MNUM – Projekt 1

Zadanie 1

Wyznaczanie dokładności maszynowej komputera

Zgodnie z definicją dokładność maszynowa jest to najmniejsza wartość, która w arytmetyce zmiennoprzecinkowej po dodaniu do jedynki daje wartość od niej większą.

$$eps = \min\{g \in M : fl(1+g) > 1, g > 0\}$$

Algorytm:
Zakładamy, że pierwszą dokładnością jest liczba $eps = 1$. Po podzieleniu jej przez 2 sprawdzimy czy suma $1 + eps > 1$. Dzielenie powtarzamy, dopóki nie znajdziemy liczby niespełniającej tego założenia. Wtedy poprzednia liczba będzie dokładnością maszynową.

Kod programu:

```
function [eps] = machinePrecision() %Obliczanie dokładności maszynowej komputera

    x = 1.0;
    t = 1.0 + x;
    while (t > 1.0)
        eps = x;    %Zapamiętujemy ostatni x który po zsumowaniu dawał więcej niż 1
        x = x/2;        %Sprawdzamy czy x/2 będzie spełniało warunek pętli
        t = 1.0 + x;       %Jeśli nie oznacza to że poprzedni x (zapamiętany w eps)
    end                    %jest dokładnością maszynową

end
```

Wynik działania:

```
>> machinePrecision()

ans =

   2.2204e-16
```

Wynik ten jest zgodny ze standardem IEEE 754 dla liczb zmiennoprzecinkowych podwójnej precyzji.

Zadanie 2

Rozwiązywanie układu n równań liniowych metodą Eliminacji Gaussa z częściowym wyborem elementu podstawowego

Algorytm eliminacji Gaussa dzieli się na dwa etapy:

1. Eliminacja zmiennych – przekształcanie macierzy A i wektora b tak, aby otrzymać macierz trójkątną górną.
2. Postępowanie odwrotne (back-substitution) – stosujemy algorytm rozwiązania układu z macierzą trójkątną.

Wybór elementu podstawowego polega na tym, że przed każdym krokiem eliminowania zmiennych wybieramy z danej kolumny największy co do modułu element i zamieniamy wiersz w którym się on znajduje z obecnie rozpatrywanym.

Kod programu (plik: `GaussElimination.m`):

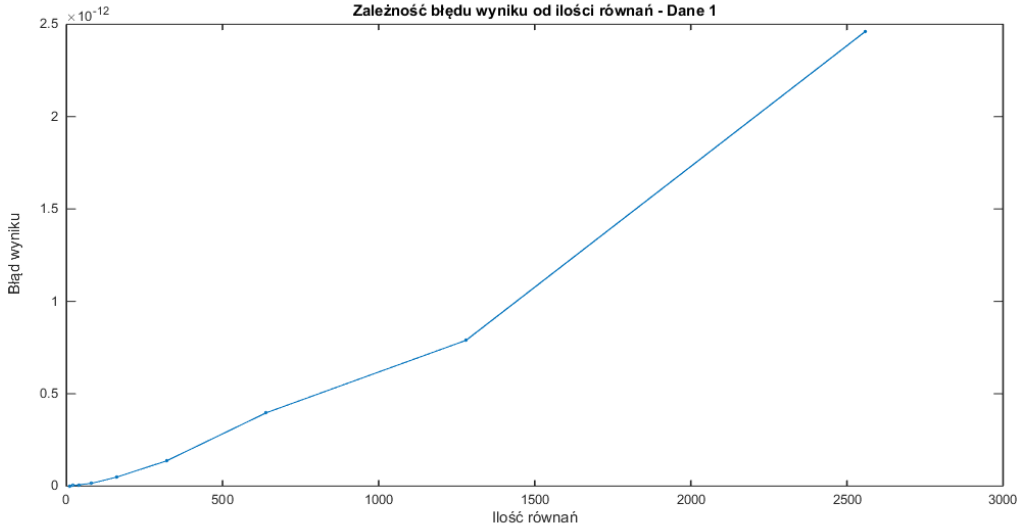
```
function [r,t] = GaussEliminate(A, b)

    tic %rozpoczęcie pomiaru czsu
    P = [A b]; %stworzenie macierzy rozszerzonej A|b
    x=size(A,1); %x oznacza ilość równań

    %w będzie naszym wektorm wynikowym
    w = zeros(x,1);

    if(det(A) == 0)
        fprintf(1,'Macierz jest osobliwa');
    end

    %Przekształcanie macierzy z cześciowym wyborem elementu podstawowego
    for j = 1:x                         %j jest indeksem wiersza
        main = 0;                       %main - główny element macierzy
        imain = j;                      %imain - indeks wiersza z main
        for i = j:x                     %poszukiwanie elementu głównego
            if abs(P(i,j))> main        %
                main = abs(P(i,j));
                imain = i;
            end
        end

        %Zamieniamy wiersz z elementem głównym z obecnym wierszem
        tmp = P(j,:);
        P(j,:) = P(imain,:);
        P(imain, :) = tmp;

        %Eliminacja Gaussa - od wierszy kolejnych odejmujemy obecny wiersz
        %pomnożony przez odpowiedni współczynnik - l
        for i = j+1:x
            l = P(i,j)/P(j,j);
            P(i,:) = P(i,:) - l * P(j,:);
        end
    end

    %Od końca będziemy obliczać kolejne wartości wynikowe i wstawiać je do
    %wektora wynikowego - w, gdzie k to indeks obliczonej wartości wynikowej
    k = x;
    while(k>=1)
        % Najpierw zapisujemy wartość wolną
        w(k) = P(k, x+1);

        %Odejmujemy po kolei obliczone już wartości wynikowe, z
        %odpowiednimi współczynnikami, zaczynając od następnego wyniku,
        %idąc aż do końca
        l = k+1;
        while(l<=x)
            w(k)= w(k) - P(k,l) * w(l);
            l = l+1;
        end

        %Na  koniec dzielimy przez współczynnik przy obecnie obliczonej
        %niewiadomej
        w(k) = w(k)/ P(k,k);

        k = k-1;
    end

    %r jest residuum
    r = b - A * w;

    %Po obliczeniu normy zwracamy ją jako błąd obliczenia
    r = norm(r);
    t = toc;
end
```

Wyniki:

Liczbę równań zwiększałem, dopóki czas na ich wykonanie nie przekraczał dwóch minut, oraz pierwszy zestaw przekraczający ten czas, we wszystkich przypadkach było to 2560 równań.

Dane 1:

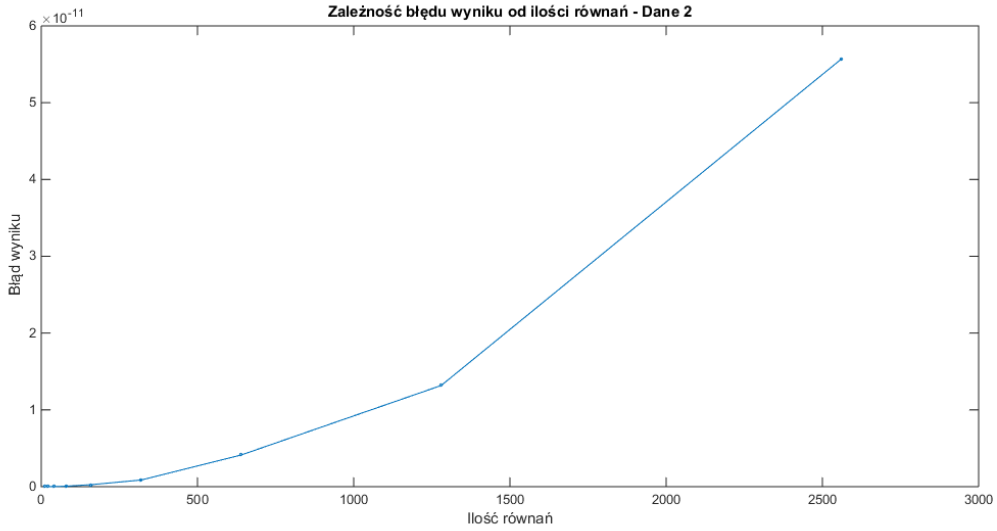
Ilość równań	Błąd wyniku
10	0.0001e-11
20	0.0003e-11
40	0.0007e-11
80	0.0015e-11
160	0.0048e-11
320	0.0136e-11
640	0.0397e-11
1280	0.0789e-11
2560	0.2463e-11



Jak widać z otrzymanych danych błąd wyniku wzrasta nieliniowo wraz ze wzrostem liczby równań, ale wartości tego błędu mimo wszystko nie są zbyt duże (na poziomie e-12). W związku z tym można przyjąć, że rozwiązanie tego zestawu tą metodą daje dokładne rezultaty.

Dane 2:

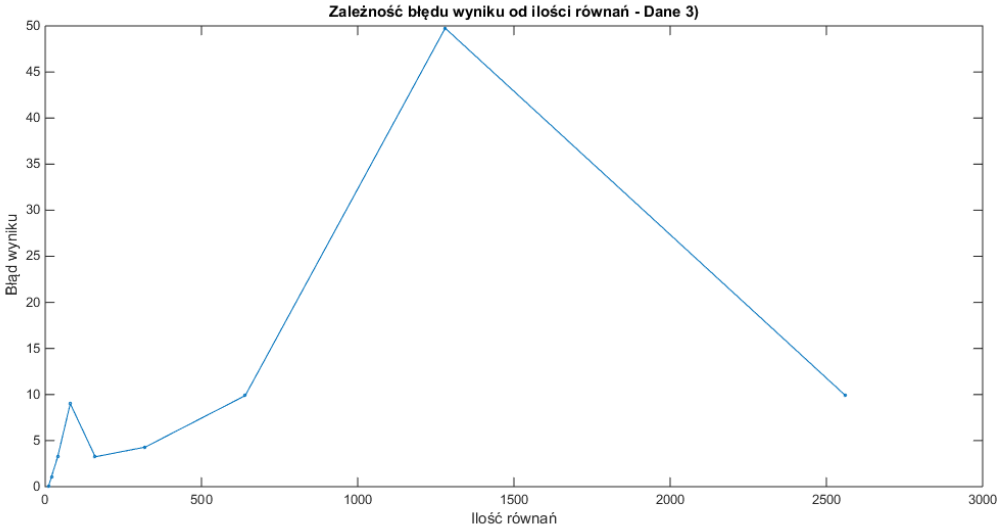
Ilość równań	Błąd wyniku
10	0.0000e-10
20	0.0001e-10
40	0.0001e-10
80	0.0006e-10
160	0.0026e-10
320	0.0087e-10
640	0.0413e-10
1280	0.1318e-10
2560	0.5567e-10



Z otrzymanych danych wynika że dla tych danych błąd wyniku także wzrasta nieliniowo wraz ze wzrostem liczby równań. Wartości tych błędów są większe niż dla poprzednich danych i widać że wzrastają szybciej niż poprzednio. Mimo tego są one cały czas niezbyt duże w związku z czym metoda elimionacji Gaussa działa też poprawnie dla tego zestawu.

Dane 3:

Ilość równań	Błąd wyniku
10	0.0007
20	1.0782
40	3.2667
80	9.0112
160	3.2430
320	4.2949
640	9.8823
1280	49.7531
2560	9.9189



Dla trzeciego zestawu błędy rozwiązania są o kilkanaście rzędów wielkości większe niż w porzednich przypadkach i ponadto nie widać tu wyraźnej zależności wartości błędu rozwiązania od liczby równań, co się odbija na tym że błąd wyniku jest całkowicie nieprzewidywalny. Dla tych danych metoda eliminacji Gaussa nie będzie dawała poprawnych rezultatów.

Zadanie 3

Rozwiązywanie układu n równań liniowych metodą Jacobiego

Rozwiązywanie układów równań metodą Jacobiego polega na zdekomponowaniu macierzy A na macierze L, D, U, gdzie $A = L + D + U$, oraz: L – macierz trójkątna dolna, D – macierz diagonalna, U – macierz trójkątna górna. Dzięki temu układ równań $Ax = b$ można zapisać w postaci:

$$Dx = -(L + U)x + b$$

Dopóki macierz A jest nieosobliwa można zaproponować metodę iteracyjną:

$$x^{(i+1)} = -D^{-1}(L + U)x^{(i)} + D^{-1}b, \quad i = 0,1,2, \ldots,$$

Równoważnie:

$$x_j^{(i+1)} = -\frac{1}{d_{jj}} \sum_{k=1}^{n} (l_{jk} + u_{jk}) x_j^{(i)} + b_j \quad i = 0,1,2, \ldots,$$

Dla tej metody istotne jest ustalenie warunku stopu, np. liczba iteracji, lub zadowalająca dokładność.

Kod program (plik `Jacobi.m`):

```
function [x] = Jacobi(A, b, iter)
    s =size(A,1);
    L = zeros(s,s);
    D = zeros(s,s);
    U = zeros(s,s);

    x = zeros(s,1);

    %Sprawdzenie warunku dostatecznego zbieżności - silnej dominacji
    %diagonalnej macierzy A
    for i = 1:s
        %Sumujemy wszystkie elementy w wierszu, poza diagonalnym
        sum = 0;
        for j = 1:s
            if i~=j
                sum = sum + abs(A(i,j));
            end
        end
        %Jeśli suma jest większa od elementu to warunek nie jest spełniony
        if sum > abs(A(i,i))
            disp('Uwaga: Warunek dominacji diagonalnej nie jest spełniony')
            break
        end
    end

    %Stworzenie macierzy trójkątnych oraz diagonalnej
    for i = 1:s
        for j = 1:s
            if(i<j)
                U(i,j) = A(i,j);
            elseif(i>j)
                L(i,j) = A(i,j);
            else
                D(i,j) = A(i,j);
            end
        end
    end
```

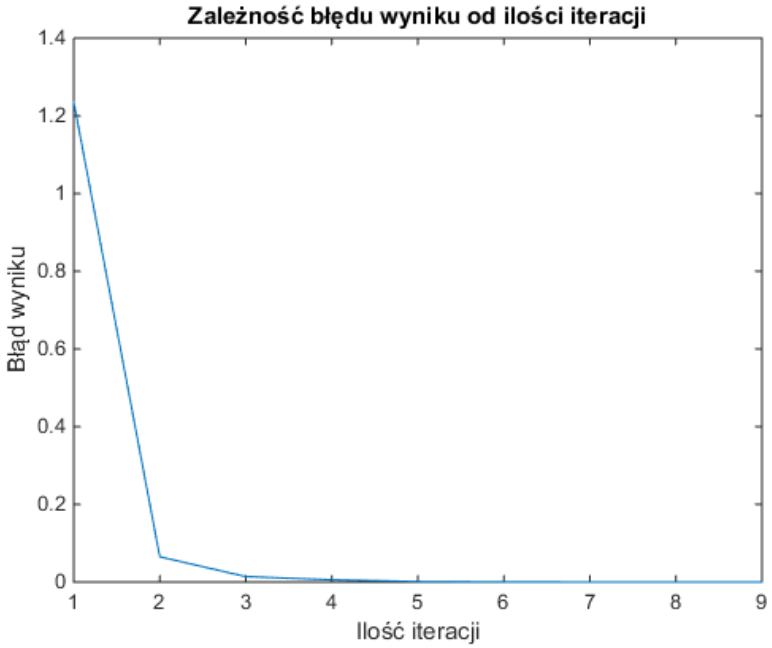
```
    %Składniki wykorzystane przy obliczniu kolejnych iteracji
    P = inv(D)*(L+U);
    O = inv(D)*b;

    for k = 2:iter
        x(:,k) = (-1)*P * x(:,k-1) + O;
    end
end
```

Wyniki:

Dane z zadania 3:

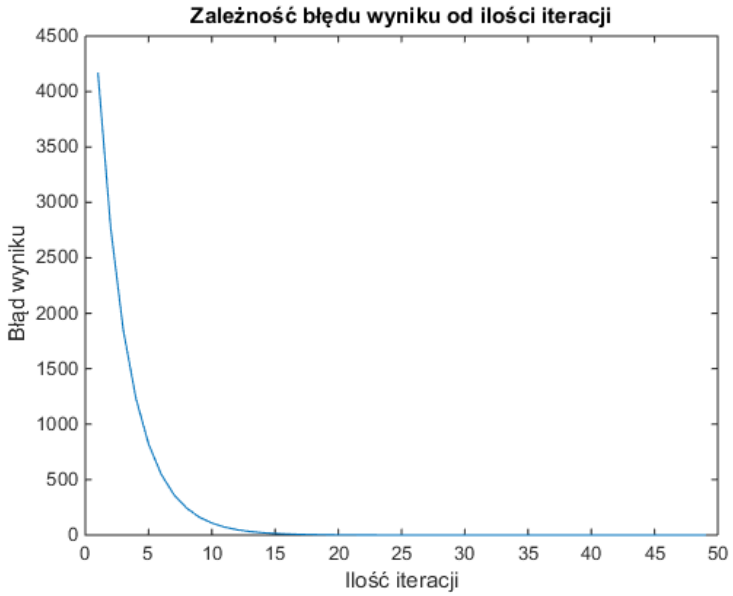
$x =$

0.4225
0.2217
0.4609
0.9758



Jak widać na wykresie dla małej liczby iteracji błąd rozwiązania jest duży, ale bardzo szybko maleje wraz ze wzrostem liczby iteracji. Dla ilości większej od 5 błąd osiąga wartość bardzo bliską zeru. Można więc przyjąć że Metoda Jacobiego daje w tym przypadku dokładne rezultaty.

Dane z zadania 2.1 (ilość równań = 2560):



Widać, że dla małych ilości iteracji wynik jest obarczony dużym błędem, jednakże bardzo szybko zbiega on do zera i dla więcej niż dwudziestu iteracji osiąga praktycznie 0. Oznacza to, że dla tych danych otrzymujemy dobre rezultaty (po odpowiedniej liczbie iteracji), a cały algorytm działa znacznie szybciej niż metoda eliminacji Gaussa.

Dane z zadania 2.2 oraz 2.3:

Dla pozostałych danych z zadania 2 nie udało się uzyskać poprawnych rezultatów metodą Jacobiego. W tych przypadkach ciągi $x^{(i)}$ były rozbieżne, przez co każda kolejna iteracja dawała wynik coraz bardziej odbiegający od rzeczywistego. Stosowanie tej metody do tych zestawów danych daje zatem błędne rezultaty.

Wniosek:

Dla określonych zestawów danych metoda Jacobiego, przy odpowiedniej ilości iteracji, daje poprawne rezultaty, a cały algorytm działa znacznie szybciej niż metoda eliminacji Gaussa. Jednakże metoda eliminacji Gaussa jest bardziej niezawodna, będzie obsługiwała większą liczbę zestawów niż algorytm Jacobiego. W związku z tym metoda Jacobiego powinna być używana dla zestawów spełniających określone warunki (warunek dostateczny: silna dominacja diagonalna macierzy), a dla pozostałych metoda eliminacji Gaussa będzie dawała pewniejsze rezultaty (ale także obarczone błędami).