4. Metoda Keller – Box

4.1. Preliminarii

Metoda Keller – Box este o metoda care utilizează diferențe finite, problemele de rezolvat reducându-se la rezolvarea unor sisteme de ecuații algebrice. Metoda a fost introdusă de Keller (1970) şi aplicată apoi de alți autori în problemele de strat limită laminar şi turbulent. Metoda a fost popularizată odată cu apariția cărții lui Cebeci şi Bradshaw (1984). Ea este folosită de un timp relativ scurt în țara noastră şi o descriere foarte amănunțită a ei se găseşte în cartea lui Pop şi Postelnicu (1999), Postelnicu (1999) şi Alexandrescu şi Alexandrescu (2001). Metoda Keller – Box este deosebit de eficientă în rezolvarea ecuațiilor diferențiale ordinare şi a ecuațiilor cu derivate parțiale de tip parabolic. Astfel de ecuații sau sisteme de ecuații apar şi în modelarea curgerilor cu strat limită, convecției, fenomenelor de transport, etc.

Pentru a rezolva ecuațiile diferențiale ordinare sau cu derivate parțiale folosind metoda Keller – Box vom introduce schema iterativă Newton – Raphson (vezi Coman, 1995) de rezolvare a ecuațiilor şi sistemelor de ecuații algebrice neliniare.

Astfel, dacă avem o ecuație algebrică neliniară de forma f(x) = 0, vom considera iterația necesară aflării rădăcinii

$$x_{n+1} = x_n + \varepsilon_n \tag{1.62}$$

unde x_n este aproximația rădăcinii la pasul n, iar ε_n este eroarea de calcul la acelaşi pas. Apoi, pentru a afla valoarea lui ε_n, dezvoltăm în serie Taylor funcția f(x) în punctul x_{n+1}:

$$0 = f(x_{n+1}) = f(x_n + \varepsilon_n) = f(x_n) + \varepsilon_n f\,'(x_n) + \ldots \tag{1.63}$$

unde f ' reprezintă derivata funcției f şi păstrând doar termenii de ordin întâi în ε_n avem:

$$\varepsilon_n \approx -\frac{f(x_n)}{f'(x_n)} \tag{1.64}$$

Întru cât ε_n este cunoscut, folosind relația (1.62), putem găsi iterația care ne va conduce la aflarea rădăcinii:

$$x_{n+1} = x_n + \varepsilon_n = x_n - \frac{f(x_n)}{f'(x_n)} \tag{1.65}$$

Putem generaliza acest raționament şi în cazul unui sistem algebric neliniar de N ecuații cu N necunoscute:

$$\begin{cases} f_1(x_1, x_2, ..., x_N) = 0 \\ f_2(x_1, x_2, ..., x_N) = 0 \\ \quad ... \\ f_N(x_1, x_2, ..., x_N) = 0 \end{cases} \tag{1.66}$$

Considerăm acum iterația (1.62) pentru aflarea rădăcinii, scrisă în formă N - dimensională:

$$(x_1, x_2, \ldots, x_N)^{(n+1)} = (x_1, x_2, \ldots, x_N)^{(n)} + (\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_N)^{(n)} \tag{1.67}$$

şi în continuare dezvoltăm sistemul (1.66) în serie Taylor:

$$0 = f_j\left(x_1^{(n+1)}, x_2^{(n+1)}, ..., x_N^{(n+1)}\right) = f_j\left(x_1^{(n)}, x_2^{(n)}, ..., x_N^{(n)}\right) + \sum_{i=1}^{N}\left(\frac{\partial f_j}{\partial x_i}\right)^{(n)} \varepsilon_i^{(n)}$$

$$j = \overline{1, N}$$

unde necunoscutele $\varepsilon_i^{(n)}$, i = 1, 2, …, N se pot determina rezolvând următorul sistem:

$$\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_N} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_N} \\ \cdots & & & \\ \frac{\partial f_N}{\partial x_1} & \frac{\partial f_N}{\partial x_2} & \cdots & \frac{\partial f_N}{\partial x_N} \end{bmatrix}^{(n)} \cdot \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \cdots \\ \varepsilon_N \end{bmatrix}^{(n)} = -\begin{bmatrix} f_1 \\ f_2 \\ \cdots \\ f_N \end{bmatrix}^{(n)} \tag{1.68}$$

Sistemul (1.68) fiind liniar poate fi rezolvat cu o rutină de tip Gauss. Totuşi, nu este uşor să găsim toate soluțiile sistemului (1.66) şi nu vom obține soluție pentru orice alegere inițială, x_1. Pentru evaluarea derivatelor din matricea sistemului (1.68) putem folosi diferențe finite şi, chiar dacă nu avem o eficiență atât de mare, scrierea codului poate fi generalizată pentru funcții foarte complicate. Astfel, de exemplu, avem:

$$\frac{\partial f_j}{\partial x_i} \approx \frac{f_j\left(x_1, ..., x_{j-1}, x_j + \delta, x_{j+1}, ..., x_N\right) - f_j\left(x_1, ..., x_{j-1}, x_j, x_{j+1}, ..., x_N\right)}{\delta} \tag{1.69}$$

unde δ este un număr mic ($\approx 10^{-5}$) şi nu neapărat pozitiv.

În cazuri extreme putem folosi un factor de relaxare, ω, pentru a subrelaxa corecțiile şi pentru a preîntâmpina o posibilă divergență:

$$(x_1, x_2, \ldots, x_N)^{(n+1)} = (x_1, x_2, \ldots, x_N)^{(n)} + \omega(\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_N)^{(n)} \tag{1.70}$$

unde $0 < \omega \leq 1$, metoda Newton – Raphson corespunzând cazului în care $\omega = 1$.

4.2. Aplicarea metodei Keller – Box în cazul ecuațiilor diferențiale ordinare neliniare

Vom arăta în continuare cum se aplică această metodă pe un caz concret, problema lui Blasius:

$$f''' + \frac{1}{2} ff'' = 0 \tag{1.71}$$

cu condițiile la limită:

$$f(0) = 0 \, , \, f'(0) = 0, \, f'(\infty) = 1 \tag{1.72}$$

În această metodă se reduce ecuația (1.71) la un sistem de ecuații diferențiale ordinare de ordinul unu făcând următoarele notații

$$a = f, \, b = f', \, c = f'' \tag{1.73}$$

Înlocuind în (1.71) şi (1.72) obținem

$$\begin{cases} a' - b = 0 \\ b' - c = 0 \\ c' + \frac{1}{2} ac = 0 \end{cases} \tag{1.74a}$$

iar condițiile la limită (1.72) devin:

$$\begin{cases} a(0) = 0 \\ b(0) = 0 \\ b(\infty) - 1 = 0 \end{cases} \tag{1.74b}$$

Definim o rețea de puncte $\eta_1, \eta_2, \ldots, \eta_N$ ($\eta_1 = 0$, $\eta_N = \eta_\infty$), paşii putând fi inegali, adică $h_i = \eta_{i+1} - \eta_i$. Pentru discretizarea derivatelor de ordinul întâi folosim diferențe finite centrale şi vom nota $a_i = a(\eta_i)$. Aproximarea pentru a'(η) în

$$\eta = \eta_{i+1/2} = \frac{\eta_{i+1} + \eta_i}{2}$$

este

$$a'_{i+1/2} \approx \frac{a_{i+1} - a_i}{h_i}$$

iar pentru $a(\eta_{i+1/2})$ vom avea

$$a_{i+1/2} \approx \frac{a_{i+1} + a_i}{2}.$$

După discretizare, considerând că avem un pas echidistant, ecuațiile (1.74a) devin

$$\begin{cases} \dfrac{a_{i+1} - a_i}{h} - \dfrac{b_{i+1} + b_i}{2} = 0 \\ \dfrac{b_{i+1} - b_i}{h} - \dfrac{c_{i+1} + c_i}{2} = 0 \\ \dfrac{c_{i+1} - c_i}{h} + \dfrac{1}{8}(a_{i+1} + a_i)(c_{i+1} + c_i) = 0 \end{cases} \tag{1.75}$$

pentru i = 1, 2,…, N-1, având 3N – 3 ecuații pentru 3N necunoscute. Numărul ecuațiilor se completează cu condițiile pe frontieră (1.74b) care, discretizate, se vor scrie:

$$a_1 = 0,\ b_1 = 0,\ b_N - 1 = 0 \tag{1.76}$$

Pentru a rezolva sistemul (1.75) – (1.76) folosim iterația Newton – Raphson, $a_i^{(n+1)} = a_i^{(n)} + \delta a_i^{(n)}$ şi utilizăm expresii similare şi pentru b şi c. Înlocuind aceste expresii în (1.75) şi (1.76) şi păstrând doar termenii de ordinul întâi în δ, rezultă:

$$\frac{\delta a_{i+1} - \delta a_i}{h} - \frac{\delta b_{i+1} + \delta b_i}{2} = -\left[\frac{a_{i+1} - a_i}{h} - \frac{b_{i+1} + b_i}{2}\right]^{(n)} \tag{1.77a}$$

$$\frac{\delta b_{i+1} - \delta b_i}{h} - \frac{\delta c_{i+1} + \delta c_i}{2} = -\left[\frac{b_{i+1} - b_i}{h} - \frac{c_{i+1} + c_i}{2}\right]^{(n)} \tag{1.77b}$$

$$\begin{aligned} &\frac{\delta c_{i+1} - \delta c_i}{h} + \frac{1}{8}(a_{i+1} + a_i)(\delta c_{i+1} + \delta c_i) + \frac{1}{8}(c_{i+1} + c_i)(\delta a_{i+1} + \delta a_i) \\ &= -\left[\frac{c_{i+1} - c_i}{h} + \frac{\lambda + 1}{8}(a_{i+1} + a_i)(c_{i+1} + c_i)\right]^{(n)} \end{aligned} \tag{1.77c}$$

Condițiile la limită (1.76) devin:

$$\delta a_1 = -a_1^{(n)},\ \delta b_1 = -b_1^{(n)},\ \delta b_N = 1 - b_N^{(n)} \tag{1.78}$$

Putem scrie sistemul (1.77) – (1.78) sub forma matricială (1.79), unde valorile pentru $(r_i)_j$ sunt date de relațiile:

$$(r_1)_i = -\left[\frac{a_{i+1} - a_i}{h} - \frac{b_{i+1} + b_i}{2}\right]$$

$$(r_2)_i = -\left[\frac{b_{i+1} - b_i}{h} - \frac{c_{i+1} + c_i}{2}\right]$$

$$(r_3)_i = -\left[\frac{c_{i+1} - c_i}{h} + \frac{1}{8}(a_{i+1} + a_i)(c_{i+1} + c_i)\right]$$

Prin rezolvarea sistemului (1.79) de mai jos, obținem mărimile $\delta a_i^{(n)}$, $\delta b_i^{(n)}$, $\delta c_i^{(n)}$, i = 1, 2, …, N şi astfel putem calcula următoarele iterații $a_i^{(n+1)}$, $b_i^{(n+1)}$, $c_i^{(n+1)}$, iar acest procedeu se continuă până la obținerea acurateții dorite. Există mai multe metode de rezolvare a sistemului algebric liniar (1.79) şi în continuare vom prezenta metoda eliminării blocurilor descrisă în cartea lui Cebeci şi Cousteix (1999), deoarece matricea sistemului are o structură tridiagonală de blocuri.

$$
\begin{bmatrix}
1 & 0 & 0 & & & & & & \\
0 & 1 & 0 & & & & & & \\
-\frac{1}{h} & -\frac{1}{2} & 0 & \frac{1}{h} & -\frac{1}{2} & 0 & & & \\
0 & -\frac{1}{h} & -\frac{1}{2} & 0 & \frac{1}{h} & -\frac{1}{2} & & & \\
\frac{1}{8}(c_2+c_1) & 0 & \frac{1}{8}(a_2+a_1)-\frac{1}{h} & \frac{1}{8}(c_2+c_1) & 0 & \frac{1}{8}(a_2+a_1)+\frac{1}{h} & & & \\
 & & & & & & & & \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots & & & \\
 & & -\frac{1}{h} & -\frac{1}{2} & 0 & \frac{1}{h} & -\frac{1}{2} & 0 & \\
 & & 0 & \frac{1}{h} & -\frac{1}{2} & 0 & \frac{1}{h} & -\frac{1}{2} & \\
 & & \frac{1}{8}(c_N+c_{N-1}) & 0 & \frac{1}{8}(a_N+a_{N-1})-\frac{1}{h} & \frac{1}{8}(c_N+c_{N-1}) & 0 & \frac{1}{8}(a_N+a_{N-1})+\frac{1}{h} & \\
 & & & & & 0 & 1 & 0 &
\end{bmatrix}
\times
\begin{bmatrix}
\delta a_1 \\ \delta b_1 \\ \delta c_1 \\ \ldots \\ \ldots \\ \ldots \\ \\ \\ \ldots \\ \delta a_N \\ \delta b_N \\ \delta c_N
\end{bmatrix}^{(n)}
=
\begin{bmatrix}
-a_1 \\ -b_1 \\ (r_1)_1 \\ (r_2)_1 \\ (r_3)_1 \\ \\ \\ \\ \\ (r_1)_{N-1} \\ (r_2)_{N-1} \\ (r_3)_{N-1} \\ 1-b_N
\end{bmatrix}^{(n)}
\tag{1.79}
$$

4.3. Metoda eliminării blocurilor

Pentru a putea aplica metoda eliminării blocurilor trebuie ca blocurile de pe diagonala principală să fie nesingulare şi în acest scop vom schimba între ele liniile corespunzătoare ecuațiilor (1.77a) şi (1.77b). După ce facem această schimbare introducem următoarele notații:

$$A_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{1}{h} & -\frac{1}{2} \end{bmatrix}, \qquad A_i = \begin{bmatrix} \frac{1}{h} & -\frac{1}{2} & 0 \\ \frac{1}{8}(c_{i+1}+c_i) & 0 & \frac{1}{8}(a_{i+1}+a_i)+\frac{1}{h} \\ 0 & -\frac{1}{h} & -\frac{1}{2} \end{bmatrix},$$

$$A_N = \begin{bmatrix} \frac{1}{h} & -\frac{1}{2} & 0 \\ \frac{1}{8}(c_N+c_{N-1}) & 0 & \frac{1}{8}(a_N+a_{N-1})+\frac{1}{h} \\ 0 & 1 & 0 \end{bmatrix}, \qquad 2 \le i \le N-2$$

$$B_{i+1} = \begin{bmatrix} -\frac{1}{h} & -\frac{1}{2} & 0 \\ \frac{1}{8}(c_{i+1}+c_i) & 0 & \frac{1}{8}(a_{i+1}+a_i)-\frac{1}{h} \\ 0 & 0 & 0 \end{bmatrix}, \quad 1 \le i \le N-1$$

$$C_i = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \frac{1}{h} & -\frac{1}{2} \end{bmatrix}, \quad 1 \le i \le N-1 \qquad (1.80)$$

$$\delta_i = \begin{bmatrix} \delta a_i \\ \delta b_i \\ \delta c_i \end{bmatrix}, \quad 1 \le i \le N$$

$$r_1 = \begin{bmatrix} -a_1 \\ -b_1 \\ (r_2)_1 \end{bmatrix}, \quad r_N = \begin{bmatrix} (r_1)_{N-1} \\ (r_3)_{N-1} \\ 1-b_N \end{bmatrix}, \quad r_i = \begin{bmatrix} (r_1)_{i-1} \\ (r_3)_{i-1} \\ (r_2)_i \end{bmatrix}, \quad 2 \le i \le N-1$$

Cu notațiile (1.80) sistemul (1.79) se poate scrie acum în forma matricială:

$$\begin{bmatrix} A_1 & C_1 & & & & & & \\ B_2 & A_2 & C_2 & & & & & \\ \dots & \dots & \dots & & & & & \\ & & B_i & A_i & C_i & & & \\ & & \dots & \dots & \dots & & & \\ & & & & B_{N-1} & A_{N-1} & C_{N-1} & \\ & & & & & B_N & A_N & \end{bmatrix} \times \begin{bmatrix} \delta_1 \\ \delta_2 \\ \dots \\ \delta_i \\ \dots \\ \delta_{N-1} \\ \delta_N \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ \dots \\ r_i \\ \dots \\ r_{N-1} \\ r_N \end{bmatrix} \tag{1.81}$$

În continuare rezolvăm sistemul (1.81) folosind metoda eliminării blocurilor şi pentru început vom afla mărimile Γ_i şi Δ_i din formulele recursive de mai jos:

$$\begin{gathered} \Delta_1 = A_1 \\ \Gamma_i \Delta_{i-1} = B_i, \ i = 2,3, \dots N \\ \Delta_i = A_i - \Gamma_i C_{i-1}, \quad i = 2,3,\dots N \end{gathered} \tag{1.82}$$

unde Γ_i are aceeaşi structură ca şi blocul B_i, apoi vom calcula w_i după formula:

$$\begin{gathered} w_1 = r_1 \\ w_i = r_i - \Gamma_i w_{i-1}, \quad i=2,3,\dots N \end{gathered} \tag{1.83}$$

În cele din urmă, putem afla valorile δ_i necesare calculării iteraţiilor pentru pasul următor:

$$\begin{gathered} \Delta_N \delta_N = w_N \\ \Delta_i \delta_i = w_i - C_i \delta_{i+1}, \ j = N-1, N-2, \dots, 1 \end{gathered} \tag{1.84}$$

Metoda eliminării blocurilor este o metodă generală care poate fi utilizată pentru rezolvarea oricărui sistem liniar de ecuaţii algebrice cu structură de blocuri tridiagonală. Complexitatea acestui algoritm depinde de ordinul matricelor A_i, B_i, C_i. Când ordinul acestor matrici este mic atunci sistemul (1.81) se poate rezolva relativ uşor fără folosirea unor cantităţi mari de resurse.

Programe Matlab

```
%Keler box Blasius
x1=0; x2=7;
h=0.01;
N=(x2-x1)/h+1;
a=ones(N,1);b=zeros(N,1);c=zeros(N,1);

for i=1:N
   b(i)=0+i/N;
end;

err=1;
nr_iter=0;
while abs(err)>=1e-6
[A,B,C,r]=init_matrix(a,b,c,N,h);
delta=elim_bloc(A,B,C,r,N);
err=max(max(delta));
[a,b,c]=reeval(delta,a,b,c,N);
nr_iter=nr_iter+1;
end;
plot(x1:h:x2,b);
fprintf('\n iteratii:%d f"(0)=%f\n',nr_iter,c(1))
-----------------------------------------------------
function [a,b,c]=reeval(delta,a,b,c,N)

for i=1:N
   a(i)=a(i)+delta(1,i);
   b(i)=b(i)+delta(2,i);
   c(i)=c(i)+delta(3,i);
end
-----------------------------------------------------

function [A,B,C,r]=init_matrix(a,b,c,N,h);
%initializam blocurile matricei sistemului si matricea coloana a
%termenilor liberi
%initializam termenii liberi
r1=zeros(N-1,1);r2=zeros(N-1,1);r3=zeros(N-1,1);
for i=1:N-1
   r1(i)=-((a(i+1)-a(i))/h-(b(i+1)+b(i))/2);
   r2(i)=-((b(i+1)-b(i))/h-(c(i+1)+c(i))/2);
   r3(i)=-((c(i+1)-c(i))/h+(a(i+1)+a(i))*(c(i+1)+c(i))/8);
end;
r(:,1)=[-a(1);-b(1);r2(1)];
for i=2:(N-1)
   r(:,i)=[r1(i-1);r3(i-1);r2(i)];
end;
r(:,N)=[r1(N-1);r3(N-1);1-b(N)];
```

```
%initializare blocurile A
A(:,:,1)=[1 0 0 ;0 1 0;0 -1/h -0.5];
for i=1:(N-2)
A(:,:,i+1)=[1/h -0.5 0;(c(i+1)+c(i))/8 0 (1/h)+((a(i+1)+a(i))/8);
0 -1/h -1/2];
end
A(:,:,N)=[1/h -1/2 0;(c(N)+c(N-1))/8 0 (1/h)+((a(N)+a(N-1))/8);
0 1 0];

%initializare blocurile B
B(:,:,1)=zeros(3,3);
for i=1:(N-1)
   B(:,:,i+1)=[-1/h -1/2 0;
(c(i+1)+c(i))/8 0 (-1/h)+((a(i+1)+a(i))/8);0 0 0];
end

%initializare blocurile C
C(:,:,1)=zeros(3,3);
for i=1:(N-1)
   C(:,:,i)=[0 0 0;0 0 0;0 1/h -0.5];
end
-----------------------------------------------
function delta=elim_bloc(A,B,C,r,N)
%se rezolva sistemul prin metoda eliminarii blocurilor
%parametrul returnat delta reprezinta solutia sistemului
dd(:,:,1)=A(:,:,1);
w(:,1)=r(:,1);
for i=2:N
   gamma(:,:,i)=B(:,:,i)*inv(dd(:,:,i-1));
   w(:,i)=r(:,i)-gamma(:,:,i)*w(:,i-1);
   dd(:,:,i)=A(:,:,i)-gamma(:,:,i)*C(:,:,i-1);
end

delta(:,N)=inv(dd(:,:,N))*w(:,N);
for i=N-1:-1:1
   delta(:,i)=inv(dd(:,:,i))*(w(:,i)-C(:,:,i)*delta(:,i+1));
end
-----------------------------------
```