Programowanie Liniowe   
Projekt 1  
Michał Safuryn 288574

**H1. Zwiększając n można uzyska¢ obwód dowolnie bliski liczbie 2Π.**

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

|  |  |  |
| --- | --- | --- |
| N | 2PI | MY\_2PI |
| 10 | 6,28318531 | 6,18034 |
| 10010 | 6,28318531 | 6,28272 |
| 20010 | 6,28318531 | 6,28021 |
| 30010 | 6,28318531 | 6,28324 |
| 40010 | 6,28318531 | 6,28208 |
| 50010 | 6,28318531 | 6,28791 |
| 60010 | 6,28318531 | 6,28664 |
| 70010 | 6,28318531 | 6,2786 |
| 80010 | 6,28318531 | 6,28573 |
| 90010 | 6,28318531 | 6,27918 |

Powyższy wykres przedstawia jak zmienia się 2Π wyliczone z sumy długości wektorów.  
Można zauważyć, że od około 320010-wierzchłkowego wielokąta 2Π jest bardzo blisko tej stałej. Jednak dla większych N Pi zaczyna się zmieniać i oscyluje w okolicy 6.1 - 6.5

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 *wi* 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 |
| 10 | 0 | 0 |
| 10010 | -2,82E-07 | 2,00E-05 |
| 20010 | 3,50E-06 | -8,80E-04 |
| 30010 | -3,65E-06 | 0,00065923 |
| 40010 | 1,95E-06 | 0,00049925 |
| 50010 | 4,12E-07 | 0,00038958 |
| 60010 | 7,55E-07 | 0,00033021 |
| 70010 | 2,18E-06 | 0,00028729 |
| 80010 | -1,36E-06 | 0,00024819 |
| 90010 | -1,65E-06 | 0,00021839 |
| 100010 | -3,50E-06 | 0,00019455 |

Jak można zauważyć zmienne nie są równe dokładnie [0, 0], ale są one dość blisko blisko.

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 *wi* można policzyć osobno, a następująca zmiana kolejności sumowania sprawi, że wynik będzie bliższy wektorowi zerowemu.**

Dla moich danych TAK, 164/204 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 metoda sumowania. Jednakże dla większych danych monte carlo działa lepiej i generuje błędy mniejsze niż sumowanie

Przykładowe dane:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| N | Pi | MY\_PI\_monte | Points\_In | MY\_PI | Błąd\_monte | Błąd\_sum |
| 10 | 3,141592654 | 3,6 | 9 | 3,09017 | 0,458407 | 0,051423 |
| 10010 | 3,141592654 | 3,1041 | 7768 | 3,14136 | 0,037493 | 0,000233 |
| 20010 | 3,141592654 | 3,12204 | 15618 | 3,140105 | 0,019553 | 0,001488 |
| 30010 | 3,141592654 | 3,12469 | 23443 | 3,14162 | 0,016903 | 0,000027 |
| 40010 | 3,141592654 | 3,13042 | 31312 | 3,14104 | 0,011173 | 0,000553 |
| 50010 | 3,141592654 | 3,13537 | 39200 | 3,143955 | 0,006223 | 0,002362 |
| 60010 | 3,141592654 | 3,14048 | 47115 | 3,14332 | 0,001113 | 0,001727 |
| 70010 | 3,141592654 | 3,14052 | 54967 | 3,1393 | 0,001073 | 0,002293 |
| 80010 | 3,141592654 | 3,14201 | 62848 | 3,142865 | 0,000417 | 0,001272 |
| 90010 | 3,141592654 | 3,14272 | 70719 | 3,13959 | 0,001127 | 0,002003 |
| 100010 | 3,141592654 | 3,14293 | 78581 | 3,145545 | 0,001337 | 0,003952 |
| 110010 | 3,141592654 | 3,14168 | 86404 | 3,145355 | 0,000087 | 0,003762 |
| 120010 | 3,141592654 | 3,1435 | 94313 | 3,145245 | 0,001907 | 0,003652 |

**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 dostać niestety gorsze wyniki, dalsze wektorowi zerowemu niż sum.

Z testowanych pkt 28/204 było bliżej 0.