Politechnika Poznańska	
Sprawozdanie nr 1	Krystian Baran 145000
Metody rozwiązywania układów równań	09/05/2020

Cel ćwiczenia:

Celem ćwiczenia było zapoznanie się z metodami rozwiązywania układów równań, a także porównanie czasów wykonania każdego z wybranych algorytmów dla różnych wielkości macierzy.

Kody metod:

Metoda Cramera:

```
function x = cramer (W, B)
    x = [];
    w = det(W);
    n = 0;
    if w~=0
        n = length(W);
        for i=1:n
            X = W;
            X(:,i) = B;
            t = det(X);
            x(i) = t/w;
        endfor
    endif
endfunction
```

Metoda prosta LU:

```
function [U, L] = simpleLU (A)
    U = A;
    n = length(A);
    L = eye(n);
    for i=1:n-1
        pivot = U(i,i);
        for k=i+1:n
            a = U(k,i);
            b = -a/pivot;
            for j=i:n
                U(k,j) = U(k,j) + b*U(i,j);
            endfor
            L(k,i) = -b;
        endfor
    endfor
endfunction
```

Metoda LU z częściowym wyborem:

```
function [L, U, P] = partialLU (A)
    U = A;
    n = length(A);
    L = eye(n);
    P = eye(n);
    swap = 1;
    for i=1:n-1
        #wybór pivota
        pivot = U(i,i);
        swap = i;
        for k=i:n
            if abs(U(k,i)) > abs(pivot)
                pivot = U(k,i);
                swap = k;
            endif
        endfor
        #permutacja
        U = Swap(U,swap,i);
        P = Swap(P,i,swap);
        #wyzerowanie elementow
        for k=i+1:n
            a = U(k,i);
            b = -a/pivot;
            for j=i:n
                U(k,j) = U(k,j) + b*U(i,j);
            endfor
            U(k,i) = -b;
        endfor
    endfor
#split
    n=length(A);
    for i=2:n
        for j=1:i-1
            L(i,j) = U(i,j);
            U(i,j) = 0;
        endfor
    endfor
endfunction

function C = Swap (B, n, m)
    tmp = B(n,:);
    B(n,:) = B(m,:);
    B(m,:) = tmp;
    C = B;
endfunction
```

Rozkład LU z całkowitym wyborem:

```
function [L, U, P, Q] = completeLU (A)
    U = A;
    n = length(A);
    L = eye(n);
    P = eye(n);
    Q = eye(n);
    swap = 1;
    for i=1:n-1
        #wybór pivota
        pivot = U(i,i);
        swap = [i,i];
        for k=i:n
            for l=i:n
                if abs(U(k,l)) > abs(pivot)
                    pivot = U(k,l);
                    swap = [k,l];
                endif
            endfor
        endfor
        #permutacja
        U = SwapRow(U,swap(1),i);
        U = SwapCol(U,swap(2),i);
        P = SwapRow(P,i,swap(1));
        Q = SwapCol(Q,swap(2),i);
        #wyzerowanie elementow
        for k=i+1:n
            a = U(k,i);
            b = -a/pivot;
            for j=i:n
                U(k,j) = U(k,j) + b*U(i,j);
            endfor
            U(k,i) = -b;
        endfor
    endfor
#split
    n=length(A);
    for i=2:n
        for j=1:i-1
            L(i,j) = U(i,j);
            U(i,j) = 0;
        endfor
    endfor
endfunction

function C = SwapRow (B, n, m)
    tmp = B(n,:);
    B(n,:) = B(m,:);
    B(m,:) = tmp;
    C = B;
endfunction

function C = SwapCol (B, n, m)
    tmp = B(:,n);
    B(:,n) = B(:,m);
    B(:,m) = tmp;
    C = B;
endfunction
```

Program główny:

```
clc,clear, format long;
A = gallery("lehmer",4)
b = A*[1;1;1;1]
cond(A)

#cramer
x = cramer(A,b)
norm(x -[1,1,1,1],2)

#simple LU
[L, U] = simpleLU(A,b);
y = forwardRow(L,b);
x = backwardsRow(U,y)
norm(x -[1,1,1,1],2)

#partial LU
[L, U, P] = partialLU(A,b);
B = P*b;#transpose(b)
y = forwardRow(L,B);
x = backwardsRow(U,y)
norm(x -[1,1,1,1],2)

#complete LU
[L, U, P, Q] = completeLU(A,b);
B = P*b;
y = forwardRow(L,B);
y = backwardsRow(U,y);
x = Q*transpose(y)
norm(x - [1;1;1;1],2)

#czasy obliczania
n = 1600, p=0;
A = gallery("lehmer",n);
b = A*ones(n,1);
disp(cond(A));

#cramer
tic;
x = cramer(A,b);
p=toc;
disp("Cramer")
if ~isempty(x)
    disp(norm(x-ones(1,n)));
    disp(p);
endif

#simple LU
tic;
[L, U] = simpleLU(A,b);
y = forwardRow(L,b);
x = backwardsRow(U,y);
p=toc;
disp("Simple LU")
disp(norm(x-ones(1,n)));
disp(p);
```

```
#partial LU
tic;
[L, U, P] = partialLU(A,b);
B = P*b;#transpose(b)
y = forwardRow(L,B);
x = backwardsRow(U,y);
p=toc;
disp("Partial LU")
disp(norm(x-ones(1,n)));
disp(p);

#complete LU
tic;
[L, U, P, Q] = completeLU(A,b);
B = P*b;
y = forwardRow(L,B);
y = backwardsRow(U,y);
x = Q*transpose(y);
p=toc;
disp("Complete LU")
disp(norm(x-ones(n,1)));
disp(p);
```

Przebieg ćwiczenia:

Do analizy algorytmów rozwiązywania układów równań wybrano macierze z dostępnej biblioteki GNU Octave typu ***gallery(„lehmer”, N)***. Są to macierze Lehmera rozmiaru $N \times N$.
Poniżej przedstawiono macierz przykładową 4×4 i wektor wyrazów wolnych tak, aby rozwiązanie było wektorem jedynek. Poniżej macierz A i wektor wyrazów wolnych.

$$\begin{bmatrix} 1.000000000000000e+00 & 5.000000000000000e-01 & 3.333333333333333e-01 & 2.500000000000000e-01 \\ 5.000000000000000e-01 & 1.000000000000000e+00 & 6.666666666666666e-01 & 5.000000000000000e-01 \\ 3.333333333333333e-01 & 6.666666666666666e-01 & 1.000000000000000e+00 & 7.500000000000000e-01 \\ 2.500000000000000e-01 & 5.000000000000000e-01 & 7.500000000000000e-01 & 1.000000000000000e+00 \end{bmatrix}$$

$$b := \begin{bmatrix} 2.083333333333333 \\ 2.666666666666667 \\ 2.750000000000000 \\ 2.500000000000000 \end{bmatrix}$$

Za pomocą wbudowanej funkcji w GNU Octave, obliczono współczynnik uwarunkowania:

$$\text{cond(A)} = 12.20626419560859$$

Macierz jest zatem źle uwarunkowana i nie możemy się spodziewać bardzo dokładnych wyników. Mimo to przeprowadzona została analiza.

Dla metody Cramera uzyskane rozwiązanie w formacie *long* jest następujące:

$$\begin{bmatrix} 1.000000000000000e+00 \\ 9.999999999999999e-01 \\ 1.000000000000001e+00 \\ 1.000000000000001e+00 \end{bmatrix}$$

Gdzie norma błędu wynosi 1.047382306668854e-15. Widzimy, że otrzymane wartości są w aproksymacji szukany wektor jedynek.

Dla prostego rozkładu LU otrzymano następujące rozwiązanie:

$$\begin{bmatrix} 9.999999999999998e-01 \\ 9.999999999999997e-01 \\ 1.000000000000000e+00 \\ 9.999999999999999e-01 \end{bmatrix}$$

Norma błędu wynosi 6.080941944488118e-16. Dla tej metody widzimy, że trzy wartości wartość należy aproksymować.

Dla rozkładu LU z częściowym wyborem otrzymano następujące rozwiązanie:

$$\begin{bmatrix} 9.999999999999998e-01 \\ 9.999999999999997e-01 \\ 1.000000000000000e+00 \\ 9.999999999999999e-01 \end{bmatrix}$$

Norma błędu wynosi 6.080941944488118e-16. Dla tej metody widzimy, że rozwiązanie jest takie samo jak dla metody prostej LU.

Dla rozkładu LU z całkowitym wyborem otrzymano następujące rozwiązanie:

$$\begin{bmatrix} 9.999999999999996e\text{-}01 \\ 9.999999999999994e\text{-}01 \\ 1.000000000000001e{+}00 \\ 9.999999999999996e\text{-}01 \end{bmatrix}$$

Norma błędu wynosi 1.391105462616079e-15. Dla tej metody wyniki są podobne do poprzednich dwóch metod, z większą niedokładnością.

Metoda prosta rozkładu LU wraz z metodą częściowego wyboru okazały się najbardziej skuteczne dla tego typu macierzy, a metoda rozkładu LU z całkowitym wyborem okazała się najmniej skuteczna. Metoda Cramera, pomimo tego, że nie jest zwyczajnie skuteczna, ponieważ wymagane jest obliczenie różnych wyznaczników, dla tej macierzy jest trzecia najbardziej skuteczna.

Następnym krokiem, po sprawdzeniu, czy metody działają poprawnie, było wyznaczenie czasów obliczania każdej metody dla różnych wielkości macierzy, to jest:

$$[50, 100, 200, 400, 800, 1600, 3200, 6400, 12800]$$

Pomiar czasów został wykonany za pomocą funkcji ***tic*** i ***toc*** w GNU Octave. Dla kolejnych metod obliczone zostało także ratio wzrostów czasów obliczania. Otrzymane wyniki zapisano w tabeli poniżej.

n	Cramer	Simple LU	Partial LU	Complete LU
50	2.78E-02	9.58E-01	9.96E-01	1.89
100	1.77E-01	7.19	7.47	15.28
200	8.49E-01	57.14	57.43	116.68
400	-	565.93	515.61	792.81
800	-	2170.18	2211.28	5128.95
1600	-	17280.86	22414.6	-
3200	-	-	-	-
6400	-	-	-	-
12800	-	-	-	-

Tabela 1 - Czasy obliczania rozwiązania

n	Cramer	Simple LU	Partial LU	Complete LU
100/50	6.34	7.51	7.51	8.11
200/100	4.81	7.94	7.68	7.64
400/200	-	9.9	8.98	6.8
800/400	-	3.83	4.29	6.47
1600/800	-	7.96	10.14	-
3200/1600	-	-	-	-
6400/3200	-	-	-	-
12800/6400	-	-	-	-
Average	5.58	7.43	7.72	7.25

Tabela 2 – Ratio czasów obliczania rozwiązania

Metoda Cramera od n = 400 okazała się niemożliwa do zastosowania, ponieważ wartości wyznacznika głównej macierzy był taki mały, że przekraczał najmniejszą liczbę maszynową dodatnią i był aproksymowany do 0 przez program GNU Octave. Zatem nie są dostępne czasy dla tej metody. Pomimo tego widzimy, że dla innych metod ratio wzrostu czasu są mniej więcej tyle samo. Można powiedzieć że czasy rosną wykładniczo.

Poniżej przedstawiono normy błędów dla różnych wymiarów i współczynniki uwarunkowania macierzy.

n	cond(A)	Cramer	Simple LU	Partial LU	Complete LU
50	2487.03	8.59E-13	1.22E-12	1.22E-12	7.77E-13
100	10268.2	6.68E-12	9.63E-12	9.63E-12	5.42E-12
200	41942.7	0	7.75E-11	7.75E-11	4.83E-11
400	170087.84	-	6.56E-10	6.56E-10	3.80E-10
800	686437.38	-	4.98E-09	4.98E-09	2.39E-09
1600	2761575.22	-	4.25E-08	4.25E-08	-
3200	-	-	-	-	-
6400	-	-	-	-	-
12800	-	-	-	-	-

Tabela 3 – Normy błędów

Dla metody Cramera norma błędu wynosi 0 dla n = 200, więc mogłoby się wydawać, że metoda ta obliczyła najbardziej poprawnie rozwiązanie. Jednak maszyna nie ma możliwości obliczenia tak dokładnie wyniku, co oznacza, że liczby wykorzystane w tej metodzie są graniczne liczby maszynowe i są aproksymowane. Wynik zerowy może wynikać z tego, że liczby obliczeń są aproksymowane do 1, ponieważ wynik rozwiązania powinien być wektorem jedynek, otrzymujemy zerową normę błędu.

Uwarunkowanie macierzy widzimy, że jest mocno większe do 1 i zwiększa się coraz bardziej ze zwiększeniem wymiarów macierzy. Macierz tego typu jest zatem macierzą źle uwarunkowaną.

Poniżej przestawiono ponownię tabele czasów w godzinach. Ponieważ od n = 1600 czasy obliczania są znacznie duże, to kolejne czasy estymowano, mnożąc poprzedni czas razy średnią z ratio poprzednich czasów. Estymowane liczby zapisano kolorem czerwonym.

n	Simple LU	Partial LU	Complete LU
50	0	0	0
100	0	0	0
200	0.02	0.02	0.03
400	0.16	0.14	0.22
800	0.6	0.61	1.42
1600	4.8	6.23	10.33
3200	35.67	48.06	74.91
6400	265.04	370.96	543.23
12800	1969.4	2863.35	3939.11

Tabela 4 – Czasy estymowane w godzinach

Widzimy zatem, że od n = 3200 nie jest czasowo wykonalne obliczenie czasów działania programu, ponieważ przekraczają one dzień liczenia. Zatem dla tych wartości nie obliczono także normę błędu.

Ponieważ spodziewane czasy obliczania okazały się dużo mniejsze niż rzeczywiste czasy obliczania, nie obliczono także czas obliczania dla metody rozkładu LU z całkowitym wyborem elementów, ponieważ brakło na to czasu. Wstawiono zatem estymowany czas.

Wnioski:

Na podstawie otrzymanych wyników możemy powiedzieć, że złe uwarunkowanie macierzy wpływa na czas obliczania dla metod podobnych do metody Gaussa. Wnioskujemy też, że metoda Cramera nie jest stosowalna dla dużych wymiarów układów równań, należałoby poszukać bardziej skutecznej metody.

Wnioskujemy także, że metoda rozkładu LU z całkowitym wyborem jest metodą o największym czasie obliczeniowym dla dużych wymiarów macierzy „A”, zatem stosowne jest unikać tej metody, jeżeli to możliwe.

Zauważono także, że macierz Lehmera nie jest dobrą macierzą testową dla zbyt dużych rozmiarów.