Sprawozdanie z ćwiczenia drugiego

Algorytmy geometryczne

Jakub Pisarek
Informatyka, WIEiT, AGH

9 listopada 2022

1 Cel ćwiczenia

Celem ćwiczenia jest wyznaczenie otoczki wypukłej dla zadanych zbiorów punktów z $\mathbb{R}^2$ przy użyciu algorytmów Grahama i Jarvisa. Uzyskane wyniki należy następnie porównać ze sobą, w zależności od użytego algorytmu i ilości punktów w zbiorze.

2 Wprowadzenie

Otoczka wypukła $\mathrm{CH}(S)$ niepustego zbioru punktów S na płaszczyźnie jest najmniejszym wielokątem wypukłym zawierającym S. Jej wyznaczenie oznacza podanie ciągu punktów będących wierzchołkami otoczki (ustalenie: wymienionych w porządku przeciwnym do ruchu wskazówek zegara).

Żadne trzy punkty należące do otoczki wypukłej nie mogą być współliniowe.

Do wyznaczenia otoczki można wykorzystać m.in. algorytm Grahama (złożoność $\mathrm{O}(n \log n)$) i algorytm Jarvisa (złożoność pesymistyczna $\mathrm{O}(n^2)$, $\mathrm{O}(nk)$ jeśli liczba wierzchołków otoczki ograniczona jest przez stałą k), które nie będą tu przytaczane.

Podczas działania obu algorytmów wymagane jest uszeregowanie punktów względem kąta, który wyznaczają z pewną prostą. Do celu tego można użyć wyznacznika odpowiedniej macierzy, analogicznie jak w poprzednim ćwiczeniu.

3 Przebieg ćwiczenia

1. Przygotować program generujący następujące zbiory punktów na płaszczyźnie (współrzędne rzeczywiste typu double):
   a) 100 losowo wygenerowanych punktów o współrzędnych z przedziału [-100, 100],
   b) 100 losowo wygenerowanych punktów leżących na okręgu o środku (0, 0) i promieniu $R = 10$,
   c) 100 losowo wygenerowanych punktów leżących na bokach prostokąta o wierzchołkach (-10, 10), (-10, -10), (10, -10), (10, 10),
   d) wierzchołki kwadratu (0, 0), (10, 0), (10, 10), (0, 10) oraz punkty wygenerowane losowo w sposób następujący: po 25 punktów na dwóch bokach kwadratu leżących na osiach i po 20 punktów na przekątnych kwadratu.

2. Uruchomić wizualizację graficzną utworzonych zbiorów punktów.

3. Zmodyfikować program z punktu 1. tak, aby możliwe było zadawanie określonych parametrów dla schematów losowania punktów podanych powyżej:

   a) liczba punktów, przedziały dla współrzędnych,

   b) liczba punktów, środek i promień okręgu,

   c) liczba punktów, wierzchołki prostokąta,

   d) wierzchołki kwadratu, liczba punktów na osiach, liczba punktów na przekątnych.

4. Zaimplementować algorytmy Grahama oraz Jarvisa wyznaczające otoczkę wypukłą dla zadanego zbioru punktów:

   - Przetestować program na zbiorach z punktu 1. i zmodyfikowanych zbiorach z punktu 3.,
   - Umożliwić zapisanie w pliku wyniku działania, czyli ciągu wierzchołków otoczki,
   - Dodać możliwość sprawdzenia czasu działania algorytmu.

5. Uruchomić aplikację graficzną tak, aby można było zilustrować graficznie poszczególne kroki realizacji algorytmu:

   - Rozróżnić punkty zbioru wyjściowego, boki aktualnie rozpatrywane i boki wielokąta otoczki,
   - Po zakończeniu wyznaczania otoczki, jej wierzchołki winny być wyróżnione.

6. Stwierdzić, czy program działał poprawnie dla wszystkich zbiorów danych oraz opisać krótko wnioski dotyczące działania algorytmu dla poszczególnych zbiorów:

   - Odpowiedzieć na pytania: Co było powodem zaproponowania takich, a nie innych zbiorów punktów? Które z nich mogły stwarzać problemy w działaniu algorytmów, jakie?,
   - Porównać czasy działania obu algorytmów w zależności od liczby punktów w poszczególnych zbiorach (A, B, C, D),
   - Dołączyć wybrane, istotne ilustracje.

4 Środowisko programistyczne

- System operacyjny Ubuntu 20.04.5 LTS x86_64,
- Procesor Intel Core i5-3320M 3.30 GHz,
- Python 3.8.10 z zewnętrznymi bibliotekami `numpy` i `matplotlib`,
- Środowisko Jupyter Notebook zintegrowane z edytorem Visual Studio Code.

5 Szczegóły implementacji algorytmów

5.1 Algorytm Grahama

Punkty bez punktu skrajnego p_0 posortowano z użyciem dostępnej w Pythonie funkcji `sorted()`, używającej algorytmu Timsort – hybrydy merge sort i insertion sort. Do funkcji tej przekazano komparator korzystający z wyznacznika odpowiedniej macierzy 3x3 we własnej implementacji. Następnie, przechodzono po posortowanych punktach, odrzucając niektóre punkty współliniowe, tzn. leżące na tej samej prostej wraz z punktem skrajnym p_0 – w przypadku znalezienia takich, co działo się również z użyciem ww. wyznacznika, notowano w odpowiedniej strukturze danych informację, by do otoczki rozważać jedynie punkt bardziej odległy od p_0 z tych dwóch. Następnie, jako stosu użyto kontenera `collections.deque` z biblioteki standardowej Pythona. Ponownie, używano wyznacznika do sprawdzenia, po której stronie krawędzi otoczki leży potencjalny nowy punkt do niej należący.

5.2 Algorytm Jarvisa

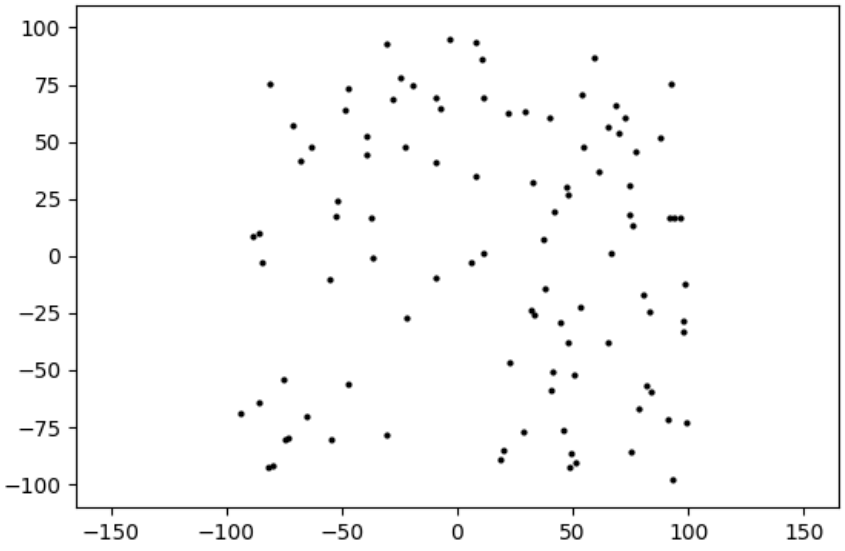
Ponownie, użyto wyznacznika odpowiedniej macierzy 3x3 w implementacji własnej, tym razem do poszukiwania najmniejszego kąta. Dodatkowo, jeżeli zwrócił on informację, że punkty są współliniowe, używano funkcji pomocniczej obliczającej dystans między punktami, by zawsze wziąć ten bardziej odległy.

6 Wyniki

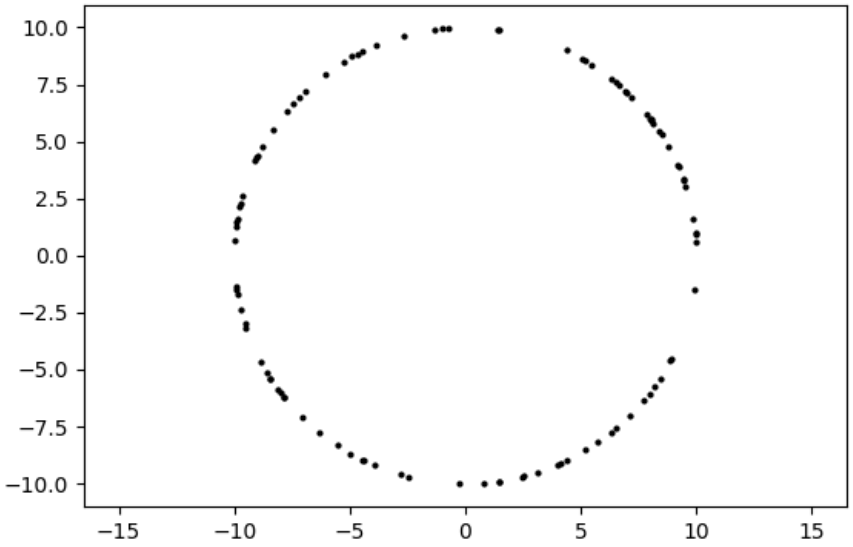
6.1 Wizualizacje zbiorów punktów

6.1.1 Oryginalne zbiory

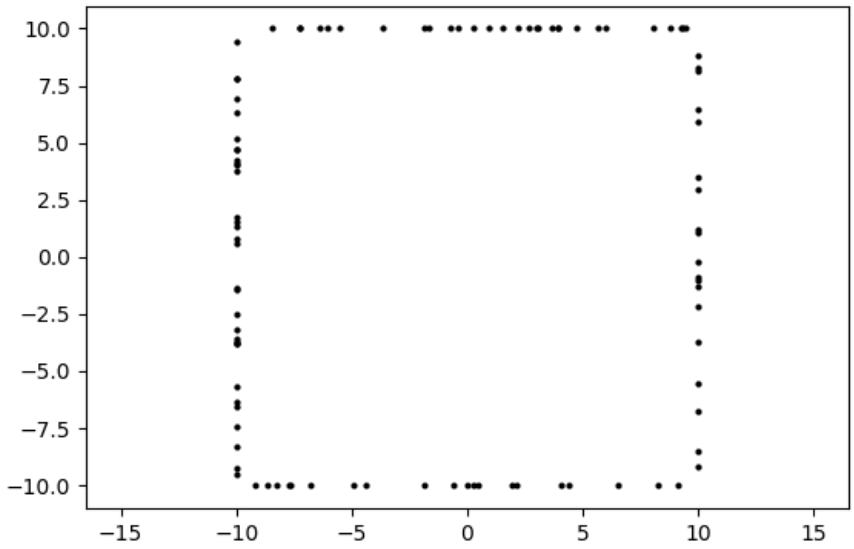
Nazywając poszczególne zbiory zgodnie z podpunktami, w których zostały nadmienione, otrzymujemy:



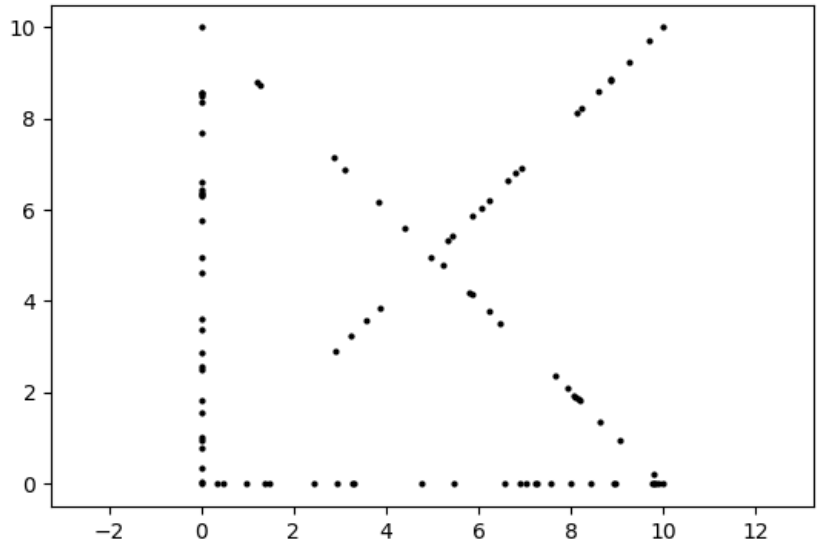
Ryc. 1: Wizualizacja punktów ze zbioru A



Ryc. 2: Wizualizacja punktów ze zbioru B



Ryc. 3: Wizualizacja punktów ze zbioru C



Ryc. 4: Wizualizacja punktów ze zbioru D

6.1.2 Zmodyfikowane zbiory

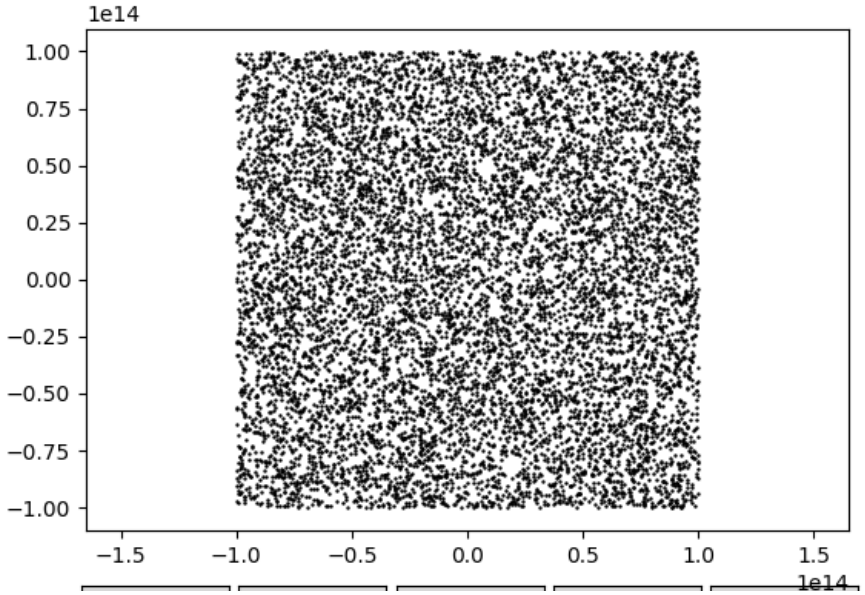
Używając tych samych metod co dla wyżej widocznych zbiorów, wygenerowano zmodyfikowane zbiory o zmienionych parametrach, tj.

- dla zbioru A: n losowo wygenerowanych punktów o współrzędnych z przedziału $[-10^{14}, 10^{14}]$,
- dla zbioru B: n losowo wygenerowanych punktów leżących na okręgu o środku $(-5 \cdot 10^3, -10^4)$ i promieniu $R = 10^4$,

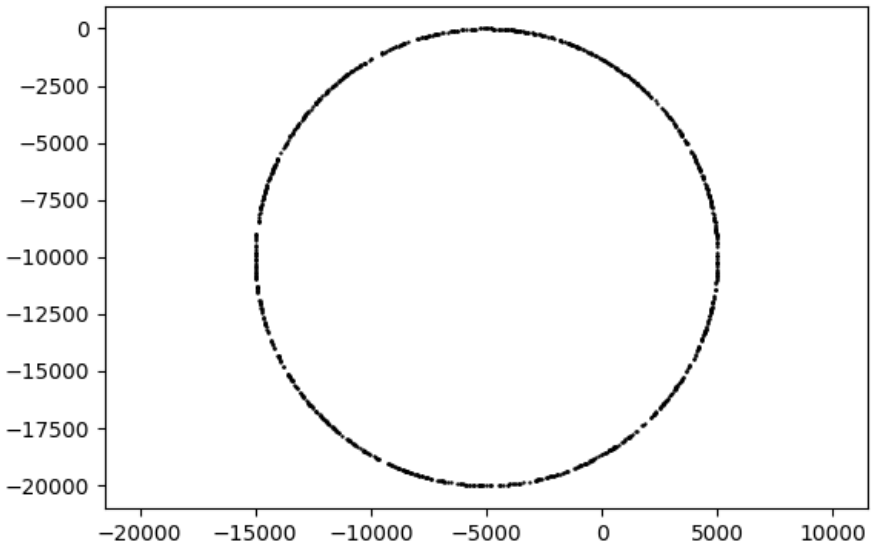
- dla zbioru C: n losowo wygenerowanych punktów leżących na bokach prostokąta o wierzchołkach $(-5 \cdot 10^3, 2 \cdot 10^4)$, $(-5 \cdot 10^3, -10^4)$, $(5 \cdot 10^3, -10^4)$, $(5 \cdot 10^3, 2 \cdot 10^4)$,
- dla zbioru D: wierzchołki kwadratu $(5 \cdot 10^4, 0)$, $(6 \cdot 10^4, 0)$, $(6 \cdot 10^4, 10^4)$, $(5 \cdot 10^4, 10^4)$ oraz punkty wygenerowane losowo w sposób następujący: po $\frac{n}{4}$ punktów na dwóch bokach kwadratu leżących na osiach i po $\frac{n}{5}$ punktów na przekątnych kwadratu;

gdzie $n \in \{10^2, 10^3, 10^4, 10^5\}$.

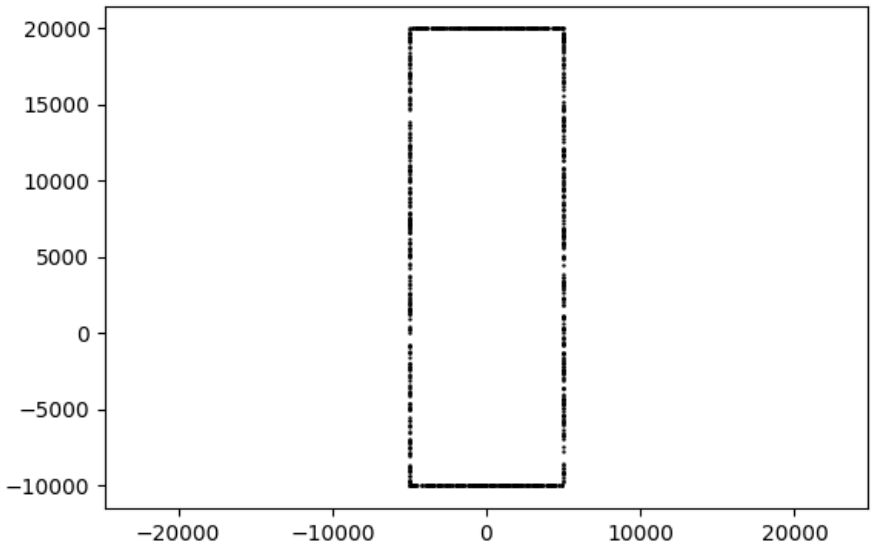
Przykładowe wizualizacje otrzymanych zbiorów przedstawiono poniżej:



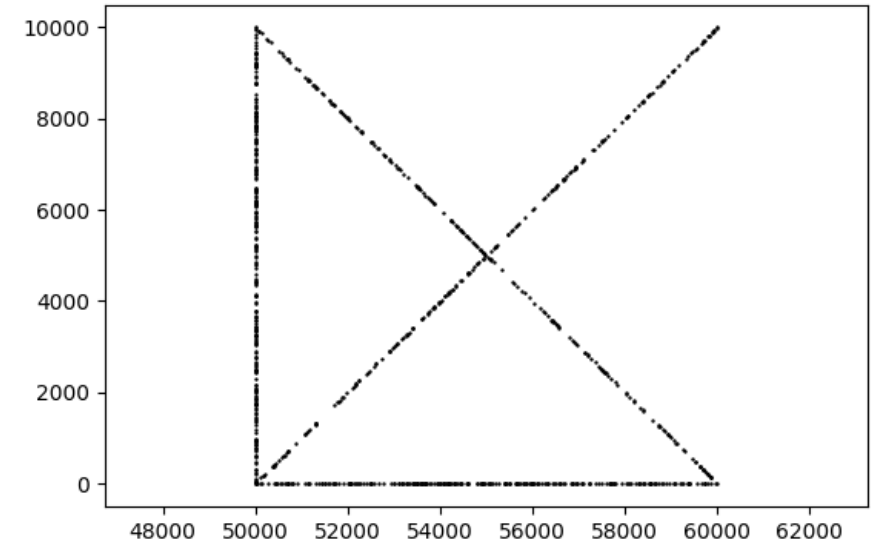
Ryc. 5: Wizualizacja punktów ze zmodyfikowanego zbioru A $(n = 10^4)$



Ryc. 6: Wizualizacja punktów ze zmodyfikowanego zbioru B $(n = 10^3)$



Ryc. 7: Wizualizacja punktów ze zmodyfikowanego zbioru C ($n = 10^3$)



Ryc. 8: Wizualizacja punktów ze zmodyfikowanego zbioru D ($n = 10^3$)

6.2 Wyniki wyznaczania otoczek – zbiory oryginalne

Podczas pracy z oryginalnymi zbiorami, do porównywania kątów używano wyznacznika odpowiedniej macierzy 3x3 w implementacji własnej, dla którego przyjmowano tolerancję dla zera $\varepsilon = 10^{-12}$.

Poniżej przedstawiono wyniki działania funkcji wyznaczania otoczki wypukłej dla konkretnych zbiorów.

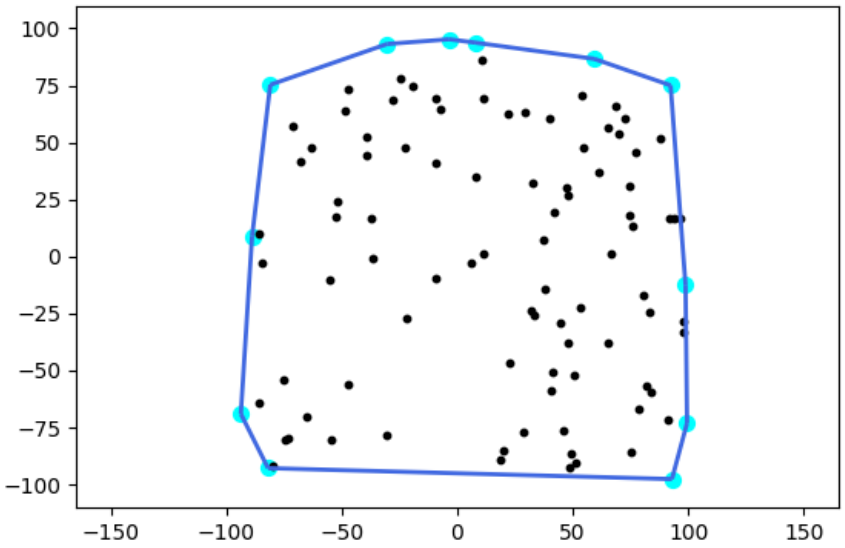
Tabela 1: Wyniki i czas działania funkcji wyznaczania CH dla oryginalnych zbiorów ($\varepsilon = 10^{-12}$)

Zbiór	**Algorytm**	**Liczba punktów otoczki**	**Czas działania funkcji [s]**
A	Grahama	12	0.019
A	Jarvisa	12	0.022
B	Grahama	100	0.020
B	Jarvisa	100	0.045
C	Grahama	8	0.022
C	Jarvisa	8	0.021
D	Grahama	4	0.019
D	Jarvisa	4	0.021

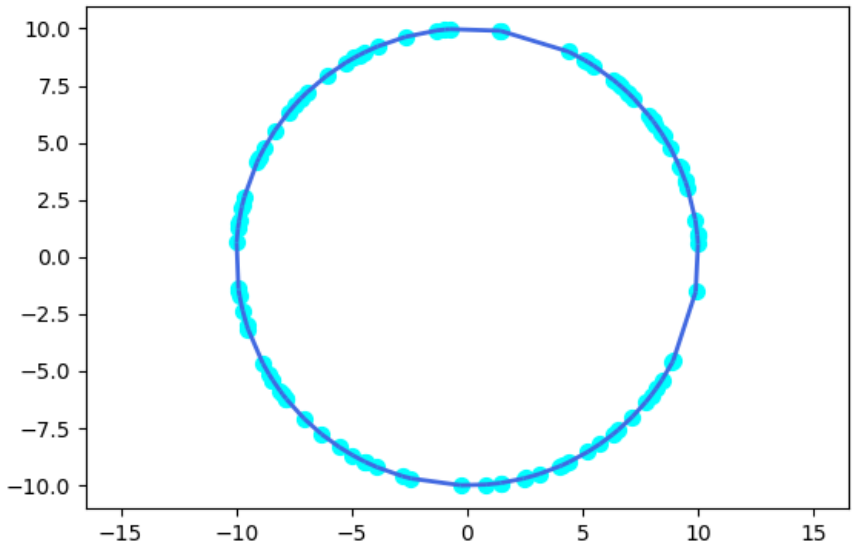
Fakt, że wszystkie wywołania funkcji dały te same wyniki, pozwala upewnić się o poprawności własnej implementacji obu algorytmów wyznaczania otoczki wypukłej.

Czasy działania funkcji są podobne we wszystkich przypadkach za wyjątkiem zbioru B, gdzie algorytm Grahama radzi sobie ponad dwukrotnie szybciej – co jest oczywiste, gdyż liczebność otoczki jest tam równa liczebności zbioru, w związku z czym algorytm Jarvisa osiąga swą pesymistyczną złożoność.

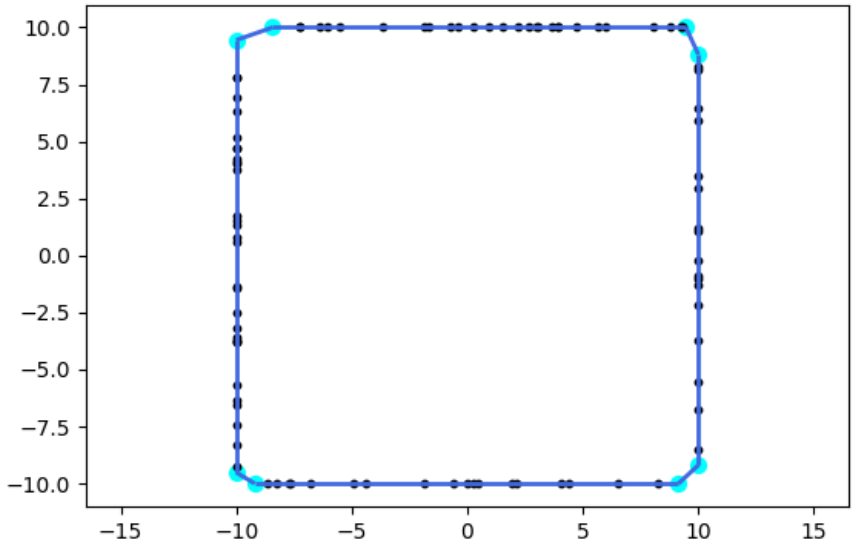
Otrzymane otoczki przedstawione są na poniższych rycinach. Punkty należące do otoczki wyróżniono poprzez pogrubienie i zaznaczenie kolorem cyjanowym, zaś krawędzie otoczki – poprzez zaznaczenie kolorem niebieskim.



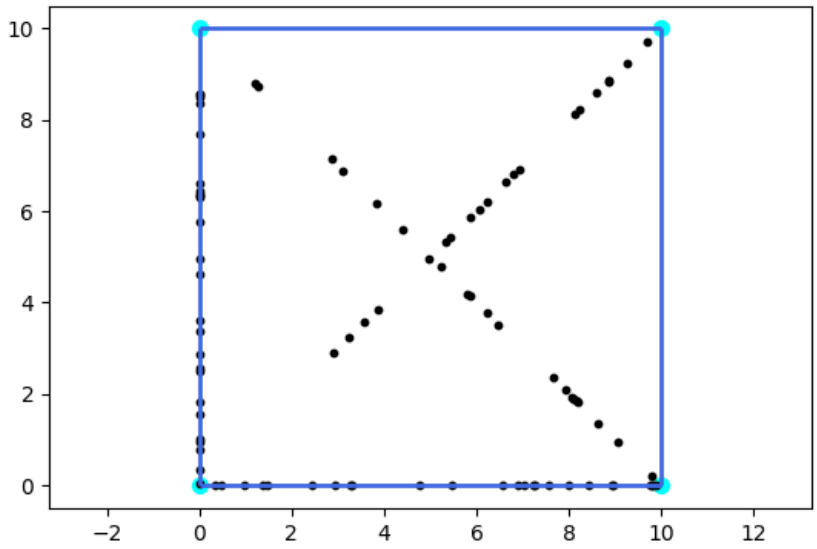
Ryc. 9: Wizualizacja otoczki wypukłej zbioru A



Ryc. 10: Wizualizacja otoczki wypukłej zbioru B



Ryc. 11: Wizualizacja otoczki wypukłej zbioru C



Ryc. 12: Wizualizacja otoczki wypukłej zbioru D

Wśród otrzymanych wyników nie sposób znaleźć żadnych błędów.

Pełne wizualizacje kroków działania poszczególnych algorytmów dostępne są w załączonym notebooku Jupyter. Krawędzie już obecne w otoczce zaznaczono tam kolorem zielonym, krawędź aktualnie rozważaną – kolorem czerwonym.

6.3 Wyniki wyznaczania otoczek – zbiory zmodyfikowane

Otoczki wyznaczono również dla zbiorów zdefiniowanych w sekcji 5.1.2.

Ponownie, do porównywania kątów używano wyznacznika odpowiedniej macierzy 3x3 w implementacji własnej, dla którego nominalnie przyjmowano tolerancję dla zera $\varepsilon = 10^{-12}$, w niektórych przypadkach (tj. dla algorytmu Grahama na zmodyfikowanym zbiorze D) decydowano się jednak na zwiększenie tolerancji do $\varepsilon = 10^{-6}$ z racji na rażąco niepoprawne wyniki przy mniejszej.

Limit czasu wykonania funkcji ustalono na 120 sekund.

Tabela 2: Wyniki i czas działania funkcji wyznaczania CH dla zmodyfikowanych zbiorów ($\varepsilon = 10^{-12}$)

Zbiór	**n**	**Algorytm**	**Liczba punktów otoczki**	**Czas działania funkcji [s]**
A	100	Grahama	14	0.022
A	100	Jarvisa	14	0.021
A	1000	Grahama	19	0.049
A	1000	Jarvisa	19	0.065
A	10000	Grahama	27	0.274
A	10000	Jarvisa	27	0.449
A	100000	Grahama	25	2.980
A	100000	Jarvisa	25	3.961
B	100	Grahama	100	0.023
B	100	Jarvisa	100	0.044
B	1000	Grahama	1000	0.066
B	1000	Jarvisa	1000	2.049
B	10000	Grahama	–	*Przekroczony limit czasu*
B	10000	Jarvisa	–	*Przekroczony limit czasu*
B	100000	Grahama	–	*Przekroczony limit czasu*
B	100000	Jarvisa	–	*Przekroczony limit czasu*
C	100	Grahama	8	0.022
C	100	Jarvisa	8	0.021
C	1000	Grahama	8	0.049
C	1000	Jarvisa	8	0.037
C	10000	Grahama	8	0.273
C	10000	Jarvisa	8	0.191
C	100000	Grahama	8	2.781
C	100000	Jarvisa	8	1.758
D	100	Grahama	14	0.020
D	100	Jarvisa	4	0.018
D	1000	Grahama	103	0.034
D	1000	Jarvisa	4	0.032
D	10000	Grahama	763	0.269
D	10000	Jarvisa	4	0.097
D	100000	Grahama	–	*Przekroczony limit czasu*
D	100000	Jarvisa	4	0.768

- W przypadku zbioru A wyniki nie budzą wątpliwości, wszędzie są prawidłowe.

  Zauważyć można, że dla większych wartości n przewagę czasową zaczyna uzyskiwać algorytm Grahama.

- Zbiór B prezentuje pierwsze poważne problemy – dla $n \in \{10^4, 10^5\}$ oba algorytmy przekroczyły założony limit czasu 120 s. Sytuacja ta jest zrozumiała dla algorytmu Jarvisa, lecz dość zadziwiająca dla algorytmu Grahama, gdyż złożoność liniowo-logarytmiczna nie powinna powodować aż tak drastycznego wzrostu czasu wykonania. Ponowne przeprowadzenie obliczeń dla $\varepsilon = 10^{-6}$ nie przyniosło żadnej różnicy, limit czasu znów został przekroczony. Autorowi nie udało się dociec przyczyny takiego stanu rzeczy.

  Podobnie jak dla oryginalnego zbioru B, algorytm Jarvisa czasowo radzi sobie o wiele gorzej.

- Dla zbioru C wyniki nie różnią się, wszędzie są poprawne.

  Zbiór ten jednak po raz pierwszy obrazuje sytuację, gdzie algorytm Jarvisa działa szybciej od algorytmu Grahama. Wynika to oczywiście z faktu ograniczonej liczby punktów należących do otoczki.

- Zbiór D prezentuje problemy algorytmu Grahama, który w każdym z przypadków daje zupełnie niepoprawne wyniki[1]. Jeden z takich wyników przytoczony zostanie na rycinie poniżej.

  Podjęto próbę ponownego wygenerowania punktów, w pobliżu środka układu współrzędnych, a następnie przesunięcia ich na podobną odległość. Wyniki były identyczne, tj. algorytm Grahama działał poprawnie dla nieprzesuniętych punktów, zaś dla przesuniętych zwracał błędne wyniki, podobnie gdyby były generowane jako już przesunięte. Pozwoliło to wyeliminować możliwość wadliwej generacji punktów.

  Użycie innej metody sortowania punktów również nie przyniosło żadnych zmian.

  Zauważono jednak pewną prawidłowość – liczba punktów wyeliminowanych podczas sprawdzania współliniowości uległa zmianie. Gdy wygenerowano po 2500 punktów na przekątnych, po 2000 punktów na bokach i 4 w wierzchołkach, w przypadku nieprzesuniętych punktów otrzymywano 2503 punkty po eliminacji, co odpowiada jednej z przekątnych i trzem wierzchołkom – w przypadku przesuniętych/generowanych w przesunięciu punktów natomiast, uzyskiwano większe ich liczby, odpowiednio 3787 i 3883. Oznacza to, że niektóre punkty nie zostały odrzucone, co prowadzi do następującego wniosku – problem leży po stronie wyznacznika.

  Z jakiegoś powodu, zmodyfikowanie dokładnie jednej (sic!) ze współrzędnych o pokaźną wielkość dla wszystkich punktów spowodowało drastyczny spadek dokładności wyliczanego wyznacznika, co w konsekwencji doprowadziło do błędnego zaklasyfikowania faktycznie współliniowych punktów jako niewspółliniowe. Finalnie, autorowi nie udało się zgłębić przyczyny takiego stanu rzeczy.

[1] Z racji na ten fakt, nie jest możliwe porównanie czasu wykonania funkcji dla tego zbioru i tej dokładności

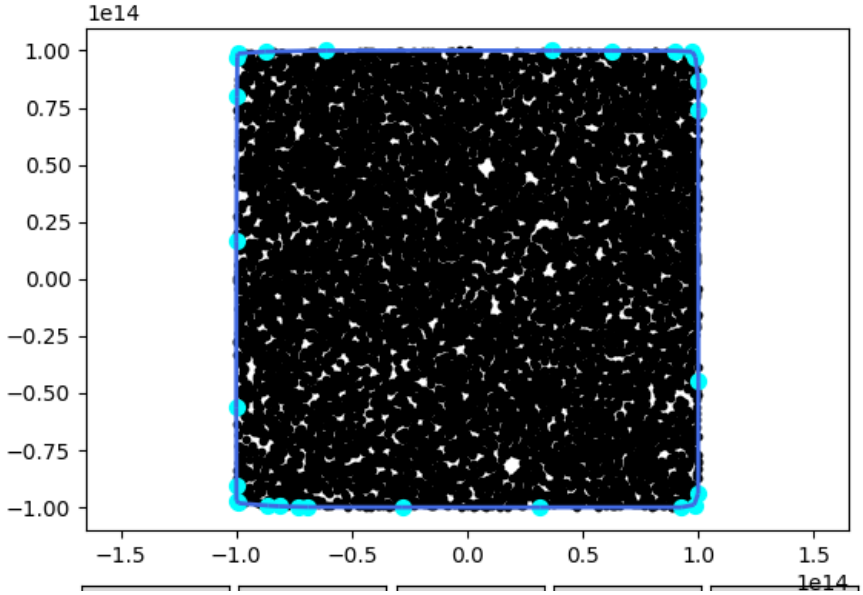
W celu zachowania ciągłości wniosków, zdecydowano się na niezmienianie zbioru D – zamiast tego, obliczono otoczki ponownie, z tolerancją dla zera $\varepsilon = 10^{-6}$.

Tabela 3: Wyniki i czas działania funkcji wyznaczania CH dla zmodyfikowanego zbioru D ($\varepsilon = 10^{-6}$)

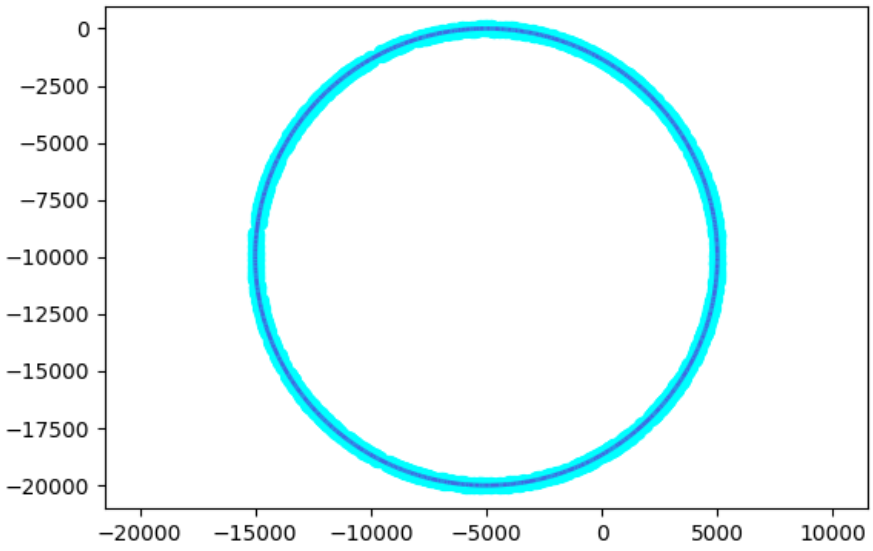
Zbiór	**n**	**Algorytm**	**Liczba punktów otoczki**	**Czas działania funkcji [s]**
D	100	Grahama	4	0.021
D	100	Jarvisa	4	0.020
D	1000	Grahama	4	0.040
D	1000	Jarvisa	4	0.031
D	10000	Grahama	4	0.175
D	10000	Jarvisa	4	0.094
D	100000	Grahama	4	1.660
D	100000	Jarvisa	4	0.795

Wyniki uzyskane przy większej tolerancji dla zera są poprawne we wszystkich przypadkach. Pozwala to, ponownie, na zaobserwowanie szybszego działania algorytmu Jarvisa przy ograniczonej liczbie punktów należących do otoczki.

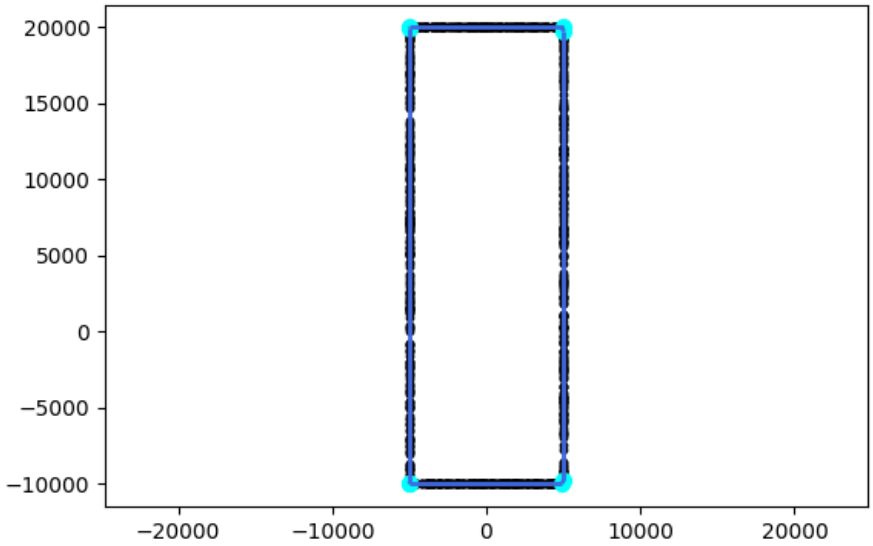
Uzyskane otoczki dla przykładowych zmodyfikowanych zbiorów przedstawiono na poniższych rycinach:



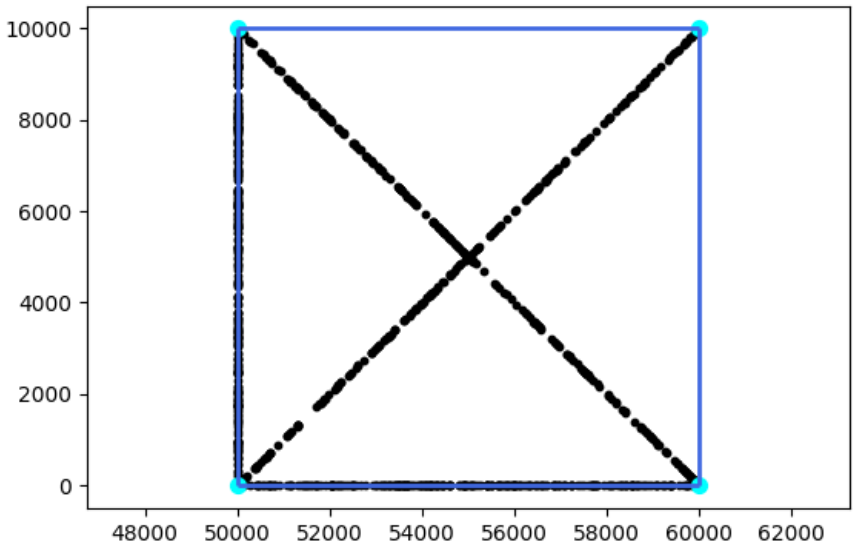
Ryc. 13: Wizualizacja otoczki wypukłej zmodyfikowanego zbioru A ($n = 10^4$)



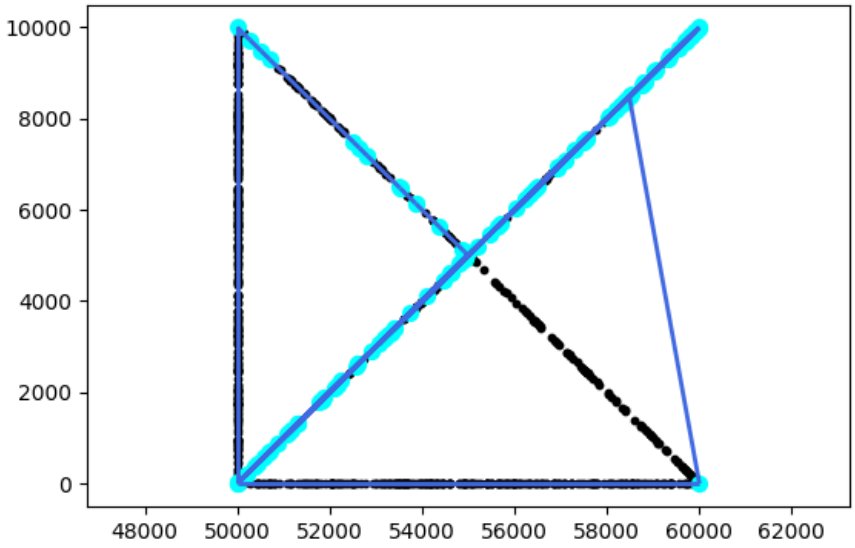
Ryc. 14: Wizualizacja otoczki wypukłej zmodyfikowanego zbioru B ($n = 10^3$)



Ryc. 15: Wizualizacja otoczki wypukłej zmodyfikowanego zbioru C ($n = 10^3$)



Ryc. 16: Wizualizacja otoczki wypukłej zmodyfikowanego zbioru D ($n = 10^3$, alg. Jarvisa)



Ryc. 17: Wizualizacja błędnie wyznaczonej otoczki wypukłej zmodyfikowanego zbioru D ($n = 10^3$, $\varepsilon = 10^{-12}$, alg. Grahama)

7 Wnioski

- Zaproponowane zbiory punktów zostały specjalnie dobrane, by ukazać możliwe problemy przy wyznaczaniu otoczki wypukłej zbioru punktów na płaszczyźnie,
- Problemy w działaniu algorytmów mogły sprawiać wszystkie zbiory, w szczególności B - problem czasowy, zwłaszcza dla algorytmu Jarvisa (wszystkie punkty ze zbioru należące do otoczki); C, D - błędne wyznaczenie otoczki, problem czasowy (w mniejszym stopniu; wiele punktów współliniowych), zwłaszcza dla algorytmu Grahama,
- W zasadzie każdy zbiór, odsunięty od środka układu współrzędnych, będzie sprawiał problemy dla algorytmu Grahama, w związku ze spadkiem dokładności wyznacznika,
- Algorytm Grahama lepiej się sprawdza dla liczniejszych otoczek, algorytm Jarvisa – dla otoczek o ograniczonej liczbie punktów.