# Laboratorio di Dinamica dei Fliudi
# Esercitazione 01 – a.a. 2008-2009

Dott. Simone Zuccher

15 Maggio 2009

**Nota**. Queste pagine potrebbero contenere degli errori: chi li trova è pregato di segnalarli all'autore (zuccher@sci.univr.it).

## 1 Risoluzione numerica delle equazioni dello strato limite 2D su lamina piana in corrente uniforme

Consideriamo il problema dello strato limite bidimensionale su una lamina piana per un fluido incomprimibile in coordinate cartesiane $(x, y)$

$$u_x + v_y = 0 \tag{1.1}$$
$$uu_x + vu_y = \nu u_{yy} \tag{1.2}$$

e le relative condizioni al contorno

$$u(x, 0) = v(x, 0) = 0, \qquad u(x, \infty) = 1$$

e condizioni iniziali

$$u(0, y) = 1, \qquad v(0, y) = 0.$$

Moltiplicando l'equazione di continuità per $u$ e sommandola all'equazione della quantità di moto, si ottiene il sistema in forma conservativa

$$u_x + v_y = 0 \tag{1.3}$$
$$(u^2)_x + (uv)_y = \nu u_{yy}, \tag{1.4}$$

che risulta più agevole da risolvere numericamente in quanto "assorbe" meglio la discontinuità della soluzione al bordo d'attacco $x = 0$. Si noti che le equazioni sono paraboliche in $x$. Pertanto, partendo dalla condizione iniziale a $x = x_{\text{in}}$ (posizione inziale), è possibile marciare in $x$ tramite uno schema esplicito fino a $x = x_{\text{fi}}$ (posizione finale).

Per quanto riguarda la discretizzazione, siccome i gradienti di velocità sono più forti in prossimità di $y = 0$ e $x = 0$, utilizziamo una grigia cartesiana $(x_i, y_j)$ non equispaziata con i punti maggiormente addensati in prossimità di tali zone (si veda la figura 1). Per

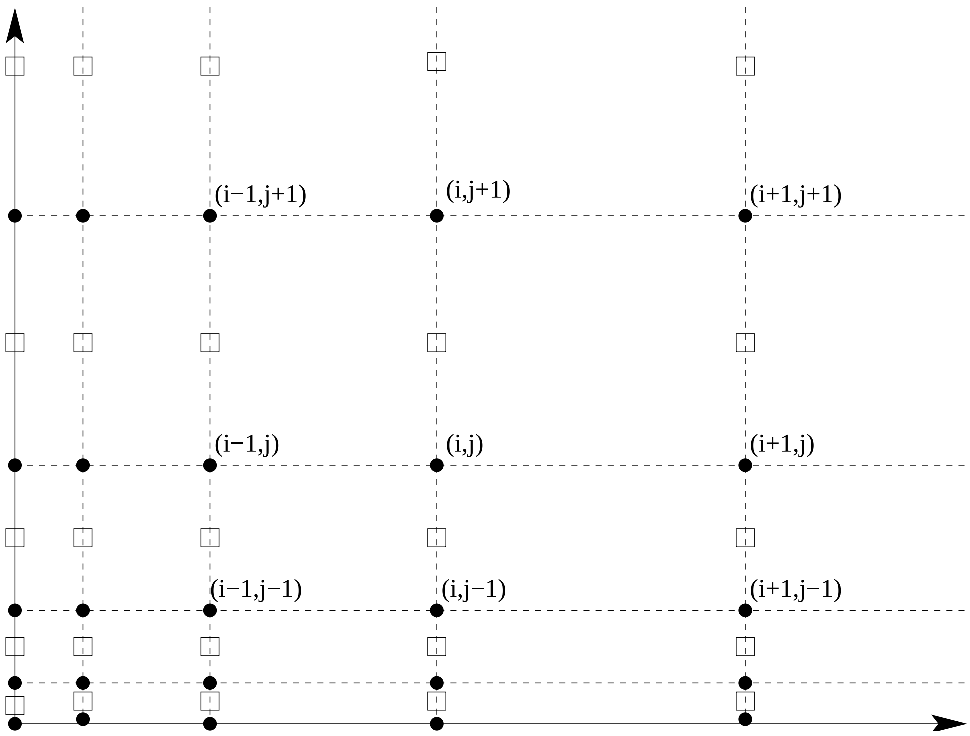


Figura 1: Griglia non equispaziata con nodi concetrati vicino alla parete ($y = 0$) e vicino al bordo d'attacco ($x = 0$). Le variabili $u$ e $v$ sono riferite ai nodi ($\bullet$); l'equazione di continuità è verificata nei punti $(x_i, \frac{y_j+y_{j-1}}{2})$, indicati con $\square$, mentre l'equazione della quantità di moto è verificata nei nodi ($\bullet$).

semplicità utilizziamo uno schema a differenze finite: Eulero esplicito (a passo variabile) in $x$ e differenze finite del second'ordine (non equispaziate) in $y$. Inoltre, assumiamo che le variabili $u_{i,j}$ e $v_{i,j}$ siano note nei nodi della griglia cartesiana, ma ad ogni $x$ soddisfiamo l'equazione di continuità in $(x_i, \frac{y_j+y_{j-1}}{2})$ (punti intermedi della griglia in $y$, indicati con $\square$), mentre l'equazione della quantità di moto viene soddisfatta in $(x_i, y_j)$, indicati con $\bullet$ in figura 1. Pertanto, le equazioni diventano:

$$\frac{\frac{u_{i,j}+u_{i,j-1}}{2} - \frac{u_{i-1,j}+u_{i-1,j-1}}{2}}{x_i - x_{i-1}} + \frac{v_{i,j} - v_{i,j-1}}{y_j - y_{j-1}} = 0 \qquad (1.5)$$

$$\frac{u_{i,j}^2 - u_{i-1,j}^2}{x_i - x_{i-1}} + \frac{u_{i,j+1}v_{i,j+1} - u_{i,j-1}v_{i,j-1}}{y_{j+1} - y_{j-1}} - \nu \frac{\frac{u_{i,j+1}-u_{i,j}}{y_{j+1}-y_j} - \frac{u_{i,j}-u_{i,j-1}}{y_j-y_{j-1}}}{\frac{y_{j+1}-y_{j-1}}{2}} = 0 \qquad (1.6)$$

Si osservi che le derivate rispetto ad $x$ sono del prim'ordine esplicite, mentre quelle rispetto ad $y$ sono del second'ordine esplicite. In realtà le derivate nell'equazione di continuità sono effettivamente differenze finite centrate (in $y$), mentre nell'equazione della quantità di moto le derivate (prime e seconde) rispetto a $y$ sarebbero corrette se valutate nel punto $(x_i, \frac{y_{j+1}+y_{j-1}}{2})$. Tuttavia questo errore non è maggiore di quello (del second'ordine) che si commette utilizzando le differenze finite centrate.

Un griglia come quella in figura 1 prende il nome di *staggered grid* (griglia "staggherata") ed è utilizzata qui perché altrimenti se l'equazione di continuità fosse soddisfatta nei nodi ($\bullet$) allora i valori delle variabili nei nodi dispari (in $y$, a $x$ fissata) risulterebbero disaccoppiati da quelli pari e la soluzione sarebbe oscillante.

Fissato $x = x_i$, per risolvere il problema è necessario solamente conoscere la soluzione alla $x$ precedente, $x = x_{i-1}$. Le incognite correnti possono essere ordinate in un unico vettore $f = [u_1, v_1, u_2, v_2, \ldots, u_N, v_N]^T$ che varia solo con $y$, essendo $y_1 = 0$ e $y_N = y_{\max} \approx \infty$. In altre parole,

$$u_1 = u_{i,1}, \quad v_1 = v_{i,1}, \quad u_2 = u_{i,2}, \quad v_2 = v_{i,2}, \quad \ldots, \quad u_N = u_{i,N}, \quad v_N = v_{i,N}.$$

Così facendo, ad ogni $x = x_i$, si ottiene il sistema non lineare formato dalle equazioni (1.5)-(1.6) che può essere riscritto in modo compatto come

$$\mathbf{b}(\mathbf{f}) = \mathbf{0},$$

e risolto utilizzando il metodo di Newton:

$$\mathbf{b}(\mathbf{f}) \approx \mathbf{b}(\bar{\mathbf{f}}) + [\boldsymbol{J}(\bar{\mathbf{f}})](\mathbf{f} - \bar{\mathbf{f}}) = \mathbf{0} \qquad \Rightarrow \qquad \mathbf{f} = \bar{\mathbf{f}} - [\boldsymbol{J}(\bar{\mathbf{f}})]^{-\mathbf{1}}\mathbf{b}(\bar{\mathbf{f}}),$$

dove $\bar{\mathbf{f}}$ è una soluzione di tentativo e $[\boldsymbol{J}(\bar{\mathbf{f}})]J$ è lo Jacobiano ivi valutato.

Come detto, questa procedura viene ripetuta da $x_{\text{in}}$ a $x = x_{\text{fi}}$.

**Esercizio 1.7** *Scrivere un proprio script (in Octave o Matlab) che risolva il problema dato e confrontare i risultati con quelli ottenuti con lo script* `BLfp.m` *marciando da $x = 0$ a $x = 1$.*

**Esercizio 1.8** *Capire come funziona lo script* `BLfp.m`*.*

**Esercizio 1.9** *Utilizzare lo script* `BLfp.m` *cambiando i vari parametri (numero di punti in $x$, $y$, "addensamento", $y_{\max}$, verificando i risultati ottenuti. Quando si può pensare di aver risolto abbastanza correttamente il problema numerico?*

**Esercizio 1.10** *Riscrivere lo script* `BLfp.m` *in modo da evitare i cicli (altrimenti non c'è guadagno in Octave o Matlab).*

**Esercizio 1.11** *Riscrivere lo script* `BLfp.m` *in modo che lo schema sia del second'ordine esplicito in $x$.*

# A Script `BLfp.m`

```
## Copyright (C) 2009 Simone Zuccher
##
## This program is free software; you can redistribute it and/or modify it
## under the terms of the GNU General Public License as published by
## the Free Software Foundation; either version 2, or (at your option)
## any later version.
##
## This program is distributed in the hope that it will be useful, but
## WITHOUT ANY WARRANTY; without even the implied warranty of
## MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
## General Public License for more details.
```

```
##

## Solve the Boundary-layer equations past a flat plate
##
##  |
##  |  u_x + v_y = 0
## <
##  |  u u_x + v u_y = nu u_{yy}
##  |
##
##  with BC:
##  @ y = 0:     u(x,0) = v(x,0) = 0
##  @ y = inf:   v_y(x,inf) = 0
##  and IC:
##  @ x = 0:     u(0,y) = 1
##  @ x = 0:     v(0,y) = 0
##
##  In the conservative form
##
##  |
##  |  u_x + v_y = 0
## <
##  |  (u^2)_x + (v u)_y = nu u_{yy}
##  |

##
## Since the system is parabolic-in-space, a marching technique is employed
## from x=0 to x=1.


## Author:   Simone Zuccher <zuccher@sci.univr.it>
## Created:  12 May 2009
## Modified:


clear all
close all
N = 100;     % number of y points
ymax = 20;   % maximum value of y (inf)
xmin = 0;    % initial x
xmax = 1;    % final x-location
nx = 10;     % number of grid points in x
nu = 1;      % viscosity

% Generate y grid clustering points close to the wall
y=((0:(N-1))./(N-1)).^3*ymax;
% Generate x grid clustering points close to x=0
x=xmin + ([0:(nx-1)]./(nx-1)).^2*(xmax-xmin);

% Initial guess for Newton:
% u is 1 except at the wall
% v is zero everywhere
f=zeros(2*N,1);
f(3:2:2*N) = ones(N-1,1);
% Now perturb a little bit to make sure nothing is zero
f = f + 0.01*(0.5-rand(size(f)));

% Solution at previous x
fold = f;

% These constants are use in the FD code
nvar=2;
neq=2;
uv=0;
vv=1;
eq1=0;
eq2=1;
```

```
% Loop on x
for ix=2:nx
dx = x(ix) - x(ix-1);

% Set the error to start the Newton's loop
err=1;
while(err>1e-6)
%  Set back everything to zero
   J=spalloc(2*N,2*N,5*2*N);
   rhs=zeros(2*N,1);
   eq=1;
   rhs(eq+eq1)=f(eq+uv)-0;
   rhs(eq+eq2)=f(eq+vv)-0;
   J(eq+eq1,eq+uv)=1;
   J(eq+eq2,eq+vv)=1;
   for i=2:N-1
      d10 = 1/(y(i+1)-y(i-1));
      d1p = 1/(y(i+1)-y(i));
      d1m = 1/(y(i)-y(i-1));
      d2p = 2*d1p/(y(i+1)-y(i));
      d20 = -2*d1p*(1/(y(i+1)-y(i)) + 1/(y(i)-y(i-1)));
      d2m = 2*d1p/(y(i)-y(i-1));
      eq = i*2-1;
      rhs(eq+eq1)=(f(eq+uv)+f(eq+uv-nvar))/2/dx - ...
                  (fold(eq+uv)+fold(eq+uv-nvar))/2/dx + ...
  d1m*(f(eq+vv)-f(eq+vv-nvar));
      rhs(eq+eq2)=(f(eq+uv)^2 - fold(eq+uv)^2)/dx + ...
                  d10*(f(eq+uv+nvar)*f(eq+vv+nvar) - ...
       f(eq+uv-nvar)*f(eq+vv-nvar)) ...
  - nu*(f(eq+uv+nvar)*d2p + f(eq+uv)*d20 + f(eq+uv-nvar)*d2m);
      %
      J(eq+eq1,eq+uv-nvar)= 1/2/dx;
      J(eq+eq1,eq+uv)= 1/2/dx;
      J(eq+eq1,eq+vv-nvar)= -d1m;
      J(eq+eq1,eq+vv)= d1m;
      %
      J(eq+eq2,eq+uv-nvar)= - d10*f(eq+vv-nvar) - nu*d2m;
      J(eq+eq2,eq+uv)= 2*f(eq+uv)/dx - nu*d20;
      J(eq+eq2,eq+uv+nvar)= d10*f(eq+vv+nvar) - nu*d2p;
      J(eq+eq2,eq+vv-nvar)= - d10*f(eq+uv-nvar);
      J(eq+eq2,eq+vv+nvar)= d10*f(eq+uv+nvar);
   end
   eq = N*2-1;
   rhs(eq+eq1)=f(eq+uv)-1;
   rhs(eq+eq2)=(f(eq+vv)-f(eq+vv-nvar))/(y(N) - y(N-1))-0;
   J(eq+eq1,eq+uv)=1;
   J(eq+eq2,eq+vv)=1/(y(N) - y(N-1));
   J(eq+eq2,eq+vv-nvar)=-1/(y(N) - y(N-1));

   df = -J\rhs;
%  Update solution
   f = df + f;
%  Check the error
   err=norm(df);
endwhile
err;
fold = f;
end

plot(f(1:2:end),y,'-',f(2:2:end),y,'-')
ylabel('y');
xlabel('u, v');
legend('u','v')
```