1 Wstęp

Celem ćwiczenia była implementacja, testy i wizualizacja podstawowego predykatu geometrycznego - określenia po której stronie prostej znajduje się punkt.

1.1 Wstęp teoretyczny

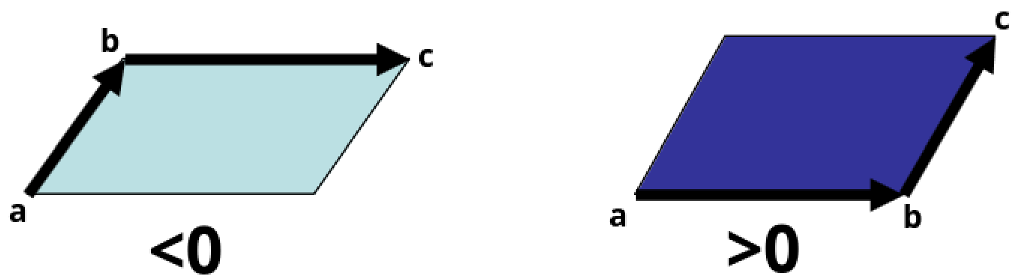
Niech punkty $a = (a_x, a_y)$ i $b = (b_x, b_y)$ wyznaczają jednoznacznie prostą k.

Pozycja punktu $c = (c_x, c_y)$ względem k zależy od wyznacznika $det(a, b, c)$, który możemy wyliczyć na dwa sposoby:

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

Zależność jest następująca:

$$
\begin{aligned}
det(a,b,c) > 0 &\iff c \text{ leży po lewej } k \\
det(a,b,c) < 0 &\iff c \text{ leży po prawej } k \\
det(a,b,c) = 0 &\iff c \text{ leży na } k
\end{aligned} \tag{2}
$$



Rysunek 1: Wizualizacja zależności jako pola równoległoboku

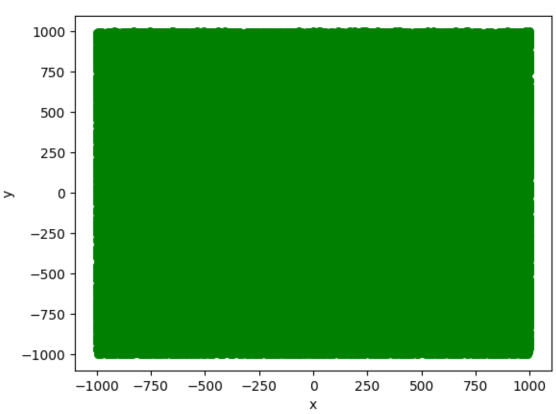
1.2 Specyfikacja narzędzi i sprzętu

Wszystkie potrzebne obliczenia zostały wykonane przy użyciu interpretera języka Python wersji 3.12. Ponadto, w celu wygenerowania zbiorów punktów i sprawdzenia bibliotecznych funkcji wyliczania wyznacznika, użyłem biblioteki `numpy`. Wykresy i wizualizacja wyników została przygotowana za pomocą narzędzia przygotowanego przez koło naukowe Bit. Kod znajduje się w załączonym pliku.

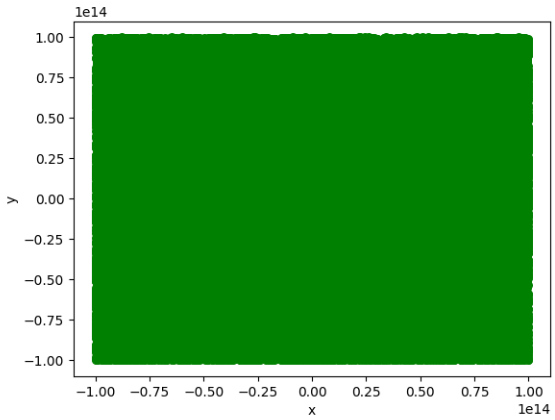
Przedstawione wyniki zostały wygenerowane na komputerze z systemem operacyjnym Debian 12 i procesorem Intel Core i5-8250U 1.6 GHz.

2 Realizacja obliczeń

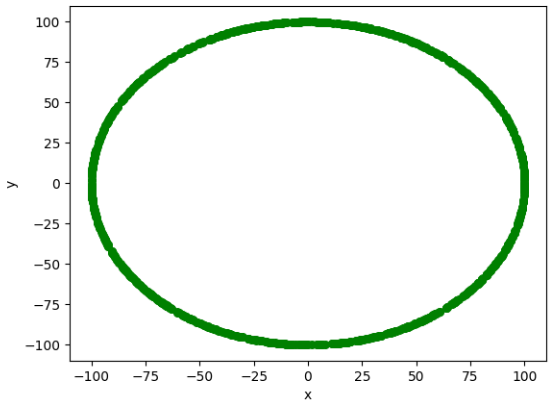
Obliczenia rozpocząłem od przygotowania czterech zbiorów losowych punktów:



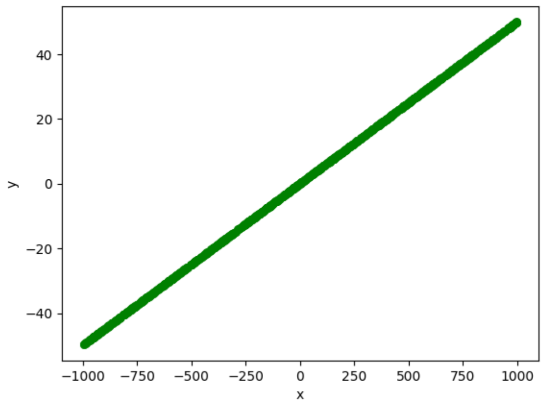
(i) Zbiór A
10^5 punktów o współrzędnych
z przedziału $[-1000, 1000]$.



(ii) Zbiór B
10^5 punktów o współrzędnych
z przedziału $[-10^{14}, 10^{14}]$.



(iii) Zbiór C
1000 punktów na okręgu: $x^2 + y^2 = 100$.



(iv) Zbiór D
1000 punktów o współrzędnej $x \in [-1000, 1000]$,
leżących na prostej wyznaczonej przez punkty
$a = (-1, 0)$, $b = (1, 0.1)$.

Rysunek 2: Wygenerowane zbiory

Następnie każdy z tych zbiorów podzieliłem względem prostej wyznaczonej przez punkty $a = (-1, 0)$, $b = (1, 0.1)$ korzystając z różnych implementacji liczenia wyznacznika, tolerancji dla zera i precyzji liczb zmiennoprzecinkowych. Funkcje, używane przeze mnie do liczenia wyznacznika, to:

- `mat_det_2x2`

  (implementacja w załączonym pliku)

- `mat_det_2x2_lib`

  (funkcja biblioteczna `numpy.linalg.det`)

- `mat_det_3x3`

  (implementacja w załączonym pliku)

- `mat_det_3x3_lib`

  (funkcja biblioteczna `numpy.linalg.det`)

3 Wyniki

Tabele 1, 2, 3, 4 i Rysunki 3, 4, 5, 6 przedstawiają wyniki podziałów punktów dla każdego zbioru w zależności od funkcji, tolerancji ($\varepsilon = 0, 10^{-14}, 10^{-12}, 10^{-10}, 10^{-8}$) i precyzji zmiennych (`float64` lub `float32`). Za wyniki w tabelach przyjmuję ilość punktów zakwalifikowanych kolejno po lewej, po prawej i na prostej.

Na wykresach punkty leżące na lewo i prawo od prostej będą zaznaczane kolejno na zielono i pomarańczowo, a punkty leżące na prostej - na fioletowo. Sama prosta jest zaznaczona na czerwono.

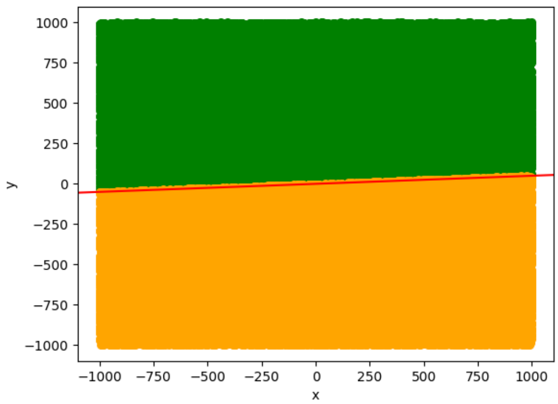
3.1 Zbiór A

Funkcja	**Po lewej**	**Po prawej**	**Na prostej**	**Po lewej**	**Po prawej**	**Na prostej**
$\varepsilon = 0, 1\text{e}{-14}, \ldots$	`float64`			`float32`		
`mat_det_3x3`	50084	49916	0	50084	49916	0
`mat_det_3x3_lib`	50084	49916	0	50084	49916	0
`mat_det_2x2`	50084	49916	0	50084	49916	0
`mat_det_2x2_lib`	50084	49916	0	50084	49916	0

Tabela 1: Rozkład punktów dla zbioru A

Dla każdej sprawdzonej tolerancji, precyzji i funkcji wyniki podziału są identyczne i nie występują żadne błędy wynikające z reprezentacji liczb rzeczywistych za pomocą liczb zmiennoprzecinkowych.

Na Rysunku 3 znajduje się wykres podziału punktów, który jest taki sam dla wszystkich sprawdzonych przypadków:



Rysunek 3: Wykres rozkładu punktów w zbiorze A

3.2 Zbiór B

Funkcja	**Po lewej**	**Po prawej**	**Na prostej**	**Po lewej**	**Po prawej**	**Na prostej**
$\varepsilon = 0$	`float64`			`float32`		
`mat_det_3x3`	50120	49880	0	50120	49880	0
`mat_det_3x3_lib`	50120	49880	0	50120	49880	0
`mat_det_2x2`	50116	49876	8	0	0	100000
`mat_det_2x2_lib`	50116	49875	9	6545	6621	86834
$\varepsilon = 1\text{e}{-14}$	`float64`			`float32`		
`mat_det_3x3`	50120	49880	0	50120	49880	0
`mat_det_3x3_lib`	50120	49880	0	50120	49880	0
`mat_det_2x2`	50116	49876	8	0	0	100000
`mat_det_2x2_lib`	50116	49875	9	6545	6621	86834
$\varepsilon = 1\text{e}{-12}$	`float64`			`float32`		
`mat_det_3x3`	50120	49880	0	50120	49880	0
`mat_det_3x3_lib`	50120	49880	0	50120	49880	0
`mat_det_2x2`	50116	49876	8	0	0	100000
`mat_det_2x2_lib`	50116	49875	9	6545	6621	86834
$\varepsilon = 1\text{e}{-10}$	`float64`			`float32`		
`mat_det_3x3`	50120	49880	0	50120	49880	0
`mat_det_3x3_lib`	50120	49880	0	50120	49880	0
`mat_det_2x2`	50116	49876	8	0	0	100000
`mat_det_2x2_lib`	50116	49875	9	6545	6621	86834
$\varepsilon = 1\text{e}{-8}$	`float64`			`float32`		
`mat_det_3x3`	50120	49880	0	50120	49880	0
`mat_det_3x3_lib`	50120	49880	0	50120	49880	0
`mat_det_2x2`	50116	49876	8	0	0	100000
`mat_det_2x2_lib`	50116	49875	9	6545	6621	86834

Tabela 2: Rozkład punktów dla zbioru B

Jak widać w Tabeli 2, wyniki są niezależne od przyjętej tolerancji. Jest tak, ponieważ prawdopodobieństwo znalezienia losowego punktu w zbiorze B na naszej prostej jest astronomicznie małe. Możemy z tego wywnioskować, że zakwalifikowanie jakichkolwiek punktów jako leżących na prostej, wynika z błędu i niewystarczającej precyzji reprezentacji liczb w komputerze.

Pominę wyniki dla funkcji liczących wyznacznik macierzy 3×3, ponieważ dzielą one nasz zbiór, tak jakbyśmy się tego spodziewali. Najciekawsze wyniki widzimy dla wyznacznika 2×2, w którym pojawiają się pierwsze błędy. Dla precyzji `float64` występuje ich nieznaczna ilość, lecz dla `float32` wyniki są kompletnie błędne.

Weźmy dla przykładu punkt $c = (-4.507\,465\,4\text{e}13, -4.607\,413\,6\text{e}13)$. Podążając algorytmem na obliczanie wyznacznika macierzy 2×2:

$$\begin{aligned}
a_x - c_x &= -1 - (-4.507\,465\,4\text{e}13) =_{\texttt{float32}} 4.507\,465\,4\text{e}13 \\
b_y - c_y &= 0.1 - (-4.607\,413\,6\text{e}13) =_{\texttt{float32}} 4.607\,413\,6\text{e}13 \\
a_y - c_y &= 0 - (-4.607\,413\,6\text{e}13) =_{\texttt{float32}} 4.607\,413\,6\text{e}13 \\
b_x - c_x &= 1 - (-4.507\,465\,4\text{e}13) =_{\texttt{float32}} 4.507\,465\,4\text{e}13 \\
(a_x - c_x)(b_y - c_y) &- (a_y - c_y)(b_x - c_x) =_{\texttt{float32}} 0
\end{aligned} \tag{3}$$

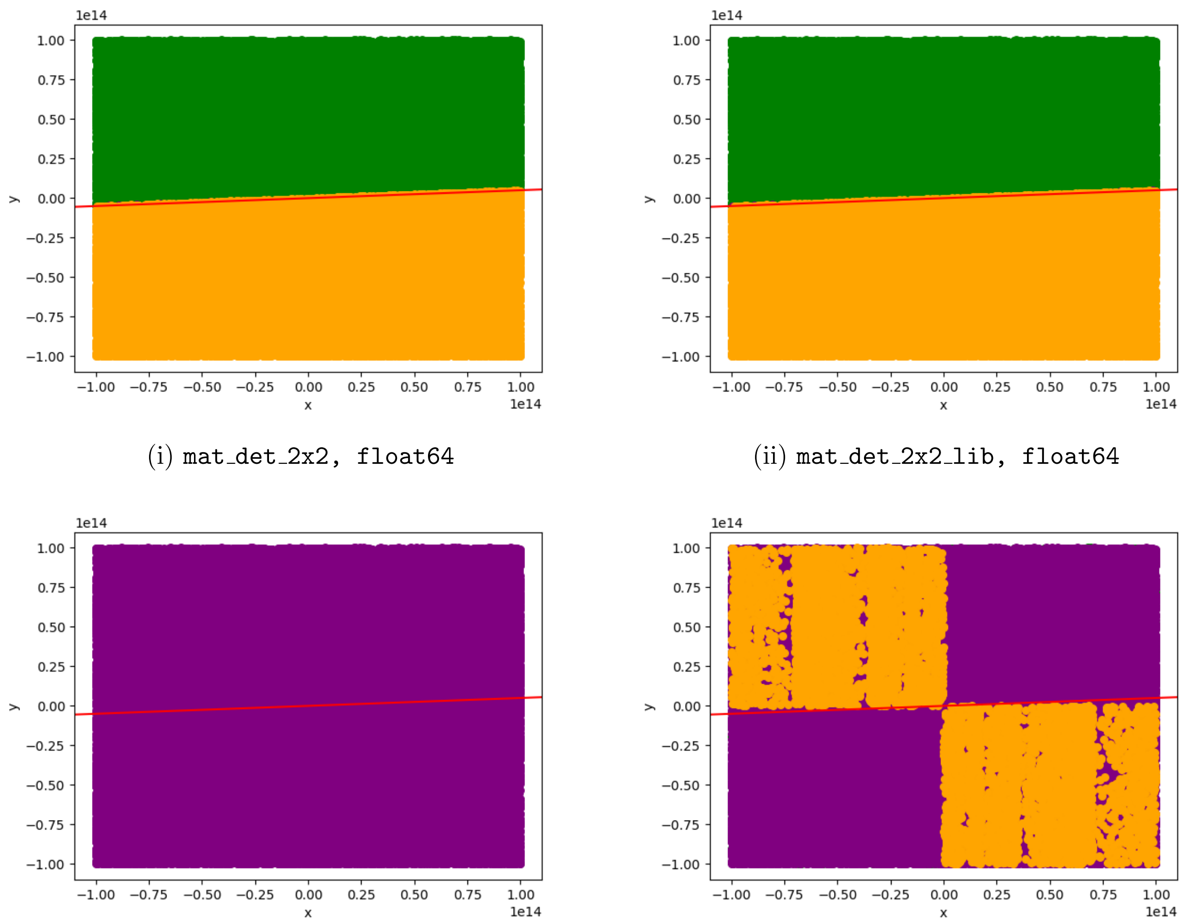
Widzimy więc, że reprezentowalne wartości wokół $\pm 10^{13}$, są dla precyzji `float32` na tyle rzadkie, iż nie jesteśmy w stanie rozróżnić bliskich siebie wartości.

Dla tego samego punktu funkcja biblioteczna zwraca jednak inną wartość:

$$\underset{\text{(punkt zakwalifikowany po lewej)}}{\texttt{mat\_det\_2x2\_lib}(a, b, c) = 3.521\,457\,4\text{e}11 > 0} \tag{4}$$

Wynika to prawdopodobnie z innej implementacji liczenia wyznacznika macierzy i wewnętrznych optymalizacji.

Na Rysunku 4 znajdują się podziały, w których występiły błędy:



(i) `mat_det_2x2, float64` (ii) `mat_det_2x2_lib, float64`

(iii) `mat_det_2x2, float32` (iv) `mat_det_2x2_lib, float32`

Rysunek 4: Wykresy wybranych rozkładów punktów w zbiorze B

3.3 Zbiór C

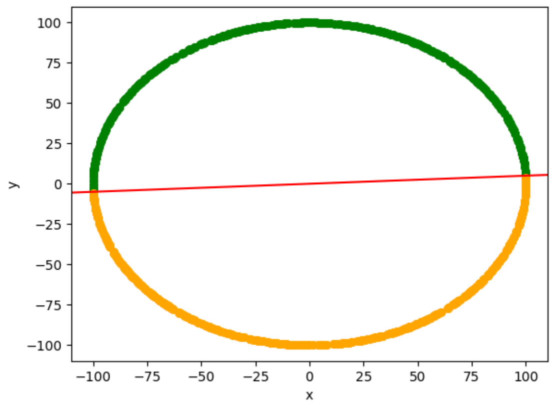
Funkcja	**Po lewej**	**Po prawej**	**Na prostej**	**Po lewej**	**Po prawej**	**Na prostej**
$\varepsilon = 0, 1\text{e}{-14}, \ldots$	`float64`			`float32`		
`mat_det_3x3`	484	516	0	484	516	0
`mat_det_3x3_lib`	484	516	0	484	516	0
`mat_det_2x2`	484	516	0	484	516	0
`mat_det_2x2_lib`	484	516	0	484	516	0

Tabela 3: Rozkład punktów dla zbioru C

Podobnie jak dla zbioru A, wszystkie wyniki są identyczne.

Wartości współrzędnych zarówno w tym zbiorze, jak i zbiorze A, są małe, więc wyniki operacji zmiennoprzecinkowych na nich są dokładniejsze. Ponadto nie sprawdzamy dokładnej wartości wyznacznika, lecz tylko jego znak, co jeszcze bardziej ułatwia nam zdobycie poprawnych wyników.

Na Rysunku 5 znajduje się wykres podziału punktów, który jest taki sam dla wszystkich sprawdzonych przypadków:



Rysunek 5: Wykres rozkładu punktów w zbiorze B

3.4 Zbiór D

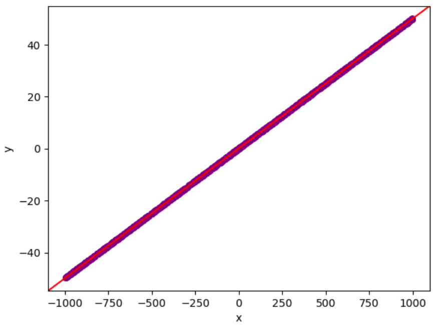
Funkcja	**Po lewej**	**Po prawej**	**Na prostej**	**Po lewej**	**Po prawej**	**Na prostej**
$\varepsilon = 0$	`float64`			`float32`		
`mat_det_3x3`	269	384	347	339	290	371
`mat_det_3x3_lib`	344	286	370	504	442	54
`mat_det_2x2`	156	152	692	191	178	631
`mat_det_2x2_lib`	166	161	673	510	490	0
$\varepsilon = 1\mathrm{e}{-14}$	`float64`			`float32`		
`mat_det_3x3`	0	0	1000	339	290	371
`mat_det_3x3_lib`	29	40	931	441	387	172
`mat_det_2x2`	149	148	703	191	178	631
`mat_det_2x2_lib`	157	153	690	510	490	0
$\varepsilon = 1\mathrm{e}{-12}$	`float64`			`float32`		
`mat_det_3x3`	0	0	1000	339	290	371
`mat_det_3x3_lib`	0	0	1000	431	374	195
`mat_det_2x2`	84	91	825	191	178	631
`mat_det_2x2_lib`	114	105	781	510	490	0
$\varepsilon = 1\mathrm{e}{-10}$	`float64`			`float32`		
`mat_det_3x3`	0	0	1000	339	290	371
`mat_det_3x3_lib`	0	0	1000	431	374	195
`mat_det_2x2`	0	0	1000	191	178	631
`mat_det_2x2_lib`	0	0	1000	510	490	0
$\varepsilon = 1\mathrm{e}{-8}$	`float64`			`float32`		
`mat_det_3x3`	0	0	1000	337	290	373
`mat_det_3x3_lib`	0	0	1000	429	373	198
`mat_det_2x2`	0	0	1000	191	178	631
`mat_det_2x2_lib`	0	0	1000	508	488	4

Tabela 4: Rozkład punktów dla zbioru D

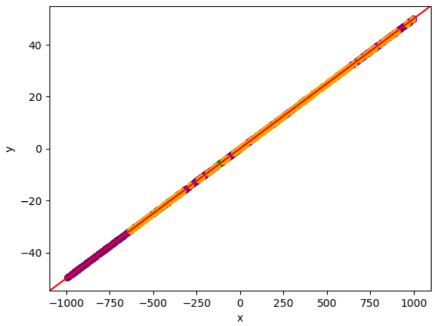
Zbiór D zawiera tylko wartości leżące na omawianej przez nas prostej. Z tego powodu, w teorii, wszystkie punkty powinny zostać zakwalifikowane jako leżące na prostej. Jak widzimy jednak w Tabeli 4, idealne wyniki udało się uzyskać jedynie dla relatywnie dużych tolerancji i precyzji. Jest tak, ponieważ ilość reprezentowalnych wartości zmiennoprzecinkowych rośnie blisko zera. Sprawia to, że z większym prawdopodobieństwem wyznaczniki wyliczane są jako bardzo bliskie zeru, co wymusza na nas zwiększenie tolerancji.

Dla precyzji `float32` natomiast wszystkie wyniki są bardzo zbliżone do siebie, niezależnie od tolerancji. Wynika to oczywiście z jeszcze większej niedokładności obliczeń. Uzyskanie idealnego podziału wymagałoby więc od nas zwiększenia tolerancji.

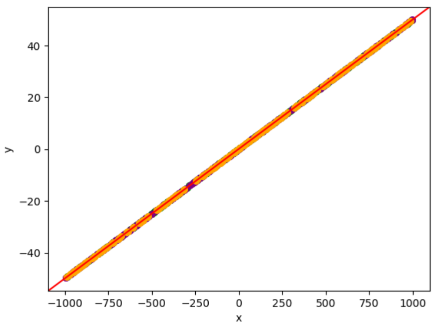
Na Rysunku 6 znajdują się wykresy wybranych przeze mnie rozkładów, które moim zdaniem najlepiej obrazują różne niedokładności w wynikach.



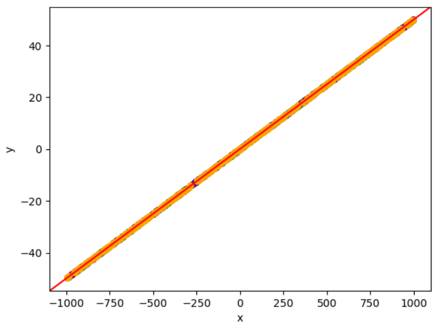
(i) `mat_det_3x3 float64` $\varepsilon = 1\mathrm{e}{-10}$



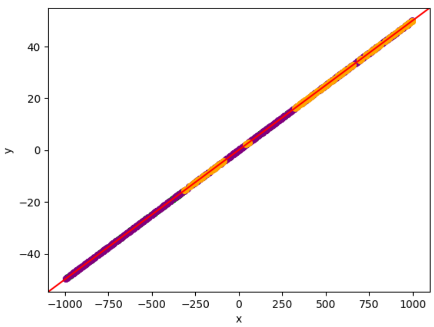
(ii) `mat_det_3x3 float32` $\varepsilon = 1\mathrm{e}{-10}$



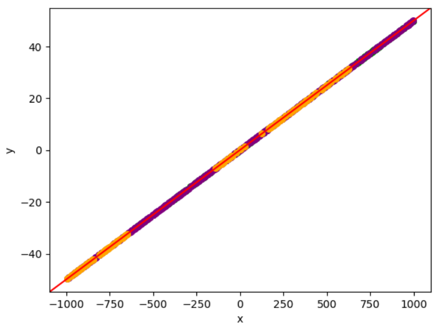
(iii) `mat_det_3x3_lib float64` $\varepsilon = 0$



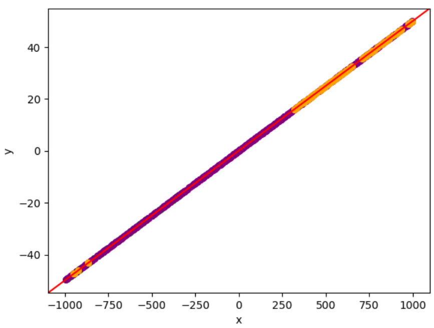
(iv) `mat_det_3x3_lib float32` $\varepsilon = 1\mathrm{e}{-10}$



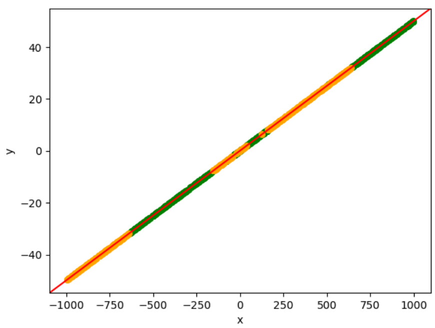
(v) `mat_det_2x2 float64` $\varepsilon = 1\mathrm{e}{-14}$



(vi) `mat_det_2x2 float32` $\varepsilon = 1\mathrm{e}{-10}$



(vii) `mat_det_2x2_lib float64` $\varepsilon = 1\mathrm{e}{-12}$



(viii) `mat_det_2x2_lib float32` $\varepsilon = 1\mathrm{e}{-10}$

Rysunek 6: Wykresy wybranych rozkładów punktów w zbiorze D

4 Podsumowanie

Na podstawie przeprowadzonych obliczeń jesteśmy w stanie wywnioskować, że dobranie tolerancji dla zbiorów A, B i C nie zmieniało w żaden sposób wyniku. W zbiorze B najważniejsza okazała się precyzja i dobór funkcji. Dla wyznaczników 2×2 nawet precyzja `float64` oznaczała kilka błędów, natomiast dla `float32` wyniki były kompletnie błedne. Zbiór D okazał się najbardziej czuły na zmiany w tolerancji, która musiała być całkiem duża, żeby wyniki były w pełni poprawne.

Ostatecznie najbardziej dokładne wyniki zwracały funkcje liczące wyznacznik macierzy 3×3, z czego najlepiej poradziła sobie funkcja `mat_det_3x3` będąca najprostszą implementacją liczenia wyznacznika metodą Sarrusa. Wyjątkiem jest zbiór D przy użyciu precyzji `float32`, gdzie najbliżej poprawnych wyników była funkcja `mat_det_2x2`.