

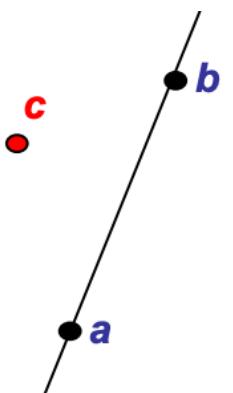
Algorytmy Geometryczne

Zadanie 1, laboratorium 1 - sprawozdanie

Radosław Kawa

1. Opis ćwiczenia

Zadaniem na laboratorium 1 było określenie po jakiej stronie znajduje się punkt od danego odcinka. Można było to zadanie wykonać poprzez obliczenie wyznaczników:


$$(1) \det(a, b, c) = \begin{vmatrix} a_x & a_y & 1 \\ b_x & b_y & 1 \\ c_x & c_y & 1 \end{vmatrix}$$
$$\text{lub } (2) \det(a, b, c) = \begin{vmatrix} a_x - c_x & a_y - c_y \\ b_x - c_x & b_y - c_y \end{vmatrix} \begin{cases} < 0 \\ > 0 \\ = 0 \end{cases}$$

Gdzie oczywiście a,b,c to są punkty, a,b należące do odcinka a punkt c to punkt badany. Jeśli dany wyznacznik jest większy od 0/zadanej tolerancji to ten punkt znajduje się po lewej stronie od odcinka, analogicznie, gdy jest mniejszy to po prawej stronie, a gdy równy 0 należy do odcinka/prostej przechodzącej przez ten odcinek (2 punkty tworzące). Wyniki wyznaczników mogą się nieco różnić w zależności od tego w jaki sposób jest liczony, a zarazem w jaki sposób komputer przechowuje liczby zmiennoprzecinkowe, przez co mogą wychodzić minimalnie inne liczby wyliczane przez wyznacznik.

2.Biblioteki oraz specyfikacja

Zadanie było wykonane w jupyter notebook, w języku Python. Wersja 3.10.0. Do obliczeń wykorzystałem bibliotekę numpy, aby uzyskać dokładniejsze wartości liczb i szybciej je generować. Do rysowania wykresów wykorzystałem standardowe narzędzie oparte na bibliotece matplotlib.

Specyfikacja:

System operacyjny: macOS Monterey

Procesor: Apple M1

Pamięć: 8GB

3. Algorytmy

Nazewnictwo:

2x2 – napisany własnoręcznie program do liczenia wyznacznika 2 na 2

3x3 – napisany własnoręcznie program do liczenia wyznacznika 3 na 3

np2x2/numpy2x2 – biblioteczny program do liczenia wyznacznika 2 na 2

np3x3/numpy3x3 – biblioteczny program do liczenia wyznacznika 2 na 2

Porównanie szybkości algorytmów na sprawdzaniu na podstawie wygenerowanych zbiorów (wszystkich):

	2x2	3x3	numpy2x2	numpy3x3
Czas [s]	0.5114981	0.6142389	2.8481873	3.0523504

Jak widzimy dużo wolniejsze są biblioteczne prawie 6 krotnie. Zaś najszybszy własnoręcznie zaimplementowany 2x2.

4.Wygenerowane zbiory oraz plan zadania

Na początku wygenerowałem punkty typu float64:

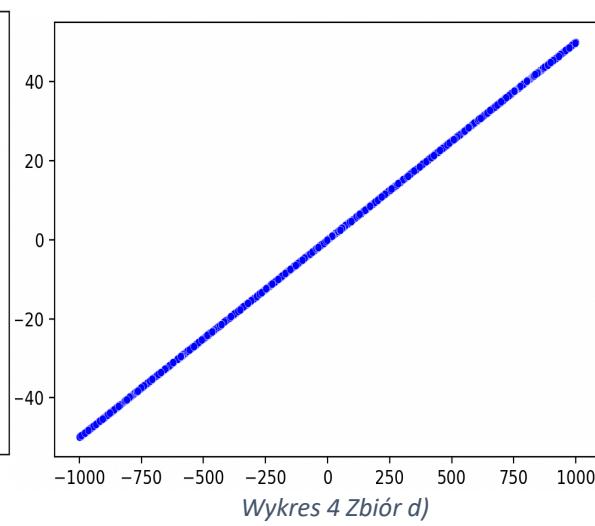
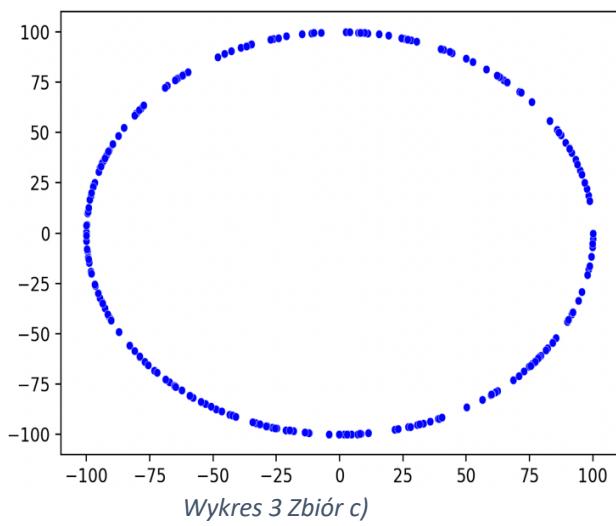
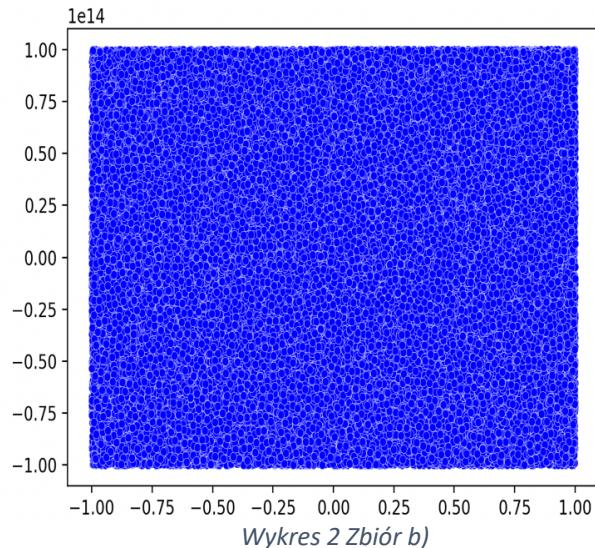
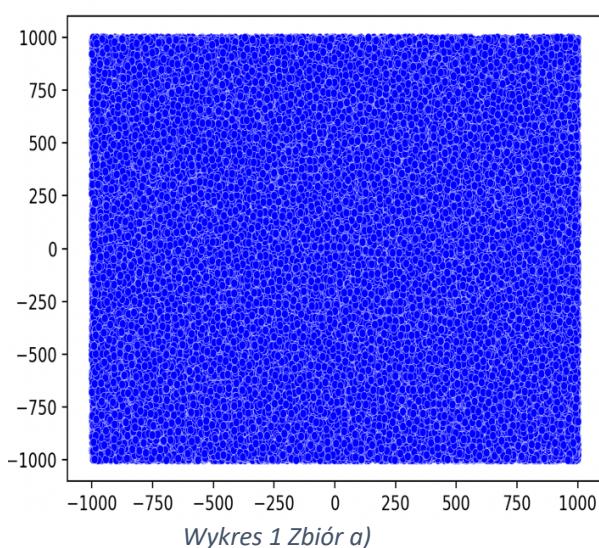
- 10^5 losowych punktów o współrzędnych z przedziału [-1000, 1000]
- 10^5 losowych punktów o współrzędnych z przedziału [-1014, 1014]
- 1000 losowych punktów leżących na okręgu o środku (0,0) i promieniu R=100

d) 1000 losowych punktów o współrzędnych z przedziału $[-1000, 1000]$ leżących na prostej wyznaczonej przez wektor (a, b) , $a = [-1.0, 0.0]$, $b = [1.0, 0.1]$.

Do generowania tych zbiorów użyłem funkcji `numpy.random.uniform` z biblioteki `numpy`.

Następnie do każdego zakresu danych utworzyłem wykresy obrazujące rozkład punktów.

Wykresy wygenerowanych punktów:



Następnie zaimplementowano funkcje liczące poszczególne wyznaczniki i zbierające te punkty do odpowiednich tablic:

Generuj2x2 – generuje zbiory na podstawie własnego wyznacznika 2x2

Generuj3x3 – generuje zbiory na podstawie własnego wyznacznika 3x3

Generujnp2x2 – generuje zbiory na podstawie wyznacznika bibliotecznego 2x2

Generujnp3x3 – generuje zbiory na podstawie wyznacznika bibliotecznego 3x3

5. Klasyfikacja punktów:

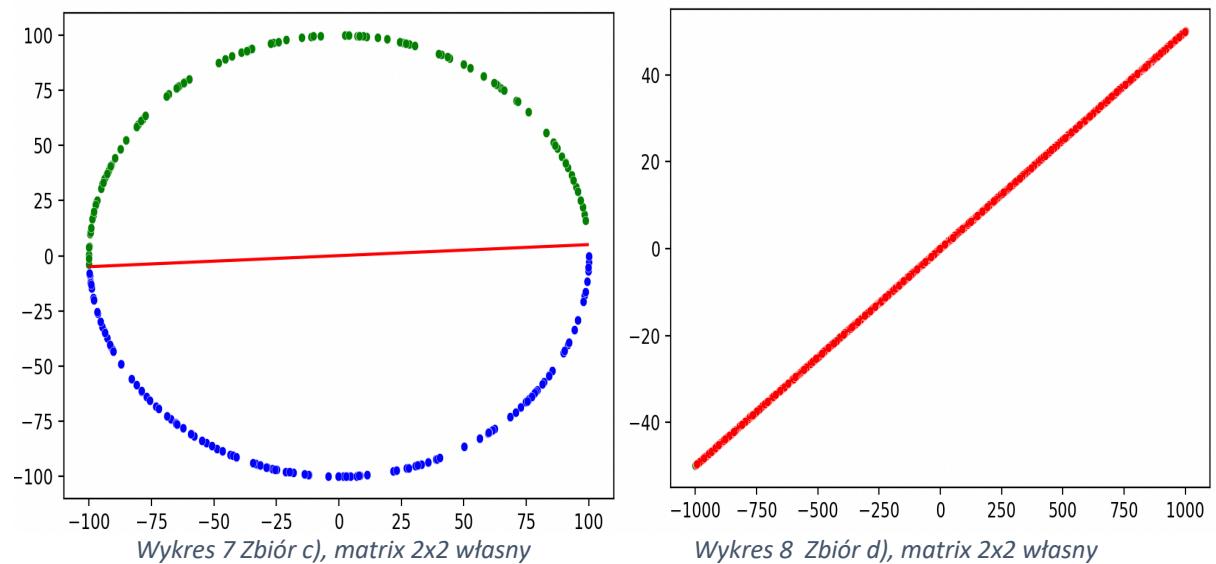
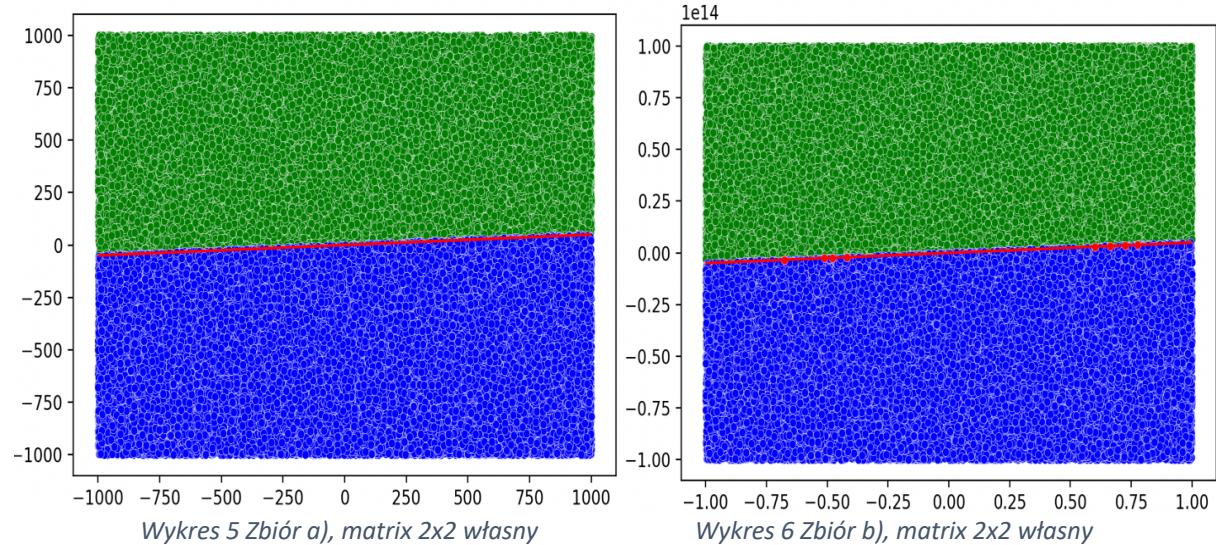
Na początek klasyfikuje te punkty w jaki sposób są rozmieszczone poprzez obliczania innym wyznacznikiem własnym lub bibliotecznym i zbieram dane do tabel, w tym przypadku tolerancja wynosi 0, typ danych dalej float64

Podział punktów		Lewo	Środek	Prawo
Matrix 2x2 własny	a)	49734	0	50266
	b)	50057	8	49935
	c)	119	0	131
	d)	140	716	144
Matrix 3x3 własny	a)	49734	0	50266
	b)	50061	0	49939
	c)	119	0	131
	d)	267	362	371
Matrix 2x2 numpy biblioteczny	a)	49734	0	50266
	b)	50061	0	49939
	c)	119	0	131
	d)	463	0	537
Matrix 3x3 numpy biblioteczny	a)	49734	0	50266
	b)	50061	0	49939
	c)	119	0	131
	d)	499	0	501

a), b), c), d) to kolejne zbiory, które generowałem

Jak widzimy na podstawie danych najbardziej precyzyjnym dla nas będzie matrix 2×2 , gdyż najwięcej punktów zakwalifikowało jako współliniowe.

Wykresy dla matrixa 2×2 , reszta wykresów znajduje się na jupyter notebook'u:



Następnym krokiem jest zbadanie przykładowych punktów jak się zachowują przy różnej tolerancji wybrałem dla nich 2 tolerancje, gdyż, przy większych tolerancjach nie ma takiej różnicy.

To pierwsze 5 punktów w tym przypadku

Legenda: Liczba 1 oznacza, że znajduje się po lewej stronie, liczba -1 po prawej a liczba 0, że leży na prostej.

Tabela1: Zbiór a)

	Tolerancja 10^{-14}				Tolerancja 10^{-12}			
	2x2	3x3	np2x2	np3x3	2x2	3x3	Np2x2	Np3x3
(383.944890284577, -977.2505612482809)	-1	-1	-1	-1	-1	-1	-1	-1
(484.3016989654416, 878.3875792519711)	1	1	1	1	1	1	1	1
(932.158259147619, 14.070560113649321)	-1	-1	-1	-1	-1	-1	-1	-1
(-183.5398833982647, 775.8155535341004)	1	1	1	1	1	1	1	1
(260.0275788990632, 314.42566998135953)	1	1	1	1	1	1	1	1

Tabela2: Zbiór b)

	Tolerancja 10^{-14}				Tolerancja 10^{-12}			
	2x2	3x3	np2x2	np3x3	2x2	3x3	Np2x2	Np3x3
(7156975058715.969, -40411588513231.85)	-1	-1	-1	-1	-1	-1	-1	-1
(-13219233900958.203, 43387385739829.59)	1	1	1	1	1	1	1	1
(-99240335045315.78, -27685850097775.266)	-1	-1	-1	-1	-1	-1	-1	-1
(-18270489336224.125, 34471373573280.844)	1	1	1	1	1	1	1	1
(-77386523626729.86, -95277236444560.81)	-1	-1	-1	-1	-1	-1	-1	-1

Tabela3: Zbiór c)

	Tolerancja 10^{-14}				Tolerancja 10^{-12}			
	2x2	3x3	np2x2	np3x3	2x2	3x3	Np2x2	Np3x3
(7.664743463496633, 99.70582584602961)	1	1	1	1	1	1	1	1
(79.29931107943382, -60.92306018518099)	-1	-1	-1	-1	-1	-1	-1	-1
(81.981176364554, -57.26331043420291)	-1	-1	-1	-1	-1	-1	-1	-1
(82.9117708978437, 55.907407797030615)	1	1	1	1	1	1	1	1
(28.212229130186955, 95.93784512644544)	1	1	1	1	1	1	1	1

Tabela4: Zbiór d)

	Tolerancja 10^-14	Tolerancja 10^-12							
		2x2	3x3	np2x2	np3x3	2x2	3x3	np2x2	Np3x3
(491.2828500744349, 24.614142503721748)	0	0	-1	0	0	0	0	0	
(144.391520771047, 7.2695760385523505)	1	0	1	0	0	0	0	0	
(346.07670866584795, 17.3538354332924)	-1	0	-1	0	0	0	0	0	
(583.423703139568, 29.221185156978404)	0	0	1	0	0	0	0	0	
(39.10163587552279, 2.0050817937761396)	-1	0	0	0	0	0	0	0	

Jak widzimy pierwsze 3 zbiory zachowują się bardzo podobnie niezależnie od ε oraz niezależnie od użytego znacznika, natomiast już dla 4 zbioru, gdzie są bardziej precyzyjne dane, przy tolerancji 10^{-14} już zaczynają widnieć jakiekolwiek różnice.

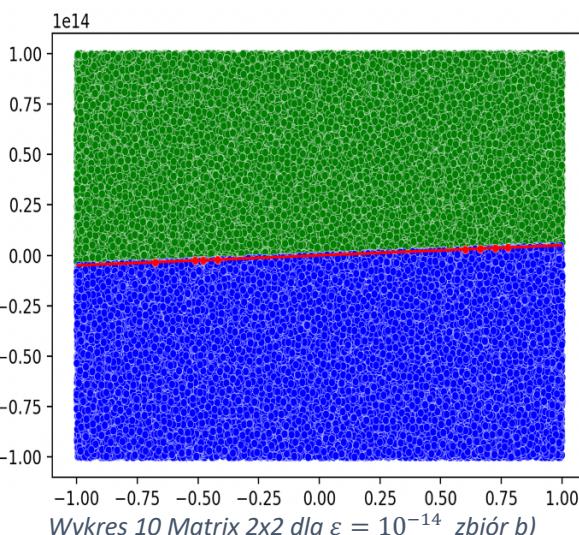
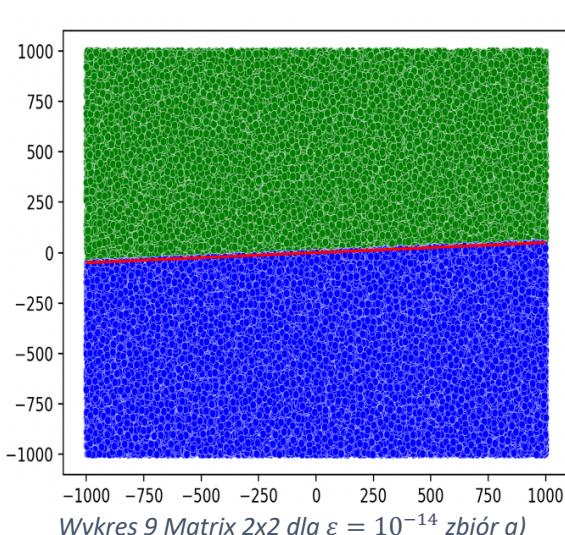
Zbiór a)				Zbiór b)				
	L	S	P		L	S	P	
10^-8	1	49734	0	50266	1	50057	8	49935
	2	49734	0	50266	2	50061	0	49939
	3	49734	0	50266	3	50061	0	49939
	4	49734	0	50266	4	50061	0	49939
10^-10	1	49734	0	50266	1	50057	8	49935
	2	49734	0	50266	2	50061	0	49939
	3	49734	0	50266	3	50061	0	49939
	4	49734	0	50266	4	50061	0	49939
10^-12	1	49734	0	50266	1	50057	8	49935
	2	49734	0	50266	2	50061	0	49939
	3	49734	0	50266	3	50061	0	49939
	4	49734	0	50266	4	50061	0	49939
10^-14	1	49734	0	50266	1	50057	8	49935
	2	49734	0	50266	2	50061	0	49939
	3	49734	0	50266	3	50061	0	49939
	4	49734	0	50266	4	50061	0	49939
10^-16	1	49734	0	50266	1	50057	8	49935
	2	49734	0	50266	2	50061	0	49939
	3	49734	0	50266	3	50061	0	49939
	4	49734	0	50266	4	50061	0	49939

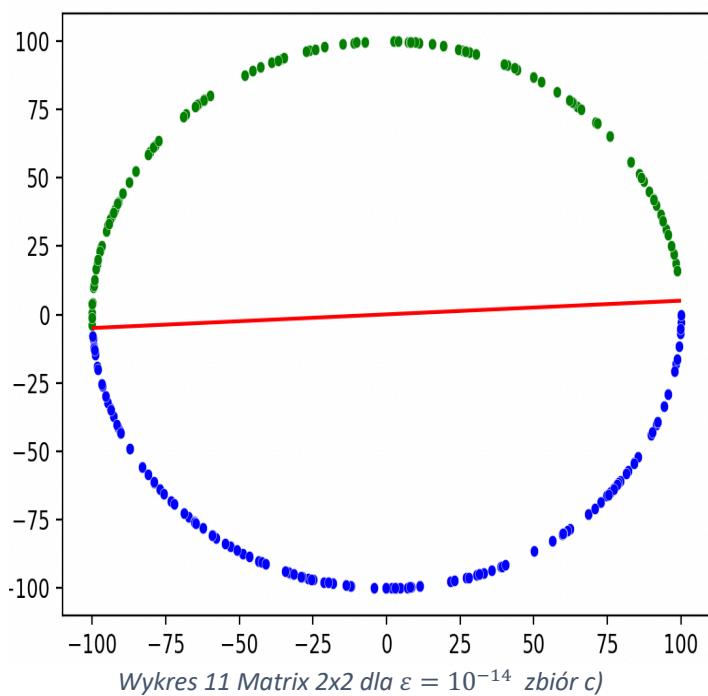
Zbiór c)				Zbiór d)				
		L	S	P		L	S	P
10^-8	1	119	0	131	1	0	1000	0
	2	119	0	131	2	0	1000	0
	3	119	0	131	3	0	1000	0
	4	119	0	131	4	0	1000	0
10^-10	1	119	0	131	1	0	1000	0
	2	119	0	131	2	0	1000	0
	3	119	0	131	3	0	1000	0
	4	119	0	131	4	0	1000	0
10^-12	1	119	0	131	1	75	849	76
	2	119	0	131	2	0	1000	0
	3	119	0	131	3	149	671	180
	4	119	0	131	4	0	1000	0
10^-14	1	119	0	131	1	134	726	140
	2	119	0	131	2	0	1000	0
	3	119	0	131	3	424	84	492
	4	119	0	131	4	47	910	43
10^-16	1	119	0	131	1	139	717	144
	2	119	0	131	2	264	365	371
	3	119	0	131	3	462	5	533
	4	119	0	131	4	481	32	487

Liczby 1,2,3,4 oznaczają kolejno wyznaczniki 2x2, 3x3, numpy2x2, numpy3x3

Litery L, S, P kolejno po lewej, środek, po prawej

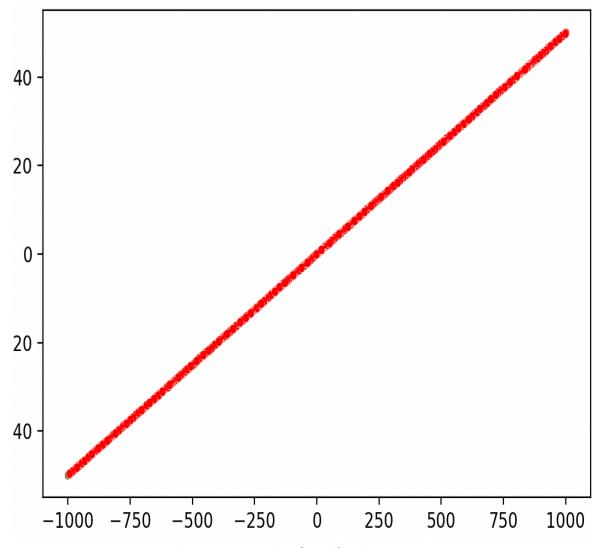
6. Wybranie różnych wykresów i zaprezentowanie tej różnicy na wykresach między tolerancjami



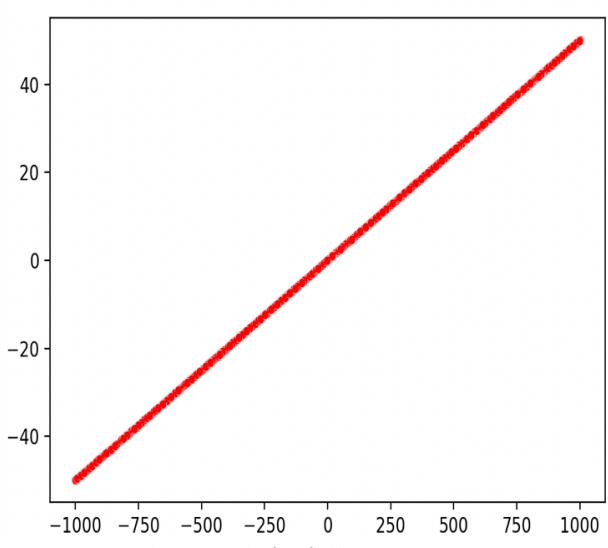


Wykres 11 Matrix 2x2 dla $\varepsilon = 10^{-14}$ zbiór c)

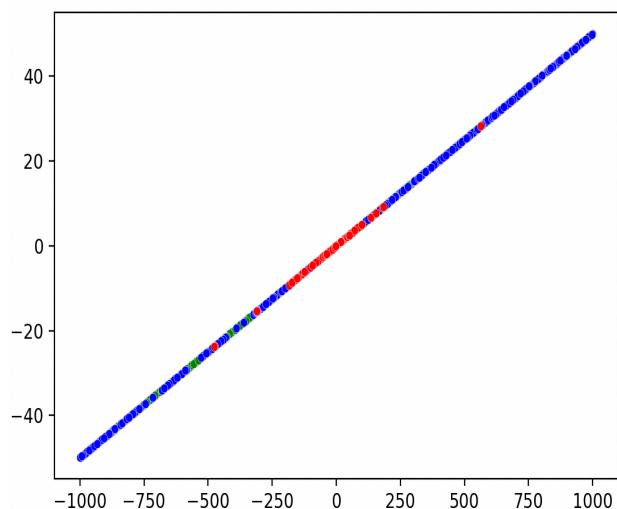
Przykładowe wykresy dla precyzji 10^{-14} dla ostatniego zbioru:



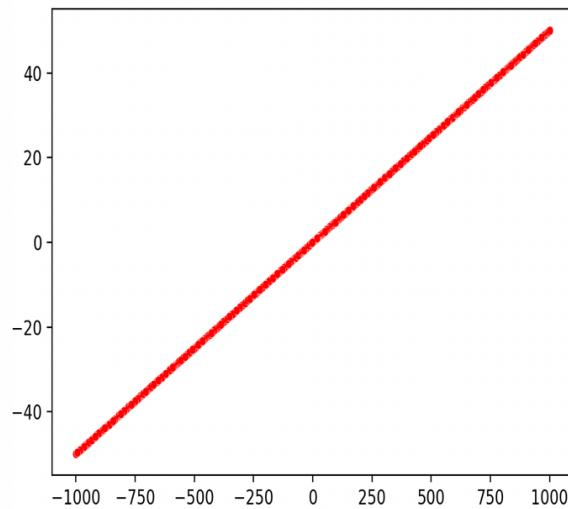
Wykres 12 Zbiór d) dla matrixa 2x2



Wykres 13 Zbiór d) dla matrixa 3x3



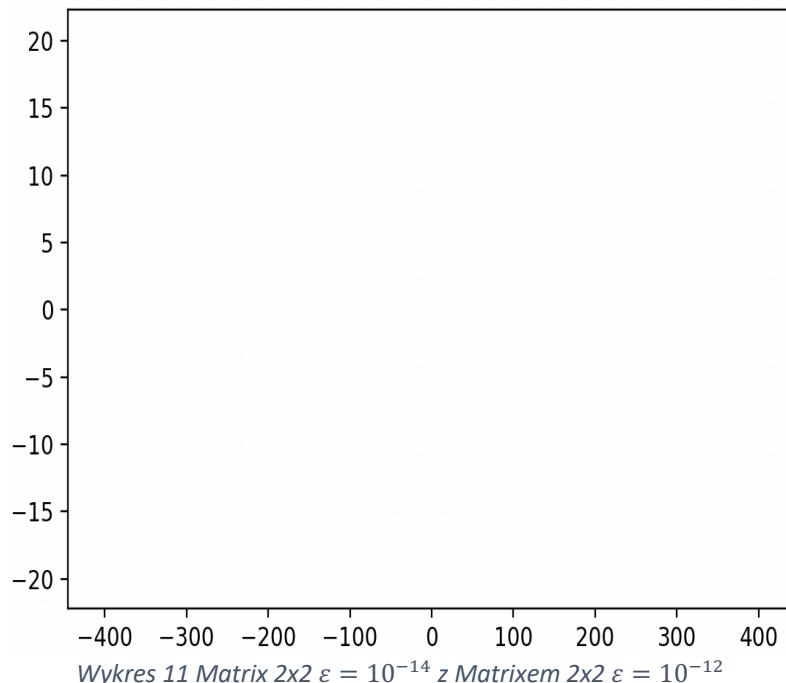
Wykres 14 Zbiór d) dla matrixa numpy 2x2



Wykres 15 Zbiór d) dla matrixa numpy 3x3

Porównywanie takich samych punktów między precyzją 10^{-14} a 10^{-12} , ile takich punktów i jakie są klasyfikowane do różnych miejsc no i również w zależności od użytego wyznacznika.

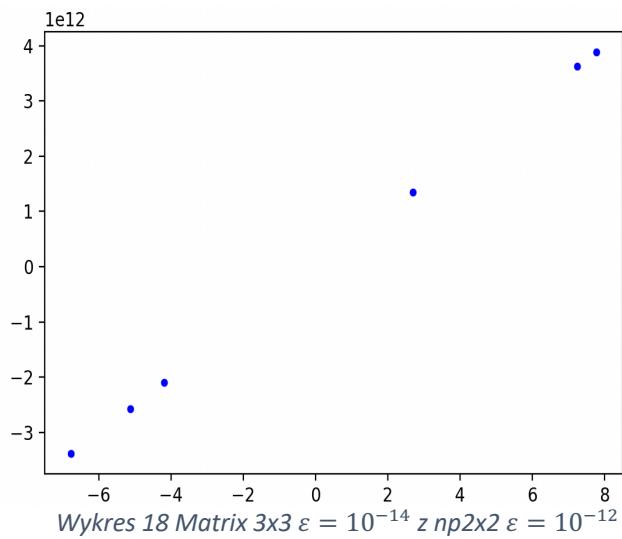
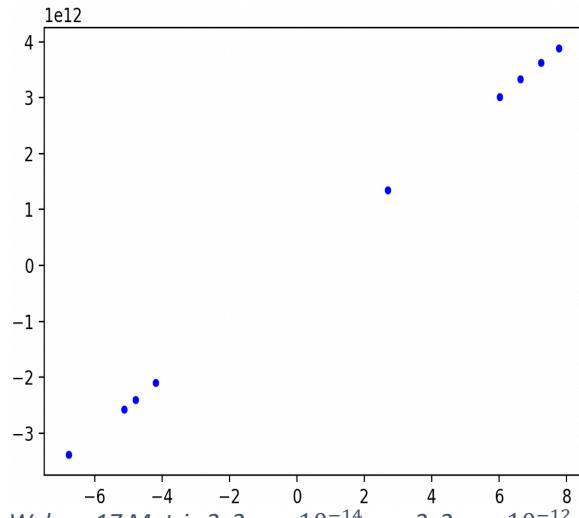
W podpunkcie a) wszystkie punkty zostały sklasyfikowane tak samo więc się nie ma różnic:



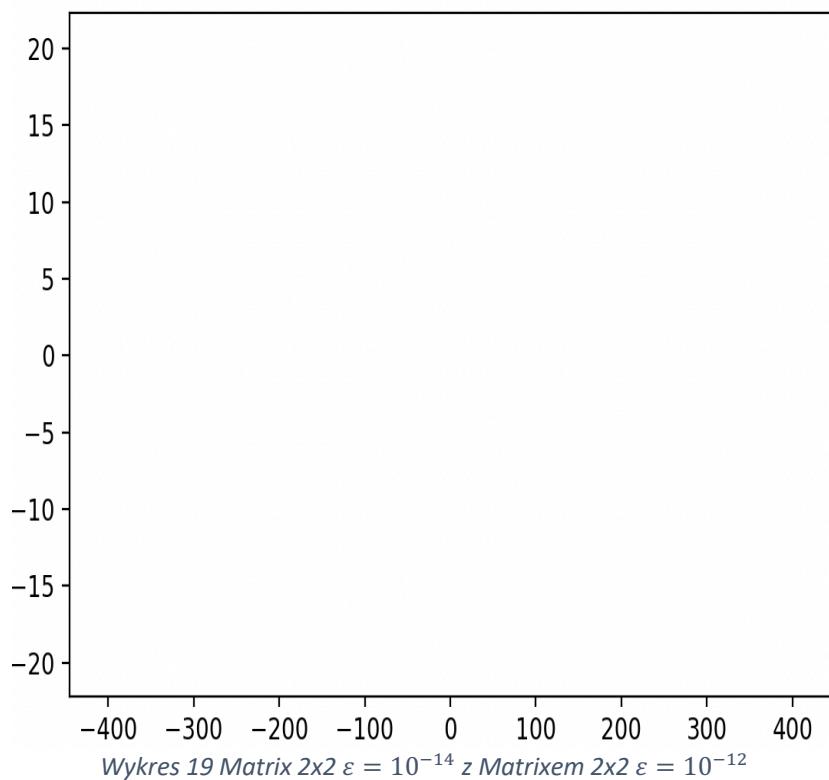
Wykres 11 Matrix 2x2 $\varepsilon = 10^{-14}$ z Matrixem 2x2 $\varepsilon = 10^{-12}$

Uwaga! Puste wykresy nie są błędem one obrazują, że nie ma różnic między precyzjami w danym przykładzie.

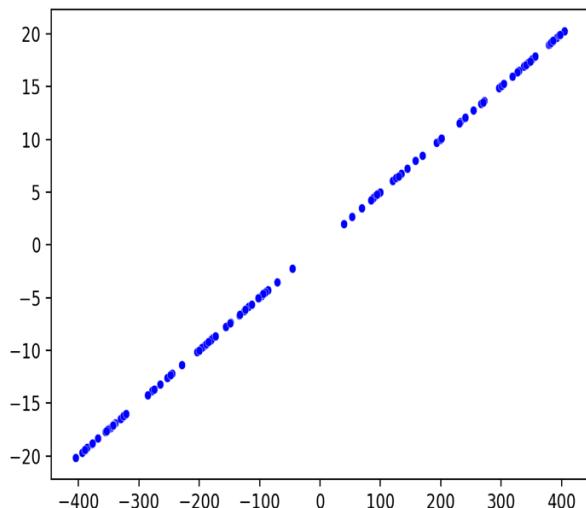
Zbiory b) delikatnie się różnią paroma punktami między matrixami oraz precyzjami



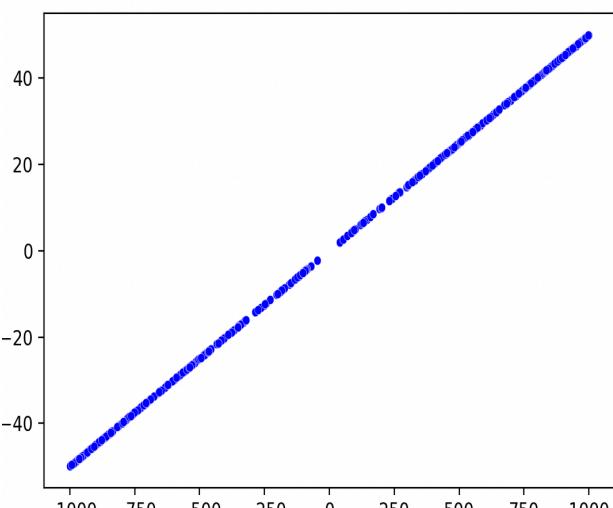
Zbiory c) podobnie jak w punkcie a) się nie zmieniają:



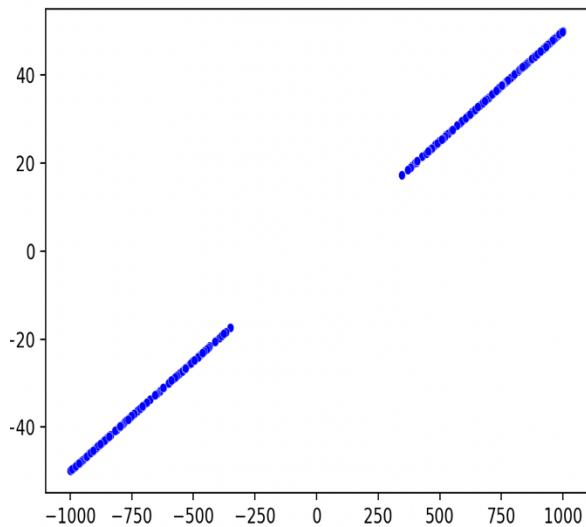
Zaś zbiory d) różnią się dość mocno w zależności od ε oraz wyznacznika



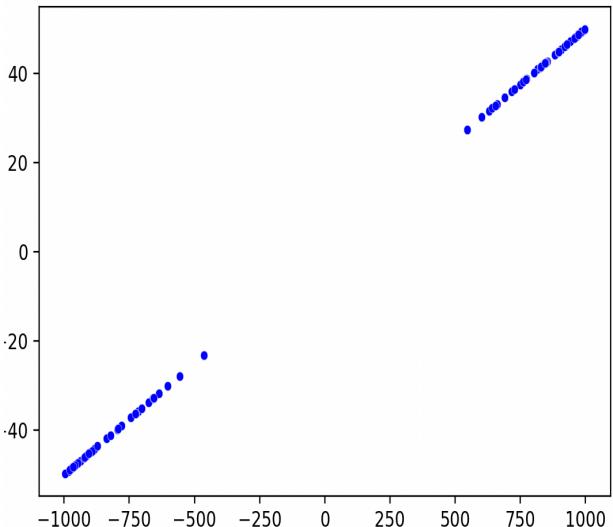
Wykres 20 Matrix $2 \times 2 \varepsilon = 10^{-14}$ z Matrixem $2 \times 2 \varepsilon = 10^{-12}$



Wykres 21 Matrix $2 \times 2 \varepsilon = 10^{-14}$ z $np2 \times 2 \varepsilon = 10^{-12}$



Wykres 22 Matrix $3 \times 3 \varepsilon = 10^{-14}$ z $np2 \times 2 \varepsilon = 10^{-12}$

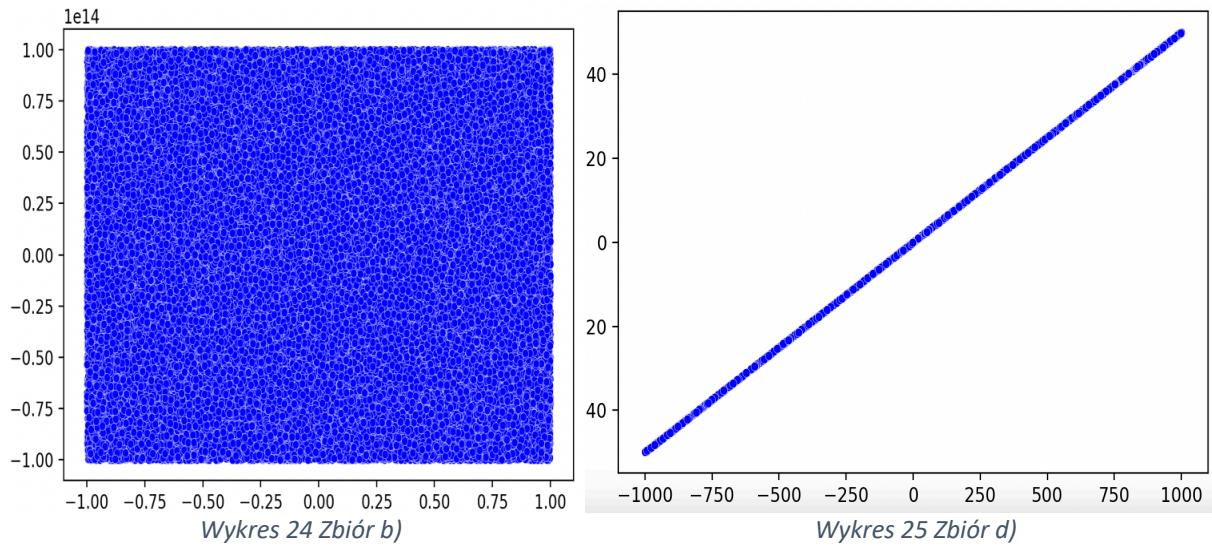


Wykres 23 $np3 \times 3 \varepsilon = 10^{-14}$ z $np3 \times 3 \varepsilon = 10^{-12}$

7. Użycie innej precyzji (float32)

W przypadku precyzji float32 wybrałem wykresy dla podpunktów b) oraz d) gdyż one były najbardziej rozbieżne w poprzednich wynikach, więc one będą się najbardziej różniły utworzyłem do tego tych samych danych.

Wygenerowane wykresy:



Tym razem również sprawdzam w jaki sposób są rozmieszczone przykładowe punkty (Pierwsze 5 punktów)

Legenda: Liczba 1 oznacza, że znajduje się po lewej stronie, liczba -1 po prawej a liczba 0, że leży na prostej.

Liczby 1,2,3,4 oznaczają kolejno wyznaczniki 2×2 , 3×3 , $\text{numpy}2 \times 2$, $\text{numpy}3 \times 3$

Tabela1: Zbiór b)

	Tolerancja 10^{-16}				Tolerancja 10^{-14}				Tolerancja 10^{-12}			
	1	2	3	4	1	2	3	4	1	2	3	4
(7156975300000.0, -40411590000000.0)	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
(-13219234000000.0, 43387386000000.0)	1	1	1	1	1	1	1	1	1	1	1	1
(-99240340000000.0, -27685850000000.0)	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
(-18270489000000.0, 34471374000000.0)	1	1	1	1	1	1	1	1	1	1	1	1
(-77386530000000.0, -95277240000000.0)	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

Tabela2: Zbiór d)

	Tolerancja 10^-16				Tolerancja 10^-14				Tolerancja 10^-12			
	1	2	3	4	1	2	3	4	1	2	3	4
(491.28284,24.614143)	1	1	1	1	1	1	1	1	1	1	1	1
(144.39153,7.269576)	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
(346.07672,17.353836)	-1	0	-1	-1	-1	0	-1	0	0	0	0	0
(583.4237,29.221186)	1	1	1	1	1	1	1	1	1	1	1	1
(39.101635,2.005082)	1	1	1	1	1	1	1	1	1	1	1	1

Jak widzimy te same punkty o innej precyzyji (w tabeli d)) jeśli wróćimy parę stron wcześniej zauważymy ze nie zostały zakwalifikowane do prostej.

Utworzyłem 2 tabelki pokazujące rozmieszczenie wszystkich punktów w zależności od tolerancji i od użytego wyznacznika

Zbiór b)				Zbiór d)				
		L	S	P		L	S	P
10^-8	1	50058	7	49935	1	389	192	419
	2	50061	0	49939	2	389	192	419
	3	50062	0	49938	3	389	192	419
	4	50061	0	49939	4	389	192	419
10^-10	1	50058	7	49935	1	392	189	419
	2	50061	0	49939	2	392	189	419
	3	50062	0	49938	3	392	189	419
	4	50061	0	49939	4	392	189	419
10^-12	1	50058	7	49935	1	392	189	419
	2	50061	0	49939	2	392	189	419
	3	50062	0	49938	3	419	123	458
	4	50061	0	49939	4	392	189	419
10^-14	1	50058	7	49935	1	408	166	426
	2	50061	0	49939	2	392	189	419
	3	50062	0	49938	3	482	15	503

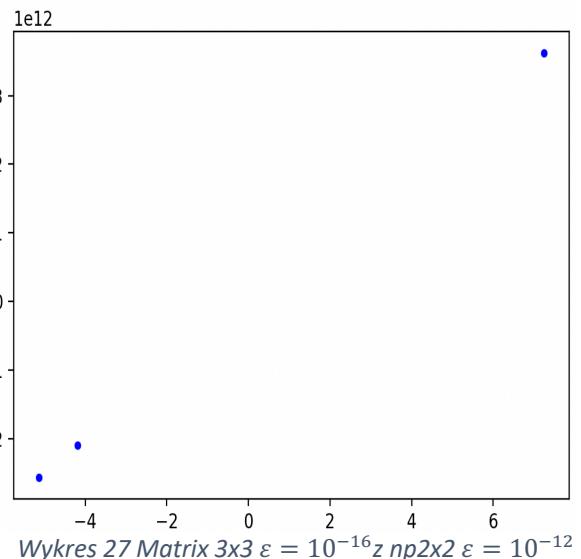
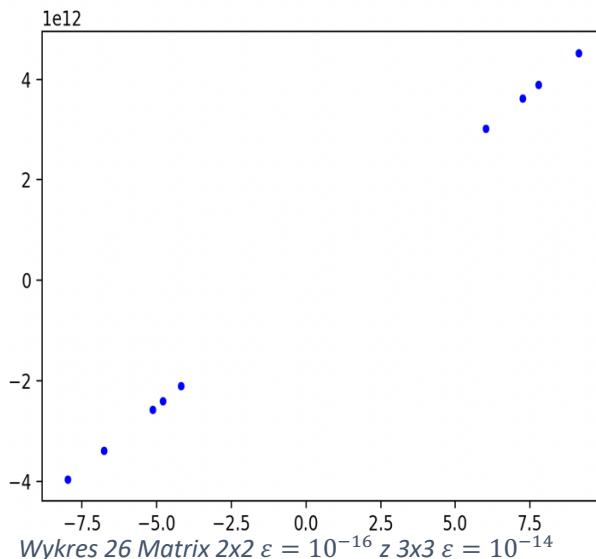
	4	50061	0	49939	4	401	170	429
10^-16	1	50058	7	49935	1	408	165	427
	2	50061	0	49939	2	447	56	497
	3	50062	0	49938	3	489	0	511
	4	50061	0	49939	4	480	3	517

Liczby **1,2,3,4** oznaczają kolejno wyznaczniki **2x2, 3x3, numpy2x2, numpy3x3**
 Litery L, S, P kolejno po lewej, środek, po prawej

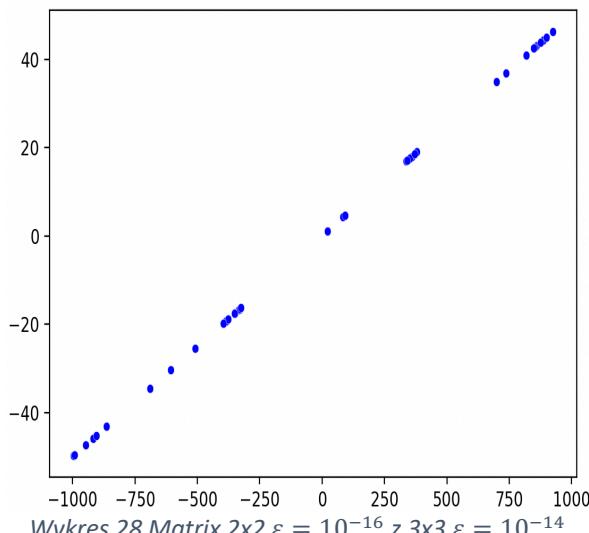
Jak widzimy im większa ta tolerancja tym dużo mniej zakwalifikowanych punktów do prostej. Dodatkowo przy użyciu precyzji float32 dużo mniej punktów w środku jest niż przy użyciu float64, co czyni tę precyzję mniej dokładną.

Porównywanie takich samych punktów między precyzją 10^{-16} a 10^{-14} oraz 10^{-12} , ile takich punktów i jakie są klasyfikowane do różnych miejsc no i również w zależności od użytego wyznacznika.

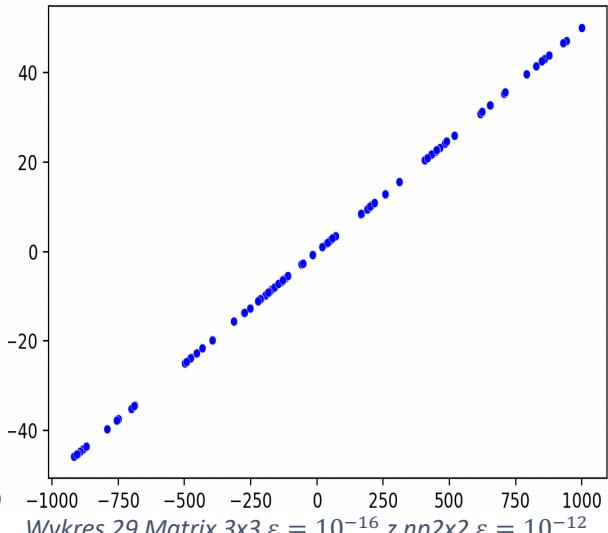
Zbiory b) delikatnie się różnią paroma punktami między matrixami oraz precyzjami



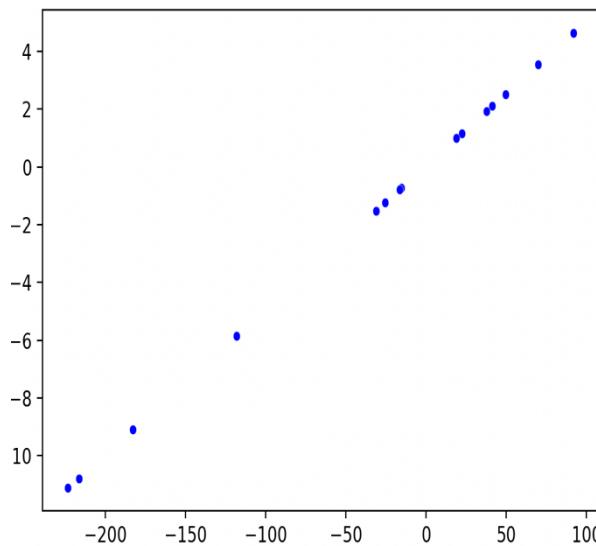
Zbiory d) będą się bardziej różnić



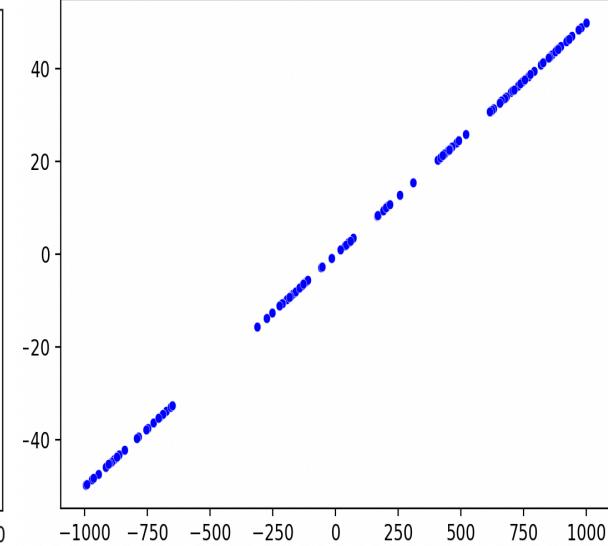
Wykres 28 Matrix 2x2 $\varepsilon = 10^{-16}$ z 3x3 $\varepsilon = 10^{-14}$



Wykres 29 Matrix 3x3 $\varepsilon = 10^{-16}$ z np2x2 $\varepsilon = 10^{-12}$



Wykres 30 Matrix np2x2 $\varepsilon = 10^{-16}$ z np2x2 $\varepsilon = 10^{-14}$



Wykres 31 Matrix 3x3 $\varepsilon = 10^{-16}$ z np3x3 $\varepsilon = 10^{-12}$

Jak widzimy w tym przypadku bardzo losowo są kwalifikowane te punkty to prostej, po zamianie na float32 kompletnie inaczej się zachowuje niż przy obserwacji tego zbioru przy float64

8. Ogólne wnioski:

Klasyfikacja punktów z podpunktów a) i c) w większości przypadków dawała identyczne wyniki i zbytnio one się nie różniły. Istnieje bardzo niskie prawdopodobieństwo ze wygenerowany punkt znajdzie się akurat na prostej przy generowanych losowych wynikach.

Dla przypadku b) zdarzyło się że jakieś punkty znalazły się na prostej, lecz jest ich bardzo mało, tylko przy użyciu własnego matrixa 2×2 . Jest to prawdopodobnie spowodowane, zapisem liczb w komputerze, mantysa jest skończona więc w przypadku niektórych liczb oraz wyznaczników, operacje na liczbach mogą inaczej wyliczać wynik przez co nie są kwalifikowane do prostej. Tolerancja nie wpłynęła na wyniki, a sam sposób liczenia wyznaczników.

Co do przypadku d) tutaj wyniki są mocno rozbieżne, w zależności od wybranej precyzji oraz wybranego sposobu liczenia wyznaczników. Można zauważyć, że najbardziej dokładne są wyniki przy użyciu tolerancji $\varepsilon \geq 10^{-14}$.

W zależności od użytej metody liczenia, dość mocno się różnią punkty w jaki sposób są kwalifikowane po jakiej stronie lub czy należą do prostej.

Na niektórych wykresach można zauważyć, że ilość punktów zmienia się w zależności od wartości współrzędnych x i y. Im większe te wartości (bezwzględnie) tym bardziej widać różnice między punktami. Możemy przez to stwierdzić, że najlepiej wyznaczniki działają dla punktów o mniejszej wartości (bezwzględnie).

W przypadku zamiany precyzji na float32 wyniki są dużo gorsze i dużo mniej punktów klasyfikuje do prostej. Najlepsze wyniki dawały wyznaczniki zaimplementowane zwłaszcza wyznacznik 2×2 , a najgorzej wyznacznik z biblioteki numpy2x2.

Ostatecznie można stwierdzić, że dużo lepiej jest użyć precyzji float64 oraz własnych zaimplementowanych wyznaczników, poza tym biblioteczne są wolniejsze.