|  |  |
| --- | --- |
| Wojciech Michaluk | grupa nr 6 godz. 13:00-14:30 czw. A |

**Sprawozdanie z laboratorium nr 1: Predykaty geometryczne**

|  |  |
| --- | --- |
| Data wykonania: 05.10.2023r. | Data oddania: 20.10.2023r. |

**1. Cel ćwiczenia**

Celem ćwiczenia jest implementacja podstawowych predykatów geometrycznych – m. in. określenie po której stronie prostej znajduje się punkt, a także przeprowadzenie testów, wizualizacja   
i opracowanie wyników.

**2. Wstęp teoretyczny**

Do wyznaczania, po której stronie prostej znajduje się punkt, używam odpowiedniego wyznacznika macierzy.

|  |  |
| --- | --- |
| Obraz zawierający zrzut ekranu, linia  Opis wygenerowany automatycznie | Niech a=(ax,ay), b=(bx,by), c=(cx,cy). Wtedy można obliczyć na dwa sposoby wyznacznik :  1)  2) . |
| **Rys 2.1** Przykładowe punkty |  |

punkt c leży po lewej stronie prostej ab

punkt c leży po prawej stronie prostej ab

punkt c leży na prostej ab

**3. Metodologia, specyfikacja narzędzi i sprzętu**

Punkty potrzebne do wykonania ćwiczenia zostały wygenerowane losowo z użyciem funkcji random.uniform() z biblioteki numpy. Każda z funkcji obliczających wyznacznik macierzy została napisana w dwóch wersjach: z wykorzystaniem funkcji linalg.det() z biblioteki numpy (nazwy tych funkcji to mat\_det\_3x3\_lib oraz mat\_det\_2x2\_lib) a także bez ich używania (mat\_det\_3x3, mat\_det\_2x2). Poszczególne wyniki przedstawię dla dwóch różnych przyjętych tolerancji dla zera: 10-12 i 10-8 oraz różnych precyzji floata: float32 i float64. Wykresy przedstawiające wygenerowane losowo punkty powstały przy użyciu biblioteki matplotlib oraz dzięki narzędziu przygotowanemu przez koło naukowe Bit. Program (w pliku „michaluk\_kod\_1.ipynb”) jest napisany w języku Python w środowisku Jupyter Notebook. Przedstawione wyniki pochodzą z uruchomienia programu na komputerze z systemem Windows 11 i procesorem Intel Core i5-8300H 2.30 GHz.

**4. Program ćwiczenia**

Na początku przygotowałem następujące zbiory punktów:

|  |  |  |  |
| --- | --- | --- | --- |
|  | a) 105 losowych punktów o współrzędnych  z przedziału [-1000, 1000] (zbiór A) |  | b) 105 losowych punktów o współrzędnych  z przedziału [-1014, 1014] (zbiór B) |
|  | |  | |
| **Rys. 4.1** Zbiór A | | **Rys. 4.2** Zbiór B | |
|  | c) 1000 losowych punktów leżących na okręgu o środku w punkcie (0,0) i promieniu R=100 (zbiór C) |  | d) 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) (zbiór D) |
|  | |  | |
| **Rys. 4.3** Zbiór C | | **Rys 4.4** Zbiór D | |

Teraz przedstawię dla każdego zbioru dane dotyczące podziału punktów względem prostej przechodzącej przez punkty a i b, gdzie a = [-1.0, 0.0], b = [1.0, 0.1] – liczbę punktów znajdujących się po lewej stronie prostej, liczbę punktów po prawej stronie prostej oraz leżących na prostej. Każdy punkt jest kwalifikowany na podstawie wyznacznika, którego sposób obliczania przedstawiono   
w **rozdziale 2**. W graficznym przedstawieniu kolorem zielonym są oznaczone punkty leżące na lewo od prostej, żółtym – na prawo oraz fioletowym – na prostej. Podane wyniki uwzględniają funkcję użytą do obliczania wyznacznika, przyjętą tolerancję dla zera () oraz wybraną precyzję floata.

**5. Otrzymane wyniki oraz ich analiza**

|  |  |  |  |
| --- | --- | --- | --- |
| Używane funkcje do obliczania wyznacznika | Liczba punktów po lewej | Liczba punktów po prawej | Liczba punktów na prostej |
| tolerancja , precyzja float32 | | | |
| mat\_det\_3x3 | 49941 | 50059 | 0 |
| mat\_det\_3x3\_lib | 49941 | 50059 | 0 |
| mat\_det\_2x2 | 49941 | 50059 | 0 |
| mat\_det\_2x2\_lib | 49941 | 50059 | 0 |
| tolerancja , precyzja float32 | | | |
| mat\_det\_3x3 | 49941 | 50059 | 0 |
| mat\_det\_3x3\_lib | 49941 | 50059 | 0 |
| mat\_det\_2x2 | 49941 | 50059 | 0 |
| mat\_det\_2x2\_lib | 49941 | 50059 | 0 |
| tolerancja , precyzja float64 | | | |
| mat\_det\_3x3 | 49941 | 50059 | 0 |
| mat\_det\_3x3\_lib | 49941 | 50059 | 0 |
| mat\_det\_2x2 | 49941 | 50059 | 0 |
| mat\_det\_2x2\_lib | 49941 | 50059 | 0 |
| tolerancja , precyzja float64 | | | |
| mat\_det\_3x3 | 49941 | 50059 | 0 |
| mat\_det\_3x3\_lib | 49941 | 50059 | 0 |
| mat\_det\_2x2 | 49941 | 50059 | 0 |
| mat\_det\_2x2\_lib | 49941 | 50059 | 0 |

**Tabela 5.1** Rozkład punktów dla zbioru A

Jak widać w tabeli 5.1, dla zbioru A wyniki są identyczne dla przyjętych różnych tolerancji, precyzji oraz funkcji obliczającej wyznacznik. Ich graficzne przedstawienie wygląda następująco:

|  |  |
| --- | --- |
| Obraz zawierający tekst, zrzut ekranu, wyświetlacz, Prostokąt  Opis wygenerowany automatycznie | **Rys 5.5** Graficzne przedstawienie rozkładu punktów w zbiorze A |

|  |  |  |  |
| --- | --- | --- | --- |
| Używane funkcje do obliczania wyznacznika | Liczba punktów po lewej | Liczba punktów po prawej | Liczba punktów na prostej |
| tolerancja , precyzja float32 | | | |
| mat\_det\_3x3 | 50115 | 49985 | 0 |
| mat\_det\_3x3\_lib | 50115 | 49985 | 0 |
| mat\_det\_2x2 | 50113 | 49984 | 3 |
| mat\_det\_2x2\_lib | 50113 | 49983 | 4 |
| tolerancja , precyzja float32 | | | |
| mat\_det\_3x3 | 50115 | 49985 | 0 |
| mat\_det\_3x3\_lib | 50115 | 49985 | 0 |
| mat\_det\_2x2 | 50113 | 49984 | 3 |
| mat\_det\_2x2\_lib | 50113 | 49983 | 4 |
| tolerancja , precyzja float64 | | | |
| mat\_det\_3x3 | 50115 | 49985 | 0 |
| mat\_det\_3x3\_lib | 50115 | 49985 | 0 |
| mat\_det\_2x2 | 50113 | 49984 | 3 |
| mat\_det\_2x2\_lib | 50113 | 49984 | 3 |
| tolerancja , precyzja float64 | | | |
| mat\_det\_3x3 | 50115 | 49985 | 0 |
| mat\_det\_3x3\_lib | 50115 | 49985 | 0 |
| mat\_det\_2x2 | 50113 | 49984 | 3 |
| mat\_det\_2x2\_lib | 50113 | 49984 | 3 |

**Tabela 5.2** Rozkład punktów dla zbioru B

Wyniki są praktycznie identyczne dla różnych precyzji i tolerancji dla zera. Widać różnicę w zależności od użytej funkcji – zgodnie z tabelą 5.2 wyznaczniki 2x2 „wyłapują” kilka punktów na prostej (przy czym przy takim zakresie współrzędnych, prawdopodobieństwo, że punkt znajdzie się na prostej jest znikome), podczas gdy wyznaczniki 3x3 nie. Dla zwiększonej precyzji floata wyznacznik obliczany za pomocą funkcji mat\_det\_2x2\_lib kwalifikuje na prostej jeden punkt mniej (a podpada on punktom po prawej).

Przy takiej liczbie punktów różnic w graficznym przedstawieniu nie ma (zaznaczone na fioletowo punkty i tak są „przesłaniane” przez pozostałe).

Obraz zawierający tekst, zrzut ekranu, Prostokąt, linia

Opis wygenerowany automatycznie

**Rys. 5.6** Graficzne przedstawienie rozkładu punktów w zbiorze B

|  |  |  |  |
| --- | --- | --- | --- |
| Używane funkcje do obliczania wyznacznika | Liczba punktów po lewej | Liczba punktów po prawej | Liczba punktów na prostej |
| tolerancja , precyzja float32 | | | |
| mat\_det\_3x3 | 480 | 520 | 0 |
| mat\_det\_3x3\_lib | 480 | 520 | 0 |
| mat\_det\_2x2 | 480 | 520 | 0 |
| mat\_det\_2x2\_lib | 480 | 520 | 0 |
| tolerancja , precyzja float32 | | | |
| mat\_det\_3x3 | 480 | 520 | 0 |
| mat\_det\_3x3\_lib | 480 | 520 | 0 |
| mat\_det\_2x2 | 480 | 520 | 0 |
| mat\_det\_2x2\_lib | 480 | 520 | 0 |
| tolerancja , precyzja float64 | | | |
| mat\_det\_3x3 | 480 | 520 | 0 |
| mat\_det\_3x3\_lib | 480 | 520 | 0 |
| mat\_det\_2x2 | 480 | 520 | 0 |
| mat\_det\_2x2\_lib | 480 | 520 | 0 |
| tolerancja , precyzja float64 | | | |
| mat\_det\_3x3 | 480 | 520 | 0 |
| mat\_det\_3x3\_lib | 480 | 520 | 0 |
| mat\_det\_2x2 | 480 | 520 | 0 |
| mat\_det\_2x2\_lib | 480 | 520 | 0 |

**Tabela 5.3** Rozkład punktów dla zbioru C

Analizując tabelę 5.3, widać że podobnie jak dla zbioru A, wyniki są identyczne dla każdej kombinacji precyzji, tolerancji dla zera i użytej funkcji do obliczania wyznacznika. Wpływ na to może mieć stosunkowo mała (w porównaniu do poprzednich zbiorów) liczba losowanych punktów. Przykładowe graficzne przedstawienie podziału punktów w zbiorze C prezentuję poniżej.

Obraz zawierający linia, diagram, Wykres, zrzut ekranu

Opis wygenerowany automatycznie

**Rys. 5.7** Graficzne przedstawienie rozkładu punktów w zbiorze C

Zanim przedstawię wyniki kwalifikacji punktów w zbiorze D, zastanówmy się – zgodnie z poleceniem były one losowane **na prostej**, więc w teorii wszystkie z nich powinny tak być zakwalifikowane. A jak to wygląda w praktyce?

|  |  |  |  |
| --- | --- | --- | --- |
| Używane funkcje do obliczania wyznacznika | Liczba punktów po lewej | Liczba punktów po prawej | Liczba punktów na prostej |
| tolerancja , precyzja float32 | | | |
| mat\_det\_3x3 | 419 | 397 | 184 |
| mat\_det\_3x3\_lib | 419 | 397 | 184 |
| mat\_det\_2x2 | 419 | 397 | 184 |
| mat\_det\_2x2\_lib | 419 | 397 | 184 |
| tolerancja , precyzja float32 | | | |
| mat\_det\_3x3 | 419 | 400 | 181 |
| mat\_det\_3x3\_lib | 419 | 400 | 181 |
| mat\_det\_2x2 | 419 | 400 | 181 |
| mat\_det\_2x2\_lib | 435 | 420 | 145 |
| tolerancja , precyzja float64 | | | |
| mat\_det\_3x3 | 0 | 0 | 1000 |
| mat\_det\_3x3\_lib | 0 | 0 | 1000 |
| mat\_det\_2x2 | 0 | 0 | 1000 |
| mat\_det\_2x2\_lib | 0 | 0 | 1000 |
| tolerancja , precyzja float64 | | | |
| mat\_det\_3x3 | 0 | 0 | 1000 |
| mat\_det\_3x3\_lib | 0 | 0 | 1000 |
| mat\_det\_2x2 | 77 | 84 | 839 |
| mat\_det\_2x2\_lib | 103 | 99 | 798 |

**Tabela 5.4** Rozkład punktów dla zbioru D – co tu się zadziało?

Pytanie zadane w tytule tabeli 5.4 nasuwa się na myśl. Można zauważyć, że bezbłędne wyniki udało się uzyskać jedynie dla „dużej” tolerancji dla 0 i precyzji floata. Kluczowe znaczenie w podziale punktów miała właśnie precyzja – nawet dla mniejszej tolerancji wyznaczniki 3x3 zaklasyfikowały punkty bezbłędnie, wyznaczniki 2x2 już poradziły sobie nieco gorzej (ale przyjmijmy, że w miarę przyzwoicie). Natomiast w przypadku mniejszej precyzji floata, dla obu przyjętych tolerancji wyniki są dość podobne (oprócz odchyłu mat\_det\_2x2\_lib przy mniejszej tolerancji), ponadto bardzo niedokładne – zaledwie 18% punktów zostało uznanych za leżące na prostej, a w przypadku wspomnianego odchyłu – zaledwie 14,5%.

Na następnej stronie przedstawiam reprezentacje graficzne dla wybranych parametrów: precyzja floata, tolerancja dla zera, funkcja obliczająca wyznacznik. Dzięki temu można obserwować, jak w zależności od nich linia staje się mniej lub bardziej fioletowa.

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  | |
|  | **Rys. 5.8** Precyzja float32, ,  funkcja mat\_det\_3x3 |  | **Rys 5.9** Precyzja float32, ,  funkcja mat\_det\_2x2\_lib |
|  | |  | |
|  | **Rys 5.10** Precyzja float64, ,  funkcja mat\_det\_2x2\_lib |  | **Rys. 5.11** Precyzja float64, ,  funkcja mat\_det\_3x3\_lib |
|  | |  | |
|  | **Rys. 5.12** Precyzja float64, , funkcja mat\_det\_2x2 |  | **Rys. 5.13** Precyzja float64, , funkcja mat\_det\_2x2\_lib |

**6. Podsumowanie i wnioski**

Graficzne przedstawienie dla wszystkich kombinacji danych znajduje się w pliku z kodem,   
w sprawozdaniu umieściłem najważniejsze, dla których widać różnicę w kwalifikowaniu punktów lub aby pokazać rozkład klasyfikacji punktów w zbiorze.

Na podstawie przeprowadzonych testów i wyników można wywnioskować, że dla zbiorów A,B i C dobranie precyzji floata i tolerancji dla zera oraz metody obliczania wyznacznika nie przynosiło znacznych różnic w podziale punktów (dla zbiorów A i C wyniki są identyczne; dla zbioru B obliczanie wyznacznikiem 2x2 dawało delikatnie odmienne rezultaty). Inaczej jest w przypadku zbioru D – de facto zbioru punktów leżących na prostej. Obnaża on niedoskonałości komputera w kontekście przechowywania liczby rzeczywistej – skończona reprezentacja liczby. Było to szczególnie widoczne dla mniejszej precyzji, gdzie większość punktów była uznawana za **nieleżące** na prostej! W przypadku większej precyzji też nie było idealnie, jeżeli tolerancja dla zera była zbyt mała.

Ponadto, potencjalnie funkcje obliczające wyznacznik z wykorzystaniem funkcji bibliotecznych powinny być wolniejsze. Jednakże, przyjęte zbiory punktów jak na dzisiejsze standardy technologiczne nie zawierały **aż tak wielu** punktów, więc różnice na tym polu były praktycznie niewidoczne.

To zadanie pozwoliło zademonstrować, że należy uważać, operując na komputerze na liczbach rzeczywistych, bowiem sposób reprezentacji liczb rzeczywistych (w pewnym sensie podyktowany ograniczeniami sprzętowymi – nieskończone rozwinięcie dziesiętne nie brzmi dla komputera zbyt przyjemnie) może w znaczącym stopniu „zafałszować” rzeczywistość.