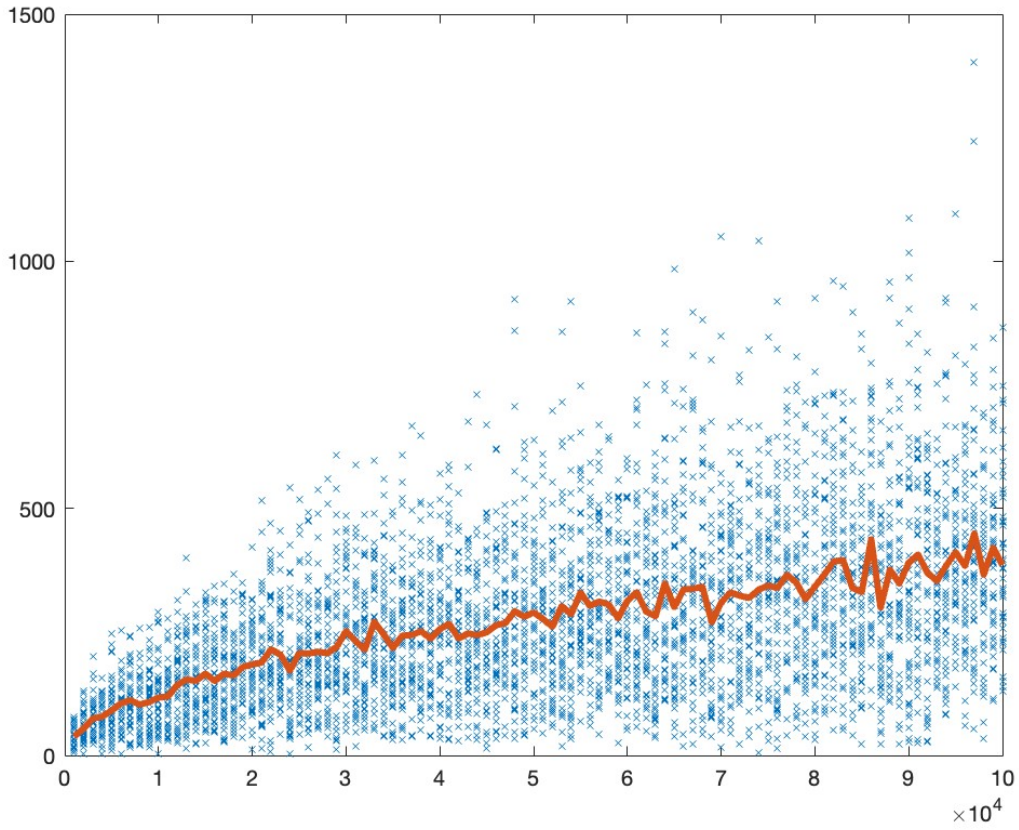
Kajetan Plewa 282212

Sprawozdanie z symulacji problemu *Balls and Bins*

Celem zadania było przeprowadzenie symulacji modelu kul i urn, w celu znalezienia średnich wartości B_n, U_n, C_n, D_n, $Difference_n$ dla n, gdzie n oznacza liczbę urn w modelu. Pamiętając o wnioskach z poprzedniego zadania, tym razem zdecydowałem się na wykorzystanie języka C++. W celu zapewnienia jednostajności rozkładu dla generatora losowych liczb w przedziale [0,n-1], skorzystałem z algorytmu Mersenne Twister i klasy uniform_int_distribution (weryfikacja przy pomocy dokumentacji: https://en.cppreference.com/w/cpp/numeric/random/uniform_int_distribution).

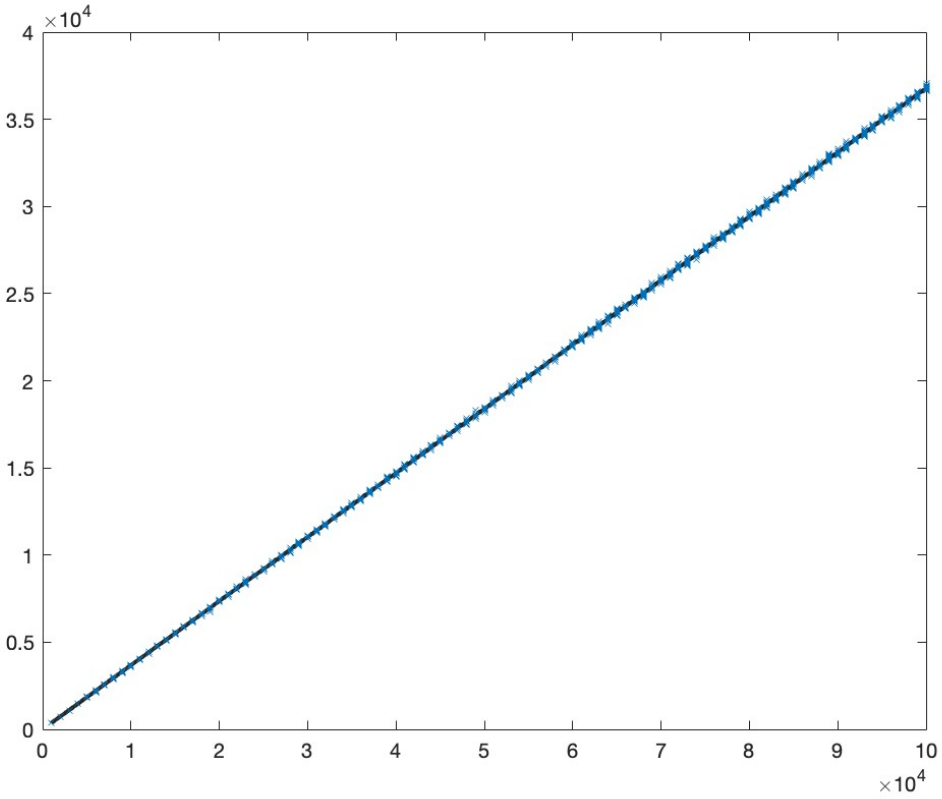
Wyniki kolejnych iteracji zostały zapisane w pliku .csv (Comma Separated Values) za pomocą klasy CSVwriter. Do obróbki danych zdecydowałem się użyć programu MatLab.

Poniżej zaprezentuje otrzymane wyniki w następującym schemacie. Każdy wykres będzie składał się z dwóch warstw - punktów ('x') i funkcji pokazującej rozkład średnich dla danego n.



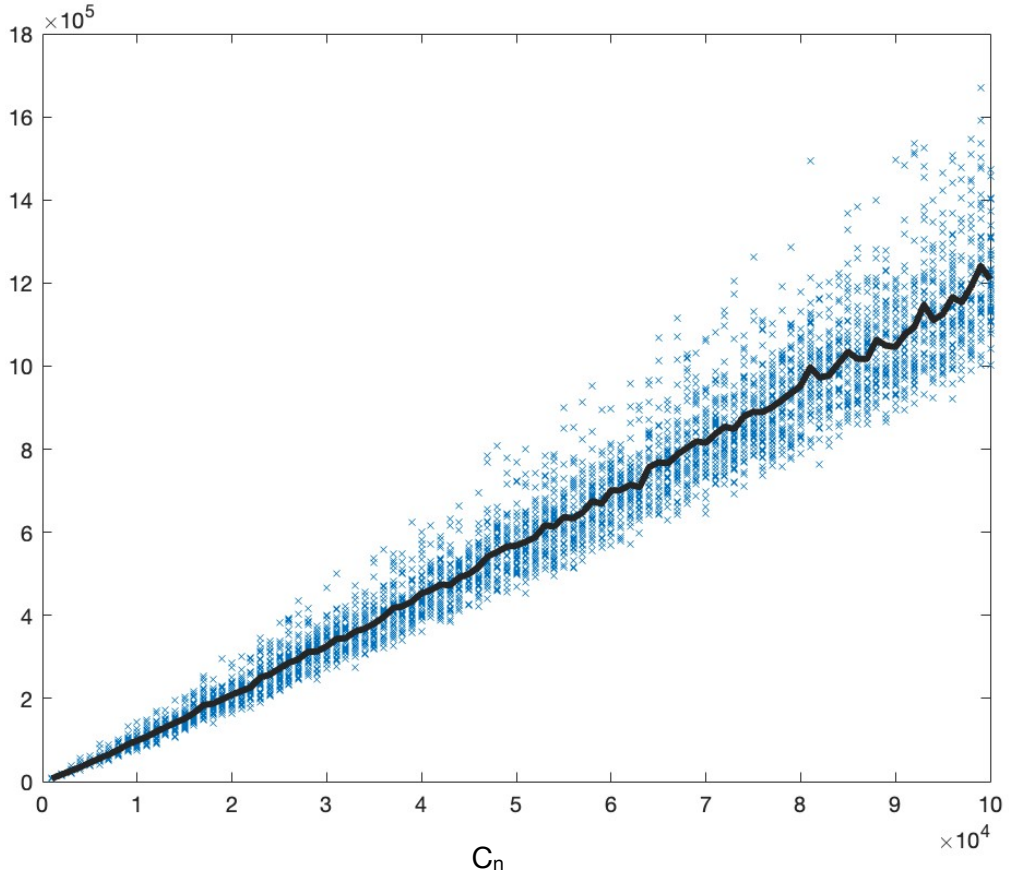
Wykres dla B_n

Pierwszy graf przedstawia liczbę rzutów do pierwszej kolizji w urnach. Jak widać rozrzut danych wokół wartości średniej jest znaczący. Widoczna jest duża ilość "outlier'ów" oraz podobieństwo wykresu do funkcji $y=\sqrt{x}$. Fakt ten sugeruje, że B_n dąży do pewnej wartości.



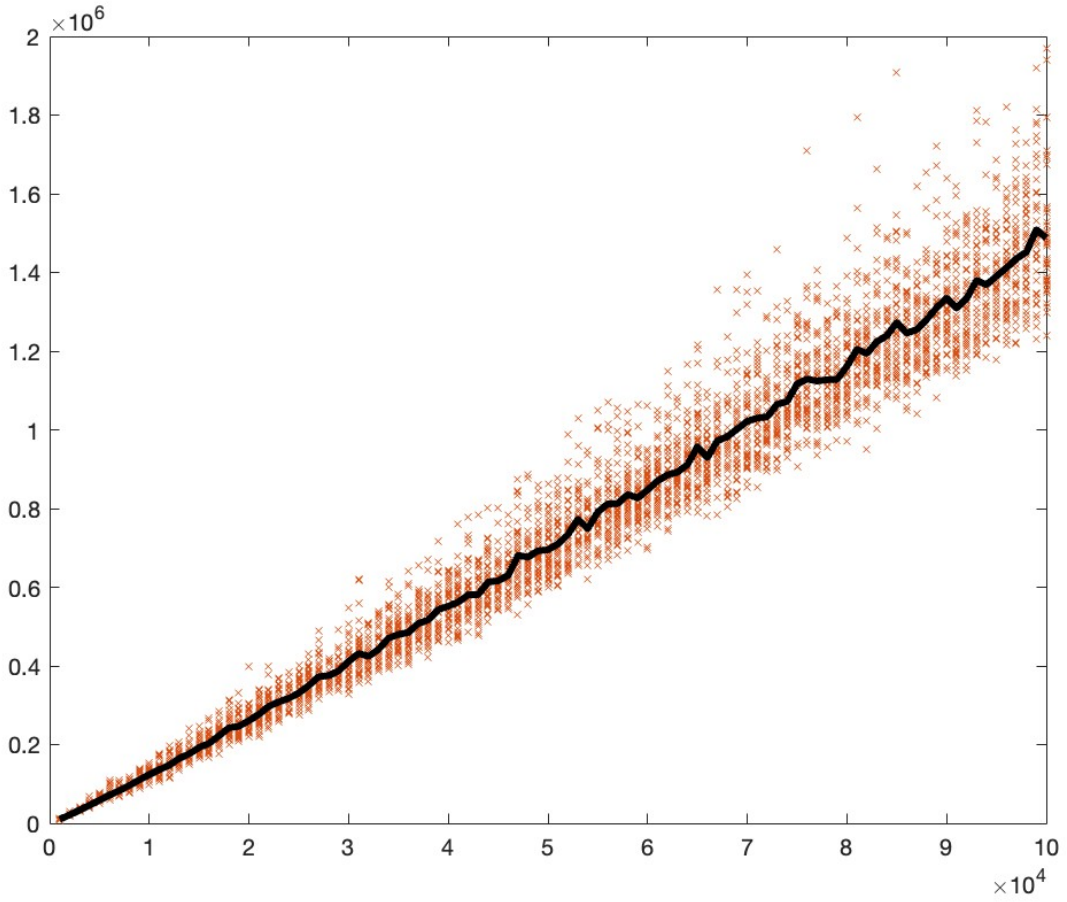
U_n

W przeciwieństwie do poprzedniego wykresu, w tym wypadku punkty nie oscylują wzdłuż średniej - wynik każdej symulacji dla dowolnego n i k posiada wysoką dokładność oraz precyzję. Skąd tak niski błąd pomiaru? Moja hipoteza jest następująca: w przypadku reszty zmiennych skupiamy się na specyficznym zdarzeniu - np. Pierwsza kolizja dotyczy stanu pojedynczej urny - podczas gdy dla Un liczy się zjawisko globalne, czyli stan wszystkich urn.



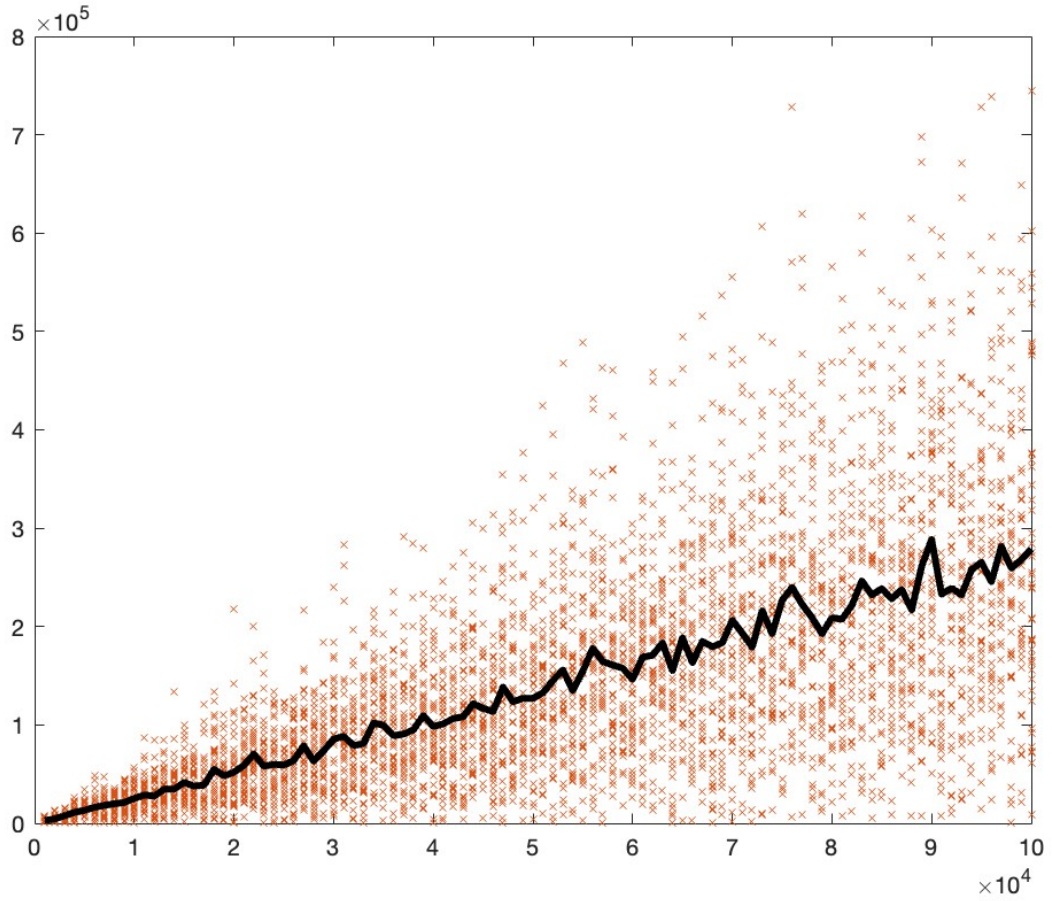
C_n

Wykres przedstawiający C_n - liczbę rzutów, po której każda urna posiada minimum 1 kulkę - podobnie jak U_n odnosi się do stanu wszystkich urn, dlatego wariancja jest mniejsza niż dla B_n. Występują outlier'y lecz ich stosunek do średniej jest mniejszy niż w przypadku B_n - 17/12 w porównaniu do 1500/350 (w zaokrągleniu). Mimo to, punkty w widoczny sposób są blisko funkcji - duża dokładność lecz niska precyzja.



D_n

W przypadku zmiennej D_n dane na wykresie układają się podobnie - jedyną różnica są wartości - różnica pomiędzy nimi pokazana zostanie na kolejnym wykresie.

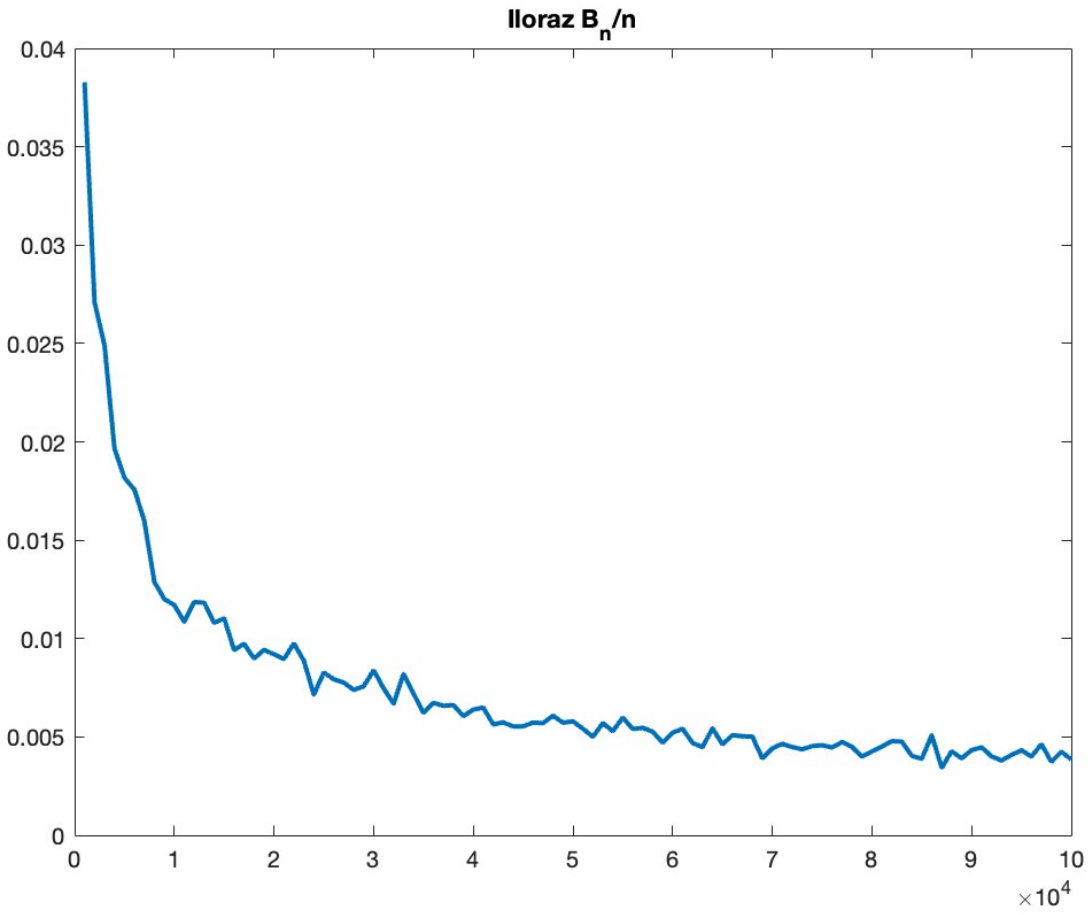


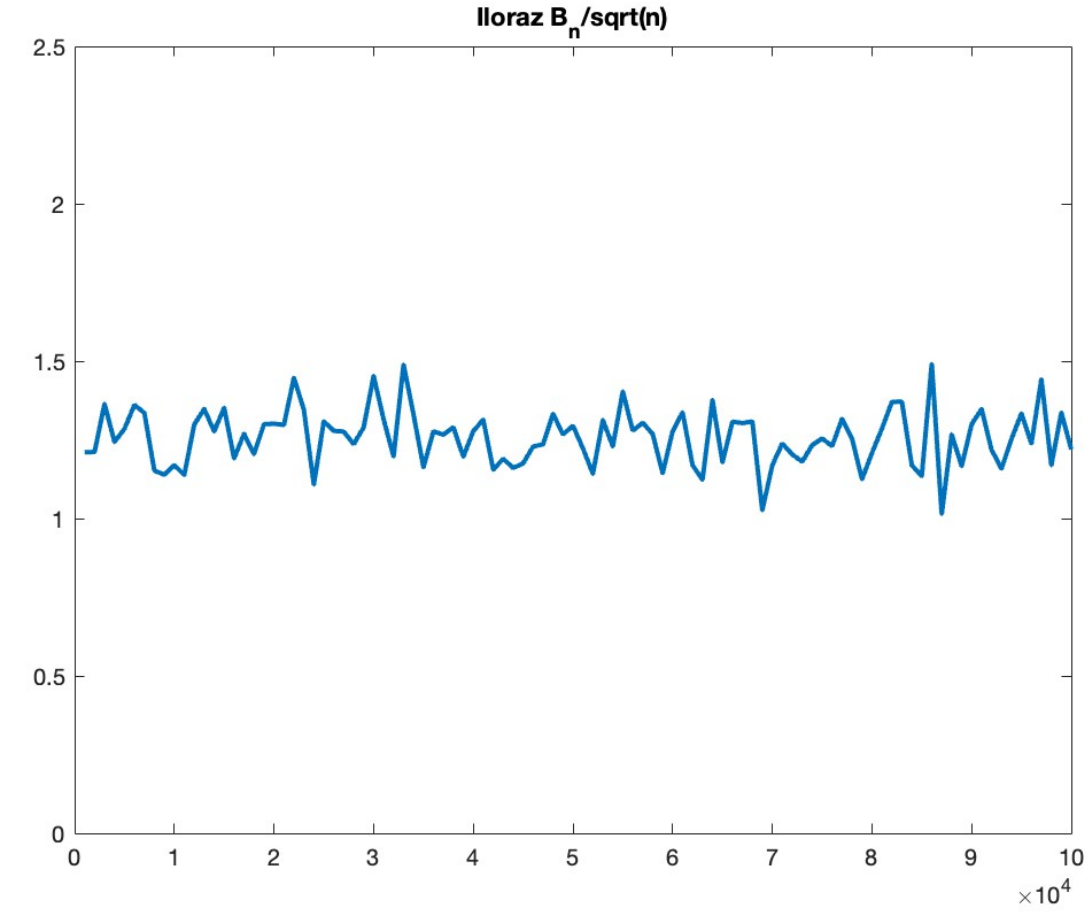
Difference between D_n and C_n

W przypadku $difference_n$ procentowy błąd pomiaru rośnie ponieważ błędy z C_n i D_n należy do siebie dodać. Oznacza to, że dokładność i precyzja pomiaru spada (zgodnie z zasadami propagacji błędu - https://en.wikipedia.org/wiki/Propagation_of_uncertainty).

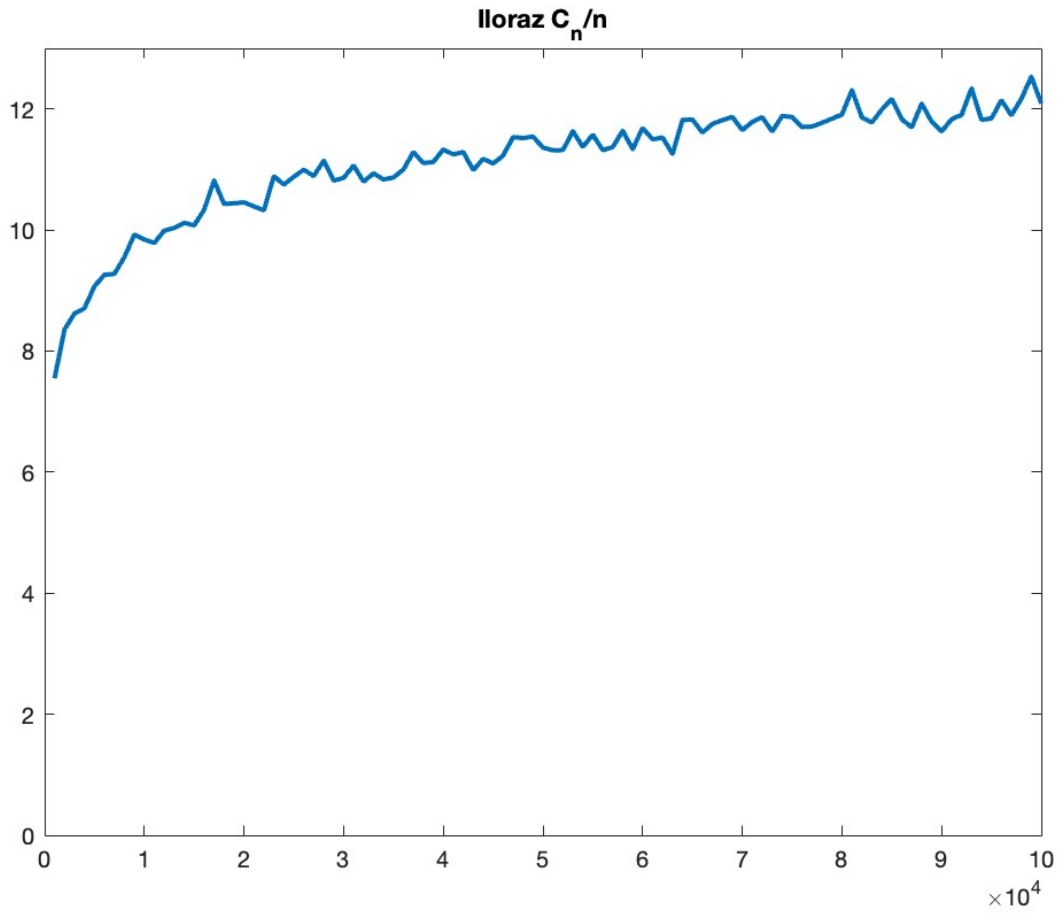
W celu określenia asymptotycznego tempa wzrostu średnie wartości B_n, C_n, U_n, D_n, $Difference_n$ zostały podzielone przez odpowiednie dzielniki:

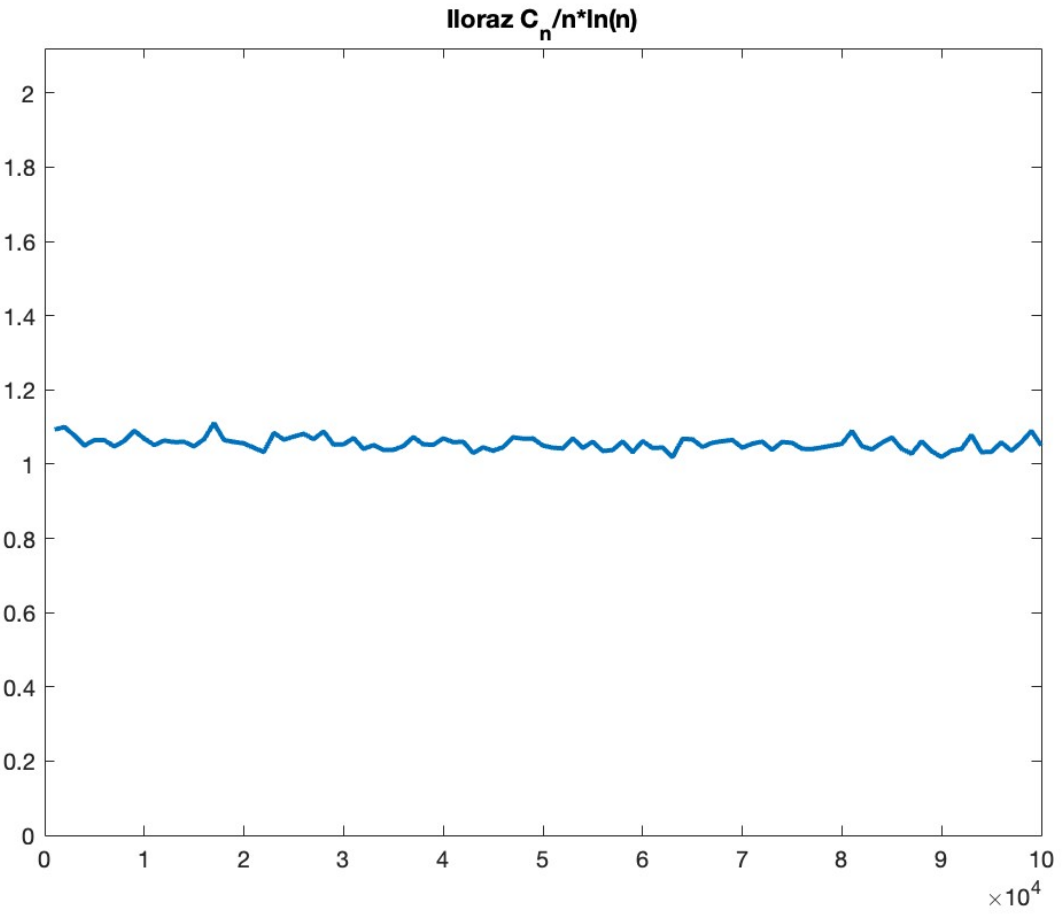
Dla B_n mamy dwa przypadki B_n/n oraz $B_n/\sqrt{n}$.
Dla pierwszego widzimy jak funkcja zbliża się do zera dla większych wartości n. Z tego możemy wywnioskować, że B_n rośnie wolniej od n. Korzystając z definicji notacji duże O wiemy, że $B_n = O(n)$.
Dla drugiego otrzymana funkcja jest stała i wynosi w aproksymacji 1.25 czyli:
$Bn \sim 1.25\sqrt{n}$
Jak widać po wykresach, dla funkcji $\sqrt{n}$ tempo wzrostu jest dokładniejsze.

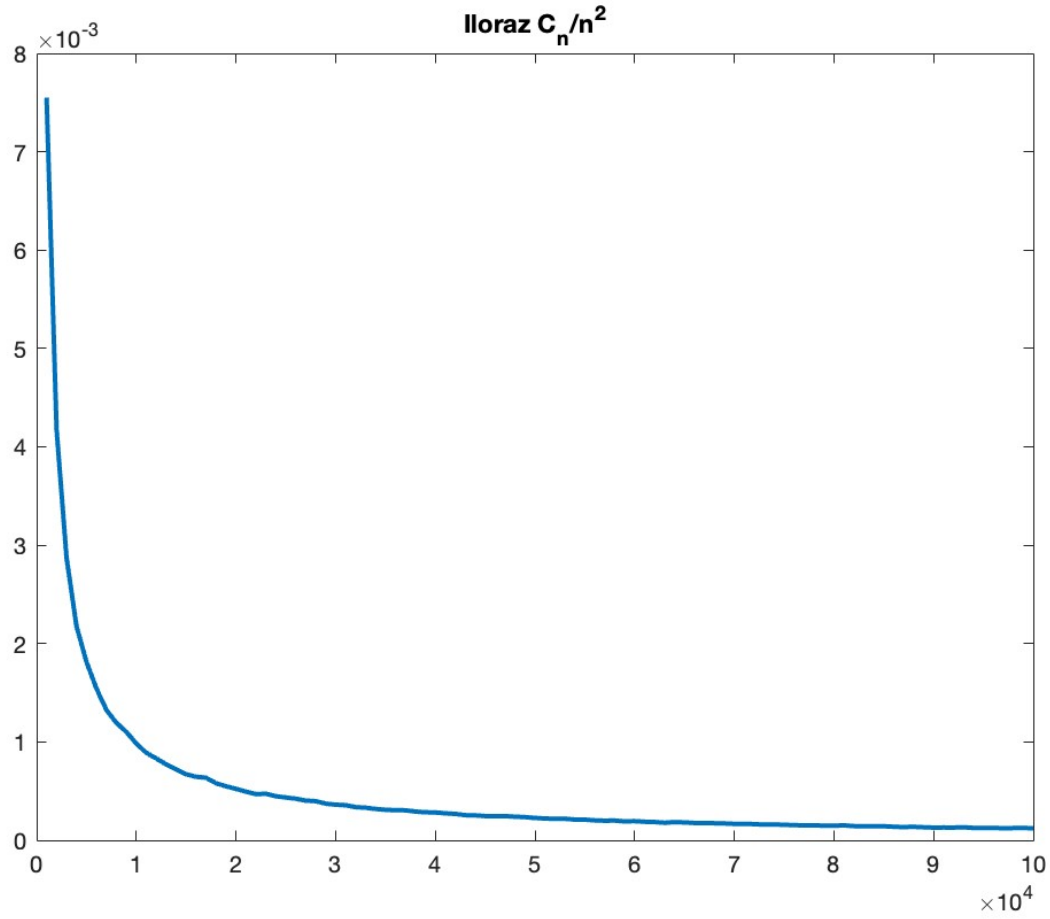




Dla C_n mamy 3 przypadki: C_n/n, $C_n/(n*\ln(n))$, C_n/n^2.


Iloraz C_n/n
12
10
8
6
4
2
0
0
1
2
3
4
5
6
7
8
9
10
$\times 10^4$


Iloraz $C_n/n*\ln(n)$
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
1
2
3
4
5
6
7
8
9
10
$\times 10^4$



Dla pierwszej funkcji widzimy, że dla większych n wartość zbliża się do 12. Możemy więc użyć zapisu:

$$C_n=O(n)$$

$$C_n \sim 12n$$

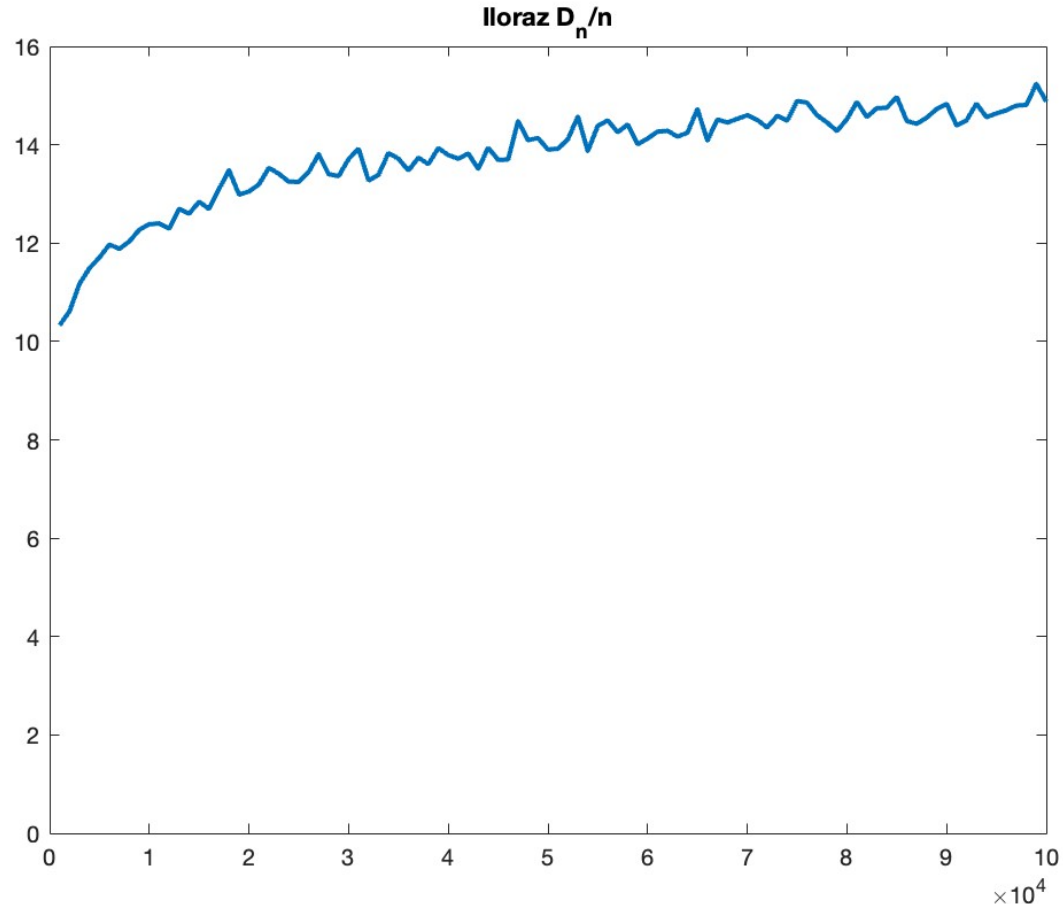
Drugi przypadek to funkcja stała - jest więc też przypadkiem najdokładniejszym. Można więc stwierdzić, że:

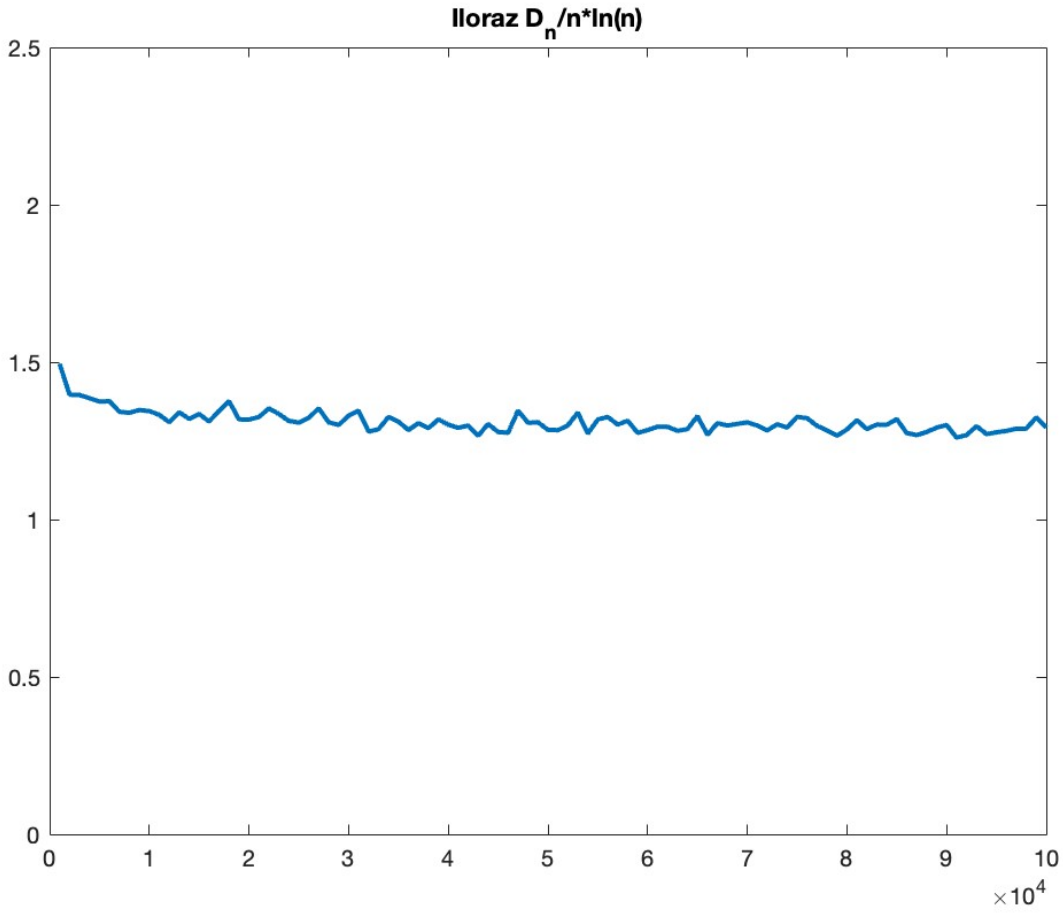
$$C_n \sim 1.1*n*\ln(n)$$

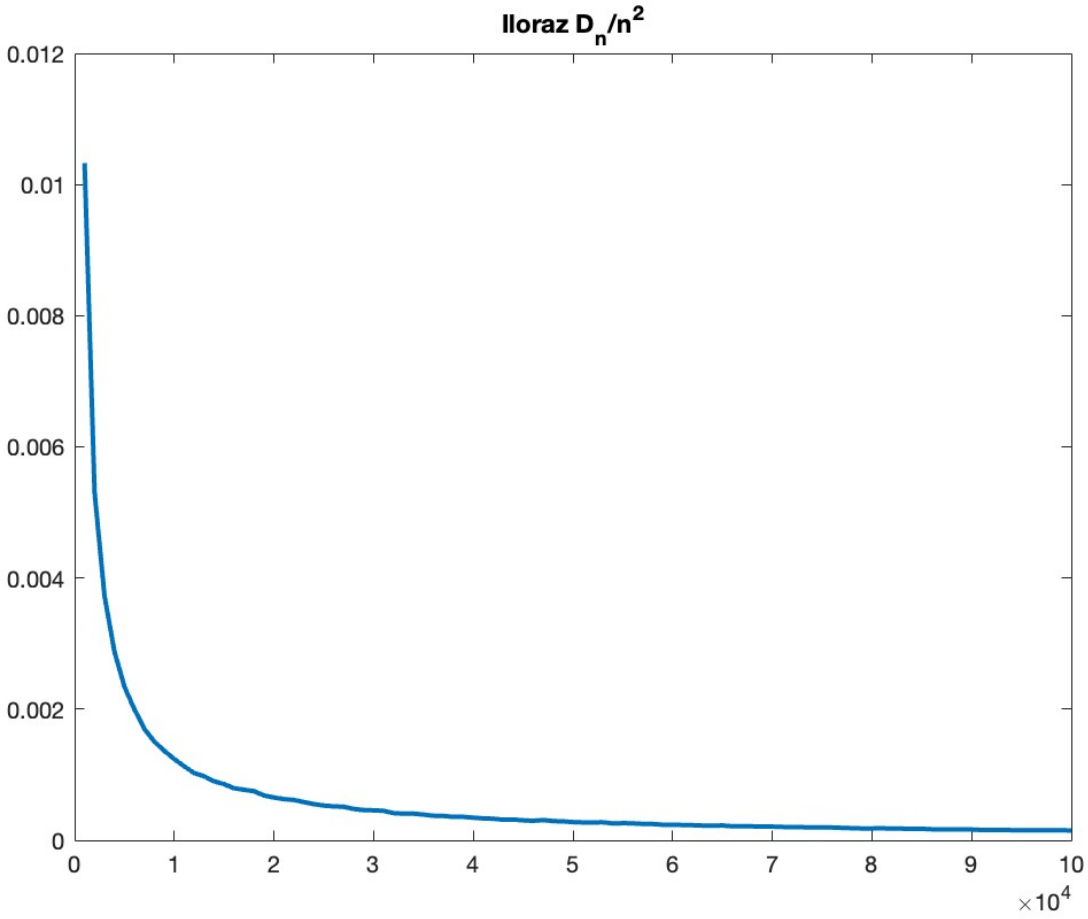
Ostatnia funkcja dla C_n dąży do zera. Z tego wynika, że dla n->∞ n^2 rośnie szybciej od C_n - czyli istnieje n_0, od którego C_n<=n^2:

$$C_n=O(n^2)$$

Dla D_n mamy również trzy dzielniki: n, n*ln(n), n^2. Biorąc pod uwagę podobieństwo między tymi dwoma zmiennymi możemy spodziewać się podobnego tempa wzrostu lecz z innymi stałymi c.







Korzystając z identycznej logiki otrzymujemy:

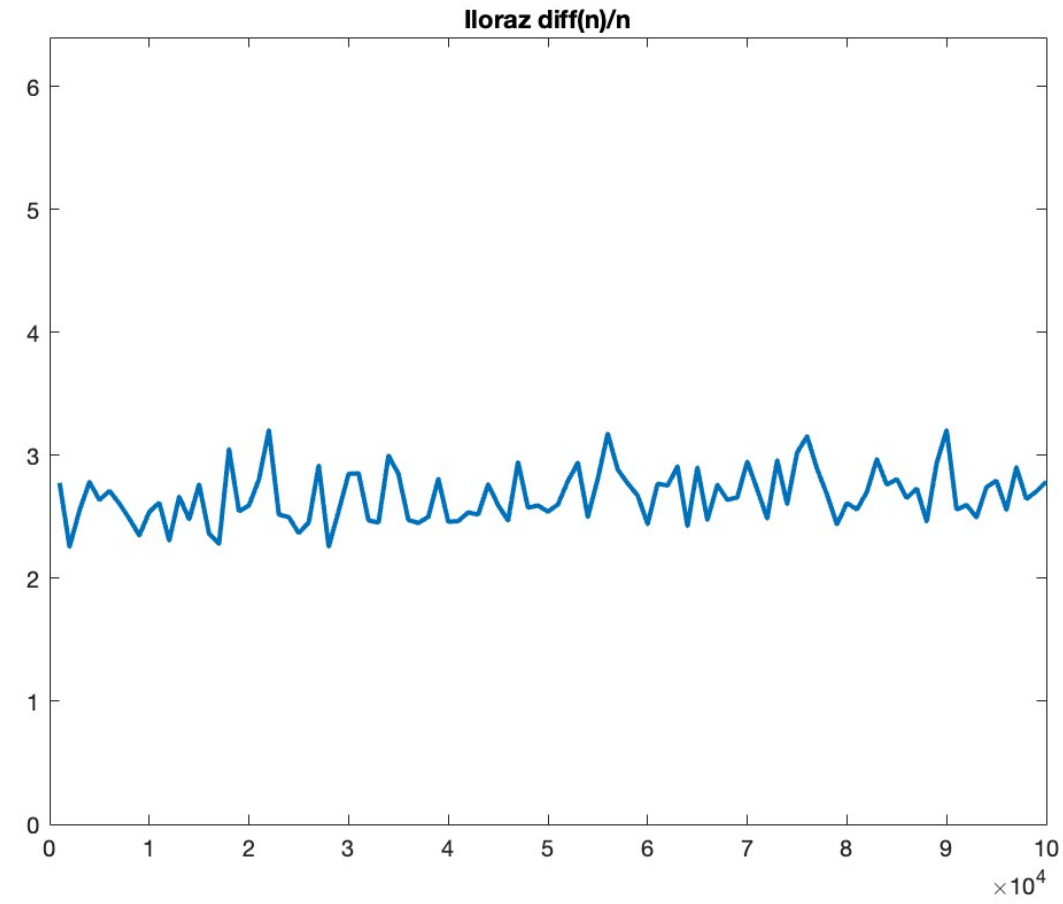
$D_n=O(n)$

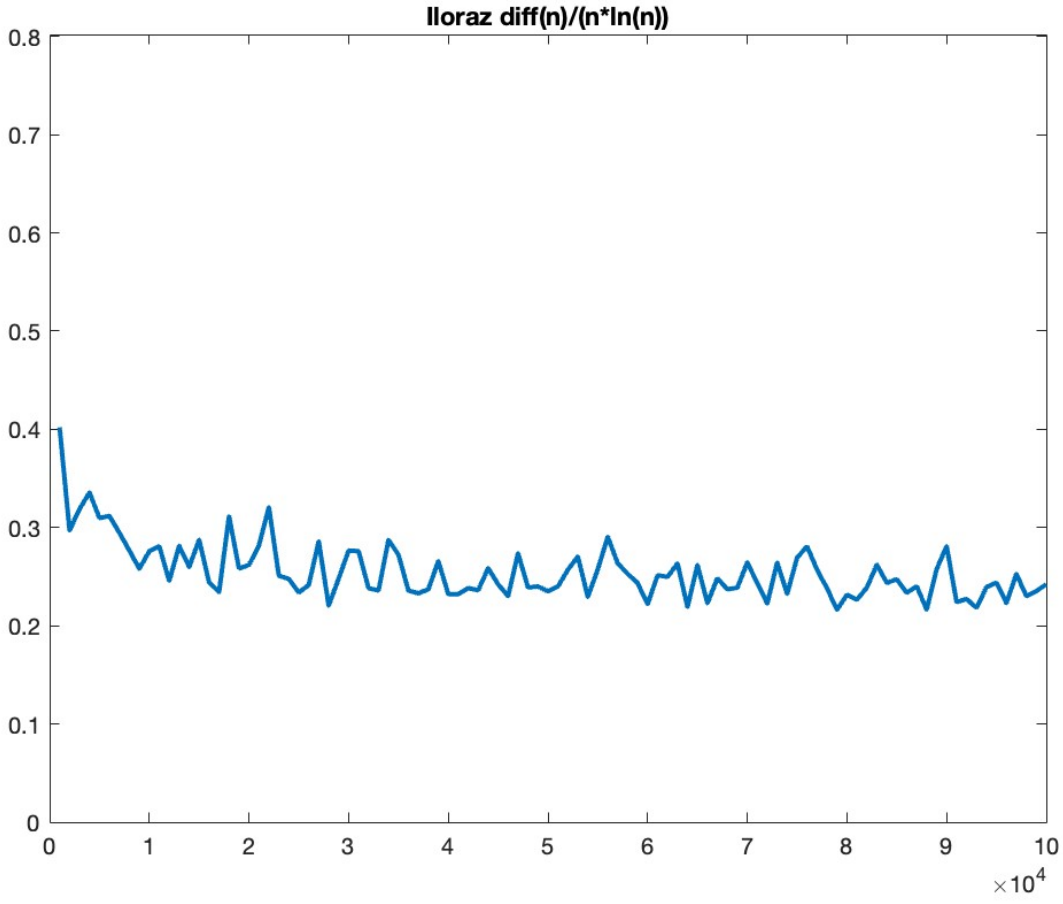
$D_n\sim16n$

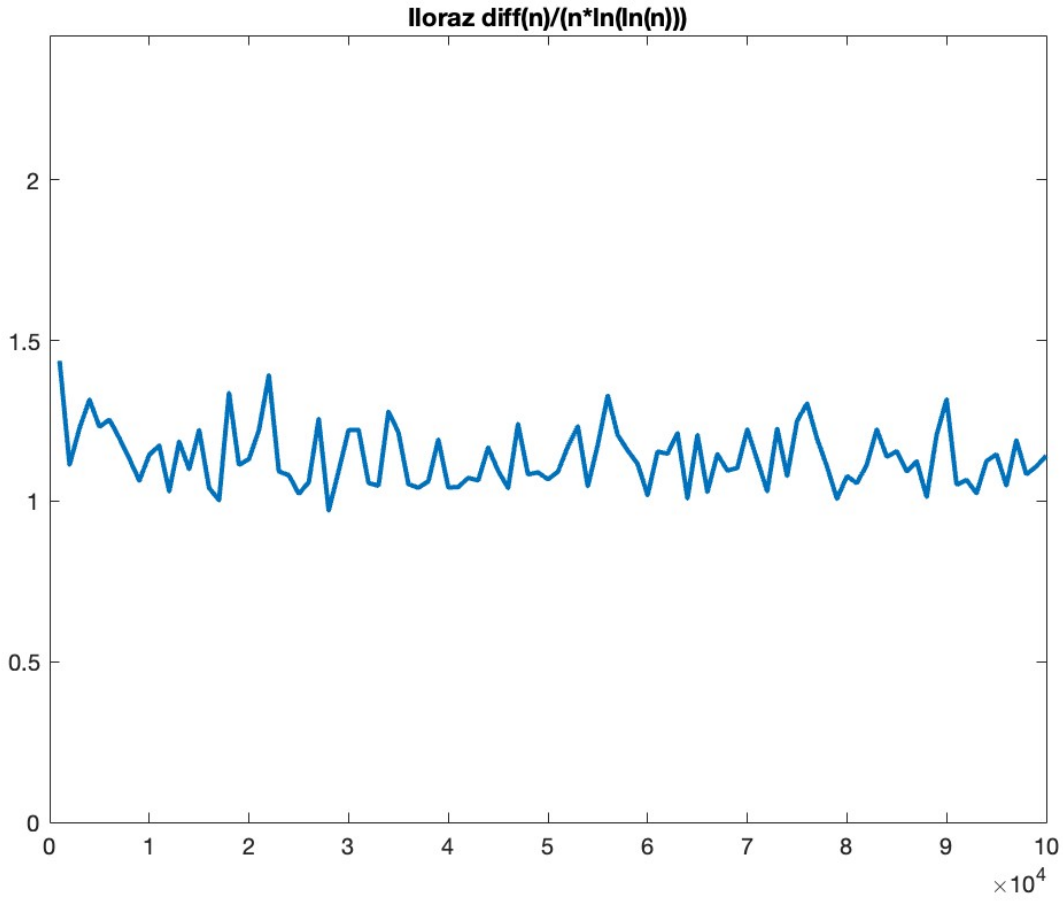
$D_n\sim1.4*n*\ln(n)$

$D_n=O(n^2)$

Ostatnią przypadkiem jakim musimy rozważyć jest różnica między D_n i C_n.


lloraz diff(n)/n
0
1
2
3
4
5
6
0
1
2
3
4
5
6
7
8
9
10
×10^4


lloraz diff(n)/(n*ln(n))
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
1
2
3
4
5
6
7
8
9
10
×10^4



W tym przypadku po podzieleniu powstały 3 funkcje stałe. Korzystamy więc z notacji tylda:

$Diff_n \sim 2.5n$

$Diff_n \sim 0.25*n*\ln(n)$

$Diff_n \sim 1.2*n*\ln(\ln(n))$

Połączenie modelu kul i urn z birthday paradox i coupon collector's problem

Wielkość B_n, którą otrzymaliśmy w trakcie symulacji możemy utożsamiać z *birthday paradox*. Paradoks ten polega na znalezieniu prawdopodobieństwa tego, że z grupy n ludzi przynajmniej dwie z nich mają urodziny w tym samym dniu. Symulacja odpowiada na to pytanie - gdybyśmy wykonali k iteracji dla n = 365 (urny reprezentują dni, a kulki ludzi), moglibyśmy znaleźć ile ludzi średnio potrzeba aby doszło do kolizji w urnie. Idąc tym śladem, możliwym byłoby znalezienie eksperymentalnej odpowiedzi na ten problem.

W przypadku *coupon collector's problem* odnosi się on do C_n. Pytanie na jakie tym razem należy odpowiedzieć to: jak dużo kuponów należy wyciągnąć z jakiegoś pudełka, żeby wylosować każdy kupon (jest n rodzajów)?
Dla n jest to oczywiście równe obliczonej przez nas liczbie rzutów, potrzebnych do zapełnienia każdej urny przynajmniej jedną kulą. Korzystając z wyznaczonych w badaniu asymptotyk, można łatwo policzyć odpowiedź dla danego n.

Birthday paradox, a funkcje haszujące

Żeby odpowiedzieć na to pytanie należy najpierw wytłumaczyć czym jest funkcja haszująca. Przypisuje ona dowolnemu ciągowi danych wartość z określonego zakresu - jego ograniczenie, powoduje, że przy wystarczająco dużej liczbie próbek istnieje szansa, że dwie wartości będą miały taki sam hash.
To zjawisko jest podstawą ataków urodzinowych polegających na znalezieniu dwóch różnych wartości które posiadają taki sam hash. Aby temu zapobiec należy jak najbardziej zwiększyć zakres tak, by liczba prób potrzebna do znalezienia kolizji była obliczeniowo nieosiągalna.