Algorytmy Geometryczne - Laboratorium 1

Badanie dokładności predykatów geometrycznych

Imię i nazwisko: Wermiński Gabriel **Data wykonania:** 19.10.2025

Grupa: 6

1. Środowisko obliczeniowe

Eksperymenty zostały przeprowadzone w następującej konfiguracji:

- **Procesor:** CPU 12th Gen Intel(R) Core(TM) i5-1235U, 1.30 GHz
- **Pamięć RAM:** 16.0 GB, 3200 MT/s
- **System operacyjny:** Microsoft Windows 11 Home
- **Środowisko programistyczne:** Jupyter Notebook, Pycharm
- **Język programowania:** Python 3.13.5
- **Biblioteki:** numpy, pandas, matplotlib, pathlib

2. Wprowadzenie i cel

Celem laboratorium jest zbadanie dokładności podstawowych operacji geometrycznych, w szczególności predykatu określającego położenie punktu względem prostej. Analiza skupia się na wpływie różnych czynników na poprawność klasyfikacji punktów:

- metody obliczania wyznacznika (implementacja własna vs biblioteczna),
- rodzaju wyznacznika (macierz 2×2 vs 3×3),
- precyzji reprezentacji liczb zmiennoprzecinkowych,
- wartości tolerancji dla zera.

Predykat geometryczny oparty jest na obliczeniu wyznacznika macierzy. Dla trzech punktów a, b i c wartość wyznacznika określa, po której stronie prostej przechodzącej przez punkty a i b znajduje się punkt c.

3. Podstawy teoretyczne

3.1. Orientacja punktu względem prostej

Położenie punktu $c = \left(c_x, c_y\right)$ względem prostej wyznaczonej przez punkty $a = \left(a_x, a_y\right)$ i $b = \left(b_x, b_y\right)$ można określić obliczając wyznacznik:

Macierz 2×2:

$$\det_{2x2(a,b,c)} = \begin{vmatrix} a_x - c_x & a_y - c_y \\ b_x - c_x & b_y - c_y \end{vmatrix}$$

Macierz 3×3:

$$\det_{3x3(a,b,c)} = \begin{vmatrix} a_x & a_y & 1 \\ b_x & b_y & 1 \\ c_x & c_y & 1 \end{vmatrix}$$

Interpretacja wyniku:
- $\det > 0 \rightarrow$ punkt leży po lewej stronie prostej,
- $\det = 0 \rightarrow$ punkt leży na prostej,
- $\det < 0 \rightarrow$ punkt leży po prawej stronie prostej.

W praktyce, ze względu na błędy zaokrągleń w arytmetyce zmiennoprzecinkowej, wprowadza się tolerancję ε. Punkt uznaje się za leżący na prostej, gdy $|\det| \leq \varepsilon$.

3.2. Problematyka numeryczna

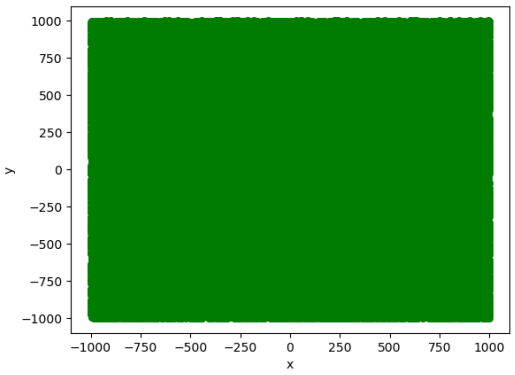
Reprezentacja liczb zmiennoprzecinkowych w komputerze jest ograniczona, co prowadzi do błędów zaokrągleń. Dla dużych wartości liczbowych dokładność względna maleje, ponieważ stała liczba bitów mantysy musi reprezentować coraz większe odległości między kolejnymi liczbami. Operacje na dużych, podobnych liczbach (np. odejmowanie) są szczególnie podatne na utratę cyfr znaczących.

4. Realizacja eksperymentu

4.1. Generowanie zbiorów testowych

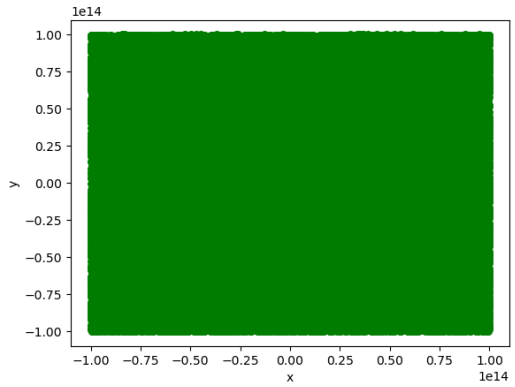
Przygotowano cztery zbiory punktów o różnych charakterystykach:

Zbiór A: Zawiera 10^5 punktów losowo rozłożonych w kwadracie o współrzędnych z przedziału $[-1000, 1000]^2$. Jest to zbiór o umiarkowanym zakresie wartości.



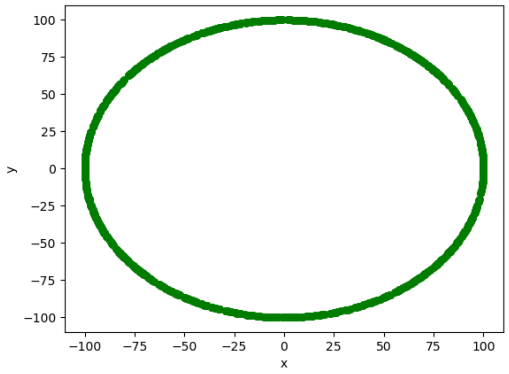
Rysunek 1: Wizualizacja zbioru A

Zbiór B: Obejmuje 10^5 punktów w znacznie większym obszarze: $\left[-10^{14}, 10^{14}\right]^2$. Duże wartości współrzędnych testują stabilność numeryczną algorytmów.



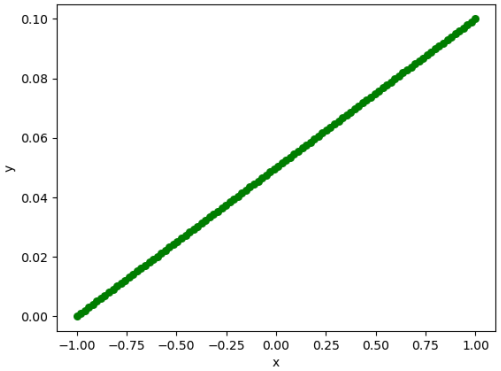
Rysunek 2: Wizualizacja zbioru B

Zbiór C: Składa się z 1000 punktów równomiernie rozłożonych na okręgu o środku w początku układu współrzędnych i promieniu $R = 100$. Punkty generowano w układzie biegunowym, a następnie przekształcano do kartezjańskiego.



Rysunek 3: Wizualizacja zbioru C

Zbiór D: Zawiera 1000 punktów leżących na prostej określonej przez punkty $a = (-1.0, 0.0)$ i $b = (1.0, 0.1)$. Punkty generowano parametrycznie jako kombinacje liniowe końców odcinka, co pozwala zbadać zachowanie predykatu w przypadku idealnym.



Rysunek 4: Wizualizacja zbioru D

4.2. Implementacja wyznaczników

Zaimplementowano cztery warianty obliczania wyznacznika:

Własna implementacja 2×2: Bezpośrednie obliczenie według wzoru $(a_x - c_x)(b_y - c_y) - (a_y - c_y)(b_x - c_x)$.

Biblioteczna implementacja 2×2: Wykorzystanie funkcji `numpy.linalg.det()` dla macierzy 2×2.

Własna implementacja 3×3: Rozwinięcie Laplace'a względem pierwszego wiersza.

Biblioteczna implementacja 3×3: Funkcja `numpy.linalg.det()` dla macierzy 3×3.

4.3. Parametry badania

Klasyfikację punktów przeprowadzono dla następujących konfiguracji:

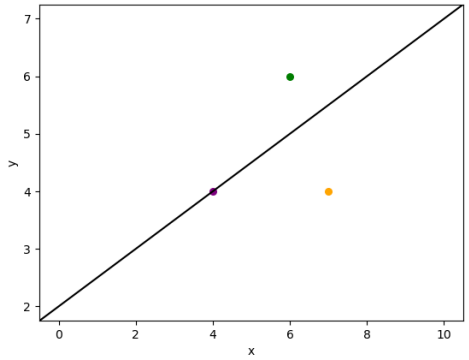
- **Precyzja zmiennych:** `float64` (podwójna) i `float32` (pojedyncza),
- **Tolerancja dla zera:** $\varepsilon \in \{0, 10^{-14}, 10^{-12}, 10^{-10}, 10^{-8}\}$,
- **Metody obliczania:** wszystkie cztery warianty wyznacznika.

Dla każdej konfiguracji rejestrowano liczbę punktów sklasyfikowanych jako leżące po lewej stronie, na prostej oraz po prawej stronie.

4.4. Klasyfikacja punktów względem prostej

Na wykresie przedstawiono przykład klasyfikacji punktów względem prostej (czarna linia):

- Punkty po lewej stronie prostej są oznaczone kolorem zielonym,
- Punkty leżące dokładnie na prostej – kolorem fioletowym,
- Punkty po prawej stronie prostej – kolorem pomarańczowym.



Rysunek 5: klasyfikacja punktów

5. Wyniki i analiza

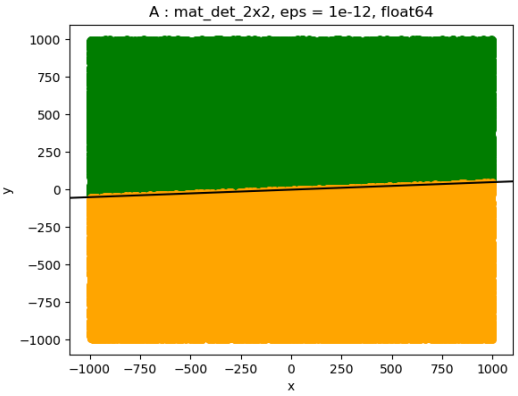
5.1. Zbiór A – zakres umiarkowany (float64)

Metoda	Lewo	Na prostej	Prawo
Wszystkie	50261	0	49739

Tabela 1: Klasyfikacja punktów zbioru A

Dla zbioru A wszystkie konfiguracje (precyzja, tolerancja ε, metoda wyznacznika) dały identyczne wyniki. Rozkład jest niemal równomierny względem prostej. Żaden punkt nie został sklasyfikowany jako leżący na prostej, co jest zgodne z oczekiwaniami.

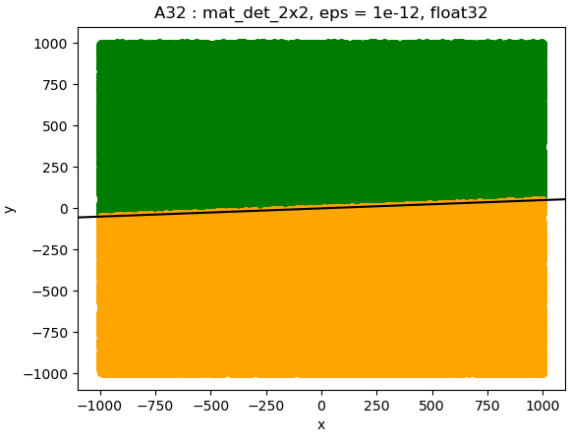
Wniosek: Dla współrzędnych z przedziału $[-1000, 1000]$ wszystkie metody są wystarczająco dokładne.



Rysunek 6: Rozkład punktów zbioru A

5.2. Zbiór A32 – zakres umiarkowany (float32)

Wyniki dla float32 są identyczne jak dla float64: 50261 punktów po lewej, 0 na prostej, 49739 po prawej. Mniejsza precyzja nie wpływa na klasyfikację przy tym zakresie wartości, co potwierdza wystarczającą dokładność float32 dla umiarkowanych współrzędnych.



Rysunek 7: Rozkład punktów zbioru A32

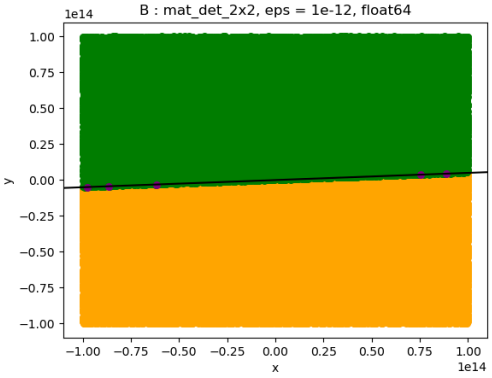
5.3. Zbiór B – duże wartości (float64)

ε	Metoda	Lewo	Na prostej	Prawo
0	mat_det_2x2	50236	2	49762
0	mat_det_3x3	50238	0	49762
10^{-8} do 10^{-14}	mat_det_2x2	50236	2	49762
10^{-8} do 10^{-14}	mat_det_3x3	50238	0	49762

Tabela 2: Klasyfikacja dla zbioru B (obie impl. 3×3 i biblioteczne)

Obserwacje: Funkcja `mat_det_2x2` jako jedyna klasyfikuje 2 punkty jako leżące „na linii" przy $\varepsilon = 0$. Wyznaczniki 3×3 nie wskazują żadnych punktów na prostej. Wyniki są stabilne dla różnych wartości ε od 10^{-8} do 10^{-14}.

Interpretacja: Przy wartościach rzędu 10^{14} float64 zapewnia wystarczającą precyzję. Niewielkie różnice między metodami wynikają z odmiennej propagacji błędów zaokrągleń.

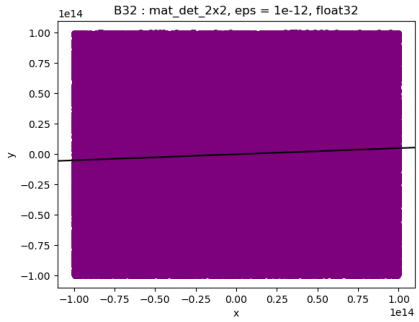


Rysunek 8: Rozkład punktów zbioru B

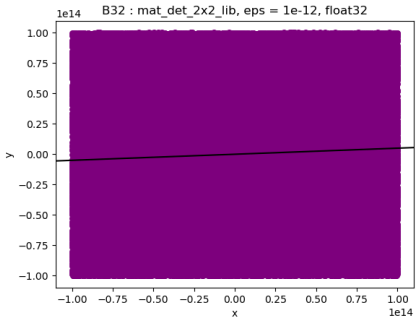
5.4. Zbiór B32 – duże wartości (float32)

ε	Metoda	Lewo	Na prostej	Prawo
0	mat_det_2x2	0	100000	0
0	mat_det_2x2_lib	8299	83487	8214
0	mat_det_3x3	50238	0	49762
10^{-8} do 10^{-14}	mat_det_2x2	0	100000	0
10^{-8} do 10^{-14}	mat_det_2x2_lib	8299	83487	8214
10^{-8} do 10^{-14}	mat_det_3x3	50238	0	49762

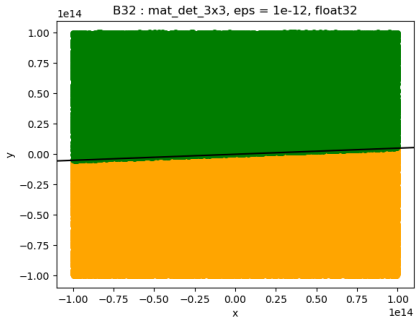
Tabela 3: Klasyfikacja dla zbioru B32 (– obie impl. 3×3)



Rysunek 9: B32: det_2x2



Rysunek 10: B32: det_2x2_lib



Rysunek 11: B32: det_3x3

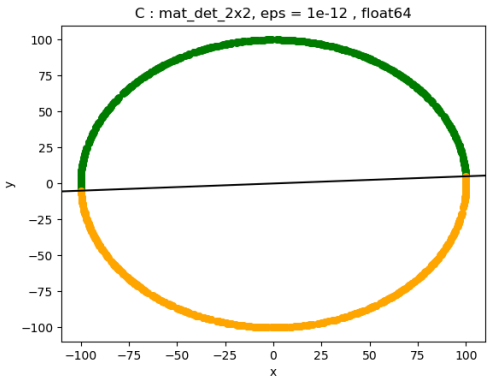
Obserwacje: Własna implementacja `mat_det_2x2` całkowicie zawiodła – wszystkie 100000 punktów sklasyfikowała jako leżące na prostej. Biblioteczna `mat_det_2x2_lib` działała lepiej, ale wciąż błędnie sklasyfikowała 83% punktów. Tylko wyznaczniki 3×3 zachowują poprawność, dając wyniki identyczne jak dla float64.

Interpretacja: Float32 (7 cyfr znaczących) jest całkowicie niewystarczający dla wartości 10^{14}. Odejmowanie dużych, podobnych liczb prowadzi do katastrofalnej utraty precyzji. Wyznaczniki 3×3, pomimo większej liczby operacji, są numerycznie stabilniejsze.

5.5. Zbiór C – punkty na okręgu (float64)

Wszystkie konfiguracje dały identyczne wyniki: 516 punktów po lewej i 484 po prawej stronie prostej. Żaden punkt nie został sklasyfikowany jako leżący na prostej.

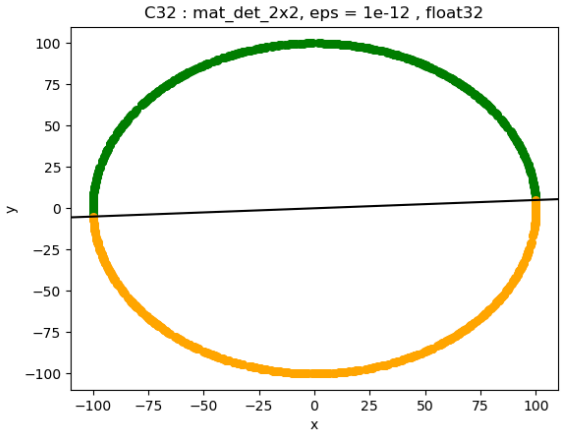
Interpretacja: Równomierny rozkład na okręgu z zakresem $|x|, |y| \leq 100$ zapewnia pełną stabilność obliczeń niezależnie od metody i precyzji.



Rysunek 12: Rozkład punktów zbioru C

5.6. Zbiór C32 – punkty na okręgu (float32)

Wyniki identyczne jak dla C (float64): 516 po lewej, 0 na prostej, 484 po prawej. Zmniejszona precyzja nie wpływa na klasyfikację przy tym zakresie współrzędnych, co potwierdza stabilność algorytmów dla punktów na okręgu.

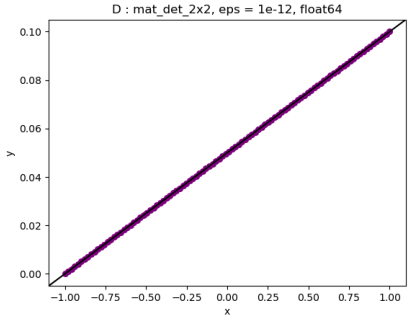


Rysunek 13: Rozkład punktów zbioru C32

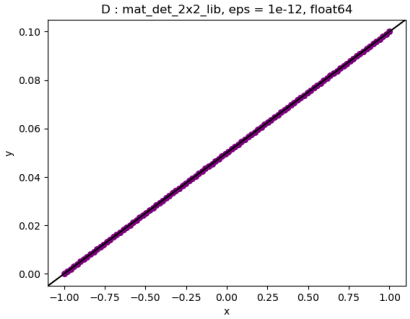
5.7. Zbiór D – punkty na prostej (float64)

ε	**Metoda**	**Lewo**	**Na prostej**	**Prawo**
0	mat_det_2x2	175	650	175
0	mat_det_2x2_lib	184	631	185
0	mat_det_3x3	106	788	106
0	mat_det_3x3_lib	406	188	406
10^{-14}	mat_det_2x2	139	722	139
10^{-14}	mat_det_2x2_lib	150	700	150
10^{-14}	mat_det_3x3	0	1000	0
10^{-14}	mat_det_3x3_lib	54	892	54
10^{-12} do 10^{-8}	wszystkie	0	1000	0

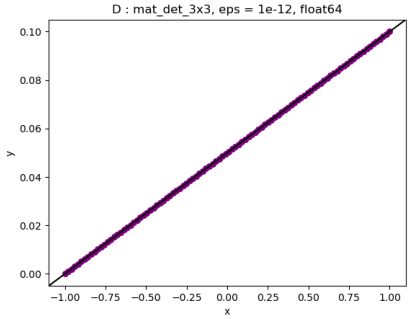
Tabela 4: Wybrane wyniki dla zbioru D



Rysunek 14: D: det_2x2, $\varepsilon = 10^{-12}$



Rysunek 15: D: det_2x2_lib, $\varepsilon = 10^{-12}$



Rysunek 16: D: det_3x3, $\varepsilon = 10^{-12}$

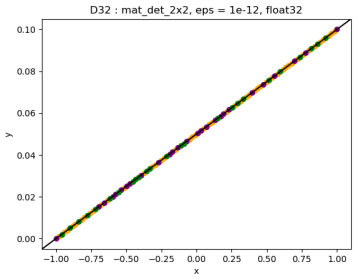
Obserwacje: Dla $\varepsilon = 10^{-12}$ wszystkie metody osiągają pełną dokładność (1000/1000 punktów na prostej), co widać na powyższych wykresach – wszystkie punkty są zaznaczone kolorem fioletowym (na prostej). Przy mniejszych wartościach ε (10^{-14}) tylko `mat_det_3x3` utrzymuje 100% dokładność. Bez tolerancji ($\varepsilon = 0$) nawet najlepsza metoda klasyfikuje tylko 78,8% poprawnie.

Interpretacja: Punkty generowane parametrycznie powinny teoretycznie leżeć dokładnie na prostej, ale błędy zaokrągleń podczas generacji i obliczeń sprawiają, że wyznacznik jest bliski, ale nie równy zeru. Wartość $\varepsilon = 10^{-12}$ stanowi optymalny kompromis zapewniający pełną dokładność.

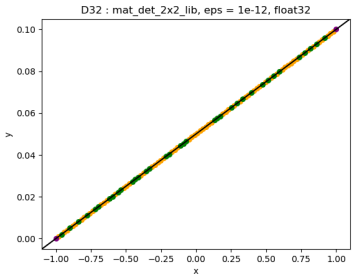
5.8. Zbiór D32 – punkty na prostej (float32)

ε	**Metoda**	**Lewo**	**Na prostej**	**Prawo**
0 do 10^{-10}	mat_det_2x2	172	656	172
0 do 10^{-10}	mat_det_2x2_lib	500	0	500
0 do 10^{-10}	mat_det_3x3	233	534	233
0 do 10^{-10}	mat_det_3x3_lib	469	62	469
10^{-12} do 10^{-8}	mat_det_2x2	172	656	172
10^{-12} do 10^{-8}	mat_det_2x2_lib	500	0	500
10^{-12} do 10^{-8}	mat_det_3x3	233	534	233
10^{-12} do 10^{-8}	mat_det_3x3_lib	469	62	469

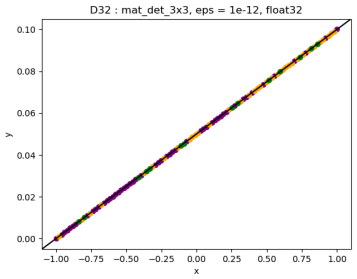
Tabela 5: Wyniki dla zbioru D32



Rysunek 17: D32: det_2x2



Rysunek 18: D32: det_2x2_lib



Rysunek 19: D32: det_3x3

Obserwacje: Dla float32 żadna metoda nie osiąga pełnej dokładności przy $\varepsilon = 10^{-12}$. Najlepsza (`mat_det_2x2`) klasyfikuje 65,6% poprawnie. Biblioteczna `mat_det_2x2_lib` w ogóle nie wykrywa punktów na prostej (0%). Wyznacznik 3×3 osiąga 53,4% dokładności. Wyniki są stabilne dla różnych ε, co sugeruje, że problem leży w ograniczonej precyzji, a nie w tolerancji.

Interpretacja: Float32 jest niewystarczający nawet dla umiarkowanego zakresu współrzędnych, gdy wymagana jest wysoka dokładność klasyfikacji punktów leżących na prostej. Własne implementacje radzą sobie lepiej niż biblioteczne, ale wciąż daleko od idealnych wyników float64.

6. Podsumowanie i wnioski

Na podstawie przeprowadzonych eksperymentów dotyczących klasyfikacji punktów względem prostej przy użyciu różnych metod obliczania wyznaczników można sformułować następujące wnioski:

1. **Zakres wartości współrzędnych ma krytyczne znaczenie dla stabilności numerycznej.** Dla zbiorów A i C (współrzędne rzędu setek lub tysięcy) wszystkie badane metody dały identyczne i poprawne wyniki, niezależnie od precyzji i tolerancji. Problem pojawia się dopiero przy wartościach ekstremalnych (zbiór B: 10^{14}).

2. **Float32 jest niewystarczający dla dużych wartości liczbowych.** W zbiorze B32 własna implementacja wyznacznika 2×2 całkowicie zawiodła, klasyfikując wszystkie 100000 punktów jako leżące na prostej. Biblioteczna implementacja 2×2 błędnie sklasyfikowała 83% punktów. Tylko wyznaczniki 3×3 zachowały poprawność również dla float32.

3. **Wyznaczniki 3×3 są bardziej stabilne numerycznie niż 2×2,** szczególnie w ekstremalnych warunkach. Mimo większej liczby operacji arytmetycznych, ich struktura zapewnia lepszą odporność na błędy zaokrągleń przy dużych wartościach współrzędnych.

4. **Tolerancja $\varepsilon = 10^{-12}$ jest optymalnym kompromisem dla float64.** Dla zbioru D (punkty na prostej) wartość ta zapewniła 100% dokładność klasyfikacji dla wszystkich metod. Mniejsze wartości (10^{-14}) powodowały błędne klasyfikacje w niektórych metodach, a brak tolerancji ($\varepsilon = 0$) prowadził do masowych błędów nawet w najlepszych implementacjach.

5. **Własne implementacje mogą przewyższać funkcje biblioteczne w specyficznych przypadkach.** Dla zbioru D32 własna implementacja det_2x2 osiągnęła 65,6% dokładności, podczas gdy biblioteczna w ogóle nie wykryła punktów na prostej (0%). Różnice wynikają prawdopodobnie z odmiennego porządku operacji arytmetycznych.

6. **Precyzja float64 jest niezbędna dla aplikacji wymagających wysokiej dokładności.** Zbiory A32 i C32 pokazały, że float32 wystarcza dla prostych zadań klasyfikacyjnych, ale zbiory B32 i D32 udowodniły jego całkowitą nieadekwatność w wymagających scenariuszach.

7. **Problem klasyfikacji punktów leżących dokładnie na prostej jest szczególnie trudny numerycznie.** Nawet punkty wygenerowane parametrycznie jako kombinacje liniowe (zbiór D) nie dają wyznacznika równego dokładnie zero ze względu na błędy zaokrągleń. Właściwy dobór tolerancji jest kluczowy.

8. **Wyniki są stabilne względem zmiany tolerancji w szerokim zakresie** (10^{-8} do 10^{-14}) dla zbiorów o dobrych właściwościach numerycznych (A, B, C). Wrażliwość na ε pojawia się tylko dla zbioru D, gdzie punkty faktycznie leżą blisko prostej.

9. **Różnice między implementacjami własną i biblioteczną są niewielkie dla float64** w większości przypadków (np. zbiór B: 2 punkty różnicy), ale stają się krytyczne dla float32 i trudnych przypadków testowych.

10. **Wizualizacja graficzna potwierdza wyniki ilościowe.** Obrazy dla $\varepsilon = 10^{-12}$ wyraźnie pokazują poprawność klasyfikacji (np. wszystkie punkty fioletowe dla zbioru D w float64) oraz dramatyczne różnice między metodami (np. zbiór B32).

6.1. Rekomendacje praktyczne

Dla zastosowań praktycznych w algorytmach geometrycznych zaleca się:

- Używanie **float64** jako standardu, szczególnie gdy zakres wartości współrzędnych jest duży lub nieznany z góry.
- Wybór **wyznacznika 3×3** jako uniwersalnej metody zapewniającej najlepszą stabilność numeryczną.
- Ustawienie **tolerancji $\varepsilon = 10^{-12}$** dla float64 jako rozsądnego kompromisu między dokładnością a odpornością na błędy.
- **Testowanie algorytmów** na różnych zbiorach danych, w tym przypadkach ekstremalnych (duże wartości, punkty blisko prostej).
- Jeśli wydajność jest krytyczna i współrzędne są niewielkie (rzędu tysięcy), można rozważyć **wyznacznik 2×2 z float64**, który będzie szybszy przy zachowaniu dokładności.
- **Unikanie float32** w aplikacjach wymagających wysokiej precyzji geometrycznej lub operujących na dużym zakresie wartości.