Parcijalne sume harmonijskog niza u programskom jeziku C

Uršan, Josip

Undergraduate thesis / Završni rad

2019

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:363572

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-19**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

Parcijalne sume harmonijskog niza u programskom jeziku C

Završni rad

Josip Uršan

Osijek, 2019

SADRŽAJ

1. UVOD 1

1.1. Zadatak završnog rada 1

2. IMPLEMENTACIJA ALGORITMA 2

2.1. Algoritam 3

2.2. Formule za izračun parcijalne sume 5

2.3. Funkcije za izračun broja elemenata 7

3. REZULTATI 10

3.1. Rezultati C-a 10

3.1.1. Iteracija 10

3.1.2. Izračun parcijalnih suma formulama 15

3.1.3. Izračun broja elemenata 19

3.2. Usporedba rezultata C-a i Python-a 27

3.3. Usporedba integracije i iteracije 32

4. ZAKLJUČAK 34

LITERATURA 35

Sažetak 36

Abstract 37

Životopis 38

PRILOZI 39

C kod 39

Datoteka source.c : 39

Datoteka functions.c: 40

Datoteka header.h: 42

Python kod 43

Funkcije 43

Glavni dio koda 46

Specifikacije računala 47

Tablice 48

Tablica C rezultata 48

Tablice Python rezultata 56

1. UVOD

Harmonijski red ima raznolike primjene, od kojih je među najpoznatijima primjena u glazbi. Na primjer, trzanjem žice gitare ostvaruje se određeni ton, ali uz taj osnovni ton javljaju se podtonovi i nadtonovi, koji zapravo u prosjeku daju osnovni ton. Ti nadtonovi i podtonovi su cjelobrojni višekratnici osnovne frekvencije promatrane žice. Ova pojava u glazbi direktno je povezana i sa Fourierovom analizom.

Cilj završnog rada jest prikazati neke od metoda izračuna parcijalne sume harmonijskog niza, te postaviti temelj za poboljšavanje ostvarenih rezultata u budućnosti.

Prvo poglavlje sadrži opis zadatka završnog rada.

U drugom poglavlju predstavljen je i objašnjen korišteni algoritam, te su prikazane i objašnjene formule za izračun parcijalne sume i za izračun broja elemenata.

Treće poglavlje sadrži rezultate ostvarene izvođenjem koda, objašnjenja tih rezultata, i usporedbu rezultat ostvarenih C-om i Python-om.

U četvrtom poglavlju nalazi se kratki osvrt na cijeli rad i postignute rezultate, te ideje za poboljšanja ostvarenih rezultata.

1.1. Zadatak završnog rada

Zadatak ovog završnog rada je demonstracija razlika u preciznosti različitih algoritama za ostvarivanje željene parcijalne sume, usporedba istih algoritama u C-u i Python-u, te demonstracija algoritama za izračun broja elemenata potrebnih za dostizanje određene parcijalne sume.

2. IMPLEMENTACIJA ALGORITMA

Harmonijski red je beskonačni red koji divergira, ali njegov opći član reda konvergira nuli.

$$\sum_{n=1}^{\infty} \frac{1}{n} \tag{2-1}$$

Divergenciju harmonijskog reda prvi je dokazao Nicole Oresme u 14. stoljeću, nakon čijeg dokaza su slijedili dokazi Pietra Mengoli-a, Johanna Bernoulli-a, i Jacoba Bernoulli-a iz 17. stoljeća.

Na prvi pogled, harmonijski red može djelovati kao konvergentan red zbog njegovog oblika koji navodi na intiutivnu pomisao da cijeli red konvergira prema nuli. Ipak, ta konvergencija ne vrijedi za cijeli red, već samo za općeg član reda. Prvi dokaz divergentnosti harmonijskog reda dao je Nicole Oresme u 14. stoljeću :

$$\sum_{n=1}^{\infty} \frac{1}{n} = 1 + \frac{1}{2} + \left(\frac{1}{3} + \frac{1}{4}\right) + \left(\frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{8}\right) + \ldots > 1 + \frac{1}{2} + \frac{1}{2} + \frac{1}{2} + \ldots \tag{2-2}$$

[1]

Oresme je grupirao članove harmonijskog reda obzirom na potencije 2^n, izuzevši prva dva člana. Svaka skupina 2^n članova ima sumu veću od $\frac{1}{2}$. Obzirom da bekonačna suma polovina divergira, Oresme je zaključio kako i harmonijski red divergira.

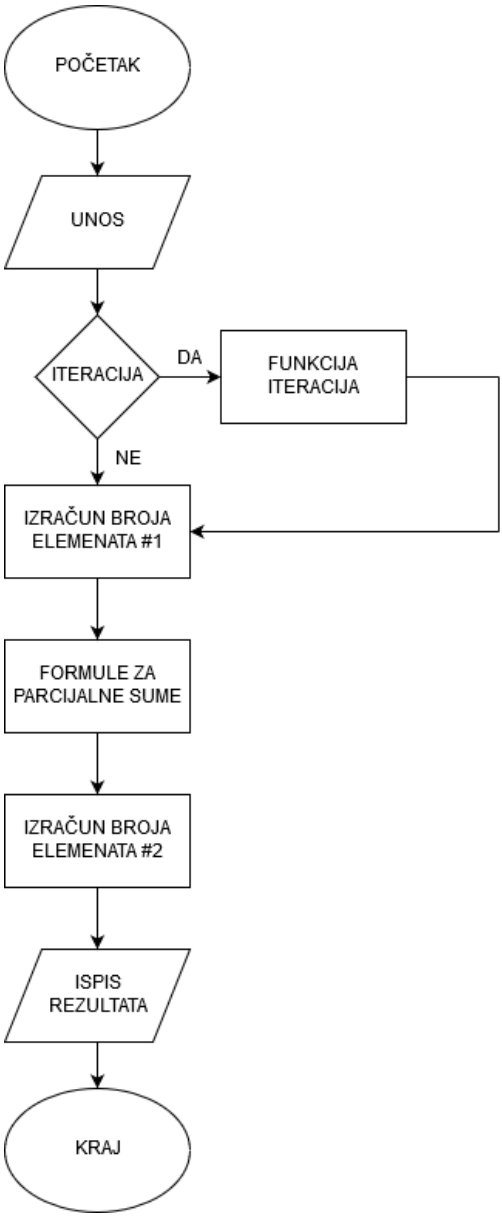
U završnom radu demonstrirani su algoritmi za dostizanje željene parcijalne sume harmonijskog reda. Glavni problem predstavlja efikasnost različitih algoritama. Uporaba iteracije je najjednostavnija metoda, ali ujedno i najskuplja u pogledu vremena i računalnih resursa. Primjena iterativne metode je vrlo vremenski neefikasna za parcijalne sume veće od 22. Uz iterativnu metodu, postoji niz jednadžbi koje su određene numerički, ili na temelju poboljšanja prethodno poznatih jednadžbi, te koje omogućuju približni izračun vrijednosti željene parcijalne sume uz iznimno dobru točnost.

Uz različite algoritme za dostizanje određene parcijalne sume, prikazani su i izvedeni algoritmi koji omogućuju određivanje potrebnog broja elemenata za dostizanje željene sume. Izračun potrebnog broja elemenata za određenu parcijalnu sumu također služi i kao potvrda isporavnosti različitih metoda izračuna parcijalne sume.

Također su prikazane razlike u efikasnosti istog koda izvedenog u C-u i Python-u, tj. prikazane su razlike u vremenima izvođenja koda i preciznosti izračuna različitih vrijednosti parcijalnih suma.

2.1. Algoritam

Za potrebu ovog završnog rada razvijen je računalni kod u programskim jezicima C i Python. Cjeloukupan kod nalazi se u prilogu, a u ovom poglavlju predstavljen je i objašnjen algoritam izračuna parcijalne sume i broja potrebnih članova harmonijskog niza za izračun željene parcijalne sume.



Slika 2.1 – algoritam programskog koda

Od korisnika se, na početku izvođenja programa, zahtjeva unos parcijalne sume koju želi postići. Nakon toga se zahtjeva odabir korisnika želi li koristiti iteraciju, ili ne želi, obzirom da funkcija koja

izvodi iteraciju može trajati iznimno dugo ako se unese dovoljno velika željena suma. Ukoliko se odabere izvođenje funkcije iteracije, ona se izvodi, i nakon nje slijedi ostatak algoritma. Ukoliko se odbije izvođenje funkcije iteracije, program nastavlja dalje.

Slijedi funkcija koja izračunava broj elemenata za dostizanje željene sume. Nadalje, izvodi se funkcija koja koristi gotove formule za izračun parcijalne sume, nakon koje slijedi druga funkcija koja izračunava potreban broj elemenata drugačijom metodom od prve funkcije koja obavlja istu zadaću.

2.2. Formule za izračun parcijalne sume

Korišteno je pet različitih formula koje omogućuju brzi izračun parcijalne sume na temelju potrebnog broja elemenata. Postoji niz formula koje pružaju vrlo slične rezultate, a većinom su dobivene računalnim aproksimacijama ili korištenjem pojednostavljenih principa poput Ramanujan-ove sumacije ili Euler-Maclaurin-ove formule.

Sljedeće formule su korištene u radu :

$$S_1(n) \approx ln(n) + \frac{1}{n} + \gamma \cdot \left(1 + ln\left(\frac{n}{n+1}\right)\right) \tag{2-3}$$

$$S_2(n) \approx ln(n+1) + \gamma \cdot \left(1 + ln\left(\frac{n}{n+1}\right)\right) \tag{2-4}$$

$$S_3(n) \approx ln(n+1) + \gamma \cdot \left(1 + \frac{50}{51n} + ln\left(\frac{n - \frac{\gamma}{10}}{n + \frac{\gamma}{10}}\right)\right) \tag{2-5}$$

$$S_4(n) \approx ln(n) + \gamma \tag{2-6}$$
[2]

$$S_5(n) \approx ln(n) + \gamma + \frac{1}{2n} - \frac{1}{12n^2} + \frac{1}{120n^4} - \frac{1}{252n^6} \tag{2-7}$$
[3]

U svim izrazima oznaka S_x označava sumu, indeks x označava redni broj formula, n označava broj elemenata, a γ označava Euler-Mascheronijevu konstantu.

Formula S_5 predstavlja poseban slučaj Euler-Maclaurinove integracijske formule. Euler-Maclaurinova formula koristi se, među ostalim, i za određivanje konačnih parcijalnih suma.

Općenito, za procjenu pacijalnih suma se koriste različite asimptotske procjene promatranog niza elemenata. Euler-Maclaurinova formula je jedna od formula iz koje se izvode takve procjene.

2.3. Funkcije za izračun broja elemenata

Za izračun broja elemenata potrebnih za dostizanje određene parcijalne sume, koriste se tri različite metode. Iterativna funkcija ima najjednostavniju metodu. Za svaku izvedenu iteraciju inkrementira se brojač koji vodi brigu o broju iteracija, a time ujedno i o broju elemenata.

```
1       double* iteracija(int *trazena_suma) {
2       int i = 1;
3       double *sum = 0;
4
5       sum = (double*)calloc(1, sizeof(double));
6
7       printf("Iteriram...\n");
8       while (*sum < *trazena_suma)
9       {
10              *sum += 1.0 / (double)i;
11              i++;
12      }
13      printf("Potreban broj elemenata: %d\n", i);
14      return sum;
15      }
```

Slika 2.2. – funkcija iteracija

U 11. redu (slika 2.1.) vidljivo je inkrementiranje varijable koja ujedno služi kao brojač elemenata potrebnih za dostizanje određene parcijalne sume.

Uz funkciju iteracije, koja za svaku iteraciju inkrementira brojač, postoje i dvije izvedene formule koje izračunavaju broj elemenata koji sudjeluju u promatranoj parcijalnoj sumi. Prva takva formula je izvedena na temlju izraza (2-8) :

$$\int_b^a \frac{1}{x}\,dx = \ln(a) - \ln(b) \tag{2-8}$$

Donja granica, odnosno b, u ovom slučaju je proizvoljno uzeta kao 1.

$$\gamma = \lim_{n\to\infty}\left(\sum_{k=1}^{n}\frac{1}{k} - ln(n)\right) = \lim_{n\to\infty}\left(H_n - ln(n)\right) \tag{2-9}$$

[4]

U izrazu (2-9), H_n označava parcijalnu sumu prvih n članova harmonijskog niza.

Iz izraza (2-9) izlučen je n, te je na temelju metode pokušaja i pogreške utvrđeno da oduzimanje γ konstane u eksponentu daje točniji rezultat, te je izveden izraz (2-10) :

$$n = e^{H_n - \gamma + ln(1)} \tag{2-10}$$

Implementacija izraza (2-10) prikazana je na slici 2.3.

```
1       double* pronadi_broj_elemenata(int *trazena_suma) {
2               int i = 1;
3               float e = 2.718281828459045;
4               float EM_constant = 0.577215664901532;
5               double *broj_potrebnih elemenata = 0;
6
7               broj_potrebnih_elemenata = (double*)calloc(1, sizeof(double));
8
9               *broj_potrebnih_elemenata = pow((double)e, ((*trazena_suma - EM_constant) +
10      log(1.0)));
11
12      return broj_potrebnih_elemenata;
13      }
```

Slika 2.3. – prva funkcija za pronalazak broja elemenata

Druga izvedena formula za pronalazak broja elemenata koji čine određenu parcijalnu sumu izvedena je iz izraza (2-3). Uzimajući u obzir činjenicu da se n, tj. element koji označava broj elemenata, nalazi u izrazu (2-3) i kao slobodan član, i kao član pod prirodnim logaritmom, prilikom izvođenja izraza za n, on će svakako ostati sa obje strane jednakosti. Zato što ostaje s obje strane jednakosti, ova formula će se rješavati iterativno.

Izvođenjem dobijamo izraz (2-11) :

$$n = \frac{\left(e^{S-\gamma} - e^{\frac{1}{n}}\right) \cdot (n+1)^{\gamma}}{n^{\gamma}} \tag{2-11}$$

Oznaka *S* u izrazu (2-11) označava predviđenu sumu.

Implementacija izraza (2-11) prikazana je na slici 2.4.

```
1       void pronadi_broj_elemenata_iterativno(int *trazena_suma) {
2               int i = 1, broj_iteracija = 0;
3               double br_cl = 1;
4               double rjesenje_iteracije = 0;
5               double e = 2.718281828459045;
6               double EM_const = 0.577215664901532;
7               double suma_clanova_tijekom_iteracije = 0;
8
9               double doljnja_granica = 0.9*(*trazena_suma);    //0.99999
10              double gornja_granica = 1.01*(*trazena_suma); //1.00001
11
12              printf("\n\Izvrsavam iteraciju s trazenom varijablom s obje strane jednakosti
13      : \n");
14              do
15              {
16                      rjesenje_iteracije = (((pow(e, *trazena_suma - EM_const) - pow(e, (1
17                      / br_cl))) * (pow(br_cl + 1, EM_const)))) /
18                      (pow(br_cl, EM_const));
19
20                      br_cl = rjesenje_iteracije;
21
22                      suma_clanova_tijekom_iteracije =
23                      izracun_parcijalne_sume_formulama(&br_cl);
24                      broj_iteracija++;
25              }while(suma_clanova_tijekom_iteracije < doljnja_granica ||
26                      suma_clanova_tijekom_iteracije > gornja_granica);
27
28              printf("Broj iteracija : %d\n", broj_iteracija);
29              printf("Potreban broj elemenata : %.14lf\n", rjesenje_iteracije);
30      }
```

Slika 2.4. – druga funkcija za pronalazak broja elemenata

3. REZULTATI

U ovom poglavlju prikazane su razlike u vremenima izvođenja istog koda u C-u i u Python-u, moguća odstupanja u ostvarenim rezultatima, te preciznost ostvarenih rezultata. Također su prikazane razlike između izračuna parcijalne sume iteracijom te pokušaja izračuna sume integriranjem površine.

Izračuni parcijalnih suma, i broja njihovih elemenata, su u C-u i Python-u izvršeni za vrijednosti parcijalne sume od 2 do 50. Iteracije se nije provodila za parcijalne sume veće od 28, obzirom da zahtjeva vrlo veliku količinu vremena.

3.1. Rezultati C-a

U ovom potpoglavlju su izneseni i objašnjeni rezultati dobiveni izvođenjem C koda. Kod se u cijelosti nalazi u prilogu, a tablice s potpunim rezultatima također.

3.1.1. Iteracija

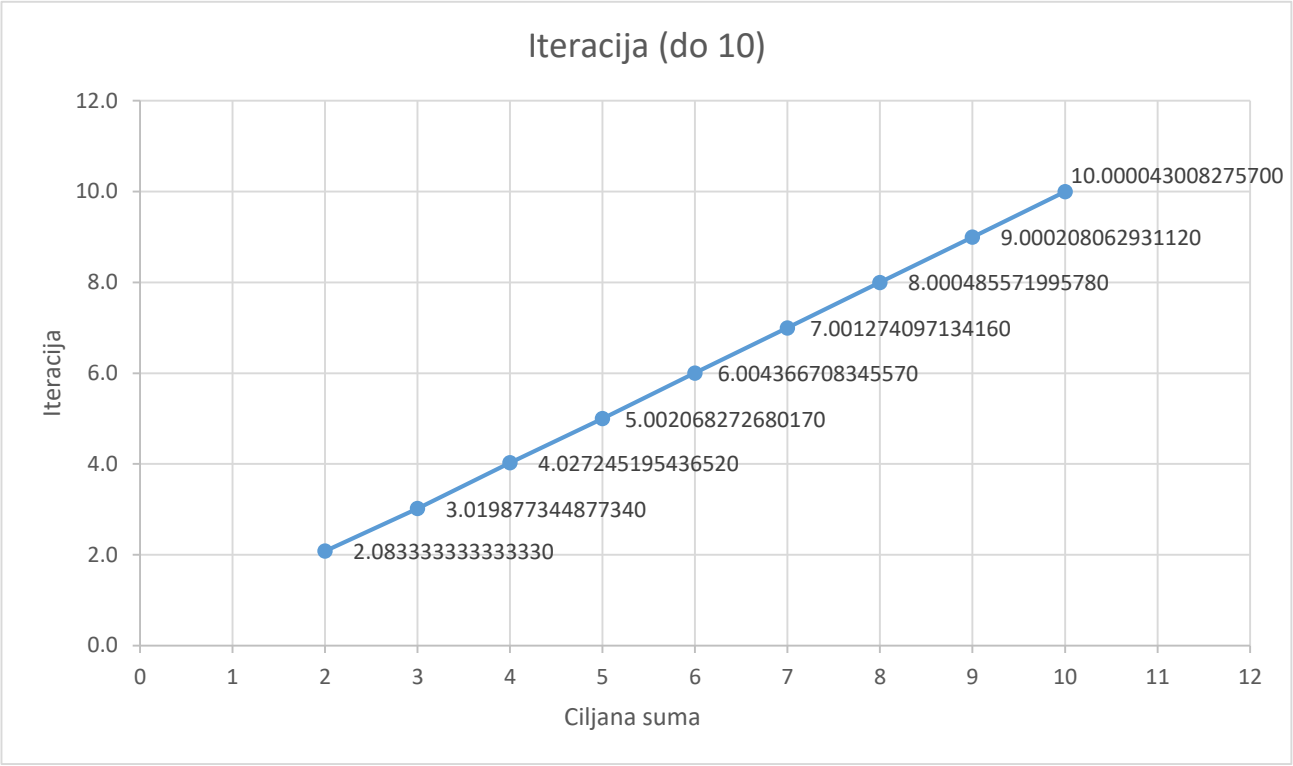
Iteracija je izvođena samo do parcijalne sume iznosa 28 obzirom da se radi o vremenski i sklopovski zahtjevnom procesu. Inicijalno iteracija ima relativno veliku pogrešku, odnosno za sume do 10, odstupanja se javljaju od druge, pa do pete decimale, što je i vidljivo na slici 3.1.

Prosječno odstupanje parcijalnih suma od 2 do 10, koje su dobivene iteracijom, je 0.015433510556632. Prosječno odstupanje za sve parcijalne sume dobivene iteracijom iznosi 0.005145387078340.

Kao što je vidljivo u tablici 3.1., iterativno izračunavanje parcijalne sume ostvarivalo je sve veću preciznost što je tražena parcijalna suma bila veća. Najveća greška u iteraciji javlja se prilikom izračuna prve parcijalne sume. Tražena je bila parcijalna suma iznosa 2, a iteracijom je dobivena vrijednost $2.8\dot{3}$. Ostvarena je greška od $0.8\dot{3}$. Najmanja greška javlja se prilikom izračuna posljednje parcijalne sume koja se izračunavala iteracijom, odnosno sume iznosa 28. Vrijednost ostvarena iteracijom iznosila je 28.000000000001, odnosno greška je iznosila $1 \cdot 10^{-12}$.

Ciljana suma	Izračun parcijalne sume iteracijom
2	2.083333333333330
3	3.019877344877340
4	4.027245195436520
5	5.002068272680170
6	6.004366708345570
7	7.001274097134160
8	8.000485571995780
9	9.000208062931120
10	10.000043008275700
11	11.000017708636400
12	12.000003051665600
13	13.000001229480900
14	14.000001362053200
15	15.000000378267200
16	16.000000095452500
17	17.000000014847700
18	18.000000003717900
19	19.000000009732000
20	20.000000001618200
21	21.000000000402300
22	22.000000000140300
23	23.000000000035000
24	24.000000000042500
25	25.000000000004700
26	26.000000000005300
27	27.000000000002800
28	28.000000000001000

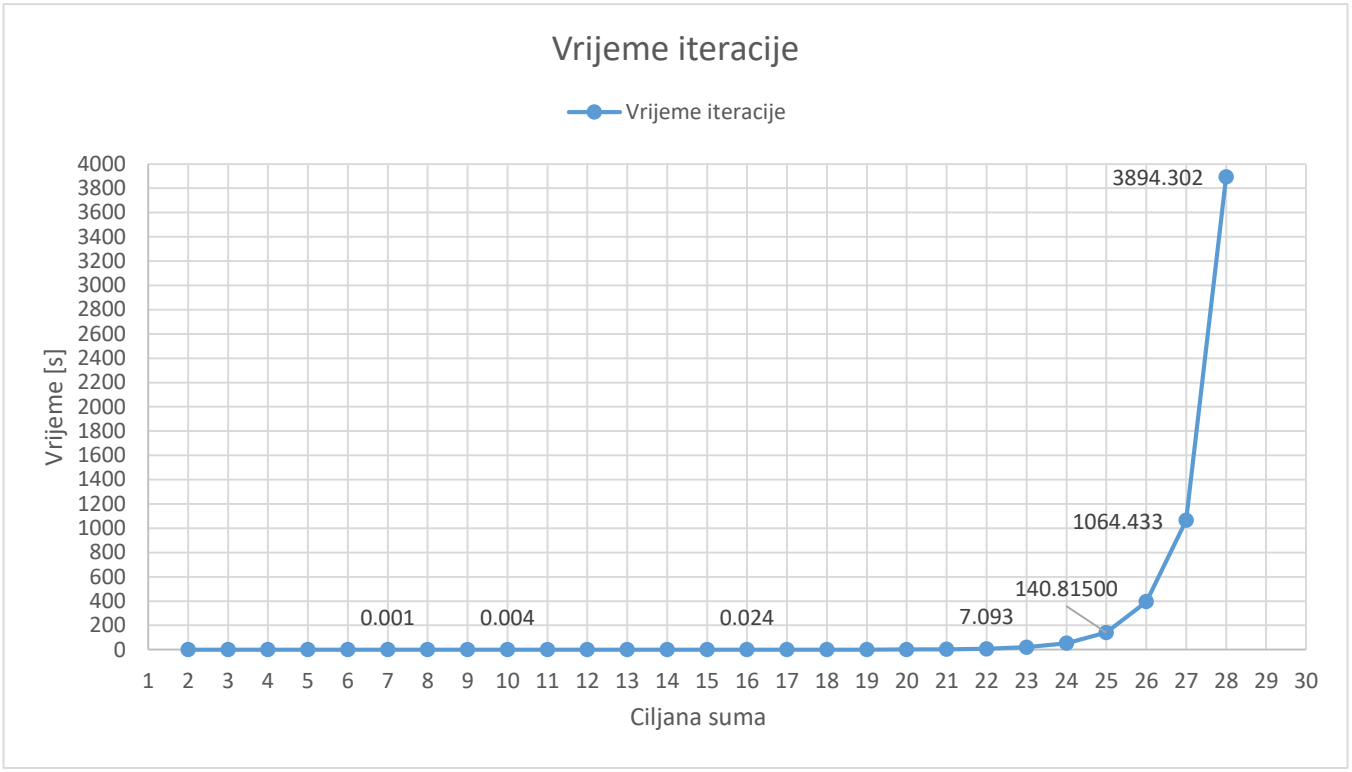
Tablica 3.1. – rezultati izvođenja iteracije



Slika 3.1. – graf parcijalnih suma iteracijom do parcijalne sume 10

Vrijeme izvođenja iteracije za svaku parcijalnu sumu bio je iznimno zanimljiv faktor. Očekivano je da će vrijeme u određenom trenutku počet ostvarivat velike pomake u vremenu potrebnom za izračun parcijalne sume. Navedena pretpostavka se i ostvarila, što je prikazano na slici 3.2.

Na slici 3.2. prikazano je nekoliko karakterističnih vrijednosti vremena izvođenja iteracije koja prikazuju velike vremenske pomake između pojedinih iznosa. Omjeri između vremena su varirali oko sličnih iznosa, ali obzirom da se svako slijedeće vrijeme izvođenja povečavalo za oko 2.5 puta u odnosu na prethodno, brzo su dostignute velike vrijednosti.



Slika 3.2. – vremena izvođenja iteracije

Vrijeme potrebno za izračun parcijalne sume iteracijom bilo je vrlo stabilno do parcijalne sume vrijednosti 14. U većini slučajeva je ono iznosilo 0.001s, ali prilikom izračuna parcijalne sume do 15, javlja se prvi veliki skok. Vrijeme potrebno za izračun parcijalne sume harmonijskog niza do 15 iznosilo je 0.01s, dok je vrijeme za izračun parcijalne sume do 14 iznosilo 0.004s. Izračun parcijalne sume do 15 trajao je 2.5 puta više nego izračun parcijalne sume do 14.

Vrlo sličan odnos između vremena javlja se i u daljnjim slučajevima. Vrijeme potrebno za ostvarivanje parcijalne sume iznosa 16 iznosilo je 0.024s. U odnosu na vrijeme koje je potrebno da bi se dostigla parcijalna suma 15, vrijeme se povečalo 2.4 puta.

Slični omjeri javljaju se u omjerima svih vremena. Pregled svih omjera dan je u tablici 3.1. U tablici 3.1. Sx predstavlja veću sumu, a Sy predstavlja manju, pa se tako u slučaju S15/S14 radi o omjeru vremena iteracije potrebnog da se ostvari suma do 15 i vremena iteracije potrebnog da se ostvari parcijalna suma do 14.

Uz pretpostavku da će se ovi omjer održati, tj. da će se ostvarivati slični pomaci svakog idućeg vremena u odnosu na prethodno, možemo pokušati približno izračunati vrijeme potrebno za izračun parcijalne sume harmonijskog niza do 29.

Ako izračunamo prosječan iznos omjera, dobijamo iznos vrijednost 2.702. Obzirom da je za izračun do 28 bilo potrebno 3894.302s, znamo da bi za izračun do parcijalne sume iznosa 29 bilo potrebno ~10522.404s, odnosno oko 175 minuta.

Sx/Sy	Omjer
S15/S14	2.5
S16/S15	2.4
S17/S16	2.833
S18/S17	2.132
S19/S18	2.524
S20/S19	2.606
S21/S20	3.332
S22/S21	2.23
S23/S22	2.769
S24/S23	2.676
S25/S24	2.677
S26/S25	2.807
S27/S26	2.692
S28/S27	3.658

Tablica 3.2 – omjeri povećanja vremena između dvaju promatranih vremena

3.1.2. Izračun parcijalnih suma formulama

Za brzi izračun parcijalnih suma korištene su formule koje su navedene u poglavlju 2.2. pod oznakama izraza (2-3), (2-4), (2-5), (2-6), (2-7). Izraz (2-3) predstavlja formulu 1, izraz (2-4), predstavlja formulu 2, izraz (2-5) predstavlja formulu 3, izraz (2-6) predstavlja izraz 4, te izraz (2-7) predstavlja formulu 5 u tablici 3.3. Djelomični rezultati dani su u tablici 3.3., dok se cjeloviti rezultati nalaze u prilogu.

Formulama su izračunate sve parcijalne sume, odnosno od 2 do 50. Najveće odstupanje od ciljane vrijednosti je 0.12418706628079, i ono je ostvareno prilikom izračuna prve sume, dok je najmanje odstupanje bilo 0, te je ono ostvareno prilikom izračuna sume iznosa 50 svim formulama.

Kod svake formule javlja se trend smanjivanja pogreške. Što je veća tražena parcijalna suma, to je manje odstupanje od ciljane vrijednosti. Zanimljivo je da formula 4, odnosno izraz (2-6), daje točan izračun za sve tražene parcijalne sume. Ujedno se radi i o najjednostavnijem izrazu koji u sebi sadrži samo prirodni logaritam broja elemenata, i Euler-Mascheroni konstantu.

Korištene formule imaju sljedeća prosječna odstupanja od ciljanih vrijednosti :

Formula 1 – 0.00636399496385

Formula 2 – 0.00531372027127

Formula 3 – 0.00699514278886

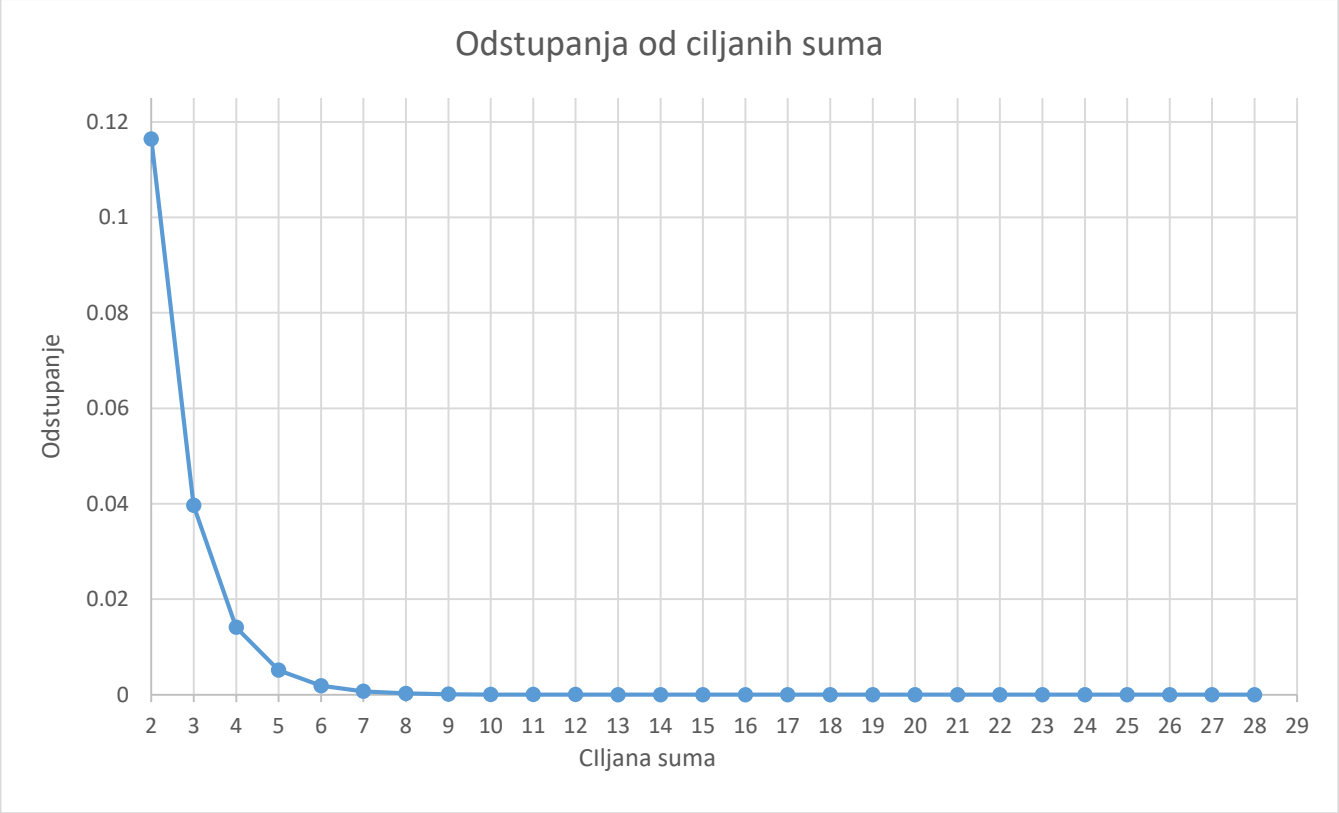
Formula 4 – 0

Formula 5 – 0.00661034475057

Ukupno prosječno odstupanje je 0.00505664055491.

Ciljana suma	Formula 1	Formula 2	Formula 3	Formula 4	Formula 5
2	2.11639146630194	2.09130082724987	2.12418706628079	2.000000	2.11570655494791
3	3.03963367784228	3.03592008928571	3.04567666260775	3.000000	3.04368243838014
4	4.01409245559630	4.01357167072667	4.01622206897784	4.000000	4.01622206897784
5	5.00511497351863	5.00504353523476	5.00618102829194	5.000000	5.00598838440665
6	6.00187213267398	6.00186241586877	6.00227384997407	6.000000	6.00220579433737
7	7.00068741626327	7.00068609879398	7.00083649950790	7.000000	7.00081184409278
8	8.00025270953974	8.00025253111770	8.00030773054512	8.000000	8.00029871186806
9	9.00009294269851	9.00009291854563	9.00011320768326	9.000000	9.00010989687302
10	10.00003418846610	10.0000341851971	10.0000416467714	10.000000	10.0000404297364
11	11.00001257679500	11.0000125763526	11.0000153209899	11.000000	11.0000148733955
12	12.00000462668490	12.0000046266250	12.0000056362770	12.000000	12.0000054716336
13	13.00000170205420	13.0000017020461	13.0000020734704	13.000000	13.0000020129038
14	14.00000062614960	14.0000006261485	14.0000007627871	14.000000	14.0000007405062
15	15.00000023034740	15.0000002303473	15.0000002806137	15.000000	15.0000002724170
16	16.00000008474000	16.0000000847400	16.0000001032320	16.000000	16.0000001002166
17	17.00000003117410	17.0000000311741	17.0000000379769	17.000000	17.0000000368676
18	18.00000001146830	18.0000000114683	18.0000000139709	18.000000	18.0000000135628
19	19.00000000421890	19.0000000042189	19.0000000051396	19.000000	19.0000000049894
20	20.00000000155200	20.0000000015520	20.0000000018907	20.000000	20.0000000018355
21	21.00000000057090	21.0000000005709	21.0000000006955	21.000000	21.0000000006752
22	22.00000000021000	22.0000000002100	22.0000000002558	22.000000	22.0000000002484
23	23.00000000007720	23.0000000000772	23.0000000000941	23.000000	23.0000000000913
24	24.00000000002840	24.0000000000284	24.0000000000346	24.000000	24.0000000000336
25	25.00000000001040	25.0000000000104	25.0000000000127	25.000000	25.0000000000123
26	26.00000000000380	26.0000000000038	26.0000000000046	26.000000	26.0000000000045
27	27.00000000000140	27.0000000000014	27.0000000000017	27.000000	27.0000000000016
28	28.00000000000050	28.0000000000005	28.0000000000006	28.000000	28.0000000000006

Tablica 3.3. – djelomični rezultati izračuna parcijalnih suma formulama

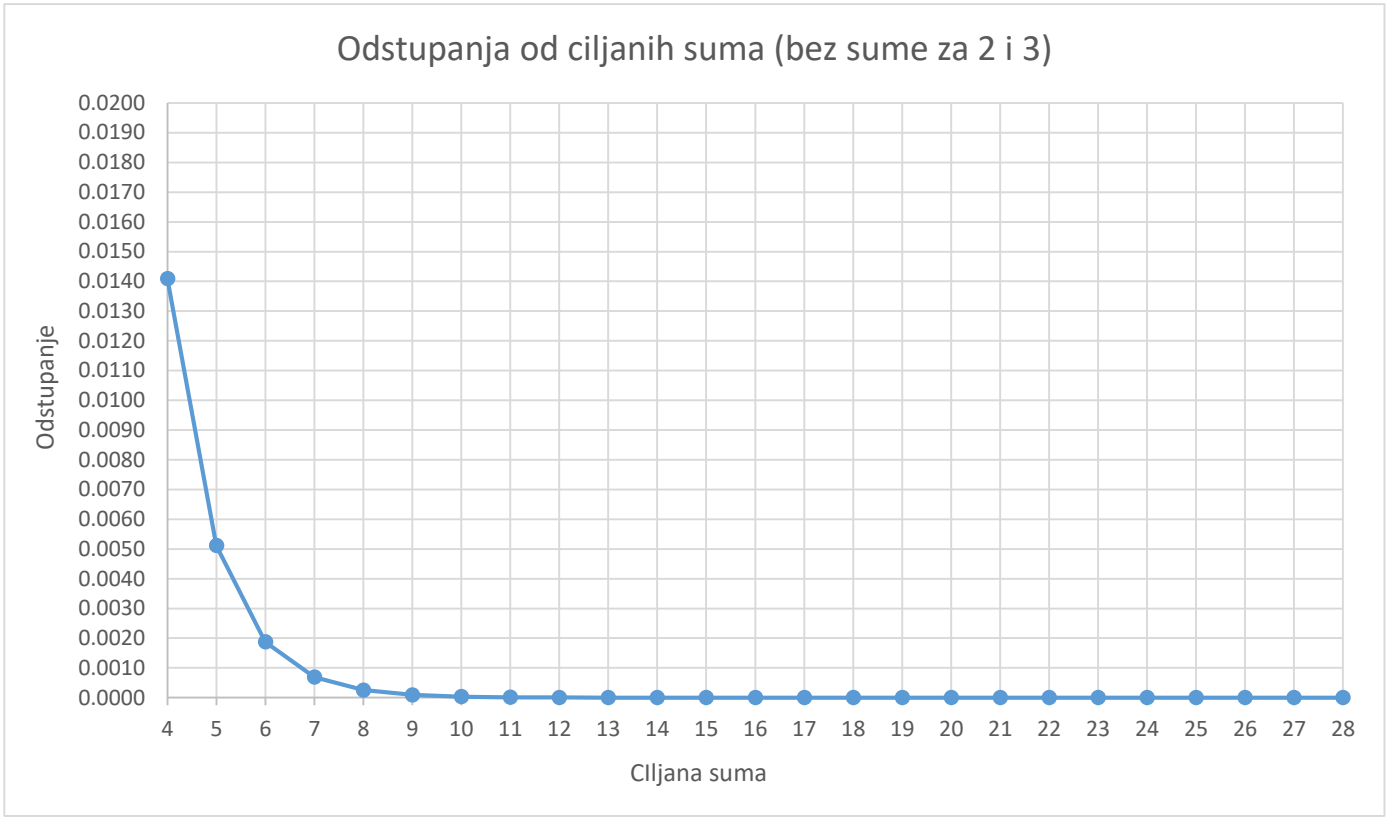


Slika 3.3. – odstupanja izračuna parcijalnih suma formulama

Na slici 3.3. prikazana su odstupanja parcijalnih suma, koje su izračunate formulama, od ciljanih suma. Prikazana su odstupanja za sve parcijalne sume koje se nalaze u tablici 3.3. s djelomičnim rezultatima.

Vidljivo je da osim prve dvije parcijalne sume, nigdje drugdje na krivulji ne postoje značajni pomaci u odstupanju, odnosno za parcijalne sume iznad 9, radi se o odstupanjima koja iza decimalne točke imaju barem četiri nule prije prve znamenke koja nije nula.

Ako uklonimo odstupanja za parcijalne sume 2 i 3, dobijamo graf prikazan na slici 3.4.



Slika 3.4. – odstupanja od ciljanih suma bez odstupanja za parcijalne sume 2 i 3

Krivulja na slici 3.4. i dalje ima vrlo izražen pad u odstupanju od prve do druge, vrijednosti, ali se ipak radi o puno manjim pomacima.

Vremena izvođenja koda koji koristi navedene formule su zanemariva. Radi se o vremenima u rasponu od 0.007s do 0.018s. Uvjerljivo najviše vremena i hardvera zahtjeva izvođenje iteracije.

3.1.3. Izračun broja elemenata

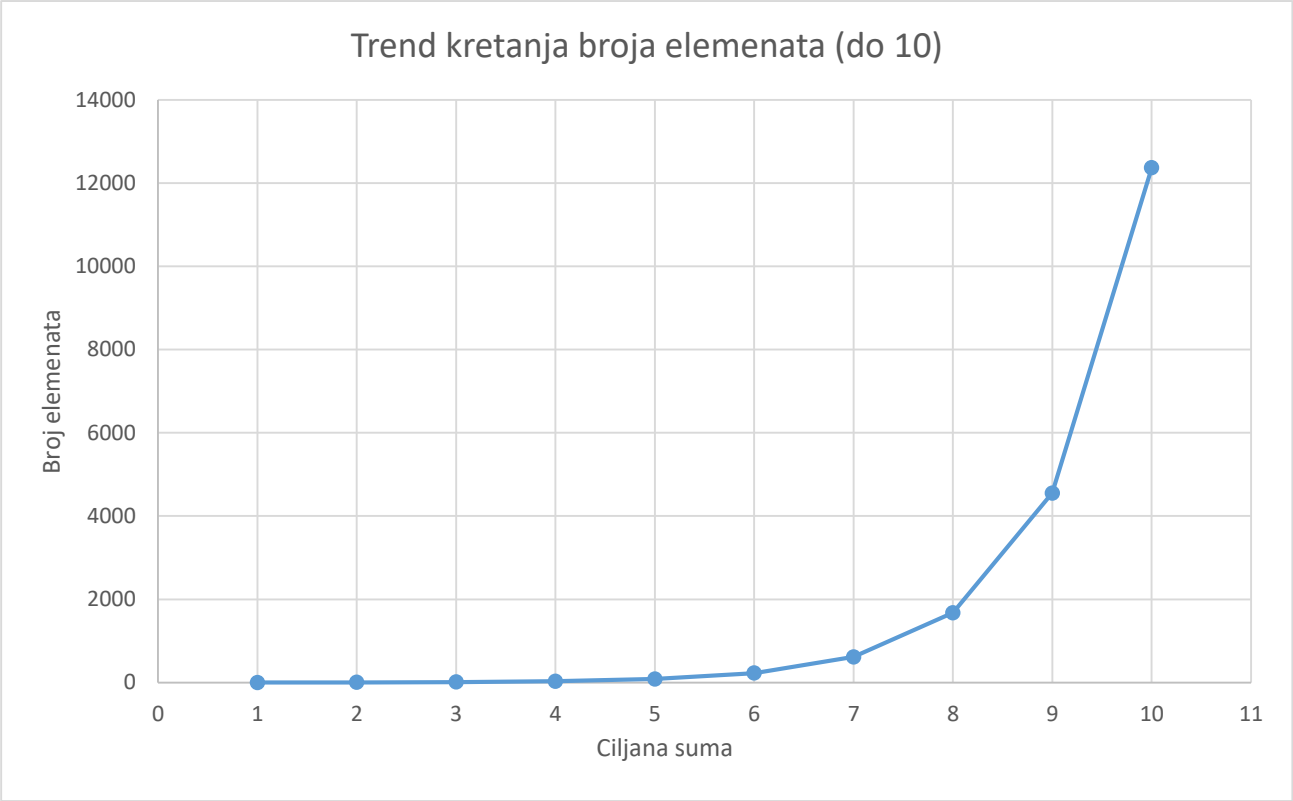
Kao što je i prije navedeno, koriste se dvije metode za izračun broja elemenata potrebnih za izračun određene parcijalne sume. U ovom potpogavlju biti će predstavljeni rezultati tih metoda.

Na temelju izraza (2-10), dobiveni su rezultati prikazani u tablici 3.4. Rezultati nisu prikazani u cjelosti, a cijela tablica nalazi se u prilogu.

Ciljana suma	Potreban broj elemenata (izraz 2-10)
2	4.14865562135235
3	11.27721518805650
4	30.65464912131650
5	83.32797566426260
6	226.50892205044200
7	615.71508679356400
8	1673.68713193903000
9	4549.55331727560000
10	12366.96810995580000
11	33616.90468642540000
12	91380.221138150000
13	248397.1946004020000
14	675213.5803224790000
15	1835420.805719360000
16	4989191.023762610000
17	13562027.298604800
18	36865412.362863100
19	100210580.524620000
20	272400600.059407000
21	740461601.202826000
22	2012783315.221330000
23	5471312310.39170000
24	14872568831.1620000
25	40427833596.25410000
26	109894245428.6630000
27	298723530400.9550000
28	812014744422.0490000

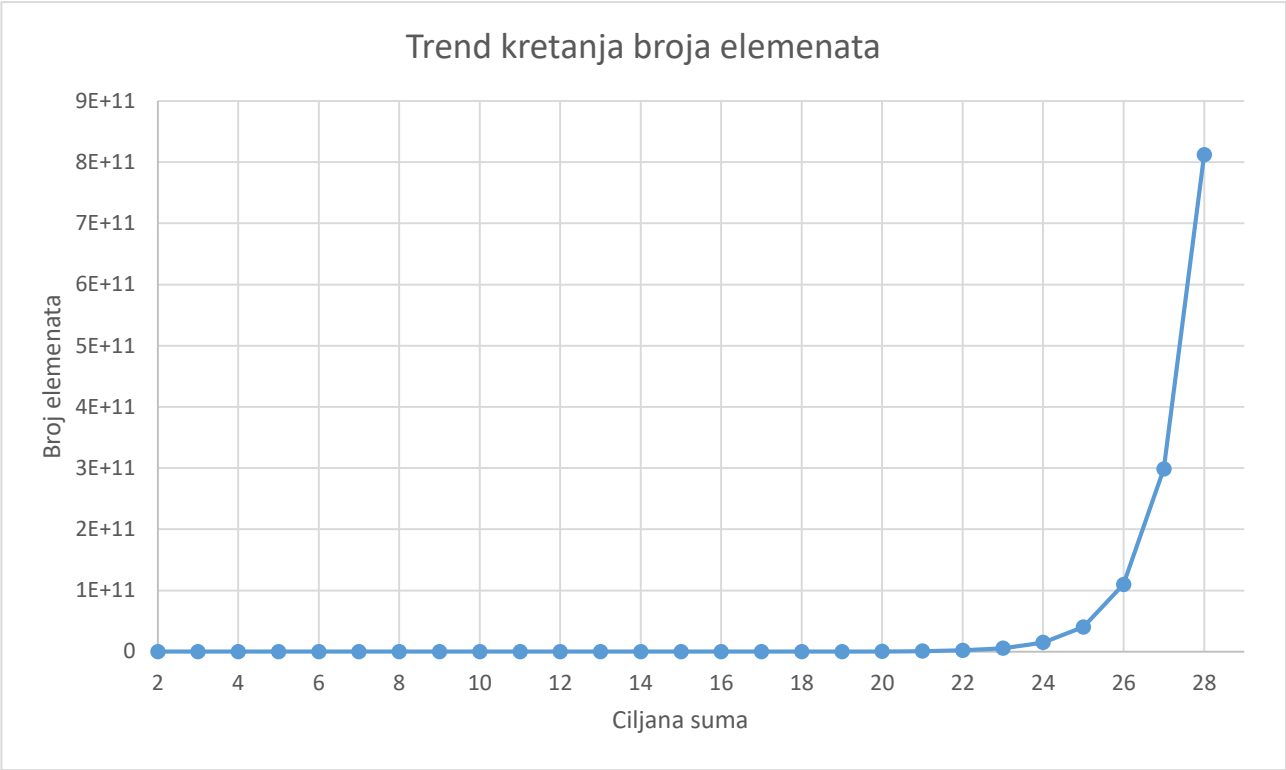
Tablica 3.4. – djelomični rezultati izračuna broja elemenata izrazom (2-10)

Obzirom da se u izrazu (2-10) potencira konstanta e, najčešće nećemo dobiti cijeli broj, što na prvi pogled može biti kontraintuitivno. Obzirom da se radi o zbrajanju onoliko elemenata koliko je potrebno da se dosegne željena parcijalna suma (a ne zbrajanju određenog broja elemenata), jasno je da broj elemenata ne mora nužno biti cijeli broj.



Slika 3.5. – prikaz trenda kretanja broja elemenata za parcijalne sume do 10

Na slici 3.5 prikazan je trend kretanja broja elemenata za parcijalne sume do 10. Zbog sve manjeg doprinosa svakog elementa, vidljiv je iznimno velik porast broja elemenata potrebnih za dostizanje parcijalne sume 10 u odnosu na parcijalnu sumu 1.



Slika 3.6. – prikaz trenda kretanja broja elemenata za parcijalne sume do 28

Na slici 3.6. prikazan je trend kretanja broja elemenata za sve parcijalne sume iz tablice 3.4. Vidljivo je da se, kao i na slici 3.5., nastavlja iznimno brzi rast broja elemenata.

Vrijednosti iz tablice 3.4. iskazuju zanimljivu pravilnost, sličnu onoj koja je uočena kod vremena iteracije. Dijeljenjem broja elemenata potrebnih za sumu do 3 i sumu do 2, dobijamo vrijednost 2.71828182846. Dijeljenjem broja elemenata potrebnih za sumu do 4 i sumu do 3, dobijamo ponovo vrijednost 2.71828182846. Omjeri za ostale sume broja elemenata prikazane su u tablici 3.5.

Nx/Ny	Omjer
N3/N2	2.71828182846
N4/N3	2.71828182846
N5/N4	2.71828182846
N6/N5	2.71828182846
N7/N6	2.71828182846
N8/N7	2.71828182846
N9/N8	2.71828182846
N10/N9	2.71828182846
N11/N10	2.71828182846
N12/N11	2.71828182846
N13/N12	2.71828182846
N14/N13	2.71828182846
N15/N14	2.71828182846
N16/N15	2.71828182846
N17/N16	2.71828182846
N18/N17	2.71828182846
N19/N18	2.71828182846
N20/N19	2.71828182846
N21/N20	2.71828182846
N22/N21	2.71828182846
N23/N22	2.71828182846
N24/N23	2.71828182846
N25/N24	2.71828182846
N26/N25	2.71828182846
N27/N26	2.71828182846
N28/N27	2.71828182846

Tablica 3.5. – omjeri između broja elemenata

U tablici 3.5., N označava broj elemenata, a broj uz N označava na koju parcijalnu sumu se taj broj elemenata odnosi. Omjeri su računati tako da je uzet broj elemenata (n+1) parcijalne sume, i n parcijalne sume.

Dijeljenjem, prema iznad navedenom pravilu, svaki put dobijamo vrijednost koja se do desetog decimalnog mjesta slaže s Euler-ovom konstantom. Svaki sljedeći broj elemenata potreban za ostvarivanje određene parcijalne sume veći je za e konstantu u odnosu na prethodni.

Usporedbom vrijednosti omjera iz tablice 3.5., i omjera iz tablice 3.2., vidljivo je također da omjeri u tablici 3.2. teže prema e konstanti, ali ni s približnom preciznošću koja je vidljiva u tablici 3.5.

Uz izraz (2-10), korišten je i izraz (2-11) koji zahtjeva iterativno izvođenje kako bi se došlo do broja elemenata. U tablici 3.6. nalaze se djelomični rezultati, dok se potpuna tablica nalazi u prilogu.

Ciljana suma	Potreban broj elemenata (izraz 2-11)
2	3.18444642056633
3	10.64926806107240
4	30.03865093415770
5	82.71402350205290
6	225.89552371905000
7	615.10186924947900
8	1673.2635431603300
9	4548.9401863032300
10	12366.3549873918000
11	33616.2915669477000
12	91379.6080198067000
13	248396.5814824760000
14	675212.9672047060000
15	1835420.1926016500000
16	4989190.410644920000
17	13562026.685487100
18	36865411.749745400
19	100210579.911502000
20	272400599.446289000
21	740461600.589709000
22	2012783314.60821000
23	5471312309.77858000
24	14872568830.5489000
25	40427833595.641000
26	109894245428.05000
27	298723530400.3410000
28	812014744421.4360000

Tablica 3.6. – djelomični rezultati izračuna broja elemenata izrazom (2-11)

Obzirom da se u izrazu (2-11) varijabla koju želimo pronaći nalazi s obje strane jednakosti, izraz se rješava iteracijom. Za izračun broja elemenata potrebnih za ostvarivanje ciljane parcijalne sume bile su potrebne dvije iteracije za svaku parcijalnu sumu.

Značajnije razlike između izračuna broja elemenata izrazom (2-10) i (2-11) javljaju se za prvih nekoliko vrijednosti parcijalnih suma, dok se kasnije te razlike javljaju daleko iza decimalne točke.

Omjer koji se javio između broja elemenata koji su izračunati izrazom (2-10) javlja se i kod broja elemenata izračunatih izrazom (2-11). Rezultati su dani u tablici 3.7.

Nx/Ny	Omjer
N3/N2	3.34415049105
N4/N3	2.82072446312
N5/N4	2.75358649373
N6/N5	2.73104262318
N7/N6	2.72294846362
N8/N7	2.72030313483
N9/N8	2.71860353672
N10/N9	2.71851342971
N11/N10	2.71836702094
N12/N11	2.71831316782
N13/N12	2.71829335740
N14/N13	2.71828606970
N15/N14	2.71828338872
N16/N15	2.71828240245
N17/N16	2.71828203962
N18/N17	2.71828190614
N19/N18	2.71828185704
N20/N19	2.71828183897
N21/N20	2.71828183233
N22/N21	2.71828182988
N23/N22	2.71828182898
N24/N23	2.71828182865
N25/N24	2.71828182853
N26/N25	2.71828182849
N27/N26	2.71828182847
N28/N27	2.71828182846

Tablica 3.7. – omjeri između broja elemenata

U tablici 3.7. se ponovo javlja e konstanta kao omjer između broja elemenata, ali puno kasnije se ostvaruje preciznost ostvarena izrazom (2-10).

Nakon što je izrazom (2-11) izračunat broj elemenata potreban za traženu sumu, izrazima (2-3), (2-4), (2-5), (2-6), (2-7) je izvršena provjera hoće li se izračunatim brojem elemenata dobiti inicijalno ciljana parcijalna suma. Djelomični rezultati su dani u tablici 3.8.

Ciljana suma	Formula 1	Formula 2	Formula 3	Formula 4	Formula 5
2	1.89188519162339	1.85095482122774	1.89729883666817	1.73549413051982	1.88436681259230
3	2.98480362670078	2.98065264696316	2.99107720853337	2.94270682803532	2.98892423982644
4	3.99408818432917	3.99354605651302	3.99684740162313	3.97970058180626	3.99625345912380
5	4.99775806906454	4.99768557059347	4.99883173158735	4.99260482331811	4.99863756729834
6	5.99916550436130	5.99915573479207	5.99956829729798	5.99728827285845	5.99950005240324
7	6.99969166237889	6.99969034228285	6.99984089349382	6.99900356004715	6.99981621327763
8	7.99999965431427	7.99999947580190	8.00005468922228	7.99974688077472	8.00004566825454
9	8.99995817885555	8.99995815469616	8.99997844656985	8.99986522362783	8.99997513531281
10	9.99998461149680	9.99998460822744	9.99999207017177	9.99995042133548	9.99999085307643
11	10.99999433843060	10.99999433798810	10.99999708267550	10.99998176140620	10.99999663507300
12	11.99999791716330	11.99999791710350	11.99999892676220	11.99999329044740	11.99999876211770
13	12.99999923375880	12.99999923375070	12.99999960517600	12.99999753170040	12.99999954460920
14	13.99999971811450	13.99999971811340	13.99999985475210	13.99999909196420	13.99999983247120
15	14.99999989629990	14.99999989629980	14.99999994656620	14.99999966595240	14.99999993836960
16	15.99999996185080	15.99999996185080	15.99999998034280	15.99999987711070	15.99999997732740
17	16.99999998596570	16.99999998596570	16.99999999276850	16.99999995479150	16.99999999165920
18	17.99999999483700	17.99999999483700	17.99999999733960	17.99999998336870	17.99999999693160
19	18.99999999810060	18.99999999810060	18.99999999902130	18.99999999388170	18.99999999887120
20	19.99999999930120	19.99999999930120	19.99999999963990	19.99999999774920	19.99999999958470
21	20.99999999974290	20.99999999974290	20.99999999986750	20.99999999917190	20.99999999984720
22	21.99999999990540	21.99999999990540	21.99999999995120	21.99999999969530	21.99999999994380
23	22.99999999996520	22.99999999996520	22.99999999998200	22.99999999988790	22.99999999997930
24	23.99999999998720	23.99999999998720	23.99999999999340	23.99999999995870	23.99999999999230
25	24.99999999999520	24.99999999999520	24.99999999999750	24.99999999998480	24.99999999999720
26	25.99999999999820	25.99999999999820	25.99999999999910	25.99999999999440	25.99999999999890
27	26.99999999999930	26.99999999999930	26.99999999999960	26.99999999999790	26.99999999999960
28	27.99999999999970	27.99999999999970	27.99999999999980	27.99999999999920	27.99999999999980

Tablica 3.8. – djelomični rezultati dobiveni izrazima (2-3), (2-4), (2-5), (2-6), (2-7) na temelju izračuna broja elemenata izrazom (2-11)

Tablica 3.8. služi kao potvrda ispravnosti vrijednosti broja elemenata koje su izračunate izrazom (2-11).

Prilikom implementiranja izraza (2-11), najveći problem je predstavljalo postavljanje gornje i doljnje granice. Navedene granice su služile za kontrolu petlje u kojoj se izvodila iteracija, odnosno ukoliko izračun tražene parcijalne sume pomoću broja elemenata izračunatog u tekućoj petlji nije davao odgovarajuću vrijednost, petlja je nastavljala s radom. U suprotnom, ako je izračun sume na temelju izračunatog broja elemenata u trenutnoj petlji dao rezultat koji upada u odstupanja postavljena gornjom i doljnjom granicom, petlja je završavala s radom.

3.2. Usporedba rezultata C-a i Python-a

Potpuni rezultati python-a su dani u tablicama u prilogu. Važno je napomenuti kako je u python-u iteracija računata samo za parcijalne sume iznosa 24 zbog velikog utroška vremena.

Jedino iznimno uočljivo odstupanje između rezultata ostvarenih python-om i C-om je u vremenima izvođenja iteracije. Vrijeme izvođenja iteracije u python-u se toliko brže povečavalo u odnosu na vrijeme izvođenja iteracije u C-u, da python-om nisu izračunate određene vrijednosti parcijalnih suma. Rezultati su dani u tablici 3.9.

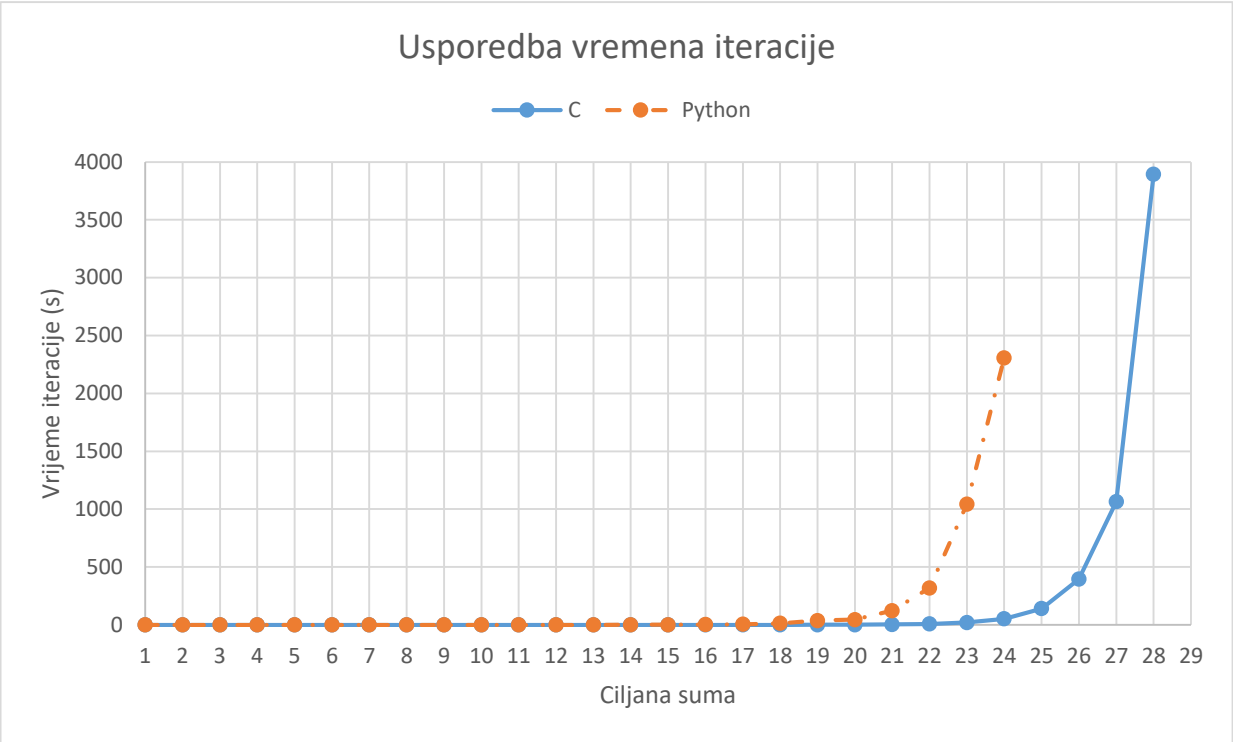
U python-u se ranije javljaju veliki vremenski skokovi. Za parcijalnu sumu 16, python uvelike nadilazi vrijeme od jedne sekunde, dok se u C-u vrijeme od jedne sekunde nadilazi tek za parcijalnu sumu 21. Python je za izračun parcijalne sume do 21 iteracijom iskoristio oko 120 sekundi vremena, dok je C za oko 140 sekundi izračunao parcijalnu sumu do 25 iteracijom. Na prvi pogled čini se da se ne radi o velikim razlikama između suma za koje su postignuta vremena, ali se radi o iznimno velikim pomacima u broju elemenata koji su korišteni za izračun parcijalnih suma. Za dostizanje parcijalne sume 21 potrebno je oko 740461601 elemenata, dok je za postizanje parcijalne sume 25 potrebno oko 40427833596 elemenata. Za parcijalnu sumu 25 potrebno je oko 54 puta više elemenata harmonijskog niza nego za postizanje parcijalne sume 21. Ovaj odnos, gdje je C postigao slično vrijeme za veću parcijalnu sumu u odnosu na python, vidljiv je i kod ostalih vrijednosti iteracije.

Kao što je i ranije navedeno, python-om nisu računate parcijalne sume iteracijom za vrijednosti iznad 24, obzirom da bi navedene radnje zahtjevale iznimno puno vremena. Uzimajući u obzir prosječnu vrijednost povećanja vremena izračunatu iz tablice 3.2., možemo izračunati da bi za izračun parcijalne sume do 28, iteracijom u python-u, bilo potrebno oko 122550 sekundi, odnosno oko 2042 minute.

Ovakva odstupanja u vremenima postignutim za različite promatrane parcijalne sume imaju dva moguća razloga : potreba za poboljšanjem koda, ili fundamentalne razlike između korištenih programskih jezika. Python je programski jezik visoke razine, dok je C jezik niske razine. Kada govorimo o programskom jeziku niske razine, govorimo o jeziku koji pruža nizak stupanj apstrakcije (i bolji pristup hardveru), dok su programski jezici visoke razine jezici koji pružaju visok stupanj apstrakcije.

Ciljana suma	Vrijeme iteracije (s) (C kod)	Vrijeme iteracije (s) (Python kod)
2	0	0
3	0	0
4	0	0
5	0	0
6	0.001	0
7	0.001	0
8	0.001	0
9	0.001	0
10	0.004	0.01517
11	0.001	0.01562
12	0.001	0.04657
13	0.002	0.09345
14	0.004	0.24955
15	0.010	0.67158
16	0.024	1.81220
17	0.068	5.01533
18	0.145	13.42157
19	0.366	36.88311
20	0.954	45.07782
21	3.179	122.03352
22	7.093	318.43726
23	19.64500	1041.46724
24	52.58500	2306.83964
25	140.81500	0
26	395.36400	0
27	1064.433	0
28	3894.302	0

Tablica 3.9. – usporedba vremena iteracije u C-u i Python-u



Slika 3.7. – usporedba trenda kretanja vremena iteracije u C-u i python-u

Na slici 3.7. prikazane su krivulje vremena iteracije u C-u i python-u. Kako je i prethodno navedeno, python puno prije počinje ostvarivati velike vremenske pomake.

Osim razlike u vremenima iteracija, ostale izračunate vrijednosti ne pokazuju značajna odstupanja. Za većinu računatih parametara, vrijednosti izračunate C-om i Python-om se podudaraju u svim danim decimalnim mjestima.

U tablici 3.10. prikazana je usporedba između vrijednosti parcijalnih suma izračunatih iteracijom u C-u i Python-u. Rezultati se u potpunosti, ili gotovo u potpunosti, podudaraju.

Ciljana suma	Parcijalna suma ostvarena iteracijom (C kod)	Parcijalna suma ostvarena iteracijom (Python kod)
2	2.083333333333330	2.08333333333333
3	3.019877344877340	3.01987734487734
4	4.027245195436520	4.02724519543652
5	5.002068272680170	5.00206827268016
6	6.004366708345570	6.00436670834556
7	7.001274097134160	7.00127409713416
8	8.000485571995780	8.00048557199578
9	9.000208062931120	9.0002080629311
10	10.0000430082757	10.0000430082757
11	11.0000177086364	11.0000177086364
12	12.0000030516656	12.0000030516656
13	13.0000012294809	13.0000012294809
14	14.0000013620532	14.0000013620532
15	15.0000003782672	15.0000003782672

Tablica 3.10. – usporedba vrijednosti parcijalnih suma ostvarenim C-om i python-om

Ciljana suma	Potreban broj elemenata za parcijalnu sumu (izraz 2-10) (C kod)	Potreban broj elemenata za parcijalnu sumu (izraz 2-10) (Python kod)
2	4.14865562135235	4.14865562135235
3	11.2772151880565	11.2772151880565
4	30.6546491213165	30.6546491213165
5	83.3279756642626	83.3279756642626
6	226.508922050442	226.508922050442
7	615.715086793564	615.715086793565
8	1673.68713193903	1673.68713193903
9	4549.55331727560	4549.5533172756
10	12366.9681099558	12366.9681099558
11	33616.90468642540	33616.9046864254
12	91380.221138150	91380.2211381501
13	248397.194600402	248397.1946004020
14	675213.580322479	675213.580322479
15	1835420.80571936	1835420.80571936

Tablica 3.11. – usporedba broja elemenata, izračunatih C-om i Python-om, potrebnih za određenu parcijalnu sumu

U tablici 3.11. prikazana je usporeda vrijednosti broja elemenata, koje su izračunate prema izrazu 2-10 u C-u i Python-u, potrebnih za dostizanje određene parcijalne sume. Kao i kod tablice 3.10., radi se o potpuno istim vrijednostima.

Kao posljednja usporedba, dana je tablica 3.12., u kojoj su uspoređene vrijednosti parcijalnih suma ostvarene uporabom izraza 2-3 i 2-4 u C-u i Python-u.

Ciljana suma	Formula 1 (C kod)	Formula 2 (C kod)	Formula 1 (Python kod)	Formula 2 (Python kod)
2	2.11639146630194	2.09130082724987	2.11639146630193	2.09130082724987
3	3.03963367784228	3.03592008928571	3.03963367784228	3.03592008928570
4	4.01409245559630	4.01357167072667	4.01409245559630	4.01357167072667
5	5.00511497351863	5.00504353523476	5.00511497351863	5.00504353523475
6	6.00187213267398	6.00186241586877	6.00187213267397	6.00186241586876
7	7.00068741626327	7.00068609879398	7.00068741626326	7.00068609879398
8	8.00025270953974	8.00025253111770	8.00025270953974	8.00025253111770
9	9.00009294269851	9.00009291854563	9.00009294269851	9.00009291854563
10	10.00003418846610	10.0000341851971	10.00003418846610	10.0000341851971
11	11.00001257679500	11.0000125763526	11.00001257679500	11.0000125763526
12	12.00000462668490	12.0000046266250	12.00000462668490	12.0000046266250
13	13.00000170205420	13.0000017020461	13.00000170205420	13.0000017020461
14	14.00000062614960	14.0000006261485	14.00000062614960	14.0000006261485
15	15.00000023034740	15.0000002303473	15.00000023034740	15.0000002303472

Tablica 3.12. – usporedba vrijednost parcijalnih suma izračunatih na temelju izraza 2-3 i 2-4 u C-u i Python-u

Formula 1 u tablici 3.12. označava izraz 2-3, a formula 2 označava izraz 2-4.

U tablici 3.12., kao prethodne dvije tablice, C kod i Python kod daju iste vrijednosti.

3.3. Usporedba integracije i iteracije

Na temelju izraza (2-10), uz proizvoljan odabir vrijednosti 1 za granicu b, te interpretaciju da se u granicu a uvrste brojevi elemenata, učinjena je usporedba između vrijednosti ostvarnih izračunom površine ispod krivulje $\frac{1}{x}$ i vrijednosti parcijalne sume ostvarene iteracijom za predviđeni broj elemenata. Rezultati su dani u tablici ispod.

Broj elemenata	Izracun integralom	Izracun iteracijom	Razlika
100	4.605170186	5.17737758	0.572207390
200	5.298317367	5.872031008	0.573713641
300	5.703782475	6.279330607	0.575548132
400	5.991464547	6.567429751	0.575965204
500	6.214608098	6.790823491	0.576215393
1000	6.907755279	7.484470923	0.576715644
1500	7.313220387	7.890102745	0.576882358
2000	7.60090246	8.177868167	0.576965707
3000	8.006367568	8.583416621	0.577049053
4000	8.29404964	8.871140364	0.577090724
5000	8.517193191	9.094308918	0.577115727
10000	9.210340372	9.787506101	0.577165729
25000	10.1266311	10.703826834	0.577195734
50000	10.81977828	11.396984014	0.577205734
75000	11.22524339	11.802452456	0.577209066
100000	11.51292546	12.090136192	0.577210732
150000	11.91839057	12.495602970	0.577212400
300000	12.61153775	13.188751817	0.577214067
400000	12.89921983	13.476434306	0.577214476
500000	13.12236338	13.699578107	0.577214727
600000	13.30468493	13.881899831	0.577214901
700000	13.45883561	14.036050630	0.577215020
800000	13.59236701	14.169582112	0.577215102
900000	13.71015004	14.287365217	0.577215177
1000000	13.81551056	14.392725788	0.577215228
1500000	14.22097567	14.798191063	0.577215393
2000000	14.50865774	15.085873219	0.577215479
3000000	14.91412285	15.491338410	0.577215560

Tablica 3.13 – usporedba izračuna iteracijom i izračuna površine ispod krivulje

Parcijalne sume iteracijom su računate do zadanog broja elemenata, dok je prilikom izračuna integralom zadani broj elemenata uvršten u gornju granicu.

Prosječna razlika između vrijednosti izračunate integralom, te vrijednosti izračunate iteracijom, iznosi 0.576715482. Vrijednost Euler-Mascheroni konstante je 0.5772156649. Iz tablice je vidljivo da porastom broja elemenata, raste i podudarnost razlike s Euler-Mascheroni konstantom.

4. ZAKLJUČAK

Cilj ovog završnog rada bio je prezentirati dvije metode izračuna tražene parcijalne sume, te usporediti njihovu točnost. Također su dane i dvije metode za izračun broja elemenata potrebnih za ostvarivanje određene parcijalne sume. Izvedena je usporedba između vrijednosti dobivenih izvršavanjem C koda i Python koda, te između vremena potrebnih za izračun parcijalne sume iteracijom.

Iterativnim izvođenjem C koda dosegnuta je parcijalna suma iznosa 28, nakon čega iteracija nije više izvođena zbog velikih zahtjeva u pogledu vremena. Jednadžbe, koje su korištene za izračun željene parcijalne sume na temelju broja elemenata, su pokazale porast točnosti izračuna što je tražena parcijalna suma veća, dok su se jednadžbe za izračun broja elemenata, potrebnih za određenu parcijalnu sumu, pokazale točnim na temelju provjere prethodno spomenutim formulama.

Uočeni su zanimljivi omjeri između vremena iteracija. Omjeri između vremena upućivali su na Euler-ovu konstantu. Sličan omjer javio se i između broja elemenata potrebnih za promatranu parcijalnu sumu. Omjeri između broja elemenata su kao rezultat davali Euler-ovu konstantu s preciznošću do desete decimale. Na temelju tog odnosa, te na temelju poznatog broja elemenata za neku paracijalnu sumu, možemo izračunati broj elemenata za neku drugu parcijalnu sumu.

Usporedbom vrijednost izračunatih u C-u i Python-u, uočena je velika razlika jedino u vremenima izvođenja iteracije. Python ranije ostvaruje značajne vremenske pomake u trajanju iteracije. Na primjer, za izračun parcijalne sume 24, python-u je potrebno oko 38 minuta, dok C istu parcijalnu sumu izračunava za oko 52 sekunde.

S ciljem poboljšanja postojećeg koda, i postizanja boljih rezultata, implementiranje neke od biblioteka koje pružaju proizvoljnu preciznost, poput GNU MPFR, GNU MPL, ili MAPM bi ostvarilo pozivitne pomake. Također, kako bi se smanjilo vrijeme iteracije, sumacija za određeni broj parcijalnih suma se može izvršiti, i te vrijednosti trajno spremiti, kako bi se mogle iskoristiti prilikom iterativnog izračuna većih parcijalnih suma.

Sav kod izvršen je na računalu ACER Aspire 3 A315-41-R9FP. Specifikacije su u prilogu.

LITERATURA

[1] : E.W.Weisstein, "Harmonic Series.", MathWorld : http://mathworld.wolfram.com/HarmonicSeries.html (Nicole Oresme izvod) [10.6.2019.]

[2] : A. Grinshpan, The partial sums of the harmonic series, Drexel University : http://www.math.drexel.edu/~tolya/123_harmonic.pdf [24.8.2019.]

[3] : J.Sondow, E.W.Weisstein, "Harmonic Number.", MathWorld : http://mathworld.wolfram.com/HarmonicNumber.html

[4] : Python Software Foundation, Python Documentation contents : https://docs.python.org/2/contents.html [6.7.2019.]

[5] : DevDocs, C Programming Language : https://devdocs.io/c/ [6.7.2019.]

[6] : cppreference, C programming language : https://en.cppreference.com/w/c/language [6.7.2019.]

[7] : D.Jukić, R.Scitovski, Matematika 1, 1998., Osijek, 97.str

SAŽETAK

Naslov : Parcijalne sume harmonijskog niza u programskom jeziku C

U ovom radu predstavljeni su osnovni pojmovi vezani uz harmonijski niz, dvije različite metode za izračun parcijalne sume, te dvije različite metode za izračun broja elemenata potrebnog za dostizanje zadane parcijalne sume. Napisan je programski kod u C-u i Python-u, te je uspoređen na temelju izračunatih vrijednosti te utrošenog vremena.

Pokazano je kako vrijeme iteracije iznad određenih vrijednost parcijalnih suma vrlo brzo raste, te postaje neekonomično za uporabu. Formule za izračun parcijalne sume daju točnije rezultate s porastom vrijednosti tražene parcijalne sume.

Fundamentalnih razlika između rezultata ostvarnih C-om i Python-om nema, ali jasne razlike se javljaju u vremenima iteracija ostvarnih C-om i Python-om. C se pokazao kao uvelike bolji izbor kada je u pitanju izračun parcijalne sume iteracijom.

Kako bi se poboljšali ostvareni rezultati, trebalo bi implementirati neku od biblioteka koja nudi proizvoljnu preciznost podataka, te izvršiti sumaciju i skladištenje prvih petnaest do dvadeset parcijalnih suma, kako bi se izračun većih parcijalnih suma olakšao i ubrzao.

Ključne riječi : C, python, parcijalne sume, harmonijski niz

ABSTRACT

Title : Partial sums of harmonic series in C programming language

In this paper, basic terms regarding harmonic series have been presented, as well as two different methods of partial sum calculation, and two different methods of calculating the amount of series members needed to calculate the wanted partial sum. Program code has been written in C and Python, and was compared using calculated values and time spent running the code.

It was shown that the iteration time grows significantly when certain partial sum values have been exceeded, thus becoming very time consuming. The growth of wanted partial sum causes our formulas to grow in accuracy.

There are no fundamental differences between values calculated using C and Python, but there are clear differences in iteration times achieved by C and Python. C is a tremendously better choice when it comes to calculating partial sums of harmonic series using iteration.

In order to improve results, one of the libraries offering arbitrary precision should be implemented. Also, first fifteen to twenty partial sums should be summed and stored for later use, in order to facilitate higher value partial sum calculations.

Key words : C, python, partial sums, harmonic series

ŽIVOTOPIS

Josip Uršan rođen je 11. veljače 1997. godine u Zagrebu. Nakon pohađanja osnovne škole u Osijeku, upisao je I. gimnaziju u Osijeku. Tijekom osnovnoškolskog i srednjoškolskog obrazovanja sudjelovao je na dva županijska natjecanja iz povijesti, te jednom županijskom natjecanju iz engleskog jezika. Tijekom osnovnoškolskog obrazovanja pohađao je tri napredna tečaja engleskog jezika u školi stranih jezika Lanico, dok je tijekom srednjoškolskog obrazovanja položio dva ispita znanja njemačkog jezika Konferencije ministara kulture Savezne Republike Njemačke za učenike u inozemstvu (DSD I i DSD II).

Nakon položene mature, 2016. godine upisuje prvu godinu preddiplomskog studija elektrotehnike na FERIT-u. Finalist je EWOB (Entrepreneurs without borders) natjecanja 2018. godine, koje organiziraju studenti Ekonomskog fakulteta u Osijeku, te sudionik Ericsson Summer Camp-a 2019. godine.

PRILOZI

C kod

Datoteka source.c :

```
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include "header.h"

int main() {
	double clock_utroseno = 0, iteration_utroseno = 0;
	int i = 1;
	int trazena_suma = 0;
	long double *krajnja_suma = 0;
	long double *potreban_broj_cl = 0;
	char yes_no;

	clock_t start_iteracija = 0, end_iteracija = 0;

	printf("Unesite zeljenu sumu : ");
	scanf("%d", &trazena_suma);
	putchar('\n');

	printf("Zelite li koristiti iteraciju?(Y/N) : ");
	scanf(" %c", &yes_no);

	clock_t start = clock();

	if(yes_no == 'y' || yes_no == 'Y')
	{
		start_iteracija = clock();
		end_sum = brute_force_check(&trazena_suma);
		end_iteracija = clock();
		printf("*Krajnja suma (iteracija) : %.14Lf\n\n", *krajnja_suma);
	}


	potreban_broj_cl = pronadi_broj_elemenata(&trazena_suma);
	printf("*Potreban broj elemenata : %.19Lf\n", *potreban_broj_cl);
	printf("Zaokruzena vrijednost potrebnog broja elemenata : %.4lf\n",
round(*potreban_broj_cl));

	izracun_parcijalne_sume_formulama(potreban_broj_cl);

	pronadi_broj_elemenata_iterativno(&trazena_suma);

	free(krajnja_suma);
	free(potreban_broj_cl);
	clock_t end = clock(NULL);

	clock_utroseno = end - start;
```

```
	iteracija_utroseno = end_iteracija - start_iteracija;

	printf("\Broj taktova (total) : %lf\n", clock_utroseno);
	printf("Utroseno vrijeme  %.6lf (total) s\n", (double)(clock_utroseno/CLOCKS_PER_SEC));

	printf("Broj taktova (iteration) : %lf\n", iteracija_utroseno);
	printf("Utroseno vrijeme (iteration) : %.6lf\n", iteracija_utroseno/CLOCKS_PER_SEC);
	return 0;
}
```

Datoteka functions.c:

```
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#include "header.h"

double* iteracija(int *trazena_suma) {
	long double i = 1;
	long double *sum = 0;

	sum = (double*)calloc(1, sizeof(double));

	printf("Iteriram...\n");
	while (*sum < *trazena_suma)
	{
		*sum += 1.0 / (long double)i;
		i++;

	}
	printf("Potreban broj elemenata (iterativna formula): %Lf\n", i);

	return sum;
}

double* pronadi_broj_elemenata(int *trazena_suma) {
	int i = 1;
	double e = 2.718281828459045;
	double EM_constant = 0.577215664901532;
	long double *broj_potrebnih_elemenata = 0;

	broj_potrebnih_elemenata = (double*)calloc(1, sizeof(double));

	*  broj_potrebnih_elemenata  =  pow((double)e,  ((*trazena_suma  -  EM_constant)  +
log(1.0)));


	return broj_potrebnih_elemenata;
}

double izracun_parcijalne_sume_formulama(double *potreban_broj_cl) {
	double e = 2.718281828459045;
	double EM_constant = 0.577215664901532;
	double formula1 = 0, formula2 = 0, formula3 = 0, formula4 = 0, formula5 = 0;
```

```
	formula1 = log(*potreban_broj_cl) + (1 / (*potreban_broj_cl)) + EM_constant * (1 +
log((*potreban_broj_cl) / (*potreban_broj_cl + 1)));
	formula2 = log(*potreban_broj_cl + 1) + EM_constant * (1 + log((*potreban_broj_cl) /
(*potreban_broj_cl + 1)));
	formula3 = log(*potreban_broj_cl) + EM_constant * (1 + (50 / (51 * (*potreban_broj_cl)))
+ log((*needed_members - (EM_constant / 19)) / (*needed_members + (EM_constant / 10))));
	formula4 = log(*potreban_broj_cl) + EM_constant;
	formula5 = log(*potreban_broj_cl) + EM_constant + (1 / (2 * (*potreban_broj_cl))) -
(1/(12*pow(*potreban_broj_cl,     2)))     +     (1/(120*pow(*potreban_broj_cl,     4)))     -
(1/(252*pow(*potreban_broj_cl, 6)));

	printf("\nFormula 1 : %.14lf\n", formula1);
	printf("Formula 2 : %.14lf\n", formula2);
	printf("Formula 3 : %.14lf\n", formula3);
	printf("Formula 4 : %.14lf\n", formula4);
	printf("Formula 5 : %.14lf\n", formula5);

	return formula1;
}


void pronadi_broj_elemenata_iterativno (int *trazena_suma) {
	int i = 1,  broj_iteracija = 0;
	double br_cl = 1;
	double rjesenje_iteracije = 0;
	double e = 2.718281828459045;
	double EM_const = 0.577215664901532;
	double suma_clanova_tijekom_iteracije = 0;

	double doljnja_granica = 0.99999*(*trazena_suma);	//0.99999
	double gornja_granica = 1.00001*(*trazena_suma); //1.00001

	printf("\n\Obavljam iteraciju za pronalazak broja elemenata s trazenom varijablom s
obje strane jednakosti : \n");
	do
	{
		rjesenje_iteracije = (((pow(e, *trazena_suma - EM_const) - pow(e, (1 / br_cl)))
* (pow(br_cl + 1, EM_const)))) / (pow(br_cl, EM_const));

		br_cl = rjesenje_iteracije;

		suma_clanova_tijekom_iteracije = izracunaj_parcijalne_sume_formulama(&br_cl);
		broj_iteracija++;
	}while(suma_clanova_tijekom_iteracije	<	doljnja_granica	||
suma_clanova_tijekom_iteracija > gornja_granica);

	printf("Potreban broj iteracija : %d\n", broj_iteracija);
	printf("Potreban	broj	elemenata	prema	iterativnoj	formuli	:	%.14lf\n",
rjesenje_iteracije);
}
```

Datoteka header.h:

```
#ifndef FUNCTIONS_H
#define FUNCTIONS_H

double* iteracija(int *trazena_suma);
double* pronadi_broj_elemenata(int *trazena_suma);
double izracun_parcijalnih_suma_formulama (double *potreban_broj_cl);
void pronadi_broj_elemenata_iterativno(int *trazena_suma);


#endif
```

Python kod

Funkcije

```
def iteracija(trazena_suma = 1):
    i = 1
    krajnja_suma = 0
    print("Iteriram...")
    while(krajnja_suma < trazena_suma):
        krajnja_suma += 1/i
        i+=1
    print("Potreban broj elemenata : {}" .format(i));
    print("Krajnja suma : {}" .format(krajnja_suma))


def pronadi_broj_elemenata(trazena_suma = 1):
    i = 1
    euler_mascheroni_const = 0.577215664901532
    broj_potrebnih_cl = 0
    broj_potrebnih_cl = math.pow(math.e, wanted_sum - euler_mascheroni_const + math.log(1, math.e))
    return broj_potrebnih_cl


def formule(potreban_broj_cl = 1):
    euler_mascheroni_const = 0.577215664901532
    formula1 = 0
```

```
formula2 = 0

formula3 = 0

formula4 = 0

formula5 = 0


formula1 = math.log(potreban_broj_cl , math.e) + (1/potreban_broj_cl ) + euler_mascheroni_const * (1+math.log((potreban_broj_cl /(potreban_broj_cl + 1)),math.e))

formula2 = math.logpotreban_broj_cl + 1, math.e) + euler_mascheroni_const * (1+math.log(((potreban_broj_cl )/(potreban_broj_cl + 1)), math.e))

formula3=math.log(potreban_broj_cl,math.e)+euler_mascheroni_const*(1+(50/(51*potreban_broj_cl))+math.log(((needed_members-(euler_mascheroni_const/10))/(potreban_broj_cl+(euler_mascheroni_const/10))), math.e))

formula4 = math.log(potreban_broj_cl ,math.e) + euler_mascheroni_const

formula5 = math.log(potreban_broj_cl , math.e) + euler_mascheroni_const + (1/(2*potreban_broj_cl )) - (1/(12*math.pow(potreban_broj_cl , 2))) + (1/(120*math.pow(potreban_broj_cl , 4))) - (1/(252*math.pow(potreban_broj_cl , 6)))

print("Formula 1 : {}" .format(formula1))

print("Formula 2 : {}" .format(formula2))

print("Formula 3 : {}" .format(formula3))

print("Formula 4 : {}" .format(formula4))

print("Formula 5 : {}" .format(formula5));

return formula1
```

```
def broj_elemenata_iterativno (trazena_suma = 1):
    i = 1
    broj_iteracija = 0
    rjesenje_iteracije = 0
    br_cl = 1
    euler_mascheroni_const = 0.577215664901532
    suma_clanova = 0
    doljnja_granica = 0.99999*trazena_suma
    gornja_granica = 1.00001*trazena_suma

    print("\n\Obavljam iteraciju da pronadem broj elemenata : \n")
    while(suma_clanova<= doljnja granica or suma_clanova > gornja_granica):

        rjesenje_iteracije = ((math.pow(math.e, trazena_suma - euler_mascheroni_const) - math.pow(math.e,1/needed_members))*(math.pow(br_cl + 1, euler_mascheroni_const)))/(math.pow(br_cl, euler_mascheroni_const))
        broj_iteracija = rjesenje_iteracije
        suma_clanova = formule(br_cl)
        broj_iteracija+=1

        print("Broj iteracija : {}" .format(broj_iteracija))
        print("Potreban broj elemenata izracunat iterativnom formulom : {}" .format(br_cl))
```

Glavni dio koda

```
sum_ulaz = int(input("Unesite broj : "))

odabir_iteracije = str(input("Zelite li koristiti iteraciju (Y/N)? : "))

start_time_iteration = 0

end_time_iteration = 0


start_program_time = time.time()

if(odabir_iteracije == 'y' or odabir_iteracija == 'Y') :

        start_time_iteration = time.time()

        iteracija(sum_ulaz)

        end_time_iteration = time.time()


potreban_broj_cl = pronadi_broj_elemenata(sum_ulaz)

print("Broj potrebnih elemenata : {}" .format(potreban_broj_cl))


formule(potreban_broj_cl)

broj_elemenata_iterativno(sum_ulaz)

end_program_time = time.time()


total_iteration_time = end_time_iteration - start_time_iteration

print("Vrijeme iteracije : {} s" .format(total_iteration_time))

total_program_time = end_program_time - start_program_time

print("Ukupno vrijeme programa : {} s" .format(total_program_time))
```

Specifikacije računala

Računalo : ACER Aspire 3 A315-41-R9FP

Chipset : AMD

Procesor : AMD Ryzen 5 2500U

-4 jezgre, 8 niti

-osnovna brzina : 2.0GHz

-turbo brzina : 3.6GHz

Grafička kartica : AMD Radeon Vega 8

RAM : 8GB DDR4 SRAM (2133MHz)

Pohrana podataka : 256GB SSD

Tablice

Tablica C rezultata

Ciljana suma	Parcijalna suma ostvarena iteracijom	Potreban broj elemenata (izraz 2-10)
2	2.083333333333330	4.14865562135235
3	3.019877344877340	11.27721518805650
4	4.027245195436520	30.65464912131650
5	5.002068272680170	83.32797566426260
6	6.004366708345570	226.50892205044200
7	7.001274097134160	615.71508679356400
8	8.000485571995780	1673.68713193903000
9	9.000208062931120	4549.55331727560000
10	10.000043008275700	12366.96810995580000
11	11.000017708636400	33616.90468642540000
12	12.000003051665600	91380.221138150
13	13.000001229480900	248397.194600402
14	14.000001362053200	675213.580322479
15	15.000000378267200	1835420.805719360
16	16.000000095452500	4989191.023762610
17	17.000000014847700	13562027.298604800
18	18.000000003717900	36865412.362863100
19	19.000000009732000	100210580.524620000
20	20.000000001618200	272400600.059407000
21	21.000000000402300	740461601.202826000
22	22.000000000140300	2012783315.221330
23	23.000000000035000	5471312310.391700
24	24.000000000042500	14872568831.162000
25	25.000000000004700	40427833596.254100
26	26.000000000005300	109894245428.663
27	27.000000000002800	298723530400.955
28	28.000000000001000	812014744422.049
29	0.000	2207284924203.270
30	0.000	6000022499693.350
31	0.000	16309752131261.800
32	0.000	44334502845079.600
33	0.000	120513673457548.000
34	0.000	327590128640500.000
35	0.000	890482293866032.0
36	0.000	2420581837980560.0
37	0.000	6579823624480560.0
38	0.000	17885814992891000.0
39	0.000	48618685882356000.0

Ciljana suma	Parcijalna suma ostvarena iteracijom	Potreban broj elemenata (izraz 2-10)
40	0.000	132159290357566000.0
41	0.000	359246197441016000.0
42	0.000	976532410446925000.0
43	0.000	2654490306219180000.0
44	0.000	7215652763216300000.0
45	0.000	19614177786721100000.0
46	0.000	53316863057809200000.0
47	0.000	144930260000482000000.0
48	0.000	393961292153155000000.0
49	0.000	1070897821576160000000.0
50	0.000	2911002088526870000000.0

Ciljana suma	Parcijalna suma (izraz 2-3)	Parcijalna suma (izraz 2-4)	Parcijalna suma (izraz 2-5)	Parcijalna suma (izraz 2-6)	Parcijalna suma (izraz 2-7)
2	2.11639146630194	2.09130082724987	2.12418706628079	2.000000	2.11570655494791
3	3.03963367784228	3.03592008928571	3.04567666260775	3.000000	3.04368243838014
4	4.01409245559630	4.01357167072667	4.01622206897784	4.000000	4.01622206897784
5	5.00511497351863	5.00504353523476	5.00618102829194	5.000000	5.00598838440665
6	6.00187213267398	6.00186241586877	6.00227384997407	6.000000	6.00220579433737
7	7.00068741626327	7.00068609879398	7.00083649950790	7.000000	7.00081184409278
8	8.00025270953974	8.00025253111770	8.00030773054512	8.000000	8.00029871186806
9	9.00009294269851	9.00009291854563	9.00011320768326	9.000000	9.00010989687302
10	10.00003418846610	10.0000341851971	10.0000416467714	10.000000	10.0000404297364
11	11.00001257679500	11.0000125763526	11.0000153209899	11.000000	11.0000148733955
12	12.00000462668490	12.0000046266250	12.0000056362770	12.000000	12.0000054716336
13	13.00000170205420	13.0000017020461	13.0000020734704	13.000000	13.0000020129038
14	14.00000062614960	14.0000006261485	14.0000007627871	14.000000	14.0000007405062
15	15.00000023034740	15.0000002303473	15.0000002806137	15.000000	15.0000002724170
16	16.00000008474000	16.0000000847400	16.0000001032320	16.000000	16.0000001002166
17	17.00000003117410	17.0000000311741	17.0000000379769	17.000000	17.0000000368676
18	18.00000001146830	18.0000000114683	18.0000000139709	18.000000	18.0000000135628
19	19.00000000421890	19.0000000042189	19.0000000051396	19.000000	19.0000000049894
20	20.00000000155200	20.0000000015520	20.0000000018907	20.000000	20.0000000018355
21	21.00000000057090	21.0000000005709	21.0000000006955	21.000000	21.0000000006752
22	22.00000000021000	22.0000000002100	22.0000000002558	22.000000	22.0000000002484
23	23.00000000007720	23.0000000000772	23.0000000000941	23.000000	23.0000000000913
24	24.00000000002840	24.0000000000284	24.0000000000346	24.000000	24.0000000000336
25	25.00000000001040	25.0000000000104	25.0000000000127	25.000000	25.0000000000123
26	26.00000000000380	26.0000000000038	26.0000000000046	26.000000	26.0000000000045
27	27.00000000000140	27.0000000000014	27.0000000000017	27.000000	27.0000000000016
28	28.00000000000050	28.0000000000005	28.0000000000006	28.000000	28.0000000000006
29	29.00000000000010	29.0000000000001	29.0000000000002	29.000000	29.0000000000002
30	30.000000	30.000000	30.000000	30.000000	30.000000
31	31.000000	31.000000	31.000000	31.000000	31.000000
32	32.000000	32.000000	32.000000	32.000000	32.000000
33	33.000000	33.000000	33.000000	33.000000	33.000000
34	34.000000	34.000000	34.000000	34.000000	34.000000
35	35.000000	35.000000	35.000000	35.000000	35.000000
36	36.000000	36.000000	36.000000	36.000000	36.000000
37	37.000000	37.000000	37.000000	37.000000	37.000000
38	38.000000	38.000000	38.000000	38.000000	38.000000
39	39.000000	39.000000	39.000000	39.000000	39.000000
40	40.000000	40.000000	40.000000	40.000000	40.000000
41	41.000000	41.000000	41.000000	41.000000	41.000000

Ciljana suma	Parcijalna suma (izraz 2-3)	Parcijalna suma (izraz 2-4)	Parcijalna suma (izraz 2-5)	Parcijalna suma (izraz 2-6)	Parcijalna suma (izraz 2-7)
43	43.000000	43.000000	43.000000	43.000000	43.000000
44	44.000000	44.000000	44.000000	44.000000	44.000000
45	45.000000	45.000000	45.000000	45.000000	45.000000
46	46.000000	46.000000	46.000000	46.000000	46.000000
47	47.000000	47.000000	47.000000	47.000000	47.000000
48	48.000000	48.000000	48.000000	48.000000	48.000000
48	49.000000	49.000000	49.000000	49.000000	49.000000
50	50.000000	50.000000	50.000000	50.000000	50.000000

Ciljana suma	Potreban broj elemenata (izraz 2-11)
2	3.18444642056633
3	10.64926806107240
4	30.03865093415770
5	82.71402350205290
6	225.89552371905000
7	615.10186924947900
8	1673.2635431603300
9	4548.9401863032300
10	12366.3549873918000
11	33616.2915669477000
12	91379.6080198067000
13	248396.5814824760000
14	675212.9672047060000
15	1835420.1926016500000
16	4989190.4106449200000
17	13562026.685487100
18	36865411.749745400
19	100210579.911502000
20	272400599.446289000
21	740461600.589709000
22	2012783314.60821000
23	5471312309.77858000
24	14872568830.5489000
25	40427833595.641000
26	109894245428.05000
27	298723530400.3410000
28	812014744421.4360000000
29	2207284924202.650000000
30	6000022499692.740000000
31	16309752131261.200000000
32	44334502845079.600000000
33	120513673457547.000000000
34	327590128640499.00000
35	890482293866031.00000
36	2420581837980560.00000
37	6579823624480560.00000
38	17885814992891000.00000
39	48618685882356000.00000
40	132159290357566000.00000
41	359246197441016000.0000
42	976532410446925000.0000

Ciljana suma	Potreban broj elemenata (izraz 2-11)
43	2654490306219180000.0000
44	7215652763216300000.0000
45	19614177786721100000.0000
46	53316863057809200000.000
47	144930260000482000000.000
48	393961292153155000000.000
49	1070897821576160000000.0000
50	2911002088526870000000.0000

Ciljana suma	Ukupno vrijeme (s)	Vrijeme iteracije (s)
2	0.00800	0
3	0.00400	0
4	0.01000	0
5	0.01800	0
6	0.00700	0.001
7	0.01000	0.001
8	0.00700	0.001
9	0.00300	0.001
10	0.00400	0.004
11	0.00500	0.001
12	0.00400	0.001
13	0.00500	0.002
14	0.00700	0.004
15	0.01600	0.010
16	0.02700	0.024
17	0.0720000	0.068
18	0.1500000	0.145
19	0.3700000	0.366
20	0.9580000	0.954
21	3.1870000	3.179
22	7.0990000	7.093
23	19.6500000	19.64500
24	52.5910000	52.58500
25	140.8220000	140.81500
26	395.3700000	395.36400
27	1064.4390000	1064.433
28	3894.3070000	3894.302
29	0.0160000	0
30	0.0070000	0
31	0.0080000	0
32	0.0060000	0
33	0.0070000	0
34	0.0070000	0
35	0.0060000	0
36	0.0070000	0
37	0.0070000	0
38	0.00600	0
39	0.00700	0
40	0.00700	0
41	0.00700	0
42	0.00700	0

Ciljana suma	Ukupno vrijeme (s)	Vrijeme iteracije (s)
43	0.00700	0
44	0.00700	0
45	0.00700	0
46	0.00500	0
47	0.00700	0
48	0.00700	0
49	0.00700	0
50	0.00700	0

Tablice Python rezultata

Ciljana suma	Parcijalna suma izračunata iteracijom
2	2.08333333333333
3	3.01987734487734
4	4.02724519543652
5	5.00206827268016
6	6.00436670834556
7	7.00127409713416
8	8.00048557199578
9	9.0002080629311
10	10.0000430082757
11	11.0000177086364
12	12.0000030516656
13	13.0000012294809
14	14.0000013620532
15	15.0000003782672
16	16.0000000954525
17	17.0000000148477
18	18.0000000037179
19	19.0000000097320
20	20.0000000016182
21	21.0000000004023
22	22.0000000001403
23	23.000000000035
24	24.0000000000425
25	0.000
26	0.000
27	0.000
28	0.000
29	0.000
30	0.000
31	0.000
32	0.000
33	0.000
34	0.000
35	0.000
36	0.000
37	0.000
38	0.000

Ciljana suma	Parcijalna suma izračunata iteracijom
39	0.000
40	0.000
41	0.000
42	0.000
43	0.000
44	0.000
45	0.000
46	0.000
47	0.000
48	0.000
49	0.000
50	0.000

Ciljana suma	Izračun broja elemenata prema izrazu 2-10
2	4.14865562135235
3	11.27721518805650
4	30.65464912131650
5	83.32797566426260
6	226.508922050442
7	615.715086793565
8	1673.687131939030
9	4549.5533172756
10	12366.9681099558
11	33616.9046864254
12	91380.2211381501
13	248397.1946004020
14	675213.5803224790
15	1835420.805719360
16	4989191.0237626100
17	13562027.298604800
18	36865412.362863100
19	100210580.524620000
20	272400600.059407000
21	740461601.2028260
22	2012783315.2213300
23	5471312310.391700
24	14872568831.16200
25	40427833596.254100
26	109894245428.663000
27	298723530400.955000
28	812014744422.0490000
29	2207284924203.270
30	6000022499693.350
31	16309752131261.80
32	44334502845080.30
33	120513673457548.0
34	327590128640500.0
35	890482293866032.0
36	2420581837980560.0
37	6579823624480560.0
38	17885814992891000.0
39	48618685882356000.0
40	132159290357566000.0
41	359246197441016000.0
42	976532410446925000.0

Ciljana suma	Izračun broja elemenata prema izrazu 2-10
43	2654490306219180000.0
44	7215652763216300000.0
45	19614177786721100000.0
46	53316863057809200000.0
47	144930260000482000000.0
48	393961292153155000000.0
49	1070897821576160000000.0
50	2911002088526870000000.0

Ciljana suma	Parcijalna suma (izraz 2-3)	Parcijalna suma (izraz 2-4)	Parcijalna suma (izraz 2-5)	Parcijalna suma (izraz 2-6)	Parcijalna suma (izraz 2-7)
2	2.11639146630193	2.09130082724987	2.12034207493036	2	2.11570655494790
3	3.03963367784228	3.03592008928570	3.04427170499802	3	3.04368243838013
4	4.01409245559630	4.01357167072667	4.01628666650846	4	4.01622206897783
5	5.00511497351863	5.00504353523475	5.00599153059087	5	5.00000000000000
6	6.00187213267397	6.00186241586876	6.00220416096621	6	6.00220579433736
7	7.00068741626326	7.00068609879398	7.00081086550653	7	7.00081184409278
8	8.00025270953974	8.00025253111770	8.00029830074950	8	8.00029871186806
9	9.00009294269851	9.00009291854563	9.00010973871303	9	9.00010989687301
10	10.00003418846610	10.0000341851971	10.0000403706164	10	10.0000404297364
11	11.00001257679500	11.0000125763526	11.0000148515198	11	11.0000148733955
12	12.00000462668490	12.0000046266250	12.0000054635688	12	12.0000054716335
13	13.00000170205420	13.0000017020461	13.0000020099346	13	13.0000020129038
14	14.00000062614960	14.0000006261485	14.0000007394136	14	14.0000007405062
15	15.00000023034740	15.0000002303472	15.0000002720150	15	15.0000002724170
16	16.00000008474000	16.0000000847400	16.0000001000687	16	16.0000001002166
17	17.00000003117410	17.0000000311741	17.0000000368132	17	17.0000000368676
18	18.00000001146830	18.0000000114683	18.0000000135428	18	18.0000000135628
19	19.00000000421890	19.0000000042189	19.0000000049821	19	19.0000000049894
20	20.00000000155200	20.0000000015520	20.0000000018328	20	20.0000000018355
21	21.00000000057090	21.0000000005709	21.0000000006742	21	21.0000000006752
22	22.00000000021000	22.0000000002100	22.0000000002480	22	22.0000000002484
23	23.00000000007720	23.0000000000772	23.0000000000912	23	23.0000000000913
24	24.00000000002840	24.0000000000284	24.0000000000335	24	24.0000000000336
25	25.00000000001040	25.0000000000104	25.0000000000123	25	25.0000000000123
26	26.00000000000380	26.0000000000038	26.0000000000045	26	26.0000000000045
27	27.00000000000140	27.0000000000014	27.0000000000016	27	27.0000000000016
28	28.00000000000050	28.0000000000005	28.0000000000006	28	28.0000000000006
29	29.00000000000010	29.000000	29.000000	29	29.000000
30	30.00000000	30.000000	30.000000	30	30.000000
31	31.000000	31.000000	31.000000	31	31.000000
32	32.000000	32.000000	32.000000	32	32.000000
33	33.000000	33.000000	33.000000	33	33.000000
34	34.000000	34.000000	34.000000	34	34.000000
35	35.000000	35.000000	35.000000	35	35.000000
36	36.000000	36.000000	36.000000	36	36.000000
37	37.000000	37.000000	37.000000	37	37.000000
38	38.000000	38.000000	38.000000	38	38.000000
39	39.000000	39.000000	39.000000	39	39.000000
40	40.000000	40.000000	40.000000	40	40.000000
41	41.000000	41.000000	41.000000	41	41.000000
42	42.000000	42.000000	42.000000	42	42.000000

Ciljana suma	Parcijalna suma (izraz 2-3)	Parcijalna suma (izraz 2-4)	Parcijalna suma (izraz 2-5)	Parcijalna suma (izraz 2-6)	Parcijalna suma (izraz 2-7)
43	43.000000	43.000000	43.000000	43	43.000000
44	44.000000	44.000000	44.000000	44	44.000000
45	45.000000	45.000000	45.000000	45	45.000000
46	46.000000	46.000000	46.000000	46	46.000000
47	47.000000	47.000000	47.000000	47	47.000000
48	48.000000	48.000000	48.000000	48	48.000000
49	49.000000	49.000000	49.000000	49	49.000000
50	50.000000	50.000000	50.000000	50	50.000000

Ciljana suma	Potreban broj elemenata (izraz 2-11)
2	3.24575391175727
3	10.56615199229410
4	30.00654108225710
5	82.70212154588780
6	225.89113702989400
7	615.10025453531400
8	1673.0733838287600
9	4548.939967715740
10	12366.354906975900
11	33616.291537364000
12	91379.608008923400
13	248396.581478472000
14	675212.967203233000
15	1835420.19260110
16	4989190.41064472
17	13562026.68548710
18	36865411.74974540
19	100210579.9115020
20	272400599.4462890
21	740461600.5897090
22	2012783314.608210
23	5471312309.778580
24	14872568830.54890
25	40427833595.641000
26	109894245428.05000
27	298723530400.34200
28	812014744421.4360
29	2207284924202.7
30	6000022499692.7
31	16309752131261.20
32	44334502845079.70
33	120513673457547.0
34	327590128640499.0
35	890482293866031.0
36	2420581837980560.0
37	6579823624480560.0
38	17885814992891000.0
39	48618685882356000.0
40	132159290357566000.0
41	359246197441016000.0
42	976532410446925000.0

Ciljana suma	Potreban broj elemenata (izraz 2-11)
43	2654490306219180000.0
44	7215652763216300000.0
45	19614177786721100000.0
46	53316863057809200000.0
47	144930260000482000000.0
48	393961292153155000000.0
49	1070897821576160000000.0
50	2911002088526870000000.0

Ciljana suma	Ukupno vrijeme izvođenja programa(s)	Vrijeme izvođenja iteracije (s)
2	0.0000000000000000	0
3	0.0152914524078369	0
4	0.0152320861816406	0
5	0.0152964591979980	0
6	0.0153338909149169	0
7	0.0156421661376953	0
8	0.0151643753051757	0
9	0.0153071880340576	0
10	0.0151724815368652	0.0151724815368652
11	0.0312473773956298	0.0156221389770507
12	0.0465741157531738	0.0465741157531738
13	0.109063863754272	0.093454122543335
14	0.249549865722656	0.249549865722656
15	0.687189579010009	0.671581029891967
16	1.812201499938960	1.812201499938960
17	5.0153343677521	5.0153343677521
18	13.4215698242187	13.4215698242187
19	36.8987169265747	36.8831086158752
20	45.0778193473815	45.0778193473815
21	122.04915094376	122.03352022171
22	318.43726110458	318.43726110458
23	1041.48275923728	1041.46724390983
24	2306.85527276992	2306.83964014053
25	0.0000000	0.00000
26	0.0000000	0.00000
27	0.0000000	0
28	0.0000000	0
29	0.0000000	0
30	0.0152965	0
31	0.0152984	0
32	0.0152888	0
33	0.0000000	0
34	0.0000000	0
35	0.0000000	0
36	0.0000000	0
37	0.0000000	0
38	0.00000	0
39	0.00000	0
40	0.00000	0
41	0.00000	0
42	0.00000	0

Ciljana suma	Ukupno vrijeme izvođenja programa(s)	Vrijeme izvođenja iteracije (s)
43	0.00000	0
44	0.00000	0
45	0.00000	0
46	0.00000	0
47	0.00000	0
48	0.00000	0
49	0.00000	0
50	0.00000	0