Pracownia z analizy numerycznej
Sprawozdanie do zadania **P1.2**

Prowadzący: dr Rafał Nowak
Mateusz Hazy

Wrocław, dnia 3 listopada 2017

1 Wstęp

Jednym z podstawowych problemów numerycznych jest utrata dokładności wraz ze wzrostem liczby wykonanych operacji arytmetycznych. Błąd spowodowany jest sposobem maszynowej reprezentacji liczb, jednak poprzez wybór odpowiedniej kolejności, precyzji arytmetyki oraz sposobu wykonywania działań arytmetycznych można go zminimalizować.

Celem niniejszego sprawozdania jest sprawdzenie i porównanie jakości wyników w zależności od wyboru metody obliczania.

W dalszych rozdziałach zostały przedstawione wyniki sumowania ciągów $\frac{1}{n}$ oraz $\frac{1}{n^2}$ oraz błędy względne jakie pojawiły się podczas obliczeń.

2 Opis eksperymentu

Sumowanie ciągów zostało przeprowadzone na 3 różne sposoby:

- Zwykłe sumowanie
- Sumowanie w odwrotnej kolejności
- Sumowanie za pomocą algorytmu sumacyjnego Kahana:

```
function Kahan(input, n)
        sum = input[1]
        c = 0
        for i = 2 to n
                y = c + input[i]
                t = sum + y
                c = (sum - t) + y
                sum = t
        end
        return sum
end
```

Wyjaśnienie:
$t = sum + y$ - część bitów y zostanie utracona
$(sum - t) = -y$ ze straconymi bitami
$(sum - t) + y =$ stracone bity y, które mogą zostać dodane w następnych iteracjach

Dla ciągu $\frac{1}{n^2}$ zostało zsumowanych 10^4, natomiast dla $\frac{1}{n}$ - 10^7 elementów. Działania zostały przeprowadzone w arytmetyce Float32 oraz Float64.

3 Przewidywane wyniki

3.1 Kolejność sumowania

Rozważmy błąd powstały przy maszynowym dodawaniu 2 liczb:

$$fl(x+y) = (x+y)(1+\xi)$$

gdzie $|\xi|$ nie przekracza precyzji arytmetyki. $(u, 2^{-t})$
Operator + łączy w lewo, więc:

$$fl(s) = fl(\sum_{i=1}^{n} x_i) = fl(fl(\sum_{i=1}^{n-1} x_i) + x_n) = (fl(\sum_{i=1}^{n-1} x_i) + x_n)(1+\xi_n)$$

Łatwo zauważyć, że:

$$fl(s) = \sum_{i=1}^{n}(x_i \prod_{j=i}^{n}(1+\xi_j))$$

Rozważmy pesymistyczny przypadek, gdy $\xi_1 = \xi_2 = ... = \xi_n = u$
Wtedy:

$$fl(s) = \sum_{i=1}^{n} x_i(1+u)^{n-i+1}$$

Widać, że początkowe elementy sumy obarczone są największym błędem.
Intuicyjnie:

Mamy n liczb (x_i) oraz n współczynników (a_j). Każdej liczbie należy przyporządkować współczynnik tak, aby $\sum a_j x_i$ była jak najmniejsza. Jeśli więc najmniejszym x_i przyporządkujemy największe a_i to intuicyjnie suma będzie najmniejsza.

Twierdzenie 1. *Niech* $0 < x_1 < x_2 < ... < x_n$ *oraz* $a_1 > a_2 > ... > a_n > 0$
σ *- permutacja n-elementowa.*
Wówczas wartość $S(\sigma) = \sum_{i=1}^{n} a_{\sigma(i)} x_i$ *jest najmniejsza, gdy* $\sigma = id$

Dowód. Nie wprost.
Załóżmy, że dla pewnej $\sigma \neq id$, $S(\sigma)$ jest najmniejsze. Weźmy najmniejsze takie i, że $\sigma(i) \neq i$. Wtedy $\sigma(i) = j, j > i$
Niech:

$$s_1 = a_j x_i + a_i x_k,\ \sigma(k) = i,\ \text{więc } k > i$$

$$s_2 = a_i x_i + a_j x_k$$

Wówczas:

$$s_1 - s_2 = (a_j - a_i)x_i + (a_i - a_j)x_k = (x_k - x_i)(a_i - a_j) \geqslant 0$$

Niech:

$$\sigma'(x) = \begin{cases} \sigma(x)\ dla & x \neq i, k \\ i & \text{dla } x = i \\ j & \text{dla } x = k \end{cases}$$

Wówczas:

$$S(\sigma) - S(\sigma') = s_1 - s_2 > 0, \text{ co przeczy minimalności } S(\sigma)$$

□

Analogicznie można pokazać, że suma będzie największa, jeśli współczynniki a_i posortujemy rosnąco.

Można się więc spodziewać, że w przypadku malejęcych ciągów, zwykłe sumowanie będzie mniej dokładne, niż sumowanie w odwrotnej kolejności. Co więcej, posortowanie elementów ciągu rosnąco da najlepsze wyniki, a malejąco - najgorsze.

3.2 Algorytm Sumacyjny Kahana

Dowodzi się, że:

$$Kahan(s) = \sum_{i=1}^{n}(1+\xi_i)x_i, \text{ gdzie } |\xi_i| \leqslant 2 \cdot 2^{-t} + \mathcal{O}(n2^{-2t})$$

Natomiast:

$$Suma(s) = \sum_{i=1}^{n}(1+\psi_i)x_i, \text{ gdzie } |\psi_i| \approx \mathcal{O}(n2^{-t})$$

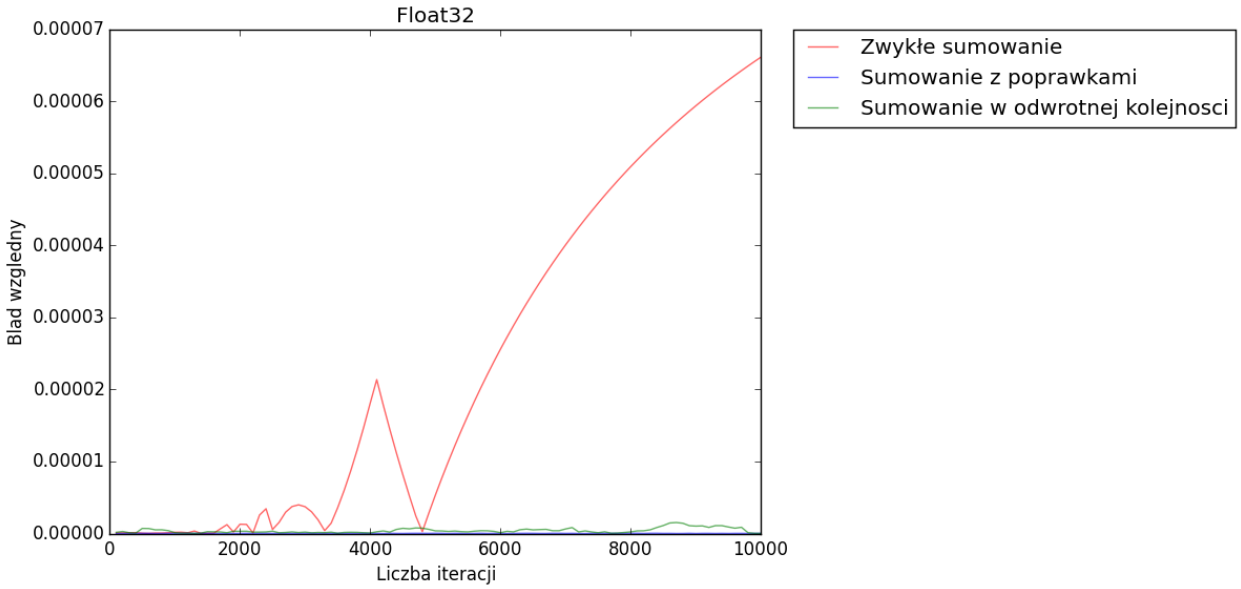
gdzie *Kahan, Suma* są odpowiednio algorytmem sumacyjnym Kahana i zwykłym sumowaniem.

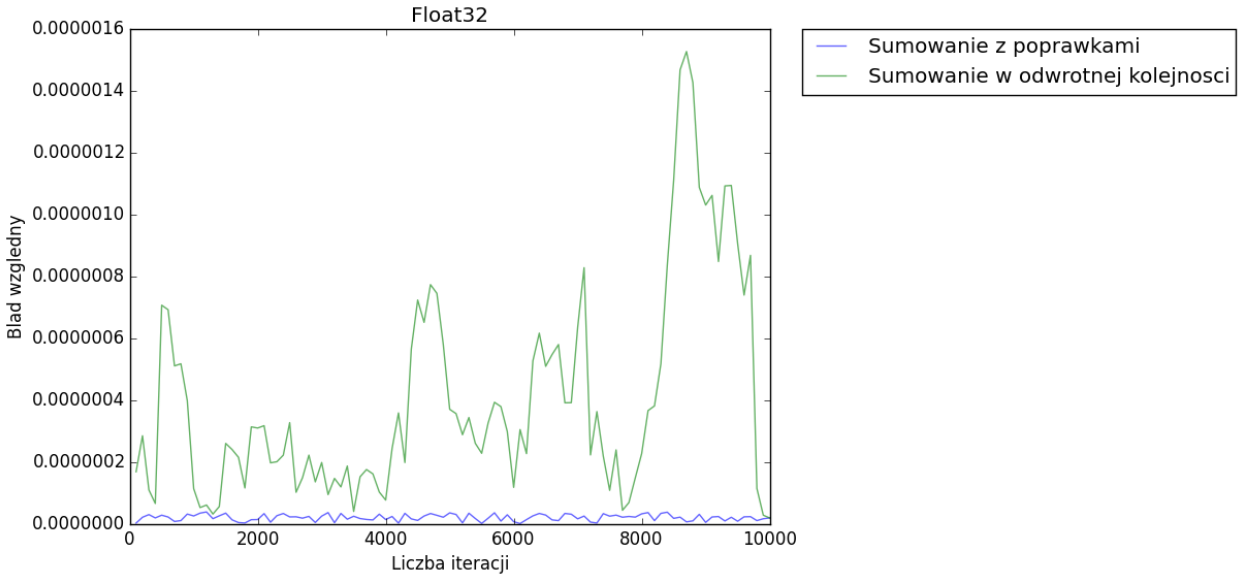
Można się spodziewać, że wyniki uzyskane za pomocą algorytmu Kahana będą najdokładniejsze. Dla $n < 2^t$ błąd względny wyniku można oszacować przez $3 \cdot 2^{-t}$.

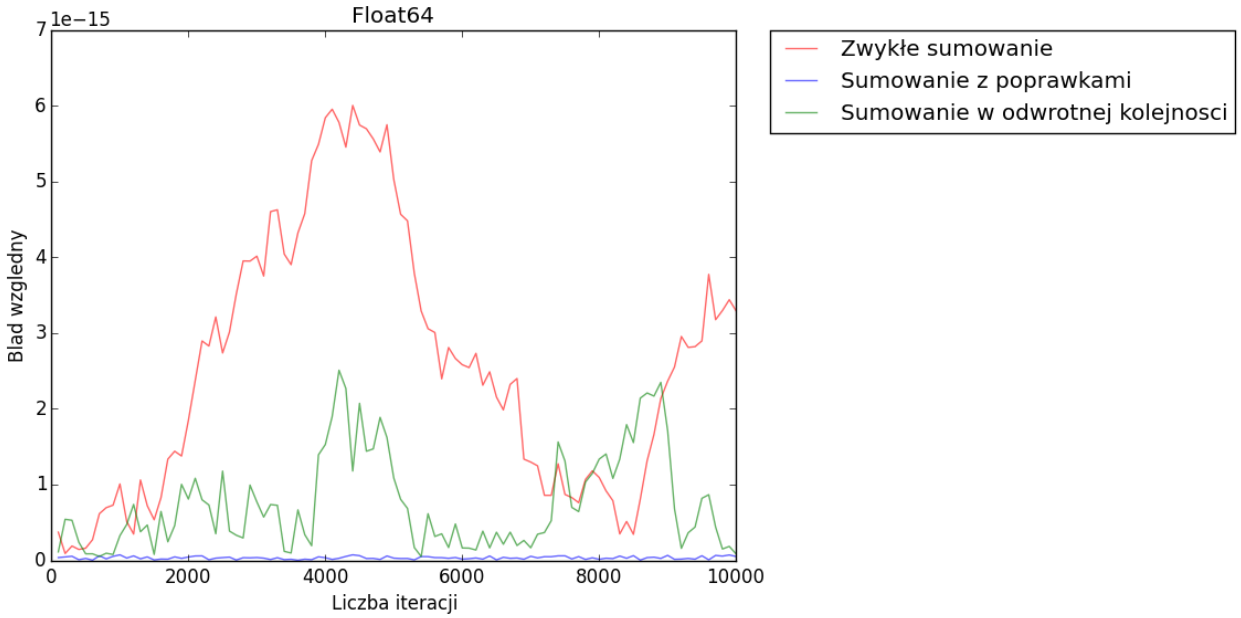
4 Wyniki doświadczalne

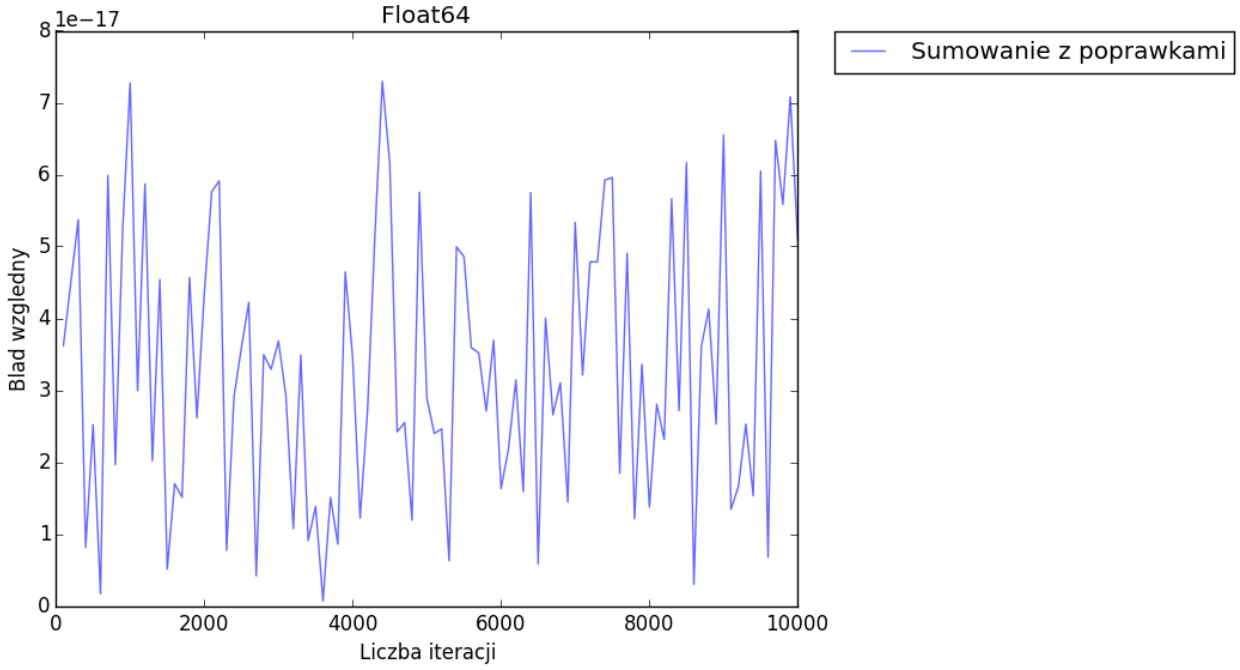
4.1 Sumowanie ciągu $\frac{1}{n^2}$

Poniżej przedstawione zostały wykresy błędu względnego sumowania ciągu $\frac{1}{n^2}$ w zależności od liczby zsumowanych elementów przy użyciu arytmetki Float32 oraz Float64.




Float32
Blad wzgledny
Liczba iteracji
Sumowanie z poprawkami
Sumowanie w odwrotnej kolejnosci

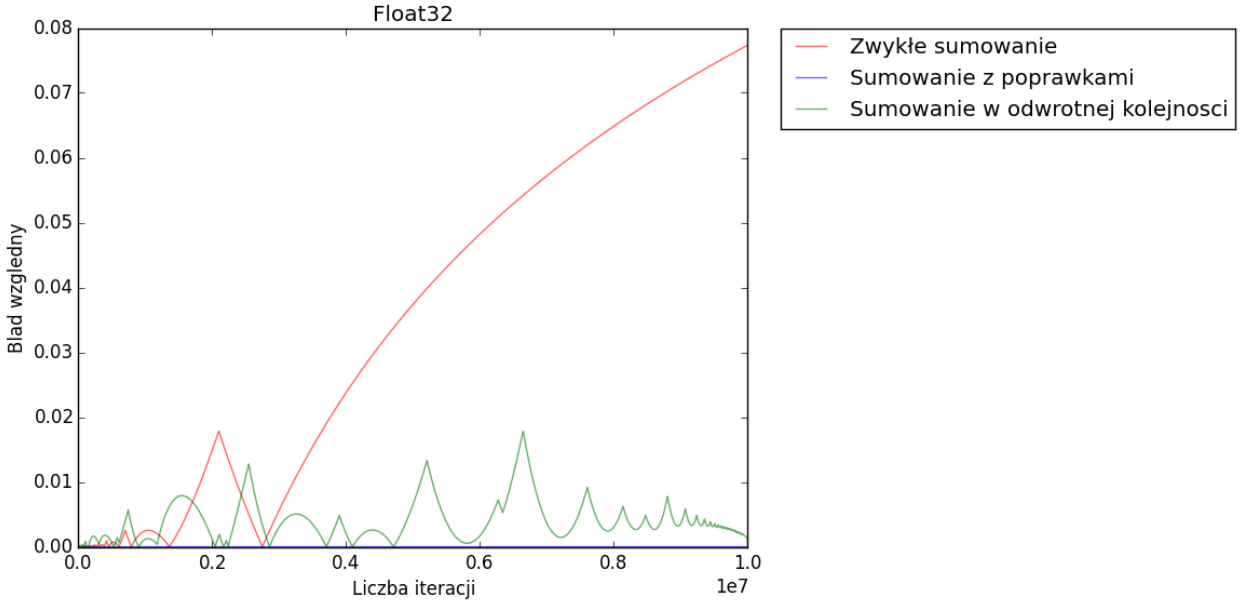

Float64
1e−15
Blad wzgledny
Liczba iteracji
Zwykłe sumowanie
Sumowanie z poprawkami
Sumowanie w odwrotnej kolejnosci

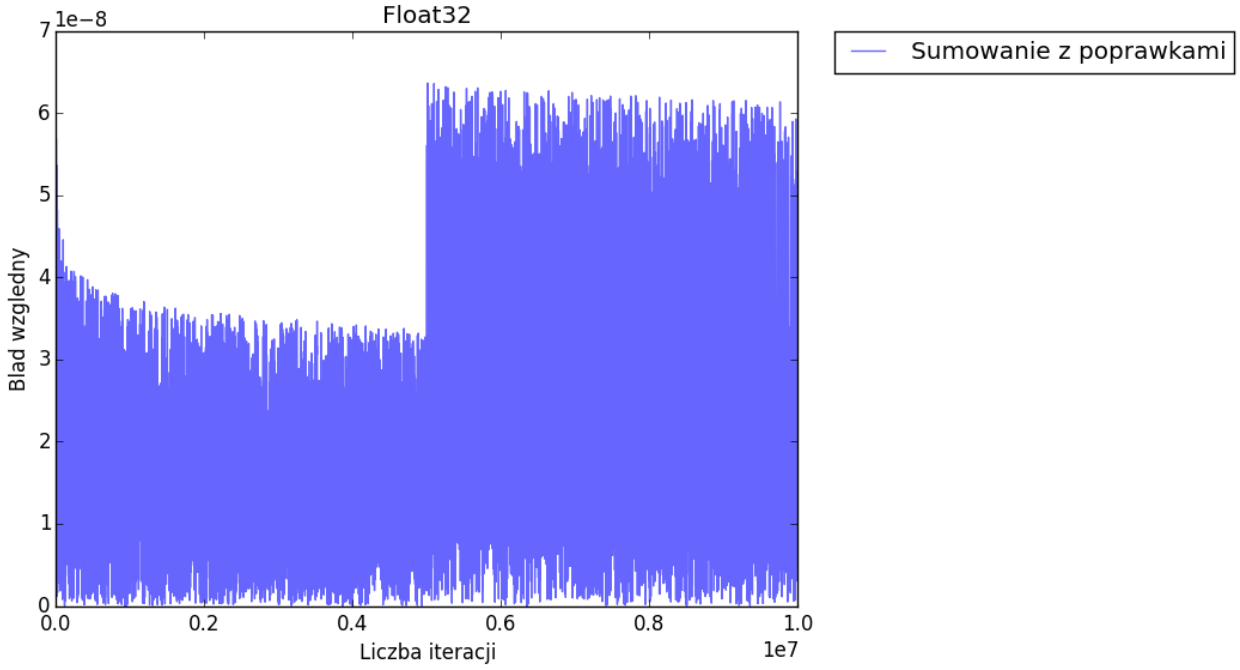


4.2 Sumowanie ciągu $\frac{1}{n}$

Motywacja

W przeciwieństwie do ciągu $\frac{1}{n^2}$, ciąg $\frac{1}{n}$ jest rozbieżny, dzięki czemu błąd bezwzględny może być dowolnie duży, więc możliwe, że błędy względne też będą większe, niż w przypadku ciągu $\frac{1}{n^2}$.





4.3 Wnioski

- Dokładności poszczególnych sumowań są zgodne z przewidywaniami.
- Wykresy nie są monotoniczne, więc błędy są w pewnym stopniu losowe.
- Przy sumowaniu w odwrotnej kolejności końcowy błąd względny był zaskakująco mały - porównywalny do błędu przy sumowaniu algorytmem Kahana. Jednak na podstawie wykresów można zauważyć, że przez większą część sumowania (sumowanie mniejszych składników), błąd był duży.
- Błąd względny sumowania algorytmem Kahana był rzędu:
  - 10^{-8} dla arytmetyki Float32
  - 10^{-17} dla arytmetyki Float64
  Jak widać, około dwukrotne zwiększenie precyzji dało około dwukrotnie dokładniejszy wynik.
- Sumowanie ciągu $\frac{1}{n}$ odniosło oczekiwany efekt. Dla arytmetyki Float32 uzyskany błąd względny był istotnie większy niż przy sumowaniu ciągu $\frac{1}{n^2}$. Faktem jest, że dla ciągu $\frac{1}{n^2}$ zsumowanych zostało 10^4 elementów, a dla ciągu $\frac{1}{n}$ - 10^7 elementów. Jednak dalsze elementy ciągu $\frac{1}{n^2}$ są tak małe, że w precyzji Float32 nie zostałyby dodane do sumy. Z drugiej strony ciąg ten jest zbieżny, więc błąd względny nie zwiększyłby się znacząco.

5 Asymptotyka błędu algorytmu Kahana

Do doświadczalnego wyznaczenia asymptotycznego błędu względnego przy sumowaniu algorytmem Kahana zostały użyte wyniki sumowania ciągu $\frac{1}{n}$ w arytmetyce Float32.
Hipoteza

$$fl(s) = \sum_{i=1}^{n}(1+\xi_i)x_i \ , \ gdzie \ |\xi_i| \leqslant 2^*2^{-t} + \mathcal{O}(n2^{-2t}) \ , \ \sum_{i=1}^{n} x_i = s$$

Niech:

$$\xi_i = \xi_j = \xi, \text{ dla każdych i, j}$$

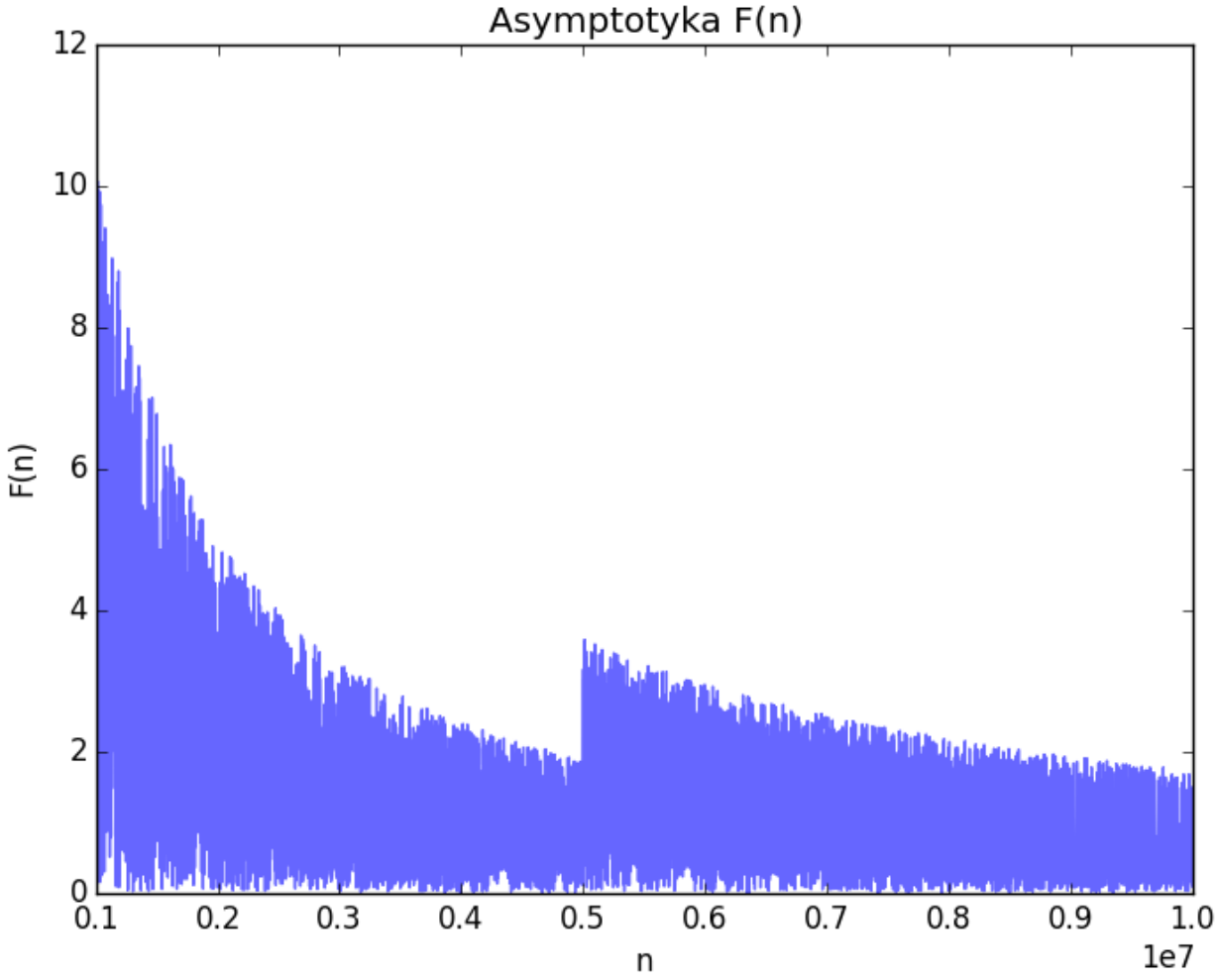
$$\xi = \mathcal{O}(n2^{-2t})$$

Po przekształceniu wzoru :

$$\xi = \frac{fl(s) - s}{s}$$

$$F(n) = \frac{\xi}{n2^{-2t}} = \mathcal{O}(1)$$

Należy sprawdzić, czy dla dużych wartości n ostatnia równość jest prawdą.



Na powyższym wykresie można zauważyć, że wartości $F(n)$ dążą do pewnej stałej dla dużych n. Można z tego wywnioskować, że asymptotyczne tempo wzrostu błędu względnego przy sumowaniu algorytmem Kahana wynosi $\mathcal{O}(n2^{-2t})$.

6 Literatura

1. `https://en.wikipedia.org/wiki/Kahan_summation_algorithm`
2. Rafał Nowak, *Analiza numeryczna 1. Analiza błędów*, Wrocław 2017
3. Mateusz Hazy, *Notatki do wykładu z Analizy Numerycznej*, Wrocław 2017