Sprawozdanie MNUM Projekt 01

Autor: **TOMASZ SACHANOWSKI**

Nr. Indexu: **276467**

Nr. Zadania**: 1.55**

Spis treści

[Treść zadań 2](#_Toc35694314)

[Zadanie 1 3](#_Toc35694315)

[Cel: 3](#_Toc35694316)

[Teoria: 3](#_Toc35694317)

[Rozwiązanie: 3](#_Toc35694318)

[Wynik: 3](#_Toc35694319)

[Podsumowanie: 4](#_Toc35694320)

[Zadanie 2 4](#_Toc35694321)

[Cel: 4](#_Toc35694322)

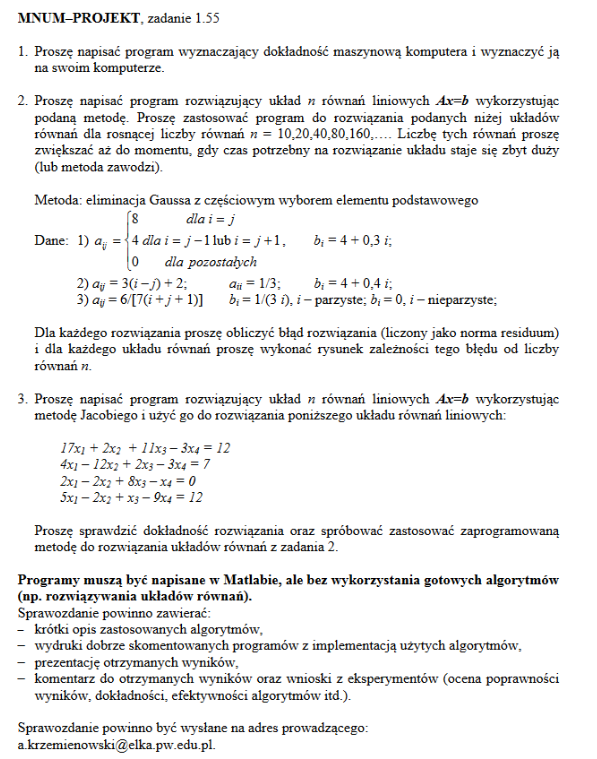
[Teoria: 4](#_Toc35694323)

[Rozwiązanie: 7](#_Toc35694324)

[Wynik: 8](#_Toc35694325)

[Podsumowanie: 13](#_Toc35694326)

# Treść zadań



# Zadanie 1

## Cel:

Celem zadania jest wyznaczenie dokładności maszynowej komputera

## Teoria:

Korzystając z definicji na dokładność maszynową.



Zgodnie z nią, dokładność maszynowa komputera to najmniejsza dodatnia liczba g taka, że**𝑓𝑙(1+𝑔)>1**, czyli najmniejsza liczba maszynowa (zmiennoprzecinkowa), która dodana do liczby 1 daje w wyniku więcej niż 1.

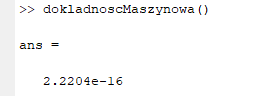
## Rozwiązanie:

Algorytm opiera się na badaniu warunku **1+g > 1**, gdzie g jest zmieniane iteracyjnie. W każdym obiegu pętli wartość g jest dzielona przez 2, aż do momentu niespełnienia warunku. Zwracana jest przedostatnia wartość obiegu, która jest naszą szukaną wartością.

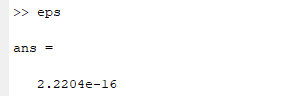
Dzielenie przez 2 nie wprowadza dodatkowych błędów z uwagi na sposób komputerowej reprezentacji liczb zmiennoprzecinkowych, gdzie dzielenie przez 2 oznacza tylko zmniejszanie o 1 wykładnika liczby.

## Wynik:

Funkcja zwraca wartość 2.2204e-16.



Jest to wartość zgodna z wbudowaną funkcja MATLAB’a **eps**



## Podsumowanie:

Wyznaczona dokładność maszynowa pokrywa się z dokładnością standardu **IEEE 754** dla liczb zmiennoprzecinkowych o podwójnej precyzji. W eksperymencie wyszła ona równa co do wartości stałej **eps** zapisanej w programie MATLAB. Ilość iteracji to 52, czyli tyle ile bitów jest przeznaczonych na mantysę.

# Zadanie 2

## Cel:

Celem zadania jest rozwiązanie układu **n** równań liniowych w postaci **Ax = b**, przy pomocy metody **ELIMINACJI GAUSSA Z CZĘŚCIOWYM WYBOREM ELEMENTU PODSTAWOWEGO** liczbę równań (**n**) należy zwiększać aż do momentu osiągnięcia zbyt dużego czasu potrzebnego na znalezienie rozwiązania.

## Teoria:

Algorytm eliminacji Gaussa dzieli się na dwa etapy:

1. Eliminacja zmiennych – w wyniku przekształceń macierzy A i wektora b otrzymamy równoważny układ równań z macierzą trójkątną górną.
2. Postępowanie odwrotne (ang. ”back-substitution”) – stosujemy algorytm rozwiązania układu z macierzą trójkątną.

Klasyczny algorytmu eliminacji Gaussa nie może znaleźć rozwiązania wszystkich układów posiadających rozwiązanie. Gdyż w trakcie realizacji metody eliminacji Gaussa wymaga się, by wszystkie współczynniki na przekątnej były różne od zera, bo musimy przez nie dzielić kolejne wiersze. Wykonując przedstawiony algorytm eliminacji Gaussa, możemy spotkać blokującą obliczenia sytuację, gdy dla **k-tego** kroku 𝑎𝑘𝑘(𝑘)=0. Unikamy tego, stosując algorytm eliminacji Gaussa z wyborem elementu głównego. Wybór ten może być częściowy lub pełny.

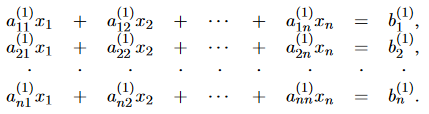
**Metoda z częściowym wyborem elementu głównego:**

Na początku k-tego kroku jako element główny wybieramy

|𝑎𝑖𝑘(𝑘)|= max𝑗 { |𝑎𝑗𝑘(𝑘)|, 𝑗=𝑘,𝑘+1,…,𝑛}, a następnie zamieniamy wiersz 𝑖-ty z 𝑘-tym, dalej stosujemy algorytm standardowy 𝑘-tego kroku. Jest to stosowne w każdym k-tym kroku, nawet jak akk(k) jest różne od zera.

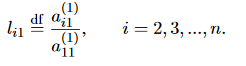
**Etap eliminacji zmiennych:**

Wyjściowy układ równań (górny indeks „(k)” – układ równań przed **k-tym** krokiem metody):

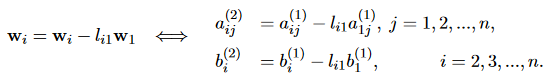


**Krok 1**

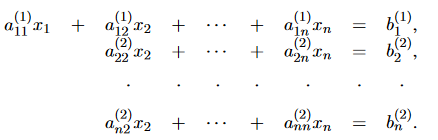
Znalezienie elementu głównego w pierwszej kolumnie (ai1). Następnie zamiana wiersza 1 z wierszem i. Potem wyzerowanie elementów kolumny pierwszej, oprócz elementu w wierszu pierwszym, dzięki wyznaczeniu współczynników.



Pierwszy wiersz w1 mnożymy przez li1 i odejmujemy od wiersza i-tego wi, kolejno dla i= 2,3,...,n:

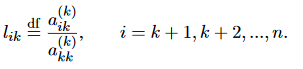


Otrzymujemy:

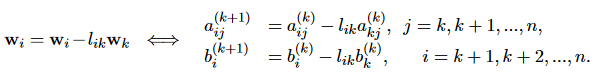


**Krok k-ty**

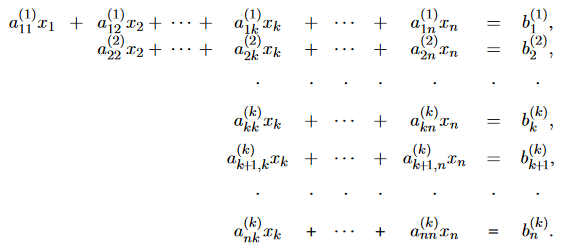
Znalezienie elementu głównego w kolumnie k-tej (aik). Następnie zamiana wiersza k-tego z wierszem i. Potem wyzerowanie elementów kolumny k-tej, w wierszach poniżej k-tego, dzięki wyznaczeniu współczynników:



Tak więc przekształcenia w k-tym kroku dane są zależnościami:



Ogólnie, po k−1 krokach otrzymujemy układ równań:



W rezultacie, po n−1 krokach uzyskujemy układ równań:

,

gdzie to macierz trójkątna górna.

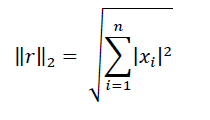
**Postępowanie odwrotne (ang. ”back-substitution”):**

1. Iteracja po wierszach od ostatniego do pierwszego
2. Dla każdego wiersza korzystając ze wszystkich wierszy poniżej zerujemy wszystkie wartości na prawo od diagonali
3. Dzielimy wartość w wektorze wyników w danym wierszu przez wartość na diagonali.

Układ równań liniowych po podstawieniu otrzymanego algorytmem numerycznym rozwiązania (oznaczmy je przez 𝑥(1)) na ogół nie jest dokładnie spełniony, tzn:



gdzie błąd niespełnienia równań 𝑟(1) nazywany jest resztą lub residuum. Błąd rozwiązania liczyłem jako normę residuum (norma euklidesowa):



## Rozwiązanie:

Program został podzielony na funkcje. Na początku korzystam z skryptów tworzących odpowiednie macierze A i wektory b. Mamy trzy funkcje które to realizują: ***CreatMatrix\_A, CreatMatrix\_B, CreatMatrix\_C***. Każda działa dla odopowiedniego typu macierzy z zadania. Następnym krokiem jest metoda eliminacji gaussa z częściowym wyborem elementu głównego. Jest to realizowane przez funkcję ***gauss.***

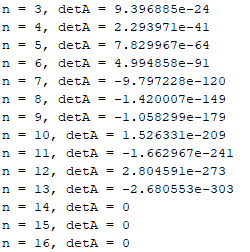
Funkcja ta na samym początku sprawdza, czy wyznacznik macierzy jest różny od zera. Zgodnie z Twierdzeniem Cramera (dla układów, w których 𝑙𝑖𝑐𝑧𝑏𝑎 𝑟ó𝑤𝑛𝑎ń=𝑙𝑖𝑐𝑧𝑏𝑎 𝑛𝑖𝑒𝑤𝑖𝑎𝑑𝑜𝑚𝑦𝑐ℎ) jeśli 𝑑𝑒𝑡𝐴 ≠0, to układ ma nieskończenie wiele rozwiązań lub jest sprzeczny. Następnie liczony jest wskaźnik uwarunkowania rozwiązania układu równań liniowych. Potem przejście do algorytmu gaussa, w którym to iterujemy po kolumnach i dla każdego kroku szukamy elementu głównego w danej kolumnie i odpowiednie zamienienie wierszy. Kolejnym etapem jest obliczenie współczynników do zerowania, dzięki którym możliwe będzie wyzerowanie elementów poniżej aktualnego elementu głównego jak przeliczenie pozostałych elementów w wierszu względem wyliczonego współczynnika. W wyniku tych operacji otrzymujemy macierz schodkowa(tójkątna górna). Ostatnim etapem funkcji jest wyliczenie wektora rozwiązań. Jest to realizowane od końca ze względu na macierz trójkątną. W wyniku tej matematycznej operacji przechodzimy do obliczenia normy residuum przy pomocy wzoru i funkcji **norm**:



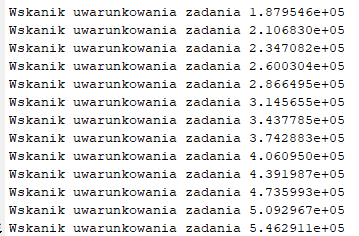
Ostatnim krokiem jest badanie zmiany residuum i czasu dla coraz większej ilości równań. Operacja ta realizowana jest w funkcji ***wykres\_residuum\_time***.

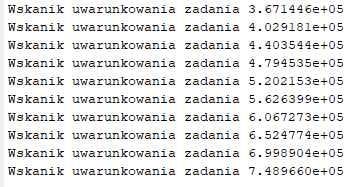
## Wynik:

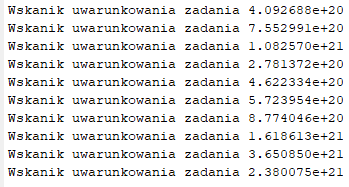
Okazało się jednak, że w przypadku danych numer 3 wyznacznik macierzy zbiega do zera. Dlatego przy odpowiednich ilościach równań MATLAB wskazuje zero.



Dla każde z typów macierzy przy zwiększaniu ilości równań zwiększa się również wskaźnik uwarunkowań. Największe wartości osiąga przy typie C macierzy.

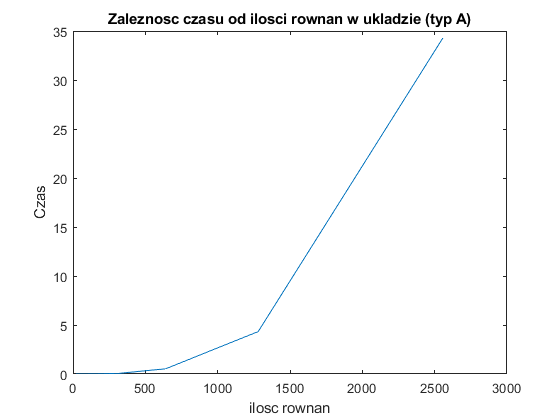




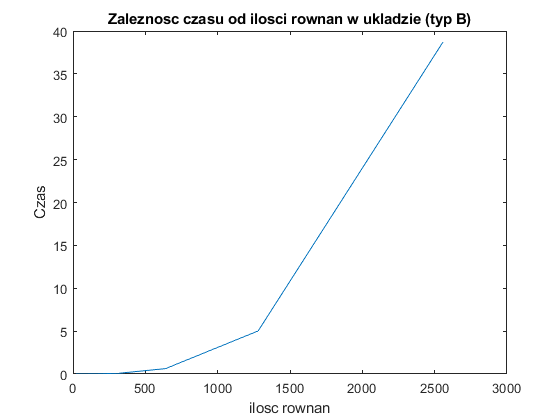


Wykresy przedstawiające zależność czasu od ilości równań:

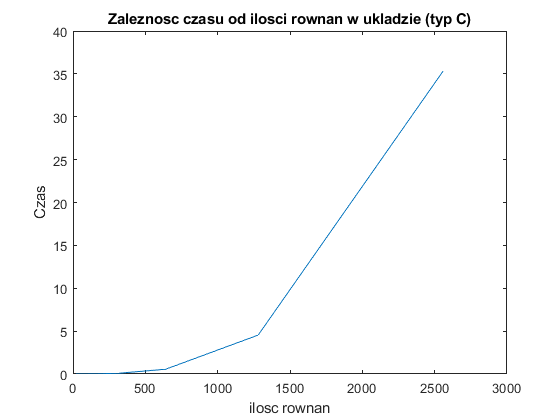
* Typ A



* Typ B

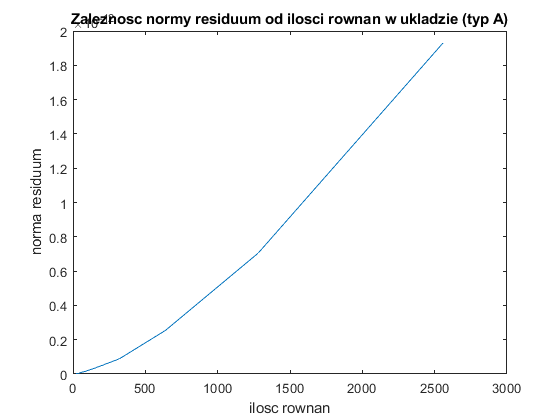


* Typ C

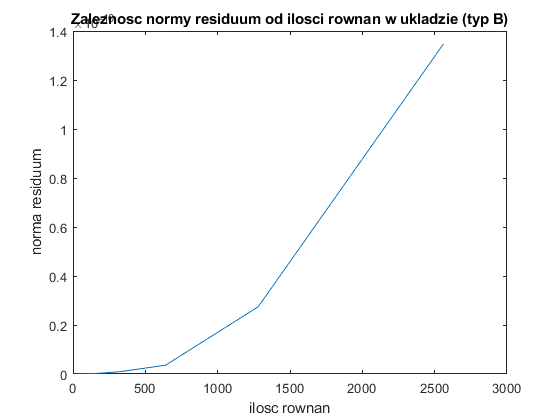


Wykresy przedstawiające zależność normy residuum od ilości równań:

* Typ A



* Typ B



* Typ C

Obraz zawierający mapa

Opis wygenerowany automatycznie

## Podsumowanie:

Dla wszystkich rodzajów macierzy czas wzrasta wraz z zwiększaniem się ilości równań co jest spodziewanym efektem. Natomiast macierz typu B obliczała się najdłużej z drugiej strony macierz A najkrócej.

Zarówno dla macierzy typa A jak i dla typu B wartość błędu rozwiązania wzrasta nieliniowo wraz z liczbą równań, ale w obu przypadkach są to względnie nieduże wartości (rząd wielkości 10−13− 10−12) , więc możemy uznać otrzymane wyniki za prawidłowe.

Problemy pojawiają się dla macierzy typu C. Tam problem był związany z wyzznacznikeim macierzy detA. Przy zwiększającej się ilości równań zbiegał on do zera. Co mogło sugerować o nietypowym zachowaniu. Jako pierwszy rzuca się w oczy nietypowy kształt wykresu, co może oznaczać o kumulacji błędów przy obliczeniach numerycznych. Jednak największym problemem jest rząd wielkości tych błędów . Jest to spowodowane złym uwarunkowaniem zadania. Jak już wspominałem wskaźnik uwarunkowań dochodzi do co jest bardzo dużą wartością.

Pojęcie numerycznego uwarunkowania zadania określa wrażliwość wyniku na zaburzenia danych. Niski (względny) wskaźnik uwarunkowania oznacza zadanie dobrze uwarunkowane, wysoki (względny) wskaźnik uwarunkowania – źle uwarunkowane. Zadanie jest źle uwarunkowane, jeśli niewielkie (względne) zaburzenia danych powodują duże (względne) zmiany jego rozwiązania.

Uwarunkowanie układu równań liniowych określa, jak zmieni się rozwiązanie układu, jeśli dane, tj. macierz 𝐴 lub wektor 𝑏 zaburzymy.

Wskaźniki uwarunkowania dla macierzy typu C mają ogromne wartości, jest to zdecydowanie źle uwarunkowane zadanie. Tak więc powraca wcześniej wspomniany problem 𝑑𝑒𝑡𝐴. Jednak jak widzimy na wykresie, czasami obliczenia układają się fortunnie i błędy się znoszą, a innym razem kumulują.