S.L. BLASIUS : RISOLUZIONE NUMERICA

L'eq. da risolvere è:

$$(1) \quad 2f''' + f \cdot f'' = 0$$

EQ. DIFFERENZIALE ORDINARIA (ODE) DI ORDINE 3

Con condizioni al contorno:

$$f(\eta = 0) = 0$$

$$f'(\eta = 0) = 0$$

$$f'(\eta \to \infty) = 1$$

Questo è un classico BOUNDARY VALUE PROBLEM (BVP) che in MATLAB è risolvibile utilizzando la funzione <u>bvp4C</u>. Questa funzione risolve ODE di ordine 1 nella forma generale $y' = f(x, y)$. Per poter utilizzare bvp4C ed integrare l'eq. (1) è necessario riscrivere tale equazione nella forma integrabile da bvp4C.

Pertanto è innanzitutto necessario definire un vettore y in cui memorizzare i valori di f e delle sue derivate:

$$f = y(1)$$

$$f' = y(2) = y'(1)$$

$$f'' = y(3) = y'(2)$$

$$\Downarrow$$

$$2f''' + f \cdot f'' = 0 \rightsquigarrow 2 \cdot y'(3) + y(1) \cdot y(3) = 0$$

$$y'(3) = -\frac{1}{2} y(1) \cdot y(3)$$

In linguaggio MATLAB, detto dydeta il vettore in cui sono memorizzate solo le derivate di f:

$$\text{dydeta} = \begin{vmatrix} y'(1) \\ y'(2) \\ y'(3) \end{vmatrix} = \begin{vmatrix} y(2) \\ y(3) \\ -\frac{1}{2} y(1) \cdot y(3) \end{vmatrix}$$

Si pone:

```
dydeta = [ y(2)
           y(3)
           -0.5*y(1)*y(3) ];
```

```
function dydeta = Blasius (eta, y)
```

Bisogna poi definire le c.c. Detto res il vettore in cui queste vengono memorizzate, si ha:

$$\text{res} = \begin{vmatrix} 0 \\ 0 \\ 0 \end{vmatrix} = \begin{vmatrix} y0(1) \\ y0(2) \\ yinf(2) - 1 \end{vmatrix} \begin{matrix} \leftsquigarrow f(0)=0 \\ \leftsquigarrow f'(0)=0 \\ \leftsquigarrow f'(\infty)=1 \end{matrix}$$

In linguaggio MATLAB:

```
res = [ y0(1)
        y0(2)
        yinf(2) - 1 ];

function res = mot4bc(y0, yinf)
```

NOTA: y0(1) è il valore iniziale di y(1), pari a 0

y0(2) " " " " di y(2), pari a 0

yinf(2) " " " per $\eta \to \infty$ di y(2), pari a 1

Una volta definite le c.c. bisogna inizializzare la soluzione. In MATLAB ciò si può fare con la funzione bvpinit di bvp4c:

SOLUZ. I° TENT. PER f — SOLUZ. I° TENT. PER f' — SOLU[Z.] I° TE[NT.] PER f[']

```
solinit = bvpinit([0:2:8], [0 0 1]),
```

GRIGLIA DI CALCOLO SU CUI VIENE INTEGRATA L'ODE A PARTIRE DAI VALORI INIZIALI SPECIFICATI QUI

SOLUZIONE DI PRIMO TENTATIVO IN FORMA DI VETTORE

Pertanto : `bvpinit([0:0.2:8],[0 0 1]);`

VALORE DI η CHE DEFINISCE L'ESTREMO SX DEL RANGE SU CUI SI INTEGRA

SPAZIATURA DEL RANGE DI η

VALORE DI η CHE DEFINISCE L'ESTREMO DX DEL RANGE SU CUI SI INTEGRA

integrerà l'eq. da $\eta=0$ fino a $\eta=1$ con incrementi $\Delta\eta=0.2$ $(0; 0.2; 0.4; \dots ; 7.6; 8)$.

Per ogni valore discreto di η, vengono assegnati i valori iniziali di f, f' ed f'' ($f_0=0$, $f'_0=0$, $f''_0=1$).

L'istruzione che effettivamente integra l'eq. (1) è:

```
sol = bvp4c(@Blasius, @mat4bc, solinit)
```

```
function Blasius_solution

% Function to calculate solution to Blasius equation using bvp4c, a built
% in Matlab function that solves boundary value problems (BVPs) for
% ordinary differential equations (ODEs) using finite difference method.

clc

% Initialize solution

solinit = bvpinit([0:2:8],[0 0 1]);

% Call BVP solver

sol = bvp4c(@Blasius,@mat4bc,solinit);

% Calculate bvp solution at a range of points

eta_int = [0:0.2:8.0];
f_int = deval(sol,eta_int);

% Print out results

fprintf('n \t f \t df/dn \t d2f/dn2 \n' );
for i = 1:length(eta_int)
    fprintf('%3.2f %8.4f %8.4f %8.4f \n', ...
              eta_int(i), f_int(1,i), f_int(2,i), f_int(3,i));
end

% Plot results

figure('Position',[600,500,500,300],'Color','white')
plot(f_int(2,:),eta_int)
xlabel('{\it u}/{\it U}','FontWeight','b')
ylabel(texlabel('eta'), 'FontWeight','b')

function dfdeta = Blasius(eta,y)

% Blasius equation is given in state space.

% Original equation:    f'''  + 0.5 * f * f"  = 0

% Define:               y(1) = f
%                       y(2) = y(1)'
%                       y(3) = y(2)'

% Rewrite equation as:  y(3)' + 0.5*y(1)*y(3) = 0

% In matrix form:                |y(1)'|   |      y(2)        |
%                     dydeta =   |y(2)'| = |      y(3)        |
%                                |y(3)'|   | -0.5*y(1)*y(3)   |

dfdeta = [   y(2)
             y(3)
             -0.5*y(1)*y(3)];

function res = mat4bc(y0,yinf)

% Define boundary conditions where:
% f0    intial vector at eta = 0
% finf  final vector as eta goes to infinity

% Original boundary conditions: f0 = 0, f0' = 0, finf' = 1

% Rewrite as: y0(1) = 0, y0(2) = 0, yinf(2) = 1

% In matrix form:         |0|    | y0(1)       |
```

```
%                    res = |0| = |  y0(2)        |
%                          |0|   | yinf(2) - 1   |

res = [  y0(1)
         y0(2)
         yinf(2)-1];
```