Algorytmy geometryczne
Sprawozdanie z Laboratorium 3

Adam Naumiec

naumiec@student.agh.edu.pl
410936
Grupa 4 – czwartek, 11:20-12:50, tydzień B

Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie
Wydział Informatyki, Elektroniki i Telekomunikacji

Listopad MMXXII

Spis treści

1. Dane techniczne komputera, na którym wykonywano obliczenia

Wykorzystano:

- komputer z 64-bitowym systemem $macOS\ 13\ Ventura$;
- czterordzeniowy procesor $Intel\ Core\ i5$;
- środowisko $Jupyter\ Notebook$, $JetBrains\ PyCharm$;
- język programowania $Python\ 3$ (3.11);
- wykorzystane biblioteki: $math$ (wartość π), $matplotlib$ (rysowanie wykresów), $time$ (sprawdzanie czasu wykonania obliczeń);
- do przygotowania sprawozdania wykorzystano programy $Microsoft\ Word$, $Microsoft\ Excel$;
- do rysowania wielokątów: dostarczone na laboratoriach narzędzie graficzne oparte na bibliotekach $matplotlib$ i $numpy$.

2. Cel ćwiczenia

W laboratorium przygotowano aplikację graficzną tak, aby można było zadawać proste wielokąty przy użyciu myszki z możliwością zapisu i odczytu przygotowanych konstrukcji.

W ćwiczeniu zaimplementowano:

- procedurę pozwalającą na sprawdzenie czy podany wielokąt jest y-monotoniczny;
- algorytm, który dla zadanego wielokąta wyszukuje i odpowiednio koloruje wierzchołki początkowe, końcowe, łączące, dzielące i prawidłowe;
- procedurę triangulacji wielokąta monotonicznego wraz z prezentacją kroków działania.

Na koniec przetestowano działanie programu na różnych zestawach danych.

3. Plan ćwiczenia

Plan ćwiczenia:

- Przygotowanie aplikacji graficznej pozwalającej zadawać proste wielokąty przy użyciu myszki, z możliwością zapisu i odczytu przygotowanych konstrukcji;
- Implementacja procedury sprawdzającej czy podany wielokąt jest y-monotoniczny;

- Implementacja algorytmu, który dla zadanego wielokąta wyszukuje wierzchołki początkowe, końcowe, łączące, dzielące i prawidłowe oraz odpowiednio je koloruje;
- Implementacja procedury triangulacji wielokąta monotonicznego, która prezentuje kolejne kroki wykonania algorytmu;
- Przetestowanie programów na różnych zestawach danych, prezentacja testów wydajnościowych.

4. Tolerancje dla zera

Klasyfikacji figur geometrycznych dokonywano z przyjętą tolerancją dla zera (oznaczaną jako *epsilon*). Wykorzystanie tolerancji związane jest z reprezentacją liczb rzeczywistych w komputerze i wynikającym z tego obarczeniem wyników obliczeń niedokładnością przy wykorzystaniu liczb zmiennoprzecinkowych (w języku *Python* typ zmiennej $float$ – 64-bitowa liczba zmiennoprzecinkowa).

Dokonano obliczeń dla tolerancji:

$$\varepsilon = 10^{-12}.$$

5. Wyznacznik

Do obliczania wyznacznika przygotowano własną implementację funkcji wykonującej stosowne obliczenia odpowiednimi sposobami, w laboratorium wykorzystano wyznacznik 3x3.

Punkty w zbiorach sklasyfikowano na te znajdujące się po lewej, po prawej oraz współliniowe (znajdujące się na jednej prostej) na podstawie wartości obliczonego wyznacznika.

Klasyfikacji dokonano na podstawie wartości wyznacznika dla punktów a, b i badanego punktu c. Wykorzystano następującą zależność:

Punkt c jest wobec punktów (prostej łączącej punkty) a i b:

$$\det(a,b,c)\begin{cases}(-\infty,0) \Rightarrow po\ prawej,\\ 0 \Rightarrow współliniowy,\\ (0,+\infty) \Rightarrow po\ lewej.\end{cases}$$

6. Generacja, wizualizacja, zapisywanie i wczytywanie wielokątów

6.1. Generowanie wielokątów

Wielokąty generowane były w przestrzeni $\mathbb{R}^2$ z metryką euklidesową (współrzędne punktów wyrażone były liczbami rzeczywistymi) za pomocą myszki w zmodyfikowanym programie. Punkty należało generować przeciwnie do ruchu wskazówek zegara ($CCW - counterclockwise$).

6.2. Wizualizacja wielokątów

Do wizualizacji wielokątów na płaszczyźnie kartezjańskiej wykorzystano dostarczone na laboratoriach narzędzie graficzne korzystające z bibliotek *matplotlib* oraz *numpy*. Pierwsza z nich pozwala na rysowanie wykresów, druga dostarcza wiele narzędzi przydatnych w algorytmice i obliczeniach komputerowych.

6.3. Zapisywanie wielokątów

Klasa *Plot* posiadała metodę $toJson()$, która zwracała łańcuch znaków zawierający listę scen w formacie $JSON$. Taki łańcuch można zapisać do pliku stosując standardowe sposoby dostępne w *Pythonie*. Do zapisu skorzystano z metody *dumps* z biblioteki *json*.

6.4. Wczytywanie wielokątów

Kolejne etapy budowy wielokąta zapisywane były jako sceny. Wczytanie listy scen z pliku dokonywało się poprzez podanie parametru *json* w wywołaniu klasy *Plot*, która przygotowywała każda scenę poprzez jej wczytanie z użyciem metody *load* z biblioteki *json*.

7. Definicje

7.1. Wielokąt (ściśle) monotoniczny

Wielokąt ściśle monotoniczny – wielokąt, dla którego można wskazać prostą L (tzw. kierunek monotoniczności) taką, że każda prosta prostopadła do niej przecina wielokąt w najwyżej dwóch punktach. Wtedy i tylko wtedy możliwe jest podzielenie tego wielokąta na dwa spójne łańcuchy takie, że każda prosta prostopadła do L przecina każdy z łańcuchów co najwyżej jeden raz (w jednym punkcie).

Wielokąt słabo monotoniczny – powyższa definicja wielokątów ściśle monotonicznych rozszerzająca tę definicję na wielokąty posiadające krawędzie prostopadłe do L.

Podczas wykonywania laboratorium uwagę skupiano na wielokątach ściśle monotonicznych, zatem wielokąty słabo monotoniczne były klasyfikowane jako wielokąty niemonotoniczne (niespełniające definicji wielokąta ściśle monotonicznego). Możliwa jest łatwa modyfikacja programu pozwalająca na klasyfikację wielokątów słabo monotonicznych jako wielokątów monotonicznych.

Ilekroć w niniejszym sprawozdaniu występuje określenie „wielokąt monotoniczny" należy przez to rozumieć wielokąt ściśle monotoniczny zgodnie z powyższą definicją.

Wielokąty wypukłe są monotoniczne w każdym kierunku. Dla wielokąta monotonicznego możliwe jest znalezienie wszystkich jego kierunków monotoniczności w czasie liniowym ze względu na liczbę wierzchołków $O(n)$. Wielokąty monotoniczne mają duże znaczenie w geometrii obliczeniowej, ponieważ w czasie liniowym ($O(n)$) można dokonać ich triangulacji oraz istnieje algorytm, który pozwala w czasie liniowym ($O(n)$) rozłożyć dowolny wielokąt na sumę wielokątów monotonicznych.[1]

7.2. Klasyfikacje punktów wierzchołka wielokąta

W algorytmie klasyfikacji wierzchołków dokonywano klasyfikacji i kolorowania wierzchołków.

Wierzchołki na podstawie klasyfikacji kolorowano na kolor:

- Zielony – wierzchołek początkowy,
- Czerwony – wierzchołek końcowy,
- Niebieski – wierzchołek łączący,
- Błękitny – wierzchołek dzielący.
- Czarny – wierzchołek prawidłowy.

Dokonano przeglądu i klasyfikowano wierzchołków podczas wykonywania algorytmu zgodnie z przyjętymi zasadami:

- **wierzchołek początkowy**:
  - obaj sąsiedzi znajdują się poniżej,
  - kąt wewnętrzny tworzony przez wierzchołek i jego sąsiadów jest mniejszy od π;
- **wierzchołek końcowy**:
  - obaj sąsiedzi znajdują się powyżej,
  - kąt wewnętrzny tworzony przez wierzchołek i jego sąsiadów jest mniejszy od π;

[1] Definicja zgodna z definicją z wykładu, na podstawie: https://pl.wikipedia.org/wiki/Wielok%C4%85t_monotoniczny

- **wierzchołek łączący**:
  - obaj sąsiedzi znajdują się powyżej,
  - kąt wewnętrzny tworzony przez wierzchołek i jego sąsiadów jest większy od π;
- **wierzchołek łączący**:
  - obaj sąsiedzi znajdują się poniżej,
  - kąt wewnętrzny tworzony przez wierzchołek i jego sąsiadów jest większy od π;
- **wierzchołek prawidłowy** – we wszystkich pozostałych przypadkach (jeżeli wierzchołek nie spełnia żadnej powyższej definicji).

7.3. y-monotoniczność

Wielokąt y-monotoniczny – wielokąt, który jest monotoniczny względem prostej pionowej L (równoległej do osi Oy), ponieważ każda prosta pozioma (prostopadła do prostej L i równoległa do osi Ox) przecina go co najwyżej w dwóch punktach.

7.4. Triangulacja wielokąta monotonicznego

Triangulacją wielokąta monotonicznego nazywamy taki podział tego wielokąta na elementy, że:

- każdy z nich ma niepuste wnętrze,
- wnętrza różnych elementów są rozłączne,
- przecięcie (iloczyn zbiorów) dwóch elementów może być punktem, krawędzią lub zbiorem pustym, gdy odpowiednio: mają wspólny bok, mają wspólny wierzchołek, nie leżą obok siebie i nie mają wspólnego wierzchołka.

7.5. Wierzchołek położony najwyżej, najniżej, najbardziej na prawo i najbardziej na lewo

Wierzchołek uznawano za położony:

- najwyżej wtedy i tylko wtedy, gdy posiadał największą wartość współrzędnej y,
- najniżej wtedy i tylko wtedy, gdy posiadał najmniejszą wartość współrzędnej y,
- najbardziej na prawo wtedy i tylko wtedy, gdy posiadał największą wartość współrzędnej x,
- najbardziej na lewo wtedy i tylko wtedy, gdy posiadał najmniejszą wartość współrzędnej x.

W przypadku, gdy kilka punktów spełniało dane kryterium o wyborze punktu decydowała druga współrzędna. Wybierano punkt o największej lub

najmniejszej wartości drugiej współrzędnej wśród punktów, które pierwotnie spełniały kryterium z definicji.

8. Algorytmy

8.1. Badane algorytmy oraz ich złożoności

W laboratorium pochylono się nad zagadnieniem następujących algorytmów:

1. Algorytm sprawdzania y-monotoniczności wielokąta,
2. Algorytm klasyfikacji i kolorowania wierzchołków wielokąta,
3. Algorytm triangulacji wielokąta.

8.2. Algorytm sprawdzania y-monotoniczności wielokąta

8.2.1. Opis algorytmu

Algorytm sprawdzał, czy zadany wielokąt był y-monotoniczny. Każda prosta prostopadła do osi Oy posiada z takim wielokątem co najwyżej 2 punkty wspólne.

8.2.2. Działanie algorytmu

Algorytm przechodził po wszystkich wierzchołkach po kolei w kolejności przeciwnej do ruchu wskazówek zegara i sprawdzał czy, każda kolejna trójka kolejnych punktów (aktualnie sprawdzany, lewy i prawy sąsiad) nie tworzy wierzchołka łączącego lub dzielącego. Jeżeli taka trójka istniała oznaczało to, że zadany trójkąt nie był y-monotoniczny, w przeciwnym przypadku oznaczało to, że wielokąt był y-monotoniczny.

8.3. Algorytm klasyfikacji i kolorowania wierzchołków wielokąta

8.3.1. Opis algorytmu

Algorytm pozwalał na odpowiednie sklasyfikowanie i pokolorowanie wierzchołków wielokąta według odpowiednich reguł. Wierzchołki dzielono na wierzchołki: początkowe, końcowe, łączące, dzielące i prawidłowe według kryteriów klasyfikacyjnych.

8.3.2. Działanie algorytmu

Algorytm przechodził po kolejnych wierzchołkach i na podstawie wysokości (wartości współrzędnej y) i kąta jaki wierzchołek tworzył (wyznaczonego poprzez obliczenie wartości wyznacznika) z sąsiednimi wierzchołkami klasyfikował go odpowiednio i kolorował.

Wierzchołki o współrzędnej y większej są klasyfikowane były jako położone wyżej, a o współrzędnej y mniejszej jako wierzchołki położone niżej. Każdy wierzchołek posiada klasyfikację, a co za tym idzie także kolorowanie. Dwa wierzchołki obok siebie mogą być skalsyfikowane w tej samej kategorii i posiadać ten sam kolor.

Kryteria kolorowania i użyte kolory opisane były w sekcji *Definicje*.

8.4. Algorytm triangulacji wielokąta

8.4.1. Opis algorytmu

Sprawdzenie czy trójkąt należał do wielokąta odbywało się poprzez obliczenie wartości wyznacznika (3x3) dla punktów należących do trójkąta i sprawdzenia znaku (funkcja *signum*) wyznacznika. Łańcuchy miały przeciwne znaki. TBD

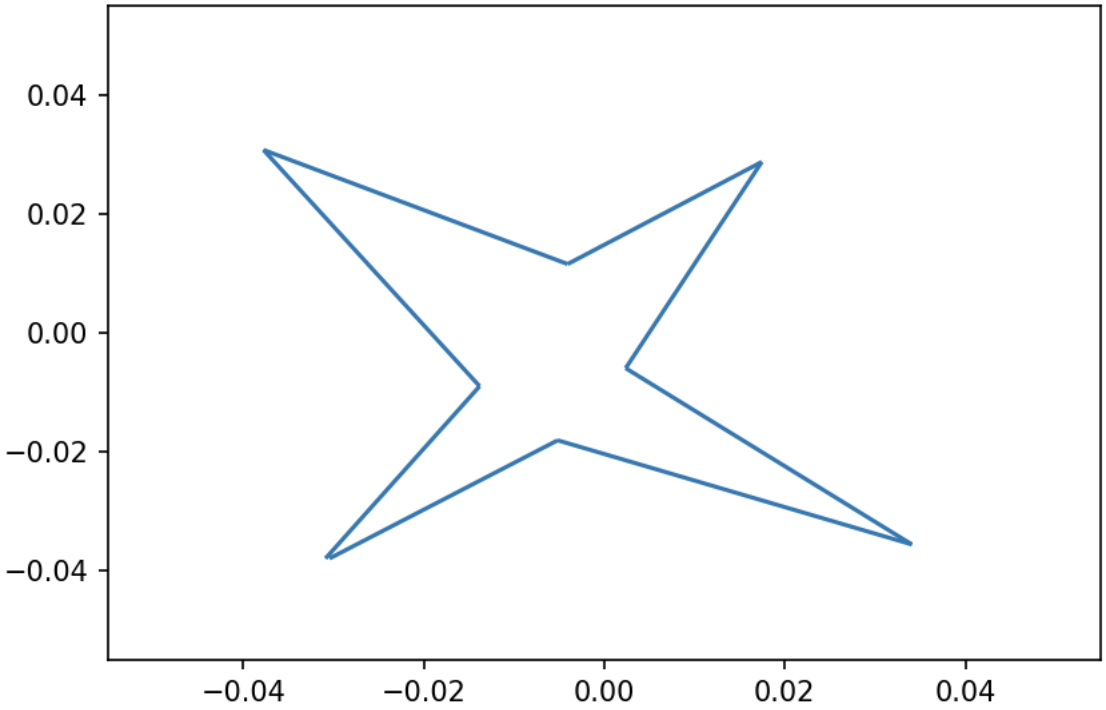
8.4.2. Działanie algorytmu

Lista kroków algorytmu:

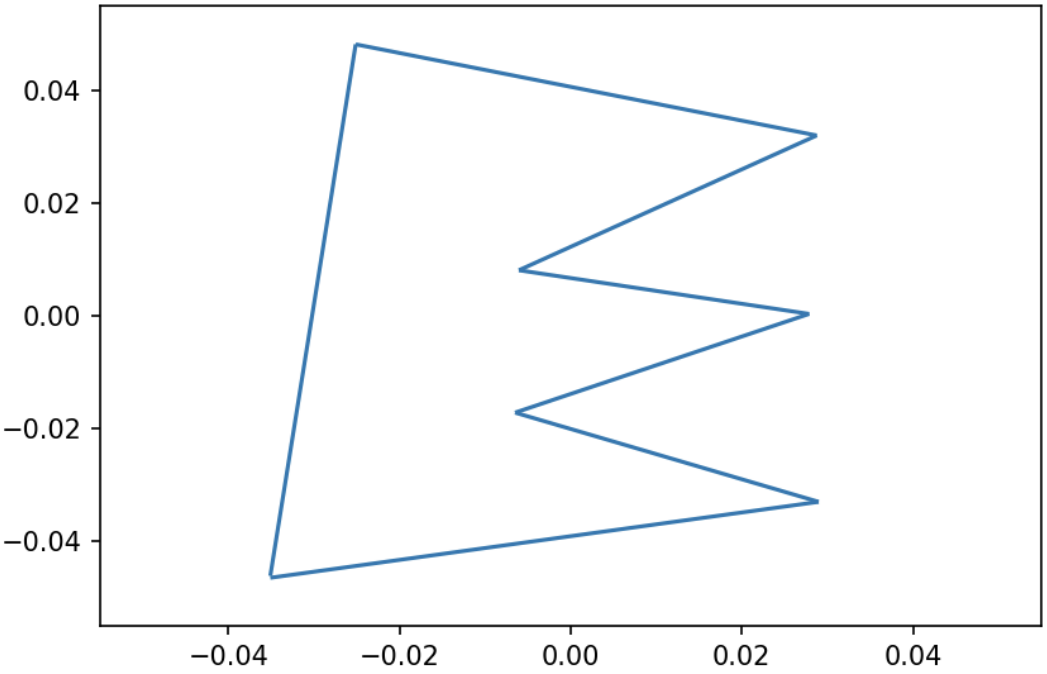
1. Sprawdzenie, czy zadany wielokąt jest y-monotoniczny. W przypadku wielokąta, który nie był y-monotoniczny algorytm kończył działanie.
2. Znalezienie wierzchołka o największej współrzędnej y w wielokącie h i wierzchołka o najmniejszej współrzędnej y w wielokącie v.
3. Przypisanie wierzchołkom innym niż h i v
4. Włożenie wierzchołka h i pierwszego wierzchołka z posortowanej listy na stos.
5. Przeglądanie reszty wierzchołków z posortowanej listy:
    a. Jeżeli kolejny wierzchołek należy do innego łańcucha niż wierzchołek stanowiący szczyt stosu, to należy go połączyć ze wszystkimi wierzchołkami ze stosu, z którymi nie tworzy on boku wielokąta. Na stosie pozostaje wierzchołek ze szczytu stosu i obecnie przeglądany punkt.
    b. W przeciwnym przypadku należy analizować trójkąty utworzone przez przeglądany wierzchołek z wierzchołkami ze stosu i tak długo jak trójkąty nalezą do wielokąta, należy połączyć ich wierzchołki (tam, gdzie połączenie już nie istnieje). Na stosie zostają wierzchołki, dla których nie udało się utworzyć trójkąta.
6. Połączenie punktów ze stosu z wierzchołkiem v (o ile wierzchołki nie są już połączone).

9. Analiza wyników

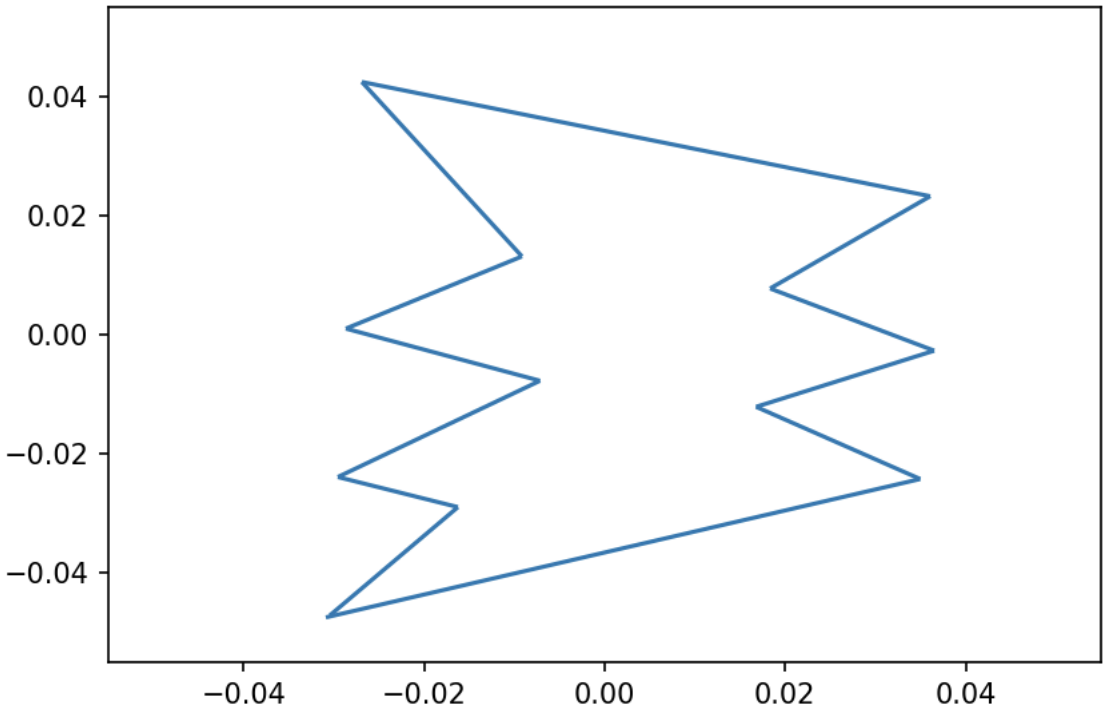
9.1. Algorytm sprawdzania y-monotoniczności wielokąta



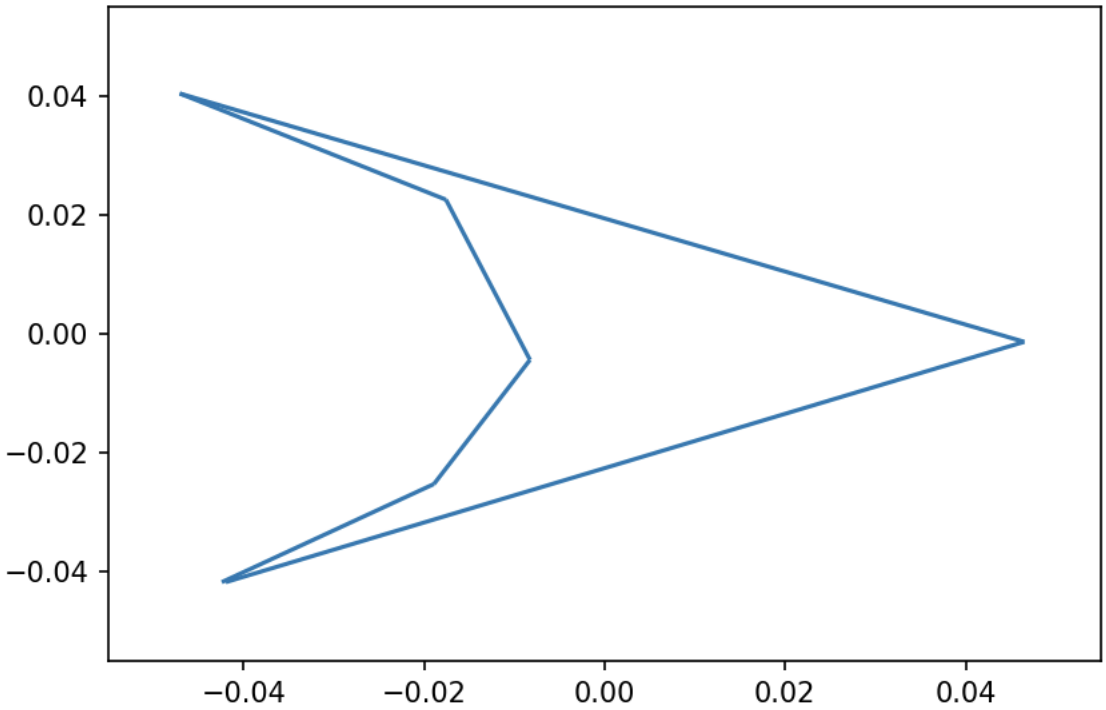
Rysunek 9.1. a) Przykład figury poprawnie zidentyfikowanej jako wielokąt nie będący wielokątem y-monotonicznym



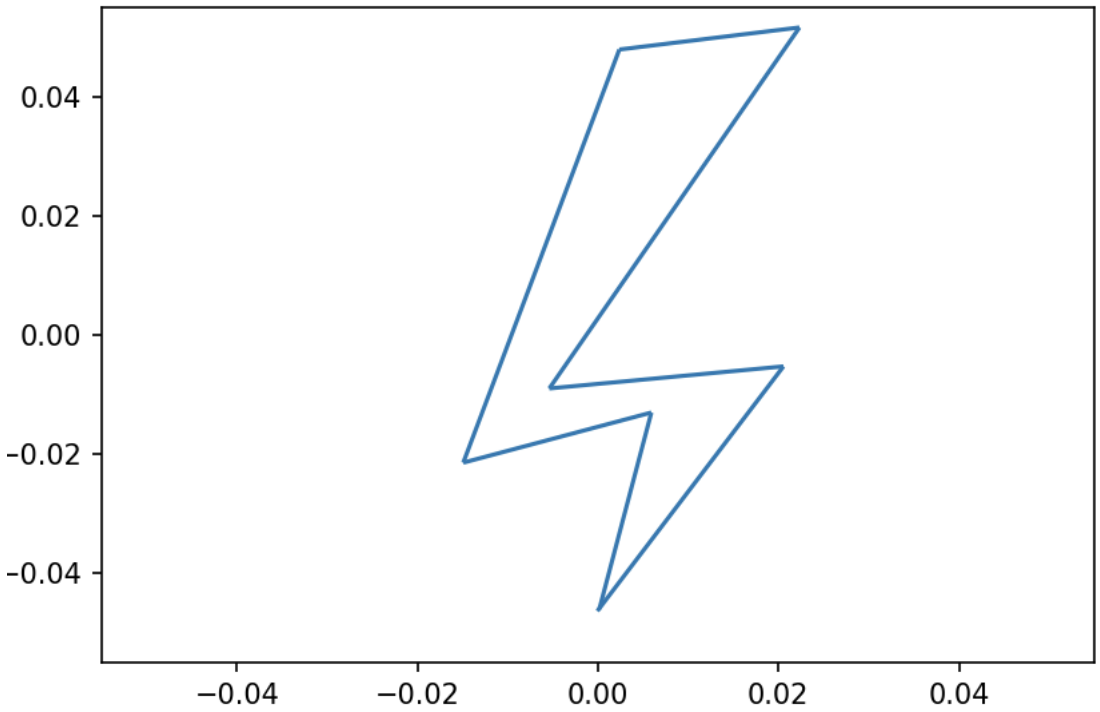
Rysunek 9.1. b) Przykład figury poprawnie zidentyfikowanej jako y-monotoniczny wielokąt



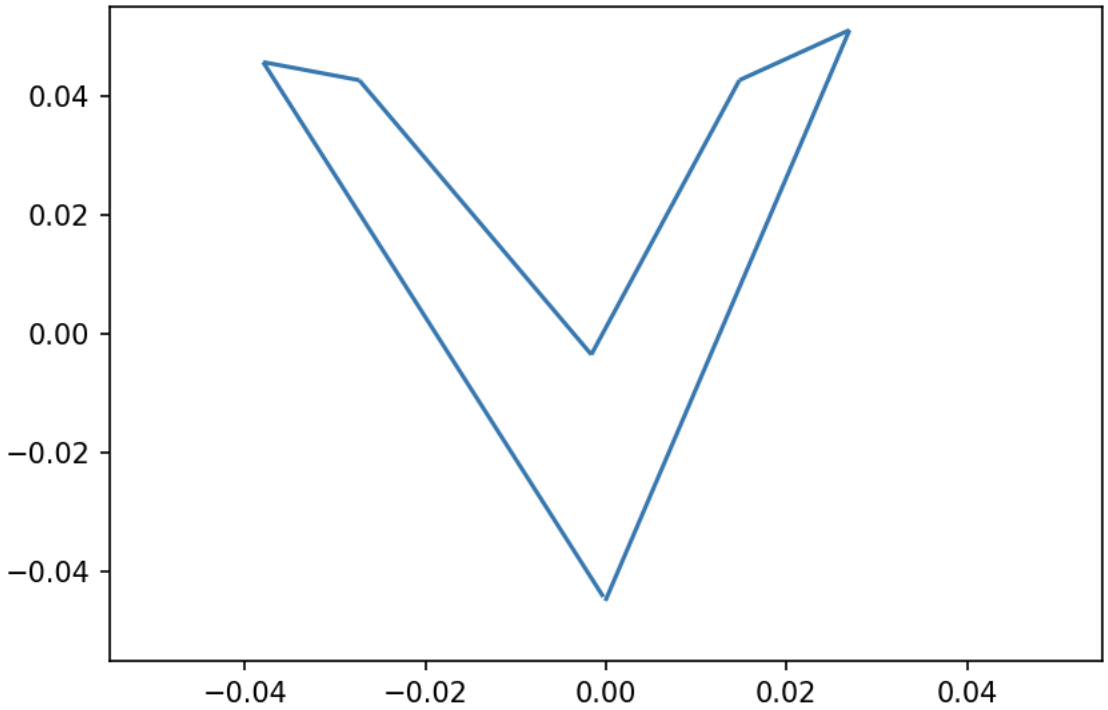
Rysunek 9.1. c) Przykład figury poprawnie zidentyfikowanej jako y-monotoniczny wielokąt



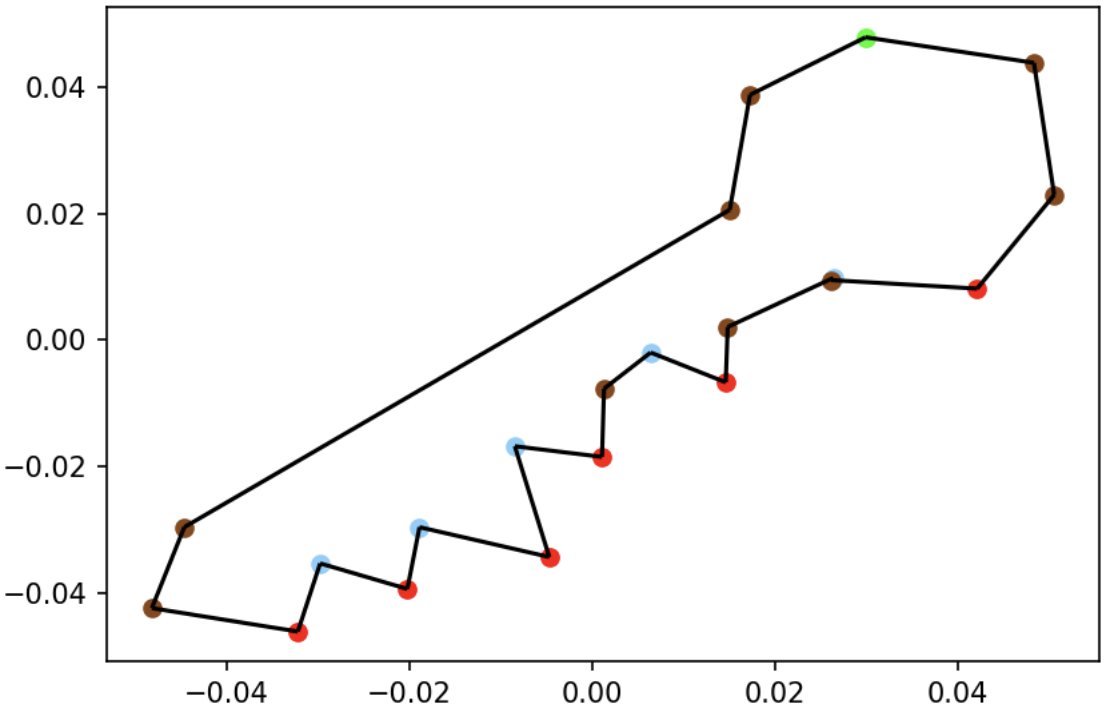
Rysunek 9.1. d) Przykład figury poprawnie zidentyfikowanej jako y-monotoniczny wielokąt



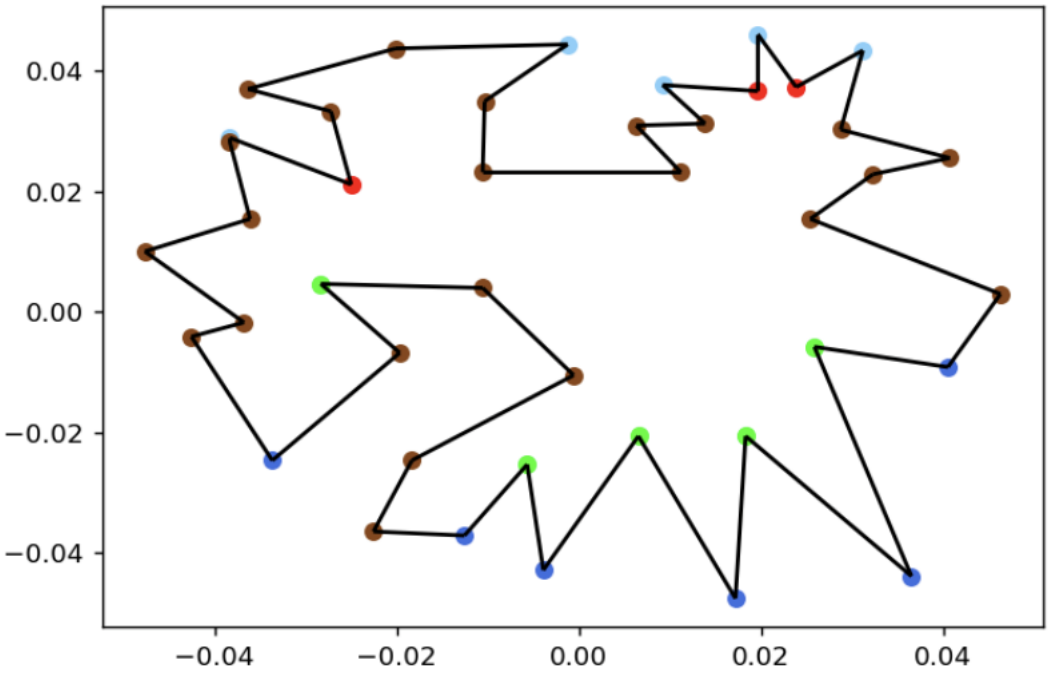
Rysunek 9.1. e) Przykład figury poprawnie zidentyfikowanej jako wielokąt nie będący wielokątem y-monotonicznym



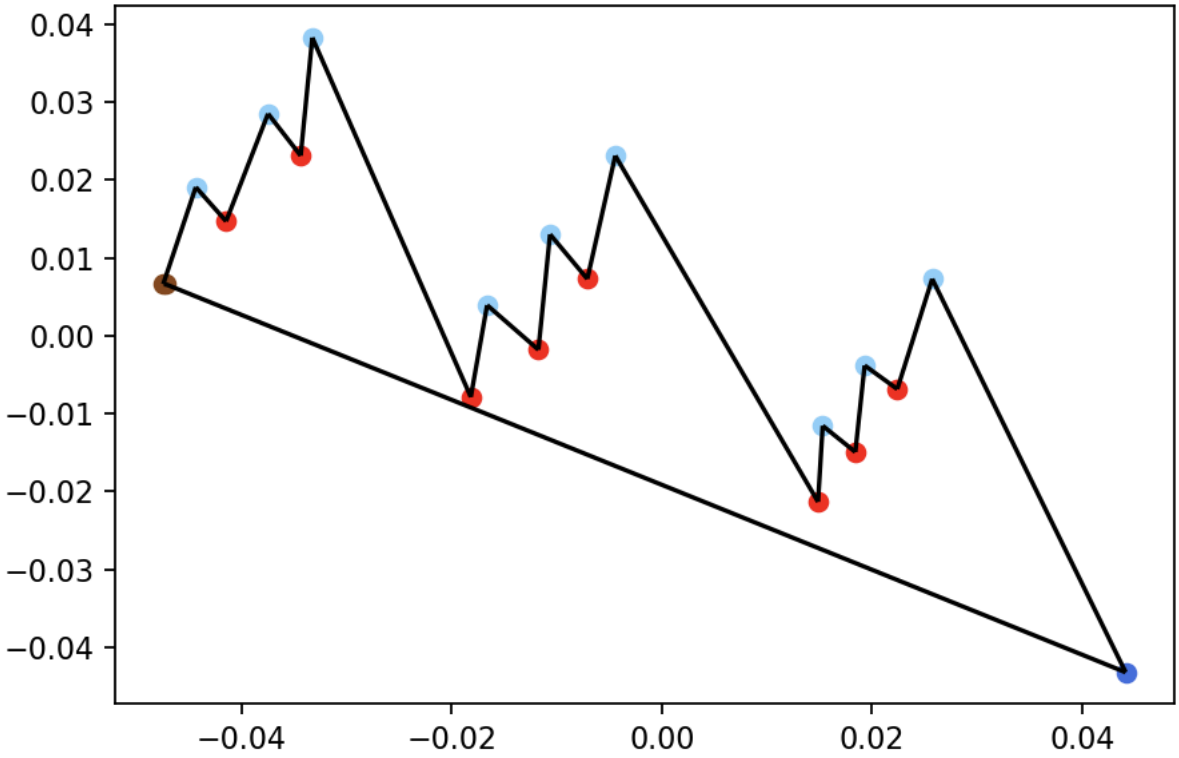
Rysunek 9.1. f) Przykład figury poprawnie zidentyfikowanej jako wielokąt nie będący wielokątem y-monotonicznym



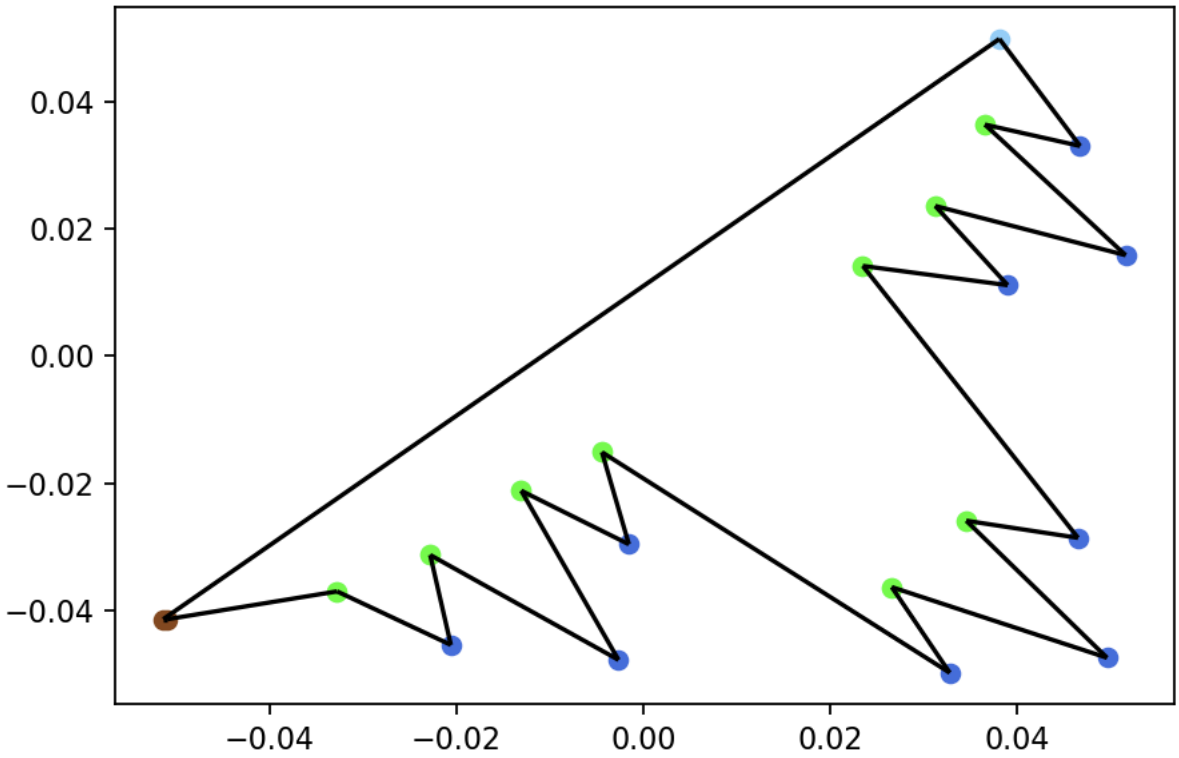
Rysunek 9.2. a) Prezentacja przykładowej klasyfikacji i kolorowania wierzchołków wielokąta



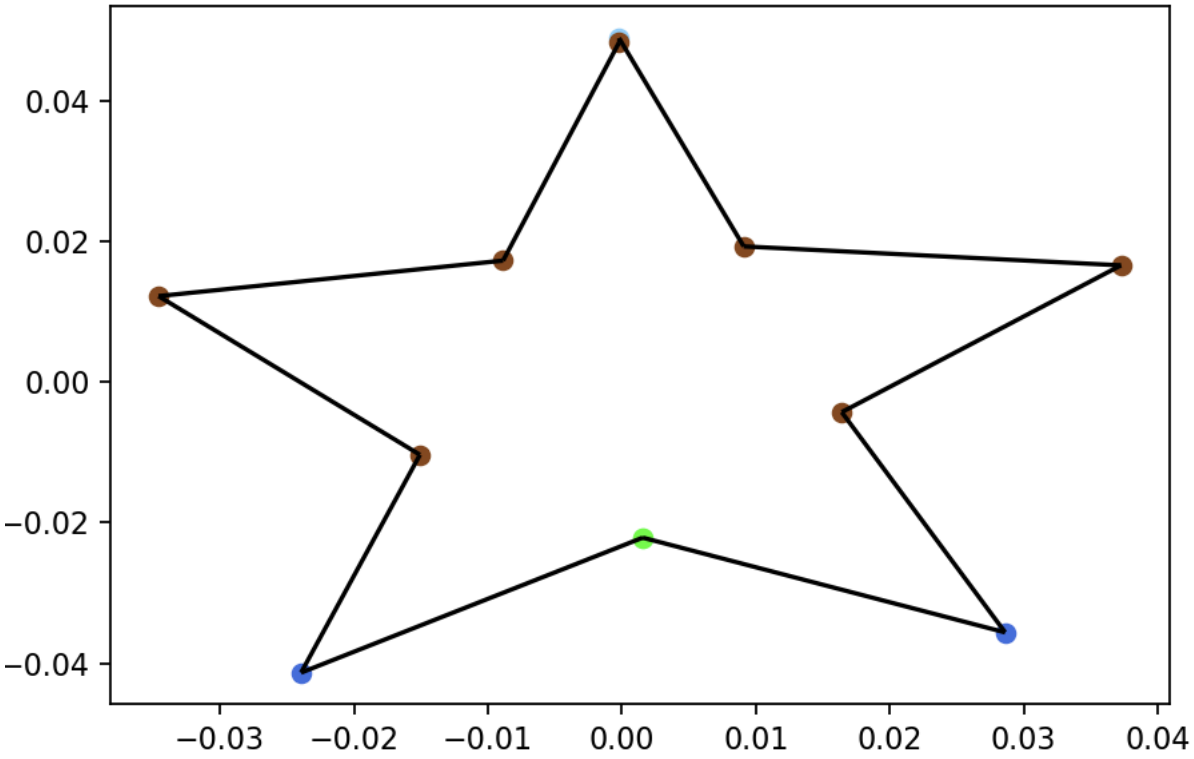
Rysunek 9.2. b) Prezentacja przykładowej klasyfikacji i kolorowania wierzchołków wielokąta



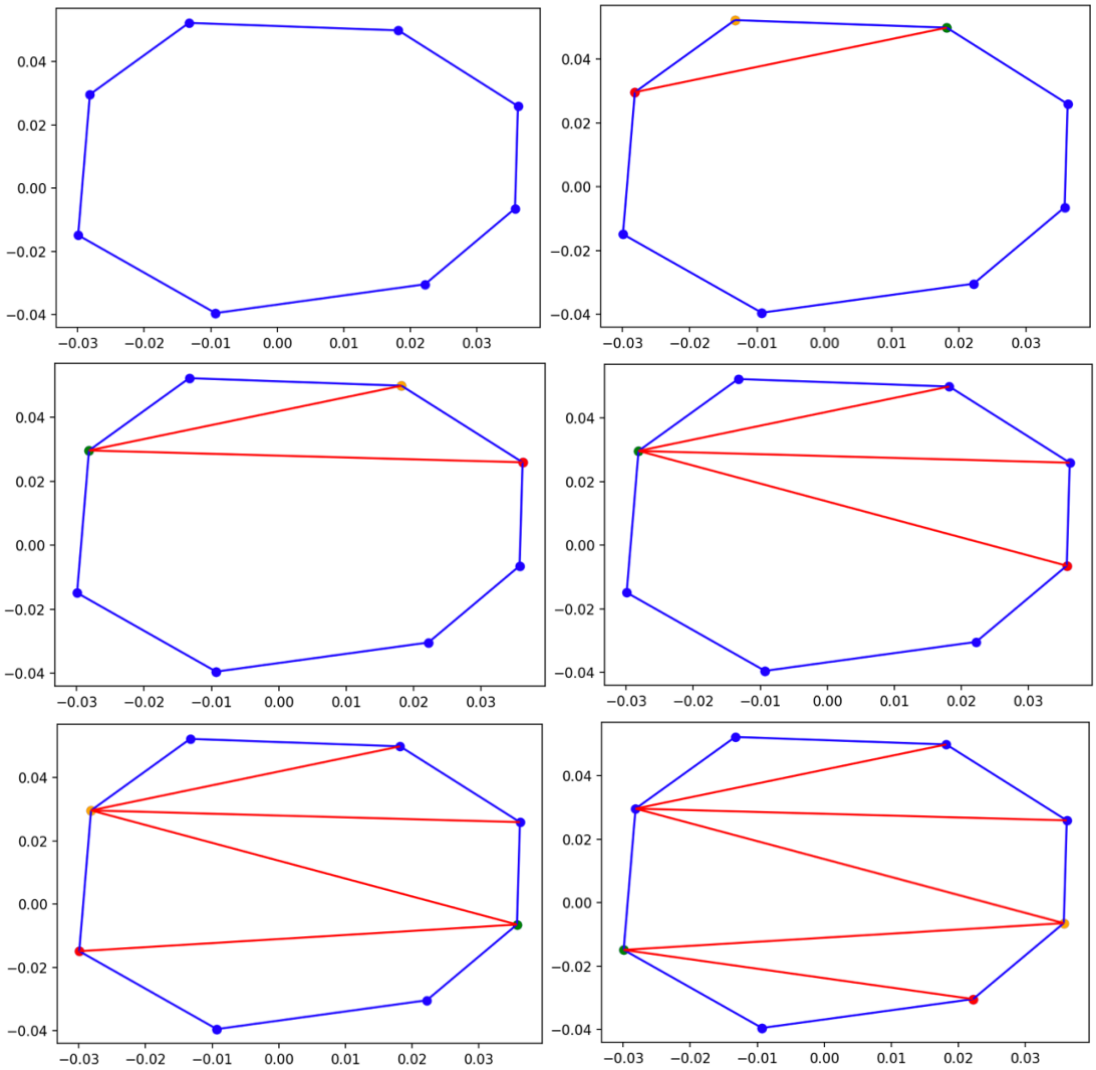
Rysunek 9.2. c) Prezentacja przykładowej klasyfikacji i kolorowania wierzchołków wielokąta



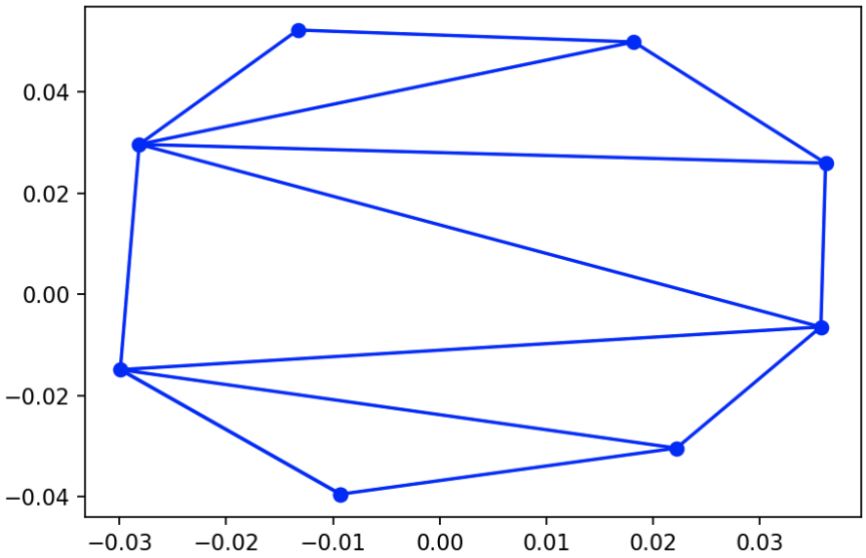
Rysunek 9.2. d) Prezentacja przykładowej klasyfikacji i kolorowania wierzchołków wielokąta



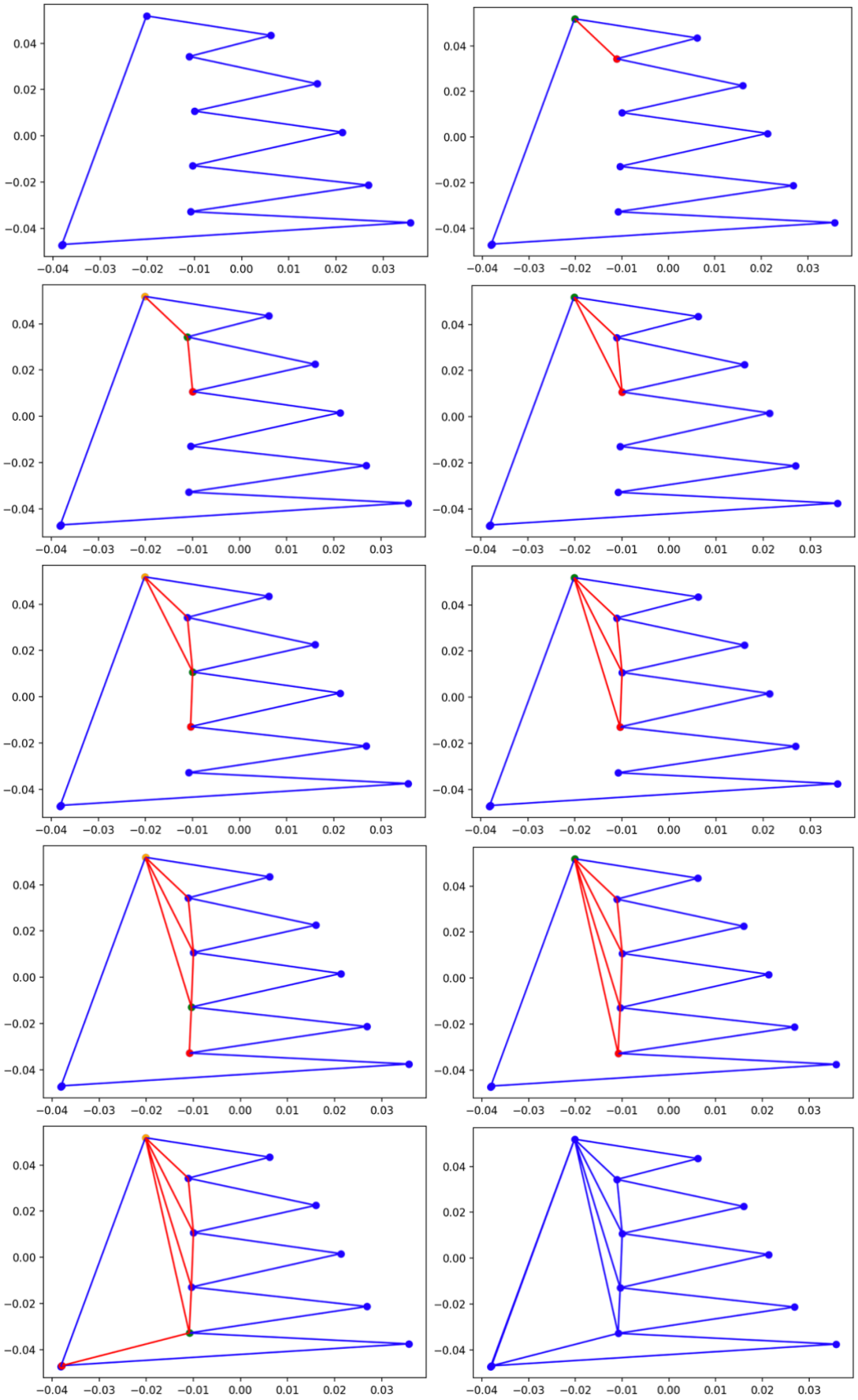
Rysunek 9.2. e) Prezentacja przykładowej klasyfikacji i kolorowania wierzchołków wielokąta



Rysunek 9.3. a1) Etapy triangulacji pierwszego wielokąta



Rysunek 9.3. a2) Efekt triangulacji pierwszego wielokąta



Rysunek 9.3. b) Wizualizacja etapów i rezultatu triangulacji drugiego wielokąta

10. Wnioski i podsumowanie

Programy zostały przetestowane na różnych zbiorach danych, w szczególności uwagę skupiono na przypadkach brzegowych, w których sprawdzana była poprawność działania dodatkowych instrukcji sprawdzających dane. Dobrym tego przykładem jest niesprawdzanie y-monotoniczności wielokąta, jeśli posiadał on mniej niż 3 boki. Poprzez przeanalizowanie otrzymanych rezultatów można stwierdzić, że programy działają poprawnie w każdym przypadku. Testowano także wykrywanie przez algorytm wielokątów niebędących wielokątami y-monotonicznymi by mieć pewność właściwego działania. Strukturami przechowującymi wielokąty były kolekcje punktów i linii przechowywane w listach. Pozwalało to na łatwą modyfikację danych, dodawanie i usuwanie wierzchołków i linii. Umożliwiało to także łatwo generować rysunki oraz dawało dostęp do wszystkich elementów wielokątów w każdym kroku. Pozwoliło to w pełni wykorzystać funkcjonalność list w języku *Python*. Do zapisu i odczytu skorzystano z biblioteki *json*, która pozwalała na zapis i odczyt zadanych wielokątów w popularnym formacie JSON, który jest wszechstronny i wygodny w użyciu/ Umożliwia to łatwą pracę z programem i wykorzystanie przetworzonych danych w innych projektach.

Wnioski te są szczególnie cenne dla projektów informatycznych, w których precyzja obliczeń i identyfikacja pewnych danych z wykorzystaniem algorytmów geometrycznych są sprawą kluczową.

* * *