite di u per y che tende a infinito) e condizioni di Neumann sui bordi est e ovest.

Le principali differenze col caso precedente stanno nelle condizioni al bordo: infatti l'avere condizioni di Neumann comporta l'avere più valori incogniti, perciò tutte le matrici utilizzate dal programma vanno ridimensionate per risolvere anche questi valori. Soprattutto nel momento in cui si utilizza la griglia staggered è necessario porre estrema attenzione alle dimensioni delle matrici. Per cui è stato necessario verificare quali nodi erano implicati in ogni passaggio, in modo da adattare le matrici utilizzate.

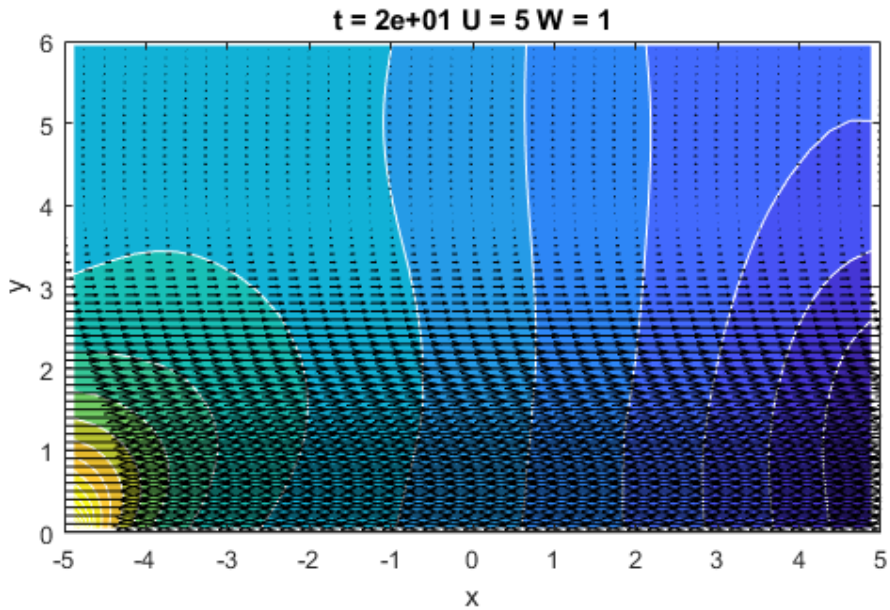
Ora analizzo il comportamento della soluzione numerica. Come nel caso precedente pongo $\omega = 1, U = 5$.



Già da questa figura si possono fare molte osservazioni sul comportamento della soluzione numerica. Innanzitutto la pressione non è costante, a differenza della soluzione esatta. In realtà, osservando i valori esatti della pressione in ogni nodo, si nota che oscillano tutti intorno a 10e-15, quindi questa variazione diventa trascurabile e in linea col risultato teorico.

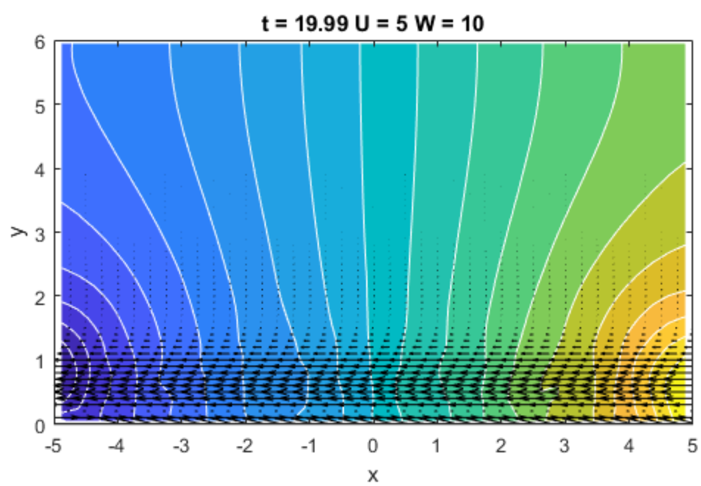
In secondo luogo si nota che la velocità vale zero per tutti i punti aventi $y \geq 4$, mentre per la soluzione esatta ciò non può accadere. La causa di questa discrepanza sta nei dati iniziali forniti al calcolatore. Infatti abbiamo posto inizialmente le matrici U e V nulle, mentre la soluzione esatta, all'istante 0, non è nulla (come si può osservare nei grafici mostrati in precedenza). Diventa quindi importante fare in modo che l'errore tra soluzione numerica e soluzione esatta non sia calcolato fin dall'istante 0, ma almeno dopo in certo intervallo di tempo (che possiamo supporre dipenda principalmente dalla viscosità del fluido).

Osserviamo inoltre che la velocità lungo l'asse y sembra nulla, infatti i valori calcolati tendono alla precisione macchina, perciò possiamo affermare che anche questo risultato teorico è stato rispettato dall'implementazione numerica.



Ad un tempo successivo la soluzione si presenta in questa forma: osserviamo che la soluzione è definita su tutti i punti, in particolare intorno a $y = 4$ si nota un cambio di fase. Inoltre l'errore, che è stato calcolato per i tempi maggiori di 4, vale 0.0708.

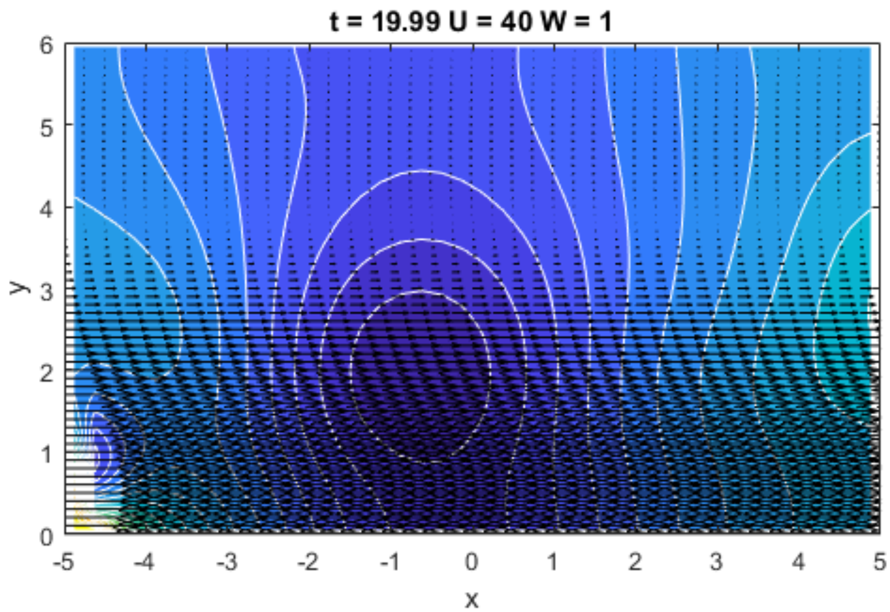
Pongo ora $\omega = 10, U = 5$. Osserviamo come varia la soluzione e l'errore rispetto al caso precedente:



Si osserva subito che, aumentando ω, aumenta anche il coefficiente k. Ciò implica sia una maggiore frequenza temporale della sinusoide del dato iniziale, sia una più veloce decrescita del modulo della velocità. Quest'ultima osservazione, è estremamente evidente dal grafico mostrato, mentre per la prima sarebbe meglio osservare l'evoluzione nel tempo del sistema.

L'ordine di grandezza dell'errore in questo caso è simile al caso precedente, ovvero l'errore vale 0.0620.

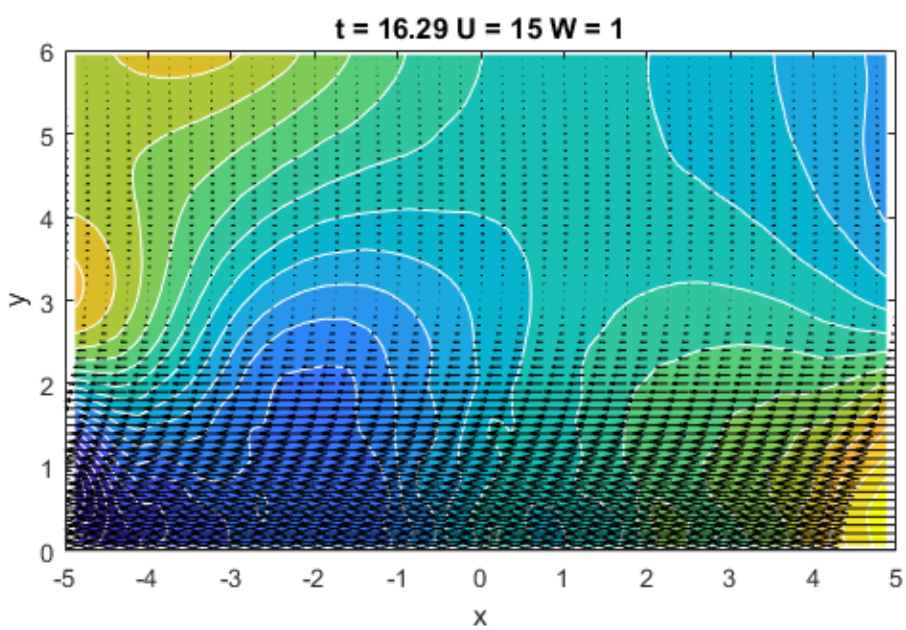
Infine osserviamo come varia il comportamento della soluzione numerica variando U, prendiamo quindi questi valori per le incognite: $\omega = 1, U = 50$.



Notiamo che le sinusoidi che sono generate dal campo vettoriale hanno ampiezza molto maggiore rispetto a tutti i casi precedenti. Inoltre la pressione varia molto più velocemente rispetto ai casi precedenti e presenta valori più alti, intorno a 1e-11. Infine l'errore è 10 volte i casi precedenti: in questo caso vale 0.5664.

La variazione di U è sicuramente la causa di tutti questi fenomeni. Suppongo che avendo diminuito la decaduta della soluzione esatta, essa assume un valore molto diverso da 0 al limite superiore, quindi la condizione la bordo diventa errata, portando ad un aumento dell'errore.

Suppongo quindi che l'errore cresca linearmente con U, per verificare questa tesi pongo $\omega = 1, U = 15$.



Il comportamento è simile al caso precedente, inoltre l'errore vale 0.2124, come supposto. Ipotizzo che scegliendo un dominio Ω con una base di lunghezza più adatta (quindi almeno lo stesso ordine di grandezza), l'errore diminuisca fino ad arrivare all'ordine di 0,01.

MATLAB Codes

Exercise 1 Code

Esericizio 1 (A)

```
clear all, close all

%soluzione con schemi alle DF esplicite del
%pb lineare di trasporto
%du/dt+a*du/dx=0 +u(x,0)=u0(x) + BCs , a=2

a=2; %velocita di propagazione

T=5;
x0=-5;  xf=5;
dx=1/25;
dt_cr=dx/abs(a); %valore critico di dt (condiz CFL)

dt=1*dt_cr;

nT=ceil(T/dt);
nX=(xf-x0)/dx;
x=linspace(x0,xf,nX+1);
t=linspace(0,T,nT+1);


%ICs
u0=@(x) 0.*x + exp(-(x).^2).*(and((x>=x0),(x<=xf)));
uex=@(t,x) u0(x-a*t);

lambda=dt/dx;
u=zeros(nX+1,nT+1);
u(:,1)=u0(x); %ICs
metodo='LF';
switch (metodo)
    case{'upwind'}
```

```
            disp('metodo--->upwind');
            for n=1:nT
                 u(1,n+1)=uex((n+1)*dt,x0);
                 u(nX+1,n+1)=uex((n+1)*dt,xf);
                for i=2:nX
                    u(i,n+1)=u(i,n)-lambda*a/2*(u(i+1,n)-u(i-1,n))...
                        +lambda*abs(a)/2*(u(i+1,n)-2*u(i,n)+u(i-1,n));
                end
             err(n)=(max(abs(uex(n*dt,x)-u(:,n)')));
             end
        case{'LF'}
            disp('metodo--->Lax-Friedrichs');
            for n=1:nT
                u(1,n+1)=uex((n+1)*dt,x0);
                u(nX+1,n+1)= uex((n+1)*dt,xf);
                for i=2:nX
                    u(i,n+1)=1/2*(u(i+1,n)+u(i-1,n))...
                        -lambda*a/2*(u(i+1,n)-u(i-1,n));
                end
                err(n)=(max(abs(uex(n*dt,x)-u(:,n)')));
             end
end %switch
errore=max(err)
% plot
figure,
subplot(2,2,1)
time=(0)*dt;
 plot(x,uex(time,x),'m-'), hold on
 %traslo avanti di uno perche parto da 1
 plot(x,u(:,1),'b-.');
 legend('Exact for t=0','LF')
 xlabel('x')
 ylabel('u(x,t)')
 title('Plot for t=0')

 subplot(2,2,2)
time=(50)*dt;
 plot(x,uex(time,x),'m-'), hold on
 %traslo avanti di uno perche parto da 1
 plot(x,u(:,51),'b-.');
 legend('Exact for t=0.5','LF')
 xlabel('x')
 ylabel('u(x,t)')
 title('Plot for t=0.5')

subplot(2,2,3)
```

```
title('Plot for t=1')
 time=(100)*dt;
 plot(x,uex(time,x),'m-'), hold on
 %traslo avanti di uno perche parto da 1
 plot(x,u(:,101),'b-.');
 legend('Exact for t=1','LF')
 xlabel('x')
 ylabel('u(x,t)')
 title('Plot for t=1')

 time=(200)*dt;
 subplot(2,2,4)

 plot(x,uex(time,x),'m-'), hold on
 %traslo avanti di uno perche parto da 1
 plot(x,u(:,201),'b-.');
 legend('Exact for t=2','LF')
 xlabel('x')
 ylabel('u(x,t)')
 title('Plot for t=2')

%  [Y,X]=meshgrid(t,x);
%  mesh(Y,X,uex(Y,X)),
%  title('Exact Solution')
%  ylabel('X')
%  xlabel('T')
%  zlabel('U(t,x)')
%  figure,
%  mesh(Y,X,u)
%  title('Numerical Solution')
%  ylabel('X')
%  xlabel('T')
%  zlabel('U(t,x)')

% figure, hold on
% plot(x,x/a,'b')
% for i=1:10
%     plot(x,x/a+3*i/2,'r')
%     plot(x,x/a-i*3/2,'g')
% end
% axis([-10 10 0 5])
% xlabel('x')
% ylabel('t')
% title('characterisitic lines for the method (A)')
```

Esericizio 1 (B)

```
clear all, close all

%soluzione con schemi alle DF esplicite del
%pb lineare di trasporto
%du/dt+x*du/dx=0 +u(x,0)=u0(x)=e^(-x^2) + BCs drichlet esatte


T=5;
x0=-5;  xf=5;
dx=1/200;
dt_cr=dx/max(abs(x0),abs(xf)); %valore critico di dt (condiz CFL)

dt=1*dt_cr;

nT=ceil(T/dt);
nX=(xf-x0)/dx;
x=linspace(x0,xf,nX+1);
t=linspace(0,T,nT+1);


%ICs
u0=@(x) 0.*x + exp(-(x).^2).*(and((x>=x0),(x<=xf)));
uex=@(t,x) u0(x.*exp(-t));

lambda=dt/dx;
u=zeros(nX+1,nT+1);
u(:,1)=u0(x); %ICs
metodo='LF';
switch (metodo)
    case{'upwind'}
         disp('metodo--->upwind');
         for n=1:nT
             u(1,n+1)=uex((n+1)*dt,x0);
             u(nX+1,n+1)=uex((n+1)*dt,xf);
            for i=2:nX
                u(i,n+1)=u(i,n)-lambda*x(i)/2*(u(i+1,n)-u(i-1,n))...
                    +lambda*abs(x(i))/2*(u(i+1,n)-2*u(i,n)+u(i-1,n));
                %keyboard
            end
            %BCs
         err(n)=(max(abs(uex(n*dt,x)-u(:,n)')));
         end
    case{'LF'}
         disp('metodo--->Lax-Friedrichs');
         for n=1:nT
            u(1,n+1)=uex((n+1)*dt,x0);
```

```
                u(nX+1,n+1)= uex((n+1)*dt,xf);
                for i=2:nX
                    u(i,n+1)=1/2*(u(i+1,n)+u(i-1,n))...
                        -lambda*x(i)/2*(u(i+1,n)-u(i-1,n));
                    %keyboard
                end
                err(n)=(max(abs(uex(n*dt,x)-u(:,n)')));
            end
end %switch
errore=max(err)

time=(0)*dt;
hold on
%traslo avanti di uno perche parto da 1
plot(x,uex(time,x),'g-')
plot(x,u(:,1),'-.','Color',[1 0 0]);
legend('Exact Solution','LF')
xlabel('x')
ylabel('u(x,t)')

time = 1000*dt;
%traslo avanti di uno perche parto da 1
plot(x,uex(time,x),'g-')
plot(x,u(:,1001),'-.','Color',[0.9 0.1 0])
legend('Exact','LF')
%
time = 2000*dt;
%traslo avanti di uno perche parto da 1
plot(x,uex(time,x),'g-')
plot(x,u(:,2001),'-.','Color',[0.8 0.2 0])
legend('Exact','LF')

time = 3000*dt;
%traslo avanti di uno perche parto da 1
plot(x,uex(time,x),'g-')
plot(x,u(:,3001),'-.','Color',[0.6 0.4 0])
legend('Exact','LF')

% [Y,X]=meshgrid(t,x);
% mesh(Y,X,uex(Y,X)),
% hold on,
% ylabel('X')
% xlabel('T')
% zlabel('U')
```

Esericizio 1 (C)

```
clear all, close all

%soluzione con schemi alle DF esplicite del
%pb lineare di trasporto
%du/dt+x*du/dx=-u +u(x,0)=u0(x)=e^(-x^2) + BCs drichlet esatte

T=5;
x0=-10;  xf=10;
dx=1/20;
dt_cr=dx/max(abs(x0),abs(xf)); %valore critico di dt (condiz CFL)

dt=1*dt_cr;

nT=ceil(T/dt);
nX=(xf-x0)/dx;
x=linspace(x0,xf,nX+1);
t=linspace(0,T,nT+1);


%ICs
u0=@(x) 0.*x + exp(-(x).^2).*(and((x>=x0),(x<=xf)));
uex=@(t,x) exp(-t).*u0(x.*exp(-t));

lambda=dt/dx;
u=zeros(nX+1,nT+1);
u(:,1)=u0(x); %ICs
metodo='LF';
switch (metodo)
    case{'upwind'}
        disp('metodo--->upwind');
        for n=1:nT
            u(1,n+1)=uex((n+1)*dt,x0);
            u(nX+1,n+1)=uex((n+1)*dt,xf);
            for i=2:nX
                u(i,n+1)=u(i,n)-lambda*x(i)/2*(u(i+1,n)-u(i-1,n))...
                    +lambda*abs(x(i))/2*(u(i+1,n)-2*u(i,n)+u(i-1,n))...
                    -dt/2*(u(i+1,n)+u(i,n));

                %keyboard
            end
            %BCs
        err(n)=(max(abs(uex(n*dt,x)-u(:,n)')));
        end
    case{'LF'}
        disp('metodo--->Lax-Friedrichs');
        for n=1:nT
```

```
            u(1,n+1)=uex((n+1)*dt,x0);
            u(nX+1,n+1)= uex((n+1)*dt,xf);
            for i=2:nX
                u(i,n+1)=1/2*(u(i+1,n)+u(i-1,n))...
                    -lambda*x(i)/2*(u(i+1,n)-u(i-1,n))...
                    -dt/2*(u(i+1,n)+u(i-1,n));
            end
            err(n)=(max(abs(uex(n*dt,x)-u(:,n)')));
        end
end %switch
errore=max(err)
% plot soluzione dopo 25 step temporali
% time=25*dt;
% plot(x,uex(time,x),'m-'), hold on
% plot(x,u(:,25),'b-.');
% legend('esatta','Lax-Friedrichs')
% xlabel('x')
% ylabel('t')


time=(0)*dt;
hold on
%traslo avanti di uno perche parto da 1
plot(x,uex(time,x),'g-')
plot(x,u(:,1),'-.','Color',[1 0 0]);
legend('Exact Solution','UpWind')
xlabel('x')
ylabel('u(x,t)')

time = 1000*dt;%t=0.4
%traslo avanti di uno perche parto da 1
plot(x,uex(time,x),'g-')
plot(x,u(:,1001),'-.','Color',[0.9 0.1 0])
legend('Exact','UpWind')
%
time = 2000*dt;%t=0.8
%traslo avanti di uno perche parto da 1
plot(x,uex(time,x),'g-')
plot(x,u(:,2001),'-.','Color',[0.8 0.2 0])
legend('Exact','LF')

time = 4000*dt;%t=2
%traslo avanti di uno perche parto da 1
plot(x,uex(time,x),'g-')
plot(x,u(:,4001),'-.','Color',[0.6 0.4 0])
legend('Exact','LF')
```

```
% [Y,X]=meshgrid(t,x);
% mesh(Y,X,uex(Y,X)),
% ylabel('X')
% xlabel('T')
% zlabel('U')

% figure, hold on
% plot(t*0,t,'b')
% for k=1:10
%     plot(x,log(x/(k/2)),'r')
% end
% axis([-10 10 0 T])
% xlabel('x')
% ylabel('t')
% title('characterisitic lines for the method (B) and (C)')
```

Exercise 2 Code

K1

```
function A = K1(n,h,a11)
%discretization with finite centered difference of the
%second derivative for both x and y
% a11: Neumann=1, Dirichlet=2, Dirichlet mid=3;
A = spdiags([-1 a11 0;ones(n-2,1)*[-1 2 -1];0 a11 -1],-1:1,n,n)'/h^2;
% spdiags creates a sparse diag Matrix
```

Average

```
function B=avg(A)


if nargin<2, k = 1; end
if size(A,1)==1, A = A'; end
if k<2, B = (A(2:end,:)+A(1:end-1,:))/2; else, B = avg(A,k-1); end
if size(A,2)==1, B = B'; end



%if(size(A,1)==1), A=A'; end %se A riga allora la traspongo in colonna


%B=(A(1:end-1,:)+A(2:end,:))/2; %media tra due elementi consecutivi per colonna

```

```
%if(size(A,2)==1), B=B'; end %se A era riga, la metto come prima

return
%It work with martix, column and row

%% EXAMPLE
% x=0:0.1:1;
% xa=avg(x);

%% OSS

%U_a=(avg(U'))' for an average per row, where U is matrix
%else it will to an average on column
```

Esercizio 2

```
clear all, close all
% -----------------------------------------------------------------------

Re = 1;          % Reynolds number
dt = 1e-2;       % time step
tf = 10e0;       % final time
lx = 5;          % half width of box
ly = 6;          % height of box
nx = 30;         % number of x-gridpoints
ny = 50;         % number of y-gridpoints
nsteps = 200;    % number of steps with graphic output
U0=5;
%U0=15;
%U0=40;
omega=1;
%omega=10;
kk=sqrt(omega*Re/2);
% -----------------------------------------------------------------------

nt = ceil(tf/dt); dt = tf/nt;        %aggiusto dt
x = linspace(-lx,lx,nx+1); hx = 2*lx/nx;
y = linspace(0,ly,ny+1); hy = ly/ny;
[X,Y] = meshgrid(y,x);

% soluzione esatta
uex=@(y,t) U0*exp(-kk*y).*cos(omega*t-kk*y);
% -----------------------------------------------------------------------
```

```
% initial conditions
U = zeros(nx+1,ny); V = zeros(nx,ny-1);
%dato che gli estremi est e ovest di U sono incogniti, e necessario
%aggiungerli alla nostra matrice soluzione

% boundary conditions
uN = x*0;          vN = avg(x)*0;
                   vS = avg(x)*0;
uW = avg(y)*0;   vW = y*0;
uE = avg(y)*0;   vE = y*0;
%-------------------------------------------------------------------------------
%condizioni al bordo per v
Vbc = dt/Re*([vS' zeros(nx,ny-3) vN']/hx^2+...
      [2*vW(2:end-1);zeros(nx-2,ny-1);2*vE(2:end-1)]/hy^2);

%-------------------------------------------------------------------------------
%Calcolo la soluzione esatta in alcuni istanti
tt=[0 0.5 5 10];
for r= 1: length(tt)
    t=tt(r);
    ex=uex(y,t);

    figure
    plot(ex, y), hold on,
    axis([-lx lx 0,ly]),
    xlabel('soluzione esatta'), ylabel('y'),
    title(sprintf('t= %0.1g',t)), hold off;

end

pause
clear t;
%-------------------------------------------------------------------------------
fprintf('initialization')

%Matrix for the pressure
Lp = kron(speye(ny),K1(nx,hx,1))+kron(K1(ny,hy,1),speye(nx));

Lp(1,1) = 3/2*Lp(1,1);
perp = symamd(Lp);
Rp = chol(Lp(perp,perp));
```

```matlab
Rpt = Rp';
Lu = speye((nx+1)*ny)+dt/Re*(kron(speye(ny),K1(nx+1,hx,1))+...
      kron(K1(ny,hy,3),speye(nx+1)));
peru = symamd(Lu);
Ru = chol(Lu(peru,peru));
Rut = Ru';

Lv = speye(nx*(ny-1))+dt/Re*(kron(speye(ny-1),K1(nx,hx,3))+...
      kron(K1(ny-1,hy,2),speye(nx)));
perv = symamd(Lv);
Rv = chol(Lv(perv,perv));
Rvt = Rv';

%vettore dell'errore
err=zeros(1,nt);
[A,B]=meshgrid(x,avg(y));

fprintf(', time loop\n---20%%---40%%---60%%---80%%--100%%\n')
for k = 1:nt
    time=(k-1)*dt;
    %la condizione al bordo varia ad ogni passo temporale
    uS = x*0+U0*cos(omega*time);

    % treat nonlinear terms
    Ubc = dt/Re*[2*uS' zeros(nx+1,ny-2) 2*uN']/hy^2;

    gamma = min(1.2*dt*max(max(max(abs(U)))/hx,max(max(abs(V)))/hy),1);
    %if gamma = 0 the we don't use UpWind
    %if gamma = 1 we use only UpWind
    %gamma is a CFL number!
    Ue = U;
    Ue = [2*uS'-Ue(:,1) Ue 2*uN'-Ue(:,end)];

    Ve = [vS' V vN'];
    Ve = [2*vW-Ve(1,:); Ve; 2*vE-Ve(end,:)];

    Ua = avg(Ue')'; Ud = diff(Ue')'/2;
    Va = avg(Ve);    Vd = diff(Ve)/2;
    UVx = diff(Ua.*Va-gamma*abs(Ua).*Vd)/hx;
    UVy = diff((Ua.*Va-gamma*Ud.*abs(Va))')'/hy;
    Ua = avg([Ue(2,2:end-1); Ue(:,2:end-1); Ue(end-1,2:end-1)]);
    Ud = diff([Ue(2,2:end-1); Ue(:,2:end-1); Ue(end-1,2:end-1)])/2;
    Va = avg(Ve(2:end-1,:)')';
    Vd = diff(Ve(2:end-1,:)')'/2;
    U2x = diff(Ua.^2-gamma*abs(Ua).*Ud)/hx;
    V2y = diff((Va.^2-gamma*abs(Va).*Vd)')'/hy;
```

```
    %abbiamo trattato U in questo modo perche U contiene righe (agli estremi) che non
    %interagiscono con V nel calcolo della soluzione

    U = U-dt*(UVy+U2x);
    V = V-dt*(UVx(:,2:end-1)+V2y);

    % implicit viscosity
    rhs = reshape(U+Ubc,[],1);%creates and decides itself the column size
    u(peru) = Ru\(Rut\rhs(peru));%we solve the system we described before
    % but using only one line!
    U = reshape(u,nx+1,ny);
    rhs = reshape(V+Vbc,[],1);
    v(perv) = Rv\(Rvt\rhs(perv));
    V = reshape(v,nx,ny-1);

    % pressure correction
    rhs = reshape(diff(U)/hx+diff([vS' V vN']')'/hy,[],1);
    p(perp) = -Rp\(Rpt\rhs(perp));
    P = reshape(p,nx,ny);
    U = U-[P(2,:)-P(1,:); diff(P);P(end,:)-P(end-1,:)]/hx;
    V = V-diff(P')'/hy;

    % visualization
    if floor(25*k/nt)>floor(25*(k-1)/nt), fprintf('|'), end
    if k==1||floor(nsteps*k/nt)>floor(nsteps*(k-1)/nt)

       clf, contourf(avg(x),avg(y),P',20,'w-'), hold on

       Ue = [uS' avg(U')' uN'];
       Ve = [vW;avg([vS' V vN']);vE];
      % Len = sqrt(Ue.^2+Ve.^2+eps);
      % quiver(x,y,(Ue./Len)',(Ve./Len)',.4,'k')
      quiver(x,y,(Ue)',(Ve)',4,'k') %tolgo normalizzazione lunghezze
       hold off, axis equal, axis([-lx lx 0 ly])
       p = sort(p); caxis(p([8 end-7]))
       title(sprintf('t = %0.4g U = %0.2g W = %0.2g',(k-1)*dt,U0,omega))
       xlabel('x'), ylabel('y'),
       drawnow

    end

  %analizzo l'errore
  if(time>=4)
```

```
  Uex=uex(B,time);
  err(1,k)=max(max(abs(Uex'-U)));
  end

end
fprintf('\n')
max(err)
%================================================================
```