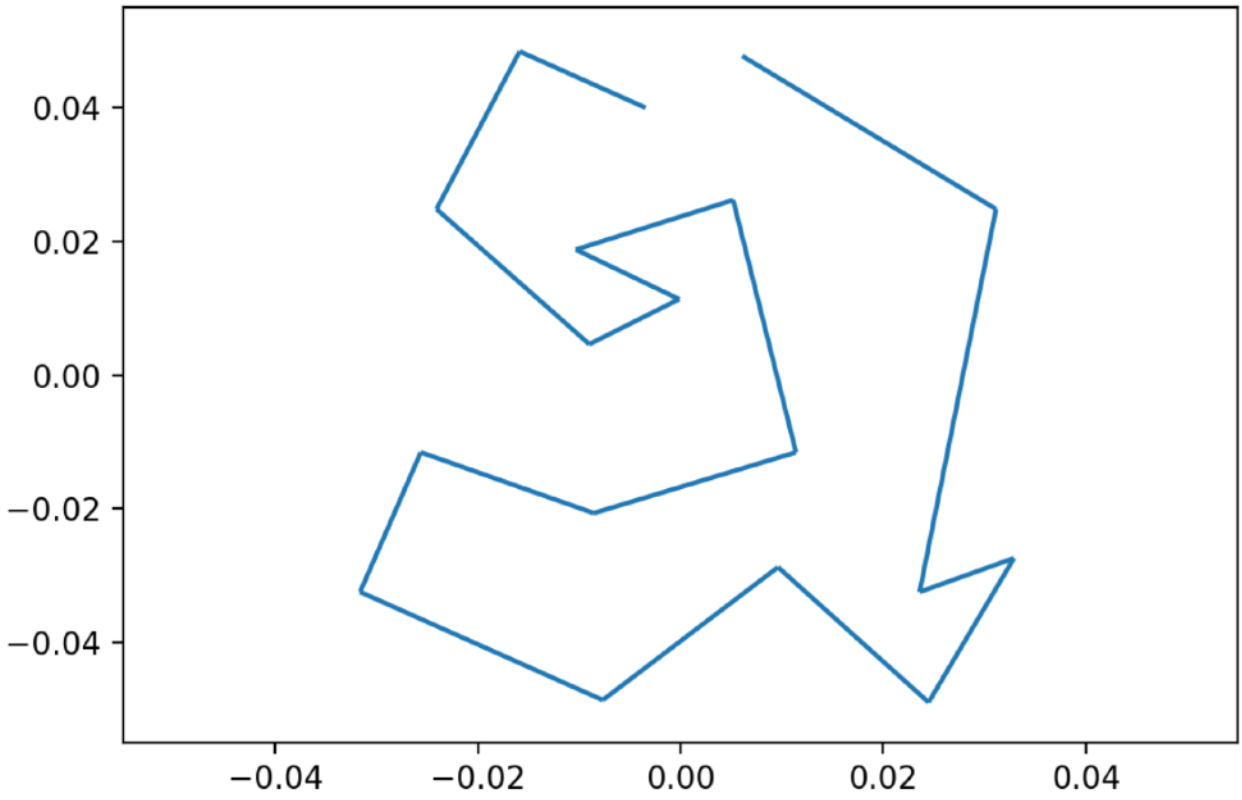
Geometria obliczeniowa
Lab 3
Oscar Teeninga

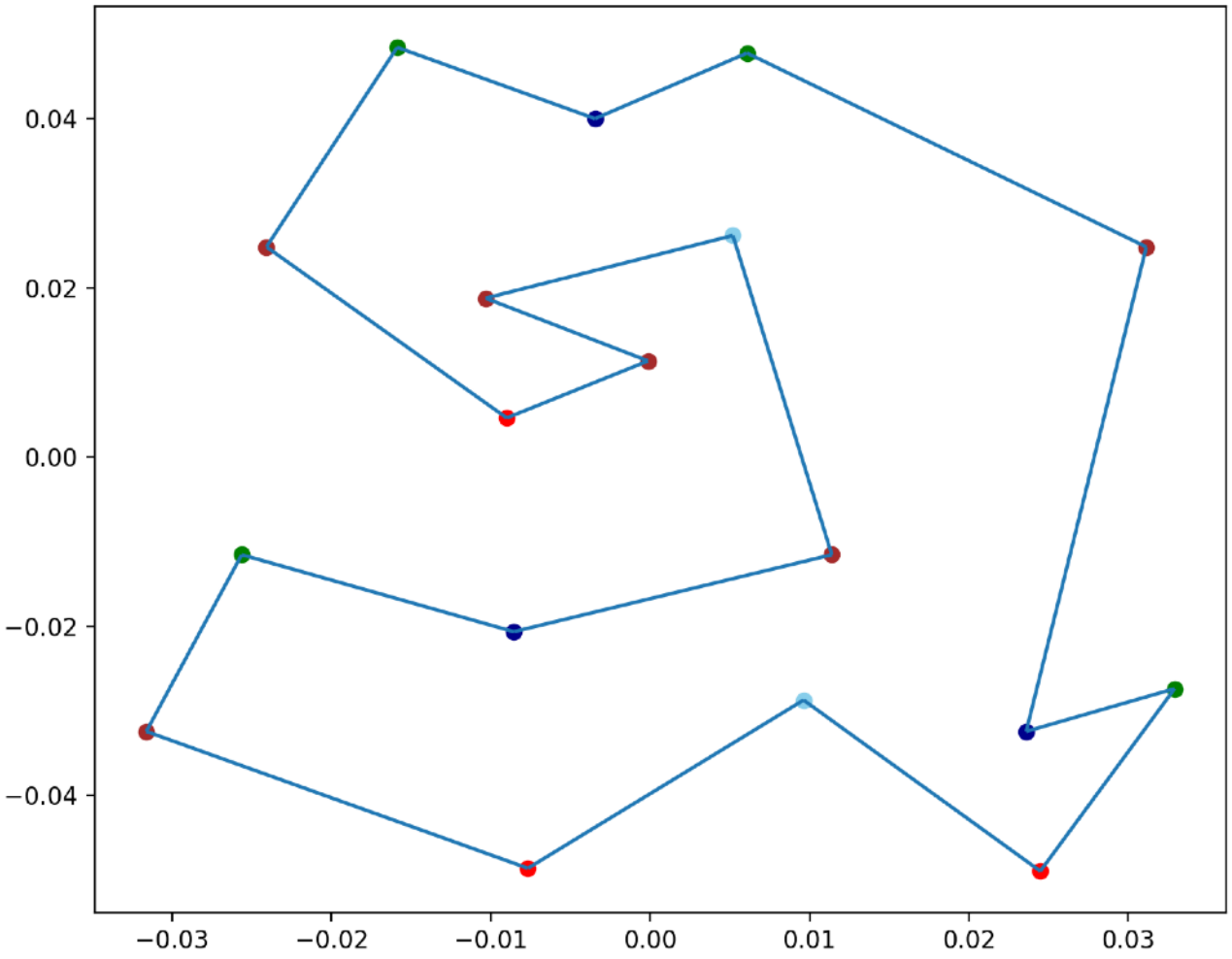
1. Aplikacja graficzna

   Korzystałem z dostarczonych na upel narzędzi oraz z opcji "Dodaj figurę". Założenie jest takie, że ostatnia krawędź w momencie definiowania nie jest rysowana, tzn. zaznaczamy tylko wierzchołki, a ostatnia krawędź zostaje potem dodana automatycznie w dalszych krokach.



2. Algorytm sprawdzania y-monotoniczność oraz wyszukiwania wierzchołków

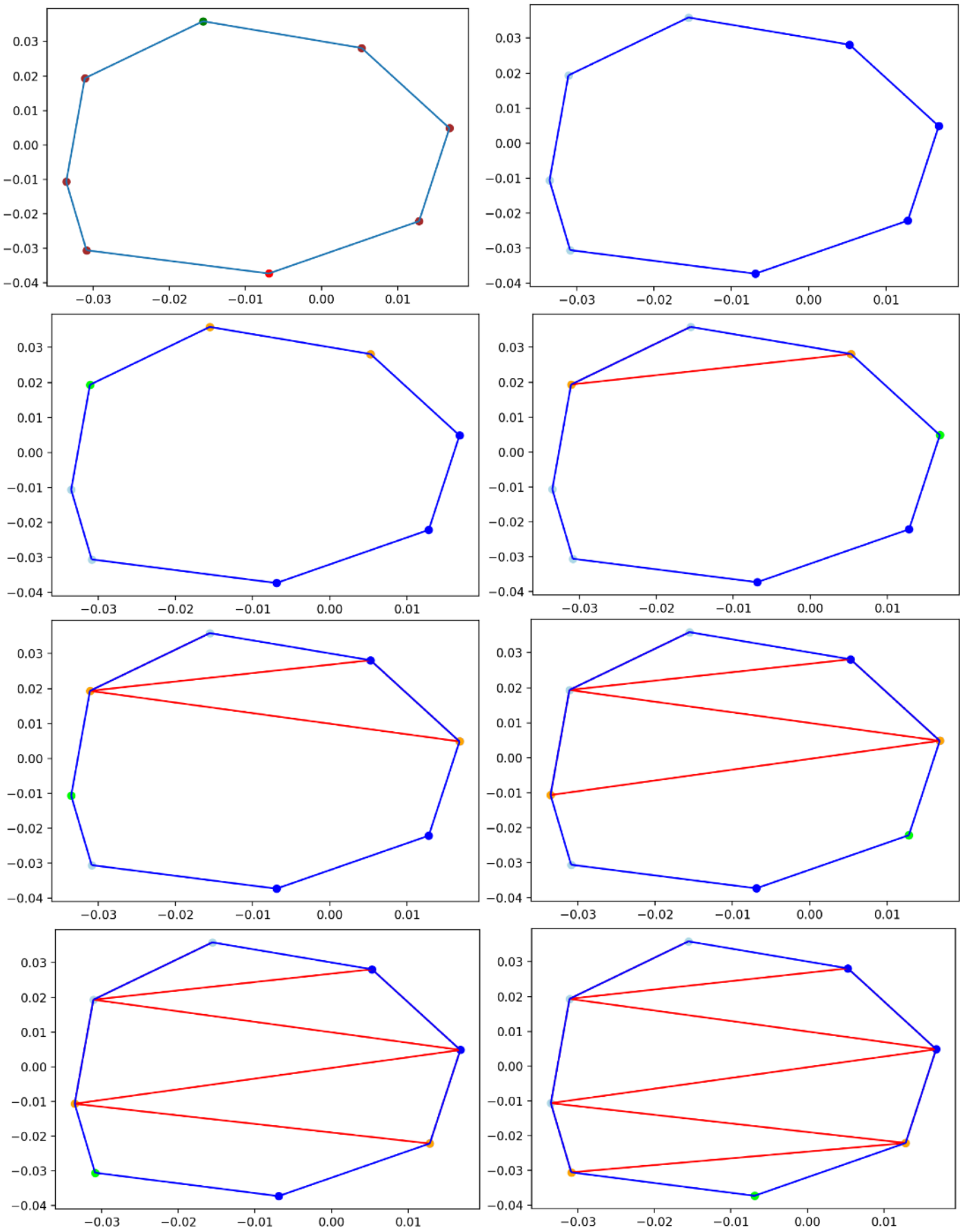
   Korzystając z własności sprawdzania położenia punktów względem siebie za pomocą wyznacznika omawianego na pierwszych labach sprawdzałem, czy kąt między trzeba punktami jest większy bądź mniejszy od pi sprowadzało się do sprawdzenia, czy punkt jest po lewej, czy prawej stronie prostej wyznaczanej przez dwa punkty. Następnie iterowałem po wszystkich punktach i kwalifikowałem każdy punkt do jednej z 5 kategorii. Ostatecznie sprawdzenie y-monotoniczności sprowadzało się do sprawdzenia czy istnieją jakieś punkty łączące lub dzielące. Kolory są zgodne z tymi na wykładzie.




```
In [135]: print(is_y_monotone(points))
          False
```

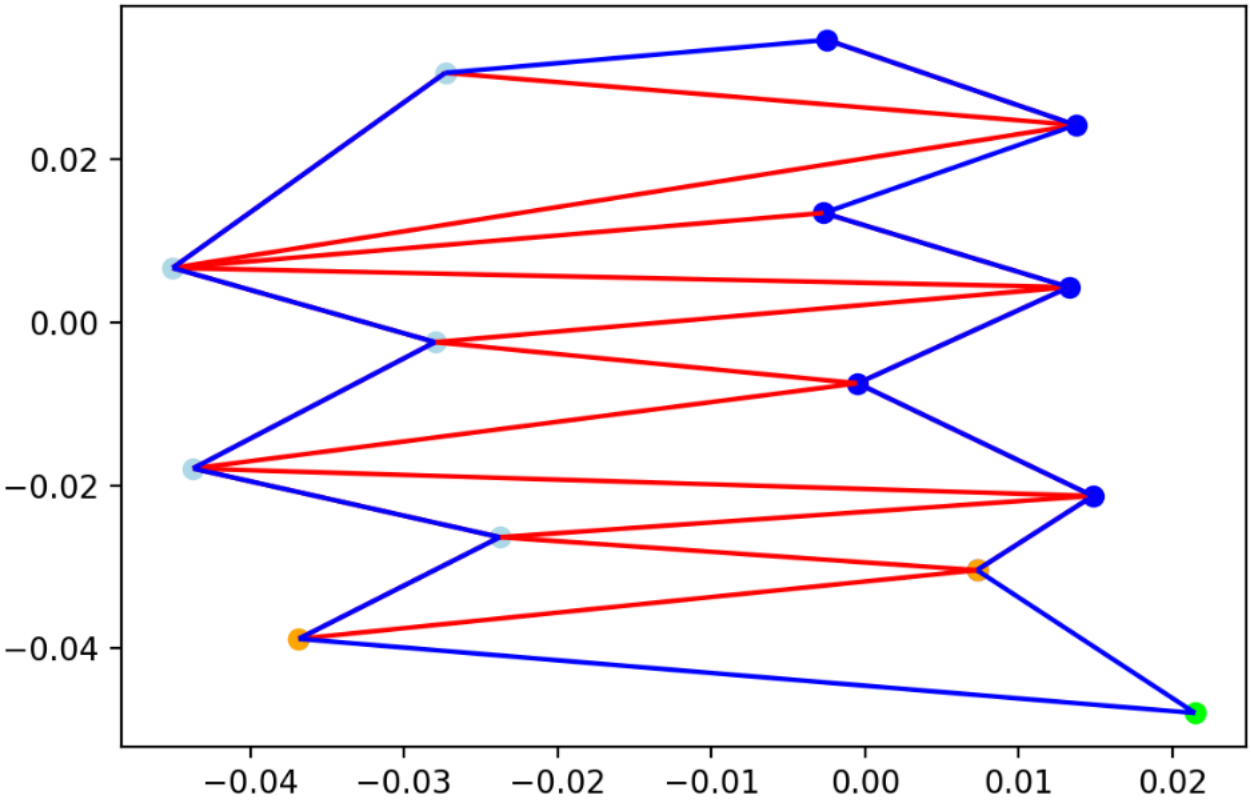
3. Triangulacja wielokąta y-monotonicznego

W poprzednim przykładnie wielokąt nie był y-monotoniczny, więc stworzyłem nowy. Punkty należy interpretować: **na stosie**, **przetwarzany**, **w lewym/prawym łańcuchu.** Trójkąty były zapisywane w formie trzyelementowej tablicy.

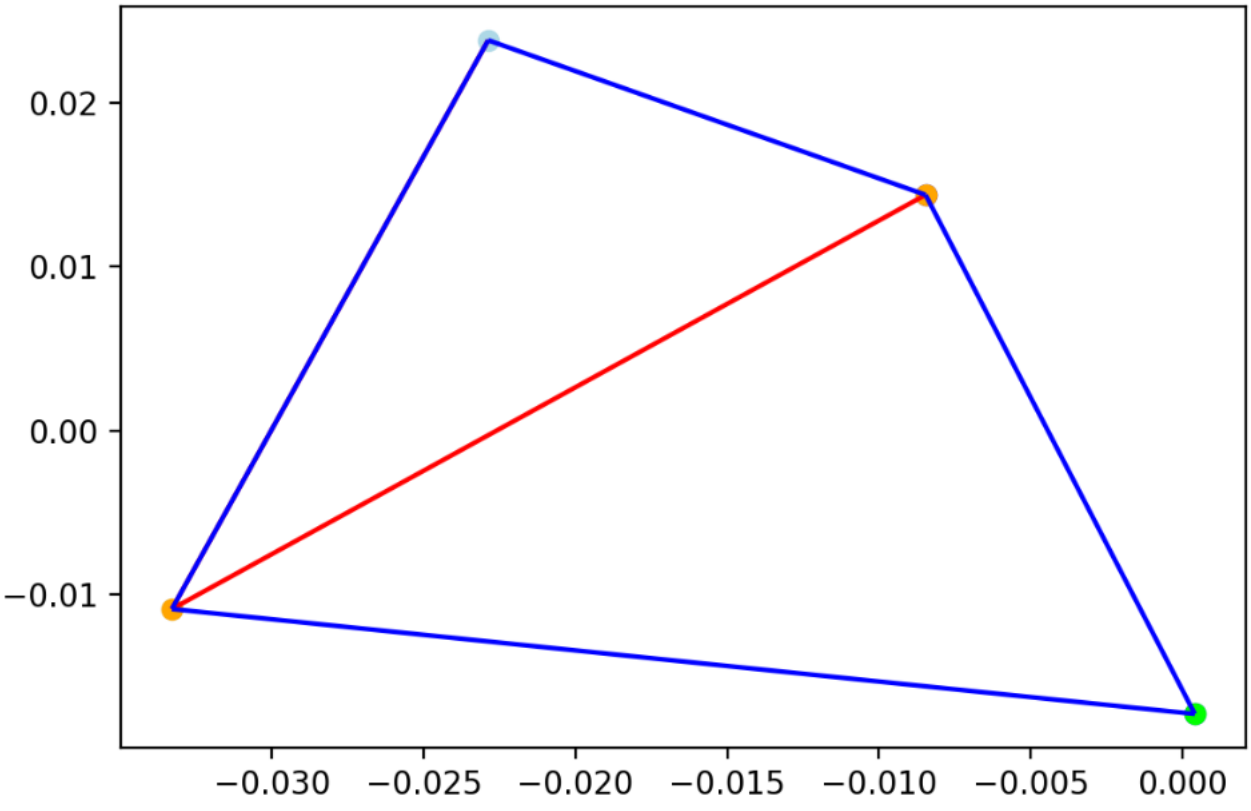


4. Inne wielokąty

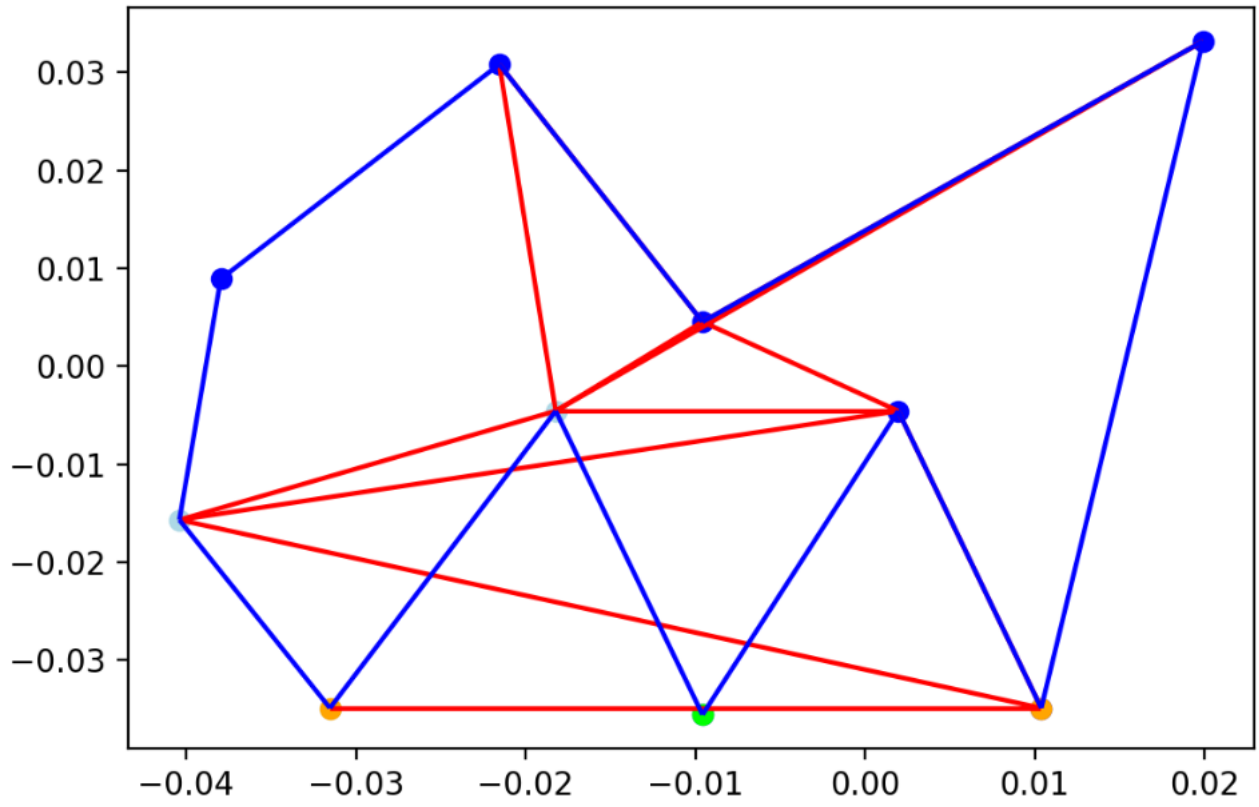
Do zaprezentowania poprawności działania algorytmu stworzyłem kilka dodatkowych wielokątów.



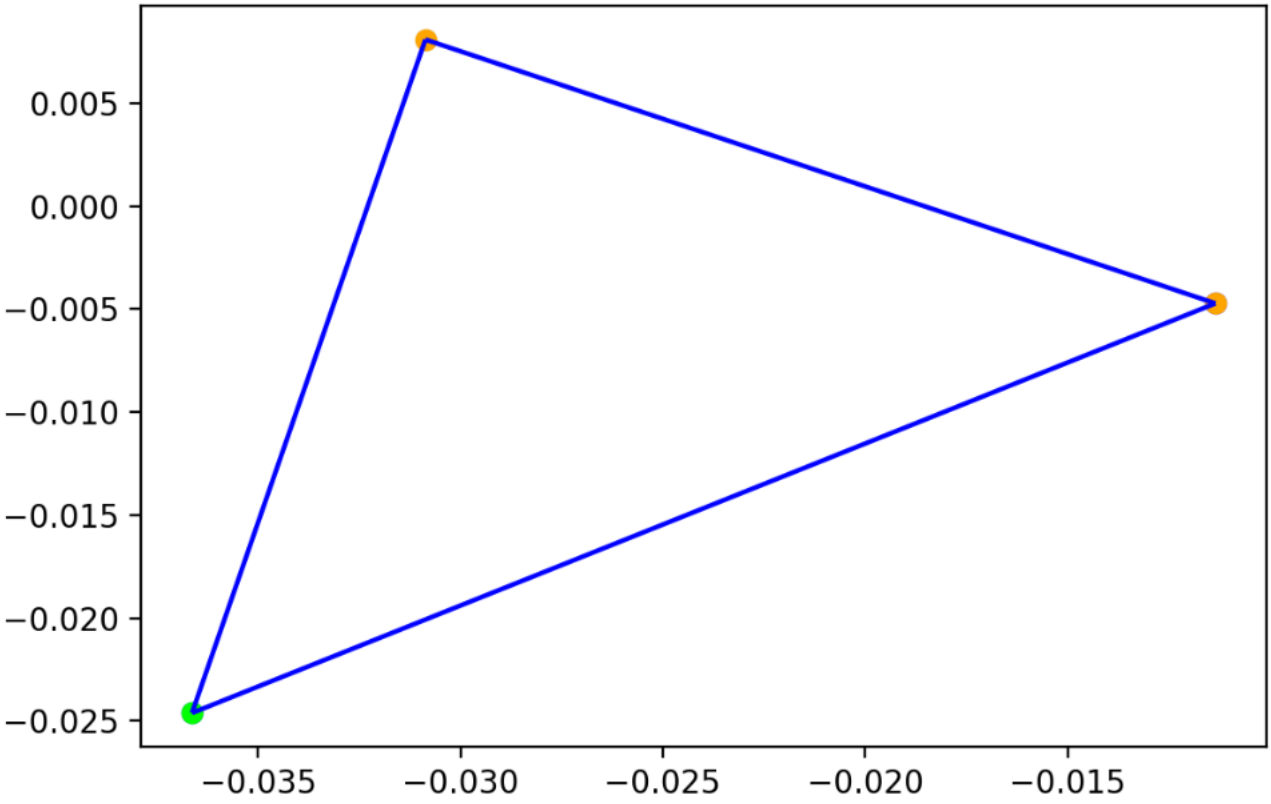
Triangulacja na wielokącie o większej liczbie wierzchołków



Triangulacja na wielokącie o małej liczbie wierzchołków



Triangulacja na wielokącie nie y-monotonicznym



Triangulacja na wielokącie będącym trójkątem

5. Łapki

Dodatkowo przeprowadziłem testy na algorytmie z bardziej specifycznym wielokątem, proponowanym na labolatorium