Uvod u Monte Carlo metodu

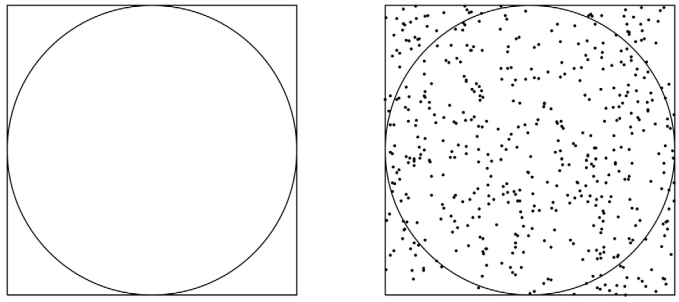
Milivoj Uroić[1]

Ovaj puta nije riječ o auto-utrkama, niti o glamuru, niti o turizmu ovog poznatog gradića u kneževini Monako, iako ima dodirnih točaka s kazinima i kockarnicama kojih je tamo u izobilju. No umjesto koristoljublja koje diktira događanja oko "igara na sreću", motiv za ovu, u fizici sve češće korištenu metodu, je želja za znanstvenom spoznajom. O čemu se radi?

Monte Carlo simulacije, analize i obrade podataka plodonosno koristimo u fizici kada je ispunjen sljedeći uvjet: problem koji želimo analizirati možemo dobro opisati matematičkim modelom, ali taj model ne znamo ili ne želimo egzaktno riješiti. Umjesto egzaktnog matematičkog rezultata koji pokriva sve moguće ulazne i izlazne parametre fizičkog sistema, Monte Carlo metoda služi se slučajnim odabirom i statističkom obradom. Iako nije uvjet, metoda je vrlo pogodna za računalnu obradu, uz korištenje generatora slučajnih brojeva i statističke obrade velikih količina podataka, gdje je brzina računanja velika prednost.

Ako mislite da nakon ovog uvoda slijedi rješavanje "nerješivih" matematičkih problema varate se, trebalo bi početi s nečim jednostavnim, a pomaže i ako odgovor unaprijed znamo. Takav je sljedeći primjer.

Kolika je površina kruga jediničnog radijusa? Pa naravno, $P = r^2\pi = 1^2\pi = \pi$. Očito, poanta nije rezultat, nego prikaz metode na jednostavnom primjeru. Zamislimo kvadrat 2×2 s upisanim krugom (slika 1a). Naš bi zadatak mogli izraziti pitanjem *Koliki udio površine kvadrata zauzima krug?* Monte Carlo nudi neobičnu metodu odgovaranja: Odaberemo nasumice točke unutar kvadrata. Omjer broja točaka unutar kruga i ukupnog broja točaka (unutar kvadrata) bit će, za velike brojeve točaka vrlo blizu omjera površina (slika 1b, 500 točaka).



Slika 1a i 1b.

Potrebno matematičko znanje ne uključuje poznavanje broja π (egzaktan postupak), nego samo kriterij pripadanja krugu, dakle računanje udaljenosti odabrane točke i središta Pitagorinim poučkom. Rezultati otprilike ovako izgledaju: Od prvih 10 točaka 8 je u krugu, od 50 u krugu je 37, od 100 u krugu je 74, od 1000...

Primijetimo da se porastom broja točaka smanjuje pogreška. A računalo je idealan stroj za ponavljanje jednostavne operacije mnogo puta.

[1] Autor je znanstveni suradnik na Institutu "Ruđer Bošković" u Zagrebu.

n	k	P
10	8	3.2
100	74	2.96
1000	788	3.152
10000	7804	3.1216
100000	78498	3.13992

Osim osnovne ideje, namjera mi je grafički prikazati neke jednostavne slučajeve. Za to mi trebaju dva jednostavna programerska zahtjeva: kako odabrati slučajan broj i kako iscrtati točku i dužinu na ekranu? Iskusniji programeri će sami naći svoje rješenje, ja ću (na nečiji užas) ponuditi rješenje u programskom jeziku BASIC. Ako vas programiranje ne zanima, koristit ćete se samo tuđim programima, uz rizik da oni ne rade točno ono što vi želite...

U BASIC-u, kao i većini programskih jezika postoji ugrađena funkcija za dobivanje slučajnog broja: RND. Jednostavan program koji ispisuje 10 slučajnih brojeva bi bio:

```
for i=1 to 10
 print rnd
next i
sleep
```

Naredba "sleep" jednostavno čeka na pritisak tipke ili zatvaranje prozora, bez nje bi se rezultat prikazao i nestao tako brzo da ga ne bismo ni vidjeli na ekranu. U navedenom primjeru brojevi će biti decimalni između 0 i 1 (nula uključena, jedan nije). Ako bi umjesto toga htjeli cijele brojeve između 1 i 1000, drugi red bi zamijenili s

```
print int(rnd*1000)+1,    (a)
```

Množenjem s 1000 interval $[0,1\rangle$ postaje $[0,1000\rangle$. Funkcija INT daje najveće cijelo dobivenog broja, dakle cijeli broj od 0 do 999. Dodamo 1 da bi dobili broj od 1 do 1000.

Tako bi npr. za interval od 4 do 8 na tri decimale stavili izraz

```
print int(rnd*4000)/1000+4,    (b)
```

a za dvoznamenkaste parne brojeve $(10, 12, 14, ..., 98)$ izraz

```
print int(rnd*45)*2+10.    (c)
```

U navedenoj tablici prikazani su dobiveni rezultati u navedene četiri varijante programa:

rnd	(a)	(b)	(c)
0.266235352	267	5.064	32
0.990631104	991	7.962	98
0.782470703	783	7.129	80
0.821899414	822	7.287	82
0.028594971	29	4.114	12
0.224212646	225	4.896	30
0.318572998	319	5.274	38
0.106414795	107	4.425	18
0.685943604	686	6.743	70
0.151153564	152	4.604	22

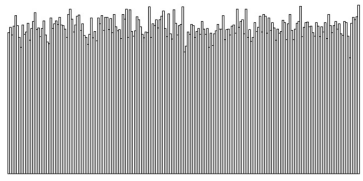
Grafički prikaz

Prikaz rezultata u gornjoj tablici pogodan je za desetak brojeva, ali bi bio vrlo nepraktičan za veći broj vrijednosti, npr. koordinate 500 točaka ucrtanih na slici 1b. Za brojnije skupove podataka pogodnije je rezultat prikazati grafom, histogramom ili slikom. Danas uobičajeni način je programom stvoriti tablicu (nalik na gornju), pa ju onda uvesti (importirati) u neki od programa za grafički prikaz. Druga je mogućnost da vaš program obavi oba posla, ali onda treba znati kako crtati po grafičkom prozoru.

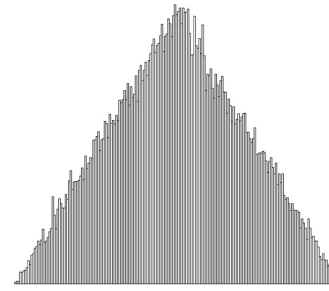
Bez daljnjeg odugovlačenja, navest ću program u FreeBasic-u (dijalekt BASIC-a):

```
#lang"fblite"       'oznaka BASIC dijalekta
screenres 800,600   '800*600 prozor za crtanje
dim f(200)          'cjelobrojni niz
for i=1 to 50000    'broj događaja
  x=int(rnd*200)+1  'računanje slučajnog broja od 1 do 200
  f(x)+=1           'pribrajanje događaja
next i              'kraj petlje
for x=1 to 200      'crtanje histograma f(x)
  line (3*x+100,550)-step(3,-f(x)),15,B
next x
sleep               'pauza do pritiska tipke
```

Ovaj program će nacrtati histogram raspodjele 50 tisuća slučajnih brojeva u 200 intervala jednake duljine. Trebalo bi ga shvatiti kao "Hello world" program za Monte Carlo: Ako vam ispravno proradi na računalu, bez problema ćete isprobati sve navedene primjere u ovom članku, a i otvorene su mogućnosti izmijeniti neki detalj i vidjeti kako to utječe na ishod. Program bi trebao otvoriti grafički prozor i iscrtati histogram kao na slici 2.



Slika 2.



Slika 3.

Dobili smo jednoliku distribuciju događaja, što nije baš interesantno, osim da uočimo i procijenimo da učestalost događaja odstupa od srednje vrijednosti. Ako bismo htjeli neku drugu distribuciju, možemo u peti red staviti drukčiji izraz, npr.

```
x=int((rnd+rnd)*100)+1
```

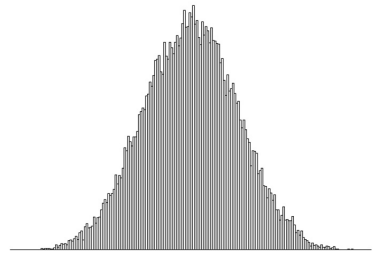
Netko će primijetiti: "Pa nije li rnd*200 isto što i (rnd+ rnd)*100?" Odmah ćemo vidjeti da nije. Histogram naime izgleda kao na slici 3.

Razlika je u tome što prvi izračun daje ravnomjernu raspodjelu po cijelom intervalu, dok drugi preferira sredinu, zato što uzimamo dva slučajna broja i njihove (ravnomjerne)

raspodjele se isprepliću (matematički strogo: konvoluiraju). Tako je npr. bacanjem jedne kocke jednaka učestalost brojeva 1 do 6, dok je za zbroj na dvije kocke najvjerojatnija srednja vrijednost 7, dok su male i velike vrijednosti (2 i 12) najmanje vjerojatne. Nastavimo li slučajne brojeve zbrajati međusobno, dobit ćemo odstupanje od srednje vrijednosti određeno nizom malih nezavisnih odstupanja, što teži prema Gaussovoj raspodjeli:

```
x=int((rnd+rnd+rnd+rnd+rnd)*40)+1
```

Histogram (približno Gaussijan) za 25000 događaja izgleda kao na slici 4.



Slika 4.

Odabir točke na površini

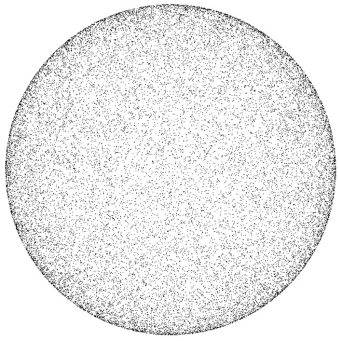
Iz gornjih se primjera vidi kako treba odabrati vrijednost iz nekog intervala da dobijemo ravnomjernu ili neke neravnomjerne raspodjele. Kako bismo odabrali točku unutar kvadrata (kao u prvom primjeru)? Ravnomjernu distribuciju bismo dobili odabirom x i y koordinate na način opisan u primjeru na slici 2, za svaku koordinatu posebno. No kako bi odabrali točku npr. na sferi? Ako želimo ravnomjeran raspored, ne možemo jednostavno odabrati sferne koordinate ravnomjernom raspodjelom, jer bi tada dobili gušće točke oko polova sfere, nego na "ekvatoru". No poslužit ćemo se matematičkim trikom: raspodjela *projekcija* na polarnu os je ravnomjerna, pa umjesto da odaberemo jednoliku gustoću sfernih koordinata (θ, ϕ), uzet ćemo jednoliku gustoću $(\cos\theta, \phi)$ i izračunati θ. To bi ovako izgledalo, uz crtanje na ekran u projekciji kugle zarotirane za kut *rot*:

```
#lang"fblite"
screenres 800,600
dim as double cth,sth,x,y,z,phi,pi=4*atn(1)
dim as double rot=0.8
for i=1 to 50000
  cth=1-rnd*2
  sth=sqr(1-cth*cth)
  phi=rnd*2*pi
  x=280*sth*cos(phi)
  y=280*(sth*sin(phi)*cos(rot)-cth*sin(rot))
  z=280*(sth*sin(phi)*sin(rot)+cth*cos(rot))
  if z>0 then pset (400+x,300+y)
next i
sleep
```

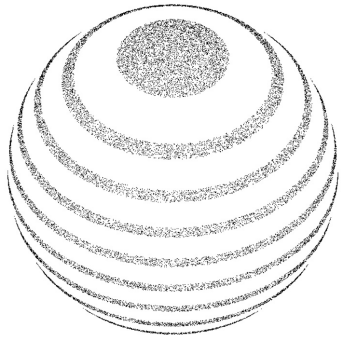
Rezultat je prikazan na slici 5. Naravno, ako su točke ravnomjerno raspoređene, nećemo vidjeti učinak rotiranja kugle. No ako odabir $\cos\theta$ zamijenimo nečim neravnomjernim, npr.

```
cth=1-(int(rnd*10)+rnd*0.3)/5
```

Dobit ćemo sliku 6. Rado bi potaknuo čitatelje da sami isprobaju razne izraze, i pogledaju pod raznim kutevima (*rot*).



Slika 5.



Slika 6.

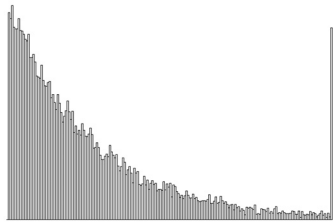
Gdje su tu rezultati?

Evo nas pri kraju članka, a tek znamo kako odabrati slučajan broj ili točku, a "izračunali" smo jedino π, i to prilično grubo. Uistinu, upotreba Monte Carlo metode tek počinje ovdje, gdje članak završava. Ipak, za kraj bi pokazao ni manje ni više nego zakon radioaktivnog raspada (isti matematički model ima brojne primjene u drugim oblastima osim nuklearne fizike).

Zamislimo da se sustav sastoji od mnoštva atoma, i da se u određenom vremenskom intervalu raspadne njih 2%. Kako će njihov broj ovisiti o vremenu? Rješenje Monte Carlo metodom bi izgledalo ovako:

```
#lang"fblite"
screenres 800,600
dim f(200)
for i=1 to 25000
  t=0
  do
    t+=1
  loop until rnd<0.02 or t=200
  f(t)+=1
next i
for x=1 to 200
  line (3*x+100,550)-step(3,-f(x)),15,B
next x
sleep
```

Histogram koji prikazuje koliko je atoma "preživjelo" x vremenskih intervala izgleda ovako:



Slika 7.

Vidi se da program izvodi do-loop petlju dok se ne dogodi događaj s 2% vjerojatnosti $(rnd < 0.02)$, a ako atom preživi odabranih 200 intervala, pripisuje se zadnjem stupcu. Histogram potvrđuje eksponencijalnu ovisnost broja atoma u vremenu, dok zadnji stupac "sumira" sve preostale događaje.

Zaključak

Namjera članka nije da čitatelj samo pročita, već i da isproba na računalu pojedine korake opisane u tekstu. Ako vam se ne sviđa Freebasic, odaberite programski jezik koji vam se sviđa, i na njega prevedite ove naredbe. Za pomoć, na internet adresi [3] naći ćete ovaj tekst i primjere u njemu, a na adresi [2] sve o korištenju Freebasica. Kao i uvijek, pojašnjenje pojmova potražite on-line na Wikipediji [1].

Literatura

[1] `en.wikipedia.org/wiki/Monte_Carlo_method`

[2] `www.freebasic.net`

[3] `lnr.irb.hr/milivoj/MFL-MonteCarlo.zip`