# 109: Integracije z metodo Monte Carlo

Peter Rupnik

19. december 2018

# 1 Prva naloga

## 1.1 Naloga

Oglej si presek treh medsebojno pravokotnih valjev z enakimi polmeri, katerih osi so poravnane s koordinatnimi osmi. Izračunaj prostornino in vztrajnostni moment dobljenega telesa,

- če je gostota konstantna,
- če se gostota spreminja z razdaljo od središča kot $\rho = (r/r_0)^p$ in je $r_0$ polmer telesu očrtane krogle, r pa razdalja od središča telesa. Kakšen učinek ima vrednost parametra p?

Oceni natančnost rezultatov.

## 1.2 Implementacija

Osnovna enačba za integracijo po metodi Monte Carlo je

$$\hat{\theta} \approx \sum_{i=1}^{N} w_i(\vec{r})\, g(\vec{r})\, f(\vec{r}), \tag{1}$$

pri čemer s *cenilko* $\hat{\theta}$ ocenjujemo v prvem primeru maso in v drugem vztrajnostni moment. Prepišem si (1) v obliko, ki jo bom potreboval za izračun mase:

$$\hat{m} = \frac{V_0}{N} \sum_{i=1}^{N} \rho(\tilde{\mathrm{r}})\, \Delta(\tilde{\mathrm{r}}_\mathrm{i}), \tag{2}$$

kjer je $\hat{m}$ ocena za maso, $V_0$ volumen, v katerem integriramo po metodi Monte Carlo, $\rho(\tilde{\mathrm{r}}_\mathrm{i})$ gostota telesa na mestu $\tilde{\mathrm{r}}$, $\Delta(\tilde{\mathrm{r}}_\mathrm{i})$ pa je simbolni operator, ki preveri, ali je izžrebana točka znotraj telesa:

$$\Delta(\tilde{\mathrm{r}}_\mathrm{i}) = \begin{cases} 1: & \tilde{\mathrm{r}}_\mathrm{i} \text{ znotraj integriranega telesa,} \\ 0: & \text{sicer} \end{cases} \tag{3}$$

Slično lahko napravim tudi za izračun ocene vztrajnostnega momenta (okrog težišča telesa) po [1]:

$$I = \int_V \tilde{\mathrm{r}}^{\,2}\,\mathrm{d}m = \int_V \tilde{\mathrm{r}}^{\,2}\,\rho(\tilde{\mathrm{r}})\,\mathrm{d}V, \tag{4}$$

to pa lahko brž̌čas aproksimiram takole:

$$\hat{I} = \frac{V_0}{N}\sum_{i=1}^{N} \tilde{\mathrm{r}}^{\,2}\,\rho(\tilde{\mathrm{r}})\,\Delta(\tilde{\mathrm{r}}_\mathrm{i}) \tag{5}$$

### 1.2.1 Analiza časovne zahtevnosti posameznih načinov generacije enakomerno porazdeljenih slučajnih števil

Na tri različne načine sem generiral $10^6$ števil in jih seštel, kot neke sorte primitivno analogijo procesa, ki omogoča metodo Monte Carlo.

1. `sum(np.random.uniform(low=-1, high=1, size=int(1e6)))`

   Trajanje: `206 ms ± 2.37 ms`

2. ```
   rez = np.zeros(int(1e6))
   for i in range(int(1e6)):
       rez[i] = np.random.uniform(low=-1, high=1)
   rez.sum()
   ```

   Trajanje: `3.88 s ± 11.1 ms`

3. ```
   rez = np.random.uniform(low=-1, high=1, size=int(1e6))
   rez.sum()
   ```

   Trajanje: `27.1 ms ± 294 µs`

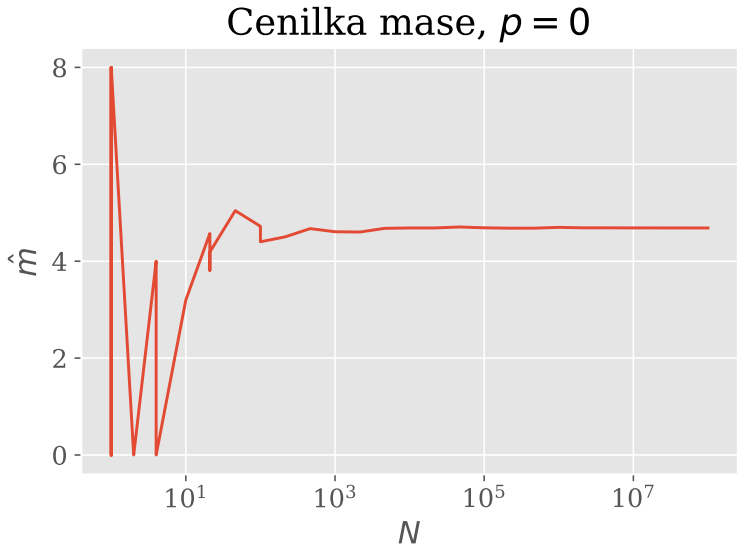
Evidentno je, da je rede velikosti bolje operirati z `numpy` seznami in uporabljati pripadajoče funkcije iz tega modula. Morda je struktura `for` zank intuitivno bolj razumljiva, a se očitno izplača potruditi in kodo vektorizirati.

### 1.2.2 Zakaj se kode ne izplača vektorizirati

Moja zgoraj opisana metoda ima napako, ki bo jo moral predvideti že *a priori*, a je nisem: ko z njo delamo Monte Carlo, najprej napravi `numpy.array` (v nadaljevanju: array) oblike $(N, 3)$ in ga drži 'v glavi' cel čas. Dejansko v programu pravimo "Prosim, seštej mi z gostoto utežene točke danega arraya, ki izpolnjujejo nek pogoj.", kar je super, koncizno, enovrstično, nenazadnje bolj *pythonično*... a zaradi velike dolžine arraya tak poskus klavrno strmoglavi. Če pa uporabimo zanko, v vsaki iteraciji katere generiramo 3 števila, preverimo pogoj, in nato neko spremenljivko samo zvečamo za prispevek trenutne iteracije, pa sicer zadeva ne bo tako počasna, a bo delovala tudi za velike $N$.

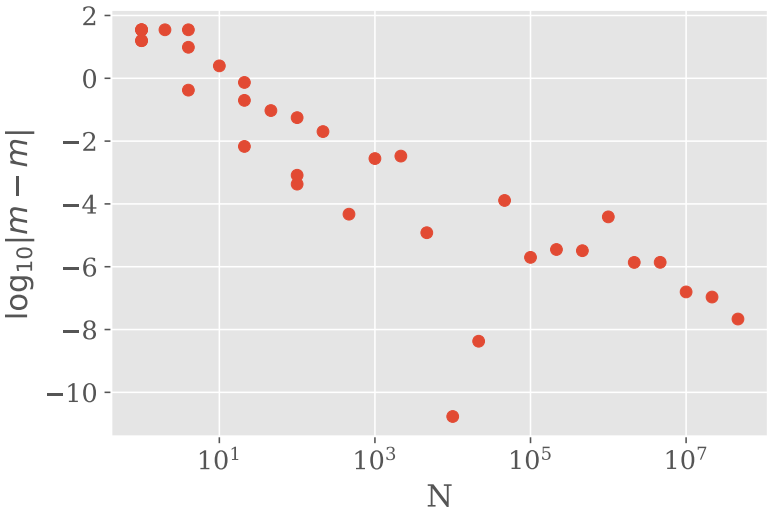
## 1.3 Redosled

Za izračun mase telesa: pripravil sem funkcijo, ki z uporabo `for` zanke preizkuša moj volumen, ki znaša $[-1, 1]^3$ (očitno bi zaradi simetrije lahko integracijo izvajal le v enem izmed oktantov, vendar se za to opcijo nisem odločil.). Funkcijo sem klical večkrat z različnim številom generiranih točk $N$ in spremljal rezultate. Množica vseh $N$-jev je bila porazdeljeno logaritmično enakomerno med $10^0$ in $10^8$. Zanimala me je predvsem vrednost $\hat{m}$. Hitro sem opazil, da je proces časovno dokaj zahteven, sploh če integracijo ponavljamo zaporedno z vedno večjimi $N$. Čas čakanja na izračune sem izkoristil za definiranje funkcije, ki sproži zvočni signal, funkcijo bom v nadaljevanju klical ob koncu izračunov. Kljub

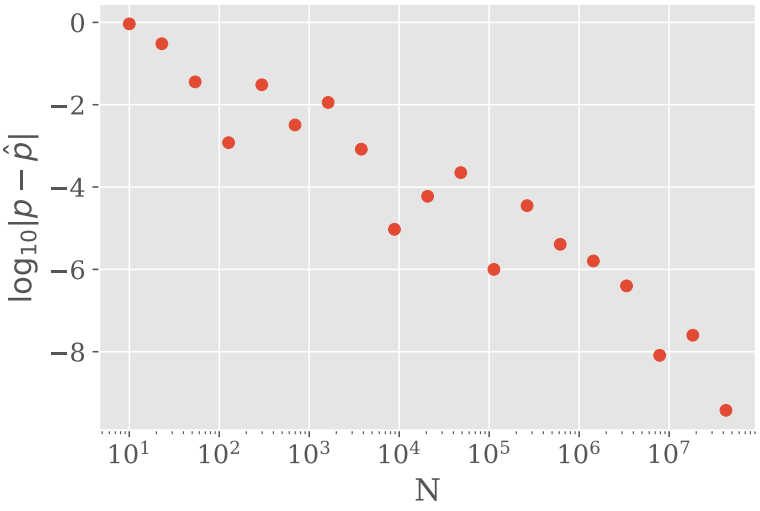



Slika 1: Potek vrednosti cenilke mase za prostorsko časovno gostoto in 25 različnih $N$-jev, porazdeljenih enakomerno v logaritemskem prostoru.

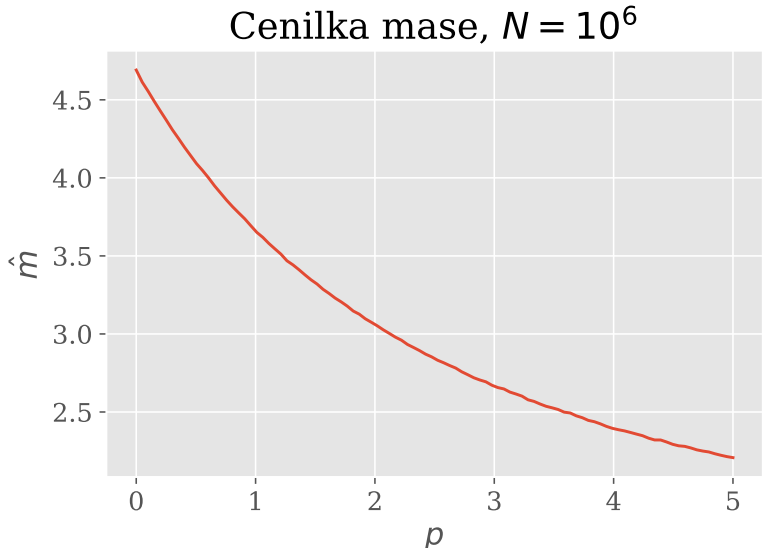
vsemu se mi zdi, da postopek traja sumljivo počasi. Krivdo iščem v sejanju mojega psevdoslučajnega generatorja, ki sem ga seedal preko dvobajtnega števila, ki sem ga dobil iz `dev/random`. Za ta generator je značilno, da 'blokira', če mu umanjka *entropije* [3]. Da bi potrdil ali ovrgel svoje sume, sem z metodo Monte Carlo generiral hitre serije zaporednih integracij z relativno nizkimi $N$ z dvema funkcijama, prva je seedala generator z `dev/random`, druga pa z `dev/urandom`, ki problema z 'blokiranjem' nima [3]. Razmerje časov je znašalo približno 60, zato sem popravil svojo kodo in od zdaj dalje seedal vse z `dev/urandom`. Kljub temu, da se seedanje izvaja le na začetku vsake Monte Carlo integracije, je čas med pričetki očitno premajhen, da bi strojno zgenerirali dovolj naključnih bajtov.



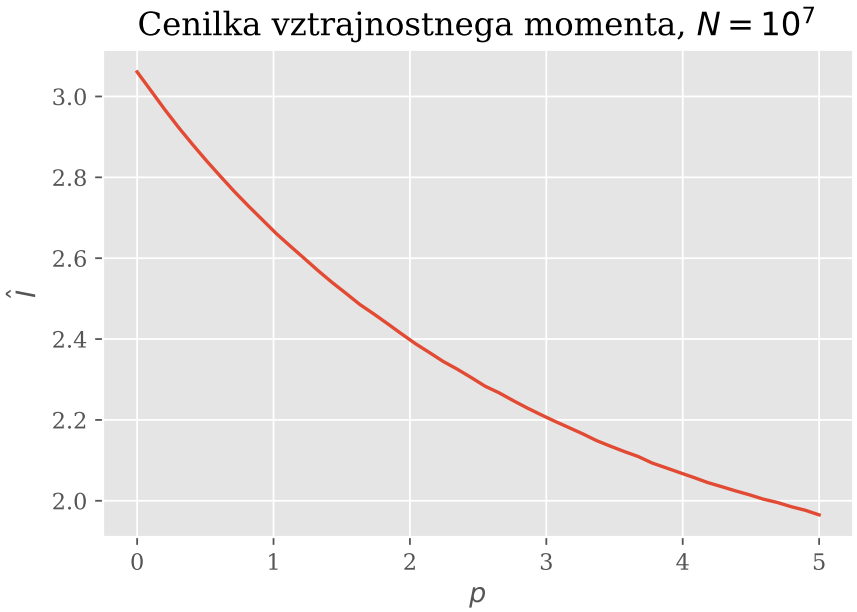

Slika 2: Boljši prikaz konvergence integracije. Lepo je viden pričakovan trend padanja z naklonom $N^{-0.5}$. Eksaktno vrednost mase povzemam po [2]. Iz tega grafa lahko ocenim negotovost integracije za poljuben $N$.




Slika 3: Spreminjanje vrednosti cenilke za vztrajnostni moment v odvisnosti od števila poskusov $N$. Lepo je viden pričakovan trend padanja z naklonom $N^{-0.5}$. Ker tu ne poznam eksaktne vrednosti, sem kot točno vzel vrednost, ki sem jo dobil z integracijo pri največjih $N$. Ponovno velja, da lahko na podlagi teh podatkov ocenim negotovost integracije.




Slika 4: Vpliv parametra $p$ na maso telesa. Odvisnost me je rahlo presenetila, a je razumljiva, ko se spomnimo, da se gostota spreminja kot $\rho(r) = \left(\frac{r}{r_0}\right)^p$, definicija s tem da je $r_0$ definiran kot radij telesu očrtanega kroga, kar pomeni, da bo razmerje $r/r_0$ za vse točke v telesu manjše, kvečjemu enako 1, zaradi česar dobimo odvisnost, ki je na sliki.




Slika 5: Za vsako točko na grafu sem z Monte Carlo integracijo z $10^7$ točkami izračunal vztrajnosti moment. Na vodoravni osi je parameter $p$, območje sem diskretiziral v 50 enakomerno razporejenih točk. Celoten izračun je trajal 4 ure in 21 minut.

# 2 Druga naloga

## 2.1 Naloga

V krogli se rojevajo žarki gama. Njihova povprečna prosta pot v snovi, iz katere je krogla, je enaka radiju krogle. Kolikšen delež fotonov uide iz krogle? Kako se verjetnost pobega spreminja z razmerjem povprečne proste poti in radija krogle?

## 2.2 Pristop

Kot sugerirano na predavanju bom obravnavo poenostavil tako, da kroglo vsakič ob nastanku fotona zavrtim tako, da se foton rodi na isti črti. Tedaj lahko točko nastanka opišem s parametroma $r$, oddaljenostjo od središča, in $\theta$, azimutalnim kotom.

Iz prejšnje naloge že vemo, da bomo sferično enakomerno porazdeljene podatke dobili, če transformiramo $u_1, u_2 \in \mathcal{U}(0,1)$ takole:

$$r_i = \sqrt[3]{u_1} \tag{6}$$
$$\theta_i = \arccos(2u_2 - 1) \tag{7}$$

Uporabil bom tudi izraz za dolžino tetive, ki jo odreže tak žarek in znaša:

$$d = -r\cos\theta + R\sqrt{1 - \frac{r^2}{R^2}(1 - \cos^2\theta)}, \tag{8}$$

kjer je $R$ radij krogle.

Poti $s$, ki jih prepotuje foton pred absorpcijo, so porazdeljene eksponentno:

$$f(s) = \frac{\mathrm{d}P}{\mathrm{d}s} = \frac{1}{\mu}e^{\frac{s}{\mu}}, \tag{9}$$

kjer je $\mu$ absorpcijska dolžina, iz česar lahko izračunamo, kako naj transformiramo slučajna števila iz $\mathcal{U}(0,1)$:

$$s_i = -\mu\,\ln(1 - u_i). \tag{10}$$

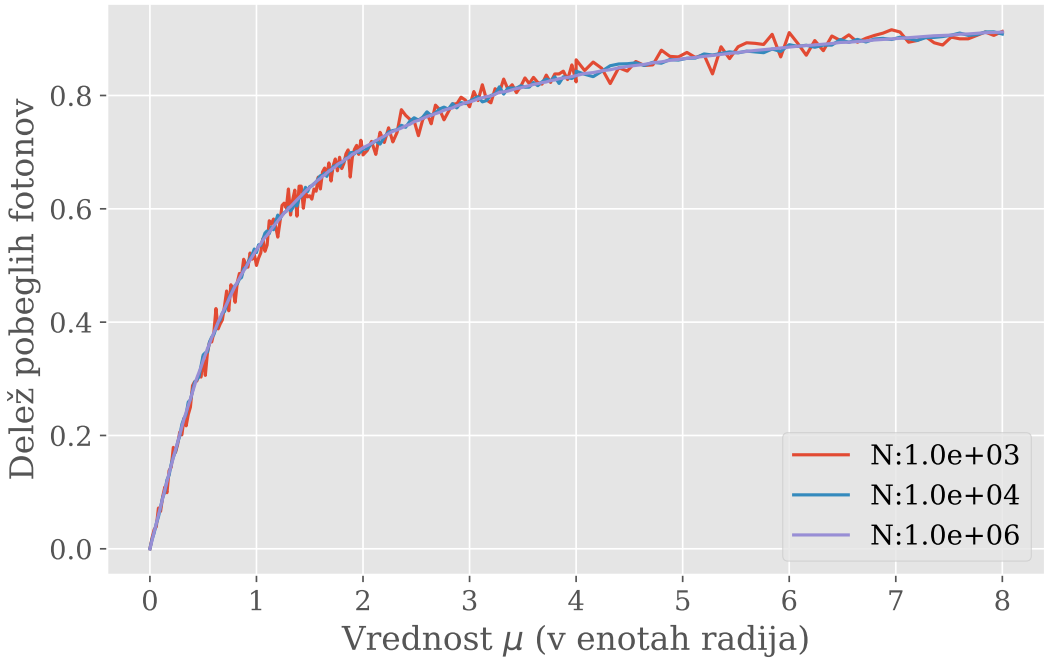
Ko tako generiramo pozicijo, smer in dolžino poti fotona, slednjo primerjamo z dolžino tetive 8, kar nam pove, če fotonu uspe 'pobegniti'.

### 2.2.1 Spet o vektorizirani kodi...

Tokrat sem se poslužil vektorirane kode, ki sem jo omenil zgoraj. Ideja je bila sledeča: četudi ne morem hkrati generirati poljubno mnogo števil in jih obdelovati vektorsko, lahko svojo vektorsko kodo še vedno uporabim znotraj zanke, ki namesto obdelave vsakega elementa posebej obdela denimo $10^7$ elementov, in nato rezultate združi v končni rezultat. Razlika je v tem, da se zanka ponavlja *dosti dosti redkeje* kot bi se sicer, znotraj zanke pa se zanašam na modul `numpy`, ki optimizira numeriko po svoji volji. Tako sem dosegel občutna povečanja hitrosti (faktor 10—100).

## 2.3 Rezultati

Rezultate prikazujem na sliki 6.




Slika 6: Za različne vrednosti absorpcijske dolžine $\mu$ sem z metodo Monte Carlo računal delež pobeglih fotonov. Spreminjal sem tudi število $N$, torej število trojk $(r, \theta, s)$. Opazno je, da pri $N = 10^6$ že dobimo zadostno konvergenco, da fluktuacij na oko ne opazimo več. Diskretizacija vodoravne osi je bila neenakomerna, najgostejša v najstrmejšem delu in najredkejša na položnem. Skupno je na vodoravni osi poračunanih okrog 200 točk. Čas simulacije je trajal slabe 3 minute.

# 3 Tretja naloga

## 3.1 Naloga

Model nevtronskega reflektorja: tok nevtronov vpada pravokotno na ploščo, v kateri se nevtroni sipljejo in nič ne absorbirajo, pri čemer je njihova prosta pot enaka polovici debeline plošče. V poenostavljenem modelu privzamemo, da se sipljejo samo naprej in nazaj, in to z enako verjetnostjo. Kakšna je porazdelitev po številu sipanj? Kolikšna je prepustnost reflektorja? Oceni natančnost. Nekoliko bolj realen je model z izotropnim sipanjem. Z njim preveri, koliko so rezultati poenostavljenega modela uporabni.

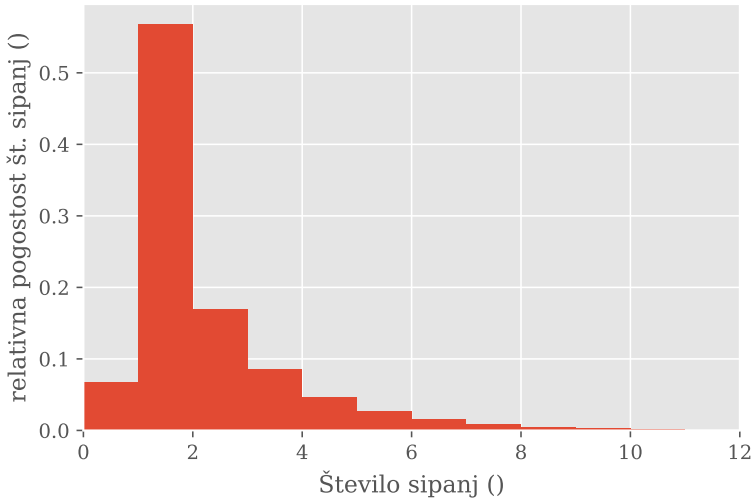
**(neobvezno)* Določi kotno porazdelitev odbitih in prepuščenih nevtronov ter odvisnost prepustnosti od debeline plošče v modelu z izotropnim sipanjem.

## 3.2 Postopek

Formalizem za generiranje slučajnih dolžin po eksponentni porazdelitvi imamo pripravljen že iz prejšnje podnaloge. Problem sem zastavil brezdimenzijsko, v poenostavljenem podproblemu sem vsak korak simulacije iniciiral v koordinatnem izhodišču, nato pa korake povečeval za slučajno dolžino poti, pomnoženo s slučajnim predznakom, torej bodisi +1 ali -1. To sem počel iterativno, vse dokler ni položaj presegel 1 ali se vrnil pod 0. Beležil sem število sipanj in število pobeglih delcev, bodisi nazaj ali skozi reflektor, natančneje:

če je bila koordinata nevtrona v $i$-tem koraku manjša od 0, sem število sipanj povečal za 1, število reflektiranih nevtronov povečal za 1 in končal iteracijo. Ča je bila koordinata delca večja od 1, sem število transmitiranih nevtronov povečal za 1, števca sipanih pa nisem spreminjal, iteracija se prekine. Sicer sem število sipanih povečal za 1 in nadaljeval iteracijo.

Za $10^8$ simuliranih prehodov sem pogledal normaliziran histogram števila sipanj, prikazujem ga na sliki 7. Transmisivnost znaša 0.75006, refleksivnost pa 0.24994.




Slika 7: Porazdelitev po številu sipanj. Kot opazimo, nekaj nevtronov preleti plast brez interakcij, daleč najpogostejši pa so nevtroni, ki interagirajo le enkrat.

Nadalje me je zanimala odvisnost transmisivnosti in reflektivnosti od parametra $\mu$, zato sem si spet pripravil množico različnih $\mu$–jev in za vsakega od njih simuliral $10^4$ nevtronskih interakcij. Rezultat prikazuje slika 8.

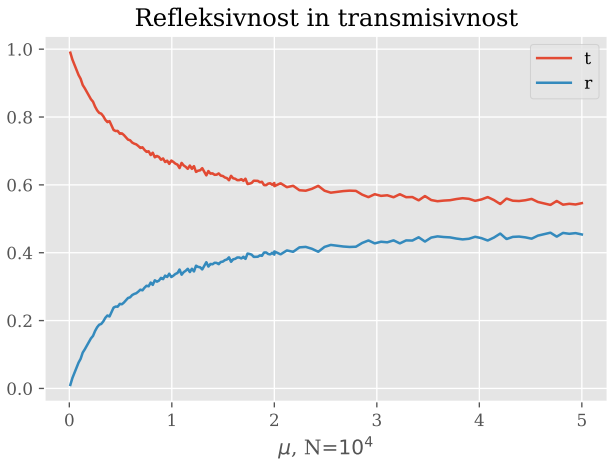
## 3.3 Izotropen model

Če zahtevamo, naj se po sipanju nevtron odbije v poljubno smer, nas v ploskovitem, v dve strani neomejenem reflektorju zanima samo koordinata v smeri vpadnega nevtrona. Zaradi tega popravim svoje algoritme; prej sem generirano slučajno dolžino bodisi množil z 1 ali -1, zdaj pa generiram naključni azimutalni kot $\theta$ po postopkih iz prejšnjih nalog in naključno generirano dolžino množim s $\cos\theta$. Porazdelitev po številu sipanj najdemo na sliki 9.

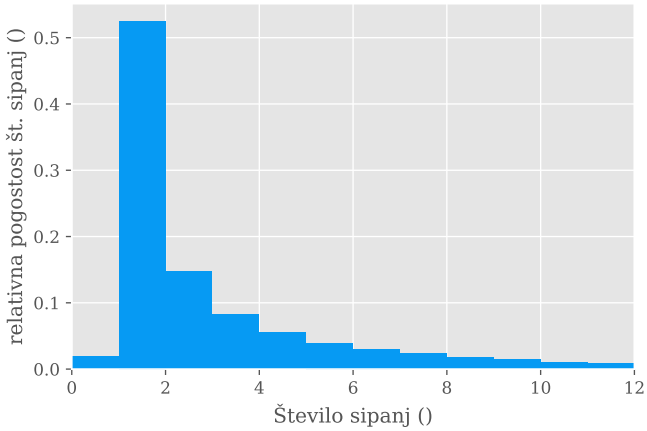
## 3.4 Ocena negotovosti izračunanih količin

Z utemeljitvijo, da 'opletanje' okrog pričakovano gladke krivulje na slikah 10 in 8 *izgleda* konstantne amplitude po celotnem območju, sem ocenil negotovost posameznih količin pri konstantnem $N = 10^5$ za oba modela:

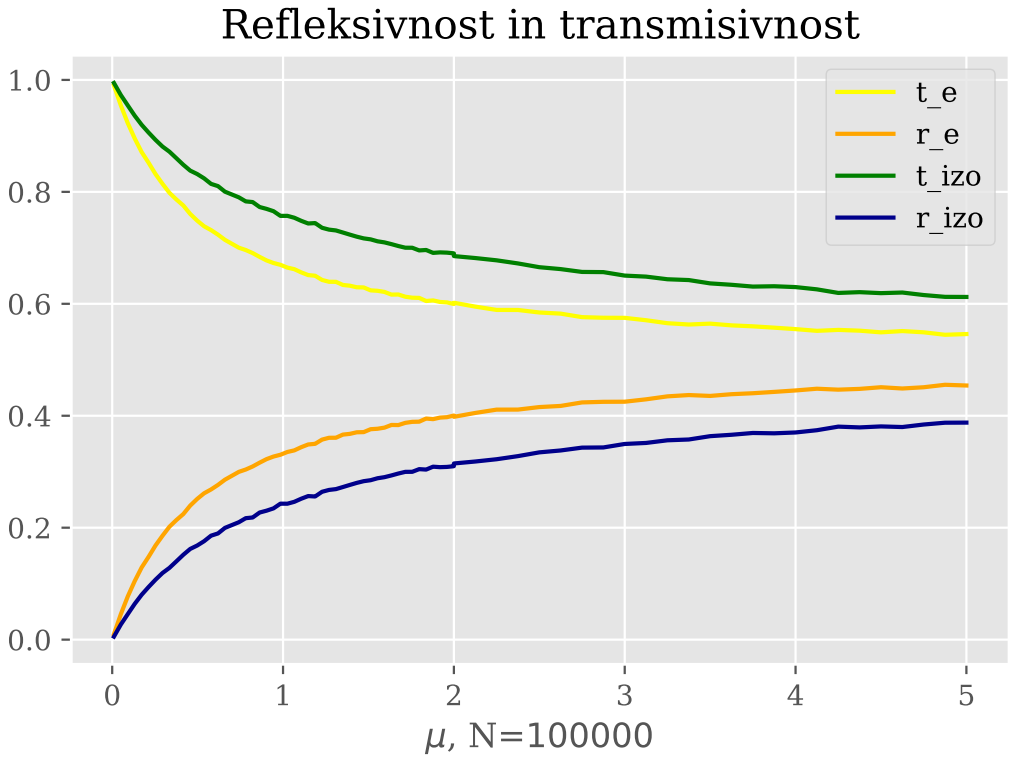
$$\begin{bmatrix} \sigma_{r_{eno}} \\ \sigma_{t_{eno}} \\ \sigma_{r_{izo}} \\ \sigma_{t_{izo}} \end{bmatrix} = \begin{bmatrix} 0.003016 \\ 0.003016 \\ 0.002299 \\ 0.002299 \end{bmatrix} \tag{11}$$




Slika 8: Transmisivnost in reflektivnost v odvisnosti od parametra $\mu$. Z razmislekom pojasnimo limiti $\lim \mu \to 0$ in $\lim \mu \to \infty$: ko se sipalna dolžina manjša, se prosta pot poveča, zaradi česar večina nevtronov preleti reflektor brez sipanja, to pa posledično pomeni 100% transmisivnost. Na drugi skrajnosti pa večanje $\mu$ pomeni isto kot večanje debeline reflektorja. Ko nevtron vstopi, se zelo verjetno sipa in z naključno hojo med plastema koleba med obema mejnima ploskvima, zaradi česar je enako verjetno, da se prebije čez katerokoli izmed njiju. Simulirano je bilo $10^4$ vpadnih nevtronov za vsako točko v $\mu$ prostoru, le–ta pa je bil diskretiziran neenakomerno, z večimi točkami na strmejšem delu.




Slika 9: Porazdelitev po številu sipanj za izotropno sipanje. Kot opazimo, je porazdelitev kvalitativno enaka poenostavljenemu modelu s sipanjem naprej–nazaj, daleč najpogostejši so nevtroni, ki interagirajo le enkrat, večkratno sipanje pa je manj verjetno.




Slika 10: Transmisivnost in reflektivnost v odvisnosti od parametra $\mu$ za enostaven model $(t_e, r_e)$ in model z izotropnim sipanjem $(t_{izo}, r_{izo})$. Število simuliranih nevtronov je v vsaki točki na $\mu$ osi znašalo $10^5$, čas simulacije pa okrog 23 minut.

# Literatura

[1] https://en.wikipedia.org/wiki/Moment_of_inertia#Angular_momentum, dostopno dne 19. december 2018.

[2] https://en.wikipedia.org/wiki/Steinmetz_solid, dostopno dne 19. december 2018.

[3] https://en.wikipedia.org/wiki//dev/random, dostopno dne 19. december 2018.