Sprawozdanie z ćwiczenia pierwszego

Algorytmy geometryczne

Jakub Pisarek
Informatyka, WIEiT, AGH

23 października 2022

1 Cel ćwiczenia

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

Interpretacja wyników ćwiczenia polega na podziale punktów z $\mathbb{R}^2$ względem ich orientacji w stosunku do danego odcinka (skierowanego) $\bar{ab}$, znajdującego się na tej samej płaszczyźnie, co osiągane jest przy użyciu wyznaczników odpowiednich macierzy. Podział ten należy powtórzyć kilkukrotnie, przy użyciu różnych dokładności obliczeń, różnych wyznaczników i tolerancji dla zera, a następnie porównać uzyskane wyniki.

2 Wprowadzenie

Niech $a, b, c \in \mathbb{R}^2$, $a = (a_x, a_y)$, $b = (b_x, b_y)$, $c = (c_x, c_y)$.

Orientację punktu c w stosunku do odcinka skierowanego $\bar{ab}$ określić można za pomocą wyznacznika macierzy

$$\begin{vmatrix} a_x & a_y & 1 \\ b_x & b_y & 1 \\ c_x & c_y & 1 \end{vmatrix} \tag{1}$$

albo macierzy

$$\begin{vmatrix} a_x - c_x & a_y - c_y \\ b_x - c_x & b_y - c_y \end{vmatrix} \tag{2}$$

Gdy wyznacznik ten jest ujemny, punkt c leży po prawej stronie prostej wyznaczonej przez odcinek $\bar{ab}$, gdy jest dodatni – po lewej, gdy jest równy 0, punkty są współliniowe.

3 Przebieg ćwiczenia

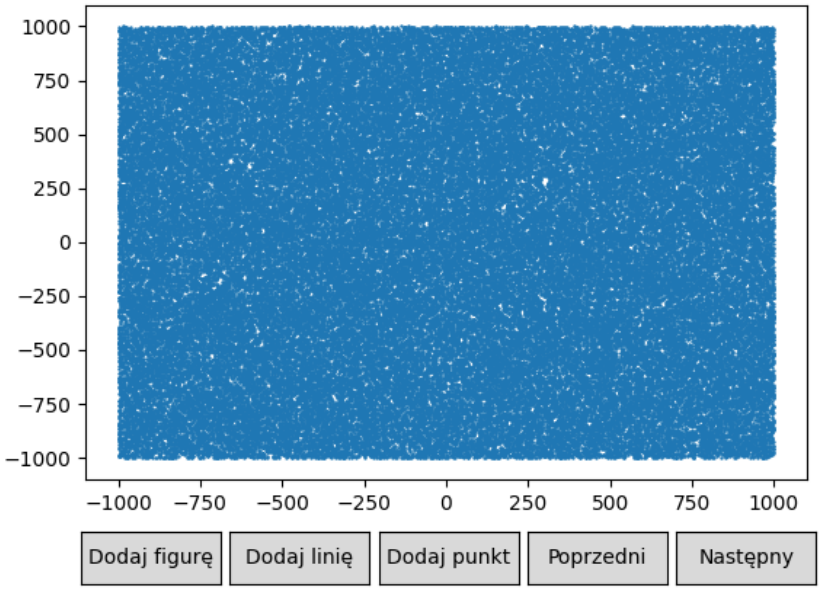
1. Przygotować następujące zbiory punktów (2D, współrzędne rzeczywiste typu double):
   a) 10^5 losowych punktów o współrzędnych z przedziału [-1000, 1000],
   b) 10^5 losowych punktów o współrzędnych z przedziału $[-10^{14}, 10^{14}]$,
   c) 1000 losowych punktów leżących na okręgu o środku (0, 0) i promieniu $R = 100$,
   d) 1000 losowych punktów o współrzędnych z przedziału [-1000, 1000] leżących na prostej wyznaczonej przez wektor $(\mathbf{a}, \mathbf{b})$,
   dla $\mathbf{a} = (-1.0, 0.0), \mathbf{b} = (1.0, 0.1)$.
2. Uruchomić wizualizację graficzną utworzonych zbiorów punktów.

3. Przygotować program, który dla każdego ze zbioru danych dokona podziału punktów względem ich orientacji w stosunku do odcinka $\bar{ab}$, $a = (-1.0, 0.0)$, $b = (1.0, 0.1)$ – punkty znajdujące się po lewej stronie, po prawej stronie oraz współliniowe:
   - Obliczenia wykonać przy pomocy wyznacznika (1) i następnie (2) zaimplementowanego samodzielnie,
   - Wyszukać w bibliotekach numerycznych procedury obliczania wyznacznika 3x3 i 2x2,
   - Dla każdego zbioru danych porównać wyniki (podział punktów) uzyskane przy pomocy obu wyznaczników wyliczanych procedurami własnymi i bibliotecznymi,
   - Określić, ile punktów (i jakich) zostało inaczej zakwalifikowanych dla różnych sposobów liczenia wyznacznika,
   - Zbadać wyniki dla różnej tolerancji dla zera oraz różnych precyzji obliczeń,
   - Odpowiednio zaprezentować otrzymane wyniki w tabelach.
4. Przedstawić graficznie różnice w podziale punktów.
5. Opisać wnioski.

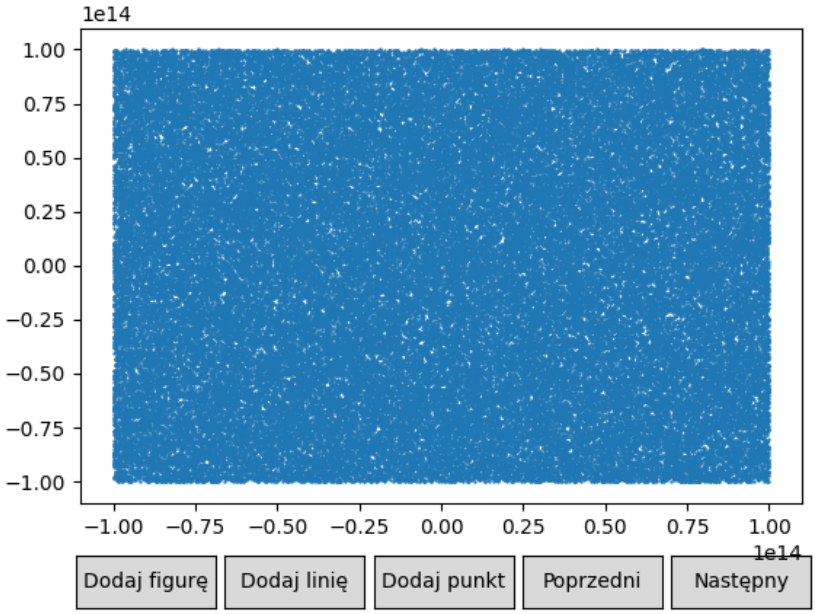
4 Wyniki

4.1 Wizualizacja zbiorów punktów

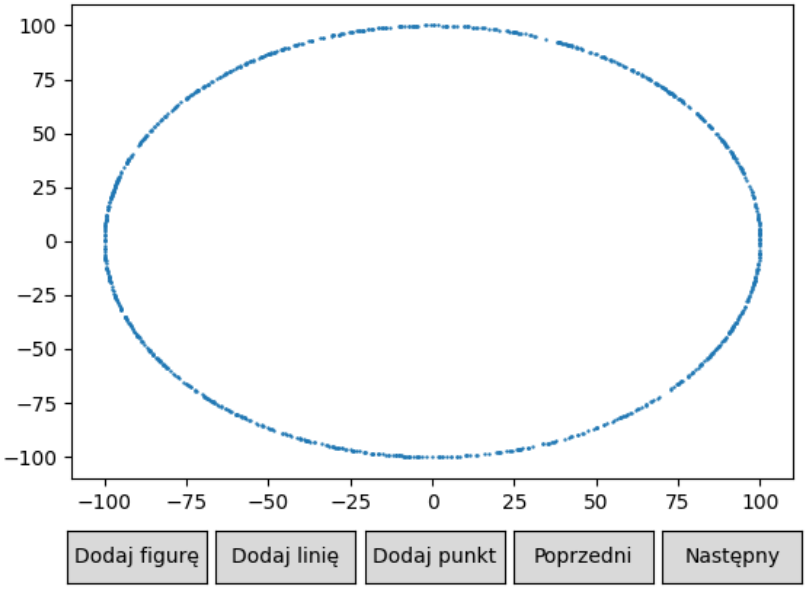
Nazywając poszczególne zbiory zgodnie z podpunktami, w których zostały nadmienione, otrzymujemy:



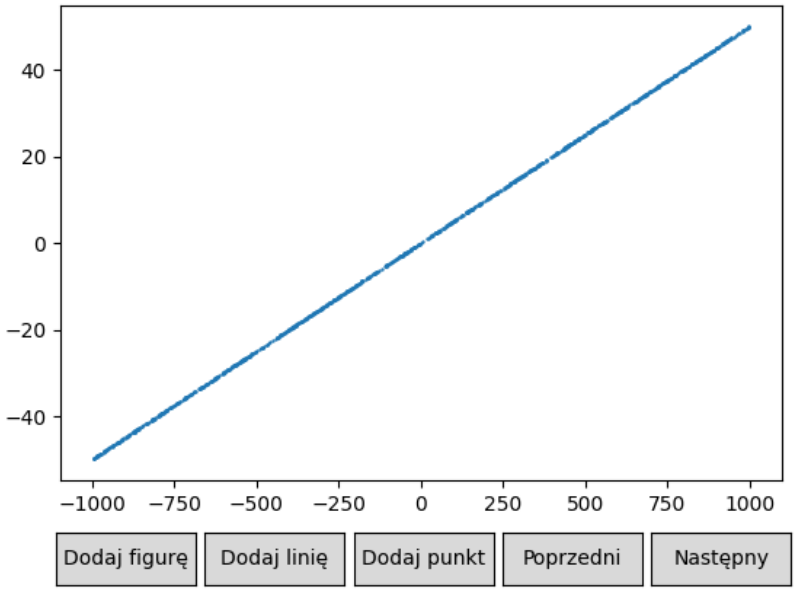
Ryc. 1: Wizualizacja punktów ze zbioru A



Ryc. 2: Wizualizacja punktów ze zbioru B



Ryc. 3: Wizualizacja punktów ze zbioru C



Ryc. 4: Wizualizacja punktów ze zbioru D

4.2 Podziały punktów

Poniżej przedstawiono wyniki działania funkcji podziału punktów dla konkretnych zbiorów. Wyniki powtarzające się scalono w celu zaoszczędzenia miejsca.

4.2.1 Zbiór A

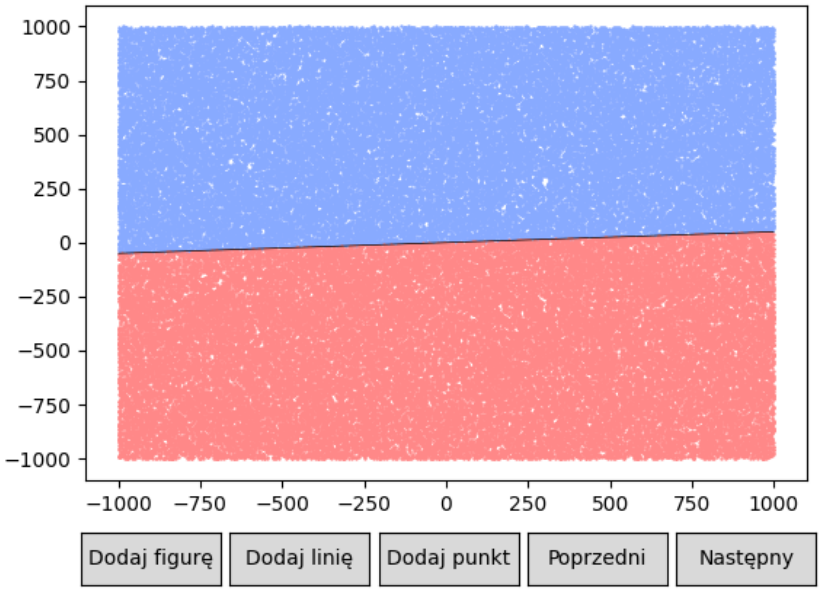
Tabela 1: Wyniki działania funkcji podziału punktów dla zbioru A (precyzja *float64*)

Wyznacznik	**Tolerancja dla zera**	**Podział punktów**		
		po lewej	współliniowe	po prawej
(1) własny, (1) biblioteczny, (2) własny, (2) biblioteczny	10^{-10}, 10^{-12}, 10^{-14}	50121	0	49879

Fakt, że wszystkie wywołania funkcji dały te same wyniki, również przy zmniejszającej się tolerancji dla zera, pozwala upewnić się o poprawności własnej implementacji funkcji liczących wyznacznik.

Otrzymany podział przedstawiony jest na poniższej rycinie. Kolorem niebieskim zaznaczono punkty znajdujące się po lewej od prostej będącej przedłużeniem odcinka $\bar{ab}$, kolorem czerwonym – po prawej, kolorem zielonym – punkty określone jako współliniowe; samą prostą zaznaczono kolorem czarnym[1].

[1] Opisany tu schemat kolorystyczny pozostanie niezmienny przez resztę sprawozdania.



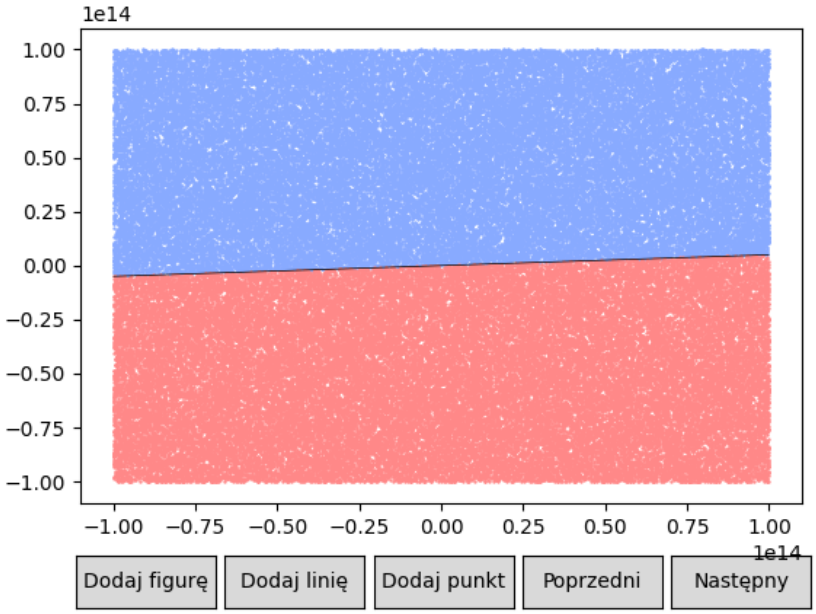
Ryc. 5: Wizualizacja podziału punktów ze zbioru A (wszystkie wyznaczniki, wszystkie tolerancje)

4.2.2 Zbiór B

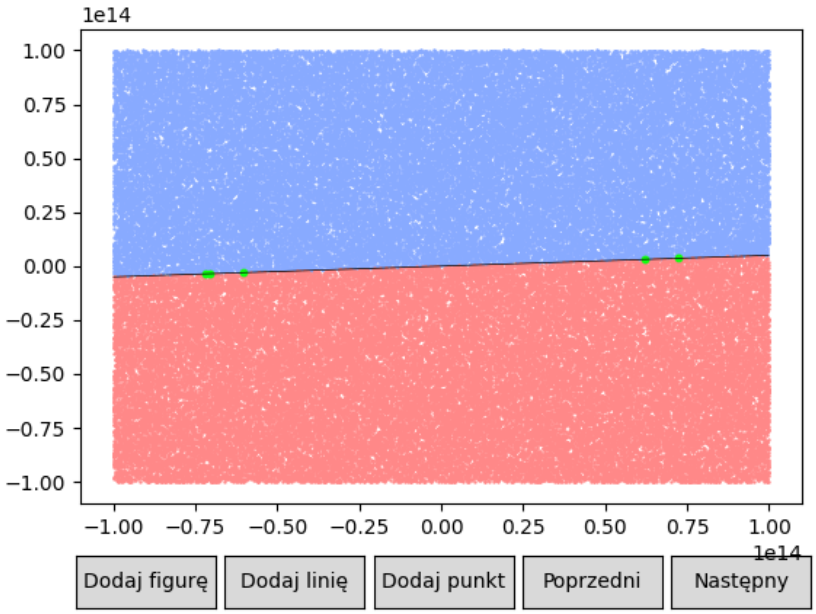
Tabela 2: Wyniki działania funkcji podziału punktów dla zbioru B (precyzja *float64*)

Wyznacznik	**Tolerancja dla zera**	**Podział punktów**		
		po lewej	współliniowe	po prawej
(1) własny	10^{-10}, 10^{-12}, 10^{-14}	49916	0	50084
(1) biblioteczny	10^{-10}, 10^{-12}, 10^{-14}	49916	0	50084
(2) własny	10^{-10}, 10^{-12}, 10^{-14}	49913	5	50082
(2) biblioteczny	10^{-10}, 10^{-12}, 10^{-14}	49911	5	50084

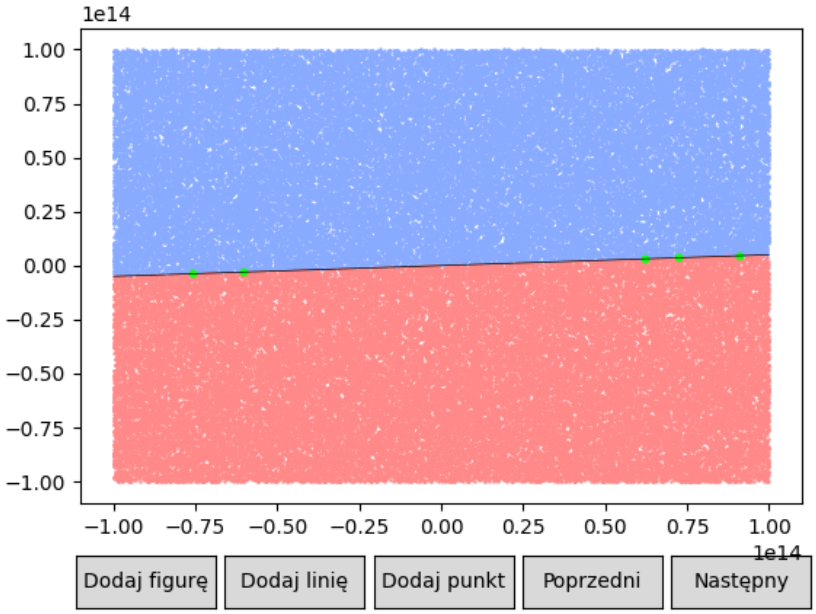
W przypadku każdego wyznacznika, wyniki są identyczne, niezależnie od zmniejszania tolerancji dla zera. Występują jednak różnice między wyznacznikami (1) i (2), a nawet w obrębie wyznacznika (2), między implementacją własną i biblioteczną. Wyniki zaprezentowano na poniższych rycinach:



Ryc. 6: Wizualizacja podziału punktów ze zbioru B (wyznaczniki (1), wszystkie tolerancje)



Ryc. 7: Wizualizacja podziału punktów ze zbioru B (wyznacznik własny (2), wszystkie tolerancje, wyróżnione punkty współliniowe)



Ryc. 8: Wizualizacja podziału punktów ze zbioru B (wyznacznik biblioteczny (2), wszystkie tolerancje, wyróżnione punkty współliniowe)

Przeprowadzono ponowną klasyfikację punktów, z użyciem większej precyzji obliczeń (*float128*). Było to możliwe jedynie dla wyznaczników w implementacji własnej, gdyż używana funkcja biblioteczna nie obsługiwała takiej precyzji.

Tabela 3: Wyniki działania funkcji podziału punktów dla zbioru B (precyzja *float128*)

Wyznacznik	**Tolerancja dla zera**	**Podział punktów**		
		po lewej	współliniowe	po prawej
(1) własny, (2) własny	10^{-10}, 10^{-12}, 10^{-14}, 10^{-16}, 10^{-18}	49916	0	50084

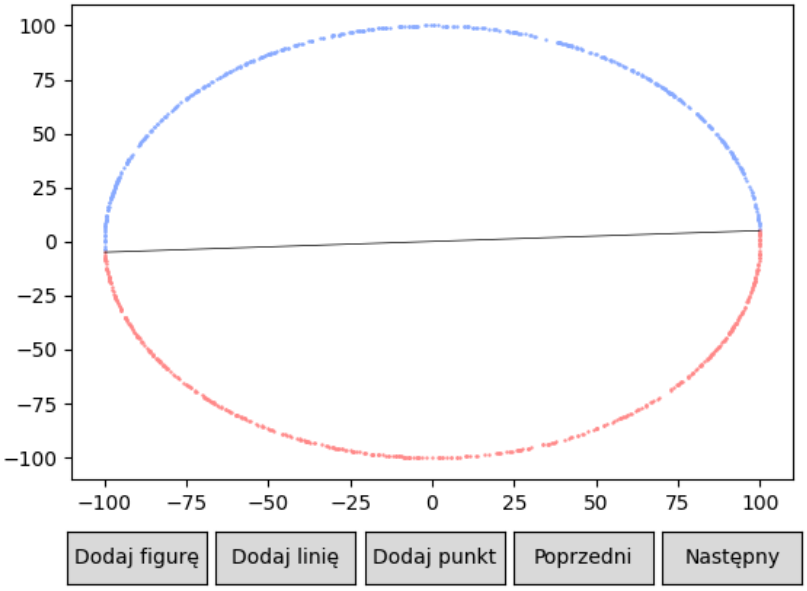
Wyniki dla zwiększonej precyzji obliczeń są jednakowe w obu przypadkach, nawet dla znacznie zmniejszonej tolerancji dla zera. Wywnioskować z tego można, że wyznacznik (1) jest dokładniejszy i daje poprawne wyniki w przypadkach granicznych nawet przy mniejszej precyzji obliczeń.

4.2.3 Zbiór C

Tabela 4: Wyniki działania funkcji podziału punktów dla zbioru C (precyzja *float64*)

Wyznacznik	**Tolerancja dla zera**	**Podział punktów**		
		po lewej	współliniowe	po prawej
(1) własny, (1) biblioteczny, (2) własny, (2) biblioteczny	10^{-10}, 10^{-12}, 10^{-14}	512	0	488

Tak jak w przypadku zbioru A, wyniki są identyczne we wszystkich przypadkach.



Ryc. 9: Wizualizacja podziału punktów ze zbioru C (wszystkie wyznaczniki, wszystkie tolerancje)

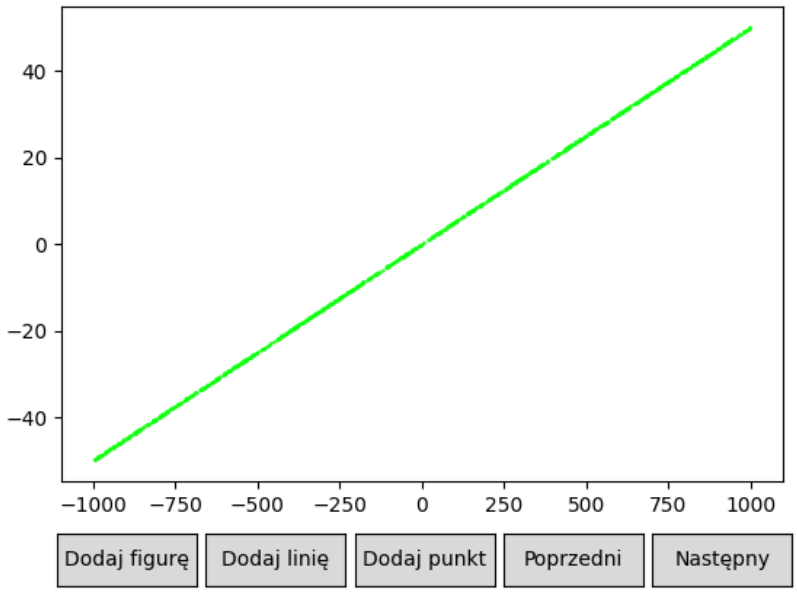
4.2.4 Zbiór D

Tabela 5: Wyniki działania funkcji podziału punktów dla zbioru D (precyzja *float64*)

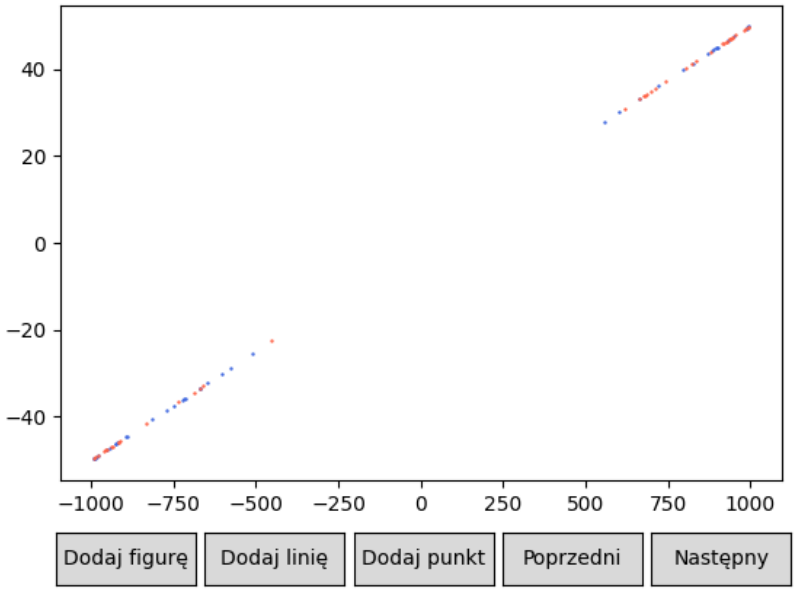
Wyznacznik	**Tolerancja dla zera**	**Podział punktów**		
		po lewej	współliniowe	po prawej
(1) własny	10^{-10}	0	1000	0
	10^{-12}	0	1000	0
	10^{-14}	0	1000	0
(1) biblioteczny	10^{-10}	0	1000	0
	10^{-12}	0	1000	0
	10^{-14}	42	916	42
(2) własny	10^{-10}	0	1000	0
	10^{-12}	68	828	104
	10^{-14}	121	722	157
(2) biblioteczny	10^{-10}	0	1000	0
	10^{-12}	96	782	122
	10^{-14}	153	691	156

Zauważyć można spadek dokładności wraz ze zmniejszaniem tolerancji dla zera dla wszystkich wykładników za wyjątkiem (1) w implementacji własnej. Widać również, że własna implementacja wyznacznika dla mniejszych tolerancji w obydwu przypadkach daje lepsze wyniki od funkcji bibliotecznych.

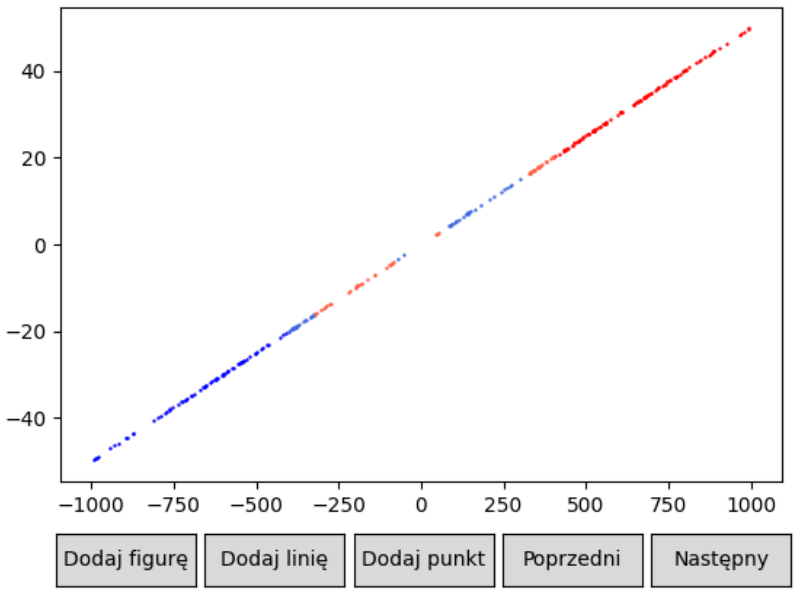
Na poniższych rycinach przedstawiono wyniki podziałów. Dla wyznaczników innych niż (1) w implementacji własnej ukazano jedynie punkty błędnie sklasyfikowane – zaznaczone kolorem tym ciemniejszym, przy im większej tolerancji dla zera ów błąd wystąpił dla danego punktu.



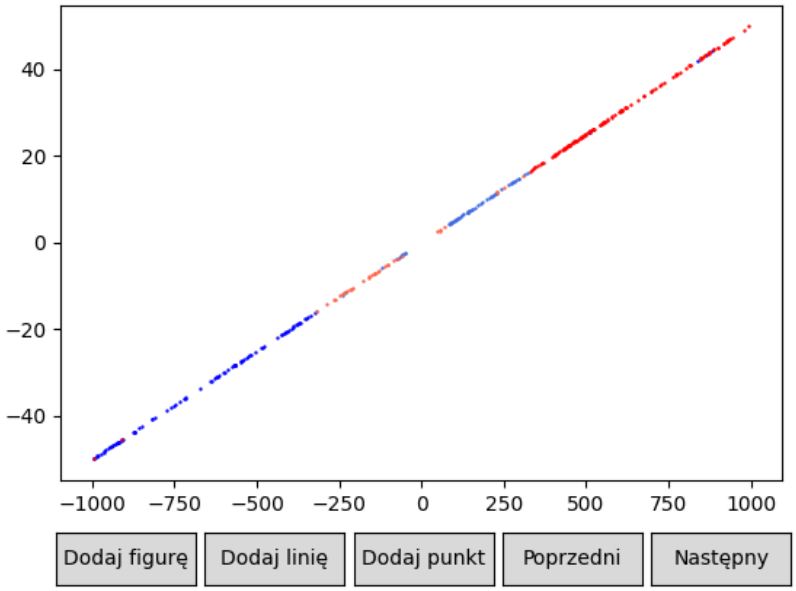
Ryc. 10: Wizualizacja podziału punktów ze zbioru D (wyznacznik własny (1), wszystkie tolerancje)



Ryc. 11: Wizualizacja podziału punktów ze zbioru D (wyznacznik biblioteczny (1), wszystkie tolerancje, wyłącznie punkty sklasyfikowane nieprawidłowo)



Ryc. 12: Wizualizacja podziału punktów ze zbioru D (wyznacznik własny (2), wszystkie tolerancje, wyłącznie punkty sklasyfikowane nieprawidłowo)



Ryc. 13: Wizualizacja podziału punktów ze zbioru D (wyznacznik biblioteczny (2), wszystkie tolerancje, wyłącznie punkty sklasyfikowane nieprawidłowo)

Można zauważyć, że w przypadku wyznacznika bibliotecznego (1), większość błędów nastąpiła dla punktów z końców odcinka; wyznaczniki (2) zachowały się podobnie, tj. dla tolerancji 10^{-12}, błędny wynik uzyskały punkty głównie z końców odcinka, a dopiero dla niższej 10^{-14}, z jego środka. Co ciekawe, wyznaczniki (2) wykazały pewien trend do przekłamywania punktów z prawego końca odcinka na stronę prawą, a z lewego na stronę lewą[2].

Podobnie jak w przypadku zbioru B, dla wyznaczników zaimplementowanych własnoręcznie przeprowadzono ponowną klasyfikację punktów, z użyciem większej precyzji obliczeń (*float128*).

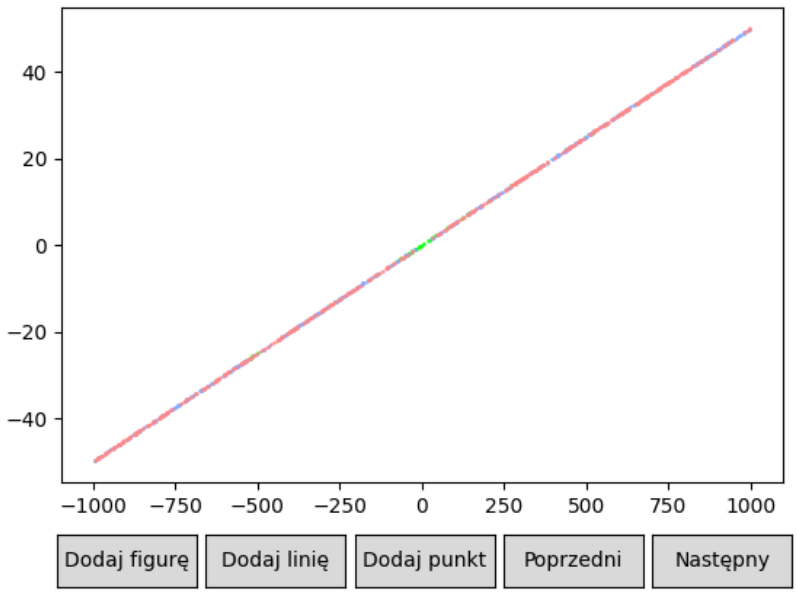
Tabela 6: Wyniki działania funkcji podziału punktów dla zbioru D (precyzja *float128*)

Wyznacznik	**Tolerancja dla zera**	**Podział punktów**		
		po lewej	współliniowe	po prawej
(1) własny	10^{-10}	0	1000	0
	10^{-12}	0	1000	0
	10^{-14}	0	1000	0
	10^{-16}	464	52	484
	10^{-18}	491	1	508
(2) własny	10^{-10}	0	1000	0
	10^{-12}	0	1000	0
	10^{-14}	0	1000	0
	10^{-16}	415	143	442
	10^{-18}	433	109	458

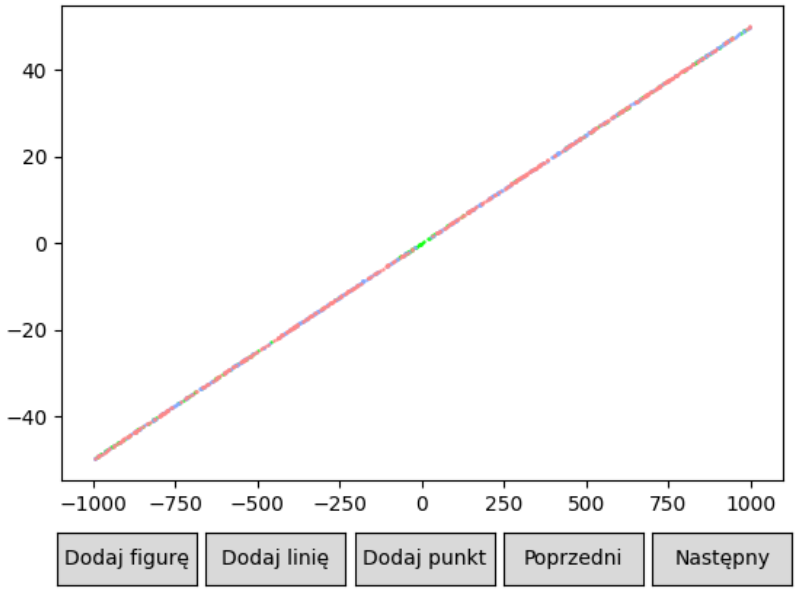
[2]Stało się tak głównie dla punktów skrajnych (co ciekawsze, dla punktów ze środka odcinka można zaobserwować odwrotną tendencję).

W obu przypadkach zauważalna jest drastyczna zmiana w podziale punktów między tolerancjami 10^{-14} a 10^{-16}. Spowodowane jest to tym, że współrzędne wyznaczonych punktów zostały wygenerowane jako wartości typu *float64*, więc zazwyczaj przy obliczeniach z większą precyzją okazywało się, że punkty te wcale nie leżą na prostej wyznaczonej przez odcinek $\bar{ab}$. W takiej sytuacji, paradoksalnie, lepiej poradził sobie wyznacznik (2), z racji na mniejszą dokładność wyniku.

Uzyskany podział dla dokładności 10^{-16} przedstawiono poniżej:



Ryc. 14: Wizualizacja podziału punktów ze zbioru D (wyznacznik własny (1), tolerancja 10^{-16}, precyzja *float128*)



Ryc. 15: Wizualizacja podziału punktów ze zbioru D (wyznacznik własny (2), tolerancja 10^{-16}, precyzja *float128*)

Ciężko tu wykazać jakiekolwiek prawidłowości inne niż grupa punktów pośrodku układu współrzędnych, które jako jedyne zostały zakwalifikowane jako współliniowe.

4.3 Czas działania funkcji

Poniższa tabela przedstawia czasy działania funkcji podziału punktów dla różnych funkcji liczenia wyznacznika. Pomiar czasu dotyczył podziału punktów ze wszystkich zbiorów, dla jednej precyzji obliczeń i trzech tolerancji dla zera.

Pięciokrotnie zmierzono czas dziesięciu następujących po sobie wywołań funkcji podziału punktów dla danej funkcji liczenia wyznacznika. Za uśredniony czas wywołania funkcji przyjęto najlepszy spośród pięciu wynik, podzielony przez liczbę wywołań (10).

Tabela 7: Pomiary czasu działania funkcji podziału punktów dla różnych funkcji liczenia wyznacznika

Wyznacznik	**Pomiary [s]**					**Uśredniony czas**
	1	2	3	4	5	**wywołania funkcji [s]**
(1) własny	3.70	3.51	3.44	3.44	3.49	0.344
(1) biblioteczny	50.28	50.15	50.23	50.13	50.19	5.013
(2) własny	2.86	2.79	2.83	2.83	2.78	0.278
(2) biblioteczny	43.87	43.76	43.63	43.65	43.61	4.361

Nietrudno zauważyć, że funkcje biblioteczne działają znacznie wolniej (1457% wolniej dla wyznacznika (1), 1569% wolniej dla wyznacznika (2)) niż te zaimplementowane własnoręcznie. Najpewniej jest to konsekwencja użycia w nich nieoptymalnego algorytmu liczenia wyznacznika, koniecznego, by funkcje te mogły radzić sobie również z wyznacznikami macierzy o większej ilości wierszy i kolumn.

Co więcej, wyznacznik (2) w obu przypadkach liczony jest szybciej niż wyznacznik (1) (19% szybciej dla implementacji własnych, 13% szybciej dla bibliotecznych), co jest stosunkowo łatwe do wyjaśnienia – wyznacznik (1) jest bardziej kosztowny obliczeniowo, z racji na większą ilość operacji mnożenia wymaganych do wyliczenia go.

5 Wnioski

- Funkcje biblioteczne liczące wykładniki macierzy działają kilkunastokrotnie wolniej od funkcji zaimplementowanych własnoręcznie,
- Wyznacznik (1) jest bardziej kosztowny obliczeniowo, lecz daje dokładniejsze wyniki niż (2),
- Przyjęcie zbyt małej tolerancji dla zera może powodować przekłamanie wyników, zwłaszcza przy nieadekwatnie dużej dokładności obliczeń.