GENERARE NUMERE ALEATOARE UNIFORM DISTRIBUITE

Numerele aleatoare uniform distribuite pe un interval sint numere care au aceiasi probalitate de aparitie pentru intregul interval. Densitatea lor de repartitie pe intervalul (a,b) este:

$$f(x) = \begin{cases} k & daca \quad x \in (a,b) \\ 0 & daca \quad x \notin (a,b) \end{cases} \quad unde \quad k = \frac{1}{(b-a)}$$

Exista 3 metode de generare:

1. Tabele de numere aleatoare
2. Generatoare de numere aleatoare prin procedee fizice
3. Numere pseudo-aleatoare

1. Tabele de numere aleatoare

Se amesteca cifrele 0,1,2,...,8,9 si se extrage cite o cifra, se noteaza cifra apoi se reintroduce cifra si se amesteca din nou. Urmeaza o noua extractie pina cind rezulta 5 cifre care alcatuiesc un numar. Se repeta pentru urmatorul numar rezultind in final tabele cu numere aleatoare care se pot introduce in memoria calculatorului.

Cea mai completa tabela are 1.000.000 de numere.

Problema este amestecul cifrelor. Se poate folosi o ruleta electrica care contine cifrele 0, 1,..., 9. Distributia ideala este:

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \end{pmatrix}$$

Numerele generate se supun testelor. Un exeplu de test simplu: N fiind numarul total de numere, se calculeaza V_0 = numarul de cifre 0 din toate numerele, V_1 = numarul de cifre 1 s.a.m.d. La sfirsit se calculeaza deviatia care este egala cu suma intervalelor:

$(V_i - 0.1*N)^2$ si care nu trebuie sa fie prea mare.

2.Generatoare de numere aleatoare prin procedee fizice

Se utilizeaza ca generatoare de numere aleatoare semnalele de zgomot. Daca numarul de semnale ce depasesc un anumit nivel intr-un interval de timp *t* este par rezulta cifra 0. Daca este impar rezulta cifra 1. Se folosesc simultan mai multe asemenea generatoare rezultind intr-un ciclu de masuratori un numar binar de *m* biti. Numarul aleator este :

$$0,X_{(1)}\,X_{(2)}\,....\,X_{(i)}$$

cu $X_{(i)}$ avand distributia $\begin{pmatrix} 0 & 1 \\ 0.5 & 0.5 \end{pmatrix}$

Problema este ca numerele generate sint greu de verificat pentru ca nu sint retinute, la un moment dat fiind disponibil un singur numar. Daca numerele sunt retinute rezulta metoda 1.

3.Numere pseudo-aleatoare

Se genereaza numere aleatoare bazate pe o formula si se verifica corectitudinea lor prin teste speciale.

Numerele calculate pe baza unei formule si care simuleaza o variabila aleatoare *X* se numesc *NUMERE PSEUDO-ALEATOARE*.

Prin *SIMULEAZA* se intelege ca numerele satisfac un set de teste asociat variabilei aleatoare.

Relatia de recurenta dintre doua numere pseudo-aleatoare se exprima astfel:

$X_{k+1} = F(X_k)$ unde F = set de transformari aplicate lui X_k

Cerintele unui generatoar de numere pseudo-aleatoare sint :

- sa fie simplu si rapid
- sa produca siruri lungi de numere care sa nu contina repetitii (perioada numerelor sa fie cit mai lunga)
- numerele sa fie cit mai putin corelate
- repartitia sa fie uniforma, sa satisfaca testele de concordanta(χ^2 , Kolmogorov).

3.1 Metoda partii de mijloc a patratului
(Mid Square Method, J. Von Neuman)

Exemplu 1: pentru numere subunitare cu 4 zecimale

$X_0 = 0.9876$

$X_0^2 = 0.97535376 \Rightarrow X_1 = 0.5353$

$X_1 = 0.28654609 \Rightarrow X_2 = 0.6546$

Exemplu 2: pentru numere supraunitare cu 4 cifre

$X_0 = 9876$

$X_0^2 = 97535376 \Rightarrow X_1 = 5353$

$X_1 = 28654609 \Rightarrow X_2 = 6546$

Formula de recurenta pentru exemplul 2 este :

$X_{k+1} = int(X_k^2 / b^a) - int(X_k^2 / b^{3a}) * b^{2a}$ unde a = 2, b = 10

sau

$X_{k+1} = int(X_k^2 / b^a) (mod\ b^{2a})$

Problema este repetabilitatea unor numere:

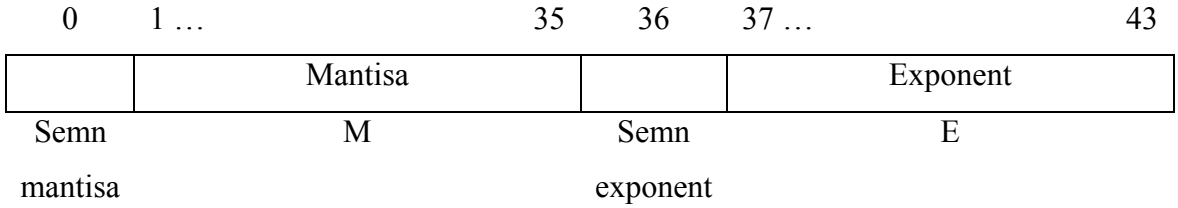
$3792^2 = 14379264$

Tema 1.1 : Sa se realizeze un program care implementeaza metoda partii de mijloc a patratului pentru numere subunitare cu 6 zecimale. Programul va genera 10.000 de numere care le va salva intr-un fisier.

Tema 1.2 : Sa se realizeze un program care implementeaza metoda partii de mijloc a patratului pentru numere supraunitare cu 6 cifre. Programul va genera 10.000 de numere care le va salva intr-un fisier.

3.2 Algoritmul Strela

Se utilizeaza pe calculatoare cu virgula flotanta cu 43 digiti binari



$X = M * 2^E \qquad 0.5 \le M < 1$

$X_{k+1} = F(X_k)$ prin trei operatii:

- X_k se inmulteste cu o constanta mare, uzual 1017
- $X_k * 10$ se deplaseaza (shift) cu 7 pozitii la stinga

- Valoarea absoluta obtinuta = X_{k+1}

Exemplu :

$X_0 = 1$ rezulta 80 000 numere diferite dupa care se repeta.

Avantaj: program scurt si rapid.

Knuth a construit un generator de numere super-aleatoare care spre surprinderea lui genera acelasi numar (algoritmul K, vezi paragraful 3.7). Concluzia lui a fost ca metodele de generare a numerelor aleatoare trebuie sa se bazeze pe teorii riguroase.

3.3 Metode congruentiale

liniare $X_{n+1} = (a * X_n + c)(\text{mod } M)$ (1)

X_0 = termen initial (mod M) = modulo M

a, c, M > 0 si intregi

Sirul de numere generat de relatia (1) = sir congruential liniar si se noteaza cu (X_0 , a, c, M)

Relatia (1) desemneaza generator *MIXT CONGRUENTIAL*

Daca c = 0 rezulta generator *MULTIPLICATIV CONGRUENTIAL*

Daca $X_{n+1} = X_n + X_{n+k}$ (mod M) rezulta generator *ADITIV CONGRUENTIAL*

Obs. M va fi ales cit mai mare si cit mai apropiat de marimea unui cuvint din calculator. Alegerea lui a, c se face astfel incit perioada sa fie maxima

c este prim cu M

b = a - 1 este multiplu de p , p = div M (divisor prim)

b este multiplu de 4 daca M este multiplu de 4

Exemplu : c = 1 $M = p^e$ => $a = p^{k+1}$ $k \in [2,e]$

Generatorul MULTIPLICATIV CONGRUENTIAL

$X_{n+1} = a * X_n \pmod M$ (2)

Pentru a avea peroiada maxima trebuiesc indeplinite cerintele:

- X_0 relativ prim cu M
- a radacina primitiva modulo M, pentru $M=2^e$, e>3 rezulta a = 3 sau 5 modulo 8

Pentru sistemele de calcul cu cuvint mic (L < 32) M <= 2^{31} -1 s-a constatat ca *a* este aproximativ sqrt(*M*).

Daca $a = a_1 * a_2 ... * a_k$ a_i intregi $1 \le i \le k$

cu proprietatea $a_i * X \le 2^{31} - 1$

atunci pentru $0 < X_n < M$ numarul

$X_{n+1} = a * X_n$

se obtine din relatiile:

$X_n^{(1)} = a_1 * X_n \pmod M$

$X_n^{(i+1)} = a_{i+1} * X_n^{(i)} \pmod M \quad 1 \le i \le k-2$ (3)

$X_{n+1} = a_k * X_n^{(k-1)} \pmod M$

Generatorul definit de relatiile (3) se noteaza cu:

$(X_0 , (a_1, a_2 ...a_k), 0, M)$

Exemple:

$(X_0 , (51, 57), 0 , 2^{25})$ $M = 2^{25} = 33554432$

$a = 2907 = 51 * 57 = 3 \pmod 8$

$(X_0 , (25, 25), 0, 76108864)$

Tema 2.1 : Sa se realizeze un program care implementeaza metoda congruentiala pentru numere subunitare. Programul va genera 10.000 de numere care le va salva intr-un fisier. Se va considera $M = 2^{31}$, iar pentru *a* si *c* vor fi date valori din program.

Atentie! Metoda congruentiala genereaza numere intregi in intervalul [0 , M-1], in consecinta pentru a putea obtine numere subunitare ar trebui sa impartim rezultatul la M-1.

Tema 2.2 : Sa se realizeze un program care implementeaza metoda congruentiala pentru numere supraunitare. Programul va genera 10.000 de numere care le va salva intr-un fisier. Se va considera $M = 2^{31}$, iar pentru *a* si *c* vor fi date valori din program.

3.4 Metode combinate

Se combina doi sau mai multi generatori pentru a obtine numere cit mai aleatoare. MacLaren si Marsaglia au combinat 2 generatori G_1 si G_2. Se genereaza cu G_1 un set de n numere din care se alege numarul corespunzator ca si ordine cu numarul generat cu G_2. Numarul selectat se inlocuieste cu unul nou generat cu G_1 (algoritmul SUPER0).

Algoritmul SUPER0.

Pas 0 - Initializari: se genereaza cu G_1, k numere X_i pseudo-aleatoare uniforme pe [0,1] si $T[i]=X_i$ pentru $1 \leq i \leq k$

Pas 1 - Se genereaza cu G_2 un numar pseudo-aleator Y pe (0,1)

Pas 2 - Se calculeaza j= 1 + int(k*Y) indice aleator

Pas 3 - Se gen. cu G_1 un nou X

Pas 4 - U=T[j], T[j]=X

Rezulta un numar aleator U. repetind pasii 1-4 rezulta sir de numere "destul de aleatoare".

Algoritmul este destul de bun pt. k=128. Conditie pt. G_1 si G_2 este sa fie generatori multiplicativi congruentiali si a_1, a_2 diferiti de X_0 ,Y_0 .

Tema 3.1 : Sa se realizeze un program care implementeaza algoritmul SUPER0. In locul generatorilor G_1 si G_2 se vor folosi 2 fisiere cu numere aleatoare obtinute cu ajutorul programului de la tema 2.1.

Algoritmul SUPER1.

Combinatie de trei generatori multiplicativi congruentiali G_1, G_2, G_3 cu a_1, a_2, a_3 numere impare si diferite.

Pas 0 - Initializare; se genereaza cu G_3, *i* numere pseudo-aleatoare si se incarca in *N[i]*
$1 <= i <= 128$

Pas 1 - Genereaza cu G_1 numarul L

Pas 2 - Genereaza cu G_2 numarul M

Pas 3 - Calculeaza $J = 1 + int(L / 2^{31} * 128)$

Pas 4 - Calculeaza $U = 0.5 + (N[J] + L + M) / 2^{31}$

Pas 5 - Genereaza K cu G3

Pas 6 - N[J]=K

Se repeta pasii 1-6 si se obtine astfel la fiecare repetare un numar aleator U.

Exemplu:

$G_1 = (X_0^{(1)} , 65539, 0, 2^{31})$

$G_2 = (X_0^{(2)} , 33554433, 0, 2^{31})$

$G_3 = (X_0^{(3)} , 362436069, 0, 2^{31})$

$i = 64$

Cu generatorul SUPER1 se obtin surse sigure de numere aleatoare.

Tema 3.2 : Sa se realizeze un program care implementeaza algoritmul SUPER1. In locul generatorilor G_1, G_2 si G_3 se vor folosi 3 fisiere cu numere aleatoare obtinute cu ajutorul programului de la tema 2.2.

3.5 Algoritmul K (Knuth)
generator de numere super-aleatoare

Transforma un numar dat X din zece cifre zecimale in numarul care ar trebui sa-l urmeze intr-un sir aleator. Pasii algoritmului sint:

K1 -Alege numarul de iteratii: Y=int($X/10^9$) cea mai semnificativa cifra a lui X (se vor executa pasii K2-K13 de Y+1 ori)

K2 -Alege pasul aleator: Z=int($X/10^8$)(mod 10) a doua cifra a lui X se trece la pasul k(3+Z)

K3 -Daca $X < 5*10^9$atunci $X = X + 5*10^9$

K4 -Mijloc patrat: $X = int(X^2 / 10^5)(mod\ 10^{10})$

K5 -Inmultire: $X = (1001001001 * X)(mod\ 10^{10})$

K6 -Pseudocomplementat: Daca $x < 10^8$ atunci $X = X + 9814055677$

altfel $X = 10^{10} - X$

K7 -Interschimba ultimele 5 cifre cu primele 5

$X = 10^5 * (X \bmod 10^5) + int(X/10^5)$

K8 -Inmultire: idem pas K5

K9 -Micsorare cifre: se scade o unitate din fiecare cifra#0 din reprezentarea zecimala a lui X

K10 -Modificare cu 99999: Daca $X < 10^5$ atunci $X = X^2 + 99999$

altfel $X = X - 99999$

K11 -Normalizare: in cadrul acestui pas X trebuie sa fie diferit de 0

Cit timp $X < 10^9$ cilceaza $X = 10 * X$

K12 -Mijlocul patratului modificat: $X = int(X*(X-1)/10^5)(mod\ 10^{10})$

K13 -Daca $Y > 0$ atunci $Y = Y - 1$ si se trece la pasul K2

altfel sfirsit algoritm

Stupoare: X=6065038420 se transforma in el insusi. In tabelul de mai jos sunt afisate valorile lui X dupa fiecare pas.

Pentru alt numar initial sirul incepe sa se repete dupa 7401 valori cu o perioada a ciclului de lungime 3178.

Pas	X (dupa)		Pas	X (dupa)	
K1	6065038420		K9	1107855700	
K3	6065038420		K10	1107755701	
K4	6910360760		K11	1107755701	
K5	8031120760		K12	1226919902	Y=3
K6	1968879240		K5	0048821902	
K7	7924019688		K6	9862877579	
K8	9631707688		K7	7757998628	
K9	8520606577		K8	2384626628	
K10	8520506578		K9	1273515517	
K11	8520506578		K10	1273415518	
K12	0323372207	Y=6	K11	1273415518	
K6	9676627793		K12	5870802097	Y=2
K7	2779396766		K11	5870802097	
K8	4942162766		K12	3172562687	Y=1
K9	3831051655		K4	1540029446	
K10	3830951656		K5	7015475446	
K11	3830951656		K6	2984524554	
K12	1905867781	Y=5	K7	2455429845	
K12	3319967479		K8	2730274845	
K6	6680032521		K9	1620163734	
K7	3252166800		K10	1620063735	
K8	2218966800		K11	1620063735	
			K12	6065038420	Y=0

Tema 4 : Sa se realizeze un program care implementeaza algoritmul K(Knuth). Programul va genera 10.000 de numere care le va salva intr-un fisier.