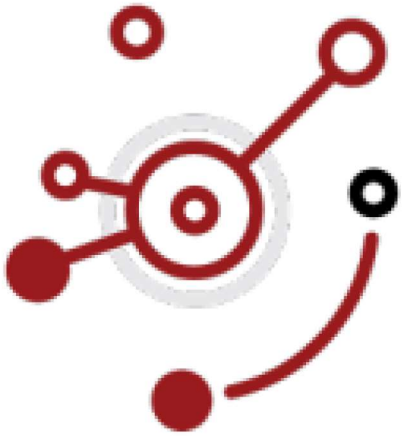
Monte Carlo - integracija

Vučković, Ilija

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

ODJEL ZA FIZIKU



Ilija Vučković

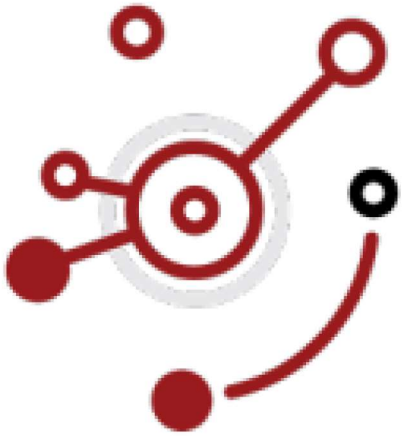
Monte Carlo - integracija

Završni rad

Osijek 2023.

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

ODJEL ZA FIZIKU



Ilija Vučković

Monte Carlo - integracija

Završni rad

Predložen Odjelu za fiziku Sveučilišta Josipa Jurja Strossmayera u Osijeku radi stjecanja zvanja prvostupnika fizike

Osijek 2023.

Ovaj završni rad je izrađen u Osijeku pod vodstvom doc. dr. sc. Zvonka Glumca u sklopu Sveučilišnog preddiplomskog studija fizike na Odjelu za fiziku Sveučilišta Josipa Jurja Strossmayera u Osijeku.

Sveučilište Josipa Jurja Strossmayera u Osijeku
Odjel za fiziku

Završni rad

Monte Carlo – integracija

Ilija Vučković

Sažetak

Monte Carlo integracija je jedna od najkorisnijih metoda za rješavanje složenih integrala u više dimenzija. Pripada numeričkim metoda rješavanja integrala, a osnova su joj slučajno generirane točke u n-dimenzijskom prostoru koje se nalaze ili unutar ili izvan neke krivulje čiji volumen ili površinu koju zatvara želimo izračunati. Sigurno se pitate jesu li ti brojevi savršeno slučajni ili koje su još metode numeričke integracije? Na sve to, uz nekoliko primjera ćemo odgovoriti u narednim poglavljima.

Rad je pohranjen u knjižnici Odjela za fiziku

Ključne riječi: Monte Carlo – integracija / numerička integracija / pseudoslučajni brojevi / kvazislučajni brojevi

Mentor: doc. dr. sc. Zvonko Glumac

Ocjenjivači:

Rad prihvaćen:

University Josip Juraj Strossmayer Osijek

Department of Physics Bachelor of Physics Thesis

Monte Carlo – integration

Ilija Vučković

Abstract

Monte Carlo integration is one of the most useful methods for solving complex integrals in multiple dimensions. It belongs to numerical methods of solving integrals, and its basis is randomly generated points in n-dimensional space that are located either inside or outside a curve, whose volume or area that curve encloses we want to calculate. You must be wondering if these numbers are perfectly random or what other methods of numerical integration are there? We will answer all this, with a few examples in the following chapters.

Thesis deposited in Department of Physics library

Keywords: Monte Carlo integration/ numerical integration / pseudo-random numbers/ quasi-random numbers

Supervisor: Zvonko Glumac, Ph.D.

Reviewers:

Thesis accepted:

Sadržaj

1. Uvod 5
2. Numerička integracija 6
2.1. Jednostavna integracija (Riemannov integral) 7
2.2. Trapezoidno pravilo 8
2.3. Simpsonovo pravilo 9
3. Slučajni brojevi 11
3.1. Pseudoslučajni brojevi 11
3.1.1. Linearni kongruencijalni generator 12
3.2. Kvazislučajni brojevi 13
3.2.1. Richtmyerovi nizovi 14
4. Monte Carlo integracija 14
4.1. Pogreška Monte Carlo integracije 17
5. Zaključak 25
6. Dodatak A 26
7. Popis literature 29

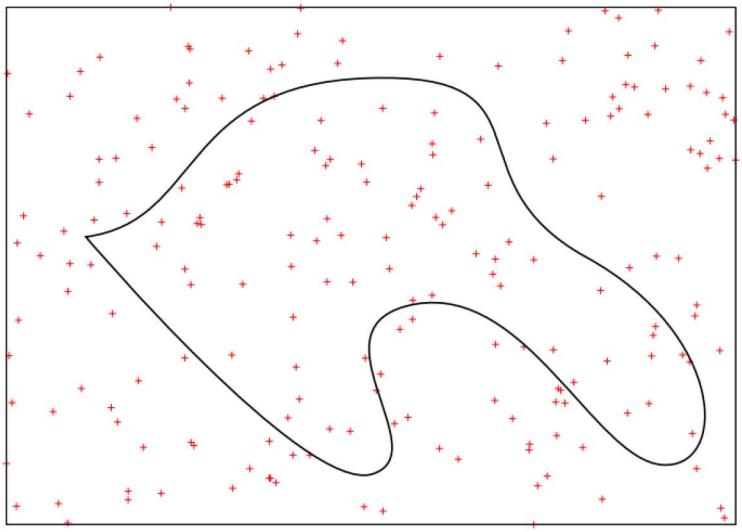
1. Uvod

Monte Carlo je poznati grad u kneževini Monaco i jedan od najljepših i najluksuznijih gradova Europe. Obiluje skupocjenim hotelima i kasinima te se u njemu događaju brojne manifestacije. No taj gradić nije tema ovoga završnog rada, nego numerička metoda za rješavanje integralnih jednadžba i višedimenzijskih integrala nazvana po njemu, Monte Carlo integracija.

Slika 1 Monte Carlo (izvor: internetski pristup Forbes).

Monte Carlo integracija pripada numeričkim metodama rješavanja problema. Prvi put se spominje u 18. stoljeću u radovima matematičara Comte de Buffona[1]. Do njenog je razvoja došlo tek s pojavom prvih boljih računala u drugoj polovici 20. stoljeća. Među najzaslužnijima za razvoj ovoga načina integracije ubrajaju se John von Neumann[2] i Stanislaw Ulam[3]. Metoda se zasniva na računanju integrala pomoću slučajno generiranih brojeva, odnosno točaka u n-dimenzijskom ograničenom prostoru. Što je veći broj slučajno generiranih točaka, manja je greška procjene.

[1] Georges-Louis Leclerc, Comte de Buffon; (1707. – 1788.), francuski prirodoslovac, matematičar i enciklopedist.
[2] John von Neumann; (1903. – 1957.), mađarsko - američki matematičar i polihistor.
[3] Stanislaw Ulam; (1909. – 1984.), poljsko-američki matematičar i nuklearni fizičar.

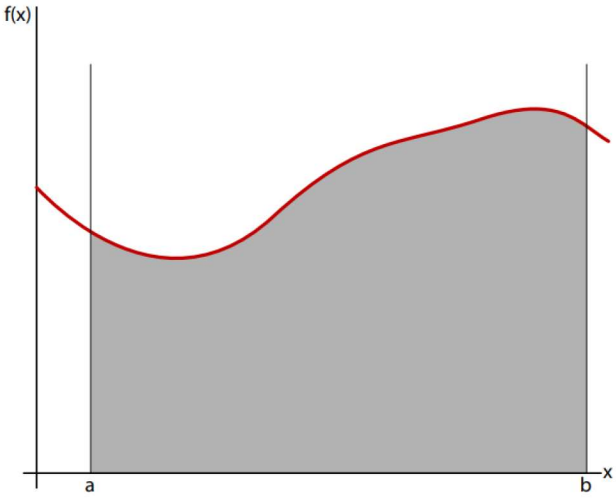


Slika 2 Monte Carlo integracija: traženje površine nepravilnog lika (izvor: Ayars-computational physics with Python).

Ova metoda ima veliku primjenu u raznim poljima fizike kao što su statistička fizika, fizika visokoenergijskih čestica, kvantna fizika, a može se pronaći njezina primjena i u biologiji, kemiji i ostalim znanostima. Sama primjena i način na koji ova metoda funkcionira bit će obrađeni u daljnjem tekstu. Reći ćemo nešto više i o slučajnim brojevima i ostalim numeričkim metodama za rješavanje integrala čisto kako bi dobili spoznaju koliko je zapravo ova metoda korisna i kako uvelike olakšava rješavanje složenih problema. Za sami kraj ostaju nam brojni primjeri računanja volumena i površina određenih geometrijskih tijela.

2. Numerička integracija

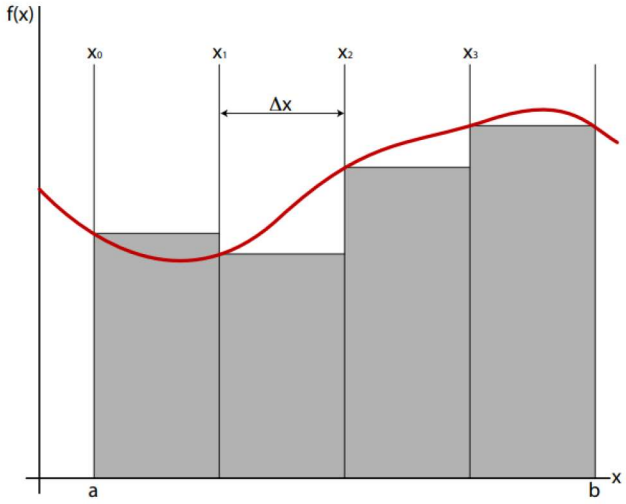
Numerička integracija je dio matematike koji se razvija od 16. stoljeća te postoji mnoštvo metoda za određivanje vrijednosti integrala. Svaka metoda ima veće ili manje odstupanje od prave vrijednosti. Ovisno o tome što računamo i trebamo li precizniji rezultat odlučujemo se za pojedinu metodu ili pravilo. U navedenom poglavlju izračunat ćemo površinu ispod krivulje $f(x)$, odnosno integral funkcije f u granicama od a do b.(Slika 3.)



Slika 3 Računanje površine ispod krivulje *f(x)*(izvor: Ayars-Computational physics with Python).

2.1. Jednostavna integracija (Riemannov[4] integral)

Neka je $f : [\, a, b \,] \rightarrow R$ nenegativna funkcija i omeđena na $[\, a, b \,]$. Označimo sa S skup točaka ravnine omeđen grafom funkcije f i pravcima $x = a$, $x = b$ i osi x. Taj skup točaka nazivamo pseudotrapez. Ovime je detaljnije opisana funkcija f i pseudotrapez čiju površinu želimo izračunati.



Slika 4 Pseudotrapez podijeljen na *n* jednakih dijelova (izvor: Ayars-Computational physics with Python) .

Podjelimo sada površinu pseudotrapeza na $n \in N$ jednakih ekvidistantnih dijelova pri čemu vrijedi $a = x_0 < x_1 < x_2 < \ldots < x_n = b$, $\Delta x = (b - a) / n$. Povučemo li pravce $x = x_0$, $x = x_1$, ...

[4] Georg Friedrich Bernhard Riemann; (1826. – 1866.) njemački matematičar.

$,x = x_n$ podijelili smo pseudotrapez S na pseudotrapeze S_1, S_2, …, S_n takve da je segment $[x_i, x_{i+1}]$ baza pseudotrapeza S_i.

Aproksimiramo površinu pseudotrapeza S_i površinom pravokutnika sa stranicama $\Delta x = x_{i+1} - x_i$ i visinom koja je jednaka vrijednosti funkcije f u lijevom ili desnom rubu segmenta $[x_i, x_{i+1}]$

$$P(S_i) \approx f(x_i)\,\Delta x. \tag{1}$$

Tada se površina pseudotrapeza S može aproksimirati zbrojem površina svih takvih pravokutnika

$$P(S) = \int f(x)\,dx \approx \sum_{i=1}^{n} f(x_i)\Delta x\,. \tag{2}$$

Ako uzmemo dovoljno velik broj n, odnosno ako podjelimo naš početni pseudotrapez S na beskonačno mnogo manjih tada aproksimacija postaje bolja.

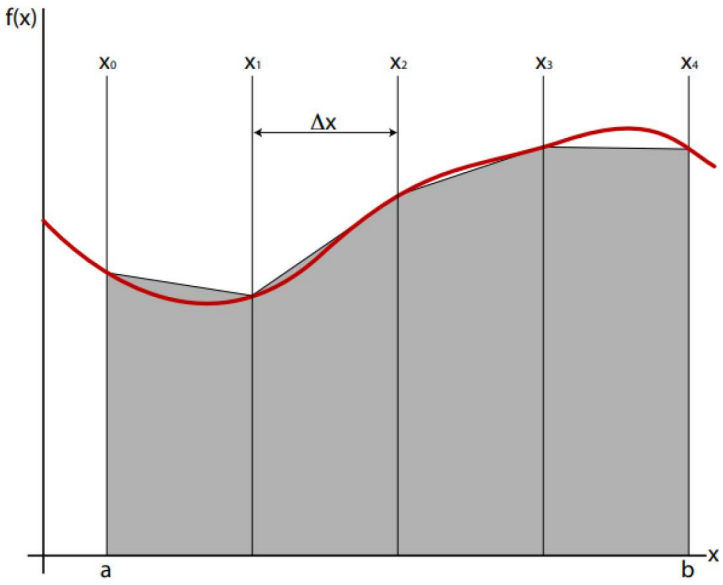
$$P(S) = \int f(x)\,dx = \lim_{n\to\infty} \sum_{i=1}^{n} f(x_i)\Delta x. \tag{3}$$

Ova metoda je vrlo intuitivna i jako dobro objašnjava koji postupak treba provesti kako bi se došlo do površine ispod krivulje. No, ona nije najpreciznija te postoje brojne druge metode koje na drugačije načine dolaze do rješenja s manjim odstupanjem od prave vrijednosti. Naravno, i ova metoda postaje bolja ukoliko početni pseudotrapez podijelimo na mnoštvo manjih. Međutim, to zahtjeva više vremena za izvršavanje računa, ali smanjuje grešku procjene. Možemo reći da je greška procjene obrnuto srazmjerna broju n.

2.2. Trapezoidno pravilo

Vrlo slična prethodnoj metodi, trapezoidna metoda umjesto pravokutnika koristi trapeze. Zbog toga dobivamo rezultat bliži pravoj vrijednosti integrala. Analogno prethodnoj metodi, podijelimo površinu na $n \in N$ dijelova i zbrajamo površine svih malih trapeza. Bitna razlika je što sada računamo vrijednost funkcije f u oba ruba segmenta $[x_i, x_{i+1}]$ i uzimamo srednju vrijednost. Prema tome slijedi

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{n} \frac{f(x_i)+f(x_{i+1})}{2}\Delta x. \tag{4}$$



Slika 5 Trapezoidna metoda (izvor: Ayars-Computational physics with Python).

Aproksimacija je bolja što je manja greška procjene, odnosno što je *n* veći. Važno je za primjetiti da greška procjene ove metode je obrnuto proporcionalna s kvadratom broja n. A ono što nam je svima intuitivno je to da sa dva puta većim brojem n, računalo izvršava dva puta više operacija, pa mu treba više vremena.

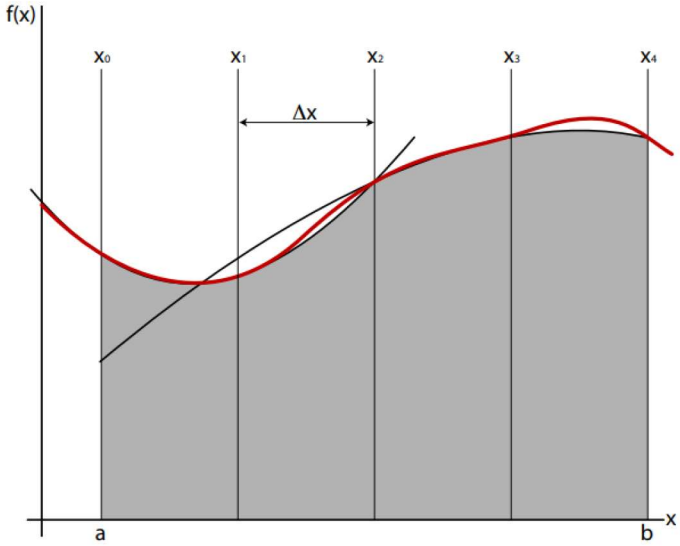
2.3. Simpsonovo pravilo

Kako bi se još više približili pravoj vrijednosti integrala, u Simpsonovoj[5] metodi uparujemo dva odreska (pseudotrapeza) i definiramo pomoću tri karakteristične točke parabolu na tom području. Takvom podjelom zapravo se broj odrezaka smanjio dva puta što vidimo i na slici 6. Time integral postaje

$$\int_a^b f(x)\,dx \approx \sum_{j=1}^{\frac{N-1}{2}} \int_{x_{2j-1}}^{x_{2j+1}} g_j(x)\,dx, \tag{5}$$

gdje je $g_j(x)$ funkcija pojedine parabole.

[5] Thomas Simpson; (1710. – 1761.), britanski matematičar i izumitelj.



Slika 6 Simpsonova metoda: parabola se proteže kroz svaka dva odreska (izvor: Ayars-Computational physics with Python).

Ako primijenimo Lagrageovu interpolaciju na funkciju $g_j(x)$ na intervalu $[\, x_{i-1},\, x_{i+1}\,]$ slijedi

$$g_j(x)=\frac{(x-x_i)(x-x_{i+1})}{(x_{i-1}-x_i)(x_{i-1}-x_{i+1})}f(i-1)+\frac{(x-x_{i-1})(x-x_{i+1})}{(x_i-x_{i-1})(x_i-x_{i+1})}f(i)+\frac{(x-x_{i-1})(x-x_i)}{(x_{i+1}-x_{i-1})(x_{i+1}-x_i)}f(i+1). \quad (6)$$

Nakon izlučivanja dobivamo

$$g_j(x)=x^2\left[\frac{f(x_{i-1})}{(x_{i-1}-x_i)(x_{i-1}-x_{i+1})}+\frac{f(x_i)}{(x_i-x_{i-1})(x_i-x_{i+1})}+\frac{f(x_{i+1})}{(x_{i+1}-x_{i-1})(x_{i+1}-x_i)}\right]+$$
$$x\left[\frac{-(x_i+x_{i+1})f(x_{i-1})}{(x_{i-1}-x_i)(x_{i-1}-x_{i+1})}+\frac{-(x_{i-1}+x_{i+1})f(x_i)}{(x_i-x_{i-1})(x_i-x_{i+1})}+\frac{-(x_{i-1}+x_i)f(x_{i+1})}{(x_{i+1}-x_{i-1})(x_{i+1}-x_i)}\right]+\left[\frac{x_ix_{i+1}f(x_{i-1})}{(x_{i-1}-x_i)(x_{i-1}-x_{i+1})}+\right.$$
$$\left.\frac{x_{i-1}x_{i+1}f(x_i)}{(x_i-x_{i-1})(x_i-x_{i+1})}+\frac{x_{i-1}x_if(x_{i+1})}{(x_{i+1}-x_{i-1})(x_{i+1}-x_i)}\equiv x^2[A]+x[B]+[C].\right. \quad (7)$$

Daljnjim matematičkim manipulacijama i uvrštavanjem dobivamo

$$\int_{x_{i-1}}^{x_{i+1}} g_i(x)\,dx=\frac{1}{3}[A](x_{i+1}^3-x_{i-1}^3)+\frac{1}{2}[B](x_{i+1}^2-x_{i-1}^2)+[C](x_{i+1}-x_{i-1}). \quad (8)$$

Kombinacijom prethodnih jednadžba dolazimo do rezultata

$$\int_a^b f(x)\,dx\approx\frac{\Delta x}{3}\sum_{i=2n}^{N-1}(f(x_{i-1})+4f(x_i)+f(x_{i+1})). \quad (9)$$

Ovo što smo dosad izveli je bilo samo za paran broj pseudotrapeza. Kada imamo neparan broj, dodajemo površinu posljednjeg pseudotrapeza odvojeno na ukupno rješenje. Površina posljednjeg pseudotrapeza je

$$P(S_n)=\frac{\Delta x}{12}[5f(N)+8f(N-1)-f(N-2)]. \quad (10)$$

Kao što vidimo rezultat je puno bolji jer pogreška kod Simpsonovog pravila ili metode obrnuto je srazmjerna četvrtoj potenciji broja n.

Osim ovih nabrojanih metoda, postoje i druge koje se više odnose na specijalne funkcije. Uglavnom cilj svake metode je isti, doći do što preciznijeg rješenja sa minimalnim naporom, odnosno sa minimalnom potrošnjom vremena na izračun.

3. Slučajni brojevi

Kada krećemo sa Monte Carlo integracijom, poželjno je imati slučajne brojeve odnosno slučajnu raspodjelu točaka iz zadanog skupa. Mali problem je generirati slučajne brojeve. Takve brojeve teško je definirati. Postoje brojna tumačenja i knjige samo o slučajnim brojevima i njihovim svojstvima. Za Monte Carlo integraciju su nam prijeko potrebni dobro definirani slučajni brojevi. Dobivamo ih pomoću programa korištenjem računala koja nisu nimalo slučajna. Svako računalo radi ono što mu je zapisano u nekom programu, algoritmu pa zbog toga ti slučajni brojevi dobiveni pomoću računala često nisu savršeno slučajni. Postoje algoritmi koji su samo bolji ili lošiji, u ovisnosti što nam treba. Tako razlikujemo pseudoslučajne i kvazislučajne brojeve. Da bismo imali što bolji generator slučajnih brojeva pogledajmo neka pravila koja bi on trebao zadovoljavati:

- brojevi moraju biti jednoliko raspoređeni
- ne smije postojati bilo kakva poveznica između brojeva
- ne smije uzimati previše vremena za računanje
- nakon određenog niza brojeva, slučajno generirani brojevi se kreću ponavljati, pa treba postojati što duži period
- mora moći ponoviti izračun sa istim slučajnim brojevima.

3.1. Pseudoslučajni brojevi

Pseudoslučajni brojevi su brojevi napravljeni korištenjem jednostavnog algoritma. Zanimljivo je da osoba koja ne poznaje algoritam ili nije vidjela period slučajnih brojeva, teško može zaključiti da su samo posljedica algoritma.

Postoje brojni testovi koji provjeravaju kvalitetu pojedinog algoritma. Najpoznatiji test je test frekvencije i za njega imamo dvije metode: χ^2 test i Kolomogorov-Smirnov[6] test. Oni nam pokazuju koliki je stupanj sličnosti između teorijske jednolike raspodjele i našeg niza slučajnih brojeva. Primjerice, ako imamo interval *[0, 5]*, podijelimo interval na a dijelova. Zatim pogledamo n slučajno generiranih brojeva iz intervala *[0, 5]* i promatramo kojoj sekciji *b* ($1 \leq b \leq a$) pripada koliko brojeva. Zatim radimo χ^2 i gledamo kakvu smo raspodjelu dobili. Ovaj test provjerava svojstvo ujednačenosti, a za ostala svojstva postoje drugačiji testovi.

Prije nego što krenemo sa testiranjem slučajnih brojeva, treba ih prvo generirati. Neki od najpoznatijih algoritama za generatore pseudoslučajnih brojeva su:

- Linearni kongruencijalni generator
- Lagged Fibonacci generator
- Shift register generator
- RANMAR.

3.1.1. Linearni kongruencijalni generator

Neka su $a, b \in Z$ i neka je $m \gg 1$ cijeli broj. Neka je poznat x_0 i rekurzijska relacija

$$x_i = (ax_{i-1} + b) \bmod m. \tag{11}$$

Tako možemo pronaći $x_1, x_2, \ldots$ koji predstavljaju naš niz pseudoslučajnih brojeva.

$x_1 = (ax_0 + b) \bmod m$

$x_2 = (ax_1 + b) \bmod m$

$x_3 = \ldots itd.$

Primjer 3.1.

Neka je $m = 121$, $a = 5$, $b = 3$ i $x_0 = 34$. Primijenimo formulu (11)

$x_i = (x_{i-1}a + b) \bmod m$

$x_1 = (34 \cdot 5 + 3) \bmod 121 = 52$

[6] Nikolai Vasilyevich Smirnov; (1900. – 1966.), ruski matematičar.
Andrej Nikolajevič Kolmogorov;(1903. – 1987.), ruski matematičar.

$x_2 = (52 \cdot 5+3) \bmod 121 = 21$

$x_3 = (21 \cdot 5+3) \bmod 121 = 108$

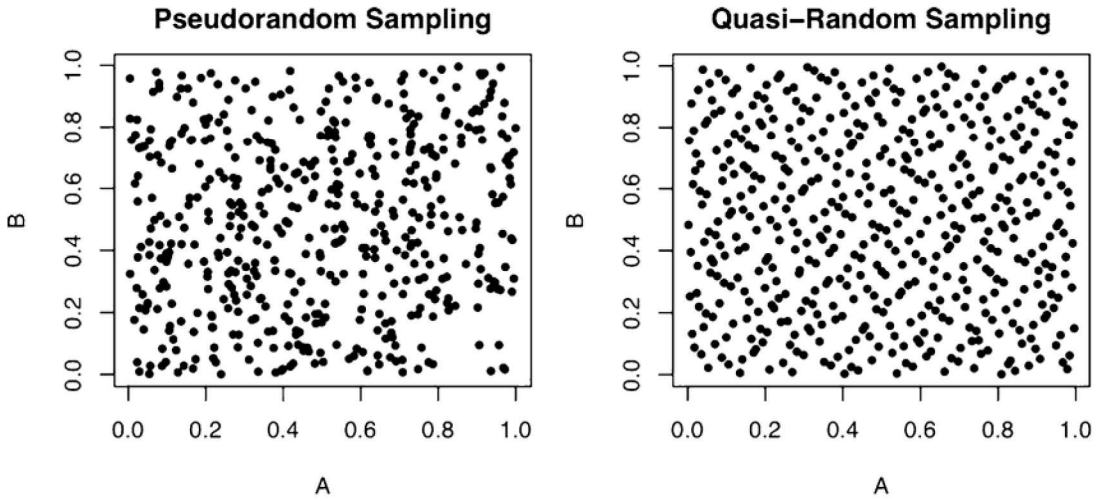
$x_4 = (108 \cdot 5+3) \bmod 121 = 59.$

Brojevi $x_1, x_2, x_3, x_4, \ldots$ odnosno *52, 21, 108, 59* nazivaju se pseudoslučajnim brojevima.

Nakon određenog vremena primijetit ćemo da se brojevi u nizu ponavljaju. Kako bi to izbjegli jer želimo što bolje slučajne brojeve koji imaju što veći period, moramo odabrati što veći m.

3.2. Kvazislučajni brojevi

Kao što smo vidjeli pseudoslučajni brojevi imaju svojih nedostataka što se tiče jednolike raspodjele. Kako bi se riješio taj problem, uvodimo kvazislučajne brojeve. Ako pogledamo sliku 3.1. možemo uočiti glavni problem i razliku između ove dvije vrste slučajnih brojeva.



Slika 7 Razlika u raspodjeli točaka pseudoslučajnih i kavzislučajnih brojeva (izvor: internetski pristup ResearchGate).

Budući da se radi o jednolikoj raspodjeli, svaka točka na grafu ima istu vjerojatnost pojavljivanja, ali svejedno vidimo da su na lijevom dijelu slike 3.1. prisutna mjesta sa više točaka i mjesta sa prazninama. Ako bi povećali broj točaka taj problem bi bio riješen i dobili bi izgled sličan izgledu desne strane slike 3.1. No, ono što je cilj kod većine simulacija pa i u Monte Carlo simulaciji je što manji broj točaka zbog uštede vremena računanja.

Za razliku od pseudoslučajnih brojeva koji ovise o prethodnima, kvazislučajni brojevi „pamte“ prethodne i pokušavaju se smjestit tako da ne smetaju drugima, odnosno žele se udaljiti od ostalih.

Postoji mnogo metoda pomoću kojih se mogu dobiti kvazislučajni brojevi, a ovo su samo neke od njih:

- Richtmyerovi nizovi
- Haltonovi nizovi
- Sobolovi nizovi
- Faureovi nizovi
- Haselgroveovi nizovi.

3.2.1. Richtmyerovi nizovi

Upoznajmo se sada sa naizgled jednostavnim generatorom kvazislučajnih brojeva osmišljenom od strane Richtmyerea[7]. Ovaj generator stvara beskonačan niz za proizvoljno odabranu dimenziju d. Za *n*-tu točku niza $x_n = (x_{n,0}, x_{n,1}, x_{n,2}, \ldots, x_{n,d})$. Ona je definirana relacijom

$$x_{n,i} = nS_i \bmod 1 \qquad n=1,\ldots,d, \tag{12}$$

gdje su S_i iracionalni brojevi čiji je zadatak osigurati beskonačan period niza. Kako bi olakšali potražnju iracionalnih brojeva, uobičajeno je da S_i predstavlja korijen iz i-tog prostog broja.

4. Monte Carlo integracija

Monte Carlo integracija je jedna od metoda za rješavanje složenih višedimenzijskih integrala korištenjem slučajnih brojeva. Osnova su joj pseudoslučajni brojevi s kojima smo i mi riješili nekoliko problema na kraju poglavlja. Ono što je najbitnije kod slučajnih brojeva koji se koriste u Monte Carlo integraciji je da su neovisni, jednoliko raspoređeni te da ih ima jako puno kako bi dobili što precizniji rezultat.

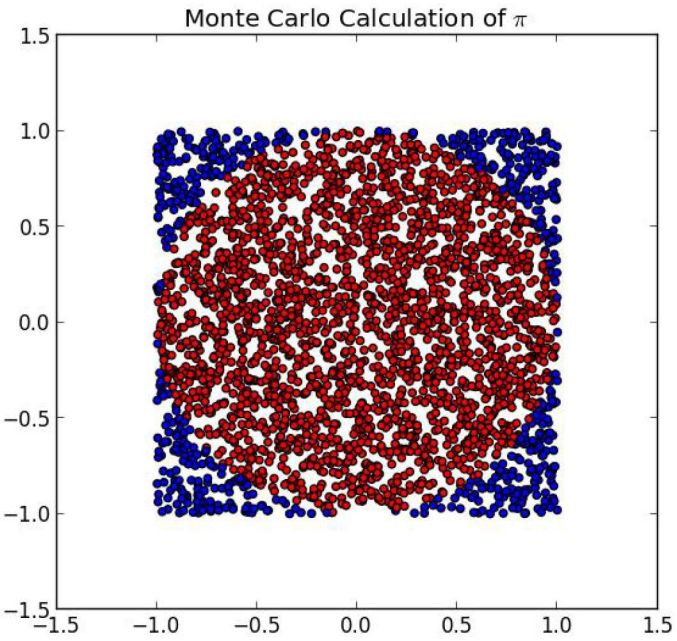
Primjer 4.1. Izračunajte površinu kruga polumjera 1.

U ovom primjeru koristit ćemo se Monte Carlo tehnikom integracije. Oko kruga ćemo napraviti kvadrat čija je površina jednaka 2 x 2 = 4. Napomenimo samo da lik kojim želimo opisati krug ne mora biti kvadrat, nego bilo koji geometrijski lik čiju površinu lako možemo izračunati. Zatim ćemo koristeći jednu od metoda navedenih u prethodnim poglavljima generirati veliki broj točaka i prebrojavati koja točka je unutar kruga, a koja izvan. Matematički zapisano, za točku sa koordinatama A_i (x_i, y_i) kažemo da pripada krugu ako zadovoljava jednadžbu

[7] Robert Davis Richtmyer;(1910. – 2003.), američki fizičar, matematičar, pedagog, pisac i glazbenik.

$$x_i^2 + y_i^2 \leq 1\,, \qquad i=1,\ 2,\ \ldots$$

Za sve ostale točke kažemo da ne pripadaju krugu.



Slika 8 Računanje površine kruga pomoću Monte Carlo integracije (izvor: internetski pristup ResearchGate).

Kako bi izračunali traženu površinu kruga, sve uvrstimo u jednadžbu

$$\frac{Broj\ točaka\ unutar\ kruga}{Ukupan\ broj\ točaka} \cdot Površina\ kvadrata = Površina\ kruga = \pi. \qquad (13)$$

Ovakav primjer nam je jako dobar pokazatelj koliko nam je točaka potrebno kako bi dobili što bližu vrijednost već dobro poznatoj $P = r^2\pi = 1^2\pi = \pi$.

Tablica 1 Analiza preciznosti Monte Carlo integracije površine kruga.

Broj točaka *n*	Površina kruga	Pogreška	Vrijeme / s
500	3.2800	0.1384	0.0029
5000	3.1376	0.0039	0.0537
50000	3.1371	0.0044	0.1231
500000	3.1407	0.0008	1.6519
5000000	3.1428	0.0011	13.1685

Pogledamo li tablicu 1, vidjet ćemo rezultate koji se dobiju korištenjem Python koda 1 u dodatku A. Nacrtajmo i graf kako bi lakše došli do zaključka.

Slika 9 Grafički prikaz podataka Tablice 1.

Kako bi dobili što precizniji broj, moramo generirati više točaka. Više točaka uzima više vremena za računanje, što je mana Monte Carlo integracije. Sa grafa lako možemo primijetiti da sa povećanjem broja točaka n graf postaje konstantan odnosno sve više se približava funkciji $y = \pi$. Vidimo da nam je za već jako precizne rezultate dovoljan broj točaka 5000 pa nadalje.

Ovo generiranje točaka, tj. vizualizaciju Monte Carlo integracije za računanje površine kruga jako dobro opisuje igra pikado. Ako imamo igrača pikada koji cijelo vrijeme savršeno nasumično baca strelice u jednom trenutku će cijela kvadratna ploča biti ispunjena strelicama. Prebrojimo strelice koje su pogodile metu i uvrstimo sve u jednadžbu (13) te imamo traženi rezultat.

Ovo je bio jednostavan primjer koji smo mogli riješiti na mnoštvo načina i uz pomoć numeričke integracije spomenute u prethodnim poglavljima. Postavlja se pitanje, zašto onda uopće koristiti Monte Carlo integraciju kada ona iziskuje puno vremena? Razlog tome je što neće svi problemi biti jednostavni kao računanje površine kruga. Samim povećanjem broja dimenzija mnoge numeričke metode integracije postaju beskorisne. S druge strane Monte Carlo integracije ostaje i dalje onakva kakva je bila. Jedina razlika je dodatna koordinata našoj točki x_i, odnosno dodali smo još jedan stupanj slobode.

4.1. Pogreška Monte Carlo integracije

Neka je točka x dana sa koordinatama $(q_1, q_2, \ldots, q_n)$ i neka se nalazi u n-dimnezijskom prostoru. Neka je $f(x) = f(q_1, q_2, \ldots, q_n)$ integrabilna funkcija na cijelom tom prostoru. Pomoću Monte Carlo integracije možemo riješiti slijedeći integral

$$I = \int f(x)dx = \int f(q_1, q_2, \ldots, q_n)d^n q. \tag{14}$$

Približno rješenje je dano formulom

$$\frac{1}{N}\sum_{n=1}^{N} f(q_n). \tag{15}$$

Ako primijenimo pretpostavku da n teži beskonačnosti slijedi

$$\lim_{n\to\infty}\frac{1}{N}\sum_{n=1}^{N} f(q_n) = I. \tag{16}$$

Kako bi došli do pogreške Monte Carlo integracije uvodimo varijancu definiranu formulom

$$\sigma(f)^2 = \int (f(x) - I)^2 dx. \tag{17}$$

Kombiniranjem prethodnih jednadžba dobivamo

$$\int dq_1 \ldots \int dq_n \left(\frac{1}{N}\sum_{n=1}^{N} f(q_n) - I\right)^2 = \frac{\sigma(f)^2}{N}. \tag{18}$$

Ovom jednadžbom je pokazano da pogreška Monte Carlo integracije ovisi o faktoru $\frac{1}{\sqrt{N}}$. Ako se prisjetimo ostalih metoda numeričke integracije imali smo da njihova pogreška sa dva puta većim brojem točaka opada dva puta za jednostavnu integraciju, četiri puta za trapezoidnu metodu i 16 puta za Simpsonovu metodu. Vidimo zapravo koliko je Monte Carlo integracija korisna i zašto se često koristi za složene izračune.

Primjer 4.2. Izračunajte metodom Monte Carlo integracije $\int_0^{\pi} sin(x)dx$.

Za početak izračunajmo ovaj integral pomoću trapeziodnog pravila za $n = 8$.

$\Delta x = \frac{b-a}{n} = \frac{\pi-0}{8} = \frac{\pi}{8}$

f(0) = sin(0) = 0

$$f\left(0+\frac{\pi}{8}\right) = f\left(\frac{\pi}{8}\right) = sin\left(\frac{\pi}{8}\right) = 0.383$$

$$f\left(\frac{\pi}{8}+\frac{\pi}{8}\right) = f\left(\frac{\pi}{4}\right) = sin\left(\frac{\pi}{4}\right) = 0.707$$

$$f\left(\frac{\pi}{4}+\frac{\pi}{8}\right) = f\left(\frac{3\pi}{8}\right) = sin\left(\frac{3\pi}{8}\right) = 0.924$$

$$f\left(\frac{3\pi}{8}+\frac{\pi}{8}\right) = f\left(\frac{\pi}{2}\right) = sin\left(\frac{\pi}{2}\right) = 1$$

$$f\left(\frac{\pi}{2}+\frac{\pi}{8}\right) = f\left(\frac{5\pi}{8}\right) = sin\left(\frac{5\pi}{8}\right) = 0.924$$

$$f\left(\frac{5\pi}{8}+\frac{\pi}{8}\right) = f\left(\frac{3\pi}{4}\right) = sin\left(\frac{3\pi}{4}\right) = 0.707$$

$$f\left(\frac{3\pi}{4}+\frac{\pi}{8}\right) = f\left(\frac{7\pi}{8}\right) = sin\left(\frac{7\pi}{8}\right) = 0.383$$

$$f\left(\frac{7\pi}{8}+\frac{\pi}{8}\right) = f(\pi) = sin(\pi) = 0$$

Iskoristimo sada formulu (4) iz odjeljka 2.2.

$$\int_a^b sin(x)dx = \frac{b-a}{2n}[f(x_0) + 2f(x_1) + 2f(x_2) + 2f(x_3) + 2f(x_4) + 2f(x_5) + 2f(x_6) + 2f(x_7) + f(x_8)].$$

Uvrštavanjem poznatih vrijednosti dobivamo

$$\int_0^{\pi} sin(x)dx = 1.974.$$

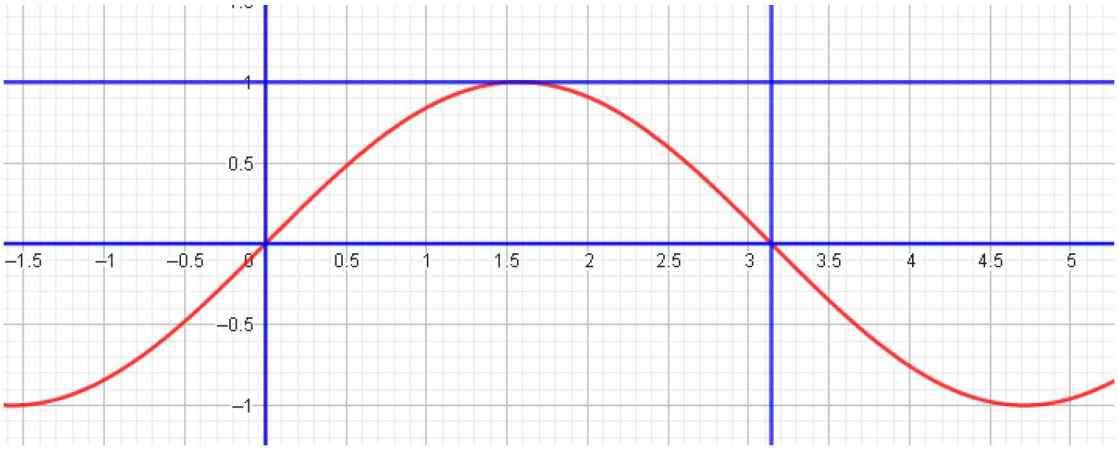
Budući da je ovo vrlo jednostavna funkcija, pomoću Newton-Leibnizove[8] formule dobivamo točan rezultat

$$\int_0^{\pi} sin(x)dx = -cos(\pi) + cos(0) = 2.$$

Skicirajmo sada funkciju *f(x) = sin(x)* i uz nju pravce $x = 0$, $x = \pi$, $y = 0$ i $y = 1$. Kao što već znamo, cilj nam je izračunati površinu ispod grafa *f(x)* u intervalu od 0 do π, ali ovaj puta pomoću Monte Carlo integracije. Razlog zbog kojeg smo uveli ova četiri pravca (Slika 9), je isti kao razlog zbog kojeg smo uveli kvadrat pri računanju površine kruga u primjeru 4.1.

[8] Isaac Newton; (1643. – 1727.), engleski fizičar, matematičar i astronom.
Gottfried Wilhelm Leibniz; (1646. – 1716,), njemački filozof, matematičar, fizičar i diplomat.

Moramo poznavati jednu vanjsku površinu koju ćemo pomnožiti sa omjerom točaka koje se nalaze unutar tražene površine i ukupnog broja točaka.



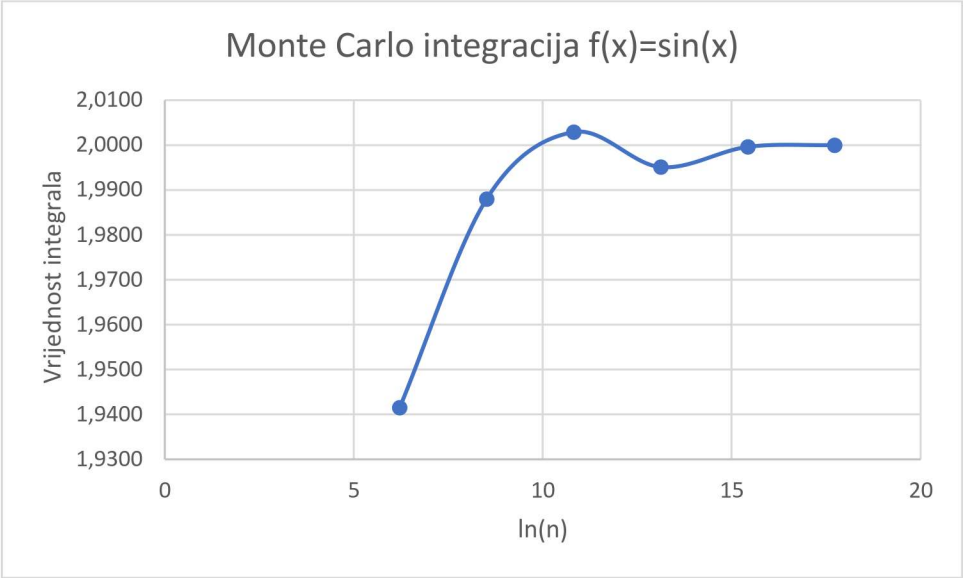
Slika 10 Graf funkcije f(x) = sin(x) i pomoćni pravci.

Rezultati koji se dobiju za različiti broj točaka i odstupanja od poznate vrijednosti integrala su sljedeći

Tablica 2 Monte Carlo integracija funkcije $f(x) = sin(x)$ korištenjem Python koda 2 iz dodatka A.

Broj točaka *n*	Vrijednost integrala	Pogreška	Vrijeme / s
500	1.9415	0.0585	0.0000
5000	1.9880	0.0120	0.0207
50000	2.0029	0.0029	0.1085
500000	1.9951	0.0049	0.9998
5000000	1.9996	0.0004	8.3567
50000000	2.0000	0.0000	82.9629

Grafički prikazana ovisnost broja točaka n o vrijednosti integrala prikazana je na slici 11. Vidimo da je kao i u primjeru 4.1. za neke savršenije rezultate broj potrebnih točaka minimalno 5000.



Slika 11 Ovisnost broja točaka *n* o vrijednosti integrala $\int_0^{\pi} sin(x)dx$.

Primjer 4.3. Izračunajte volumen polukugle jediničnog polumjera.

Za rješavanje ovoga problema potrebno je razmisliti kojim poznatim volumenom ćemo „obgrliti“ polukuglu. Najlakše je uzeti kvadar čije stranice odgovaraju promjeru, a visina polumjeru kugle. Gledano u dvije dimenzije, situacija izgleda isto kao na slici 8. Ono što je bitno za našu točku T(x, y, z) je kada će se ona nalaziti unutar polukugle? Dakle, kada vrijede ove nejednakosti, točka T je unutar gornje polukugle

$$x^2 + y^2 + z^2 \leq 1 \quad \text{i} \quad z \geq 0.$$

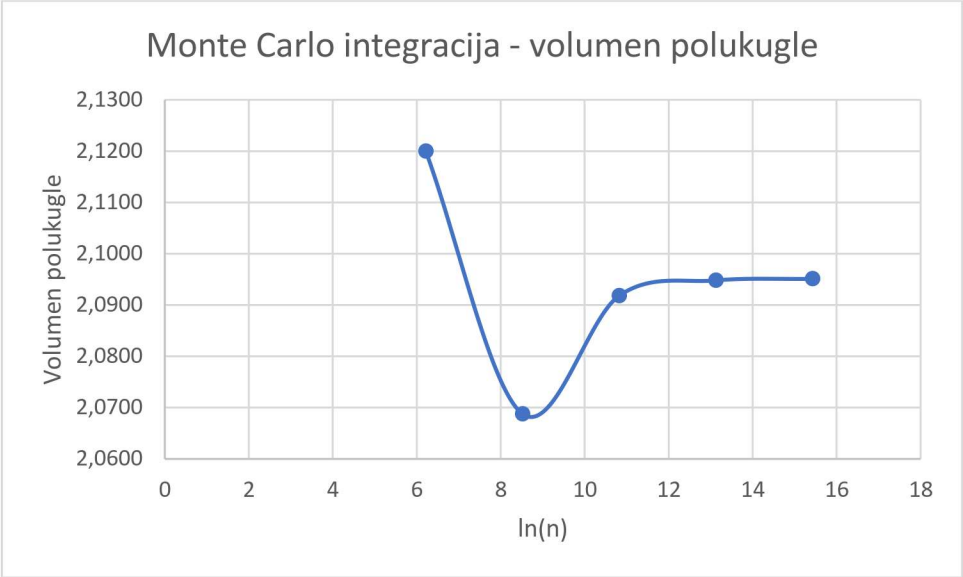
Primijenimo ove podatke u pisanju koda i dobivamo tablicu rezultata.

Tablica 3 Računanje volumena polukugle korištenjem Python koda 3 iz dodatka A.

Broj točaka *n*	Volumen polukugle	Pogreška	Vrijeme / s
500	2.1200	0.0256	0.0010
5000	2.0688	0.0256	0.0105
50000	2.0918	0.0026	0.1354
500000	2.0948	0.0004	1.6101
5000000	2.0951	0.0007	14.1568

Kao što svi već znamo volumen kugle se računa po formuli $V = \frac{4}{3}r^3\pi$. Samim tim je volumen polukugle jednak $V = \frac{2}{3}1^3\pi = \frac{2\pi}{3} = 2.0944$.

Prikažimo i grafički dobivene rezultate.



Slika 12 Grafička ovisnost volumena polukugle o broju generiranih točaka.

Pogledamo li sliku 12, možemo zaključiti da kada broj slučajno generiranih točaka prođe 5000 graf polako postaje konstantan kao u prethodnim primjerima.

Primjer 4.4. Izračunajte volumen nastao presijecanjem kugle i valjaka čiji plašt prolazi promjerom kugle.

Neka je kugla jediničnog polumjera smještena u ishodištu. Neka valjak ima polumjer 0.5 i neka je njegova os paralelna sa z osi i prolazi točkom (0,0.5,0).

Ovo je jedan od zahtjevnijih problema što se tiče same vizualizacije, ali koncept rješavanja problema je i dalje isti. Za točku *T(x, y, z)* razmišljat ćemo kakve sve uvjete mora zadovoljiti da se nađe u presjeku valjka i kugle. U tome će nam više pomoći slika 13. Znamo da svaka točka da bude unutar kugle mora zadovoljiti jednadžbu

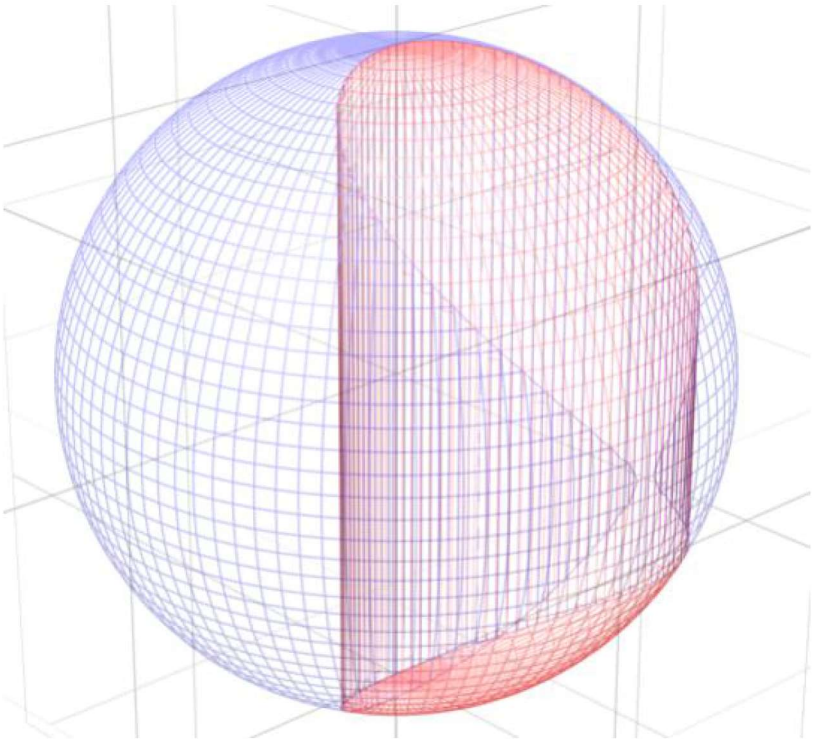
$$x^2 + y^2 + z^2 \leq 1.$$

Osim što želimo da bude unutar kugle želimo da bude i unutar kružnice označene plavom bojom na slici 13, odnosno da bude unutar kružnice koju čini plašt valjka.

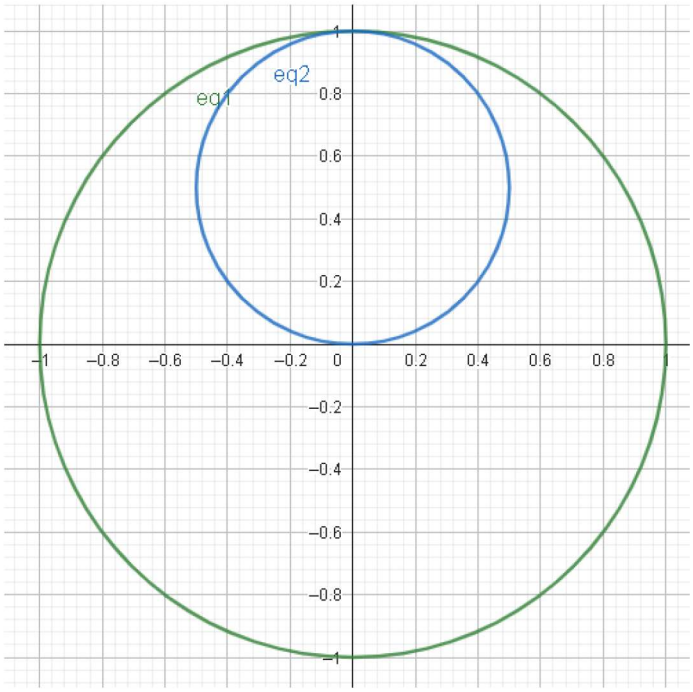
$$x^2 + (y - 0.5)^2 \leq 0.5^2.$$

Kako bi spriječili bilo kakve pogreške dodajemo još dva uvjeta na koordinate točke T

$$y > 0 \; i \; |x| < 0.5.$$



Slika 13 Presjek valjka i kugle u 3D (izvor: Ayars-Computational physics with Python).



Slika 14 Presjek valjka i kugle u 2D .

Sve ove uvjete uvažavamo u pisanju programa i dobivamo tablicu rezultata.

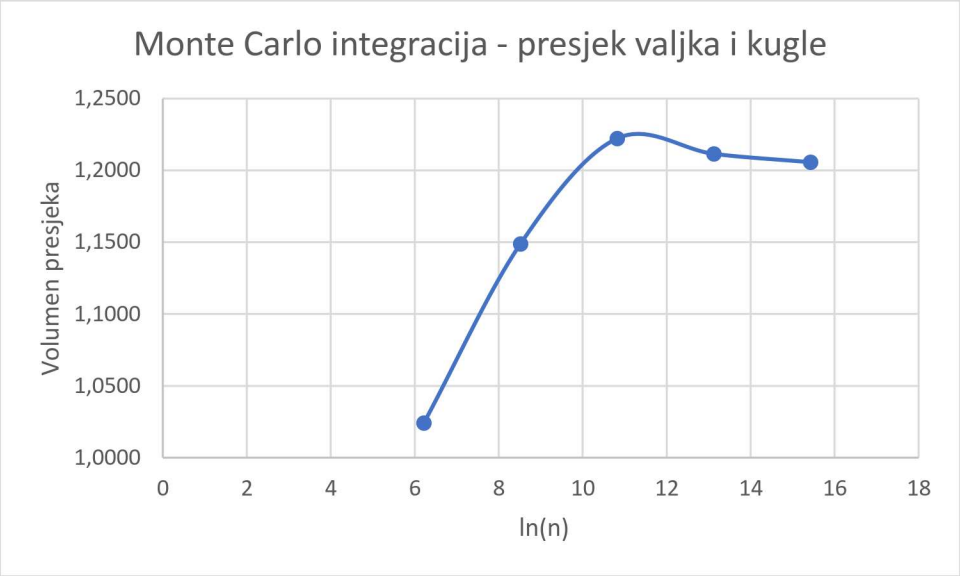
Tablica 4 Monte Carlo integracija presjeka valjka i kugle korištenjem Python koda 4 iz dodatka A.

Broj točaka *n*	Volumen presjeka	Pogreška	Vrijeme / s
500	1.0240	0.1815	0.0000
5000	1.1488	0.0567	0.0200
50000	1.2222	0.0167	0.1919
500000	1.2115	0.0060	2.2034
5000000	1.2057	0.0002	18.8764

Rezultat koji koristimo za usporedbu i izračun pogreške dobiven je drugim metoda integracije i daje nam volumen presjeka kao

$$V = \frac{2r^3\pi}{3} - \frac{8}{9} = 1.2055.$$

Prikažimo te rezultate i grafički.



Slika 15 Grafički prikaz ovisnosti broja slučajno generiranih točaka i volumena presjeka.

Primjer 4.5. Izračunajte volumen četverodimenzijske „kugle“ jediničnog polumjera.

Kako bi došli do volumena četverodimenzijske „kugle“, prvo treba dobro razumjeti kako doći do volumena kugle u tri dimenzije pomoću Monte Carlo integracije. Za kuglu u 3D svaka točka koja se nalazi unutar nje zadovoljava uvjet

$$x^2 + y^2 + z^2 \leq 1.$$

Slično je i sa „kuglom“ u četiri dimenzije. Samo dodajemo još jedan stupanj slobode, odnosno još jednu koordinatu točki T(x, y, z, w). Za točku unutar četverodimenzijske „kugle“ mora vrijediti

$$x^2 + y^2 + z^2 + w^2 \leq 1.$$

Uvrstimo taj uvjet u naš program i dobijemo volumen četverodimenzijske „kugle“. Volumen kocke koje obavija kuglu u 3D je $2^3 = 8$ dok je volumen kocke u 4D jednak $2^4 = 16$. To su poznati volumeni s kojima množimo omjer točaka koje su unutar kugle i ukupnog broja točaka u jednadžbi (13).

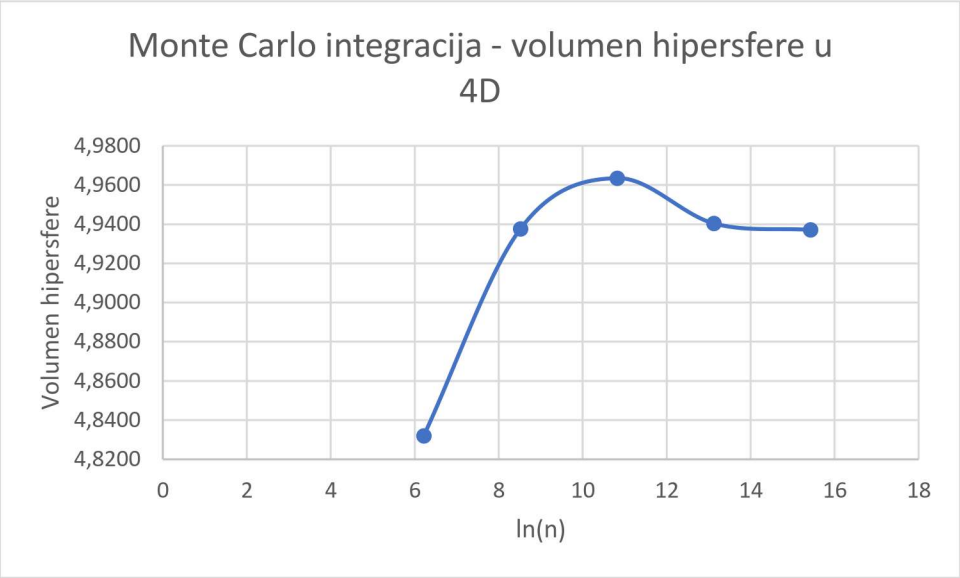
Za izračun pogreške, odnosno odstupanja od prave vrijednosti koristimo formulu za volumen hipersfere u četiri dimenzije

$$V = \frac{1}{2}\pi^2 r^4 = 4.9348.$$

Rezultati koje dobijemo su sljedeći.

Tablica 5 Monte Carlo integracija volumena hipersfere u četiri dimenzije korištenjem Python koda 5 iz dodatka A.

Broj točaka *n*	Volumen hipersfere	Pogreška	Vrijeme / s
500	4.8320	0.1028	0.0012
5000	4.9376	0.0028	0.0269
50000	4.9635	0.0287	0.1899
500000	4.9404	0.0056	2.2743
5000000	4.9371	0.0023	20.4560



Slika 16 Grafički prikaz ovisnosti broja slučajno generiranih točaka i volumena četverodimenzijske „kugle“.

5. Zaključak

Monte Carlo integracija je numerička metoda rješavanja integrala koja na vrlo zanimljiv i jednostavan način neke probleme čini vrlo jednostavnima. Vrijeme izvršavanja joj se povećava kako se povećava broj slučajno generiranih točaka, a greška procjene opada sa faktorom $\frac{1}{\sqrt{N}}$. Upoznali smo se u ovom radu sa dvije vrste slučajnih brojeva te možemo zaključiti da nam je ako želimo što manji broj generiranih točaka bolje koristiti kvazislučajne brojeve zbog toga što su oni više jednoliko raspoređeni. Pseudoslučajni brojevi nisu ništa manje bitni jer pomoću njih smo uspješno riješili nekoliko primjera koji bi se numeričkom integracijom rješavali puno duže vremena. Svaka metoda ima svoje prednosti i mane, ovisno o problemu i našim vještinama sami biramo koja nam najviše odgovara.

6. Dodatak A

Python kod 1 Računanje površine kruga.

```
import random, math
import time
pocetak=time.time()
R = 1
n= 50
ukupno, unutar_kruga = 0, 0
for i in range(n):
    x = random.uniform(-R, R)
    y = random.uniform(-R, R)
    if x ** 2 + y ** 2 <= R ** 2:
        unutar_kruga += 1
    ukupno += 1
povrsina_kruga = 4 * unutar_kruga / ukupno
poznata_vrijednost = math.pi * R ** 2
print("Površina kruga je: ",povrsina_kruga)
print(f"odstupanje od prave vrijednosti je: {abs(povrsina_kruga -
poznata_vrijednost):.08f}")
kraj=time.time()
print("vrijeme izvršavanja je ", kraj-pocetak)
```

Python kod 2 Računanje vrijednosti integrala $\int_0^{\pi} sin(x)dx$.

```
import math
import random as rnd
import time
pocetak=time.time()
N=50000000
brojac=0
a=0.0
b=math.pi
c=1.0
for i in range(N):
    x=rnd.uniform(a,b)
    y=rnd.uniform(a,c)
    tocka=(x,y)
    f=math.sin(tocka[0])
    y_slucajni=tocka[1]
    if y_slucajni<=f:
        brojac+=1
P=b*brojac/N
delta=abs(P-2)
kraj=time.time()
print("integral je %.4f" %P)
print("razlika je %.4f" %delta)
print("vrijeme izvršavanja je %.4f " %(kraj-pocetak))
```

Python kod 3 Računanje volumena polukugle.

```
import math
import random as rnd
import time
pocetak=time.time()
N=5000000
brojac=0
a=0.0
c=1.0
for i in range(N):
    x=rnd.uniform(a,c)
    y=rnd.uniform(a,c)
    z=rnd.uniform(a,c)
    tocka=(x,y,z)
    rk2=tocka[0]**2+tocka[1]**2+tocka[2]**2
    if rk2<1 :
        brojac+=1
V=4*brojac/N
delta=abs(V-(2*math.pi/3))
kraj=time.time()
print("integral je %.4f" %V)
print("razlika je %.4f" %delta)
print("vrijeme izvršavanja je %.4f " %(kraj-pocetak))
```

Python kod 4 Računanje volumena presjeka kugle i valjka.

```
import math
import random as rnd
import time
pocetak=time.time()
N=5000000
brojac=0
a=0.0
b=0.5
c=1.0
for i in range(N):
    x=rnd.uniform(-c,c)
    y=rnd.uniform(-c,c)
    z=rnd.uniform(-c,c)
    tocka=(x,y,z)
    rk2=tocka[0]**2+tocka[1]**2+tocka[2]**2
    rv2=tocka[0]**2+(tocka[1]-b)**2
    if rk2<1 and (rv2<0.25 and tocka[1]>0 and abs(tocka[0])<0.5):
        brojac+=1
V=8*brojac/N
delta=abs(V-(2*math.pi/3-8/9))
```

```
kraj=time.time()
print("integral je %.4f" %V)
print("razlika je %.4f" %delta)
print("vrijeme izvršavanja je %.4f " %(kraj-pocetak))
```

Python kod 5 Računanje volumena četverodimenzijske „kugle“.

```
import math
import random as rnd
import time
pocetak=time.time()
N=5000000
brojac=0
a=1
#3D

# for i in range(N):
#   x=rnd.uniform(-a,a)
#   y=rnd.uniform(-a,a)
#   z=rnd.uniform(-a,a)
#   tocka=(x,y,z)
#   r2=tocka[0]**2+tocka[1]**2+tocka[2]**2
#   if r2<1:
#       brojac+=1
# V=8*brojac/N

# print("Volumen je %.4f" %V)

#4D
for i in range(N):
    x=rnd.uniform(-a,a)
    y=rnd.uniform(-a,a)
    z=rnd.uniform(-a,a)
    w=rnd.uniform(-a,a)
    tocka=(x,y,z,w)
    r2=tocka[0]**2+tocka[1]**2+tocka[2]**2+tocka[3]**2
    if r2<1:
        brojac+=1
V=16*brojac/N
kraj=time.time()
print("Volumen je %.4f" %V)
print("odstupanje od poznate vrijednsoti je %.4f" %abs(V-(1/2)*math.pi**2))
print("vrijeme izvršavanja je %.4f" %(kraj-pocetak))
```

7. Popis literature

[1] Mark Newman. Computational Physics. 2012.

[2] Tao Pang. An Introduction to Computational Physics. Cambridge University Press, 2nd edit, 2006.

[3] Dr. Eric Ayars, Computational Physics With Python, California State University, Chico, 2013.

[4] Introduction to Monte Carlo methods Stefan Weinzierl 1 NIKHEF Theory Group Kruislaan 409, 1098 SJ Amsterdam, The Netherlands, 2000.

[5] Zhiqiang TAN , On a Likelihood Approach for Monte Carlo Integration, Journal of the American Statistical Association, December 2004

[6] Philip J. Davis, Philip Rabinowitz, Methods of Numerical Integration, 2007.