Kacper Garus, 25.11.2023, ćwiczenie nr.3 Triangulacja

Dane techniczne:

Język - python, translator - Visual Studio Code, procesor -Intel(R) Core (TM) i7-8565U CPU @ 1.80GHz, system operacyjny - Windows 10 Home 64bit

Realizacja ćwiczenia:

Wszystkie sprawdzane wielokąty były w formie listy dwuelementowych list.

Najpierw zaimplementowałem algorytm sprawdzający czy dany wielokąt jest y-monotoniczny, algorytm polegał na znalezieniu w wielokącie wierzchołka o maksymalnej współrzędnej y. Następnie idąc od tego wierzchołka indeksami w górę, czyli w dół po lewej gałęzi wielokąta, sprawdzałem czy kolejne wierzchołki maja mniejszą lub równą współrzędną y, jeśli współrzędna y kolejnego wierzchołka była większa od poprzedniego, to zaczynałem iść w górę po prawej gałęzi i analogicznie jak w lewej sprawdzałem czy wierzchołki tym razem maja coraz większą współrzędną y. Jeśli udało mi się tak przejść wszystkie punkty, to znaczyło, że dany wielokąt jest y-monotoniczny.

Następnie napisałem algorytm, który klasyfikuje wierzchołki wielokąta na:

- Początkowe- obaj sąsiedzi niżej, oraz kąt wewnętrzny $< 180°$
- Końcowe- obaj sąsiedzi wyżej, oraz kąt wewnętrzny $< 180°$
- Dzielące- obaj sąsiedzi niżej, oraz kąt wewnętrzny $> 180°$
- Łączące- obaj sąsiedzi wyżej, oraz kąt wewnętrzny $> 180°$
- Prawidłowe- jeden sąsiad wyżej, drugi niżej

Algorytm polegał na sprawdzaniu w pętli każdego wierzchołka wedle warunków opisanych powyżej, wartość kąta wewnętrznego sprawdzałem za pomocą własnej zaimplementowanej funkcji wyznaczania wartości wyznacznika macierzy: mat_det_2x2.

Ostatni zaimplementowany przeze mnie algorytm dokonywał triangulacji wielokąta y-monotonicznego, jako pierwszy krok algorytm sprawdzał, czy wielokąt jest y-monotoniczny następnie, wywołuję funkcję: sort, która najpierw przydziela, do której gałęzi należy dany wierzchołek(lewa lub prawa), a potem wykorzystując, informację w której gałęzi jest wierzchołek sortuję wierzchołki w złożoności liniowej, według współrzędnej y malejąco. Następnie tworzę stos i dodaję na niego dwa pierwsze wierzchołki. Dla każdego wierzchołka zaczynając od trzeciego sprawdzam czy jest na tej samej gałęzi co wierzchołek ze szczytu stosu, jeśli nie: tworzę krawędzie od tego punktu do wszystkich punktów ze stosu. Jeśli tak: sprawdzam po kolei trójkąty, które tworzą: dwa punkty z góry stosu i dany wierzchołek, jeśli trójkąt mieści się w wielokącie to dodaję przekątną od naszego wierzchołka do drugiego od góry elementu ze stosu i zdejmuje górny punkt ze stosu, jeśli trójkąt nie mieści się w stosie to wrzucam na stos nasz wierzchołek. Funkcja zwraca listę dwuelementowych list gdzie każda dwuelementowa lista to indeksy wierzchołków utworzonych przy triangulacji krawędzi.

Zaimplementowałem również wizualizacje etapów triangulacji wielokąta, pokazującą również punkty które są na stosie i aktualnie rozpatrywany punkt, które można zobaczyć w pliku z kodem.

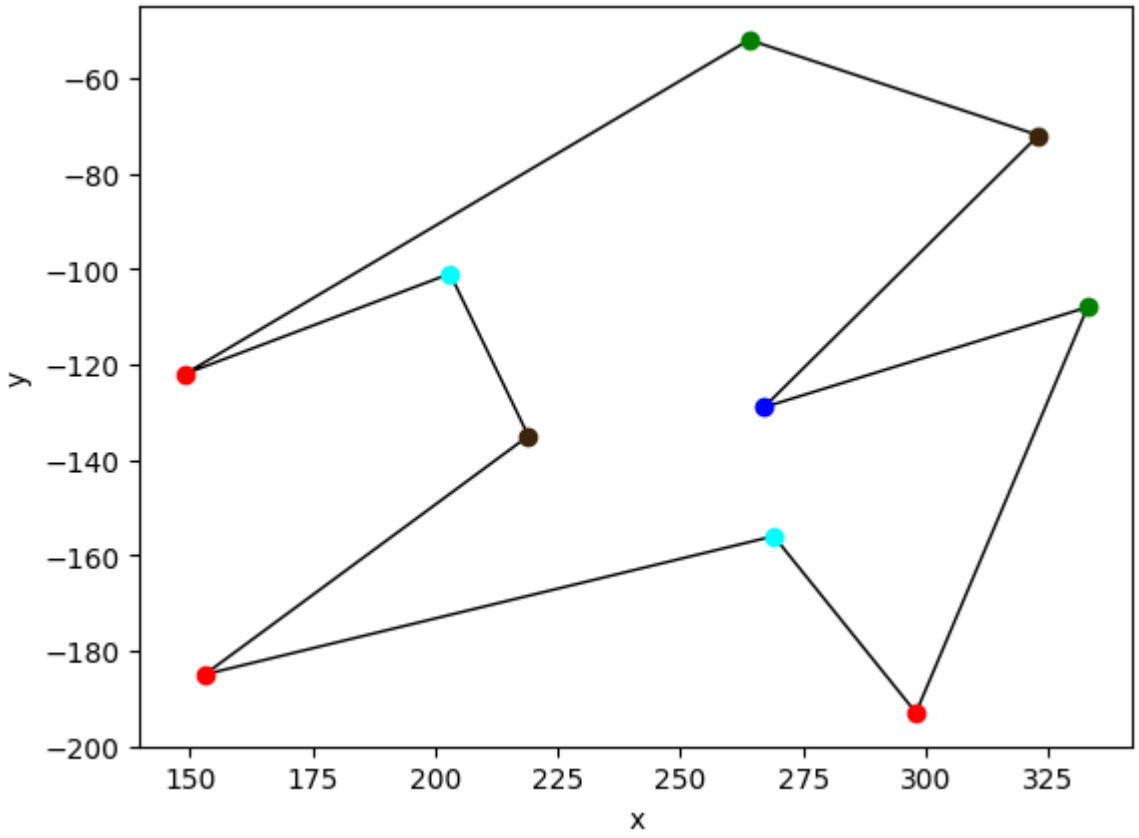
Przygotowany został również program napisany w pythonie, dzięki któremu za pomocą myszki możemy wprowadzać wielokąty. Program zwróci wprowadzone wielokąty w postaci listy punktów.

Wyniki i analiza:

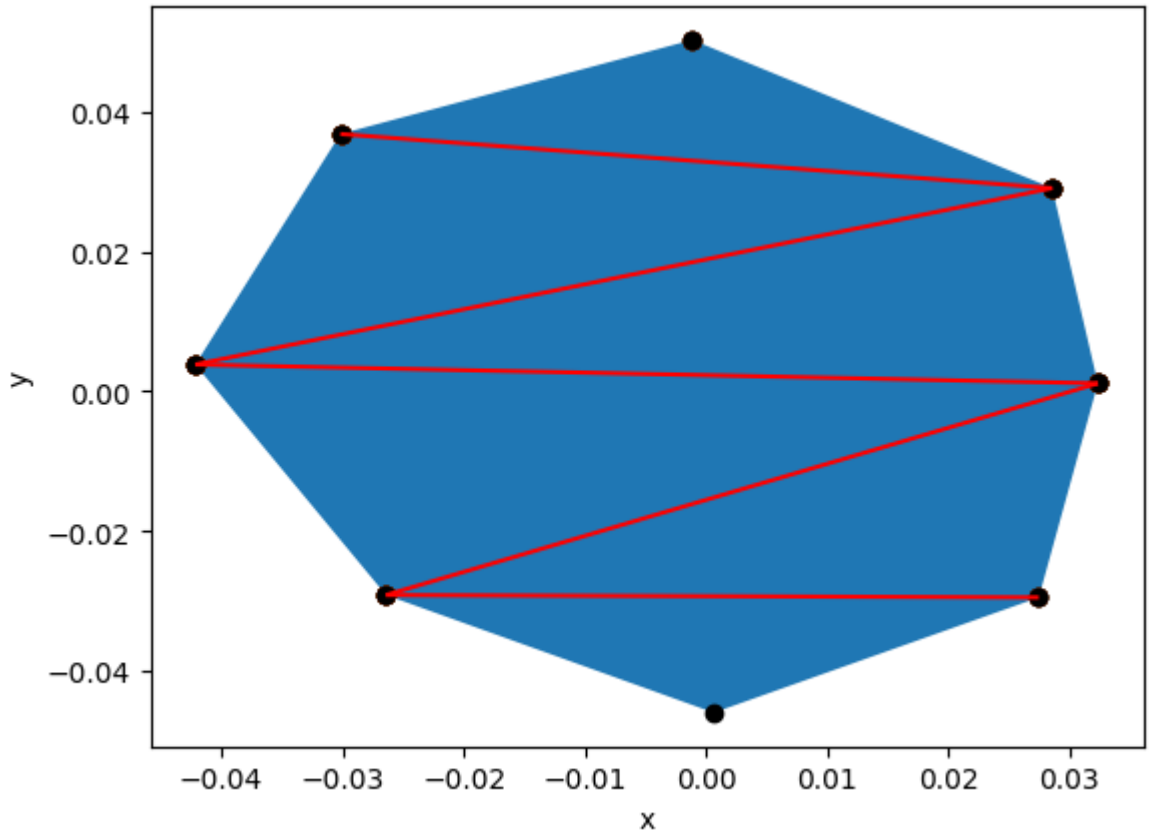
Klasyfikacja punktów: aby odróżnić jakiego typu jest dany punkt, w wizualizacji punkty zostały pokolorowane następująco:

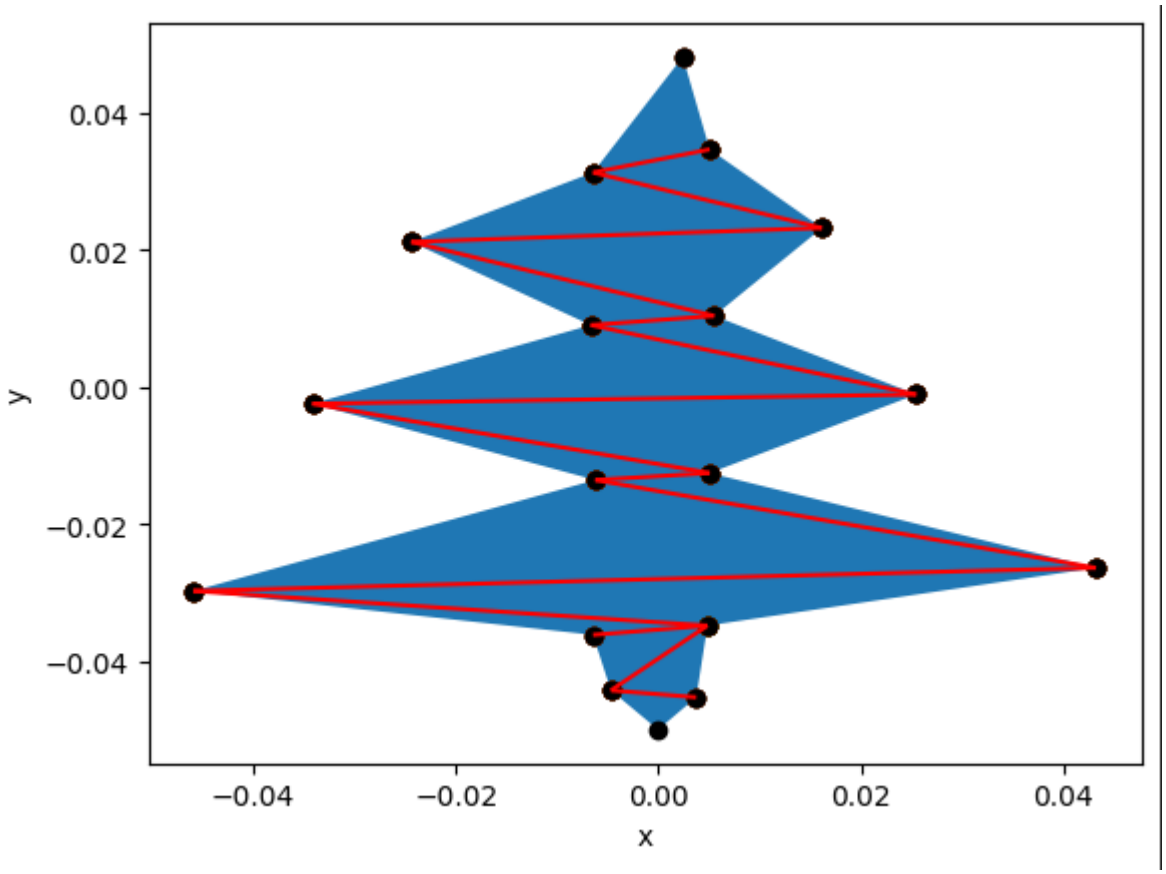
- Początkowe- zielony
- Końcowe- czerwony
- Dzielące- turkusowy
- Łączące- niebieski
- Prawidłowe- brązowy

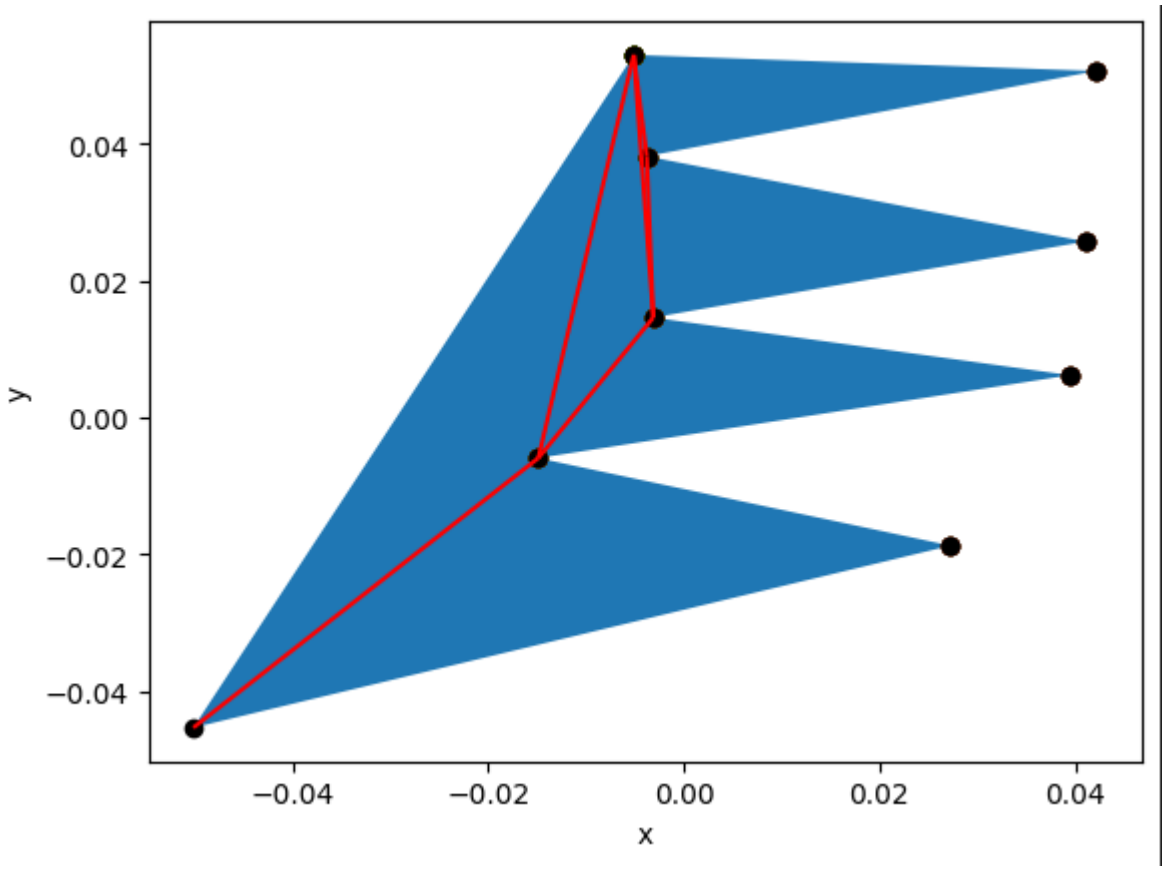
Przykładowy wielokąt posiadający wszystkie typy wierzchołków:

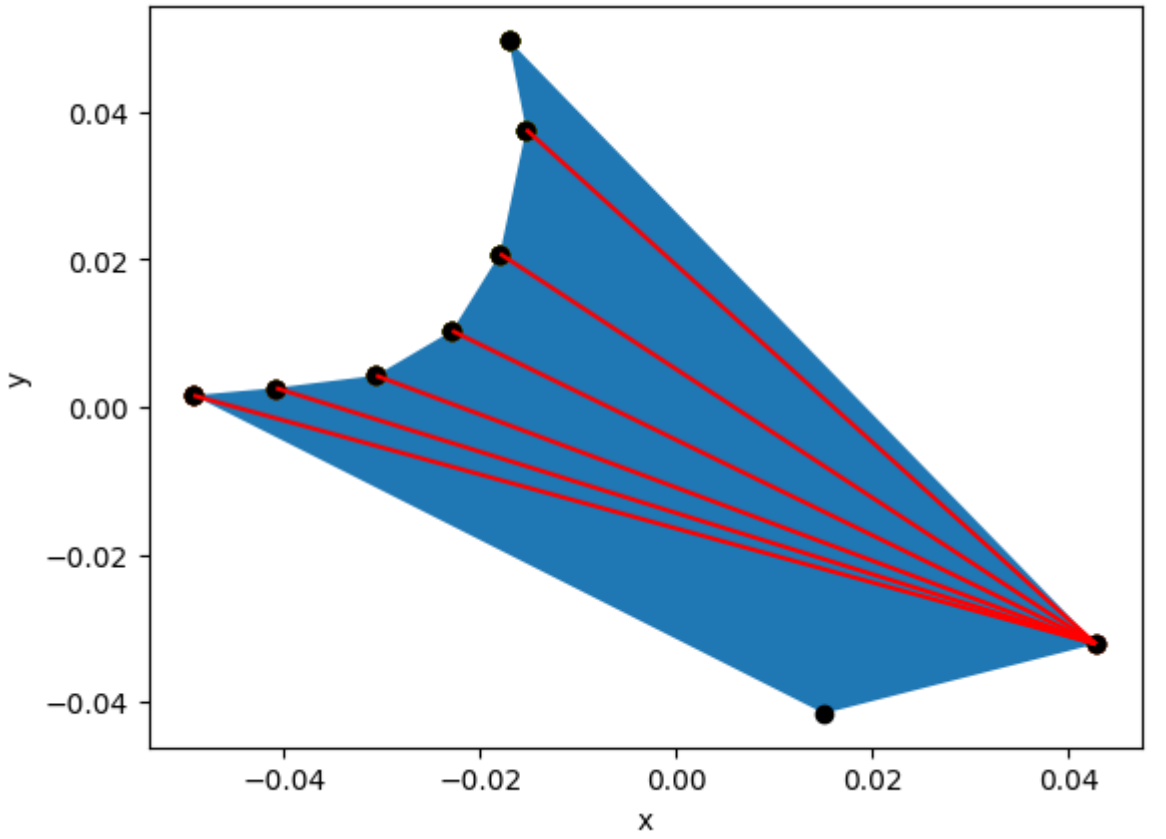


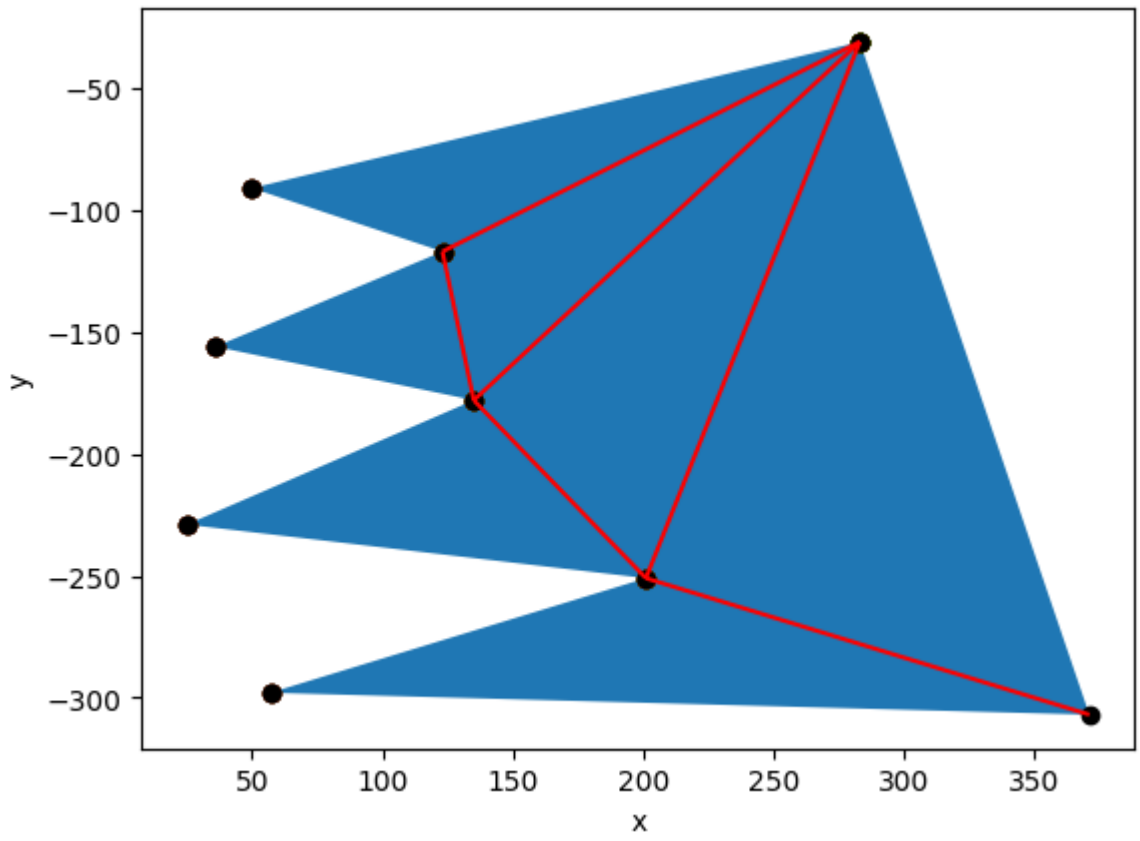
Przykładowe triangulacje:

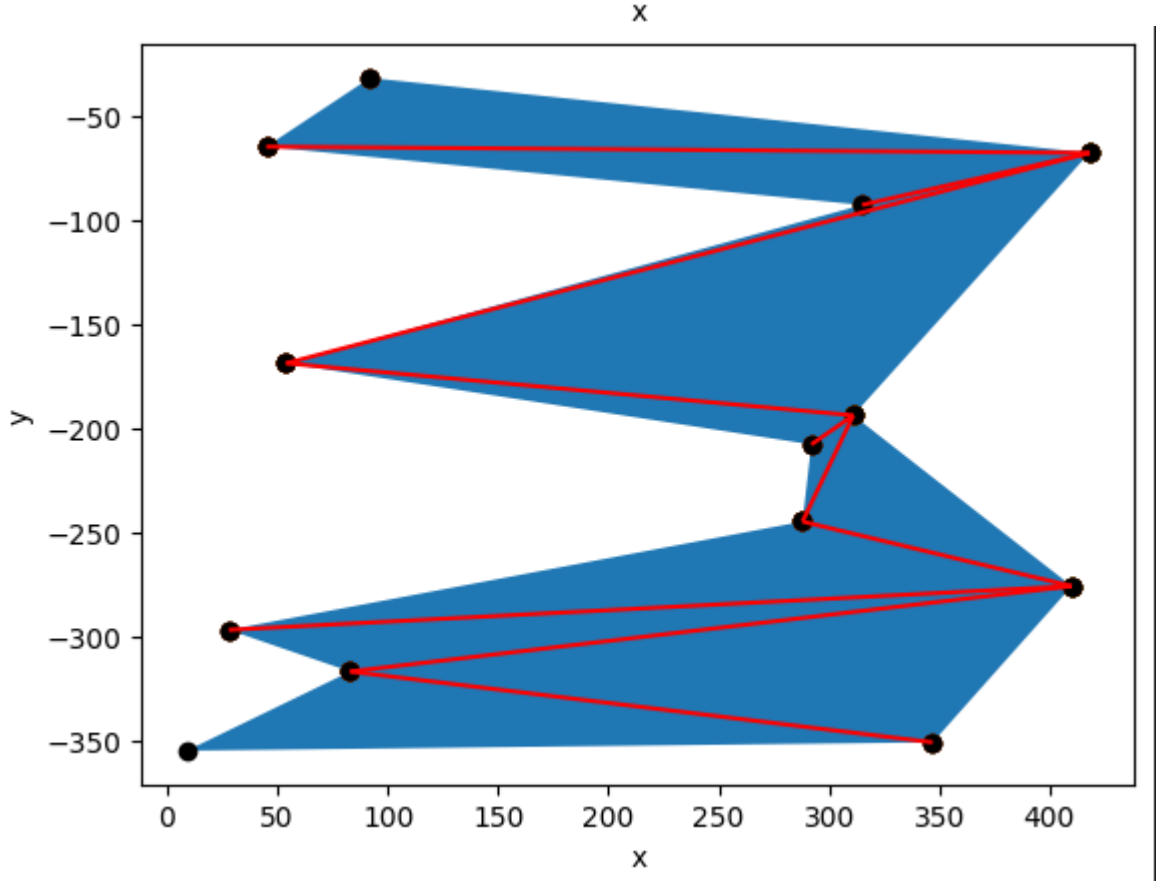





0.04
0.02
0.00
−0.02
−0.04
y
−0.04
−0.02
0.00
0.02
0.04
x


0.04
0.02
0.00
−0.02
−0.04
y
−0.04
−0.02
0.00
0.02
0.04
x


−50
−100
−150
−200
−250
−300
y
50
100
150
200
250
300
350
x


−50
−100
−150
−200
−250
−300
−350
y
0
50
100
150
200
250
300
350
400
x

Wnioski:

Wsztstkie algorytmy przeszły testy dane od KN BiT, a także moje własne. Wszystkie zwizualizowane triangulacje wygladają poprawnie, każdy wielokąt który pokazuję w sprawozdaniu, jest unikatowy i różniący się od innych, dzięki czemu algorytm został przetestowany na różnych zbiorach danych. Mogę zatem stwierdzić iż wszystko zostało zaimplementowane poprawnie.