Algorytmy geometryczne

Sprawozdanie 1 | Predykaty Geometryczne

Paweł Fornagiel | Informatyka rok II | Grupa 1

Data Wykonania: 14.10.2024 | Data Oddania: 31.10.2024

1. Opis ćwiczenia i realizacja

1.1. Informacje wstępne

Celem ćwiczenia jest implementacja podstawowych klasyfikacji położenia punktów względem prostej w zależności od ustalonej tolerancji uznania odległości za równą zeru, metody obliczania wyznacznika oraz użytej precyzji typów liczbowych dla 4 zdefiniowanych zbiorów punktów o równomiernym rozkładzie, tj.:

1. **Zestaw A** - 10^5 punktów na płaszczyźnie $[-1000, 1000]^2$
2. **Zestaw B** - 10^5 punktów na płaszczyźnie $\left[-10^{14}, 10^{14}\right]^2$
3. **Zestaw C** - 1000 punktów leżących na okręgu w układzie współrzędnych o środku $O = (0, 0)$ i promieniu $R = 100$
4. **Zestaw D** - 1000 punktów o współrzędnych niezależnych z przedziału $[-1000, 1000]$ leżących na prostej zdefiniowanej przez wektor (a, b), gdzie $a = [-1.0, 0.0]$, $b = [1.0, 0.1]$

1.2. Generowanie punktów

Równomierny rozkład punktów został zrealizowany za pomocą metody `random.uniform` biblioteki `numpy`, generującej liczby typu `float64` z zadanego zakresu domkniętego. Dodatkowo, każdy zestaw punktów został wygenerowany używając typów liczbowych `float64` i `float32` w celu analizy ich doboru wpływu na wyniki. Aby uzyskać powtarzalność wyników ustawiono wartość`random.seed` biblioteki `numpy` na stałą wartość.

Punkty położone na okręgu w *Zestawie C* zostały wygenerowane używając funkcji trygonometrycznych korzystając z parametrycznego równania okręgu.

```
1| angles = np.random.uniform(0, 2*np.pi, n)
2| x = R * np.cos(angles) + O[0]
3| y = R * np.sin(angles) + O[1]
4| tuples = [(x[i], y[i]) for i in range(n)]
```

Punkty położone na prostej w *Zestawie D* zostały wygenerowane używając równania parametrycznego prostej przechodzącej przez dwa punkty a i b w przestrzeni dwuwymiarowej.

```
1| a1,a2 = a
2| b1,b2 = b
2| v = (a1-b1,a2-b2)
3| x_normalised_lower_bound = (x_lower_bound-a[0])/abs(v[0])
4| x_normalised_upper_bound = (x_upper_bound-a[0])/abs(v[0])
5| points = np.random.uniform(x_normalised_lower_bound,x_normalised_upper_bound,n)
6| tuples = [(a1 + point * v[0], a2 + point * v[1]) for point in points]
```

1.3. Obliczanie wyznacznika

Kategoryzacja położenia punktów względem prostej była realizowana za pomocą wyznacznika macierzy zawierającej współrzędne punktów a i b należących do prostej oraz punktu c będącego przedmiotem kwalifikacji. Metody obliczania wyznaczników dla poszczególnych funkcji opisane są w Tabela 1

Nazwa funkcji	Metoda obliczenia
mat_det_3x3	Reguła Sarrusa dla $\det(a,b,c) = \begin{vmatrix} a_x & a_y & 1 \\ b_x & b_y & 1 \\ c_x & c_y & 1 \end{vmatrix}$
mat_det_2x2	Reguła Sarrusa dla $\det(a,b,c) = \begin{vmatrix} a_x - c_x & a_y - c_y \\ b_x - c_x & b_y - c_y \end{vmatrix}$
mat_det_3x3_lib	Wykorzystanie implementacji w bibliotece `numpy`
mat_det_2x2_lib	Wykorzystanie implementacji w bibliotece `numpy`

Tabela 1 : Metody obliczania wyznacznika

Kategoryzację punktów umożliwiała funkcja `categorize_points`, która dla kolekcji współrzędnych przechowywanych w obiekcie `DataFrame` biblioteki `pandas` stosuje opisane wyżej metody i zwraca obiekt `DataFrame`, w którym do każdego punktu przypisana jest cecha `category` przyjmująca wartości odpowiadające kwalifikacji punktu zależnie od wyniku.

1.4. Tolerancja zera

Wynik był uznawany za zero przyrównując jego wartość bezwzględną do zmiennej `epsilon` o następujących wartościach:

0	10^{-8}	$10^{-}10$	$10^{-}12$	$10^{-}14$

1.5. Uzyskiwanie danych i wizualizacja

Dane dla każdego z zestawów zostały wygenerowane dla wszystkich kombinacji następujących cech: metody obliczania wyznacznika, tolerancji zera, typu liczbowego. Łącznie podejście to dało 40 przypadków dla każdego z zestawów. Informacje o zastosowanych parametrach podczas generacji oraz ilości punktów, które przyjęły daną wartość `category`, zostały zapisane w plikach `CSV` w katalogu `data`. Nazwy plików odpowiadają nazwą poszczególnych zbiorów.

Do wizualizacji wygenerowanych danych posłużyła biblioteka `Seaborn`. Wszystkie wykresy zostały uzyskane używając metody `jointplot` z typem wizualizacji `scatter`, pozwalając dodatkowo na uzyskanie wykresów gęstości oraz histogramów rozmieszczenia punktów. Kwalifikacjom punktów na wykresach odpowiadają kolory:

- Zielony - oznaczenie na wykresach odpowiadające punktom zakwalifikowanym, jako leżące na prostej
- Niebieski - oznaczanie na wykresach odpowiadające punktom leżącym po lewej stronie prostej
- Pomarańczowy - oznaczenie na wykresach odpowiadające punktom leżącym po prawej stronie prostej

Do osi każdego z wykresów zostały dodane wykresy gęstości odzwierciedlający liczebność punktów na obejmowanym przedziale. Wykresy zostały zapisane w plikach `JPG` w katalogu `plots`. Nazwa każdego z plików odpowiada parametrom z jakimi został wygenerowany wykres.

2. Dane techniczne

Zadanie zostało przeprowadzone z użyciem narzędzi o następujących parametrach:

- Komputer HP EliteBook 840 G6:
  - System operacyjny: Windows 11 x64
  - Procesor Intel(R) Core(TM) i5-8365U CPU 1.60GHz 1.90 GHz
  - Pamięć RAM: 8GB
- Środowisko: Jupyter Notebook
- Język: Python 3.9.20
- Biblioteki języka: Numpy, Seaborn, Pandas, Matplotlib

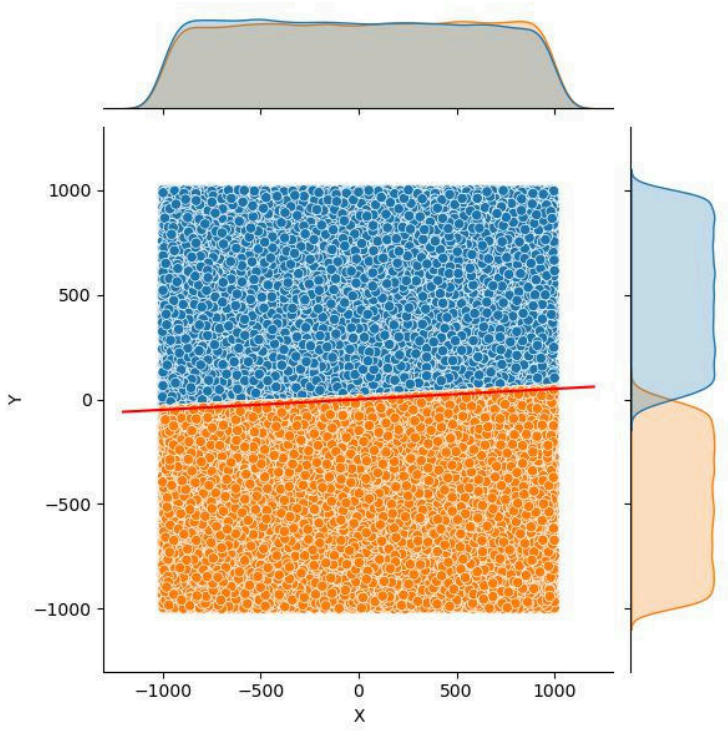
3. Analiza Wyników

3.1. Zestaw A

W *Zestawie A*, niezależnie od parametrów generowania danych, kategoryzacja punktów była identyczna. Liczba punktów w każdej z kategorii przedstawia Tabela 2 oraz Wykres 1.

Liczba punktów po lewej stronie prostej	Liczba punktów na prostej	Liczba punktów po prawej stronie prostej
50302	0	49698

Tabela 2 : Wyniki kategoryzacji punktów *Zestawu A*



Wykres 1 : A float32 1e-14 mat_det_3x3_lib
Przykładowy wykres rozkładu punktów *Zestawu A*

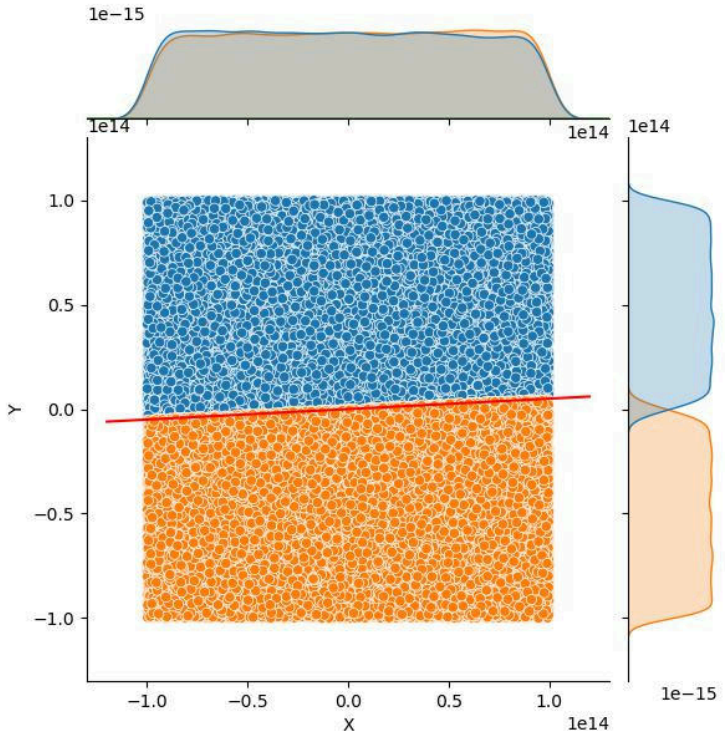
3.2. Zestaw B

3.2.1. Omówienie wyników

W *Zestawie B* precyzja typu zmiennoprzecinkowego nie miała wpływ na wyniki. W przeciwieństwie do *Zestawu A* metoda obliczania wyznacznika miała marginalny wpływ na liczbę punktów przydzielonych do odpowiednich kategorii. Wystąpiły również pojedyńcze punkty, które zostały zakwalifikowane jako leżące na prostej przy użyciu metody obliczania wyznacznika `mat_det_2x2` dla `epsilon` $\geq 10^{-14}$. Przez nieznaczne różnice w kwalifikacji punktów, różnice między wykresami nie są zauważalne. Przykładowe wyniki kategoryzacji są przedstawione w Tabela 3 oraz Wykres 2

Precyzja typu `float`	Epsilon	Metoda obliczania wyznacznika	Liczba punktów po lewej stronie prostej	Liczba punktów na prostej	Liczba punktów po prawej stronie prostej
float64	1e-14	mat_det_2x2	50300	8	49692
float64	1e-14	mat_det_2x2_lib	50302	0	49698
float64	1e-14	mat_det_3x3	50304	0	49696
float64	1e-14	mat_det_3x3_lib	50304	0	49696
float32	1e-14	mat_det_2x2	50299	11	49690
float32	1e-14	mat_det_2x2_lib	50302	0	49698
float32	1e-14	mat_det_3x3	50304	0	49696
float32	1e-14	mat_det_3x3_lib	50304	0	49696

Tabela 3 : Wyniki kategoryzacji *Zestawu B* dla `epsilon`$= 10^{-14}$



Wykres 2 : B float32 1e-14 mat_det_2x2
Przykładowy wykres rozkładu punktów *Zestawu B*

3.2.2. Analiza zakwalifikowanych punktów

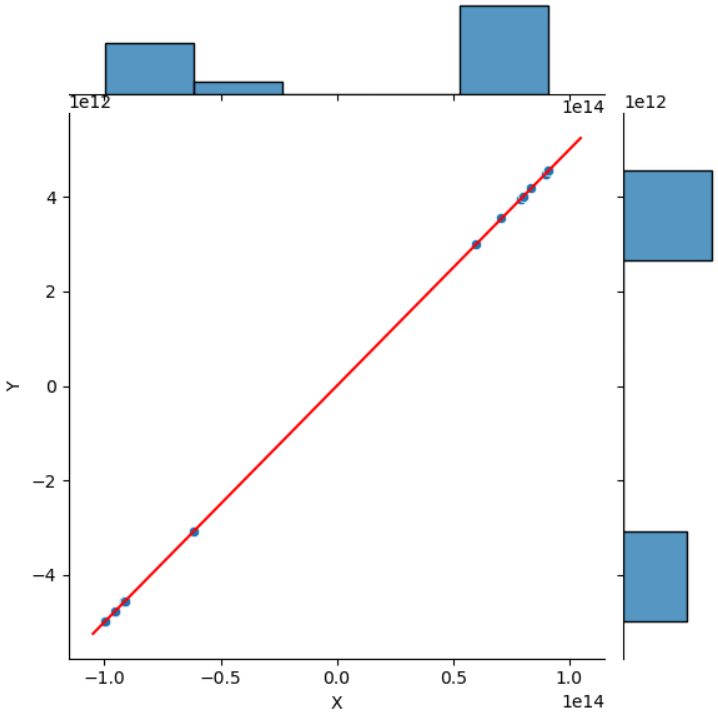
Punkty, które zostały uznane jako leżące na proste, zostały w każdym przypadku zostały zakwalifikowane używając metody obliczania wyznacznika `mat_det_2x2` dla wszystkich dokładności

`Eps`$\geq 10^{-14}$, różniąc się jedynie precyzją kwalifikacji. Punkty te zostały przedstawione w poniższej Tabela 4 oraz na wykresie Wykres 3.

Mimo, że punkty znajdują się w pobliżu prostej, ręczna weryfikacja ich przynależności wykazała, że ich odległość od prostej jest większa niż dowolna z przyjętych precyzji `Eps`. Wskazuje to prawdopodobnie na mniejszą precyzję kwalifikacji punktów używając metody `mat_det_2x2` względem pozostałych metod.

Współrzędna X	Współrzędna Y	Precyzja typu `float`
−99550446485504.00	−4973350879232.00	float32
−95395370301686.34	−4760374840241.67	float64, float32
−91521813667435.31	−4556439929295.75	float64, float32
−91016212250624.00	−4560953802752.00	float64, float32
−61373260235653.51	−3070303412621.33	float32
59903653111930.81	2989497329631.28	float64, float32
70668089830303.72	3548766641817.48	float64, float32
79277612364346.72	3954381882519.00	float64, float32
80103709691155.47	4002107667469.69	float64, float32
83403429491424.22	4202409245064.83	float64
89794268889088.00	4495321858048.00	float32
91209653551104.00	4556835520512.00	float32

Tabela 4 : Punkty *Zestawu B* zakwalifikowane jako leżące na prostej



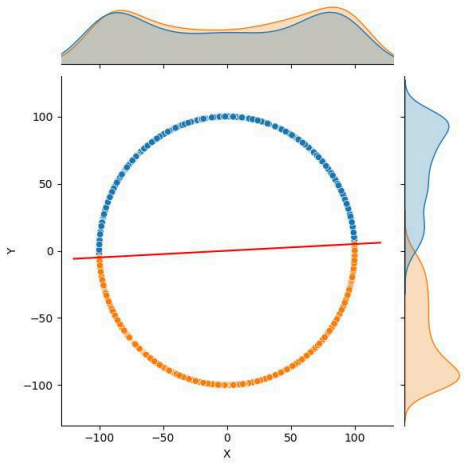
Wykres 3 : Wizualizacja punktów z Tabela 4

3.3. Zestaw C

Podobnie jak w *Zestawie A*, w *Zestawie C* niezależnie od parametrów generowania danych, kategoryzacja punktów była identyczna. Liczba punktów w każdej z kategorii przedstawia Tabela 5 oraz Wykres 4.

Liczba punktów po lewej stronie prostej	Liczba punktów na prostej	Liczba punktów po prawej stronie prostej
507	0	493

Tabela 5 : Wyniki kategoryzacji punktów *Zestawu C*



Wykres 4 : C float64 1e-12 mat_det_3x3

3.4. Zestaw D

W *Zestawie D* występuje duża rozbieżność w kwalifikacji punktów dla różnych doborów parametrów testowych, mimo że teoretycznie każdy z punktów został wygenerowany na prostej. Wyniki przedstawione są w Tabela 6.

Własne implementacje metody obliczania wyznacznika (`mat_det_2x2`, `mat_det_3x3`) w znacznie większej liczbie kategoryzowały punkty jako leżące na prostej w porównaniu z implementacjami bibliotecznymi (`mat_det_2x2_lib`, `mat_det_3x3_lib`), co widoczne jest szczególnie dla `epsilon=0`. Przykładem są Wykres 9 oraz Wykres 10.

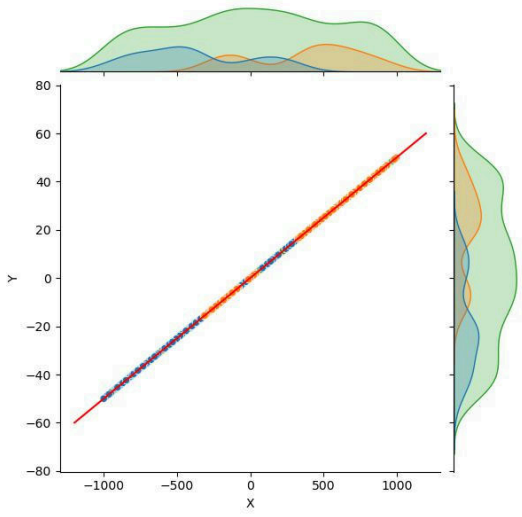
Precyzja liczby zmiennoprzecinkowej w sposób znaczący wpływała na grupowanie danych. Typ `float64` w znacząco większej liczbie uznawał przynależność punktu do prostej niż `float32` w każdym z przypadków testowych. Przykładem są Wykres 5 i Wykres 6

Tolerancja zera `epsilon` miała wpływ na wyniki w następujący sposób:

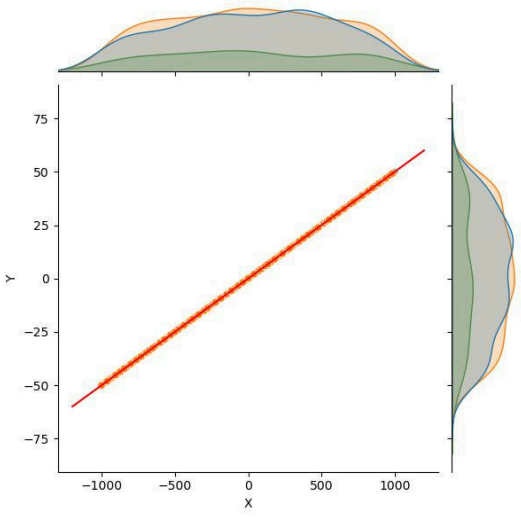
- `epsilon` $= 0$, w żadnym z testowanych przypadków nie uznano, że wszystkie z punktów znajdują się na prostej. Pozytywnego grupowania dokonywały jedynie metody implementacji obliczania wyznacznika `mat_det_2x2` oraz `met_det_3x3`.
- `epsilon`$= 10^{-14}$, jedyną metodą uznającą przynależność wszystkich punktów do prostej była `met_det_3x3` (precyzja `float64`). Wyznaczniki `3x3` zdecydowanie częściej przydzielały punktom kategorię `Middle`.
- `epsilon`$= 10^{-12}$, obydwie metody obliczania wyznacznika `3x3` przypisały przynależność do prostej dla całego zbioru (precyzja `float64`). Porównanie wyznacznika `3x3` i `2x2` widoczne jest na Wykres 7 oraz Wykres 8.
- `epsilon`$= 10^{-10}$ i `epsilon`$=10^{-8}$, dla obydwu przypadków wyniki są zbliżone, każdy z wyznaczników w precyzji `float64` kwalifikował cały zbiór jako leżący na prostej.

Precyzja typu `float`	Epsilon	Metoda obliczania wyznacznika	Liczba punktów po lewej stronie prostej	Liczba punktów na prostej	Liczba punktów po prawej stronie prostej
float64	1e-08	mat_det_2x2	0	1000	0
float64	1e-08	mat_det_2x2_lib	0	1000	0
float64	1e-08	mat_det_3x3	0	1000	0
float64	1e-08	mat_det_3x3_lib	0	1000	0
float64	1e-10	mat_det_2x2	0	1000	0
float64	1e-10	mat_det_2x2_lib	0	1000	0
float64	1e-10	mat_det_3x3	0	1000	0
float64	1e-10	mat_det_3x3_lib	0	1000	0
float64	1e-12	mat_det_2x2	88	828	84
float64	1e-12	mat_det_2x2_lib	128	744	128
float64	1e-12	mat_det_3x3	0	1000	0
float64	1e-12	mat_det_3x3_lib	0	1000	0
float64	1e-14	mat_det_2x2	146	719	135
float64	1e-14	mat_det_2x2_lib	452	98	450
float64	1e-14	mat_det_3x3	0	1000	0
float64	1e-14	mat_det_3x3_lib	34	932	34
float64	0.0	mat_det_2x2	154	704	142
float64	0.0	mat_det_2x2_lib	494	0	506
float64	0.0	mat_det_3x3	314	306	380
float64	0.0	mat_det_3x3_lib	476	0	524
float32	1e-08	mat_det_2x2	420	188	392
float32	1e-08	mat_det_2x2_lib	420	188	392
float32	1e-08	mat_det_3x3	420	188	392
float32	1e-08	mat_det_3x3_lib	420	188	392
float32	1e-10	mat_det_2x2	422	185	393
float32	1e-10	mat_det_2x2_lib	422	185	393
float32	1e-10	mat_det_3x3	422	185	393
float32	1e-10	mat_det_3x3_lib	422	185	393
float32	1e-12	mat_det_2x2	422	185	393
float32	1e-12	mat_det_2x2_lib	438	139	423
float32	1e-12	mat_det_3x3	422	185	393
float32	1e-12	mat_det_3x3_lib	422	185	393
float32	1e-14	mat_det_2x2	431	165	404
float32	1e-14	mat_det_2x2_lib	499	21	480
float32	1e-14	mat_det_3x3	422	185	393
float32	1e-14	mat_det_3x3_lib	428	171	401
float32	0.0	mat_det_2x2	431	164	405
float32	0.0	mat_det_2x2_lib	507	0	493
float32	0.0	mat_det_3x3	485	44	471
float32	0.0	mat_det_3x3_lib	509	0	491

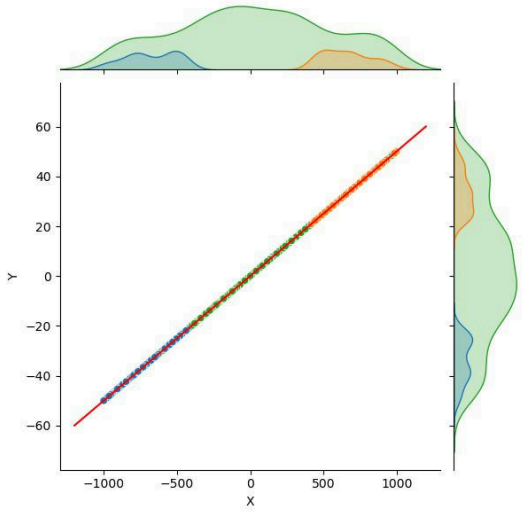
Tabela 6 : Wyniki kategoryzacji *Zestawu D*



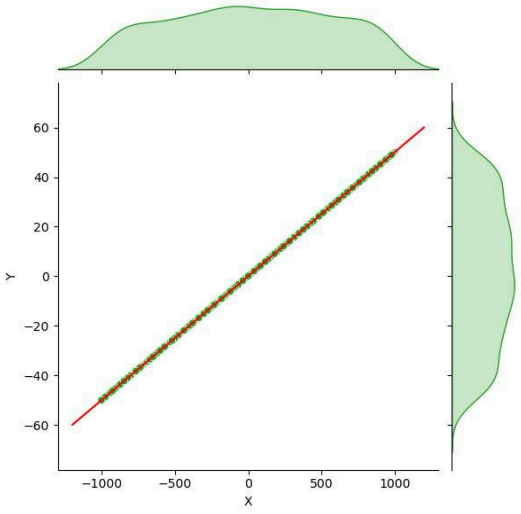
Wykres 5 : D float64 0 mat_det_2x2



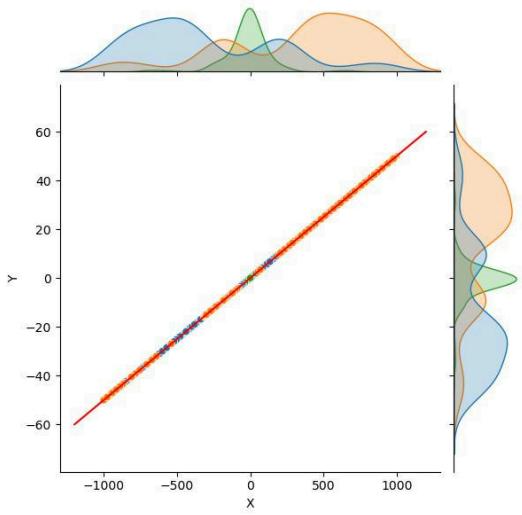
Wykres 6 : D float32 0 mat_det_2x2



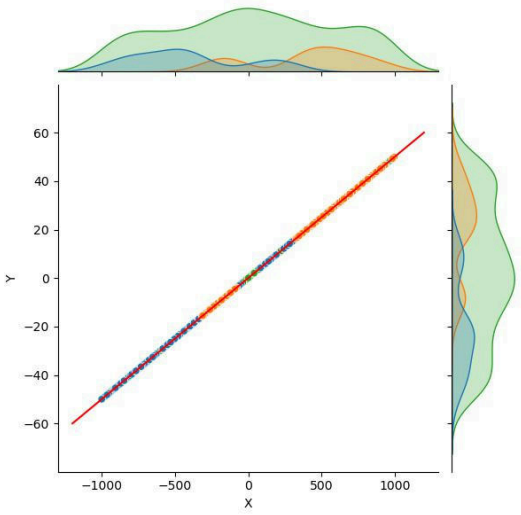
Wykres 7 : D float64 1e-12 mat_det_2x2



Wykres 8 : D float64 1e-12 mat_det_3x3



Wykres 9 : D float64 1e-14 mat_det_2x2_lib



Wykres 10 : D float64 1e-14 mat_det_2x2

4. Wnioski

Na podstawie przeprowadzonego eksperymentu można wyciągnąć następujące wnioski:

- **Stabilność klasyfikacji dla zestawów losowych**: Dla zestawów A i C, rozkład klasyfikacji punktów pozostawał niezmienny niezależnie od użytych parametrów. Oznacza to, że generowanie punktów z losowego rozkładu w płaszczyźnie jest mniej podatne na wpływ zmiennych, takich jak precyzja liczbowa czy tolerancja zera.
- **Wpływ parametrów na zestaw punktów na prostej**: W przypadku punktów leżących na prostej (*Zestaw D*) zauważalny był znaczny wpływ doboru metod wyznaczania wyznaczników oraz tolerancji zera na końcowe wyniki klasyfikacji. Dodatkowo typ danych (`float64`, `float32`) istotnie wpływał na rezultaty, zwłaszcza przy niższych wartościach tolerancji, co potwierdza wyraźną zależność między precyzją obliczeń a dokładnością klasyfikacji.
- **Efektywność bibliotecznych implementacji wyznacznika**: Zaimplementowane metody wyznacznika (szczególnie mat_det_2x2 i mat_det_3x3) lepiej klasyfikowały punkty na prostej niż ich biblioteczne odpowiedniki dla mniejszych wartości epsilon w pryzpadku *Zestawu D*, lecz nie były precyzyjne w przypadku pojedyńczych punktów *Zestawu B*. Biblioteczne implementacje okazały się mniej precyzyjne w rozpoznawaniu położenia punktów dokładnie na prostej w *Zestawie D* co sugeruje, że zależnie od sytuacji manualnie dopasowane algorytmy mogą oferować lepszą dokładność niż gotowe funkcje i vice versa.