

Univerza v Ljubljani
Fakulteta za *matematiko in fiziko*



Naključna števila in integracije z metodo Monte Carlo

7. naloga pri Modelski Analizi 1

Avtor: Marko Urbanč (28232019)
Predavatelj: prof. dr. Simon Širca
Asistent: doc. dr. Miha Mihovilovič

23.11.2023

Kazalo

1	Uvod	2
2	Naloga	2
2.1	Zvezdasti lik	2
2.2	Rojevanje žarkov gamma	2
2.3	Nevtronski reflektor	3
3	Opis reševanja	3
3.1	Zvezdasti lik	3
3.2	Rojevanje žarkov gamma	4
3.3	Nevtronski reflektor	4
4	Rezultati	4
4.1	Zvezdasti lik	5
4.2	Rojevanje žarkov gamma	9
4.3	Nevtronski reflektor	10
5	Komentarji in izboljšave	12

1 Uvod

Danes bojo pomembna naključna števila, saj bomo z metodo Monte Carlo računali integrale. Dobro je, da se zavedamo dejstva, da naključna števila, ki nam jih da recimo `np.random` niso dejansko naključna, ampak so generirana po nekem algoritmu, ki je determinističen. Težko (beri: nemogoče) je računalniku dati natančna navodila o tem kako naj nekaj nenatančno oz. poljubno naredi. Zato se poslužujemo nekaterih algoritmov, ki nam pomagajo pri generiranju naključnih števil. V tem primeru bomo uporabili `numpy.random.uniform`, ki nam vrne naključno število iz enakomerne porazdelitve na intervalu $[0, 1)$. To my surprise je, da `numpy` uporablja Mersenne Twister algoritem. In hindsight bi lahko vzel tudi kakšen preprostejši algoritem, ki bi bil hitrejši, čeprav `numpy` uporablja nekaj trikov, da je hitrejši.

Pri Monte Carlo integraciji naključno izbiramo točke v prostoru, kjer imamo neko omejeno območje, ki ga integriramo. Recimo, da je to območje $[a, b] \times [c, d]$, kjer smo od tega še nekaj odrezali, da dobimo trikotnik v profilu. V takem primeru bi izbirali naključne točke (x, y) , kjer je $x \in [a, b]$ in $y \in [c, d]$ in vsakič, ko izberemo točko, preverimo, če je znotraj območja, ki ga integriramo (v temu preprostem primeru, ko nekaj odrežemo, preverimo ali je točka znotraj trikotne prizme). Če je točka znotraj, potem inkrementiramo števec, ki nam pove koliko točk je znotraj. Na koncu integriramo tako, da delimo število točk znotraj z številom vseh točk in to pomnožimo z volumnom območja, ki ga integriramo. Tako smo dobili volumen, v našem primeru, trikotne prizme. Če imamo neko funkcijo $f(x, y)$, ki jo integriramo, potem to funkcijo pomnožimo v vsaki točki znotraj območja, ko jo izberemo. To si bomo pogledali na treh primerih.

2 Naloga

Danes sem zgrešil rok za oddajo in namesto, da bi imel edini prosti večer, pišem report. Posledično sem nekoliko bolj redkobeseden, kot bi bil sicer, zato kar pojdimo na naloge.

2.1 Zvezdasti lik

Najprej si pogledajmo zvezdasti lik, ki ga omejuje ploskev:

$$\sqrt{|x|} + \sqrt{|y|} + \sqrt{|z|} = 1 \quad (1)$$

Naloga želi, da izračunamo maso in vztrajnostni moment tega lika v dveh primerih. Prvi primer je, ko je gostota konstantna in v drugem primeru, ko je gostota odvisna od razdalje od izhodišča z neko potenco r^p .

2.2 Rojevanje žarkov gamma

V drugi nalogi si zamislimo, da se v središču krogle rojevajo žarki gamma. Naloga želi, da izračunamo verjetnost za pobeg žarka iz krogle in pogledamo, kako se ta verjetnost spreminja z povprečno prosto potjo žarkov in radijem krogle.

2.3 Nevtronski reflektor

V zadnji nalogi si pogledjmo nevtronski reflektor, ki je pravzaprav plošča z neko debelino. Na reflektor pošljamo tok nevtronov, ki se v njej sipajo in absorbirajo. Naloga želi, da izračunamo verjetnost, da se nevtron, ki ga pošljemo v reflektor, pride skozi reflektor. Povprečna prosta pot je enaka polovici debeline reflektorja. Dodatno lahko potem stvari še malo bolj zakompliciramo in dodamo še možnost gibanja v prečnih smereh.

3 Opis reševanja

Ker imam rad HPC, sem za namen naloge napisal integrator, ki je sposoben delati na več jeder. To sem naredil tako, da sem si napisal razred `NumberNecromancer`, ki sprejme funkcijo, ki jo integriramo skupaj z območjem, ki ga integriramo in število točk, ki jih generiramo. Nato razred razdeli točke na več delov, ki jih lahko integriramo na različnih procesorjih. Točke se generirajo na vsakem procesorskem jedru ločeno, tako da je manj pošiljanja podatkov med procesorji in je manj verjetno, da bi zašli v težave s ponavljanjem naključnih števil. Ime inspired od tega, da rabim Necromancy, da sem še pokonci.

Torej s tako napisanim integratorjem je reševanje vsake naloge prevedeno na to, da pravilno napišemo funkcijo, ki jo integrator sprejme. Jaz sem to imenoval *condition function*, ker je to funkcija, ki preverja, če je točka znotraj območja, ki ga integriramo in če je, potem nekaj z njo naredi. Integrator sem preveril prvo na primeru kroga v kvadratu, kjer sem z integracijo izračunal π . To sem naredil tako, da sem integriral funkcijo $f(x, y) = 1$ in primerjal razmerje točk znotraj kroga in vseh točk z razmerjem ploščin kroga in kvadrata.

$$\frac{4N_{\text{in}}}{N_{\text{tot}}} \rightarrow \pi. \quad (2)$$

Skoraj vsa integracija je potekala na mojem Ghetto clustru, kjer je bilo tokrat na voljo 152 jeder. To je bilo fino, ker sem lahko jemal večja števila točk in ker je lahko cluster dobil queue dela in je delal, ko sem jaz spal, čeprav as you might know, sem spal predolgo in posledično sem zgrešil rok za oddajo. Duša je želela, telo je reklo ne.

3.1 Zvezdasti lik

Za maso zvezdastega lika sem pravzaprav integriral funkcijo $f(x, y, z) = 1$ in jo pomnožil z gostoto, ki je bila postavljena na 1, saj je tako ali tako konstantna. Za vztrajnostni moment pa sem integriral funkcijo $f(x, y, z) = x^2 + y^2 + z^2$. No pravzaprav sem malce slabo prebral navodila in sem računal vrtilno količino, ampak ker je bila kotna hitrost enaka 1, je to isto kot vztrajnostni moment.

V drugem primeru, ko je gostota odvisna od razdalje od izhodišča, sem funkcijo $f(x, y, z)$ pomnožil z funkcijo $g(x, y, z) = (x^2 + y^2 + z^2)^p$, ki predstavlja radialni profil gostote.

3.2 Rojevanje žarkov gamma

Ker `NumberNecromancer` žreba točke uniformno na dani domeni (kar se sicer v splošnem lahko override-a, ampak je tako preprostejše) sem računal naključen radij in naključen kot takole:

$$r = \xi^{1/3}, \quad (3)$$

$$\theta = \arccos(2\xi - 1), \quad (4)$$

kjer je ξ naključno število iz enakomerne porazdelitve na intervalu $[0, 1)$. Tako sem dobil naključno točko na projekciji krogle na ravnino xy . V tem primeru so bile vse točke by definition znotraj krogle. Treba je bilo še izžrebat povprečno prosto pot takole:

$$\lambda = -\frac{1}{\mu} \ln(\xi), \quad (5)$$

Če je bila ta daljša od razdalje d , potem je žarek pobegnil, sicer pa je bil absorbiran. d sem izračunal kot slednje:

$$d = -r \cos \theta + \sqrt{1 - (r/R)^2 (1 - \cos^2 \theta)}, \quad (6)$$

kjer je R radij krogle, ki je tako ali tako bil postavljen na 1.

3.3 Nevtronski reflektor

Za nevtronski reflektor sem napisal funkcijo, ki je bila zelo podobna funkciji za rojevanje žarkov gamma. Pravzaprav sem si jaz problem zastavil tako, da so se nevtroni rojevali na sredini reflektorja in so se potem gibali. Torej tistih prvotno sipanih nevtronov tu ni zraven. Spomnim se, da je profesor to omenil na predstavitvi naloge in sem se odločil, da bom tako naredil. Glede na to, da sem tu gledal planarno gibanje, je razdalja d enaka:

$$d = d_0 + \lambda, \quad (7)$$

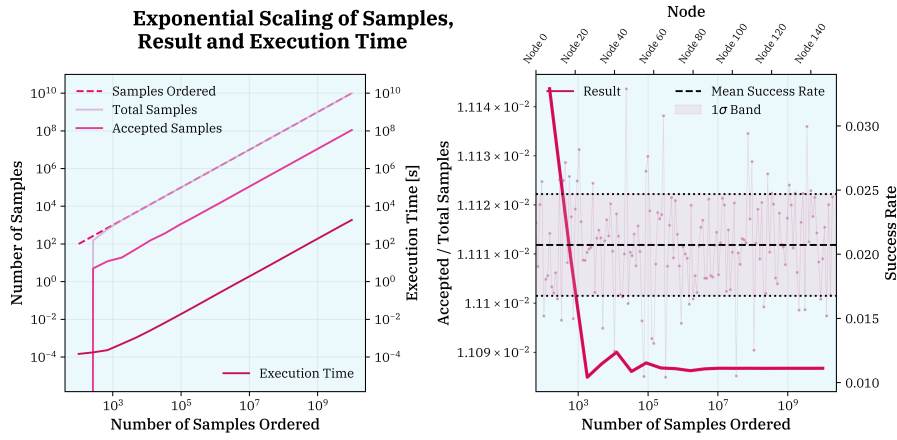
kjer je d_0 razdalja v središču reflektorja, λ pa povprečna prosta pot, ki je bila izžrebana tako kot prej. Zdaj ko to pišem, pomislim sicer na to, da mogoče v enem primeru, ko sem spreminjal debelino reflektorja, nisem upošteval tega, da se d_0 spreminja. To bi razložilo nekoliko strange rezultate, ki sem jih dobil.

4 Rezultati

Okay, kar na rezultate. Skušam se naučiti risanja kompaktificiranih grafov, tako da bi to preiskusil tukaj.

4.1 Zvezdasti lik

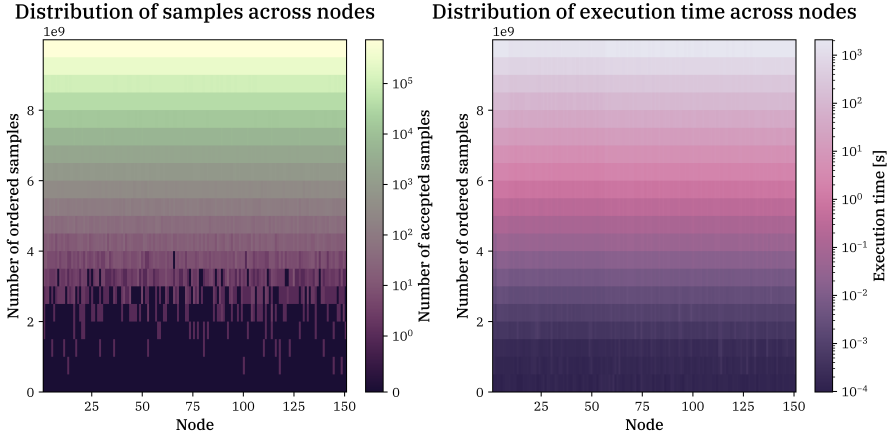
Lahko bi rekel, da sem več fokusa dal na skaliranje rezultatov, kot pa na dejansko fiziko za problemom. Problem sam po sebi se mi zdi precej dolgočasen in zdi se mi, da se več koristnega naučim, če poskusim narediti numeriko na čim boljši način. Slika (1) prikazuje rezultate za maso zvezdastega lika, ko je gostota konstantna. oz. kako se le ta spreminja z večanjem številom točk, ki jih generiramo. Tu je mogoče vredno omeniti, da je med številom zahtevanih točk in dejanskim številom točk, ki jih integrator generira, nekaj razlike. To je zaradi celoštevilskega deljenja, ki se zgodi, ko integrator razdeli točke na procesorska jedra.



Slika 1: Masa zvezdastega lika, ko je gostota konstantna.

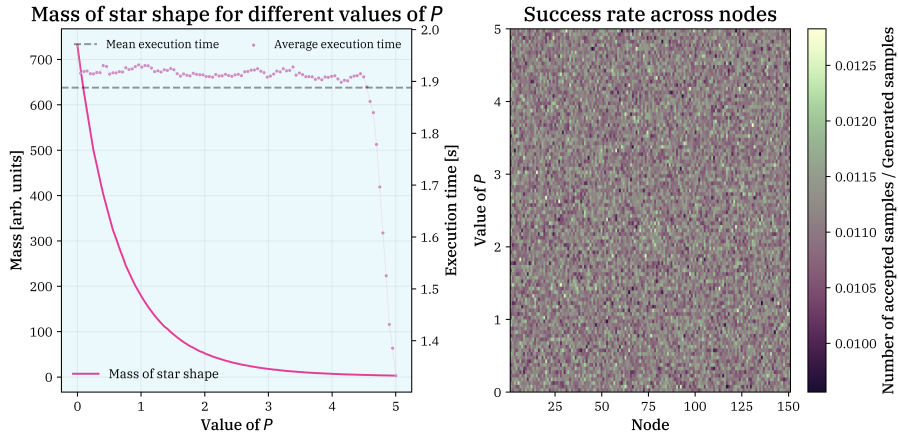
Na levi vidimo, kako se v log-log skali spreminja število vzorcev, kot se spreminja število vzorcev, ki jih sprejmemo. Pomembno je opaziti tudi naračanje časa izvajanja. Na desni vidimo, kako se vrednost mase spreminja, dokler ne doseže konvergenco, za dovolj veliko število vzorcev. V ozadju pa je prikazana res CompSci metrika in sicer success rate, oz. kolikšen delež točk je bil znotraj območja, ki ga integriramo, v povprečju čez vse data point-e za različne zahtevane števila vzorcev. S tem lahko narišemo potem srednjo vrednost povprečne vrednosti mase in njen ena-sigma interval.

Dodatno si lahko pogledamo še naključno naravo rezultatov, če si pogledamo raztros po posameznih procesorskih jedrih. To je prikazano na sliki (2). Na levi je prikazan rezultat, kjer je na x osi i -to jedro, na y osi pa je število zahtevanih vzorcev. Barva prikazuje maso, ki jo je vrnilo posamezno jedro. Pričakovali oz. želeli bi, da so barve znotraj vrstice enake, kar pomeni, da je masa neodvisna od tega, na katerem jedru se izvaja, ampak kot vidimo je nekaj naključnosti prisotne. Zanimivo mi je bilo potem še narisati čas izvajanja v istem formatu, kar je prikazano na desni. Barve se presenetljivo dobro ujemajo z manjšimi odstopanji. Lahko pa dobro oko opazi prvi namig sistemske napake, ki lahko kvari statistiko glede časa izvajanja na procesorsko jedro. Na levi strani desnega plota je viden malo temnejši pas, ki se pojavi na intervalu $i \in [8, 56]$. Ta efekt bo boljše viden na kasnejših slikah. Vmes pa lahko bralec ugiba, kaj bi lahko bil vzrok za to.



Slika 2: Masa zvezdastega lika, ko je gostota konstantna.

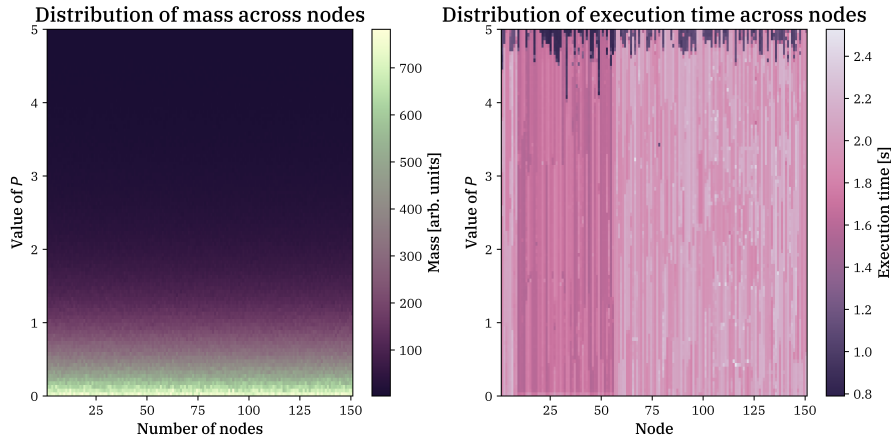
Sedaj pa si pogledjmo še, kako se spreminja masa, ko je gostota odvisna od razdalje od izhodišča. Rezultati so prikazani na sliki (3). Na levi je prikazana odvisnost mase od vrednosti p , ki je potenca v radialnem profilu gostote. V ozadju je tokrat prikazan čas izvajanja v povprečju čez vsa jedra. Število vzorcev je bilo tu postavljeno na 10^7 . Na desni pa je prikazan success rate za različne vrednosti p in njegov raztros po jedrih. Tukaj bi pričakovali, da dobimo popoln šum in res sure enough dobimo sliko, ki zglada precej naključna.



Slika 3: Masa zvezdastega lika, ko je gostota odvisna od razdalje od izhodišča.

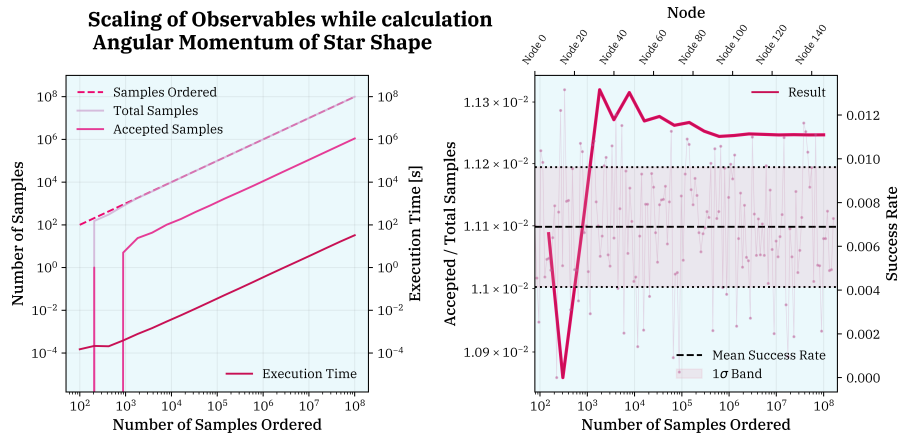
Tako kot prej lahko pogledamo tudi tu raztros po jedrih, kar je prikazano na sliki (4). Na levi je prikazan rezultat za maso, na desni pa za čas izvajanja. Na levi je zanimivo opaziti večje odstopanje za večje vrednosti mase. To mi da misliti, da je mogoče relativna napaka pravzaprav konstantna in je tu napaka navidezno večja samo zaradi skale rezultata. Na desni strani pa človek prvo zagleda tisti

efekt, ki sem ga omenil prej. Tu se izrazito vidi pas, ki sem ga prej omenjal. To je posledica tega, da cluster ni idealiziran in sestavljen iz identičnih delov, ampak sta bili pri računanju uporabljeni dve mašini, ki imata boljše procesorje. Teh jeder je skupaj 48. Čeprav sta na začetku clustra pride initial shift 8 jeder od kontrolnega računalnika, ki je tu tudi računal. Pravzaprav, če bi pogledali side-by-side Prvih 8 jeder in potem preostanek od 56 dalje, bi videli, da je prvih 8 svetlejših, kar ge tu za tretjo vrsto procesorja, ki je najbolj.. recimo temu budget oriented.



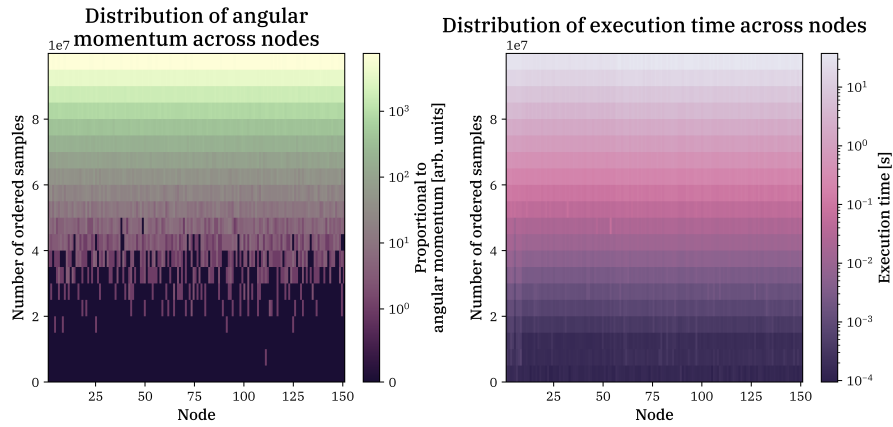
Slika 4: Masa zvezdastega lika, ko je gostota odvisna od razdalje od izhodišča.

Podobno se vsi ti efekti vidijo tudi pri izračunih za vztrajnostni moment. Ti so za konstantno vrednost gostote prikazani na sliki (5). Graf je zelo podoben tistemu za maso. Tudi vse količine na njemu so pravzaprav enake, z razliko tega, da je funkcija, ki jo integriramo drugačna.



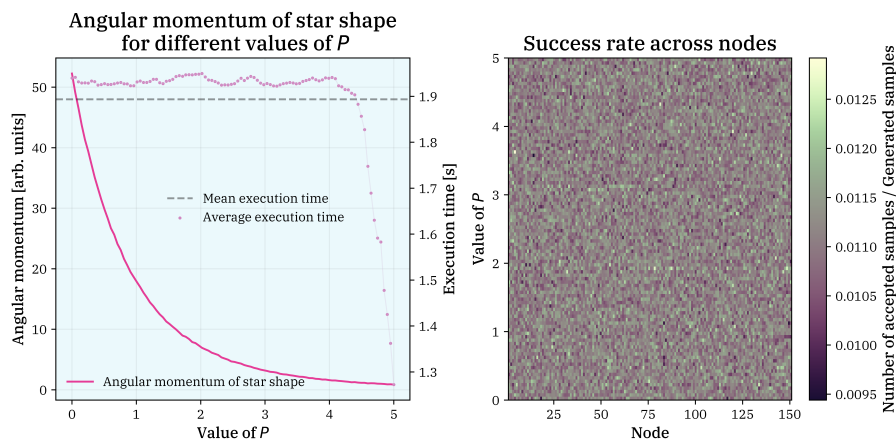
Slika 5: Vztrajnostni moment zvezdastega lika, ko je gostota konstantna.

Tudi tu sem iz radovednosti narisal še raztros po jedrih, kar je prikazano na sliki (6). Na levi je prikazan rezultat za vztrajnostni moment, na desni pa za čas izvajanja. Da se razlika med jedri boljše vidi, sem tu namesto vrtilne količine risal vrednost, ki je proporcionalna vrtilni količini; pravzaprav samo reskalacija, da so absolutne napake večje. Z risanjem relativne napake sem imel težave z postavitvijo normalizacije barve tako, da bi bile razlike lepo vidne po celem polju. Nekako na občutek pa se mi zdi, da je večja razlika v času izvajanja, kot pa v pri masi, kljub temu, da ne bi smelo biti nobenega vpliva. Not sure.



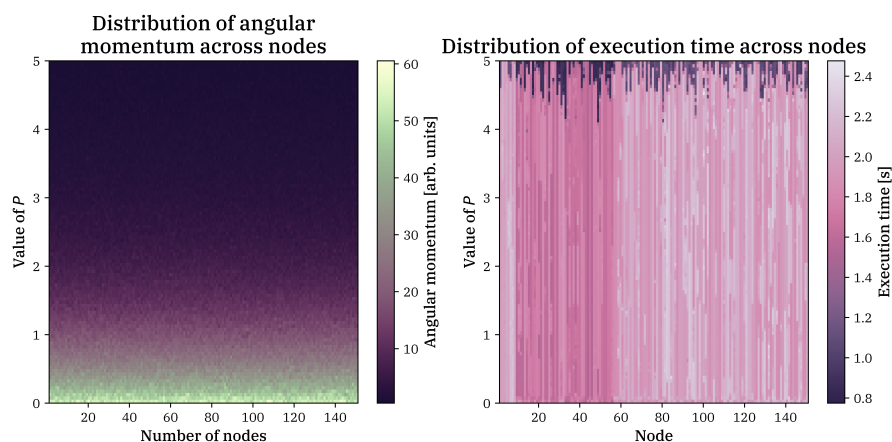
Slika 6: Vztrajnostni moment zvezdastega lika, ko je gostota konstantna.

Za primer, ko je gostota odvisna od razdalje od izhodišča, pa so rezultati prikazani na sliki (7). Vidimo, da se zgodi ista stvar kot pri masi, ko je gostota odvisna od razdalje od izhodišča. Zanimivo je tudi, da se isto pojavi ta drop-off v času računanja pri večjih vrednostih p . Not exactly sure kaj bi lahko bil vzrok za to. Spet vidimo, da je success rate res naključen, kar je prikazano na desni strani.



Slika 7: Vztrajnostni moment zvezdastega lika, ko je gostota odvisna od razdalje od izhodišča.

Tudi tu sem iz radovednosti narisal še raztros po jedrih, kar je prikazano na sliki (8). Vidno kar je bilo pričakovano in vidno že prej, a vseeno prijetno dobiti rezultat, ki je v skladu z našimi pričakovanji. Na levi je prikazan rezultat za vztrajnostni moment, na desni pa za čas izvajanja.

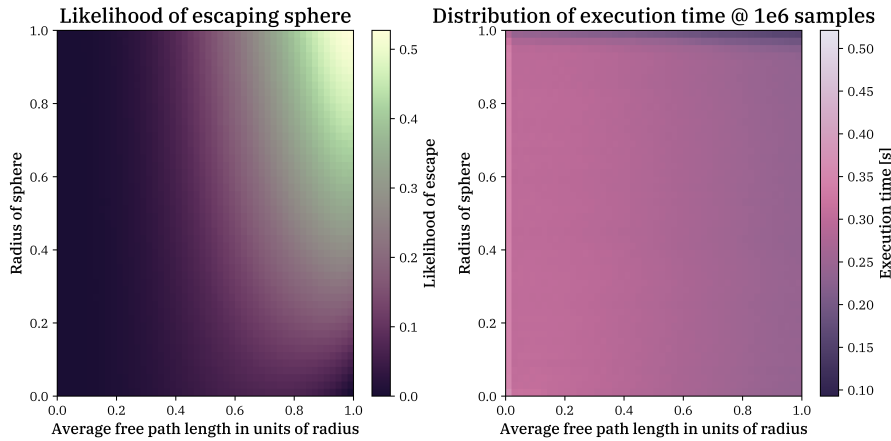


Slika 8: Vztrajnostni moment zvezdastega lika, ko je gostota odvisna od razdalje od izhodišča.

4.2 Rojevanje žarkov gamma

V duhu kompaktnih plotov in sploh ne zaradi tega, ker čas tako hitro teče, sem narisal en plot, ki hopefully uspešno pokaže obe zanimivi odvisnosti pri tej nalogi. To je prikazano na sliki (9). Na levi je prikazana verjetnost za pobeg žarka iz krogle, na desni pa čas izvajanja. Na obeh grafih je prikazana odvisnost od povprečne proste poti, ki je (**PAZI**) v enotah radija krogle. Lepo je vidno kako

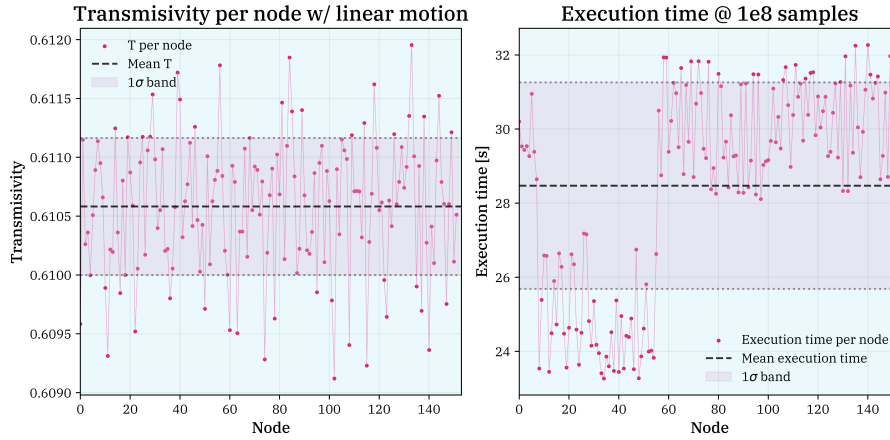
gre rezultat proti pričakovanim 52% za vrednosti porvpene proste poti, ki so večje blizu 1. Ne razumem pa čisto popolnoma, zakaj ni ta vrednost konstantna za vse radije. Lahko da je tu kje error on my part, ampak ne vem. Na desni strani pa vidimo, da čas izvajanja pade, ko imamo večjo povprečno prosto pot. Očitno je računski del za absorpcijo žarkov bolj zahteven, kot pa za pobeg.



Slika 9: Verjetnost za pobeg žarka iz krogle in čas izvajanja.

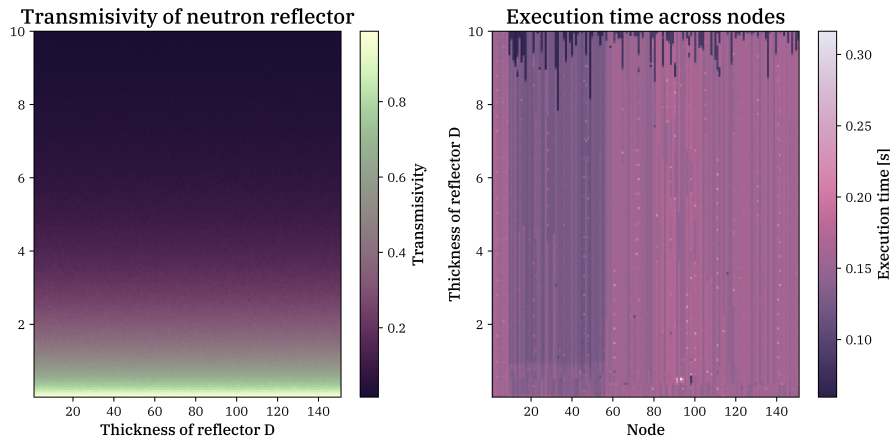
4.3 Nevtronski reflektor

Za konec pa še ta nevtronski reflektor. Meni je bilo pri danih parametrih iz naloge najbolj zanimivo pogledati tranzmisivnost. To sem potem dodatno še malo razširil in pogledal raztros po procesorjih iz katerega sem učinkovito dobil oceno za napako. To je prikazano na sliki (10). Na levi je prikazana povprečna vrednost za prepustnost nevtronov skozi reflektor raztresena po jedrih in njena srednja vrednost z ena-sigma intervalom. Na desni pa je, kot je že trend, prikazan povprečen čas izvajanja. Tu se mogoče malo boljše vidi efekt neenakosti procesorjev, kjer je res očiten padec v času izvajanja na jedrih, ki so boljša.



Slika 10: Prepustnost nevtronov skozi reflektor in čas izvajanja.

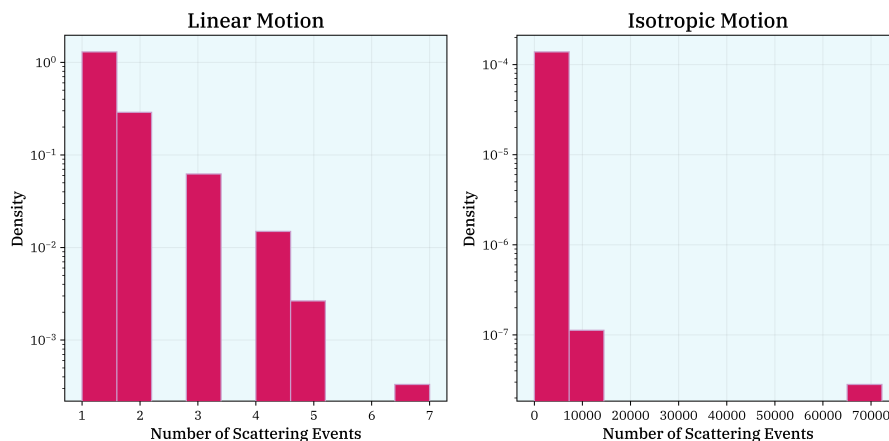
Podoben graf lahko narišemo tudi za izotropno (v planarnem) gibanje, kar je prikazano na sliki (11). Tu je razlika, da sem računal za različne debeline reflektorja. Na levi je prikazana prepustnost, na desni pa čas izvajanja. Zdaj ob pisanju tega reporta mi je padlo na pamet, da mogoče nisem upošteval, da se d_0 spreminja z debelino reflektorja. Sem prav s tem namenom znova prižgal cluster in pognal izračune za popravljeno kodo in sem dobil bolj smiselne rezultate. Te so te, ki so prikazani na sliki (11). Tu je očitno, da je prepustnost večja, če je debelina reflektorja manjša.



Slika 11: Prepustnost nevtronov skozi reflektor in čas izvajanja.

In na zadnje porazdelitev po številu sipanj. V prejšnjih primerih sem vedno gledal samo za eno sipanje, kar se po moji porazdelitvi tako ali tako izkaže za najbolj verjetno. Desen plot bi lahko verjetno bil mnogo boljši še bi outlier pri $N = 700000$ odstranil, ampak se mi je zdelo tako imenitno, da se mi je nekako posrečilo ujeti nesrečni nevtron, ki se je sipal skoraj milijon krat. Sicer pa ja,

ko dopuščam polno planarno gibanje število sipanj očitno zelo naraste, kar mi je zanimivo, ali pa samo tako zgleda zaradi nekaj outlier-jev.



Slika 12: Porazdelitev po številu sipanj.

5 Komentarji in izboljšave

Se mi zdi, da se opazi razlika med prvo nalogo in potem drugo in tretjo. Ugotovil sem, da mi je vedno manj všeč, kako je zastavljen predmet, ker se mi zdi, da je preveč fokusa na sheer volume of work, ne pa na to, da bi lahko nek problem res globinsko raziskal. Veliko bolj bi preferiral strožje ocenjevanje in samo prvo nalogo, kjer bi si lahko potem res pogledal še detaile in se iz tega naučil nekaj novega, namesto da rešujem isti problem na surface level-u. in na koncu se naučim bolj nič in so slike half-assed. To bi mi tudi omogočilo da se **končno** spustim globlje v fiziko samega problema. Drugače imam preferenco nad drugimi stvarmi in potem raje delam tisto, kar mi je zanimivo. Pravzaprav sem razmišljal kaj je boljše, delati nalogo tako kot jaz mislim, da bom maksimiziral novo znanje in zadovoljstvo, ali tako, da dobim čim boljšo oceno. Trenutno, priznam, jih delam bolj v slogu prve možnosti.

Nevem če je to sprememba, ki bi se reasonably kadarkoli uvedla v predmet, ampak one can dream. No ja, jaz bi si že tako na sploh želel, da bi bilo več fokusa tudi na kvaliteto numerike, ne samo na to, kdo naredi najbolj pisane plote. Še sploh ko med fiziki menda pogosto, da je koda popolnoma neberljiv obup. Hell jaz bi dal 20% ocene za to, da je koda v zglednem formatu. Ocena based on kako dolgo rabim, da ugotovim kaj se dogaja etc.