

# Algorytmy geometryczne

## laboratorium 1 - sprawozdanie

Radosław Rolka  
Informatyka WI, II rok  
Gr 6, tydz. A czwartek 13:00

### Spis treści

<b>1</b>	<b>Wstęp</b>	<b>2</b>
1.1	Cel ćwiczenia	2
1.2	Program ćwiczenia	2
<b>2</b>	<b>Wykorzystane narzędzia</b>	<b>3</b>
2.1	Środowisko	3
2.2	Sprzęt	3
<b>3</b>	<b>Przebieg ćwiczeń</b>	<b>3</b>
3.1	Generowanie punktów	3
3.2	Przygotowanie wyznaczników	4
<b>4</b>	<b>Klasyfikacja poszczególnych podziałów punktów</b>	<b>4</b>
4.1	Ustalenie podziału na prostej	4
4.2	Klasyfikacja pozostałych zbiorów	7
4.3	Użycie precyzji float32	9
<b>5</b>	<b>Wnioski</b>	<b>10</b>

# 1 Wstęp

## 1.1 Cel ćwiczenia

Ćwiczenie wprowadzające w zagadnienia geometrii obliczeniowej – implementacja podstawowych predykatów geometrycznych, przeprowadzenie testów, wizualizacja i opracowanie wyników.

## 1.2 Program ćwiczenia

- Przygotuj następujące zbiory punktów (2D, współrzędne rzeczywiste typu double):
  - 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  $a = [-1.0, 0.0]$ ,  $b = [1.0, 0.1]$ .
- Uruchom wizualizację graficzną utworzonych zbiorów punktów.
- Przygotuj program, który dla każdego ze zbioru danych dokona podziału punktów względem ich orientacji w stosunku do odcinka  $ab$  ( $a = [-1.0, 0.0]$ ,  $b = [1.0, 0.1]$ ) – punkty znajdujące się po lewej stronie, po prawej stronie oraz współliniowe. Obliczenia wykonaj przy pomocy wyznacznika (1) i następnie (2) zaimplementowanego samodzielnie. Wyszukaj w bibliotekach numerycznych procedury obliczania wyznacznika  $3 \times 3$  i  $2 \times 2$ . Dla każdego zbioru danych porównaj wyniki (podział punktów) uzyskane przy pomocy obu wyznaczników wyliczanych procedurami własnymi i bibliotecznymi. Określ, ile punktów (i jakich) zostało inaczej zakwalifikowanych dla różnych sposobów liczenia wyznacznika. Zbadaj wyniki dla różnej tolerancji dla zera oraz różnych precyzji obliczeń. Odpowiednio zaprezentuj otrzymane wyniki w tabelach.
- Przedstaw graficznie różnice w podziale punktów.
- Opisz wnioski.

## 2 Wykorzystane narzędzia

### 2.1 Środowisko

Ćwiczenie zostało wykonane w Jupyter Notebook wykorzystując język programowania Python oraz dodatkowe biblioteki, które zostały zawarte w projekcie dostarczonym na zajęciach.

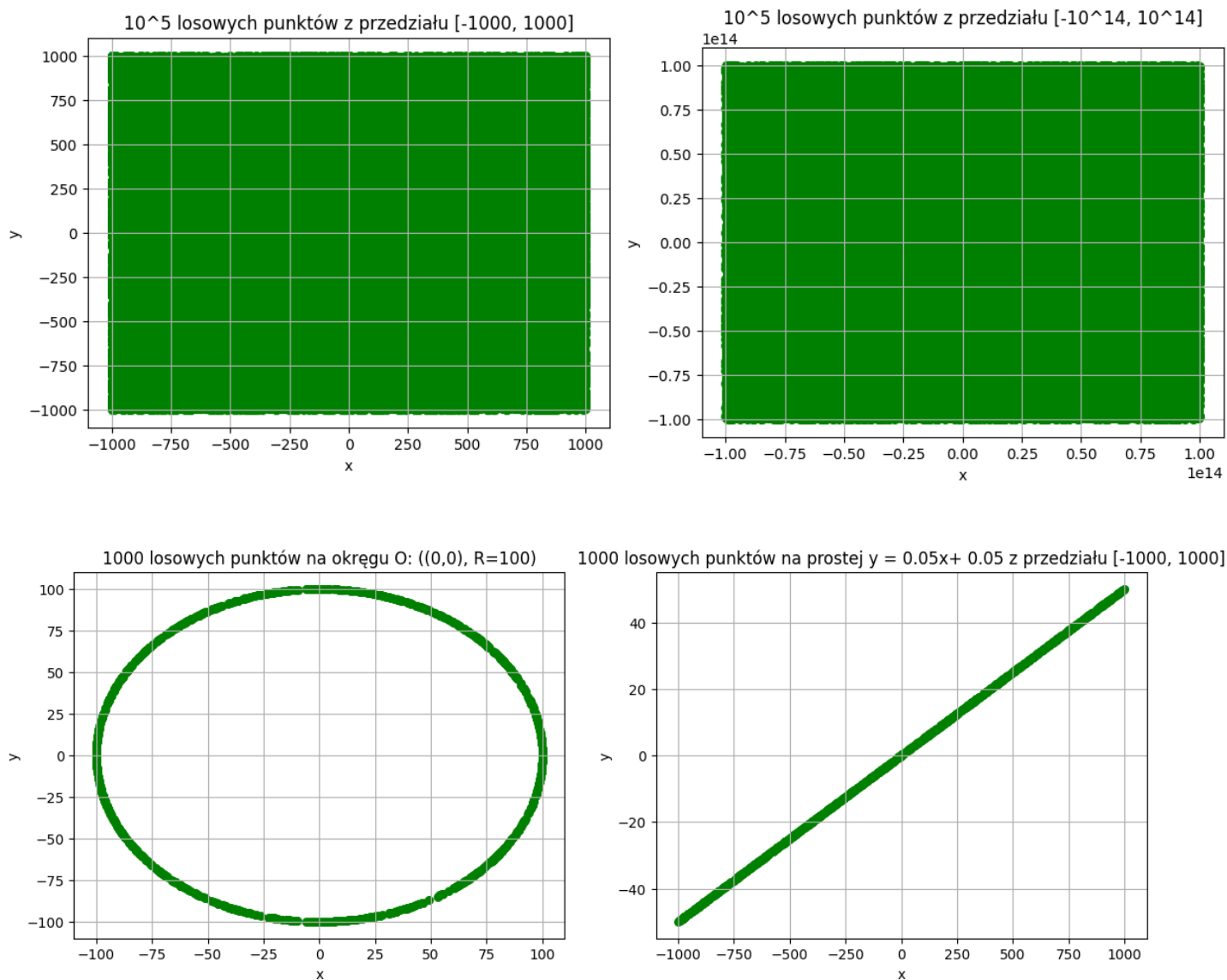
### 2.2 Sprzęt

Do wykonania został wykorzystany procesor Intel(R) Core(TM) i5-10300H 2.50GHz oraz system operacyjny Microsoft Windows 10 64bit ver 22H2.

## 3 Przebieg ćwiczeń

### 3.1 Generowanie punktów

Do generowania punktów wykorzystałem funkcję `numpy.random.uniform`, aby uzyskać równomierny rozkład punktów.



Rysunek 1: Wykresy rozkładów wygenerowanych punktów

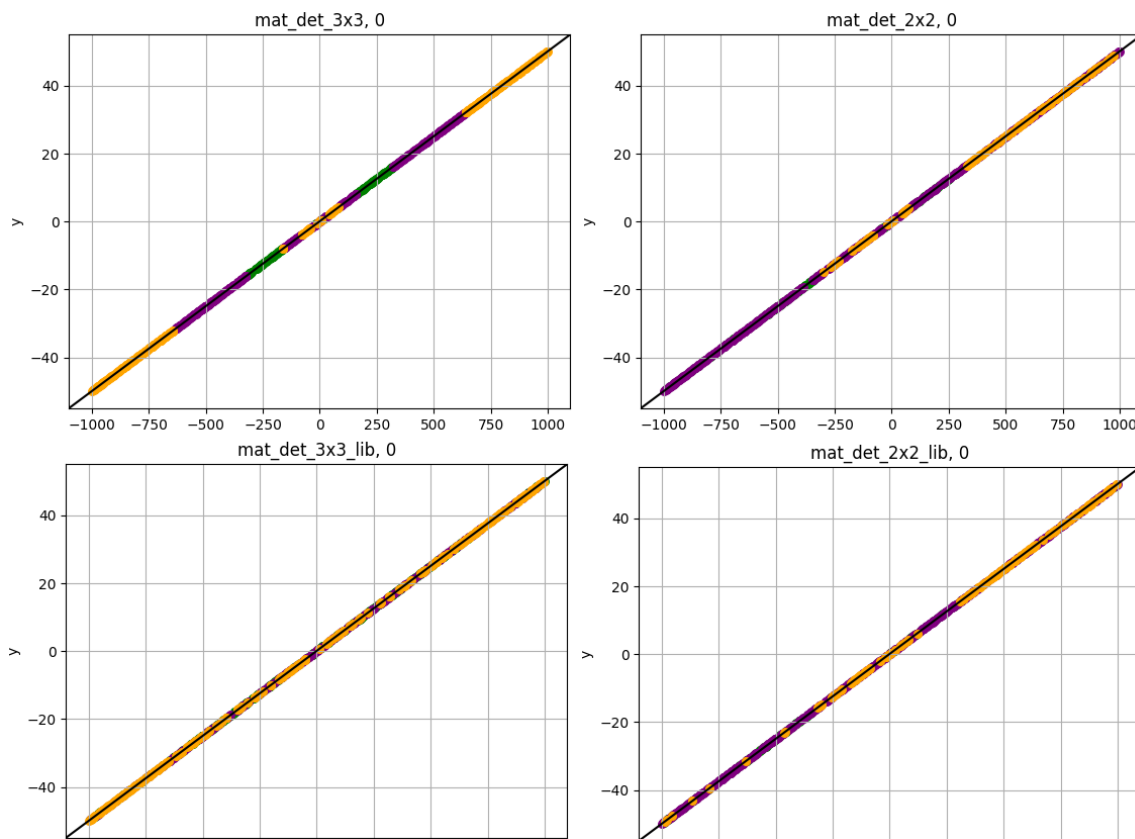
## 3.2 Przygotowanie wyznaczników

Do obliczenia wyznacznika macierzy 2x2 oraz 3x3 wykorzystałem funkcje zaimplementowane przez siebie oraz funkcję *numpy.linalg*.

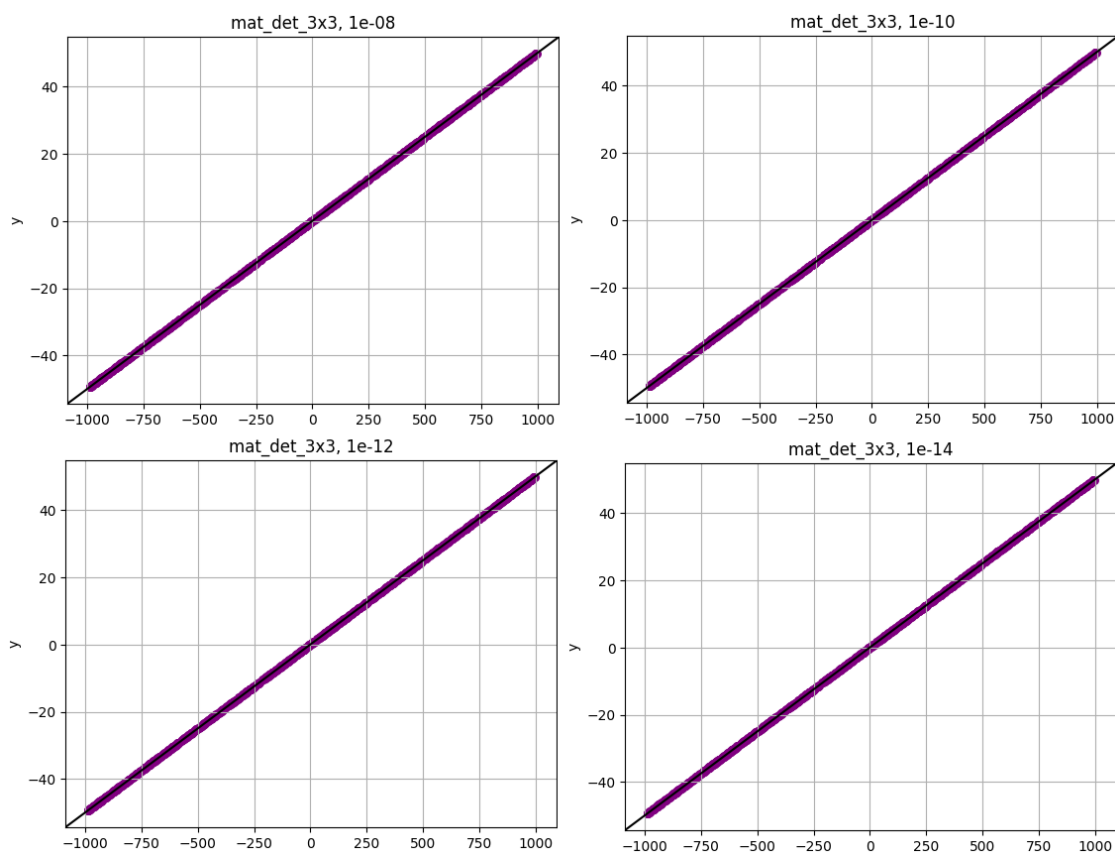
## 4 Klasyfikacja poszczególnych podziałów punktów

### 4.1 Ustalenie podziału na prostej

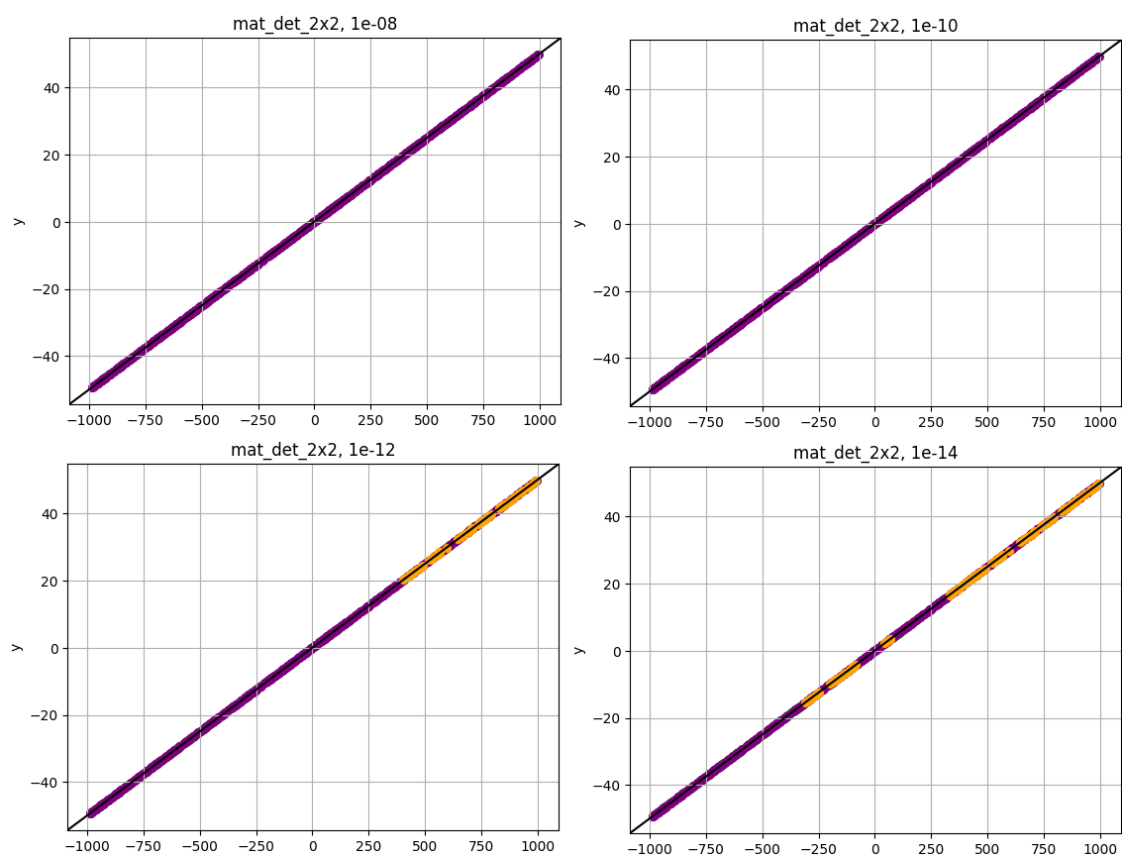
Najpierw przystąpię do klasyfikacji zbioru wygenerowanego na prostej  $ab : y = 0.05x + 0.05$ . Według przewidywań wszystkie punkty powinny się zakwalifikować jako punkty na prostej. Punkty oznaczone kolorem fioletowym oznaczają klasyfikację na prostej (prawidłowa), natomiast kolor żółty/zielony oznacza klasyfikację poza prostą.



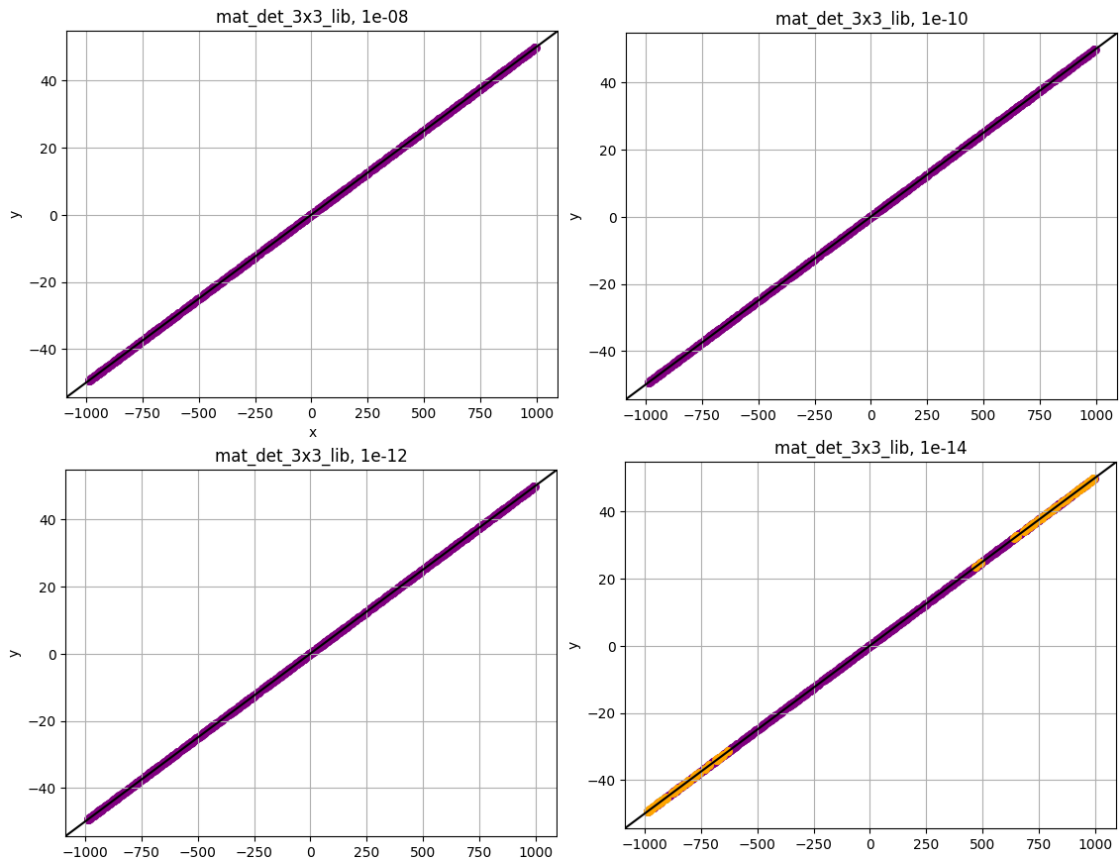
Rysunek 2: Klasyfikacje dla epsilon równo 0



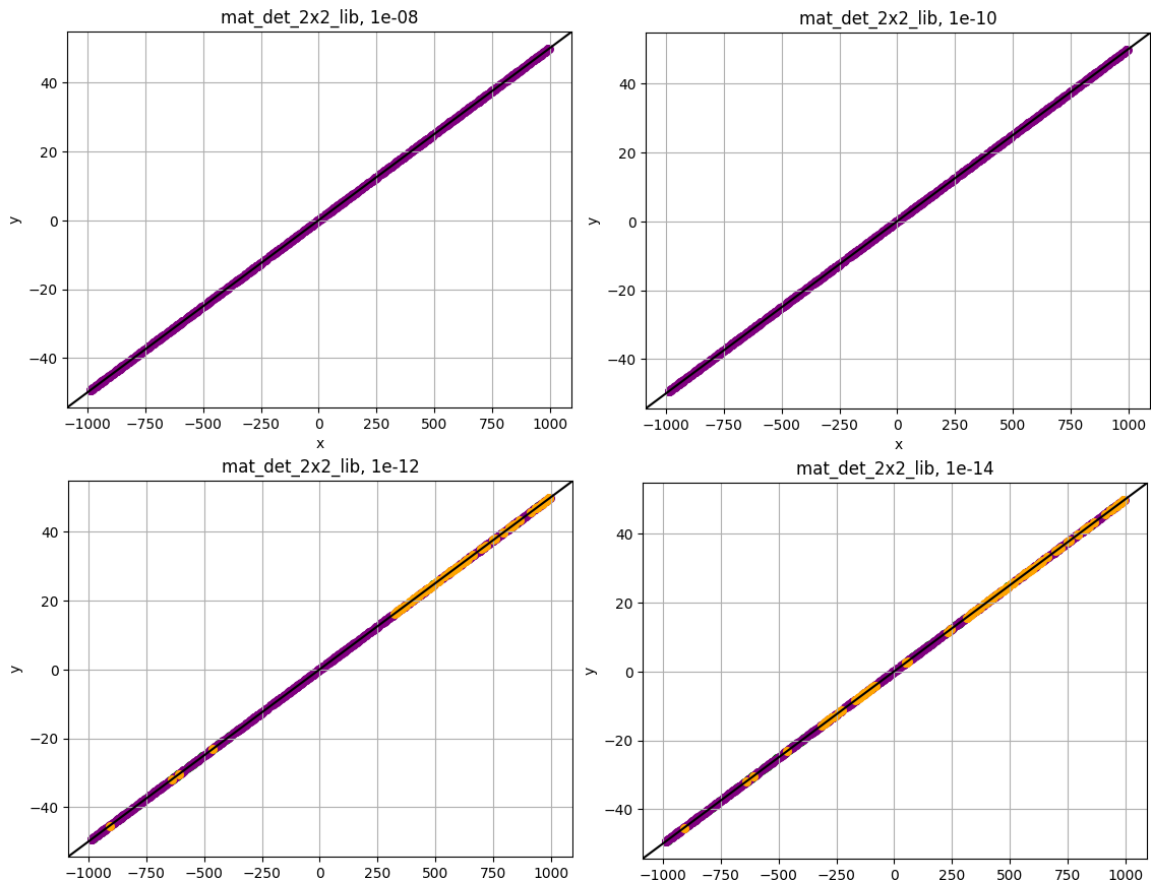
Rysunek 3: Klasyfikacje dla wyznacznika `mat_det_3x3`



Rysunek 4: Klasyfikacje dla wyznacznika `mat_det_2x2`



Rysunek 5: Klasyfikacje dla wyznacznika `mat_det_3x3_lib`



Rysunek 6: Klasyfikacje dla wyznacznika `mat_det_2x2_lib`

Wszystkie wyniki zebrałem w poniższej tabeli z uwzględnieniem procentowej zgodności klasyfikacji dla poszczególnych epsilon'ów. Gdzie zgodność oznacza stosunek poprawnie sklasyfikowanych punktów przez liczbę wszystkich punktów

Determinant	0.00e-00	1.00e-08	1.00e-10	1.00e-12	1.00e-14
mat_det_3x3	0,43	1,00	1,00	1,00	1,00
mat_det_2x2	0,70	1,00	1,00	0,84	0,74
mat_det_3x3_lib	0,30	1,00	1,00	1,00	0,83
mat_det_2x2_lib	0,67	1,00	1,00	0,82	0,73

Kolejność "sprawności" wyznaczników na podstawie przeprowadzonych klasyfikacji:

- mat\_det\_3x3
- mat\_det\_3x3\_lib
- mat\_det\_2x2
- mat\_det\_2x2\_lib

## 4.2 Klasyfikacja pozostałych zbiorów

Teraz sprawdzę klasyfikację pozostałych zbiorów dla wszystkich czterech wyznaczników oraz dla epsilon'ów równych  $0, 10^{-8}, 10^{-10}, 10^{-12}, 10^{-14}$ .

Determinant	0.00e-00	1.00e-08	1.00e-10	1.00e-12	1.00e-14
mat_det_3x3	(50030, 0, 49970)	(50030, 0, 49970)	(50030, 0, 49970)	(50030, 0, 49970)	(50030, 0, 49970)
mat_det_2x2	(50030, 0, 49970)	(50030, 0, 49970)	(50030, 0, 49970)	(50030, 0, 49970)	(50030, 0, 49970)
mat_det_3x3_lib	(50030, 0, 49970)	(50030, 0, 49970)	(50030, 0, 49970)	(50030, 0, 49970)	(50030, 0, 49970)
mat_det_2x2_lib	(50030, 0, 49970)	(50030, 0, 49970)	(50030, 0, 49970)	(50030, 0, 49970)	(50030, 0, 49970)

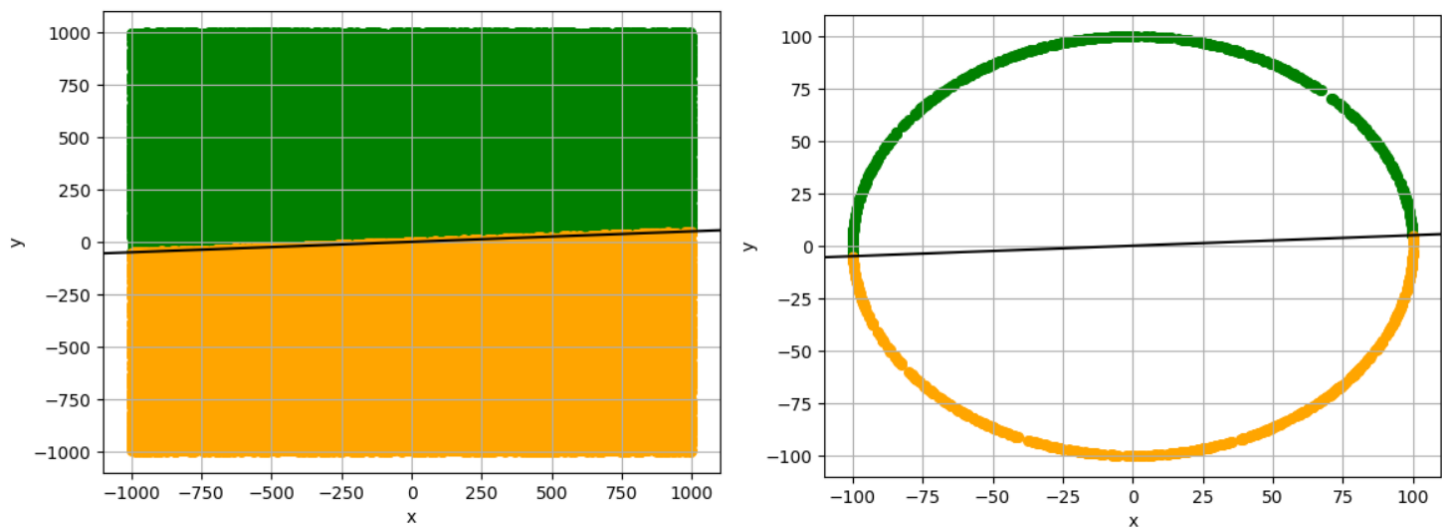
Rysunek 7: Klasyfikacja dla pierwszego zbioru  $[-1000, 1000]$

Determinant	0.00e-00	1.00e-08	1.00e-10	1.00e-12	1.00e-14
mat_det_3x3	(50139, 0, 49861)	(50139, 0, 49861)	(50139, 0, 49861)	(50139, 0, 49861)	(50139, 0, 49861)
mat_det_2x2	(50137, 4, 49859)	(50137, 4, 49859)	(50137, 4, 49859)	(50137, 4, 49859)	(50137, 4, 49859)
mat_det_3x3_lib	(50139, 0, 49861)	(50139, 0, 49861)	(50139, 0, 49861)	(50139, 0, 49861)	(50139, 0, 49861)
mat_det_2x2_lib	(50136, 6, 49858)	(50136, 6, 49858)	(50136, 6, 49858)	(50136, 6, 49858)	(50136, 6, 49858)

Rysunek 8: Klasyfikacja dla drugiego zbioru  $[-10^{14}, 10^{14}]$

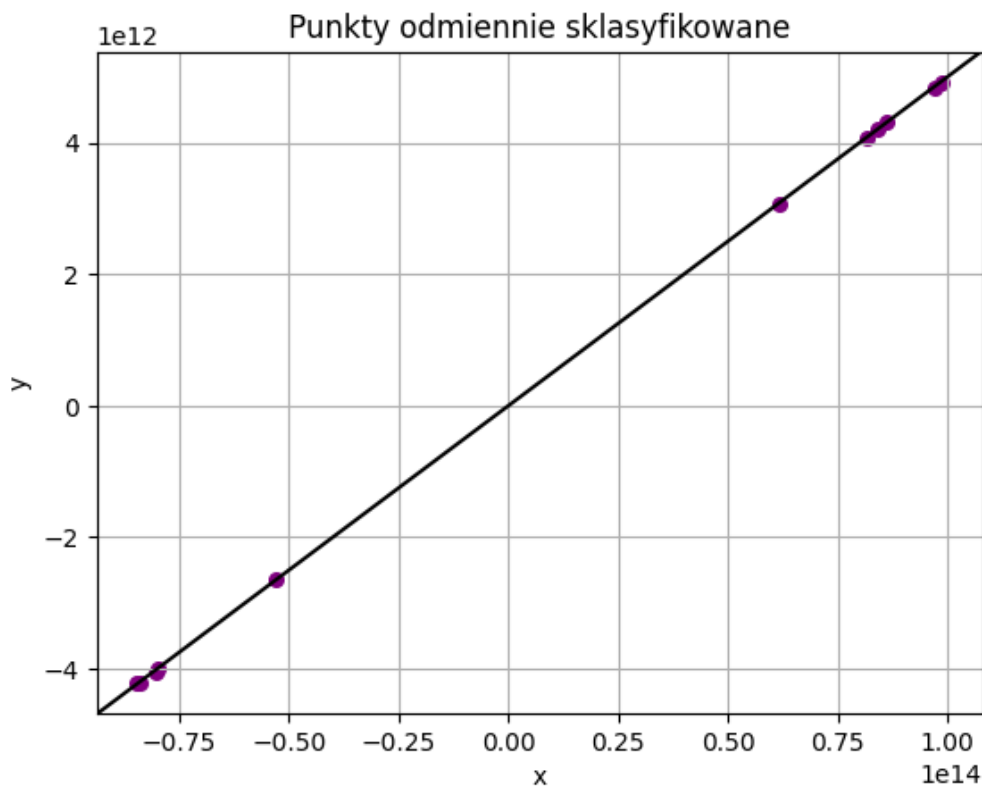
Determinant	0.00e-00	1.00e-08	1.00e-10	1.00e-12	1.00e-14
mat_det_3x3	(488, 0, 512)	(488, 0, 512)	(488, 0, 512)	(488, 0, 512)	(488, 0, 512)
mat_det_2x2	(488, 0, 512)	(488, 0, 512)	(488, 0, 512)	(488, 0, 512)	(488, 0, 512)
mat_det_3x3_lib	(488, 0, 512)	(488, 0, 512)	(488, 0, 512)	(488, 0, 512)	(488, 0, 512)
mat_det_2x2_lib	(488, 0, 512)	(488, 0, 512)	(488, 0, 512)	(488, 0, 512)	(488, 0, 512)

Rysunek 9: Klasyfikacja dla trzeciego zbioru [okrąg]



Rysunek 10: Przedstawienie Klasyfikacja zbiorów

Wszystkie wyznaczniki poradziły sobie jednakowo. Jednakże pojawiły się niespodziewane klasyfikacje na linii dla zbioru  $[10^{-14}, 10^{14}]$ .



Rysunek 11: Przedstawienie odstających punktów



### 4.3 Użycie precyzji float32

Dla zmniejszonej precyzji float32 (Python używa domyślnie float64) wygeneruję punkty i zklasyfikuje w taki sam sposób jak powyżej.

Determinant	0.00e-00	1.00e-08	1.00e-10	1.00e-12	1.00e-14
mat_det_3x3	0,43	1,00	1,00	1,00	1,00
mat_det_2x2	0,70	1,00	1,00	0,84	0,74
mat_det_3x3_lib	0,30	1,00	1,00	1,00	0,83
mat_det_2x2_lib	0,67	1,00	1,00	0,82	0,73

Rysunek 12: Klasyfikacja float32 dla prostej

Determinant	0.00e-00	1.00e-08	1.00e-10	1.00e-12	1.00e-14
mat_det_3x3	(50030, 0, 49970)	(50030, 0, 49970)	(50030, 0, 49970)	(50030, 0, 49970)	(50030, 0, 49970)
mat_det_2x2	(50030, 0, 49970)	(50030, 0, 49970)	(50030, 0, 49970)	(50030, 0, 49970)	(50030, 0, 49970)
mat_det_3x3_lib	(50030, 0, 49970)	(50030, 0, 49970)	(50030, 0, 49970)	(50030, 0, 49970)	(50030, 0, 49970)
mat_det_2x2_lib	(50030, 0, 49970)	(50030, 0, 49970)	(50030, 0, 49970)	(50030, 0, 49970)	(50030, 0, 49970)

Rysunek 13: Klasyfikacja float32 dla pierwszego zbioru  $[-1000, 1000]$

Determinant	0.00e-00	1.00e-08	1.00e-10	1.00e-12	1.00e-14
mat_det_3x3	(50139, 0, 49861)	(50139, 0, 49861)	(50139, 0, 49861)	(50139, 0, 49861)	(50139, 0, 49861)
mat_det_2x2	(50136, 6, 49858)	(50136, 6, 49858)	(50136, 6, 49858)	(50136, 6, 49858)	(50136, 6, 49858)
mat_det_3x3_lib	(50139, 0, 49861)	(50139, 0, 49861)	(50139, 0, 49861)	(50139, 0, 49861)	(50139, 0, 49861)
mat_det_2x2_lib	(50136, 6, 49858)	(50136, 6, 49858)	(50136, 6, 49858)	(50136, 6, 49858)	(50136, 6, 49858)

Rysunek 14: Klasyfikacja float32 dla drugiego zbioru  $[-10^{14}, 10^{14}]$

Determinant	0.00e-00	1.00e-08	1.00e-10	1.00e-12	1.00e-14
mat_det_3x3	(488, 0, 512)	(488, 0, 512)	(488, 0, 512)	(488, 0, 512)	(488, 0, 512)
mat_det_2x2	(488, 0, 512)	(488, 0, 512)	(488, 0, 512)	(488, 0, 512)	(488, 0, 512)
mat_det_3x3_lib	(488, 0, 512)	(488, 0, 512)	(488, 0, 512)	(488, 0, 512)	(488, 0, 512)
mat_det_2x2_lib	(488, 0, 512)	(488, 0, 512)	(488, 0, 512)	(488, 0, 512)	(488, 0, 512)

Rysunek 15: Klasyfikacja float32 dla trzeciego zbioru [okrąg]

Wszystkie wyniki są identyczne, oprócz drobnych różnic w zbiorze drugim dla mat\_det\_2x2

## 5 Wnioski

Klasyfikacje zbiorów w znacznej większości były w większości przypadków wręcz identyczne, różnice między poszczególnymi wynikami różniły się o  $< 0.01\%$ . Wybrany epsilon (tolerancja) nie miała wpływu wyniki, jedynie w klasyfikacji punktów leżących na prostej. W tym najbardziej wymagającym dokładności przypadku można dojść do wniosku, że epsilon rzędu  $10^{-10}$  będzie prawdopodobnie wystarczającą tolerancją w obliczeniach (co przy dokładności float64, która może maksymalnie zawierać 15-17 cyfr po przecinku, jest rezultatem całkiem racjonalnym).

Przy klasyfikacji punktów na prostej można zauważyć, że punkty sklasyfikowane błędnie w znacznej większości były tymi o większej odległości od środka układu współrzędnych. Jest to spowodowane tym, że klasyfikacja wymaga dużej dokładności, która jest tracona z powodu dużych wartości współrzędnych.

Epsilon równy zero nie ma racji bytu ze względu na tracone wartości poprzez ograniczoną pamięć komputera. Z tego powodu poprawność klasyfikacji punktów na prostej nie przekraczała 70%, a średni wynik dla wszystkich wyznaczników jest gorszy o co najmniej (!) 30%.

W klasyfikacji spośród czterech wyznaczników, tymi lepiej spełniającymi swoje zadanie były wyznaczniki własnoręcznie zaimplementowane, dodatkowo są one zauważalnie szybsze w obliczeniach. Jest to spowodowane przeznaczeniem funkcji bibliotecznych, które są w stanie obliczyć wyznacznik znacznie większych macierzy, przy których implementacja własnego wyznacznika będzie zbyt zasobochłonna.

Wyniki klasyfikacji przy zmniejszonej precyzji float32 były gorsze niż ich 64-bitowy odpowiednik, co było przewidywalne ze względu na ilość danych jaką mogą pomieścić. Ich wyniki w zbiorach 1,2,3 są identyczne, co jest spowodowane wykraczeniem epsilon poza zakres tych liczb (float32 może mieć maksymalnie 5-7 cyfr po przecinku).