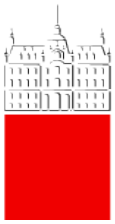
Univerza *v Ljubljani*
Fakulteta za *matematiko in fiziko*



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

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 deterministično. 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 preprostemu primeru, ko nekaj odrežemo, preverimo ali je točka znotraj trikotne prizme). Če je točka znotraj, potem inkrimentiramo 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 poglejmo zvezdasti lik, ki ga omejuje ploskev:

$$\sqrt{|x|} + \sqrt{|y|} + \sqrt{|z|} = 1 \tag{1}$$

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 poglejmo nevtronski reflektor, ki je pravzaprav plošča z neko debelino. Na reflektor pošiljamo tok nevtronov, ki se v njej siplejo 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.

$$\frac{4N_{\text{in}}}{N_{\text{tot}}} \to \pi \,. \tag{2}$$

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:

$$r = \xi^{1/3}\,, \tag{3}$$
$$\theta = \arccos(2\xi - 1)\,, \tag{4}$$

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 izžrebati povprečno prosto pot takole:

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

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

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

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:

$$d = d_0 + \lambda\,, \tag{7}$$

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

5 Komentarji in izboljšave

Literatura