

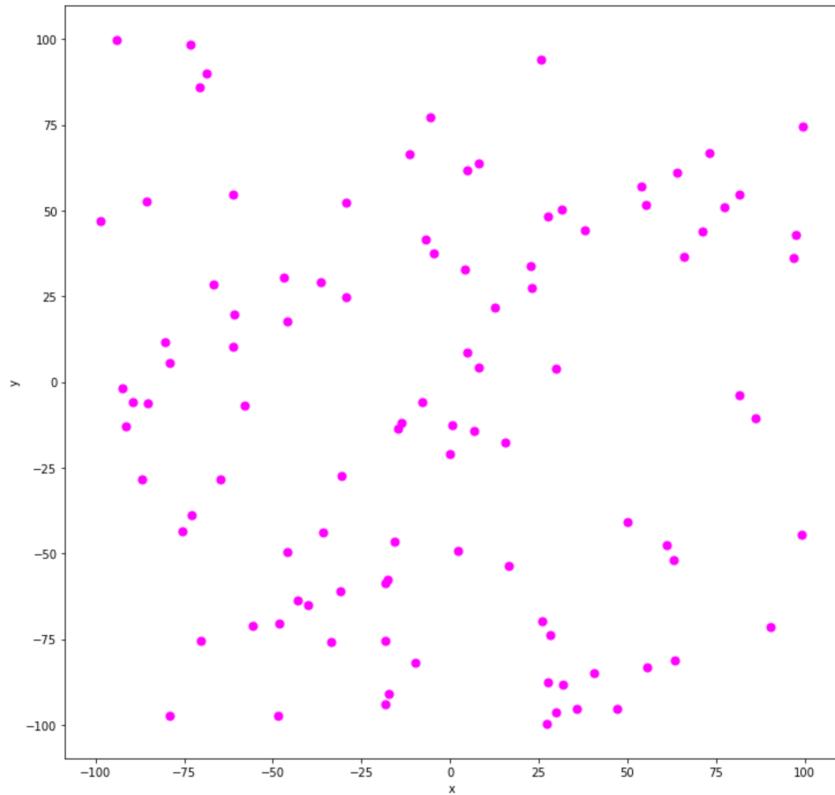
Geometria obliczeniowa

Otoczka wypukła

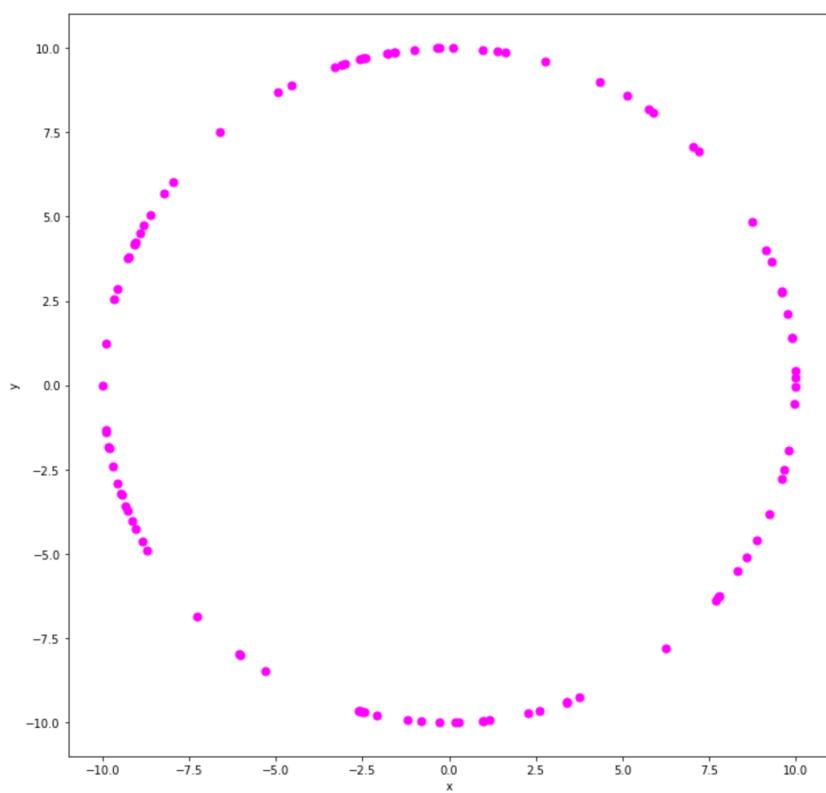
Oscar Teeninga

1. Zbiory

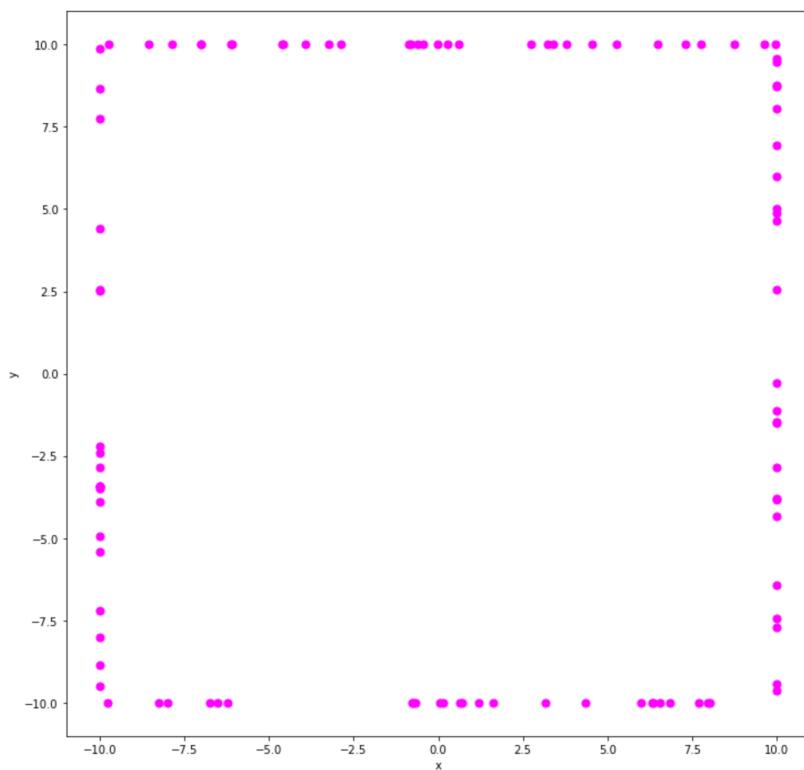
A. Zbiór zawierający 100 losowo wygenerowanych punktów o współrzędnych $[-100, 100]$



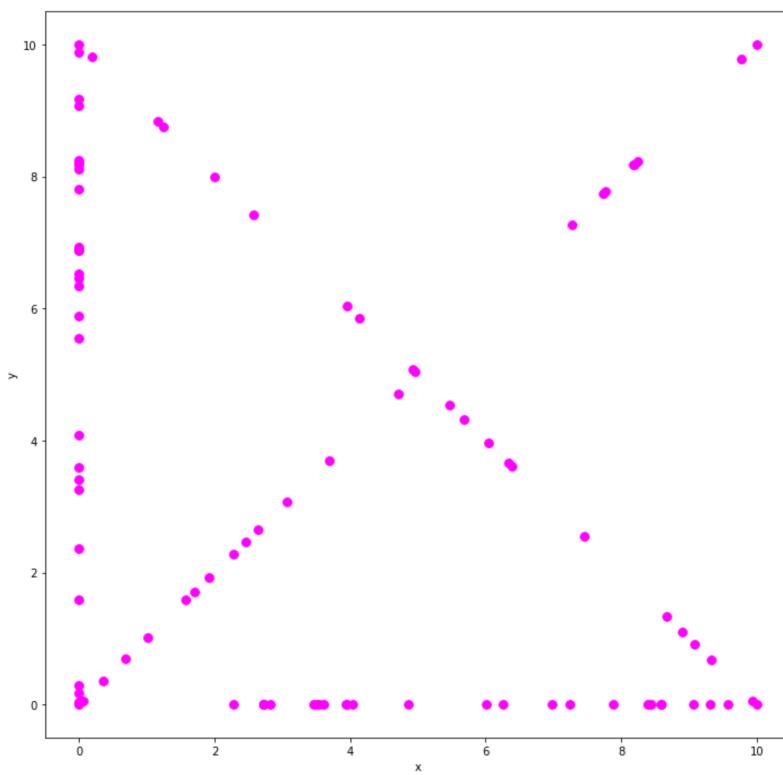
B. Zbiór zawierający 100 losowa wygenerowanych punktów leżących na okręgu o środku $(0, 1)$ i promieniu $R=10$



- C. Zbiór zawierający 100 losowo wygenerowanych punktów leżących na bokach prostokąta o wierzchołkach $(-10,10)$, $(-10,-10)$, $(10,-10)$, $(10,10)$

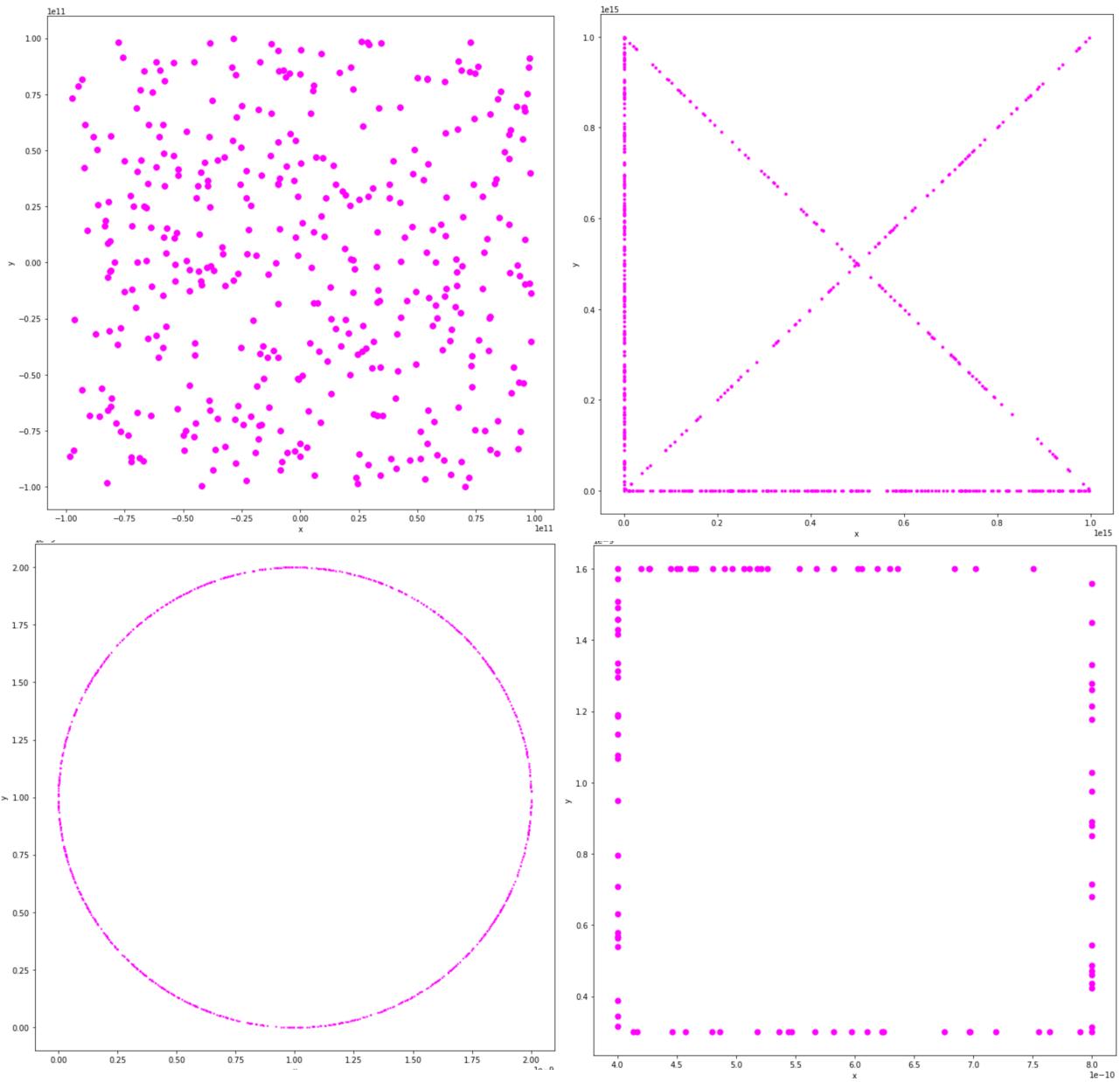


- D. Zbiór zawierający wierzchołki kwadratu $(0,0)$, $(10,0)$, $(10,10)$, $(0,10)$ oraz punkty wygenerowane losowo w sposób: po 25 punktów na dwóch bokach kwadratu leżących na osiach i po 20 punktów na przekątnych kwadratu



2. Modyfikacje zbiorów

Każdy ze zbiorów jest konfigurowalny w pewnym stopniu. Wszystkie zbiory o kształcie prostokąta wymagają by boki były równoległe do osi.

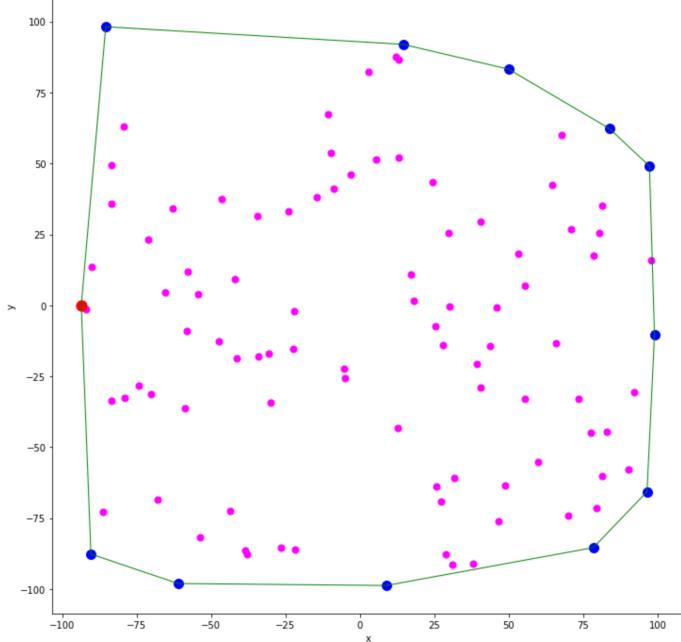


3. Algorytm Grahama

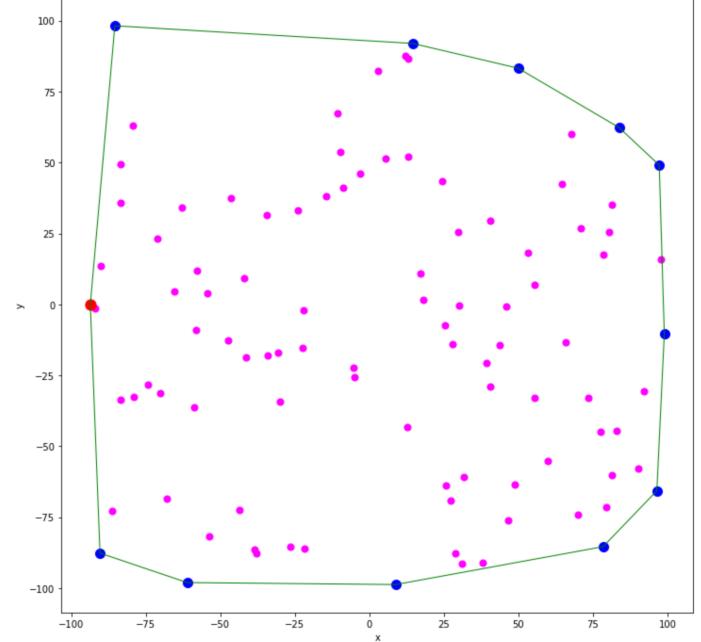
A. Wzorcowe zbiory

Zbiór A nie posiadał punktów wspólniowych, więc wzięcie zerowego epsilonu dawało dobre efekty.

Eps: 0
Liczba punktów w otoczce: 12

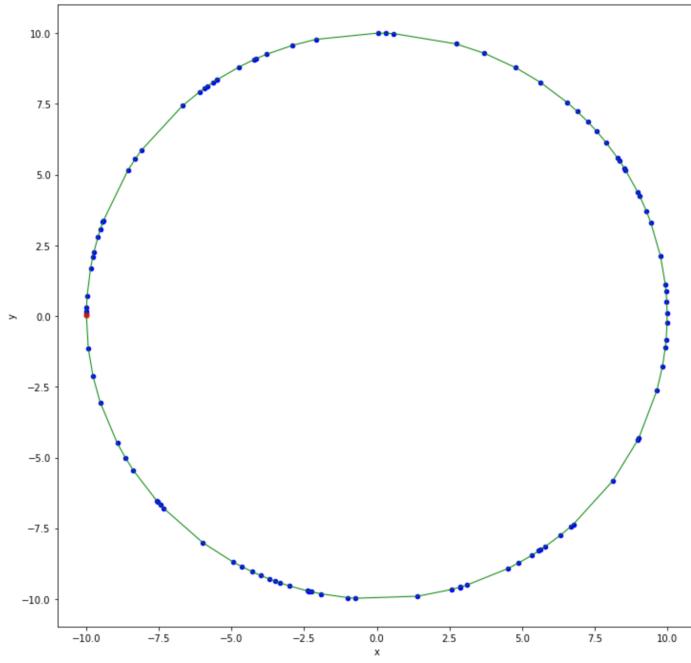


Eps: 0.001
Liczba punktów w otoczce: 12

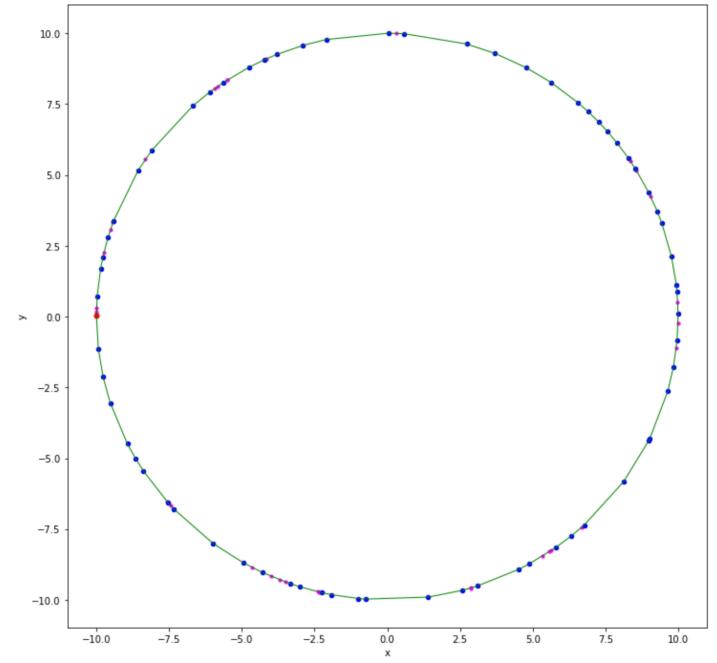


Zbiór B nie posiadał również punktów wspólniowych, a więc wzięcie zerowego epsilonu wskazywało wszystkie punkty. Dopiero 0.01 pozwalało uznać niektóre punkty za wspólniowe.

Eps: 0
Liczba punktów w otoczce: 100

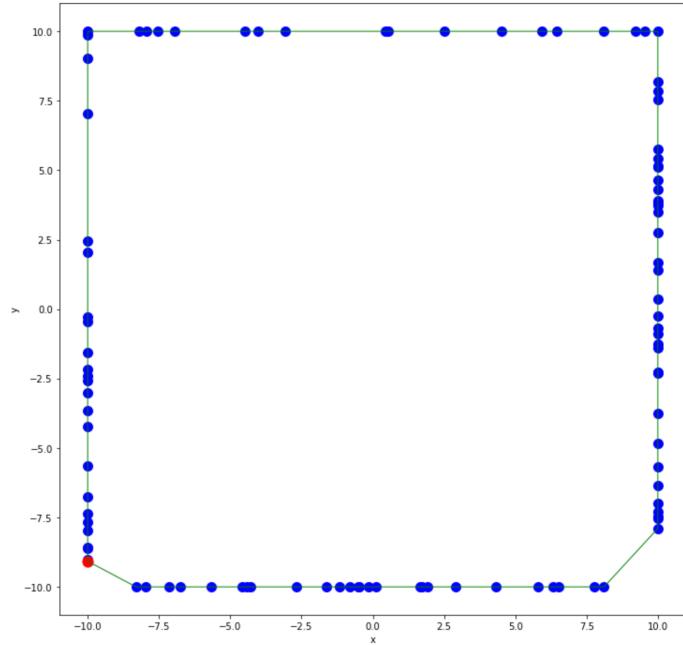


Eps: 0.01
Liczba punktów w otoczce: 67

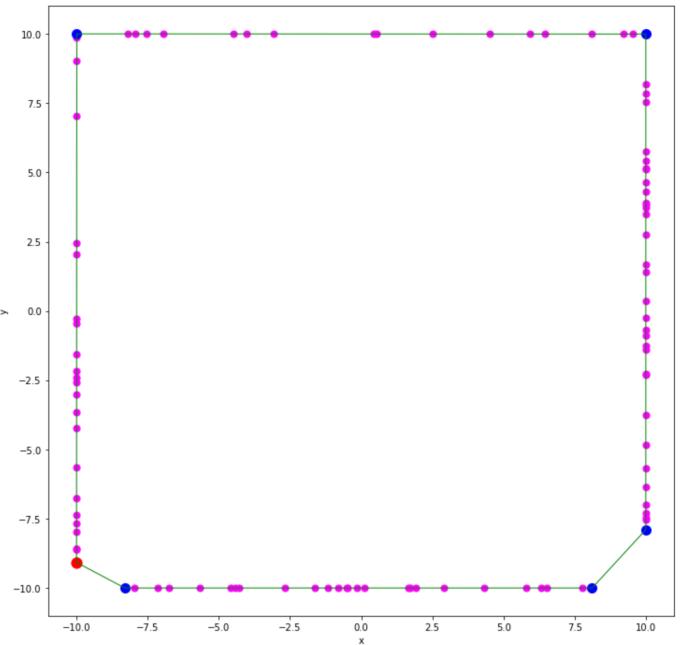


Zbiór C posiadał właściwie same punkty wspólniowe, dlatego dla zerowego epsilon otrzymaliśmy wszystkie punkty. Zwiększenie epsilon pozwoliło ograniczyć się do niezbędnych punktów.

Eps: 0
Liczba punktów w otoczce: 100

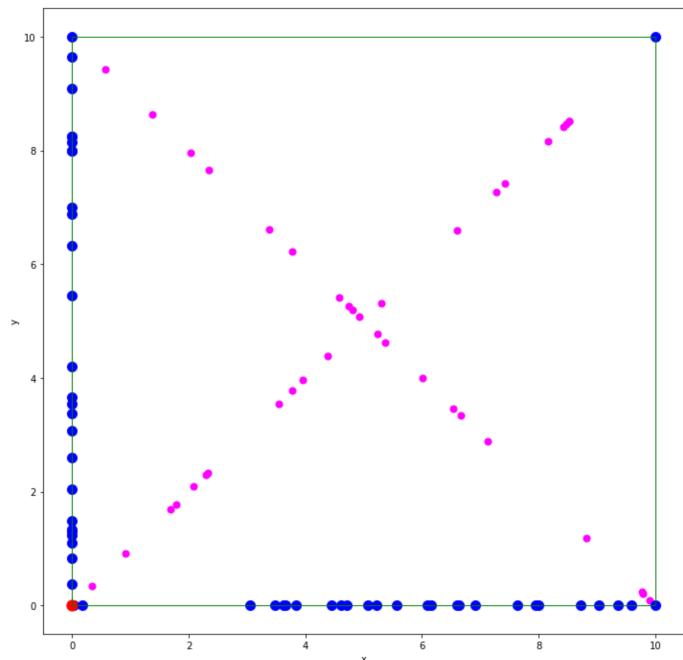


Eps: 0.1
Liczba punktów w otoczce: 6

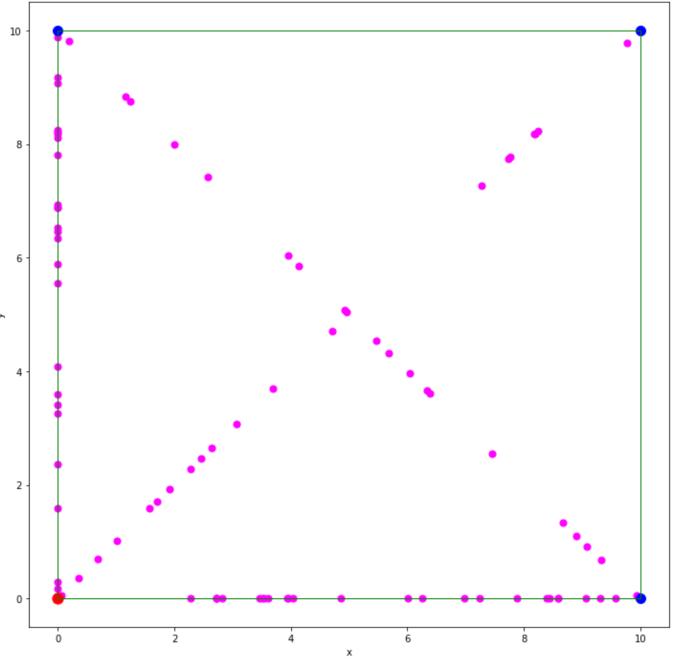


Zbiór D posiadał ten sam problem co zbiór C, zwiększenie epsilon również pomogło.

Eps: 0
Liczba punktów w otoczce: 54



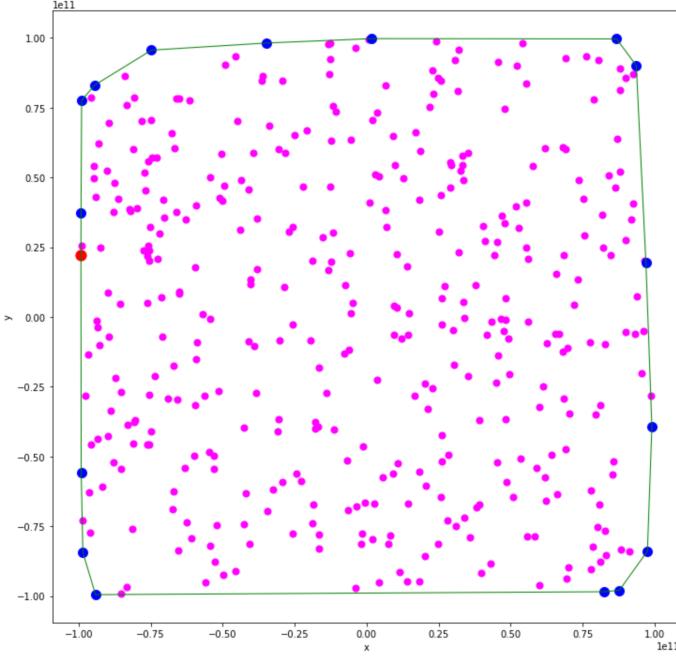
Eps: 1e-09
Liczba punktów w otoczce: 4



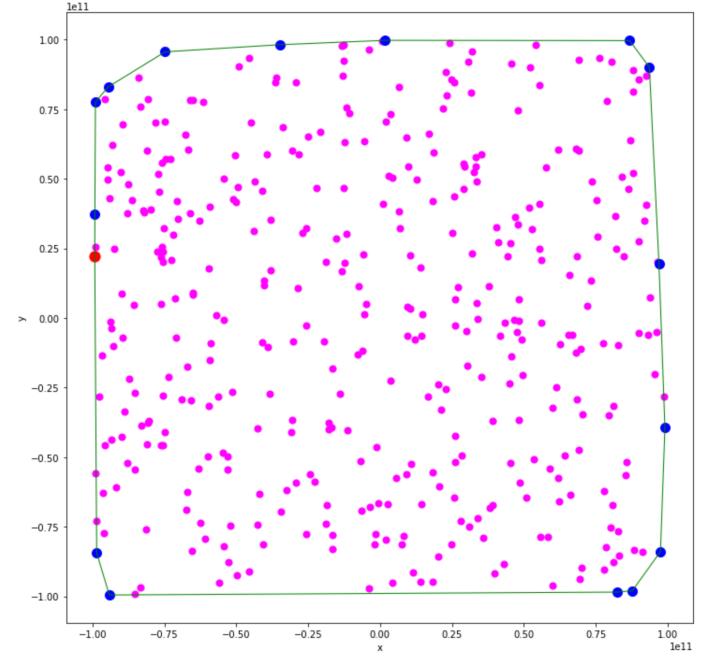
B. Alternatywne zbiory

Zbiór A nie posiadał punktów wspólniowych, więc wzięcie zerowego epsilonu dawało dobre efekty.

Eps: 0
Liczba punktów w otoczce: 17

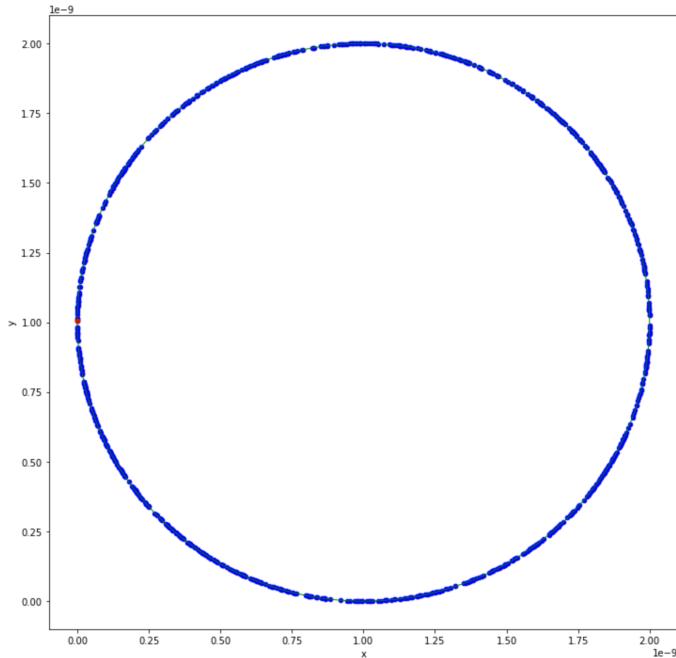


Eps: 100000000.0
Liczba punktów w otoczce: 16

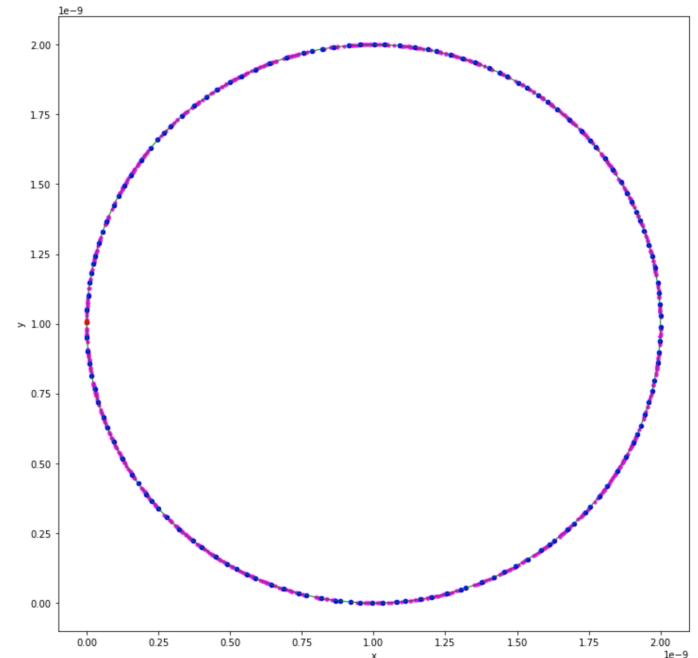


Zbiór B nie posiadał również punktów wspólniowych, a więc wzięcie zerowego epsilonu wskazywało wszystkie punkty.

Eps: 0
Liczba punktów w otoczce: 1000

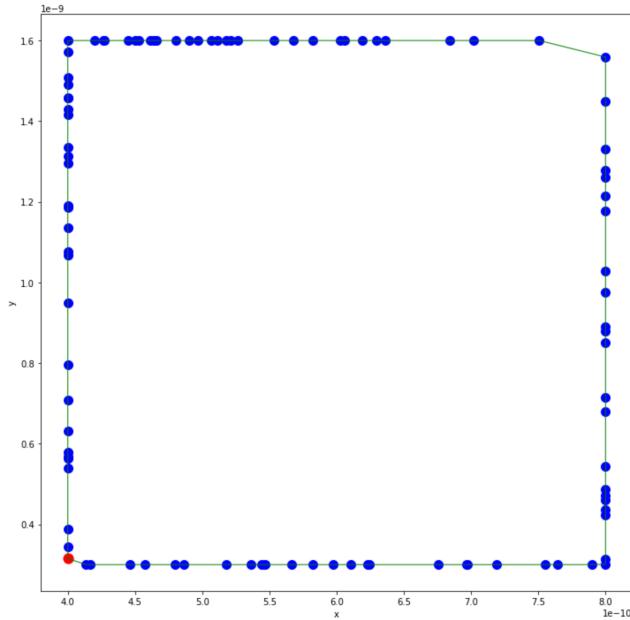


Eps: 1e-23
Liczba punktów w otoczce: 139

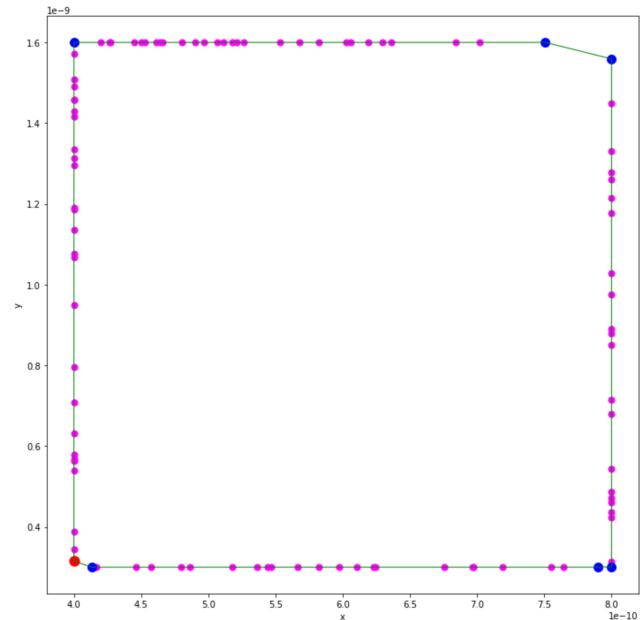


Zbiór C posiadał właściwie same punkty współliniowe, dlatego dla zerowego epsilon otrzymaliśmy wszystkie punkty. Zwiększenie epsilon pozwoliło ograniczyć się do niezbędnych punktów.

Eps: 0
Liczba punktów w otoczce: 100

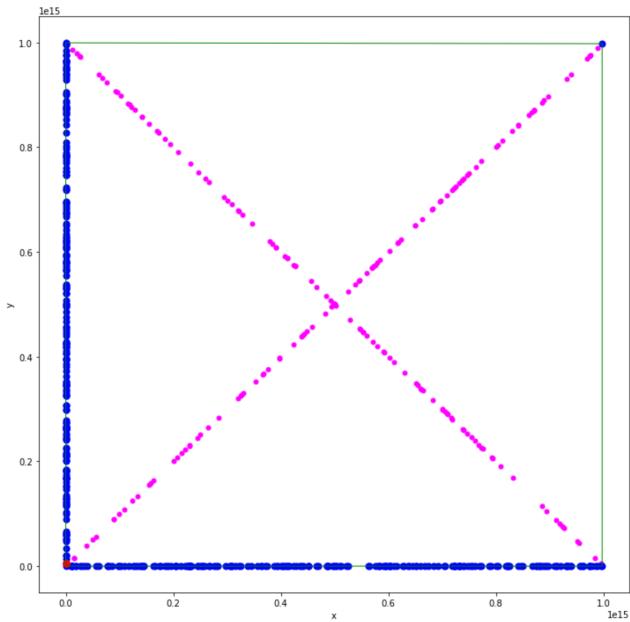


Eps: 1e-23
Liczba punktów w otoczce: 7

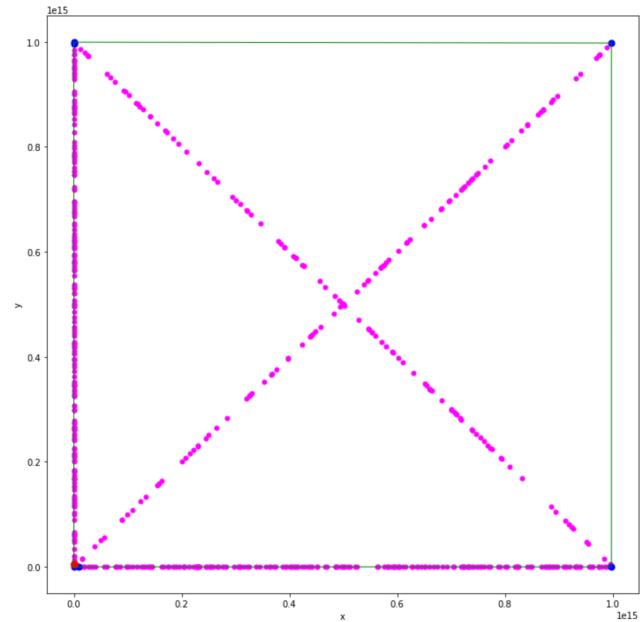


Zbiór D posiadał ten sam problem co zbiór C, zwiększenie epsilon również pomogło.

Eps: 0
Liczba punktów w otoczce: 403



Eps: 1e-09
Liczba punktów w otoczce: 7

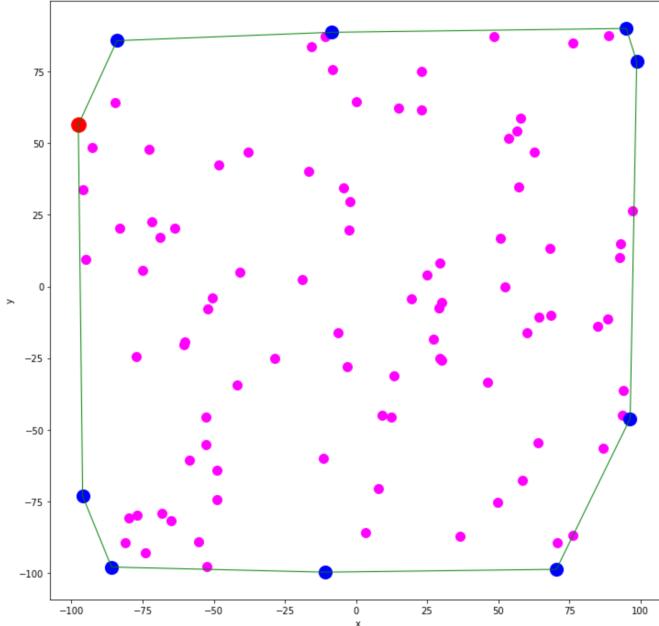


4. Algorytm Jarvisa

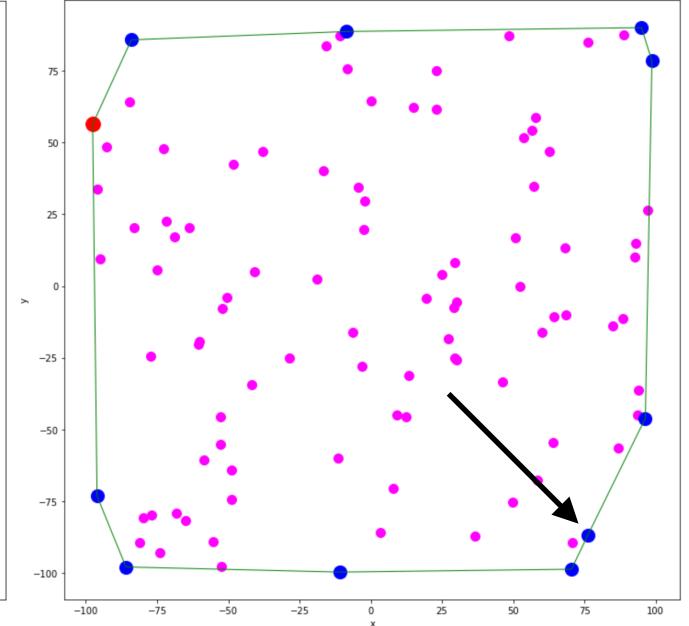
A. Zbiory wzorcowe

Zbiór A nie stanowił dla algorytmu wyzwania, co ciekawe, dopiero oszacowanie epsilon na 10 pozwoliło osiągnąć nadmiarowy punkt.

Eps: 0
Liczba punktów w otoczeniu: 10

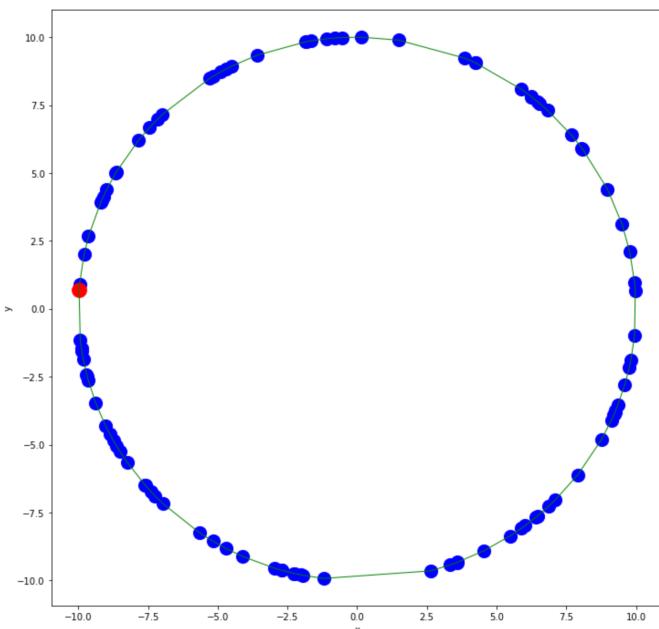


Eps: 10
Liczba punktów w otoczeniu: 11

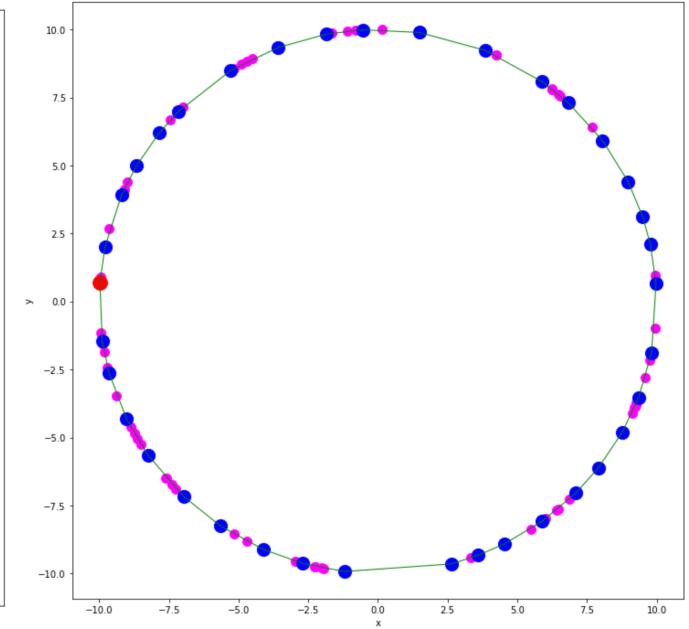


Zbiór B również pozytywnie reaguje na algorytm Jarvisa. Nawet duży epsilon nie powoduje błędów, a najwyżej niedokładności.

Eps: 0
Liczba punktów w otoczeniu: 100

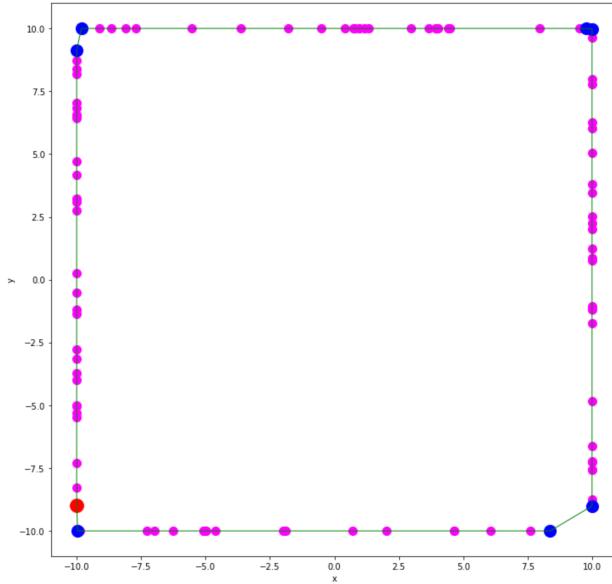


Eps: 0.1
Liczba punktów w otoczeniu: 37

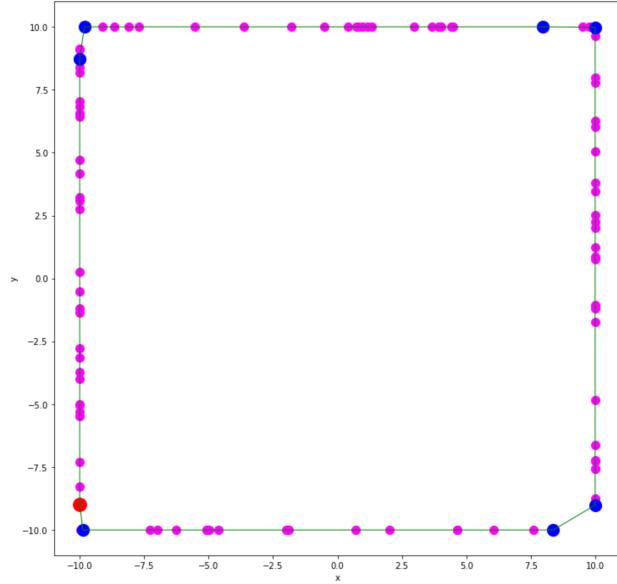


Zbiór C jest najtrudniejszy, a jednak algorytm poradził sobie świetnie, nawet z zerowym epsilon udało się osiągnąć prawidłowy wynik. Zwiększenie epsilon nie zmieniało otoczki, a dla odpowiednio dużego epsilon, algorytm niekończył działania.

Eps: 0
Liczba punktów w otoczce: 8

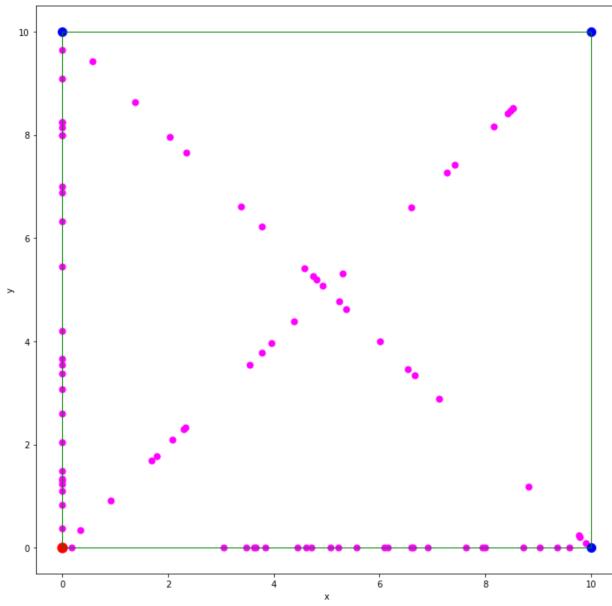


Eps: 0.1
Liczba punktów w otoczce: 8

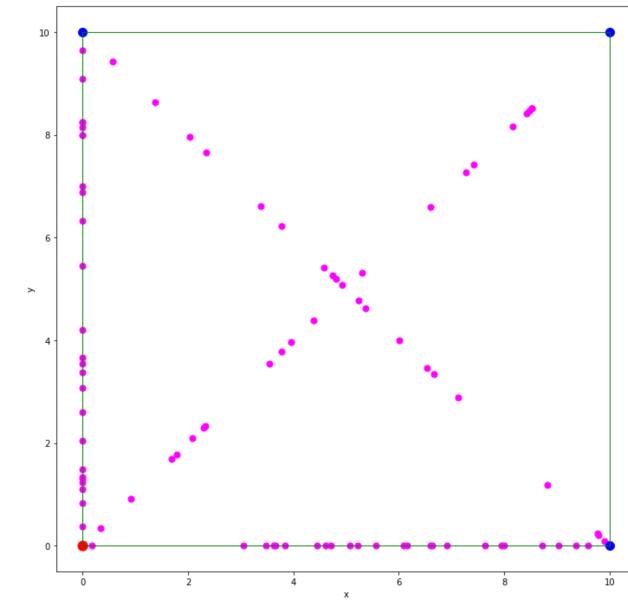


Zbiór D równie newralgiczny jak zbiór C również nie stanowił problemu, tutaj jednak zwiększenie epsilon daje efekty.

Eps: 0
Liczba punktów w otoczce: 4

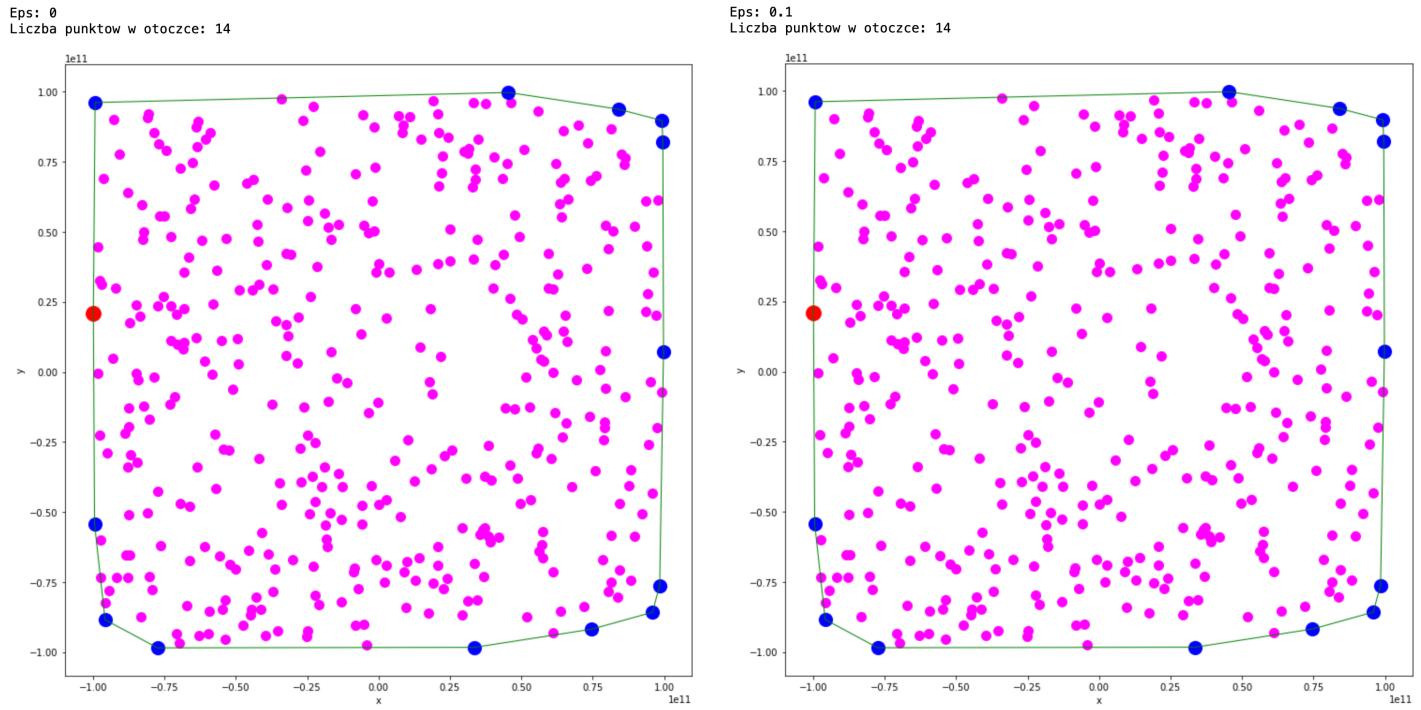


Eps: 0.1
Liczba punktów w otoczce: 4

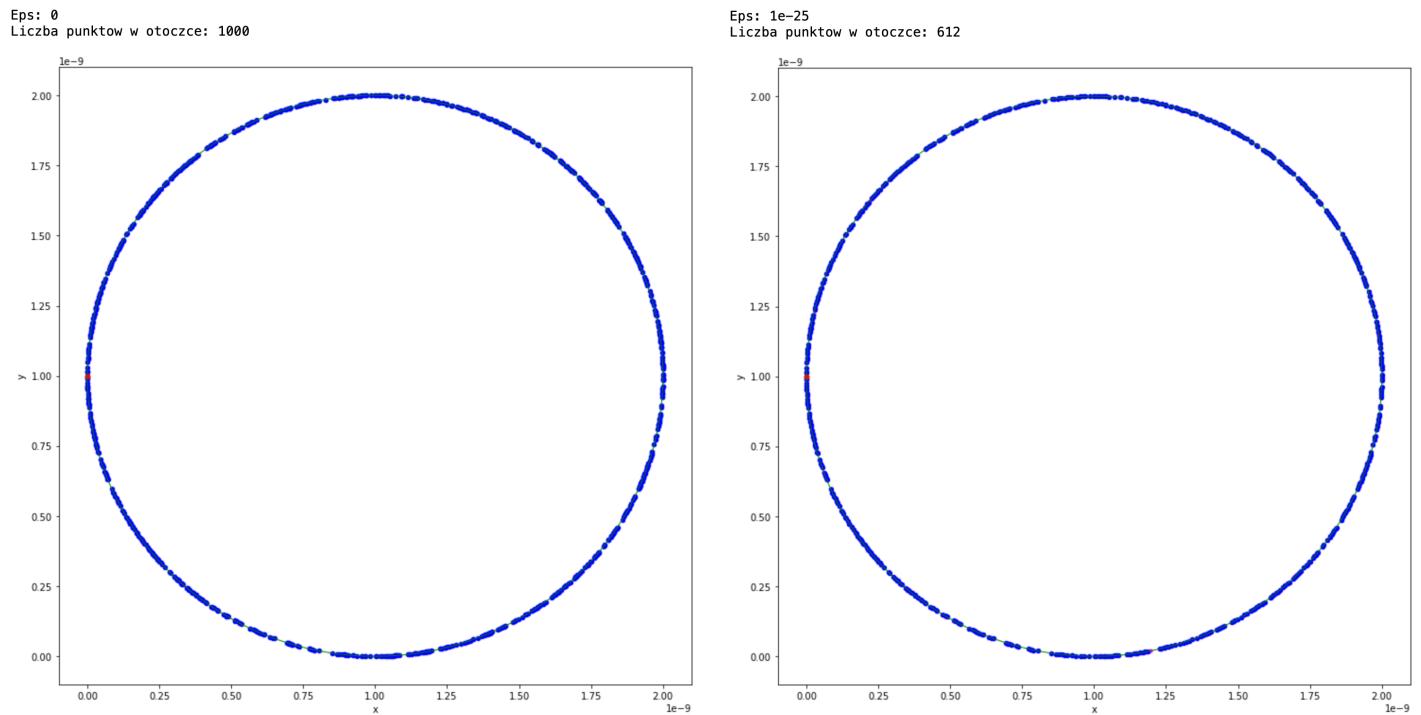


A. Zbiory alternatywne

Zbiór A nie posiadał punktów wspólniowych, więc wzięcie zerowego epsilonu dawał dobre efekty, natomiast zwiększenie epsilonu nie zmieniał otoczki.

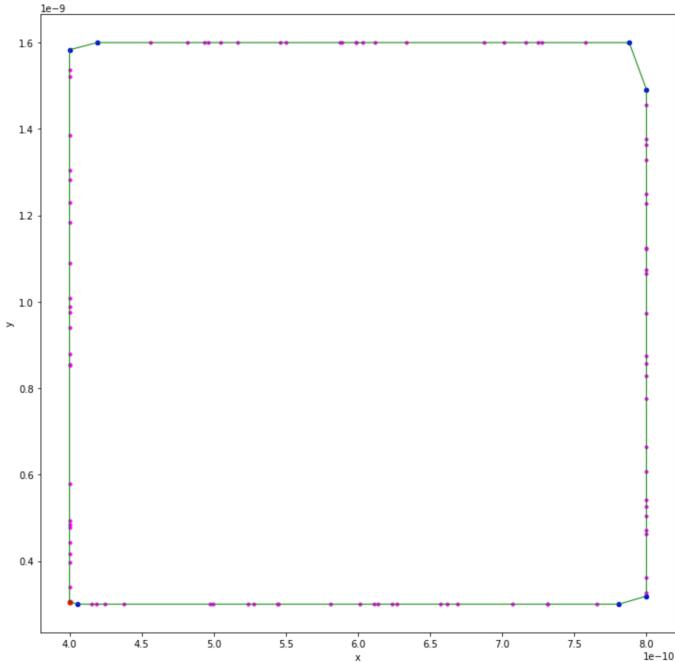


Zbiór B nie posiadał również punktów wspólniowych, a więc wzięcie zerowego epsilonu wskazywało wszystkie punkty. Zwiększenie epsilonu ze względu na kształt okręgu pozwalał ograniczyć liczbę punktów otoczki.

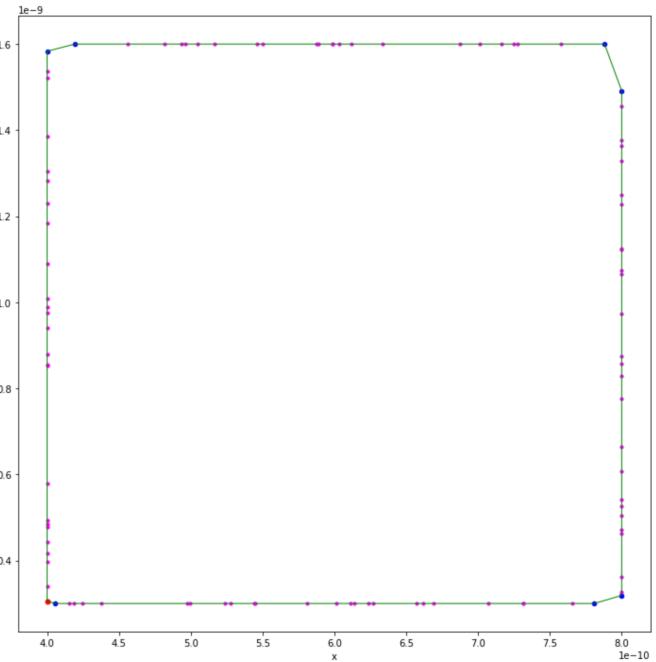


Zbiór C jest najtrudniejszy, a jednak algorytm poradził sobie świetnie, nawet z zerowym epsilon udało się osiągnąć prawidłowy wynik. Zwiększenie epsilon nie zmieniało otoczki, a dla odpowiednio dużego epsilon, algorytm niekończył działania.

Eps: 0
Liczba punktów w otoczce: 8

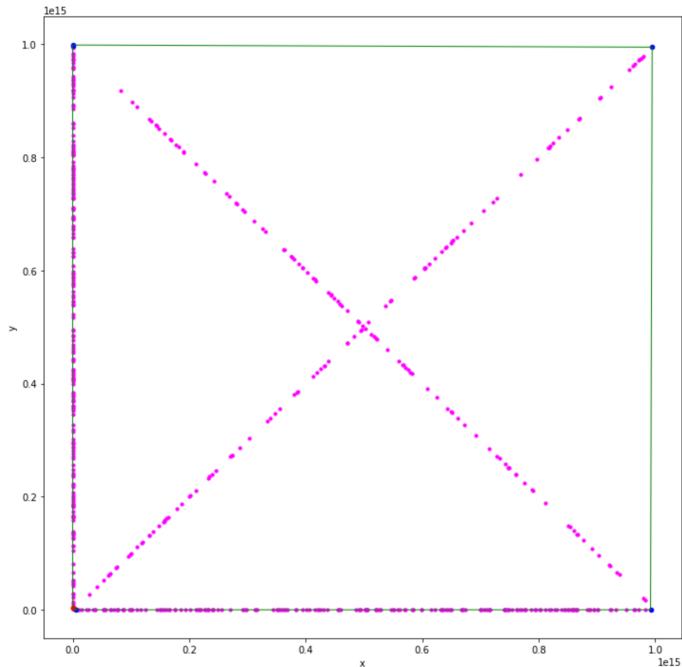


Eps: 1e-23
Liczba punktów w otoczce: 8

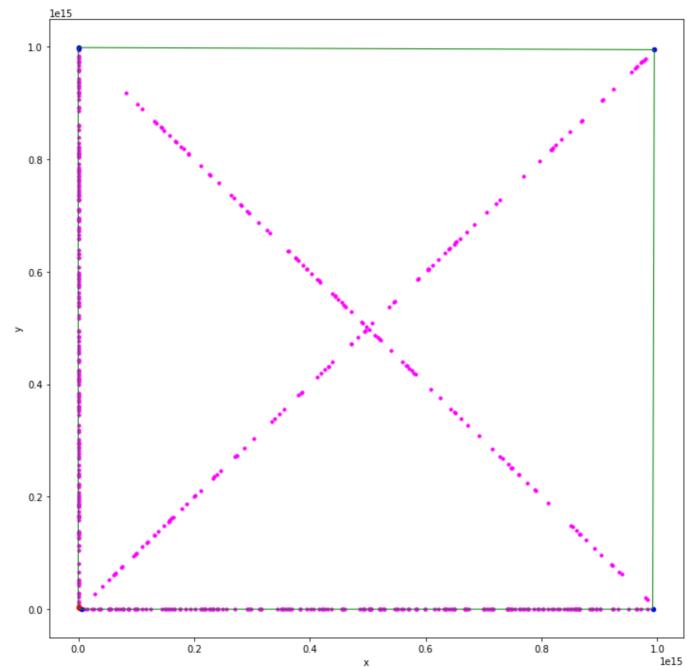


Zbiór D równie newralgiczny jak zbiór C również nie stanowił problemu, tutaj jednak zwiększenie epsilon daje efekty.

Eps: 0
Liczba punktów w otoczce: 7



Eps: 1e-23
Liczba punktów w otoczce: 7



5. Pomiary czasu

A. Algorytm Grahama

Ten algorytm ma teoretyczną złożoność $O(n \log n)$. Oznacza to, że złożoność nie zależy od liczby punktów otoczki, a jedynie od ilości punktów w zbiorze.

	100	200	300	400	500	600	700	800	900	1000
A	0.000528	0.001042	0.001341	0.001721	0.002224	0.002292	0.002507	0.003141	0.005513	0.004650
B	0.000348	0.000730	0.001284	0.001512	0.001972	0.002235	0.002556	0.003563	0.003606	0.004345
C	0.000446	0.000887	0.001338	0.001809	0.002090	0.002203	0.002603	0.004191	0.004308	0.004897
D	0.000493	0.000966	0.001424	0.001968	0.002189	0.002418	0.002838	0.005401	0.004370	0.005046

Zgodnie z wynikami widać, że najważniejszym parametrem jest liczba punktów, a nie rozłożenie punktów w zbiorze.

B. Algorytm Jarvisa

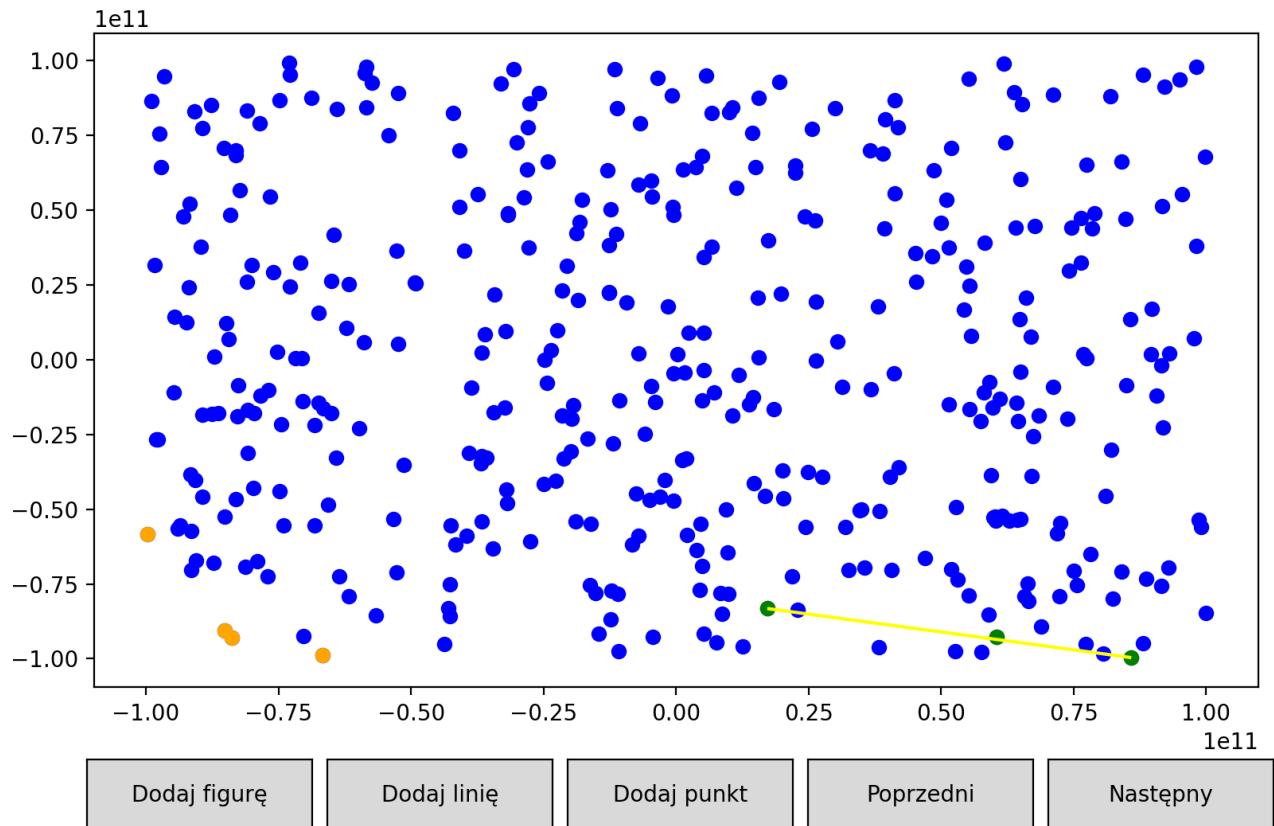
Tym razem mamy algorytm o złożoności $O(n^2)$, jednakże tak naprawdę złożoność jest równa $O(kn)$, gdzie k jest liczbą punktów należących do otoczki, a więc mając zbiór A, C, D będziemy mieli realnie złożoność $O(<10n)$. Kwadratową złożoność osiągnie zbiór B, ponieważ tam wszystkie punkty należące będą do otoczki.

	100	200	300	400	500	600	700	800	900	1000
A	0.002037	0.003531	0.008830	0.009005	0.007710	0.016307	0.022461	0.032883	0.022788	0.035917
B	0.015457	0.058557	0.132646	0.232888	0.348658	0.590090	0.966733	1.045250	1.094336	1.367838
C	0.001434	0.002741	0.004473	0.005485	0.006931	0.011902	0.013362	0.010648	0.012524	0.013581
D	0.001435	0.002749	0.004339	0.005696	0.006821	0.010258	0.012346	0.010507	0.010838	0.013555

Jak można było się spodziewać, w przypadku zbioru B osiągnęliśmy znacznie gorsze czasy, ponieważ punktów otoczki było najwięcej, a więc realna złożoność była kwadratowa.

6. Sceny

Korzystając z implementacji rysowania scen z `geometria.ipynb` stworzyłem alternatywną wersję algorytmu zwracającego listę scen do reprezentowania korków algorytmu. Pomarańczowe punkty oznaczają aktualny zbiór otoczki, zielonym punkty aktualnie analizowane, natomiast żółta linia to linia podejrzewana o bycie otoczką i ewentualnie eliminowanie punktów wspólniowych między nimi.



7. Zapisywanie do pliku

Po raz kolejny korzystam z gotowej implementacji dostarczonej w `geometria.ipynb`, a mianowicie `plot.toJson()`.