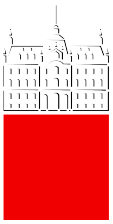
Univerza *v Ljubljani*
Fakulteta za *matematiko in fiziko*



Naloga 2: Naključni sprehodi

Matematično-fizikalni praktikum

Dimitrije Pešić

Vpisna številka: 28201072

Predavatelj: prof. dr. Borut Paul Kerševan

Ljubljana, 10.10.2023

1 Uvod

Naloga je od nas zahtevala, da naredimo računalniško simulacijo dvorazsežne naključne hoje za Lvyeve polete in sprehode. Narisati nekaj značilnih slik sprehodov za 10, 100, 1000 in 10000 korakov. Nato pa še poskusiti iz velikega števila sprehodov z velikim številom korakov določiti eksponent γ za nekaj izbranih parametrov μ v posameznih primerih, ter določiti, za kakšno vrsto difuzije gre.

2 Ozadje

Naključni sprehodi so vrsta gibanja, pri katerem v velikem številu korakov napredujemo iz izhodišča v neko končno lego, tako da se parametri vsakega naslednjega koraka sproti naključno določajo. Običajni zgled je Brownovo gibanje (difuzija) drobnih delcev barvila po mirujoči homogeni tekočini, kjer je spočetka barvilo zbrano v izhodišču. "Težišče" barvila $\langle x(t)\rangle$ v povprečju ostane v izhodišču, razen če v tekočini vzpostavimo kako anizotropijo (na primer v dveh razsežnostih z vsiljeno rotacijo). "Razmazanost" po dolgem času je sorazmerna s časom,

$$\sigma^2(t) \equiv \langle x^2(t)\rangle - \langle x(t)\rangle^2 = 2Dt\,.$$

Sorazmernostni koeficient je običajna difuzijska konstanta, priča smo normalni difuziji

Zanimiveje je opazovati naključne sprehode, pri katerih dovolimo nadpovprečno dolge korake. Verjetnostno gostoto porazdelitve po dolžinah posameznih korakov parametrizirajmo v potenčni obliki

$$p(l) \propto l^{-\mu}\,, \tag{1}$$

kjer naj bo $1 < \mu < 3$. Tedaj postane drugi moment porazdelitve

$$\langle l^2\rangle = \int l^2 p(l)\mathrm{d}l$$

neskončen. Govorimo o anomalni difuziji, prisotni pri celi družini kinematičnih distribucij dolžin poti z "debelimi repi".

Ustrezno sliko naključnega gibanja, povezanega s temi dolgimi koraki, lahko interpretiramo na dva načina:

- Lévyjev pobeg oz. polet (*flight*), implicira, da vsak korak iz porazdelitve (1) traja enako dolgo, medtem ko se hitrost gibanja med koraki (divje) spreminja.
- Lévyjev sprehod (*walk*), ki interpretira korak iz porazdelitve (1) kot gibanje s konstantno hitrostjo in tako koraki trajajo različno dolgo časa (dolžina koraka je sorazmerna s časom).

Pri anomalni difuziji razmazanost (varianca) velike množice končnih leg naključnih Lévyjevih **sprehodov (walks)** narašča z drugačno potenco časa. Velja $\sigma^2(t) \sim t^\gamma$, kjer je

$$\begin{aligned}
1 < \mu < 2\,, &\qquad \gamma = 2 && \text{(balistični režim)}\,,\\
2 < \mu < 3\,, &\qquad \gamma = 4 - \mu && \text{(super-difuzivni režim)}\,,\\
\mu > 3\,, &\qquad \gamma = 1 && \text{(normalna difuzija)}\,.
\end{aligned}$$

Za $\mu = 2$ pričakujemo $\sigma^2(t) \sim t^2/\ln t$, za $\mu = 3$ pa $\sigma^2(t) \sim t\ln t$.

Slika je nekoliko drugačna pri opazovanju naključnih Lévyjevih **poletov (flights)**. Spet vzamemo zvezo $\sigma^2(t) \sim t^\gamma$ in dobimo odvisnosti

$$\begin{aligned}
1 < \mu < 3\,, &\qquad \gamma = \frac{2}{\mu - 1} && \text{(super-difuzivni režim)}\,,\\
\mu > 3\,, &\qquad \gamma = 1 && \text{(normalna difuzija)}\,.
\end{aligned}$$

Pri $\mu = 2$ očitno pričakujemo $\sigma^2(t) \sim t^2$, torej balistični režim.

Simulacijo sem vedno začel v izhodišču ($x = y = 0$) in določili naslednjo lego tako, da naključno izberem smer koraka in statistično neodvisno od te izbire še njegovo dolžino, torej

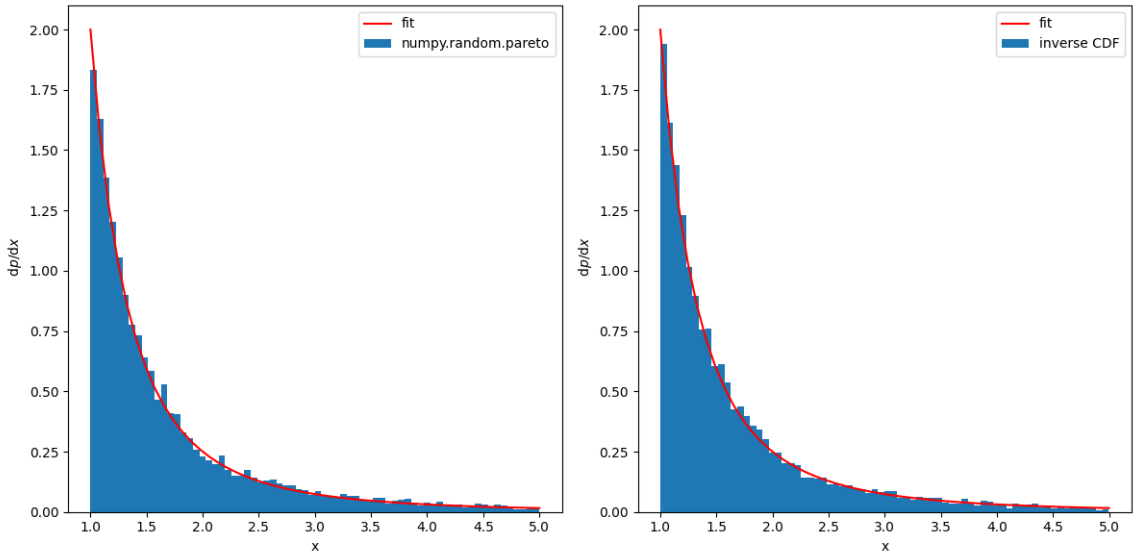
$$\begin{aligned} x &\leftarrow x + l\cos\varphi\,, \\ y &\leftarrow y + l\sin\varphi\,, \end{aligned}$$

kjer je φ enakomerno naključno porazdeljen po intervalu $[0, 2\pi]$, dolžina koraka l pa naj bo porazdeljena v skladu s potenčno obliko.

3 Rezultati

Program sem napisal v PYTHON jeziku in uporabil pakete *Numpy*, *Matplotlib* ter *Scipy*. Za psevdo naključna števila pri simulaciji sem uporabil funkciji *numpy.random.random*, ki nam vrne naključno število med $[0, 1]$ ter funkcija *numpy.random.pareto*, ki nam kot samo ime pove, vrne naključno število po Pareto distribuciji. Za obe funkciji sem nastavil začetno vrednost random seed z uporabo funkcije *datetime.datetime.now* iz knjižice *datetime*. Tako sem lahko bil siguren, da imam zares naključno porazdelitev ob vsakem ponovnem zagonu.

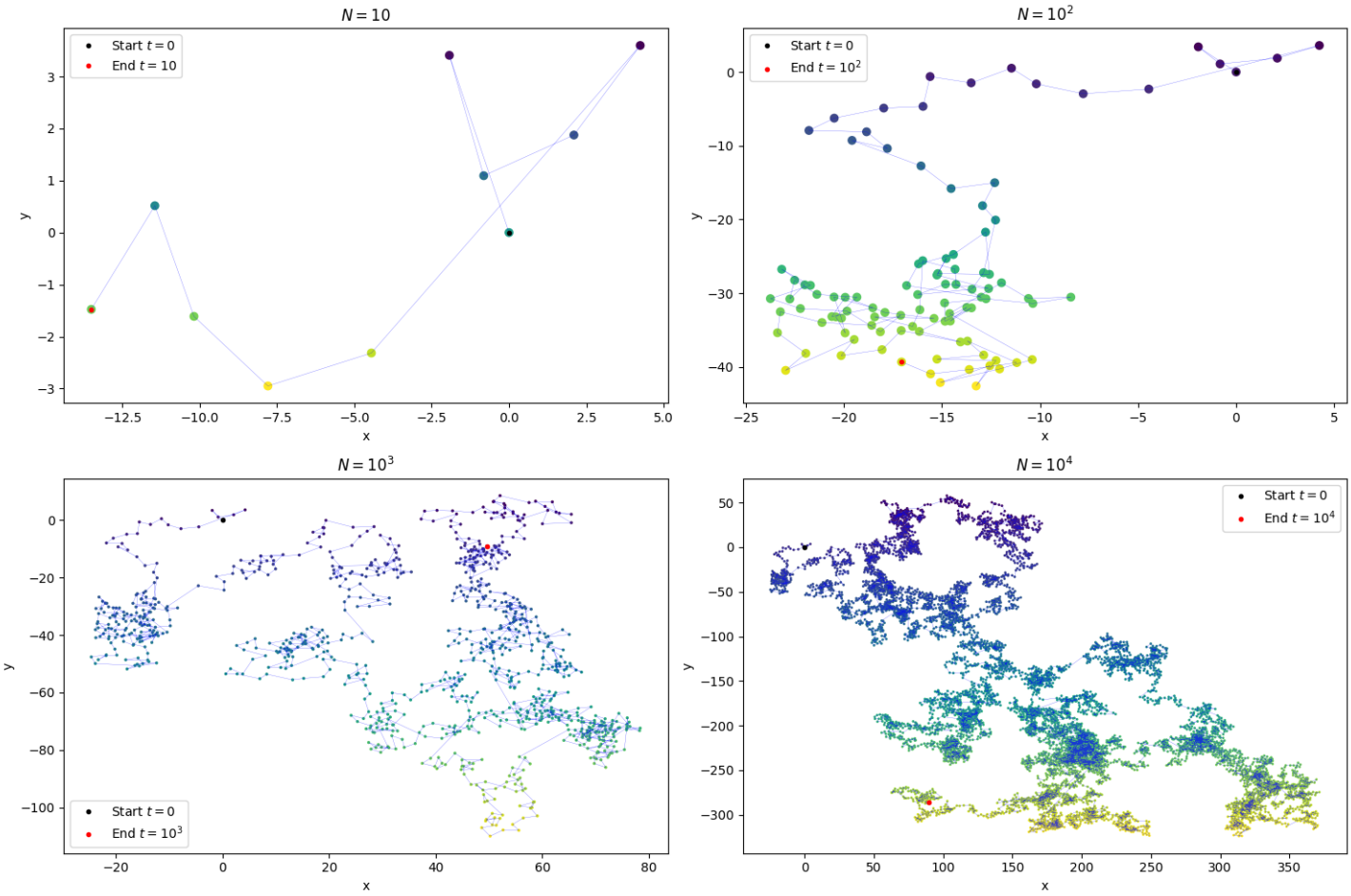
Preveril sem ali je bolje uporabiti že implementirano porazdelitev ali pa morda bolje da svojo implementacijo napiš. Rezultat tega je prikazan na grafu 1. Primerjava pokaže ekvivalentnost, odločil sem se za uporabo *numpy* funkcije.



Slika 1: Primerjava implementacije generatorjev porazdelitve.

Ker je program veliko vrednosti računal sem se odločil za implementacijo angl. "multithreading- ali večnitnosti v programu z uporabo izvorne pythonove knjižnice *threading*. Tega problema bi se lahko tudi lotil tako da bi implementiral povezavo z SQL podatkovno bazo, ali z uporabo *ROOT* knjižnice ali pa morda kar shranjeval v datoteko, vendar se mi je ta način najbolj dopadel.

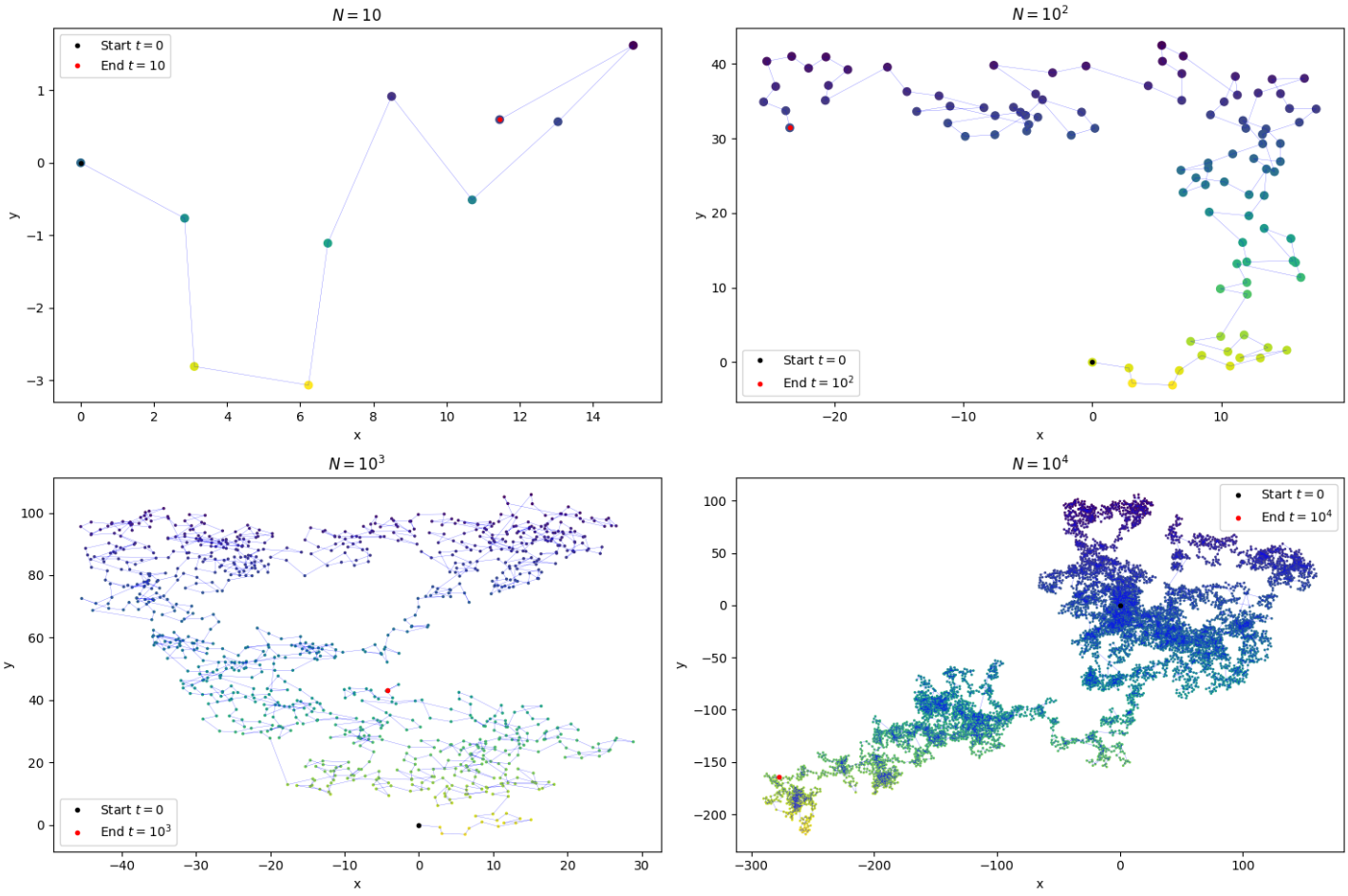
Rezultat implementacije funkcij iz prejšnjega poglavja je prikazan na slikah spodaj. Na sliki 2 je prikazana simulacija pri korakih 10, 100, 1000 in 10000 za $\mu = 3.5$ kjer lahko vidimo normalno difuzijo. Enako velja za sliko 3, kjer je prikazana simulacija za $\mu = 2.5$ kar ustreza superdifuzivnemu



Slika 2: Prikaz simulacije za $\mu = 3.5$

režimu. Na obeh slikah nam barvni gradient od rumene do vijolične barve predstavlja čas ali korak v simulaciji.

Ustvaril sem tudi kratko animacijo korakanja, ki sem jo priložil h dokumentu, vendar jo ne morem prikazati tukaj.



Slika 3: Prikaz simulacije za $\mu = 2.5$

4 Numerično računanje γ

Varianca oddaljenosti od izhodišča v odvisnosti od časa je sorazmerna z t^γ

$$\sigma^2(t) \propto t^\gamma.$$

Vrednost γ določa vrsto difuzije, določimo jo pa tako, da simuliramo let in sprehod za veliko število delcev. Ob več časih določimo prepotovano pot in tako za vsak tak čas izračunamo varianco oz. še bolj robustnejšo mero MAD, ki sem jo uporabil pri reševanju naloge. S pomočjo teh vrednosti lahko z linearno regresijo v logaritemski skali pri nekem μ določimo γ, in sicer je γ kar naklon premice. Velja:

$$\begin{aligned} \sigma^2(t) \propto \mathrm{MAD}(t)^2 &\propto t^\gamma, \\ \mathrm{MAD}^2(t) &= k \cdot t^\gamma, \\ \implies 2\,\ln(\mathrm{MAD}(t)) &= \gamma\,\ln(t) + \ln(k); \end{aligned}$$

kjer je MAD - Median Absolute Deviation:

$$\mathrm{MAD} \equiv \mathrm{median}_i\left(|X_i - \mathrm{median}_j X_j|\right)\ .$$

Metodo MAD sem implementiral s pomočjo *Scipy* knjižnice, in sicer funkcije *scipy.stats.median_abs_deviation*.

4.1 Lévyjevi poleti

Kot že opisano je pri letu/pobegu čas odvisen kar od števila korakov. Za prepotovano pot po času $t = n$ lahko vzamemo kar razdaljo od izhodišča po n korakih. Oddaljenost od izhodišča izračunamo po Pitagorovem izreku.

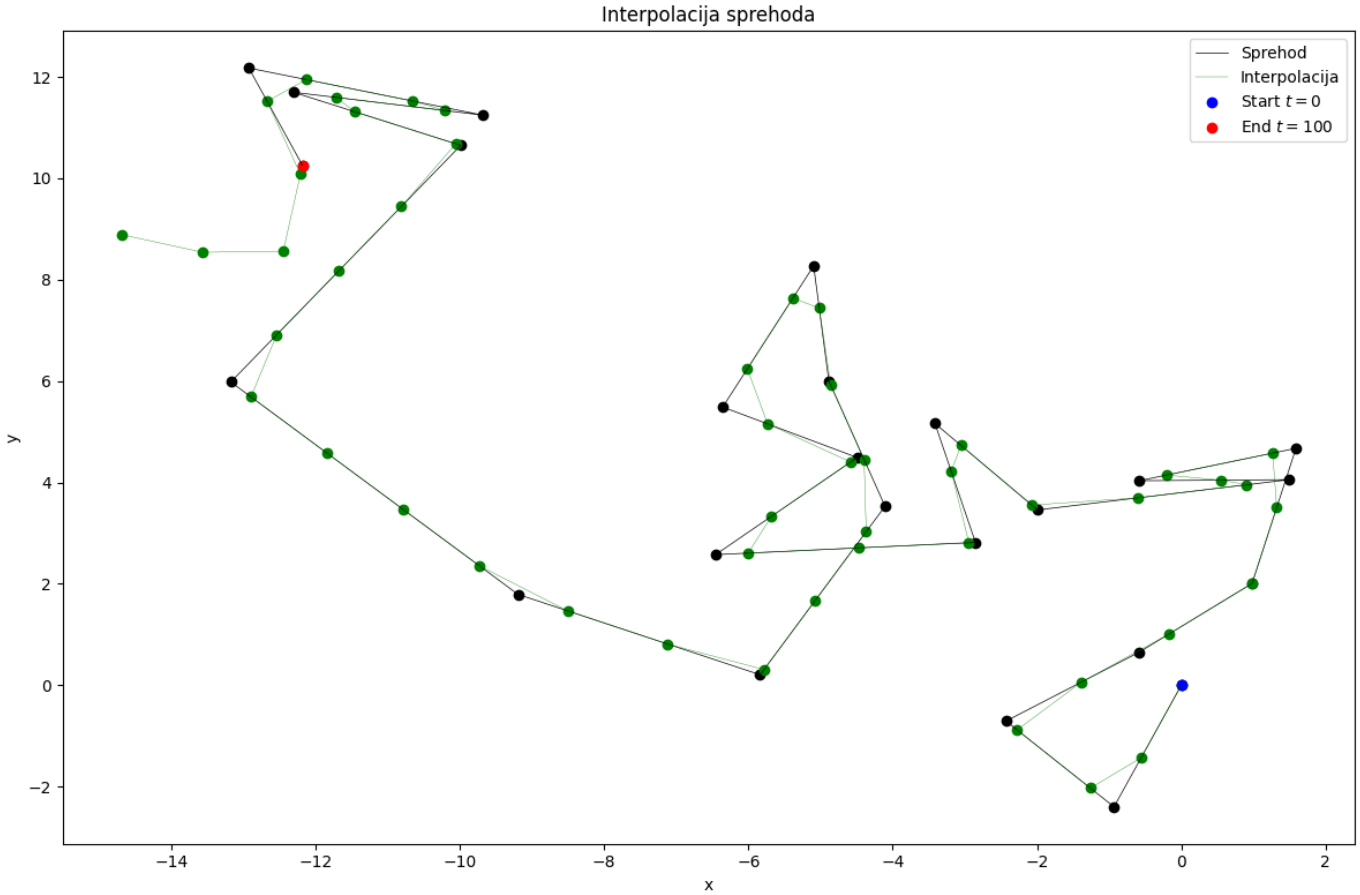
4.2 Lévyjevi sprehodi

Pri sprehodu je čas vsakega koraka nakljućen saj je sorazmeren z dolžino koraka deljeno z hitrostjo, ki pa je konstantna. Tako nimamo razdalj od izhodišča ob istem času za vsako simulacijo. Kje se delec nahaja ob času t, izračunamo s pomočjo linearne interpolacije med koraki:

$$\mathbf{r}(t) = \mathbf{r}(t_n) + \frac{t - t_n}{t_{n+1} - t_n}(\mathbf{r}_{n+1} - \mathbf{r}_n); \qquad t_n < t < t_{n+1}.$$

Tudi tokrat sem se zanašal na pametnejše od mene in sem zato kar uporabil funkcijo *numpy.interpolate*.

Na sliki 4 sem prikazal kako izgleda implementacije interpolacije. Interpolirani podatki imajo dvakrat več točk.



Slika 4: Prikaz interpolacije pri Lévyjevem sprehodu.

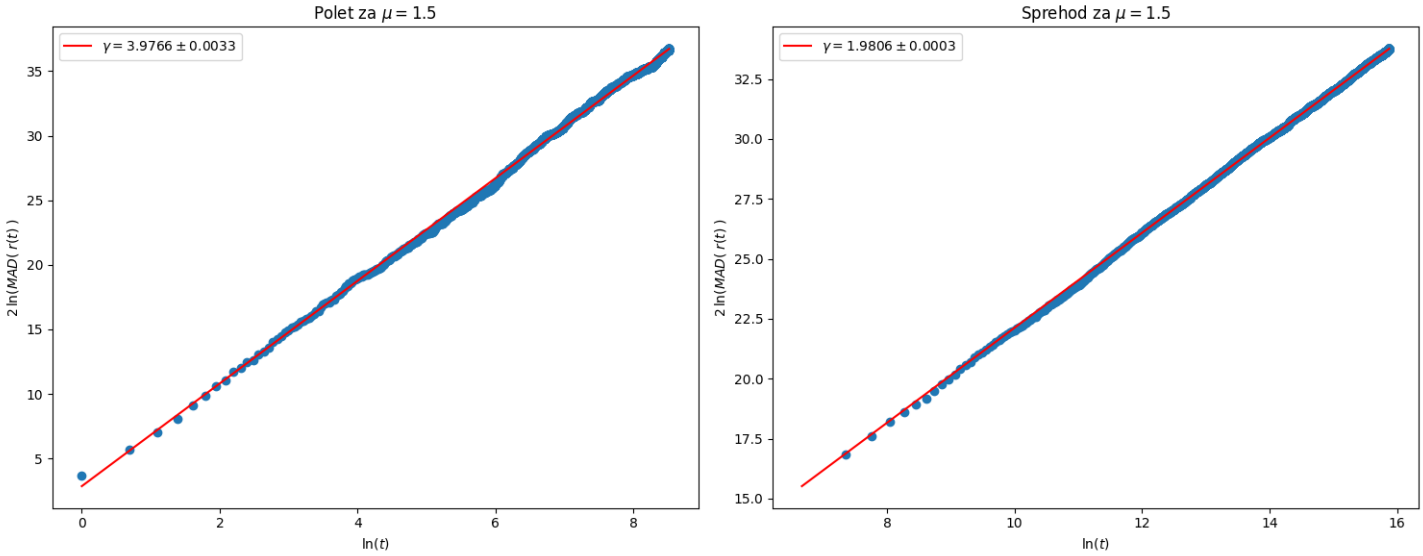
Za napako vrednosti MAD ob posameznem času sem vzel enako formulo kot za računanje negotovosti variance. Pogoj za računanje napak na takšen način je normalna razporeditev izmerkov. Enačba:

$$\delta\mathrm{MAD}(t) = \frac{\mathrm{MAD}(t)}{\sqrt{(2n)}},$$

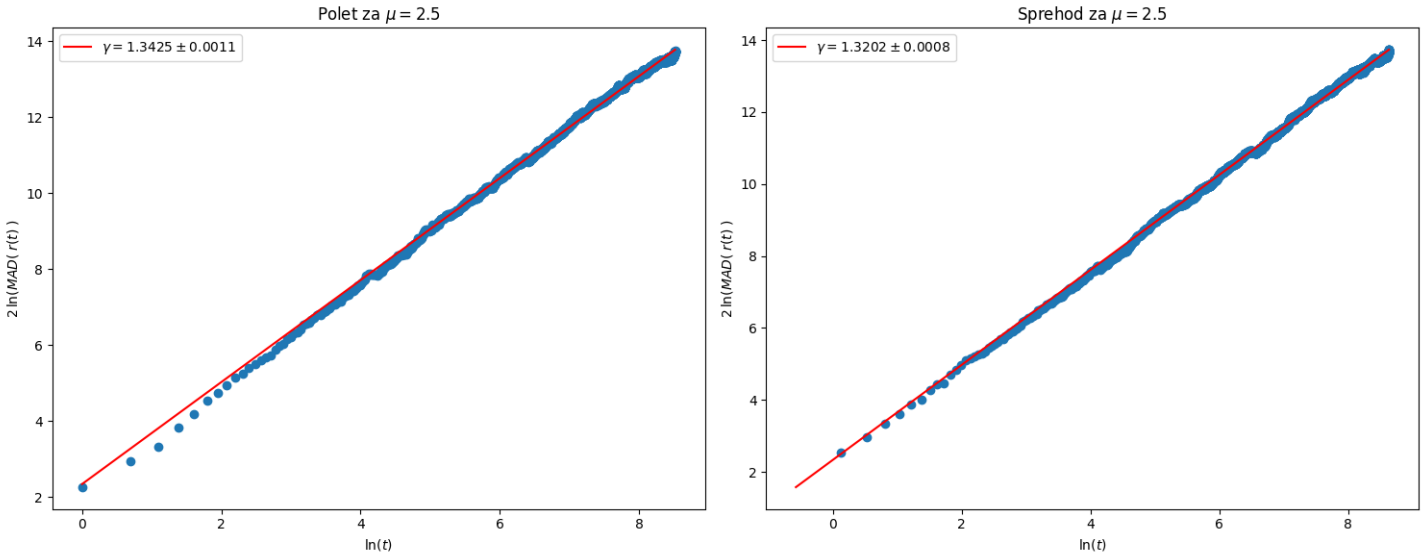
kjer je n število elementov v vzorcu, katerega MAD računamo. Za linearno regresijo sem uporabil funkcijo *scipy.optimize.curve_fit*, ki vrne tudi kovariantno matriko, katero korenimo in dobimo vrednosti absolutnih napak parametrov.

5 Grafi

Spodaj prikazani grafi na slikah 5, 6, 7 in 8 so rezultati izvajanj simulacij 500 različnih sprehodov/poletov z po 5000 koraki za različne vrednosti μ.



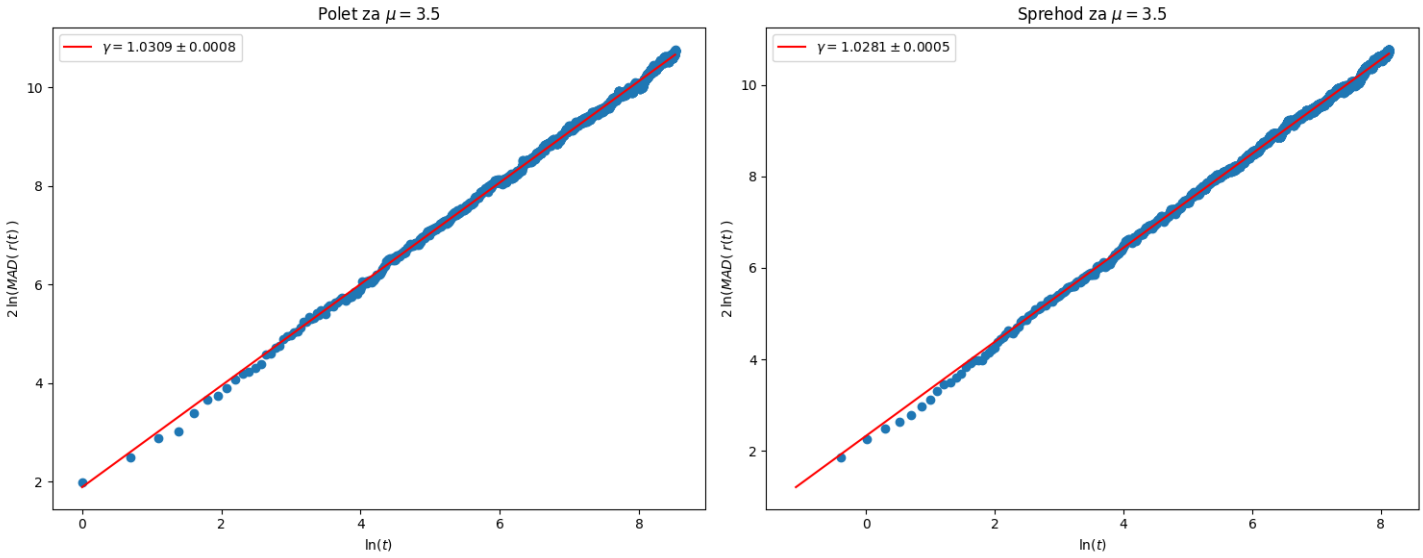
Slika 5: Prikaz simulacije pri $\mu = 1.5$.



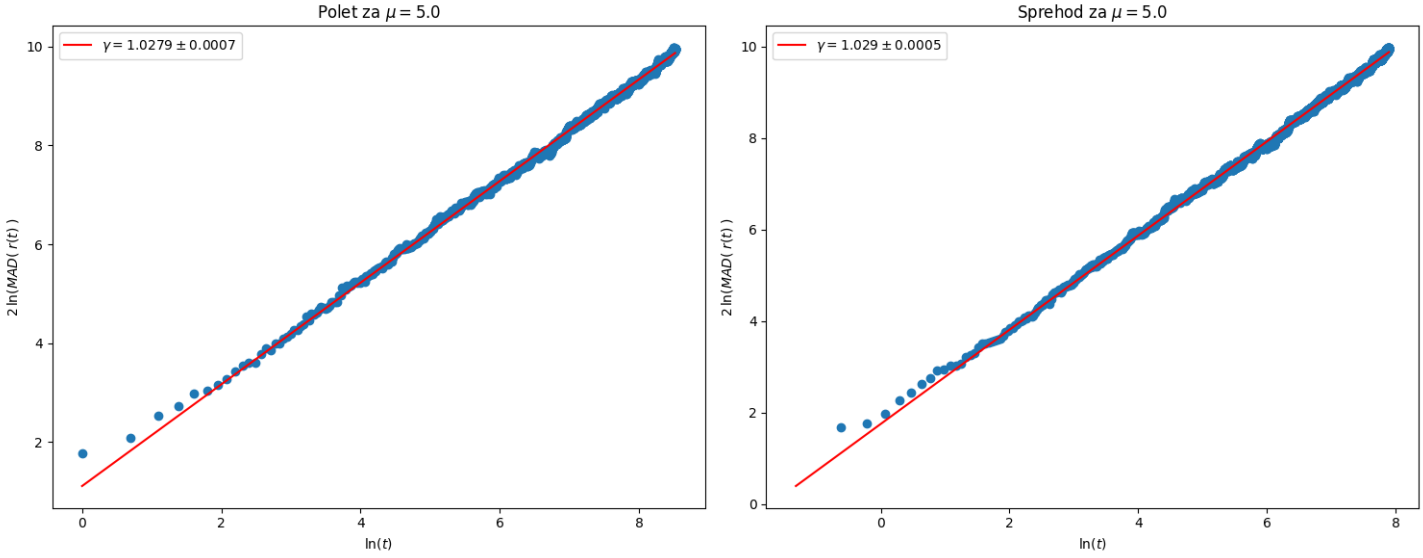
Slika 6: Prikaz simulacije pri $\mu = 2.5$.

V spodnji tabeli 1 so prikazane vrednosti regresije za zgoraj prikazane grafe. Dobljene napake se mi zdijo premajhne.

Kot je razvidno iz grafov in tabele, imamo pri vsakem μ ustrezni napovedan režim za difuzijo. Rezultati se skladajo z napovedjo.



Slika 7: Prikaz simulacije pri $\mu = 3.5$.



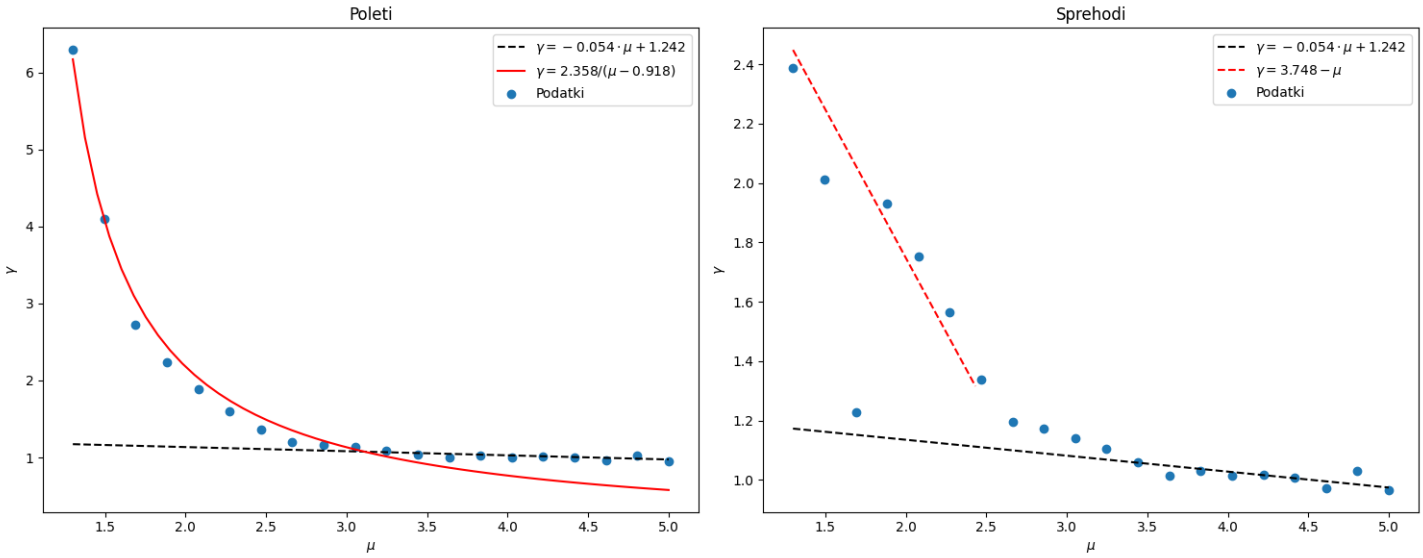
Slika 8: Prikaz simulacije pri $\mu = 5.0$.

	μ	pričakovan γ	modeliran γ	$\Delta\gamma$	režim
sprehodi	1.5	2	1.980	0.0003	balistični
	2.5	1.5	1.320	0.001	super-difuzivni
	3.5	1	1.028	0.001	normalna difuzija
	5.0	1	1.029	0.001	normalna difuzija
poleti	1.5	4	3.977	0.003	super-difuzivni
	2.5	1.3	1.343	0.001	super-difuzivni
	3.5	1	1.031	0.001	normalna difuzija
	5.0	1	1.028	0.001	normalna difuzija

Tabela 1: Vrednosti γ iz koeficienta linearne regresije za različne μ za sprehode in polete.

Za konec sem še narisal graf $\gamma(\mu)$ za več vrednosti hkrati, in primerjal dobljeno z pričakovanim modelom. Prikaz tega je na sliki 9. Tudi tu je razvidno dobro ujemanje s teoretičnim modelom, razen

pri nizkih vrednosti μ za sprehode.



Slika 9: Prikaz odvisnosti $\gamma(\mu)$.

6 Zaključek

Cilj te naloge je bil obravnavanje normalnih in anormalnih difuzij v obliki naključnih Lévyjevih sprehodov in poletov. Pri tej nalogi mi je v veliko pomoč prišlo *numpy* obdelovanje seznamov in znanje multithreadinga, saj mi je to pospešilo hitrost programa. Zopet navajam nekaj možnih dodatnih izboljšanj, in sicer lahko bi dodal test časovne zahtevnosti. Z malo več razpoložljivega časa bi morda bilo vredno zapisati kodo v $C++$ jeziku in uporaba SQL podatkovnih baz za hitrejše računanje MAD.