WTKŁAD 4.

JAK LICZYĆ SCHEMATEM BOX NA WIELU PROCESORACH?

Zajmiemy się tu przypadkiem, gdy $\boldsymbol{\alpha > 0}$, to znaczy, że warunek brzegowy jest po lewej stronie, i ruch odbywa sie z lewej do prawej. Zauważmy najpierw, że schemat BOX to poprostu układ równań liniowych, o macierzy dwudiagonalnej

$$\boldsymbol{a u_{k+1}^{n+1} + b u_k^{n+1} = b u_{k+1}^n + a u_k^n.} \tag{4.1}$$

Zapiszmy ten układ inaczej:

$$\begin{bmatrix} \boldsymbol{a} & & & & & \\ \boldsymbol{b} & \boldsymbol{a} & & & & \\ & \boldsymbol{b} & \boldsymbol{a} & & & \\ & & & & & \\ & & & & & \\ & & & \boldsymbol{b} & \boldsymbol{a} & \\ & & & & \boldsymbol{b} & \boldsymbol{a} \end{bmatrix} \begin{bmatrix} \boldsymbol{x_0} \\ \boldsymbol{x_1} \\ \\ \\ \\ \boldsymbol{x_M} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f_0} \\ \boldsymbol{f_1} \\ \\ \\ \\ \boldsymbol{f_M} \end{bmatrix} \tag{4.2}$$

W tym zapisie $\boldsymbol{x_j = u_j^{n+1}}$, zaś $\boldsymbol{f_j}$ oznacza odpowiednie wyrażenie po prawej stronie schematu. [1] Podzielimy teraz układ (4.2), ładując do procesora nr. $\boldsymbol{s}$, $\boldsymbol{R+1}$ wierszy numerowanych (lokalnie) od $\boldsymbol{0}$ do $\boldsymbol{R}$; niech $\boldsymbol{s = 0, 1, 2, \cdots, lp - 1}$, gdzie $\boldsymbol{lp}$ to liczba procesorów w użyciu. Dla wygody zapisu załóżmy, że $\boldsymbol{lp = 5}$. Układ po podziale zapiszemy tak:

$$\left[\begin{array}{c|c|c|c|c} \boldsymbol{D} & & & & \\ \hline \boldsymbol{A} & \boldsymbol{D} & & & \\ \hline & \boldsymbol{A} & \boldsymbol{D} & & \\ \hline & & \boldsymbol{A} & \boldsymbol{D} & \\ \hline & & & \boldsymbol{A} & \boldsymbol{D} \end{array}\right] \begin{bmatrix} \underline{\boldsymbol{X_0}} \\ \underline{\boldsymbol{X_1}} \\ \underline{\boldsymbol{X_2}} \\ \underline{\boldsymbol{X_3}} \\ \boldsymbol{X_4} \end{bmatrix} = \begin{bmatrix} \underline{\boldsymbol{F_0}} \\ \underline{\boldsymbol{F_1}} \\ \underline{\boldsymbol{F_2}} \\ \underline{\boldsymbol{F_3}} \\ \boldsymbol{F_4} \end{bmatrix}.$$

[1] Pamiętajmy, że wyrażenie $\boldsymbol{f_0}$ musi być sumą 'właściwej prawej strony' i

$$\boldsymbol{aBV^n - bBV^{n+1}},$$

gdzie $\boldsymbol{BV^{\cdot}}$ oznacza lewy warunek brzegowy Dirichleta na poziomie czasowym $\boldsymbol{n}$ i $\boldsymbol{n+1}$ odpowiednio

Bloki $\boldsymbol{D}$ i $\boldsymbol{A}$ są wymiaru $\boldsymbol{R+1 \times R+1}$, $\boldsymbol{e_0}$ i $\boldsymbol{e_R}$ są kolumnowemi wersorami osi zerowej i $\boldsymbol{R}$-tej odpowiednio; $\boldsymbol{D}$ jest blokiem dwudiagonalnym, zaś $\boldsymbol{A = be_0 e_R^T}$ ma tylko jeden element $\boldsymbol{b}$ różny od zera umieszczony w prawym górnym rogu. Każdy z wierszy blokowych mnożymy przez macierz $\boldsymbol{D^{-1}}$, otrzymamy:

$$\begin{aligned} \boldsymbol{X_0} &= \boldsymbol{\tilde{X}_0} \\ \boldsymbol{X_1} &= \boldsymbol{\tilde{X}_1 - D^{-1} A X_0} \\ \boldsymbol{X_2} &= \boldsymbol{\tilde{X}_2 - D^{-1} A X_1} \\ \boldsymbol{X_3} &= \boldsymbol{\tilde{X}_3 - D^{-1} A X_2} \\ \boldsymbol{X_4} &= \boldsymbol{\tilde{X}_4 - D^{-1} A X_3} \end{aligned} \tag{4.3}$$

gdzie $\boldsymbol{\tilde{X}_k = D^{-1} F_k}$ można uważać za "pierwsze przybliżenie" wektora $\boldsymbol{X_k}$, zaś $\boldsymbol{D^{-1} A X_{k-1}}$ za "poprawkę". Zajrzyjmy teraz do "poprawek". Mamy $\boldsymbol{b D^{-1} e_0 e_R^T X_{k-1} = b \underline{w} e_R^T X_{k-1}}$, gdzie

$$\boldsymbol{\underline{w} = [w_0, w_1, \cdots, w_R]^T = D^{-1} e_0.}$$

Widać odrazu, że oba wektory $\boldsymbol{\tilde{X}_k}$ i $\boldsymbol{\underline{w}}$ można znaleźć jednocześnie rozwiązując układ o macierzy dwudiagonalnej z podwójną prawą stroną

$$\boldsymbol{D[\tilde{X}_k, \underline{w}] = [F_k, e_0].} \tag{4.4}$$

Teraz pomnóżmy lewostronnie wszystkie (blokowe) równania układu (4.3) przez $\boldsymbol{e_R^T}$. Jeśli oznaczymy: $\boldsymbol{z_k = e_R^T X_k}$ i $\boldsymbol{c = b w_R}$ to z równań (4.3) otrzymamy dwudiagonalny układ $\boldsymbol{lp-1}$ równań liniowych (skalarnych), zwany **układem Schura**:

$$\begin{aligned} \boldsymbol{z_0} &= \boldsymbol{e_R^T \tilde{X}_0} \\ \boldsymbol{z_1 + c z_0} &= \boldsymbol{e_R^T \tilde{X}_1} \\ \boldsymbol{z_2 + c z_1} &= \boldsymbol{e_R^T \tilde{X}_2} \\ & \cdots\cdots\cdots\cdots \\ \boldsymbol{z_{lp-1} + c z_{lp-2}} &= \boldsymbol{e_R^T \tilde{X}_{lp-1}} \end{aligned} \tag{4.5}$$

Rozwiązując układ Schura znajdziemy "poprawki" i wyznaczymy "dokładne" rozwiązanie

$$\boldsymbol{[X_0, X_1, \cdots X_{lp-1}]^T}$$

$$\boldsymbol{X_0 = \tilde{X}_0}$$
$$\boldsymbol{X_1 = \tilde{X}_1 - b\underline{w}z_0}$$
$$\boldsymbol{X_2 = \tilde{X}_2 - b\underline{w}z_1}$$

(4.6) $$\cdots\cdots\cdots\cdots$$

$$\boldsymbol{X_{lp-1} = \tilde{X}_{lp-1} - b\underline{w}z_{lp-2}}$$

ALGORYTM
dla układu równań schematu BOX

Potrzebne dane dla programu to:

$$\boldsymbol{a = 1 + \lambda\alpha, \quad b = 1 - \lambda\alpha, \quad \lambda = \frac{\tau}{h}}$$

1. W procesorze numer $\boldsymbol{s}$ rozwiązujemy układ $\boldsymbol{R+1}$ równań z dwiema prawymi stronami
$$\boldsymbol{D[\tilde{X}_s, \underline{w}] = [F_s, e_0].}$$
W każdym z procesorów mamy "pierwsze przybliżenie" $\boldsymbol{\tilde{X}_s}$ i wektor $\boldsymbol{\underline{w}}$:
$$\boldsymbol{\tilde{X}_s = [\tilde{X}_{s,0}, \tilde{X}_{s,1}, \cdots, \tilde{X}_{s,R}]^T, \quad \underline{w} = [w_0, w_1, \cdots, w_R]^T.}$$
Stąd mamy współczynnik macierzy Schura $\boldsymbol{c = w_R b}$ i $\boldsymbol{s}$-tą współrzędną wektora prawej strony układu Schura $\boldsymbol{\tilde{X}_{s,R}}$. W każdym z procesorów tworzymy macierz Schura i chwilowo niepełny wektor prawej strony tego układu. Przy pomocy komendy **MPI_Allgather($\cdots$)** uzupełniamy wektor prawej strony układu Schura. Ostatecznie w procesorze nrumer $\boldsymbol{s, \quad s = 0, 1, \cdots, lp-1}$ mamy cały układ Schura, i oba wektory $\boldsymbol{\tilde{X}_s}$, oraz $\boldsymbol{\underline{w}}$

2. Teraz w każdym z procesorów
   - Rozwiazujemy układ Schura (4.5)
   - Obliczamy i wprowadzamy poprawki (4.6)

W w procesorze numer $s, s = 0, 1, \cdots, lp - 1$ mamy część "dokładnego" rozwiązania X_s.

Jesli jest taka potrzeba, można jeszcze użyć komendy **MPI_Gather**$(\cdots)$ lub **MPI_Allgather**$(\cdots)$ aby skomasować wszystkie części rozwiązania.

Uwaga. Przedstawiony tu algorytm jest modyfikacją algorytmu z macierzą Schura dla układów trójdiagonalnych. Algorytm dla macierzy trójdiagonalnych po raz pierwszy opisał Stefan Bondeli w pracy "Divide and Conquer..." (Lecture Notes in Computer Science vol. 457/1990) Algorytm z macierzą Schura będzie opisany w jednym z następnych "Wykładów".

To wszystko co napisane wyżej dotyczy rozwiązywania lewej strony schematu BOX, wzór (3.5). Aby mieć potrzebne tu wektory

$$F_s, s = 0, 1, \cdots, lp - 1$$

musimy te wektory utworzyć, pamiętając o tym, że będziemy musieli sięgnąć do sąsiednich procesorów. Można to zrealizować na przykład przy pomocy odpowiednio użytych komend **MPI_Bsend**$(\cdots)$ i **MPI_Recv**$(\cdots)$. Musimy także **pamiętać o warunku brzegowym**, który należy włączyć w prawą stronę układu, a dokładniej w wektor F_0.

APPENDIX

Calling $\boldsymbol{MPI}$ comands needed in realizing the above descripted program (FORTRAN version).

- **INITIATION**

$$\boldsymbol{call\ MPI\_Init(ierr)}$$

$$\boldsymbol{call\ MPI\_Comm\_Size(MPI\_Comm\_World, size, ierr)}$$

$$\boldsymbol{call\ MPI\_Comm\_Rank(MPI\_Comm\_World, rank, ierr)}$$

$$\boldsymbol{lp = size}$$

$$\boldsymbol{s = rank}$$

$$\boldsymbol{MPI\_Buffer\_Attach(BUFF, size, ierr)}$$

where

$$\boldsymbol{size \geq MPI\_Bsend\_Overhead}$$

- **WHEN FORMING THE RHS OF THE BOX-SCHEME**

$$\boldsymbol{if(s\ .\ lt\ .\ (lp-1))then}$$

$$\boldsymbol{call\ MPI\_Bsend(X(R), 1, MPI\_Real8, s+1, s,}$$

$$\boldsymbol{MPI\_Comm\_World, ierr)}$$

$$\boldsymbol{endif}$$

$$\boldsymbol{if(s\ .\ gt\ .\ 0)then}$$

$$\boldsymbol{call\ MPI\_Recv(STOCK, 1, MPI\_Real8, s-1, s-1,}$$

$$\boldsymbol{MPI\_Comm\_World, status, ierr)}$$

$$\boldsymbol{endif}$$

- **WHEN FORMING THE SCHUR SYSTEM**

$$\boldsymbol{call\ MPI\_Allgather}$$

$$\boldsymbol{(e_R^T \tilde{X}_s, 1, MPI\_Real8,}$$

$$\boldsymbol{RHS, 1, MPI\_Real8, MPI\_Comm\_World, info)}$$

- **AT THE END OF THE PROGRAM**

$$call\ MPI\_Buffer\_Detach(BUFF, size, ierr)$$

$$call\ MPI\_Finalize(ierr)$$

ZADANIE (4.1). Uogólnij wyżej opisany algorytm tak, aby mógł służyć do rozwiązywania wieloprocesorowego wkładów dwudiagonalnych o zmiennych współczynnikach

$$a_0, a_1, \cdots a_M$$

$$b_1, b_2, \cdots, b_M.$$

i zadanej prawej stronie. Napisz podprogram realizujący to zadanie.