Sprawozdanie z ćwiczenia trzeciego

Algorytmy Geometryczne

Iwo Szczepaniak, Windows 11 – 4,2GHz

Jupyter Notebook, VS Code Python 3.9.12

1) Wstęp

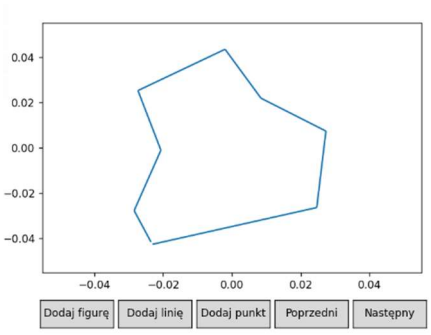
Zadaniem które należy wykonać na laboratorium 3 jest implementacja oraz przetestowanie czterech algorytmów:

- algorytm umożliwiający zadawanie wielokątów przy użyciu myszki, ich zapis oraz odczyt
- algorytm sprawdzający czy dany wielokąt jest y-monotoniczny(względem osi OY)
- algorytm klasyfikujący wierzchołki wielokąta jako: początkowe, końcowe, łączące, dzielące i prawidłowe wraz z ich wizualizacją poprzez odpowiednie kolorowanie
- algorytm wykonujący triangulację wielokąta y-monotonicznego wraz z wizualizacją powstawania kolejnych trójkątów

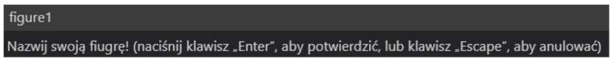
2) Opis algorytmów

2.1 Rysowanie figur przy użyciu myszki, ich zapis oraz odczyt:

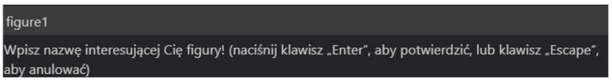
Rysowanie możliwe jest wewnątrz pustego plot'u w nagłówku „Narysuj figurę". Następnie, by zapisać figurę należy przejść do nagłówka „Zapisz narysowaną figurę". W nim użyta została funkcja plot.get_added_figure(), która wczytuje narysowaną figurę i po przetworzeniu(poprawieniu figury, jeśli ostatnia krawędź nie kończy się w punkcie początkowym) zapisuje pod nazwą wskazaną przez użytkownika w folderze projektu. Odczytanie wybranej figury możliwe jest w nagłówku „Zobacz wybraną figurę". Użytkownik musi podać nazwę pliku, w którym zapisana jest interesująca go figura.



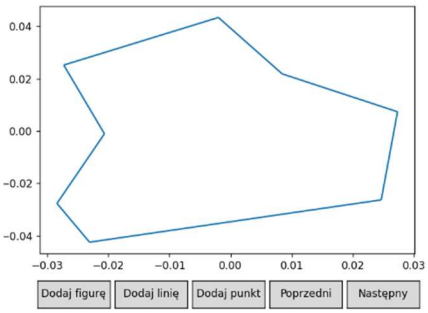
Rys1. rysowanie figury



Rys2. nazwanie figury



Rys3. wczytywanie figury z pliku



Rys4. wyświetlenie figury

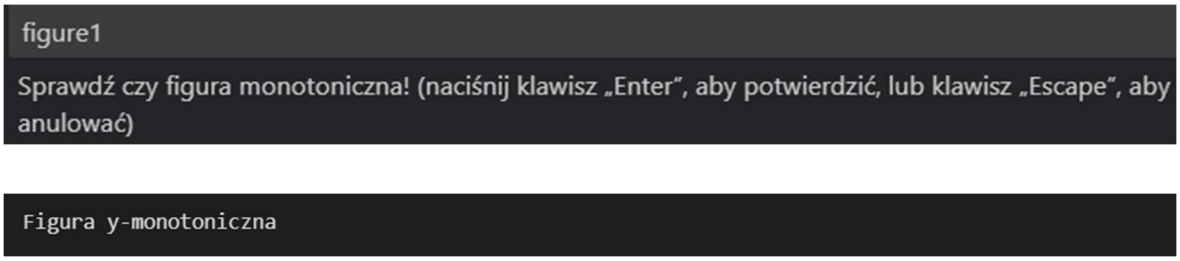
2.2 Y-monotoniczność figury:

Algorytm wykorzystuje twierdzenie z wykładu które mówi, że dla figura jest y-monotoniczna, gdy nie ma wierzchołków dzielących i łączących.

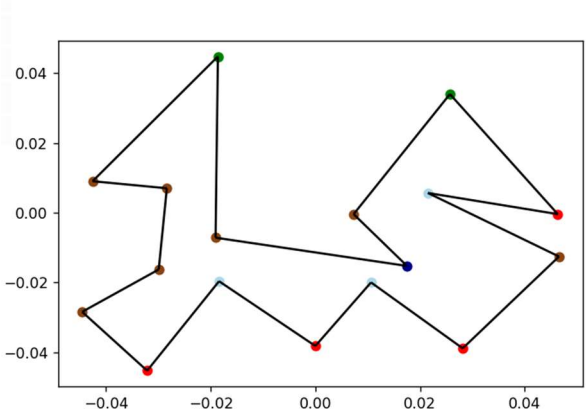
Algorytm polega na przejściu „trójkami” przez daną figurę i sprawdzanie jakim punktem(kolorem) jest środkowy wierzchołek. Gdy okaże się że jakikolwiek z wierzchołków jest dzielący lub łączący, zwracany jest False. Jeśli po przejściu przez wszystkie wierzchołki nie napotkamy dzielących i łączących punktów, to zwracany jest True.

Złożoność obliczeniowa: n
n – jednokrotne odwiedzenie każdego z n punktów O(n) i sprawdzenie czy nie są łączące bądź dzielące O(1)

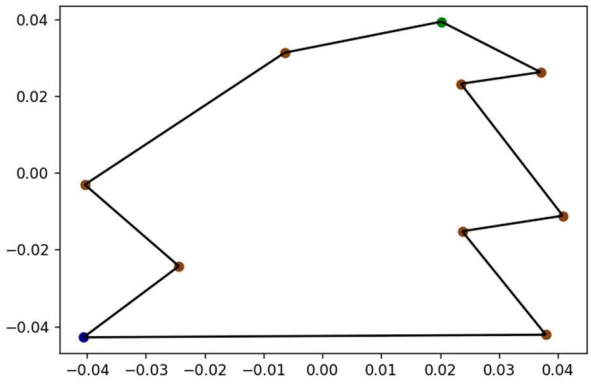
Tab1. Złożoność algorytmu 2)



Rys5. Sprawdzanie czy figura jest monotoniczna



Rys5a. Przykładowa figura niemonotoniczna



Rys5b. Przykładowa figura monotoniczna

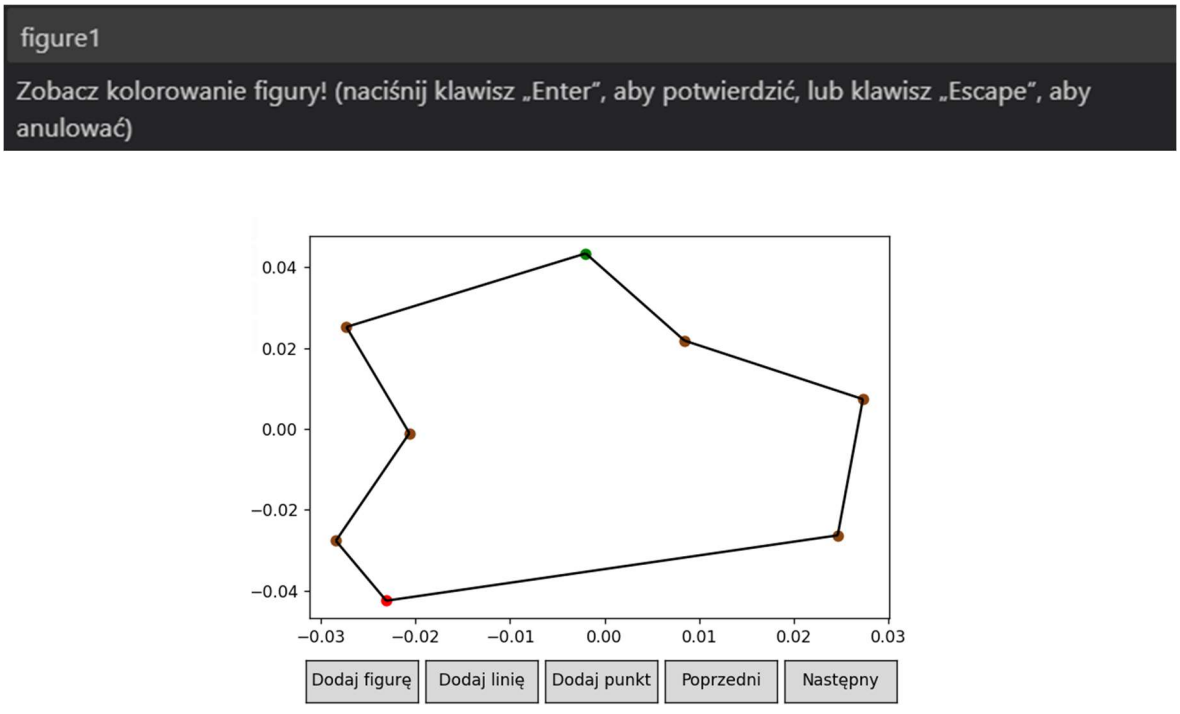
2.3 Klasyfikacja i kolorowanie wierzchołków:

Algorytm polega na przeglądnięciu wszystkich wierzchołków wielokąta i określeniu na podstawie wysokości sąsiadów aktualnie sprawdzanego wierzchołka oraz kąta jaki z nimi tworzy ten wierzchołek czy jest on:

- wierzchołkiem początkowym - obaj sąsiedzi leżą poniżej i kąt wewnętrzny tworzony przez wierzchołek i jego sąsiadów jest < π - kolor zielony
- wierzchołkiem końcowym - obaj sąsiedzi leżą powyżej i kąt wewnętrzny tworzony przez wierzchołek i jego sąsiadów jest < π – kolor czerwony
- wierzchołkiem łączącym - obaj sąsiedzi leżą powyżej i kąt wewnętrzny tworzony przez wierzchołek i jego sąsiadów jest > π - kolor ciemnoniebieski
- wierzchołkiem dzielącym - obaj sąsiedzi leżą poniżej i kąt wewnętrzny tworzony przez wierzchołek i jego sąsiadów jest > π – kolor jasnoniebieski
- wierzchołkiem prawidłowym - w pozostałych przypadkach – kolor brązowy

Złożoność obliczeniowa: n
n – jednokrotne odwiedzenie każdego z n punktów O(n) i przyporządkowanie do odpowiedniego zbioru O(1)

Tab2. Złożoność algorytmu 3)



Rys6. Wizualizacja kolorowania

2.4 Triangulacja figury:

Wejściowym warunkiem jest sprawdzenie czy figura jest y-monotniczna, do sprawdzenia tego jest wykorzystywana funkcja z punktu 2.2. Następnie dzielimy wierzchołki na lewy łańcuch i prawy łańcuch. Następnie sortujemy listę wierzchołków po współrzędnej y malejąco oraz zapisujemy oryginalną listę(po to by sprawdzać sąsiedztwo). Pierwsze dwa elementy posortowanej listy wrzucamy na stos.

Przeglądamy wszystkie pozostałe wierzchołki z posortowanej listy i dodajemy krawędzie tworzące trójkąty. Jednak dodawanie krawędzi ma następujące warunki:

- jeśli aktualnie rozpatrywany wierzchołek znajduje się na innym łańcuchu niż szczyt stosu, to łączymy go z wszystkimi wierzchołkami które znajdują się na stosie i nie są jego sąsiadami, po wykonaniu całej operacji na stosie zostawiamy dwa ostatnio analizowane wierzchołki,
- jeśli aktualnie rozpatrywany wierzchołek znajduje się na tym samym łańcuchu co szczyt stosu, to rozpatrujemy dwa przypadki:
    - Utworzony trójkąt należy do wnętrza wielokąta oraz szczyt stosu nie jest sąsiadem aktualnie rozpatrywanego wierzchołka.
      Wtedy usuwamy wierzchołek ze stosu a wierzchołki łączymy krawędzią
    - Utworzony trójkąt nie należy do wnętrza wielokąta.
      Wtedy umieszczamy badane wierzchołki na stosie.

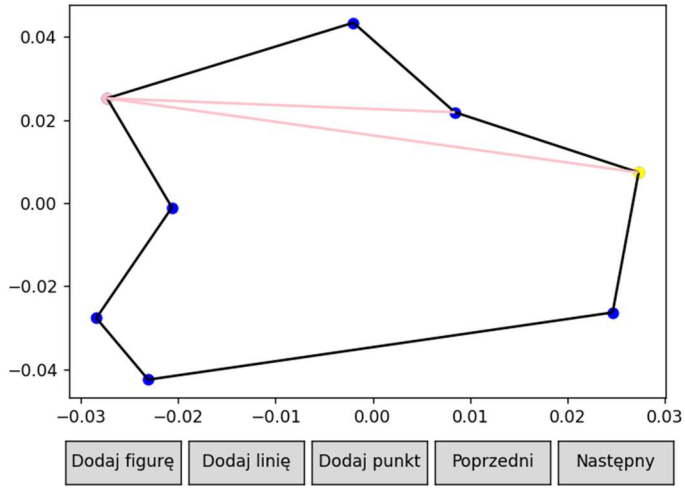
Algorytm zwraca listę par indeksów tworzących triangulację oraz sceny, które zostały dodane w trakcie działania algorytmu wizualizujące działanie algorytmu. Triangulacja jest kolorowana na różowo, a obecnie rozpatrywany wierzchołek na żółto.

Wielokąt jest przechowywany jako dwie listy – punktów i krawędzi. Triangulacja jest przechowywana jako list. Wybrano te metody ze względu na prostą implementację.

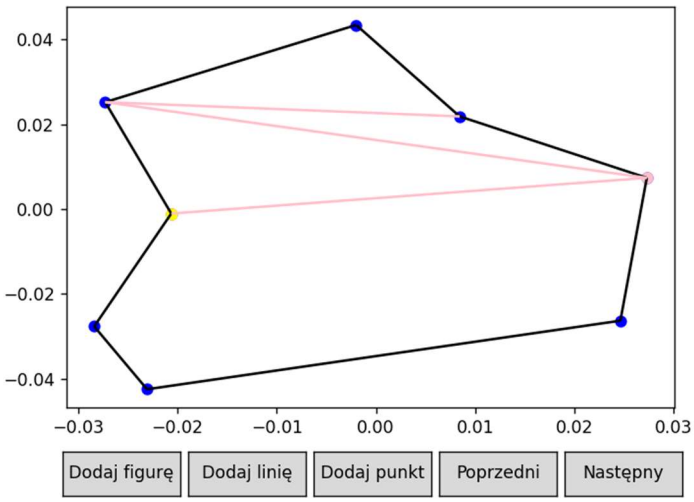
Jednokrotne odwiedzenie każdego z n punktów - O(n) funkcje pomocnicze: highest_points – O(n) – na zewnątrz pętli find_chains – O(n) – na zewnątrz pętli same_chains – O(1) triangle_inside - O(1) neighbours - O(1) **Złożoność obliczeniowa : n**

Tab3. Złożoność algorytmu 4)

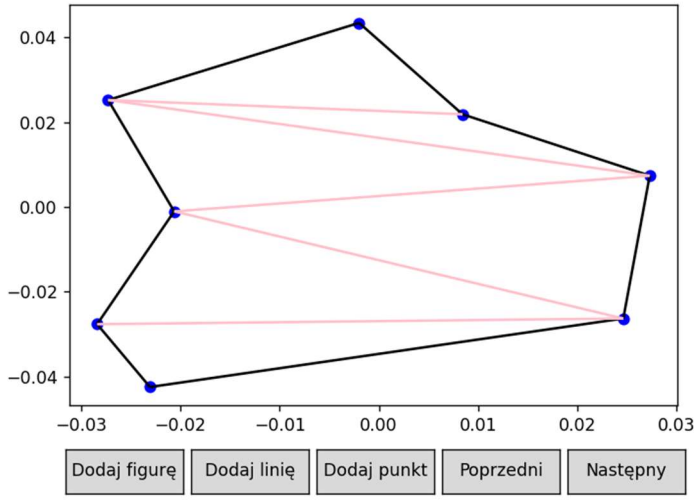
Wizualizacja triangulacji



Rys7. Początek triangulacji

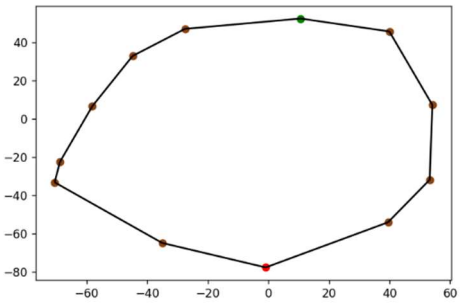


Rys8. Kolejny krok triangulacji

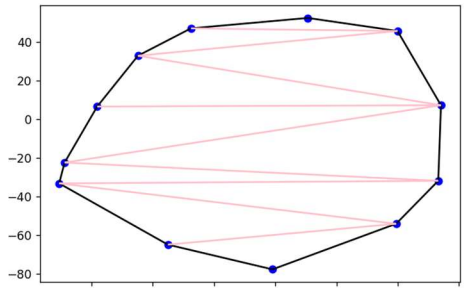


Rys9. Pełna wizualizacja triangulacji

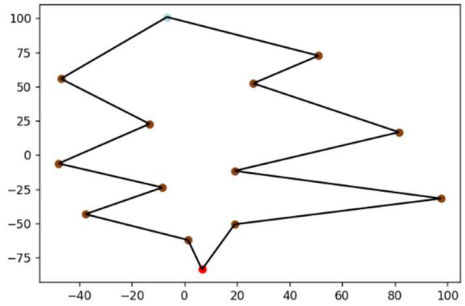
3) Różne figury – kolorowanie i triangulacja



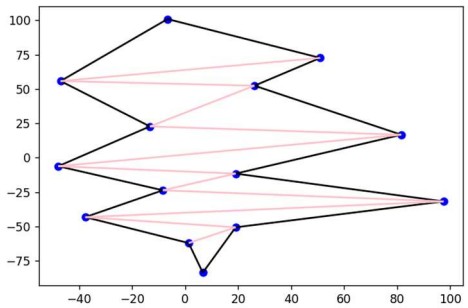
Rys10.



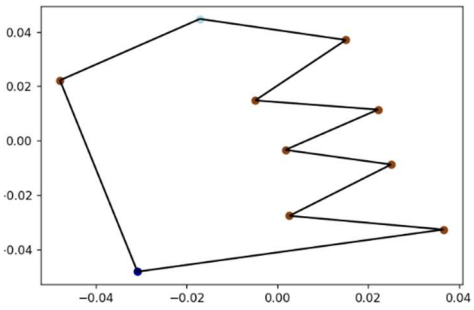
Rys11.



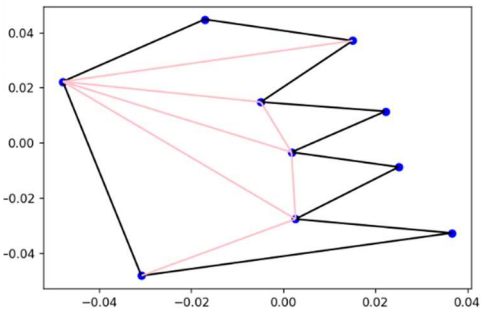
Rys12.



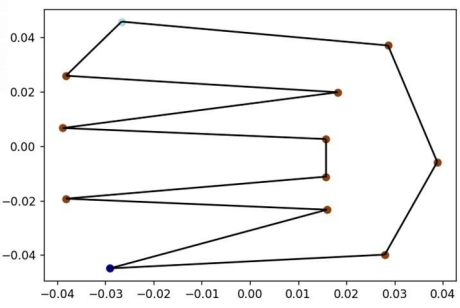
Rys13.



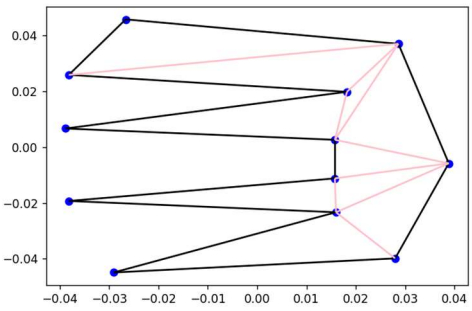
Rys14.



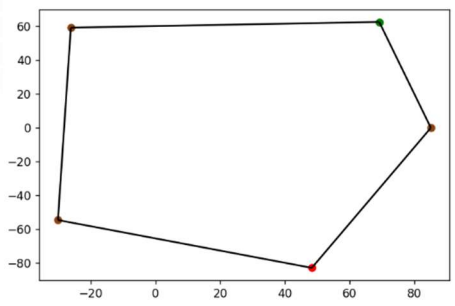
Rys15.



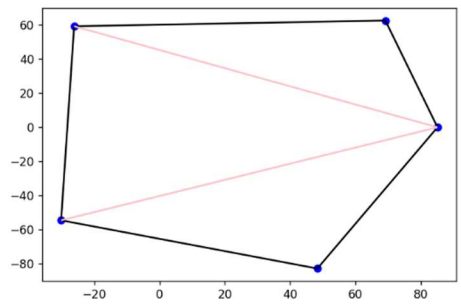
Rys16.



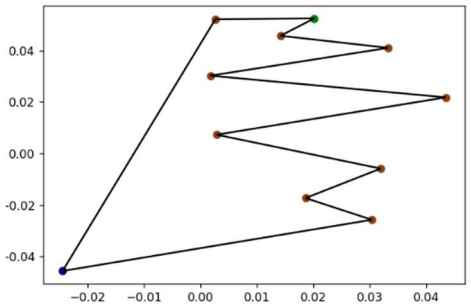
Rys17.



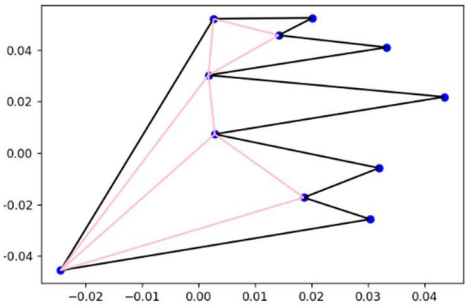
Rys18.



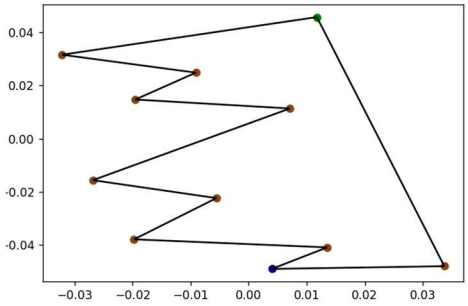
Rys19.



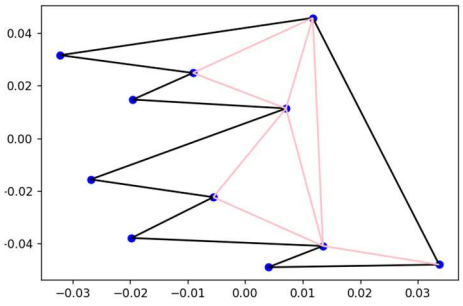
Rys20.



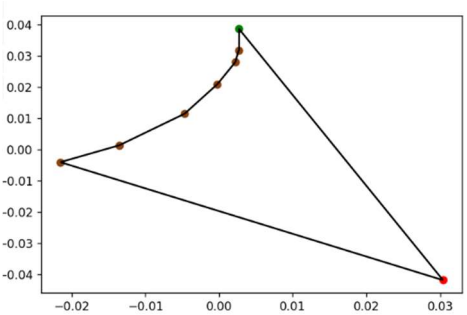
Rys21.



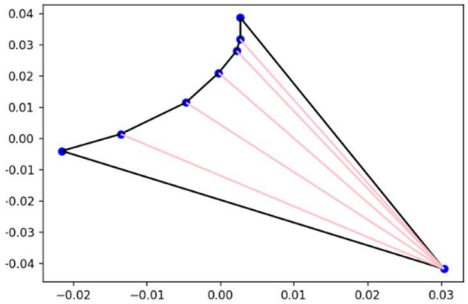
Rys22.



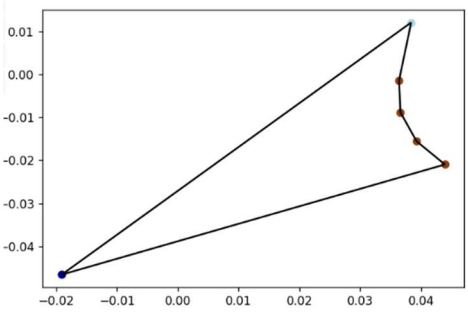
Rys23.



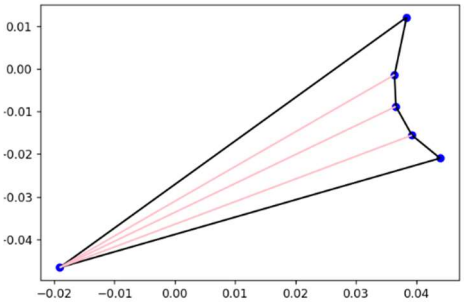
Rys24.



Rys25.



Rys26.



Rys27.

4) Wnioski:

Zbiory punktów dostosowano tak, by znaleźć możliwie dziwne figury oraz te możliwie proste.

Wszystkie algorytmy po ich przetestowaniu na powyższych zbiorach, działają poprawnie i nie zauważono w nich żadnych błędów w działaniu. Można zatem założyć, że są one poprawnie zaimplementowane.