Juliusz Wasieleski
Informatyka II rok, grupa 1

Algorytmy geometryczne, laboratorium 1 - sprawozdanie

1.Opis ćwiczenia

Na pierwszych zajęciach laboratoryjnych naszym zadaniem było określanie po której stronie prostej leżą punkty. Do wykonania go używaliśmy jednego z wyznaczników:
Wyznacznik macierzy 3x3:

$$\det(a,b,c) = \begin{vmatrix} a_x & a_y & 1 \\ b_x & b_y & 1 \\ c_x & c_y & 1 \end{vmatrix}$$

Wyznacznik macierzy 2x2:

$$\det(a,b,c) = \begin{vmatrix} a_x - c_x & a_y - c_y \\ b_x - c_x & b_y - c_y \end{vmatrix}$$

W powyższym wzorze ax, ay, bx, by, cx, cy to współrzędne x i y odpowiednio punktów a, b i c. Jeżeli wyznacznik okaże się większy od 0, to punkt znajduje się po prawej stronie od prostej, a jeśli mniejszy od zera, to po lewej stronie od prostej.

Będę 0 przybliżał wartościami ε, ponieważ porównywanie liczb rzeczywistych w komputerach nie jest idealne i należy brać je z pewnym przybliżeniem. Dodatkowo mimo tego, że powyższe dwa wyznaczniki pod względem matematycznym są identyczne to przez niedokładności obliczeniowe komputerów wyniki uzyskane nimi będą mogły się różnić.

2. Środowisko, biblioteki, założenia oraz użyte narzędzia

Ćwiczenie wykonałem w Jupyter Notebook i napisałem w języku Python. Aby policzyć wyznaczniki korzystałem z biblioteki *numpy*, funkcji *det* z modułu *linalg.*

Do rysowania wykresów użyłem biblioteki *matplotlib*, która obrazowała rozkład danych w postaci graficznej. Za wartości ε przyjąłem 10^{-18}, 10^{-14}, 10^{-10}.
Do graficznej reprezentacji punktów przyjąłem konwencję:

- na niebiesko punkty powyżej prostej
- na zielono punkty poniżej prostej
- na czerwono punkty na prostej

Wszystkie obliczenia prowadziłem na komputerze Lenovo Y50-70 z systemem Windows 10 Pro w wersji 10.0.19043, procesor Intel Core i7-4720HQ 2.60GHz, 2601 MHz, rdzenie: 4, procesory logiczne: 8.

3. Plan i sposób wykonania ćwiczenia

Na początku należało wygenerować zbiory punktów o współrzędnych rzeczywistych typu float (w Pythonie mają one 64 bity):

a) 10^5 losowych punktów o współrzędnych z przedziału [-1000, 1000],
b) 10^5 losowych punktów o współrzędnych z przedziału [-10^14 , 10^14],
c) 1000 losowych punktów leżących na okręgu o środku (0,0) i promieniu R=100,
d) 1000 losowych punktów o współrzędnych z przedziału [-1000, 1000] leżących na prostej wyznaczonej przez wektor (A, B), przyjmij:

A = [-1.0, 0.0], **B** = [1.0, 0.1].

Kolejnym krokiem było zaimplementowanie funkcji obliczających wyznaczniki 2x2, 3x3 oraz skorzystanie z gotowych wersji bibliotecznych.

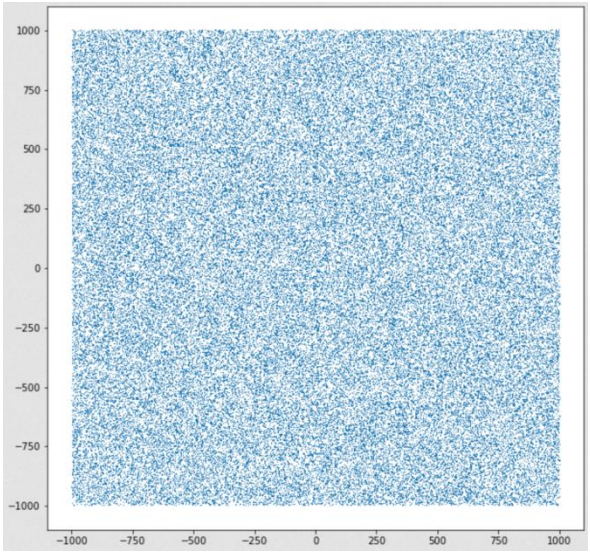
Następnie przedstawiłem uzyskane wyniki na wykresach oraz sprawdziłem jakie są różnice między moim wyznacznikiem 2x2 i bibliotecznym oraz moim wyznacznikiem 3x3 i bibliotecznym. Na koniec porównałem wyniki które otrzymywałem przy różnych wartościach ε.

4. Wykonanie ćwiczenia

4.1 Wygenerowanie punktów

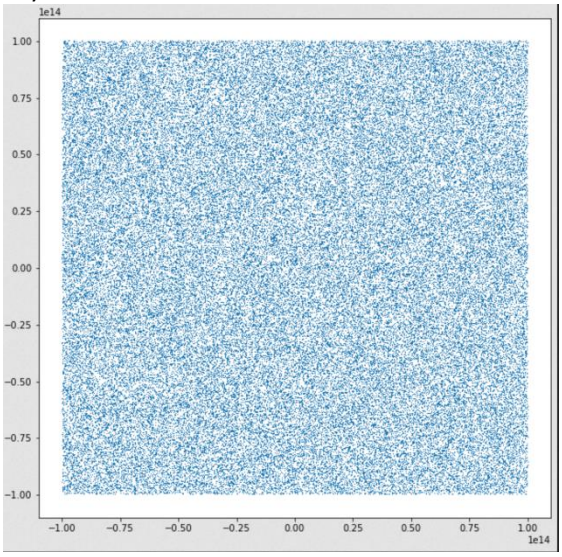
Do wygenerowania punktów użyłem funkcji *uniform* z biblioteki *random*, następnie przedstawiłem na wykresach otrzymane zbiory punktów:

a) Zbiór A:



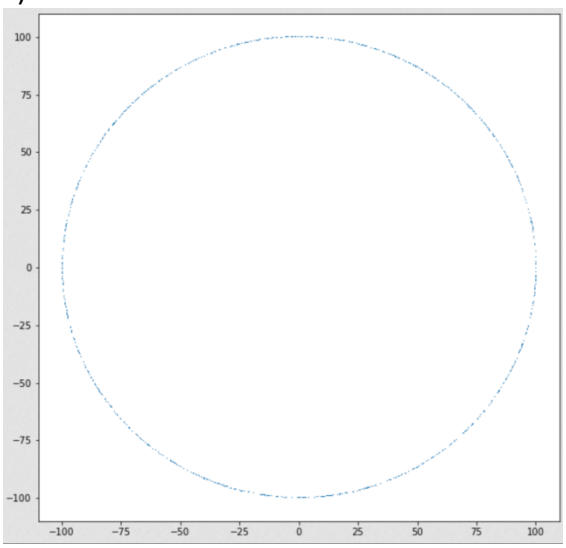
(rys. 1)

b) Zbiór B:



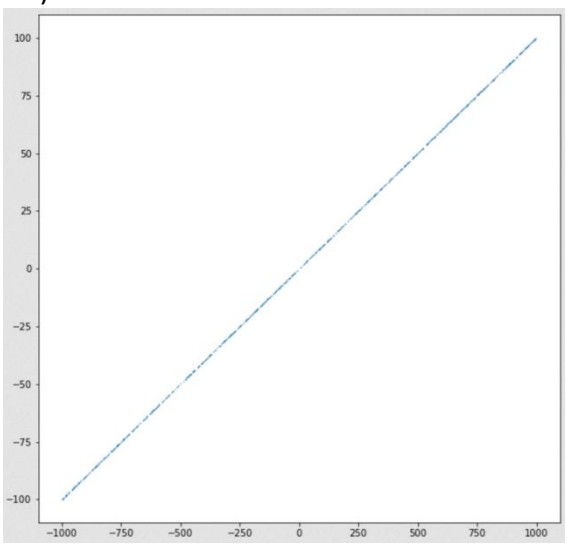
(rys.2)

c) Zbiór C:



(rys.3)

d) Zbiór D:



(rys.4)

4.2 Klasyfikacja punktów zależnie od wyznacznika i tolerancji

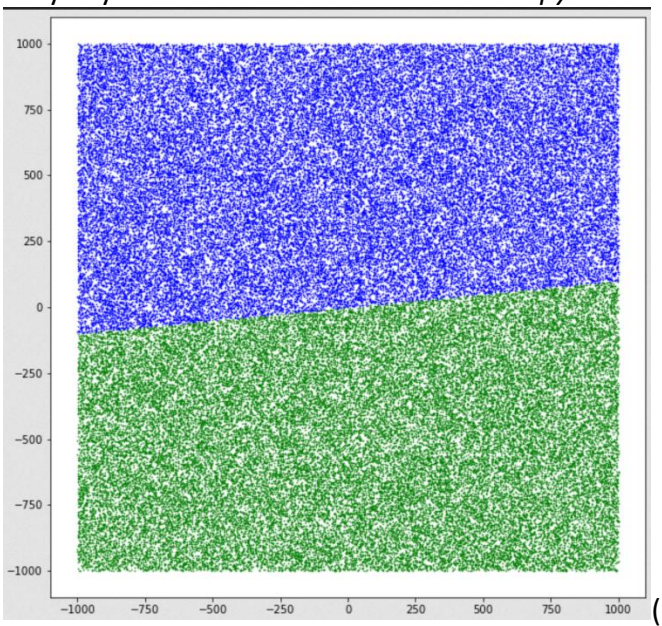
W celu klasyfikacji punktów zaimplementowałem funkcję *make_groups,* która dodatkowo pogrupowała te punkty jako znajdujące się nad prostą, pod prostą oraz na prostej. Grupowania wykonałem w trzech wariantach zależnych od wartości ε.

Spośród wszystkich szesnastu wykresów wybrałem przykładowe cztery (rys.5, rys.6, rys.7, rys.8). Na wykresach pokazałem podział dla najmniejszej wartości ε = 10^{-18}. W poniższej tabeli przedstawiłem wyniki tego grupowania.

Wyznacznik: / **Zbiór:**	**2x2 z biblioteki *numpy***	**2x2 mojej implementacji**	**3x3 z biblioteki *numpy***	**3x3 mojej implementacji**
A	Nad: 50157 Na: 0 Pod: 49843	Nad: 50157 Na: 0 Pod: 49843	Nad: 50157 Na: 0 Pod: 49843	Nad: 50157 Na: 0 Pod: 49843
B	Nad: 49838 Na: 0 Pod: 50162	Nad: 49825 Na: 24 Pod: 50151	Nad: 49834 Na: 0 Pod: 50166	Nad: 49834 Na: 0 Pod: 50166
C	Nad: 505 Na: 0 Pod: 495	Nad: 505 Na: 0 Pod: 495	Nad: 505 Na: 0 Pod: 495	Nad: 505 Na: 0 Pod: 495
D	Nad: 510 Na: 0 Pod: 490	Nad: 179 Na: 668 Pod: 153	Nad: 342 Na: 195 Pod: 463	Nad: 0 Na: 1000 Pod: 0

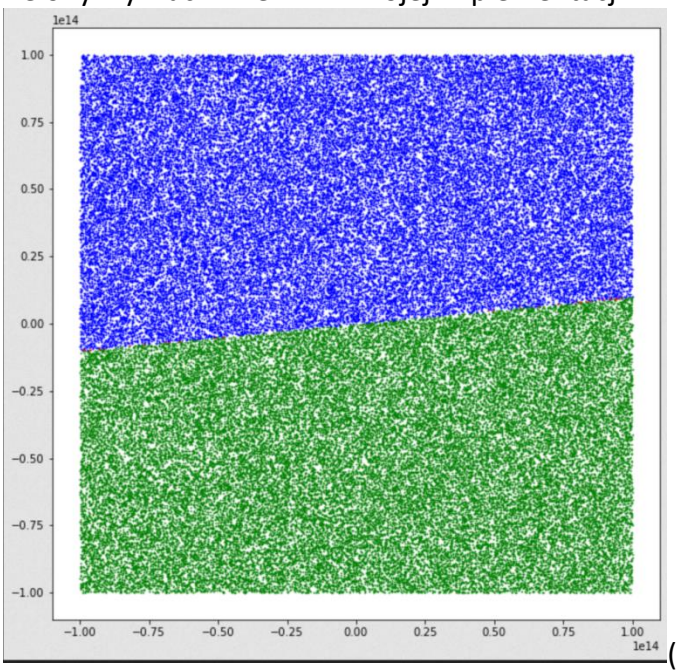
(Tabela 1)

a) Zbiór A podzielony wyznacznikiem 2x2 z biblioteki *numpy*:



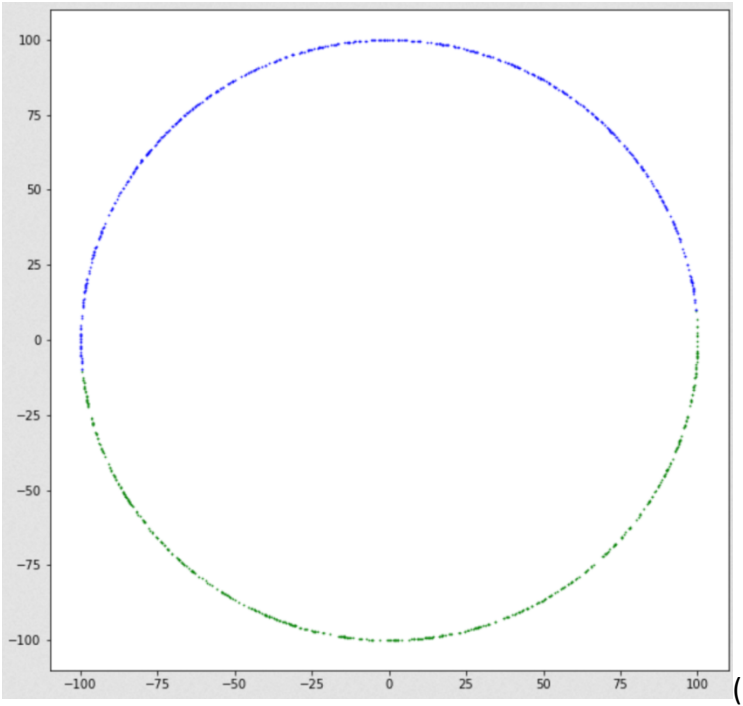
(rys. 5)

b) Zbiór B podzielony wyznacznikiem 2x2 mojej implementacji:



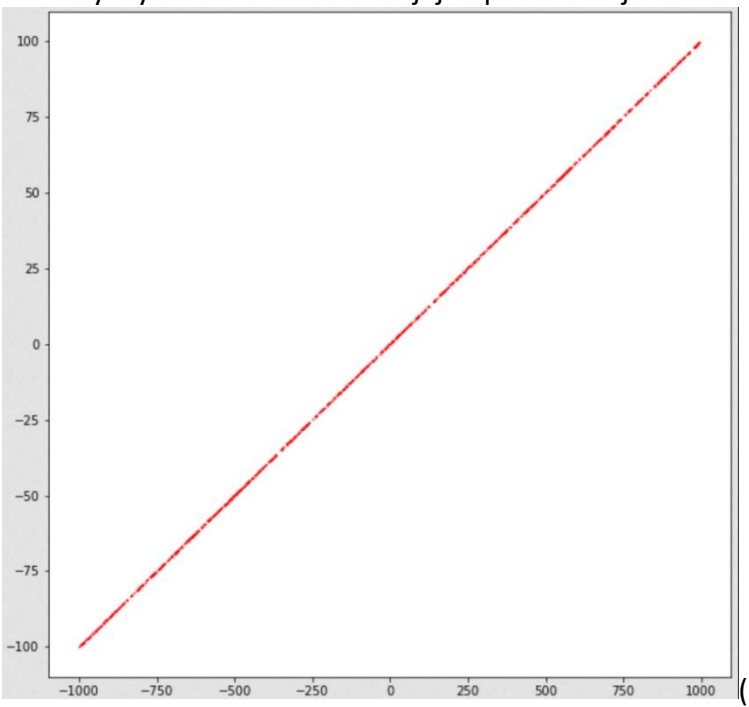
(rys. 6)

c) Zbiór C podzielony wyznacznikiem 3x3 z biblioteki *numpy*:



(rys. 7)

d) Zbiór D podzielony wyznacznikiem 3x3 mojej implementacji:



(rys. 8)

4.3 Porównanie klasyfikacji punktów ze względu na wyznacznik

W celu realizacji zaimplementowałem funkcję *differences*, a do wizualizacji wyników funkcję *show_differences*.

Porównywałem wyznaczniki 2x2 z sobą oraz 3x3 z sobą. Następnie wypisywałem, które punkty się nie zgadzają między tymi wyznacznikami oraz rysowałem je na wykresie. Podsumowanie otrzymanych wyników:

a) W zestawie A nie ma różnic
b) W zestawie B różnice są tylko między wyznacznikami 2x2 (28 punktów jest zaklasyfikowanych inaczej)
c) W zestawie C nie ma różnic
d) W zestawie D wyznaczniki 2x2 różnią się 695 punktami, a 3x3 786 punktami

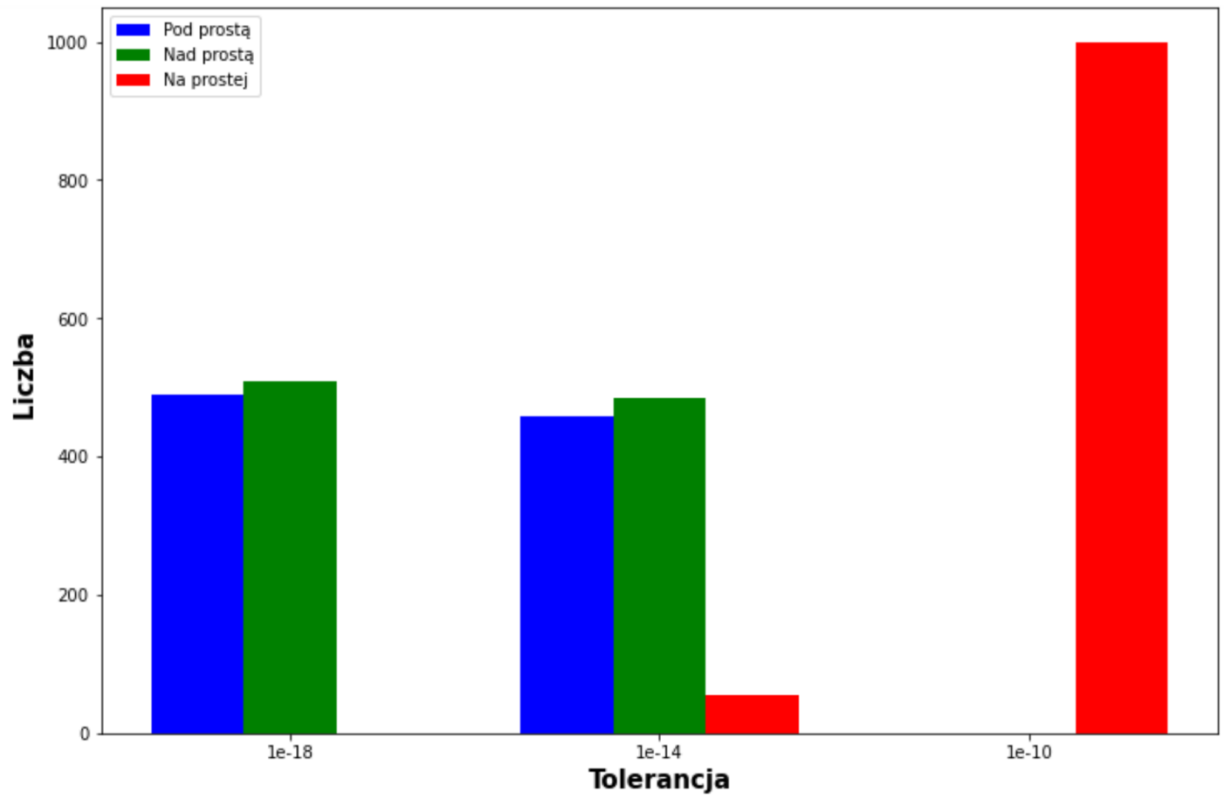
Powyższe wyniki były wyznaczane dla $\varepsilon = 10^{-18}$. Przy niższych tolerancjach nie byłoby aż tak dużych różnic. Dokładniej omówię to w podpunkcie 4.4. Wykresy punktów są dostępne w pliku Jupyter Notebook, nie umieszczam ich poniżej gdyż wszystkie leżą prawie na prostej wskazywanej przez wektor AB, więc na wykresach upodabniają tę prostą.

4.4 Różnice między wynikami ze względu na tolerancję

Jako ostatnią zaimplementowałem funkcję *generate_diagram*, która rysuje wykresy słupkowe przedstawiające dla każdej tolerancji ε ile punktów jest klasyfikowanych pod, ile nad, a ile na prostej.

Znaczące różnice pojawiły się jedynie w danych z zestawu D. Poniżej zamieszczam wykresy tych różnic oraz podsumowanie ile punktów jak było zaklasyfikowanych. Pozostałe wykresy są w Jupyter Notebook.

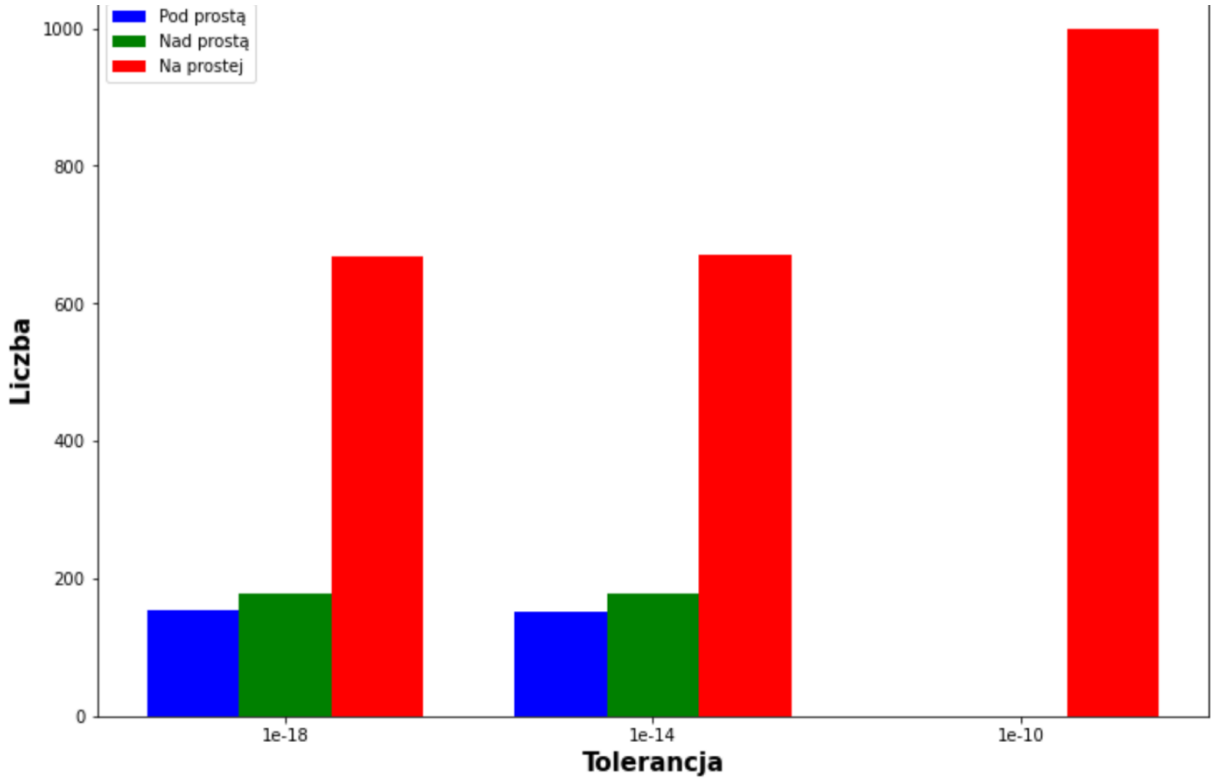
a) Obliczenia wyznacznikiem 2x2 z bilbioteki *numpy:*




```
                Pod  :  Nad  : Na linii
Precyzja 1e-18 :  490  :  510  :   0
Precyzja 1e-14 :  459  :  485  :  56
Precyzja 1e-10 :   0   :   0   :  1000
```

(rys. 9)

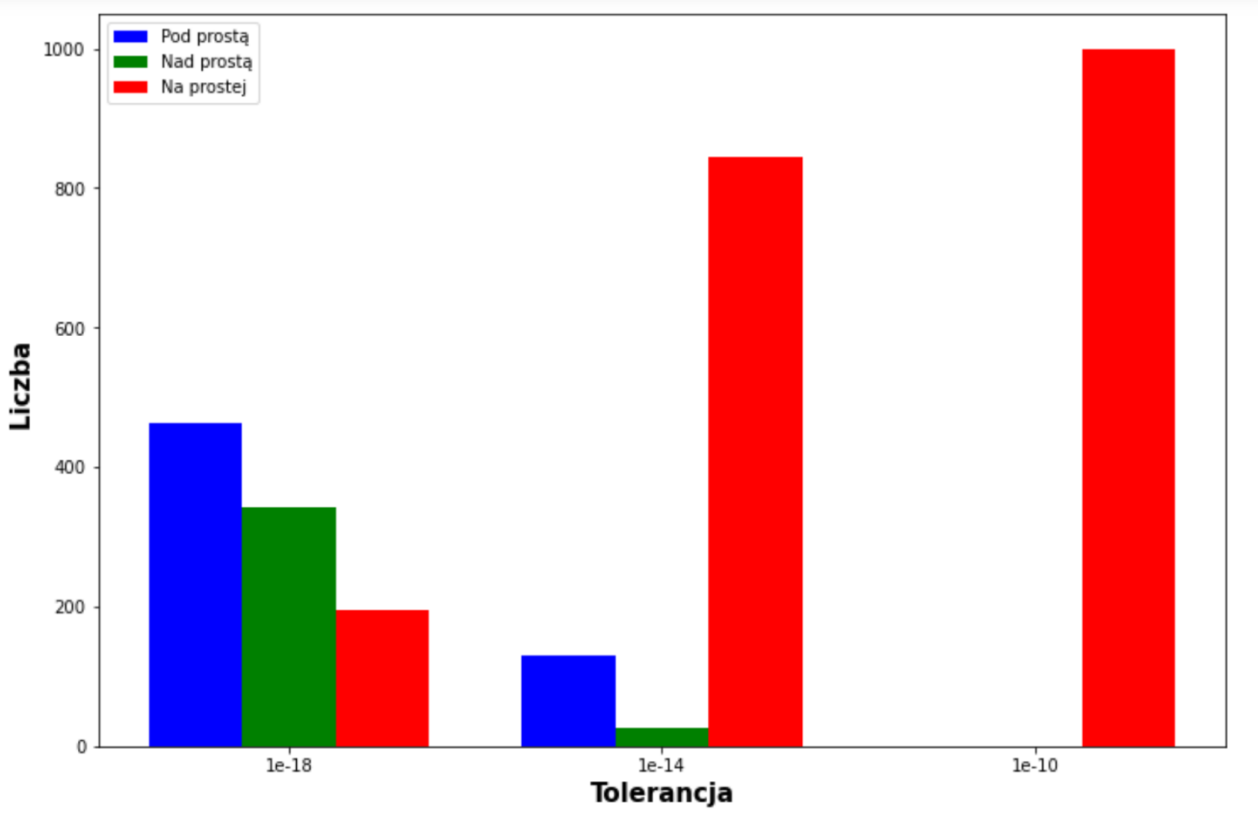
b) Obliczenia wyznacznikiem 2x2 mojej implementacji:




```
                Pod  :  Nad  : Na linii
Precyzja 1e-18 :  153  :  179  :  668
Precyzja 1e-14 :  151  :  178  :  671
Precyzja 1e-10 :   0   :   0   :  1000
```

(rys. 10)

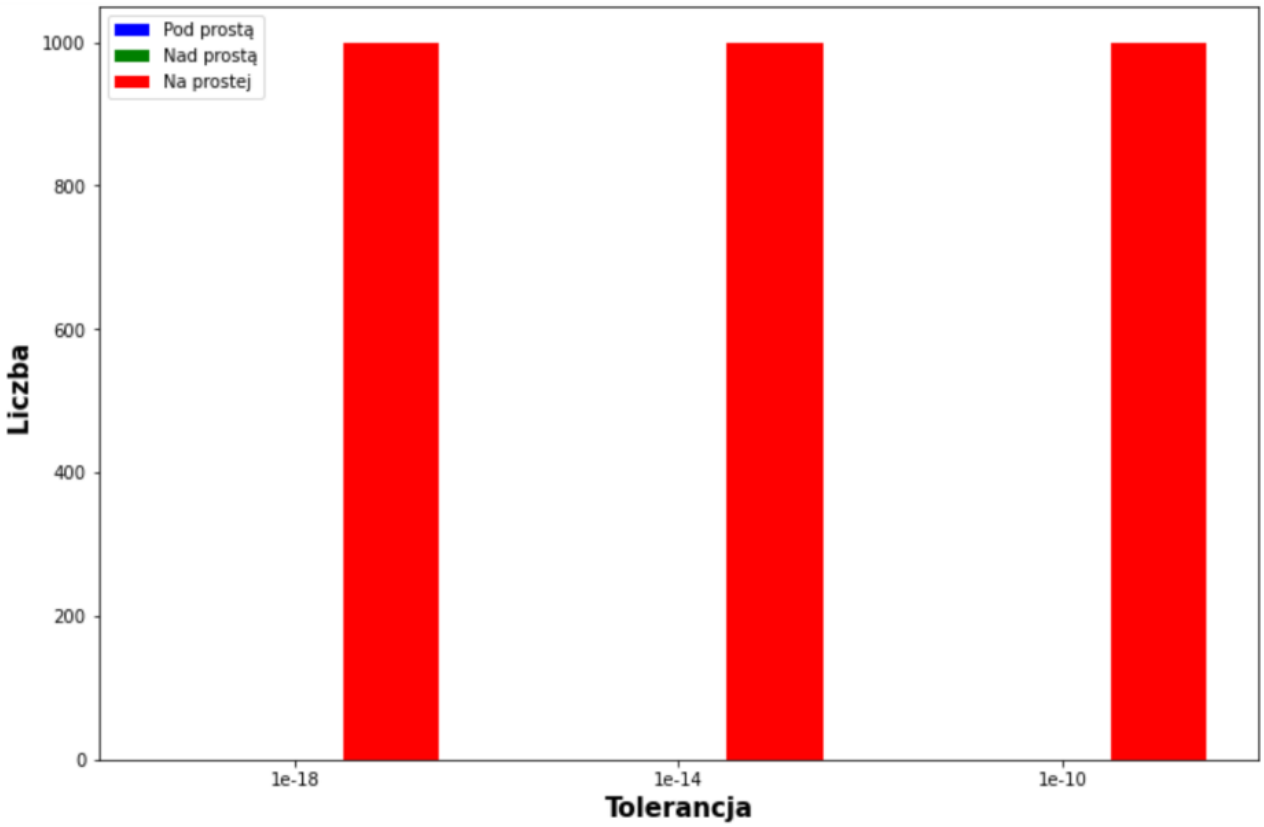
c) Obliczenia wyznacznikiem 3x3 z biblioteki *numpy*:




```
                Pod  :  Nad  : Na linii
Precyzja 1e-18 :  463  :  342  :  195
Precyzja 1e-14 :  130  :  25   :  845
Precyzja 1e-10 :   0   :   0   :  1000
```

(rys. 11)

d) Obliczenia wyznacznikiem 3x3 mojej implementacji:




```
                  Pod :  Nad  : Na linii
Precyzja 1e-18 : 0 : 0 : 1000
Precyzja 1e-14 : 0 : 0 : 1000
Precyzja 1e-10 : 0 : 0 : 1000
```

(rys. 12)

7. Wnioski

Dla danych ze zbioru:

- A: ani wyznacznik, ani ε nie mają znaczenia
- B: wyznacznik ma znaczenie, ale ε już nie
- C: ani wyznacznik, ani ε nie mają znaczenia
- D: zarówno wyznacznik jak i ε mają znaczenie

Wyniki dla zbiorów A i C są najprawdopodobniej spowodowane tym, że istnieje bardzo małe prawdopodobieństwo, iż punkt będzie znajdował się dokładnie na prostej, z dokładnością do użytej tolerancji oraz punkty są z relatywnie nie aż tak dużego zakresu.

W wynikach dla zbioru B jedynie wyznacznik 2x2 mojej implementacji dał inne wyniki od pozostałych. Zapewne wynika to z faktu, że współrzędne z tego zbioru mają duże wartości bezwzględne i podczas klasyfikacji dokładność jest tracona na rzecz dużych liczb.

Dla zbioru D najlepsze wyniki (najwięcej punktów zaklasyfikowanych do prostej) uzyskałem przy dokładności 10^{-10}. Wtedy już wszystkie punkty były klasyfikowane do prostej niezależnie od wyznacznika.

Co ciekawe, lepsze wyniki dawały wyznaczniki mojej implementacji niż biblioteczne. Możliwe, że jest to wynikiem mniejszej dokładności funkcji bibliotecznych, która jest spowodowana optymalizacją obliczeń. Widać też, że wyznaczniki 3x3 „lepiej” radzą sobie

z tymi obliczeniami niż 2x2. Najprawdopodobniej jest tak, ponieważ w wyznacznikach 3x3 dokonujemy mniej przybliżeń.

Dodatkowo, zaskakujące jest to, że nawet przy tolerancji rzędu 10^{-18} mój wyznacznik klasyfikuje wszystkie punkty ze zbioru D jako znajdujące się na prostej (czyli poprawnie). Zdarzało się tak za każdym razem podczas generowania nowego zbioru punktów (wykonałem 7 generowań). Nie jestem pewien co może być tego przyczyną, być może przypadkiem generuję punkty zbyt tendencyjnie lub miałem szczęście, że za każdym razem mój wyznacznik „poradził sobie” z taką niedokładnością.