

Algorytmy Geometryczne

Zadanie 4, laboratorium 4 - sprawozdanie

Radosław Kawa

1. Opis ćwiczenia

Na laboratorium 4 na początek należało dostosować aplikacje graficzna, aby można było zadawać odcinki myszką

Następnie zaimplementować oraz przetestować działanie 4 algorytmów:

- Funkcje generującą odcinki z wybraną ilością z zadanego przedziału 2D
- Algorytm zamiatańia sprawdzający, czy choć jedna para odcinków w zadanym zbiorze się przecina.
- Algorytm wyznaczający wszystkie przecięcia zadanego odcinków, na wyjściu podający liczbę wykrytych przecięć, współrzędne przecięć oraz dla każdego przecięcia odcinki, które się przecinają

2. Biblioteki oraz specyfikacja

Zadanie było wykonane w jupyter notebook, w języku Python. Wersja 3.10.0. Do rysowania wykresów zostało użyte narzędzie dostarczone na laboratorium, które wykorzystuje bibliotekę matplotlib. Umożliwiło to zadawanie odcinków myszką. Do wyświetlania informacji o przecięciach użyto dataframe z biblioteki pandas, aby było to bardziej czytelne

Specyfikacja:

System operacyjny: macOS Monterey

Procesor: Apple M1

Pamięć: 8GB

3. Opis algorytmów

Algorytm zamiatańia sprawdzający, czy choć jedna para odcinków w zadanym zbiorze się przecina:

W implementacji użyto struktury sortedset (biblioteka sortedcontainers), użyto ją jako stan miotły. Struktura ta wykonuje operacje takie jak dodawanie usuwanie, wyciąganie elementu (pop()) w czasie $O(\log(n))$, całkowity czas działania to $O(n\log(n))$, natomiast informacja czy dany element się znajduje w strukturze to $O(1)$. Przechowuje ona odcinki posortowane względem współrzędnej y.

Natomiast do struktury zdarzeń użyłem zwykłego arraya, na którym utworzyłem strukturę heap, (kolejka priorytetowa z użyciem kopca), wyciąganie elementu $O(\log(n))$, sortowanie $O(n\log(n))$, wykorzystuje funkcje heapify i heappop. Struktura sortuje względem współrzędnej x-owej, a w niej przechowuje punkty odcinków, aby przeglądać te zdarzenia po kolej (od lewej do prawej). Z racji tego, że potrzebuje tylko informacji czy jakieś linie się przecięły nie musze przechowywać pozycji przecięć, więc sortuje tylko raz oraz ilość danych w strukturze się nie zmienia.

Algorytm wyszukuje przecięcia na podstawie algorytmu zamiatańia, przechodzę od lewej do prawej współrzędnych x, następnie biorę punkt sprawdzam czy jest początkiem, czy końcem odcinka. Jeśli to jest początek to sprawdzam czy istnieje przecięcie tego odcinka z jego sąsiadem. Natomiast jeśli jest to koniec odcinka to sprawdzam czy istnieje przecięcie jego sąsiadów. Sprawdzanie przecięcia nie ma miejsca ponieważ, przy znalezieniu przecięcia algorytm zwraca informacje o przecięciu w postaci wykresu.

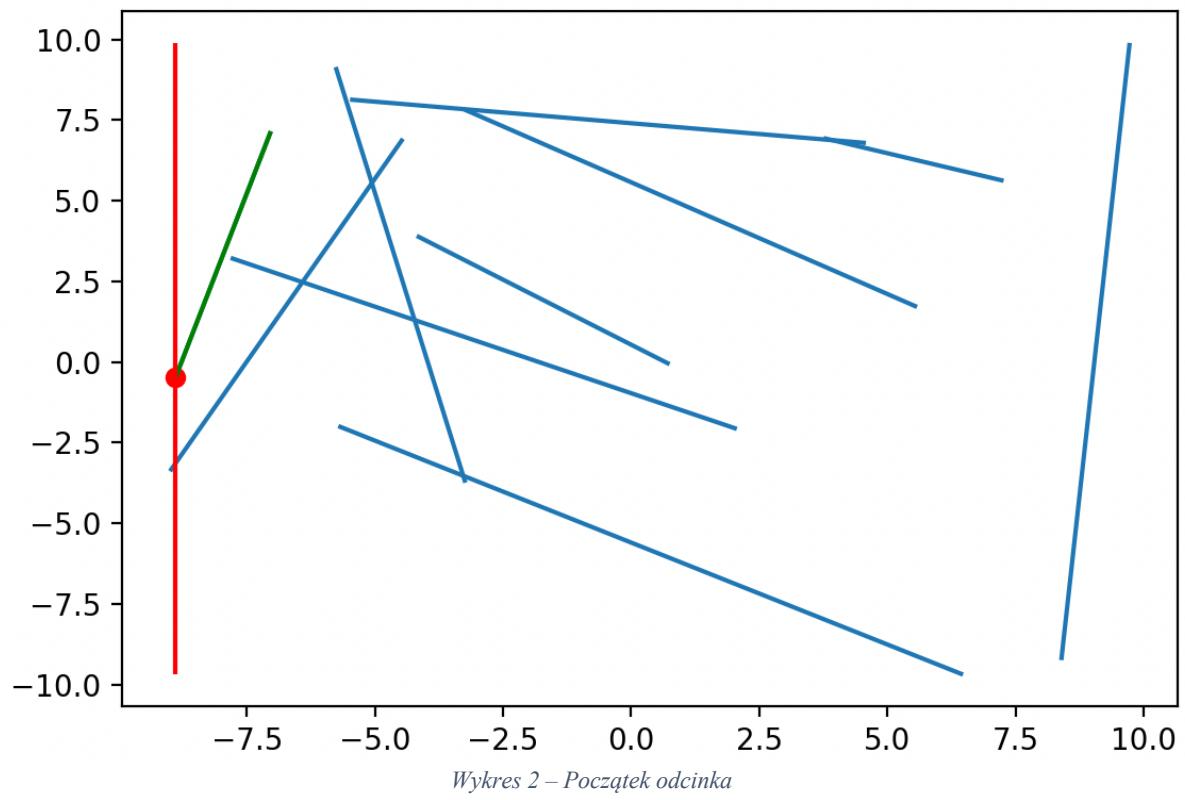
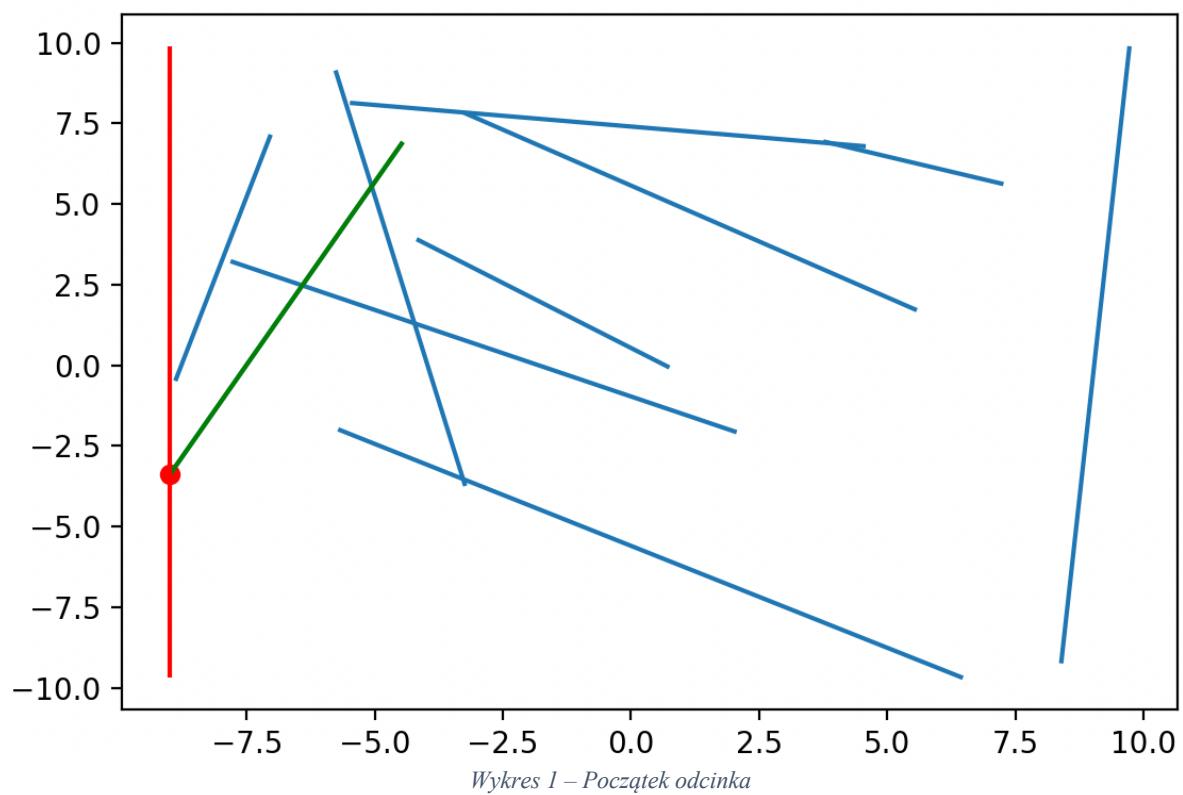
Przykładowe działanie algorytmu:

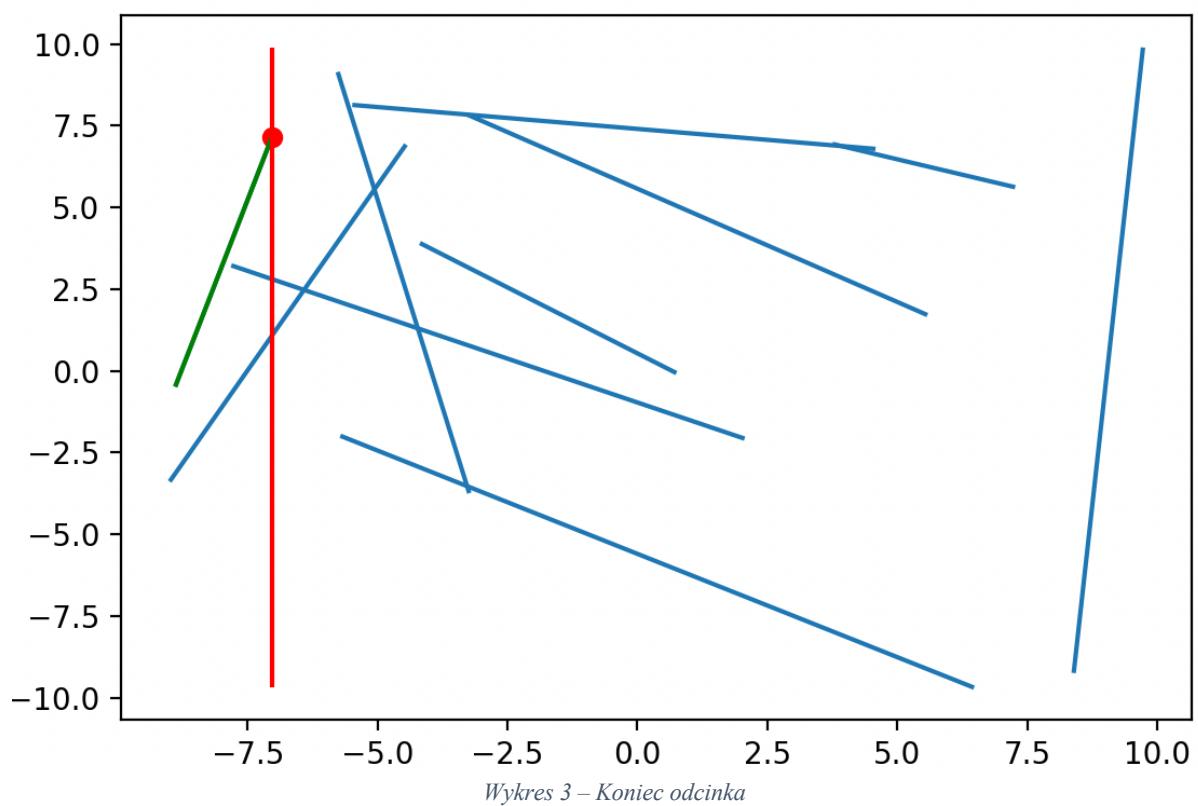
Informacja o wykresach:

Czerwona linia razem z punktem – Miotła

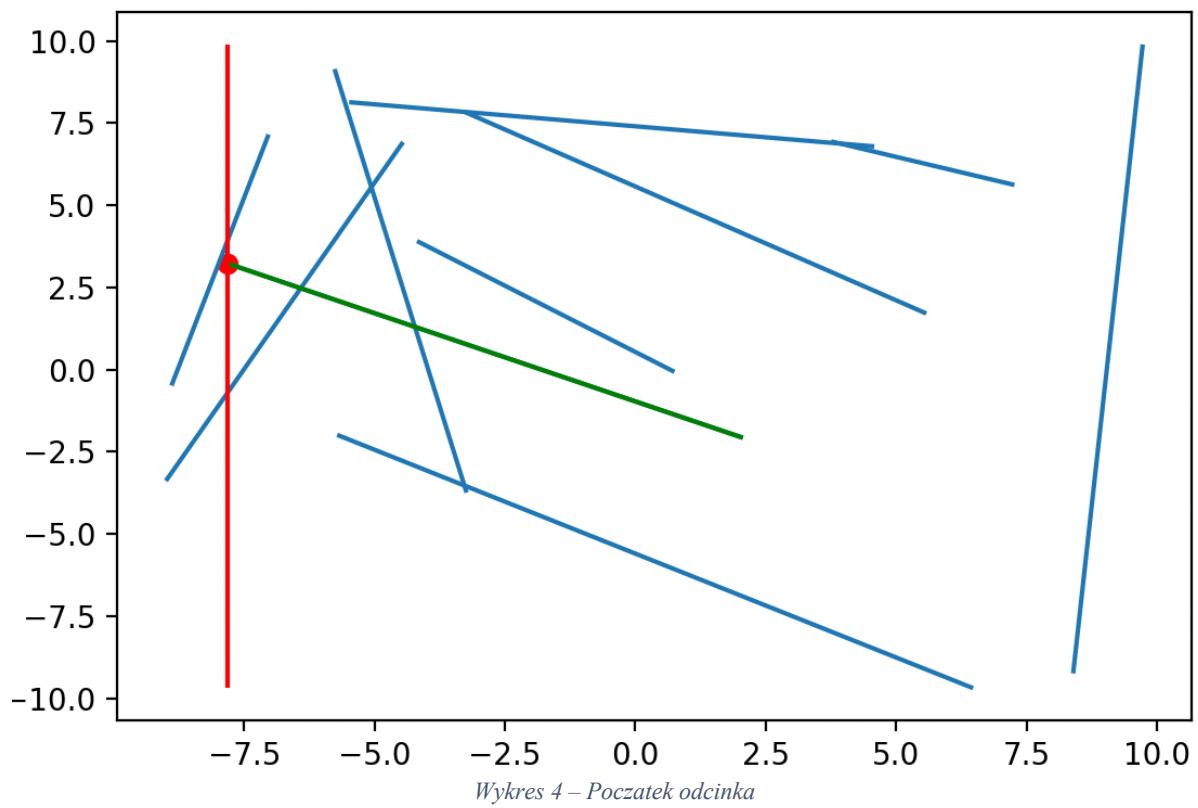
Czerwone punkty – punkty przecięcia

Zielona linia – aktualnie badana linia

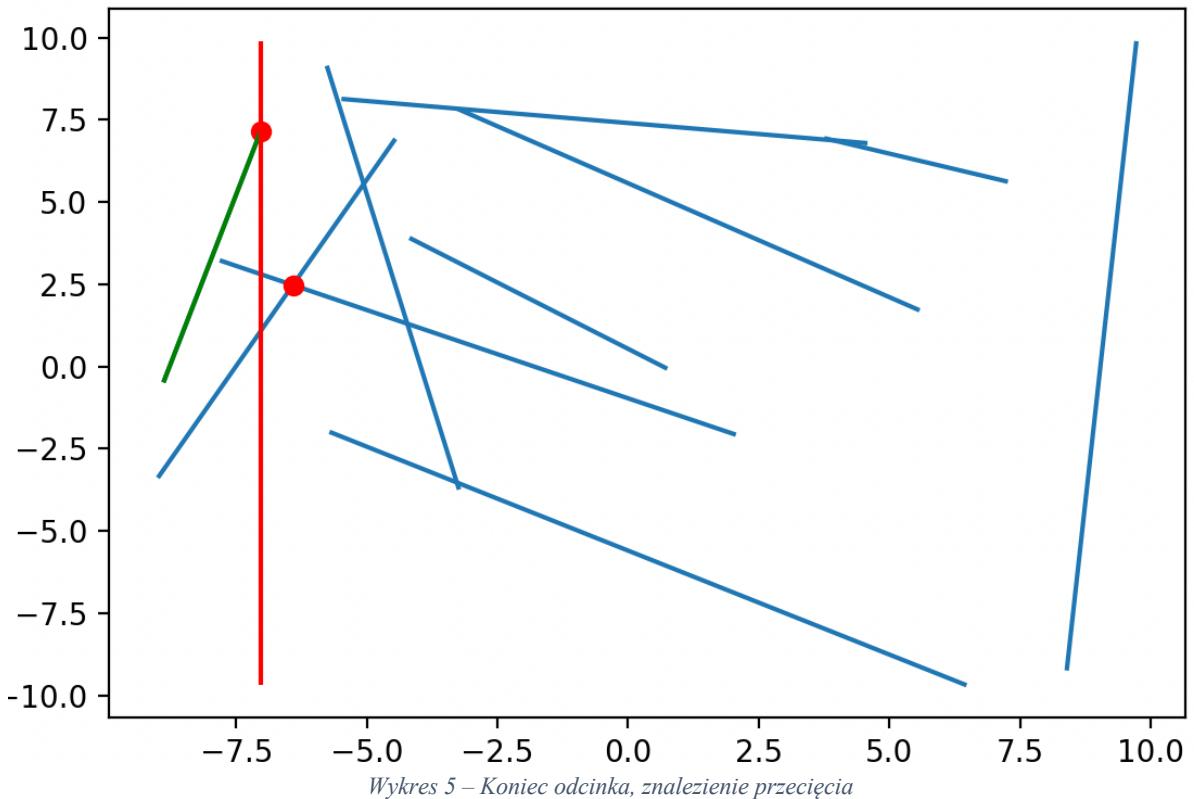




Wykres 3 – Koniec odcinka



Wykres 4 – Początek odcinka



W tym przypadku znalezione przecięcie tworzyły linie już wcześniej dodane (sąsiedzi)

Algorytm wyznaczający wszystkie przecięcia zadanych odcinków, na wyjściu podający liczbę wykrytych przecięć, współrzędne przecięć oraz dla każdego przecięcia odcinki, które się przecinają:

Algorytm działa praktycznie tak samo jak poprzedni, struktura stanu miotły to dalej sortedset natomiast jako strukturę zdarzeń tym razem wykorzystałem sortedset zamiast heap. Głównie ze względu na szybkość przeszukiwania tej struktury, wyciągania i dodawania elementów. Dodatkowo struktura nie pozwala mi zwielokrotniać elementów. (np. przecięć)

Aby zabezpieczyć się przed zwielokrotnionymi przecięciami związanymi z sposobem wyznaczania wyznaczników i liczbami zmiennoprzecinkowymi, za każdym razem gdy dodaje przecięcie sprawdzam czy dane przecięcie już było to znaczy:

przechowuje przecięcia w zbiorze w postaci (odcinek1, odcinek2) – za każdym gdy takie znajdę dodaje do tej struktury tę informację.

w momencie gdy mam dodać przecięcie sprawdzam czy takie odcinki już się nie znajdują w tej strukturze czyli sprawdzam czy (odcinek1, odcinek2) znajduje się w secie lub (odcinek2, odcinek1) – w przypadku gdy zamienione są miejscami Są to odcinki, które nie są w żaden sposób modyfikowane, więc działa to tak samo jakbym przechowywał ich indexy.

Algorytm wyszukuje przecięcia na podstawie algorytmu zamiatania, przechodzę od lewej do prawej współrzędnych x, następnie biorę punkt (zdarzenie) sprawdzam czy jest początkiem, czy końcem odcinka czy przecięciem. Gdy jest początkiem to dodaje ten odcinek do stanu miotły i następnie sprawdzam czy istnieje przecięcie z jego sąsiadem, jeśli tak to aktualizuję strukturę zdarzeń, dodaje do niej przecięcie oraz zapisuje informacje o przecięciu.

Gdy jest końcem to najpierw sprawdzam czy przypadkiem nie istnieje przecięcie sąsiadów odcinka tego zdarzenia jeśli tak to dodaje przecięcie do struktury zdarzeń oraz zapisuje informacje o przecięciu. Na koniec usuwam odcinek ze stanu miotły.

Gdy jest przecięciem, to najpierw dodaje przecięcie do zbioru przecięć następnie wyciągam oba odcinki tego przecięcia, usuwam ze stanu miotły (działa tak samo jak usuwanie w podejściu końca odcinka), zmieniam klucz sortowania w stanie miotły, na podstawie którego zamienią mi się miejscami te dwa odcinki, tzn. zwiększam w kluczu sortowania punkt x o epsilon = 10^{-12} i wyznaczam y po którym te wartości są sortowane. Dzięki temu punkty delikatnie się przesuną w prawo, a odcinki które aktualnie badam zamienią się miejscami. Aktualizuje strukturę po zmianie klucza. Na koniec dodaje odcinki ponownie (działa tak samo jak w podejściu początku odcinka)

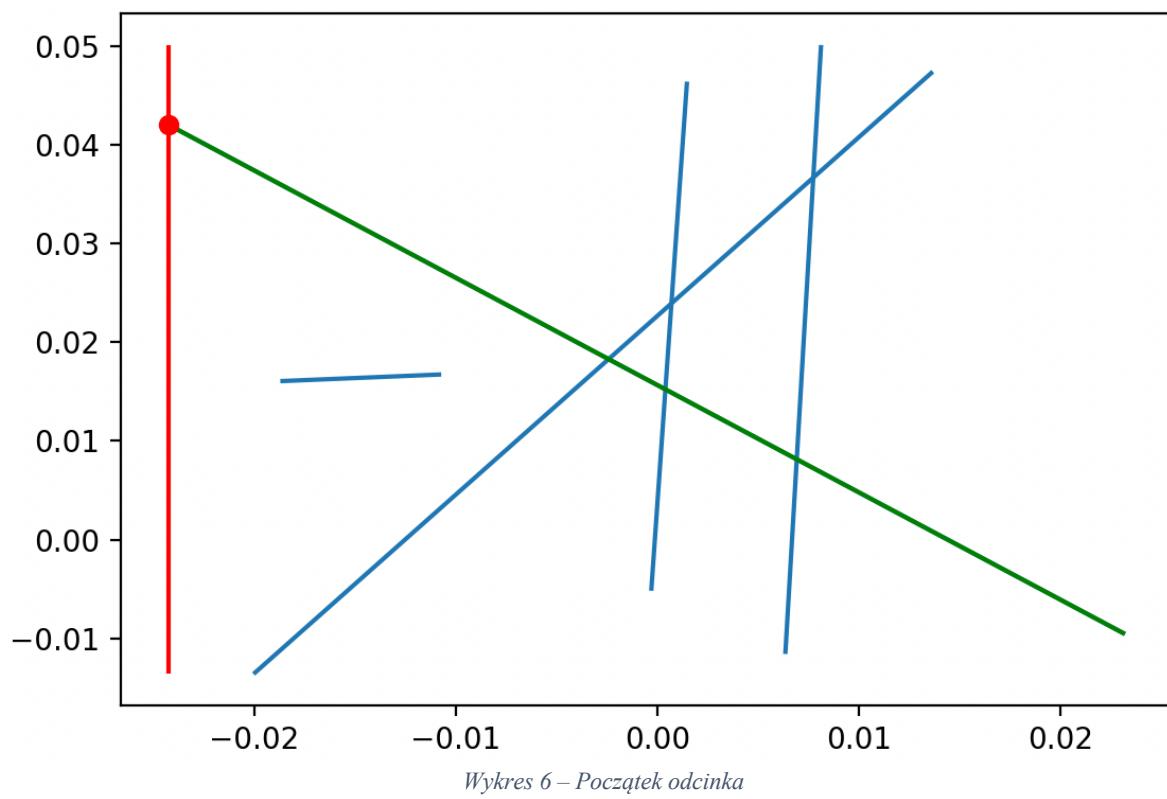
Przykładowe działanie algorytmu:

Informacja o wykresach:

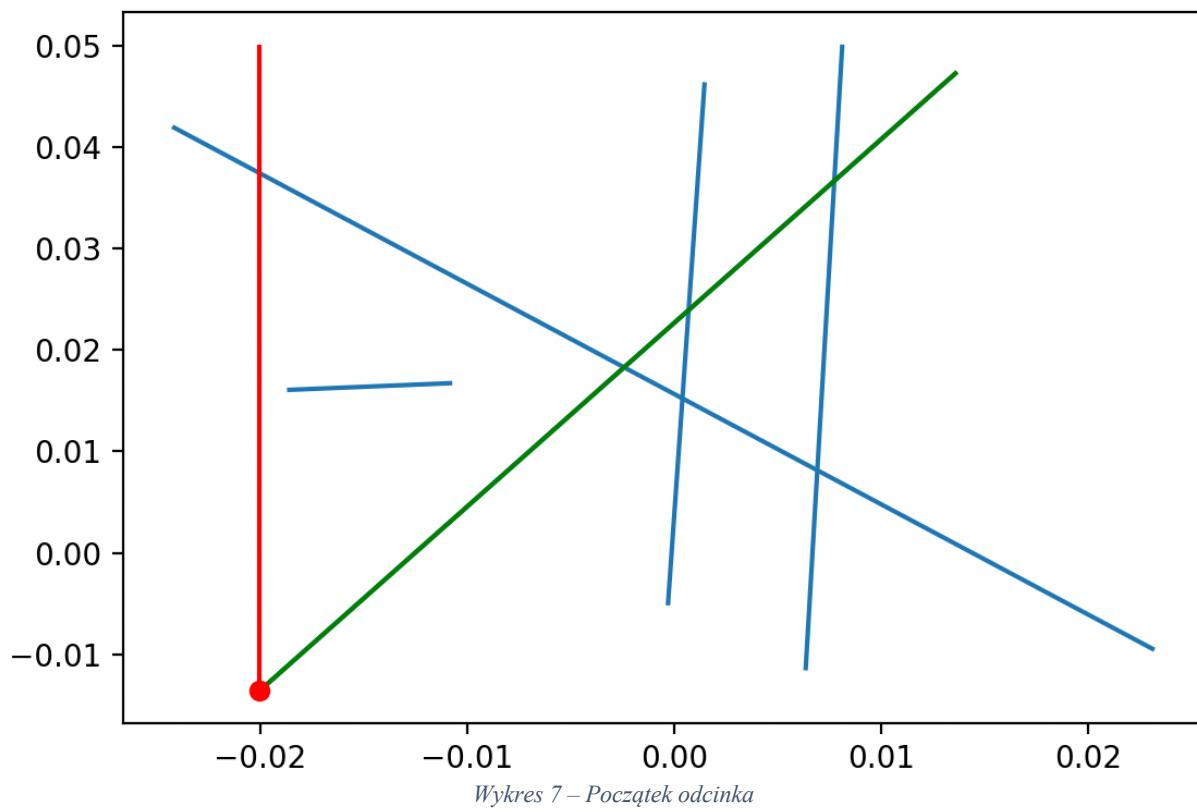
Czerwona linia razem z punktem – Miotła

Czerwone punkty – punkty przecięcia

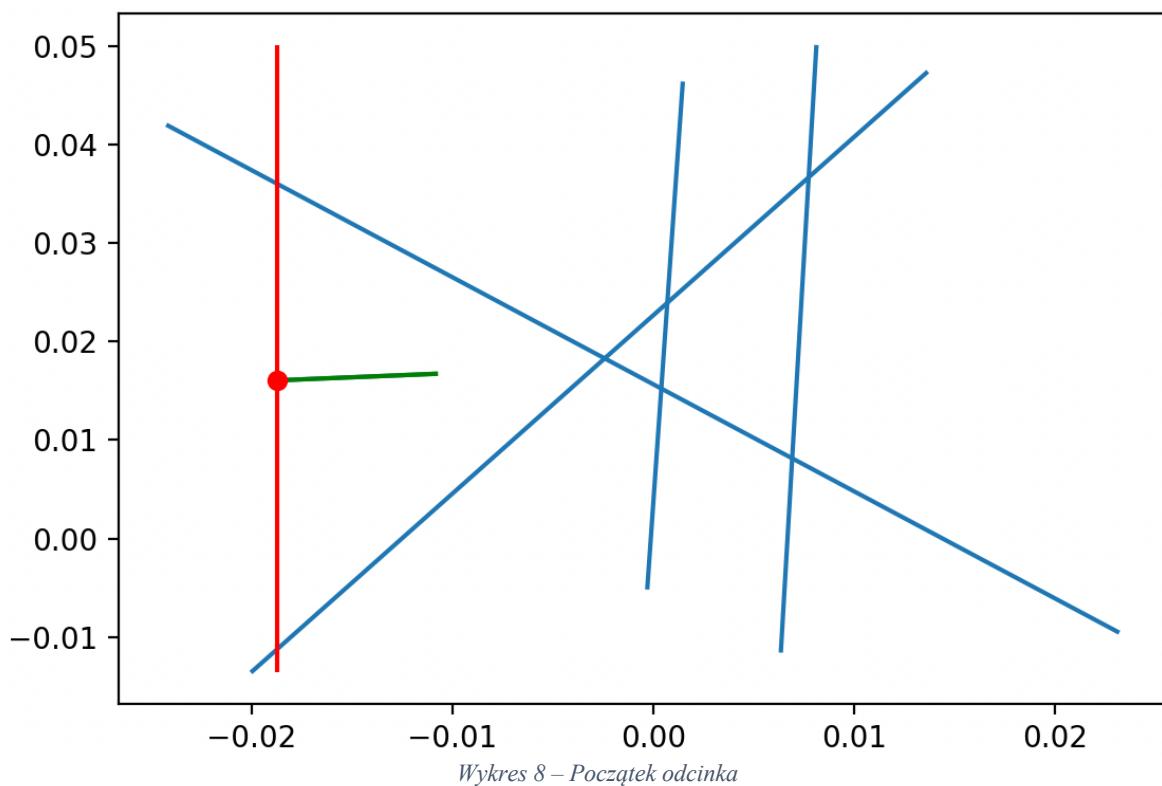
Zielona linia – aktualnie badana linia



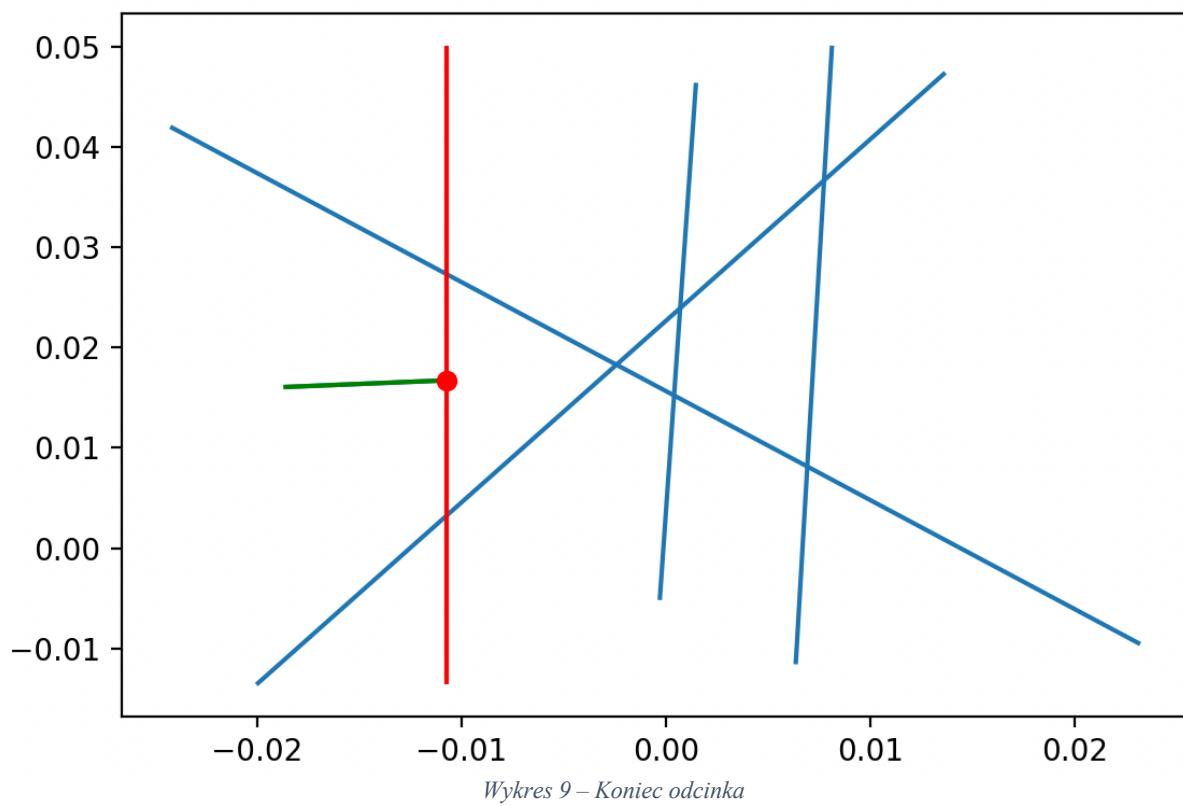
Wykres 6 – Początek odcinka



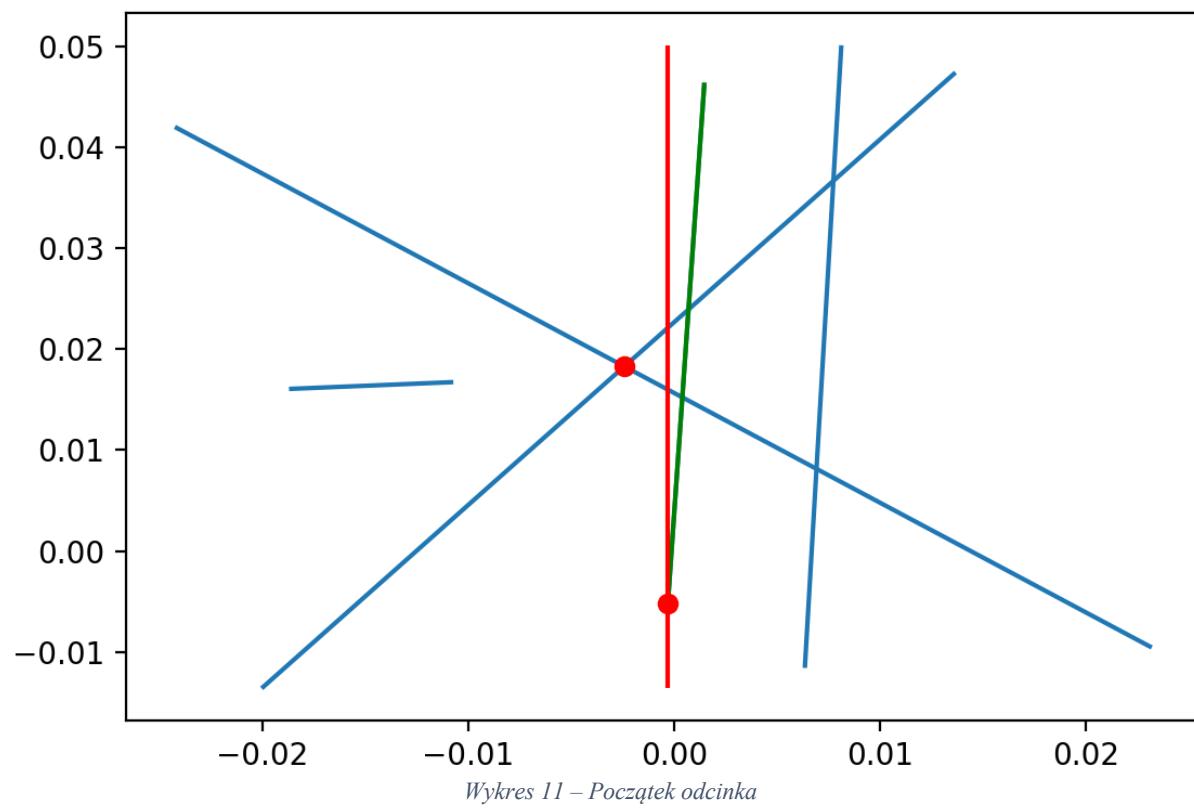
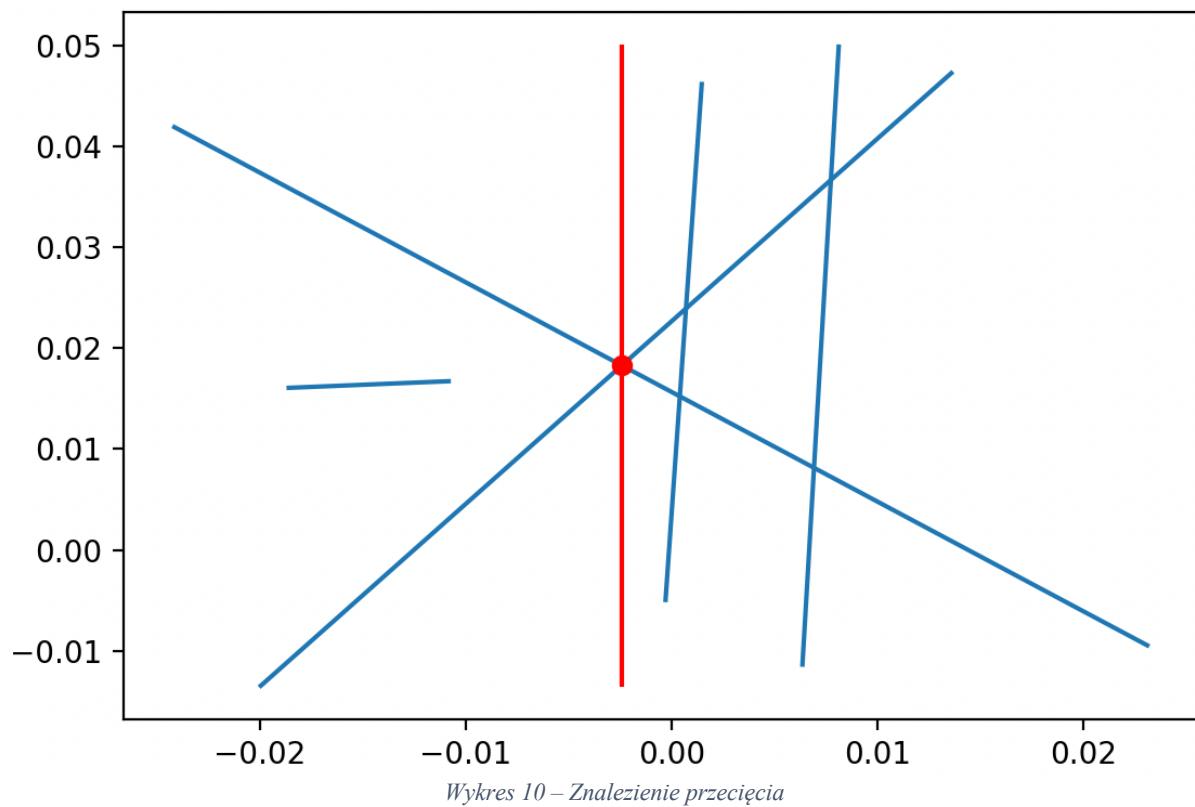
Wykres 7 – Początek odcinka

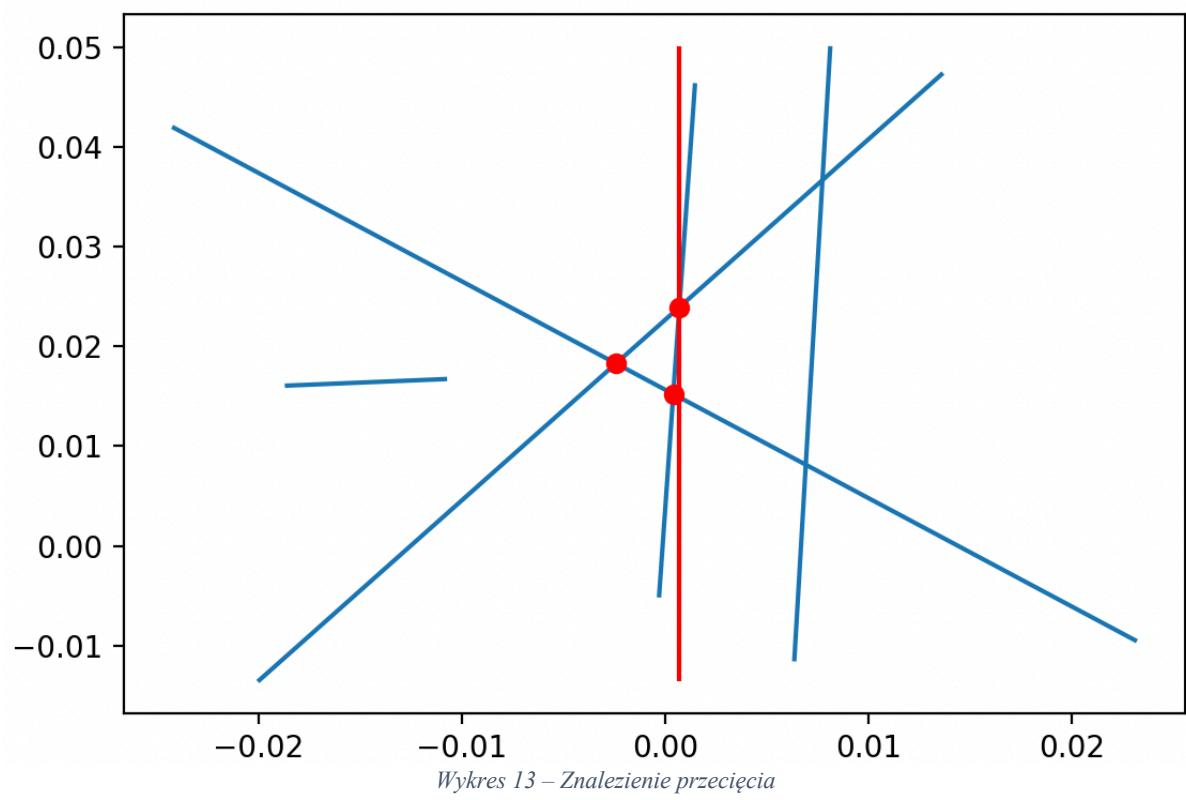
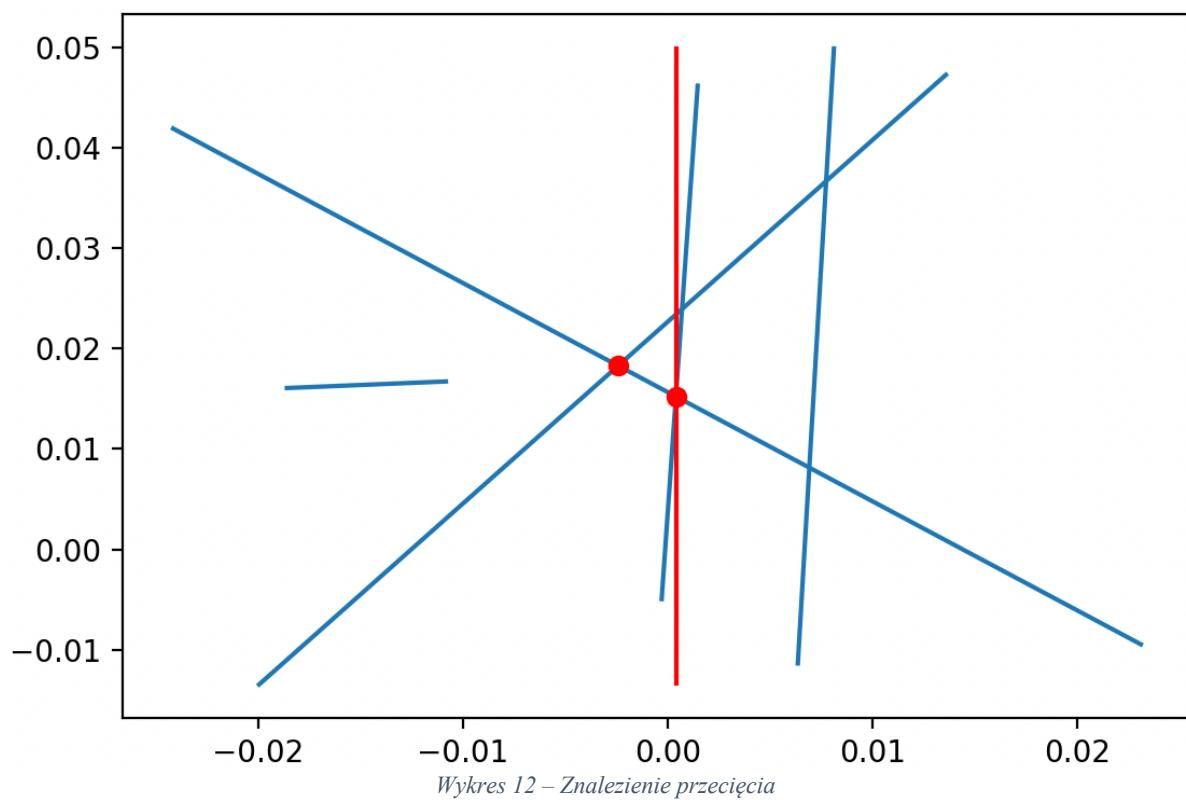


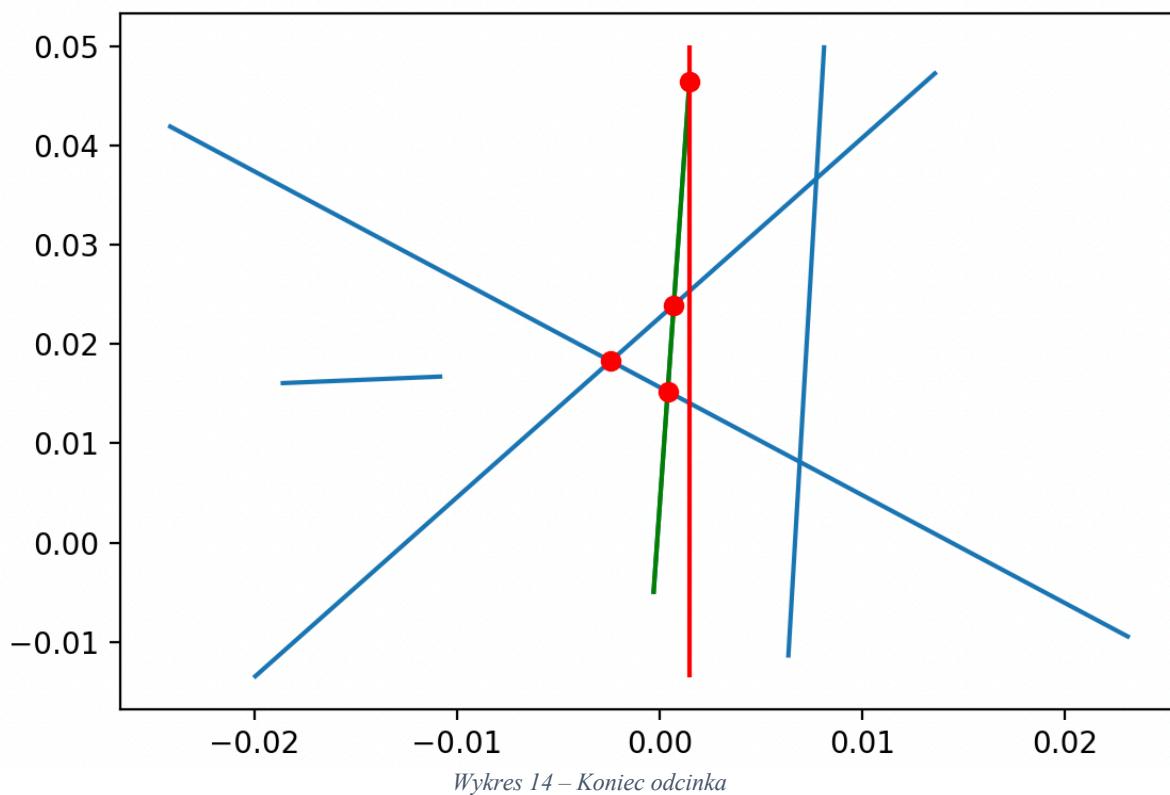
Wykres 8 – Początek odcinka



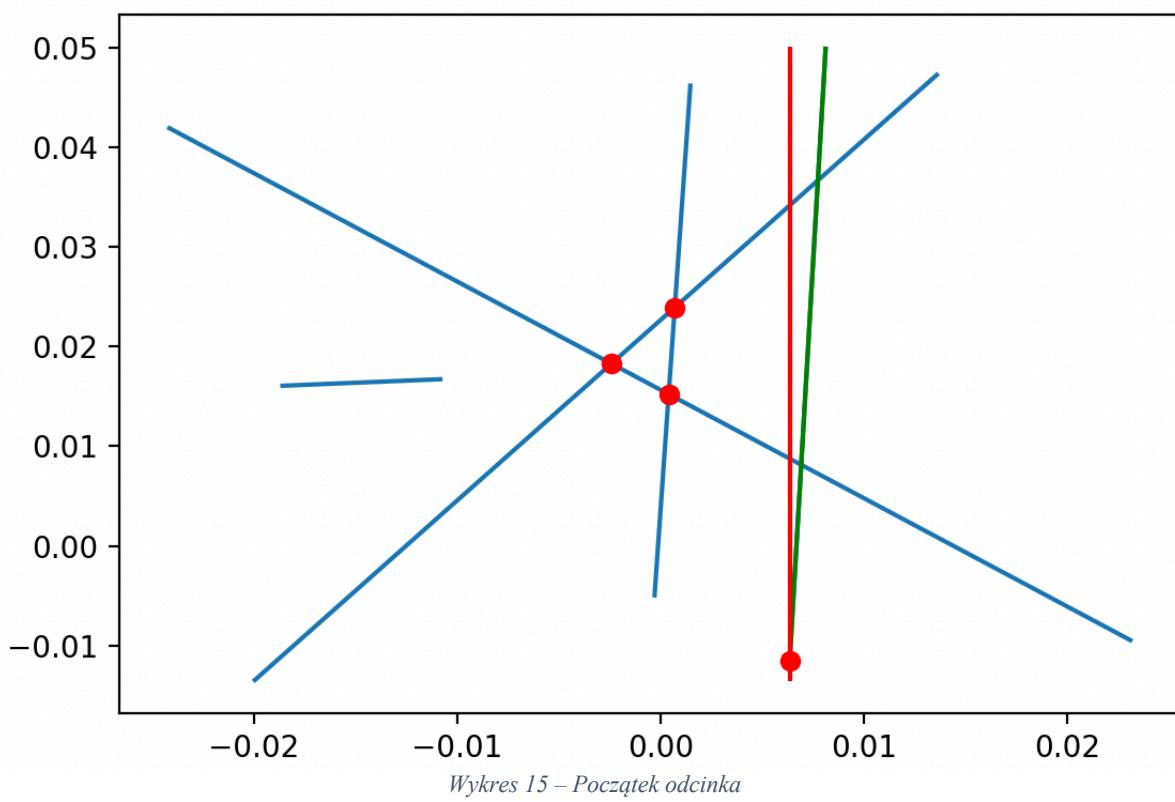
Wykres 9 – Koniec odcinka



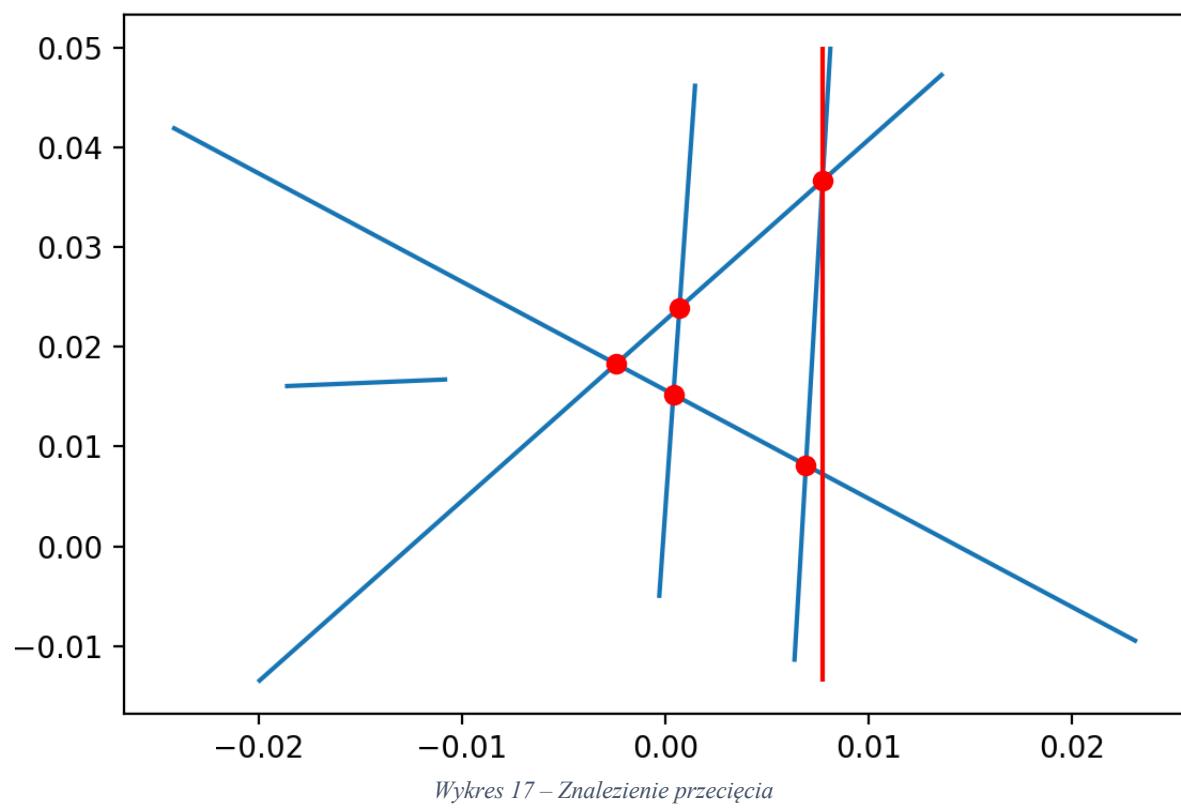
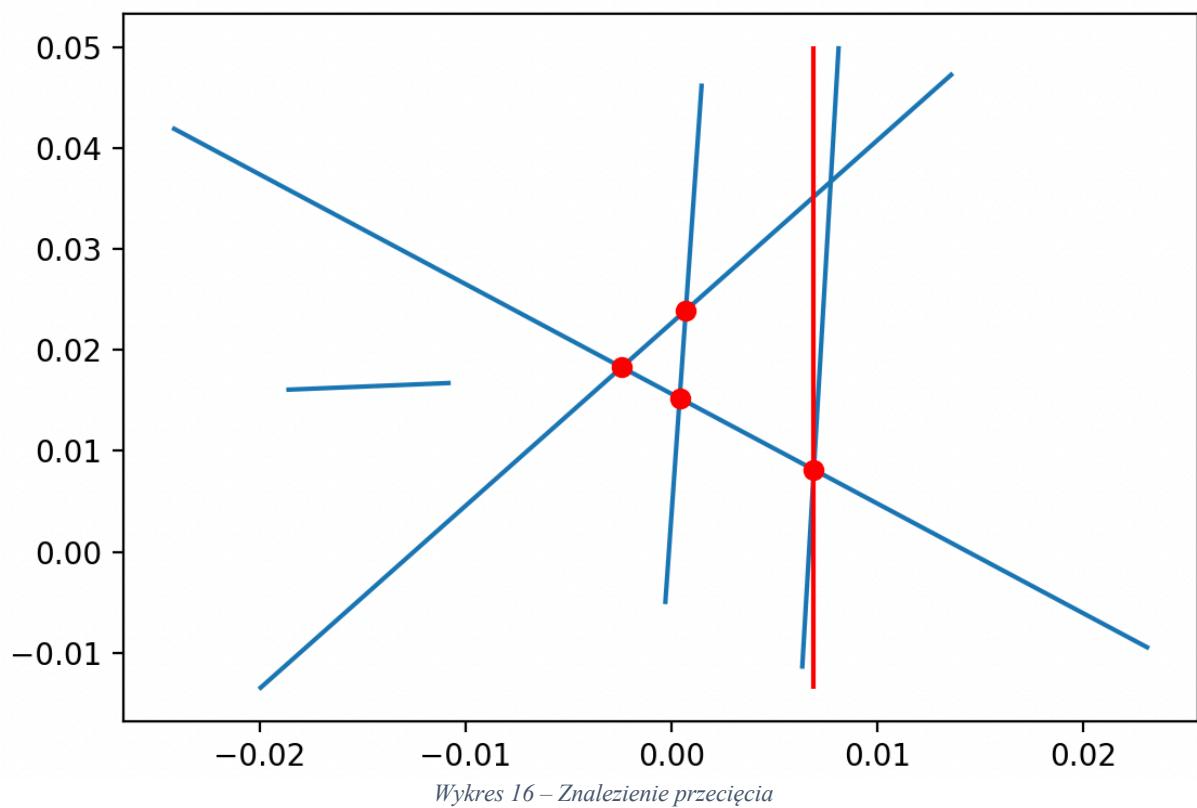


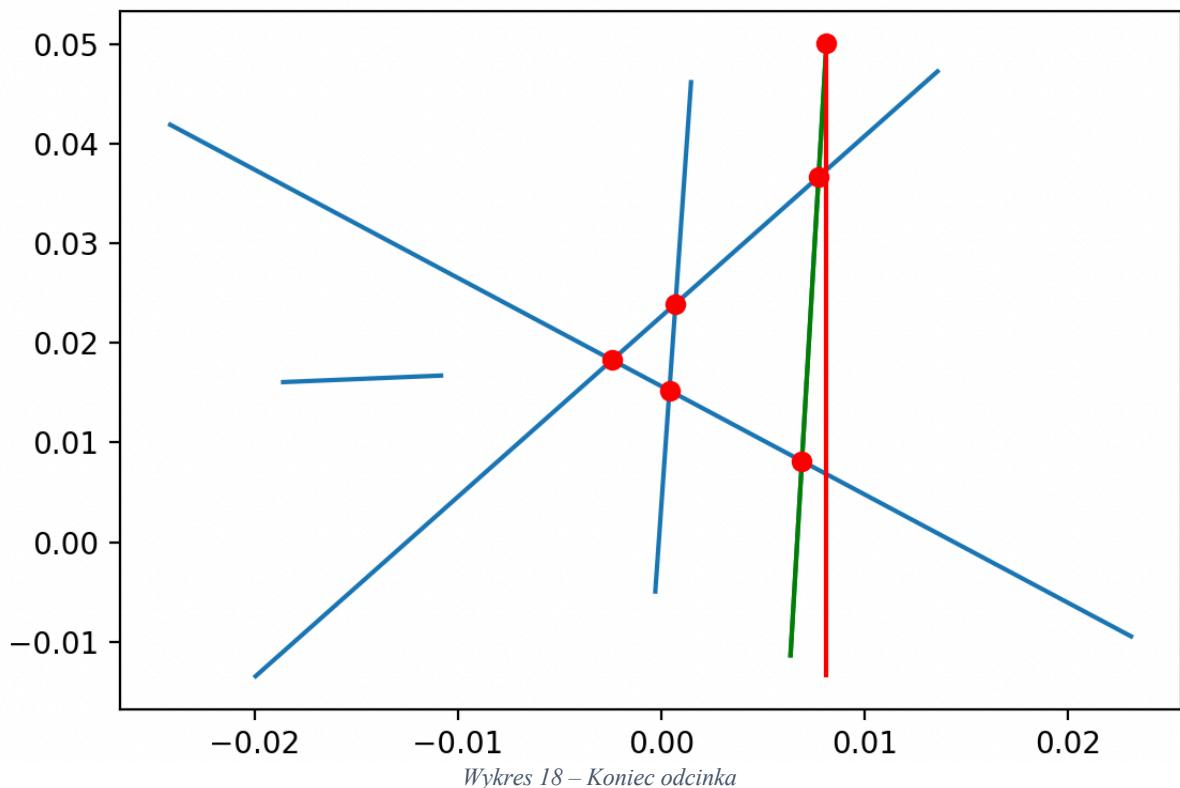


Wykres 14 – Koniec odcinka

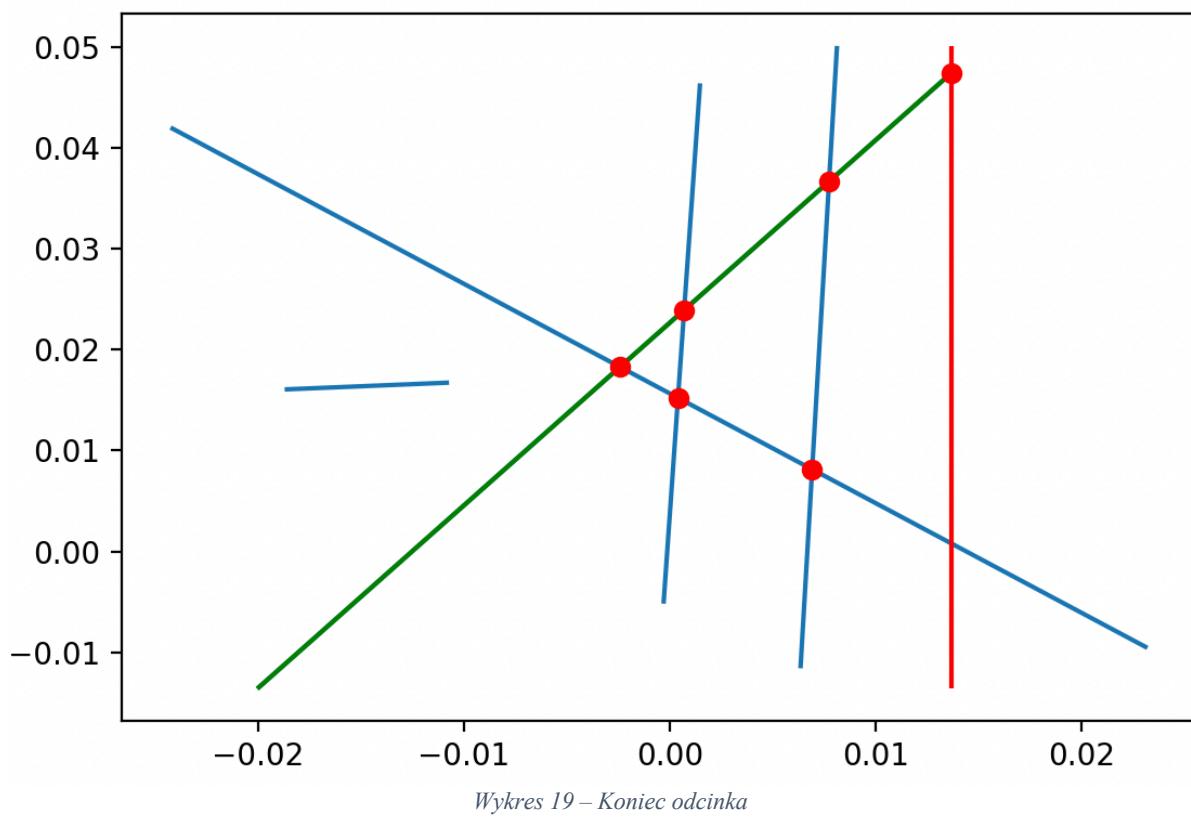


Wykres 15 – Początek odcinka

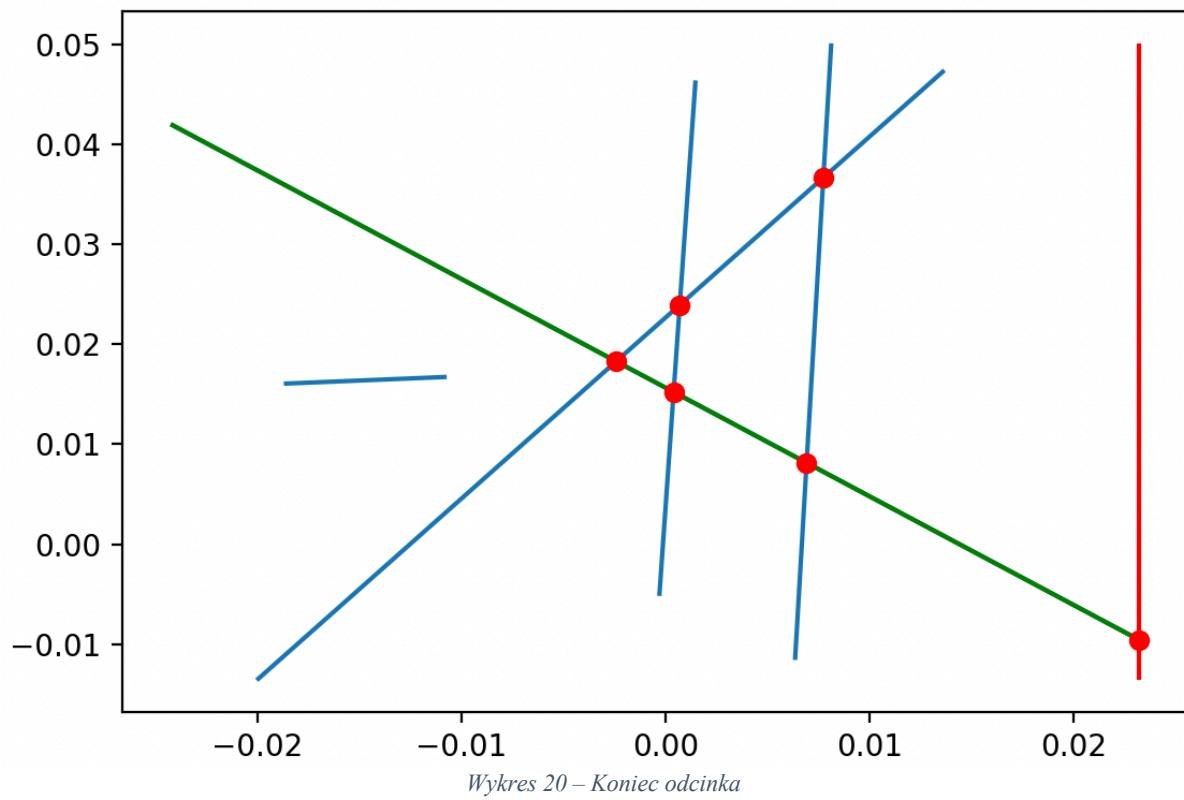




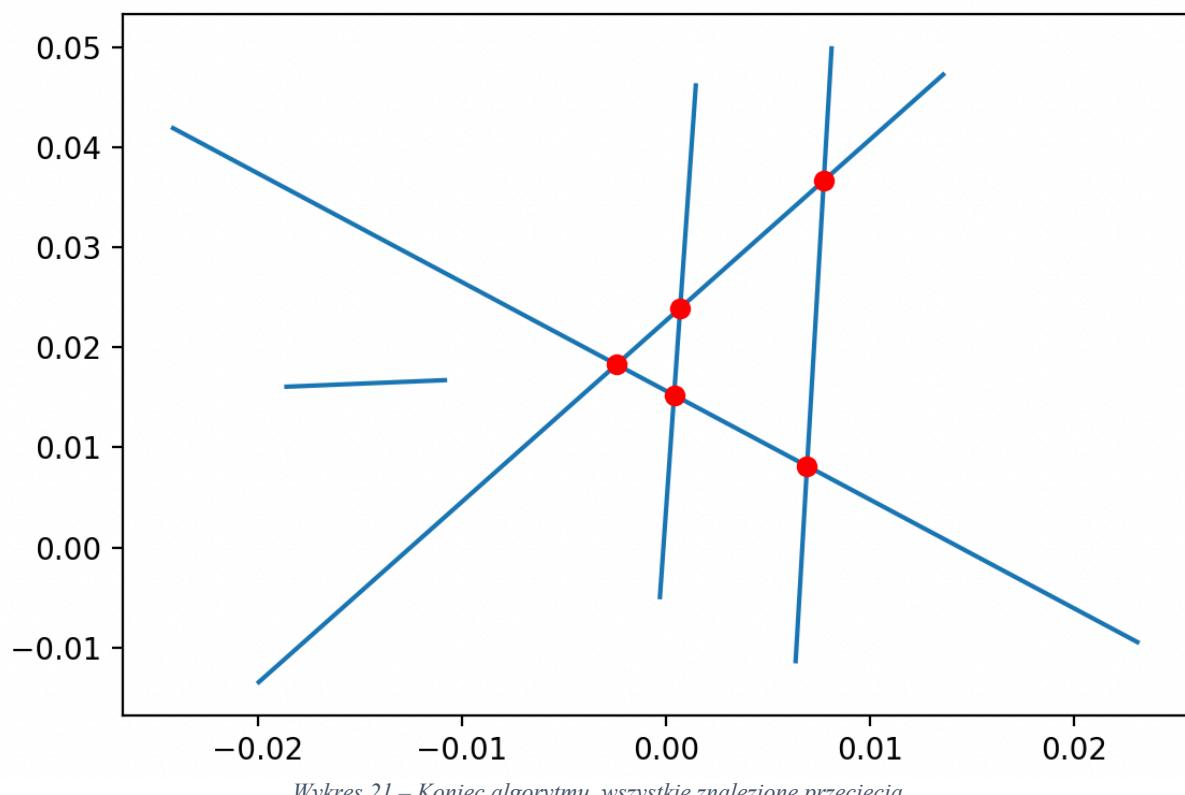
Wykres 18 – Koniec odcinka



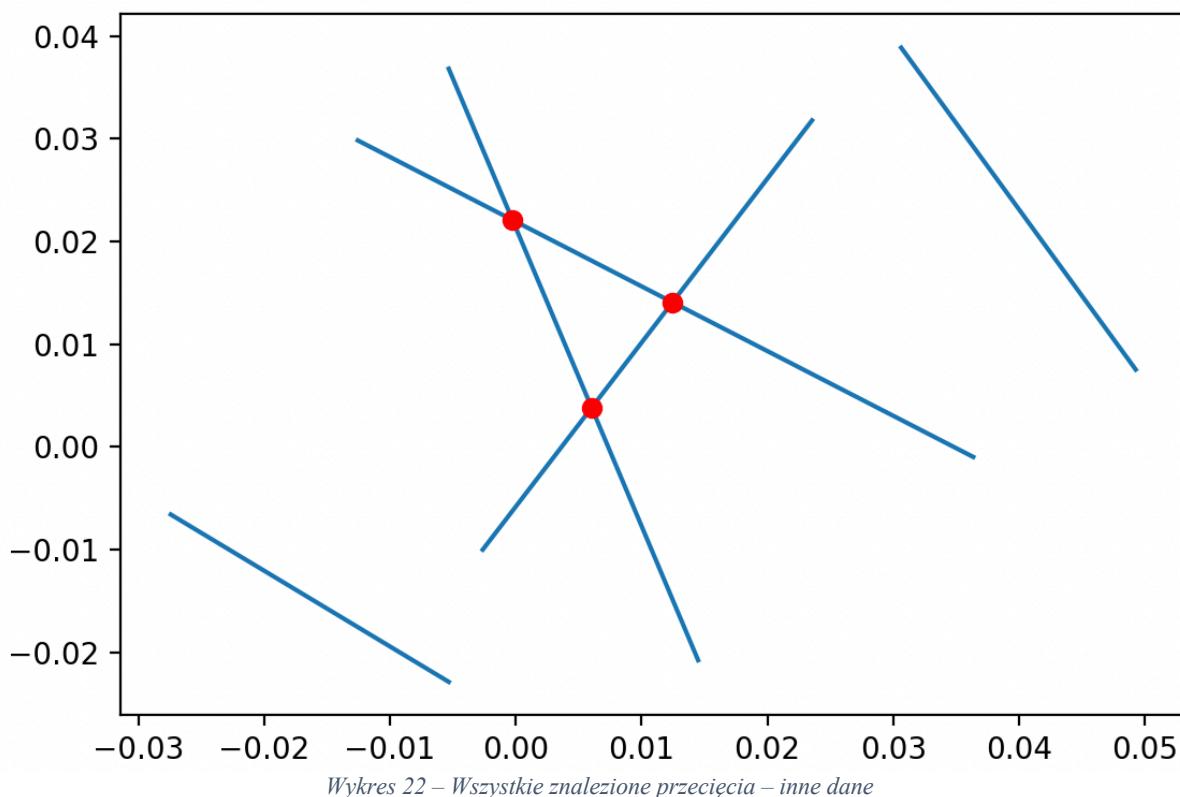
Wykres 19 – Koniec odcinka



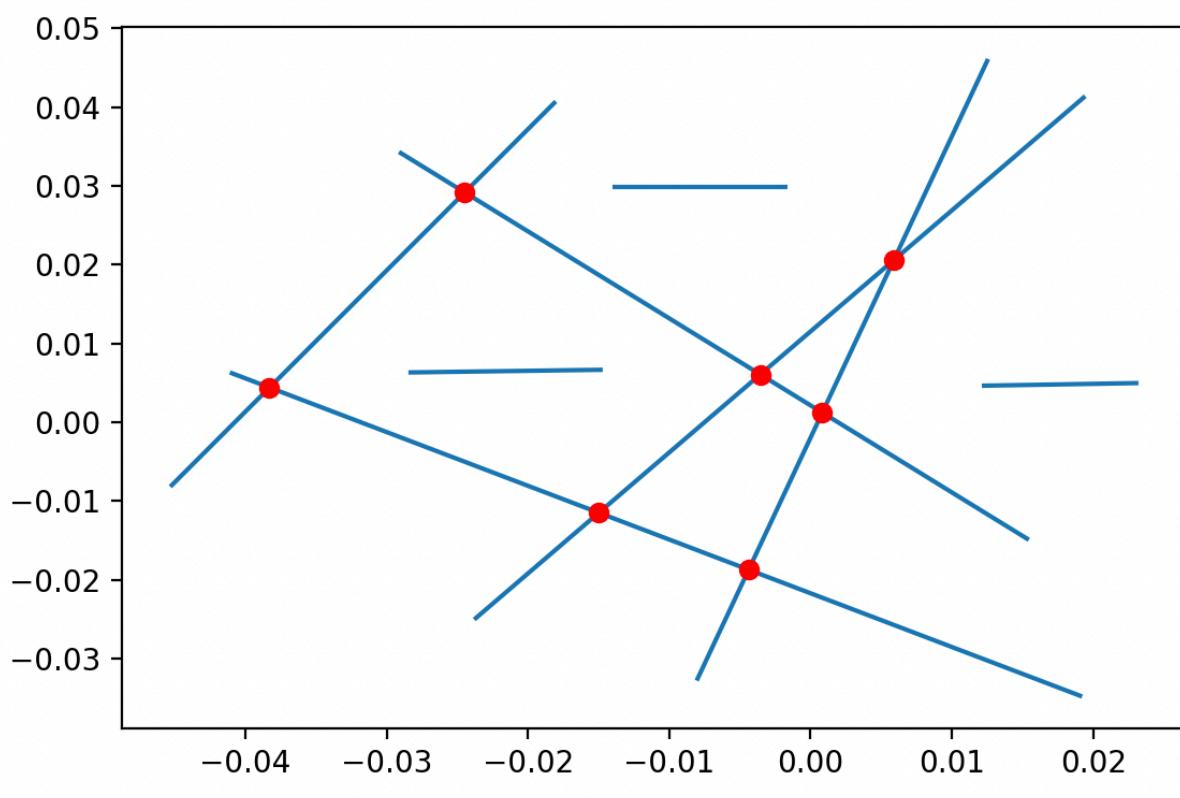
Wykres 20 – Koniec odcinka



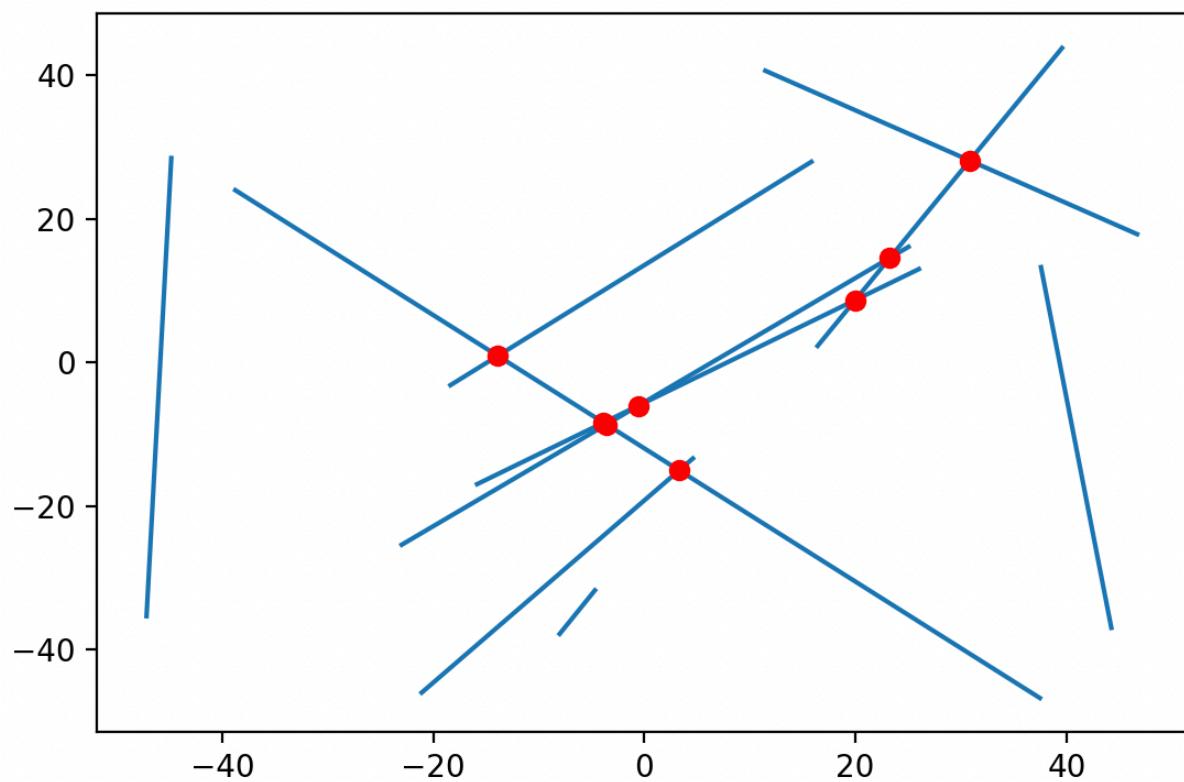
Wykres 21 – Koniec algorytmu, wszystkie znalezione przecięcia



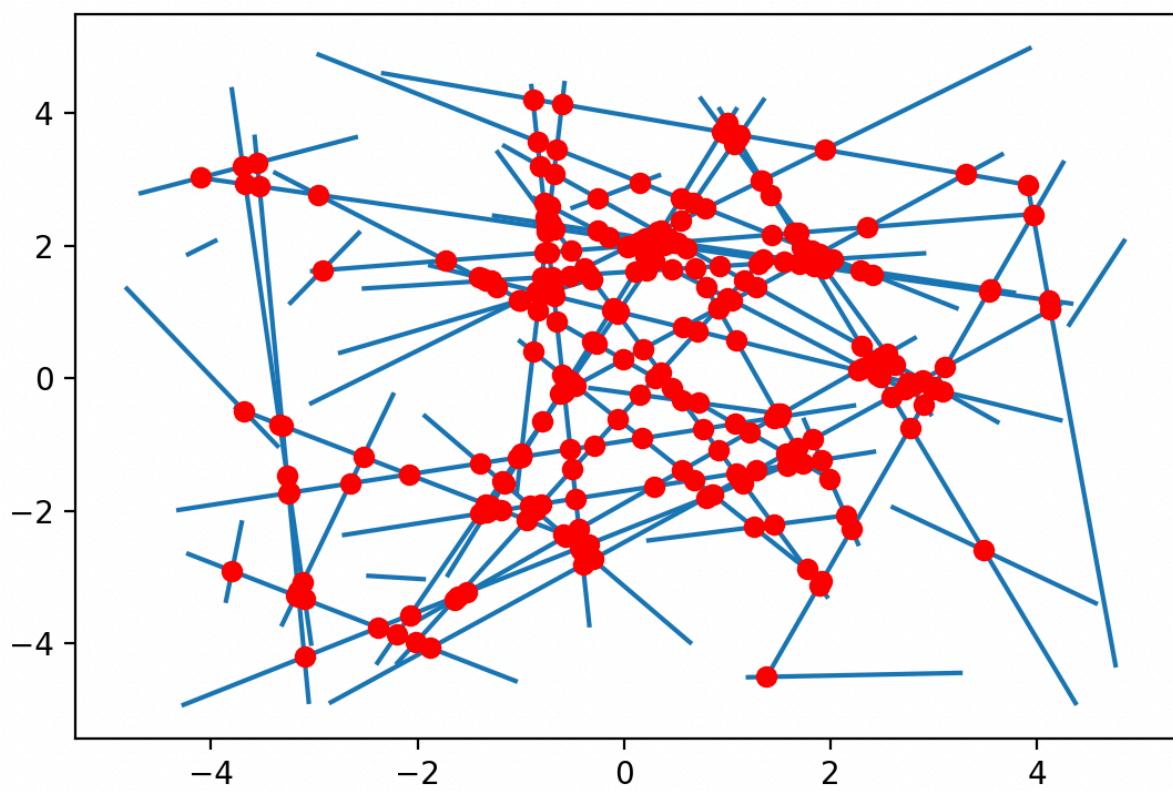
Wykres 22 – Wszystkie znalezione przecięcia – inne dane



Wykres 23 – Wszystkie znalezione przecięcia – inne dane



Wykres 24 – Wszystkie znalezione przecięcia – inne dane



Wykres 25 – Wszystkie znalezione przecięcia – inne dane

Przykładowe dane przecięć (przechowywane w dataframe, do wyświetlenia użyto funkcji head()):

	Point	Line1	Line2
0	(-2.380724916143316, -3.7606350027277626)	((-4.285964863642084, -4.93761602784156), (1.7...	((-4.2401450416407, -2.635958228823637), (-1.0...
1	(-4.10642379010593, 3.0293453816140827)	((-4.70205901180014, 2.7881523961070487), (-2....	((-4.1386450743723024, 3.0365938620685444), (4...
2	(-3.7968908185322965, -2.904062097782024)	((-3.8590156566116898, -3.3883633278410796), (...	((-4.2401450416407, -2.635958228823637), (-1.0...
3	(-3.691431652550023, 3.1973898305879906)	((-4.70205901180014, 2.7881523961070487), (-2....	((-3.801648425637402, 4.397445553404818), (-3....
4	(-2.0856591447371957, -1.4477018384069127)	((-4.337031657631211, -1.9899800612392005), (2...	((-3.7800609864742887, -0.42948166352527384), ...

Wszystkie informacje do wykresów znajdują się w samym programie.

4. Wnioski

Na podstawie powyższych wykresów można stwierdzić, że zaimplementowano poprawnie oba algorytmy, po dokładnej analizie nie znaleziono żadnego błędu lub braku jakiegoś przecięcia. Pierwszy algorytm wykorzystuje sortedset do stanu miotły, aby był szybki dostęp do danych, $\log(n)$, struktura zdarzeń tego nie potrzebowała, ponieważ żadnych przecięć nie musiała przechowywać. Natomiast dodatkowo do drugiego algorytmu również użyto sortedset'a do struktury zdarzeń. Pozwoliło to przyspieszyć działanie algorytmów ze względu na złożoność operacji na tych strukturach. Set z informacją przecięć pozwolił na wyeliminowanie ewentualnych duplikatów/zwielokrotnień.