Metody numeryczne zadanie 1

Bartosz Kucypera

19 listopada 2023

Niech $A \in \mathbb{R}^{N \times N}$ silnie diagonalnie dominującą macierzą w postaci Hessenberga
Niech s.d.d. = silnie dominująca diagonalnie.

Algorytm wyznaczania rozkładu LU dla A

Wykorzystam rekurencyjny algorytm rozkładu LU z wykładu, skorzystam ze specyficznej formy A i wykonam go w $O(N^2)$.

$$A = \left(\begin{array}{c|c} a_{11} & w^T \\ \hline v & A' \end{array}\right)$$

Niech $c = \frac{1}{a_{11}}$ (z postaci A wiemy, że $a_{11} \neq 0$). Wiemy też, że $v = \begin{bmatrix} v_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$, co znacznie ułatwia obliczenia.

Zachodzi

$$A = \left(\begin{array}{c|c} 1 & 0 \\ \hline c \cdot v_1 & \\ 0 & \\ \vdots & I_{N-1} \\ 0 & \end{array}\right) \left(\begin{array}{c|c} a_{11} & w^T \\ \hline 0 & A' - cvw^T \end{array}\right)$$

Macierz $A' - cvw^T$ jest postaci Hessenberga (cvw^T to macierz mająca tylko pierwszy wiersz niezerowy). Załóżmy, że jest też s.d.d. (dowód na dole).

Możemy wtedy rekurencyjnie szukać rozkładu LU dla macierzy kwadratowej rozmiaru o jeden mniejszego.

Dla przypadku bazowego $N = 1$

$$A = (a)$$

neich jej rozkład LU to

$$A = (a) = (1)(a)$$

Niech, więc $L'U'$ będzie rekurencyjnie znalezionym rozkładem LU macierzy $A' - cvw^T$.

Wtedy rozkład LU macierzy A to:

$$A = \left(\begin{array}{c|c} 1 & 0 \\ \hline c \cdot v_1 & \\ 0 & \\ \vdots & L' \\ 0 & \end{array}\right) \left(\begin{array}{c|c} a_{11} & w^T \\ \hline 0 & P' \end{array}\right)$$

Wykonujemy $O(N)$ kroków zmniejszając problem i w każdym z nich wykonujemy nie więcej niż $O(N)$ operacji (bo koszt wyliczenia $A' - cvw^T$ w najgorszym wypadku to N).
Czyli cały algorytm działa w $O(N^2)$.

Pokażmy jeszcze, że $A' - cvw^T$ jest s.d.d.
Niech $n = N - 1$.
Dla pierwszego wiersza chcemy by zachodziła nierówność, (dla reszty wierszy jest to oczywiste bo A jest s.d.d., więc i A' jest s.d.d.):

$$|a'_{11} - w_1 v_1 c| \geq \sum_{i=2}^{n} |a'_{1i} - w_i v_1 c|$$

mamy:

$$|a'_{11} - w_1 v_1 c| \geq |a'_{11}| - |w_1 v_1 c|$$

Ponieważ A jest s.d.d. mamy:

$$|a_{22}| > \sum_{i=1, i \neq 2}^{N} |a'_{2i}| = |a_{21}| + \sum_{i=3}^{N} |a_{2i}|$$

$$|a_{11}| > \sum_{i=2}^{N} |a_{1i}| \text{ czyli } \left| \frac{\sum_{i=2}^{N} |a_{1i}|}{a_{11}} \right| < 1$$

zachodzi więc:

$$|a_{22}| > \left| \frac{\sum_{i=2}^{N} |a_{1i}|}{a_{11}} \right| |a_{21}| + \sum_{i=3}^{N} |a_{2i}|$$

przypominamy sobie, że $a_{21} = v_1$, $\frac{1}{a_{11}} = c$, $a_{1i} = w_{i-1}$ dla $i \geq 2$ i $a_{2i} = a'_{1i-1}$, dla $i \geq 2$, czyli powyższą nierówność można zapisać jako:

$$|a'_{11}| > |w_1 v_1 c| + \sum_{i=2}^{n} |a'_{1i}| + |w_i v_1 c|$$

i z nierówności trójkąta:

$$\sum_{i=2}^{n} |a'_{1i}| + |w_i v_1 c| \geq \sum_{i=2}^{n} |a'_{1i} - w_i v_1 c|$$

czyli zachodzi:

$$|a'_{11}| > |w_1 v_1 c| + \sum_{i=2}^{n} |a'_{1i} - w_i v_1 c|$$

czyli:

$$|a'_{11} - w_1 v_1 c| \geq |a'_{11}| - |w_1 v_1 c| > \sum_{i=2}^{n} |a'_{1i} - w_i v_1 c|$$

więc macierz $A' - cvw^T$ jest silnie dominująca diagonalnie.

Kod w matlabie:

```
% LU Factorization for strongly diagonal dominant Householders matrix
function [L, U] = LUFH(A)
    N = size(A, 1);
    % będziemy budować L i U "w miejscu"
    L = eye(N);
    U = A;

    for i = 1:(N-1)
        c = 1/U(i, i);
        % ustawiamy v1*c z opisu
        L(i+1, i) = U(i+1, i) * c;
        % zerujemy pierwszą kolumne pod diagonalą
        U(i+1, i) = 0;

        % wyliczamy A' - c .* v * transpose(w)
        for j = i+1:N
            U(i+1, j) -= c * U(i, j) * A(i+1, i);
        end
    end
end
```

Algorytm rozwiązujący układ równań z macierzą A

Skoro mamy już maszynkę do rozkładu LU macierzy A, to rozwiązanie układu równań z tą macierzą jest bardzo proste.

Niech $b \in \mathbb{R}^N$ zadanym wektorem.
Szukamy takiego $x \in \mathbb{R}^N$, że $Ax = b$.

Niech, więc $A = LU$ (wyznaczone kosztem $O(N^2)$).
Możemy najpierw rozwiązać równanie $Ly = b$ i potem $Ux = y$.
L dolnotrójkątna, więc równanie $Ly = b$ rozwiązujemy w $O(N^2)$ (podstawieniami).
Tak samo skoro U górnotrójkątna to $Ux = y$ rozwiązujemy w $O(N^2)$.

Czyli cały algorytm zajmuje nam $O(N^2)$.

Kod w matlabie:

```
% LUFH - Linear Equations Solver
function [x] = LUFH_LES(A, b)
    % korzystamy z wcześniejszego algorytmu
    [L, U] = LUFH(A);
    N = size(b, 1);

    y = zeros(N, 1);
    % pierwszą zmienną wyliczamy ręcznie
    y(1) = b(1)/L(1, 1);
    for i = 2:N
        % podstawiamy już wyliczone zmienne i "przerzucamy" na drugą stronę równania
        for j = 1:(i-1)
            b(i) -= L(i,j)*y(j);
        end
        % wyliczamy kolejną zmienną
        y(i) = b(i)/L(i, i);
    end

    x = zeros(N, 1);
    % pierwszą zmienną wyliczamy ręcznie
    x(N) = y(N)/U(N,N);
    for i = (N-1):-1:1
        % podstawiamy już wyliczone zmienne i "przerzucamy" na drugą stronę równania
        for j = (i+1):N
            y(i) -= U(i, j)*x(j);
        end
        % wyliczamy kolejną zmienną
        x(i) = y(i)/U(i, i);
    end
end
```