Modelska analiza I
1999/2000
1. naloga - generatorji slučajnih števil

Andrej Studen, MA-FI

14. oktober 1999

1 Običajni testi generatorjev naključnih števil

Načeloma bi radi le, da nam generator daje povsem enakomerno porazdeljene vrednosti v določenem intervalu. Moti pa nas, ker ne moremo preizkušati v neskončnost, kjer bomo zares dobili konstanto po celem intervalu za idelani generator, pač pa se moramo že prej ustaviti. Takrat pa vsaka vrednost še nosi v sebi nekaj napake. K sreči lahko to napako ocenimo in primerjamo porazdelitev s pravo. Pri tem nam zelo pomaga χ^2 test. To število je namreč podano z enačbo:

$$\chi^2 = \frac{1}{M}\sum_{i=1}^{M}\frac{(N_i - C)^2}{\sigma_i^2}$$

za konstantno porazdelitev (C). M je število kanalov, v katerih zbiramo vrednosti, σ pa določimo iz Poissonove porazdelitve za naključne dogodke - $\sigma = \sqrt{N_i}$. Ugotovili so, da je za porazdelitve, ki se dobro ujemajo z napovedjo, χ blizu 1. Tako lahko opazujemo tudi porazdelitev vrednosti χ^2 za različne poskuse.

Pri tem nismo nič govorili o tem, kako smo prišli do vrednosti v posameznem kanalčku, niti o tem, kakšna je prava vrednost v vsakem kanalu. Da lahko izračunamo χ, rabimo vedeti le vrednost C-ja v vsakem kanalčku. Torej lahko enako računamo χ za enodimenzionalne in dvodimenzionalne teste. Le pri 2-D je treba vzeti $M = m^2$, kjer je m število delilnih točk na eni stranici, rezultat pa je enak.
Slike kažejo porazdelitve za 1-D in 2-D test (to pomeni, da gledam korelacijo dveh zaporednih naključnih števil in sliko para na ravnini) za generator `ran2.c`. Da pa ne bi govorili samo o lepih primerih, sem delal slike še s kaotičnim generatorjem (`quad.c`), ki deluje po formuli:

$$x_{n+1} = -4(1 - x_n)x_n$$

Lepo je videti 2-D korelacijo in tudi veliko odstopanje pri χ^2 testu.

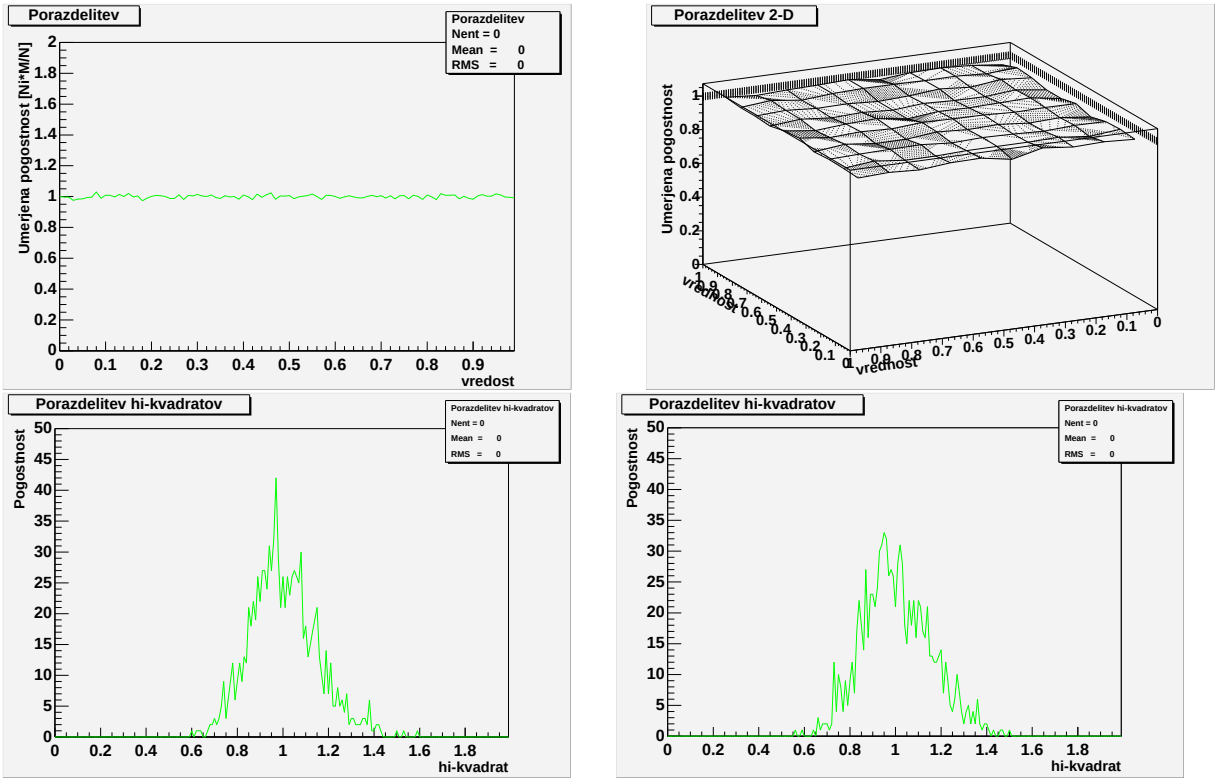
2 *Run-up* test

To je preprost postopek štetja naraščajočih ali padajočih nizov. Ker nam da teorija rezultate za povsem naključno zaporedje, jih lahko primerjamo z vrednostmi naših generatorjev.
Sevda bi morali za točne vrednosti počakati do neskončnosti in spet bi lahko s χ^2 testom preverili odstopanje od pravih rezultatov. A naj bo primerjava vrednosti za en poskus dovolj. Že tu se pokaže, kako slabo deluje kvadratični generator (`quad.c`).

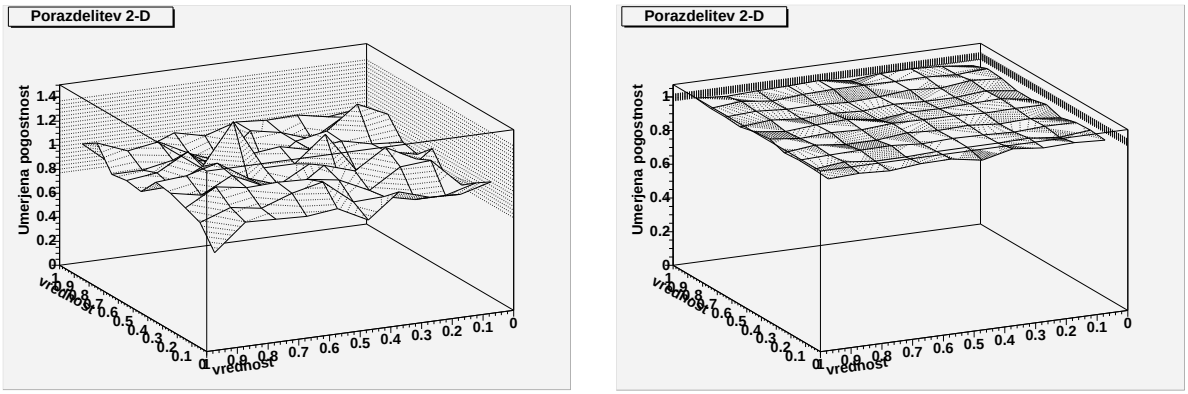
3 Kalkulatorski generator naključnih števil

še en generator deluje z realnimi števili. Formula zanj je:

$$x_{n+1} = \left\{(x_n + \pi)^5\right\},$$



Slika 1: 1-D in 2-D testi za zgleden generator `ran2.c`. To so torej slike, ki jih pričakujemo.

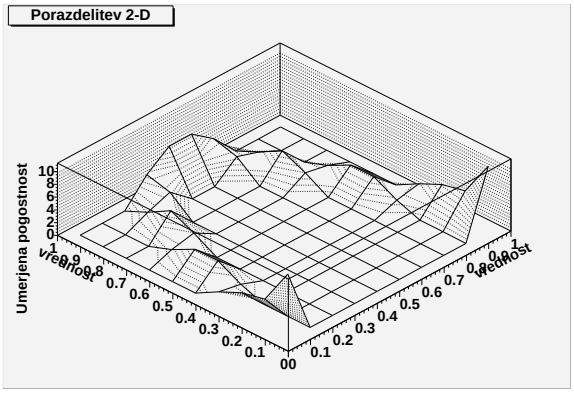


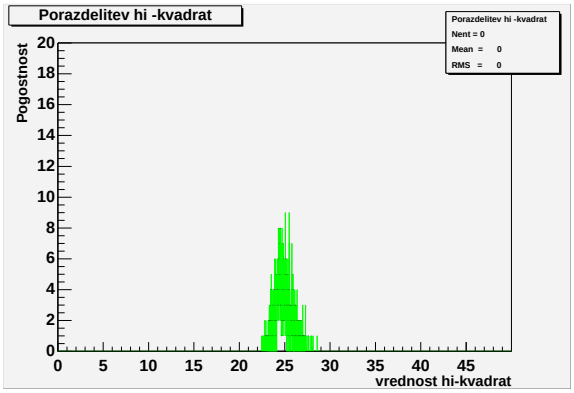
Slika 2: Umirjanje porazdelitve za naraščajoče število izžrebanih števil. Levo za $N = 10000$, desno pri $N = 1000000$. Vse za lep generator `ran2.c`.

kjer pomeni zaviti oklepaj neceli del števila (enak rezulatatu funkcije `floor`, torej številki med nič in ena. Čeprav je v navodilih predstavljen kot generator iz starih časov, presenetljivo dobro deluje. Celo tako dobro, da ga težko ločimo od kombiniranih kongruenčnih generatorjev naključnih števil. Opravljene teste kažejo slike.

4 Test s polnjenjem celic

To je še eden izmed množice testov, ki jih lahko opravimo na generatorju naključnih števil. Preprosto gledamo, kako hitro se zmanjšuje število praznih(nezadetih) celic. To lahko počnemo v 1-D ali kateremkoli





Slika 3: Tako pa je, če generator ni dober. Povsem očitna je parabolična korelacija na ravninskem testu, pa tudi 1-D testa naše zaporedje ne zdrži. Gre torej za povsem zgrešeno idejo gneratorja naključnih števil, a meni bo dobro služil kot primer, česa od dobrega generatorja ne pričakujem.

Tabela 1: Primerjava teoretičnih in eksperimantalnih dolžin nizov pri run-up testu. Skoraj vsi generatorji (vključno s starim kalkulatorskim) se izvrstno izkažejo (napaka v okviru σ)

dolžine nizov	točno	`ran0.c`	`ran1.c`	`ran2.c`	`ran3.c`	`calc.c`	`quad.c`
n(1)	1667	1644	1648	1655	1682	1675	3382
n(2)	2083	2131	2064	2125	2063	2142	3309
n(3)	916	907	920	907	902	884	0
n(4)	263	254	284	249	270	262	0
n(5)	58	54	48	62	63	53	0
n($\geq$6)	12	4	14	11	15	12	0

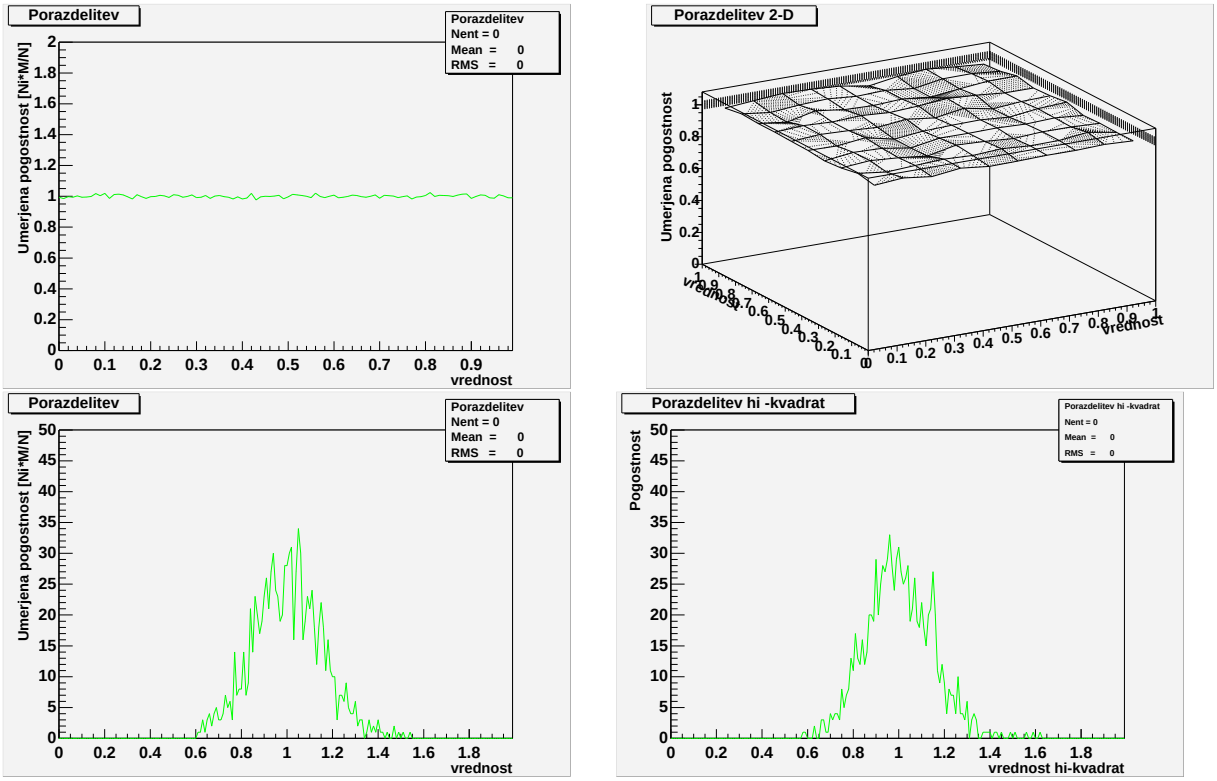
drugače dimenzionalnem prostoru. Naredil sem 3-D primer, tako da imamo teste v treh najuporabnejših dimenzijah. Ko bomo opravili M žrebanj, kjer je M število celic, bomo v povprečju neskončno poskusov z M žrebanji zadeli vsako celico natanko enkrat. Pri naključnih dogodkih (potresih), katerih verjetnostno porazdelitev opi"emo z eksponentno krivuljo, je relaksacijski čas ravno enak povprečnemu času, v katerem te doleti en dogodek. Torej bo za nas ta čas ravno število celic in bomo lahko primerjali krivuljo $\exp^{-t}$ z deležem praznih celic po M*t poskusih.

Spet se `calc.c` izkaže tako dobro kot ostali generatorji, primerjava pa je narejena za `ran2.c` in očitno slab `quad.c`.

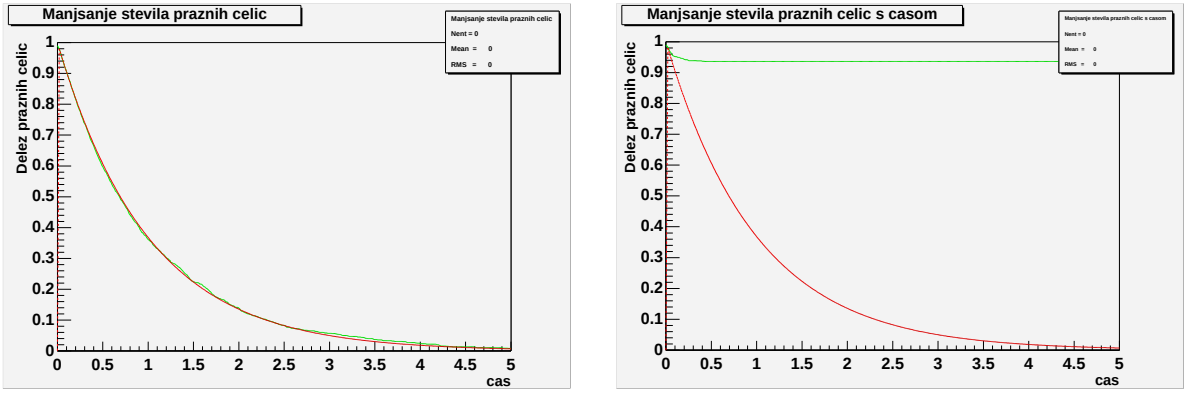
5 *Random walk*

Pri naključnem sprehodu po mreži je zanimivo, da se oddaljujemo od naše izhodiščne točke. To se zdi na prvi pogled protislovno. A že kratek premislek nas prepriča o tem, da se v povprečju oddaljujemo. Hec je v tem, da se v vsakem koraku nekam premaknemo. Vsak od naših štirih sosedov je enako verjeten. Ko pa pridemo v sosednjo točko, je pot nazaj le malo verjetna. Recimo, da smo začeli gor(S). Če opazujemo samo razdaljo, nas bosta dve točki (levo ali desno) pripeljali v dveh korakih na razdaljo $\sqrt{2}$ koraka, ena (spet gor) celo na razdaljo 2, ena sama (nazaj dol) pa nazaj v izhodišče. Verjetnost za povratek je le četrtinka, porazdelitev po razdaljah pa taka, kot smo jo pričakovali, z maksimumom pri razdalji $\sqrt{2}$ za N=2.

Opazoval sem sprehode z 10000 koraki in njihove cilje. Za večino generatorjev je porazdelitev taka kot na sliki(normiral sem na število korakov - tako je povprečna prepotovana razdalja $\sqrt{\frac{1}{10000}} = 0.01$. Spet pa je očitno napačen `quad` generator, ki nas odvleče predalec, pa tudi porazdelitev ciljev zaradi kvadratne korelacije ni več rotacijsko simetrična okrog izhodišča.



Slika 4: 1-D in 2-D testi za kalkulatorski generator `calc.c`. Primerjave s slikami priporočenega generatorja odkrijejo veliko podobnost obeh.



Slika 5: Kako je lahko generator napačen, nam spet kaže primerjava s pravim generatorjem. Levo se točna eksponentna krivulja in delež praznih celic praktično ujemata, medtem ko nam kvadratni generator nekaterih točk sploh ne zadene.

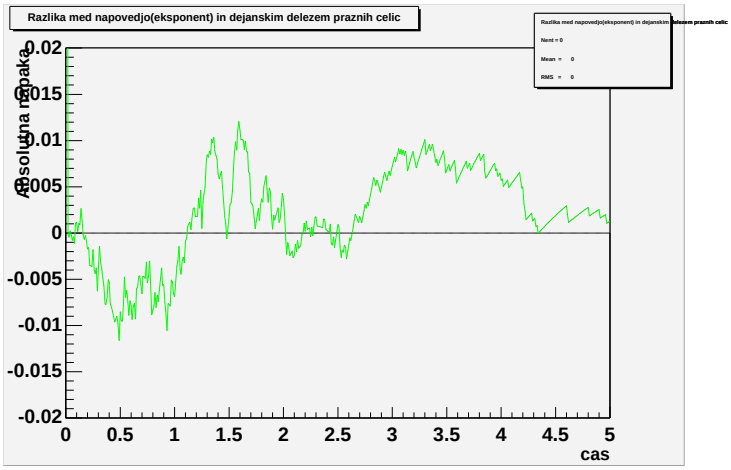
6 Permutacijski test

Da permutacijski indeks, predstavljen na listu, res deluje, je malo težko ugotoviti. Poskusimo pa lahko. Vsak od C_i-jev bo zagotovo manjši od i-1. Torej je C_0 enako 0 in je permutacijski indeks manjši od $n!$, če je n število elementov v podnizu. Nato vidimo, da v vsoti, ki nam da indeks premutacije naslednji člen C_i vedno množimo s členom, ki je večji od največjega, ki ga še lahko doseže predhodnik. Tako vsak zase kontrolira svoje območje.

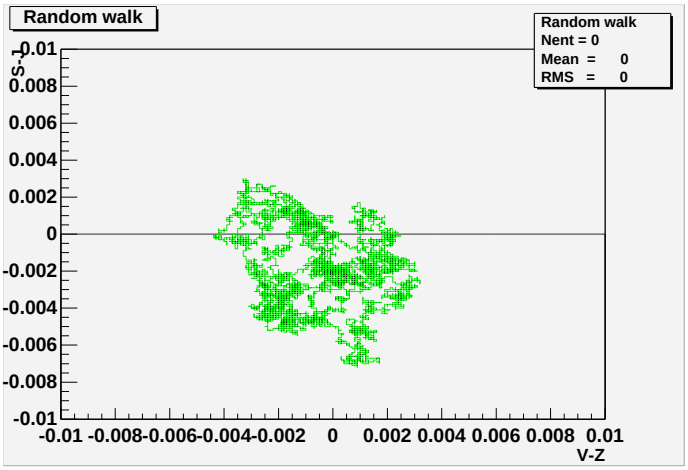
Testi ne prinesejo nič novega. Spet imamo dobro primerjavo v `quad` in lahko primerjamo dobre in slabe

Tabela 2: Indeksi permutacij za niza dolžine 4

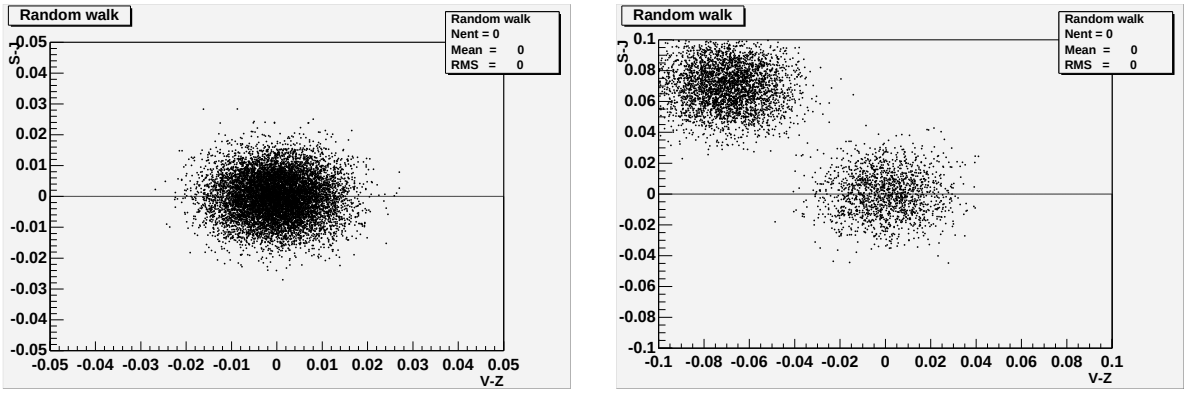
permutacija	indeks
1 2 3 4	23
1 2 4 3	22
1 3 2 4	19
1 3 4 2	18
1 4 2 3	17
1 4 3 2	21
2 1 3 4	11
2 1 4 3	10
2 3 1 4	7
2 3 4 1	6
2 4 1 3	5
2 4 3 1	9
3 1 2 4	3
3 1 4 2	2
3 2 1 4	15
3 2 4 1	14
3 4 1 2	13
3 4 2 1	1
4 1 2 3	0
4 1 3 2	8
4 2 1 3	12
4 2 3 1	20
4 3 1 2	4
4 3 2 1	16



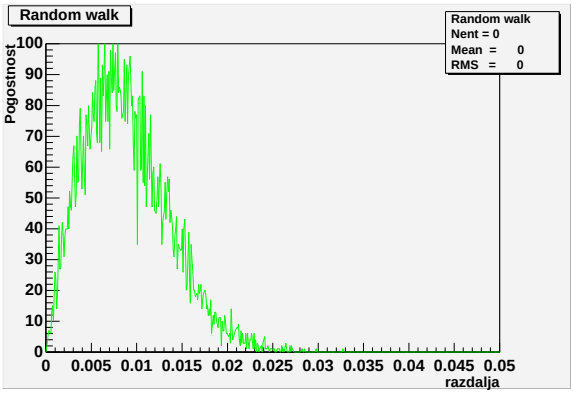
Slika 6: Še razlika med teoretično napovedjo in dejanskim deležem praznih celic. Že po enem poskusu se obe vrednosti ne razlikujeta več kot za stotinko.

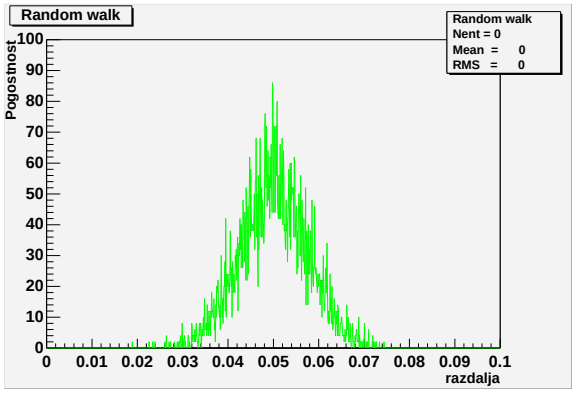


Slika 7: Sprehod v enem poskusu. Naš možic je kar priden in pokrije velik prostor. Zanimiv vzorec.



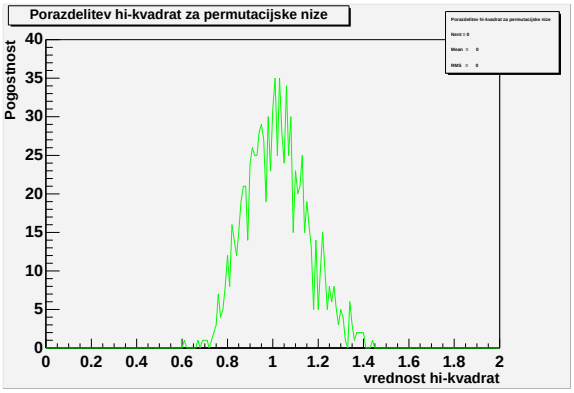
Slika 8: Spet primerjava dobrega in slabega generatorja. Cilji so pri `ran2.c` (levo) lepo porazdeljeni, medtem ko na levi vidomo oblak na levi, kamor zaide veliko točk. Pač niso vse smeri enako verjetne.

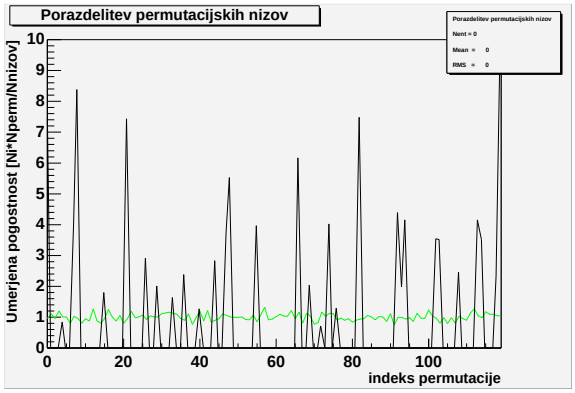




Slika 9: Še primerjava zadetih razdalj. Res se pojavi kupček z maksimumom nekje okrog 0.01, kar je $(10000)^{-\frac{1}{2}}$, medtem ko pri slabem generatorju `quad` dobimo sranje (desno).

strani generatorjev. V igri je spet χ^2 test. Vse po pričakovanjih.





Slika 10: Leva slika kaže porazdelitev χ^2 za dober generator, desno pa vidimo porazdelitev po dolžinah nizov za nize dolžine 5. Napaka `quad` je res očitna, medtem ko je porazdeitev za `ran2.c` dokaj enakomerna.