**Algorytmy geometryczne**

Sprawozdanie z ćwiczenia 3.

Paweł Lamża

Dane techniczne urządzenia na którym wykonano ćwiczenie:

Laptop z systemem Windows 10 x64

Procesor: AMD Ryzen™ 5 4600H

Pamięć RAM: 16GB

Środowisko: Jupyter notebook

Ćwiczenie zrealizowano w języku Python 3, z wykorzystaniem bibliotek numpy oraz matplotlib

**Opis ćwiczenia**

Ćwiczenie polegało na zaimplementowaniu algorytmów:

- Sprawdzania y-monotoniczności dowolnego wielokąta

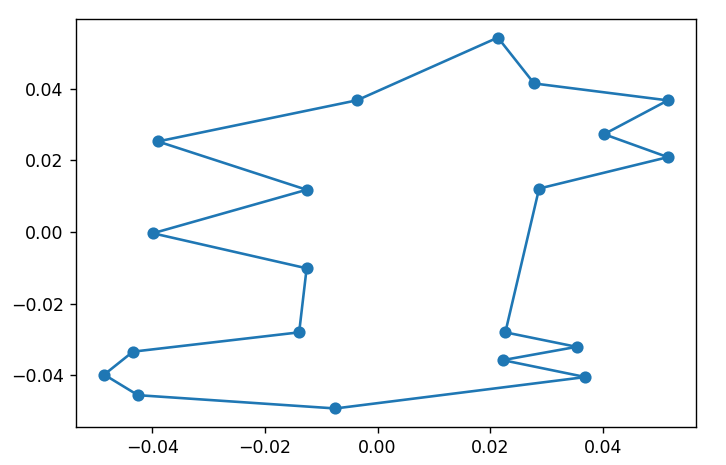
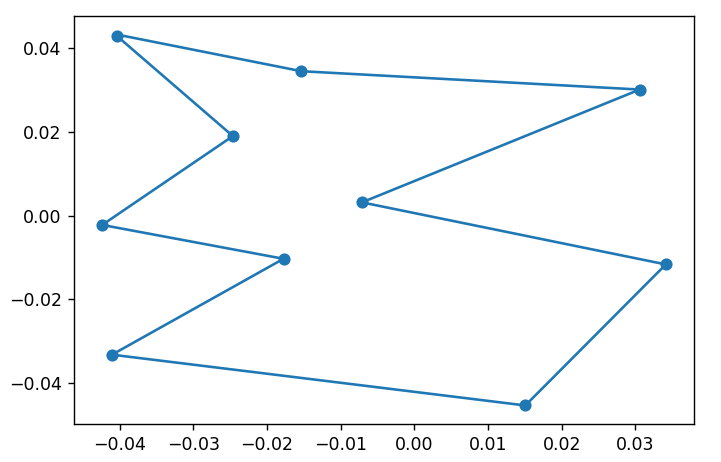
- Kwalifikacji punktów dowolnego wielokąta

- Triangulacji wielokąta y-monotonicznego

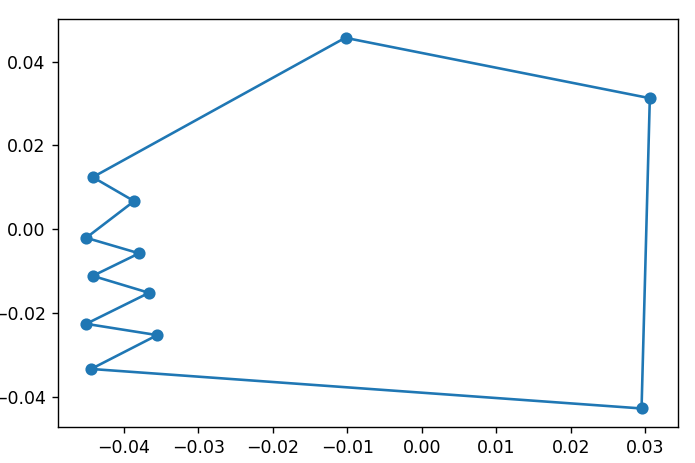
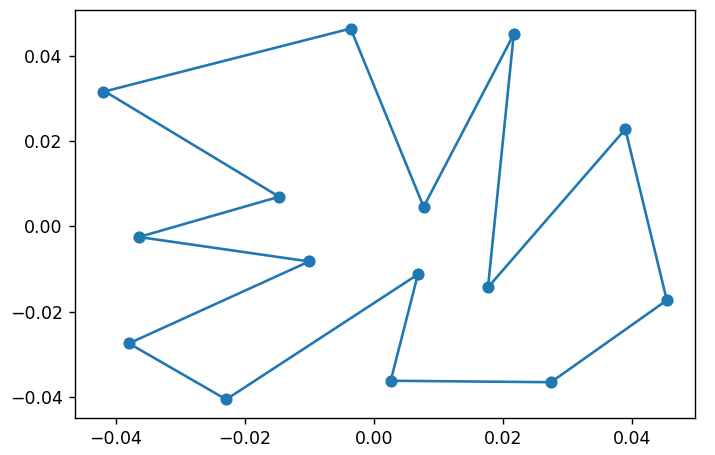
1. Generacja figur.

W celu wykonania ćwiczenia wygenerowano 6 figur w tym jedną

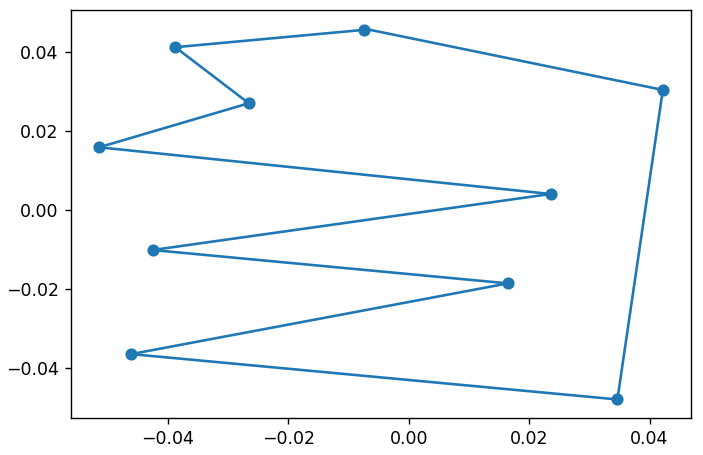
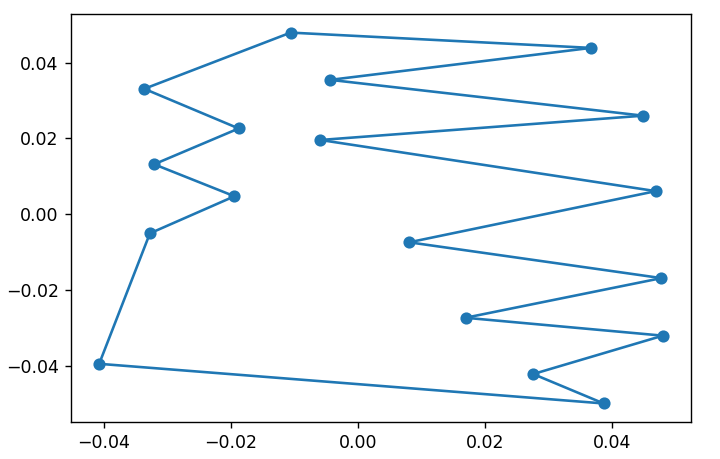
nie y-monotoniczną i 5 y-monotonicznych. Figura 2. i 5. są stosunkowo proste, figury 1. i 6. są mocno skomplikowane. Figura 4. jest przykładem wielokąta niemonotonicznego co oznacza, że występują w niej wierzchołki łączące i dzielący o których więcej można przeczytać dalej. Figura 3. Sprawdza zachowanie algorytmów w sytuacji gdy krawędzi są bliskie nakładania się, bycia współliniowymi. Poniżej kolejne figury:

Wykres 1.1, Figura 1 Wykres 1.2, Figura 2

Wykres 1.3, Figura 3 Wykres 1.4, Figura 4

Wykres 1.5, Figura 5 Wykres 1.6, Figura 6

2. Metoda obliczania wyznacznika i tolerancja dla zera.

Do obliczania wyznacznika użyłem wyznacznika macierzy 3x3 własnej implementacji, a jako tolerancje dla zera przyjąłem 10^-12.

3. Algorytm sprawdzania y-monotoniczności dowolnego wielokąta.

Opis algorytmu:

Iterujemy po kolejnych punktach w kolejności w jakiej występują na wielokącie (odwrotnie do duchu wskazówek zegara). I liczymy ile razy zostanie zmieniona monotoniczność wzrastania/malenia Y. Tzn. liczymy ile jest miejsc w których Y malał w kolejnych punktach a później zaczął rosnąć lub odwrotnie. Jak można zauważyć każdy wielokąt monotoniczny ma tylko dwa takie punkty (w punktach o najwyższej i najmniejszej współrzędnej Y tj. ‘startowym’ i ‘końcowym’)

4. Algorytm klasyfikacji punktów dowolnego wielokąta.

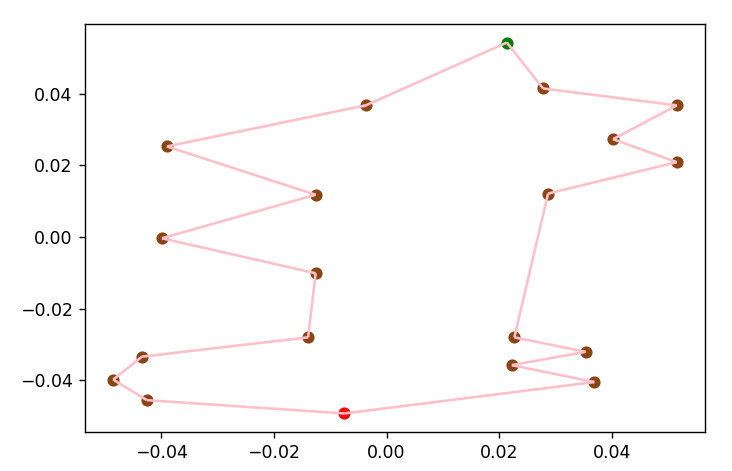
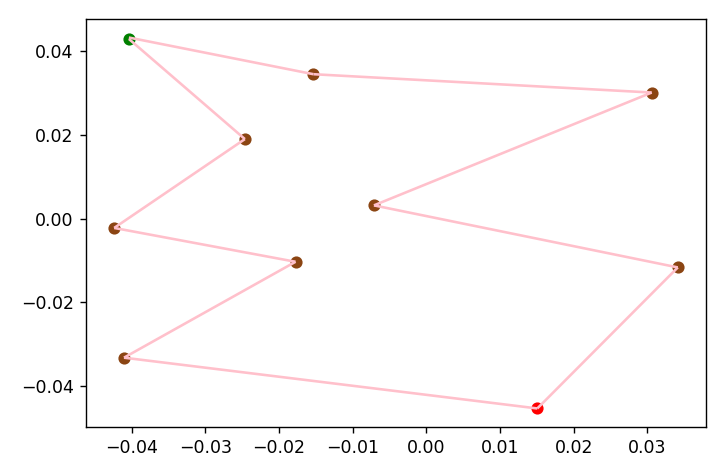
Algorytm polega na podzieleniu punktów wielokąta na 5 kategorii:

* początkowy, gdy obaj jego sąsiedzi leżą poniżej i kąt wewnętrzny < π (ZIELONY)
* końcowy, gdy obaj jego sąsiedzi leżą powyżej i kąt wewnętrzny < π (CZERWONY)
* łączący, gdy obaj jego sąsiedzi leżą powyżej i kąt wewnętrzny > π (NIEBIESKI)
* dzielący, gdy obaj jego sąsiedzi leżą poniżej i kąt wewnętrzny > π (JASNO-NIEBIESKI)
* prawidłowy (ma jednego sąsiada powyżej, drugiego – poniżej) (BRĄZOWY)

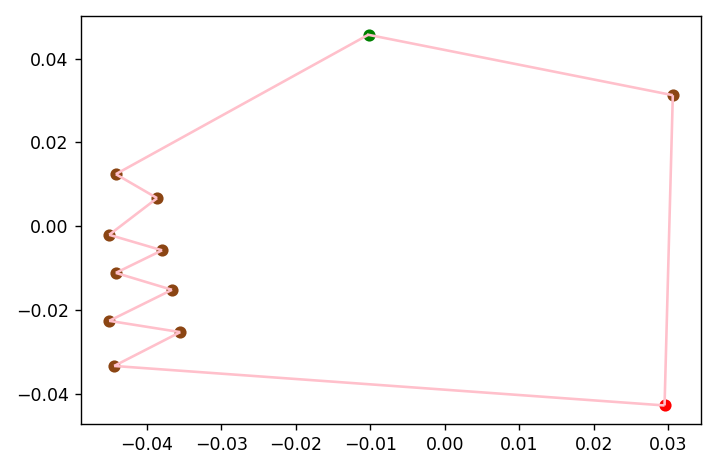
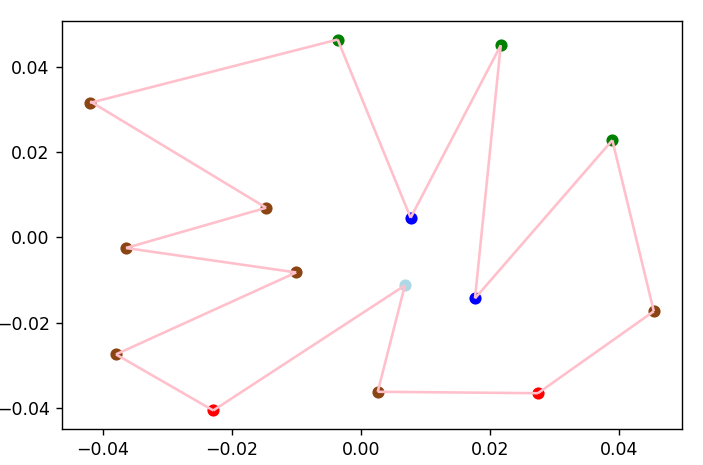
Uwagi implementacyjne:

* do sprawdzenia kąta użyto funkcji orient używającej wyznacznika

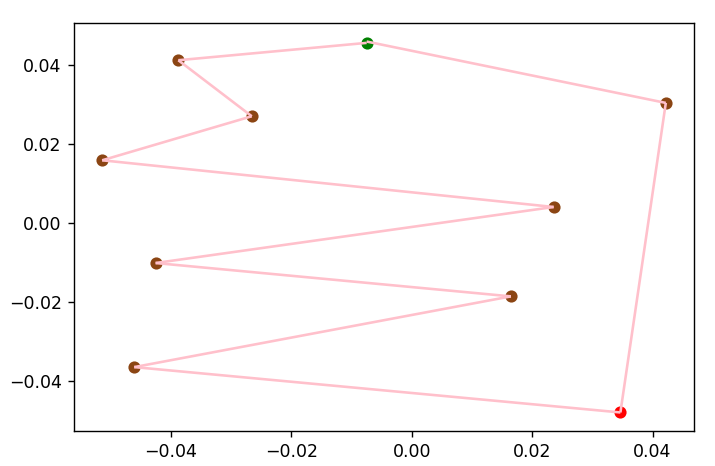
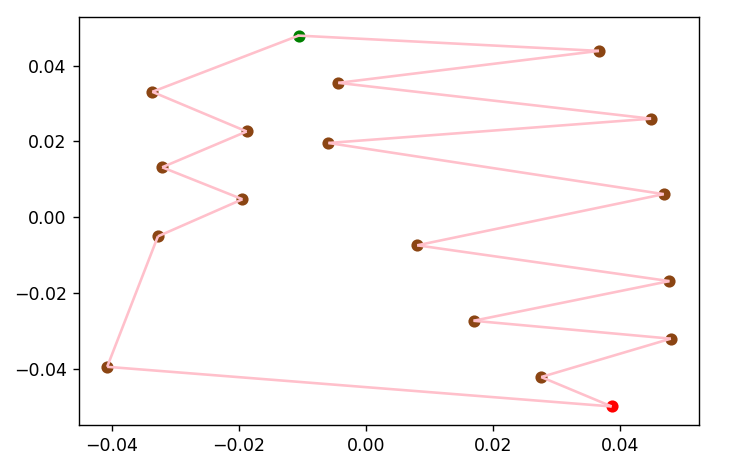
Dla wszystkich zestawów punkty zostały sklasyfikowane poprawnie, tzn. dla wielokątów monotonicznych jeden punkt początkowy i jeden końcowym oraz pozostałe prawidłowe.

Wykres 4.1 Klasyfikacja figura 1 Wykres 4.2 Klasyfikacja figura 2

Wykres 4.3 Klasyfikacja figura 3 Wykres 4.4 Klasyfikacja figura 4

Wykres 4.5 Klasyfikacja figura 5 Wykres 4.6 Klasyfikacja figura 6

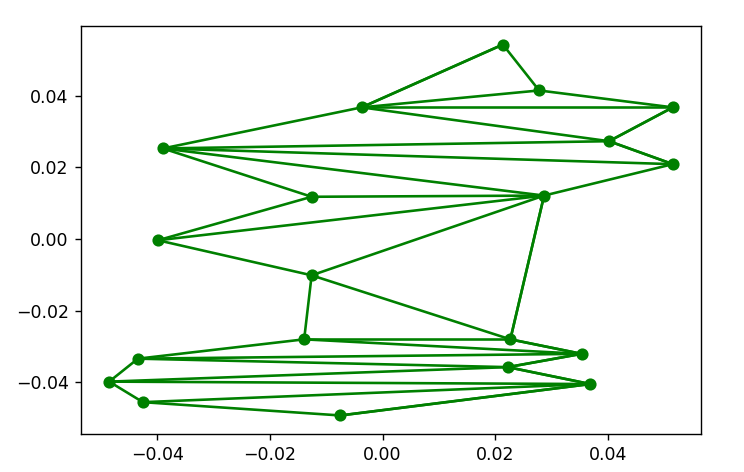
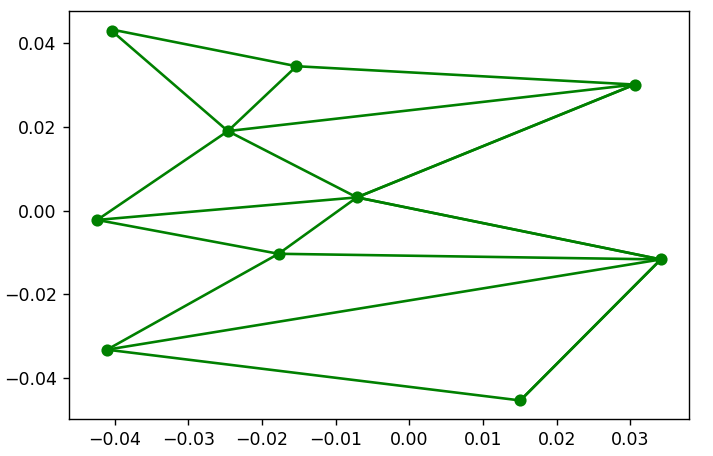
6. Algorytm triangulacji:

Opis działania algorytmu:

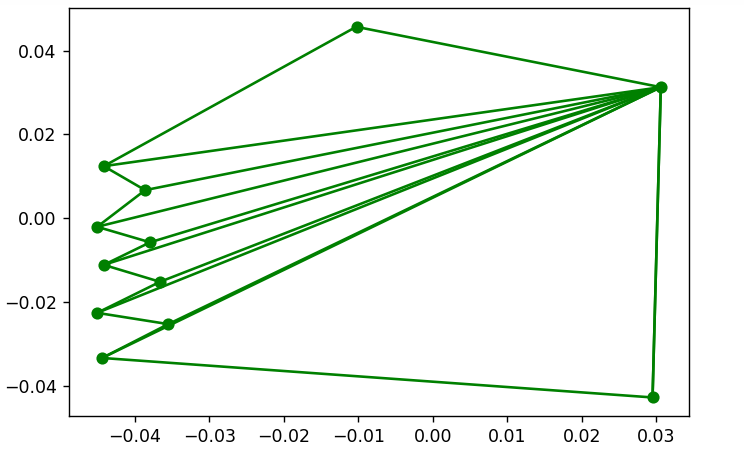
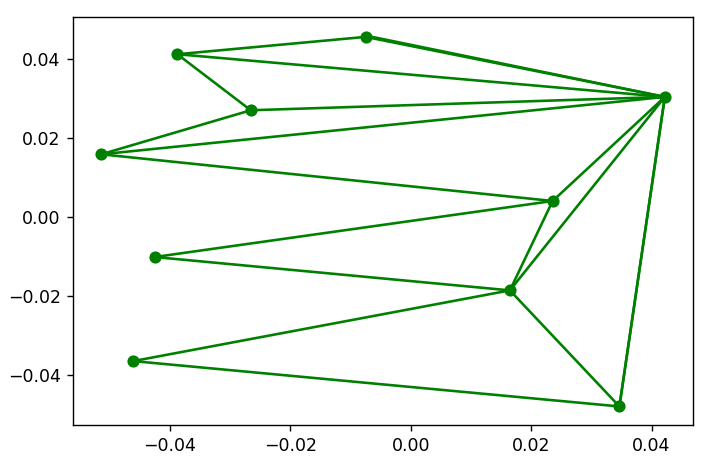
* sprawdzamy czy wielokąt jest y-monotoniczny, jeśli nie to zwracamy informację o tym i kończymy program
* dzielimy punkty na lewą i prawą gałęź, przy czym punkt startowy (najwyższy) uznajemy że jest na prawej gałęzi, a punkt końcowy(najniższy) na lewej gałęzi
* punkty sortujemy ze względu na Y od punktu najwyżej położonego do tego położonego najniżej
* na stosie umieszczamy dwa pierwsze wierzchołki
* następnie przechodzimy do głównej pętli w której rozważamy następne wierzchołki według reguły:
  + Jeśli wierzchołek należy do innego łańcucha niż ostatni wierzchołek na stosie, to łączymy ze wszystkimi wierzchołkami na stosie. Na stosie zostają dwa wierzchołki o najmniejszych Y spośród rozważonych punktów.
  + W przeciwnym przypadku analizujemy kolejne trójkąty, jakie tworzy ten wierzchołek z wierzchołkami zdejmowanymi ze stosu.
    - Jeśli trójkąt należy do wielokąta, to usuwamy wierzchołek ze szczytu stosu
    - w przeciwnym przypadku umieszczamy badane wierzchołki na stosie.

Uwagi implementacyjne:

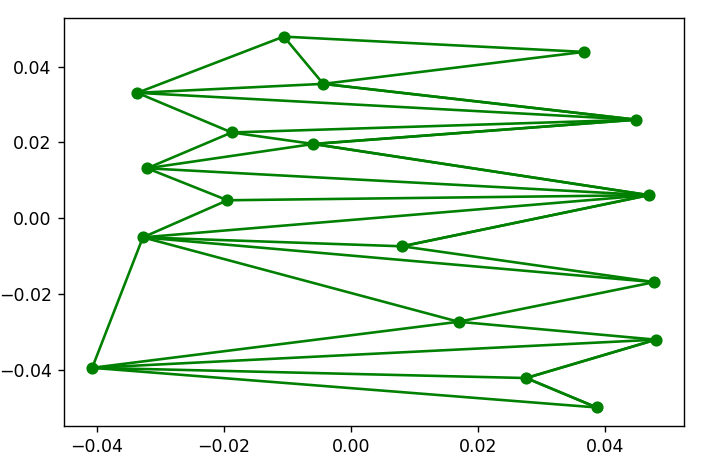
* do sprawdzenia czy trójkąt należy do wielokąta czy nie użyto wyznacznika połączonego z wiedzą na której z gałęzi (lewej czy prawej) znajduje się dany wierzchołek

Wykres 4.1 Triangulacja figura 1 Wykres 4.2 Triangulacja figura 2

Wykres 4.3 Triangulacja figura 3 Wykres 4.4 Triangulacja figura 5



Wykres 4.5 Triangulacja figura 6

7. Implementacja struktur przechowujących wielokąt oraz utworzoną triangulację.

* Wielokąt jest przechowywany jako lista punktów wraz z listą krawędzi
* Funkcja odpowiedzialna za triangulację zwraca jako wynik listę wszystkich krawędzi w powstałej figurze, oraz listę scen, przedstawiających kolejne etapy triangulacji

8. Wnioski i spostrzeżenia

* Wszystkie zaimplementowane algorytmy zadziałały poprawnie dla wielokątów na których były testowane
* Algorytm poprawnie generuje tyle krawędzi ile powinien co znaczy że dzieli wielokąt o n wierzchołkach na n-2 trójkąty co zgadza się ze znaną właściwością tego dotyczącą.
* Dodatkowo możliwość zapisu i odczytu figur pozwala na łatwe zmienianie danych do obliczeń i sprawdzanie kolejnych typów wielokątów