**Ćwiczenie I - Predykaty Geometryczne**

**Jakub Frączek - grupa nr 4**

**1. Wstęp**

**Opis ćwiczenia**

Ćwiczenie polegało na wygenerowaniu 4 różnych zbiorów punktów, a następnie sklasyfikowaniu ich ze względu na to po której stronie prostej się znajdują.

**Dane techniczne - software**

Ćwiczenie zostało zrealizowane w języku Python przy użyciu środowiska Jupyter notebook. Wykorzystane biblioteki to: numpy, random, pandas, matplotlib, bitalg.

**Dane techniczne - hardware**

Laptop z systemem operacyjnym Linux Mint, procesorem AMD Ryzen 5 5500U 2.1 GHZ oraz 8 GB pamięci RAM.

**2. Generacja punktów**

**Wygenerowane zbiory**

1. 105 losowych punktów o współrzędnych z przedziału [-1000, 1000]

2. 105 losowych punktów o współrzędnych z przedziału [-1014, 1014]

3. 1000 losowych punktów leżących na okręgu o środku (0,0) i promieniu R=100

4. 1000 losowych punktów o współrzędnych z przedziału [-1000, 1000] leżących na prostej wyznaczonej przez wektor (a, b), gdzie a = [-1.0, 0.0], b = [1.0, 0.1].

**Sposób generowania**

Punkty zostały wygenerowane za pomocą funkcji random.uniform() z biblioteki random. W przypadku punktów leżących na okręgu skorzystałem z równań okręgu zadanego parametrycznie:

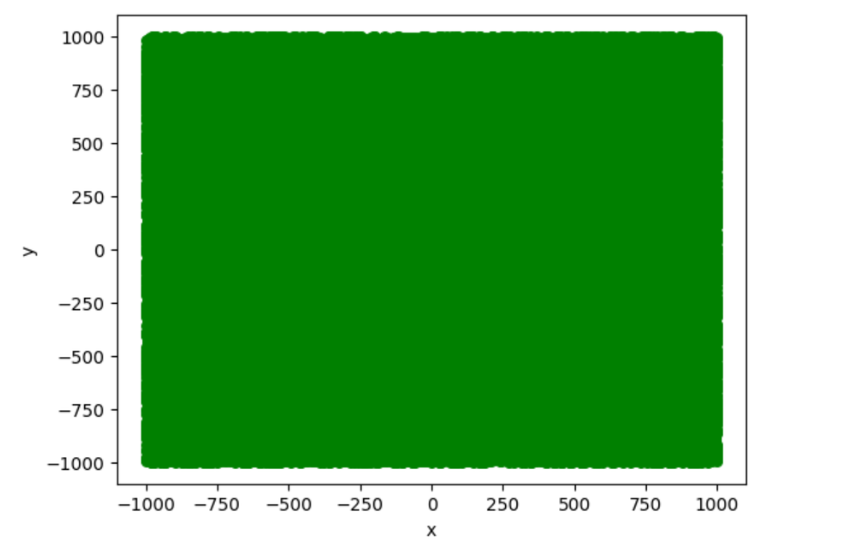
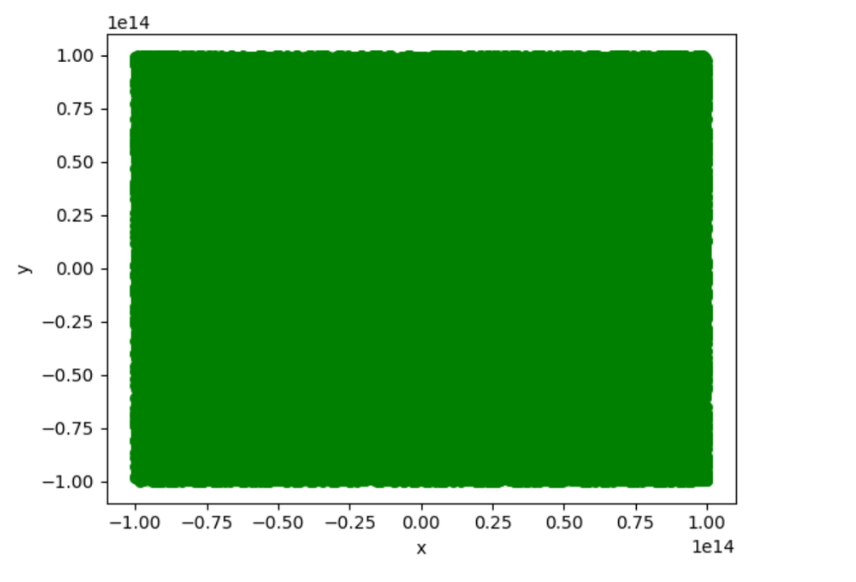
W przypadku punktów leżących na prostej skorzystałem ze wzoru na prostą w postaci kierunkowej

**Wizualizacja wygenerowanych zbiorów**

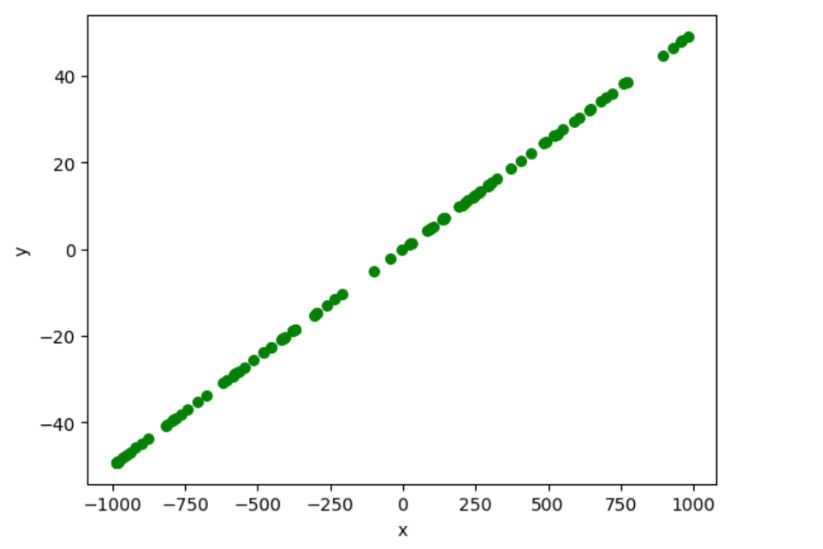
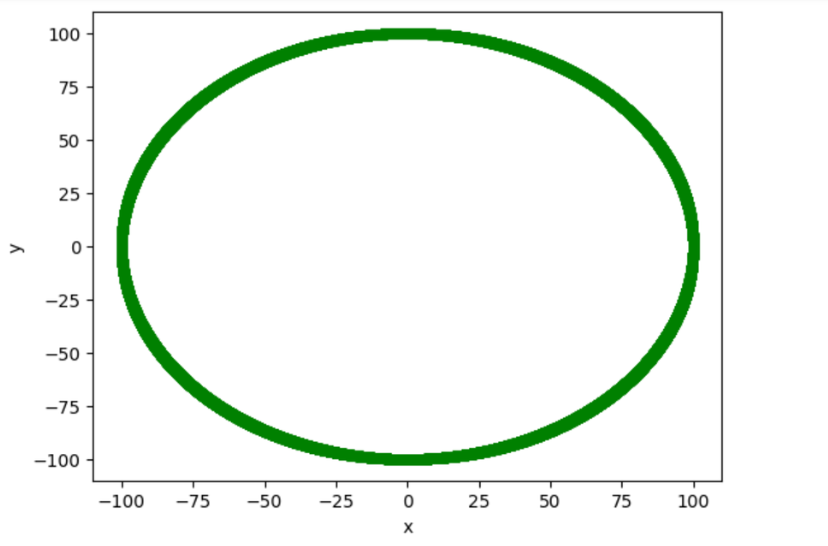
Wizualizacja została zrealizowana przy pomocy biblioteki bitalg napisanej przez koło naukowe BIT.

**Wykresy**

Poniższe wykresy (Wykres 1, Wykres 2, Wykres 3, Wykres 4) obrazują wygenerowane punkty:

****

Wykres 1. Zbiór 1 Wykres 2. Zbiór 2.



Wykres 3. Zbiór 3. Wykres 4. Zbiór 4.

**3. Określenie po której stronie znajduje się punkt**

**Sposób określania**

Położenie punktu względem prostej można łatwo określić obliczając iloczyn wektorowy . Przez punkty a i b przechodzi prosta, a punkt c jest tym którego położenie chcemy określić. Ta metoda jest równoważna z wyliczeniem wyznacznika:

macierzy 2 x 2:

|  |  |  |  |
| --- | --- | --- | --- |

lub macierzy 3x3:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |

Przyjąłem, że jeśli det(A) < 0 to punkt leży po prawej stronie prostej, det(A) > 0 po lewej, a det(A) = 0 na prostej.

**Sposoby wyliczania wyznaczników**

Zaimplementowane zostały 4 funkcje:

1. mat\_det\_3x3() - wyliczająca wyznacznik macierzy 3x3 metodą Sarrusa

2. mat\_det\_3x3\_lib() - wyliczająca wyznacznik macierzy 3x3 przy użyciu funkcji linalg.det() z biblioteki numpy

3. mat\_det\_2x2() - wyliczająca wyznacznik macierzy 2x2 mnożąc wyrazy macierzy stojące na przekątnej, a następnie odejmując je od siebie

4. mat\_det\_2x2\_lib() - wyliczająca wyznacznik macierzy 2x2 przy użyciu funkcji linalg.det() z biblioteki numpy

**4. Analiza danych**

**Sposób analizy**

Dane ze zbiorów 1-4 zostały przeanalizowane wszystkimi 4 sposobami obliczania wyznaczników, przy użyciu dwóch różnych precyzji typu danych float (float64, float32), a także w zależności od zbioru, różnych wartości tolerancji, czyli dokładności z jaką klasyfikuje punkt jako leżący na prostej. Pod każdą tabelą znajdują się wybrane przeze mnie najciekawsze wykresy. Zielony kolor na wykresie oznacza punkty na lewo od prostej, pomarańczowy na prawo, a fioletowy punkty leżące na prostej. Gwiazda w tabeli oznacza, że każda użyta metoda dała taki sam efekt.

**Dane ze zbioru 1.**

Dla każdej wybranej metody wyniki były takie same. Są one przedstawione w Tabeli 1. Przetestowane epsilony to: 10-15, 10-10, 10-5.

| Wyznacznik | Precyzja float'a | Epsilon | Punkty po lewej | Punkty na prostej | Punkty po prawej |
| --- | --- | --- | --- | --- | --- |
| \* | \* | \* | 49787 | 0 | 50213 |

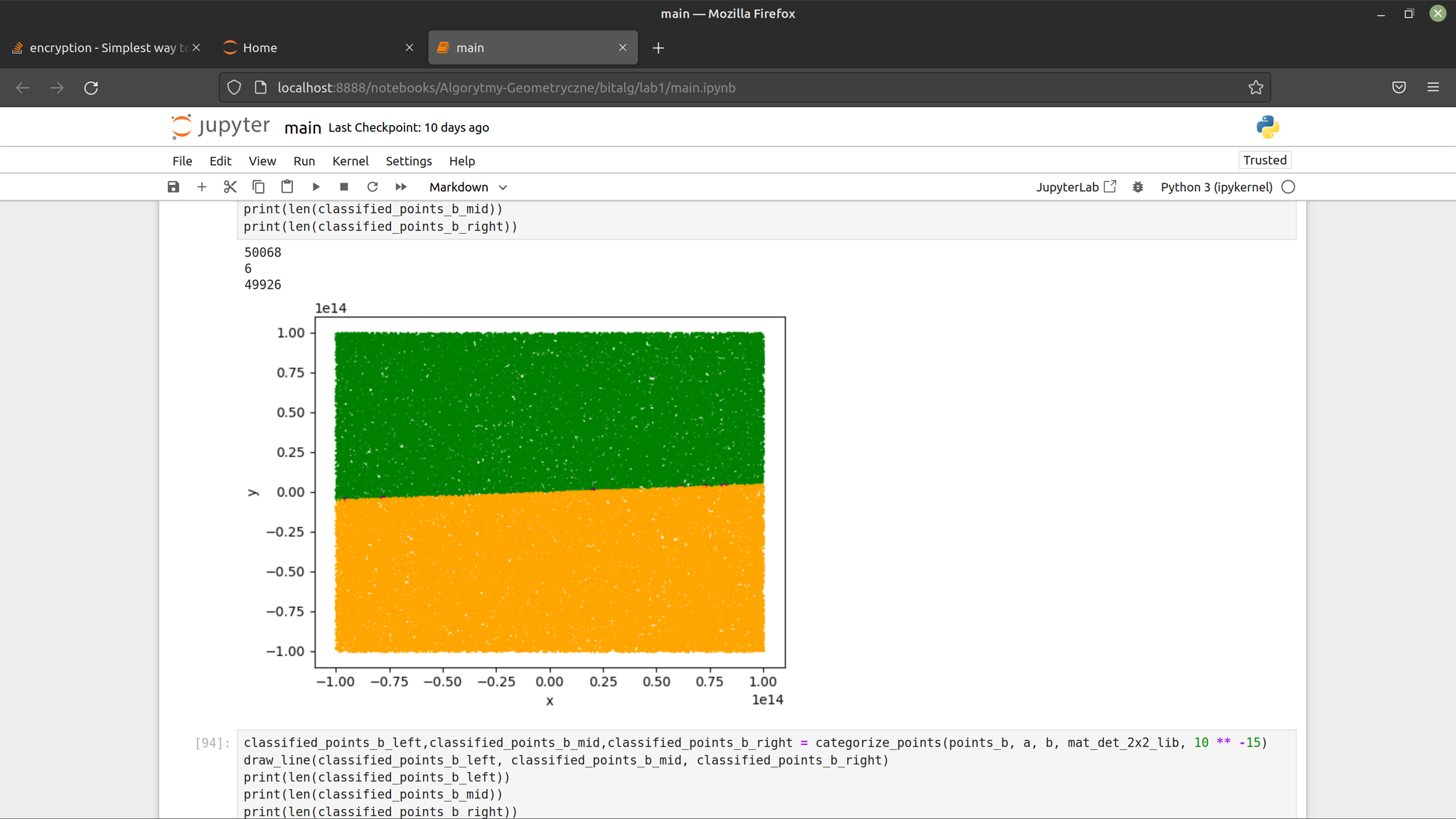
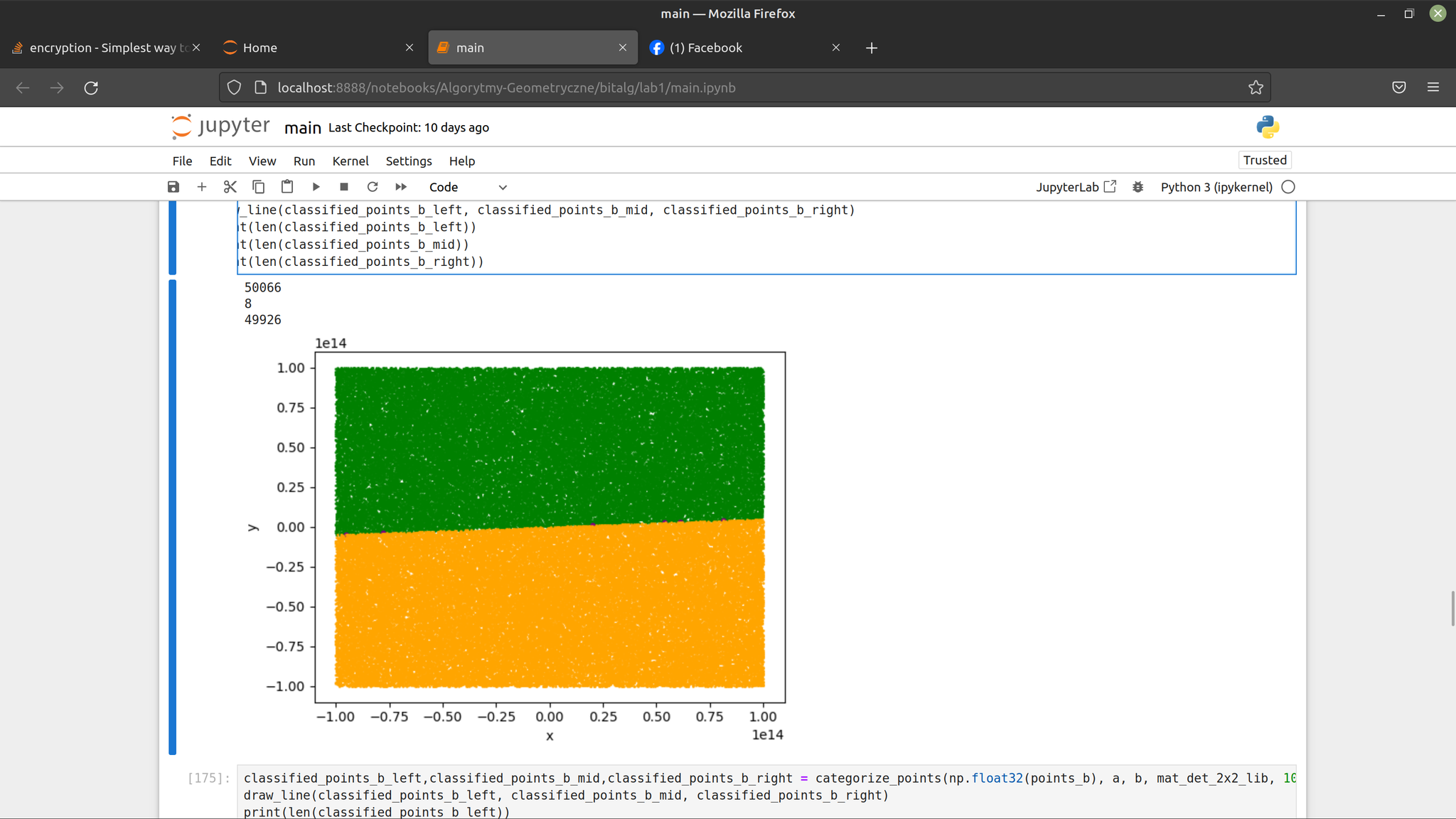
Tabela 1. Klasyfikacja danych ze zbioru 1.

**Dane ze zbioru 2.**

Wyznacznik 3x3 okazał się nieskuteczny przy wykryciu punktów leżących na prostej. Odmienne rezultaty dał wyznacznik 2x2 są one przedstawione w Tabeli 2. Po ręcznym przetestowaniu co zwracają oba wyznaczniki okazało się, że wyznacznik 2x2 zwraca dokładnie 0, a wyznacznik 3x3 bardzo dużą, lub bardzo małą liczbę w zależności od punktu. Pod tabelą (wykres 6. i wykres 7.) znajduje się graficzne porównanie dla dwóch różnych precyzji floata i funkcji mat\_det\_2x2. Przetestowane epsilony to: 10-100, 10-15, 10-10, 10-5.

| Wyznacznik | Precyzja float'a | Epsilon | Punkty po lewej | Punkty na prostej | Punkty po prawej |
| --- | --- | --- | --- | --- | --- |
| mat\_det\_2x2 | 64 bity | \* | 50068 | 6 | 49926 |
| 32 bity | \* | 50066 | 8 | 49926 |
| mat\_det\_2x2\_lib | 64 bity | \* | 50066 | 6 | 49928 |
| 32 bity | \* | 50067 | 6 | 49927 |

Tabela 2. Klasyfikacja danych ze zbioru 2.



Wykres 6. mat\_det\_2x2, eps = 10-15, float64 Wykres 7. mat\_det\_2x2, eps = 10-15, float32

**Dane ze zbioru 3.**

Podobnie jak w przypadku zbioru 1. nie udało się trafić punkty na prostej oraz metody dały ten sam wynik zaprezentowany w Tabeli 3. Przetestowane epsilony to: 10-15, 10-10, 10-5.

| Wyznacznik | Precyzja float'a | Epsilon | Punkty po lewej | Punkty na prostej | Punkty po prawej |
| --- | --- | --- | --- | --- | --- |
| \* | \* | \* | 50240 | 0 | 49760 |

Tabela 3. Klasyfikacja danych ze zbioru 3.

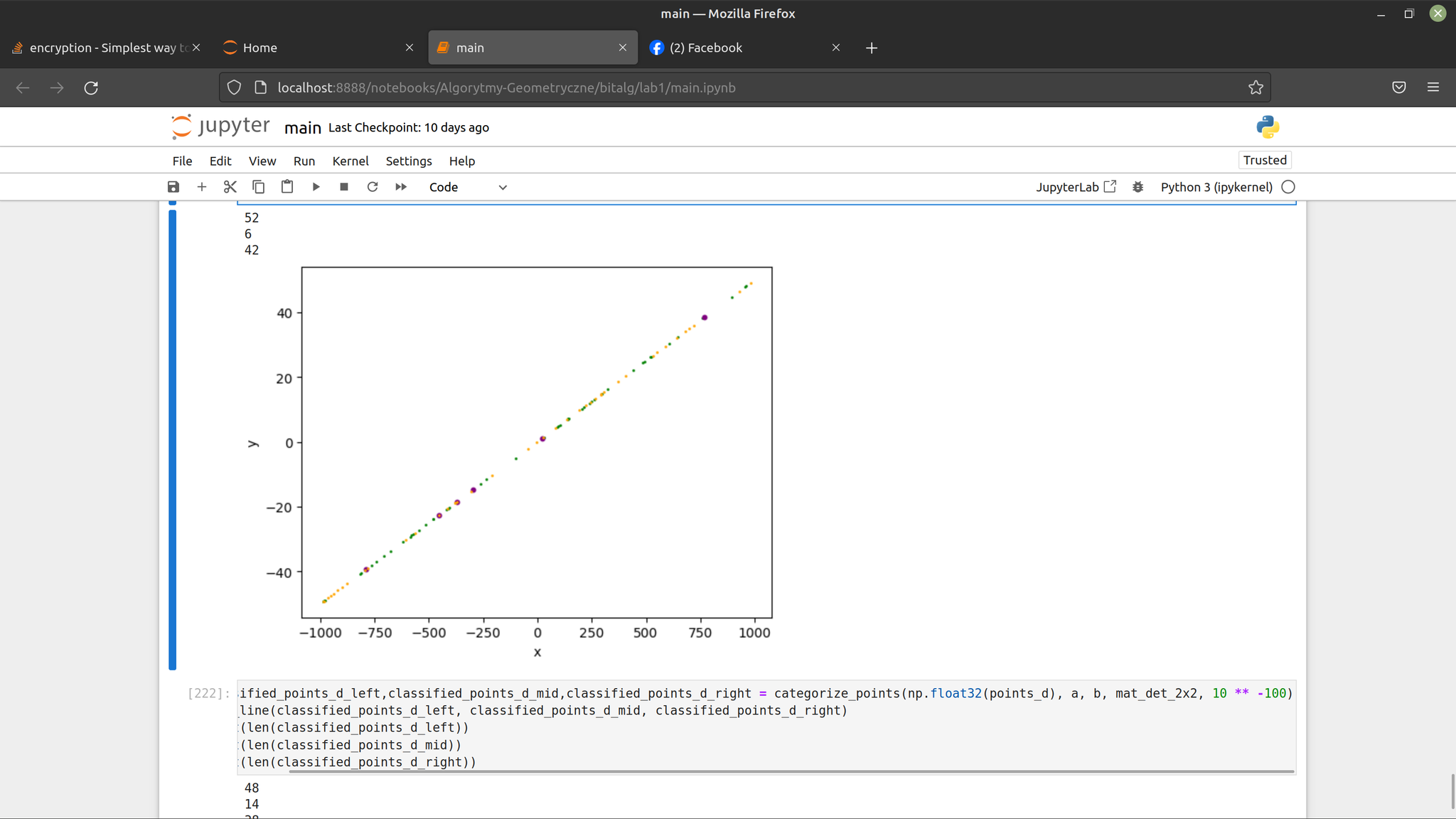
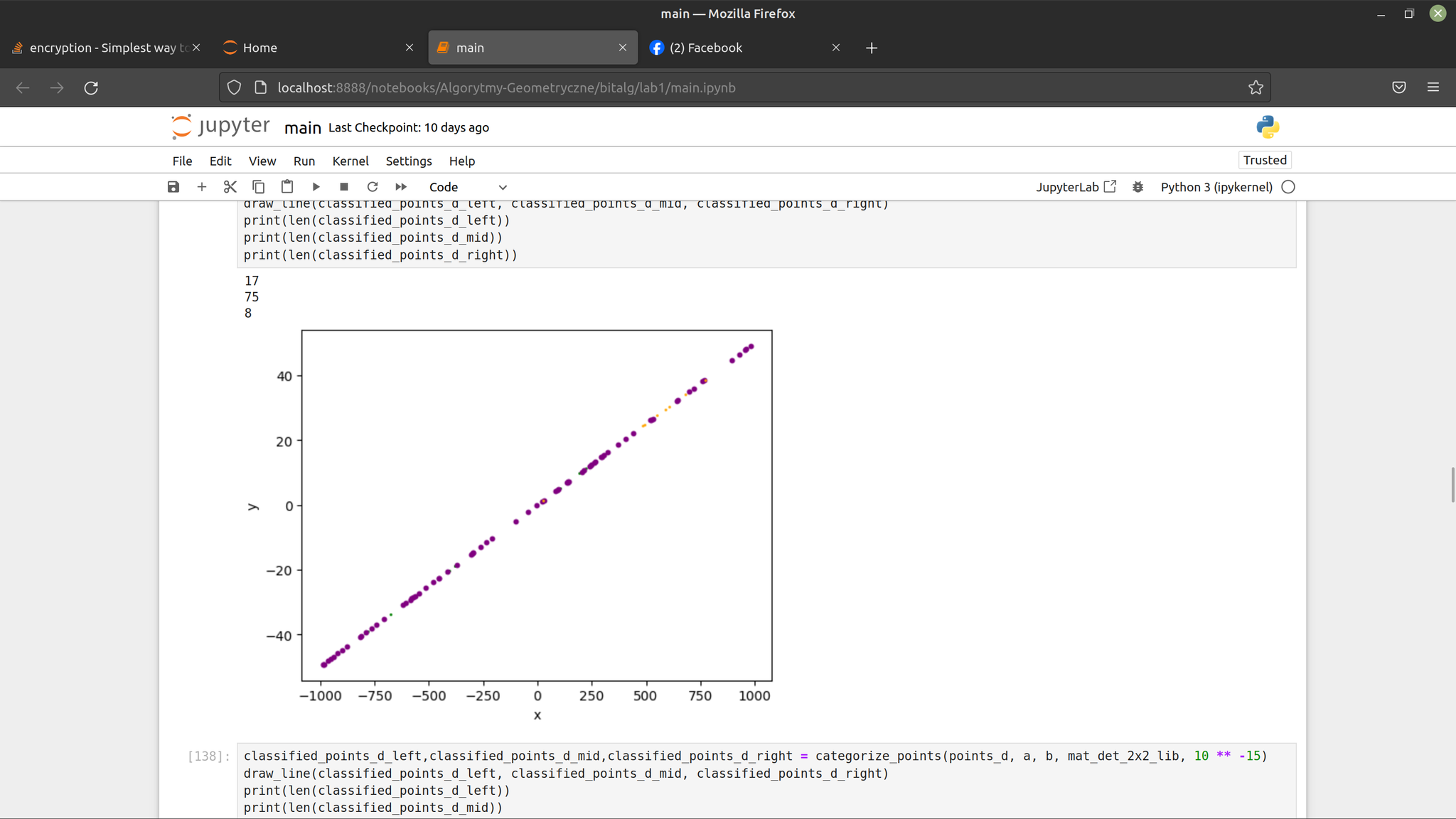
**Dane ze zbioru 4.**

Rezultaty dla zbioru 4. są zdecydowanie najciekawsze. Każda użyta metoda dała inne wyniki dla różnych tolerancji i precyzji. W tym wypadku punkty mimo, że zostały wygenerowane na prostej nie wszystkie zostały zakwalifikowane jako takowe. Dla tolerancji= 10^-10 wszystkie punkty zostały określone jako leżące na prostej. Dane zostały zestawione w tabeli 4. Pod tabelą na wykresie 8. przedstawiony jest wynik dla najmniej skutecznej metody, a na wykresie 9. dla optymalnej moim zdaniem funkcji mat\_det\_2x2 i tolerancji 10-15.

**Wybrane wykresy dla danych ze zbioru 4.**

| Zestaw 4 | | | | | |
| --- | --- | --- | --- | --- | --- |
| Wyznacznik | Precyzja float'a | Epsilon | Punkty po lewej | Punkty na prostej | Punkty po prawej |
| mat\_det\_3x3 | 64 bity | 10-10 | 0 | 100 | 0 |
| 10-15 | 20 | 46 | 34 |
| 10-20 | 20 | 44 | 36 |
| 10-100 | 20 | 44 | 36 |
| 32 bity | 10-10 | 46 | 17 | 37 |
| 10-15 | 51 | 7 | 42 |
| 10-20 | 51 | 7 | 42 |
| 10-100 | 51 | 7 | 42 |
| mat\_det\_3x3\_lib | 64 bity | 10-10 | 0 | 100 | 0 |
| 10-15 | 35 | 33 | 32 |
| 10-20 | 42 | 25 | 33 |
| 10-100 | 42 | 25 | 33 |
| 32 bity | 10-10 | 46 | 17 | 37 |
| 10-15 | 52 | 7 | 41 |
| 10-20 | 52 | 6 | 42 |
| 10-100 | 52 | 6 | 42 |
| mat\_det\_2x2 | 64 bity | 10-10 | 0 | 100 | 0 |
| 10-15 | 17 | 75 | 8 |
| 10-20 | 18 | 74 | 8 |
| 10-100 | 18 | 74 | 8 |
| 32 bity | 10-10 | 46 | 17 | 37 |
| 10-15 | 48 | 14 | 38 |
| 10-20 | 48 | 14 | 38 |
| 10-100 | 48 | 14 | 38 |
| mat\_det\_2x2\_lib | 64 bity | 10-10 | 0 | 100 | 0 |
| 10-15 | 26 | 60 | 14 |
| 10-20 | 26 | 60 | 14 |
| 10-100 | 26 | 60 | 14 |
| 32 bity | 10-10 | 46 | 17 | 37 |
| 10-15 | 51 | 7 | 42 |
| 10-20 | 48 | 12 | 40 |
| 10-100 | 48 | 12 | 40 |

Tabela 4. Klasyfikacja danych ze zbioru 4.

Wykres 8. mat\_det\_3x3\_lib, eps = 10-100, float32 Wykres 9. mat\_det\_2x2, eps = 10-15, float64

**5. Wnioski**

Dane ze zbiorów 1 i 3 dały takie same wyniki, nie udało się wygenerować żadnego punktu leżącego na prostej. Próbowałem eksperymentować z dość dużą wartością tolerancji tj. 10-1 i udało się uzyskać 3 punkty w przypadku zbioru 1 i 33 punkty w przypadku zbioru 3, które zostały sklasyfikowane jako leżące na prostej. Jednak nie można powiedzieć, że leżą one na tej prostej, a jedynie w pobliżu. W przypadku obu zbiorów, szansa że jakiś punkty wypadnie dokładnie na prostej była bardzo znikoma.

W zbiorze 2. udało się wygenerować kilka punktów leżących na prostej. Mogę być tego pewny ponieważ zarówno wyznacznik 2x2 obliczony moją metodą jak i metodą biblioteczną niezależnie dla jakiej tolerancji dawał niemal te same wyniki (6 pkt. lub 8pkt. dla mat\_det\_2x2 i float32). Należy to uznać za duże szczęście ponieważ zbiór 2. zawierał 105 punktów z przedziału [-104, 104] dużo większego niż zbiór 1.

Zdecydowanie najciekawsze rezultaty otrzymałem dla zbioru 4. Co ciekawe dla tolerancji 10-10 wszystkie punkty zostały określone jako leżące na prostej niezależnie od sposobu sprawdzenia. Jednak wraz z wzrostem tolerancji malała ich liczba. Zmiana z 64 bitowego float’a na 32 bitowy również znacznie zmniejszyła liczbę tych punktów. Najbardziej zaskoczyło mnie, że dla float32 zaledwie 17 / 100 punktów leżało na prostej, gdzie dla tej samej tolerancji (10-10) wszytkie na niej leżały. Dzieje się tak ponieważ przez taką zmianę znacznie traci się na precyzji. Przy takich obliczeniach liczby daleko po przecinku mają znaczenie, a ich utrata wiążę się z błędnym określeniem pozycji punktu.

W moim przypadku najlepsze wyniki dała funkcja obliczająca wyznacznik macierzy 2x2

zaimplementowana przeze mnie, a optymalną tolerancją jest 10-15.