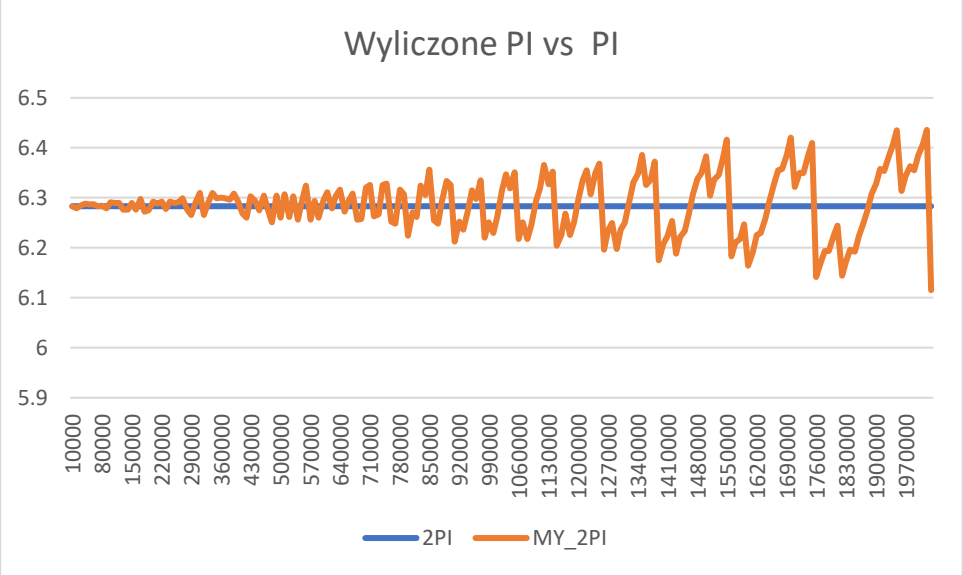
Programowanie Liniowe
Projekt 1
Michał Safuryn 288574

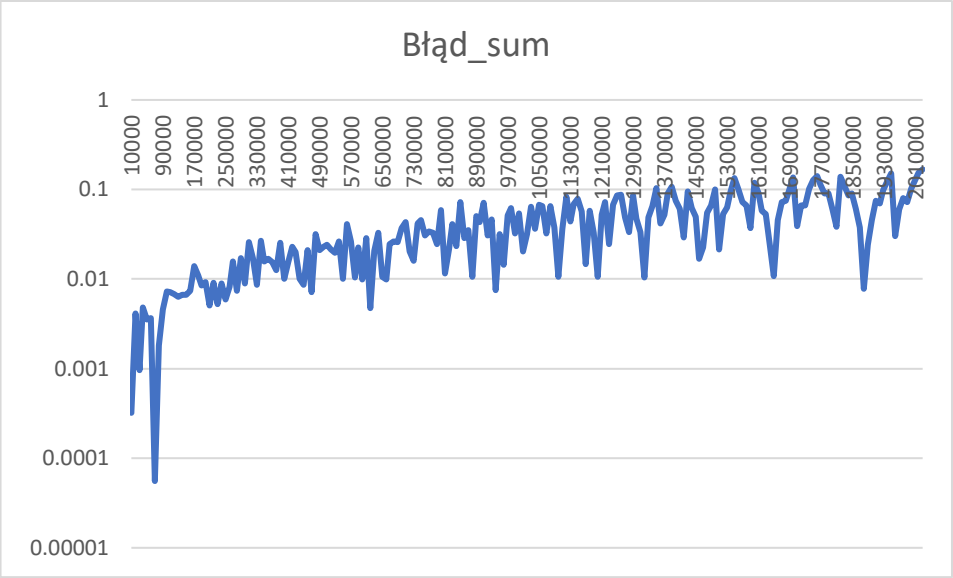
H1. Zwiększając n można uzyskać obwód dowolnie bliski liczbie 2Π.

Nie, zwiększając N nie będziemy zbliżać się do dowolnie bliskiej wartości 2Π
Przykład danych:

N	2PI	MY_2PI
10000	6,28318531	6,28350401
20000	6,28318531	6,27902412
30000	6,28318531	6,28414917
40000	6,28318531	6,28800869
50000	6,28318531	6,28675079
60000	6,28318531	6,28681755
70000	6,28318531	6,28312969
80000	6,28318531	6,28498507
90000	6,28318531	6,27860165
100000	6,28318531	6,29050064



Powyższy wykres przedstawia jak zmienia się 2Π wyliczone z sumy długości wektorów. Można zauważyć, że od około 360000-wierzchłkowego wielokąta 2Π jest bardzo blisko tej stałej. Jednak dla większych N Pi zaczyna się zmieniać i oscyluje w okolicy 6.1 - 6.5. Są to błędy wynikające z użycia typu danych float a nie double.



Na wykresie powyżej można zobaczyć błędy, różnicę między stałą 2Π, a wyliczaną. I widać ze wraz ze wzrostem N błąd też rośnie.

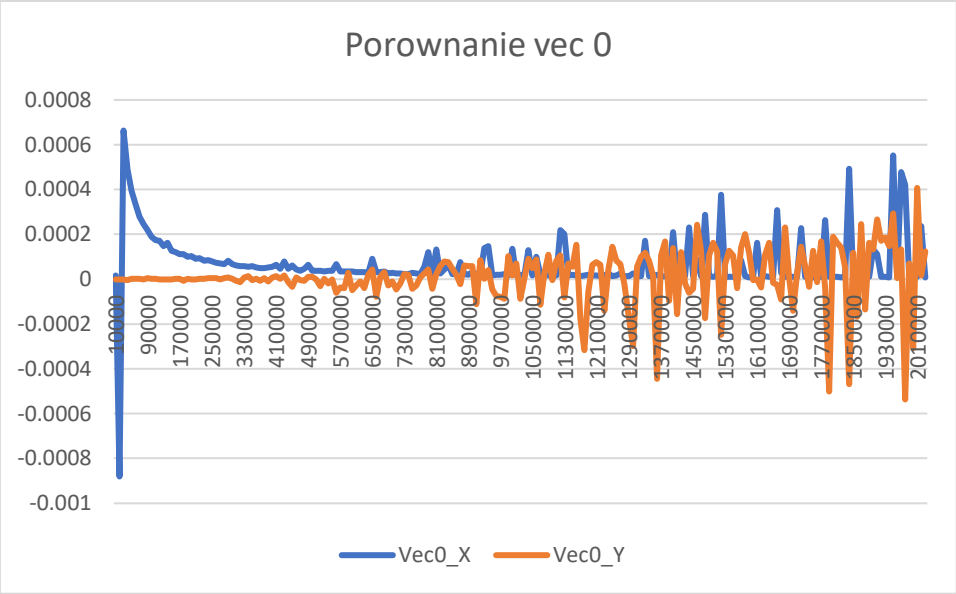
H2. Suma wszystkich wektorów w_i daje dokładnie wektor zerowy.

Nie, nie daje ona dokładnie wektora zerowego. Daje ona natomiast bardzo blisko wektorowi zerowemu

Przykład danych:

N	Vec0_X	Vec0_Y
10000	0,00001644	-0,00000014
20000	-0,00088021	-0,00000193
30000	0,00066332	-0,00000131
40000	0,00048927	-0,00000329
50000	0,00039468	0,00000098
60000	0,00033644	0,00000229
70000	0,00027964	0,00000291
80000	0,00024716	0,00000009
90000	0,00021864	0,00000376

Nie są one dokładnie równe wektorowi, ale są bliskie.

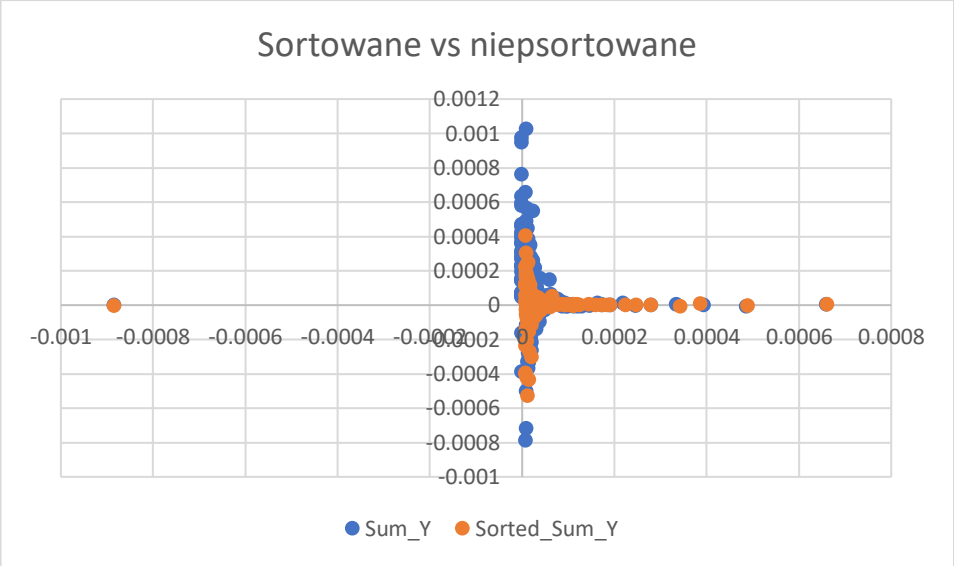


Wykres pokazuje ze oscylują one w granicy 0,0. Jednak, gdy N rosną również błędy

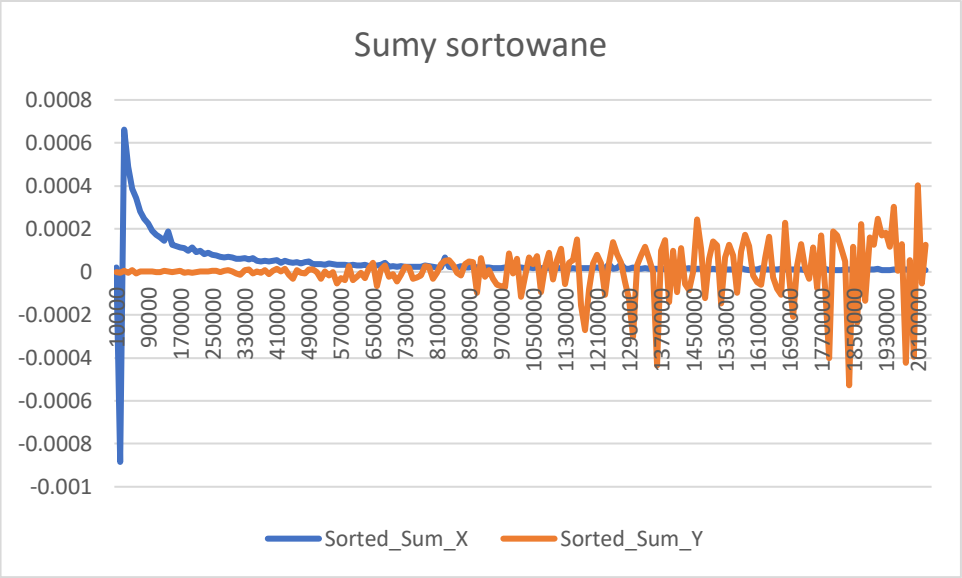
Podsumowując, NIE, nie dają dokładnie wektora zerowego.

H3. Sumy współrzędnych wektorów w_i można policzyć osobno, a następująca zmiana kolejności sumowania sprawi, że wynik będzie bliższy wektorowi zerowemu.

Dla moich danych TAK, 151/204 dane były bliżej wektora 0 po posortowaniu.

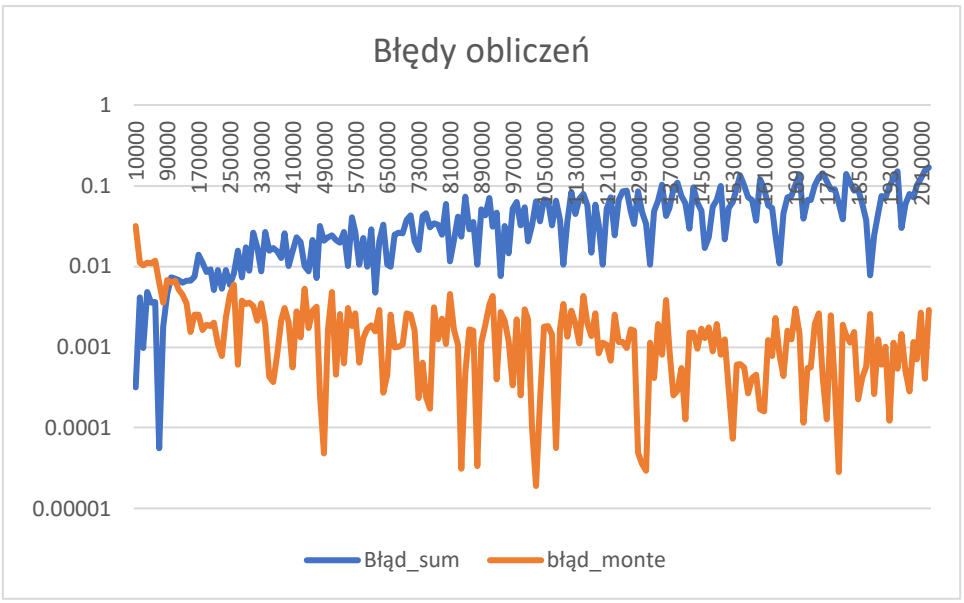


Pomarańczowe kropki przedstawiają posortowane dane i jest ich więcej bliższych [0, 0] niż tych nieposortowanych.

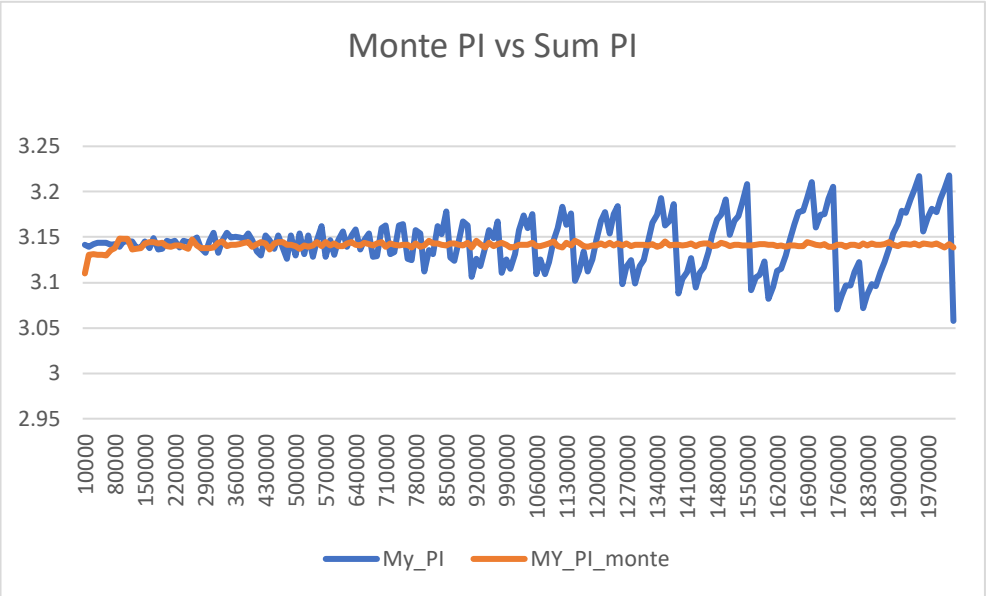


H4. Opisane zastosowanie metody Monte Carlo jest mniej efektywne niż metoda oparta o sumowanie wektorów.

Ciężko powiedzieć i



Ciężko powiedzieć, ponieważ wyniki monte carlo zależą od wartości N. Zakładając, że trafiamy około 75% wartości w ćwiartkę koła dla małych danych nasze wyniki mogą być niedokładne i wtedy sprawdza się lepiej met//oda sumowania. Jednakże dla większych danych monte carlo działa lepiej i generuje błędy mniejsze niż sumowanie

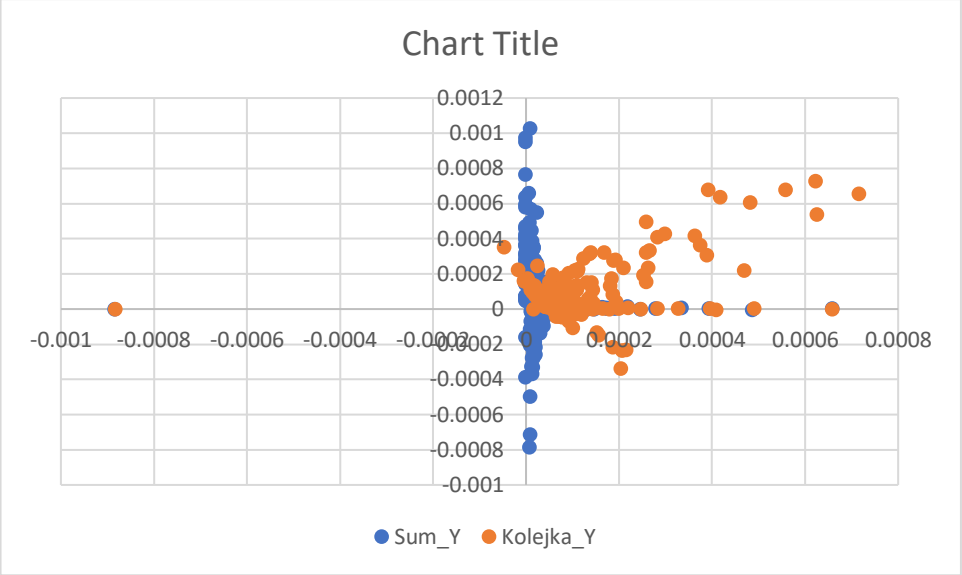


Przykładowe dane:

N	Pi	MY_PI_mont	Points_In	błąd_monte	Błąd_sum	My_PI
10000	3,14159265	3,1099999	7775	0,03159276	0,0003187	3,141752
20000	3,14159265	3,13039994	15652	0,01119271	0,00416118	3,13951206
30000	3,14159265	3,13133335	23485	0,0102593	0,00096386	3,14207458
40000	3,14159265	3,13050008	31305	0,01109258	0,00482338	3,14400434
50000	3,14159265	3,13064003	39133	0,01095262	0,00356549	3,1433754
60000	3,14159265	3,1298666	46948	0,01172605	0,00363224	3,14340878
70000	3,14159265	3,13537145	54869	0,00622121	0,00005562	3,14156485
80000	3,14159265	3,13805008	62761	0,00354257	0,00179976	3,14249253
90000	3,14159265	3,14835548	70838	0,00676283	0,00458366	3,13930082
100000	3,14159265	3,14784002	78696	0,00624737	0,00731533	3,14525032

H5. Podobnie jak w H3 ale w celu sumowania każdego ze zbiorów wybieramy dwa najmniejsze (albo największe) elementy a sumę wstawiamy z powrotem do zbioru.

Używając kolejki możemy otrzymujemy gorsze wyniki, dalsze wektorowi zerowemu niż sum.



Z testowanych pkt 93/204 było bliżej 0.