Sprawozdanie MNUM
Projekt nr.1
Zadanie 1.1

Kajetan Kaczmarek

27 marca 2018

1. Opis zastosowanych algorytmów :

   (a) Do sprawdzenia dokładności maszynowej komputera wykorzystałem prosty algorytm dzielący liczbę 1 przez 2 przy każdej kolejnej iteracji tak długo aż nie była ona równa 0.Dla mojego komputera otrzymałem wynik 1076

   (b) Przy drugim poleceniu wykorzystałem algorytm faktoryzacji Cholesky'ego Banachiewicza. Najpierw utworzyłem funkcje generateA, generateB oraz generateC, tworzące zbiory danych zgodnie z podanymi specyfikacjami.Następnie użyłem pomocniczych funkcji do faktoryzacji Cholesky'ego zgodnie z podanymi wzorami

   $$l_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2}$$

   oraz

   $$l_{ji} = \frac{a_{ji} - \sum_{k=1}^{i-1} l_{jk} \cdot l_{ik}}{l_{ii}}$$

   Otrzymałem dzięki temu macierz która pomnożona przez swoją tranzpozycję odtworzy zadaną macierz A.Następnie używam otrzymanej macierzy L do rozwiązania równania. Błędy dla kolejnych, coraz większych, iteracji, kształtują się następująco :

   (c) W metodzie Gaussa-Siedel'a rozbijamy zadaną macierz na trzy - L - część pod przekątną , D - wyłącznie przekątną oraz U - część ponad przekątną. W moim programie służy do tego podfunkcja decompose.Następnie przeprowadzamy kolejne iteracje modelu

   $$Dx * (i + 1) = -Lx * (i + 1) - Ux * (i) + b$$

   Dzięki faktowi że połowa macierzy L jest wypełniona zerami, zachowując odpowiedni porządek, możemy użyć ww. wzoru pomimo występowania po prawej stronie elementów x+1

2. Wydruki programów

   (a) Program 1 - Sprawdzenie dokładności maszynowej

```
x = 1;
i = 1;

%Program liczy w pętli ile razy należy podzielić liczbę przez 2 aby dla
%komputera wynosiła ona 0

while(x ~= 0)
  x = x/2;
  i = i + 1;
end
disp(i);
```

   (b) Program 2 - Rozwiązanie układu równań metodą faktoryzacji Cholesky'ego-Banachiewicza

- Główna funkcja (wrapper) dla drugiego zadania

```
n =10 ; % initial value of i
i = 1;
% Errors for each set of data for given n
R1 = zeros( 2,20 );
R2 = zeros( 2,20 );
R3 = zeros( 2,20 );
%Timers
tic
t1 = toc;
t = toc;

while(t - t1 <180 )
    %Factorization of first set of data

    [ A , B ] = generateA(n);

    X = Cholesky_Solve(A,B);

    R1(1,i) = sqrt(sum((A*X -B).^2));
    R1(2,i) = n;
    %Factorization of second set of data

    [ A , B ] = generateB(n);

    X = Cholesky_Solve(A,B);

    R2(1,i) = sqrt(sum((A*X -B).^2));
    R2(2,i) = n;

    %Factorization of third set of data

    [ A , B ] = generateC(n);

    X = Cholesky_Solve(A,B);

    R3(1,i) = sqrt(sum((A*X -B).^2));
    R3(2,i) = n;
    disp(n);
    i = i+1;
    n = 10*(2^(i-1));

    t1 = t;
    t = toc;
    disp(t - t1);
end
```

- Generator zbioeru danych A

```
%Generating set of data a)
function [A,B] = generateA(n)
A = zeros( n );

for  i = 1:n
    for j = 1:n
        if i==j
            A( i , j ) = 10;
        elseif i == j - 1 || j == i - 1
            A( i , j ) = 2.5;
        end
    end
end

B = zeros( n , 1 );

for i = 1 : n
    B(i) = 4 - 0.5 * i;
end
```

- Generator zbioeru danych B

```
%Generating set of data b)
function [A,B] = generateB(n)
A = zeros( n );

for  i = 1:n
    for j = 1:n
        if i==j
            A( i , j ) =  3*n^2 - i;
        else
            A( i , j ) = i + j +1;
        end
    end
end

B = zeros( n , 1 );

for i = 1 : n
    B(i) = 2.5 + 0.6 * i;
end
```

- Generator zbioeru danych C

```
%Generating set of data c)
function [A,B] = generateC(n)

A = zeros( n );

for  i = 1:n
    for j = 1:n
        if i==j
            A( i , j ) =  0.1 * n + 0.3 * i;
        else
            A( i , j ) = 1 / ( 6 * ( i + j +1 ) );
        end
    end
end

B = zeros( n , 1 );

for i = 1 : n
    B(i) = 5 / ( 3 * i );
end
end
```

- Funkcja faktoryzująca

```
%Funkcja przeprowadzająca faktoryzację metodą Cholesky'ego-Banachiewicz
function [L] = Cholesky_Banachiewicz(A)
sz = size(A); %Informację o n pobieramy bezpośrednio z macierzy przekaz
n = sz(1);
L = zeros(n);%Inicjalizujemy L zerami

for  j = 1:n
    for i = 1:n
        if i==j %Wartości poszczególnych elementów L są obliczane zgodn
                %ze wzorami ze skryptu
           L(i,j) = sqrt ( A ( i , i ) - sum(L( i  , 1 : i - 1 ).^2) )
        elseif( j > i )% Pomocnicze macierze, pojedyncze wiersze
           %używane w algorytmie - jako że nie miałem wcześniej
           %doświadczenia z MatLabem pozostawiam je w takiej formie,
           %uprzednio wprowadziłem je celem zdebugowania programu
           L1 = L(i,1:end);
           L2 = L(j,1:end);
           L(j,i) = (A( j , i) - sum(L2.*L1))/L( i , i );
        end
    end
end
return;
```

- Propagacja wsteczna

```
%Funkcja realizuje metodę propagacji wstecznej,
%używana jest przeze mnie w algorytmie Cholesky'e

function [X] = backwards_Substitution(L,B)
sz = size(B);
n = sz(1);
X = zeros(n,1);
X(n) = B(n)/L(n,n);
X(n-1) = (B(n-1)-L(n-1,n)*X(n))/L(n-1,n-1);
for  k = n -2 : -1 : 1
     L1 = L(k,k+1:end);
     X1 = X(k+1:n);
     X(k) = (B(k) - sum(L1.*X1'))/L(k,k);
end
return;
```

- Propagacja wprzód

```
function [Y] = forward_Substitution(L,B)
sz = size(B);
n = sz(1);
Y = zeros(n,1);

for  i = 1:n
    Y(i) = (B(i) - sum(L(i,1:i-1)*Y(1:i-1)))/L(i,i);
end
return;
```

- Funckja rozwiązująca równania

```
%Funkcja realizująca rozwiązanie układu równań metodą Cholesky'ego.Jako
%wyjście przekazuje rozwiązanie X, a jako argumenty przyjmuję zadane
%macierze A i B

function [X] = Cholesky_Solve(A , B)

%Pierwszym etapem jest faktoryzacja
L = Cholesky_Banachiewicz(A);
%Następnie rozwiązujemy Ly=B
Y = forward_Substitution(L,B);
%Z użyciem wyliczonych Y rozwiązujemy Ax=B
X = backwards_Substitution(L',Y);
```

(c) Program 3 - Rozwiązanie układu równań metodą Gaussa-Seidela

- Główna funkcja (wrapper) dla trzeciego zadania

```
%Wrapper function of Seidel-Gauss problem
[ A , B ] = generateB(n);
[X,k] = Gauss_Seidel( A , B, 10, 0.001 );

r = A*X - B ;

disp(r);
```

- Funkcja rozkładająca na 3 macierze

```
%Funkcja pomocnicza służąca do rozkładu macierzy zadanej przy algorytmie
%Gaussa-Seidela na 3 części wymagane przy jego użyciu

function [L , D , U] = decompose(A)
sz = size(A);% Pobieramy n z rozmiaru macierzy
n = sz(1);
%Inicjalizujemy macierze zerami
L = zeros(n);
D = zeros(n);
U = zeros(n);

for i = 1:n
    for j= 1:n
       if(i==j)%Elementy przekątnej wpisywane są do D
           D( i , j ) = A(i,j);
       elseif(i<j) % Elementy pod przekątną do L
           L( i , j ) = A(i,j);
       else %Elementy nad przekątną do U
           U( i , j ) = A(i,j);
       end
    end
end
```

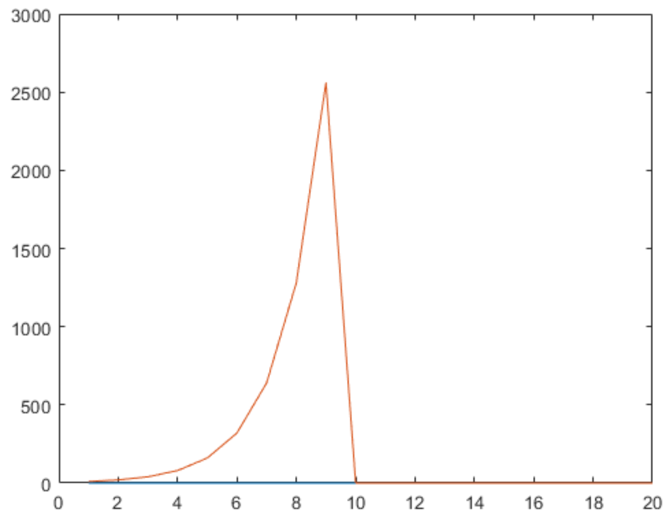
- Funkcja rozwiązująca równania

```
%Funkcja realizuje rozwiązanie układu równań metodą Gaussa-Seidela.Jako
%parametry wyjściowe zwraca rozwiązanie układu X oraz ilość iteracji k.Jako
%parametry podajemy macierze zadane A i B oraz błąd przybliżenia delta

function [X,k] = Gauss_Seidel(A,B,n,delta)
k = 1 ;
%Funkcja pomocnicza rozkłada macierz zadaną na części pierwsze
[L, D, U] = decompose(A);

%Inicjalizujemy X zerami
X = zeros(n , 1);
%Pomocnicza macierz przechowująca X z poprzedniej iteracji
X1 = ones(n , 1);
%Sprawdzamy warunek wyjściowy tj. dokładność rozwiązania
while(sum(abs(X-X1)) >= delta)
    X1 = X;
    %Obliczamy pomocniczą zmienną W = UX-B
    W = U*X - B;
    for i = 1 : n
        %Obliczamy kolejne iteracje rozwiązania zgodnie z wzorem ze skryptu
       X(i) =(-W(i) - sum(L(i,1:end).*X(1,1:end)))/D(i,i) ;
    end
    k = k + 1;
end
```

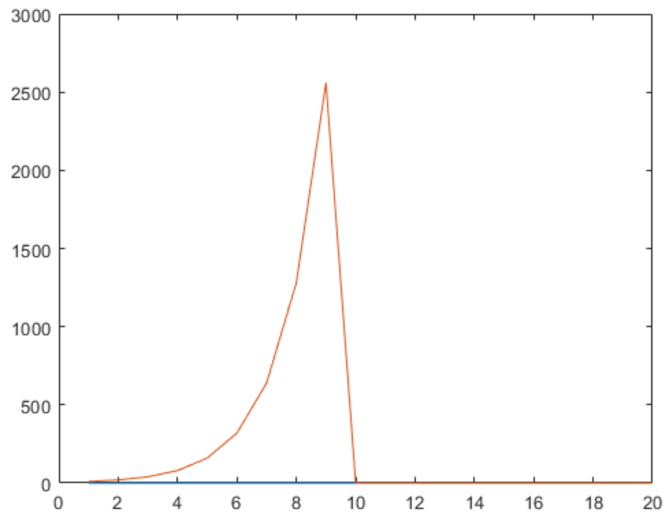
3. Wyniki oraz wnioski

- Dla pierwszego algorytmu otrzymałem wynik rzędu ponad 1000 (dokładnie 1076) Liczba ta wydaje mi się bardzo duża. niemniej algorytm jest prosty i nie mogę doszukać się w nim błędu

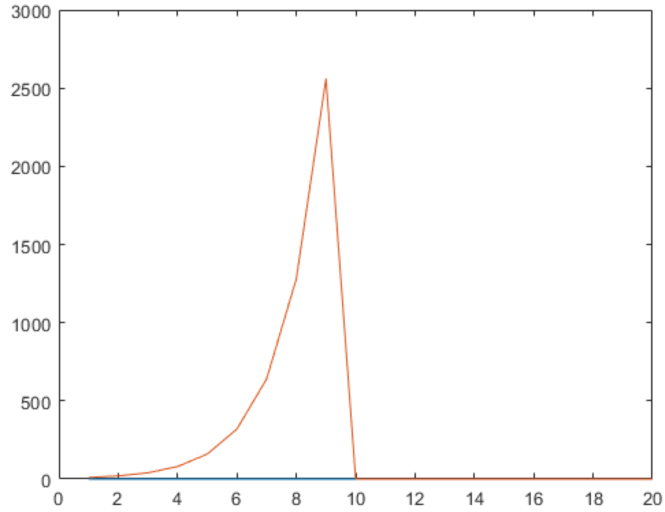
- Jeśli chodzi o drugie zadanie, zadany czas 3 minut został przekroczony dla n>2560. Otrzymałem następujące wyniki normy residuum -
  - Zbiór danych 1



  - Zbiór danych 2



  - Zbiór danych 3



- Jak widać, dane są bardzo zbliżone dla wszystkich zestawów danych - rokuje to na korzyść algorytmu.Z drugiej strony, błąd wzrastał stale aż do zakończenia pracy algorytmu - prawa strona, równa 0, to elementy dla których nie znaleziono rozwiązania z powodu przekroczenia czasu
- Dla trzeciego zadania otzymałem błędy rzędu około 15