**Krzysztof Bednarski – AiR**

**MNUM – Projekt 1**

**Zadanie 1.19**

Zadanie 1

Wyznaczanie dokładności maszynowej komputera.

Z definicji dokładność maszynowa jest to najmniejsza dodatnia liczba, która w arytmetyce zmiennoprzecinkowej jeśli zostanie dodana do jedynki, da wartość większą od niej, tzn.

Zasada działania algorytmu:

Jako początkową wartość eps przyjmujemy 1. Podczas przebiegu każdej pętli, zapamiętujemy poprzednią wartość eps, a następnie dzielimy ją przez 2. Jeśli po podzieleniu eps przez 2 i dodaniu do 1, wartość wyrażenia dalej będzie 1, pętla zostaje zakończona. Wartość eps, czyli dokładności maszynowej jest to wartość z przedostatniej pętli.

Kod źródłowy programu (plik: machinePrecision.m) :

function [eps] = machineprecision() %Obliczanie dokladnosci maszynowej komputera

x = 1.0;

rd = 1.0 + x;

while (rd > 1.0)

eps = x; %Ostatni x spelniajacy warunek petli, zapamietujemy w zmiennej eps

x = x/2;

rd = 1.0 + x; %Jesli rd bedzie równe 1, oznacza to, ze ostatni zapamiętany x to dokladnosc maszynowa

end

end

Wyniki:

>> [eps] = machinePrecision()

eps =

2.2204e-16

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

Zadanie 2

Rozwiązywanie układu n równań liniowych A×x = b metodą LU rozkładu macierzy.

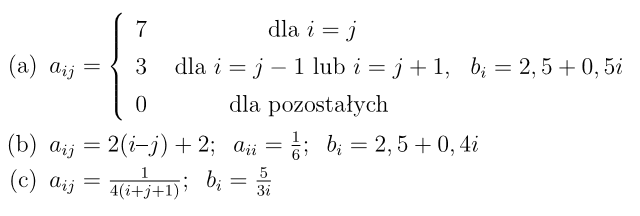
Aby otrzymać rozkład LU, w którym macierz L jest macierzą dolną trójkątną, natomiast U jest macierzą górną trójkątną, postępujemy tak jak w metodzie eliminacji Gaussa. Macierz U jest to po prostu macierz A, po przeprowadzeniu eliminacji Gaussa. Natomiast macierz L powstaje w taki sposób, że na jej diagonali znajdują się 1, natomiast pod diagonalą w każdą komórkę macierzy wpisywany jest współczynnik, który był potrzebny do wyzerowania danej komórki w eliminacji Gaussa.

Gdy mamy już rozkład LU, możemy przejść do rozwiązywania równań:

1. L×y = b, a następnie
2. U×x = y

Po rozwiązaniu obu równań, otrzymujemy wektor rozwiązań x.

Dane:



Kod źródłowy programu (plik: metodaLU.m):

function[x] = metodaLU(A, b, n)

tic %rozpoczecie pomiaru czasu

A\_P = zeros(n,n); %macierz pomocnicza

x = zeros(n,1); %wektor niewiadomych

y = zeros(n,1); %wektor potrzebny do rozwiazywania rownan

L = zeros(n,n);

U = zeros(n,n);

%Macierz w ktorej zapisujemy poczatkowe wartosci macierzy A

A\_P = A;

%Przeprowadzamy eliminajce Gaussa macierzy A, ale w komorce, ktora

%zerujemy, zapamietujemy zmienna l

for j = 1:n

for i = j+1:n

l = A\_P(i,j)/A\_P(j,j);

A\_P(i,:) = A\_P(i,:) - l \* A\_P(j,:);

L(i,j) = l;

end

end

%Przeksztalcona macierz A rozkladamy na mierze L i U

for j = 1:n

for i = 1:n

if i==j

L(i,j) = 1;

U(i,j) = A\_P(i,j);

elseif i>j

U(i,j) = 0;

elseif i<j

U(i,j) = A\_P(i,j);

end

end

end

%Rozwiazujemy dwa równania

%Najpierw rownanie L\*y=b

k = 1;

while k<=n

y(k) = b(k); %pierwsza komorke mozemy przepisac

l=k-1;

while l>=1

y(k) = y(k) - (L(k,l)\*y(l));

l = l-1;

end

k = k+1;

end

%Teraz rownanie U\*x=y

k = n;

while k>=1

x(k) = y(k); %pierwsza komorke mozemy przepisac

l=k+1;

while l<=n

x(k) = x(k) - (U(k,l)\*x(l));

l = l+1;

end

x(k) = x(k)/U(k,k);

k = k-1;

end

%r jest residuum

r = b - A\*x;

%Po obliczeniu normy zwracamy ja jako blad obliczenia

r = norm(r);

toc

end

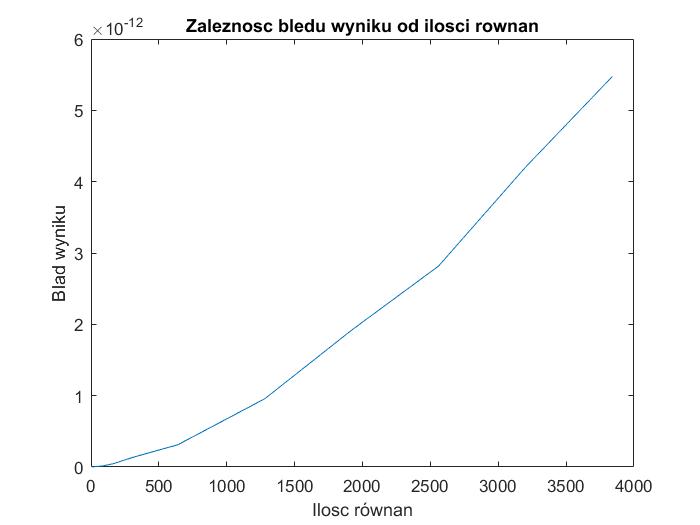
Wyniki:

Ilość równań została zwiększana, aż do momentu w którym czas potrzebny na obliczenie wektora wynikowego, przekroczył 5 minut. Stało się to po osiągnięciu 3840 równań. Ilość równań była zwiększana w następujący sposób. Do 1280 rosła wykładniczo, natomiast później była zwiększana o 640.

Podpunkt a):

Dane dla podpunktu a):

|  |  |  |
| --- | --- | --- |
| Ilość równań | Czas potrzebny do obliczenia | Błąd przybliżenia |
| 10 | 0.0050 | 1.0e-11 \*0.0001 |
| 20 | 0.00016840 | 1.0e-11 \*0.0002 |
| 40 | 0.00054938 | 1.0e-11 \*0.0003 |
| 80 | 0.0017 | 1.0e-11 \*0.0008 |
| 160 | 0.0087 | 1.0e-11 \*0.0013 |
| 320 | 0.0471 | 1.0e-11 \*0.0044 |
| 640 | 0.6033 | 1.0e-11 \*0.0142 |
| 1280 | 9.6370 | 1.0e-11 \*0.0312 |
| 1920 | 34.8817 | 1.0e-11 \*0.1919 |
| 2560 | 154.9106 | 1.0e-11 \*0.2817 |
| 3200 | 171.4815 | 1.0e-11 \*0.4201 |
| 3840 | 400.8918 | 1.0e-11 \*0.5473 |

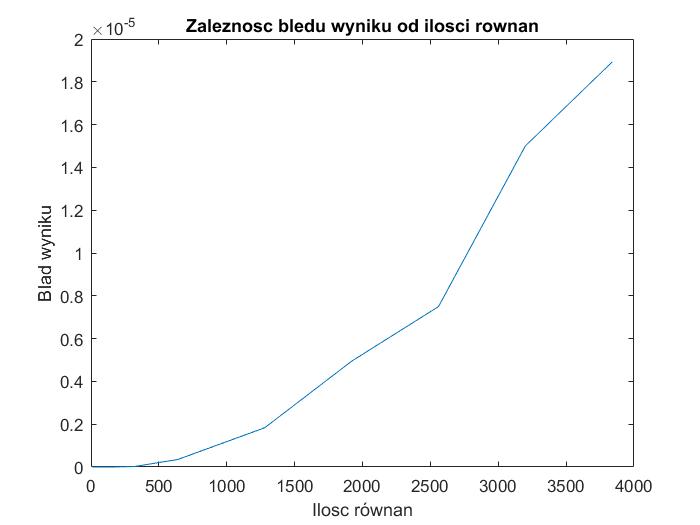


Z wykresu widać, że błąd wyniku rośnie nieliniowo względem ilości równań. Natomiast błędy te są rzędu e^(-12). Możemy z tego wywnioskować, że rozwiązanie podpunktu a) tą metodą daje dokładne rezultaty.

Podpunkt b):

Dane dla podpunktu b):

|  |  |  |
| --- | --- | --- |
| Ilość równań | Czas potrzebny do obliczenia | Błąd przybliżenia |
| 10 | 0.00060753 | 1.0e-10\*0.0123 |
| 20 | 0.00018464 | 1.0e-10\*0.1172 |
| 40 | 0.00030546 | 1.0e-10\*0.5060 |
| 80 | 0.0011 | 1.0e-10\*0.3958 |
| 160 | 0.0072 | 1.0e-06\* 0.0015 |
| 320 | 0.0437 | 1.0e-06\*0.0229 |
| 640 | 0.6650 | 1.0e-06\*0.3523 |
| 1280 | 9.7780 | 1.0e-04\*0.0184 |
| 1920 | 36.2052 | 1.0e-04\*0.0494 |
| 2560 | 157.4237 | 1.0e-04\*0.0750 |
| 3200 | 172.4699 | 1.0e-04\*0.1501 |
| 3840 | 412.9059 | 1.0e-04\*0.1893 |

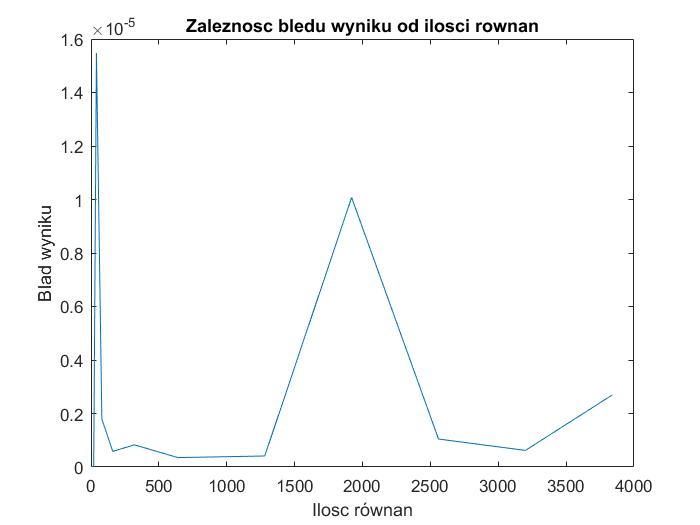


Z wykresu widać, że błąd wyniku rośnie nieliniowo względem ilości równań. Natomiast błędy są większe niż w poprzednim przypadku bo są rzędu e^(-5), również wzrost ich jest szybszy niż w podpunkcie a). Mimo to, możemy stwierdzić, że ta metoda również jest dokładna dla podpunktu b).

Podpunkt c):

Dane dla podpunktu c):

|  |  |  |
| --- | --- | --- |
| Ilość równań | Czas potrzebny do obliczenia | Błąd przybliżenia |
| 10 | 0.000058147 | 1.0e-05\*0.0001 |
| 20 | 0.000070608 | 1.0e-04\*0.0005 |
| 40 | 0.00023146 | 1.0e-04\*0.1548 |
| 80 | 0.00092243 | 1.0e-04\*0.0179 |
| 160 | 0.0072 | 1.0e-04\*0.0058 |
| 320 | 0.0520 | 1.0e-04\*0.0083 |
| 640 | 0.6813 | 1.0e-04\*0.0035 |
| 1280 | 9.9240 | 1.0e-04\*0.0041 |
| 1920 | 34.8913 | 1.0e-04\*0.1007 |
| 2560 | 159.0515 | 1.0e-04\*0.0105 |
| 3200 | 170.8698 | 1.0e-04\*0.0062 |
| 3840 | 421.2050 | 1.0e-04\*0.0269 |



Z wykresu widać, że błąd wyniku nie rośnie wraz ze wzrostem ilości równań. Dla małej ilości równań bardzo szybko rośnie, potem maleje, ale znowu w okolicach 1500 równań zaczyna rosnąć, a przy 2500 znowu maleć. Z wykresu wnioskujemy, że i ta metoda jest w miarę dokładna, bo rząd błędu wynosi około e^(-5). Natomiast nie możemy przewidzieć, mniej więcej wielkości błędu, jak to miało miejsce w poprzednich podpunktach.

Zadanie 3

Rozwiązywanie układu n równań liniowych A×x = b metodą iteracyjną Gaussa-Seidela.

Aby móc przeprowadzić metodę iteracyjną Gaussa-Seidela, na początku musimy sprawdzić, czy macierz A spełnia warunek dostateczny zbieżności, czyli silnej dominacji diagonalnej (oznacza to, że suma wszystkich elementów w wierszu poza diagonalnym, nie może być większa od elementu diagonalnego).

Gdy warunek ten jest spełniony możemy przejść do rozkładu macierzy A na macierze:   
L – macierz poddiagonalną, U – naddiagonalną i D – diagonalną. W następujący sposób:

**A = L + D + U**

Przykład:

**A L D U**

Pojedynczą iterację możemy zapisać w postaci:

Otrzymujemy stąd znowu układ n równań skalarnych. Składowe nowego wektora wyznaczamy kolejno, poczynając od pierwszej:

gdzie

,

itd.

Dla tej metody ważne jest ustalenie warunku, który gdy zostanie spełniony, ma spowodować przerwanie iterowania i wypisanie wyniku. W tym przypadku będzie to osiągnięcie założonej dokładności.

Kod źródłowy programu (plik: Gaussa\_Seidela.m):

%e oznacza ε - czyli blad przyblizenia

function[x]=Gaussa\_Seidela(A,b,n,e)

L = zeros(n,n);

D = zeros(n,n);

U = zeros(n,n);

x = zeros(n,1);

y = zeros(n,1); %zapisuje postac wektora x, z poprzedniej iteracji

w = zeros(n,1);

%Sprawdzenie warunku dostatecznego zbieznosci - silnej dominacji

%diagonalnej macierzy A

for i = 1:n

%Sumujemy wszystkie elementy w wierszu, poza diagonalnym

sum = 0;

for j = 1:n

if i~=j

sum = sum + abs(A(i,j));

end

end

%Jesli suma jest wieksza od elementu to warunek nie jest spelniony

if sum > abs(A(i,i))

disp('Warunek silnej dominacji diagonalnej nie jest spelniony');

return

end

end

%Stworzenie macierzy trojkatnych oraz diagonalnej

for i = 1:n

for j = 1:n

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

%Zaczynamy iterowanie

%e - to blad przyblizenia

r = 1;

iter = 1;

while r>e

y = x; %zapamietujemy wektor x z poprzedniej iteracji

w = U\*x - b;

for i = 1:n

x(i) = (-L(i,:)\*x- w(i))/D(i,i);

end

r = x-y;

r = norm(r); %liczmy blad z nomrmy euklidesowej, po kazdej iteracji

iter = iter + 1; %zliczamy ilosc iteracji

end

end

Wyniki:

Do danych z zadania 3:

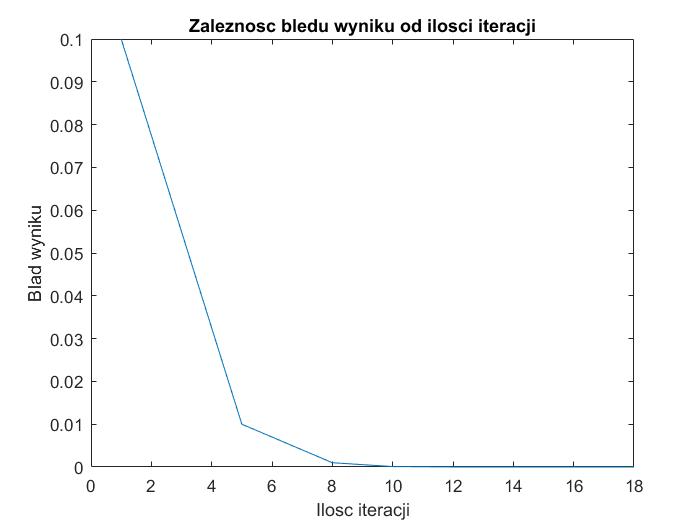
x =

-0.1635

0.1544

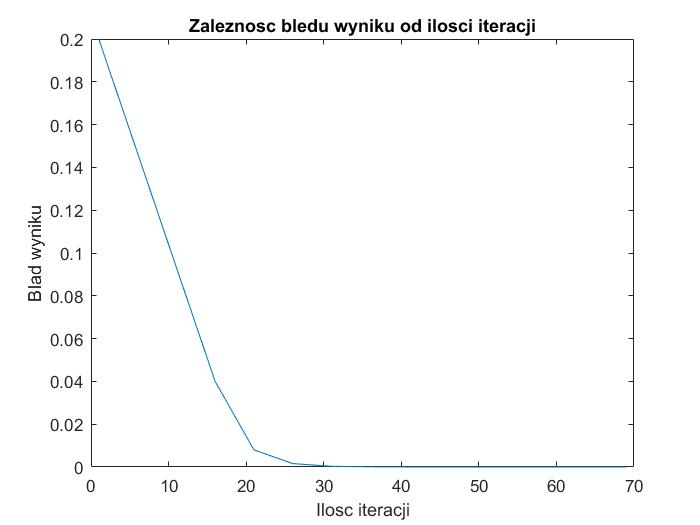
0.9236

2.7396



Jak możemy zauważyć na wykresie dla małej liczby iteracji błąd rozwiązania jest duży, ale bardzo szybko maleje wraz ze wzrostem liczby iteracji. Dla ilości iteracji większej od 10 błąd osiąga wartość bardzo bliską zeru. Można więc przyjąć, że metoda Gaussa-Seidela daje w tym przypadku dokładne rezultaty.

Dane z zadania 2 z podpunktu a) (ilość równań = 2560):



Jak widać na wykresie błąd wyniku bardzo szybko zbiega do 0. Po 30 iteracji jest on już równy prawie 0. Wnioskujemy z tego, że dla tych danych metoda Gaussa- Seidela daje dobre rezultaty. Ponadto w tym przypadku jest ona szybsza niż metoda rozkładu LU z zadania 2.

Dane z zadania 2 z podpunktu b) i c):

Dla tych danych metoda Gaussa-Seidela nie działa, ponieważ nie spełniają one warunku dostatecznego dla tej metody, czyli silnej dominacji diagonalnej. Po uruchomieniu procesu dla tych danych od razu dostajemy informacje 'Warunek silnej dominacji diagonalnej nie jest spelniony' i proces zostaje przerwany. Dzieje się tak, ponieważ w tych przypadkach ciągi były rozbieżne.

**Podsumowanie**

Porównując oba algorytmy rozwiązywania układów n równań liniowych, możemy zauważyć, że metoda Gaussa-Seidela daje lepsze rezultaty dla odpowiedniej liczby iteracji, ponadto jest ona znacznie szybsza. Natomiast nie jest ona tak uniwersalna jak metoda rozkładu LU, dzięki której możemy rozwiązać dużo większą liczbę układów równań, ale dla liczby równań przewyższającej 4000, staje się ona bardzo nieefektywna, ponieważ czas obliczenia wektora wynikowego dla takiej liczby równań przekracza 10 minut. W związku z tym metoda   
Gaussa-Seidela powinna być stosowana dla macierzy, które spełniają odpowiednie warunki (warunek dostateczny dla tej metody: silna dominacja diagonalna), natomiast dla innych metoda rozkładu LU.