MNUM - Projekt 1, AiR sem-2

zadanie 1.14

1. Napisać solwer, czyli uniwersalną procedurę, rozwiązującą układ n równań liniowych Ax = b, wykorzystując podaną metodę: eliminacji Gaussa z częściowym wyborem elementu głównego.

Metoda eliminacji Gaussa z częściowym wyborem elementu głównego pozwala uniknąć podczas rozwiązywania układu równań liniowych nieprzyjemnej sytuacji polegającej na wystąpieniu zera na diagonali macierzy. Gdyby taki właśnie układ rozwiązywano tradycyjną metodą eliminacji Gaussa, przy obliczaniu współczynników w mianowniku wystąpiłoby zero, co uniemożliwia nam dalszy proces rozwiązywania równań. Dzięki metodzie eliminacji Gaussa z częściowym wyborem elementu głównego taka sytuacja nie stanowi dla nas problemu.

Opis algorytmu:

Załóżmy, że mamy macierz 3x3, oznaczmy ją jako „A" (dla większych macierzy kwadratowych algorytm nie zmienia się):

$$\begin{matrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{matrix}$$

oraz wektor wyników równania:

$$\begin{matrix} b_{11} \\ b_{21} \\ b_{31} \end{matrix}$$

Liczby obok danego elementu macierzy oznaczają współrzędne elementu w macierzy czy wektorze (pierwsza cyfra oznacza wiersz, druga cyfra kolumnę). Z tych danych tworzymy macierz rozszerzoną Ab składającą się z macierzy A i wektora b:

$$\begin{matrix} a_{11} & a_{12} & a_{13} & b_{11} \\ a_{21} & a_{22} & a_{23} & b_{21} \\ a_{31} & a_{32} & a_{33} & b_{31} \end{matrix}$$

Wprowadźmy oznaczenie i (jako współrzędna wierszowa) oraz j (jako współrzędna kolumnowa), a także ogólne oznaczenie elementu a_{ij}. Zmienne „w" oznaczają liczbę wierszy macierzy Ab a „k" liczbę kolumn macierzy Ab. Gdzie „i" należy do przedziału <1; w> o kroku 1 a „j" do przedziału <(i+1); w> również o kroku 1.

Wystąpienie dwukropka jako jednej ze współrzędnych będziemy traktować jako cały wiersz lub kolumnę np. $a_{i:}$ oznacza i-ty wiersz macierzy.

Pierwszym krokiem w podanej metodzie jest znalezienie największego co do modułu elementu w kolumnie, którą chcemy wyzerować (zaczynamy od pierwszej kolumny). Po jego znalezieniu zamieniamy wiersze miejscami (ten, w którym znajduje się największa znaleziona co do modułu wartość z tym wierszem, gdzie występuje element diagonalny w danej kolumnie).

Kolejnym krokiem będzie wyzerowanie (pierwszej a następnie każdej kolejnej) kolumny:

W każdej kolumnie zerujemy tylko te elementy kolumny, które znajdują się pod elementem diagonalnym w kolumnie (Dla pierwszej kolumny będą to elementy a21 i a 31, a elementem diagonalnym będzie a11). Aby wyzerować dany element stosujemy wzór:

$$a_{j:} = a_{j:} - \left(\frac{a_{ji}}{a_{ii}}\right) \cdot a_{i:}$$

Na przykład dla i = 1, j = i + 1, czyli j=2 i wzór będzie wyglądał tak:

$$a_{2:} = a_{2:} - \left(\frac{a_{21}}{a_{11}}\right) \cdot a_{1:}$$

Warto dodać, że zmienną „i" zwiększamy dopiero po wyzerowaniu kolumny, czyli dopiero po przejściu przez wszystkie wartości zmiennej „j". Gdy zmienna „i" przyjęła nową wartość to zmienna „j" przyjmuje wartość początkową z przedziału i przechodzi przez wszystkie wartości znajdujące się w przedziale. Tak postępujemy w każdej następnej kolumnie w celu jej wyzerowania.

Wykonujemy te dwa kroki (znalezienie elementu i przekształcenia wierszy) aż do napotkania ostatniego elementu diagonalnego w macierzy. Jako rezultat otrzymamy macierz trójkątną górną. W naszym wypadku macierz będzie wyglądać tak:

$$\begin{matrix} a'_{11} & a'_{12} & a'_{13} & b'_{11} \\ 0 & a'_{22} & a'_{23} & b'_{21} \\ 0 & 0 & a'_{33} & b'_{31} \end{matrix}$$

Następnie stosując postępowanie odwrotne (tzw. „back – substitution") wyznaczamy zmienną reprezentowaną przez przedostatnią kolumnę macierzy Ab dzieląc ostatnią współrzędną ostatniej kolumny przez ostatni element diagonalny macierzy. Na tym etapie traktujemy jako osobne elementy kolumnę ostatnią i resztę macierzy Ab.

$$x_{31} = \frac{b'_{31}}{a'_{33}}$$

Kolejnym krokiem jest pomnożenie każdej wartości wyżej w tej kolumnie i odjęcie tej wartości od elementu wektora b znajdującego się w tym samym wierszu. Postępujemy tak samo w każdej kolejnej kolumnie poruszając się od strony prawej do lewej, aż w końcu wyznaczymy wszystkie zmienne.

Po przekształceniach otrzymamy macierz jednostkową oraz wektor rozwiązań równania:

$$\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix} \quad \begin{matrix} x_{11} \\ x_{21} \\ x_{31} \end{matrix}$$

Metoda eliminacji Gaussa z częściowym wyborem elementu głównego napisana w programie MatLab:

```
% Definicja procedury, jej parametrów wejściowych i wejściowych:
function [x, eps1] = pgauss(A, b)
% Utworzenie macierzy rozszerzonej oraz zmiennych:
%   -w - oznaczającą liczbę wierszy macierzy rozszerzonej,
%   -k - oznaczającą liczbę kolumn macierzy rozszerzonej:
Ab = [A, b];
n = size(Ab);
w = n(1);
k = n(2);
%   Sprowadzanie macierzy rozszerzonej do postaci schodkowej górnej:
for i = 1:w
    % Wybór elementu głównego i zamiana wierszy:
    a = max(abs(Ab(i:k-1, i)));
    t = find(Ab(:, i)==a);
    Ab([t i],:) = Ab([i t],:);
    % Przekształcenie danego wiersza macierzy rozszerzonej:
    for j = (i+1):w
        Ab(j, :) = Ab(j, :) - (Ab(j,i)/Ab(i,i)) * Ab(i,:);
    end
end
% Wyodrębnienie z macierzy rozszerzonej:
%   Aw - macierzy otrzymanej w wyniku przekształceń,
%   bw - wektora otrzymanego w wyniku przekształceń:
bw = Ab(:, k);
Aw = Ab(:, (1:1:k-1));
```

```
% Utworzenie zmniennej reprezentującej wektor rozwiązań x:
x = zeros(w,1);
% Wyznaczenie wektora rozwiązań stosując "back substitution":
for i = k-1:-1:1
    for j = w:-1:1
        % Wyznaczanie zmiennej składowej wektora rozwiązań:
        if j == i
            x(j,1) = bw(j,1) / Aw(j,j);
        end
        % Pomnożenie pozostałych elemntów w kolumnie gdzie leży dana
        % zmienna oraz odjęcie wyniku tego działania od wektora w bw, aby
        % móc łątwiej wyznaczyć pozostałe zmienne:
        if Aw(j,i) ~= 0
            bw(j,1) = bw(j,1) - x(i,1) * Aw(j,i);
        end
    end
end
% Wyliczenie błędu wektora rozwiązania:
eps1 = norm(A*x - b);
disp(x);
disp(eps1);
```

Wykres błędu ε_1 dla macierzy z przykładu A i B:

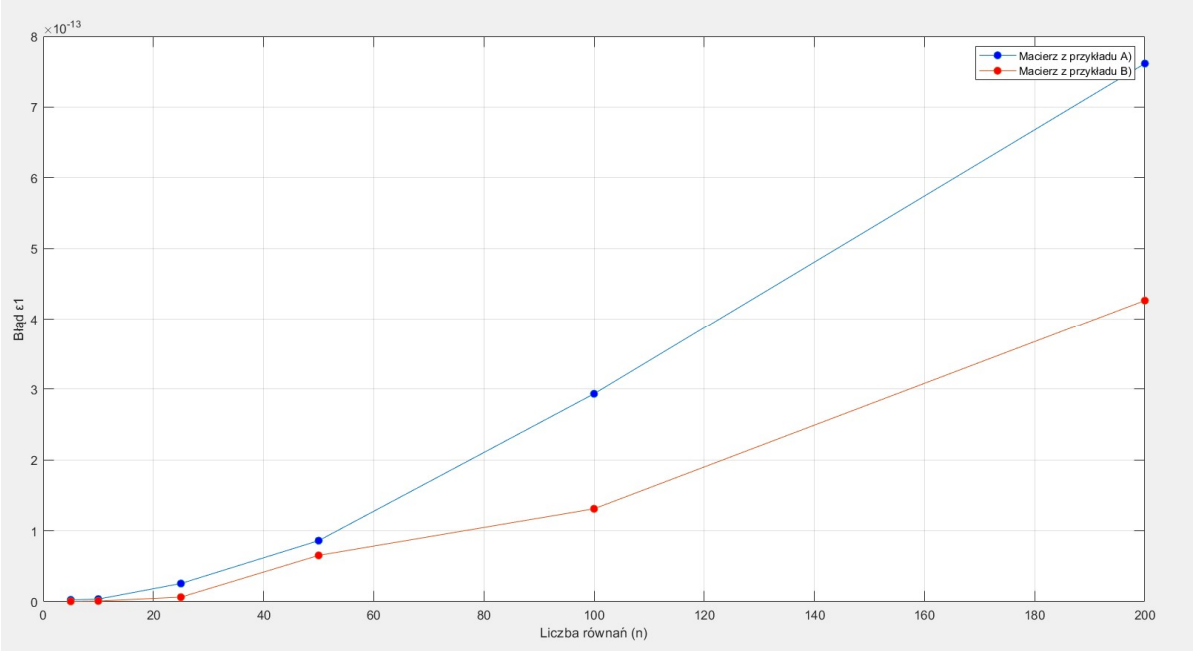

Tabela błędu ε_1 i czasu obliczeń dla macierzy z przykładu A i B:

Metoda eliminacji Gaussa z częściowym wyborem elementu głównego:				
Liczba równań (n):	Czas obliczeń dla macierzy A) [s]:	Czas obliczeń dla macierzy B) [s]:	Błąd dla macierzy A):	Błąd dla macierzy B):
5	0.0002	0.0001	2.5511e-15	4.96507e-16
10	0.0002	0.0001	3.66205e-15	1.02056e-15
25	0.0003	0.0002	2.56843e-14	6.43552e-15
50	0.0007	0.0005	8.6127e-14	6.53522e-14
100	0.0020	0.0016	2.93401e-13	1.31064e-13
200	0.0114	0.0097	7.61496e-13	4.26231e-13

Wnioski z Zad 1:

Dzięki danym prezentowanym na wykresie i w tabeli możemy zauważyć, że:

- wraz ze wzrostem liczby równań do rozwiązania zwiększa się czas obliczeń dokonywanych przez komputer dla obu macierzy,

- wraz ze wzrostem liczby równań do rozwiązania zwiększa się błąd wyniku dla obu macierzy,

- obliczenia dla macierzy B) wykonują się szybciej niż obliczenia dla macierzy A), można również zauważyć, że im większa liczba równań n tym większa różnica pomiędzy czasami obliczeń między macierzą A i B,

- błędy dla macierzy B są mniejsze niż dla macierzy A, możemy zatem wywnioskować, że macierz B jest lepiej uwarunkowana niż macierz A.

2. Napisać solwer, czyli uniwersalną procedurę, rozwiązującą układ n równań liniowych Ax = b, wykorzystując metodę Jacobiego.

Metoda Jacobiego jest metodą iteracyjną. Nie polega ona na dokładnym wyznaczeniu rozwiązań równania Ax=b, lecz na wyznaczaniu coraz dokładniejszych ich przybliżeń, obliczając kolejne elementy wektora rozwiązań. Jednak to rozwiązanie działa tylko w wypadku, gdy macierz jest dominująca diagonalnie (to znaczy, że moduł sumy elementów w wierszu niebędących diagonalami macierzy jest mniejszy od elementu diagonalnemu lub tzw. dominacja kolumnowa, czyli gdy moduł sumy elementów w kolumnie niebędących diagonalami macierzy jest mniejszy od elementu diagonalnego). Załóżmy tak jak w poprzednim zadaniu, że mamy macierz A, wektor b, zerowy wektor rozwiązań x (jako początkowa wartość). Stosowane oznaczenia będą takie same jak dla zadania poprzedniego.

Macierz A:

$$\begin{matrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{matrix}$$

Wektor b:

$$\begin{matrix} b_{11} \\ b_{21} \\ b_{31} \end{matrix}$$

Początkowy wektor rozwiązań:

$$\begin{matrix} x_{11} = 0 \\ x_{21} = 0 \\ x_{31} = 0 \end{matrix}$$

Naszym kolejnym krokiem będzie obliczenie jednego z elementów wektora rozwiązań równania na podstawie wzoru ogólnego:

$$x_{i1} = (b_{i1} - \sum_{j=<1:k>\setminus\{j=i\}}^{k} (x_{j1} \cdot A_{ij})) \div A_{ii}$$

Gdzie i należy do przedziału <1: w> o kroku 1, a j należy do przedziału <1: w> o kroku 1 z wyjątkiem elementu, gdy j = i. Zmienną i zwiększamy dopiero wtedy, gdy zostaną wyznaczone wszystkie przybliżenia wektora rozwiązań.

Następnie wyznaczamy błąd (ε_2) pomiędzy kolejnymi przybliżeniami wektora x jako normę euklidesową z ich różnicy, a później dalej wykonujemy obliczenia. Obliczenia wykonujemy dotąd aż wartość ε_2 będzie mniejsza niż podana przez nas założona tolerancja błędu.

Metoda Jacobiego napisana w programie MATLAB:

```
% Definicja procedury, jej parametrów wejściowych i wejściowych:
function [x, eps1] = jacobi(A, b, eps2w)
% Utworzenie zmiennych:
%   -w - oznaczającą liczbę wierszy macierzy rozszerzonej,
%   -k - oznaczającą liczbę kolumn macierzy rozszerzonej,
%   -x - oznaczający wektor rozwiązań w danej iteracji:
n = size(A);
w = n(1);
k = n(2);
x = zeros(w,1);
% Rozpoczęcie iteracji:
while true
    % Utworzenie wektora do którego będziemy zapisywać rozwiązanie powstałe
    % w danej iteracji, ale tylko tymczasowo w celu uniknięcia nadpisania
    % składowych wektora x:
    y = zeros(w,1);
    % Iteracja, w której liczymy każdą składową wektora rozwiązań:
    for i=1:w
        s = 0;
        for j=1:k
            if j ~= i
                s = s + A(i,j) * x(j,1);
            end
        end
        y(i,1) = (b(i,1) - s)/A(i,i);
```

```
    end
    % Obliczenie ε2:
    eps2 = norm(y - x);
    x = y;
    % Obliczenie ε1:
    eps1 = norm(A*x - b);
    % Warunek sprawdzający czy ε2 jest mniejsze od eps2w ("w" oznacza "war
    % tość graniczną błędu ε2). Gdy eps2 będzie mniejsze od wartości
    % podanej do zmiennej eps2w, pętla zakończy swoje działanie.
    if (eps2 <= eps2w)
        break;
    end
end
disp(x);
disp(eps1);
```

Wykresy błędu ε_1 dla macierzy z przykładu A:

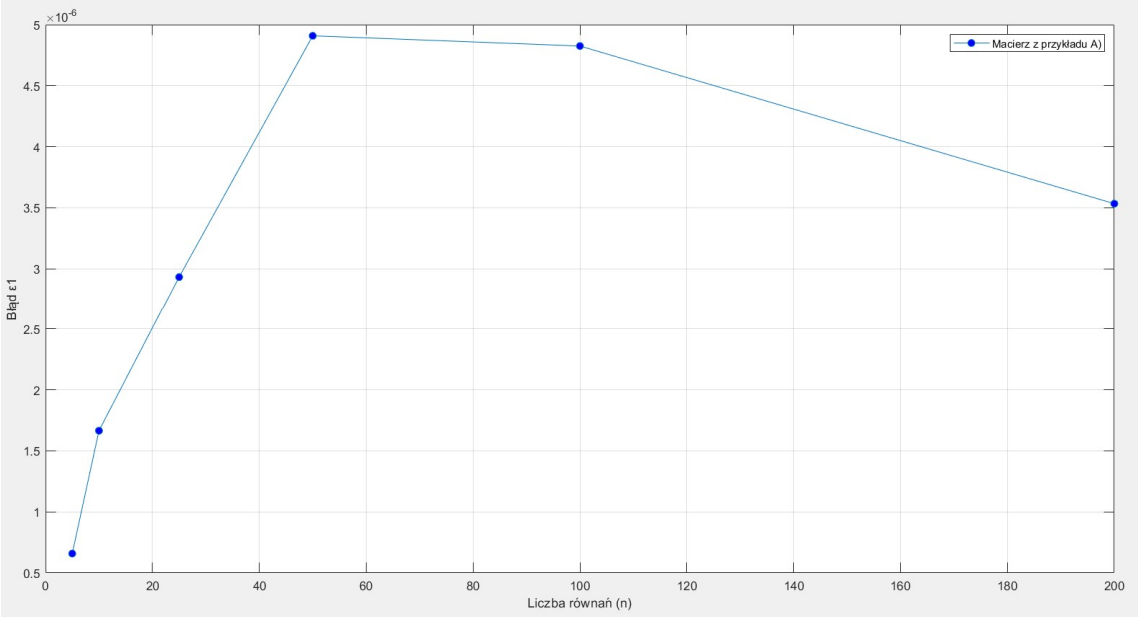

Tabela błędu ε_1 i czasu obliczeń dla macierzy z przykładu A:

Metoda iteracyjna Jacobiego:		
Liczba równań (n):	*Czas obliczeń dla macierzy A) [s]:*	*Błąd dla macierzy A):*
5	2.2970e-04	6.5868e-07
10	1.1740e-04	1.6649e-06
25	2.0720e-04	2.9315e-06
50	5.4930e-04	4.9086e-06
100	0.0011	4.8248e-06
200	0.0054	3.5330e-06

Wnioski z Zad 2:
- metoda Jacobiego była możliwa do wykonania tylko dla macierzy A, gdyż macierz B nie spełniała żadnego z warunków dominacji,
- im więcej równań tym czas obliczeń jest większy,
- błąd zwiększa się tylko do pewnej liczby równań (w tym wypadku n = 50), potem zaczyna się zmniejszać,

Wnioski ogólne:

Porównując czasy wykonywania obu metod dla macierzy A, możemy zauważyć, że przy małej liczbie równań są one pod względem szybkości tak samo efektywne, jednak wraz ze wzrostem ilości równań metoda Jacobiego okazuje się działać szybciej niż metoda eliminacji Gaussa z częściowym wyborem elementu głównego. Metoda Jacobiego wydaje się również łatwiejsza do implementacji i bardziej intuicyjna. To w czym metoda eliminacji Gaussa przeważa jest dokładność uzyskanego rozwiązania oraz jego poprawność. Różnica pomiędzy błędami w metodzie Gaussa a w metodzie Jacobiego jest bardzo duża (można przyjąć, że błąd w metodzie Jacobiego jest równy błędowi w metodzie Gaussa podniesionego do kwadratu, a nawet potęgi zawierającej się między <2;3>). Jednak, gdy liczba równań będzie bardzo duża to błędy te zaczną być coraz bliżej siebie, aż się zrównają. Od tego jaką metodę wybierzemy zależy przede wszystkim od danych, ich uwarunkowania oraz tego czy bardziej zależy nam na szybkości rozwiązania czy jego dokładności.