SZYBKI ALGORYTM Z MACIERZĄ SHURA DLA MACIERZY TRÓJDIAGONALNYCH

Rozwiązujemy układ z macierzą trójdiagonalną. Założymy dla prostoty opisu, że macierz ma stałe współczynniki, to znaczy, że na głównej diagonali jest zawsze element d, na poddiagonali element a, zaś na naddiagonali element b. Opisana tu metoda, z małą modyfikacją, może być zastosowana również do układów z dowolną macierzą trójdiagonalną.

$$A\underline{x} = \underline{f}. \tag{1}$$

Macierz A zapisana blokowo
$N+1$ bloków kwadratowych, wymiaru $R+1 \times R+1$

$$A = \begin{bmatrix} D & B & \cdot & \cdot & \cdot & \cdot & \cdot \\ A & D & B & \cdot & \cdot & \cdot & \cdot \\ \cdot & A & D & B & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & A & D \end{bmatrix} \tag{2}$$

Macierz w postaci oryginalnej ($N \ll M$)
gdzie $M = (N+1)(R+1)$

$$A = \begin{bmatrix} d & b & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ a & d & b & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & a & d & b & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & a & d \end{bmatrix}$$

Wektory $\underline{x}$ i $\underline{f}$ dostosowane do rozkładu blokowego A:

$$\underline{x} = [\underline{x}_0, \underline{x}_1, \cdots, \underline{x}_N]^T$$
$$\underline{f} = [\underline{f}_0, \underline{f}_1, \cdots, \underline{f}_N]^T$$

Układ w postaci blokowej po rozpisaniu:

$$D\underline{x}_0 + B\underline{x}_1 = \underline{f}_0 \tag{3}$$

$$A\underline{x}_{j-1} + D\underline{x}_j + B\underline{x}_j = \underline{f}_j, \quad j = 1, 2, \cdots, N-1,$$
$$A\underline{x}_{N-1} + D\underline{x}_N = \underline{f}_N$$

Mamy $N+1$ procesorów, ponumerowanych $0, 1, \cdots N$. Każdemu procesorowi jest przyporządkowane jedno równanie blokowe, które ten procesor będzie obsługiwał.

ALGORYTM.

1. **Wszystkie procesory $j = 0, 1, \cdots, N$ wykonują równolegle pierwszy krok algorytmu. W procesorze o numerze j polega to na rozwiązania trzech układów równań o wymiarze $R+1$:**

   $$D\tilde{\underline{x}}_j = \underline{f}_j \tag{4}$$

   $\tilde{\underline{x}}_j$, to pierwsze przybliżenie wektora $\underline{x}_j$,

   $$D\underline{w}_0 = \underline{e}_0, \tag{5}$$

   $$D\underline{w}_R = \underline{e}_R, \tag{6}$$

   z tą samą macierzą trójdiagonalną D, gdzie

   $$\underline{e}_k, \ 0 \leqslant k \leqslant R$$

są wersorami osi współrzednych. Zauważmy, że dla każdego j, $0 \leqslant j \leqslant N$,

$$A = a\underline{e}_0 \ \underline{e}_R^T,$$

$$B = b\underline{e}_R \ \underline{e}_0^T,$$

więc, po wykonaniu pierwszego kroku, równania (3) będzie można zapisać tak:

(7)
$$\underline{x}_0 = \underline{\tilde{x}}_0 - (b\underline{w}_R) \ \underline{e}_0^T \underline{x}_1$$

$$\underline{x}_j = \underline{\tilde{x}}_j - (a\underline{w}_0) \ \underline{e}_R^T \underline{x}_{j-1} - (b\underline{w}_R) \ \underline{e}_0^T \underline{x}_{j+1}, \ 1 \leqslant j \leqslant N-1$$

$$\underline{x}_N = \underline{\tilde{x}}_N - (a\underline{w}_0) \ \underline{e}_R^T \underline{x}_{N-1}$$

Wzory (7) określają "poprawki", które po dodaniu do $\underline{\tilde{x}}_j$ pozwolą obliczyć dokładne składowe rozwiązania $\underline{x}_j$, $0 \leqslant j \leqslant N$.

Zauważmy, że na tym etapie, w ramach jednego procesora, nie da się wykorzystać poprawek, gdyż w każdym z równań (7) występują zmienne x_j dla różnych j.

2. Drugi krok algorytmu polega na utworzeniu w każdym procesorze UKŁADU RÓWNAŃ SHURA, który pozwoli wykorzystać poprawki, i ostatecznie, znaleźć składowe $\underline{x}_j$ dokładnego rozwiązania.

   UKŁAD SHURA ma niski wymiar $(N-1)(N-1)$ i jest trójdiagonalny. Bedzie rozwiązywany w każdym procesorze. Najpierw zdefiniujemy "procedurę numeracyjną", przy pomocy której określimy zmienne UKŁADU SHURA i jego współczynniki

PROCEDURA NUMERACYJNA
NUMEROWANIE ZMIENNYCH I RÓWNAŃ UKŁADU SHURA

W oparciu o równania poprawek (7) opiszemy tę procedurę w przypadku, gdy $N = 4$.

(8)

$$
\begin{array}{cccl}
 & \underline{e}_R^T| & \underline{x}_0 = \tilde{\underline{x}}_0 & - b\underline{w}_R[\underline{e}_0^T\underline{x}_1]_0 \\
\underline{e}_0^T| & \underline{e}_R^T| & \underline{x}_1 = \tilde{\underline{x}}_1 - a\underline{w}_0[\underline{e}_R^T\underline{x}_0]_1 & - b\underline{w}_R[\underline{e}_0^T\underline{x}_2]_2 \\
\underline{e}_0^T| & \underline{e}_R^T| & \underline{x}_2 = \tilde{\underline{x}}_2 - a\underline{w}_0[\underline{e}_R^T\underline{x}_1]_3 & - b\underline{w}_R[\underline{e}_0^T\underline{x}_3]_4 \\
\underline{e}_0^T| & \underline{e}_R^T| & \underline{x}_3 = \tilde{\underline{x}}_3 - a\underline{w}_0[\underline{e}_R^T\underline{x}_2]_5 & - b\underline{w}_R[\underline{e}_0^T\underline{x}_4]_6 \\
\underline{e}_0^T| & & \underline{x}_4 = \tilde{\underline{x}}_4 - a\underline{w}_0[\underline{e}_R^T\underline{x}_3]_7 &
\end{array}
$$

Zmienne UKŁADU SHURA oznaczamy przez

$$z_s,\ 0 \leqslant s \leqslant 2N - 1 = 7,$$

gdzie

$$z_s = [\cdots]_s, 0 \leqslant s \leqslant 2N - 1 = 7.$$

Jak czytać wzory (8)?

Każde równanie blokowe z (8) jest mnożone lewostronnnie przez $\underline{e}_0^T$ lub/i przez $\underline{e}_R^T$, dzięki czemu w lewej części wzorów (8) określających poprawki, pojawią się zmienne z_s, $0 \leqslant s \leqslant 2(N-1)-1$. Te same zmienne z_s można znaleźć już po prawej stronie wzorów (8) w kwadratowych nawiasach. Zauważmy też, że mnożenie lewostronne przez $\underline{e}_0^T$ wprowadza zmienne o numerach parzystych, zaś mnożenie lewostronne przez $\underline{e}_R^T$ wprowadza zmienne o numerach nieparzystych. Powstaje w ten sposób układ

$(N-1)(N-1)$ równań skalarnych. Jest to UKŁAD SHURA. Wykorzystując teraz wektory $\underline{w}_0$ i $\underline{w}_R$ określone wzorami (5) i (6) możemy wyznaczyć współczynniki tego układu i, po uporządkowaniu, wypisać go explicite:

UKŁAD SHURA

(9) $$\Sigma \underline{z} = \underline{F}$$

$$\begin{bmatrix} h & 1 & & & & & & \\ 1 & p & q & & & & & \\ & g & h & 1 & & & & \\ & & 1 & p & q & & & \\ & & & g & h & 1 & & \\ & & & & 1 & p & q & \\ & & & & & g & h & 1 \\ & & & & & & 1 & p \end{bmatrix} \begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \\ z_6 \\ z_7 \end{bmatrix} = \begin{bmatrix} \underline{e}_R^T \tilde{\underline{x}}_0 \\ \underline{e}_0^T \tilde{\underline{x}}_1 \\ \underline{e}_R^T \tilde{\underline{x}}_1 \\ \underline{e}_0^T \tilde{\underline{x}}_2 \\ \underline{e}_R^T \tilde{\underline{x}}_2 \\ \underline{e}_0^T \tilde{\underline{x}}_3 \\ \underline{e}_R^T \tilde{\underline{x}}_3 \\ \underline{e}_0^T \tilde{\underline{x}}_4 \end{bmatrix}.$$

ELEMENTY MACIERZY SHURA

$g = a\underline{w}_{R,0}, \quad h = b\underline{w}_{R,R}$

$p = a\underline{w}_{0,0}, \quad q = b\underline{w}_{0,R}$

gdzie:

$$\underline{w}_0 = [w_{0,0}, w_{0,1}, \cdots, w_{0,R}]^T$$

i

$$\underline{w}_R = [w_{R,0}, w_{R,1}, \cdots, w_{R,R}]^T.$$

Zauważmy jeszcze, że do utworzenia macierzy UKŁADU SHURA potrzebne są tylko wektory $\underline{w}_0$ i $\underline{w}_R$, które są w każdym procesorze. Nie możemy jednak rozwiązać tego układu, ponieważ nie mamy wszystkich składowych wektora prawej strony. Zauważmy, że udało się nam dojść do tego punktu, nie korzystając z komunikacji między procesorami. Ale na tym koniec! (:-()

3. **Aby rozwiązać UKŁAD SHURA w każdym procesorze i obliczyć**

POPRAWKI

$$\begin{aligned}
\underline{x}_0 &= \underline{\tilde{x}}_0 && - b\underline{w}_R z_0 \\
\underline{x}_1 &= \underline{\tilde{x}}_1 - a\underline{w}_0 z_1 && - b\underline{w}_R z_2 \\
\underline{x}_2 &= \underline{\tilde{x}}_2 - a\underline{w}_{20} z_3 && - b\underline{w}_R z_4 \\
\underline{x}_3 &= \underline{\tilde{x}}_3 - a\underline{w}_{30} z_5 && - b\underline{w}_R z_6 \\
\underline{x}_4 &= \underline{\tilde{x}}_4 - a\underline{w}_{40} z_7
\end{aligned}$$

trzeba w każdym procesorze skompletować w całe rozwiązanie UKŁADU SHURA. Na szczęście w tym przypadku można to zrobić przy pomocy jednego rozkazu systemu MPI.

Oto jeden ze sposobów. Ponieważ w każdym procesorze (np. o numerze j) w mamy obliczony wektor

$$\underline{\tilde{x}}_j,$$

to mamy częśc wektora prawej strony. Rozpatrzmy nasz przykład dla $N = 4$, wtedy UKŁAD SHURA jest wymiaru 8×8

(a) jeśli $j = 0$ to tworzymy wektor

$$\boldsymbol{F_0} = [\underline{e}_R^T \underline{\tilde{x}}_0, 0, 0, 0, 0, 0, 0, 0]^T$$

(b) jeśli $j = 4$ to tworzymy wektor

$$\boldsymbol{F_4} = [0, 0, 0, 0, 0, 0, 0, \underline{e}_R^T \underline{\tilde{x}}_4]^T$$

(c) jeśli np. $j = 2$ to tworzymy wektor

$$\boldsymbol{F_j} = [0, 0, 0, \underline{e}_0^T \underline{\tilde{x}}_2, \underline{e}_R^T \underline{\tilde{x}}_2, 0, 0, 0]^T$$

(d) i.t.d.

Teraz w każdym procesorze rozwiązujemy UKŁAD SHURA

$$\Sigma \boldsymbol{y_j} = \boldsymbol{F_j}.$$

Przy pomocy komendy MPI "Allreduce" dla wektorów y_j z opcją "SUM" wydanej we wszystkich procesorach, w każdym z nich otrzymujemy na miejscu wektora y_j pełne rozwiązanie $\underline{z}$ UKŁADU SHURA. W każdym z procesorów, przy pomocy odpowiedniej poprawki, znajdujemy wektor $\underline{x}_j$, część dokładnego rozwiązania naszego układu równań.

==========

Taki algorytm po raz pierwszy zastosował i opisał STEFAN BONDELI z ZÜRICHU. "Divide and Conquer " Lecture Notes in Computer Science vol.457 (1990).