Laboratorio di Dinamica dei Fliudi
Esercitazione 02 – a.a. 2008-2009

Dott. Simone Zuccher

22 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 dell'equazione di Blasius

L'equazione di Blasius

$$f''' + \frac{1}{2} f f'' = 0 \tag{1.1}$$

e le relative condizioni al contorno

$$\begin{aligned} f(0) = f'(0) &= 0 && (1.2)\\ f'(\infty) &= 1 && (1.3) \end{aligned}$$

formano un problema ai limiti non lineare del terz'ordine. Esso può essere ricondotto ad un sistema non lineare di ordine più basso (secondo o primo) tramite l'introduzione di variabili ausiliarie. Qui ci fermiamo al second'ordine:

$$\begin{aligned} fu' + 2u'' &= 0 && (1.4)\\ f' - u &= 0 && (1.5) \end{aligned}$$

con condizioni al contorno

$$f(0) = 0, \qquad u(0) = 0, \qquad u(\infty) = 1 \tag{1.6}$$

Evidentemente, ai fini della soluzione numerica, l'equazione che rimpiazza la condizione al contorno per f all'infinito è $f'(\infty) = u(\infty) = 1$.

Siccome le derivate sono maggiori in prossimità della parete ($y = 0$), per risolvere numericamente l'equazione di Blasius addesiamo i punti in corrispondenza di $y = 0$. Per le derivate (prima e seconda) sulla griglia equispaziata si utilizzando le stesse idee viste nel caso del sistema di equazioni alle derivate parziali ottendo il sistema discretizzato

$$\begin{aligned} f_i \frac{u_{i+1} - u_{i-1}}{y_{i+1} - y_{i-1}} + 2\frac{\frac{u_{i+1}-u_i}{y_i - y_{i-1}} - \frac{u_i - u_{i-1}}{y_i - y_{i-1}}}{\frac{y_{i+1}-y_{i-1}}{2}} &= 0 && (1.7)\\ \frac{f_{i+1} - f_{i-1}}{y_{i+1} - y_{i-1}} - u_i &= 0, && (1.8) \end{aligned}$$

che viene risolto utilizzando il metodo di Newton.

Esercizio 1.9 *Scrivere un proprio script (in Octave o Matlab) che risolva il problema dato e confrontare i risultati con quelli ottenuti con lo script* `blasius.m`.

Esercizio 1.10 *Confrontare i risultati risolvendo l'equazione di Blasius con quelli ottenuti con lo script* `BLfp.m` *marciando da* $x = 0$ *a* $x = 1$.

Esercizio 1.11 *Capire come funziona lo script* `blasius.m`.

Esercizio 1.12 *Riscrivere lo script* `blasius.m` *in modo da evitare i cicli (altrimenti non c'è guadagno in Octave o Matlab).*

A Script `blasius.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 Blasius' equation
##
## f f'' + 2f''' = 0
##
## with BC
##
## f(0) = f'(0) = 0 and f'(ymax) = 1.
##
## By introducing u = f', the equation can be recast in the system
##
##  |
##  |  f u' + 2 u'' = 0
## <
##  |  f' - u    = 0
##  |
## with BC
##
## f(0) = u(0) = 0 and u(ymax) = 1.
##
## The last equation for f at ymax is f'(ymax) - u(ymax) = 0

## Author:   Simone Zuccher <zuccher@sci.univr.it>
## Created:  21 May 2009
## Modified:


clear all
close all
N = 200;     % number of y points
ymax = 10;   % maximum value of y (inf)

% Generate y grid clustering points close to the wall
y=([0:(N-1)]./(N-1)).^3*ymax;
```

```
% Initial guess for Newton:
% u goes from 0 to 1 linearly
% f is zero everywhere
f=zeros(2*N,1);
f(1:2:2*N) = linspace(0,1,N)';
% Now perturb a little bit to make sure nothing is zero
f = f + 0.01*rand(size(f));

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

% Set the error to start the Newton's loop
err=1;
while(err>1e-10)
%  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+fv)-0;
   J(eq+eq1,eq+uv)=1;
   J(eq+eq2,eq+fv)=1;
   for i=2:N-1
      d1p = 1/(y(i+1)-y(i-1));
      d1m = - d1p;
      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+fv)*d1p*(f(eq+uv+nvar)-f(eq+uv-nvar)) + ...
             2*(f(eq+uv+nvar)*d2p + f(eq+uv)*d20 + f(eq+uv-nvar)*d2m);
      rhs(eq+eq2)=d1p*(f(eq+fv+nvar)-f(eq+fv-nvar)) - f(eq+uv);
      %
      J(eq+eq1,eq+uv-nvar)= f(eq+fv)*d1m + 2*d2m;
      J(eq+eq1,eq+uv)= 2*d20;
      J(eq+eq1,eq+uv+nvar)= f(eq+fv)*d1p + 2*d2p;
      J(eq+eq1,eq+fv)= d1p*(f(eq+uv+nvar)-f(eq+uv-nvar));
      %
      J(eq+eq2,eq+fv-nvar)= d1m;
      J(eq+eq2,eq+uv)= -1;
      J(eq+eq2,eq+fv+nvar)= d1p;
   end
   eq = N*2-1;
   rhs(eq+eq1)=f(eq+uv)-1;
   rhs(eq+eq2)=(f(eq+fv)-f(eq+fv-nvar))/(y(N) - y(N-1))-f(eq+uv);
   J(eq+eq1,eq+uv)=1;
   J(eq+eq2,eq+fv)=1/(y(N) - y(N-1));
   J(eq+eq2,eq+uv)=-1;
   J(eq+eq2,eq+fv-nvar)=-1/(y(N) - y(N-1));

   ftmp = -J\rhs;
%  Upade solution
   f = ftmp + f;
%  Check the error
   err=norm(ftmp);
endwhile

% Plot results
U = f(1:2:end);
V = 1/2*(y'.*U-f(2:2:end));
plot(U,y,'-',V,y,'-')
ylabel('\eta');
```

```
xlabel('u, v');
legend('u','v')
axis([0 1 0 10])
```