MOWNIT lab1

Karol Hamielec

3/10/2020

zad.1 Sumowanie liczb pojedynczej precyzji

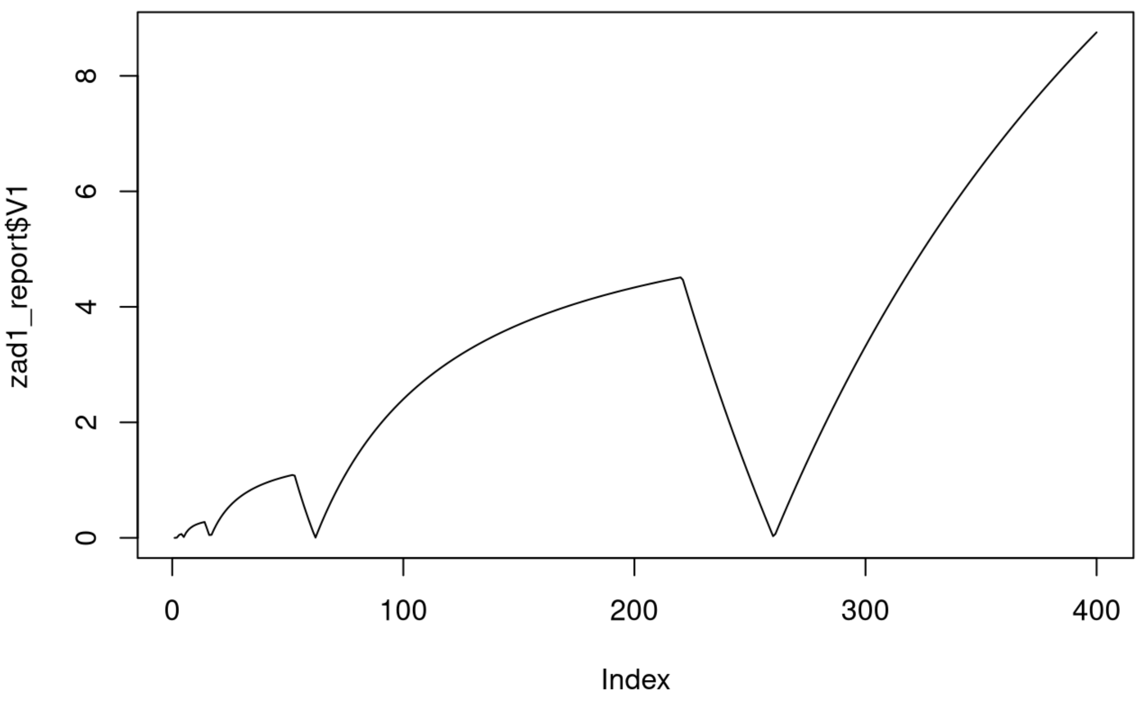
2.

bezwzględny: 175874.000000 względny: 8.793699%

Błąd względny wynika z błędu reprezentacji, ponieważ z każdą iteracją błąd względny jest przemnażany, więc na końcu wynosi N*(błąd) oraz na końcu wynika z dodawania bardzo odległych od siebie liczb (do bardzo dużej liczby dodajemy bardzo małą, przez co błąd reprezentacji wyniku jest większy)

3.

```
zad1_report <- read.table("./zad1_report.txt", header=FALSE)
plot(zad1_report$V1, type="l")
```



5.

bezwzgledny: 0.000000 wzgledny: 0.000000%

Błąd zmalał praktycznie do zera (nie wiemy czy do zera - możliwy niedomiar). Błąd zmalał, ponieważ nie wykonujemy operacji na liczbach odległych od siebie

6.

```
sumowanie normalne:
na koncu 2175874.000000
bezwzgledny: 175874.000000
wzgledny: 8.793699%
0,27s user 0,04s system 99% cpu 0,306 total


sumowanie rekurencyjne:
na koncu 2000000.000000
bezwzgledny: 0.000000
wzgledny: 0.000000%
0,33s user 0,03s system 99% cpu 0,363 total
```

zad.2 Algorytm Kahana

1. i 3.

```

sumowanie normalne:
na koncu 2175874.000000
bezwzgledny: 175874.000000
wzgledny: 8.793699%
0,27s user 0,04s system 99% cpu 0,306 total


sumowanie rekurencyjne:
na koncu 2000000.000000
bezwzgledny: 0.000000
wzgledny: 0.000000%
0,33s user 0,03s system 99% cpu 0,363 total


sumowanie Kahana:
na koncu 1999999.875000
bezwzgledny: 0.125000
wzgledny: 0.000006%
0,33s user 0,00s system 99% cpu 0,332 total
```

2.

W zmiennej `err` trzymamy błąd poprzedniego obliczenia. Służy nam do korekcji błędu przy kolejnym sumowaniu.

zad.3 Suma szeregu

zad3.1.1 - sumowanie normalne

zad3.1.2 - sumowanie normalne wstecz

zad3.4.1 - sumowanie Kahana

zad3.2.1 - sumowanie normalne (double)

zad3.2.2 - sumowanie normalne wstecz (double)

zad3.4.2 - sumowanie Kahana (double)

Niestety nie widać żadnej różnicy pomiędzy różnymi algorytmami sumowania. Ciekawy natomiast jest fakt, że nie widać różnicy w sumie dla różnych wartości **n**, może to wynikać z faktu, że wyrazu ciągu dla $n > 50$ są mniejsze od ϵ maszynowego

```
zad3.1.1
n: 50 sum: 0.5
zad3.1.2
n: 50 sum: 0.5
zad3.4.1
n: 50 sum: 0.5

zad3.1.1
n: 100 sum: 0.5
zad3.1.2
n: 100 sum: 0.5
zad3.4.1
n: 100 sum: 0.5

zad3.1.1
n: 200 sum: 0.5
zad3.1.2
n: 200 sum: 0.5
zad3.4.1
n: 200 sum: 0.5

zad3.1.1
n: 500 sum: 0.5
zad3.1.2
n: 500 sum: 0.5
zad3.4.1
n: 500 sum: 0.5

zad3.1.1
n: 800 sum: 0.5
zad3.1.2
n: 800 sum: 0.5
zad3.4.1
n: 800 sum: 0.5


-----------DOUBLE-------------

zad3.2.1
n: 50 sum: 0.5
zad3.2.2
n: 50 sum: 0.5
zad3.4.2
n: 50 sum: 0.5

zad3.2.1
n: 100 sum: 0.5
zad3.2.2
n: 100 sum: 0.5
zad3.4.2
n: 100 sum: 0.5

zad3.2.1
n: 200 sum: 0.5
```

```
zad3.2.2
n: 200 sum: 0.5
zad3.4.2
n: 200 sum: 0.5

zad3.2.1
n: 500 sum: 0.5
zad3.2.2
n: 500 sum: 0.5
zad3.4.2
n: 500 sum: 0.5

zad3.2.1
n: 800 sum: 0.5
zad3.2.2
n: 800 sum: 0.5
zad3.4.2
n: 800 sum: 0.5
```

zad.4 Epsilon maszynowy

```
epsilon for float: 1.19209e-07
epsilon for double: 2.22045e-16
```

Wartości zgadzają się zarówno dla typu `float` jak i `double`

zad.5 Algorytm niestabilny numerycznie

Pierwiastki równania kwadratowego:

Przy obliczaniu pierwiastków i korzystaniu z równań:

$\frac{b-\sqrt{b^2-4ac}}{2a}$ oraz $\frac{b+\sqrt{b^2-4ac}}{2a}$

Jeśli $b^2 >> 4ac$ to $\sqrt{b^2-4ac} \approx \sqrt{b^2} \approx |b|$

W pierwszym przypadku jeśli $b > 0$ to odejmujemy dwie prawie takie same liczby i narażamy się na błąd obcięcia

W drugim przypadku, jeśli $b < 0$ to wystąpi analogiczna sytuacja

Rozwiązaniem tego problemu może być zastosowanie wzorów Viète'a i korzystając z faktu że $x1x2 = \frac{c}{a}$ obliczyć drugi pierwiastek trójmianu, a pierwszy obliczyć w zależnosi od relacji $b > 0$ wybierając jeden ze sposobów $\frac{b-\sqrt{b^2-4ac}}{2a}$ lub $\frac{b+\sqrt{b^2-4ac}}{2a}$ tak, aby uniknąć błędu obcięcia

normal