Maciej Michoń, Informatyka II rok, gr. 2 Śr_13.00_A

Algorytmy geometryczne, sprawozdanie z ćwiczenia 3

1. Środowisko, biblioteki oraz dane techniczne urządzenia

Ćwiczenie zostało zrealizowane w Jupyter Notebooku i napisane w języku Python, z wykorzystaniem bibliotek numpy (umożliwiająca zaawansowane operacje numeryczne), pandas (do przedstawiania wyników poszczególnych obliczeń w DataFrame). Dodatkowo do rysowania wykresów (Visualizer) i sprawdzenia poprawności niektórych funkcji wykorzystałem projekt opracowany przez studentów Koła Naukowego BIT. Wszystko to było wykonane na laptopie z systemem operacyjnym Windows 11, procesorem AMD Ryzen 5 5600H 3.30 GHz oraz pamięcią RAM 32 GB.

2. Opis realizacji ćwiczenia:

Celem ćwiczenia była implementacja algorytmów: sprawdzenia y-monotoniczności zadanego wielokąta, klasyfikacja wierzchołków w dowolnym wielokącie, triangulacja wielokąta y – monotonicznego wraz z wizualizacją.

2.1. Generacja zbiorów danych (figur)

Program pozwala na tworzenie figur za pomocą własnoręcznie napisanej klasy PolygonBuilder. Za pomocą myszki należy wprowadzić kolejne punkty należące do figury. Po narysowaniu zostaje zwrócona lista punktów należących do tej figury w kolejności wprowadzania. W ten sposób łatwo można ponownie narysować figurę. Dodatkowo zostały stworzone funkcje save_() i read_(), dla których można w łatwy sposób potrzebne nam dane zapisywać i odczytywać w pliku json.

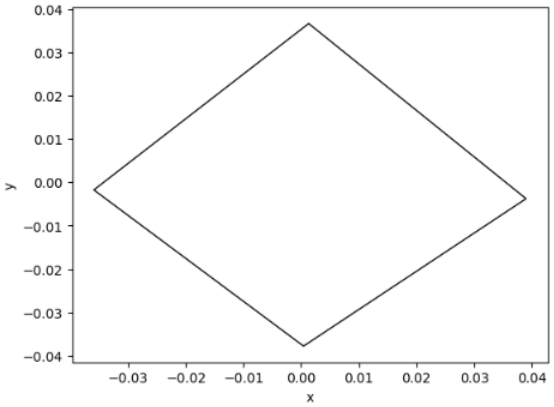
W celu przetestowania algorytmów wykorzystałem figury dostarczone przez testy z projektu oraz jedną figurę stworzoną za pomocą klasy PolygonBuilder:

- Czworokąt
- Okrąg
- Choinka
- Figura niemonotoniczna
- Wielokąt
- Grzebień (kilka wariantów)
- Strzałka
- Strzałka rozciągnięta
- Figura narysowana za pomocą PolygonBuildera

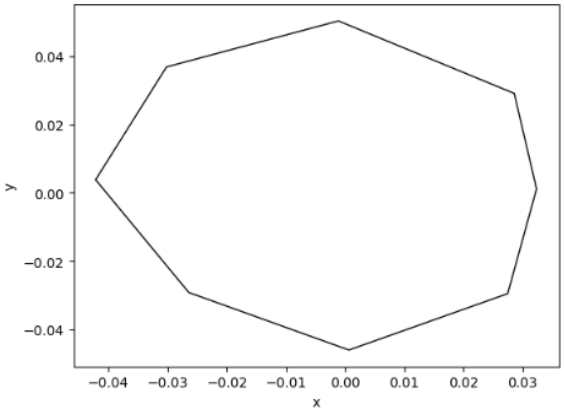
Figury ‘Czworokąt’, ‘Okrąg’ i ‘Wielokąt’ testują podstawowe przypadki, czyli figury wypukłe, dla których każde punkty można połączyć odcinkiem zawartym w tym wielokącie. Figura niemonotoniczna służy sprawdzeniu procedury określania y-monotoniczności. Pozostałe figury nie są wypukłe i testują niezależność działania algorytmu triangulacji od orientacji figury, położenia punktów ‘krytycznych’, czyli takich z których połączonych będzie wiele innych punktów, ‘odcinanie’ krawędzi z wierzchołków oraz zrównoważona wielkość krawędzi.

Wszystkie zaimplementowane w ramach tego ćwiczenia algorytmy korzystają z reprezentacji wielokątów w postaci listy wierzchołków w kolejności ich dodawania. Pozwala to na szybkie znajdowanie sąsiadów wierzchołków oraz lewej i prawej gałęzi figury.

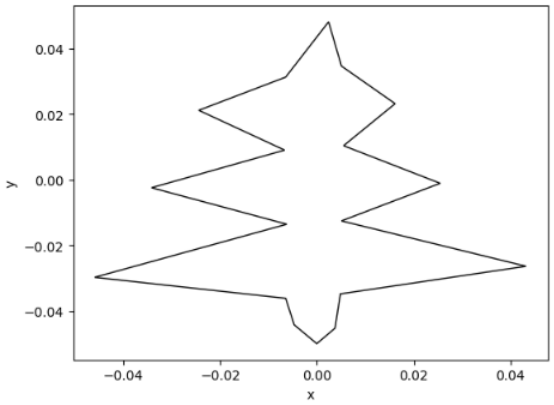
Rysunek 1.1 Figura ‘Czworokąt’



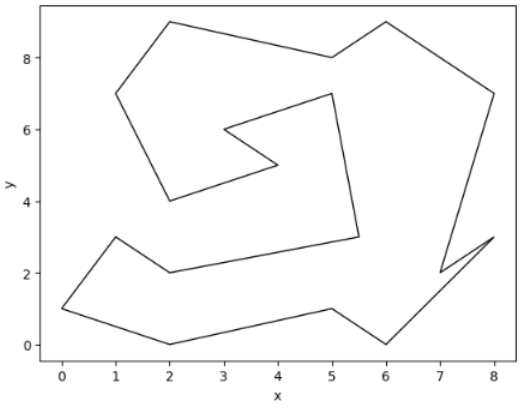
Rysunek 1.2 Figura ‘Okrąg’



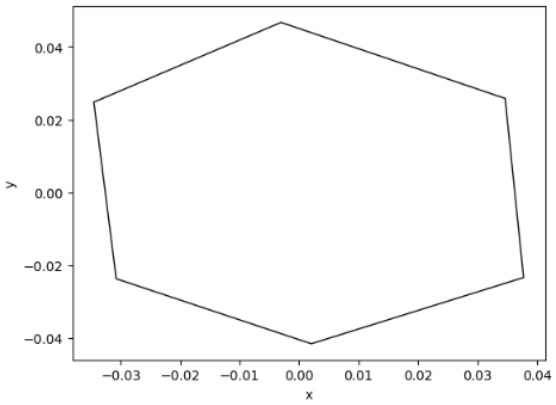
Rysunek 1.3 Figura ‘Choinka’



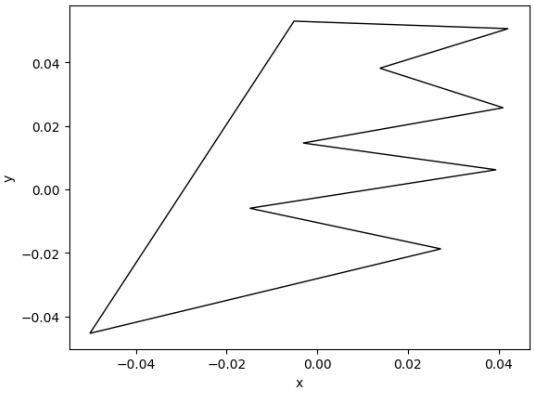
Rysunek 1.4 Figura ‘Figura niemonotoniczna’



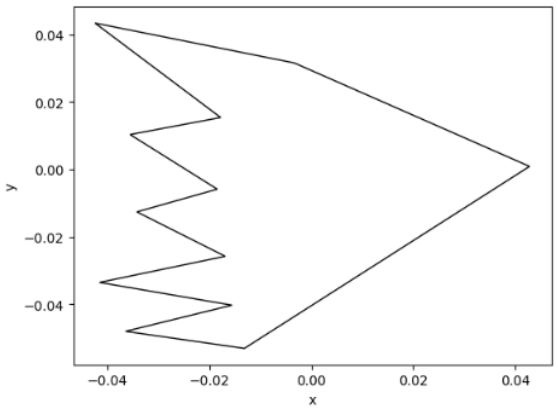
Rysunek 1.5 Figura ‘Wielokąt’



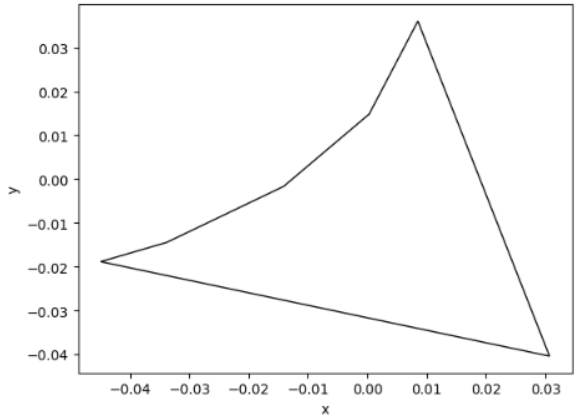
Rysunek 1.6 Figura ‘Grzebień 1’



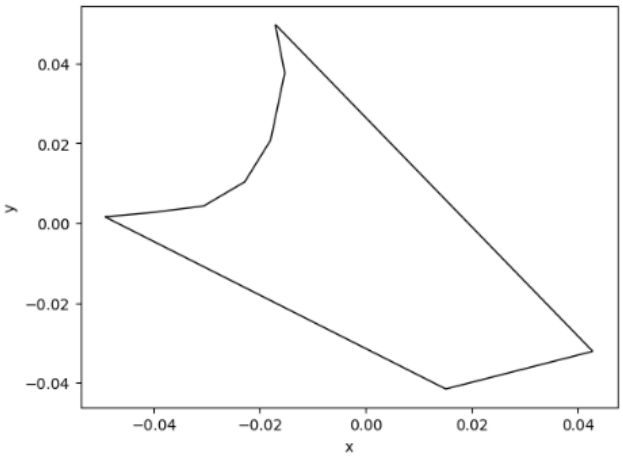
Rysunek 1.7 Figura ‘Grzebień 2’



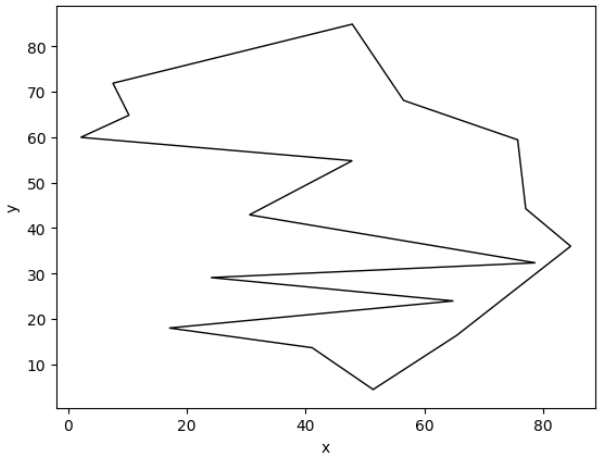
Rysunek 1.8 Figura ‘Strzałka’



Rysunek 1.9 Figura ‘Strzałka rozciągnięta’



Rysunek 1.10 Figura ‘Figura narysowana za pomocą PolygonBuildera’



2.2. Sprawdzanie monotoniczności

W celu sprawdzenia monotoniczności zadanego wielokąta wykorzystano kolejność punktów go tworzących. Wielokąt jest w postaci listy wierzchołków z których się składa i w nim znaleziono indeksy punktów o największej i najmniejszej współrzędnej y. Następnym krokiem było sprawdzenie, czy każdy kolejny punkt idąc od najwyższego do najniższego leży poniżej wcześniejszego punktu. Ponownie trzeba było sprawdzić warunek tylko tym razem od punktu najniższego do najwyższego, gdzie punkty leża powyżej wcześniejszych punktów. W przypadku przynajmniej jednego niespełnionego warunku można stwierdzić, że figura nie jest monotoniczna.

Procedura ta została przetestowana dla wszystkich zaproponowanych figur i jedynie zgodnie z oczekiwaniami figura ‘Figura niemonotoniczna’ została oznaczona jako niemonotoniczna.

2.3. Klasyfikacja wierzchołków

Algorytm klasyfikacji wierzchołków wielokąta na początkowe, końcowe, łączące, dzielące i prawidłowe polega na jednokrotnym przejściu po liście wierzchołków i ich klasyfikacji na podstawie ich położenia w relacji do sąsiednich wierzchołków.

W celu określenia kąta wewnętrznego jaki tworzą rozpatrywane wierzchołki użyto wyznacznika.

Punkty były klasyfikowane na podstawie następujących warunków z następującymi oznaczeniami:

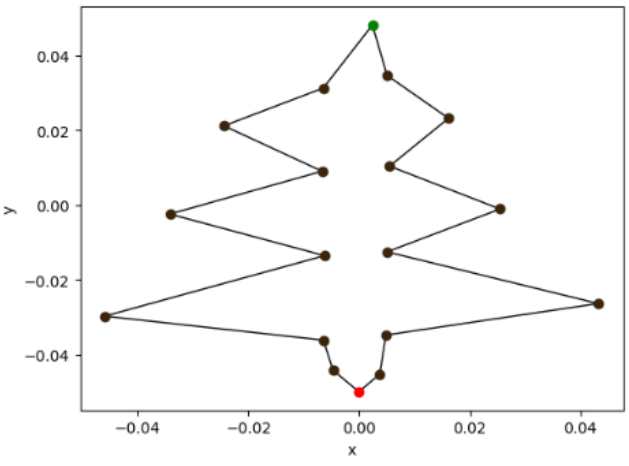
- Początkowe - gdy obaj jego sąsiedzi leżą poniżej i kąt wewnętrzny ma mniej niż 180 stopni. (zielony)
- Końcowe - gdy obaj jego sąsiedzi leżą powyżej i kąt wewnętrzny ma mniej niż 180 stopni. (czerwony)
- Dzielący - gdy obaj jego sąsiedzi leżą poniżej i kąt wewnętrzny ma więcej niż 180 stopni. (ciemnoniebieski)
- Łączący - gdy obaj jego sąsiedzi leżą powyżej i kąt wewnętrzny ma więcej niż 180 stopni. (jasnoniebieski)
- Prawidłowy - pozostałe przypadki, jeden sąsiad powyżej drugi poniżej

Dla wszystkich zestawów wierzchołki zostały sklasyfikowane poprawnie. Zgodnie z przewidywaniami wielokąty monotoniczne zawierały po jednym wierzchołku początkowym (najwyższy), końcowym (najniższym), a pozostałe wierzchołki były prawidłowe.

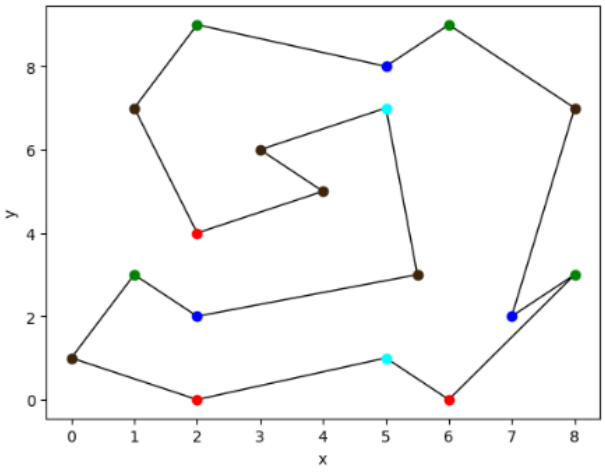
Figura 'Figura niemonotoniczna' zawierała wszystkie rodzaje wierzchołków.

Na rysunku spośród wszystkich figur monotonicznych przedstawiony jest tylko jeden przykład ponieważ w reszcie przypadków rezultat był podobny.

Rysunek 2.1 Klasyfikacja punktów w figurze 'Choinka'



Rysunek 2.2 Klasyfikacja punktów w figurze 'Figura niemonotoniczna'



2.4. Triangulacja

Algorytm triangulacji na wejściu przyjmuję listę punktów, będących wierzchołkami tworzącymi dany wielokąt w kolejności ich wprowadzenia, zgodnie z ruchem przeciwnym do wskazówek zegara.

Na początku algorytm dokonuje podziału punktów na dwie gałęzie – lewą i prawą. Jest to wykonane w ten sposób, że znajdujemy najpierw najwyższy i najniższy punkt. Lewa gałąź składa się z przedziału punktów od indeksu najwyższego do najniższego wraz z punktem najwyższym. Pozostałe punkty znajdują się w gałęzi prawej. Następnie punkty są sortowane od najwyższego do najniższego według współrzędnej y.

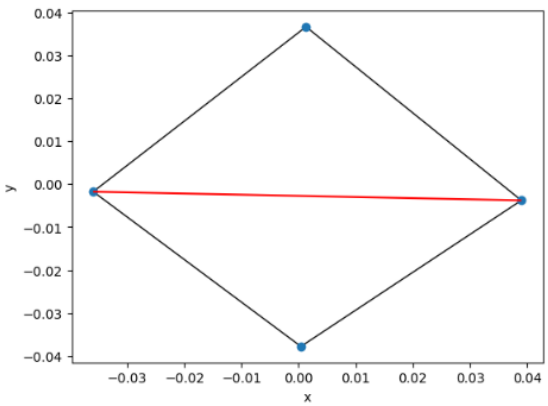
Przed rozpoczęciem pętli dodajemy do stosu dwa pierwsze najwyższe wierzchołki. Następnie w pętli rozpatrujemy pojedynczo kolejne wierzchołki. Jeśli rozpatrywany wierzchołek znajduje się w innej gałęzi niż ostatni punkt ze stosu to łączymy go ze wszystkimi wierzchołkami znajdującymi się na stosie i sprawdzamy żeby nie dodać przypadkiem krawędzi pomiędzy sąsiadami tego wierzchołka. Po wykonaniu tej operacji z powrotem dodajemy dwa ostatnio rozważane wierzchołki.

Jeśli rozważane punkty należą do tej samej gałęzi co wierzchołek ze szczytu stosu dodajemy krawędź między tymi dwoma wierzchołkami pod warunkiem, że krawędź je łącząca będzie zawierała się we wnętrzu wielokąta. Jest to sprawdzane poprzez wyliczenie wyznacznika i sprawdzenie go w zależności w jakiej gałęzi znajduję wierzchołek. Po wykonaniu tej operacji na stos umieszczamy wierzchołek rozpatrywany oraz ostatni wyjęty ze stosu wierzchołek.

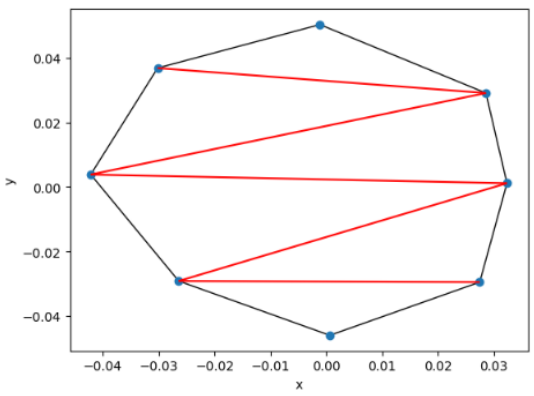
Po wykonaniu pętli głównej otrzymujemy listę przekątnych tego wielokąta. Dodatkowo tworzona jest lista triangulation do której dodajemy listy przekątnych oraz boki wielokąta w celu możliwości odczytu pełnego wyniku triangulacji. W zależności od wyboru funkcji triangulation2() zwraca przekątne w tablicy diagonals oraz tablica triangulation zawierająca przekątne i boki wielokąta. Funkcja triangulationvis dodatkowo zwraca wizualizacje przebiegu triangulacji.

Użycie listy krawędzi jako rezultat triangulacji ma wiele zalet:
Krawędzie są najprostszym typem obiektów, które definiują rezultat triangulacji. Zwracanie w postaci listy trójek punktów byłby trudniejszy w wizualizacji, ponieważ musielibyśmy uważać na duplikaty krawędzi;
Dzięki temu w prosty sposób możemy znaleźć ilość krawędzi wychodzących z danego wierzchołka.
Dodatkowo bardzo dobrze ta reprezentacja współgra z dostarczonym narzędziem do wizualizacji.

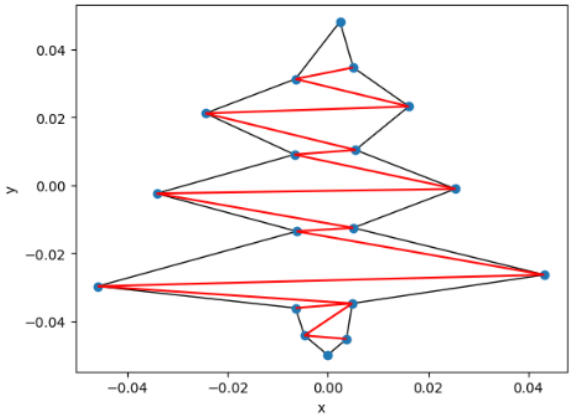
Rysunek 3.1 Wynik algorytmu triangulacji dla figury ‘Czworokąt’



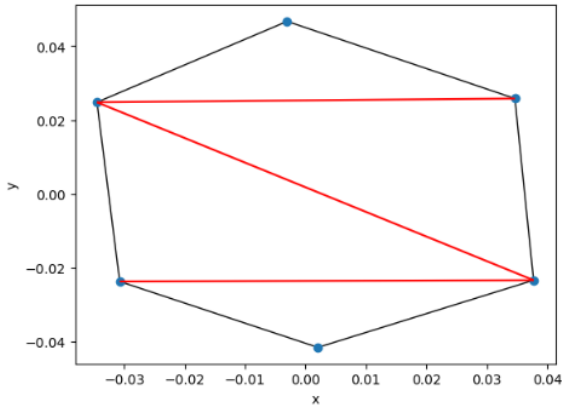
Rysunek 3.2 Wynik algorytmu triangulacji dla figury ‘Okrąg’



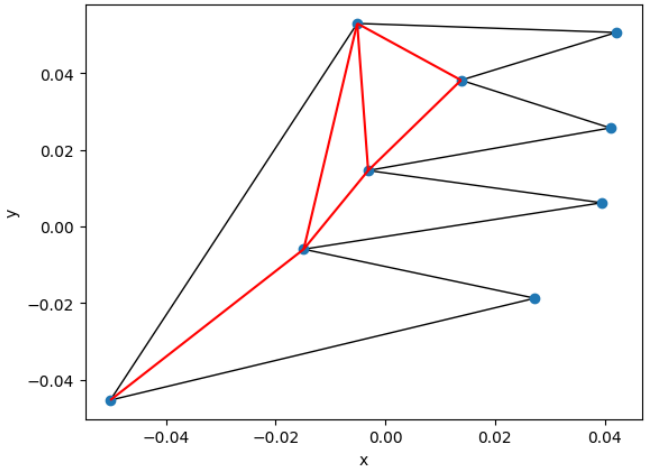
Rysunek 3.3 Wynik algorytmu triangulacji dla figury ‘Choinka’



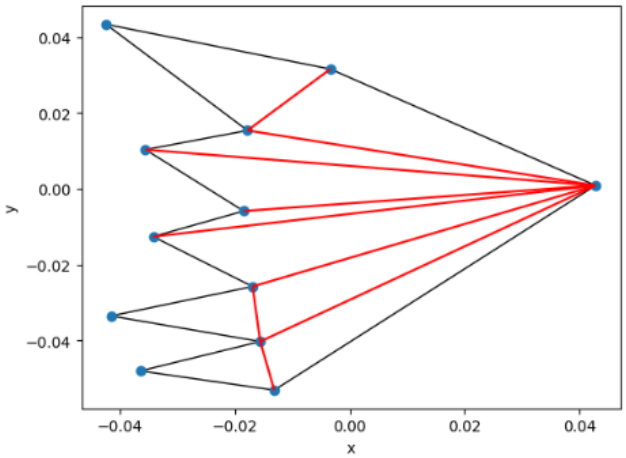
Rysunek 3.4 Wynik algorytmu triangulacji dla figury ‘Wielokąt’



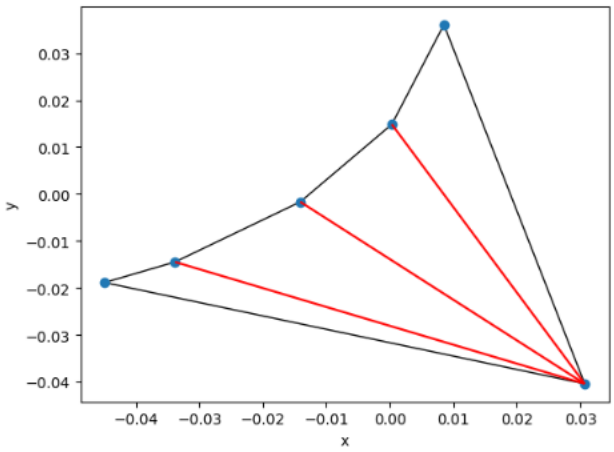
Rysunek 3.5 Wynik algorytmu triangulacji dla figury ‘Grzebień 1’



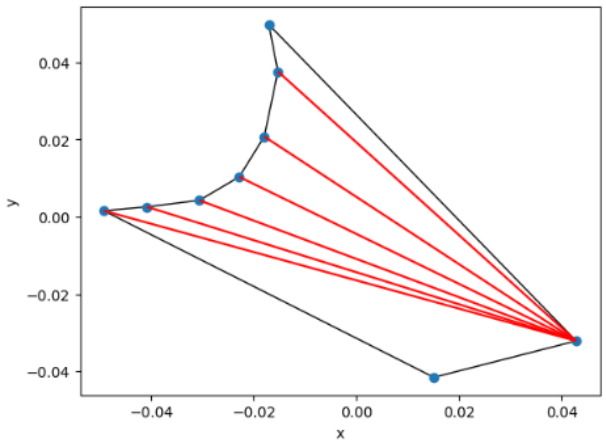
Rysunek 3.6 Wynik algorytmu triangulacji dla figury ‘Grzebień 2’



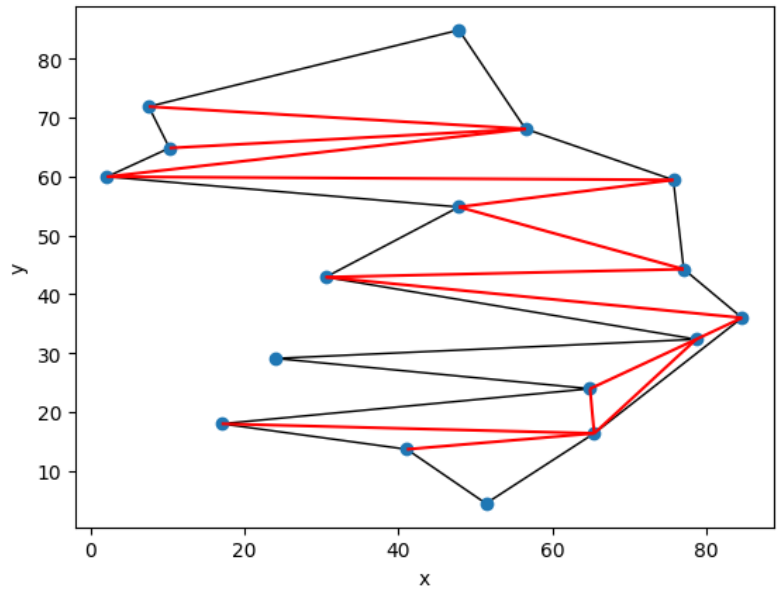
Rysunek 3.7 Wynik algorytmu triangulacji dla figury ‘Strzałka’



Rysunek 3.8 Wynik algorytmu triangulacji dla figury ‘Strzałka rozciągnięta’



Rysunek 3.9 Wynik algorytmu triangulacji dla figury 'Figura narysowana za pomocą PolygonBuildera'



3. Wnioski

Wszystkie zaimplementowane procedury i algorytmy zadziałały poprawnie dla testowanych wielokątów. Poprawność triangulacji można było łatwo zweryfikować korzystając z własności wielokątów prostych, mówiącej że dla wielokąta o n wierzchołkach efektem triangulacji będzie n-2 trójkątów. Ponadto algorytm poprawnie nie generuje duplikatów krawędzi. Dobrze to widać w przykładowych wizualizacjach znajdujących się w programie, w których dodawano tylko przekątne między punktami.