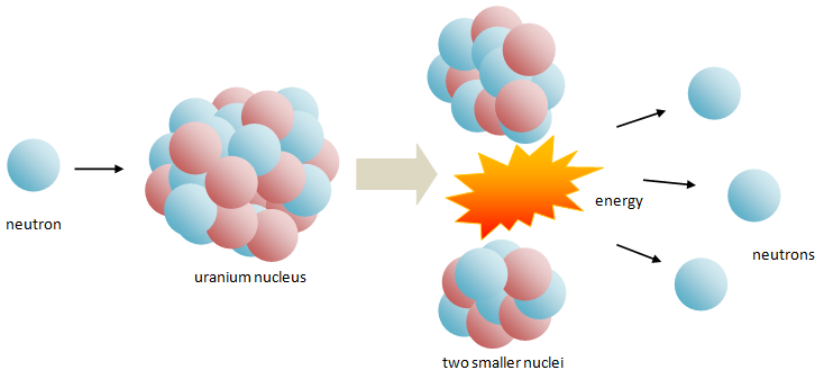
"A human is a random variable, with a hope to find a function to let his life have some value."

Jinesh Parakh

11

Simularea variabilelor aleatoare

Simulări Monte Carlo

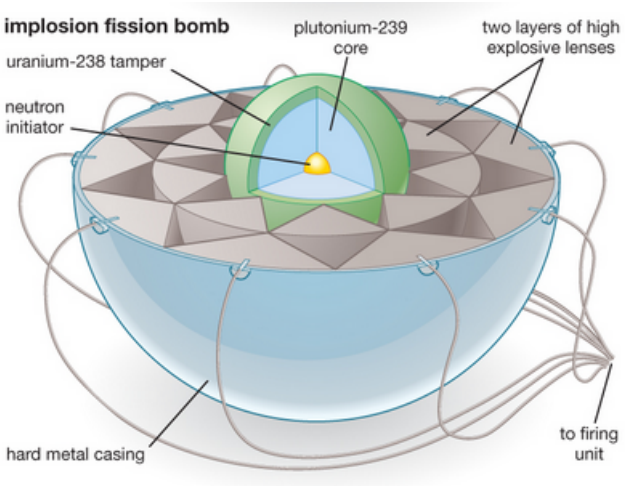


Anul 1945, doua evenimente socante au loc: explozia primei bombe nucleare si aparitia primului computer electronic. Primul eveniment a condus la doua regretabile erori, o cursa a inarmarilor, dar si la crearea unei discutabile surse de energie. Al doilea a trecut aproape neobservat, insa cateva minti luminate ca ale lui John von Neumann sau Stanislaw Ulam i-au inteles imediat impactul, cu toate ca nu aveau cum sa intuiasca pe deplin modul in care computerele ne vor schimba realitatea.

ENIAC (Electronic Numerical Integrator And Computer) a fost proiectat si construit pentru a calcula tabele balistice, pentru armata americana. John von Neumann, deja consultant in proiectul Manhattan, era preocupat de probleme termonucleare si de constructia primei bombe cu hidrogen. Prin urmare i-a contactat pe Nicholas Metropolis si Stan Frankel pentru a pregati un model computational al unei reactii termonucleare, pentru a testa ENIAC. Neumann considera ca acesta trebuia sa fie adevaratul test al calculatorului. La Los Alamos

lucrau in acea vreme si matematicienii Stanislaw Ulam si Robert Richtmyer. Impreuna au pornit de la cateva probleme cunoscute in studiul difuziei neutronilor in materialele fisionabile si au dezvoltat ceea ce astazi numim simulari sau metode Monte Carlo. Numele se pare ca a fost ales de N. Metropolis care si-a amintit ca un unchi al lui S. Ulam obisnuia sa imprumute bani de la rude pentru a pleca la Monte Carlo unde juca la cazinouri.

Una dintre probleme, discutata intr-o scrisoare adresata de Neumann lui Richtmyer, presupunea considerarea unui nucleu sferic al unui material fisionabil, invelit intr-un material numit tamper care poate reflecta inapoi neutronii evadati provocand noi coliziuni intre acestia. Se presupunea o anumita distributie a neutronilor in spatiu si viteza dar nu se luau in considerare efectele radiative sau hidrodinamice. Ideea era sa poata fi urmarita dezvoltarea unui numar mare de lanturi neutronice, ca efect al imprastierii, absortiei, fisiunii sau evadarii.



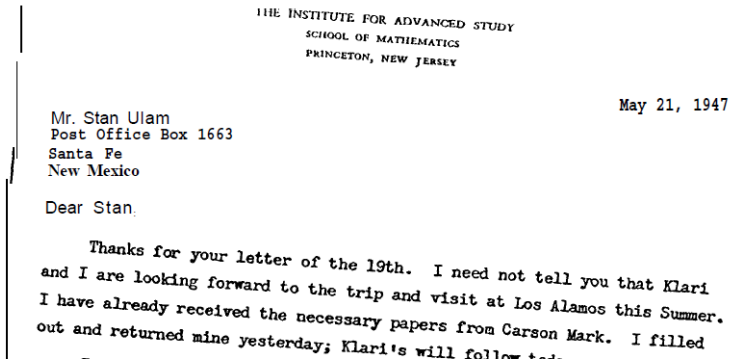
In fiecare stadiu trebuiau luate o serie de decizii bazate pe probabilitati statistice, adecvate factorilor fizici si geometrici. Printre acestea enumeram: pozitia in spatiu a neutronilor, viteza, pozitia primei coliziuni, natura acestei coliziuni. Daca se decidea ca o fisiune a avut loc trebuia decis numarul de neutroni rezultati. Daca coliziunea rezulta intr-o imprastiere trebuia decis noul impuls al neutronilor. Cand un neutron traversa un material se luau in considerare caracteristicile noului mediu, rezultand in final o istorie genealogica a fiecarui neutron. Procesul era repetat pentru alti neutroni pana ce o imagine statistica valida era obtinuta.

Initial problema transportului neutronilor fusese abordata analitic, dar rezultasera ecuatii greu de manevrat. Stanislaw Ulam povestea ca in 1946, in timp ce juca Solitaire, ii venise in gand ideea sa estimeze sansele de a castiga la acest joc, cu un pachet de 52 de carti bine amestecate.

> *"Dupa ce am pierdut o multime de timp incercand sa le estimez, prin calcule combinatoriale, m-am intrebat daca nu cumva o metoda mai practica decat "gandirea abstracta" ar insemna sa asez cartile, sa zicem de 100 de ori, si sa calculez numarul de jocuri incununate de succes. Asta era deja posibil de imaginat, odata cu inceputul noii ere a computerelor rapide, si m-am gandit imediat la problemele difuziei*

neutronilor si mai general la cum ar putea fi schimbate procese descrise de ecuatii diferentiale intr-o forma echivalenta ca sa poata fi interpretata ca fiind o succesiune de operatii aleatoare."

Ulterior, S. Ulam i-a prezentat ideea lui von Neumann, care a fost fascinat de posibilitatea de a face esantionari folosind noile metode electronice de calcul. O astfel de abordarea parea potrivita pentru explorarea reactiilor in lant ale neutronilor in dispozitivele de fisiune. In formularea lui von Neumann, a problemei difuziei neutronilor, istoria fiecarui neutron era analoaga cu un singur joc de Solitaire iar folosirea numerelor aleatorii, pentru a face alegeri de-a lungul traiectoriei acestora, era analoaga cu intoarcerea unei noi carti in timpul jocului. Astfel, pentru a realiza o simulare Monte Carlo era nevoie de un generator de numere aleatoare si de simulari ale unor variabile aleatoare, specifice fiecarui tip de decizie. Se stia din fizica ca un model bun pentru traiectoriile neutronilor intre coliziuni presupunea o distributie exponentiala. Pentru a simula o astfel de distributie von Neumann si Ulam au pornit de la o distributie uniforma pe intervalul $(0, 1)$ si au construit, intr-o serie de corespondenţe, ceea ce azi numim metodele inversării funcţiei de repartiţie si metoda respingerii.



THE INSTITUTE FOR ADVANCED STUDY
SCHOOL OF MATHEMATICS
PRINCETON, NEW JERSEY

May 21, 1947

Mr. Stan Ulam
Post Office Box 1663
Santa Fe
New Mexico

Dear Stan,

Thanks for your letter of the 19th. I need not tell you that Klari and I are looking forward to the trip and visit at Los Alamos this Summer. I have already received the necessary papers from Carson Mark. I filled out and returned mine yesterday; Klari's will follow to-

Ulterior, deoarece utilizarea simularilor Monte Carlo necesita un numar mare de numere aleatorii, s-a ajuns la dezvoltarea generatoarelor de numere pseudo-aleatorii. Este suprinzator cum natura poate genera cu usurinta evenimente aleatoare dar tot ceea ce omul creaza sfarseste prin a fi pseudo-aleator. Spre exemplu, dezintegrarea radioactiva survine aleator. S-ar putea, deci, folosi un contor de particule Geiger pentru a capta radiatii emise în urma dezintegrarii unui element radioactiv si utiliza valorile contorului pentru a genera numere aleatoare.

Astazi metodele Monte Carlo se folosesc pe scara larga, in studiul fenomenelor care implica o anumita incertitudine in desfasurarea lor, in domenii ca si grafica computerizata, inteligenta artificiala aplicata in studierea jocurilor, in operatiuni de cautare si salvare , in estimarea riscurilor (finante), programarea robotilor autonomi (localizarea Monte Carlo), procesarea semnalelor sau chiar in astrofizica (modelarea evolutiei unei galaxii).

Metode de simulare a variabilelor aleatoare

• majoritatea limbajelor de programare au functii predefinite de generare aleatoare (pseudo-aleatoare) a unor numere care de obicei apartin intervalelor $[0,1)$ sau $(0,1)$, pe care le vom nota cu **rand(0,1)** intr-un context general

• aceste functii au la baza de cele mai multe ori algoritmul Mersenne Twister (MT19937) sau batranul algoritm congruential liniar

$\Longrightarrow$ de amintit faptul ca J. von Neumann isi crease propriul generator care nu era insa suficient de performant

• plecand de la aceste functii vom construi algoritmi de simulare a valorilor unor variabile aleatoare clasice, in conformitate cu densitatile de probabilitate f sau functiile de repartitii F date

Metoda inversarii functiei de repartitie

Ideea generala: putem genera o variabila aleatoare X cu o functie de repartitie dorita F daca stim inversa sa F^{-1}, pornind de la o variabila uniform distribuita $U \sim \mathcal{U}(0,1)$, prin formula

$$X = F^{-1}(U)$$

deoarece pentru o variabila uniform distribuita avem $P(U \leq u) = u$ si astfel

$$P(X \leq x) = P(F^{-1}(U) \leq x) = P(U \leq F(x)) = F(x)$$

asadar X va avea functia de repartitie F.

• in cazul variabilelor aleatoare discrete

$$X : \begin{pmatrix} x_1 & x_2 & \dots & x_n \\ p_1 & p_2 & \dots & p_n \end{pmatrix}$$

functia de repartitie $F : \mathbb{R} \to [0,1]$, $F(x) = P(X \leq x)$, are urmatoarea forma

$$F(x) = \begin{cases} 0, & x < x_1 \\ p_1, & x_1 \leq x < x_2 \\ p_1 + p_2, & x_2 \leq x < x_3 \\ \quad \vdots & \\ p_1 + p_2 + \ldots + p_k & x_k \leq x < x_{k+1} \\ \quad \vdots & \\ 1 & x_n \leq x \end{cases}$$

• definim o "functie inversa" $F^{-1} : [0,1] \to \mathbb{R}$ prin formula

$$F^{-1}(u) = \inf\{x \in \mathbb{R} : F(x) \geq u\}$$

• daca generam $U \sim \mathcal{U}(0,1)$ pentru a obtine o variabila aleatoare cu functia de repartitie F, prin $X = F^{-1}(U)$, trebuie sa tinem cont de relatiile

$$X = \begin{cases} x_1, & u < p_1 \\ x_2, & p_1 \leq u < p_1 + p_2 \\ \vdots & \\ x_k, & p_1 + p_2 + \ldots + p_{k-1} \leq u < p_1 + p_2 + \ldots + p_{k-1} + p_k \\ \vdots & \\ x_n, & p_1 + p_2 + \ldots + p_{n-1} \leq u < 1 \end{cases}$$

• cu alte cuvinte, stim sa generam automat un numar aleator $u \in [0, 1)$ dar apoi trebuie sa il conectam la o valoare x corespunzatoare variabilei aleatoare X si vom realiza asta plecand de la faptul ca functia de repartitie F are domeniul de valori intervalul $[0, 1]$

• pentru a determina valoarea x, a lui X, trebuie sa identificam intervalul $[F(x_i), F(x_{i+1}))$ caruia ii apartine valoarea generata u

Algoritmul general

Input: functia de repartitie F
Output: o variabila X cu functia de repartitie F

Pas 1: $u := rand(0, 1)$ // genereaza o valoare uniform distribuita in $[0, 1)$
Pas 2: initializeaza $i = 1$ si $p = F(x_1)$
Pas 3: daca $u < p$ atunci seteaza $X := x_i$ si stop
Pas 4: incrementeaza $i \leftarrow i + 1$ si $p := F(x_i)$
Pas 5: repeta Pasul 3

• pentru o variabila continua X algoritmul este mai simplu, dar presupune cunoasterea expresiei lui F^{-1}

Algoritmul general

Input: inversa F^{-1} a functiei de repartitie
Output: valori ale unei variabile aleatoare X cu distributia data de F

Pas 1: $u := rand(0, 1)$
Pas 2: $x := F^{-1}(u)$
Pas 3: returneaza x

Metoda transformarilor

• foarte multe variabile aleatoare pot fi obtinute in functie de alte variabile aleatoare prin diferite transformari, de exemplu o variabila aleatoare normal distribuita $X \sim \mathcal{N}(m, \sigma^2)$ poate fi obtinuta dintr-una normal standard distribuita $Z \sim \mathcal{N}(0, 1)$

$$X = m + \sigma Z$$

• din acest motiv vom prezenta mai tarziu doar un procedeu de simulare a unei variabile Z normal standard distribuita (Box-Muller), cazul general fiind apoi obtinut prin adaugarea unei linii $x := m + \sigma * z$ in algoritm

• analog, putem obtine o variabila $X \sim \mathcal{U}(a,b)$ uniform distribuita pe $[a,b]$ prin transformarea

$$X = a + (b-a)Y$$

unde $Y \sim \mathcal{U}(0,1)$.

Metoda combinarii

• uneori o functie de repartitie F poate fi descompusa ca o suma de functii de repartitie

$$F(x) = \sum_{i=1}^{n} p_i \cdot G_i(x)$$

unde $p_i > 0$ si $\sum_{i=0}^{n} p_i = 1$

• o metoda de a simula o astfel de variabila presupune simularea unei variabile aleatoare Y cu proprietatea $p_i = P(Y = i)$ iar la pasul urmator, in functie de valoarea i obtinuta, simulam X considerand F_i ca fiind functia sa de repartitie

Algoritmul general

Input: functiile de repartitie G_i si un algoritm pentru simularea varaibilei Y
Output: valori ale unei variabile aleatoare X cu distributia data de F

Pas 1: genereaza o valoare i corespunzatoare variabilei Y
Pas 2: returneaza valoarea generata prin algoritmul de simulare al variabilei cu functia de repartitie G_i

Metoda convolutiei

• unele variabile aleatoare pot fi obtinute prin simpla insumare a unora clasice, de exemplu

$$X_1, X_2, \dots X_n \sim Ber(p) \text{ independente } \Longrightarrow Y = \sum_{i=1}^{n} X_i \sim Bin(n,p)$$

$$X_1, X_2, \dots X_n \sim Exp(\lambda) \text{ independente } \Longrightarrow Y = \sum_{i=1}^{n} X_i \sim Erlang(n,\lambda)$$

sau

$$X_1, X_2, \dots X_n \sim Geo(p) \text{ independente } \Longrightarrow Y = \sum_{i=1}^{n} X_i \sim NB(n,p)$$

• spunem ca distributia lui Y este convolutia distributiilor lui X_i, $i = \overline{1,n}$

Algoritmul general

Input: metoda de simulare a variabilei X, unde $X_i \sim X$, $i = \overline{1,n}$
Output: valori ale variabilei aleatoare Y

Pas 1: simuleaza valori ale variabilelor $X_1, X_2, \dots X_n$
Pas 2: insumeaza valorile si returneaza y

Metoda Box-Muller pentru simularea variabilelor normale

• cea mai cunoscuta metoda, numita metoda Box-Muller, presupune generarea unor variabile U_1, U_2 uniform distribuite cu valori in $(0,1)$ si apoi construirea unei bijectii

$$g : (0,1) \times (0,1) \to \mathbb{R}^2$$

astfel incat $(Y_1, Y_2) = g(U_1, U_2)$ sa fie normal distribuite

• din motive de diferentiabilitate Box si Muller au ales functia

$$g(u_1, u_2) = \left(\sqrt{-2\ln u_1}\cos(2\pi u_2), \sqrt{-2\ln u_1}\sin(2\pi u_2)\right)$$

• aceasta functie nu poate fi utilizata direct deoarece sin si cos creaza probleme de eficienţă algoritmului, prin urmare se prefera metoda polara Box-Muller

Algoritmul general

Input: doua variabile uniform distribuite U_1, U_2
Output: doua variabile normal standard distribuite Z_1, Z_2

Pas 1: simuleaza $U_1, U_2 \sim \mathcal{U}(0,1)$ generand u_1 si u_2
Pas 2: seteaza $v_1 := 2u_1 - 1$, $v_2 := 2u_2 - 1$ si $r := v_1^2 + v_2^2$
Pas 3: daca $r > 1$ repeta Pas 1
Pas 4: returneaza valorile z_1, z_2, corespunzatoarea unor variabile independente, normal standard distribuite

$$z_1 = \sqrt{\frac{-2\ln r}{r}}v_1 \quad \text{si} \quad z_2 = \sqrt{\frac{-2\ln r}{r}}v_2$$

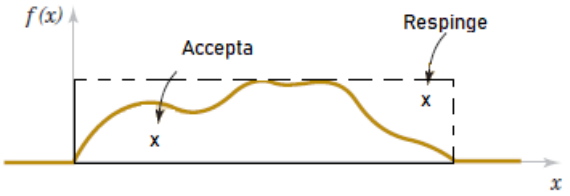
Metoda respingerii

• aflarea unei formule pentru inversa F_X^{-1} nu este intotdeauna posibila sau algoritmul propus nu este suficient de eficient, prin urmare propunem o metoda alternativa, construita de catre J. von Neumann

Ideea generala: daca stim densitatea de probabilitate $f(x)$ a unei variabile aleatoare X si aceasta densitate este nenula doar in intervalul $[a,b]$ iar valoarea maxima a lui f este c, atunci generam uniform puncte (x_i, y_i) din dreptunghiul $[a,b] \times [c,d]$ dupa care le respingem pe cele care nu convin dupa urmatorul algoritm

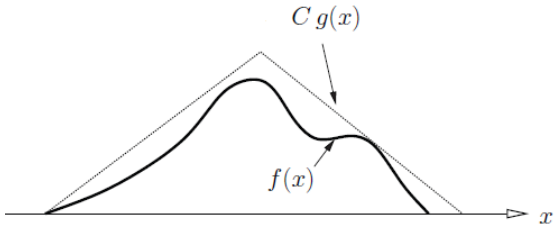
Algoritmul general

Pas 1: seteaza $x := a + (b-a) * rand(0,1)$ si $y := c * rand(0,1)$
// genereaza $x \in [a,b]$ si $y \in [0,c]$ uniform distribuite cu metoda
// transformarilor
Pas 2: daca $y \leq f(x)$ se accepta valoarea x
daca $y > f(x)$ se respinge valoarea x
Pas 3: incrementeaza contorul de numarare al incercarilor (pentru a putea controla procesul generarii) si revino la primul pas



• pentru cazul in care intervalul $[a, b]$ este nemarginit este nevoie de o metoda mai generala si anume presupunem ca f este majorata de o densitate de probabilitate g, care corespunde unei variabile aleatoare Y, pentru care avem deja un algoritm de simulare

$$f(x) \leq Cg(x), \qquad C \geq 1.$$



• un exemplu des intalnit este reprezentat de variabila aleatoare $X = |Z|$, unde $Z \sim \mathcal{N}(0, 1)$, si prin urmare are densitatea $f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$, care poate fi majorata pe intervalul $[0, \infty)$ printr-o densitate de probabilitate exponentiala $g(x) = e^{-x},\ x \geq 0$

• in cazul exponential putem folosi usor metoda inversarii functiei de repartitie caci $G(x) = 1 - e^{-x}$, pentru $x \geq 0$, si prin urmare

$$Y = G^{-1}(U) = -\ln(1 - U), \qquad U \in \mathcal{U}(0, 1)$$

• putem sa folosim aceeasi abordare pentru a simula o variabila normal standard distribuita $Z \sim \mathcal{N}(0, 1)$, prin intermediul metodei combinarii, datorita relatiei

$$F_Z(z) = \frac{1}{2} F_{-|Z|}(z) + \frac{1}{2} F_{|Z|}(z)$$

Algoritmul general

Input: densitatea de probabilitate g si o constanta C
Output: o variabila aleatoare X cu densitatea de probabilitate f

Pas 1: found$\leftarrow$ false
Pas 2: while not found do
genereaza x cu distributia Y
alege $u := C * g(x) * rand(0, 1)$
// genereaza u cu distributia uniforma pe $(0, Cg(x))$
daca $u \leq f(x)$ atunci found$\leftarrow$ true
Pas 3: returneaza x

• in cazul discret poate fi aplicata o strategie similara, acolo avand $p_i \leq Cq_i$, pentru doua variabile aleatoare X, Y, cu $p_i = P(X = i)$ si $q_i = P(Y = i)$

Simularea variabilelor in C++

• vom particulariza, pentru limbajul C++, cateva aspecte legate de cei doi piloni ai simularii variabilelor aleatoare: simularile variabilelor uniform distribuite si ale celor normal distribuite

• versiunea standard a C++ contine functia *rand* inclusa in libraria *cstdlib*, un apel al functiei *rand* returneaza o valoare intreaga intre 0 si o constanta numita $RAND\_MAX$, care de obicei este $2^{31} - 1$

• cel mai bine este sa definim propriile functii care sa genereze numere pseudo aleatoare si in principiu doua astfel de functii sunt utile: una care sa genereze numere naturale cuprinse 1 si un numar natural dat si una care sa genereze numere reale dintr-un interval fixat $[a, b]$

```
#ifndef   UNIFORM_H
#define   UNIFORM_H
double unif();
   /** genereaza un numar aleator intre 0 si 1
     *  returneaza o valoarea uniform distribuita din intervalul [0, 1]
     */
double unif(double a, double b);
   /** genereaza un numar real intr-un interval dat [a, b]
     */
long unif(long n);
   /** genereaza un numar natural intre 1 si o valoare data n
     * parametrul n reprezinta cea mai mare valoare pe care
     * o poate produce functia
     */
void seed();
   /** e nevoie sa resetam generatorul de numere pseudo-aleatoare
     */
  # endif
```

• procedura *rand* si celelalte care se bazeaza pe *rand* de mai sus returneaza un sir imprevizibil de valori, dar de fiecare data acelasi sir, din cauza ca se doreste a avea caracterul de reproductibilitate

• mai mult, valoarea de start, *seed*, este fixata

• in cazul nostru acest aspect este nedorit si putem evita elegant acest fenomen folosind *srand* si functia *time* (adaugam headerul *ctime*) care va returna o valoare *time(0)*, insemnand numarul de secunde scurse de la un moment de timp fixat (care depinde de computer)

• uneori compilatorul cere modificarea valorii timp returnata, intr-una fara semn, si atunci se poate inlocui ultima linie cu *srand(unsigned(time(0)));*

• in cele ce urmeaza este prezentat modul de definire al functiilor unif, putandu-se observa particularizarea pseudocodului de pe o pagina anterioara

```
#include "uniform.h"
#include <cstdlib>
#include <ctime>
#include <cmath>
using namespace std;

double unif() {
return rand() / double(RAND_MAX);
}

double unif(double a, double b) {
return (b-a)*unif() + a;
}

long unif(long a) {
if (a < 0) a = -a;
if (a==0) return 0;
return long(unif()*a) + 1;
}

void seed() {
srand(time(0));
 }
```

• in cele de urmeaza ne vom ocupa de implementarea metodei Box-Muller, un program C++ bazat pe metoda polara Box-Muller este prezentat mai jos

```
#include <cmath>
#include "uniform.h"
using namespace std;
double randn() {
double x,y,r;
do {
x = unif(-1.,1.);
y = unif(-1.,1.);
r = x*x + y*y;
} while (r >= 1.);
double mu = sqrt(-2.0 * log(r) / r);
return mu*x;
}
```

• header-ul *randn.h* arata in felul urmator

```
#ifndef RANDN_H
#define RANDN_H
#include "uniform.h"
double randn();
#endif
```

• o versiune completa poate arata in felul urmator

```
#include "randn.h"
#include <cmath>
using namespace std;

double randn() {
static bool has_saved = false;
static double saved;

if (has_saved) {
has_saved = false;
return saved;
}

double x,y,r;
do {
x = unif(-1.,1.);
y = unif(-1.,1.);
r = x*x + y*y;
} while (r >= 1.);

double mu = sqrt(-2.0 * log(r) / r);

saved = mu*y;
has_saved = true;

return mu*x;
}
```

• prin modificarea facuta, progamul este de doua ori mai rapid decat cel anterior care avea o problema de eficienta, deoarece genera doua variabile aleatoare normal distribuite Z_1 si Z_2

• noua versiune contine doua variabile statice: una booleana *has_saved* si una reala *saved*

• variabila *has_saved* este presetata ca fiind falsa, pentru a arata ca procedura nu contine in acel moment o valoare normal distribuita salvata, apoi se verifica daca exista una care nu a fost folosita

• apoi se genereaza doua variabile normal distribuite prin metoda Box-Muller polara si vom salva z_2 in *saved* si setam *has_saved* ca fiind *true* si returnam z_1

• prin salvarea lui z_2 partea inceata a algoritmului ruleaza doar o singura data, dubland viteza

Vectori aleatori

• pentru a analiza anumite experimente aleatorii este nevoie de doua variabile aleatoare X si Y, vezi primele doua probleme rezolvate, si vom numi perechea (X, Y) vector aleator

• in cazul discret, functia de repartitie comuna perechii X,Y este definita prin

$$F_{X,Y}(x,y) = P\left(X \leq x,\ Y \leq y\right)$$

• probabilitatea unui eveniment $(X,Y) = (x,y)$ se calculeaza prin

$$P_{X,Y}(x,y) = P\left(X = x,\ Y = y\right)$$

folosind un tabel de repartitie comuna, la fel ca in fisa seminarului 9

• insa in cazul unor variabile X, Y independente avem formula

$$P_{X,Y}(x,y) = P\left(X = x,\ Y = y\right) = P(X = x) \cdot P(Y = y)$$

• de retinut ca $\{X = x,\ Y = y\}$ reprezinta acum un eveniment al unui experiment iar multimea starilor devine

$$S_{X,Y} = \{(x,y) : \quad P_{X,Y}(x,y) > 0\}$$

• probabilitatea unui eveniment E va fi

$$P(E) = \sum_{(x,y)\in E} P_{X,Y}(x,y)$$

iar functiile de probabilitate marginale sunt

$$P_X(x) = \sum_{y\in S_Y} P_{X,Y}(x,y), \qquad P_Y(y) = \sum_{x\in S_X} P_{X,Y}(x,y)$$

aflandu-se prin adunarea pe linie sau coloana a valorilor din tabelul de repartitie comun, vezi fisa seminarului 9

• in cazul unui vector aleator (X,Y) continuu este nevoie sa manevram integrale duble, vezi fisa seminar daca e necesar

• densitatea comuna de probabilitate $f_{X,Y}(x,y)$ va avea proprietatile caracteristice

$$f_{X,Y}(x,y) \geq 0 \ \text{ si } \ \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f_{X,Y}(x,y)\ dxdy = 1$$

iar functia de repartitie comuna este definita prin

$$F_{X,Y}(x,y) = \int_{-\infty}^{x}\int_{-\infty}^{y} f_{X,Y}(u,v)dudv$$

in consecinta

$$P(a \leq X \leq b, \quad c \leq Y \leq d) = F_{X.Y}(b,d) - F_{X.Y}(b,c) - F_{X.Y}(a,d) + F_{X.Y}(a,c)$$

• pentru variabile aleatoare independente avem relatia

$$f_{X,Y}(x,y) = f_X(x) \cdot f_Y(y)$$

• probabilitatea unui eveniment E se calculeaza prin

$$P(E) = \iint\limits_{E} f_{X,Y}(x,y)\ dxdy$$

• densitatile de probabilitate marginale sunt obtinute din

$$f_X(x) = \int_{-\infty}^{\infty} f_{X,Y}(x,y)dy, \qquad f_Y(y) = \int_{-\infty}^{\infty} f_{X,Y}(x,y)dx$$

Probleme rezolvate

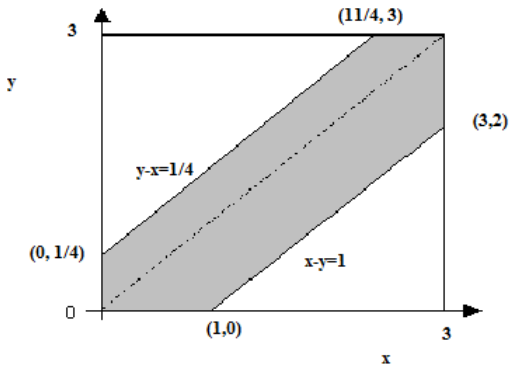
Problema 1. ***(Problema intalnirii revizitata)***
Un barbat si o femeie decid sa se intalneasca intr-un restaurant dupa ora 21. Restaurantul se inchide la ora 24. Din cauza programului incarcat al fiecaruia ei decid ca in cazul in care unul dintre ei va intarzia fiecare sa astepte dupa celalalt un anumit timp. Barbatul este dispus sa astepte o ora iar femeia doar 15 minute! Care este probabilitatea ca cei doi sa se intalneasca la acel restaurant?

Solutie: Notam cu X variabila aleatoare continua care masoara timpul la care soseste femeia (fara vreo aluzie cromozomiala) si cu Y variabila aleatoare care masoara timpul la care soseste barbatul. Fiecare poate ajunge la restaurant in orice moment de timp intre ora 21 si 24 cu aceeasi probabilitate, deci X si Y au distributia uniform continua. Mai mult, cele doua variabile sunt independente si putem sa consideram ca ambele iau valori in intervalul $[0,3]$ unde 0 inseamna ora 21 si 3 inseamna ora 24.

Avem asadar un vector aleator (X,Y) care modeleaza matematic problema si masoara timpii la care sosesc cei doi. Notam cu E evenimentul: cei doi se intalnesc in restaurant. Probabilitatea lui E se calculeaza conform formulei

$$P(E) = \iint\limits_E f_{X,Y}(x,y)\, dxdy$$

unde E este regiunea gri din desenul de mai jos.



Deoarece X si Y sunt independente densitatile de probabilitate sunt in relatia

$$f_{X,Y}(x,y) = f_X(x) \cdot f_Y(y)$$

unde

$$f_X(x) = \begin{cases} \frac{1}{3-0}, & x \in [0,3] \\ 0, & \text{altfel} \end{cases} \qquad \text{si} \qquad f_Y(y) = \begin{cases} \frac{1}{3-0}, & y \in [0,3] \\ 0, & \text{altfel} \end{cases}$$

sunt densitatile de probabilitate ale celor doua variabile aleatoare continue uniform distribuite. Asadar

$$P(E) = \iint\limits_E \frac{1}{9}\, dxdy.$$

Pentru a calcula aceasta integrala dubla notam cu Δ_1 si Δ_2 cele doua trinughiuri albe din desenul anterior. Se observa ca

$$\Delta_1 \cup A \cup \Delta_2 = [0,3] \times [0,3]$$

Prin urmare

$$\iint\limits_{[0,3]\times[0,3]} \frac{1}{9}\, dxdy = \iint\limits_{\Delta_1} \frac{1}{9}\, dxdy + \iint\limits_{A} \frac{1}{9}\, dxdy + \iint\limits_{\Delta_2} \frac{1}{9}\, dxdy.$$

Aplicam teorema lui Fubini si scriem cele doua triunghiuri ca domenii simple in raport cu axa OY

$$\int_0^3 \int_0^3 \frac{1}{9}\, dxdy = \int_0^{\frac{11}{4}} \int_{x+\frac{1}{4}}^3 \frac{1}{9}\, dydx + \iint\limits_A \frac{1}{9}\, dxdy + \int_1^3 \int_0^{x-1} \frac{1}{9}\, dydx$$

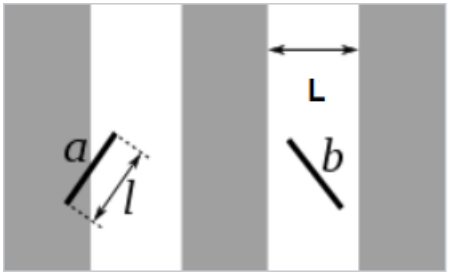
$$\iff 1 = \frac{121}{288} + \iint\limits_A \frac{1}{9}\, dxdy + \frac{4}{18}$$

$$\implies P(E) = \iint\limits_A \frac{1}{9}\, dxdy = 0.35 = 35\%$$

In final, ar fi indicata o comparare cu abordarea prezentata in introducerea fisei seminarului 8. Pe parcursul solutiei, in evaluarea unor integrale am preferat o abordare tipica integralelor duble (pentru antrenament), evitand investigarea mult mai naturala a ariilor.

Problema 2. ***(Problema acului lui Buffon)***
O podea are linii paralele situate la distanta L una de cealalta. Un ac de lungime ℓ este aruncat aleator pe podeaua. Gasiti probabilitatea ca acul sa intersecteze una dintre linii.

Solutie: Problema are o istorie aparte, fiind propusa de contele de Buffon in secolul al XVIII-lea. Poate fi interpretata ca descriind o metoda Monte Carlo (aruncari repetate ale acului pe podea) pentru aproximarea numarului π, deoarece solutia problemei este $p = \frac{2}{\pi}\frac{\ell}{L}$.



Pentru a simplifica lucrurile vom trata doar cazul in care lungimea acului este mai mica decat distanta dintre liniile podelei $\ell < L$.

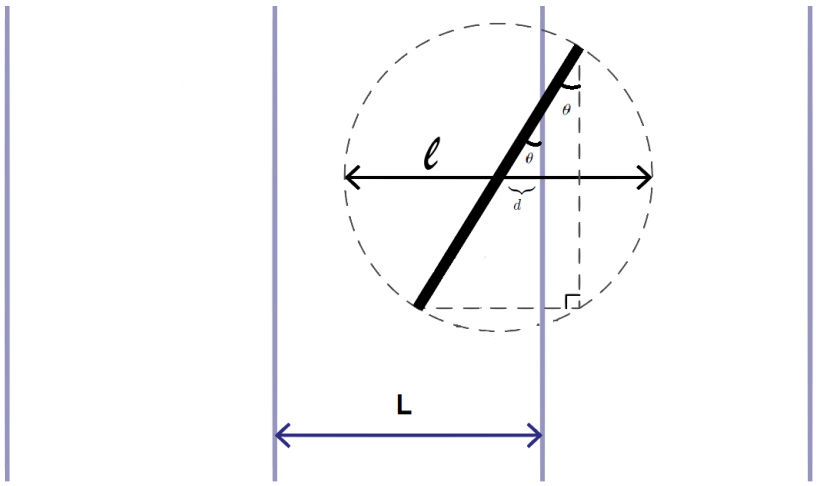
Inainte de toate trebuie remarcat faptul ca unghiul ascutit θ, pe care acul il formeaza cu directia descrisa de liniile podelei, influenteaza sansa ca acul sa atinga o linie. Un ac care cade paralel cu liniile nu va atinge prea des liniile (decat daca ajunge exact pe una dintre linii), indiferent de lungimea sa ℓ.

Se poate observa ca nu este suficient sa cunoastem unghiul θ format de ac si linii pentru a preciza pozitia acului si implicit a modela matemcatic problema, dupa cum decurge si din desen. Pozitia finala a acului depinde si de distanta dintre ac si oricare dintre liniile podelei. Vom considera pozitia fata de cea mai apropiata linie (dupa ce cade pe podea). Putem estima aceasta distanta in multe moduri, dar cea mai eleganta cale pare a fi sa masuram distanta de la mijlocul acului la aceasta cea mai apropiata linie. Notam cu d aceasta distanta este evident ca d poate avea valori intre 0 (cand mijlocul acului se afla pe linie) si $\frac{L}{2}$ (cand mijlocul acului se afla exact la mijlocul distantei dintre doua linii). Orice valoare din intervalul $[0, \frac{L}{2}]$ poate fi atinsa de catre d cu aceasi probabilitate, prin urmare d va fi o valoare a unei variabile aleatoare D cu distributie uniforma pe $[0, \frac{L}{2}]$. Si in cazul unghiului θ se poate observa ca valoarea minima este 0 iar valoarea maxima este $\frac{\pi}{2}$ si din nou sansa ca acul sa formeze oricare unghi din acest interval este aceeasi, deci θ este o valoare a unei variabile aleatoare Θ uniform distribuite pe $[0, \frac{\pi}{2}]$.

Asadar folosim vectorul aleator (D, Θ) pentru a modela matematic caderea unui ac pe podea. Prin definitie densitatile de probabilitate ale celor doua variabile uniform distribuite sunt

$$f_D(d) = \begin{cases} \frac{1}{\frac{L}{2}} & 0 \le d \le \frac{L}{2} \\ 0 & \text{in rest} \end{cases} \quad \text{si} \quad f_\Theta(\theta) = \begin{cases} \frac{1}{\frac{\pi}{2}} & 0 \le \theta \le \frac{\pi}{2} \\ 0 & \text{in rest} \end{cases}$$

Definim evenimentul E: acul intersecteaza una dintre liniile podelei. Este recomandat sa desenam aceasta situatie pentru a observa ca evenimentul are loc doar daca $d \le \frac{\ell}{2} \sin\theta$



In desenul de mai sus este descrisa situatia in care acul intersecteaza cea mai apropiata linie. Se formeaza doua triunghiuri dreptunghice cu un unghi θ si

cateta opusa d, respectiv $\frac{\ell}{2}\sin\theta$. Pentru a intersecta linia respectiva trebuie sa avem relatia $d \leq \frac{\ell}{2}\sin\theta$ altfel linia albastra se afla in afara zonei cu care acul poate avea puncte comune. Prin urmare putem defini evenimentul E ca fiind

$$E = \left\{ (d,\theta) \in \mathbb{R}^2 \ : \ d \leq \frac{\ell}{2}\sin\theta \right\}$$

si conform teoriei prezentate in fisa

$$P(E) = \iint_E f_{D,\Theta}(d,\theta)\ dddd\theta.$$

Din cauza notatiei alese apare neplacuta asociere dd, insemnand integrare in raport cu d :)). Deoarece variabilele D si Θ sunt independente

$$f_{D,\Theta}(d,\theta) = f_D(d)\cdot f_\Theta(\theta) = \begin{cases} \frac{4}{L\pi}, & 0 \leq d \leq \frac{L}{2}, \quad 0 \leq \theta \leq \frac{\pi}{2} \\ 0, & \text{in rest} \end{cases}$$

Pentru a evalua integrala dubla de mai sus vom scrie multimea E ca un domeniu simplu in raport cu axele

$$E = \left\{ (d,\theta) \in \mathbb{R}^2 \ : 0 \leq \theta \leq \frac{\pi}{2}, \quad 0 \leq \ d \leq \frac{\ell}{2}\sin\theta \right\}$$

prin urmare

$$P(E) = \iint_E f_{D,\Theta}(d,\theta)\ dddd\theta = \int_0^{\frac{\pi}{2}} \left(\int_0^{\frac{\ell}{2}\sin\theta} \frac{4}{L\pi}\ dd \right) d\theta$$
$$= \int_0^{\frac{\pi}{2}} \frac{4}{L\pi}\frac{\ell}{2}\sin\theta\ d\theta = -\frac{4}{L\pi}\frac{\ell}{2}\cos\theta\Big|_0^{\frac{\pi}{2}} = \frac{2\ell}{\pi L}$$

Problema 3. *Scrieti un algoritm (pseudocod) pentru a simula o variabila aleatoare binomiala $X \sim Bin(n,p)$.*

Solutie: Functia de probabilitate a unei variabile aleatoare $X \sim Bin(n,p)$ este $f(k) = C_n^k p^k (1-p)^{n-k}$, $k = \overline{1,n}$. Este important sa remarcam recurenta $f(k+1) = \frac{n-k}{k+1}\frac{p}{1-p}f(k)$. Vom crea un algoritm bazat pe metoda inversarii functiei de repartitie:

Algoritm

Pas 1: citeste n si p si seteaza $c := \frac{p}{1-p}$, $i := 0$, $prob := (1-p)^n$ si $F := prob$
Pas 2: genereaza $u := rand(0,1)$
Pas 3: daca $u < F$ seteaza $x := i$ si stop
Pas 4: incrementeaza prob←prob*c*(n-i)/(i+1) apoi F← F+prob
si $i \leftarrow i+1$
Pas 5: repeta Pas 3

Problema 4. *Scrieti un algoritm (pseudocod) pentru a simula o variabila aleatoare cu o distributie beta $X \sim \beta(\mu,\nu)$, pentru $\mu,\nu < 1$. Simulati o variabila aleatoare beta $X \sim \beta(2,2)$ prin metoda respingerii.*

Solutie: Reamintim ca $f(x) = \frac{x^{\mu}(1-x)^{\nu}}{\beta(\mu,\nu)}$, pentru $x \in [0,1]$, este densitatea de probabilitate corespunzatoare lui $X \sim \beta(\mu,\nu)$. In general algoritmii de simulare a valorilor lui X se bazeaza pe diverse estimari ale lui f si metoda respingerii. In cazul $\mu, \nu < 1$ putem incerca urmatorul algoritm

Algoritm

Pas 1: genereaza $u := rand(0,1)$ si $v := rand(0,1)$
Pas 2: seteaza $x := u^{1/\mu}$ si $y := v^{1/b}$
Pas 3: daca $x + y > 1$ repeta Pas 1, altfel returneaza $\frac{x}{x+y}$ ca valoare a lui X

Pentru o variabila X cu distributie $\beta(2,2)$, densitatea de probabilitate este $f(x) = 6x(1-x)$, pentru $0 \leq x \leq 1$ si $f(x) = 0$ in rest. Deoarece are loc inegalitatea $f(x) \leq 6 \cdot 1$, pentru $x \in (0,1)$, putem alege $g(x) = 1$ care este densitatea de probabilitate a unei variabile uniform distribuite $Y \sim \mathcal{U}(0,1)$, si constanta $C = 6$. Putem crea acum usor un algoritm bazat pe metoda respingerii.

Probleme propuse

Problema 1. *Functia de probabilitate comuna a variabilelor aleatoare X si Y este data in tabel*

(a) Aflati functiile probabiliste marginale ale lui X, respectiv Y.
(b) Aflati $P(1 \leq X < 3, Y \geq 1)$.
(c) Determinati daca X si Y sunt independente

X \ Y	0	1	2
0	1/18	1/9	1/6
1	1/9	1/18	1/9
2	1/6	1/6	1/18

Problema 2. *Densitatea de probabilitate comuna a doua variabile aleatoare X si Y este*

$$f(x,y) = \begin{cases} cxy, & 0 < x < 4, 1 < y < 5 \\ 0, & \text{in rest} \end{cases}$$

a) Aflati valoarea lui c
b) Aflati $P(1 < X < 2, 2 < Y < 3)$
c) Aflati densitatea de probabilitate marginala a lui X, respectiv Y

Problema 3. *Scrieti un algoritm de simulare (pseudocod) a unei variabile aleatoare Poisson prin metoda inversarii functiei de repartitie.*

Problema 4. *Folositi metoda convolutiei pentru a scrie un algoritm de simulare a unei variabile aleatoare $X \sim Erlang(n, \lambda)$. Faceti acelasi lucru apoi pentru o variabila aleatoare cu distributie negativ binomiala $Y \sim NB(n, p)$*

Problema 5. *Variabilele continue X, Y au densitatea de probabilitate comuna*

$$f_{X,Y}(x,y) = \begin{cases} ce^{-2x-3y}, & x+y \leq 1,\ x \geq 0,\ y \geq 0 \\ 0, & \text{in rest} \end{cases}$$

i) Aflati valoarea lui c

ii) Evaluati probabilitatea $P(X \leq Y)$

iii) Calculati probabilitatea $P(X + Y \leq \frac{1}{2})$

Problema 6. *Aruncam o moneda pana in momentul in care se obtine de doua ori pajura. Notam cu X_1 numarul de aruncari pana la prima aparitie a pajurei si cu X_2 numarul de aruncari aditionale necesare pana la a doua aparitie a pajurei. Consideram variabila aleatoare $Y = X_1 - X_2$. Aflati $M(Y)$ si $D^2(Y)$.*

Problema 7. *Scrieti un algoritm de generare a unor puncte uniform distribuite in discul eliptic*

$$D = \left\{ (x,y) \in \mathbb{R}^2 : \ \frac{x^2}{4} + \frac{y^2}{9} \leq 1 \right\}.$$

Bibliografie

[1] R. Eckhardt. *Stam Ulam, John von Neumann, and the Monte Carlo Method*, Los Alamos Science, Special Issue, 1987.

[2] N. Metropolis. *The Beginning of the Monte Carlo Method*, Los Alamos Science, Special Issue, 1987.

[3] R. Negrea. *Note de curs MS*, 2020.

[4] S. Ross. *Simulation*, Academic Press, 2013.

[5] R. Rubinstein and D. Kroese. *Simulation and the Monte Carlo Method*, Wiley&Sons, 2017.

[6] E. Scheinerman. *C++ for Mathematicians*, Chapmann & Hall/CRC, 2006.