Naključna števila in integracije z metodo Monte Carlo

7. domača naloga pri predmetu Modelska analiza I

Domen Vaupotič

Naključna števila

Za namene kriptografije in računalniških simulacij pogosto potrebujemo algoritem, ki generira zaporedje naključnih števil. Zaradi deterministične narave računalniških algoritmov, je možno generirati le psevdonaključna števila, medtem ko je za pravo naključnost potreben eksperiment, ki se podreja naključnim dogodkom v naravi (npr. radioaktivni razpadi ali termični šum). Algoritme, ki generirajo psevdonaključna števila, imeujemo PRNG (*pseudorandom number generator*), in jih v splošnem lahko razdelimo na navadne PRNG in kriptografsko varne PRNG (počasnejši in boljši). Pri računalniških simulacijah se zadovoljimo z navadnimi PRNG, ki delujejo na osnovi modularne aritmetike, in sicer kot linearni kongruenčni generatorji. Generator iz stanja x_i preide v naslednje stanje po formuli

$$x_{i+1} = a\,x_{i+1} + c \mod m,$$

pri čemer a imenujemo multiplikator, c inkrement in m modul. Če je modul zadostno velik in parametra a in c smiselno izbrana, ima lahko generator zelo dolge periode (tipično enakega reda kot m).

Integracija Monte Carlo

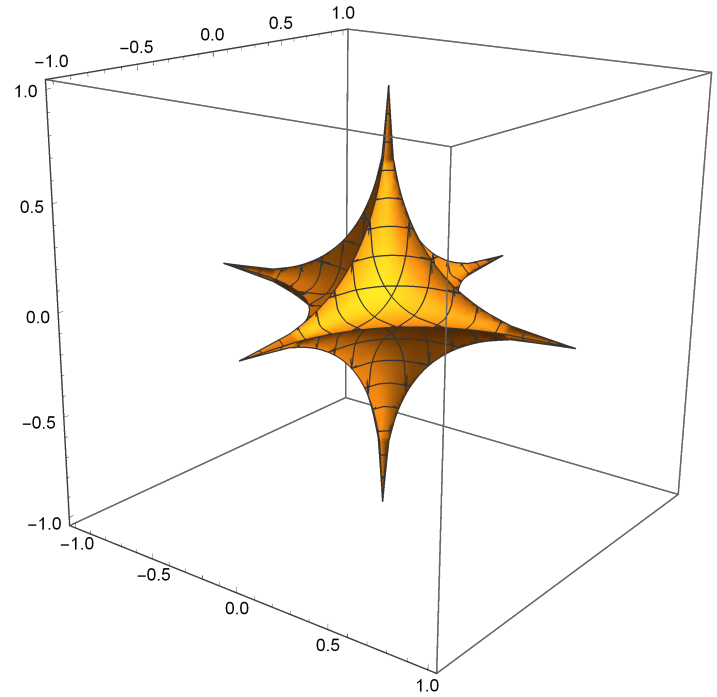
Numerično računanje integralov v več dimenzijah je posebno zahtevno, kadar so območja kompleksnejša kot navadni intervali. Ena izmed metod, ki omogoča učinkovito računanje, je metoda integracije Monte Carlo, pri kateri uporabimo generator naključnih števil, s katerim generiramo točke v neki preprostejši nadmnožici integracijskega območja.

1. naloga

Obravnavamo zvezdasto telo, ki je podano z implicitno enačbo

$$\sqrt{|x|} + \sqrt{|y|} + \sqrt{|z|} \leq 1.$$

Out[ ]=



In[ ]:=
```
zvezdica[p_] := Sqrt[Abs[p[[1]]]] + Sqrt[Abs[p[[2]]]] + Sqrt[Abs[p[[3]]]] < 1;
zvezdaOsmina =
  ImplicitRegion[Sqrt[x] + Sqrt[y] + Sqrt[z] <= 1, {{x, 0, 1}, {y, 0, 1}, {z, 0, 1}}];
```

Masa

Najprej izračunajmo maso telesa z vgrajenim numeričnim integratorjem. Pri tem zaradi simetrije integrator poženemo zgolj po prvem oktantu in rezultat množimo z 8.

In[ ]:=
```
8 NIntegrate[1, {x, y, z} ∈ zvezdaOsmina] // NumberForm[#, 16] &
```

Out[ ]//NumberForm=
```
0.0888888885246028
```

Intuicija ob rezultatu nam pravi, da je morda pravilen rezultat enak $\frac{88}{990} = \frac{4}{45} = 0,0888\ldots$ Poskusimo povečati natančnost integratorja.

In[ ]:=
```
8 NIntegrate[1, {x, y, z} ∈ zvezdaOsmina, PrecisionGoal → 10] // NumberForm[#, 16] &
```

Out[ ]//NumberForm=
```
0.088888888888885
```

Zares, vrednost integrala je očitno vsaj na 14 mest enaka vrednosti $\frac{88}{990}$. Poskusimo *Mathematico* prepričati, naj integral izračuna analitično.

```
In[•]:= Integrate[x^2 + y^2 + z^2, {x, y, z} ∈
          ImplicitRegion[Sqrt[x] + Sqrt[y] + Sqrt[z] <= 1, {{x, 0, 1}, {y, 0, 1}, {z, 0, 1}}]]
```

Out[•]=

$$\int_{\{x,y,z\}\in \text{ImplicitRegion}\left[\sqrt{x}+\sqrt{y}+\sqrt{z}\le 1\&\&0\le x\le 1\&\&0\le y\le 1\&\&0\le z\le 1,\{x,y,z\}\right]} \left(x^2+y^2+z^2\right)$$

Očitno ji bo treba nekoliko bolj pomagati ...

```
In[•]:= 8 Integrate[1, {x, 0, 1}, {y, 0, (1 - Sqrt[x])^2}, {z, 0, (1 - Sqrt[x] - Sqrt[y])^2}]
```

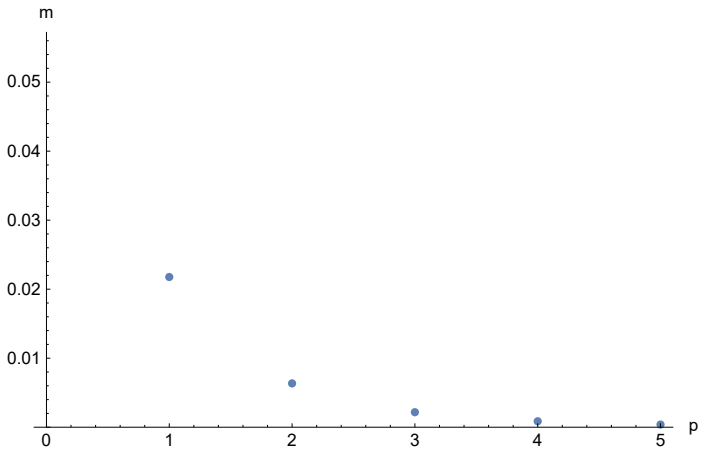
Out[•]= $\frac{4}{45}$

Gostota r^p

Izračunajmo sedaj maso, če telo ni homogeno, temveč je gostota oblike r^p. Ker se z večanjem parametra p masa premika proti izrastkom zvezde, ki predstavljajo le majhen volumen celotnega telesa, celokupna masa pada.

```
In[•]:= ListPlot[{#, 8 NIntegrate[(x^2 + y^2 + z^2)^(#/2), {x, y, z} ∈ zvezdaOsmina]} & /@
          Range[0, 5, 1], ...]
```

Out[•]=

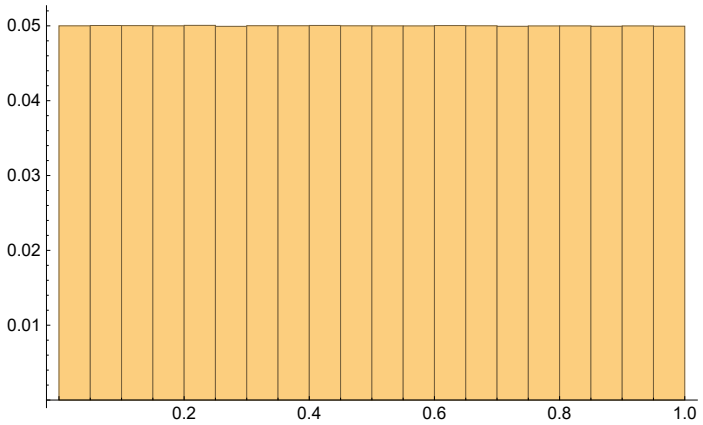


Integracija Monte Carlo

Sedaj lahko preverimo, kako natančna in točna je metoda integracije Monte Carlo. Pri tem uporabimo implementirani PRNG, natančneje funkcijo RandomReal[], ki vrača psevdonaključna števila enakomerno med 0 in 1. Preverimo na hitro vsaj, ali je porazdelitev zares enakomerna.

In[ ]:=
```
Histogram[RandomReal[{0, 1}, 10 000 000] , Automatic, "Probability"]
```

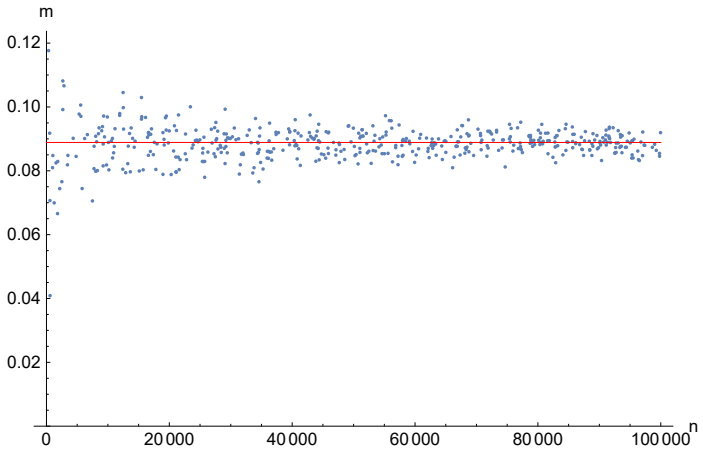
Out[ ]=



Izračun mase bomo sedaj izvedli tako, da bomo naključno žrebali točke v prvem oktantu po enakomerni porazdelitvi, nato pa izračunali, kolikšen delež jih leži znotraj želenega integracijskega območja (zvezde). Izračun bomo večkrat ponovili z različnim številom točk *n*.

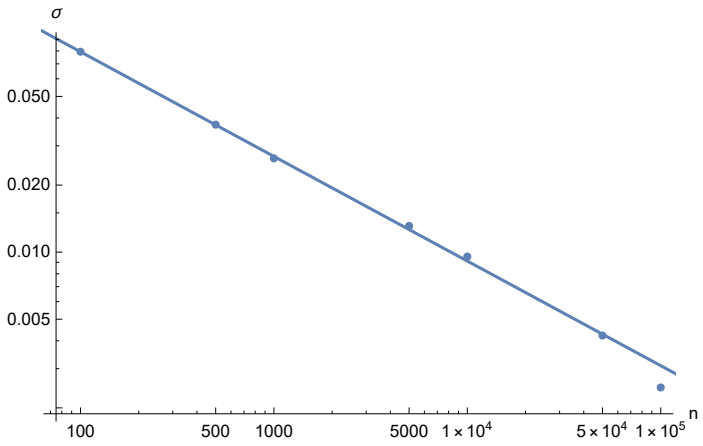
```
masaMonteCarlo[n_] := 8/n Count[zvezdica[#] & /@ RandomReal[{0, 1}, {n, 3}], True] // N;
```

Out[ ]=



Izračunajmo še raztresenost rezultatov pri različnih številih točk, tako da za različna števila točk n ponovimo integracijo 100-krat. Prilegajmo rezultatu krivuljo $A\,n^{\xi}$. Opazimo, da je parameter ξ približno -1/2, torej napaka integracije Monte Carlo pada kot $\frac{1}{\sqrt{n}}$.

Out[•]=

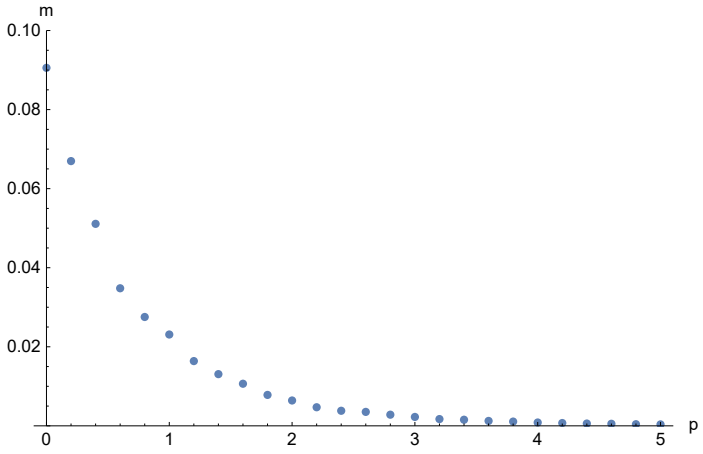


$\xi = -0.469607$

Izračunajmo še maso za nehomogeno telo z metodo Monte Carlo in 100 000 točkami.

In[•]:=
```
masaMonteCarlo[n_, p_] :=
 8/n Total[(#[[1]]^2 + #[[2]]^2 + #[[3]]^2)^(p/2) * If[zvezdica[#], 1, 0] & /@
     RandomReal[{0, 1}, {n, 3}]] // N;
ListPlot[{#, masaMonteCarlo[100000, #]} & /@ Subdivide[0, 5, 25], ...]
```

Out[•]=



2. naloga

V drugi nalogi preučujemo potovanje žarkov gama, ki nastajajo v krogli s polmerom 1. Njihova pot, ki jo prepotujejo po nastanku, je porazdeljena eksponentno s $\sim \frac{1}{\mu} e^{-\frac{s}{\mu}}$. Zanima nas, kolikšen delež žarkov gama bo ušel iz krogle. Na predavanjih smo pokazali, da je zaradi krogelne simetrije treba poznati samo radij od središča r in kot θ, tako da je oddaljenost od roba krogle nato enaka

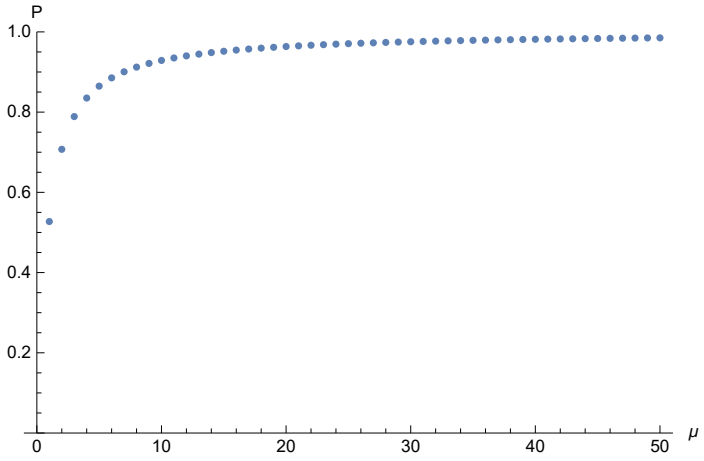
$$d(r, \theta) = -r\cos\theta + \sqrt{1 - r^2(1 - \cos^2\theta)}$$

Sledi, da je verjetnost za pobeg enaka:

$$P = \frac{\iiint e^{-\frac{d(r,\theta)}{\mu}} dV}{\iiint dV} = \frac{3}{2}\int_0^1 r^2\, dr \int_0^\pi e^{-\frac{d(r,\theta)}{\mu}} \sin\theta\, d\theta$$

Izračunajmo verjetnost najprej z uporabo vgrajenega integratorja. Pričakovano opazimo, da z večjo povprečno prosto potjo narašča delež žarkov gama, ki uidejo iz krogle, in v limiti $\mu \to \infty$ vsi žarki uidejo iz krogle.

Out[ ]=

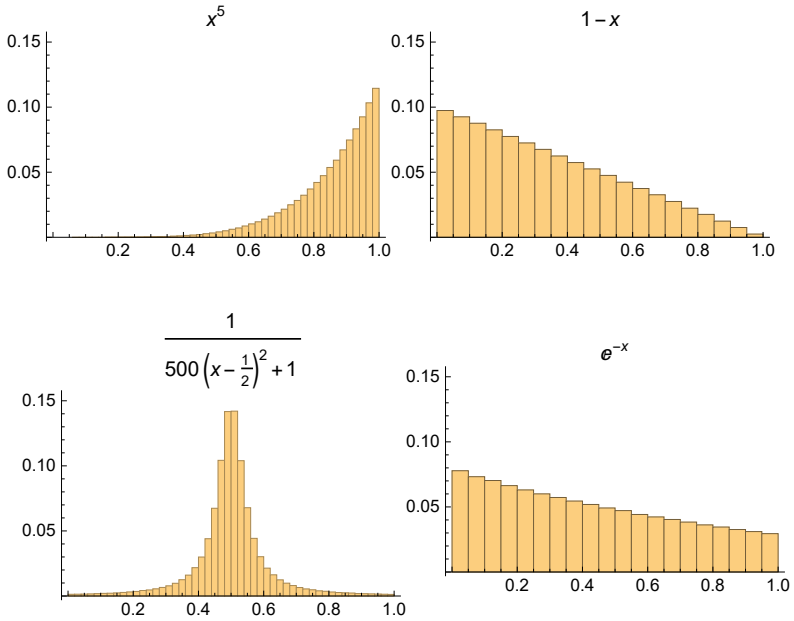


Žrebanje po poljubni porazdelitvi

Za simulacijo nastanka in potovanja žarkov gama v krogli z metodo Monte Carlo, potrebujemo algoritem, ki bo vračal naključna števila, razporejena po želeni porazdelitvi.

To lahko storimo z že vgrajeno funkcijo RandomVariate[], ki vzorči po poljubni verjetnostni porazdelitvi. Oglejmo si histograme za nekaj različnih porazdelitev, ki jih funkcija tudi sama ustrezno normira.

Out[ ]=



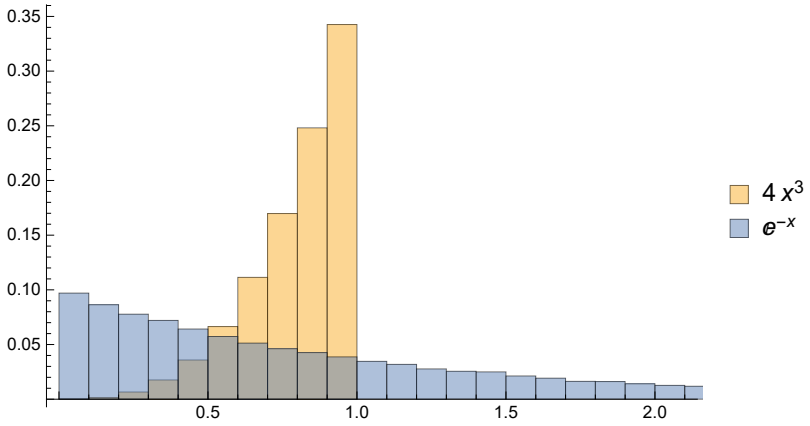
Dodajmo sedaj lastno implementacijo žrebanja, in sicer po metodi z inverzom, ki je mogoča, ko znamo poiskati inverz kumulativne porazdelitve.

In[ ]:=

```
(* Eksplicitno podamo, da naj išče inverz v pozitivnem poltraku,
sicer vrne negativnega.*)
obrniFunkcijo[f_] :=
 obrniFunkcijo[f] = InverseFunction[ConditionalExpression[f[#], # > 0] &]
najdiCDF[f_] := najdiCDF[f] = Evaluate[Integrate[f[x], {x, 0, #}]] &

zrebajPoPorazdelitvi[f_] := obrniFunkcijo[najdiCDF[f]][RandomReal[]]
```

Out[ ]=



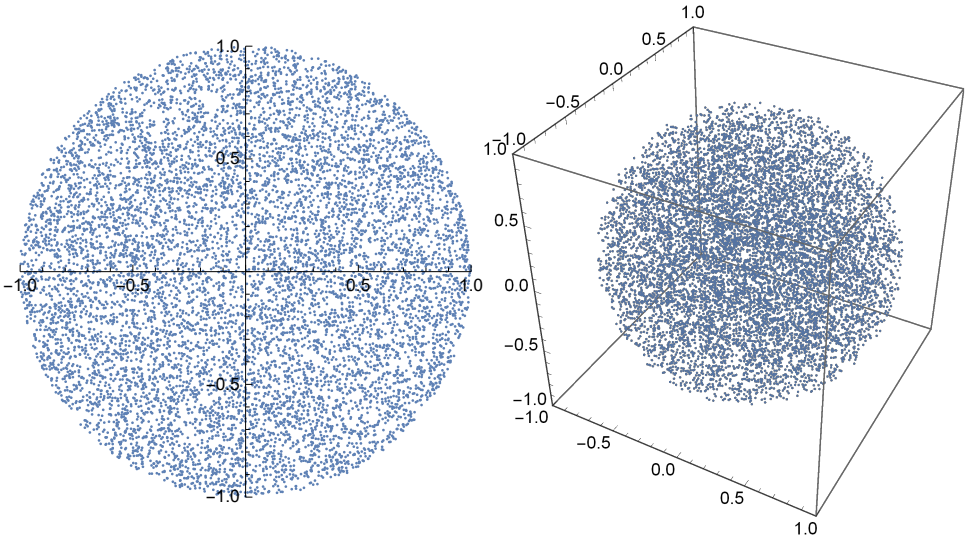
Sestavimo še funkciji za enakomerno žrebanje po krogu in po krogli.

In[ ]:=
```
zrebajPoKrogu[] := {Sqrt[RandomReal[]], 2 Pi RandomReal[]}
zrebajPoKrogli[] :=
 {CubeRoot[RandomReal[]], 2 Pi RandomReal[], ArcCos[2 RandomReal[] - 1]}

sfer2kart[p_] := Module[{r, ϕ, θ},
  {r, ϕ, θ} = p;
  {r Sin[θ] Cos[ϕ], r Sin[θ] Sin[ϕ], r Cos[θ]}]
```

Okometrično preverimo, da je žrebanje ustrezno.

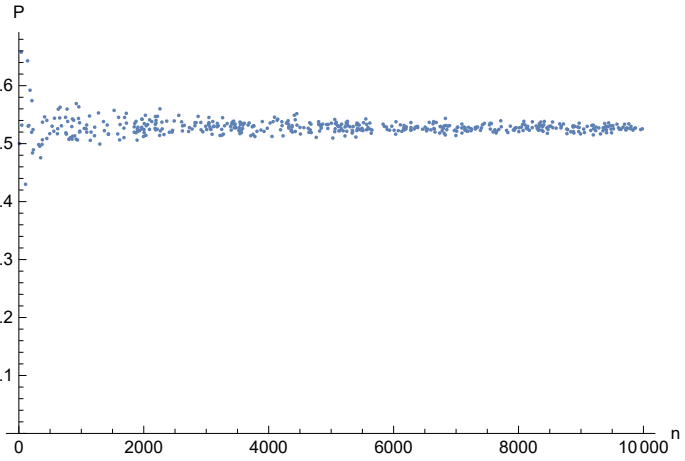
Out[ ]=



In[ ]:=
```
radijOdSredisca[r_, θ_] := -r Cos[θ] + Sqrt[1 - r^2 (1 - Cos[θ]^2)]
verjetnostZaPobeg[n_, μ_] := Count[Table[
    #[[2]] > radijOdSredisca[#[[1, 1]], #[[1, 3]]] & @
     {zrebajPoKrogli[], zrebajPoPorazdelitvi[Exp[-# / μ] / μ &]}
    ,
    n],
   True] /
  n
```

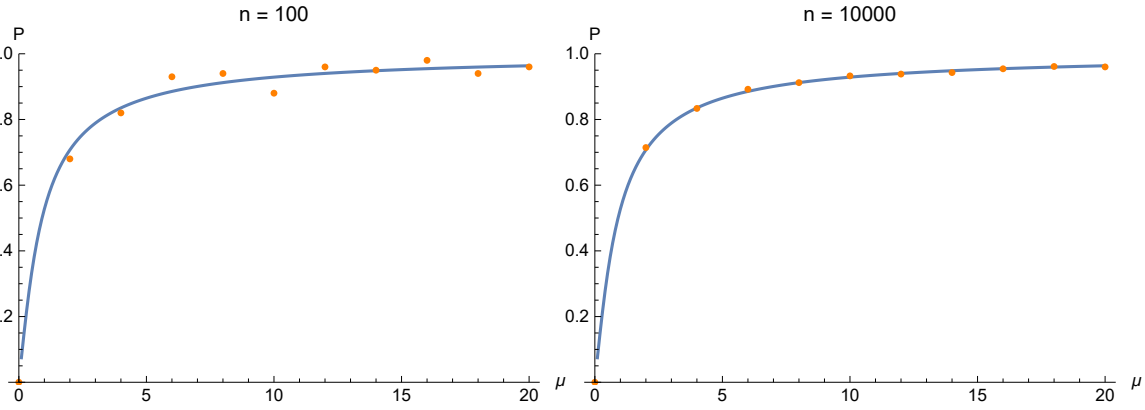
Izračunajmo delež pobeglih žarkov z različnim številom točk pri prosti poti, enaki 1. Ponovno opazimo, da se z naraščanjem števila točk zmanjšuje napaka metode Monte Carlo.

Out[ ]=



Oglejmo si še odvisnost od povprečne proste poti in primerjajmo z numerično integracijo.

Out[ ]=



3. naloga

Nevtroni se izotropno sipljejo na plošči debeline 1, tako da so njihovi zaporedni položajo (0, 0), $\vec{r}_1$, $\vec{r}_2$, $\vec{r}_3$ ...

Zveza, ki podaja naslednji položaj sipanja:

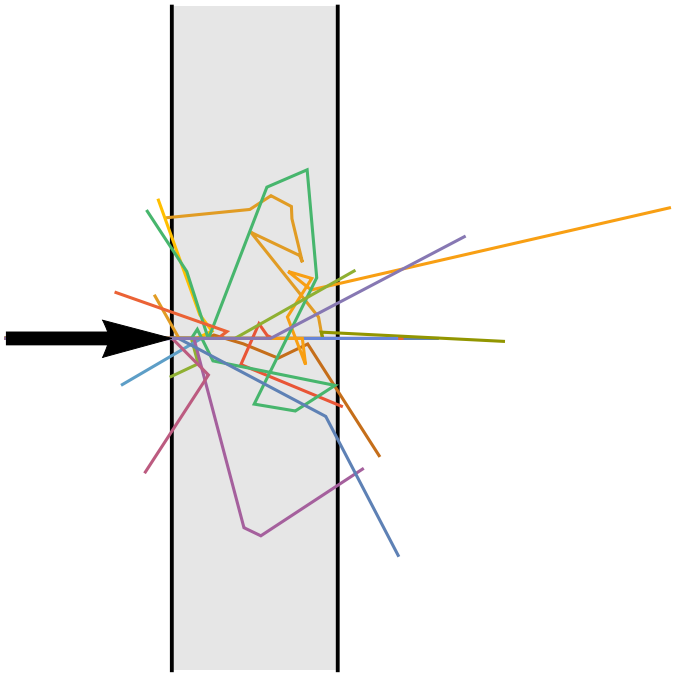
$\vec{r}_{n+1} = \vec{r}_n + \vec{N}$,

pri čemer je $\vec{N}$ naključni vektor, z dolžino, porazdeljeno po eksponentni porazdelitvi (s povprečno prosto potjo μ), in enakomerno porazdeljeno usmerjenostjo.

Nevtron bo pobegnil, če ima nek $\vec{r}_i$ komponento x večjo od 1 ali manjšo od 0.

Oglejmo si za začetek nekaj trajektorij pri $\mu = 1$.

Out[•]=



Definirajmo sedaj funkcijo, ki prejšnjemu sipanju priredi naslednje sipanje, in funkcijo, ki simulira sipanje, vse dokler nevtron ne odleti iz plošče.

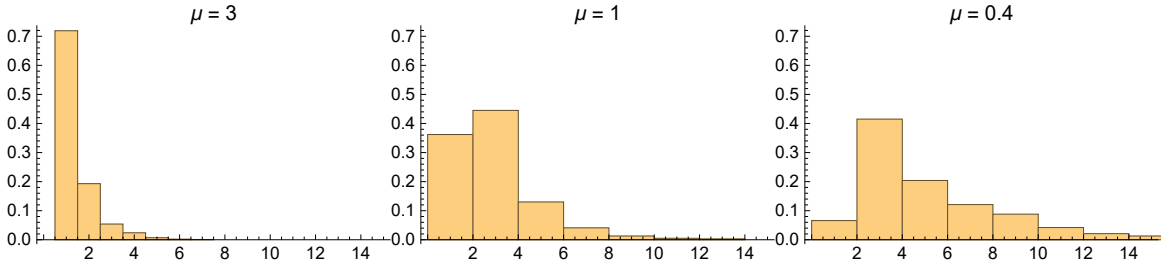
In[•]:=
```
pol2Kart[r_, ϕ_] := {r Cos[ϕ], r Sin[ϕ]};
```

In[•]:=
```
naslednjeSipanje[p_, μ_] :=
 p + pol2Kart[zrebajPoPorazdelitvi[Exp[-# / μ] / μ &], 2 Pi RandomReal[]]
izvediSipanje[μ_] := NestWhileList[naslednjeSipanje[#, μ] &,
  {zrebajPoPorazdelitvi[Exp[-# / μ] / μ &], 0}, And[#[[1]] ≥ 0, #[[1]] < 1] &]
steviloSipanj[μ_] := Length @ izvediSipanje[μ]
```

Oglejmo si porazdelitev števila sipanj pri različnih overprint prostih poteh. Opazimo, da se z naraščanjem povprečne proste poti zmanjšuje porazdelitev števila sipanj pomika v levo, torej se v povprečju

nevtron siplje manjkrat.

Out[•]=

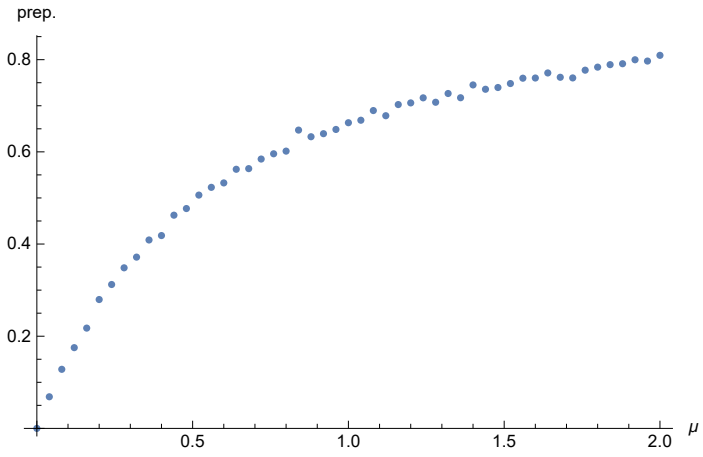


Izračunajmo sedaj še prepustnost reflektorja in izračunajmo prepustnost za različne povprečne proste poti. Pri vsak simulaciji opazujemo 5000 nevtronov.

In[•]:=
```
jePrepuscenQ[sipanje_] := Last[sipanje][[1]] > 1
delezPrepuscenih[n_, μ_] :=
 Count[#, True] & @ (jePrepuscenQ /@ Table[izvediSipanje[μ], n]) / n
```

In[•]:=
```
ListPlot[{#, delezPrepuscenih[5000, #]} & /@ Subdivide[2, 50], AxesLabel → {"μ", "prep."}]
```

Out[•]=

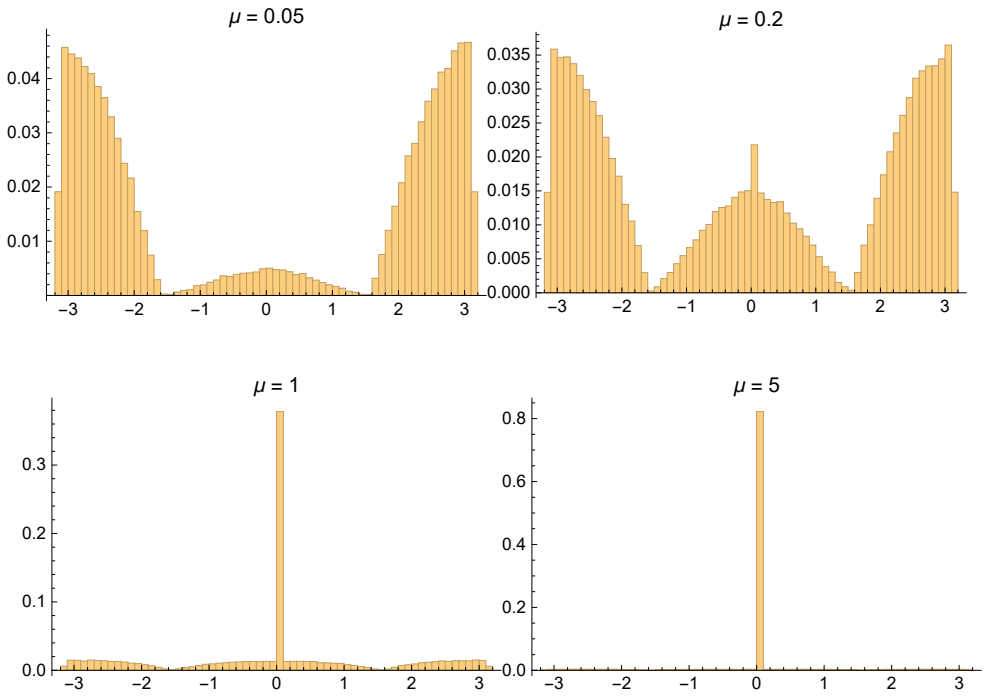


Zanimiva je tudi kotna porazdelitev prepuščenih in odbitih nevtronov.

In[ ]:=
```
kotZadnjegaNevtrona[sipanje_] :=
 ArcTan @@ (sipP[[-1]] - sipP[[-2]] /. sipP → {{0, 0}} ~Join~ sipanje)
```

In[ ]:=
```
Histogram[Table[kotZadnjegaNevtrona[izvediSipanje[#]], 100 000], 50, "Probability",
    PlotLabel → "μ = " <> ToString[#], ImageSize → 250] & /@ {.05, .2, 1, 5} // Row
```

Out[ ]=



Dodatna naloga

Preučujemo čas oddaje nalog za Modelsko analizo. Radi bi ovrednotili, ali lahko oddane čase za različne naloge interpretiramo kot različne vzorce iz iste porazdelitve.

Uvozimo podatke ...

In[•]:=
```
SetDirectory[NotebookDirectory[]];
podatki = (Import[#] // Flatten) & /@ FileNames["podatki\\*.dat"];
```

Pretvorimo čas v minute ...

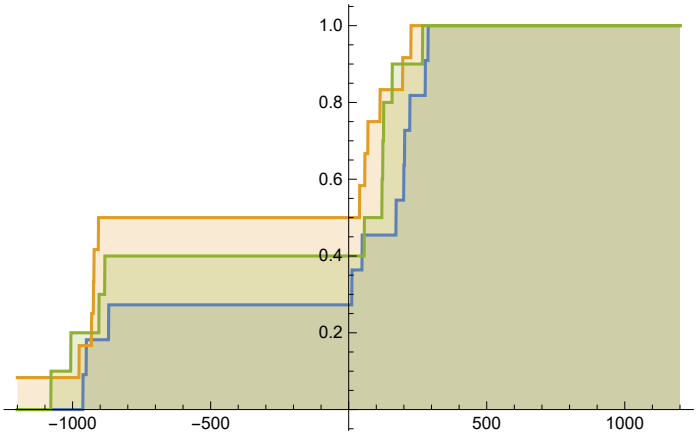
In[•]:=
```
casiOddaje = Partition[podatki, 13];
minuteOddaje =
  Map[ToExpression @ StringReplace[#, dan__ ~~ ":" ~~ ure__ ~~ ":" ~~ minute__ :>
      ToString @ ((24 * 60) ToExpression[dan] + 24 ToExpression[ure] +
         ToExpression[minute])] & , casiOddaje, {2}];
```

Za vsako nalogo sestavimo empirično diskretno porazdelitev in izračunamo kumulativne porazdelitve ...

```
distribucijeOddaje = Map[EmpiricalDistribution, minuteOddaje, {2}] // Quiet;
CDFOddaje = Map[CDF[#, t] &, distribucijeOddaje, {2}];
```

Oglejmo si nekaj primerov oddajanja za leto 2010 in naloge 1, 2, 3.

Out[•]=



S testom Kolmogorov-Smirnov lahko sedaj po parih preverimo, ali se porazdelitvi oddajanj ujemata. Pri tem postavimo hipotezi :

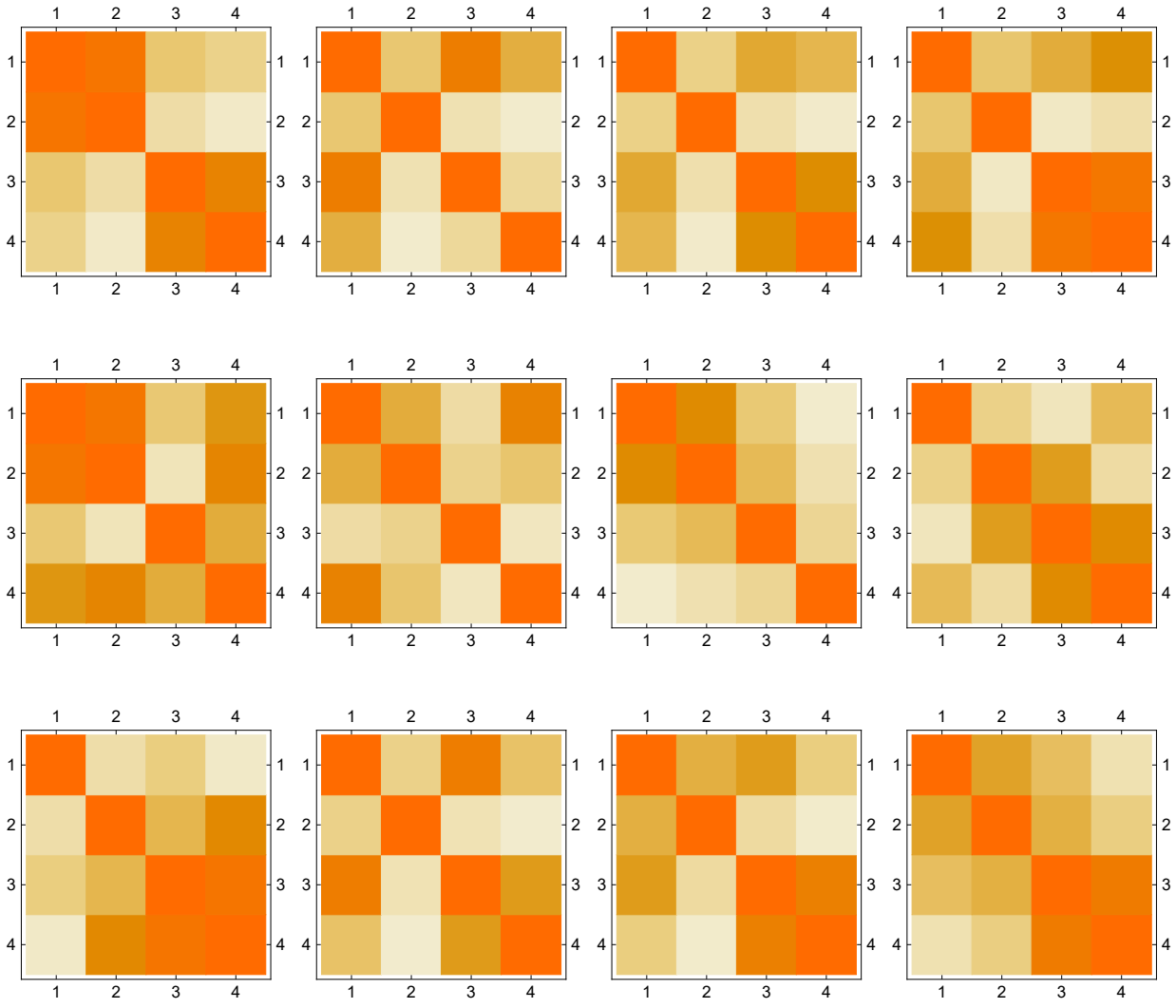
H0 : podatki prihajajo iz iste porazdelitve

H1 : podatki ne prihajajo iz različnih porazdelitev

Ničto hipotezo lahko zavržemo z verjetnostjo 1 - α, če je izračunana vrednost p testa K-S manjša kot α.

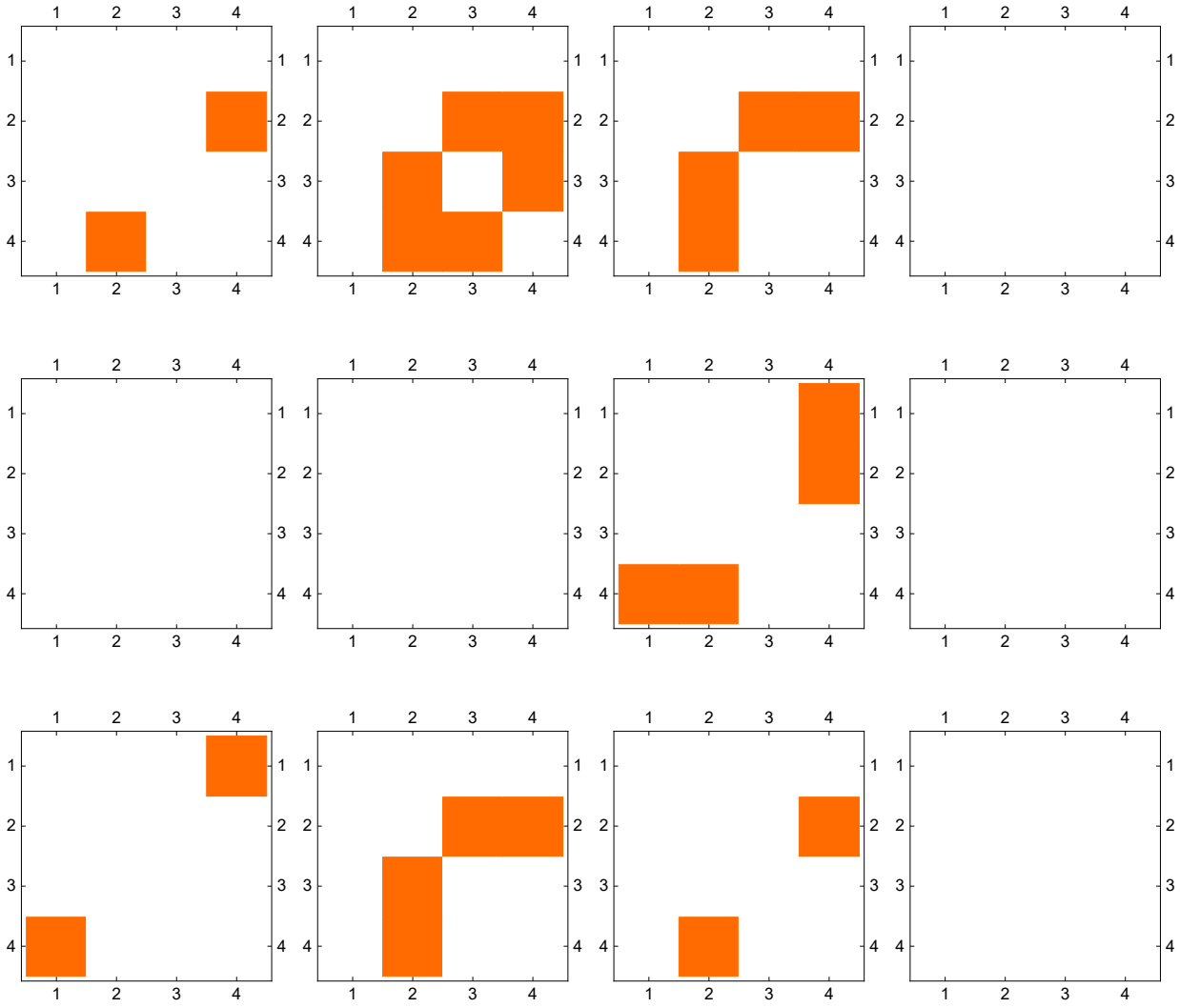
Oglejmo si vrednosti p za vse pare let in vse naloge.

Out[ ]=



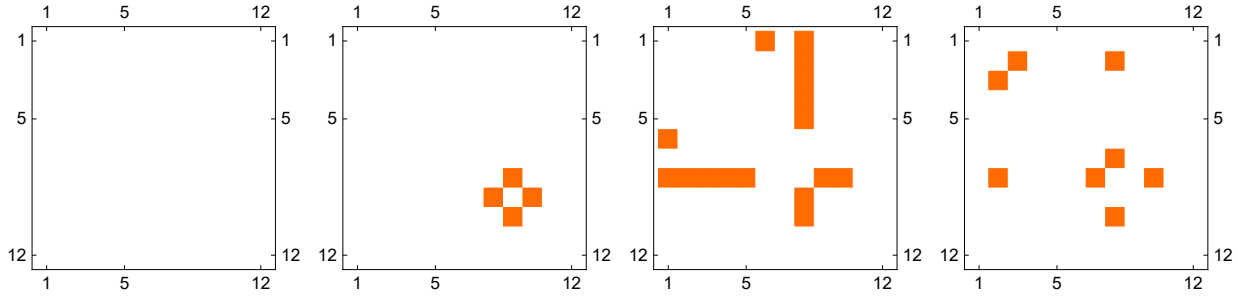
Označimo samo tiste pare, pri katerih je vrednost p manjša od 5 %. Opazimo, da za večino parov ne moremo ovreči hipoteze o isti porazdelitvi. Pri določenih nalogah (npr. 2. in 3.) daje test K-S značilne razlike med posameznimi letniki.

Out[ ]=



Preverimo, ali se naloge znotraj posameznih let med seboj razlikujejo.

Out[ ]=



Oglejmo si porazdelitvi v letu 2013 (3. leto) za nalogo 8, za katero test K-S trdi, da se značilno razlikuje od nalog 1-5. Opazimo, da so 8. nalogo študentje oddajali kasneje.

In[ ]:=
```
DiscretePlot[{CDFOddaje[[3, 8]], CDFOddaje[[3, 1]]}, {t, -1200, 1200},
 PlotLegends → {"8. naloga", "1. naloga"}, AxesLabel → {"Δt"}]
```

Out[ ]=