22 grudnia 2017

Jakub Zbrzezny
Grupa e6

5. Rozwiązywanie równania macierzowego AX = B, gdzie $A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times m}, m \geq 1$, metodą Crouta. Obliczanie det(A) na podstawie wyznaczonego rozkładu.

1 Opis metody.

W celu rozwiązania równania macierzowego AX= B zastosuję metodę Crouta. Metoda Crouta polega na wyznaczeniu rozkładu LU macierz A, gdzie $A \in \mathbb{R}^{n\times n}$, L jest macierzą dolnotrójkątną wymiaru n x n, a U macierzą górnotrójkątną wymiaru n x n (U na diagonali ma same jedynki).

Macierz L:

$$L = \begin{pmatrix} l_{11} & 0 & 0 & \dots & 0 \\ l_{21} & l_{22} & 0 & \dots & 0 \\ l_{31} & l_{32} & l_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \dots & l_{nn} \end{pmatrix} \tag{1}$$

$l_{ij} \in \mathbb{R}$ dla każdych $i, j = 1, 2, ..., n$.

Macierz U:

$$U = \begin{pmatrix} 1 & u_{12} & u_{13} & \dots & u_{1n} \\ 0 & 1 & u_{23} & \dots & u_{2n} \\ 0 & 0 & 1 & \dots & u_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix} \tag{2}$$

$u_{ij} \in \mathbb{R}$ dla każdych $i, j = 1, 2, ..., n$.

Otrzymam równanie LUX = B. Żeby otrzymać macierz X, trzeba rozwiązać poniższy układ równań:

$\begin{cases} LY = B \\ UX = Y \end{cases}$, gdzie Y jest macierzą wymiaru n x m o wyrazach rzeczywistych.

Korzystam z własności wyznacznika det(AB) = det(A) * det(B).
Wyznacznik macierzy górnotrójkątnej U jest równy 1, ponieważna przekątnej macierzy U są same jedynki.
det(A) = det(L)*det(U) = det(L) = $(l_{11} * l_{22} * ... * l_{nn})$.

Przedstawię również twierdzenie o rozkładzie LU.

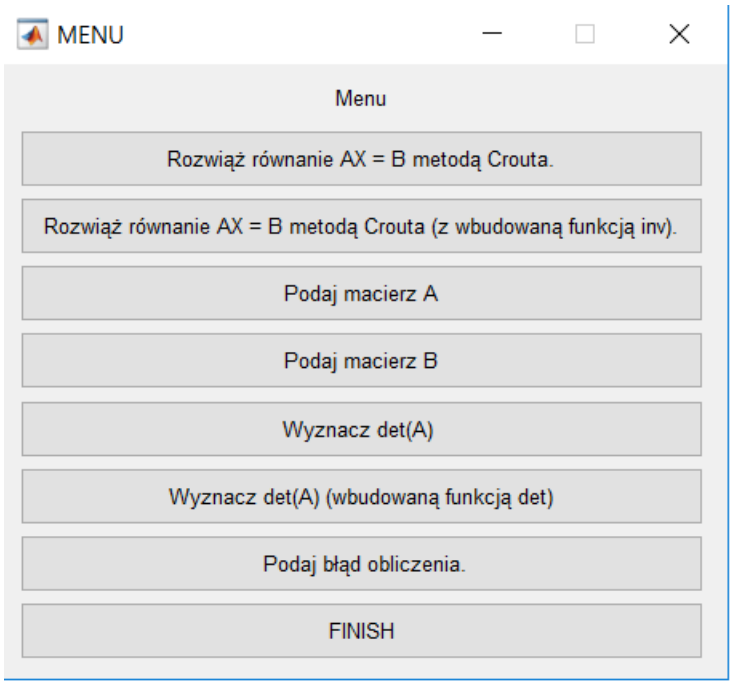
Twierdzenie: Jeśli $A \in \mathbb{R}^{n \times n}$ jest nieosobliwa i ma dwa rozkłady LU:
$A = L_1U_1$, $A = L_2U_2$, gdzie L_1, L_2 - trójkątne dolne, U_1, U_2 - trójkątne górne, to istnieje macierz diagonalna $D = diag(d_1, d_2, ..., d_n)$ taka, że $U_1 = U_2D$ i $L_1 = D^{-1}L_2$. Jeśli U_1, U_2 mają same jedynki na przekątnej, to $L_1 = L_2$ i $U_1 = U_2$.

Dowód:

$(U_2^{-1}/L_1) * L_1 = U_2 * L_2/L^{-1}$
$U_2^{-1} * U_1 = L_2 * L_1^{-1}$
Biorąc $D = U_2^{-1} * U_1$ otrzymamy $U_2 * U_2^{-1} * U1 = I * U_1 = U_1$ oraz $D^{-1} * L_2 = (U_2^{-1} * U_1)^{-1} * L_2 = (L_2 * L_1^{-1})^{-1} * L_2 = L_1 * L_2^{-1} * L_2 = L_1 * I = L_1$,
c.n.d.

2 Opis programu obliczeniowego.

Po uruchomieniu programu wyświetli się menu takie jak na poniższym obrazku.



Po wybraniu 1 opcji program wyznaczy rozkład LU macierzy A metodą Crouta. Program przyjmuje macierze A i B oraz zwraca macierz X, która jest rozwiązaniem równania macierzowego.
Żeby uruchomić funkcję, trzeba najpierw uruchomić 2 i 3 funkcję, ponieważ trzeba podać macierz A i B.
W przypadku, gdy poda się macierz A, która nie jest kwadratowa, program wyświetli komunikat: "Macierz A nie jest kwadratowa!", a następnie zakończy działanie.
Natomiast w przypadku, gdy liczba wierszy macierz A nie będzie równa liczbie wierszy macierzy B, program wyświetli komunikat: "Liczba wierszy macierzy A nie jest równa liczbie wierszy macierzy B!", a później zakończy działanie.
Następnie program wyznacza rozkład LU macierzy A, a nastęnie rozwiązuje równania macierzowe LY = B i UX = Y

W celu rozwiązania tych równań program korzysta z funkcji rozwiązującej równanie LY = B oraz z funkcji rozwiązującej równanie UX = Y.
W pierwszej funkcji wykorzystywana jest metoda Forward Substitution (zaczyna obliczać elementy macierzy Y na pierwszym wierszu i kończy na ostatnim.), natomiast w drugiej metoda Back Substitution (zaczyna obliczać elementy macierz X na ostatnim wierszu i kończy na pierwszym).

Po wybraniu 2 opcji program wyznacza rozwiązanie równania, wykorzystując wbudowaną metodę inv.

Po wybraniu 3 opcji program każe podać macierz A.

Po wybraniu 4 opcji program każe podać macierz B.

Po wybraniu 5 opcji program wyznaczy det(A), korzystając z funkcji, która liczy iloczyn wszystkich elementów na przekątnej.
Po wybraniu 6 opcji program obliczy det(A), korzystając z wbudowanej funkcji det.

Po wybraniu 7 opcji program poda błąd przybliżenia poprzez obliczenie wartości e równej norm(A-L*U)/norm(A), żeby sprawdzić, jak duży jest błąd przybliżenia (im wartość |e| jest większa, tym większy jest błąd przybliżenia).

Po wybraniu 8 opcji program zakończy działanie.

3 Przykłady obliczeniowe.

Macierz A	Macierz B	Bł. przybliżenia.	Błąd wzgl.	Bł. wzgl. (według wbud. metody inv)
eye(20)	eye(20)	0	0	0
pascal(20)	A*eye(20)	0	0	0
magic(20)	A*pascal(20)	NaN	NaN	120.6211
hilb(20)	A*pascal(20)	2.7903e-17	5.7823	2.0097
pascal(20)	A*hilb(20)	0	79.8446	110.8885
pascal(50)	A*hilb(50)	5.2511e-16	2.3962e+13	9.5197e+16

4 Analiza wyników.

Zbadam wskaźnik uwarunkowania macierzy A, współczynnik stabilności, poprawności.

A	B	det(A)	det(A) (f. wb.)	Wsk. uwar. A	Wsp. stab.	Wsp. pop.
eye(20)	A	1	1	1	0	0
pascal(20)	pascal(20)	1	27.8263	3.7274e+29	0	0
magic(20)	A*pascal(20)	NaN	-6.0892e-224	8.2375e+17	NaN	NaN
hilb(20)	A*pascal(20)	-2.5289e-194	7.4534e-196	6.5084e+18	2.7450e-18	8.8854e-18
pascal(20)	A*hilb(20)	1	27.8263	2.2465e+21	3.5542e-20	2.8938e-19
pascal(50)	A*hilb(50)	3.0260e+224	-Inf	3.7274e+29	6.4287e-17	2.0281e-30

Macierz A	Macierz B	Wsp. stab. (z met. inv)	Wsp. pop. (z met. inv)
eye(20)	eye(20)	0	0
pascal(20)	A*eye(20)	0	0
magic(20)	A*pascal(20)	1.6782e-17	3.4958e-17
hilb(20)	A*pascal(20)	9.5401e-19	1.5291e-19
pascal(20)	A*hilb(20)	4.9361e-20	1.5291e-19
pascal(50)	A*hilb(50)	2.5540e-13	3.6066e-33

Zatem metoda daje dokładniejsze wyniki, gdy macierz A jest lepiej uwarunkowana. Przy dużym wskaźniku uwarunkowania A metoda daje mniej dokładny wynik. Przy macierzy magic(20) metoda zawodzi, ponieważ macierz magic(20) nie jest ani symetryczna, ani dodatnio określona. Ponadto wskaźnik uwarunkowania jest rzędu 10^{17}.
Można łatwo zauważyć, że bezpośrednie rozwiązywanie równania AX = B z wykorzystywaniem wbudowanej metody inv daje o wiele większe błędy niż w przypadku mojej metody.
Funkcja licząca wyznacznik na podstawie rozkładu LU macierzy A daje lepsze wyniki niż wbudowana funkcja det, ponieważ liczy tylko iloczyn wszystkich elementów na przekątnej macierzy L.

5 Kody źródłowe wszystkich procedur.

1. Program.

```
% MENU
clear
clc
zakoncz_program = 8;
kontrol = 1;
while kontrol ~= zakoncz_program
    kontrol=menu('Menu', 'Rozwiaz rownanie AX = B metoda Crouta.',
     'Rozwiaz rwwnanie AX = B metoda Crouta (z wbudowana funkcja inv.'),
      'Podaj macierz A', 'Podaj macierz B', 'Wyznacz det(A) ', 'Wyznacz det(A)
      (wbudowana funkcja det)', ' Podaj blad przyblizenia.', 'FINISH');
    switch kontrol
        case 1
            [a, b] = size(A);
            if (a ~= b )
                disp('Macierz A nie jest kwadratowa!');
                continue;
            end
            n = max(size(A));
            [c, d] = size(B);
            if (n ~= c)
                disp('Liczba wierszy macierzy A nie jest rowna liczbie wierszy
                macierzy B!');
                continue;
            end
            L = zeros(n);
            U = eye(n);

            for k = 1 : n
                for i = k : n
                    L(i,k) = A(i, k) - MojaSuma(i, k, 1, k - 1, L, U);
                end
                for j = k + 1 : n
                    U(k, j) = (A(k, j) - MojaSuma(k, j, 1, k - 1, L, U)) / L(k, k);
                end
            end
            y = Rozwiaz_uklad_LYB(L, B); % Moja funkcja -tutaj L jest
            trojkatna dolna
            x = Rozwiaz_uklad_UXY(U, y);  % Moja funkcja -tutaj U jest
            trojkatna gorna
        case 2
            [a, b] = size(A);
            if (a ~= b )
```

```
                disp('Macierz A nie jest kwadratowa!');
                continue;
            end
            n = max(size(A));
            [c, d] = size(B);
            if (n ~= c)
                disp('Liczba wierszy macierzy A nie jest rowna liczbie wierszy
                 macierzy B!');
                continue;
            end
            x = A\B;
        case 3
            A = input('Podaj A.');
        case 4
            B = input('Podaj B.');
        case 5
            wyz = wyznacznik_macierzy_trojkatnej(L);
            % wyznacznik_macierzy_trojkatnej(U) = 1, bo na przekÄ...tnej ma 1-ki
            disp('Wyznacznik macierz A wynosi:');
            disp(wyz);
        case 6
            disp('Wyznacznik macierz A wynosi:');
            disp(det(A));
        case 7
            e=norm(A-L*U)/norm(A);
            disp('BĹ,Ä...d przybliĹŕenia wynosi: ');
            disp(e);
        case 8
            disp('Koniec programu');
            return;
    end
end
```

2. Funkcja pomocnicza rozwiązująca układ LY = B.

```
function [Y] = Rozwiaz_uklad_LYB(L, B)
[n, m] = size(B);
Z = zeros(m);
Y = [zeros(n) Z(:,n+1:n)];

for k = 1 : n
    for i = 1:m
        Y(k,i) = (B(k,i) - MojaSuma(k, i, 1, k - 1, L, Y)) / L(k,k);
    end
end
```

```
end
```

3. Funkcja pomocnicza rozwiązująca układ UX = Y.

```
function [X] = Rozwiaz_uklad_UXY(U, Y)
[n, m] = size(Y);
Z = zeros(m);
X = [zeros(n) Z(:,n+1:n)];

for k = n:-1:1
    for i = 1:m
        X(k,i) = Y(k,i) - MojaSuma(k, i, k + 1, n, U, X);
    end
end

end
```

4. Funkcja pomocnicza licząca sumę od 1 do m L(i,p)*U(p,k).

```
function y = MojaSuma(i, k, l, m, L, U)
% funkcja liczaca sume od l do m L(i,p)*U(p,k)

y = 0;
for p = l : m
    y = y + L(i,p) * U(p,k);
end
```

5. Funkcja pomocnicza licząca wyznacznik macierzy trójkątnej A.

```
function wyz = wyznacznik_macierzy_trojkatnej(A)
% funkcja liczaca wyznacznik macierzy trojkatnej A
% Jakub Zbrzezny

wyz = 1;
n = max(size(A));

for i = 1 : n
    wyz = wyz * A(i, i);
end
```

6. Funkcja pom. licząca błąd względny.

```
function e = blad_wzgledny(rozw_otrz, rozw_pop)
% Funkcja przyjmuje rozwiazanie otrzymane oraz rozwiazanie poprawne.
e = norm(rozw_otrz - rozw_pop) / norm(rozw_pop);
disp('Blad wzgledny:');
disp(e);
end
```

7. Funkcja pomocnicza licząca współczynnik stabilności.

```
function wsp_stab = wspolczynnik_stabilnosci(rozw otrz, rozw pop, A)
e = blad_wzgledny(rozw_otrz, rozw pop);
wsp_stab = e / cond(A);
disp('Wspolczynnik stabilnosci:');
disp(wsp_stab);
end
```

8. Funkcja pomocnicza licząca współczynnik poprawności.

```
function wsp_pop = wspolczynnik_poprawnosci(A, B, rozw_otrz)
wsp_pop = norm(B - A * rozw_otrz) / (norm(A) * norm(rozw_otrz));
disp('Wspolczynnik poprawnosci:');
disp(wsp_pop);
end
```