**Algorytmy Geometryczne**

**Laboratorium 3 – Sprawozdanie**

**Jan Smółka**

1. **Sposób wykonania zadania**
   1. **Kod programu**

**Do rozwiązania zadania posłużono się językiem programowania Julia, wraz z wbudowaną biblioteką do tworzenia wykresów i zewnętrzną biblioteką PyCall oraz językiem Python, z wykorzystaniem biblioteki MatPlotLib.  
Kod został podzielony na pakiety zawierające generator wielokątów monotonicznych, procedury klasyfikacji punktów, triangulacji oraz predykaty geometryczne – test monotoniczności oraz funkcję *orient*.  
W implementacjach przyjęto sprzętową tolerancję dla 0, implikowaną zastosowanym typem reprezentacji liczb rzeczywistych Float64. Istnieje także możliwość przyjęcia (w ramach pakietu *PolygonTriangulation i Geometric Predicates*) arbitralnej tolerancji.  
Program ma możliwość zadania punktów poprzez wczytanie z pliku lub przez interaktywny interfejs, w którym położenia punktów zadaje się poprzez kliknięcie myszą.**

* 1. **Metoda badania**

**Każdy z zadanych zbiorów punktów zbadano przy użyciu algorytmu zachłannego triangulacji. Nie zaobserwowano rozbieżności w poprawności działania. Przyjęcie tolerancji równej 0 nie prowadzi do błędów. Pomiary czasu działania wykonano za pomocą wbudowanej funkcji.**

* 1. **Wykresy**

**Sporzą**dzono kilka wykresów, dla zbiorów punktów otrzymanych różnymi metodami. Rysunki 1 i 2 Ilustrują działanie algorytmu dla wielokątów wygenerowanych losowo.  
Rysunek 3 przedstawia triangulację prostego zbioru zadanego poprzez interaktywny wykres.  
Rysunek 4 przedstawia graficzny podział wielokąta na łańcuchy użyte do triangulacji.  
Rysunek 5 przedstawia klasyfikację punktów wskazaną w poleceniu.

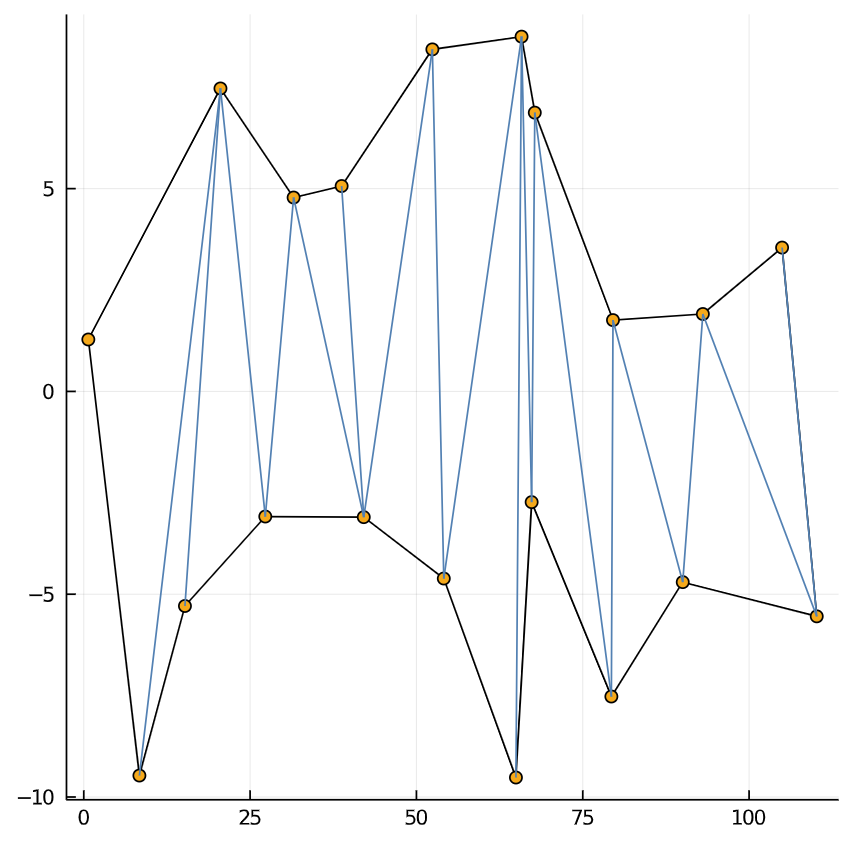
* 1. **Środowisko programistyczne**

**Do wykonania zadania użyto języka Julia w wersji 1.4, kompilowanego dedykowanym przez twórców kompilatorem oraz języka Python w wersji 1.10, działającym w wewnętrznym, automatycznie wygenerowanym środowisku Conda zagnieżdżonym w środowisku Julii. Obliczenia wykonano na komputerze z systemem Ubuntu 20.4, z procesorem Intel Core i7, o częstotliwości taktowania 2 GHz.**

* 1. **Uwaga na temat implementacji**

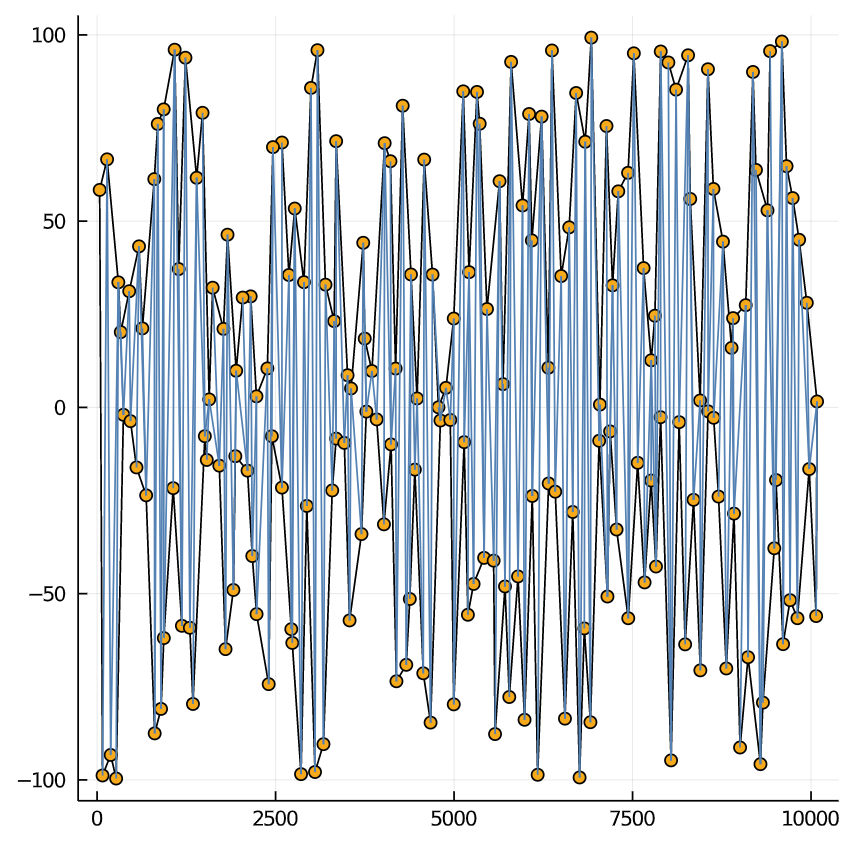
**Algorytmy klasyfikacji punktów oraz triangulacji działają dla wielokątów x-monotonicznych, z przyczyn wygody implementacyjnej. Prowadzi to do zagadnienia całkowicie równoważnego postawionemu w temacie zadania, ponieważ zmiana sprowadza się do obrotu układu współrzędnych, którego można łatwo dokonać.**

1. **Wykresy**
   1. **Zbiór I**



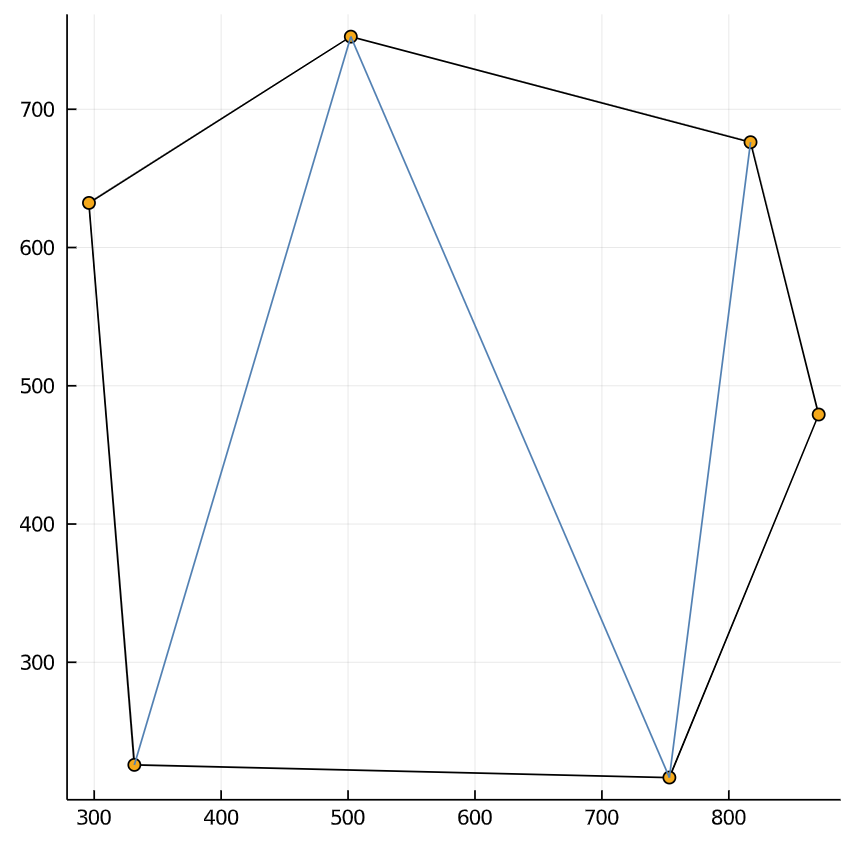
Rysunek Triangulacja zbioru wygenerowanego automatycznie

* 1. **Zbiór II**



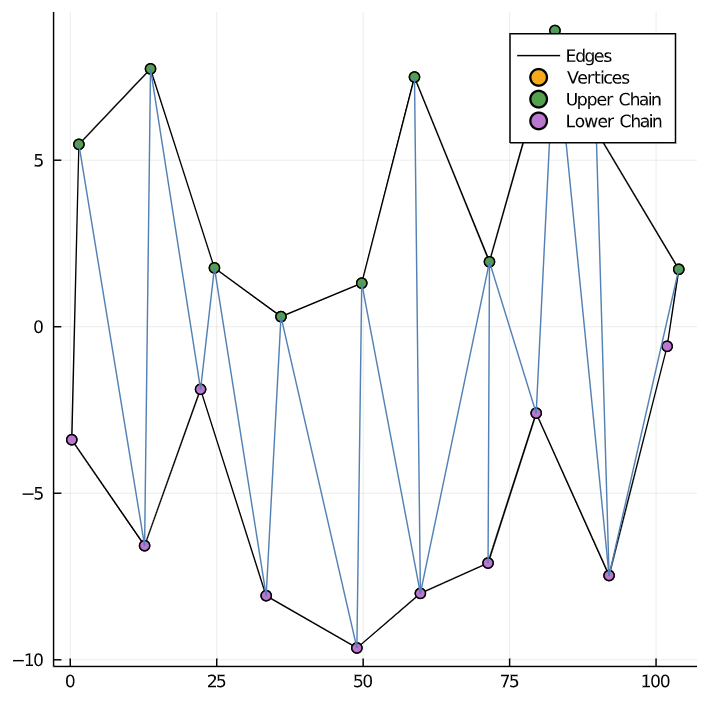
Rysunek Triangulacja zbioru wygenerowanego automatycznie

* 1. **Zbiór III**



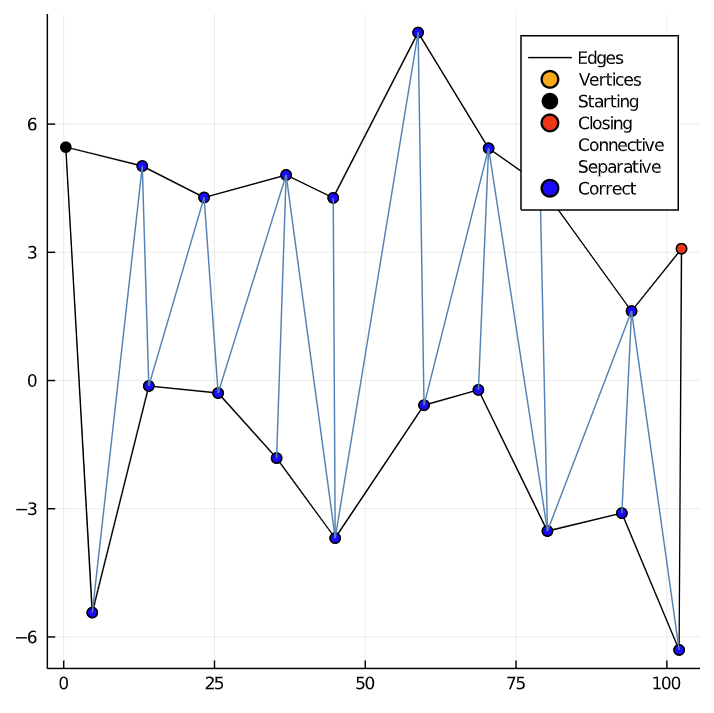
Rysunek Triangulacja zbioru podanego przez użytkownika

* 1. **Zbiór IV – Przykład wyświetlenia podziału na łańcuchy**



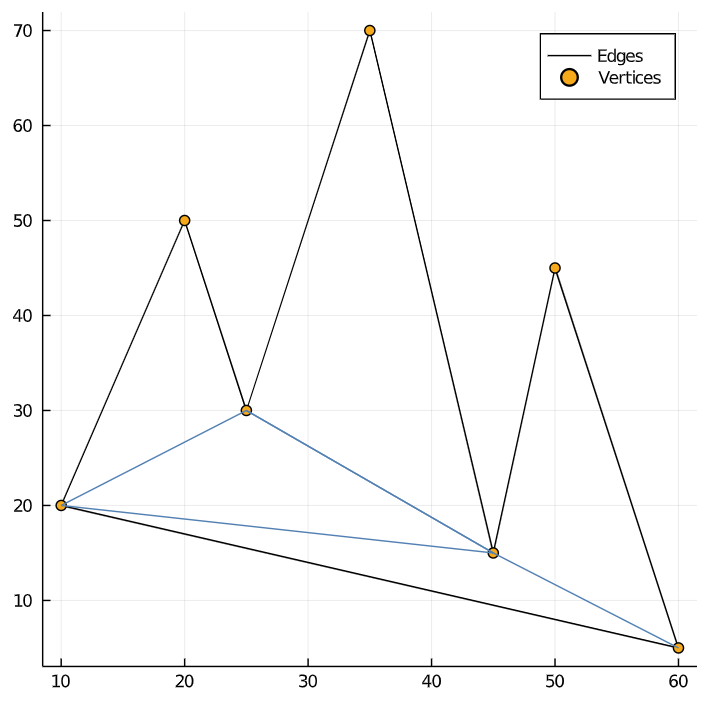
Rysunek 4 Podział na łańcuchy

* 1. **Zbiór V – Przykład topologicznej klasyfikacji punktów**



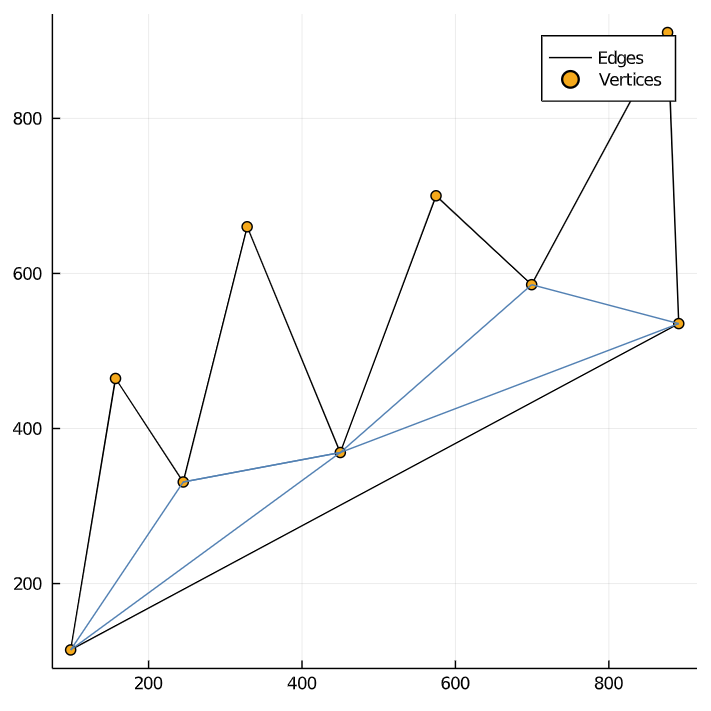
Rysunek 5 Klasyfikacja punktów na początkowe, końcowe, łączące, dzielące i prawidłowe

* 1. **Zbiór VI – Trudny przypadek I**



Rysunek 6 Triangulacja szczególnego przypadku wielokąta

* 1. **Zbiór VII – Trudny przypadek II**



Rysunek 7 Triangulacja szczególnego przypadku wielokąta

1. **Analiza działania algorytmu**
   1. **Jakość generowanych wyników i efektywność**

Algorytm klasyfikacji punktów działa dla przyjętych zbiorów niezawodnie.  
Efekt działania algorytmu triangulacji jest zgodny z oczekiwaniami. Niezależnie od *kształtu* wielokąta, algorytm jest efektywny. Dobór zbiorów pozwala zweryfikować ten fakt. Dla zbiorów skrajnie gęstych (o średniej długości boku i miarach kątów rzędu dokładności Float64) warto przyjąć niezerową tolerancję, by zapobiec tworzeniu przez algorytm przekątnych pokrywających się z bokami lub nieleżących wewnątrz wielokąta.

* 1. **Czas działania**
     1. **Tabela 1. Oszacowanie czasu działania dla przykładowych rozmiarów danych wejściowych:**

|  |  |
| --- | --- |
| **Liczność zbioru** | **Rząd wielkości czasu działania** |
| 100 |  |
| 1000 |  |
| 10000 |  |

* 1. **Dobór wielokątów**

Najwięcej najbardziej różnorodnych kroków (w tym sensie, że analizowane przez algorytm punkty pochodzą z różnych łańcuchów) można uzyskać dla takich wielokątów, dla których łańcuchy są podobnej liczności. Z tego względu generowane losowo wielokąty mają tyle samo punktów dolnego i górnego łańcucha (lub liczności tychże różnią się o 1).  
Algorytm zachłanny działa optymalnie dla tego typu zbiorów.  
Również dla zbiorów generowanych interaktywnie działanie algorytmu odbywa się w akceptowalnym czasie, ponieważ zbiory zadane przez użytkownika są w oczywisty sposób *prostsze* od zbiorów generowanych losowo.  
Rysunki 7 i 8 obrazują wynik działania algorytmu dla wielokątów o szczególnych własnościach, wskazanych przez Prowadzącą.