A logo of a building

Description automatically generated

Določeni integrali s pomočjo metode Monte Carlo

Gregor Kuhelj

26.7.2023, Ljubljana

# Določeni integrali in metoda Monte Carlo

Določeni integrali so pomemben matematični koncept, ki se uporablja za izračunavanje ploščin, volumnov, mas in drugih količin. Včasih je težko ali nemogoče izračunati določene integrale analitično, zato se uporabljajo numerične metode, kot je metoda Monte Carlo, ki aproksimiraj rešitev.

Metoda Monte Carlo je statistična metoda, ki temelji na naključnem vzorčenju. Ideja metode za dvorazsežne prostore je naslednja. Predpostavimo, da imamo funkcijo *f(x,y),* ki slika točke iz dvorazsežnega prostora ℝ² v realno število ℝ. Cilj je izračunati določeni integral funkcije *f(x,y)* na področju *[a,b]x[c,d]*, kar predstavlja pravokotno področje v ravnini. Namesto analitičnega izračuna integrala, bomo z metodo Monte Carlo ocenili rezultat z naključnim vzorčenjem točk znotraj tega področja.

Postopek metode Monte Carlo za določeni integral funkcije *f(x,y)* je sledeč:

* Določimo pravokotno področje *[a,b] x [c,d]*, znotraj katerega želimo izračunati integral.
* Generiramo *N* naključnih točk *(x,y)* znotraj tega področja. To lahko naredimo tako, da generiramo naključne vrednosti za *x* med *a* in *b* ter naključne vrednosti za *y* med *c* in *d*.
* Izračunamo vrednost funkcije *f(x,y)* pri vsaki naključni točki *(x,y)*, in izračunamo povprečno vrednost funkcije za celotno področje.
* Približek določenega integrala je enak povprečni vrednosti funkcije, pomnoženi z velikostjo področja *[a,b] x [c,d].*

Metoda Monte Carlo omogoča relativno enostavno oceno določenih integralov za zapletene funkcije ali za področja, kjer analitični izračuni niso izvedljivi. Pomembno je poudariti, da se s povečanjem števila naključnih točk *N* približek metode Monte Carlo približuje pravi vrednosti določenega integrala. Napaka Monte Carlove metode je približno .

Kljub prednostim pa metoda Monte Carlo ni vedno najbolj učinkovita. Za nekatere funkcije in področja lahko zahteva veliko število naključnih točk, da doseže zadovoljivo natančnost. V takih primerih je bolj primerno uporabiti druge numerične metode za izračun določenih integralov.

# Simulacija Monte Carlo v Matlabu

Simulacijo začnemo s tem, da naredimo dve funkciji, in sicer: *monte\_carlo\_doloceni\_integral2D(f,a\_min,a\_max,b\_min,b\_max,N)* in *monte\_carlo\_doloceni\_integral3D(f,a\_min,a\_max,b\_min,b\_max,c\_min,c\_max,N).* Prvo uporabimo za funkcije, ki slikajo iz množice ℝ² v množico ℝ, drugo pa za funkcije, ki slikajo iz množice ℝ³ v množico ℝ.

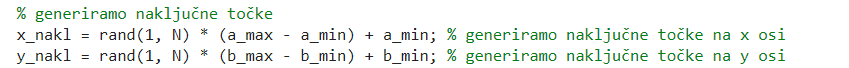
Funkciji obe sprejmeta funkcijo *f*, ki jo želimo aproksimirati, število naključnih točk *N*, ki jih želimo generirati in meje integracije za posamezno os. *a\_min* in *a\_max* predstavljata meji na *x* osi, *b\_min* in *b\_max* pa predstavljata meji na *y* osi. Funkciji se razlikujeta v tem, da funkcija *monte\_carlo\_doloceni\_integral3D* sprejme še dva parametra, in sicer *c\_min* in *c\_max*, ki predstavljata meji na *z* osi.

Pri obeh funkcijah najprej generiramo naključne točke. Pri *monte\_carlo\_doloceni\_integral2D* generiramo točke na *x* in *y* osi, medtem, ko pri *monte\_carlo\_doloceni\_integral3D* generiramo še naključne točke na *z* osi.

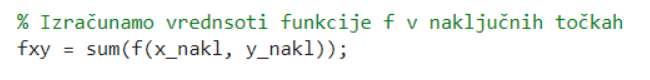
# Funkcija monte\_carlo\_doloceni\_integral2D

Oglejmo si potek aproksimacije integrala funkcije *f*, ki slika iz množice ℝ² v množico ℝ.

Najprej generiramo *N* naključnih števil med 0 in 1. Dobljena števila nato pomnožimo z razliko mej na posamezni osi (*a\_max – a\_min)*, s čimer dobimo naključne vrednosti med 0 in (*a\_max – a\_min)*. Če tem vrednostim prištejemo še *a\_min*, dobimo naključne vrednosti med *a\_max* in *a\_min*. Enako storimo za *y* os.



Nato izračunamo vsoto vrednosti funkcije *f* v naključno generiranih točkah.



Sedaj aproksimiramo integral funkcije s pomočjo metode Monte Carlo. To naredimo tako, da izračunamo ploščino pravokotnika, ki omejuje funkcijo *f*. To sta stranici *a* in *b*, ki ju dobimo kot razliko *(a\_max – a\_min)* in *(b\_max – b\_min)*. Potem še aproksimiramo integral. To naredimo tako, da pomnožimo vsoto vrednosti funkcije v naključnih točkah s ploščino območja, in delimo s številom generiranih točk *N*.S tem dobimo približek za integral funkcije *f* na območju *[a\_min, a\_max]* *x [b\_min, b\_max].*

A close up of a number

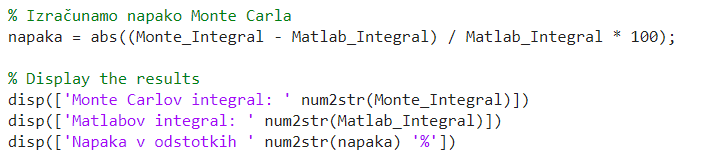
Description automatically generated

Sedaj moramo še izračunati dejansko vrednost integrala. Matlab vrednost integrala iz ℝ² dobi s funkcijo *integral2*, ki kot parametra sprejme želeno funkcijo *f* in meje *x* in *y* osi (*a\_min, a\_max, b\_min, b\_max*).

A group of black and blue letters

Description automatically generated

Sedaj nam preostane samo še to, da primerjamo dobljeno aproksimacijo integrala z dejansko vrednostjo.



Ob klicu funkcije *monte\_carlo\_doloceni\_integral2D* s spodnjimi parametri,

A computer code with black text

Description automatically generated

dobimo naslednje vrednosti.

A close up of words

Description automatically generated

Oglejmo si še nekaj primerov delovanja algoritma.

Za funkcijo *f3* in naslednje podatke,

A white background with black text

Description automatically generated

dobimo naslednje vrednosti.

A black text on a white background

Description automatically generated

Za funkcijo *f4* in naslednje podatke,

A white background with black text

Description automatically generated

dobimo naslednje vrednosti.

A black text on a white background

Description automatically generated

Kot lahko vidimo, je Monte Carlova aproksimacija integrala zelo dobra, saj se aproksimacije od dejanske vrednosti razlikujejo zelo malo.

# Funkcija monte\_carlo\_doloceni\_integral3D

Oglejmo si še delovanje funkcije *monte\_carlo\_doloceni\_integral3D.* Funkcija deluje enako kot *monte\_carlo\_doloceni\_integral2D*, le da ima 3D različica še 2 dodatna vhodna podatka (meji na *z* os).

Enako kot pri 2D različici funkcije *monte\_carlo\_doloceni\_integral3D*, si najprej generiramo naključne vrednosti za osi *x, y, z*.

A close-up of a math equation

Description automatically generated

Nato izračunamo vsoto vrednosti funkcije *f* v naključno generiranih točkah.

A close up of a text

Description automatically generated

Naslednje izračunamo prostornino intervala, znotraj katerega je naša funkcija. To naredimo tako, da pomnožimo razlike med mejami. Nato aproksimiramo vrednost integrala po enaki formuli kot pri 2D različici, le da sedaj ne gledamo površine lika, temveč prostornino telesa (torej gledamo tri osi namesto dveh).

S pomočjo vgrajene funkcije *integral3*, ki sprejme 7 parametrov (funkcijo, ki jo želimo integrirati in meje integriranja na *x, y, z* oseh) izračunamo še dejansko vrednost integrala.

A close-up of a math problem

Description automatically generated

Sedaj nam preostane samo še to, da primerjamo dobljeno aproksimacijo integrala z dejansko vrednostjo.

A computer code with many colored text

Description automatically generated with medium confidence

Ob klicu funkcije *monte\_carlo\_doloceni\_integral3D* s spodnjimi parametri,

A white background with black text

Description automatically generated

dobimo naslednje vrednosti.

A black text on a white background

Description automatically generated

Oglejmo si še nekaj primerov delovanja algoritma.

Za funkcijo *f3* in naslednje podatke,

A white background with black text

Description automatically generated

dobimo naslednje vrednosti.

A black text on a white background

Description automatically generated

Za funkcijo *f6* in naslednje podatke,

A white background with black text

Description automatically generated

dobimo naslednje vrednosti.

A number on a white background

Description automatically generated

Zopet lahko opazimo, da je aproksimacija zelo dobra, saj so odstopanja od dejnaskih vrednosti zelo majhna.

# Viri

1. Wikipedia, *Monte Carlo integration*, <https://en.wikipedia.org/wiki/Monte_Carlo_integration> [26.7.2023]
2. Phill Parisi, *2022 How to Perform Monte Carlo Integration in MATLAB | MATLAB Tutorial*, <https://www.youtube.com/watch?v=JwRTKQyFSjA&t=426s> [26.7.2023]
3. GeeksForGeeks, *Monte Carlo integration in Python*, <https://www.geeksforgeeks.org/monte-carlo-integration-in-python/> [26.7.2023]
4. Dr Manab, *Multidimensional numerical integration in Matlab | Monte Carlo integration,* [*https://www.youtube.com/watch?v=ctM1ehAzHe0*](https://www.youtube.com/watch?v=ctM1ehAzHe0) *[26.7.2023]*