Akademia Górniczo-Hutnicza
Wydział Informatyki, Elektroniki i Telekomunikacji
Kierunek Informatyka

Algorytmy geometryczne

Laboratorium 1

Ćwiczenie wprowadzające

Kyrylo Iakymenko
Czwartek 13:00 - 14:30 tydzień B

Kraków, 16 października 2023

1 Wprowadzenie

1.1 Cel ćwiczenia

To ćwiczenie ma na celu zapoznanie się z metodami generacji losowych punktów oraz badanie metod klasyfikacji położenia punktów na płaszczyźnie względem prostej.

1.2 Położenie punktu względem prostej

Położenie punktu względem prostej będziemy wyznaczać obliczając dane wyznaczniki. Wyznaczniki pozwalają określić położenie punktu c względem prostej która jest wyznaczona przez punkty a i b. Jeżeli wyznacznik jest większy od 0 to punkt znajduje się z lewej strony prostej, jeżeli jest mniejszy od 0 to punkt znajduje się po prawej stronie prostej, a jeżeli wartość wyznacznika jest równa 0 (lub jej wartość bezwzględna $< \varepsilon$) to punkt leży na prostej.

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

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

Pomimo, że powyższe wyznaczniki są sobie równoważne to na skutek niedoskonałości reprezentacji liczb rzeczywistych w komputerze wyniki mogą się różnić w zależności od użytego wyznacznika.

2 Zbiory testowe

Na potrzeby ćwiczenia wygenerujemy 4 zbiory punktów losowych.

a) 10^5 losowych punktów (x, y) w przestrzeni $\mathbb{R}^2$, gdzie $(x, y) \in [-1000, 1000]^2$.

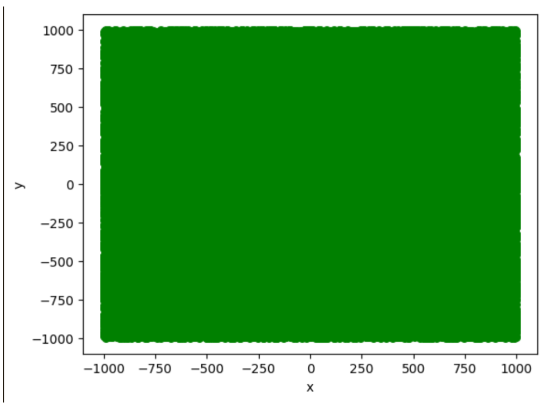
b) 10^5 losowych punktów (x, y) w przestrzeni $\mathbb{R}^2$, gdzie $(x, y) \in \left[-10^{14}, 10^{14}\right]^2$.

c) 1000 losowych punktów w przestrzeni $\mathbb{R}^2$ leżących na okręgu o środku $O = (0, 0)$ i promieniu $R = 100$.

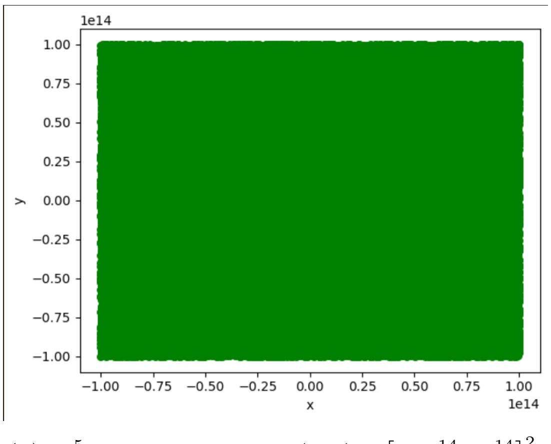
d) 1000 losowych punktów w przestrzeni $\mathbb{R}^2$ dla $x \in \langle -1000, 1000 \rangle$ leżących na prostej wyznaczonej przez wektor $\overrightarrow{ab}$. Gdzie $a = (-1.0, 0.0)$, $b = (1.0, 0.1)$.

3 Wykresy i algorytmy generacji zbiorów

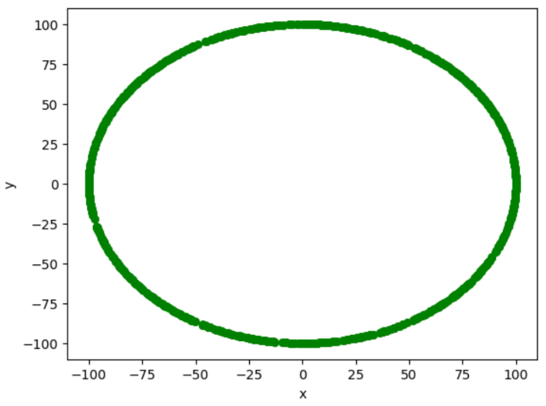
3.1 Wykresy wygenerowanych zbiorów



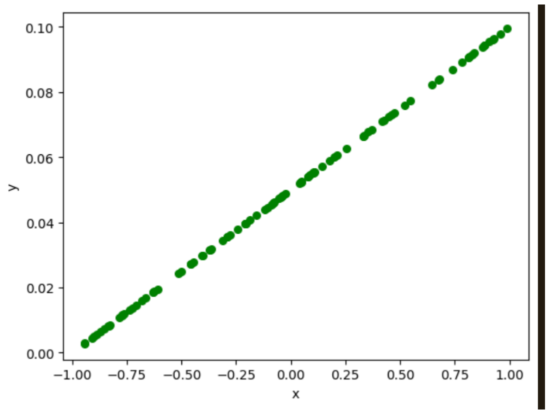
(a) 10^5 losowych punktów $(x, y) \in [-1000, 1000]^2$.



(b) 10^5 losowych punktów $(x, y) \in \left[-10^{14}, 10^{14}\right]^2$.



Rysunek 2: 1000 losowych punktów leżących na okręgu.



Rysunek 3: 1000 losowych punktów na prostej.

3.2 Algorytmy generacji zbiorów

1. Dla zbiorów a i b.

```
def generate_uniform_points(left, right, n = 10 ** 5):
    points = []
    for i in range(n):
        x = np.random.uniform(left, right)
        y = np.random.uniform(left, right)
        points.append((x, y))

    return points
```

2. Dla zbioru c.

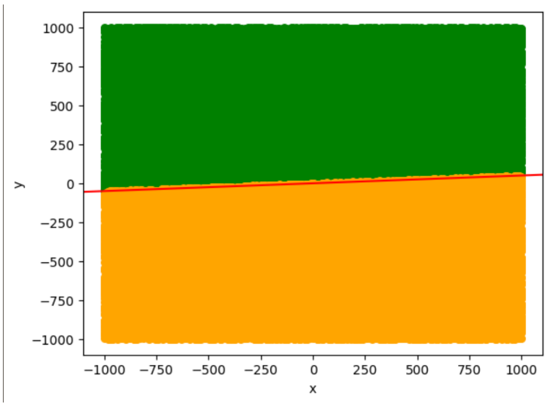
```
def generate_circle_points(O, R, n = 1000):
    points = []
    for _ in range(n):
        angle = 2 * np.pi * np.random.uniform()
        points.append((R * np.cos(angle), R * np.sin(angle)))
    return points
```

3. Dla zbioru d.

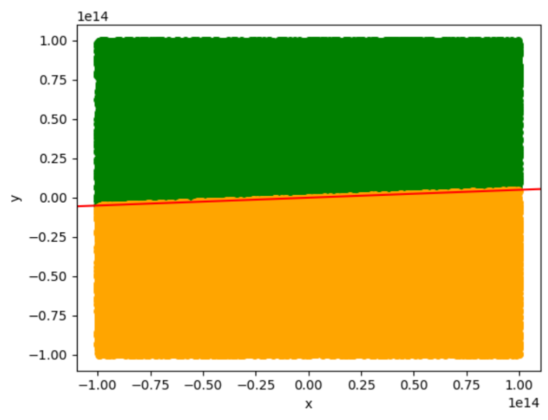
```
def generate_collinear_points(a, b, n=100):
    points = []
    xt = b[0] - a[0]
    yt = b[1] - a[1]
    for i in range(n):
        t = np.random.uniform()
        points.append((xt * t + a[0], yt * t + a[1] ))

    return points
```

4 Testy klasyfikacyjne dla różnych wartości ε

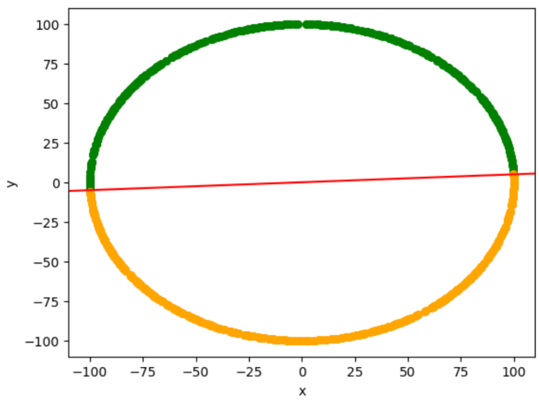
4.1 Przykładowe wykresy dla $\varepsilon = 10^{-4}$.



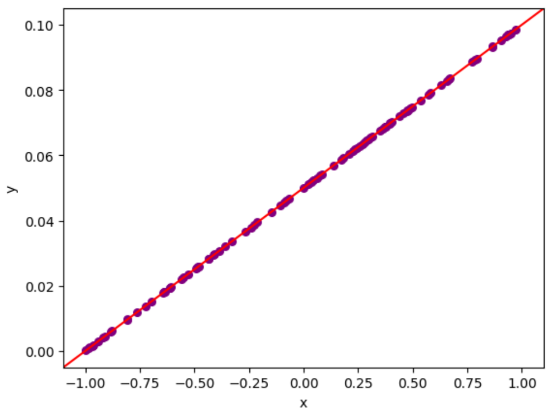
(a) 10^5 losowych punktów $(x, y) \in [-1000, 1000]^2$.



(b) 10^5 losowych punktów $(x, y) \in \left[-10^{14}, 10^{14}\right]^2$.



Rysunek 5: 1000 losowych punktów leżących na okręgu.



Rysunek 6: 1000 losowych punktów na prostej.

4.2 Tabele

Poniżej przedstawione są tabele ilości zaklasyfikowanych punktów ze zbiorów (a, b, c, d) ze względu na ich położenie od prostej, ε tolerancje wartości bliskich zera oraz funkcje użytą dla obliczenia wyznacznika.

	eps		5	3	1	-2	-4	-8	-12	-14	-16	-18	-20
0	Zbiór a	po lewej	0	24995	49942	50186	50186	50186	50186	50186	50186	50186	50186
1		na prostej	100000	49920	471	0	0	0	0	0	0	0	0
2		po prawej	0	25085	49587	49814	49814	49814	49814	49814	49814	49814	49814
3	Zbiór b	po lewej	49683	49683	49683	49683	49683	49683	49683	49683	49683	49683	49683
4		na prostej	0	0	0	0	0	0	0	0	0	0	0
5		po prawej	50317	50317	50317	50317	50317	50317	50317	50317	50317	50317	50317
6	Zbiór c	po lewej	0	0	461	481	481	481	481	481	481	481	481
7		na prostej	1000	1000	41	0	0	0	0	0	0	0	0
8		po prawej	0	0	498	519	519	519	519	519	519	519	519
9	Zbiór d	po lewej	0	0	0	0	0	0	0	0	0	17	17
10		na prostej	100	100	100	100	100	100	100	100	100	83	83
11		po prawej	0	0	0	0	0	0	0	0	0	0	0

Numpy 3×3.

	eps		5	3	1	-2	-4	-8	-12	-14	-16	-18	-20
0	Zbiór a	po lewej	0	24995	49942	50186	50186	50186	50186	50186	50186	50186	50186
1		na prostej	100000	49920	471	0	0	0	0	0	0	0	0
2		po prawej	0	25085	49587	49814	49814	49814	49814	49814	49814	49814	49814
3	Zbiór b	po lewej	49679	49679	49679	49679	49679	49679	49679	49679	49679	49679	49679
4		na prostej	7	7	7	7	7	7	7	7	7	7	7
5		po prawej	50314	50314	50314	50314	50314	50314	50314	50314	50314	50314	50314
6	Zbiór c	po lewej	0	0	461	481	481	481	481	481	481	481	481
7		na prostej	1000	1000	41	0	0	0	0	0	0	0	0
8		po prawej	0	0	498	519	519	519	519	519	519	519	519
9	Zbiór d	po lewej	0	0	0	0	0	0	0	0	0	27	29
10		na prostej	100	100	100	100	100	100	100	100	100	43	40
11		po prawej	0	0	0	0	0	0	0	0	0	30	31

Numpy 2×2.

	eps		5	3	1	-2	-4	-8	-12	-14	-16	-18	-20
0	Zbiór a	po lewej	0	24995	49942	50186	50186	50186	50186	50186	50186	50186	50186
1		na prostej	100000	49920	471	0	0	0	0	0	0	0	0
2		po prawej	0	25085	49587	49814	49814	49814	49814	49814	49814	49814	49814
3	Zbiór b	po lewej	49683	49683	49683	49683	49683	49683	49683	49683	49683	49683	49683
4		na prostej	0	0	0	0	0	0	0	0	0	0	0
5		po prawej	50317	50317	50317	50317	50317	50317	50317	50317	50317	50317	50317
6	Zbiór c	po lewej	0	0	461	481	481	481	481	481	481	481	481
7		na prostej	1000	1000	41	0	0	0	0	0	0	0	0
8		po prawej	0	0	498	519	519	519	519	519	519	519	519
9	Zbiór d	po lewej	0	0	0	0	0	0	0	0	0	17	17
10		na prostej	100	100	100	100	100	100	100	100	100	83	83
11		po prawej	0	0	0	0	0	0	0	0	0	0	0

Własna funkcja 3×3.

	eps		5	3	1	-2	-4	-8	-12	-14	-16	-18	-20
0	Zbiór a	po lewej	0	24995	49942	50186	50186	50186	50186	50186	50186	50186	50186
1		na prostej	100000	49920	471	0	0	0	0	0	0	0	0
2		po prawej	0	25085	49587	49814	49814	49814	49814	49814	49814	49814	49814
3	Zbiór b	po lewej	49679	49679	49679	49679	49679	49679	49679	49679	49679	49679	49679
4		na prostej	9	9	9	9	9	9	9	9	9	9	9
5		po prawej	50312	50312	50312	50312	50312	50312	50312	50312	50312	50312	50312
6	Zbiór c	po lewej	0	0	461	481	481	481	481	481	481	481	481
7		na prostej	1000	1000	41	0	0	0	0	0	0	0	0
8		po prawej	0	0	498	519	519	519	519	519	519	519	519
9	Zbiór d	po lewej	0	0	0	0	0	0	0	0	0	21	24
10		na prostej	100	100	100	100	100	100	100	100	100	47	42
11		po prawej	0	0	0	0	0	0	0	0	0	32	34

Własna funkcja 2×2.

4.3 Analiza wyników

Na potrzeby analizy danych wybierzemy wyniki dla bibliotecznej funkcji liczenia wyznacznika 3×3.

1. W zbiorze a dla $\varepsilon \leq 10^{-2}$ widzimy brak zmian w ilości zaklasyfikowanych do różnych zbiorów punktów. Także charakterystyczną dla tego ε cechą można nazwać 0 punktów rozpoznanych jako leżące na prostej.

   Żeby zweryfikować nasze wyniki policzmy prawdopodobieństwo, że punkt w naszym zbiorze zostanie wylosowany w miejscu dla którego będzie zaklasyfikowany, jako należący do prostej. W naszych obliczeniach dla prostoty pominiemy niepewności związane z obliczeniami komputerowymi.

   Klasyfikujemy punkty do odpowiedniej grupy obliczając iloczyn wektorowy $\overrightarrow{ab} \times \overrightarrow{ac}$, gdzie $c = (x, y)$ jest punktem, dla którego poszukujemy wiadomości o lokalizacji względem prostej przechodzącej przez punkty a i b. Metoda ta jest równoznaczna z obliczeniem wyznacznika macierzy 2×2:

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

Wiemy także, że iloczyn wektorowy $\overrightarrow{A} \times \overrightarrow{B}$ można także obliczyć ze wzoru:

$$\overrightarrow{A} \times \overrightarrow{B} = ||A|| \cdot ||B|| \cdot \sin\theta.$$

Gdzie $\sin\theta$ to kąt pomiędzy wektorami $\overrightarrow{A}$ i $\overrightarrow{B}$.
Wybierzmy teraz dla konkretnego punktu c takie a i b na prostej, żeby

(a) Dla $A = \overrightarrow{ab}$, $||A|| = 1$.

(b) $\sin\theta = 1$, gdzie θ jest kątem pomiędzy $\overrightarrow{ab}$ i $\overrightarrow{ac}$ (wektor $\overrightarrow{ac}$ jest prostopadły do naszej prostej i do wektora $\overrightarrow{ab}$).

Wtedy nasz iloczyn

$$\overrightarrow{A} \times \overrightarrow{B} = ||A|| \cdot ||B|| = 1 \cdot ||B|| = ||B||.$$

I pytanie klasyfikacji punktu jako należacego do prostej sprowadza się do sprawdzenia czy $||B|| < \varepsilon$. Dla zbioru a długość prostej o równaniu $y = \frac{1}{20}x - 1$ na całym zbiorze $(-1000, 1000)$ jest równa

$$\sqrt{(y(1000) - y(-1000))^2 + (1000 + 1000)^2} = \sqrt{10^4 + 4 \cdot 10^6} = \sqrt{4,01 \cdot 10^6} \approx 2{\cdot}10^3.$$

Pole obszaru na którym punkty będą klasyfikowane jako należące do prostej zate wynosi

$$2 \cdot 2 \cdot 10^3 \cdot \varepsilon = 4 \cdot 10^3 \cdot \varepsilon.$$

Teraz w porównaniu do całkowitego pola naszego zbioru

$$\frac{4 \cdot 10^3 \cdot \varepsilon}{(1000 - (-1000))^2} = \frac{4 \cdot 10^3 \cdot \varepsilon}{4 \cdot 10^6} = 10^{-3} \cdot \varepsilon.$$

Patrząc na ten wynik dla $\varepsilon = 10^{-2}$ nie jest zadziwiające to, że żaden z 10^5 punktów nie został zakwalifikowany jako leżący na prostej. Gdzyż prawdopodobieństwo wylosowania takiego punktu to tylko 10^{-5}.

2. Dla zbioru b widzimy, że dla wszystkich $10^{-20} < \varepsilon < 10^5$ dostajemy ten sam wynik , czego też warto było się spodziewać gdzyż stosując metodę opisaną w poprzednim podpunkcie dostajemy prawdopodobieństwo wylosowania punktu leżącego na prostej

$$\frac{4 \cdot 10^{14} \cdot \varepsilon}{(10^{14} - (-10^{14}))^2} = \frac{4 \cdot 10^{14} \cdot \varepsilon}{4 \cdot 10^{28}} = 10^{-14} \cdot \varepsilon.$$

Czyli nawet dla $\varepsilon = 10^5$ prawdopodobieństwo, że conajmniej jeden z 10^5 punktów zostanie zaklasyfikowany do prostej to około 10^{-4}.

3. Dla zbioru c widzimy podobną sytuację. Dla $\varepsilon > 10^{-2}$ każde zmniejszenie ε powoduje ogromne zmiany w wynikach, ale podalsze zwiększenie precyzji nie ma żadnego efektu na wynikach.

4. Dla zbioru d zmniejszenie ε ma największe znaczenie ze wszystkich. Nawet przejście pomiędzy $\varepsilon = 10^{-16}$ do $\varepsilon = 10^{-18}$ ma ogromne znaczenie na ilości punktów zakwalifikowanych do prostej.

5 Porównywanie czasów klasyfikacji dla różnych funkcji obliczających wyznacznik

Poniżej przedstawiona jest tabela średniego czasu klasyfikacji na zbiorze testowym specjalnie dla tego celu (zbiór ma ogromną ilość punktów). 10^6 losowych punktów (x, y) w przestrzeni $\mathbb{R}^2$, gdzie $(x, y) \in \left[-10^{14}, 10^{14}\right]^2$.

```
test_set = generate_uniform_points(-10^4, 10^4, 10 ** 6)
```

		Numpy 2x2	Numpy 3x3	Własny 2x2	Własny 3x3
0	Średni czas klasyfikacji [ms]	604	595	575	586

6 Testy precyzji float64 i float32

Na poniższym wykresie jest przedstawiona tabela ilości zaklasyfikowanych do prostej punktów w zależności od precyzji zminnej (float32, float64) i ε. zbiorze testowym specjalnie dla tego celu (zbiór punktów na prostej). 10^4 losowych punktów w przestrzeni $\mathbb{R}^2$ dla $x \in \langle -1000, 1000 \rangle$ leżących na prostej wyznaczonej przez wektor $\overrightarrow{ab}$. Gdzie $a = (-1.0, 0.0)$, $b = (1.0, 0.1)$.

	eps	-4	-8	-12	-14	-16	-18	-20	-22	-24	-26
0	float64	10000	10000	10000	10000	10000	4067	3855	3855	3855	3855
1	float32	10000	10000	1522	1522	1522	362	361	361	361	361

7 Podsumowanie

Porównując ilość zaklasyfikowanych punktów dla różnych ε dostaliśmy różne wyiniki dla róznych metod obliczenia wyznacznika. Klasyfikacja punktów ze zbiorów a oraz c dawała podobne wyniki niezależnie od użytej funkcji liczenia wyznacznika czy ε. Spowodowane jest to tym, że istnieje bardzo małe prawdopodobieństwo, iż punkt będzie znajdował się dokładnie na prostej, z dokładnością do użytej tolerancji. W podpunkcie b wybór epsilona nie miał znaczenia jednak wybór wyznacznika już tak. Jest to spowodowane dużymi odległościami pomiędzy punktami. Operacja klasyfikacji punktów wymaga jednak dużej precyzji, która jest tracona z powodu dużych wartości które są generowane. Z tego możemy dojść do wniosku, że użycie wyznaczników 3×3 daje nam większą precyzje w obliczeniach, a też patrząc na wyniki porównania wydajności funkcji liczących wyznacznik, są wykonywane tak samo szybko, jak i funkcje liczenia wyznacznika 2×2.

Patrząc na porównanie precyzji float64 z float32, możemy dojść do wniosku, że float64 daje (jak wypadało się spodziewać) dużo większą precyzje obliczeń. Widoczna jest poważna różnica w obliczeniach już dla $\varepsilon = 10^{-12}$. Na tej podstawie na potrzeby algorytmów geometrycznych zalecane jest używanie float64.

We wszystkich testach i porównaniach statystycznych dostaliśmy wyniki, które zgadzały się z wartościami teoretycznymi. Na tej podstawie możemy dojść do wniosku, że w obliczeniach nie było poważnych błędów i ćwiczenie było wykonane poprawnie.