POLITECZNIKA POZNAŃSKA NUMERYCZNA ALGEBRA LINIOWA			
Temat pracy:	Numeryczne rozwiązywanie układów równań z wykorzystaniem Octave.	Data oddania:	12.05.2022
Pracę wykonał:	Wiktoria Kaczmarek	Ocena:	

Naszym zadaniem jest rozwiązanie układu równań $Ax = B$, gdzie A jest macierzą pełną, a wektor wyrazów wolnych generujemy tak aby otrzymać wektor rozwiązań w postaci:

$$x = [1,1,1,..,1]^T.$$

Do deklaracji macierzy A wykorzystałam funkcję *A=galery("ris",n)*, gdzie n to rozmiar macierzy nxn. Macierz wygenerowana to symetryczna macierz Hankel'a, tworzona według wzoru:

$$A(i,j) = \frac{0.5}{n - i - j + 1.5}.$$

Do rozwiązania skorzystam z każdej z metody:

- Metoda Cramera,
- Rozkład LU,
- Rozkład LU z częściowym wyborem,
- Rozkład LU z całkowitym wyborem.

Każdą z metod przetestuje dla $n = [50; 100; 200; 400; 800; 1600; 3200;\ 6400; 12800]$ z ograniczeniem sprzętowym.

Zaczną od obliczenia normy błędu $\left||x^* - x|\right|_2$,gdzie: x- rozwiązanie dokładne, x^*- rozwiązanie otrzymane metodą numeryczną. Aby to porównywać skorzystam z formatu long e oraz funkcji *norm(x-[1,1,…,1],2).*

Tab. 1. Wyniki obliczeń normy błędu dla poszczególnych metod i wymiarów macierzy nxn.

n	Norma błędu			
	Cramer	LU	LU z częściowym wyborem	LU z całkowitym wyborem
50	3.616328897427208e-15	1.905784361335154e+15	3.663735981263017e-15	7.428545315910685e-15
100	1.325078039852354e-14	2.033528536138054e+17	6.035162558560572e-15	1.699185367494937e-14
200	1.250320137812093e-14	6.497109576478077e+18	1.155485647719850e-14	4.730162492227964e-14
400	2.711400117760946e-14	9.625679115795909e+20	2.864203270760085e-14	1.412736877758964e-13
800	6.888233792326469e-14	4.128448664100366e+21	5.009087111013216e-14	-
1600	NaN	4.313960509084270e+21	1.668107046463255e-13	-

Następnie przeprowadzam testy by poznać czas działania poszczególnych metod:

Tab. 2. Czas wykonywania metody [s] oraz uwarunkowanie macierzy wykorzystywanej do testów.

n	Uwarunkowanie macierzy	Czas wykonywania metody w sekundach			
		Cramer	LU	LU z częściowym wyborem	LU z całkowitym wyborem
50	3.0429	0.012894	0.034673	0.086658	1.1416
100	3.3162	0.072567	0.1939	0.4670	20.000
200	3.5905	0.4970	1.1394	1.6398	181.43
400	3.8658	4.9699	1.4715	12.735	916.67
800	4.1417	38.607	17.503	51.315	-
1600	4.4182	314.14	34.881	77.017	-
3200	4.6952	2939.2	232.88	948.97	-
6400	4.9726	-	-	-	-
12800		-	-	-	-

Tab.3. Ratio czasów wykonywania obliczeń przez program.

n/n	Ratio czasu wykonywania metody w sekundach			
	Cramer	LU	LU z częściowym wyborem	LU z całkowitym wyborem
100/50	5,627966	5,592248	5,389	17,51927
200/100	6,848843	5,876225	3,511349	9,0715
400/200	9,99979	1,291469	7,766191	5,052472
800/400	7,768171	11,89467	4,029446	-
1600/800	8,136866	1,992858	1,500867	-
3200/1600	9,356338	6,676414	12,32157	-

Wnioski:

Możemy zauważyć, że wybór metody powinien być zależny od tego czy chcemy uzyskać wynik szybko czy bardziej dokładny. W metodzie Cramera oraz rozkładach LU z częściowym wyborem i całkowitym wyborem norma błędu malała przy wzroście rozmiaru macierzy, jednak w rozkładzie LU norma błędu rosła wraz z wzrostem rozmiaru macierzy. Jednak warto zwrócić uwagę, że czas wykonywania algorytmu dla metody Cramera do n=800 był krótszy niż rozkładu LU z częściowym wyborem, przy większych rozmiarach krócej wykonywała się metoda z wykorzystaniem rozkładu LU z częściowym wyborem. Rozkład LU z całkowitym wyborem wykonywał się na tyle długo, że postanowiłam nie wykonywać dalszych obliczeń, lecz można zauważyć, że ma on najmniejszą normę błędu oraz maleje ratio wykonywania kolejnych prób. Patrząc na tabelę z ratio czasu możemy wywnioskować, że metoda Cramera ma proporcjonalny czas wykonywania dla wszystkich zbadanych rozmiarów, a rozkład LU z całkowitym wyborem woli pracować na większych macierzach.

Dodatkowo możemy stwierdzić, że wybrana przeze mnie macierz jest stosunkowo dobrze uwarunkowana, więc możemy założyć, że wyniki będą dość dokładne.

Kody źródłowe:

Cramer:

```
function [ODP]=cramer(n, F, C)
  wyzC=det(C);
  for i=1:n
    wektor=F;
    F=C(:,i);
    C(:,i)=wektor;
    k=det(C);
    ODP(i,1)=k/wyzC;
    wektor=F;
    F=C(:,i);
    C(:,i)=wektor;
   endfor
endfunction
```

LU:

```
function [L,U]=simpLU(A)
   [m,n]=size(A);
  for i=1:1:n-1
    r=i+1:1:n;
    A(r,i)=A(r,i)/A(i,i);
    A(r,r)=A(r,r)-A(r,i)*A(i,r);

  endfor
L=eye(n,n)+tril(A,-1);
U=tril(A);
```

LU z częściowym wyborem:

```
function [L, U, P] = pivot(A)
     [m, n] = size(A);
     L=eye(n);
     P=eye(n);
     U=A;
     for k=1:m-1
          [pivot,ind]=max(abs(U(k:m,k)));
          ind=ind+k-1;

          U([k,ind],k:m)=U([ind,k],k:m);
          L([k,ind],1:k-1)=L([ind,k],1:k-1);
          P([k,ind],:)=P([ind,k],:);

          for j=k+1:m
               L(j,k)=U(j,k)/U(k,k);
               U(j,k:m)=U(j,k:m)-L(j,k)*U(k,k:m);
          end
     end
end
```

LU z całkowitym wyborem:

```
function [L, U, P, Q] = completePivot(A)

  n = length(A);
  L = eye(n);
  P = eye(n);
  Q = eye(n);
  U = A;
  swap = 1;

  for i=1:n-1
    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

    U = SwapRow(U,swap(1),i);
    U = SwapCol(U,swap(2),i);
    P = SwapRow(P,i,swap(1));
    Q = SwapCol(Q,swap(2),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
     U(k,i) = -b;
   endfor

  endfor

  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
```

Całość:

```
clc,clear all, format long e
n=6400
A = gallery("ris",n);
B = A*ones(n,1);
cond(A)

%cramer
disp("cramer")
tic
n=size(A,1);
[ODP]=cramer(n, B, A);
norm(ODP-ones(n,1),2)
p=toc


%lu z czesciowy pivot
disp("LU z cz piv")
tic
[L, U, P] = pivot(A);
n=size(L,1);
y=macierz_dolno(n,P*B,L);
x=macierz_gorno(n,y,U);
norm(x-ones(n,1),2)
p=toc
```

```
%lu z calkowity pivot
disp("LU z cal pivot")
tic
[L, U, P, Q] = completePivot(A);
b=P*B;
y=macierz_dolno(n,b,L);
y=macierz_gorno(n,y,U);
x=Q*y;
norm(x-ones(n,1),2)
p=toc


%lu
disp("LU")
tic
[L,U]=simpLU(A);
y=macierz_dolno(n,B,L);
x=macierz_gorno(n,y,U);
norm(x-ones(n,1),2)
p=toc
```