Jacek Łoboda gr 6,

Pon. godz 16:45

Algorytmy geometryczne, ćwiczenie 1

1. Dane techniczne

- Procesor: Intel Core i7-7700HQ 2.80GHz
- RAM: 16GB 2133MHz
- System operacyjny: Windows 10 Home x64
- Środowisko: Visual Studio Code
- Język: Python 3.13.5

Użyto następujących bibliotek: numpy, pandas, matplotlib, random. Do graficznej wizualizacji wykorzystano narzędzie koła BIT.

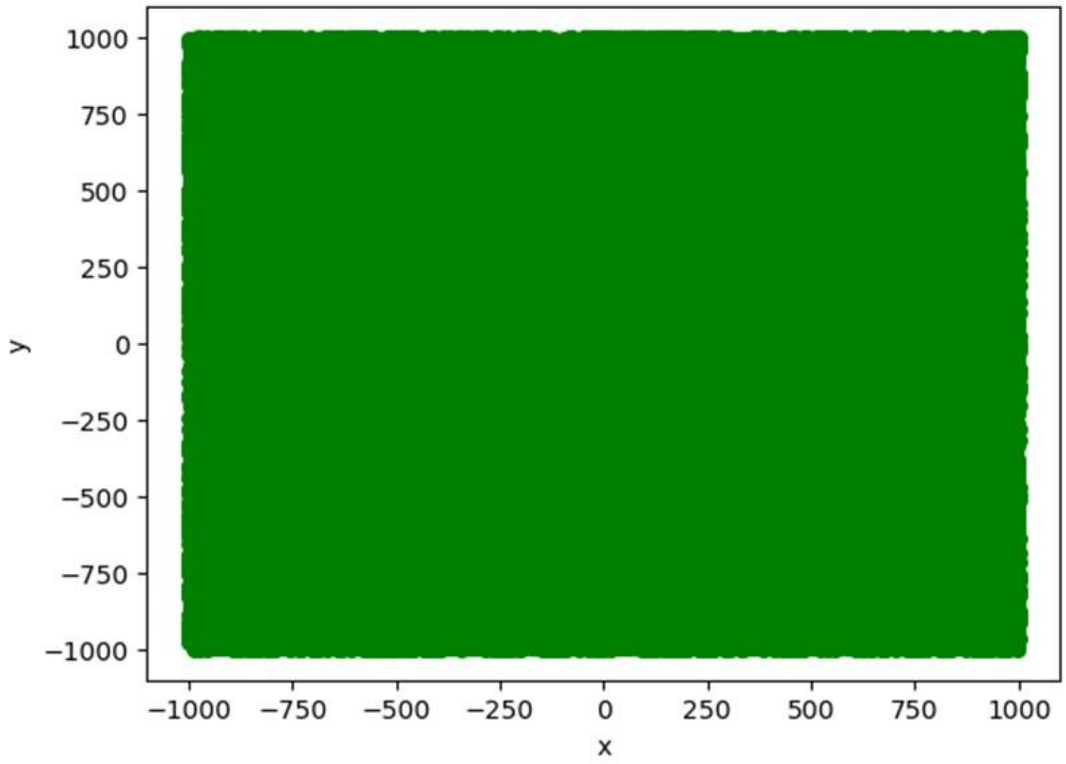
2. Cel ćwiczenia

Implementacja podstawowych predykatów geometrycznych takich jak położenie punktu względem prostej, przeprowadzenie testów, wizualizacja i opracowanie wyników.

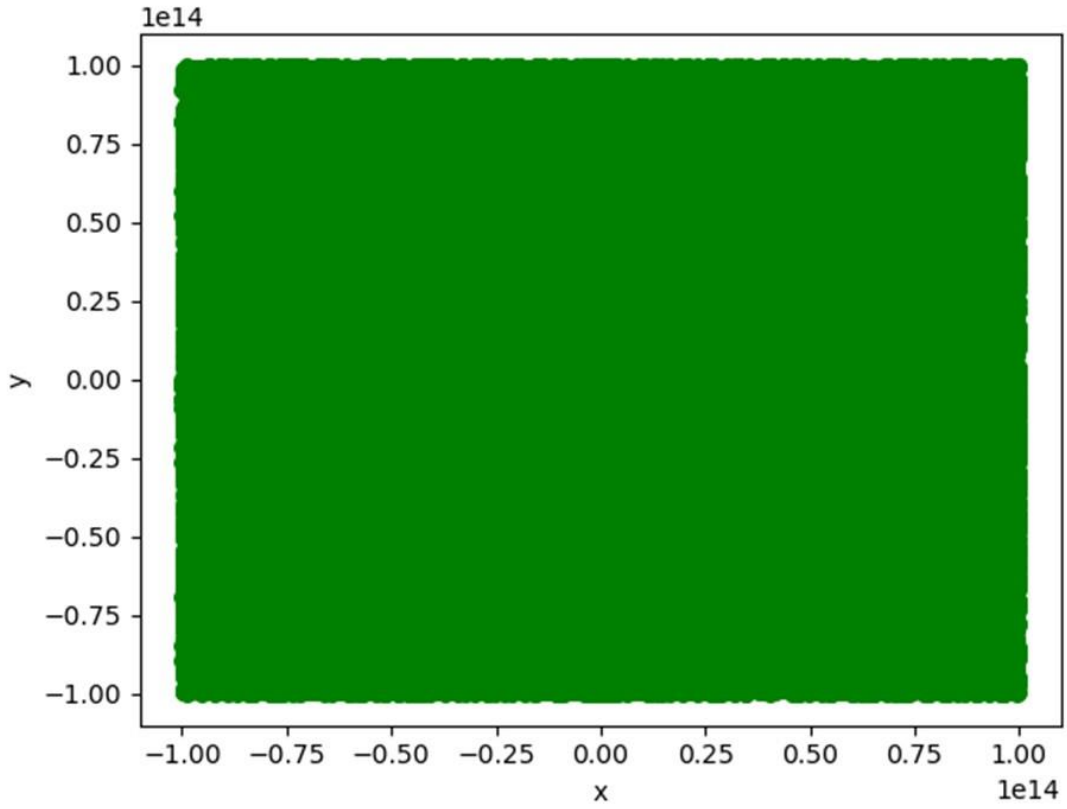
3. Przebieg ćwiczenia

3.1 Generowanie zbiorów

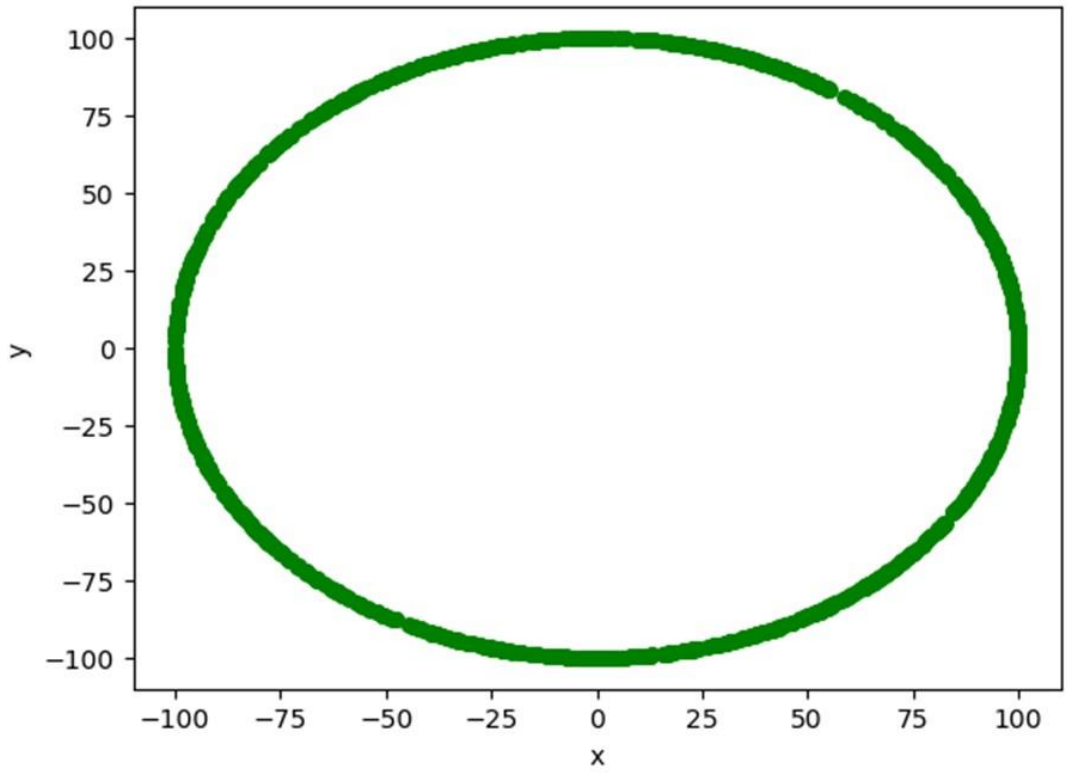
Przy pomocy funkcji random.uniform() przygotowano następujące zbiory:



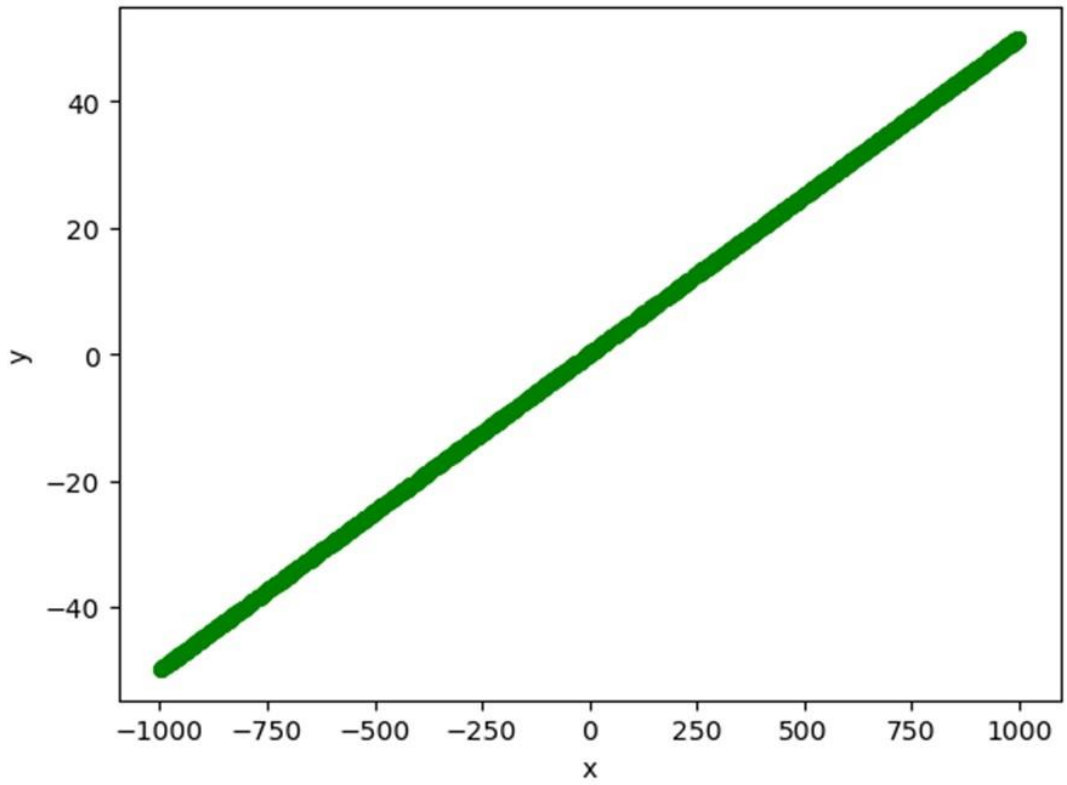
Rys. 1: Zbiór A - 10^5 losowych punktów o współrzędnych z przedziału [−1000, 1000].



Rys. 2: Zbiór B - 10^5 losowych punktów o współrzędnych z przedziału $[-10^{14}, 10^{14}]$.



Rys. 3: Zbiór C - 1000 losowych punktów leżących na okręgu o środku (0, 0) i promieniu R=100.



Rys. 4: Zbiór D - 1000 losowych punktów o współrzędnych z przedziału [−1000, 1000] leżących na prostej wyznaczanej przez dwa punkty A = (-1.0, 0.0) i B = (1.0, 0.1).

3.2 Określanie położenia

Aby określić, po której stronie prostej leży punkt, użyto wyznaczników macierzy 2x2 oraz 3x3:

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

Gdzie a i b tworzą prostą, a c jest punktem, którego położenie badamy.

Zależność wartości wyznacznika od położenia punktu:

- $\det(a,b,c) < 0$ - punkt c leży na prawo od prostej ab
- $\det(a,b,c) = 0$ - punkt c leży na prostej ab
- $\det(a,b,c) > 0$ - punkt c leży na lewo od prostej ab

Ze względu na ograniczoną precyzję arytmetyki zmiennoprzecinkowej, wyznacznik dla punktów leżących dokładnie na prostej rzadko przyjmuje wartość dokładnie zero. Zamiast tego stosuje się tolerancję eps, która definiuje przedział wokół zera, w którym punkt uznawany jest za współliniowy:

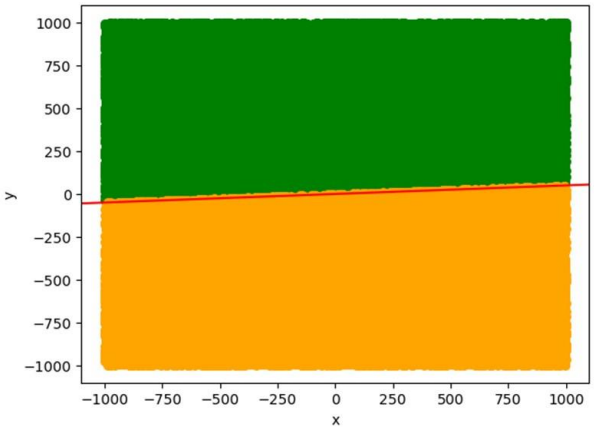
|det(a,b,c)| <= eps - punkt leży na prostej.

4. Wyniki

Punkty w każdym z czterech zbiorów zostały pogrupowane według położenia względem prostej wyznaczanej przez punkty A = (1.0, 0.0) i B = (1.0, 0.1). Położenie to było ustalane przy użyciu różnych metod obliczania wyznacznika, wartości tolerancji zera oraz dokładności zmiennej liczbowej. Do wizualizacji użyto podziału na kolory: zielony – na lewo od prostej (Left), fioletowy – współliniowy (Mid), żółty – na prawo od prostej (Right).

4.1 Zbiór A

typ	det func	eps	Left	Mid	Right
float64, float32	2x2, 2x2_lib, 3x3, 3x3_lib	0, 1e-12, 1e-8, 1e-4	49989	0	50011

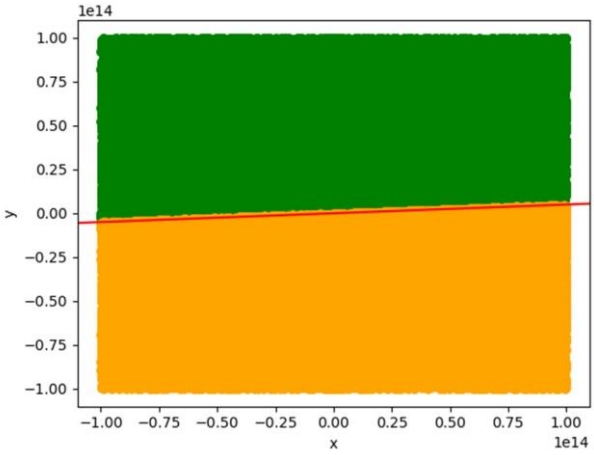


Rys. 5: Wizualizacja dla każdego z przypadków.

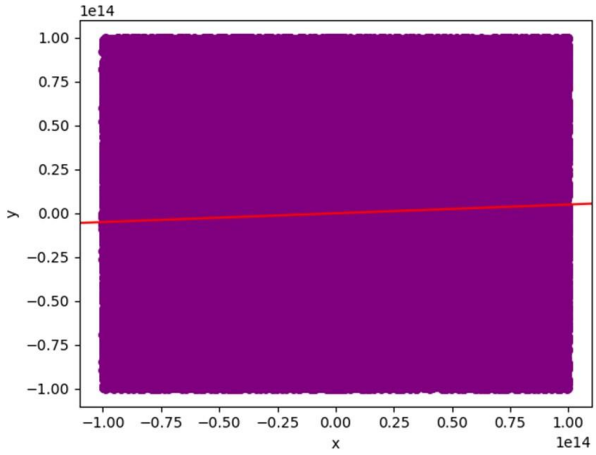
Wyniki są identyczne niezależnie od precyzji float, wybranego eps i metody wyznaczania wyznacznika.

4.2 Zbiór B

	det func	eps	Left	Mid	Right		det func	eps	Left	Mid	Right
float64	2x2	0,00E+00	49848	4	50148	float32	2x2	0,00E+00	0	100000	0
		1,00E-12	49848	4	50148			1,00E-12	0	100000	0
		1,00E-08	49848	4	50148			1,00E-08	0	100000	0
		1,00E-04	49848	4	50148			1,00E-04	0	100000	0
	2x2_lib	0,00E+00	49850	0	50150		2x2_lib	0,00E+00	0	100000	0
		1,00E-12	49850	0	50150			1,00E-12	0	100000	0
		1,00E-08	49850	0	50150			1,00E-08	0	100000	0
		1,00E-04	49850	0	50150			1,00E-04	0	100000	0
	3x3	0,00E+00	49850	0	50150		3x3	0,00E+00	49850	0	50150
		1,00E-12	49850	0	50150			1,00E-12	49850	0	50150
		1,00E-08	49850	0	50150			1,00E-08	49850	0	50150
		1,00E-04	49850	0	50150			1,00E-04	49850	0	50150
	3x3_lib	0,00E+00	49850	0	50150		3x3_lib	0,00E+00	49850	0	50150
		1,00E-12	49850	0	50150			1,00E-12	49850	0	50150
		1,00E-08	49850	0	50150			1,00E-08	49850	0	50150
		1,00E-04	49850	0	50150			1,00E-04	49850	0	50150



Rys. 6: Dla większości przypadków.



Rys. 7: Dla precyzji float32, dowolnej implementacji wyznacznika 2x2 i dowolnego eps.

Dla większości przypadków dane zostały pogrupowane w taki sam sposób (Rys. 6). Na wyniki nie wpłynęła przyjęta tolerancja dla zera. Dla precyzji float64 oraz własnej implementacji wyznacznika 2x2 do współliniowych zaliczono 4 punkty:

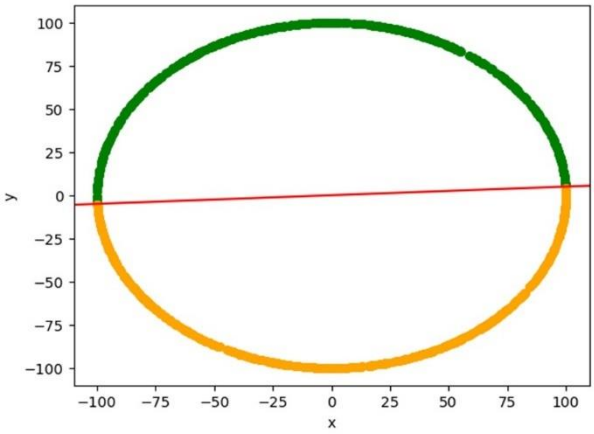
(82354795127306.97, 4098978929058.422), (-57093176688074.37, −2871983058693.0312), (57915738845373.59, 2906250984700.8594), (-38451773640830.39, −1924500862452.4219)

Problem wynika z budowy liczb zmiennoprzecinkowych, stały rozmiar mantysy przy dużych liczbach oznacza większe przerwy między reprezentowalnymi wartościami. Powoduje to utratę precyzji przy odejmowaniu bliskich sobie, dużych liczb (jak w wyznaczniku 2x2). Dla innych funkcji wyznaczania wyznaczników żaden punkt nie został zakwalifikowany jako współliniowy.

Przyjęcie precyzji float32 oraz użycie dowolnego wyznacznika 2x2 (Rys. 7) spowodowało wystąpienie błędu, każdy punkt w zbiorze został zakwalifikowany jako należący do prostej.
Typ float32 przechowuje tylko ok. 7 cyfr znaczących, co jest niewystarczające przy liczbach rzędu 10^{14}. Podczas wyliczania wyznacznika 2x2 małe współrzędne punktów A i B są nieznaczące podczas odejmowania ich od o wiele większych współrzędnych punktu C. W efekcie float32 sprowadzał A_x-C_x i B_x-C_x oraz A_y-C_y i B_y-C_y do tego samego. Całe równanie sprowadzało się więc do odejmowania dwóch identycznych wartości, co dla każdego punktu dawało wynik 0.0.

4.3 Zbiór C

typ	det func	eps	Left	Mid	Right
float64, float32	2x2, 2x2_lib, 3x3, 3x3_lib	0, 1e-12, 1e-8, 1e-4	481	0	519

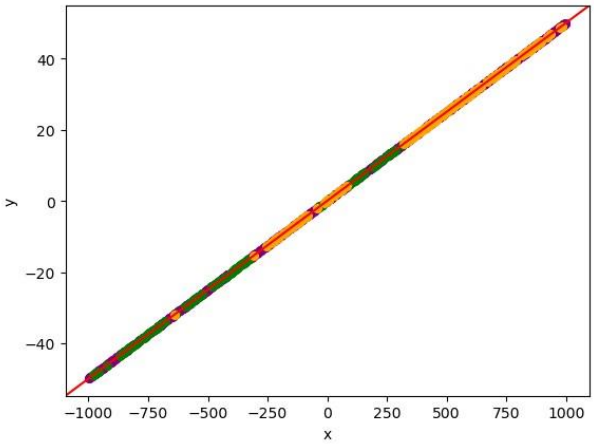


Rys. 8: Wizualizacja zbioru C dla każdego z przypadków.

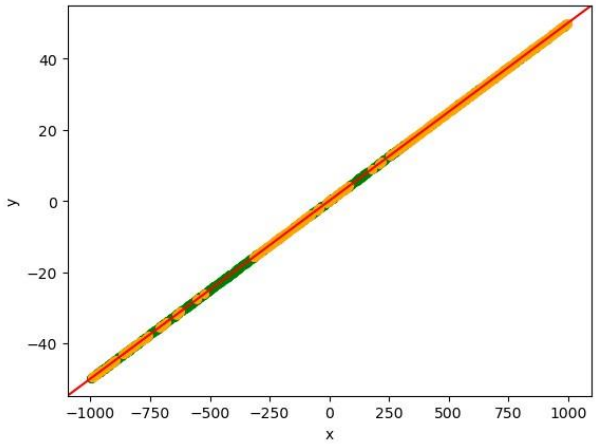
Wyniki są identyczne niezależnie od precyzji float, wybranego eps i metody wyznaczania wyznacznika.

4.4 Zbiór D

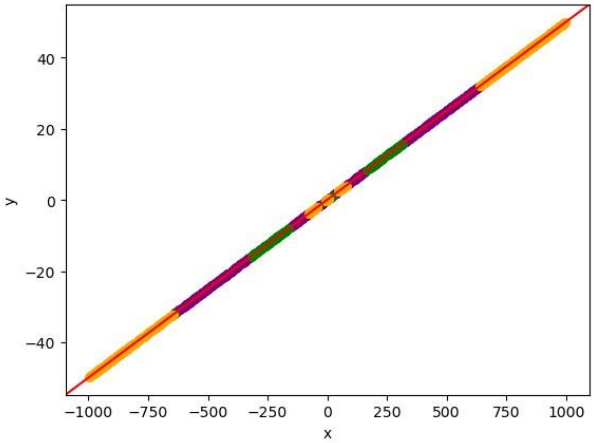
	det func	eps	Left	Mid	Right		det func	eps	Left	Mid	Right
float64	2x2	0,00E+00	131	708	161	float32	2x2	0,00E+00	170	665	165
		1,00E-12	62	847	91			1,00E-12	170	665	165
		1,00E-08	0	1000	0			1,00E-08	169	667	164
		1,00E-04	0	1000	0			1,00E-04	154	698	148
	2x2_lib	0,00E+00	468	0	532		2x2_lib	0,00E+00	497	0	503
		1,00E-12	113	738	148			1,00E-12	497	0	503
		1,00E-08	0	1000	0			1,00E-08	496	4	500
		1,00E-04	0	1000	0			1,00E-04	388	239	373
	3x3	0,00E+00	157	423	420		3x3	0,00E+00	300	391	309
		1,00E-12	0	1000	0			1,00E-12	300	391	309
		1,00E-08	0	1000	0			1,00E-08	299	393	308
		1,00E-04	0	1000	0			1,00E-04	0	1000	0
	3x3_lib	0,00E+00	459	1	540		3x3_lib	0,00E+00	491	0	509
		1,00E-12	0	1000	0			1,00E-12	416	165	419
		1,00E-08	0	1000	0			1,00E-08	415	168	417
		1,00E-04	0	1000	0			1,00E-04	0	1000	0



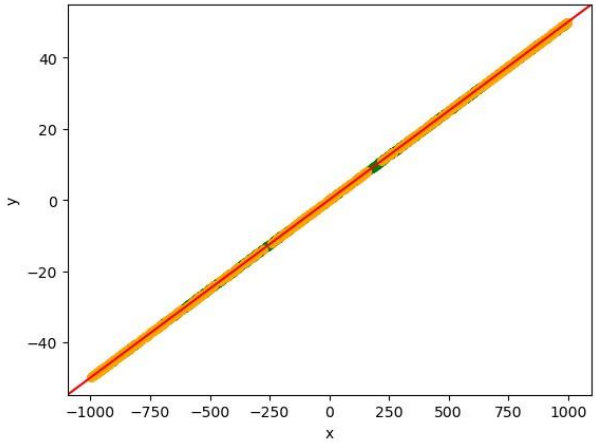
Rys. 9: float64 det2x2 eps=0



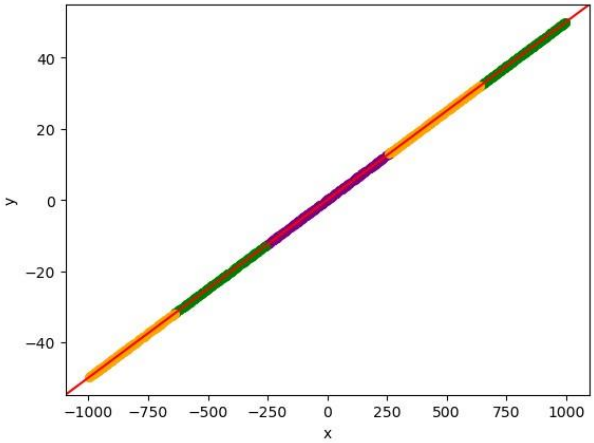
Rys. 10: float64 det2x2lib eps=0



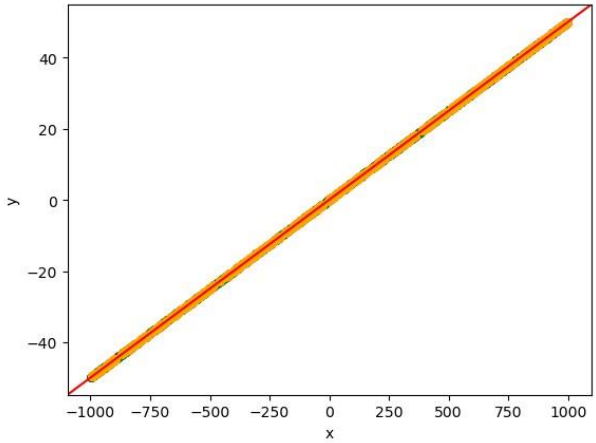
Rys. 11: float64 det3x3 eps=0



Rys. 12: float64 det3x3lib eps=0



Rys. 13: float32 det2x2lib eps=1e-4



Rys. 14: float32 det3x3lib eps=1e-8

Wyniki dla Zbioru D wykazują największe zróżnicowanie pod względem stabilności numerycznej. Implementacje własne dla float64 (det2x2, det3x3) okazują się podatne na błędy zaokrągleń w tym teście. Jest to widoczne na Rys. 9 i Rys. 11, gdzie przy eps=0 punkty są klasyfikowane we wszystkich trzech kategoriach w sposób nieuporządkowany. Tabela potwierdza, że błędy te są na tyle znaczące, iż wystarczy bardzo mała tolerancja (1e-12 lub 1e-8), aby wszystkie punkty zostały zakwalifikowane jako leżące na prostej. Dla kontrastu, implementacje biblioteczne wyznacznika na float64 (Rys. 10, Rys. 12) działają bardzo stabilnie. Przy eps=0 precyzyjnie rozdzielają punkty na leżące po lewej i po prawej stronie linii, co sugeruje znacznie lepsze zarządzanie precyzją. Co interesujące, obie

implementacje dla float 32 (Rys. 13, Rys. 14) również zachowują się w sposób stabilny i przewidywalny, podobnie do float64_lib. Na Rys. 14 widać czysty podział na lewo/prawo przy eps=1e-8. Szczególnie ciekawy jest Rys. 13, który ilustruje stabilność float32 przy dużej tolerancji 1e-4. Zamiast klasyfikować wszystkie punkty jako środkowe (co stało się z float64), algorytm pogrupował punkty w wyraźne, odseparowane od siebie skupiska, rozmieszczone symetrycznie wzdłuż prostej. Podsumowując, problemem nie jest sama precyzja float64, lecz niestabilność numeryczna algorytmu użytego w ręcznych implementacjach. Zarówno float32, jak i funkcje biblioteczne z float64 poradziły sobie z tym zbiorem w sposób bardziej przewidywalny.

5. Wnioski

5.1 Wpływ zakresu współrzędnych

Dla punktów o współrzędnych z przedziału [−1000, 1000] (zbiór A) wszystkie implementacje i precyzje dają identyczne, poprawne wyniki. Problemy numeryczne pojawiają się dopiero przy bardzo dużych wartościach współrzędnych (zbiór B, przedział $[-10^{14}, 10^{14}]$), gdzie float32 całkowicie zawodzi, klasyfikując wszystkie punkty jako współliniowe.

5.2 Precyzja obliczeń

Float64 w zbiorze B wykazuje większą stabilność niż float32, jednak wciąż występują błędy (4 punkty błędnie sklasyfikowane jako współliniowe dla implementacji 2x2). Wyniki pokazują, że precyzja float64 nie gwarantuje całkowitej eliminacji błędów numerycznych przy ekstremalnych wartościach.

5.3 Punkty na prostej (zbiór D)

Zbiór D ujawnia nieoczekiwany wynik - float32 lepiej radzi sobie z wykrywaniem punktów rzeczywiście leżących na prostej niż float64. Float64 wymaga bardzo dużych wartości eps (1e-12 dla 2x2, 1e-8 dla 3x3), aby zaklasyfikować wszystkie punkty jako współliniowe, co sugeruje akumulację błędów zaokrągleń w obliczeniach o wyższej precyzji. Float32 z eps=1e-4 daje bardziej przewidywalne wyniki z lepszą separacją punktów.

5.4 Różnice między implementacjami

Implementacje biblioteczne (det2x2lib, det3x3lib) wykazują nieco inne zachowanie niż własne implementacje, szczególnie widoczne w zbiorze D. Wyznacznik 3x3 generalnie wymaga mniejszych wartości eps niż wyznacznik 2x2 dla uzyskania porównywalnych wyników.

5.5 Tolerancja eps

Odpowiedni dobór wartości eps jest kluczowy dla poprawnego działania predykatu. Zbyt mała wartość prowadzi do błędnej klasyfikacji punktów leżących na prostej (zbiór D), zbyt duża może powodować błędne uznawanie punktów za współliniowe (zbiór B). Optymalna wartość eps zależy od zakresu współrzędnych, precyzji obliczeń i implementacji wyznacznika.