Universitatea din Bucureşti

Facultatea de Matematică şi Informatică

Modele de simulare

Autor **Prof. Dr. Ion Văduva**

BUCUREŞTI 2004

Prefaţă

Lucrarea constituie suportul de curs pentru disciplina **Modele de simulare** predată studenţilor de la secţiile de *Informatică* şi *Matematică-Informatică* din universităţi, ca disciplină obligatorie.

Materialul cuprinde un volum de noţiuni mai mare dacât s-ar cuveni pentru numărul de ore afectate disciplinei respective (un semestru), dar el aste indestulător pentru un curs de un an. Lucrarea poate fi utilizată şi de către studenţii de la unele specializări de *Master* in informatică, precum şi ca material documentar pentru lucrări de licenţă. Ea poate servi şi ca suport de curs opţional, mai ales ca material de iniţiere, fiind completată la nevoie cu bibliografie suplimentară pentru capitolele de aplicaţii, sau pentru unele perobleme noi ce nu au fost tratate aici.

Structurată pe şapte capitole, lucrarea prezintă mai intâi problemele generale ale construcţiei modelelor de simulare a sistemelor de orice fel, precum şi o introducere in limbajul specializat de simulare GPSS (General Purpose Simulation System), interesant mai ales pentru studenţii de la specializările de *informatică*. Se prezintă şi o introducere formală in studiul *sistemelor cu evenimente externe discrete,* pe baza cărora se construiesc modelele de simulare cu calculatorul, precum şi alte aplicaţii informatice.

Capitolele 2-4 se ocupă de simularea numerelor aleatoare, variabilelor aleatoare şi vectorilor aleatori, construind şi analizând algoritmi pentru simularea principalelor repartiţii de probabilitate, unidimensionale sau multidimensionale, intâlnite in practică.

Capitolul al cincilea prezintă proceduri de simulare a traiectoriilor lanţurilor Markov sau a unor procese stochastice particulare.

Capitolul al şaselea tratează câteva probleme de calcul numeric rezolvate cu ajutorul metodei Monte Carlo cum sunt: calculul integralelor, rezolvarea sistemelor de ecuaţii liniare şi a ecuaţiilor integrale, rezolvarea numerică a problemei Dirichlet.

Intrucât o problemă importantă a aplicării simulării este *validarea* modelului construit, ultimul capitol prezintă câteva modele de simulare pentru sisteme de aşteptare sau sisteme de stocuri, acestea prilejuind şi o scurtă introducere matematică in teoria firelor de aşteptare şi in teoria stocurilor. Modelele matematice corespunzătoare se folosesc de regulă la validarea modelelor de simulare.

Toate capitolele conţin şi câteva exerţiţii menite să antreneze cititorul in inţelegerea mai aprofundată a materialului tratat in lucrare. Pentru fiecare exerciţiu se dau indicaţii de rezolvare sau se prezintă chiar soluţia.

Când este cazul, prezentarea noţiunilor şi rezultatelor este insoţită de figuri sau grafice şi este presărată cu exemple menite să uşureze inţelegerea unor tehnici sau metode teoretice. Pentru sintetizarea unor rezultate sunt alcătuite uneori tabele.

Bibliografia cuprinde de regulă cărţi sau lucrări ce pot fi găsite de către studenţi sau alţi cititori in biblioteci.

Considerăm că această lucrare poate fi utilă ca documentaţie de iniţiere şi lucrătorilor din informatică ce abordează teme de proiectare asistată de calculator sau de modelare.

Prin conţinutul său, cartea de faţă constituie numai un material de iniţiere. Elaborarea modelelor de simulare pentru sistemele reale este un demers dificil care implică colaborarea unor echipe specializate. Totuşi, un curs de iniţiere ca cel de faţă poate contribui la acumularea unei experienţe minimale pentru inceput.

Autorul

Bucureşti, iunie 2004.

Cuprins

Cap. 1

Generalităţi despre simulare

1.1 Introducere

Cuvantul *simulare* este de origine latina si inseamna *capacitatea de a reproduce ceva*.In informatica, termenul de simulare a fost introdus de John von Neumann la inceputul anilor '40, odata cu aparitia primelor calculatoare electronice.J.von Neumann impreuna cu grupul de savanti de la Scoala *Los Alamos* (Ulam, Metropolis, etc) au dezvoltat primele aplicatii ale calculatoarelor. Tot ei au introdus cuvintele *Cercetari operaţionale* (pentru a desemna aplicaţiile legate de dirijarea operaţiilor militare pe arii geografice mari ale globului pamnantesc!) precum şi *metoda Monte-Carlo* (pentru a desemna aplicaţii ale calculatoarelor bazate pe utilizarea numerelor aleatoare). In accepţiunea actuala a informaticii, simularea cuprinde o serie de aplicaţii care realizează *imitarea* comportamentului unor parţi ale lumii reale (simularea stochastica), luand in considerare si comportamentul aleator al acesteia.Din domenul simularii face parte şi metoda Monte Carlo.

Lucrarea de faţa trateaza simularaea sub acest aspect general, pornind de la următarea definiţie mai putin formalizată [10].

Definiţia 1.1 *Simularea este o tehnică de realizare a experimentelor cu calculatorul, care implică utilizarea unor modele matematice şi logice care descriu comportarea unui sistem real (sau a unor componente ale sale) de-alungul unor perioade mari de timp.*

Deci simularea se realizeaza pe baza unui model special, numit *model*

de simulare, cu ajutorul căruia se realizează experimentele prin intermediul calculatorului. Modelul de simulare se construieşte pe scheletul unui model matematic şi se finalizează intr-un algoritm. De aceea in cele ce urmează vom prezenta schema generală a unui model de simulare, pornind de la descrierea principalelor elemente ale unui model matematic.

• **Model matematic.** Prin definiţie, un model [10] este un *analog* ce reprezinta o parte a lumii inconjuratoare intr-un mod uşor de perceput de catre noi. Modelul ne reprezintă uneori realitatea prin scheme, figuri geometrice sau alte obiecte ce ne sunt familiare şi pe care le intelegem uşor (i.e. la fel de bine cum le "vedem" sau le "pipăim"). Modelul matematic ne reprezinta realitatea folosind elemente sau abstracţiuni matematice.

Elementele constitutive ale unui model matematic sunt urmatoarele [10]:

a) *Variabile* (V) şi *Parametri* (P) care pot fi de **intrare** (VI, PI), dacă pot fi percepute (masurate sau inţelese *uşor*), sau de **ieşire** (VE, PE), daca dimpotrivă, masurarea sau perceperea lor este dificilă. Variabilele şi parametri pot lua valori numerice sau logice. Deosebirea dintre variabile şi parametri consta in aceea că parametrii nu işi schimbă valorile pe perioade mari de timp, in timp ce variabilele işi schimba valorile chiar pe intervale mici de timp. *Scopul* modelului este de a exprima variabilele şi parametri de ieşire in funcţie de variabilele şi parametri de intrare, cu eventuala satisfacere a unor condiţii de performanţă de către sistem (de ex. condiţii de optim). Unele VI pot fi *aleatoare*, caz in care şi unele variabile sau parametri de ieşire vor fi de asemenea aleatoare.

b) *Relaţiile funcţionale* constituie o altă categorie de elemente ale unui model matematic. Ele sunt de forma

$$F_i(VI, PI, VE, PE) = 0$$

(adică implicite) şi pot fi la randul lor de două tipuri:

- ecuaţii, care sunt satisfăcute numai de anumite valori ale variabilelor sau parametrilor, şi ;

- identităţi, care sunt satisfăcute de orice valori ale variabilelor şi parametrilor; ele exprima *condiţii de echilibru* sau *legi de conservare.*

Ecuaţiile si identitaţile pot fi relaţii algebrice sau transcendente, diferenţiale sau integrale, detrministe sau stochastice, etc.

c) *Caracteristicile operative* constituie o altă categorie de elemente ce compun un model matematic şi ele pot fi:
- ipoteze de lucru (referitoare la relaţiile funcţionale);
- ipoteze statistice (referitoare la VI-aleatoare).

d) *Tehnica de rezolvare* este un alt element constitutiv al unui model matematic. Ea este o tehnică matematică ce realizează separarea elementelor de ieşire in funcţie de elementele de intrare, adică:

$$(V, P)_E = f_j(V_I, P_I).$$

Cu alte cuvinte, tehnica de rezolvare a modelului exprimă sub forma *explicită* variabilele şi parametri de ieşire in funcţie de variabilele şi parametri de intrare.

Tehnicile matematice de rezolvare a modelelor sunt insă *sărace* atât ca varietate cât şi ca performanţă. De exemplu, ecuaţiile modelului se pot reazolva numai dacă sunt liniare sau uneori pătratice, iar daca sunt de grad superior ele se pot rezolva numai daca au forme particulare. La fel, ecuaţiile diferenţiale sau cu derivate parţiale se pot rezolva cu metode matematice deductive numai in anumite cazuri particulare. De aceea in construcţia modelelor matematice se fac de multe ori *ipoteze simplificatoare* care permit aplicarea tehnicilor de care dispune matematica. (Acesta este scopul utilizării de către model a caracteristicilor operative!). Din aceste motive, modelarea matematică este aproximativă şi ea nu permite rezolvarea *realistă* a problemelor practice. *Utilizarea calculatorului permite imbunătăţirea performanţelor modelelor matematice prin utilizarea metodelor numerice.* Dar şi in aceste conditii modelele matematice nu pot descrie corect realitatea in toată complexitatea ei deoarece nu toate relaţiile dintre obiectele lumii reale se pot exprima prin formule matematice. Intr-o atare situaţie modelul matematic trebuie *completat* cu descrieri care să *imite* anumite comportari ale lumii reale. Acestea se realizează prin descrieri algoritmice de tipul **if-then-else-** sau **if-then-** combinate cu alte structuri algoritmce (cicluri,secvenţe, etc.). In acest fel, modelul matematic se completează şi extinde sub formă de algoritm şi devine *model de simulare.* **Simularea măreşte deci mult posibilitatea de tratare realistă**

a problemelor aplicative. Construcţia unui model de simulare, care in fapt este un algoritm complex, dezvoltat pe *scheletul* unui model matematic, este o sarcină nu prea uşoară; o vom trata mai jos.

• **Clasificări ale modelelor matematice.** Mai intai să vedem insă cum pot fi clasificate modelele matematice?

(i). Clasificarea modelelor matematice dupa natura variabilelor utilizate de model: **continue/discrete**; statice/**dinamice** (daca timpul nu intervine sau dacă apare explicit ca variabilă a modelului); deterministe/**stochastice** (dupa cum nu conţine sau conţine măcar o variabilă de intrare ca variabilă aleatoare).

(ii). Clasificare topologică, după structura determinată de parţile in care se descopune modelul (când este cazul): cu *o componentă*/ **cu mai multe componente in serie , in paralel, in reţea.** (Tipurile de modele scrise cu caractere ingroşate sunt importante căci sunt *realiste* şi se construiesc cu dificultate. Modelele de simulare sunt de aceste tipuri).

Un model, fie el matematic sau de simulare, constitue de fapt o *clasă de modele*.

Pentru a ilustra caracteristicile componente ale unui model matematic vom da un exemplu.

Exemplul 1.1 [10,11]. **Sistemul de aşteptare** este o parte a lumii reale in care se produc *aglomerări*. Un astfel de sistem constă din una sau mai multe staţii de serviciu care servesc (dupa anumite re guli) clienţi care *sosesc* in sistem. Dacă sosesc mulţi clienţi şi staţiile de servire nu pot să-i servească repede, atunci se formează *cozi de aşteptare* in sistem. Proprietarul sau administratorul sistemului trebuie să dirijeze sistemul astfel incât nici clienţii să nu *aştepte* mult până primesc serviciul, dar nici staţiile de serviciu să nu *lenevească* prea mult, căci lenevirea lor aduce pierderi administratorului sistemului.

Modelarea matematică a sistemelor de aşteptare are deci ca *scop* realizarea unui echilibru intre pierderile cauzate clienţilor prin aşteptari şi pierderile cauzate administratorului prin leneviri. Modelele matematice de acest tip sunt cunoscute sub numele de *teoria matematică a cozilor sau teoria aşteptării.* Ele sunt utilizate in diverse situaţii practice: in comerţ, in managementul sistemelor de comunicaţii şi de transport,

dar şi in dirijarea şi controlul funcţionării in timp real a reţelelor de calculatoare.

Un model de asteptare contine de regula urmatoarele elemente:

Variablele de *intrare* (cunoscute) VI: AT = timp de intersosire (al clienţilor); ST = timp de serviciu al unui client (caracteristică a staţiilor); aceste variabile sunt de regulă aleatoare, fapt ce măreşte dificultatea modelării. Sosirile se pot *măsura* şi prin NA = *număr de clienţi sosiţi pe unitatea de timp* iar serviciile se pot măsura şi prin NS =*număr de clienţi serviţi pe unitatea de timp.* Acestea sunt deci variable aleatoare *discrete.*

Variabilele de *ieşire* (necunoscute) VE sunt: WT = timp de aşteptare (sau WL= lungimea cozii, variabilă discretă);

TID = timp de lenevire (sau NID= numărul de staţii de servire care *lenevesc*, variabilă discretă);

Scopul modelului: cunoscând repartiţiile de probabilitate ale lui AT şi ST să se determine informaţii despre WT (WL) sau TID (NID) şi să se stabilească cum trebuie să se realizeze ST astfel incât o anume funcţie de eficienţă (sau cost) să fie optimă sau convenabilă pentru administratorul sistemului.

Construcţia modelelor de simulare se bazează pe studierea (in funcţie de ipotezele făcute asupra variabilelor de intrare AT, ST) a procesului stochastic discret $N(t)$ = numărul de clienţi in sistemul de aşteptare la momentul de timp t. Acest proces este *proces de naştere şi deces.* Nu vom studia aceste procese deocamdată. (Acest lucru va fi făcut in ultimul capitol!). Vom prezenta mai intai *construcţia unui model de simulare.*

1.2 Construcţia unui model de simulare

Modelul de simulare (MS) presupune utilizarea calculatorului şi el se construieşte pe baza/scheletul unui model matematic; mai precis el completează modelul matematic descriind unele *relaţii* prin algoritmi, deci in final MS este un algoritm. Acest algoritm trebuie să descrie *corect* evoluţia sistemului şi să permită efectuarea de experienţe (prin rulări pe calculator), care să inlocuiască experienţele ce ar trebui realizate asupra sistemului real.

1.2.1 Structura unui model de simulare

Construcţia unui MS utilizează două concepte de bază: **ceasul simularii** şi **agenda simulării**

• **Ceasul simulării.** Ceasul asigură eşalonarea corectă in timp a evenimentelor create de model şi uneori ajută la implementarea condiţiei de terminare a simulării. El este de două feluri:

a) ceas cu creştere constantă;

b) ceas cu creştere variabilă;

Notă: Evenimentele corespund unor *modificări in sistem* adica modificari ale valorilor unor variabile care se calculează sau se *generează* prin instrucţiuni ale modelului (chiar şi dacă sunt *aleatoare!*).

In Figura 1.1 sunt prezentate cele două modalităţi de creştere a ceasului. Ceasul porneşte cu valoarea zero la inceputul simulării. Dacă modelul se bazează pe ceasul cu creştere variabilă, atunci ceasul este crescut cu valoarea ce corespunde apariţiei *primului eveniment următor*; apoi programul *prelucrează* evenimentul , dupa care ceasul creşte din nou reluându-se *ciclul simulării*, pâna când ceasul atinge o valoare data iniţial, T_{max} care corespunde perioadei pe care se realizează simularea.

Ceasul cu creştere constantă presupune că de fiecare dată creşterea se realizează cu o *cuantă* de timp c constantă; apoi se prelucrează toate evenimentele apărute pe intervalul de timp de lungime c, după care se reia ciclul simulării. Simularea se termină de asemenea când ceasul atinge valoarea T_{max}.

Terminarea simulării se poate realiza şi impunând condiţia ca modelul să prelucreze un anumit număr dat de evenimente de un tip precizat. (De exemplu, in cazul unui model de simulare a unui sistem de aşteptare, se poate impune condiţia ca simularea să se termine când s-a simulat servirea unui număr dat de clienţi).

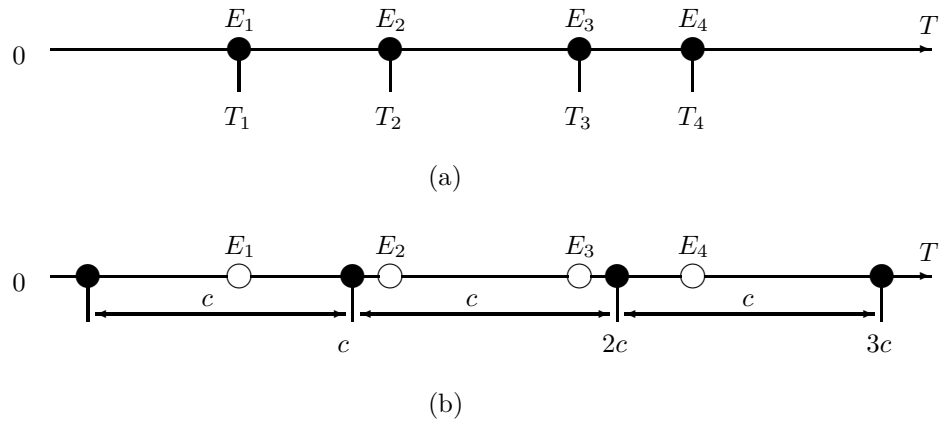

Fig. 1.1. Variaţia ceasului.
a) Cazul ceasului cu creştere variabilă. T_i = valorile ceasului.
b) Cazul ceasului cu creştere constantă c.

A rămas oarecum ne precizat ce se inţelege prin prelucrarea unui eveniment.Acest fapt se inţelege uşor dacă precizăm şi conceptul de *agendă a simulării.*

• **Agenda simulării.** Agenda se compune din elementele *memorate* de modelul de simulare. Variabilele modelului iau diverse valori pe parcursul simulării; de aici rezulta că pe parcursul simulării apar multe evenimente. Memorarea in totalitate a acestor evenimente, impreună cu caracteristicile lor, nu este nici recomandată dar nici necesară daca simularea se realizează pe perioade mari de timp. De aceea, se memorează (in agendă) numai ceea ce este strict necesar. Evenimentele sunt de *diverse tipuri*; unele variabile descriu şi *stări* ale sistemului (sau ale unor componente ale acestuia). Evenimentele sunt creiate sau generate la *momente de timp* ulterioare valorii ceasului. De aceea agenda se compune din două părţi: **agenda evenimentelor curente** AEC şi **agenda evenimentelor viitoare.**

Deci **agenda** $A = AEC \oplus AEV$**,** unde:

AEC = agenda evenimentelor curente (care au timpul de apariţie egal cu valoarea ceasului); iar

AEV = agenda evenimentelor viitoare (care au timpul de apariţie ulterior valorii curente a ceasului).

Algoritmul simularii prelucrează deci numai evenimentele din AEC; prelucrarea unui eveniment inseamnă fie determinarea apariţiei unui nou eveniment (ce se memorează in AEV) sau modificarea unei stări,

fie distrugerea unui eveniment ("stergerea") lui din agendă. Prelucrările evenimentelor ţin seama şi de stările sistemului la acel moment.

Algoritmul simulării gestionează/actualizează agenda prin *interacţiunea acesteia cu ceasul;* intr-un ciclu al simulării ceasul este acualizat, dupa care se selectează din agenda A evenimentele care fac parte din AEC şi se prelucreaza aceste evenimente pana cand AEC devine vida. Atunci, ceasul este cresut din nou şi se reia **ciclul simulării**. Deci agenda simulării se modifică pe parcursul simulării conform următoarei relaţii de dinamică

$$A = A \oplus AE_{gen} \ominus AE_{elim}$$

unde A_{gen} sunt evenimentele generate pe parcursul unui ciclu, iar A_{elim} sunt evenimentele eliminate cu ocazia prelucrării AEC. (Semnele $\oplus$ şi $\ominus$ se autexplică).

Structura generală a unui MS este descrisă in schema logică din Figura 1.2, pentru a cărei inţelegere este necesară şi urmărirea schemelor ajutătoare din Fig. 1.3, a)-g). Descrierea structurii MS este prezentată structurat.

In Figura 1.3 a) se prezintă schema logică generală a *rutinei principale;* MS este conceput ca un *joc* constituit din *mutări.* Se urmăreşte ca jocul să conducă la o performanţă a sistemului prin efectuarea a mai multe mutări, fiecare mutare fiind aleasă astfel incât să conducă in final la *satisfacerea jocului.* In blocul central al Fig.1.3 a) (rutina principală!) se execută ciclul de bază al simulării (interacţiunea ceas-agendă!).Rutina de comutaţie alege evenimentul ce urmează a fi prelucrat din AEC si transferă controlul la rutina eveniment care-l va prelucra.

Rutina eveniment prelucrează evenimente după schema:

$$A \rightarrow B,\ A..... > B$$

care inseamnă că A il detrmină *sigur* pe B, respectiv A *il poate determina* pe B (cu o probabilitate); sau,

$$A \rightarrow /B,\ A..../..... > B$$

care inseamnă că A il *distruge* (elimină) pe B sigur, respectiv A il *poate distruge* pe B cu o probabilitate dată. Prelucrarea evenimentului

ţine seama de regulile de prelucrare ale acelui tip de eveniment şi de starea/stările sistemului.

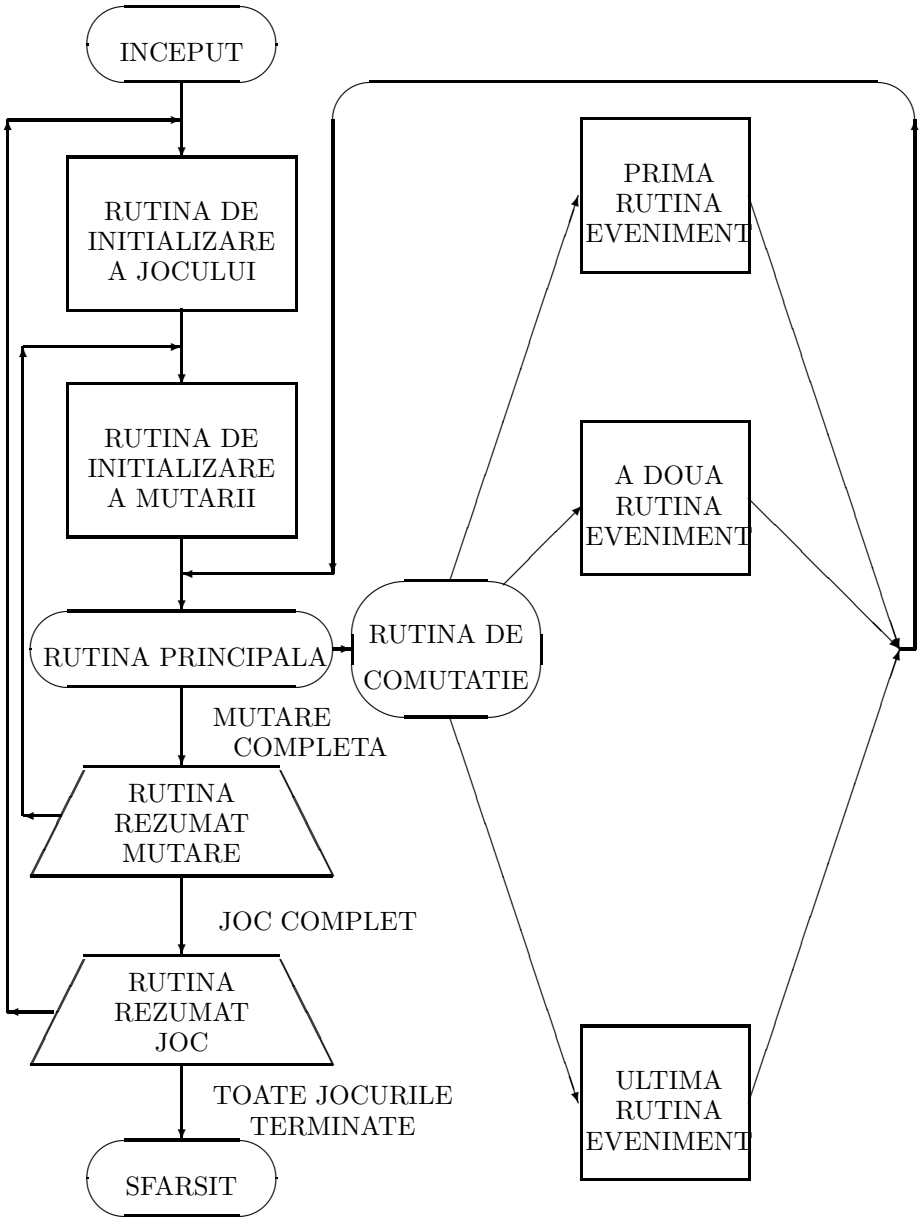

Fig. 1.2. Schema logica globală a unui program

de simulare bazat pe ceas cu creştere variabilă.

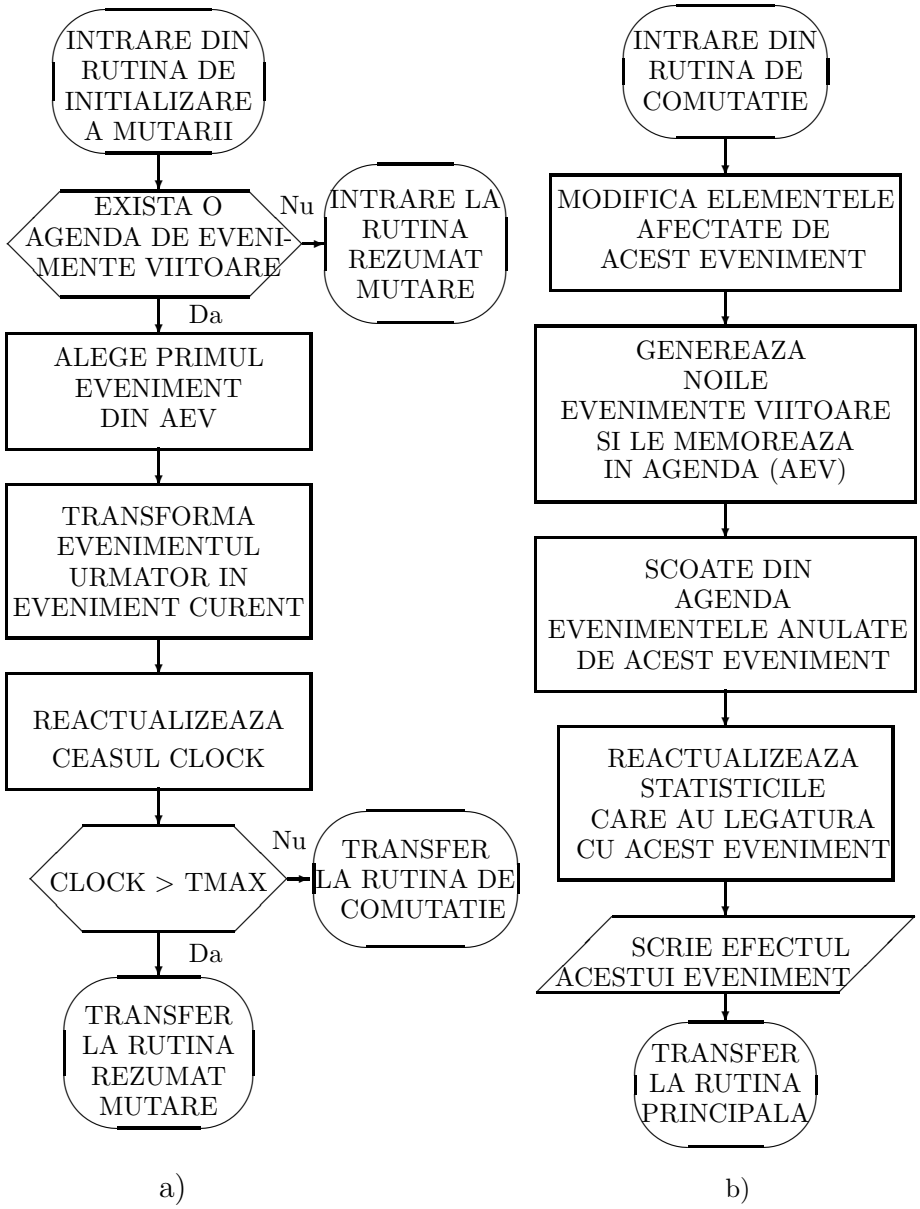

Fig. 1.3. Schemele logice ale rutinelor programului de simulare.

a) Rutina principala

b) Rutina eveniment

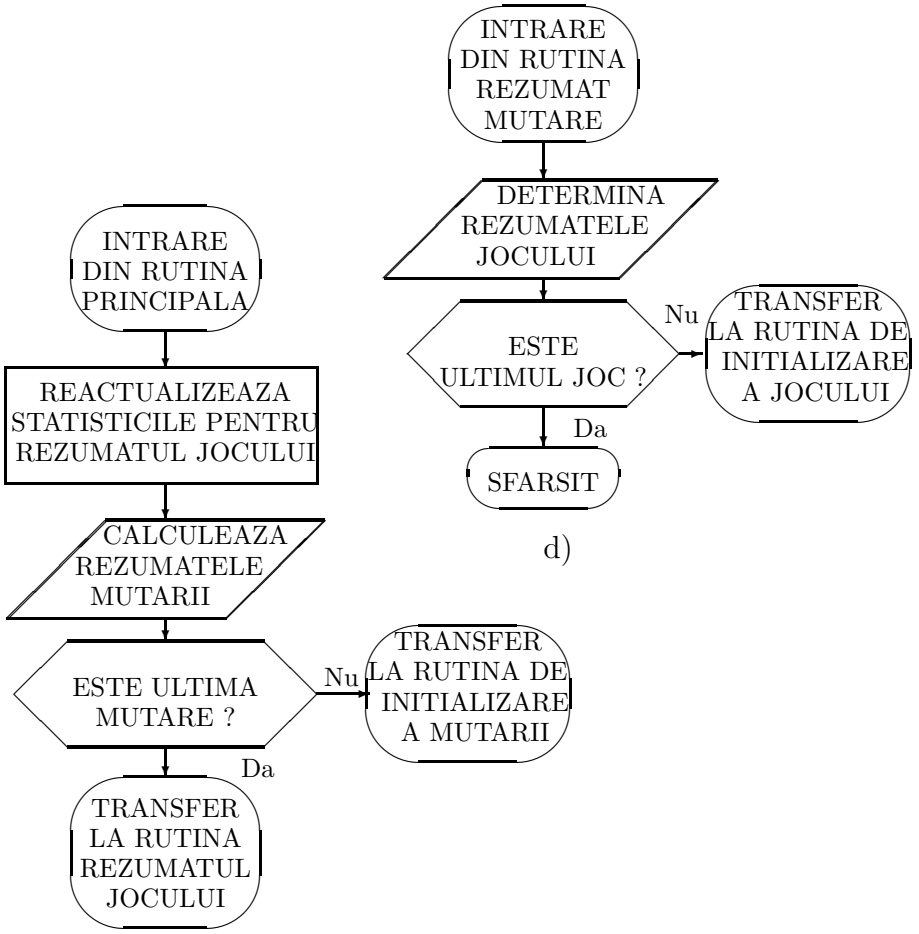

Fig.1.3(continuare) Schemele logice ale rutinelor programului de simulare.

c) Rutina rezumat mutare;

d) Rutina rezumat joc;

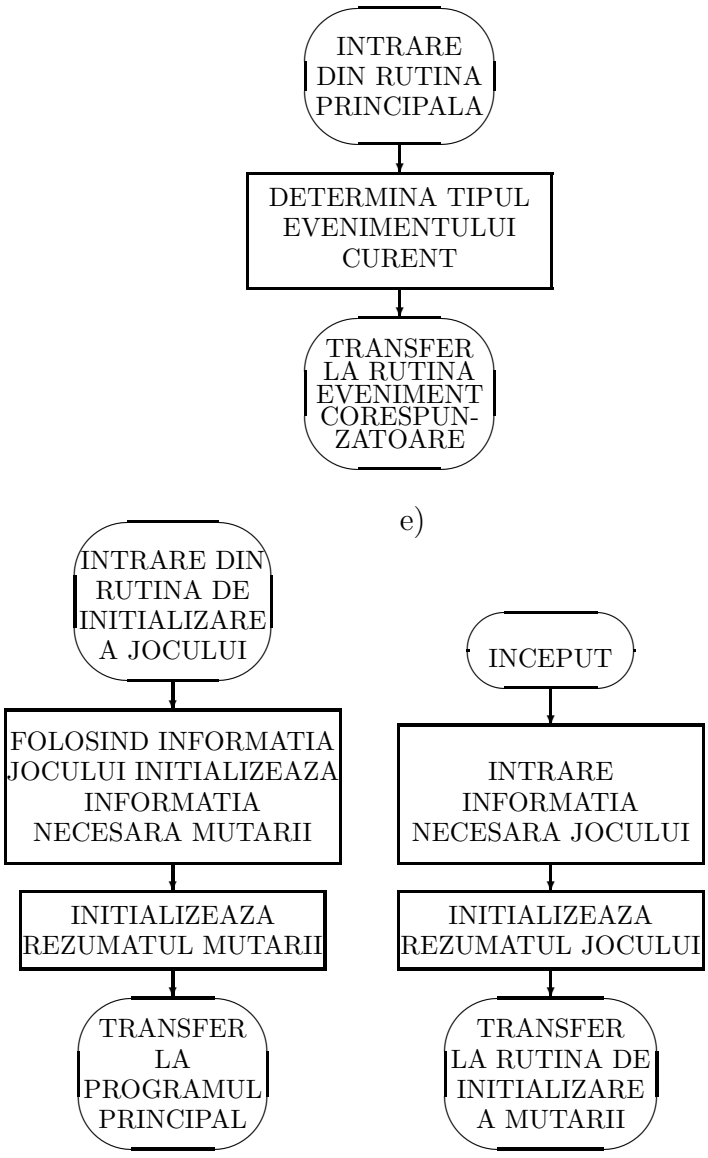

Fig.1.3(continuare) Schemele logice ale rutinelor programului de simulare.

e) Rutina de comutaţie;

f) Rutina de iniţializare a mutării;

g) Rutina de iniţializare a jocului.

1.2.2 Concepte de bază in modelarea sistemelor

Descrierea unui MS prezentată mai sus poate fi realizată intr-un cadru formal general folosind concepte din *teoria matematică a sistemelor* [8, 12, 13].

Prin sistem [8] se inţelege, sub forma cea mai vagă, o mulţime de obiecte $\mathcal{O}$ interconectate prin intermediul unei relaţii $\mathcal{R}$, adica **Sistem=** $(\mathcal{O}, \mathcal{R})$, unde

$$\mathcal{R} \subset \mathcal{O} \times \mathcal{O}.$$

Definiţia formala generala a unui sistem este [8]:

Definiţia 1.2 Un sistem este următoarea structură de mulţimi:

$$S = (T, X, \Omega, \Sigma, Y, \delta, \lambda)$$

unde:

$T =$ **timpul de bază al sistemului (ceasul sistemului)**;

$X =$ **mulţimea intrărilor**;

$\Omega =$ mulţimea **segmentelor de intrare** (forma intrărilor!)

segment= $\omega : (t_0, t_1) \mapsto X$,=$\omega_{(t_0,t_1)}$ = grafic= $\{(t, \omega(t)), t_0 \leq t \leq t_1\}$.

Se foloseste de regulă notaţia $\omega_{(t,t_1)} = \{(\tau, \omega(\tau)), t < \tau \leq t_1\}$ şi notaţiile $\omega = \omega_{(t_0,t_1)}$, $\omega_{t)} = \omega_{(t_0,t)}$, $\omega_{(t} = \omega_{(t,t_1)}$; $\omega = \omega_{t)}\omega_{(t}$;

$\Sigma =$ **mulţimea stărilor interne ale sistemului** = *memoria sistemului;*

Conceptul de *stare* este esenţial in modelarea sistemelor; el descrie structura internă (intimă!) a sistemului;

$\delta =$ **funcţia de tranziţie a stărilor** definită ca

$$\delta : \Sigma \times \Omega \mapsto \Sigma.$$

Ea satisface relaţia

$$\delta(\sigma, \omega) = \delta(\delta(\sigma, \omega_{t)}), \omega_{(t)}, \forall t,\ \omega = \omega_{t)}\omega_{(t},$$

care este *axioma semigrupului* sau proprietatea de *separabilitate a stărilor.* Mulţimea (graficul) (ω, σ) se numeşte *traiectorie a stărilor*;

Y =**mulţimea ieşirilor** (Sistemul este presupus *deschis*, intrările si ieşirile fiind exterioare sistemului).

Mulţimea Y conţine *răspunsul* sistemului la intrarea de forma ω când la momentul *iniţial* t_0 sistemul sa află in starea σ.

λ = **funcţia de răspuns,** de forma $\lambda : \Sigma \times X \times T \mapsto Y$;

$\lambda(\sigma, x, t)$ conduce la un *segment de ieşire* ce reprezintă forma răspunsului sistemului la intrarea x, la momentul t când starea la momentul iniţial $t_0, t_0 < t$, este σ.

Nota: din definiţie rezultă că pentru o stare iniţială σ, (la momentul t_0) când are loc intrarea de forma ω se realizează o *traiectorie unică a stărilor*. Cu alte cuvinte, traiectoria stărilor satisface relaţiile:

$$STRAJ_{\sigma,\omega} : (t_0, t_1) \mapsto \Sigma, \quad a.i. STRAJ_{\sigma\omega}(t_0) = \sigma$$

$$STRAJ_{\sigma,\omega}(t) = \delta(\sigma, \omega_{t)}), \forall t \in (t_0, t_1).$$

Proiecţia $STRAJ$ obtinută prin funcţia de tranziţie a stărilor compusă cu funcţia de răspuns, este *traiectoria de ieşire*

$$OTRAJ_{\sigma,\omega} : (t_0, t_1) \mapsto Y.$$

Dacă $\lambda = \lambda(\sigma)$ (*ieşirea* depinde numai de σ) atunci rezulta relaţia simplă

$$OTRAJ_{\sigma,\omega}(t) = \lambda(STRAJ_{\sigma,\omega}(t)).$$

• **Nivele de reprezentare a sistemelor.** Există mai multe *nivele de reprezentare a sistemelor* şi anume [8]:

• **Reprezentarea la nivel de comportare.** Sistemul este o cutie neagră (**black box**), exprimat formal prin relaţia

$$R_s = \{(\omega, \rho) | \omega \in \Omega,\ \rho = OTRAJ_{\sigma,\omega},\ \sigma \in \Sigma\}\,.$$

Acest nivel de reprezentare este cel mai vag. El descrie numai relaţiile de intrare/ieşire ce se pot observa dinafara sistemului (care deci se comportă ca o *cutie neagra*).

• **Reprezentarea la nivel de structură de stare.** La acest nivel se *intră* in structra internă (*intimă*) a sistemului, adică se stabilesc elementele acestuia : $(T, X, \Omega, \Sigma, Y, \delta, \lambda)$, definite ca mai sus.

• **Reprezentarea modulară** (ca structură compusă). Dacă sistemul este complex, atunci se identifică *subsisteme* ale acestuia precum si interconexiunile dintre ele in sensul că ieşirile unor subsisteme sunt intări in alte subsisteme, intrările unor subsisteme fiind intrări ale sistemului şi iesirile unor subsisteme fiind ieşiri ale sistemului. Cele trei nivele de reprezentare ale unui sistem sunt ilustrate grafic in Fig.1.4.

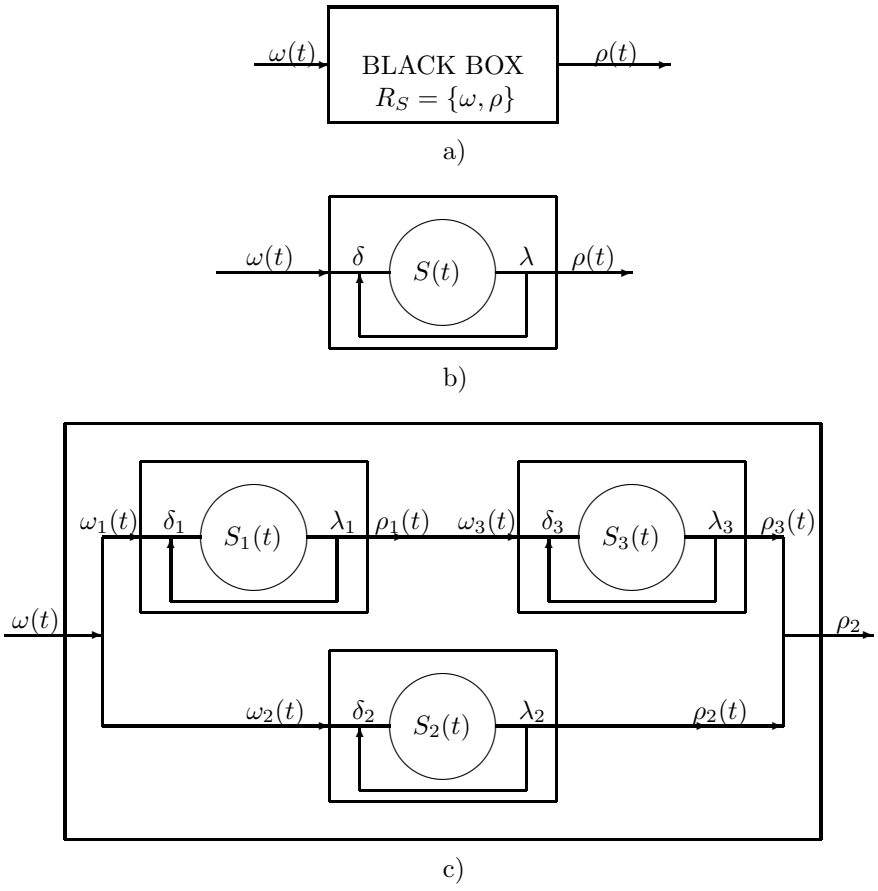

Fig. 1.4. Descrierea sistemului

a) Sistem la nivel de comportare

b) Sistem la nivel de structură de stare

c) Sistem la nivel de structură compusă.

Nivelele de reprezentare a sistemelor, pornind de la forma cea mai vagă si continuând pană la forma detaliată legitimeaza **Metodologia**

"TOP DOWN" de proiectare a sistemelor de orice fel, in particular, metodologia de proiectare a **sistemelor informatice** si in general metodologia de proiectare *ierarhică, descendentă* a programelor.

1.2.3 Modelul sistemului cu evenimente externe discrete

Modelul de simulare se incadrează in categoria *modelului de sistem cu evenimente externe discrete* (DEVS=Discrete EVent System), [8, 12,13]. Acesta este un sistem particular de forma

$$S_M = (X_M, S_M, Y_M, \delta_M, \lambda_M, \tau_M)$$

in care timpul T este considerat *implicit* iar elementele au urmatoarea semnificaţie:

X_M = mulţimea (discretă) de evenimente *externe*; (inţelegem evenimentele ca nişte entitaţi oarecari);

S_M = mulţimea secvenţelor discrete de stări interne pe care le pot parcurge evenimentele externe când acestea intră in sistem;

Y_M = mulţimea ieşirilor, sau răspunsurilor sistemului;

δ_M = funcţia de *quasi-tranziţie* exprimată prin două componente:

$\delta_M^{\emptyset} : S_M \mapsto S_M$ care spune ce stări parcurg evenimentele externe când *nu găsesc* alte evenimente intrate in sistem;

$\delta_M^{ex} : X_M \times S_M \times T \mapsto S_M$-adică $\delta_M^{ex}(x, s, e)$ care spune cum se transformă starea când intră in sistem x, dacă starea iniţială era s şi dacă e este timpul scurs de la apariţia ultimei tranziţii de stare;

λ_M = *funcţia de ieşire* (de răspuns), $\lambda_M : S_M \mapsto Y_M$, $\lambda_M(s) \in Y_M$;

τ_M = *funcţia de intârziere* (decalaj), $\tau_M : S_M \mapsto R^+$, care spune cât intârzie sistemul in starea s (intârzierea este $\tau_M(s)$);

Sa arătăm acum de ce S_M este un sistem in sensul general definit mai sus.

S_M este un $S = (T, X, \Omega, \Sigma, Y, \delta, \lambda)$ deoarece

$T = [t_0, \infty), t_0 - initial$ şi el este subinţeles in mod implicit;

$X = X_M \bigcup \emptyset$; ($\emptyset$ = nu este eveniment);

Ω = mulţimea segmentelor; segmentele sunt secvenţe discrete, finite de evenimente asociate cu momentele (ordonate) de timp când acestea intră in sistem;

Σ = mulţimea stărilor,adică

$$\Sigma = \{(s, e) | s \in S_M, 0 \leq e \leq \tau_M(s)\}.$$

δ se construieşte din δ_M astfel

$$\delta : \Sigma \times X \mapsto \Sigma;$$

$$\delta(s, e, \emptyset) = (s, e) \in \Sigma, \; daca \; e < \tau_M(s)$$
$$\delta(s, \tau_M(s), \emptyset) = (\delta_M^{\emptyset}(s), \tau_M(s))$$
$$\delta(s, e, x) = (\delta_M^{ex}(x, s, e), e);$$

Funcţia de ieşire este $\lambda = \lambda_M$.

O formalizare a teoriei modelelor de simulare discretă se găseşte in cărţile lui ZEIGLER [12,13].

Ea poate servi ca metodologie de construcţie a modelelor de simulare si conduce la construcţia **modelelor de simulare OO** (=Orientate Obiect), [13].

1.2.4 Metodologia de realizare a experimentelor de simulare (Metodologia simulării), [10, 11].

Etapele realizării unui experiment de simulare sunt:

1^0 **Formularea problemei**, care constă in a preciza intrebările la care trebuie să răspunda modelul şi a preciza domeniul lumii reale ce trebuie analizat. Aici se precizează si forma răspunsului la intrebări (de ex. dacă se vor produce grafice, tabele sau chiar texte scrise).

2^0 **Realizarea unor experimente preliminare**; in această etapă se stabilesc, pe baza observaţiilor sau datelor culese din lumea reală, sau existente, referitoare la aceasta (istorice), *variabilele şi parametri* şi care din acestea sunt de intrare sau de ieşire.

3^0 **Prelucrarea (interpretarea) primară a datelor preliminare**; acum se disting *variabilele aleatoare,* se estimează parametri şi se testează ipoteze statistice. (Statistica matematică joacă un rol important in simulare).

4^0 **Formularea unui model matematic preliminar, incomplet**; in această etapă se precizează relaţii funcţionale, ipoteze de lucru (care să concorde cu datele existente, colectate) şi se indentifica ce relaţii nu pot fi exprimate matematic şi care sunt dificultăţile ce ar trebui inlăturate pentru a răspunde la intrebările formulate.

5^0 **Evaluarea modelului** (**etapă de decizie**!); se urmăreşte să se evalueze *complexitatea* modelului, daca el poate răspunde in timp real şi *complet* la intrebări; in această etapă se pot eventual reizui unele din etapele precedente făcandu-se simplificări sau completări ale modelului matematic propus.

6^0 **Construcţia modelului de simulare** (sub formă de algoritm detaliat); modelul se construieşte conform celor precizate anterior urmărindu-se ca el sa fie *general*. La construcţia algoritmului simulării se va avea in vedere şi cum se va realiza programarea: intr-un **limbaj evoluat** sau int-un **limbaj specializat de simulare**. Există un număr mare de limbaje de simulare toate având la bază, mai mult sau mai puţin, filozofia DEVS. Un astfel de limbaj (ce va fi prezentat in secţiunea următoare) este GPSS= General Purpose Simulation System [1, 14]. O versiune românească a acestui limbaj este SIMUB (limbajul de SIMulare al Universitaţii din Bucureşti) şi el a fost implementat la sfârşitul anilor '70 pe sistemele de calcul FELIX C-256 de cercetători ai Centrului de calcul al Universităţii din Bucureşti (CCUB).

In anii '60-70 au fost construite [10] şi implementate multe limbaje de simulare pentru sistemele de calcul de generatia a III-a.

Printre limbajele de simulare introduse atunci menionăm limbajul SIMULA, care introduce conceptul de *clasă* pentru a desemna o mulţime de obiecte ale unui program care au anumite proprietati şi care se supun unui tratament comun; acest concept a generat pe cel de *obiect* şi este deci premergător *programării orientată spere obiecte.* Un alt limbaj, care se baza de fapt pe o subrutina complexă FORTRAN, ce putea fi combinată cu alte subprograme in acest limbaj şi care prezintă din această cauză multă flexibilitate in scrierea de programe de simulare este limbajul GASP IV; o versiune a acestui limbaj, numită GASP-SIMPATIC a fost implementata la inceputul anilor '80 pe sistemele de calcul FELIX C-256 şi pe minicalculatoarele CORAL şi INDEPENDENT, de către cercetatori ai CCUB.

Limbajele de simulare sunt atât limbaje de programere cat şi instrumente care facilitează construcţia modelelor de simulare. Utilizând un astfel de limbaj, analistul-programator va fi scutit de grija construcţiei instrumentelor specifice simulării (organizarea agendei, manipularea şi controlul ceasului al AEC sau AEV, etc.), el putându-se astfel concentra numai asupra definirii elementelor specifice sistemului pe care-l

modelează (tipurile de evenimente, componentele şi stările sistemului, definirea răspunsurilor la intrebări, etc).

Avantajele utilizării limbajelor de simulare: modelele se construiesc rapid, experimenţele de simulare se realizează repede (sunt experienţe artificiale ce nu au nevoie de scurgerea timpului fizic pentru a fi observate). **Dezavantajele** constau in faptul că modelele de simulare realizate in limbaj specializat sunt supuse unor restricţii determinate de posibilităţile limitate ale acestora.

De regulă modelele de simulare generează valori ale variabilelor de intrare si produc *selecţii statistice* de valori asupra variabilelor de ieşire. Pentru formularea răspunsurilor la intrebări ar trebui prelucrate in mod cât mai evoluat aceste selecţii. Limbajele de simulare nu dispun de regulă de astfel de facilităţi, mai ales presupunând (ceea ce este un fapt real!), că valorile de selecţie produse de simulare nu sunt independente stochastic.

Avantajele modelelor dezvoltate in limbaje evoluate sunt: acurateţea rezulatelor, flexibilitatea modelelor (adică pot fi extinse cu noi tehnici numerice de calcul, pot fi mai generale, etc). **Dezavantajele** constau in faptul că se constriuesc mult mai greu (utilizatorul trebuie să implementeze tehnicile de manipulare a ceasului şi de gestionare a agendei).

7^0 **Validarea modelului** este de asemenea o etapă importantă. In afară de validarea formală (testare sintactică şi logic-formală a programului), trebuie să se decidă dacă modelul rezolvă **corect** problema formulată. Acest lucru se realizează prin compararea rezultatelor simulării fie cu rezultate practice cunoscute, fie prin compararea cu soluţia obţinută cu ajutorul unui model matematic. (De ex.in **teoria cozilor** se cunoaşte soulţia matematică a modelului in anumite ipoteze restrictive; se va obţine soluţia prin simulare pe baza unei selecţii de volum mare a variabilelor de ieşire. Daca soluţia simulată se apropie de soluţia matematică, atunci, in virtutea *legii numerelor mari*, rezultă corectitudinea modelului de simulare, acesta fiind deci *valid* şi in ipoteze diferite de cele cosiderate la validare).

8^0 **Planificarea experienţelor de simulare** este următoarea etapă necesară. Aşa cum am precizat, simularea este un experiment realizat cu calculatorul pe baza unui model ce descrie sistemul real. Orice

experiment (cu caracter statistic ca in cazul simularii) trebuie să se desfaşoare pe baza unui plan experimental. Pentru planificarea experimentelor de simulare se folosesc metode ale **statisticii matematice** (aşa zisele *experimental design*).

9^0 **Prelucrarea şi interpretarea experienţelor de simulare** este o etapă la fel de importantă ca şi cele precedente. Programul de simulare rulat pe calculator produce de regulă valori de selecţie asupra variabilelorde ieşire, dar care nu definesc selecţii statistice in sensul obişnuit (nu sunt selecţii bernouliene deoarece valorile de selecţie sunt de regulă dependente). De aceea prelucrarea experimentelor de simulare se face cu mijloace statistice din cele mai sofisticate (prelucrarea seriilor dinamice). Toate limbajele de simulare posedă un minim de facilităţi pentru prelucrări statistice simple (calcule de medii, dispersii, coeficienţi de corelaţie, histograme, etc,).

• **Utilitatea simulării.** Orice sistem complex, se proiectează pe baza simulării. Alegerea parametrilor de proiectare se realizează (ca intr-un joc!) pe baza unor experienţe bazate pe modelul de simulare. Aceste experienţe sunt mult mai puţin costisitoare decat cele reale.

Cand experienţele reale sunt costisitoare este intotdeauna recomandabil ca experienţele să se realizeze prin simulare. (De exemplu, inainte de aconstrui un baraj, mai intâi **simulăm** comportarea lui pentru a alege cea mai buna variantă de proiectare şi construcţie).

In anumite situatii practice experienţele durează mult si de asemenea este nevoie să se recurgă la simulare.(De exemplu simularea **duratei de viaţă a unei comete** se realizează cu modelul de simulare in cel mult câteva minute sau zeci de minute). Experienţele legate de fiabilitatea sistemelor se realizează de asemenea mai degrabă prin simulare.

Când se construiesc sisteme mari noi, ne mai intâlnite se efectuează mai intâi simularea lor. (De exemplu **zborurile interplanetare**, sistemele hidroenergetice mari cu implicaţii ecologice, etc. se proiectează pe baza unor variante alese cu ajutorul simulării).

Când se proiectează politici complexe de dezvoltare economico-socială de asemenea se utilizează simularea. (De exemplu sistemele de producţie complexe, politicile de dezvoltare regională, proiecte macroeconomice, evoluţia mediului ambiant,etc).

Simularea joacă un rol important in dezvoltarea **tehnologiei GRID** un domeniu nou al informaticii in plină dezvoltare, care se bazează atât pe simulare cât şi pe pe calcul performant, pe inteligenţă artificială, pe reţele de comunicaţie, etc.

1.3 Generalităţi despre limbajul GPSS

Limbajele de simulare sunt in acelaş timp **limbaje de programare** şi **limbaje de modelare;** ele implementează elementele esenţiale ale simulării: manipularea **ceasului** şi **gestionarea memoriei.** Utilizatorul are numai grija descrierii **evenimentelor** şi a **prelucrării lor.**

Programele in limbaj de simulare sunt *mai scurte* dar şi *mai puţin flexibile.* Sunt limitate in privinţa prelucrării şi interpretării experimentelor de simulare.

Vom face o foarte scurta prezentare a **limbajului GPSS** (General Purpose System Simulator), urmand ca cititorul interesat să recurgă la un manual detaliat sau să folosească facilitatea *help* existentă in implementarea GPSS/PC [1, 14]. Acest limbaj a fost dezvoltat pentru prima data de firma IBM la inceputul anilor '60.

• **Entităţile limbajului GPSS.** Limbajul GPSS se compune din 16 **tipuri de entităţi** (elemente de abstractizare).

La fiecare entitate se asociază un numar de proprietăţi sau **atribute** in majoritatea lor **adresabile intern** (de către GPSS), dar unele sunt **adresabile si de către utilizator**;

Atributele sunt:**standard numerice** (numere) sau **standard logice** (valori logice);

Entităţile de bază sunt:

1) **Blocuri** (entităţi care descriu **activităţi);**

2) **Tranzacţii** (elemente **circulante**);ele sunt creiate (printr-un bloc special GENERATE) şi circulă prin model (sistem!), ca urmare a acţiunii altor blocuri. Blocurile au asociate caracteristici numerice sau logice si posedă argumente necesare descrierii activităţilor. Tranzacţiile posedă un număr de parametri standard dar şi psrametri introduşi de utilizator. Prin programul de simulare utilizatorul poate accesa parametrii tranzacţiilor.

Entităţi de echipament:

3) **Staţii de serviciu** sau **facilităţi**; (ele corespund subsistemelor cu o componentă care tratează de regulă o singura tranzacţie!);

4) **Multistaţii** de serviciu sau **depozite**; acestea tratează de regulă mai multe tranzacţii (de ex. liftul, autobuzul, pot transporta mai multi clienti considerati ca tranzactii, etc!);

5) **Comutatorii logici** sunt variabile logice care permit utilizatorului să realizeze ramificarea după dorinţă a fluxului de execuţie in programul de simulare.

Entităţi de calcul ce permit efectuarea (limitat!) a unor calcule:

6) **Variabile aritmetice**; permit evaluarea unor expresii aritmetice, iar rezultatul este memorat de o variabilă aritmetică;

7) **Variabile booleene**; permit evaluarea unor expresii booleene, iar rezultatul este memorat de o variabilă booleeană;

8) **Funcţii** descrise prin tabele sau *prin segmente* de dreaptă (liniare pe porţiuni);

9) **Funcţii analitice** descrise prin expresii mai complicate; (ele există numai in SIMUB şi au rolul de a facilita simularea diverselor variabile aleatoare prin *metoda inversa*);

Entităţile statistice, permit colectarea unor **statistici** sau **estimaţii** privind VE sau PE; printre acestea se remarcă :

10) **Cozile** care sunt entităti statistice ce memorează tranzacţiile intârziate in sistem;

11) **Tabele de frecvenţe** care descriu **histograme** ale VE;

12) **Tabele de frecventa bidimensionale,** sau **tabele de contingenţă**; (acestea sunt disponibile numai in SIMUB);

Entităţi de referinţă care memorează anumite informaţii pe care le doreşte utilizatorul şi anume:

13)**Cuvinte de salvare** care memorează valori ce corespund câte unui cuvânt de memorie al calculatorului;

14) **Matrice de cuvinte de salvare** care memorează matrici;

Entităţi de tip lanţ,care sunt de două tipuri:

15) **Lanţuri ale utilizatorilor** in care se pot depune sau scoate tranzaţii după dorinţa utilizatorului; (Există şi lanţuri ale sistemului, manipulate intern de către GPSS, care nu pot fi accesate de utilizator).

16) **Grupuri** care **separă** tranzacţii (cu anume scopuri de prelucrare dorite de utilizator, sau cu anumite proprietăţi).

NOTA. Toate entităţile trebuie definite/identificate şi declarate de către utilizator la inceputul programului. Pentru fiecare entitate se alocă memorie; la instalarea limbajului se alocă şi un **număr maxim** de entităţi care se pot utiliza intr-un program.

Entităţile se **invocă** prin instrucţiuni cu structură fixă. (Limbajul este un **interpreter**).

Limbajul permite descrierea modulară a unui model de simulare general (vezi schema generală 1.1); cel mai important este modulul principal (**ciclul de bază**). Ieşirile sunt standard; se afişează/scriu valori ale atributelor, statistici, tabele de frecvenţă, etc.

Limbajul GPSS este un **interpreter**, ceeace inseamnă că fiecare instrucţiune scrisă corect (şi recunoscută ca atare de către GPSS) se execută imediat.

• **Structura instrucţiunii GPSS.** Instrucţiunea GPSS are un format fix (vezi Figura 1.5).

1) Blocurile descriu acţiuni (sunt entităţi **active** spre deosebire de tranzacţii care sunt entităţi pasaive, in sensul că ele sunt *mişcate* prin model ca urmare a acţiunii blocurilor.

Nume simbolic (Etichetă)	Numele blocului (Cunânt cheie)	Argumente/parametri (separate prin virgulă)
Opţional pt.referiri	Verb imperativ care desemnează acţiunea blocului	A,B,C,...

Fig 1.5. Structura instrucţiunii GPSS.

Tipurile de blocuri sunt:

1a) *Blocuri de actiune*: SEIZE/RELEASE (tranzacţia curentă ocupă sau eliberează o facilitate); PREEMPT (tranzacţia intrată in acest bloc poate ocupa o staţie de serviciu indicată de parametrul-argument sau o poate **prelua** dacă este ocupată de altă tranzacţie); ENTER/LEAVE (ocupă sau eliberează o poziţie dintr-o multistaţie); QUEUE/DEPART (intră in coadă sau pleacă din coadă); LINK/UNLINK (introduce sau

scoate tranzacţia dintr-un lanţ al utilizatorului); SPLIT (descompune tranzacţia in mai multe tranzacţii care formează o *familie*); ASSEMBLE/GATHER (unifică tranzacţiile dintr-o familie fără, sau cu păstrarea atributelor de bază din familie); MATCH (sunt 2 blocuri conjugate; ele sincronizează deplasarea tranzacţiilor care intâlnesc aceste blocuri); ADVANCE (intârzie tranzacţiile; simulează de ex. **durata de serviciu** a tranzacţiei); BUFFER (inseamnă prelucrarea tranzacţiilor in ordinea priorităţilor); JOIN/REMOVE (introduc sau scot tranzacţia intr-un grup); SCAN (verifică dacă există o tranzacţie in grup cu o anumită proprietate);

1b) *Blocuri de creare şi distrugere de tranzacţii:* GENERATE/TERMINATE (generarea se face cu **generatori specializaţi de numere aleatoare**, tranzacţia putând fi prevăzută cu priorităţi);

1c) *Blocuri de control logic al tranzacţiilor:* TEST (controlează dacă 2 atribute ale tranzacţiei satisfac o condiţie); TRANSFER (asigură transfer condiţionat sau nu al fluxului de execuţie la blocul indicat prin argument); GATE (modifică drumul tranzacţiilor in funcţie de condiţii referitoare la atribute ale facilitaţilor, multistaţiilor, comutatorilor logici sau tranzacţiilor); EXAMINE (modifică fluxul in funcţie de apartenenţa la un grup); LOOP (repetă execuţia de la blocul menţionat ca argument până la blocul LOOP);

ATENŢIE: fluxul tranzacţiei se modifică şi prin blocurile:PREEMPT, LINK/UNLINK, REMOVE, ALTER, SCAN, SELECT.

1d) *Blocuri de modificare a caracteristicilor tranzacţiilor şi a valorilor unor entităţi de referinţă:* ASSIGN (atribuie valori numerice parametrilor tranzacţiilor şi/sau le modifica; când se generează o tranzacţie poate fi prevăzută cu "locuri" pentru un număr de maximum 12 parametri referibili de către utilizator; numărul parametrilor poate fi standard sau precizat de utilizator la generare); INITIAL (iniţializează cuvintele sau matricile de cuvinte păstrate); PRIORITY (modifică priorităţile; lucrează in legătură cu blocul BUFFER); LOGIC (pozitionează pe *true* sau *false* un comutator logic ce poate juca rol de *semafor*); MARK (utilizat pentru a marca intr-un cuvânt special asociat fiecărei tranzacţii, valoarea ceasului);
SAVEVALUE/MSAVEVALUE (memorează intr-un cuvânt/matrice valoarea unui atribut standard numeric precizat); COUNT (determină numărul de entităţi ce satisfac o anumită condiţie şi-l memorează intr-

un parametru al tranzacţiei specificat ca argument al lui COUNT); SELECT (selectează prima entitate dintr-o gamă de entităţi ce satisfac o anumită condiţie); ALTER (modifică condiţionat sau nu, parametri sau prioritatea uneia sau mai multor tranzacţii dintr-un grup putând modifica opţional şi drumul tranzacţiei); HELP (permite includerea unor proceduri de calcul ale utilizatorului; este un bloc pretenţios căci necesită interfaţa intre GPSS şi limbajul in care este scrisă procedura; in SIMUB se foloseau subrutine FORTRAN!);

1e) *Blocuri pentru obţinerea de statistici:* TABULATE (pentru construirea histogramei);(In SIMUB se foloseşte si BTABULATE pentru construirea tabelelor de contingenţă, adică *histograme bidimensionale*).

1f) *Blocuri pentru listări:* PRINT (permite scrierea parţială a unor statistici; cele mai multe statistici se scriu automat de GPSS la sfârşitul simulării);

2a) *Tranzacţiile* se generează cu GENERATE şi se distrug cu TERMINATE. La generare tranzaţiile primesc automat nişte parametri interni şi un număr de parametri (limitat de ex la 100) declaraţi şi care pot fi accesaţi de utilizator; in funcţie de implementare, un program GPSS poate utiliza un număr maxim de tranzacţii (acesta este un dezavantaj al limbajelor de simulare datorat faptului că limbajul **gestionează agenda**, GPSS rezolvând astfel el insuşi detaliile de programare pe care intr-un limbaj evoluat le rezolvă utilizatorul).

3a) *Staţiile* de serviciu pot fi ocupate de o singură tranzacţie la un moment dat.

4a) *Depozitele* sau multistaţiilec au o capacitate declarată in prealabil şi in ele pot intra mai multe tranzacţii (cât permite capacitatea).

5a) *Comutatorii logici* sunt de fapt variabile booleene, ce trebuie iniţializate (precizează condiţii satisfăcute de "echipoamente").

Toate entităţile de echipament sunt identificate printr-un număr (intreg). Ele posedă atribute standard numerice sau logice.

6a) *Variabilele* au cuvântul cheie VARIABLE, au un nume şi o expresie aritmetică (de dimensiune limitată) care conţine atribute standard numerice, cuvinte de salvare, parametri, etc.

7a) *Variabilele booleene* BVARIABLE, sunt construite analog (de către utilizator) folosind atribute standard logice, inclusiv comutatori logici).

8a) 9a) *Funcţia* (FUNCTION) desemnează o funcţie de o variabilă dată printr-o listă sau functie liniară precizându-se argumentul prin care aceasta este referită.

10a) *Coada* este o entitate statistică pentru care se reţine automat *lungimea medie* şi *lungimea maximă* care se listează la sfârşitul simulării.

11a) 12a) *Tabela de frecvenţă* se defineşte la inceputul programului cu TABLE şi reprezintă o histogramă a unei variabile de ieşire din model, de regulă atribut numeric.La definire se precizează argumentul tabelat şi numărul de clase de frecxvenţe.

13a) 14a) *Cuvintele* sau *matricile de cuvinte* păstrate sunt invocate prin INITIAL, SAVEVALUE sau MSAVEVALUE şi ele se folosesc pentru a reţine anumite valori pe parcursul simulării. Sunt referite printr-un număr de identificare.

15a) *Lanţurile sistemului* sunt: lanţul evenimentelor curente (LEC), lanţul evenimentelor viitoare (LEV), lanţuri ale tranzacţiilor intrerupte, lanţuri ale tranzacţiilor in aşteptare pentru sincronizări, lanţuri de intârziere (asociate echipamentelor). Orice tranzacţie *activă* la un moment dat se găseşte intr-un lanţ. Tranzacţiile care se deplasează in sistem se găsesc in lanţul evenimentelor curente. Mecanismul de actualizare a lanţului evenimentelor curente şi al celor viitoare este cel cunoscut (descris prin relaţia dintre A, AEC şi AEV din sectiunea 2.1). Tranzaţiile terminate (prin blocul TERMINATE) sunt distruse; blocul START precizează printr-un argument al sau câte tranzacţii se vor prelucra (termina).Prelucrarea tranzacţiilor din LEC inseamnă transferul lor in alt lanţ sau distrugerea. Tranzacţiile intârziate (de ex prin ADVANCE) sunt introduse in LEV.

Lanţurile sistemului sunt controlate automat de către sistemul GPSS.

Lanţurile utilizatorului (LU) sunt definite de acesta şi ele (sau atribute ale lor) pot fi referite in programul GPSS. Tranzacţiile pot fi introduse sau scoase din LU prin LINK şi UNLINK şi ele se folosesc la implementarea unor *discipline de serviciu* (in cazul prelucrarii tranzactiilor din cozi).

16a) *Grupurile* oferă utilizatorului un instrument de *clasificare* a tranzacţiilor care au anume proprietăţi. De ex. toate staţiile care la un moment dat depăşesc un anumit grad de utilizare pot fi făcute membre ale unui grup.

Un program GPSS este o listă ordonată de BLOCURI ale căror argumente se referă la diverse entităţi.

In cele ce urmează sunt prezentate patru programe GPSS care se bazează pe câteva din entităţile şi instrucţiunile uzuale ale acestui limbaj.

1.3.1 Exemple de programe GPSS.

• E1. Model de simulare pentru un sistem de aşteptare cu o staţie.

Programul GPSS pentru acest model este:

```
001     GENERATE     10,2      ;Sosiri aleatoare la 10 ± 2 minute
003     SEIZE        FRIZER    ;Ocupă staţia de serviciu "FRIZER"
005     ADVANCE      12,3      ; Durata servirii 12 ± 3 minute
007     RELEASE      FRIZER    ; Eliberează staţia
009     TERMINATE    1         ; Ieşire din sistem un client
```

Comentariu. Cifrele din faţa tabelului reprezintă etichetele instrucţiunilor/blocurilor, iar textele de la terminarea liniilor reprezintă comentarii. După denumirile blocurilor apar parametri acestora.

Modelul nu este complet deoarece in faţa blocului SEIZE pot să apară aşteptări de aceea inaintea acestui bloc trebuie introdus un bloc QUEUE şi după el un bloc DEPART ca mai jos (etichetele indică locul unde se introduc aceste blocuri respectând ordinea).

```
002     QUEUE     COADA1   ; Tranzacţia ocupă COADA1
003     SEIZE              .......
004     DEPART    COADA1   ;Tranzacţia pleacă,se actualizează aşteptările;
```

Pentru a prelucra 200 tranzacţii se va da la inceput comanda

$$000 \quad START \quad 200$$

La sfârşitul simulării apelând procedura GPSSREPT.EXE se va afişa un *raport final* privind simularea care conţine:

- durata totală a prelucrării tranzacţiilor(durata simulării);
- gradul de ocupare a staţiei de serviciu (media şi maxima);
- durata medie a aşteptărilor, etc.

• E2. Model de simulare pentru un sistem de aşteptare cu staţii paralele.

Modelul de mai jos simulează un sistem de aşteptare cu 3 staţii paralele (ghişee ale unei bănci) in care serviciul se realizează cu disciplina FIFO (First In First Out) adică serviciul se execută in ordinea sosirii clienţilor şi nu se consideră priorităţi ale unor clienţi. Clientul va trece pe la staţia multiplă de servire (depozit), dar el va ocupa o singură staţie care-l serveşte (STORAGE 3). In final el va trece şi la staţia CASIER unde primeşte un alt serviciu (primeşte sau plăteşte banii pe baza formelor elaborate de ghişeul parcurs anterior!). Desigur, pentru executarea modelului care urmează ar trebui adăugată comanda START corespunzătoare şi apelat in final modulul de ieşire GPSSREPT.EXE. Modelul este dat de programul:

```
GHISEE      STORAGE     3          ; Declararea dimensiunii multistaţiei
001         GENERATE    4,1        ;Sosesc clienţi
002         QUEUE       COADA      ;Clientul in coada căci dorim statistici
003         ENTER       GHISEE     Ocupă un loc in GHISEE (dacă esistă!)
004         DEPART      COADA      ;Clientul pleacă din coada (spre servire!)
005         ADVANCE     15,3       ;Clientul este servit (durata de servire!)
006         LEAVE       GHISEE     Se eliberează un log (un ghişeu)
007         SEIZE       CASIER     ; ocupă locul la casier;
007         ADVANCE     2          ; Servirea clientului este 2 minute
008         RELEASE     CASIER     ; Eliberează casierul(clientul pleacă)
009         TERMINATE   1          ;Plecare client.
```

• **E3. Model cu staţii paralele şi preferinţe.** Se modelează serviciul la o frizerie unde lucrează 3 frizeri (trei staţii de serviciu!):Figaro, Gică şi Tică. Clienţii care sosesc sunt primiţi in proporţie de 60% de Figaro (care este preferat!) iar restul de 40% sunt serviţi de ceilalţi doi, dar din aceştia 50% merg la Gică şi 50% merg la Tică. Modelul nu conţine comenzile de START şi cele de ieşire.

```
001         GENERATE    6,1        ;Sosesc clienţi
002         TRANSFER    .4,,ALTII  ;60% clienţi merg la Figaro
003         QUEUE       COADAFIG   ;Intră in coada lui Figaro
004         SEIZE       COADAFIG   ; Clientul se duce la Figaro
005         DEPART      COADAFIG   ;Se culeg date despre ac.coadă
006         ADVANCE     8,1        ;Clientul e servit
007         RELEASE     FIGARO     ;Figaro devine liber
008         TRANSFER    ,CASA      ; Clientul merge să plătească
ALTII       TRANSFER    ,LAGICA    ;Tică şi Gică sunt la fel
010         SEIZE       TICA       ; 50% din clienţi merg la Gică
011         ADVANCE     12,2       ; Tică serveşte mai incet ca Figaro
012         RELEASE     TICA       ; Tica este eliberat
```

```
013       TRANSFER   ,CASA      ;Clientul plăteşte
LAGICA    SEIZE      GICA       ;50% din 40% clienţi merg la Gica
015       ADVANCE    12,2       ; Gică serveşte ca şi Tică
016       RELEASE    GICA       ; Se eliberează Gica
CASA      SEIZE      CASIERA    ; Se ocupă staţia CASIERA
018       ADVANCE    1          ;Plata intr-un minut
019       RELEASE    CASIERA    ; Se eliberează CASIERA
020       TERMINATE  1          ;Tranzacţia pleacă
```

Nota. Cele trei staţii sunt folosite explicit, nu ca multistaţie. Folosim coada numai la Figaro căci numai acolo (presupunem!) ne interesează statistici finale.

In programele GPSS etichetele sunt opţionale şi ele pot fi mnemonice (care denumesc obiecte modelate ca de ex. LAGICA sau ALTII), sau pot fi constante intregi care sugerează ordinea instrucţiunilor; sunt obligatorii etichetările blocurilor la care se referă alte blocuri (vezi TRANSFER mai sus).

Parametri unor entităţi pot desemna *nume simbolice* (de ex. staţia GICA sau coada COADAFIG).

• **E4. Un Model complex.** Acest model este de complexitate ridicată şi instrucţiunile ce descriu peogramul de simulare $GPSS$ sunt prezentate mai jos.. El simulează in $GPSS/PC$ activităţile de reparaţie a unui automobil intr-un *service* care are ateliere separate de reparaţii *motor, caroserie şi tinichigerie.* Unitatea de timp este *ora* iar repartiţia de probabilitate a duratelor de reparaţie este exponenţială şi funcţia de repartiţie este derscrisă de funcţia *continuă* numită $XPDIS$ dată prin instrucţiunea '50' de mai jos care are ca argument generatorul de numere aleatoare $RN2$, este de tip *continuu* şi este descrisă prin coordonatele a 24 puncte, $C24$. Instrucţiunea 60 descrie o histogramă a timpului $M1$ petrecut de tranzacţii (automobilele ce se repară) in sistem. Blocul GENERATE 70 simulează sosiri exponenţiale cu media 48 folosind metoda inversă (vezi capitolul următor) aplicată funcţiei de repartiţie definită in preambulul programului. (Funcţia este descrisă prin coordonatele unor puncte prin care trece graficul ei şi care puncte sunt date prin perechile de coordomate, despărţite prin *slash* /.

In continuare programul descrie activităţile ce au loc in sistem: blocul 80 include in coadă automobilul sosit (prin GENERATE); blocul

90 simulează ocuparea staţiei *SEFATELIER*; blocul 100 simulează (prin 100 $ADVANCE$) activitatea de *constatare* a şefului de atelier care are repartiţie uniformă de medie 2 ± 1; apoi blocul 110 $RELEASE$ simulează eliberarea şefului de atelier, după care blocul 120 $SEIZE$ ocupă mecanicul $MEC1$ care realizează demontarea blocului motor de pe caroserie. In continuare blocul 150 $SPLIT$ trimite motorul la mecanici denumiţi in model $ATELMEC$ (blocul $SPLIT$ *ramifică* traseul tranzacţiilor). In continuare comentariul $160 * traseulcaroseriei$ indică faptul că blocurile 170-400 descriu traseul caroseriei prin intreprinderea de service,traseu care se termină la blocul
410 $ATELMEC\ SEIZE\ MEC1$ care prin eticheta $ATELMEC$ indică faptul că motorul (desprins de caroserie prin blocul 130) a fost transferat in atelierul mecanic. In secvenţa de instrucţiuni $170 - 400$ (care simulează activităţile din secţia *Caroserie,* se remarcă blocurile 230 şi 290 care sunt blocuri $MATCH$ conjugate şi care semnifică faptul că trebuie sincronizate activităţile lui $OM2$ (ajutor de sudor care demontează uşile) cu cele ale lui $OM1$ (care realizează sudura propriuzisă). In secvenţa de instrucţiuni referitoare la caroserie activităţile descrise sunt explicate prin comentarii şi prin etichetările cu mnemonice ale blocurilor; majoritatea blocurilor au fost deja intâlnite şi in cadrul celorlalte exemple de programe $GPSS$. Menţionăm ca deosebite blocurile 260 (transfer necondiţionat la blocul 320 $ASSEMBLE$ care asamblează uşile de corpul caroseriei) şi 400 (transfer necondiţionat la blocul 450 $MONFINAL$ care realizează montajul final al motorului şi caroseriei). In instrucţiunile finale ale programului $GPSS$ remarcăm secvenţa $410 - 440$ care descrie traseul motorului şi secvenţa $450 - 500$ care descrie asamblarea finală a componentelor automobilului. O instrucţiune nouă aici este 490 $TABULATE$ care inregistrează (pe parcursul simulării) frecvenţele in tabela $TABCLASE$ definită la inceput.

Descriem in continuare programul sursă $GPSS$. La inceput sunt descrise instrucţiunile de definire a funcţiei, după care urmează blocurile ce simulează activităţile in service-ul auto.

```
; GPSS/PC Program File E4.GPS. (V2, #39399) 08-02-2003 04:13:29
10  * Modelul reparatiei unui automobil, in ateliere separate
20  * pentru motor si pentru tinichigerie(, urmata de vopsire).
30  *    -Unitatea de timp a modelului este ora.-
```

```
40   * Repartitia exponentiala standard
50   XPDIS     FUNCTION     RN3,C24
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2/.75,1.38
.8,1.6/.84,1.83/.88,2.12/.9,2.3/.92,2.52/.94,2.81/.95,2.99/.96,3.2
.97,3.5/.98,3.9/.99,4.6/.995,5.3/.998,6.2/.999,7/.9998,8

60   TABCLASE     TABLE      M1,40,20,20     ;Pt.histogr. timp in sistem.
70                GENERATE   48,FN$XPDIS     ;Sosiri expon., cu media 48 ore.
80                QUEUE      DURATA          ;Incepe inregistrarea.
90                SEIZE      SEFATELIER      ;Pt.diagnostic & eval.cost.
100               ADVANCE    2,1             ;Durata constatarii.
110               RELEASE    SEFATELIER
120               SEIZE      MEC1            ;Demont.motor de pe caroserie.
130               ADVANCE    3,1             ;Durata demont. de catre Mec1
140               RELEASE    MEC1
150               SPLIT      1,ATELMEC       ;Trimitere motor la mecanici
160  *Tras.caroserie:
170               SEIZE      SUD2            ;Ajutorul de tinichigiu (sudor)
180               ADVANCE    2,1             ;demonteaza usi, capote, aripi.
190               RELEASE    SUD2
200               SPLIT      1,USI           ;La ajutor (Sud2), pt.usi, etc.
210               SEIZE      SUD1
220               ADVANCE    14,FN$XPDIS     ;Durata suduri corp (cu Sud1)
230  OM1          MATCH      OM2             ;Verificare potriviri balamale
240               ADVANCE    8,FN$XPDIS      ;Alte suduri corp (Sud1)
250               RELEASE    SUD1
260               TRANSFER   ,MONCAROS
270  USI          SEIZE      SUD2            ;Traseul usilor demontate:
280               ADVANCE    12,FN$XPDIS     ;Durata suduri usi+capote(Sud2)
290  OM2          MATCH      OM1             ;Verific la balamale cu corpul.
300               ADVANCE    9,FN$XPDIS      ;Alte suduri usi (Sud2).
310               RELEASE    SUD2            ;Ajutorul tinichigiu a terminat.
320  MONCAROS     ASSEMBLE   2               ;Usile cu corpul caroseriei.
330               SEIZE      SUD1
340               ADVANCE    2               ;Mont.la loc caroserie + reglaj
350               RELEASE    SUD1            ;usi de catre tinich.principal
360               SEIZE      VOPSITOR        ;Secventa de vopsire.
370               ADVANCE    24,3
380               RELEASE    VOPSITOR
390               ADVANCE    1               ;Transportul il face altcineva.
400               TRANSFER   ,MONFINAL       ;Pt.asamblare cu motorul.
410  * Traseu motor:
420  ATELMEC      SEIZE      MEC1
430               ADVANCE    20,FN$XPDIS     ;Reparatii, facute de Mec1.
```

```
440                     RELEASE     MEC1
445   * Reasambl.finala
450   MONFINAL          ASSEMBLE    2           ;Motorul cu caroseria.
460                     SEIZE       MEC1        ;El face asamblarea.
470                     ADVANCE     2
480                     RELEASE     MEC1
490                     TABULATE    TABCLASE    ;Timp in sistem, pe clase
500                     DEPART      DURATA      ;Inchei inreg.timpului
510                     TERMINATE   1           ;Iesire din sistem.
```

Exemplele prezentate sunt simple, alese gradual, de la simplu la complex şi au desigur un scop didactic. Ele ilustrează utilizarea celor mai importante blocuri şi instrucţiuni ale imbajului *GPSS*. Pentru rezolvarea unor probleme practice complexe este necesară consultarea unor decumetaţii complete privind limbajul [14].

In ultimul capitol vor fi prezentate modele de simulare asemănătoare primelor două de mai sus, dar care sunt implementabile intr-un limbaj evoluat; scopul tratării acelor modele va fi acela de a se ilustra cum se face validarea unui model de simulare, ceea ce nu s-a făcut aici, in cazul modelelor implementate in GPSS.

Cap. 2

Numere aleatoare

2.1 Noţiuni introductive

Vom aminti mai intâi câteva notiuni de bază. Presupunem că X este o *variabilă aleatoare* şi fie $F(x) = P(X < x)$ *funcţia sa de repartiţie.* Densitatea de repartiţie, când aceasta există, este derivata funcţiei de repartiţie, adică $f(x) = F'(x)$. Funcţia F satisface proprietăţile: $F(-\infty) = 0$, $F(+\infty) = 1, F(a) \leq F(b)$ dacă $a < b$. Intre $F(x)$ şi $f(x)$ are loc relaţia

$$F(x) = \int_{-\infty}^{x} f(u)du.$$

De obicei X reprezintă o caracteristică a unei mulţimi de obiecte care variază aleator de la un obiect la altul; acea mulţime se numeşte *populaţie statistică.* Dacă luăm la intâmplare un număr n de obiecte din populaţie si considerăm valorile $X_1, X_2, ..., X_n$ ale caracteristicii X ce corespund acestor obiecte spunem că aceste valori determină o *selecţie de volum* n asupra lui X. In statistica matematică selecţia $X_1, X_2, ..., X_n$ este considerată ca fiind o mulţime de variabile aleatoare independente şi identic repartizate ca şi X (aceasta numindu-se *selecţie bernouliană*). *Independenţa* a două variabile aleatoare se defineşte astfel: X este o variabilă aleatoare *independentă* de Y dacă $P(X < x, Y < y) = P(X < x)P(Y < y)$ sau, dacă notăm $F(x, y) = P(X < x, Y < y)$ funcţia de repartiţie *comună* a variabilelor X şi Y şi notăm $F_1(x) = P(X < x)$, $F_2(y) = P(Y < y)$ funcţiile de repartiţie *marginale,* ale lui X respectiv

Y, atunci *condiţia de independenţă* se scrie

$$F(x,y) = F_1(x)F_2(y). \tag{2.1}$$

Să observăm că se face distincţie intre variabila aleatoare X şi o valoare x a acesteia (care reprezintă un număr real fixat!).

Valorile efectiv obţinute prin procesul de selecţie constituie o mulţime de n numere care se spune că reprezintă o realizare a selecţiei bernuliene. Deoarece fiecare din aceste numere pot să reprezinte oricare valoare a lui X se considera că valorile de selecţie (numite şi variabile de selecţie) sunt independente si identic repartizate ca şi X.

Când pentru o variabilă aleatoare X se cunoaşte F sau f spunem că se cunoaşte *repartiţia de probabilitate* sau *repartiţia statistică* a lui X. Exsistă multe tipuri de repartiţii de probabilitate. Cele mai importante vor fi introduse in acest capitol, dar şi in capitolele următoare.

• **Repartiţia uniformă.** Una din repartiţiile de probabilitate importante, dar care este *naturală,* este *repartiţia uniformă pe un interval* $[a,b]$ (vezi [2, 3, 5, 6, 7, 10, 11]) care are densitatea de repartiţie de forma

$$g(x) = \begin{cases} k, daca\, x \in [a,b] \\ 0,\ in\, rest \end{cases}, \int_{-\infty}^{\infty} g(x)dx = 1 \rightarrow \quad k = \frac{1}{b-a}. \tag{2.2}$$

Variabila V având densitatea de repartiţie (2.2) se spune că este repartizată *uniform* pe $[a,b]$. Deci toate valorile variabilei V sunt *egal probabile.* Funcţia de repartiţie corespunzând densităţii (2.2) este

$$G(x) = \int_{-\infty}^{x} g(u)du = \begin{cases} 0, & daca \quad x < a \\ \frac{x-a}{b-a} & daca \quad x \in [a,b] \\ 1, & daca \quad x > b. \end{cases} \tag{2.3}$$

O repartiţie uniformă interesantă este repartiţia uniformă pe $[0,1]$ aşa cum se va vedea in continuare.

Să notăm cu U variabila aleatoare uniformă pe $[0,1]$, pe care o vom numi pe scurt variabilă uniformă $0-1$. Densitatea de repartiţie şi funcţia de repartiţie a lui U sunt respectiv

$$f(x) = \begin{cases} 1,\ daca\, x \in [0,1] \\ 0,\ in\, rest, \end{cases} \quad F(x) = \begin{cases} 0,\ daca\, x < 0, \\ x,\ daca\, x \in [0,1] \\ 1, daca\, x > 1. \end{cases} \tag{2.4}$$

In Fig. 2.1 se prezintă graficele densităţilor şi funcţiilor de repartiţie uniforme f, F, g, G. Graficele densităţilor uniforme ilustrează intuitiv faptul că valorile variabilelor uniforme sunt *egal probabile,* adică au aceleaşi şanse de apariţie intr-un experiment.

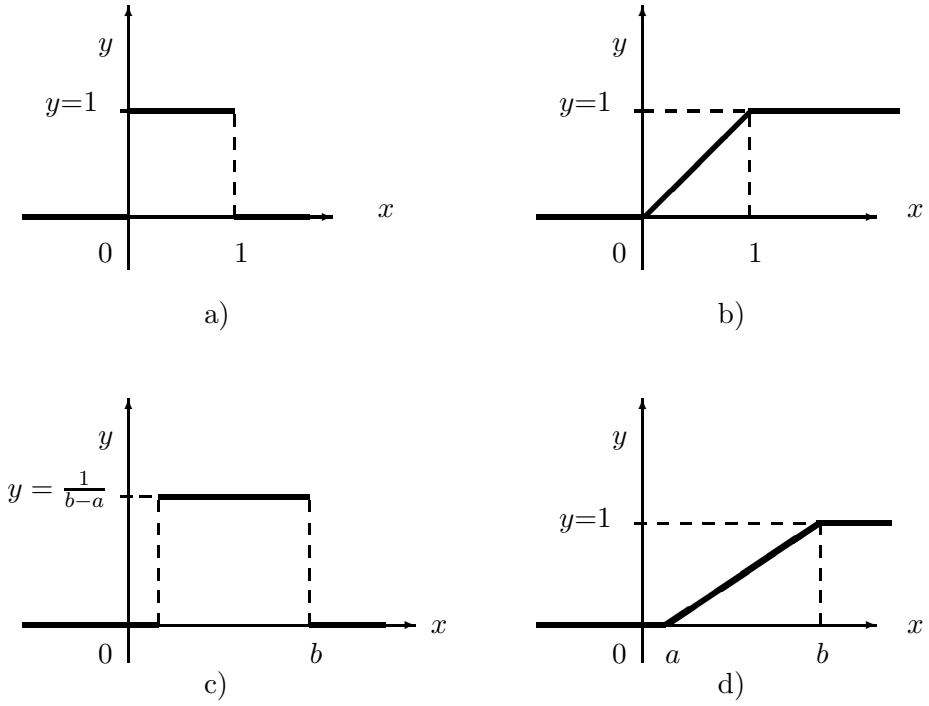

Fig. 2.1. Repartiţii uniforme
a) Densitatea uniformă $0-1$
b) Funcţia de repartiţie uniformă $0-1$
c) Densitatea uniformă pe $[a, b]$
d) Funcţia de repartiţie uniformă pe $[a, b]$

Valorile de selecţie asupra variabilelor aleatoare uniforme se numesc [2, 5, 6, 7, 10, 11] *numere aleatoare.* Aşa cum se va vedea mai jos, nu este posibil să se producă cu calculatorul, printr-un algoritm, secvenţe de numere aleatoare care să fie uniform repartizate pe intervalul $[0, 1]$ şi care să fie *independente stochastic.* De aceea numerele produse cu calculatorul, vor fi numite *numere pseudoaleatoare* şi ele vor putea fi folosite drept numere aleatoare dacă au

un comportament cât mai aleator (vezi \$2.3). Un algoritm care produce un număr aleator (pseudoaleator) se numeşte *generator* de numere aleatoare (pseudoaleatoare). Iterând un generator se poate obţine o selecţie (secvenţă) de numere aleatoare. Când este nevoie de numere aleatoare cu cât mai bune calităţi se pot aplica metode care combină doi sau mai mulţi generatori, rezultând un alt generator care produce secvenţe de numere pseudoaleatoare mai bune. In multe aplicaţii sunt insă suficient de bune numerele produse de generatori simpli, aşa cum se va vedea mai jos.

2.2 Necesitatea simulării numerelor aleatoare

Următoarea teoremă [2, 3, 5, 10] stabileşte legătura intre repartiţia uniformă $0-1$ şi repartiţia uniformă pe un interval $[a,b]$ oarecare.

Teorema 2. 1 *Dacă U este o variabilă uniformă $0-1$ atunci $V=a+(b-a)U$ este o variabilă uniformă pe $[a,b]$ şi reciproc, dacă V este o variabilă aleatoare uniformă pe $[a,b]$ atunci variabila*

$$U=\frac{V-a}{b-a}$$

este uniformă $0-1$.

Să vedem acum dece sunt necesare numerele aleatoare in simulare. Fiind dată o variabilă aleatoare oarecare X, pentru care se cunoaşte funcţia de repartiţie F, este adevărată următoarea propoziţie [2, 3, 10] care evidenţiază importanţa repartiţiei uniforme $0-1$, adică importanţa numerelor aleatoare.

Teorema 2. 2 *Variabila aleatoare $F(X)$ este uniformă $0-1$ iar dacă notăm cu F^{-1} inversa funcţiei F atunci $F^{-1}(U)$ are funcţia de repartiţie F (sau cu alte cuvinte $F^{-1}(U)=X$).*

Demonstraţie. Demonstrăm partea a doua a teoremei deoarece aceasta joacă un rol esenţial in simulare.

Funcţia de repartiţie a lui $F^{-1}(U)$ se scrie

$$P(F^{-1}(U)<x)=P(F(F^{-1}(U))<F(x))=P(U<F(x))=F(x).$$

Ultima egalitate rezultă din (2.3) cu $a=0$ şi $b=1$. Teorema este demonstrată. Ea este cunoscută sub numele de *teorema lui Hincin.*

Din teorema 2 rezultă următoarea consecinţă practică: dacă am putea produce valorile de selecţie $U_1, U_2, ..., U_n$ asupra lui U şi am cunoaşte funcţia de repartiţie F a lui X, atunci am putea produce valorile de selecţie $X_1, X_2, ..., X_n$ asupra lui X cu formula $X_i = F^{-1}(U_i), 1 \leq i \leq n$. Dacă funcţia F^{-1} se poate calcula cu un algoritm atunci valorile de selecţie X_i ar putea fi produse (*generate*) cu calculatorul folosind următorul algoritm

Metoda inversă. (Algoritm)
repetă de n **ori următoarele instrucţiuni**
generează o valoare de selecţie U **uniformă** $0-1$;
calculează $X = F^{-1}(U)$**;**

Spunem că acest algoritm *simulează* o *selecţie de volum n* asupra lui X. Dar paşii cei mai importanţi ai algoritmului sunt ultimii doi paşi care produc o valoare de selecţie X, căci dacă putem genera numere U uniforme $0-1$ si *independente* atunci prin iterare, conform primului pas, am produce valorile selecţiei de volum n asupra lui X. Din această cauză, pentru simularea diverselor variabile *ne vom concentra asupra problemei simulării unei singure valori de selecţie* care să folosească un algoritm astfel incât prin iterare să poată produce selecţia care ne interesează.

Revenind la metoda inversă, dacă am putea genera valori de selecţie $U_i, 1 \leq i \leq n$ uniforme $0-1$ independente, atunci şi $X_i = F^{-1}(U_i)$ ar fi independente.

Un algoritm pentru generarea (simularea) unei selecţii $U_i, 1 \leq i \leq n$ de variabile uniforme $0-1$ va trebui deci să producă in primul rând un număr uniform $0-1$, iar prin iterări acelaşi algoritm să fie in măsură să producă selecţia de volum n; deci selecţia se va produce printr-o relaţie iterativă de forma $U_{i+1} = g(U_i)$, iar algoritmul trebuie să permită producerea de numere U_i independente stochastic. Valorile de selecţie U_i se numesc *numere aleatoare uniforme* sau simplu *numere aleatoare.*

2.3 Metode congruenţiale liniare

Fie X o variabilă aleatoare discretă repartizată uniform având repartiţia

$$X : \begin{pmatrix} 1, & 2, & ..., & m \\ \frac{1}{m}, & \frac{1}{m}, & ..., & \frac{1}{m} \end{pmatrix}, m < \infty, \tag{2.5}$$

unde $m \in N^+$. Conform teoremei 2.1, variabila U uniformă $0-1$ se poate obţine astfel

$$U = \frac{X}{m} \in (0,1) \qquad (2.5')$$

unde in ultima relaţie impărţirea se execută *in real.*

Valori de selecţie asupra variabilei X se pot obţine prin relaţia *congruenţială liniară* [2, 5, 6, 7, 10]

$$X_{i+1} \equiv (aX_i + c)(mod\, m), \qquad (2.6)$$

care spunem că defineşte *generatorul mixt congruenţial* liniar (X_0, a, c, m) unde cele patru constante sunt numere naturale date.

Pentru ca generatorul (2.6) sa producă *numere aleatoare,* trebuie alese constantele (X_0, a, c, m) astfel incât să satisfacă condiţiile:

1^0. Numerele X_i să reprezinte o repartiţie uniformă de forma (2.5); acest lucru se realizează dacă sirul $\{X_i\}$ are o *perioadă* λ mare adică dacă cel mai mic număr λ pentru care $X_i = X_{i+\lambda}$ are loc pentru un λ cât mai mare.Acest lucru se intâmplă dacă cele patru constante se aleg astfel incât relaţia (2.6) să producă toate resturile modulo m, adică numerele $0, 1, ..., m-1$, iar m este mare.

2^0. Numerele X_i să fie independente stochastic; acest lucru nu este practic posibil deoarece X_{i+1} depinde de X_i conform relaţiei (2.6); este insă posibil ca numerele X_i, X_{i+1} să fie *slab dependente* stochastic. Această condiţie inseamnă că numerele respective au un *coeficient serial de corelaţie* ρ *apropiat de* 0, ρ definit astfel

$$\rho = \rho_1 = Corr(X_i, X_{i+1}) = \frac{E[X_i X_{i+1}] - E[X_i]E[X_{i+1}]}{\sqrt{Var[X_i]Var[X_{i+1}]}} \qquad (2.7)$$

$$Var[X_i] = E[X_i - E[x_i]]^2 = E[X_i^2] - \{E[X_i]\}^2$$

unde cu $E[Y]$ se notează valoarea medie a variabilei aleatoare Y.

Din motive impuse de aritmetica sistemelor de calcul, *modulul* m se ia de forma $m = p^e$ unde p este un număr prim [10]. Următoarea teoremă [10] precizează condiţii pentru alegerea constantei a care defineşte generatorul (2.6).

Teorema 2. 3 *Perioada maximă a generatorului (2.6) este* $\lambda = p^e$ *dacă şi numai dacă*

$$a \equiv 1(mod\ p) \quad cand \quad p > 2$$

$$a \equiv 1(mod\ 4) \quad cand \quad p = 2,\ \ 1 < a < p. \qquad (2.8)$$

O valoare a lui a poate fi $a = p^k + 1$, $2 \leq k \leq e$. Alegerea lui a folosind numai această teoremă poate insă să producă un şir de intregi $X_0, X_1, ..., X_{\lambda-1}$ care să nu fie aleator; acest şir care conţine sigur numărul 0, poate să conţină subşiruri de lungime mare care să fie sau crescătoare, sau descrescătoare, sau să prezinte o periodicitate mare. Presupunând că $X_0 = 0$ (ceea ce este desigur permis) se observă că sirul $\{X_n\}$ este de forma

$$X_n = \frac{(a^n - 1)c}{b} (mod\, m), \quad b = a - 1. \tag{2.6'}$$

Dacă in relaţia precedentă luăm $a = b + 1$ atunci (2.6') devine

$$X_n \equiv c(n + C_n^2 b + ... + C_n^s b^{s-1})(mod\ m) \tag{2.6''}$$

unde s este cel mai mic număr natural care satisface relaţia

$$b^s \equiv 0 (mod\, m). \tag{2.6'''}$$

Numărul s se numeşte *potenţa* generatorului mixt congruenţial. Din relaţia (2.6") rezultă că X_n este de forma unui polinom in b modulo m şi deci generatorul este cu atât mai bun cu cât potenţa sa s este mai mare. S-a constatat că din punct de vedere practic, potenţa trebuie să fie cel puţin 5.

Un caz interesant este acela când $m = 2^e$ unde e este apropiat de *cuvântul* calculatorului pe care se implementează generatorul (2.6). In acest caz se arată că o alegere bună a lui a este de forma $a = 2^{f_1} + 2^{f_2} + ... + 1, \quad f_1 > f_2 > ...$ Deci un generator pentru care m şi a satisfac condiţiile teoremei 3 generează un sir de numere aleatoare intregi de perioada λ mare şi care nu prezintă *regularităţi* (nu conţin subşiruri periodice de lungimi mari sau subşiruri cu monotonii periodice). Constanta c poate fi deocamdată arbitrar aleasă.

Să vedem acum cum putem alege constantele generatorului (2.6) astfel incât ρ să fie suficient de mic.

Numerele produse de (2.6) au o repartiţie de forma (2.5) in care valorile lui X sunt $0, 1, ..., m-1$. Deci in acest caz avem [10]

$$E[X] = \frac{\sum_{x=0}^{m-1} x}{m}, \quad Var[X] = \frac{\sum_{x=0}^{m-1} x^2}{m} - \{E[X]\}^2$$

$$\rho_1 = \rho = \frac{m \sum_{x=0}^{m-1} xs(x) - \left(\sum_{x=0}^{m-1} x\right)^2}{m \sum_{x=0}^{m-1} x^2 - \left(\sum_{x=0}^{m-1} x\right)^2}, \; s(x) = (ax + c)(mod\, m).$$

Evaluarea lui ρ se face cu dificultate. Se arată că

$$\rho \approx \frac{1}{a}\left(1-6\left(\frac{c}{m}\right)+6\left(\frac{c}{m}\right)^2\right) \pm \epsilon,\ \epsilon \approx \frac{a}{m}. \tag{2.9}$$

O valoare conciliantă a lui a, care să asigure o valoare mică a lui ρ este $a \approx \sqrt{m}$ iar din condiţia $\rho \approx 0$ se deduce că c/m trebuie să satisfacă ecuaţia

$$1-6x+6x^2=0 \quad adica \quad \frac{c}{m}=\frac{1}{2}-\frac{1}{6}\sqrt{3}=0.211324865. \tag{2.10}$$

Ultima relaţie dă o **condiţie pentru alegerea lui** c.

Notând $\rho_k = Corr(X_i, X_{i+k})$, se arată că $\rho_k \leq \rho_1$, $k > 1$, ceea ce asigură o dependenţă slabă a numerelor *depărtate* din secvenţa $X_1, X_2, ...$

In concluzie pentru ca generatorul (2.6) să fie bun trebuie să alegem un modul m cât mai mare, să selectăm un a astfel incât să satisfacă teorema 3 impreună cu o potenţă mai mare decât 5 şi să ia o valoare apropiată de $\sqrt{m}$, iar c să satisfacă (2.10). Numărul *de start* X_0, care se mai numeşte şi *sământa* generatorului poate fi orice număr natural mai mic decât m.

Pentru a produce un număr aleator intreg X, generatorul (2.6) necesită deci efectuarea unei inmulţiri şi a unei adunări şi apoi calcularea restului modulo m. Dacă am alege $c = 0$ atunci am obţine *generatorul multiplicativ congruenţial* $(X_0, a, 0, m)$ care are avantajul că necesită numai operaţia de inmulţire el fiind deci de forma

$$X_{n+1} \equiv (aX_n)(mod\, m). \tag{2.11}$$

In acest caz X_0 *trebuie sa fie neapărat pozitiv* căci altfel sirul produs de (2.11) ar avea toate elementele egale cu zero deci nu ar fi un sir de numere aleatoare uniforme intregi. Condiţia de *perioadă maximă* se modifică şi ea şi intr-un caz interesant din punct de vedere practic se exprimă prin teorema următoare.

Teorema 2. 4 *Perioada maximă a generatorului* $(X_0, a, 0, m)$ *se obţine când:*

(i) X_0 *este un intreg pozitiv prim cu* m;

(ii) a *este rădăcină primitivă modulo* m, *şi dacă* $p_i, 1 \leq i \leq t$ *sunt numere prime atunci perioada este*

$$\begin{gathered}\lambda(2^e) = 2^{e-2}, \quad daca \quad e \geq 3,\\ \lambda(p^e) = p^{e-1}(p-1), \quad daca \quad p > 2,\\ \lambda(p_1^{e_1}...p_t^{e_t}) = cmmmc(\lambda(p_1^{e_1})...\lambda(p_t^{e_t})).\end{gathered} \tag{2.12}$$

Desigur, generatorul recomandabil este cel de forma (2.11) deoarece necesită numai o operaţie de inmulţire. Există insă şi alti generatori de numere aleatoare intregi. Enumerăm in continuare câţiva.

2.4 Alţi generatori de numere uniforme

Există multe idei ce se folosesc pentru simularea de numere aleatoare uniforme $0-1$. Uneori sunt necesari generatori cu *calităţi* statistice foarte bune; aceşti generatori au de regulă complexitate de calcul mare şi necesită utilizarea de calculatoare cu putere de calcul mare mai ales privind efectuarea calculelor aritmetice (calculatoare cu *cuvânt mare!*). Alteori însă sunt necesari generatori mai puţin performanţi, care produc numere **pseudo-aleatoare** cu proprietăţi statistice mai sărace; de exemplu, numere U_i care sunt aproximativ independente stochastic şi repartizate numai *aproape* uniform.

Există un adevărat arsenal de **teste statistice** [6, 10] cu care se verifică proprietăţile statistice ale secvenţelor de numere aleatoare (uniformitatea repartiţiei, independenţa stochastică a numerelor succesiv generate, non-periodicitatea secvenţelor, etc.). Printre acestea, un loc important il ocupă *testele de concordanţă* care verifică ipoteza că numerele aleatoare $U \in [0,1]$, produse cu un generator, sunt uniform repartizate pe acel interval;asemenea teste sunt *testele de tip Kolmogorov-Smirnov* şi *testul general* χ^2 *de concordanţă.* Există, de asemenea, multe teste care se referă la caracterul aleator. Nu vom prezenta detalii privind aceste teste (Vezi de ex. $ 3.7). Cititorul interesat in aprofundarea acestei probleme poate consulta de exemplu [10]. Precizăm numai faptul constatat experimental că generatorii care produc secvenţe de perioade λ mari, au de regulă, proprietăţi statistice corespunzătoare.

In cele ce urmează prezentăm câţiva generatori de numere aleatoare care sunt, de regulă, implementaţi in limbajele de programare evoluate. Aproape orice astfel de limbaj are o funcţie (sau procedură) care, prin apelări succesive, produce secvenţe de numere pseudoaleatoare de bună calitate. (De ex. in Pascal şi C este funcţia *random* a cărei sămânţă este iniţializată standard la implementare sau care poate fi reiniţializată prin *randomize* de către utilizator). Fără să insistăm asupra detaliilor privind calitatea generatorilor şi implementarea lor efectivă, in cele ce urmează vom trece in revistă câteva metode ce se utilizează in construcţia practică a unor astfel de generatori (problemă care constituie mai degrabă o *temă de cercetare ştiinţifică*).

• **Generatorul aditiv congruenţial, sau Fibonacci decalat** notat $(k; X_0, X_1, ..., X_k; j; m)$ şi bazat pe relaţia

$$X_n = (X_{n-j} - X_{n-k})(mod\, m) \tag{2.13}$$

iar o alegere bună a constantelor este $j = 24$, $k = 55$, $m = 2^e$, $e \geq 31$ care asigură o perioada $\lambda \geq 2^{55} - 1$.

• **Generatorul congruenţial inversiv** notat $(X_0, a, c, m)^{-1}$ cu m-prim, definit de relaţia

$$X_n = (aX_{n-1}^{-1} + c)(mod\, m) \tag{2.14}$$

unde X_i^{-1} este *inversul* $(mod\ m)$ al lui X_i in raport cu operaţia de inmulţire a claselor de resturi, când acesta există sau este zero altfel. (Dacă m-prim şi $X_i \neq 0$ atunci inversul exista!).

• **Generatorul matricial congruenţial** care este de forma

$$\mathbf{X}_n = (A\mathbf{X}_{n-1} + C)(mod\, m) \tag{2.15}$$

unde $\mathbf{X}_n, C$ sunt vectori $d-$dimensionali iar A este matrice $d \times d$. Inmulţirea matriceală va produce vectori $\mathbf{X}_n$ cu componente corelate. Acest tip de generatori se utilizează pe calculatoare paralele.

• **Generatori bazaţi pe registre de deplasare** care utilizează *reprezentarea binară* a numerelor intregi in calculator. Dacă notăm cu $a_i, 1 \leq i \leq p$ cifrele binare ale lui x_{n-1} şi considerăm cifrele c_i nu toate egale cu zero, atunci generarea cifrelor a_i ale lui lui X_n se realizează prin relaţia

$$a_i = (c_p a_{i-p} + c_{p-1} a_{i-p+1} + ... + c_1 a_{i-1})(mod\, 2). \tag{2.16}$$

In practică se foloseşte forma mai simplă

$$a_i = (a_{i-p} + a_{i-p+q})(mod\, 2),\ p > q > 0 \tag{2.16'}$$

sau dacă notăm $\oplus$ operaţia binară *or*-exclusiv ca suma de biţi modulo 2 atunci relaţia precedentă devine

$$a_i = a_{i-p} \oplus a_{i-p+q}. \tag{2.16''}$$

Să observăm că relaţia de recurenţă referitoare la biţii a_i este aceeaşi cu relaţia de recurenţă a numerelor aleatoare X_i interpretate ca $p-$tupluri de biţi, adică

$$X_n = X_{n-p} \oplus X_{n-p+q}. \tag{2.17}$$

Un generator bun este de exemplu generatorul $X_n = X_{n-3p} \oplus X_{n-3q}$, $p = 521$, $q = 32$, care are o perioadă $\lambda = 2^{521-1}$. Formula (2.17) se poate extinde matricial sub forma

$$\mathbf{X}_n = \mathbf{X}_{n-p} \oplus \mathbf{X}_{n-p+q}. \tag{2.17'}$$

• **Amestecarea de generatori** se obţine astfel: Să presupunem că se dau doi generatori G_1, G_2 şi folosim o tabelă $T[1..k]$, (de ex $k = 64$ sau $k = 128$). Algoritmul de amestecare a celor doi generatori este

Iniţializăm $T[i] = U_i$ cu numere U_i produse cu G_1;
Generăm cu G_2 un indice aleator $j \in \{1, 2, ..., k\}$;
Luăm $Y := T[j]$; Generăm U cu G_1 şi luăm $T[j] := U$.

Generarea indicelui aleator j, care este un intreg uniform repartizat cu valori in $\{1, 2, ..., k\}$ se face astfel

Generează U cu G_2;
Ia $j = trunc(U \times k) + 1$.

Generatorul G rezultat din amestecare este *mai aleator* decât părinţii săi G_1, G_2, iar perioadele satisfac relaţia:$\lambda(G) = cmmmc(\lambda(G_1), \lambda(G_2)$.

• In final, prezentăm câteva **exemple concrete** de generatori:

1. Generatorul $(X_0, 125, 0, 2796203)$;
2. Generatorul$(x_0, 16807, 0, 2^{31} - 1)$;
3. Generatorul de numere uniforme U_i obţinute astfel:

$$X_i = 171X_{i-1}(mod\, 30269), Y_i = 172Y_{i-1}(mod\, 30307),$$

$$Z_i = 170Z_{i-1}(mod\, 30323),\ U_i = \left(\frac{X_i}{30269} + \frac{Y_i}{30307} + \frac{Z_i}{30323}\right)(mod\, 1).$$

Primii doi generatori sunt multiplicativ-congruenţiali şi satisfac condiţiile precizate anterior. Pentru cel de-al treilea se presupun date *seminţele* (X_0, Y_0, Z_0) ce corespund la trei generatori multiplicativ-congruenţiali şi se arată că U_i au perioada de ordinul 10^{12}.

In cele ce urmează vom presupune că dispunem de un generator de numere aleatoare uniforme $0 - 1$, reprezentat generic de funcţia *random.*

Exerciţii.

E2.1 Demonstraţi (2.6') şi (2.6") pentru generatorul mixt congruenţial.

E2.2 Fie $\mathbf{V} = (V_1, V_2, ..., V_k)'$ un vector aleator uniform pe intervalul $I = [a_1, b_1] \times [a_2, b_2] \times ... \times [a_k, b_k]$, $-\infty < a_i < b_i < \infty, 1 \leq i \leq k$. Să se arate că V_i sunt variabile aleatoare uniforme pe $[a_i, b_i]$ şi independente stochastic. Reciproca este de asemenea adevărată.

Indicaţie: Se consideră densităţile de repartiţie ale lui $\mathbf{V}$ şi V_i, adică

$$f(\mathbf{v}) = \begin{cases} \frac{1}{\prod_{i=1}^{k}(b_i - a_i)}, \, daca\, \mathbf{v} \in I \\ 0, \, in\, rest \end{cases}, \quad f_i(v_i) = \begin{cases} \frac{1}{b_i - a_i}, daca\, v_i \in [a_i, b_i] \\ 0, \, in\, rest \end{cases}$$

şi se observă că $f(\mathbf{v}) = \prod_{i=1}^{k} f_i(v_i)$.

E2.3 Arătaţi că dacă U este uniform $0-1$ atunci $V = a + (b-a)U$, cu $-\infty < a < b < +\infty$ este uniform pe $[a,b]$ şi reciproc.

Indicaţie: Folosind funcţia de repartiţie F a lui U dată de (2.4), se arată că funcţia de repartiţie a lui V este G dată de (2.3) şi invers.

E2.4 Să se arate că dacă U este o variabilă uniformă $0-1$ atunci şi $1-U$ este o variabilă repartizată uniform pe $[0,1]$.

Indicaţie: se arată că $P(1-U<x) = x$, $x \in [0,1]$.

E2.5 Folosind metoda de simulare a *indicelui aleator* j descrisă in ultima secţiune precizaţi in ce condiţii metoda poate fi folosită la simularea *aruncării cu zarul,* la simularea *tragerii la sorţi,* sau la simularea unui semn aleator?

Indicaţie: La aruncarea cu zarul se ia $k = 6$, iar la tragerea la sorţi se ia $k = 2$.

Simularea semnului aleator se realizează cu instruţiunea:

if $U \leq 0.5$ **then** $semn := 1$ **else** $semn := -1$;

Observaţie: pentru o variabilă aleatoare X continuă (care are densitate de repartiţie), avem $P(X = x) = 0$, deci in particular $P(U = u) = 0$, ceea ce inseamnă că semnul $\leq$ poate fi inlocuit cu semnul $\geq$ in cadrul instrucţiunii **if** precedente.

E2.6 Fie $(U,V)'$ un vector aleator uniform pe $[0,1] \times [0,1]$, unde U este independent de V. Să se calculeze $P(U+V<x), x \in R$.

Indicaţie. In general avem $P(U+V<x) = \int_D dudv$, $D = \{(u,v)|u+v<x\}$. Pornind de aici şi calculând ariile unor triunghiuri obţinem $P(U+V<x) = x^2/2$, *daca* $0 \leq x \leq 1$; $P(U+V<x) = 1-(2-x)^2/2$, *daca* $1 \leq x \leq 2$; $P(U+V<x) = 0$, *daca* $x<0$, $P(U+V<x) = 1$, *daca* $x>2$.

E2.7 Precizaţi cum se generează un indice aleator $i \in \{r, r+1, ..., s\}$ unde $r, s \in \mathcal{N}$ (adică r, s sunt numere naturale).

*Indicaţie.*Simularea se realizează cu instrucţiunea:$i = r + trunc[(s-r) \times U] + 1$, asemănător exerciţiului 5.

E2.8 Construiţi un generator de numere U uniforme pe $[0,1]$ utilizând *amestecarea* a trei generatori G_1, G_2, G_3.

Soluţie. Se va folosi o idee asemănătoare aceleia care a condus la amestecarea a doi generatori G_1, G_2, descrisă in ultimul algoritm de mai sus. Să considerăm un vector $L[1..m]$ in care se vor memora numere uniforme U generate cu G_1. Să considerăm un alt vector cu valori intregi $L1[1..k]$ ale cărui valori sunt numere intregi $L1[j] \in \{1, 2, ..., m\}, 1 \leq j \leq k$. Algoritmul obţinut prin amestecarea celor trei generatori este următorul:

Pasul 0. Incarcă vectorul L cu secvenţa de numere aleatoare $U_1, U_2, ..., U_m$, uniforme pe $[0,1]$, generate cu G_1; incarcă de asemenea vectorul $L1$

cu secvenţa de numere intregi $j_1, j_2, ..., j_k$ din $\{1, 2, ..., m\}$, generate cu G_2;

Pasul 1. Generează cu G_3 un indice aleator j din mulţimea $\{1, 2, ..., k\}$; ia $i := L1[j]$; generează cu G_2 un indice aleator i_1 in $\{1, 2, ..., m\}$ şi ia $L1[j] := i_1$;

Pasul 2. Ia $U := L[i]$; generează cu G_1 un număr aleator U_1 uniform pe $[0, 1]$ şi ia $L[i] := U_1$.

Numărul aleator pe $[0, 1]$ nou generat este numărul U şi el este obţinut prin amestecarea generatorilor G_1, G_2, G_3. Este de asteptat ca numerele U produse de acest algoritm să aibă proprietăţi de *aleatorism* mai bune decât numerele ce s-ar produce cu fiecare generator in parte. Printre altele se arată că dacă λ_i este perioada generatorului $G_i, i = 1, 2, 3$, atunci perioada generatorului dat de algoritmul precedent este $\lambda = cmmmc(\lambda_1, \lambda_2, \lambda_3)$ care poate fi (dacă alegem convenabil generatorii G_i) mai mare decât cel mai mare din λ_i.

Dacă este necesar să se producă o secvenţă $U_1, U_2, ..., U_n$ atunci se execută o singură dată pasul 0 şi se repetă de n ori paşii 1 şi 2 ai algoritmului.

Cap. 3

Simularea variabilelor neuniforme

Teorema 2.2 din capitolul precedent ne spune că cel puţin teoretic, putem simula orice variabilă aleatoare X dacă cunoaştem funcţia sa de repartiţie F şi putem calcula uşor şi fără erori funcţia inversă F^{-1}. Pornind de la acest fapt, precum şi de la alte considerente ce vor fi menţionate la locul potrivit, in literatura de specialitate s-au construit diverse metode, atât *generale* cât şi particulare, pentru simularea diverselor tipuri de repartiţii de probabilitate ne uniforme. Schema generală a acestor metode este de tipul următor: se presupun cunoscute metode de simularea unor variabile aleatoare *simple* $S_1, S_2, ..., S_n$ şi se caută un algoritm (adică metoda!) care să transforme aceste variabile in variabila X care trebuie generată. (In particular de exemplu, variabila *simplă* U se transformă in variabila $X = F^{-1}(U)$).

In cele ce urmează vom prezenta mai intâi câteva metode generale de simularea variabilelor aleatoare neuniforme. Toate aceste metode presupun simularea unor variabile aleatoare simple (cele mai simple sunt cele uniforme), iar prin transformarea/combinarea acestora se obţin variabilele dorite.

3.1 Metoda inversă

Această metodă a fost deja introdusă ca o consecinţă directă a teoremei lui Hincin. Ea se aplică, aşa cum s-a precizat, in cazul in care funcţia de repartiţie se poate inversa uşor. Dacă funcţia de repartiţie nu se inversează uşor, atunci ea se poate aproxima (pe intervale convenabil alese) cu o funcţie

liniară pe porţiuni, *cozile* fiind aproximate cu *exponenţiale* convenabil alese. In tabela următoare se prezintă câteva repartiţii de probabilitate *ce se pot simula uşor cu metoda inversă* (vezi şi [10]).

Repartiţia	Densitatea f	Inversa F^{-1}
$Exp(\lambda), \lambda > 0$	$f(x) = \lambda e^{-\lambda x}, x > 0$	$x = -\frac{1}{\lambda} ln(u)$
$Weib(0, 1, \nu), \nu > 0$	$f(x) = \nu x^{\nu-1} e^{-x^\nu}$	$x = (-ln(u))^{1/\nu}$
$Cauchy$	$f(x) = \frac{1}{\pi} \frac{1}{1+x^2}, x \in R$	$x = tan\pi(u - \frac{1}{2})$
$PearsonXI, \alpha \in R, \nu > 0$	$f(x) = \frac{\nu\alpha}{(1+\alpha x)^{\nu+1}}, x > 0$	$x = \frac{1}{u^{1/\nu}}$
$Arcsin$	$f(x) = \frac{1}{\pi} \frac{1}{\sqrt{1-x^2}}, x \in [-1, 1]$	$x = sin\pi(u - \frac{1}{2})$
$Logistica$	$f(x) = \frac{e^{-x}}{(1+e^{-x})^2}, x \in R$	$x = log\left(\frac{1-u}{u}\right)$

Tabelul 3.1. Metoda inversă

In formulele lui f din tabel se precizează numai valorile lui x pentru care $f(x) > 0$, presupunându-se că $f(x) = 0$ in rest.

In ultima coloană se scrie direct formula cu care se simulează variabila aleatoare X, deoarece se ţine cont de exerciţiul 4 din capitolul 2 (s-a inlocuit $1-u$ cu u [10], economisindu-se in acest fel o operaţie de scădere). Inlocuirea lui $1 - u$ cu u se va face ori de câte ori se va ivi ocazia, dacă este posibil.

Folosind *metoda inversă* se poate construi uşor un algoritm pentru simularea variabilei discrete

$$X : \begin{pmatrix} a_1, & a_2, & ..., & a_m \\ p_1, & p_2, & ..., & p_m \end{pmatrix}, \sum_{i=1}^{m} p_i = 1. \tag{3.1}$$

Pentru simularea variabilei aleatoare X calculăm mai intâi valorile funcţiei de repartiţie a acesteia.Avem

$$F(x) = \begin{cases} 0, \, daca \quad x < a_1 \\ p_1, \, daca \, a_1 \le x < a_2 \\ p_1 + p_2, \, daca \quad a_2 \le x < a_3 \\ ..... \\ p_1 + p_2 + ... + p_k, \, daca \quad a_k \le x < a_{k+1} \\ ..... \\ 1 \, daca \quad a_m \le x. \end{cases} \tag{3.2}$$

Algoritmul pentru simularea lui X constă deci in găsirea lui a_i a.î. $F(a_i) = U$ si este următorul:

Algoritmul SIMDISCRV [10]

Intrare tabela $T[1..m], T[i] := \sum_{j=1}^{i} p_j$; *Intrare* $a[1..m]$;
Generează U *uniform* $0-1$; (*cu* $U := random$;);
$i := 1$; **while** $U > F[i]$ **do** $i := i + 1$;
Ia $X := a[i]$.

3.2 Metoda compunerii sau amestecării

Această metodă se aplică variabilelor aleatoare X a căror repartiţie de probabilitate satisface următoarea definiţie [2, 3, 10]:

Definiţia 3. 1 *Funcţia de repartiţie* $F(x)$ *este o amestecare (sau compunere sau mixtură) discretă a mulţimii de funcţii de repartiţie* $\{F_i(x)\}_{1\leq i\leq m}$ *cu repartiţia disretă*

$$J : \begin{pmatrix} 1, & 2, & ..., & m \\ p_1, & p_2, & ..., & p_m \end{pmatrix}, \sum_{i=1}^{m} p_i = 1, \tag{3.3}$$

dacă

$$F(x) = \sum_{i=1}^{m} p_i F_i(x). \tag{3.4}$$

Funcţia de repartiţie $F(x)$ *este o amestecare continuă a familiei de funcţii de repartiţie* $\{G(x,Y)\}_{Y\in R}$, *cu funcţia de repartiţie continuă* $H(y)$ *a lui* Y *dacă ea este de forma*

$$F(x) = \int_R G(x,y)dH(y) \tag{3.5}$$

ultima integrală fiind integrala Stieltjes.

Dacă notăm cu X variabila aleatoare care are funcţia de repartiţie $F(x)$ şi cu X_i variabila aleatoare care are funcţia de repartiţie $F_i(x)$ atunci amestecarea discretă poate fi interpretată astfel

$$X = X_i \quad cu\, probabilitatea\, p_i$$

de unde rezultă următorul algoritm pentru simularea lui X

Algoritmul COMPDISCR
Generează un indice j *având repartiţia (3.3);*
Generează X_j *având funcţia de repartiţie* $F_j(x)$;
Ia $X := X_j$.

O interpretare analogă are şi amestecarea continuă, de unde se deduce următorul algoritm de simulare a variabilei aleatoare $X \rightsquigarrow F(x)$

Algoritmul COMPCONT
Generează variabila Y care are funcţia de repartiţie $H(y)$;
Generează variabila aleatoare Z_Y care are funcţia de repartiţie $G(x,Y)$;
Ia $X := Z_Y$.

Desigur, in algoritmii precedenţi se presupun cunoscute metode de generare a variabilelor aleatoare J, X_i, Y, Z_Y. Dacă in definiţia compunerii se consideră in loc de funcţii de repartiţie, densităţi de repartiţie atunci formulele (3.4) şi (3.5) se scriu respectiv [10,11]

$$f(x) = \sum_{i=1}^{m} p_i f_i(x) \tag{3.4'}$$

$$f(x) = \int_R g(x,y)h(y)dy. \tag{3.5'}$$

Exemplul 3.1, [10]. Presupunem că la o staţie de benzină sosesc m tipuri de autoturisme şi se cunoaşte p_i = probabilitatea să sosească un autoturism de tipul $i, 1 \le i \le m$. Presupunem că timpul X_i de intersosire al autoturismelor de tip i are repartiţia $Exp(\lambda_i)$ adică *exponenţială negativă de parametru* λ_i. Atunci timpul de intersosire X al autoturismelor oarecare (amestecate!) are o repartiţie *mixtexponenţială* (sau amestecat exponenţială) adică este o amestecare discretă cu densitatea

$$f(x) = \begin{cases} 0, & daca \quad x < 0 \\ \sum\limits_{i=1}^{m} p_i \lambda_i e^{-\lambda_i x_i}, & daca\, x \ge 0. \end{cases}$$

Deci simularea unui timp de intersosire a două autoturisme oarecare se va face cu algoritmul **COMPDISCR.**

Exemplul 3.1'. Fie X variabila având repartiţia $Laplace(\lambda)$ [5] a cărei densitate este

$$f(x) = \frac{\lambda}{2} e^{-\lambda|x|}, \;\; x \in R, \quad \lambda > 0.$$

Se observă că

$$f(x) = p_1 f_1(x) + p_2 f_2(x), \quad p_1 = p_2 = \frac{1}{2} = 0.5$$

$$f_1(x) = \begin{cases} \lambda e^{\lambda x}, \, daca\, x \le 0 \\ 0, \, daca\, x > 0 \end{cases}, \; f_2(x) = \begin{cases} 0, \, dacax \le 0, \\ \lambda e^{-\lambda x}, \, daca, x > 0 \end{cases}.$$

Algoritmul de compunere pentru simularea lui X este

Algoritmul LAPCOMP pentru simularea variabilei Laplace
Genereaza $U \rightsquigarrow uniform\, 0-1$;
if $U \leq 0.5$ **then** $s := -1$ **else** $s := 1$; (s=*semn aleator*);
Generează $Y \rightsquigarrow Exp(\lambda)$; *ia* $X := sY$.

Simularea variabilei $Laplace(\lambda)$ se poate face uşor şi cu metoda inversă.

Exemplul 3.2. Durata in funcţionare $X, X > 0$ a unui aparat (de ex. computer) este exponenţială de parametru $\eta\lambda$ unde $\lambda > 0$ este un parametru determinat de producător (in laborator!) iar η este un parametru aleator care indică influenţa *mediului* in care este exploatat calculatorul. Presupunând că repartiţia de probabilitate a lui η este de tip $Gamma(0, b, a)$ adică are densitatea

$$h(\eta) = \begin{cases} 0, & daca \quad x < 0 \\ \frac{b^a}{\Gamma(a)}\eta^{a-1}e^{-b\eta}, \end{cases} \tag{3.6}$$

să se determine densitatea de repartiţie a lui X.

Se observă [5,7] că este vorba de o amestecare continuă a cărei densitate este de forma (3.5') adică pentru $x \geq 0$ avem

$$f(x) = \int_0^\infty \eta\lambda e^{-\lambda\eta x}\frac{b^a}{\Gamma(a)}\eta^{a-1}e^{-b\eta}d\eta = \frac{\lambda b^a}{\Gamma(a)}\int_0^\infty \eta^a e^{-\eta(\lambda x+b)}d\eta =$$

$$= \frac{\lambda b^a\Gamma(a+1)}{\Gamma(a)(\lambda x+b)^{a+1}} = \frac{\lambda a}{b}\frac{b^{a+1}}{(\lambda x+b)^{a+1}} = \frac{a\theta}{(\theta x+1)^{a+1}}, \theta = \frac{\lambda}{b}.$$

Deci in final densitatea lui X este

$$f(x) = \begin{cases} 0, & daca \quad x < 0 \\ \frac{a\theta}{(\theta x+1)^{a+1}}, & daca \quad x \geq 0, \end{cases} \quad \theta = \frac{\lambda}{b}. \tag{3.7}$$

In calculele de mai sus s-a folosit următoarea formulă

$$\frac{\Gamma(\nu)}{a^\nu} = \int_0^\infty x^{\nu-1}e^{-ax}dx. \tag{3.8}$$

Repartiţia a cărei densitate este de forma (3.7) se numeşte repartiţie *Lomax* şi dându-se parametrii pozitivi λ, a, b simularea ei se realizează cu algoritmul **COMPCONT** cu condiţia de a cunoaşte un algoritm de simulare a variabilei $Gamma(0, b, a)$.

Exemplele de mai sus ilustrează două cazuri particulare când se poate aplica *metoda compunerii.* Totuşi următoarea teoremă ne asigură că metoda compunerii discrete se poate aplica in general.

Teorema 3. 1 *Fie X o variabilă aleatoare care are densitatea $f(x), x \in \Delta, (\Delta \subseteq R)$ şi să considerăm o diviziune a lui Δ adică $\Delta = \cup_{i=1}^{m} \Delta_i, \Delta_i \cap \Delta_j = \emptyset, \forall i \neq j$. Presupunând că $p_i = P(X \in \Delta_i) > 0$, există densităţile $f_i(x)$, nule pentru $x \notin \Delta_i$ astfel incât*

$$f(x) = \sum_{i=1}^{m} p_i f_i(x). \tag{3.9}$$

Demonstraţie [10]. Deoarece $p_i = \int_{\Delta_i} f(x)dx$ rezultă că putem defini densităţile de repartiţie

$$f_i(x) = \begin{cases} \frac{f(x)}{p_i}, & daca \quad x \in \Delta_i, \\ 0, & daca \quad x \notin \Delta_i. \end{cases} \tag{3.9'}$$

Fie un $x \in \Delta$, oarecare pentru care $f(x) \neq 0$. Atunci există un $i, 1 \leq i \leq m$ astfel incât $x \in \Delta_i$. Prin verificare directă se deduce că

$$f(x) = p_i f_i(x) = \frac{f(x)}{p_i} p_i = \sum_{j=1}^{m} p_j f_j(x), \ (caci\, f_j(x) = 0, \forall j \neq i)$$

şi teorema este demonstrată.

Metoda compunerii poate fi aplicată in combinaţie cu metoda inversă (de ex. pentru simularea variabilelor X_i din exemplul 3.1) aşa cum se va vedea in multe cazuri ce vor fi tratate in continuare. Ea poate fi aplicată şi in combinaţie cu *metoda respingerii* ce va fi tratată in continuare.

3.3 Metoda respingerii

Această metodă (denumită in mod *pesimistic* ca metodă a respingerii), ar putea fi denumită şi metoda *acceptării-respingerii.* Ea are forma generală următoare dacă ne referim la simularea unei variabile aleatoare X.

Presupunem că se cunosc următoarele elemente [10]:

- Se cunoaşte un procedeu de simulare a unei variabile aleatoare N care ia valori naturale pozitive;

- Se cunosc metode pentru simularea unor variabile aleatoare $S_i \in \mathcal{S}, i \geq 1$, unde S este o familie de variabile aleatoare dată;

- Se cunoaşte un predicat $\mathcal{P}(S_1, S_2, ..., S_n)$ care se poate calcula $\forall S_i, n$; (Acest predicat sau condiţie trebuie să poată fi evaluat fără calcule de mare complexitate!);

- Se cunoaşte funcţia Ψ astfel incât $X = \Psi(\{S_1, S_2, ..., S_n)\}$, $\mathcal{P}(S_1, S_2, ..., S_n) = true$).

Când pentru o variabilă aleatoare X este posibil să se construiască cele patru elemente cu proprietăţile precizate anterior, atunci se poate construi un algoritm pentru simularea lui X sub forma generală următoare:

ALGRES=Algoritm General de Respingere
repeat
Simulează o valoare n a lui $\mathcal{N}$;
Simulează valorile de selecţie $S_1, S_2, ..., S_n$ *din* $\mathcal{S}$;
until $\mathcal{P}(S_1, S_2, ..., S_n) = true$;
Ia $X = \Psi(S_1, S_2, ..., S_n)$.

Să observăm că dacă $\mathcal{P}(S_1, S_2, ..., S_n) = false$ atunci mulţimea de variabile aleatoare $\{S_1, S_2, ..., S_n\}$ *se respinge*, de unde provine şi numele de *metodă de respingere.* In aceeaşi ordine de idei, se observă că dacă $p_a = P(\mathcal{P}(S_1, S_2, ..., S_n) = true))$, numită *probabilitate de acceptare*, este mare (apropiată de 1), atunci algoritmul este *bun;* in caz contrar algoritmul este *lent.*

Vom prezenta in continuare câteva propoziţii care fundamentează algoritmi de respingere pentru simularea unor variabile aleatoare [2,10].

Teorema 3. 2 *Fie dată o variabilă aleatoare X care are densitatea de repartiţie $f(x), x \in R$, pe care dorim să o simulăm. Fie Y o altă variabilă aleatoare ce ştim să o simulăm şi a cărei densitate de repartiţie este $h(x)$ astfel incât densităţile f, h au acelaşi suport S (adică iau valori diferite de zero pe aceeaşi mulţime $S \subset R$). Presupunem că există o constantă $\alpha, 0 < \alpha < \infty$, astfel incât $f(x) \leq \alpha h(x), \forall x \in S$. In aceste condiţii dacă U este o variabilă uniformă $0-1$ independentă de Y, atunci densitatea de repartiţie a variabilei Y, condiţionată de $0 \leq U \leq f(Y)/(\alpha h(Y))$ este f.*

Demonstraţie. Trebuie să demonstrăm că

$$P\left[(Y < x) | (0 \leq U \leq f(Y)/(\alpha h(Y))\right] = F(x) = \int_{-\infty}^{x} f(v)dv.$$

Considerând evenimentele $A = \{Y < x\}, B = \{0 \leq U \leq f(Y)/(\alpha h(Y))\}$, ultima formulă devine

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

unde

$$P(B) = \int_{-\infty}^{\infty} \left[\int_{0}^{f(v)/(\alpha h(v))} du \right] h(v)dv = \int_{-\infty}^{\infty} \frac{f(v)}{\alpha h(v)} h(v)dv = \frac{1}{\alpha},\ \alpha > 1.$$

deci

$$P(A|B) = \alpha \int_{-\infty}^{x} \left[\int_{0}^{f(v)/(\alpha h(v))} du \right] h(v)dv = \alpha \int_{-\infty}^{x} \frac{f(v)}{\alpha h(v)} h(v) = \int_{-\infty}^{x} f(v)dv.$$

Teorema este demonstrată. Această teoremă este cunoscută sub numele de *teorema infăşurătoarei* deoarece condiţia referitoare la α inseamnă că graficul densităţii $f(x)$ se poate *infăşura* cu graficul lui $\alpha h(x)$.

Din demonstraţie rezultă că probabilitatea de acceptare este $p_a = 1/\alpha$, de unde rezultă că pentru o metodă a infăşurătoarei ne banală trebuie să avem $\alpha > 1$. Procedura de respingere este formată din următoarele elemente: $N = 2$ (variabila aleatoare constantă); $\mathcal{S} = \{U, Y\}$; $\mathcal{P}(U, Y) = true$ dacă $0 \leq U \leq f(Y)/(\alpha h(Y))$; $\Psi(U, Y) = Y$, (adică *proiecţia*).

Exemplul 3.3 [10]. Să considerăm densitatea de repartiţie $Gamma(0, 1, \nu), 0 < \nu < 1$, numită repartiţia gamma-standard cu ν subunitar. Să aplicăm metoda infăşurătoarei pentru simularea variabilei gamma notată X, folosind ca infăşurătoare densitatea $Weib(0, 1, \nu)$.

Soluţie. Densităţile de repartiţie sunt

$$f(x) = \begin{cases} 0, \, daca \, x < 0 \\ \frac{1}{\Gamma(\nu)} x^{\nu-1} e^{-x}, \, daca \, x > 0 \end{cases}, \quad h(x) = \begin{cases} 0, \, daca \, x < 0 \\ \nu x^{\nu-1} e^{-x^{\nu}}, \, daca \, x > 0 \end{cases}. \tag{3.10}$$

In vederea determinării constantei α a infăşurătoarei analizăm raportul

$$r(x) = \frac{f(x)}{h(x)} = \frac{1}{\nu\Gamma(\nu)} e^{-x+x^{\nu}}.$$

Punctul de maxim al funcţiei $r(x)$ este $x_0 = \nu^{1/(1-\nu)}$ de unde rezultă

$$\alpha = \frac{e^{\zeta(1-\nu)}}{\Gamma(\nu+1)}, \quad \zeta = \nu^{\frac{\nu}{1-\nu}}. \tag{3.11}$$

Deci in final algoritmul pentru simularea lui X este

Algoritmul GAMRESM1 [10] (algoritm de respingere pentru variabila gamma standard cu parametru subunitar)

Intrare ν; *Calculează* $c := 1/\nu$, $\zeta := \nu^{\frac{\nu}{1-\nu}}$ $a := e^{\zeta(\nu-1)}$;

repeat

Generează U *uniform* $0 - 1$;

Calculează $Z := -ln(U)$, $Y := Z^c$; {se generează $Y \rightsquigarrow Weib(0, 1, \nu)$;}

Generează alt U *uniform* $0 - 1$;

until $U \leq ae^{Z-Y}$;
Ia $X := Y$.

Primul pas al algoritmului este un *pas pregătitor;* in cazul când se simulează o selecţie de volum $n > 1$ asupra lui X atunci acest pas se execută o singură dată şi numai ceilalţi paşi se execută de n ori.

O altă metodă de respingere se poate obţine din teorema următoare [2,10].

Teorema 3. 3 *Fie X o variabilă aleatoare care are funcţia de repartiţie de forma*

$$F(x) = c \int_{-\infty}^{x} Q(\phi(x))dR(x) \tag{3.12}$$

unde $Q(z)$ este funcţia de repartiţie a unei variabile aleatoare $Z, Z \in [0, M]$, $\phi(x)$ este o funcţie ce ia valori in $[0, M]$ (unde M poate lua şi valoarea ∞) iar $R(y)$ eate funcţia de repartiţie a unei variabile aleatoare $Y, Y \in R$, iar variabilele Z, Y sunt independente stochastic. In aceste condiţii funcţia de repartiţie a variabilei Y, condiţionată de $Z \leq \phi(Y)$ este $F(x)$.

Demonstraţie [10]. Notând ca in Teorema 3

$$A = \{Y < x\}, B = \{Z \leq \phi(Y)\}, P(A|B) = \frac{P(A \cap B)}{P(B)}, \tag{3.12'}$$

ar trebui să demonstrăm că $P(A|B) = F(x)$. Să observăm mai intâi că constanta c din (3.12) este o constantă de normare adică

$$c = \left[\int_{-\infty}^{+\infty} Q(\phi(x))dR(x) \right]^{-1}. \tag{3.12''}$$

Să calculăm acum numitorul din (3.12'). Avem

$$P(B) = \int_{-\infty}^{+\infty} \left(\int_{0}^{\phi(x)} dQ(y) \right) dR(x) = \int_{-\infty}^{\infty} Q(\phi(x))dR(x) = \frac{1}{c}.$$

In continuare avem

$$P(A|B) = cP(A \cap B) = c \int_{-\infty}^{x} \left(\int_{0}^{\phi(t)} dQ(v) \right) dR(t) = c \int_{-\infty}^{x} Q(\phi(y))dR(y) = F(x).$$

Teorema este demonstrată. Se observă din nou că probabilitatea de acceptare a algoritmului este $p_a = P(B)$. Este evident că teorema descrie un procedeu de respingere ale cărui elemente sunt uşor de precizat.

Teorema rămâne valabilă dacă condiţia (3.12) se scrie in termeni de densităţi şi anume

$$f(x) = cQ(\phi(x))r(x),\ r(x) = R'(x). \tag{3.13}$$

O formă *dual*ă a teoremei se obţine dacă $F(x)$ este de forma [10]

$$F(x) = c\int_{-\infty}^{x}(1 - Q(\phi(x)))\,dR(x); c = \left[\int_{-\infty}^{+\infty}(1 - Q(\phi(y))dR(y)\right]^{-1} \tag{3.14}$$

când predicatul devine $\{Z \geq \phi(Y)\}$ iar condiţia (3.13) se scrie in termeni de densităţi

$$f(x) = c(1 - Q(\phi(x))r(x). \tag{3.13'}$$

Şi in acest caz avem $p_a = P(Z \geq \phi(Y)) = 1/c$.

Exemplul 3.4 [5,7]. Fie X variabila aleatoare a cărei densitate de repartiţie este

$$f(x) = c(1 - e^{-\lambda x})e^{-\mu x},\ x \geq 0 \tag{3.13''}$$

adică de forma (3.13) şi se cere să se precizeze un algoritm de respingere pentru simularea lui X.

Soluţie. Se observă că $\phi(x) = x$ şi $Q(z) = 1 - e^{-\lambda z}, z > 0$, deci

$$c = \left[\int_{0}^{+\infty}(1 - e^{-\lambda x})e^{-\mu x}dx\right]^{-1} = [\frac{\lambda}{\lambda + \mu}]^{-1}.$$

Algoritmul de respingere penru simularea lui X este

Algoritmul RESP34
repeat
Generează $Z \rightsquigarrow Exp(\lambda)$;
Generează $Y \rightsquigarrow Exp(\mu)$;
until $Z \leq Y$;
Ia $X := Y$.

Acest algoritm este rapid dacă $\mu << \lambda$.

Dacă se calculează in detalii funcţia de repartiţie $F(x)$ a densităţii (3,13") se poate constata că rezolvarea ecuaţiei $F(X) = U$ necesită o abordare numerică şi deci metoda inversă nu este recomandabilă.

Următoarea teoremă [2, 10], ce va fi numită *teorema şirului descendent,* datorată lui Forsythe, fundamentează un algoritm introdus de John von Neumann pentru simularea exponenţialei $Exp(1)$, algoritm ce va fi prezentat intr-o secţiune următoare.

Teorema 3. 4 *Presupunem date variabilele aleatoare $Z_i \rightsquigarrow G(x), i \geq 1$ şi $Z_0 \rightsquigarrow G_0(x)$, independente stochastic. Atunci, următoarele afirmaţii sunt adevărate*

(1) Dacă x este fixat şi numărul k este fixat, atunci

$$P(x \geq Z_1 \geq Z_2 \geq ... \geq Z_{k-1} < Z_k) = \frac{[G(x)]^{k-1}}{(k-1)!} - \frac{[G(x)]^k}{k!}. \tag{3.15}$$

(2) Dacă x este fixat şi K este indicele aleator la care se "rupe" subşirul descendent (ca la pct (1)), atunci

$$P(K = nr.impar) = P(K(mod 2) = 1) = e^{-G(x)}. \tag{3.15'}$$

(3) Dacă subşirul descendent este $Z_0 \geq Z_1 \geq ... \geq Z_{K-1} < Z_K$ (adică se rupe la un K aleator şi incepe cu $Z_0 \rightsquigarrow G_0(x)$), atunci

$$P[Z_0 < x | K(mod 2) = 1] = \frac{1}{p_a} \int_{-\infty}^{x} e^{-G(t)} dG_0(t), \tag{3.16}$$

unde p_a este constanta de normare

$$p_a = \left[\int_{-\infty}^{+\infty} e^{-G(X)} dG_0(x) \right]^{-1}. \tag{3.16'}$$

Demonstraţie [10]. (1) Fie evenimentele $A = \{x \geq Z_1 \geq Z_2 \geq ... \geq Z_{k-1}\}$, $B = \{x \geq Z_1 \geq Z_2 \geq ... \geq Z_k\}$. Se observă că $B \subset A$ şi $P(Z_i \leq x) = G(x)$. Deoarece subşirul care defineşte A conţine numai una din cele $(k-1)!$ ordini relative in care se pot afla cele $k-1$ variabile aleatoare $Z_i, 1 \leq i \leq k-1$, rezultă că

$$P(A) = \frac{(G(x))^{k-1}}{(k-1)!}.$$

Pentru a demonstra (3.15) să observăm că probabilitatea din membrul stâng se scrie $P(A \setminus B)$ şi deoarece $A \subset B$ avem

$$P(A \setminus B) = P(A) - P(B) = \frac{[G(x)]^{k-1}}{(k-1)!} - \frac{[G(x)]^k}{k!}$$

şi afirmaţia (1) este demonstrată.

Afirmaţia (2) rezultă din următoarele relaţii:

$$P(K = nr.impar) = P(K = 1) + P(K = 3) + ... =$$

$$= 1 - \frac{G(x)}{1!} + \frac{[G(x)]^2}{2!} - \frac{[G(x)]^3}{3!} + ... = e^{-G(x)}.$$

Pentru a demonstra (3) să observăm că atunci când Z_0 este aleator avem

$$P(K = nr.impar) = P(K \, mod \, 2 = 1) = \int_{-\infty}^{\infty} e^{-G(x)} dG_0(x)$$

de unde folosind p_a dat de (3.16') obţinem

$$P(Z_0 < x | K = nr.impar) = \frac{1}{p_a} \int_{-\infty}^{x} e^{-G(t)} dG_0(t)$$

şi teorema este demonstrată.

Teorema reprezintă un prototip complex de metodă de respingere; se atribuie ca valoare a lui X valoarea Z_0 din *ultimul subşir descendent* care este acceptat.

Algoritmul de respingere rezultat din teoremă este urmărorul

Algoritmul RESPSIRD bazat pe siruri descendente.
repeat
Generează $Z_0 \rightsquigarrow G_0(x)$; *Ia* $Z^* = Z_0, K := 1$; *generează* $Z_1 \rightsquigarrow G(x)$;
while $Z_0 \geq Z_1$ **do begin**
$K := K + 1; Z_0 := Z_1;$
Generează $Z_1 \rightsquigarrow G(x)$;
end;
until $K \, mod 2 = 1$;
Ia $X := Z^*$.

Să analizăm acum performanţa algoritmului. Observăm că p_a ne dă informaţie asupra vitezei algoritmului: cu cât p_a este mai mare, cu atât mai repede ajungem să acceptăm un Z_0 (când $K = nr.impar$). Dar p_a (probabilitatea de acceptare) nu este de ajuns pentru a caracteriza performanţa algoritmului; ar trebui să vedem şi *câte numere* $Z_i, i \geq 0$ sunt necesare pentru a obţine un Z_0- acceptat.Vom utiliza notaţiile:

p_r este probabilitatea de a respinge un Z_0 (de a respinge un subşir!); avem $p_r = 1 - p_a$;

N_a = numărul de valori $Z_i, i \geq 1$ dintr-un subşir in care se acceptă un Z_0;

N_r = numărul de valori $Z_i, i \geq 1$ dintr-un subşir in care se respinge Z_0;

N^* = Numărul total de valori de selecţie $Z_i, i \geq 0$ necesare până la acceptarea unui Z_0.

Folosind aceste notaţii putem calcula valorile medii ale numerelor N. Avem

$$E(N^*) = p_a E(N_a) + p_r[E(N_r) + E(N^*)]$$

de unde

$$E(N^*) = \frac{1}{p_a}[E(N_a)p_a + E(N_r)p_r].$$

Dar

$$E(N_a)p_a + E(N_r)p_r = E(K+1), \quad deci \quad E(N^*) = \frac{1}{p_a}E(K+1).$$

Se observă că

$$E(K+1) = \sum_{k=1}^{\infty}(k+1)\int_{-\infty}^{+\infty}\left[\frac{(G(x))^{k-1}}{(k-1)!} - \frac{(G(x))^k}{k!}\right]dG_0(x) =$$

$$= \sum_{k=0}^{\infty}(k+2)\int_{-\infty}^{+\infty}\frac{(G(x))^k}{k!}dG_0(x) - \sum_{k=1}^{\infty}(k+1)\int_{-\infty}^{+\infty}\frac{(G(x))^k}{k!}dG_0(x) =$$

$$= 1 + \int_{-\infty}^{+\infty}\sum_{k=0}^{\infty}\frac{(G(x))^k}{k!}dG_0(x) = 1 + \int_{-\infty}^{+\infty}e^{G(x)}dG_0(x).$$

In concluzie

$$E(N^*) = \frac{1}{p_a}\left(1 + \int_{-\infty}^{+\infty}e^{G(x)}dG_0(x)\right). \tag{3.17}$$

Exemplul 3.5 [10]. Presupunem că in Teorema 3.4, variabilele $Z_i, i \geq 0$ sunt toate uniforme $0-1$ şi independente, adică $Z_i = U_i, i \geq 0$. Atunci, conform teoremei avem

$$P(U_0 \leq x | K = nr.impar) = \frac{1}{p_a}\int_0^x e^{-x}dx,\ p_a = \int_0^1 e^{-x}dx = 1 - e^{-1}.$$

Cu alte cuvinte U_0-acceptat are funcţia de repartiţie

$$F(x) = \begin{cases} 0, & daca \quad x < 0, \\ \frac{1-e^{-x}}{1-e^{-1}}, & daca \quad 0 \leq x \leq 1, \\ 1, & daca \quad x > 1 \end{cases} \tag{3.18}$$

care este *repartiţia exponenţială negativă de parametru 1, trunchiată pe intervalul* $[0, 1]$. Deci simularea variabilei care are funcţia de repartiţie $F(x)$ din (3.18) se poate face cu algoritmul **RESPSIRD.** De fapt, intr-un capitol următor vom arăta cum acest algoritm va fi completat astfel incât să permită simularea unei variabile exponenţiale negativă netrunchiată. Numărul mediu $E(N^*)$ de variabile $\{Z_i\}_{i\geq 0}$ ce trebuie generate este conform formulei (3.17)

$$E(N^*) = \frac{1}{1-e^{-1}}(1 + \int_0^1 e^x dx) = \frac{e}{1-e^{-1}} = \frac{e^2}{e-1}. \tag{3.17'}$$

3.4 Alte metode

Aşa cum am menţionat, metodele de simulare a variabilelor aleatoare neuniforme X, pot fi obţinute fie prin transformarea unor variabile aleatoare uniforme $0-1$, fie sub forma

$$X = T(Y_1, Y_2, ..., Y_n) \tag{3.19}$$

unde variabilele Y_i *pot* fi simulate. De fapt, metodele prezentate in primele trei subsecţiuni ale acestui capitol sunt toate de acelaş tip: simularea lui X se *reduce* la simularea unor variabile mai simple $Y_i, 1 \leq i \leq n$, unde şi n *poate fi aleator.*

In această secţiune vom prezenta metode de simulare a unor variabile aleatoare particulare care au tocmai astfel de proprietăţi: repartiţiile lor derivă din alte repartiţii mai simple.

In tabelul 3.2. (vezi [10]) se dau câteva repartiţii de probabilitate a căror simulare se realizează prin transformări simple de variabile U uniforme $0-1$ dar nu prin metoda inversă. In ultima coloană se precizează transformarea T.

Nr. crt.	Numele repartiţiei	Densitatea de repartiţie	Transformarea T
1	Normală $N(0,1)$	$f(x)=\frac{1}{\sqrt{2\pi}}e^{-x^2/2}$ $-\infty<x<\infty$	$X=\sum_{i=1}^{12}U_i-6$
2	Modul	$f(x)=\begin{cases}1-\lvert x\rvert,\ x\in[-1,1]\\ 0,\ altfel\end{cases}$	$X=U_1-U_2$
3	Repartiţia maximului	$f(x)=\begin{cases}nx^{n-1},\ x\in[0,1]\\ 0,\ altfel\end{cases}$	$X=\max\{U_1,U_2,...,U_n\}$
4	Repartiţia minimului	$f(x)=\begin{cases}n(1-x)^{n-1},\ x\in[0,1]\\ 0,\ altfel\end{cases}$	$X=\min\{U_1,U_2,...,U_n\}$
5	Repartiţia $Erlang(k)$	$f(x)=\begin{cases}o,\ daca\ x<0\\ \frac{1}{\Gamma(k)},\ daca\ x\geq 0\end{cases}$ $k\in N^+$	$X=-ln\{\prod_{i=1}^{k}U_i\}$

Tabelul 3.2. Repartiţii simulate prin transformări de uniforme

Variabilele uniforme U_i care intervin in ultima coloană a tabelului sunt presupuse independente. Să vedem cum se justifică metodele prezentate sintetic in tabel.

1. Aici se foloseşte *teorema limită centrală* intr-o forma simplificată şi anume: dacă $\{V_n\}_{n\in N}$ este un şir de variabile aleatoare independente şi identic distribuite care au momente de primele două ordine şi dacă notăm $S_n=\sum\limits_{i=1}^{n}V_i$, atunci

$$\lim_{n\to\infty}P\left(\frac{S_n-E(S_n)}{\sqrt{Var(S_n)}}<x\right)=\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x}e^{-\frac{t^2}{2}}dt. \tag{3.20}$$

In membrul al doilea este funcţia de repartiţie normală $N(0,1)$. Dacă considerăm ca variabile V_i variabilele uniforme U_i avem

$$E(U_i)=\frac{1}{2},\quad Var(U_i)=\frac{1}{12}.$$

Viteza de convergenţă in (3.20) depinde de viteza de generare a variabilelor V_i. In cazul când $V_i=U_i$ membrul stâng din (3.20) se apropie suficient de mult de funcţia de repartiţie normală $N(0,1)$ din membrul drept dacă

$n \geq 10$. Luând $n = 12$ se obţine expresia din ultima coloană a tabelului (3.20) care are (aproximativ!) repartiţia normală $N(0,1)$.

Densitatea de repartiţie *normală* $N(m,\sigma)$ este

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-m)^2}{2\sigma^2}}. \tag{3.21}$$

Notând Y variabila $N(m,\sigma)$ avem

$$m = E(Y), \quad \sigma^2 = Var(Y). \tag{3.21'}$$

Dacă notăm $Z \rightsquigarrow N(0,1)$ atunci se poate arăta cu uşurinţă că intre Y şi Z au loc relaţiile

$$Y = m + Z\sigma, \quad Z = \frac{Y-m}{\sigma} \tag{3.21''}$$

de unde rezultă că simularea lui Y se reduce la simularea lui Z, care la rândul său se simulează folosind teorema limită centrală ca in tabelă.

2. Repartiţia *modul* se obţine pe o cale analogă celei folosite in exerciţiul **E2.6**, Cap.2.

3. Justificarea este următoarea

$$F(x) = P(\max\{U_1, U_2, ..., U_n\} < x) =$$

$$= P(X_1 < x, ..., U_n < x) = \begin{cases} 0, \, daca \, x < 0, \\ x^n, \, daca \, x \in [0,1], \\ 1, \, daca \, x > 1 \end{cases}$$

şi derivând $F(x)$ se obţine densitatea din tabel.

4. Observăm că

$$P(\min\{U_i, U_2, ..., U_n\} < x) = 1 - P(\max\{U_1, U_2, ..., U_n\} \geq x) =$$

$$= 1 - (1-x)^n, \forall x \in [0,1]$$

şi derivând ultima expresie se obţine densitatea de repartiţie din tabel.

5. Repartiţia $Erlang(k), k \in N^+$, este un caz particular al repartiţiei $Gamma(\alpha, \lambda, \nu)$, $\alpha, \lambda, \nu \in R^+$ care are densitatea de repartiţie [3, 6, 7, 10]

$$f(x) = \begin{cases} 0, \, daca \, x < \alpha, \\ \frac{\lambda^\nu}{\Gamma(\nu)}(x-\alpha)^{\nu-1} e^{-\lambda(x-\alpha)}, \, daca \, x \geq \alpha. \end{cases} \tag{3.22}$$

Să notăm cu Y variabila $Gamma(\alpha, \lambda, \nu)$. Relaţia dintre Y şi *variabila gama standard* $X \rightsquigarrow Gamma(0,1,\nu)$ este

$$Y = \alpha + \frac{X}{\lambda}, \quad X = (Y-\alpha)\lambda. \tag{3.22'}$$

Pentru a arăta că variabila $Erlang(k)$ (care este *gamma standard* $Gamma(0,1,k), k \in N^+$) se simulează ca in tabelă, vom demonstra teorema

Terema 3. 5 *Dacă $Z_i, 1 \leq i \leq k$ sunt variabile $Exp(1)$ independente, atunci variabila*

$$X = \sum_{i=1}^{k} Z_i \tag{3.23}$$

este o variabilă $Erlang(k)$.

Demonstraţie [10]. Vom folosi tehnica funcţiei caracteristice şi vom arăta că funcţia caracteristică a variabilei X dată de (3.23) este aceeaşi cu funcţia caracteristică a variabilei $Erlang(k)$ de unde se deduce că cele două repartiţii coincid. Pentru variabila $Erlang(k)$ funcţia caracteristică este

$$\varphi(t) = E[e^{itX}] = \int_0^{\infty} e^{itx} \frac{1}{(k-1)!} x^{k-1} e^{-x} dx =$$

$$= \frac{1}{(k-1)!} \int_0^{\infty} x^{k-1} e^{-x(1-it)} dx == \frac{1}{(k-1)!} \frac{\Gamma(k)}{(1-it)^k} = \frac{1}{(1-it)^k}.$$

Pe de-altă parte funcţia caracteristică a variabilei X din (3.23) este

$$\varphi_X(t) = e[e^{it\sum_j Z_j}] = \prod_{j=1}^{k} \varphi_{Z_j}(t) = (\varphi_Z(t))^k .$$

Dar

$$\varphi_Z(t) = E[e^{itZ}] = \int_0^{\infty} e^{-(1-it)x} dx = \frac{1}{1-it}$$

deci

$$\varphi_X(t) = \left(\frac{1}{1-it}\right)^k,$$

adică aceeaşi cu $\varphi(t)$, ceea ce demonstrează teorema.

Folosind aceeaşi tehnică de demonstraţie se poate demonstra şi următoarea teoremă [10].

Teorema 3. 6 *Dacă $X \rightsquigarrow Gamma(0,1,\nu)$, $Y \rightsquigarrow Gamma(0,1,\mu)$ şi X şi Y sunt independente stochastic, atunci*

$$(Z = X + Y) \rightsquigarrow Gamma(0,1,\nu+\mu).$$

Când o familie de variabile aleatoare are proprietatea că suma a două variabile independente este tot o variabilă din aceeaşi familie spunem că aceasta este o *familie stabilă* de variabile aleatoare. Teorema 3.6 spune deci că *familia variabilelor aleatoare gamma standard este stabilă.*

Tot cu ajutorul funcţiei caracteristice se poate demonstra teorema

Teorema 3. 7 *Familia variabilelor normale este stabilă.*

Demonstraţie. Se foloseşte faptul că dacă $X \rightsquigarrow N(m,\sigma)$ atunci funcţia sa caracteristică este $\varphi_X(t) = E[e^{itX}] = e^{itm-\frac{t^2\sigma^2}{2}}$. Asemănător dacă $Y \rightsquigarrow N(p,\lambda)$ atunci $\varphi_Y(t) = e^{itp-\frac{t^2\lambda^2}{2}}$, iar dacă X, Y sunt independente atunci $\varphi_{X+Y}(t) = e^{it(m+p)-\frac{t^2(\sigma^2+\lambda^2)}{2}}$, ceea ce inseamnă că $(X+Y) \rightsquigarrow N(m+p,\sqrt{\sigma^2+\lambda^2})$.

3.4.1 Simularea repartiţiilor inrudite cu repartiţia normală

Vom defini acum o serie de variabile aleatoare care derivă din alte variabile aleatoare şi care au importanţă in statistica matematică.

Exemple 3.6. Repartiţii inrudite cu repartiţia normală [5, 7, 10].

Exemplul 3.6.1. Repartiţia χ^2. Dacă $Z_i, 1 \leq i \leq f$ sunt variabile normale $N(0,1)$ independente atunci

$$\chi^2 = \sum_{i=1}^{f} Z_i^2 \tag{3.24}$$

se numeşte *variabilă hi patrat centrată cu f grade de libertate* notată χ_f^2.

Dacă $Z_i \rightsquigarrow N(m_i,1)$ atunci variabila (3.24) se numeşte *variabilă hi patrat necentrată cu f grade de libertate şi cu parametrul de excentricitate* δ, *notată* $\chi_{f,\delta}^2$, *unde*

$$\delta^2 = \sum_{i=1}^{f} m_i^2. \tag{3.24'}$$

Se arată că variabila χ_f^2 centrată este o variabilă de tip $Gamma(0,\frac{1}{2},\frac{f}{2})$. Formula (3.24) permite simularea *directă,* pornind de la definiţie a variabilelor $\chi_f^2, \chi_{f,\delta}^2$.

Exemplul 3.6.2. Dacă $Z \rightsquigarrow N(0,1)$ este independentă de variabila hi patrat χ^2_f atunci variabila

$$t_f = \frac{Z}{\sqrt{\frac{\chi^2_f}{f}}} \tag{3.25}$$

se numeşte *variabila t a lui Student cu f grade de libertate.* Dacă in (3.25) in loc de χ^2_f se introduce $\chi^2_{f,\delta}$ atunci se obţine variabila notată $t_{f,\delta}$ care se numeşte *variabila t Student ne centrată cu f grade de libertate şi cu parametrul de excentricitate* δ. Variabilele de tip t Student se simulează deci direct cu formule de tipul (3.25).

Exemplul 3.6.3. Dacă variabilele $\chi^2_{f_1}, \chi^2_{f_2}$ sunt independente atunci variabila

$$F_{f_1,f_2} = \frac{f_2 \chi_{f_1}}{f_1 \chi_{f_2}} \tag{3.26}$$

se numeşte variabila F a lui Snedecor centrată cu f_1, f_2 grade de libertate (in notaţia indicilor contează ordinea!). Dacă in (3.26) se foloseşte câte una din $\chi_{f_1,\delta_1}, \chi_{f2,\delta_2}$ sau ambele, atunci se obţin variabilele F Snedecor *simplu necentrate* $F_{f_1,f_2;\delta_1,0}, F_{f_1,f_2;0,\delta_2}$, cu parametri corespunzători de excentricitate, sau variabila F Snedecor *dublu necentrată* $F_{f_1,f_2;\delta_1,\delta_2}$. Variabilele F pot fi de asemenea simulate direct prin formule de tipul (3.26).

In exemplele precedente mediile sunt $m = 0$ sau $m \neq 0$ iar dispersiile sunt $\sigma = \sigma^2 = 1$.

Exemplul 3.7 [10]. Variabila aleatoare pozitivă Y se numeşte *lognormală* $LN(\mu, \sigma)$ de parametri μ, σ dacă variabila $X = log(Y)$ este normală $N(\mu, \sigma)$. Avem deci densităţile

$$X \rightsquigarrow f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, x \in R;$$

$$Y \rightsquigarrow g(y) = \frac{1}{\sqrt{2\pi} y \sigma} e^{-\frac{(log(y)-\mu)^2}{2\sigma^2}}, y \in R+.$$

Calculând integralele corespunzătoare, in care se fac schimbări de variabile potrivite, se obţin următoarele relaţii

$$m = E[Y] = e^{\mu+\sigma^2/2}, \quad s^2 = Var[Y] = m^2(e^{\sigma^2} - 1) \tag{3.27}$$

de unde prin rezolvarea acestui sistem se obţin formulele

$$\mu = log(m) - \frac{1}{2} log\left[\frac{s^2}{m^2} + 1\right], \quad \sigma^2 = log\left[\frac{s^2}{m^2} + 1\right]. \tag{3.27'}$$

Dacă se dau media m şi dispersia s^2 pentru variabila lognormală Y atunci media μ şi dispersia σ^2 pentru X se calculează cu formulele (3.27'); deci simularea variabilei Y se realizează cu algoritmul simplu

Algoritmul LNORM pentru simularea lognormalei.
Intrare m, s^2; *calculează* μ, σ; (acesta este un pas pregatitor!);
Generează $Z \rightsquigarrow N(0,1)$; (ca in tabelul 3.1);
Calculează $X = \mu + Z\sigma$; (unde $\sigma = \sqrt{\sigma^2}$);
Ia $Y = e^X$. ($Y \rightsquigarrow LN(\mu, \sigma)$).

• **Familia de variabile de tip Johnson** [5, 7]. Din aceasta familie fac parte o serie de variabile ce se obţin prin transformări de variabile normale $X \rightsquigarrow N(m, \sigma)$. Aceste transformări sunt:

$Y = \lambda e^X + \xi$, - **variabila lognormală;**

$Y = \lambda \left[1 + e^X\right]^{-1} + \xi$ - **variabila logit normală;**

$Y = \lambda sinh(X) + \xi$ - **Variabila** $Sinh^{-1}$ **normală**,

$sinh(x) = \frac{e^x - e^{-x}}{2}$, unde $\lambda > 0, \xi \geq 0$, aceste constante alegânduse de obicei astfel incât $E\{Y\} = 0$, $Var\{Y\} = 1$.

Variabila *logonormală* a fost studiată mai sus,variabila *logit normală* este de tip *logistică,* iar variabila $Sinh^{-1}$ **normală** este de tip *sinus hiperbolic.*

3.5 Simularea unor variabile particulare

In secţiunile precedente am prezentat diverse metode, unele generale, pentru simularea variabilelor aleatoare ne uniforme. In această secţiune vom prezenta metode pentru simularea tipurilor de variabile aleatoare particulare, bazate fie pe metodele generale, fie pe anumite particularităţi, dar care conduc la algoritmi de regulă mai buni decât cei ilustraţi in exemplele prezentate până acum. (A se vedea [2, 3, 5, 6, 7, 10, 11]).

3.5.1 Repartiţia exponenţială

Este suficient să ne ocupăm de repartiţia variabilei $Z \rightsquigarrow Exp(1)$ a cărei densitate de repartiţie este

$$f(x) = \begin{cases} 0, & daca\, x \leq 0 \\ e^{-x}, & daca\, x > 0, \end{cases} \tag{3.28}$$

căci simularea variabilei $X \rightsquigarrow Exp(\lambda)$ se realizează cu $X = \frac{Z}{\lambda}$. Metoda inversă pentru simularea lui Z, adică $Z = -log(U)$, nu este recomandabilă deoarece dacă U este apropiat de zero atunci $log(U)$ nu se poate calcula. De

aceea vom prezenta o metodă de *respingere* datorată lui *John von Newmann* care se bazează pe teorema următoare.

Teorema 3. 8 *Să luăm in teorema subşirului descendent (Teorema 3.4),* $Z_0 = U_0, Z_i = U_i, i \geq 1$, *unde* U_0, U_i *sunt uniforme* $0-1$. *Dacă notăm cu* N *numărul aleator de subşiruri descendente respinse până când se acceptă un subşir, atunci* $X = N + Z_0 \rightsquigarrow Exp(1)$, *unde* Z_0 *este cel acceptat (din ultimul subşir descendent).*

Demonstraţie [10]. Din **Exemplul 3.5** de mai sus, reţinem că pentru $x \in [0,1]$ avem

$$P(Z_0 \leq x | K = nr.impar) = \frac{1}{p_a}\int_0^x e^{-x}dx = F(x) = \frac{1-e^{-x}}{p_a},\ p_a = 1-e^{-1},$$

unde F(x) este dat de (3.18). De aici rezultă că probabilitatea de a respinge un şir descendent (de forma $Z_0 \geq Z_1 \geq ... \geq Z_{K-1} < Z_K$) este $p_r = 1 - p_a = e^{-1}$. Deci, $P(N = n) = e^{-n}(1-e^{-1})$. Rămâne să arătăm că

$$P(N + Z_0 \leq x) = \begin{cases} 0, & daca\, x < 0 \\ 1 - e^{-x}, & daca\, x \geq 0. \end{cases} \qquad (3.28')$$

Intr-adevăr, dacă pentru un $x > 0$ dat notăm $x = k + z, k = [x], z \in [0,1)$ (k= partea intreagă) atunci avem

$$P(N + Z_0 \leq x) = P(N + Z_0 \leq k + z) = P(N < k) + P(N = k, Z_0 \leq z) =$$

$$= \sum_{j=0}^{k-1}(1-e^{-1})e^{-j} + (1-e^{-1})\frac{e^{-k}}{1-e^{-1}}\int_0^z e^{-u}du =$$

$$= 1 - e^{-k} + e^{-k}(1-e^{-z}) = 1 - e^{-(k+z)} = 1 - e^{-x},\ x > 0$$

şi teorema este demonstrată.

De aici se deduce următorul algoritm pentru simularea lui Z.

Algoritmul EXRJ de simulare a exponenţilalei prin metoda respingerii
Iniţializează $N := 0$;
repeat
generează U_0, U_1 *uniforme* $0-1$ *şi independente; Ia* $U^* := U_0, K := 1$;
while $U_0 \geq U_1$ **do begin**
$K := K + 1$, $U_0 := U_1$, *generează* U_1 *uniform* $0-1$;
end; (s-a simulat un subşir descendent);

if $K \, mod \, 2 = 0$ **then** $N := N + 1$; (se numără subşirurile respinse);
until $K \, mod \, 2 = 1$;
Ia $Z := N + U^*$.

Pe lângă probabilitatea de acceptare $p_a = 1 - e^{-1}$, performanţa algoritmului este caracterizată şi de numărul N^* al variabilelor uniforme $\{U_i\}_{i \geq 0}$ necesare a fi generate până când se obţine o valoare de selecţie exponenţială Z. Conform (3.17) avem

$$E(N^*) = \frac{1}{p_a}(1 + \int_0^1 e^z dz) = \frac{1}{1 - e^{-1}}(1 + e - 1) = \frac{e^2}{e - 1} \approx 3.8, \qquad (3.17')$$

adică trebuie simulate in medie aproape 4 numere uniforme pentru a obţine o valoare de selecţie exponenţială Z.

3.5.2 Repartiţia Gama

Repartiţia $Gamma(\alpha, \lambda, \nu)$ are densitatea dată de formula (3.22) iar formula (3.22') spune că este necesar şi esenţial să construim metode pentru simularea unei repartiţii *Gamma standard,* adică $Gamma(0, 1, \nu)$, $\nu \in R^+$. Dacă $\nu = k \in N^+$ atunci repartiţia gama devine repartiţie Erlang şi simularea acestei variabile (notată E_k) se face simplu prin insumare de exponenţiale, conform Teoremei 3.5. Dacă $0 < \nu < 1$ atunci ţinând seama de Exemplul 3.3, simularea variabilei $Ganna(0, 1, \nu)$ se realizează cu o metodă de respingere bazată pe *infăşurarea* cu o densitate $Weibull(0, 1, \nu)$.

In această subsecţiune vom prezenta şi alte metode pentru simularea variabilelor gamma [10].

• **O metodă de compunere-respingere pentru cazul** $0 < \nu < 1$.
Vom scrie densitatea de repartiţie $Gamma(0, 1, \nu), 0 < \nu < 1$ dată de expresia

$$f(x) = \begin{cases} 0, \, daca \, x < 0, \\ \frac{1}{\Gamma(\nu)} x^{\nu - 1} e^{-x}, \, daca \, x \geq 0, \end{cases}$$

sub forma

$$f(x) = p_1 f_1(x) + p_2 f_2(x),$$

$$f_1(x) = \begin{cases} \frac{f(x)}{p_1}, \, daca \, x \in [0, 1], \\ 0, \, altfel \end{cases}, \quad f_2(x) = \begin{cases} \frac{f(x)}{p_2}, \, daca \, x \in (1, +\infty), \\ 0, \, altfel \end{cases} \qquad (3.29)$$

$$p_1 = \frac{\Gamma(1; \nu)}{\Gamma(\nu)}, \, p_2 = 1 - p_1 = \frac{\Gamma(\nu) - \Gamma(1; \nu)}{\Gamma(\nu)},$$

$$\Gamma(\nu)=\int_0^{\infty}x^{\nu-1}e^{-x}dx,\ \Gamma(1;\nu)=\int_0^{1}x^{\nu-1}e^{-x}dx.$$

Să notăm $X_1\rightsquigarrow f_1(x)$, $X_2\rightsquigarrow f_2(x)$. Atunci are loc următoarea teoremă

Teorema 3. 9 *Variabila X_1 se simulează folosind Teorema 3.4 a subşirului descendent unde $Z_0=U_0^{1/\nu}$, $Z_i=U_i$, cu $\{U_i\}_{i\geq 0}$ uniforme $0-1$, iar X_2 se simulează cu ajutorul Teoremei 3.3 duale unde densitatea $f_2(x)$ este de forma*

$$f_2(x)=c(1-Q(x))r(x),\ x>0,\quad c=\frac{1}{e(\Gamma(\nu)-\Gamma(1;\nu))},$$

$$r(x)=\begin{cases}0,\ daca\, x<1\\ e^{-x+1},\ daca\, x\geq 1\end{cases},\quad Q(x)=\begin{cases}0,\ daca\, x<1,\\ 1-x^{\nu-1},\ daca\, x\geq 1.\end{cases}$$

Demonstraţie [10]. Intr-adevăr, din enunţul teoremei rezultă că pentru $x\in[0,1]$, in Teorema 3.4 avem

$$G_0(x)=P(U_0^{1/\nu}<x)=P(U_0<x^{\nu})=x^{\nu},$$

$$H(x)=P(U_0<x|K=nr.impar)=\frac{1}{p_a}\int_0^{x}e^{-t}dG_0(t)=\frac{1}{p_a}\int_0^{x}e^{-t}\nu t^{\nu-1}dt$$

$$p_a=\int_0^{1}e^{-t}\nu t^{\nu-1}dt=\nu\Gamma(1;\nu),$$

de unde derivând $H(x)$ avem

$$H'(x)=h(x)=\frac{e^{-x}x^{\nu-1}}{\Gamma(1;\nu)}=f_1(x),\quad x\in[0,1]$$

şi prima parte a teoremei este demonstrată.

Cea de-a doua parte a teoremei se demonstrează dacă scriem forma detaliată a lui $f_2(x)$ adică

$$f_2(x)=\begin{cases}0,\ daca\, x<1,\\ \frac{x^{\nu-1}e^{-x}}{\Gamma(\nu)-\Gamma(1;\nu)}\end{cases},\ x\geq 1$$

care se observă că se reprezintă sub forma din enunţul teoremei.

Demonstraţia este completă.

Presupunând că s-au calculat in prealabil

$$g = \Gamma(\nu),\ g_1 = \Gamma(1;\nu),\ p_1 = \frac{g_1}{g},\ p_2 = \frac{g - g_1}{g},$$

$$c = \frac{1}{e(\Gamma(\nu) - \Gamma(1;\nu))};\quad a = \frac{1}{\nu},\quad b = -\frac{1}{1-\nu}$$

algoritmul de simulare al variabilei $X \rightsquigarrow Gamma(0,1,\nu)$, $0 < \nu < 1$, este

Algoritmul GAMCOMNL1 pentru simularea variabilei gamma de parametru subunitar prin compunere şi respingere:

Intrare p_1, p_2;
Generează $U \rightsquigarrow uniform\, 0-1$;
if $U \leq p_1$ **then begin**
Generează $X_1 \rightsquigarrow f_1(x)$; Ia $X := X_1$;
end
else begin
Generează $X_2 \rightsquigarrow f_2(x)$; Ia $X := X_2$;
end.

Variabilele X_1 şi X_2 se generează cu algoritmii de respingere prezentaţi in continuare.

Algoritmul GAMRESNL1 pentru simularea variabilei X_1 :
repeat
Generează $U := random$; (U=uniform $0-1$);
Ia $Z_0 := U^a$; *Generează* $Z_1 := random$; *Ia* $K := 1$; $Z^* := Z_0$;
While $Z_0 \geq Z_1$ **do begin**
$Z_0 := Z_1$; *Generează* $Z_1 := random$; $K := K + 1$;
end;
until $Kmod2 = 1$;
Ia $X_1 := Z^*$.

Pentru acest algoritm avem

$$p_a = \nu\Gamma(1;\nu);\ E(N_1^*) = \frac{1}{p_a}(1 + \int_0^1 \nu t^{\nu-1} e^t dt)$$

ultima integrală putând fi calculată numeric (de ex şi cu metoda Monte Carlo, cf. Cap.6).

Algoritmul GAMRESNL2 pentru simularea variabilei X_2 :
repeat
Generează $U := random$; *Ia* $Z := U^b$;

Generează $X_0 \rightsquigarrow Exp(1)$; *Ia* $Y := X_0 + 1$; $(Y \rightsquigarrow r(x))$;
until $Y \leq Z$;
Ia $X_2 := Y$.

Pentru acest algoritm avem

$$p_2 = 1 - p_1; E(N_2^*) = \frac{2}{p_2}$$

deci numărul mediu de variabile $Z_i, i \geq 0, Z, Y$ necesare pentru a genera in final un X este

$$E(N^*) = p_1 E(N_1^*) + p_2 E(N_2^*) = p_1 E(N_1^*) + 2.$$

• **Metode pentru simularea variabilei** $Gamma(0, 1, \nu)$, $\nu > 1$. Vom prezenta doi algoritmi de respingere bazaţi pe metoda infăşurătoarei.

G1. Să considerăm $X \rightsquigarrow f(x) \rightsquigarrow Gamma(0, 1, \nu), \nu > 1$ şi să luăm ca infăşurătoare densitatea $h(x) \rightsquigarrow Exp(\frac{1}{\nu})$ adică

$$h(x) = \begin{cases} 0, \, daca \, x < 0, \\ \frac{1}{\nu} e^{-\frac{x}{\nu}}, \, daca \, x \geq 0. \end{cases} \tag{3.30}$$

Analizând raportul

$$r(x) = \frac{f(x)}{h(x)} = \frac{\nu x^{\nu-1} e^{-x}}{\Gamma(\nu) e^{-\frac{x}{\nu}}}, \, \nu > 1$$

se constată că acesta are ca punct de maxim $x = \nu$, deci constanta α din Teorema 3.2 (a infăşurătoarei) este

$$\alpha = r(\nu) = \frac{\nu^\nu e^{1-\nu}}{\Gamma(\nu)}.$$

Algoritmul de respingere in acest caz este simplu de construit şi nu-l mai prezentăm. Probabilitatea de acceptare este deci

$$p_a = \frac{\Gamma(\nu)}{\nu^\nu e^{1-\nu}} \approx \sqrt{\frac{e^2 2\pi}{\sqrt{\nu - 1}}}$$

unde in ultima relaţie s-a folosit aproximarea lui Stirling pentru $\nu \to \infty$ adică

$$\Gamma(\nu) \approx (\nu - 1)^{\nu-1} e^{-(\nu-1)} \sqrt{2\pi(\nu - 1)},$$

deci această probabilitate scade de ordinul $\sqrt{\nu}$.

G2 Algoritmul precedent este lent pentru ν foarte mare. De aceea vom prezenta in acest caz o altă metodă de respingere, bazată pe infăşurarea cu o *densitate Cauchy nonstandard trunchiată pe* $[0,\infty)$ de forma

$$h(x)=\frac{k}{1+\frac{(x-(\nu-1))^2}{c}},\quad x\geq 0 \tag{3.31}$$

unde k este o constantă de normare. Putem enunţa teorema [10]

Teorema 3. 10 *Dacă infăşurăm densitatea* $Gamma(0,1,\nu)$, $\nu>1$, *cu densitatea* $h(x)$ *dată de (3.31) atunci pentru* $c\geq 2\nu-1$ *avem*

$$r(x)=\frac{f(x)}{h(x)}\leq \alpha=\frac{1}{k\Gamma(\nu)}(\nu-1)^{\nu-1}e^{-(\nu-1)}. \tag{3.31'}$$

Demonstraţie. Avem

$$r(x)=\frac{1}{k\Gamma(\nu)}\varphi(x),\quad \varphi(x)=x^{\nu-1}e^{-x}\left[1+\frac{(x-(\nu-1))^2}{c}\right]$$

$$\varphi'(x)=-\frac{e^{-x}x^{\nu-1}}{c}[x-(\nu-1)][(x-\nu)^2+c-(2\nu-1)]$$

de unde rezultă că ecuaţia $\varphi'(x)=0$ are soluţia $x_0=\nu-1>0$ iar dacă $c\geq 2\nu-1$ atunci x_0 este punct de maxim. Dacă luăm $c=2\nu-1$ atunci avem

$$\alpha=\frac{1}{k\Gamma(\nu)}(\nu-1)^{\nu-1}e^{-(\nu-1)}$$

ceeace demonstrează teorema.

Pentru a descrie algoritmul dedus din teorema precedentă presupunem calculate in prealabil constantele

$$b=\nu-1,\ c=\nu+b,\ s=\sqrt{2\nu-1}.$$

Atunci algoritmul pentru generarea variabilei $X\rightsquigarrow Gamma(0,1,\nu)$, $\nu>1$ este este

Algoritmul GAMCAUCHY (simularea variabilei $Gamma(0,1,\nu)$, $\nu>1$ prin infăşurarea cu o densitate Cauchy)
repeat
  repeat
  Generează $U:=$ *random şi ia* $T=s.tg[\pi(U-0.5)]$; (T este Cauchy standard pe $(-\infty,+\infty)$);

Ia $Y = b + T$; (Y este Cauchy non standard pe $(-\infty, +\infty)$);
until $Y > 0$; (Se aplica respingerea pentru a obţine Y-trunchiată);
Generează $U_1 := random$;
until $U_1 \leq e^{b\log(Y/b) - T + \log(1+T^2/c)}$;
Ia $X = Y$.

Se observă că constanta k nu intervine in construcţia algoritmului dar ea este necesară pentru a calcula $p_a = \frac{1}{\alpha}$. Un calcul simplu arată că

$$k = \left[\frac{\pi}{2} + arctg(-\frac{\nu - 1}{\sqrt{2\nu - 1}})\right]^{-1}.$$

Referitor la repartiţia $Gamma(0, 1, \nu), \nu > 1$ mai observăm şi faptul că dacă descompunem ν sub forma $\nu = k + p$, $k = [\nu] \in N^+, p = \nu - k \in [0, 1)$ şi considerăm variabilele $X \rightsquigarrow Gamma(0, 1, \nu)$, $E_k \rightsquigarrow Erlang(k)$, $Y \rightsquigarrow Gamma(0, 1, p)$, atunci simularea lui X se realizează cu relaţia $X = E_k + Y$, $(E_k, Y)-$independente.

3.5.3 Repartiţia Beta

Variabila X are repartiţia $Beta(a, b), a > 0, b > 0$ [5, 7, 10, 11] dacă densitatea sa de repartiţie este

$$f(x) = \begin{cases} \frac{1}{B(a,b)} x^{a-1}(1-x)^{b-1}, daca\, x \in [0,1] \\ 0,\ altfel \end{cases} \tag{3.32}$$

unde

$$B(a, b) = \int_0^1 x^{a-1}(1-x)^{b-1}dx, \quad B(a, b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}. \tag{3.32'}$$

Următoarea teoremă [10] permite simularea variabilei $Beta(a, b)$.

Teorema 3. 11 *Dacă* $X_1 \rightsquigarrow Gamma(0, 1, a)$, $X_2 \rightsquigarrow Gamma(0, 1, b)$, X_1 *independent de* X_2 *atunci variabila*

$$X = \frac{X_1}{X_1 + X_2} \tag{3.33}$$

este o variabilă $Beta(a, b)$.

Demonstraţie. Densitatea comună de repartiţie a lui (X_1, X_2) este

$$f(x_1, x_2) = \frac{1}{\Gamma(a)\Gamma(b)} x_1^{a-1} x_2^{b-1} e^{-(x_1+x_2)}.$$

Făcând in ultima integrala transformarea

$$u = \frac{x_1}{x_1 + x_2}, \quad v = x_2, \quad J = \frac{\partial(x_1, x_2)}{\partial(u, v)} = \frac{v}{1-u}, \; 0 < u < 1,$$

avem

$$f(x_1(u,v), x_2(u,v)) = g(u,v) = \frac{1}{\Gamma(a)\Gamma(b)} \frac{u^{a-1} v^{a+b-1}}{(1-u)^a} e^{-\frac{v}{1-u}}, \; 0 < v < \infty.$$

Densitatea de repartiţie a variabilei $X_1/(X_1 + X_2)$ este

$$h(u) = \int_0^\infty g(u,v)dv = \frac{1}{\Gamma(a)\Gamma(b)} \int_0^\infty \frac{u^{a-1} v^{a+b-1}}{(1-u)^{a+1}} e^{-\frac{v}{1-u}} dv$$

adică după calcule deducem

$$h(u) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} u^{a-1}(1-u)^{b-1}.$$

şi teorema este demonstrată.

Simularea variabilei $Beta(a, b)$ poate fi deci făcută cu (3.33). Dar deoarece această formulă presupune generarea prealabilă a două variabile gamma, rezultă că această metodă are o complexitate mare. De aceea, in cazuri particulare se pot folosi următoarele teoreme [5, 10]:

Teorema 3. 12 *Fie $a, b \in N^+$ şi $n = a + b - 1$ şi să considerăm variabilele $U_1, U_2, ..., U_n$ uniforme $0-1$. Să construim statisticile de ordine $U_{(1)} < U_{(2)} < ... < U_{(n)}$ prin ordonarea valorilor $\{U_i\}_{1 \le i \le n}$. Atunci $U_{(a)} \rightsquigarrow Beta(a, b)$.*

Nu prezentăm demonstraţia acestei teoreme.

Teorema 3. 13 *Dacă $0 < a < 1, 0 < b < 1$ şi U_1, U_2 sunt variabile aleatoare uniforme $0 - 1$ şi independente şi dacă $V = U_1^{\frac{1}{a}}, T = U_2^{\frac{1}{b}}$, atunci repartiţia variabilei $X = \frac{V}{V+T}$ condiţionată de $V + T < 1$ este $Beta(a, b)$.*

Teorema 3. 14 *Dacă $0 < a < 1, b > 1$ şi U_1, U_2 sunt variabile uniforme $0 - 1$ independente şi considerăm $V = U_1^{\frac{1}{a}}, T = U_2^{\frac{1}{b-1}}$, atunci repartiţia variabilei V condiţionată de $V + T < 1$ este $Beta(a, b)$.*

Demonstraţie. Intrucât demonstraţiile ultimelor două teoreme sunt asemănătoare, vom prezenta numai demonstraţia ultimei teoreme, folosind calea urmată cu ocazia demonstraţiei teoremei 3.11.

Mai intâi observăm că

$$F(x) = P(V < x) = P(U^{\frac{1}{a}} < x) = P(U < x^a) = x^a,\ x \in [0,1]$$

de unde rezultă că densitatea de repartiţie a lui V este

$$f(x) = ax^{a-1}, \quad x \in [0,1].$$

In mod asemănător densitatea de repartiţie a lui T este

$$h(y) = (b-1)y^{b-2}, \quad y \in [0,1].$$

Rezultă că densitatea comună de repartiţie a variabilelor V, T-independente este

$$g(x,y) = a(b-1)x^{a-1}y^{b-2}$$

iar

$$P(V+T<1) = a(b-1)\int_0^1 (\int_0^{1-x} y^{b-2}dy)x^{a-1}dx = aB(a,b).$$

Deci densitatea comună a variabilelor (V,T) condiţionată de $V+T<1$ este

$$p(x,y) = \frac{b-1}{B(a,b)}x^{a-1}y^{b-2}, x \in [0,1], y \in [0,1].$$

De aici rezultă că densitatea lui V condiţionată de $V+T<1$ este

$$q(x) = \int_0^{1-x} p(x,y)dy = \frac{1}{B(a,b)}x^{a-1}(1-x)^{b-1}$$

şi teorema este demonstrată.

Algoritmii ce rezultă din aceste ultime teoreme sunt uşor de construit. Vom prezenta algoritmul ce rezultă din teorema 3.13.

Algoritmul BETA13 pentru simularea variabilei $Beta(a,b), 0 < a, b < 1$

repeat

Generează U_1, U_2 *uniforme* $0-1$ *şi independente;*

Ia $V = U_1^{\frac{1}{a}}, T = U_2^{\frac{1}{b}}$;

until $V + T < 1$;

Calculează $X = \frac{V}{V+T}$.

Probabilitatea de acceptare a acestui algoritm de respingere este

$$p_a = \frac{ab}{a+b}B(a,b). \tag{3.34}$$

Algoritmul de respingere construit pe baza teoremei 3.14 are probabilitatea de acceptare

$$p_a = aB(a,b). \tag{3.34'}$$

3.5.4 Repartiţia normală

Desigur, ne vom opri la simularea variabilei $Z \rightsquigarrow N(0,1)$ căci dacă ştim să simulăm această variabilă atunci $X \rightsquigarrow N(m,\sigma)$ se simulează cu formula $X = m+Z\sigma$. In tabelul 3.2 este prezentată metoda de simulare a variabilei $N(0,1)$, bazată pe teorema limită centrală, dar această metodă este aproximativă. De aceea in această subsecţiune vom prezenta alte metode, exacte.

• **O metodă de compunere-respingere.** Să considerăm variabila normală standard $N(0,1)$, notată cu X_1, care are densitatea

$$f_1(x) = \begin{cases} 0, \, daca\, x < 0 \\ \sqrt{\frac{2}{\pi}}e^{-\frac{x^2}{2}}, \, daca\, x \geq 0. \end{cases} \tag{3.35}$$

De aici deducem că densitatea lui $X_2 = -X_1$ este

$$f_2(x) = \begin{cases} \sqrt{\frac{2}{\pi}}e^{-\frac{x^2}{2}}, \, daca\, x < 0 \\ 0, daca\, x \geq 0 \end{cases}. \tag{3.35'}$$

Deci densitatea lui $Z \rightsquigarrow N(0,1)$ se scrie

$$f(x) = \frac{1}{2}f_1(x) + \frac{1}{2}f_2(x)$$

adică este o compunere discretă a densităţilor $f_1(x)$ şi $f_2(x)$. Pentru a construi un algoritm de simulare a lui Z va trebui să construim mai intâi un algoritm de simulare a lui X_1.

Vom infăşura densitatea $f_1(x)$ cu $h(x) \rightsquigarrow Exp(1)$. Rezultă teorema

Teorema 3. 15 *Dacă infăşurăm* $f_1(x)$ *cu* $h(x)$ *avem*

$$\frac{f_1(x)}{h(x)} \leq \alpha = \sqrt{\frac{2e}{\pi}} \tag{3.36}$$

şi deci putem aplica teorema de respingere a infăşurătoarei pentru simularea variabilei X_1.

Demonstraţie. Observăm că

$$r(x) = \frac{f_1(x)}{h(x)} = \sqrt{\frac{2}{\pi}} e^{-\frac{x^2}{2}+x}$$

iar ecuaţia $r'(x) = 0$, are soluţia $x_0 = 1$ care este un punct de maxim pentru $r(x)$, adică,

$$r(x) \leq r(x_0) = \sqrt{\frac{2e}{\pi}}$$

ceea ce demonstrează teorema.

Algoritmul pemtru simularea lui $Z \rightsquigarrow N(0,1)$ este

Algoritmul RJNORM de compunere respingere pentru simularea normalei $N(0,1)$

repeat

Generează U *uniform* $0-1$;

Generează $Y \rightsquigarrow Exp(1)$;

until $u \leq e^{-\frac{Y^2}{2}+Y-0.5}$;

Ia $X_1 := Y$;

Generează U *uniform* $0-1$;

if $U \leq 0.5$ **then** $s := 1$ **else** $s := -1$; (s este un *semn aleator*);

Ia $Z := sX_1$.

Se observă că probabilitatea de acceptare este

$$p_a = \sqrt{\frac{\pi}{2e}} \approx 0.72 \tag{3.36'}$$

adică, in medie, din patru perechi (U, Y), trei sunt acceptate pentru a produce un X_1.

• **Metoda polară.** O altă metodă interesantă de simulare a variabilei $N(0,1)$ este *metoda polară* dată de următoarea teoremă [10] datorată lui Box şi Muller.

Teorema 3. 16 *Dacă variabilele* U_1, U_2 *sunt uniforme* $0-1$ *şi independente, atunci variabilele aleatoare*

$$Z_1 = V_1\sqrt{\frac{-2logS}{S}}, \quad Z_2 = V_2\sqrt{\frac{-2logS}{S}} \tag{3.37}$$

unde

$$V_1 = 2U_1 - 1, \quad V_2 = 2U_2 - 1, \quad S = V_1^2 + V_2^2,\ S < 1$$

sunt variabile normale $N(0,1)$ *independente.*

Demonstraţie. Să observăm mai intâi că (V_1, V_2) este un vector aleator uniform pe pătratul $I^2 = [-1,1] \times [-1,1]$ iar $V_i, i = 1,2$ sunt uniforme pe $[-1,1]$ şi independente. Condiţia $S < 1$ face ca vectorul $(V_1, V_2)|\{S < 1\}$ (condiţionat de $S < 1$!) să fie vector uniform pe cercul unitate. De aceea să exprimăm V_i in *coordonate polare,* adică

$$V_1 = Rcos\theta, \quad V_2 = Rsin\theta, \quad 0 \leq R \leq 1,\ 0 \leq \theta \leq 2\pi.$$

Identificând ultimele relaţii cu (3.37) rezultă că

$$S = R^2, \quad Z_1 = \sqrt{-2logS}\, cos\theta,\ Z_2 = \sqrt{-2logS}\, sin\theta. \tag{3.37'}$$

Pe de altă parte, insăşi Z_1, Z_2 se pot exprima direct in coordonate polare şi anume

$$Z_1 = R'cos\theta', \quad Z_2 = R'sin\theta' \tag{3.38}$$

de unde identificând (3.37') cu (3.38) avem

$$\theta' = \theta, \quad R' = \sqrt{-2logS}$$

şi deoarece (V_1, V_2) independente pe I^2, rezultă că şi (R, θ) sunt independente, precum şi (R', θ') sunt independente (dar nu pe cercul unitate!). Deoarece (V_1, V_2) are o repartiţie uniformă pe cercul unitate, rezultă că θ' are o repartiţie uniformă pe $[0, 2\pi]$ adică are densitatea

$$\varphi(\theta) = \begin{cases} \frac{1}{2\pi},\ daca\, \theta \in [0, 2\pi] \\ 0,\ altfel. \end{cases} \tag{3.39}$$

Să determinăm acum repartiţia lui R'. Avem

$$F(r) = P(R' \leq r) = P(\sqrt{-2logS} \leq r).$$

Dar deoarece $S = R^2$ este uniformă pe $[0,1]$ rezultă că

$$F(r) = P(S > e^{-\frac{r^2}{2}}) = 1 - e^{-\frac{r^2}{2}}$$

deci densitatea de repartiţie a lui R' este

$$\psi(r) = re^{-\frac{r^2}{2}}, \quad r \in [0,1]. \tag{3.40}$$

Pentru a demonstra teorema trebuie să arătăm că funcţia de repartiţie a variabilelor Z_1, Z_2 date de (3.37) este produsul a două funcţii de repartiţie normale $N(0,1)$. Pentru aceasta să considerăm domeniile

$$D_{(r,\theta)} = \{(r,\theta); rcos\theta \leq z_1,\ rsin\theta \leq z_2\}$$

$$D_{(x,y)} = \{(x,y); x \leq z_1,\ y \leq z_2\}.$$

Avem

$$F(z_1,z_2) = P(Z_1 \leq z_1, Z_2 \leq z_2) = \int\int_{D_{(r,\theta)}} \frac{1}{2\pi} e^{-\frac{r^2}{2}} r dr d\theta$$

şi după efectuarea schimbărilor de variabilă

$$\theta = arctg(\frac{x}{y}), \quad r = \sqrt{x^2+y^2}$$

deducem

$$F(z_1,z_2) = \frac{1}{2\pi}\int\int_{D_{(x,y)}} e^{-\frac{x^2+y^2}{2}} dxdy = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{z_1} e^{-\frac{x^2}{2}} dx \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{z_2} e^{-\frac{y^2}{2}} dy$$

şi teorema este demonstrată.

Algoritmul corespunzător metodei polare se deduce cu uşurinţă şi el produce simultam două valori de selecţie $N(0,1)$, Z_1 şi Z_2, independente.

Din demonstraţia teoremei rezultă că variabilele Z_1, Z_2 pot fi simulate şi cu formulele

$$Z_1 = \sqrt{-2logU_1}cos(2\pi U_2),\ Z_2 = \sqrt{-2logU_1}sin(2\pi U_2). \tag{3.41}$$

In varianta bazată pe (3.37) se resping valorile pentru care $S \geq 1$ si deci probabilitatea de acceptare este

$$p_a = \frac{\pi}{4} \tag{3.41$'$}$$

in timp ce varianta bazată pe (3.41) nu presupune nicio respingere. Totuşi complexitatea calculului expresiilor (3.41) poate să mărească timpul de calcul faţă de cazul (3.37), ceea ce poate da câştig de cauză variantei (3.37) (din cauza funcţiilor sin, log,etc., consumatoare de timp).

3.6 Simularea unor variabile discrete

Pentru simularea unei variabile discrete ce ia un număr finit de valori se poate folosi algoritmul **SIMDISCRV** din secţiunea 3.1 bazat pe metoda inversă. Dacă X este o variabilă discretă care ia ca valori şirul $\{a_n\}_{1\leq n\leq\infty}$ şi se cunoaşte *funcţia de fracvenţă* $f(i) = P(X = a_i), f-$calculabilă, atunci folosind proprietatea $\lim_{i\to\infty} f(i) = 0$, (dedusă din $\sum_{i=1}^{\infty} f(i) = 1$) se poate construi uşor un algoritm bazat pe metoda respingerii (măcar aproximativ!) şi in acest caz.

In această secţiune vom prezenta insă algoritmi pentru simularea unor repartiţii particulare.

3.6.1 Simularea unor repartiţii bazate pe probe Bernoulli

Vom prezenta mai intâi ce inţelegem prin *probe Bernoulli* [10].

Fie un eveniment aleator observabil A care are probabilitatea *constantă* $p = P(A) > 0$. Intr-o experienţă intâmplătoare se poate produce A cu probabilitatea p sau evenimentul contrar $\overline{A}$ cu probabilitatea $q = 1 - p$. O astfel de experienţă se numeşte *probă Bernoulli.* Când se produce A spunem că avem de-a face cu un *succes,* iar când a nu se produce spunem că se realizează un *eşec.* Să asociem unei probe Bernoulli variabila aleatoare Z astfel incât $Z = 1$ dacă se produce A şi $Z = 0$ dacă se produce $\overline{A}$, adică Z are repartiţia

$$Z : \begin{pmatrix} 0 & 1 \\ q & p \end{pmatrix}, \; E(Z) = p, \; Var(Z) = pq = p(1-p). \tag{3.42}$$

Funcţia de repartiţie a lui Z este

$$F(x) = P(Z < x) = \begin{cases} 0, & daca \quad x < 0 \\ q, & daca \quad 0 \leq x < 1 \\ 1 & daca \quad x \geq 1. \end{cases} \tag{3.42'}$$

De aici rezultă că algoritmul de simulare a lui Z prin metoda inversă este

Algoritmul BERN de simulare a unei variabile (probe) Bernoulli
Generează $U := random$;
if $U > p$ **then** $Z := 0$ **else** $Z := 1$.

Să observăm din nou că dacă $p = \frac{1}{2}$ atunci suntem in cazul particular al *aruncării la intâmplare cu banul (tragerea la sorţi).*

• **Repartiţia binomială.** Se spune că variabila aleatoare discretă $X \in N$ este o *variabilă binomială* $Binom(n, p), n \in N^+, 0 < p < 1$ dacă $X =$*numărul de succese in n probe Bernoulli independente,* adică

$$X = \sum_{i=1}^{n} Z_i$$

unde Z_i sunt variabile identic repartizate Bernoulli, independente.

Simularea variabilei X se face deci simplu, prin *numărarea de succese in n probe Bernoulli independente.*

Se poate deduce cu uşurinţă că

$$P(X = \alpha) = C_n^{\alpha} p^{\alpha} q^{n-\alpha}, \quad q = 1 - p,$$

adică $P(X = \alpha)$ este termenul general al dezvoltării binomului $(p + q)^n$, de unde derivă şi denumirea de *repartiţie binomială.*

Funcţia caracteristică a variabilei binomiale este

$$\varphi(t) = E[e^{itX}] = E[e^{it\sum_j Z_j}] = (q + pe^{it})^n. \tag{3.43}$$

Cu ajutorul funcţiei caracteristice se calculează uşor media şi dispersia lui X, adică

$$EX = E(\sum_{i=1}^{n} Z_i) = \sum_{i=1}^{n} E(Z_i) = np,$$

$$Var(X) = Var(\sum_{i=1}^{n} Z_i) = \sum_{i=1}^{n} Var(Z_i) = npq \tag{3.43'}$$

iar din *teorema limită centrală* se deduce că pentru n suficient de mare ($n \to \infty$) variabila

$$W_n = \frac{X - np}{\sqrt{npq}} \rightsquigarrow N(0, 1).$$

De aici rezultă următorul algoritm simplu de generare a lui $X \rightsquigarrow Binom(n, p), n = mare$ [10].

Algoritmul BINNORM

Generează $W \rightsquigarrow N(0, 1)$;

Calculează $X := \{np+W\sqrt{npq}\}$. (Notaţia $\{E\}$ inseamnă "cel mai apropiat intreg de E").

Observaţie. Variabila $Binom(n, p)$ are şi o interpretare in termeni de experiment cu o urnă. Astfel, să presupunem că intr-o urnă avem A bile albe şi B bile negre, $A + B = N$. Presupunem că se realizează extracţii succesive din urnă şi după fiecare extracţie se introduce bila extrasă la loc in urnă (*experienţa cu bila "intoarsă"*). Fie $p = A/N$ probabilitatea de a extrage o bilă albă intr-o extracţie. De aici rezultă că X= numărul de bile albe in n extracţii succesive *cu intoarcere* este o variabilă binomială $Binom(n, p)$.

• **Repartiţia Pascal** [10]. Variabila X are *repartiţia* $Pascal(k, p), k \in N^+, 0 < p < 1$, dacă $X =$ *numărul de eşecuri până la apariţia a k succese* intr-un şir oarecare de probe Bernoulli independente. De aici rezultă că variabila $X \rightsquigarrow Pascal(k, p)$ se simulează cu următorul algoritm care numără eşecurile realizate până la realizarea a k succese intr-un şir de probe Bernoulli independente.

Algoritmul PASCAL

Intrare $k \in N^+, p, 0 < p < 1$; $X := 0; j := 0$; (X numără eşecurile şi j-succesele);
repeat
Generează $U := random$; **if** $U < p$ **then** $j := j + 1$ **else** $X := X + 1$;
until $j = k$.(X este valoarea de selecţie generată).

Să observăm că

$$P(X = \alpha) = C_{\alpha+k-1}^{k-1} p^k q^\alpha, \ \alpha = 0, 1, 2, ...$$

care este termenul general al dezvoltării in serie a expresiei $p^k(1-q)^{-k}$ din care cauză repartiţia $Pascal(k, p)$ se mai numeşte şi *repartiţia binomială cu exponent negativ.* De aici se deduce şi faptul că dacă $X_1 \rightsquigarrow Pascal(k_1, p)$ şi $X_2 \rightsquigarrow Pascal(k_2, p)$, sunt variabile independente, atunci $(X = X_1 + X_2) \rightsquigarrow Pascal(k_1 + k_2, p)$, adică repartiţia Pascal este *stabilă.* (Demonstraţia se poate face simplu utilizând funcţia caracteristică).

Se arată că

$$E(X) = \frac{kq}{p}, \quad Var(X) = \frac{kq}{p^2}, \tag{3.44}$$

formule ce se folosesc la validarea algoritmului.

Observaţie. Interpretarea *cu urnă* din cazul repartiţiei $Binom(n, p)$ se poate adapta şi in cazul repartiţiei $Pascal(k, p)$. Astfel, numărul X de bile negre extrase *cu intoarcere* până când se obţin k bile albe, este o variabilă $Pascal(k, p)$.

• **Repartiţia geometrică** $Geom(p)$ [10] este un caz particular de repartiţie Pascal, când $k = 1$. Simularea variabilei $X \rightsquigarrow Geom(p)$ se poate realiza cu algoritmul **PASCAL** sau cu metoda inversă după cum urmează:

Fie $P(X = x) = pq^x, x = 0, 1, 2, ...$ şi să notăm

$$F(x) = P(X < x) = \sum_{i=0}^{x-1} pq^i = 1 - q^x, x = 0, 1, 2, ...$$

care este o funcţie de repartiţie discretă. ($P(X = x)$ este termenul unei *progresii geometrice,* de unde şi numele de repartiţie geometrică). Simularea variabilei geometrice se poate realiza prin metoda inversă cu formula

$$X = \left[\frac{log(U)}{log(q)}\right]. \tag{3.45}$$

unde$[a]$ este partea intreagă a lui a. Din (3.44) se deduce că pentru variabila geometrică X avem

$$E(X) = \frac{q}{p}, \quad Var(X) = \frac{q}{p^2}, \tag{3.44'}$$

formule care de asemena se pot folosi la validarea algoritmului.

3.6.2 Repartiţia hipergeometrică

Această repartiţie se introduce după cum urmează [10].

Să considerăm experimentul cu urnă descris in legătură cu repartiţia binomială, cu deosebirea că aici cele n bile se extrag la intâmplare din urnă, *fără intoarcere.* In acest caz numărul X de bile albe extrase este o *variabilă hipergeometrică.* Să notăm cu u evenimentul care reprezintă extragerea unei bile albe şi cu v evenimentul care constă in extragerea unei bile negre; atunci probabilităţile de a extrage in prima extragere o bilă albă respectiv neagră, sunt respectiv $p = P(u) = A/N$, $P(v) = B/N$. Probabilităţile de extragere a unei bile albe sau negre in a doua extragere sunt condiţionate de rezultatele primei extrageri adică

$$P(u|u) = \frac{A-1}{N-1},\ P(u|v) = \frac{A}{N-1},\ P(v|u) = \frac{B}{N-1},\ P(v|v) = \frac{B-1}{N-1}.$$

Se observă deci că la fiecare extragere *compoziţia urnei se schimba* şi probabilitatea de a extrage o bilă albă sau neagră este variabilă in funcţie de extragerile anterioare. Variabila hipergeometrică se notează $H(N, p, n)$, $0 < p < 1$, $n < N$, de unde $A = \{Np\}$ ($\{x\}, x \in R$ este *cel mai apropiat intreg de* x), $B = N - A$. Se demonstrează că probabilitatea ca in n extracţii succesive *fără intoarcere,* să se extragă a bile albe este

$$P(X = a) = \frac{C_A^a C_B^{n-a}}{C_N^n}, \quad 0 \le a \le n, \quad n < N. \tag{3.46}$$

De asemenea se arată că

$$E(X) = np, \quad E(X^2) = \frac{np}{N-1}[n(Np-1) + N(1-p)],$$

$$Var(X) = np(1-p)\frac{N-n}{N-1}. \tag{3.47}$$

Având in vedere cele menţionate, algoritmul de simulare a variabilei hipergeometrice X este

Algoritmul HIPERGEOM

Intrare A, B, N, $N = A + B$, n;(Acesta este un pas pregăritor);

Calculează $p = A/N$; *Iniţializează* $j := 0$, $X := 0$;

repeat

Generează $U := random$; *Ia* $j := j + 1$;

if $U < p$ **then begin**
$X := X + 1$; $S := 1$ (S-a estras o bilă albă);
end else$S := 0$; (S-a extras o bilă neagră);
Calculează $N := N - 1$, $A := A - S$, $p := \frac{A}{N}$;
until $j = n$. $(X \rightsquigarrow H(N, p, n))$;

Formulele (3.47) se pot folosi la validarea algoritmului.

3.6.3 Repartiţia Poisson

Variabila aleatoare X, $X \in \mathcal{N}$ are repartiţia $Poisson(\lambda)$, $\lambda > 0$ dacă

$$P(X = \alpha) = \frac{\lambda^\alpha}{\alpha!} e^{-\lambda}, \quad \lambda > 0. \tag{3.48}$$

Funcţia caracteristică a variabilei $Poisson(\lambda)$ este

$$\varphi(t) = E(e^{itX}) = \sum_{\alpha=0}^{\infty} \frac{(e^{it}\lambda)^\alpha}{\alpha!} e^{-\lambda} = e^{\lambda e^{it}} e^{-\lambda} = e^{\lambda(e^{it}-1)} \tag{3.49}$$

de unde deducem

$$E(X) = \lambda, \quad E(X^2) = \lambda^2 + \lambda \quad Var(X) = E(X^2) - \lambda^2 = \lambda. \tag{3.49'}$$

Repartiţia $Poisson(\lambda)$ este repartiţia *evenimentelor rare* in sensul următor: evenimentele sunt independente şi se produc la intervale aleatoare de timp astfel incât un eveniment se produce pe intervalul de timp $[t, t + \Delta t]$ cu probalilitatea $\lambda \Delta t + O(\delta t)$ unde

$$\lim_{\Delta t \to 0} O(\Delta t) = 0, \quad \lim_{\Delta t \to 0} \frac{O(\Delta t)}{\Delta t} = 0$$

($O(\Delta T)$ este *neglijabilă* in raport cu Δt) iar probabilitatea ca pe acelaş interval să se producă *mai mult* de un eveniment (condiţia de *raritate*) este $O(\Delta t)$ (adică este neglijabilă). *Numărul de evenimente rare ce se produc pe unitatea de timp* este o variabilă aleatoare $Poisson(\lambda)$. Numărul λ este *intensitatea* cu care se produc evenimentele rare.

Se arată că intervalul de timp θ la care se produc două evenimente rare consecutive are o repartiţie $Exp(\lambda)$, fapt care spune că $X = j - 1$ dacă $\sum_{i=1}^{j-1} \theta_i \leq \lambda < \sum_{i=1}^{j} \theta_i$. Ţinând acum seama de faptul că $\theta_i = -logU_i/(\lambda)$, U_i uniforme $0 - 1$, atunci rezultă că

$$X = j - 1 \quad daca \quad , \prod_{i=1}^{j-1} U_i \geq e^{-\lambda} > \prod_{i=1}^{j} U_i. \tag{3.50}$$

Pe baza relaţiei (3.50) se poate construi cu uşurinţă un algoritm pentru simularea lui X iar cu formulele (3.49') se poate realiza validarea algoritmului.

Repartriţia Poisson poate fi dedusă din repartiţia binomială $Bin(n,p)$; să notăm $\lambda = np$ şi să presupunem că $n \to \infty$ şi $p \to 0$, λ rămânând constant. Tinând seama de (3.43) rezultă că funcţia caracteristică a variabilei $Binom(n,p)$ se scrie sub forma

$$\varphi(t) = \left(1 + \frac{\lambda(e^{it}-1)}{n}\right)^n \rightsquigarrow e^{\lambda(e^{it}-1)}$$

care conform (3.49) este funcţia caracteristică a variabilei Poisson.

De aici rezultă că simularea variabilei $Poissonb(\lambda)$ se poate realiza (aproximativ!) in felul următor.

0. Se alege o probabilitate $p \approx 0$ (de ex $p = 0.001$);

1. Se determină $n = \lambda/p$ intreg (n este in acest caz mare);

Se simulează $X \rightsquigarrow Binom(n,p)$.

(Simularea variabilei binomiale se poate face utilizând ca mai sus teorema limită centrală).

3.7 Validarea generatorilor

Vom prezenta in finalul acestui capitol cum se poate face *validarea algoritmilor* cu care se simulează diversele tipuri de variabile aleatoare. Validare inseamnă nu numai verificarea corectitudinii formale a programelor ci in special dacă algoritmul implementat (programat) produce valori de selectţie asupra variabilei aleatoare in cauză, adică dacă pe baza unei selecţii $X_1, X_2, ..., X_n$, de volum n suficient de mare se verifică ipoteza statistică $H : X \rightsquigarrow F(x)$, care se mai numeşte şi ipoteză *de concordanţă.*

Mai intâi ar trebui să verificăm intuitiv dacă *repartiţia empirică* sau *repartiţia de selecţie* se aseamănă cu cea *teoretică.* Mai precis trebuie să construim grafic *histograma* şi să vedem dacă ea se aseamănă cu densitatea de reparticţie.

• **Construcţia histogramei** se face astfel:

- se determină mai intâi $m = \min(X_1, X_2, ..., X_n)$,
$M = \max(X_1, X_2, ..., X_n)$ care reprezintă *intervalul de variaţie* $[m, M]$;

- se alege un intreg pozitiv k, (practica recomandă $15 \leq k \leq 40$) care reprezintă *numărul de intervale ale histogramei;*

- se imparte intervalul $[m, M]$ in k intervale $I_i = [a_{i-1}, a_i)$, $1 \leq i \leq k, a_0 = m$, $a_k = M$;

- se determină f_i = numărul valorilor de selecţie ce cad in intervalul $[a_{i-1}, a_i)$, $1 \leq i \leq k$; f_i se numesc *frecvenţe absolute;*

- se determină *frecvenţele relative* $r_i = \frac{f_i}{n}$.

Repartiţia empirică este

$$\begin{pmatrix} I_1, & I_2, & ..., & I_k \\ r_1, & r_2, & ..., & r_k \end{pmatrix}. \tag{3.51}$$

Acum să reprezentăm grafic (3.51) astfel: luăm pe abscisă intervalele I_i şi construim *dreptunghiuri* care au ca bază aceste intervale şi ca inălţimi r_i. Graficul obţinut reprezintă *histograma* selecţiei $X_1, X_2, ..., X_n$ date. (Ea este *histograma frecvenţelor relative*; asemănător se obţine şi *histograma frecvenţelor absolute*).

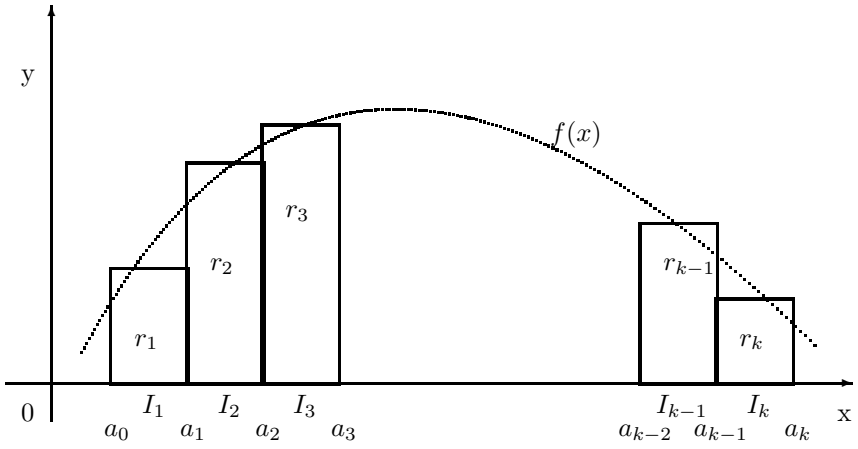

Fig.3.1. Histogramă.

In Fig. 3.1 se prezintă o histogramă; ea sugerează că linia punctată reprezintă forma *densităţii de repartiţie.* Din această reprezentare grafică putem formula ipoteza H cu privire la funcţia de repartiţie $F(x)$ a variabilei X asupra căreia s-a efectuat selecţia simulată $X_1, X_2, ..., X_n$. In Fig. 3.2 se prezintă forma graficelor unor densităţi de repartiţie considerate in acest capitol.

Când construim histograma unei selecţii obţinută prin simulare nu putem determina mai intâi m, M şi apoi să construim I_i şi $f_i, 1 \leq i \leq k$ decât dacă memorăm *ad literam* toată selecţia (ceea ce impune folosirea unei selecţii de

volum n mic) sau dacă *repetăm* simularea selecţiei folosind aceeaşi sămânţă a generatorului (ceea ce impune consum de timp de calcul dublu).

De aceea vom folosi următoarele idei pentru construcţia histogramei:

- simulăm mai intâi un număr mic n_1 de valori de selecţie $X_1, X_2, ..., X_{n_1}$, $n_1 << n$ pe care le memorăm;

- cu ajutorul acestei selecţii determinăm următoarele limite ale intervalelor histogramei: $a_1 = \min\{X_1, X_2, ..., X_{n_1}\}$, $a_{k-1} = \max\{X_1, X_2, ..., X_{n_1}\}$ şi cu ajutorul lor determinăm intervalele $I_2, ..., I_{k-1}$ de lungimi egale cu $h = (a_{k-1} - a_1)/(k-2)$, adică $I_i = [a_{i-1}, a_i)$, $2 \leq i \leq k-1$, $a_i = a_{i-1} + hi$;

- cu ajutorul selecţiei de volum n_1 determinăm (parţial!) frecvenţele absolute $f_i, 2 \leq i \leq k-1$ (utilizând pentru numărare formula (3.52) de mai jos ;deocamdată a_0 şi a_k nu sunt cunoscute!); luăm $a_0 = a_1$, $a_k = a_{k-1}$ care vor fi modificate ulterior; iniţializăm $f_1 = 0$, $f_k = 0$;

- simulăm pe rând celelalte $n - n_1$ valori de selecţie şi pentru fiecare X astfel simulat efectuăm următoarele operaţii: dacă $X < a_1$ atunci luăm $a_0 = min(a_0, X)$ şi $f_1 := f_1 + 1$; dacă $X > a_{k-1}$ atunci luăm $a_k = max(a_k, X)$ şi $f_k := f_k + 1$; altfel dacă $a_1 \leq X < a_{k-1}$ atunci calculăm

$$j := \left[\frac{X - a_1}{h}\right] + 2,\ f_j := f_j + 1; \tag{3.52}$$

Când s-au prelucrat toate cele $n - n_1$ valori de selecţie, s-au determinat toate elementele histogramei, care după reprezentarea şi interpretarea grafică conduce la formularea ipotezei de concordanţă.

Să remarcăm că această ultimă construcţie conduce la o histogramă cu $k-2$ intervale egale, iar intervalele I_1, I_k sunt de lungimi oarecare.

Dacă am fi ales arbitrar m, M şi am fi construit apoi histograma ca in construcţia precedentă, atunci histograma obţinută ar fi putut avea multe frecvenţe $f_i = 0$ şi eventual frecvenţele f_1 şi/sau f_k foarte mari, fapt care nu ne-ar fi putut conduce la o formulare corectă a ipotezei de concordanţă H. In concluzie, algoritmul cel mai bun, de construcţie al histogramei unei selecţii simulate asupra unei variabile aleatoare X, cu scopul de a valida algoritmul de simulare, este următorul:

Algoritmul HISTOGRAMA

Intrare n, n_1, k; *Consideră tabela* $f[1..k]$ *şi iniţializăm:*

for $j := 1$ **to** k **do** $f[j] := 0$;

Consideră tabela $xi[1..n_1]$; *Consideră tabela* $a[0..k]$;

for $i := 1$ **to** n_1 **do begin** *Generează* X şi *ia* $xi[i] := X$; **end;**

Calculează $a[1] := \min(xi[1], xi[2], ..., xi[n_1])$,

$a[k-1] = \max(xi[1], xi[2], ..., xi[n_1])$, $h := (a[k-1] - a[1])/(k-2)$;

for $i := 1$ **to** n_1 **do** $j := trunc(\frac{xi[i]-a[1]}{h}) + 2,\ f[j] := f[j] + 1;$
Ia $a[0] := a[1],\ a[k] := a[k-1];$
for $i := 1$ **to** $n - n_1$ **do begin**
Generează un X;
if $a[1] < X < a[k-1]$ **then begin**
$j := trunc(\frac{X-a[1]}{h}) + 2,\ f[j] := f[j] + 1;$
end
else if $X \leq a[1]$ **then begin**
$a[0] := \min\{a[0], X\},\ f[1] := f[1] + 1;$
end
else begin
$a[k] := \max\{a[k], X\};\ f[k] := f[k] + 1$
end;
end.

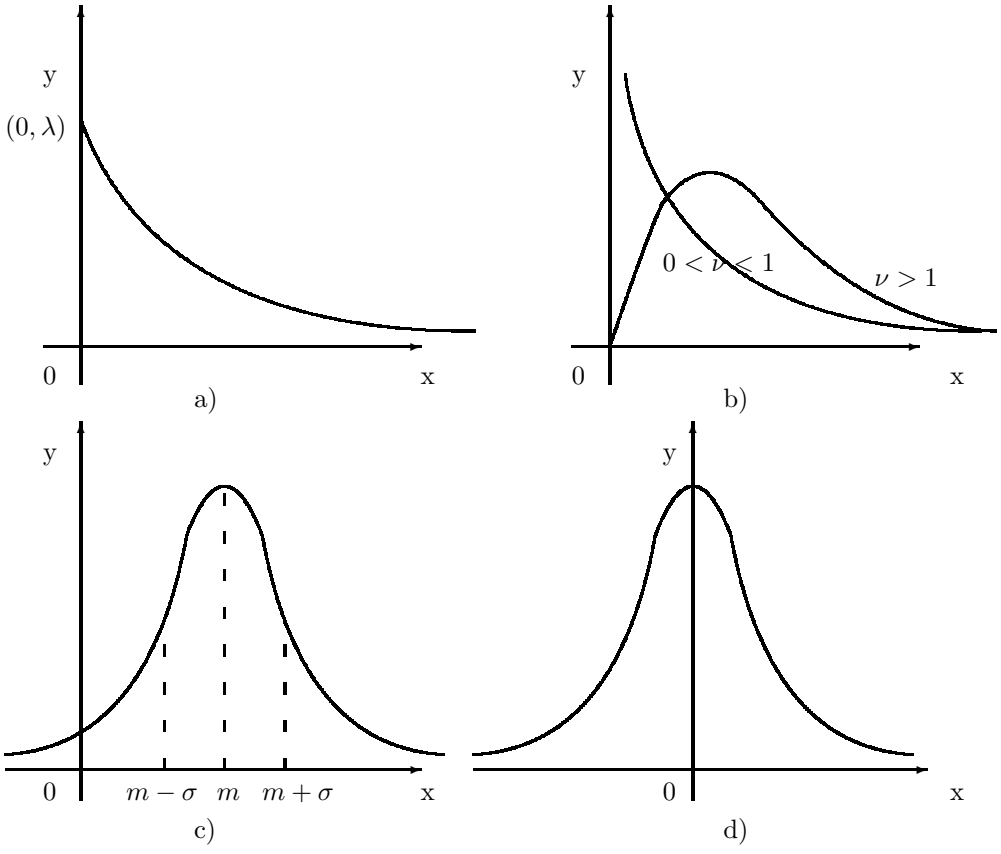

Fig. 3.2. Graficele unor densităţi de repartiţie.
a) Densitatea exponenţială Exp(λ).
b) Densităţi de tipul Gamma$(0,1,\nu)$, Weibull$(0,1,\nu)$.
c) Densitatea normala $N(m,\sigma)$.
d) Densitatea de repartiţie de tip Student.

• **Testul χ^2.** Odată construită histograma putem aplica testul de concordanţă χ^2 pentru verificarea ipotezei $H : X \rightsquigarrow F(x)$.

Pentru aceasta calculăm intâi probabilităţile

$$p_1 = F(a_1),\ p_i = F(a_i) - F(a_{i-1}),\ 2 \leq i \leq k-2,\ p_k = 1 - F(a_{k-1}).$$

Calculăm apoi

$$\chi^2 = \sum_{j=1}^{k} \frac{(f_i - np_i)^2}{np_i} \tag{3.53}$$

care se ştie că are repartiţia hi patrat cu $k-1$ grade de libertate, (χ^2_{k-1} este variabila corespunzătoare). Fiind dat *riscul de genul I,* α, (o probabilitate mică, apropiată de zero) determinăm $\chi^2_{k-1,\alpha}$ (numită α-cuantilă superioară) astfel incât

$$P(\chi^2_{k-1} > \chi^2_{k-1,\alpha}) = \alpha. \tag{3.53'}$$

Ipoteza H se acceptă dacă

$$\chi^2 < \chi^2_{k-1,\alpha} \tag{3.53''}$$

şi se respinge in caz contrar.

• **Un test simplu.** Cel mai simplu mod de testare a unui generator care simulează o variabilă neuniformă X se poate realiza astfel:

- Se determină câteva momente teoretice ale lui X ca de ex. $m = E[X]$ şi $\sigma^2 = Var[X]$;

- Cu ajutorul selecţiei simulate $X_1, X_2, ..., X_n$ se calculează momentele empirice de selecţie

$$\overline{X} = \frac{\sum_{i=1}^n X_i}{n},\ s^2 = \frac{\sum_{i=1}^n X_i^2}{n} - \overline{X}^2;$$

Se consideră că generatorul este bun dacă pentru n suficient de mare ($n > 1000$) valorile lui $\overline{X}$ şi s^2 sunt tot mai apropiate de m şi σ^2, ca o consecinţă a *legii numerelor mari.* Se presupune că momentele teoretice m, σ^2 sunt cunoscute.

Exerciţii

E.3.1. Fie variabila $X \rightsquigarrow f(x)$ unde

$$f(x) = kg(x) \quad g(x) = \begin{cases} x(2-x),\ daca\, x \in [0,2] \\ 0, altfel \end{cases}.$$

Sa se determine constanta k şi să se precizeze o metodă de simulare a lui X.

Soluţie. Impunând condiţia $\int_R f(x)dx = 1$, se obţine $k = 4/3$ şi

$$F(x) = \begin{cases} 0, & daca \quad x < 0 \\ x^2 - \frac{x^3}{3}, & 0 \leq x \leq 2 \\ 1 & daca \quad x > 1 \end{cases}$$

iar simularea se poate face prin metoda inversă calculându-se numeric soluţia $X \in [0, 2]$ a ecuaţiei $F(X) = U$.

E.3.2. Variabila aleatoare Y are densitatea de repartiţie

$$f(y) = \begin{cases} 0 \quad daca \quad y < 0 \\ 2\lambda y e^{-\lambda y^2}, \quad daca \quad y \geq 0. \end{cases}$$

Dacă $X \rightsquigarrow Exp(\lambda)$ să se arate că $Y = \sqrt{X}$ de unde se deduce metoda de simulare a lui Y.

Soluţie Intr-adevăr

$$P(Y < t) = P(\sqrt{X} < t) = P(X < t^2) = \begin{cases} 0, & daca \quad t < 0 \\ 1 - e^{-\lambda t^2} & daca \quad t \geq 0. \end{cases}$$

E.3.3. Fie $X_1 \rightsquigarrow Binom(n_1, p)$ $X_2 \rightsquigarrow Binom(n_2, p)$ X_1, X_2 independente. Arătaţi că $(X_1 + X_2) \rightsquigarrow Binom(n_1 + n_2, p)$.

Soluţie. Se foloseşte funcţia caracteristică

$$\varphi_{X_j}(t) = (q + pe^{it})^{n_j}, j = 1, 2$$

şi $\varphi_{X_1+X_2}(t) = \varphi_{X_1}(t)\varphi_{X_2}(t)$. (Repartiţia binomială este stabilă).

E.3.4. Fie $X \rightsquigarrow F(x)$, $Y \rightsquigarrow G(x)$ independente. Să se determine funcţia de repartiţie a variabilei $Z = max(X, Y)$. Să se precizeze de aici cum se poate simula variabila $Z \rightsquigarrow H(x)$ unde

$$H(x) = \begin{cases} (1 - e^{-\lambda x})(1 - \frac{1}{(1+\theta x)^c}) & daca \quad x \geq 0 \\ 0, \quad daca \quad x < 0. \end{cases}$$

Soluţie. Se observă că $Z \rightsquigarrow H(x) = F(x)G(x)$, $X \rightsquigarrow Exp(\lambda)$, $Y \rightsquigarrow Lomax(\theta, c)$ de unde $Z = max(X, Y)$.

E.3.5. Fie $X \rightsquigarrow Poisson(\lambda)$, $Y \rightsquigarrow Poisson(\mu)$ independente.Să se arate că $(X|X + Y = n) \rightsquigarrow Binom(n, \frac{\lambda}{\lambda+\mu})$.

Soluţie. Avem $(X + Y) \rightsquigarrow Poisson(\lambda + \mu)$ şi deci

$$P(X = r|X + Y = n) = \frac{P(X = r, X + Y = n)}{P(X + Y = n)} =$$

$$= \frac{e^{-\lambda}\lambda^r e^{-\mu}\mu^r}{r!(n-r)!}\frac{n!}{e^{-(\lambda+\mu)}(\lambda+\mu)^n} = C_n^r \frac{\lambda^r \mu^{n-r}}{(\lambda+\mu)^n}, r = 0, 1, ..., n.$$

E.3.6. Fie $X \rightsquigarrow f(x)$ unde

$$f(x) = \begin{cases} 0, & daca \quad x < 0 \\ \frac{1}{\lambda} exp\{-\frac{e^x - 1}{\lambda} + x\} & daca \quad x \geq 0 \end{cases}, \lambda > 0$$

numită *repartiţia valorii extreme.* Să se descrie o metodă pentru simularea lui X.

Soluţie. Se aplică metoda inversă.

E.3.7. Fie $Y \rightsquigarrow Exp(1)$ şi $X = k \exp(\frac{Y}{a})$. Să se determine repartiţia lui X.

*Soluţie.*Se deduce prin calcul direct că

$$F(x) = P(X < x) = \begin{cases} 0 & daca \quad x < 0 \\ 1 - (\frac{x}{k})^{-a} & daca \quad x \geq 0. \end{cases}$$

Se aplică metoda inversă.

E.3.8. Fie Z_1, Z_2 variabile normale $N(0,1)$, independente. Să se arate că

$$U = \frac{2}{\pi} arctg \frac{Z_1}{Z_2} \rightsquigarrow uniforma\, U(-1, 1).$$

Soluţie. Densitatea de repartiţie comună a lui Z_1, Z_2 este

$$f(x_1, x_2) = \frac{1}{2\pi} e^{-\frac{x_1^2 + x_2^2}{2}}.$$

Deci

$$P(U < y) = P(\frac{Z_1}{Z_2} < tg(2\pi y)) = \int\int_{x_1/x_2 < tg(2\pi y)} e^{-\frac{x_1^2+x_2^2}{2}} dx_1 dx_2 = y.$$

E.3.9. Fie X şi Y două variabile exponenţiale de parametru 1, independente. Să se determine repartiţia variabilei $T = X - Y$.

Soluţie. Densitatea comună de repartiţie este

$$g(x, y) = e^{-x-y},\ x > 0,\ y > 0.$$

Facem transformarea

$$t = x - y, \quad v = y, \quad J = \frac{D(x, y)}{D(v, t)} = 1$$

şi calculăm

$$h(t) = \int_0^\infty g(x(t,v), y(v,t))|J|dv.$$

Se obţine in final

$$h(t) = \begin{cases} \frac{1}{2}e^x & daca\, x < 0, \\ \frac{1}{2}e^{-x} & daca\, x \geq 0. \end{cases}$$

Deci densitatea lui T este de tip *Laplace*; variabila T se simulează deci ca diferenţa de două variabile $Exp(1)$ independente.

E.3.10. Variabila aleatoare X are densitatea $f(x) = \frac{3}{4}(1-x^2)$, $x \in [-1,1]$ şi $f(x) = 0$, $x \notin [-1,1]$. Să se simuleze X.

Soluţie. Se poate aplica metoda respingerii folosind infăşurătoarea $h(x) = \frac{1}{2}$, $x \in [-1,1]; h(x) = 0$, $x \notin [-1,1]$ (adică $h(x)$-uniformă pe $[-1,1]$). Se obţine

$$\frac{f(x)}{h(x)} \leq \alpha = \frac{3}{2},\ etc.$$

Se poate folosi şi metida inversă. Avem

$$F(x) = \begin{cases} 0, & x < 0, \\ \frac{3x-x^3+2}{4}, & x \in [-1,1], \\ 1, & x > 1 \end{cases}.$$

Se ia soluţia $x_0 \in [-1,1]$ a ecuaţiei $F(x) = U$. (Ecuaţia poate avea trei soluţii!).

Cap. 4

Simularea vectorilor aleatori

4.1 Generalităţi

Un vector $\mathbf{X} = (X_1, X_2, ..., X_k)'$ (vector coloană) ale cărui componente X_i sunt variabile aleatoare, se numeşte *vector aleator.* Cazul in care componentele sunt independente stochastic este desigur banal de aceea cazul cel mai interesant este când componentele X_i sunt dependente sau *corelate.*

Funcţia de repartiţie a vectorului $\mathbf{X}$ este

$$F(\mathbf{x}) = F(x_1, x_2, ..., x_k) = P(X_1 < x_1, X_2 < x_2, ..., X_k < x_k) \tag{4.1}$$

si ea se mai numeşte funcţia de repartiţie *comună* a lui $\{X_1, X_2, ..., X_k\}$ sau funcţia de repartiţie multidimensională. Desigur,

$$F(-\infty, ..., -\infty) = 0,\ F(+\infty, ..., +\infty) = 1,\ F(..., x_i, ...) \leq F(..., y_i, ...),\ x_i < y_i$$

(adică F este monotonă pe componente). Funcţia $f(x_1, x_2, ..., x_k)$ definită de relaţia

$$f(\mathbf{x}) = f(x_1, x_2, ..., x_k) = \frac{\partial^k F(x_1, x_2, ..., x_k)}{\partial x_1 \partial x_2 ... \partial x_k} \tag{4.2}$$

când derivata parţială există, se numeşte *densitate de repartiţie multidimensională* a lui $\mathbf{X}$.

Funcţia de repartiţie $P(X_i < x_i) = F(+\infty, ..., +\infty, x_i, +\infty, ..., +\infty) = F_i(x_i)$ se numeşte *funcţie de repartiţie marginală a lui* X_i, iar $f_i(x_i) = F_i'(x_i)$ (când derivata există) se numeşte *densitate de repartiţie marginală* a lui X_i. Se pot defini funcţii de repartiţie marginale pentru $2, 3, ..., k-1$ componente, etc. De exemplu funcţia de repartiţie marginală $F_{ij}(x_i, x_j) = P(X_i <$

$x_i, X_j < x_j) = F(+\infty, ..., x_i, ..., x_j, ... + \infty)$, $1 \leq i, j \leq k$ corespunde componentelor X_i, X_j ale vectorului $\mathbf{X}$. Au loc relaţiile

$$F(\mathbf{x}) = F(x_1, ..., x_k) = \int_{-\infty}^{x_1} ... \int_{-\infty}^{x_k} f(u_1, ..., u_k) du_1 ... du_k = \int_{-\infty}^{\mathbf{x}} f(\mathbf{u}) d\mathbf{u}. \tag{4.3}$$

Dacă variabilele $X_i, 1 \leq i \leq k$ sunt independente atunci

$$F(x_1, x_2, ..., x_k) = F_1(x_1) F_2(x_2) ... F_k(x_k) \tag{4.4}$$

sau analog pentru densităţi

$$f(x_1, x_2 ..., x_k) = f_1(x_1) f(x_2) ... f_k(x_k). \tag{4.4'}$$

Când există integralele

$$m_i = E(X_i) = \int_{-\infty}^{+\infty} x_i f_i(x_i) dx_i,$$

$$m_{ij} = E(X_i X_j) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} x_i x_j f_{ij}(x_i x_j) dx_i dx_j$$

acestea se mumesc momente *de ordinul I*, respectiv *de ordimul II*. Expresiile

$$\sigma_{ij} = E[(X_i - m_i)(X_j - m_j)] = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} (x_i - m_i)(x_j - m_j) f_{ij}(x_i, x_j) dx_i dx_j$$

când integralele există, se numesc momente centrate; in acest caz, σ_{ij} se numeşte *covarianţa* lui X_i cu X_j şi se notează $\sigma_{ij} = Cov(X_i, X_j)$. Se observă că $\sigma_{ii} = Cov(X_i, X_i) = Var(X_i) = \sigma_i^2$. Este uşor de arătat că dacă X_i este independent de X_j, $i \neq j$, atunci

$$m_{ij} = E(X_i X_j) = m_i m_j = E(X_i) E(X_j), \quad \sigma_{ij} = Cov(X_i, X_j) = 0.$$

Este satisfăcută *inegalitatea lui Schwarz,* adică

$$|Cov(X_i, x_j)| \leq \sqrt{Var(X_i) Var(X_j)} \quad sau \quad |\sigma_{ij}| \leq \sigma_i \sigma_j.$$

Se numeşte *coeficient de corelaţie al variabilelor* X_i şi X_j mărimea

$$\rho_{ij} = Corr(X_i, X_j) = \frac{Cov(X_i, X_j)}{\sqrt{Var(X_i) Var(X_j)}}. \tag{4.5}$$

Se observă că $\rho \in [-1, 1]$ şi el are următoarea interpretare: $\rho_{ij} \approx 0 \Rightarrow X_i \, depinde \, foarte \, putin \, de \, X_j$; dacă $\rho_{ij} < 0$, atunci X_i şi X_j sunt *invers*

corelate iar dacă $\rho_{ij} > 0$, atunci X_i şi X_j sunt *direct* corelate; dacă $|\rho_{ij}| \approx 1$ atunci X_i şi X_j sunt *puternic corelate.*

Pentru un vector aleator mediile m_i definesc un vector $\mathbf{m} = (m_1, m_2, ..., m_k)'$ numit *vectorul valoare medie* notat $\mathcal{E}(\mathbf{X})$ iar covarianţele σ_{ij} definesc o matrice $\Sigma = ||\sigma_{ij}, ||$ (care este pozitiv definită, $\Sigma \succ 0$) şi se notează $\Sigma = \mathcal{C}ov(\mathbf{X}, \mathbf{X'})$.

Vom defini acum noţiunea de repartiţie condiţionată. Presupunem că $\mathbf{X}$ are densitatea de repartiţie $f(\mathbf{x})$ şi să considerăm descopmunerea lui $\mathbf{X}$ in doi subvectori $\mathbf{X} = (\mathbf{X}^{(1)}, \mathbf{X}^{(2)})'$ unde $\mathbf{X}^{(1)}$ este un vector $r-$dimensional iar $\mathbf{X}^{(2)}$ este un vector $q-$dimensional, $r < k$, $r + q = k$. Definim *densitatea de repartiţie condiţionată* $f(\mathbf{x}^{(1)}|\mathbf{x}^{(2)})$ a lui $\mathbf{X}^{(1)}$ când $\mathbf{X}^{(2)} = \mathbf{x}^{(2)}$ astfel

$$f(\mathbf{x}^{(1)}|\mathbf{x}^{(2)}) = \frac{f(\mathbf{x}^{(1)}, \mathbf{x}^{(2)})}{\int_{-\infty}^{+\infty} f(\mathbf{y}^{(1)}, \mathbf{x}^{(2)}) d\mathbf{y}^{(1)}}, \text{ unde } f(\mathbf{x}) = f(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}). \quad (4.6)$$

Cu această densitate se definesc, când există, $E[\mathbf{X}^{(1)}|\mathbf{X}^{(2)} = \mathbf{x}^{(2)}]$, $\mathcal{C}ov(\mathbf{X}^{(1)}, \mathbf{X}^{(1)'}|\mathbf{X}^{(2)} = \mathbf{x}^{(2)})$ adică *media condiţionată,* respectiv *matricea de covarianţă condiţionată.* Uneori este necesar să simulăm $(\mathbf{X}^{(1)}|\mathbf{X}^{(2)} = \mathbf{x}^{(2)})$, când se cunoaşte repartiţia condiţionată.

Metoda inversă este valabilă şi in cazul simulării vectorilor aleatori. Fie $F(\mathbf{x}) = F(x_1, x_2, ..., x_k) = P(X_1 < x_1, ..., X_k < x_k)$, funcţia de repartiţie a lui $\mathbf{X}$ şi fie $F_1(x_1) = P(X_1 < x_1)$ funcţia de repartiţie marginală a lui X_1, şi $F_{1...i}(x_1, ..., x_i) = P(X_1 < x_1, ..., X_i < x_i)$ funcţia de repartiţie marginală a lui $(X_1, ..., X_i)'$. Fie de asemenea funcţiile de repartiţie condiţionate
$F_2(x_1, x_2) = P(X_2 < x_2|X_1 = x_1) = F(x_2|x_1)$,
$F_3(x_1, x_2, x_3) = P(X_3 < x_3|X_1 = x_1, X_2 = x_2) = F(x_3|x_1, x_2)$,
$F_i(x_1, x_2, ..., x_i) = P(X_i < x_i|X_1 = x_1, ..., X_{i-1} = x_{i-1}) = F(x_i|x_1, ..., x_{i-1})$,
$2 \leq i \leq k$. Să notăm cu F_i^{-1} inversa funcţiei $F_i(x_1, x_2, ..., x_i)$ in raport cu x_i. Atunci are loc următoatea teoremă (metoda inversă pentru vectori).

Teorema 4. 1 *Dacă $U_i, 1 \leq i \leq k$ sunt variabile aleatoare uniforme $0-1$ independente atunvi vectorul aleator $\mathbf{Y} = (Y_1, Y_2, ..., Y_k)'$ ale cărui componente sunt*

$$Y_1 = F_1^{-1}(U_1),\ Y_2 = F_2^{-1}(Y_1, U_2),\ Y_i = F_i^{-1}(Y_i, ..., Y_{i-1}, U_i),\ 2 \leq i \leq k \quad (4.6')$$

are funcţia de repartiţie $F(\mathbf{x})$.

Demonstraţie. Avem

$$P(Y_1 < x_1) = P(F_1^{-1}(U_1) < x_1) = P(U_1 < F(x_1)) = F_1(x_1),$$

$$P(Y_1 < x_1, Y_2 < x_2) = P(Y_2 < x_2 | Y_1 = x_1) P(Y_1 < x_1) =$$

$$= P(F_2^{-1}(Y_1, U_2) < x_2 | Y_1 = x_1) F_1(x_1) = P(U_2 < F_2(x_1, x_2)) F_1(x_1) =$$

$$= F_2(x_1, x_2) F_1(x_1) = F_{12}(x_1, x_2).$$

Repetând procedeul precedent obţinem pentru $(Y_1, ..., Y_i)'$

$$P(Y_1 < x_1, ..., Y_{i-1} < x_{i-1}, Y_i < x_i) =$$

$$= P(Y_1 < x_1, ..., Y_{i-1} < x_{i-1}, F_i^{-1}(Y_1, ..., Y_{i-1}, U_i) < x_i) = F_{1...i}(x_1, ..., x_i).$$

Dar

$$F_{1...k}(x_1, ..., x_k) = F(x_1, ..., x_k)$$

ceeace trebuia demonstrat. De aici se deduce următorul algoritm pentru simularea vectorului aleator $\mathbf{X}$ când se cunosc inversele F_i^{-1} ale funcţiilor de repartiţie condiţionate $F_i(x_1, ..., x_{i-1}, x_i)$ in raport cu x_i.

Algoritmul AIMULT pentru metoda inversă multidimensională.
Generează $U := random$; *Calculează* $Y_1 = F_1^{-1}(U)$;
for $i := 2$ **to** k **do begin**
Generează $U := random$;
Calculează $Y_i := F_i^{-1}(Y_1, ..., Y_{i-1}, U)$;
end.

Vectorul $\mathbf{Y} = (Y_1, Y_2, ..., Y_k)'$ are funcţia de repartiţie $F(\mathbf{x})$ conform teoremei 1.

Exemplul 4.1. Simularea repartiţiei Gumbel bidimensionale. Presupunem că vectorul aleator $(X, Y)'$ are funcţia de repartiţie

$$F(x, y) = \begin{cases} 1 - e^{-x} - e^{-y} + e^{-(x+y+\theta xy)}, \; daca \; x > 0, y > 0, \\ 0, \; altfel \end{cases}, 0 < \theta < 1.$$

Se observă că

$$F_1(x) = \begin{cases} 1 - e^{-x}, \; daca \; x > 0, \\ 0, \; altfel \end{cases}$$

adică repartiţia marginală a lui X este $Exp(1)$.

Prin calcule simple (utilizând (4.6)) se deduce că

$$f_2(y|x) = \begin{cases} e^{-y(1+\theta x)} \{(1 + \theta x)(1 + \theta y) - \theta\}, \; daca \; y > 0, \\ 0, \; altfel \end{cases}$$

şi deci funcţia de repartiţie a lui Y condiţionată de $X = x$ este

$$F_2(y|x) = \begin{cases} [1 - e^{-y(1+\theta x)}][1 + \theta(1 + x)], \, daca \; y > 0 \\ 0, \, altfel \end{cases}$$

Putem deci aplica teorema 1 pentru simularea lui $(X, Y)'$ astfel

Algoritmul GUMBEL
Intrare θ; (Acesta este un pas pregătitor);
Generează $X \rightsquigarrow Exp(1)$;
Generează $U := random$; *Generează* Y *rezolvând ecuaţia*

$$[1 - e^{-Y(1+\theta X)}][1 + \theta(1 + X)] = U; U \in (0, \infty).$$

Teorema infăşurătoarei se poate aplica şi in cazul vectorilor aleatori. (Vezi exerciţiul **E4.5**).

4.2 Simularea vectorilor uniformi.

Vectorul aleator $k-$dimensional $\mathbf{V} = (V_1, V_2, ..., V_k)'$ are repartiţie uniformă pe domeniul mărginit $D \subset \mathbf{R}^k$ dacă densitatea sa de repartiţie este

$$f(\mathbf{v}) = \begin{cases} k, \, daca \, \mathbf{v} \in D, \\ 0, \, daca \, \mathbf{v} \notin D \end{cases}, \quad k = \frac{1}{mesD}, \, mesD = \int_D d\mathbf{v}. \tag{4.7}$$

Rezultă că $\mathbf{V}$ are repartiţie uniformă pe intervalul $I = [a_1, b_1] \times [a_2, b_2] \times ... \times [a_k, b_k]$ $-\infty < a_i < b_i < +\infty, 1 \leq i \leq k$ dacă

$$f(\mathbf{v}) = \begin{cases} \frac{1}{\prod_{i=1}^{k}(b_i - a_i)}, \, daca \, \mathbf{v} \in I \\ 0, \, daca \, \mathbf{v} \notin I. \end{cases} \tag{4.7'}$$

Din (4.7') rezultă că dacă $\mathbf{V}$ este uniform pe I atunci V_i sunt variabile uniforme pe $[a_i, b_i]$ şi independente deoarece

$$f(v_1, v_2, ..., v_k) = \prod_{i=1}^{k} f_i(v_i), \quad f_i(v_i) = \begin{cases} \frac{1}{b_i - a_i}, \, daca \; v_i \in [a_i, b_i] \\ 0, daca \; v_i \notin [a_i, b_i]. \end{cases} \tag{4.7''}$$

De aici se deduce următorul algoritm simplu pentru simularea vectorului $\mathbf{V} \rightsquigarrow uniform \, pe \, I$.

Algoritmul VUNIFINT de simulare a vectorului uniform pe interval

Intrare $a_i, b_i, 1 \leq i \leq k$;
for $i := 1$ **to** k **do begin**
Generează $U := random$; *Ia* $V_i := a_i + (b_i - a_i)U$;
end.

Pentru a simula un vector aleator $\mathbf{W}$ uniform pe $D \subset \mathbf{R}^k$ (D fiind un domeniu oarecare), nu mai este adevărată (4.7"). Dar să observăm că dacă $D \subseteq I$ şi $\mathbf{V}$ e uniform pe I atunci restricţia lui $\mathbf{V}$ la D (vectorul notat $\mathbf{W}$), este un vector uniform pe D.

De aici rezultă următorul algoritm de *respingere* pentru simularea lui $\mathbf{W}$.

Algoritmul VECTUNIFD de simulare a vectorului uniform pe domeniu oarecare:

Construieşte un interval minimal $I = [a_1, b_1] \times ... \times [a_k, b_k]$ *astfel incât* $D \subseteq I$; (Acest interval există deoarece D este mărginit).
repeat
Generează V *uniform pe* I;
until $\mathbf{V} \in D$;
Ia $\mathbf{W} := \mathbf{V}$.

Probabilitatea de acceptare a acestui algoritm este

$$p_a = \frac{mesD}{mesI} = \frac{mesD}{\prod_{i=1}^{k}[b_i - a_i]}$$

de unde se deduce că pentru a obţine un interval I cât mai bun, acesta trebuie să fie *minimal* astfel incât $D \subseteq I$.

4.3 Simularea vectorilor normali

Vectorul aleator $k-$dimensional $\mathbf{X}$ are repartiţia normală $N(\mu, \Sigma)$ dacă densitatea sa este

$$f(\mathbf{x}; \mu, \Sigma) = \frac{1}{(2\pi)^{k/2}[det(\Sigma)]^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\mu)'\Sigma^{-1}(\mathbf{x}-\mu)}, \ \mathbf{x} \in \mathbf{R}^k. \tag{4.8}$$

Se arată că

$$\mathcal{E}(\mathbf{X}) = \mu, \quad \mathcal{C}ov(\mathbf{X}, \mathbf{X}') = \Sigma. \tag{4.8'}$$

Formulele precedente se deduc uşor cu ajutorul funcţiei caracteristice. In cazul variabilei normale $X \rightsquigarrow N(m, \sigma)$ conform teoremei 3.7 funcţia caracteristică este

$$\varphi(t) = e^{itm - \frac{t^2\sigma^2}{2}}.$$

Prin analogie, in cazul multidimensional funcţia caracteristică este

$$\varphi(\mathbf{t}) = E[e^{i\mathbf{t}'\mathbf{X}}] = \int_{R^k} e^{i\sum_{j=1}^{k} t_j x_j} f(\mathbf{x};\mu,\Sigma)d\mathbf{x}$$

care după efectuarea unor calcule devine

$$\varphi(\mathbf{t}) = e^{i\mathbf{t}'\mu - \frac{\mathbf{t}'\Sigma\mathbf{t}}{2}}.$$

Făcând deci analogie cu funcţia caracteristică a variabilei normale $N(m,\sigma)$ rezultă că μ şi Σ au semnificaţia din formulele (4.8'). Să notăm cu $\mathbf{Z}$ vectorul normal $N(\mathbf{0},I)$ unde I este matricea unitate $k-$dimensională iar $\mathbf{0}$ este vectorul nul din $\mathbf{R}^k$. Se constată cu uşurinţă că densitatea $f(\mathbf{z})$ a lui $\mathbf{Z}$ satisface proprietatea

$$f(\mathbf{z}) = \prod_{i=1}^{k} f_i(z_i), \quad f_i(z_i) = \frac{1}{\sqrt{2\pi}} e^{-\frac{z_i^2}{2}} \tag{4.9}$$

adică componentele Z_i ale lui $\mathbf{Z}$ sunt independente şi normale $N(0,1)$. Deci, simularea lui $\mathbf{Z} \rightsquigarrow N(\mathbf{0},I)$ se realizează simplu astfel

Algoritmul VECNORMSTD de generare a vectorului normal standard.

for $i := 1$ **to** k **do begin**
Generează $T \rightsquigarrow N(0,1)$; *Ia* $Z_i := T$;
end.

Pentru a simula $\mathbf{X} \rightsquigarrow N(\mu,\Sigma)$ vom folosi mai intâi teorema

Teorema 4. 2 *Dacă* $\mathbf{Y} \rightsquigarrow N(\mathbf{0},\Sigma)$ *şi* $\mathbf{C}$ *este o matrice* $k \times k$ *iar* μ *este un vector* $k \times 1$ *atunci*

$$\mathbf{X} = \mu + \mathbf{C}\mathbf{Y} \rightsquigarrow N(\mu, \mathbf{C}\Sigma\mathbf{C}').$$

Demonstraţie. Deoarece $\mathbf{X}$ este o combinaţie liniară de variabile normale (componentele lui $\mathbf{Y}$), el are tot o repartiţie normală şi deci avem

$$\mathcal{E}(\mathbf{X}) = \mu + \mathbf{C}\mathcal{E}(\mathbf{Y}) = \mu + \mathbf{0} = \mu,$$

$$\mathcal{C}ov(\mathbf{X},\mathbf{X}') = \mathcal{C}ov[(\mathbf{C}\mathbf{Y}),(\mathbf{C}\mathbf{Y})'] = \mathbf{C}\Sigma\mathbf{C}' \tag{4.10}$$

şi teorema este demonstrată.

Fiind dată matricea $\Sigma \succ 0$ (pozitiv definită) se ştie că există o matrice $C = ||c_{ij}||, C \succ 0$, *inferior triunghiulară* (adică $c_{ij} = 0,\ i < j$), astfel incât $CC' = \Sigma$. Matricea C este *matricea Choleski.* De aici deducem teorema

Teorema 4. 3 *Dacă* $\mathbf{Z} \rightsquigarrow N(\mathbf{0}, \mathbf{I})$, μ *este un vector* $k \times 1$ *şi* C *este matricea Choleski a lui* Σ, *atunci* $\mathbf{X} = \mu + C\mathbf{Z} \rightsquigarrow N(\mu, \Sigma)$.

Demonstraţie. Din teorema 4.1 avem $\mathcal{E}(\mathbf{X}) = \mu$ şi $\mathcal{C}ov(\mathbf{X}, \mathbf{X}') = C\mathbf{I}C' = CC' = \Sigma$ şi teorema este demonstrată.

Matricea C se determină cu ajutorul matricii $\Sigma = ||\sigma_{ij}||$ cu formulele lui Choleski

$$c_{i1} = \frac{\sigma_{i1}}{\sigma_{11}}, \quad 1 \leq i \leq k$$

$$c_{ii} = \sqrt{\sigma_{ii} - \sum_{r=1}^{i-1} c_{ir}^2}, \quad 1 < i \leq k$$

$$c_{ij} = \frac{\sigma_{ij} - \sum_{r=1}^{j-1} c_{ir}c_{jr}}{c_{jj}}, \quad 1 < j < i \leq k \tag{4.11}$$

$$c_{ij} = 0, \quad 1 < i < j < k.$$

Simularea vectorului $\mathbf{X} \rightsquigarrow N(\mu, \Sigma)$, după ce se calculează intr-un pas pregătitor matricea C a lui Choleski se realizează cu următorul algoritm

Algoritmul NORMMULT de simulare a vectorului normal
Intrare μ, C;
Generează $\mathbf{Z} \rightsquigarrow N(\mathbf{0}, \mathbf{I})$ *cu algoritmul* **VECNORMSTD;**
Calculează $\mathbf{X} = \mu + C\mathbf{Z}$.

Uneori este necesar să simulăm un subvector al unui vector normal când se dă celălalt subvector. Mai precis dacă $\mathbf{X} = (\mathbf{X}^{(1)}, \mathbf{X}^{(2)})' \rightsquigarrow N(\mu, \Sigma)$, cu $\mathbf{X}^{(1)}$ vector $r-$dimensional şi $\mathbf{X}^{(2)}$ vector $q-$dimensional, $r + q = k$, se cere să se simuleze $\mathbf{X}^{(1)}$ condiţionat de $\mathbf{X}^{(2)} = \mathbf{x}^{(2)}$.

Să considerăm scrierea *celulară* a lui μ şi Σ, adică

$$\mu = \begin{pmatrix} \mu^{(1)} \\ \mu^{(2)} \end{pmatrix}, \quad \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}. \tag{4.12}$$

Dacă μ şi Σ sunt cunoscute atunci este adevărată următoarea teoremă

Teorema 4. 4 *Repartiţia condiţionată a lui* $\mathbf{X}^{(1)}$ *când se dă* $\mathbf{X}^{(2)} = \mathbf{x}^{(2)}$ *este normală* $N(\mu_*^{(1)}, \Sigma_{11}^*)$ *unde*

$$\mu_*^{(1)} = \mu^{(1)} + \Sigma_{12}\Sigma_{22}^{-1}(\mathbf{x}^{(2)} - \mu^{(2)}), \quad \Sigma_{11}^* = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}. \tag{4.13}$$

Nu prezentăm demonstraţia acestei teoreme.

De aici se deduce algoritmul de simulare a lui $\mathbf{X}^{(1)}$ condiţionat de $\mathbf{X}^{(2)} = \mathbf{x}^{(2)}$ şi anume

Algoritmul NORMCOND de simulare condiţionată.
Intrare $\mathbf{x}^{(2)}$, $\mu^{(i)}$, Σ_{ij}, $i, j = 1, 2$;
Calculează $\mu_*^{(1)}$, Σ_{11}^*;
Generează $\mathbf{Z}^{(1)} \rightsquigarrow N(\mathbf{0}, \mathbf{I}^{\mathbf{r}\times\mathbf{r}})$;
Calculează C^* astfel incât $C^*C^{*'} = \Sigma_{11}^*$;
Calculează $\mathbf{X}_*^{(1)} = \mu_*^{(1)} + C^*\mathbf{Z}^{(1)}$.

Rezultatul este $\mathbf{X}_*^{(1)} \rightsquigarrow N(\mu_*^{(1)}, \Sigma_{11}^*)$.

• **O metodă specială.** Presupunem că matricea de covarianţă Σ are elementele de forma

$$\sigma_{ij} = \begin{cases} \sigma_i^2 \quad daca \quad i = j,\ \sigma_i > 0 \\ \sigma_i\sigma_j\lambda_i\lambda_j \quad daca \quad i \neq j,\ \lambda_i \in [-1, 1]. \end{cases} \tag{4.13'}$$

Se arată simplu că dacă $\mathbf{Z} \rightsquigarrow N(\mathbf{0}, \mathbf{I})$ este un vector normal $(k+1)-$dimensiolal, $\mathbf{Z} = (Z_0, Z_1, Z_2, ..., Z_k)'$, atunci

$$X_i = \mu_i + \sigma_i(\sqrt{1-\lambda_i^2}Z_i + \lambda_i Z_0),\ 1 \leq i \leq k \tag{4.13''}$$

este un vector normal $N(\mu, \Sigma)$, $\mu = (\mu_1, \mu_2, ..., \mu_k)'$.

Intr-adevăr, deoarece $Z_i \rightsquigarrow N(0, 1)$ rezultă că X_i sunt normale şi deci $\mathbf{X} = (X_1, X_2, ..., X_k)'$ are repartiţie normală multidimensională. In plus avem

$$E(X_i) = \mu_i,\ Var(X_i) = \sigma_i^2(1 - \lambda^2 + \lambda^2) = \sigma_i^2,\ \sigma_{ij} = Cov(X_i, X_j) =$$

$$E[(X_i - \mu_i)(X_j - \mu_j)] = E[\sigma_i(\sqrt{1-\lambda_i^2}Z_i + \lambda_i Z_0)\sigma_j(\sqrt{1-\lambda_j^2}Z_j + \lambda_j Z_0)] =$$

$$= \sigma_i\sigma_j E[\sqrt{(1-\lambda_i^2)(1-\lambda_j^2)}Z_iZ_j + \lambda_j\sqrt{1-\lambda_i^2}Z_0Z_j +$$

$$+\lambda_i\sqrt{1-\lambda_j^2}Z_0Z_i + \lambda_i\lambda_j Z_0^2] = \sigma_i\sigma_j\lambda_i\lambda_j, \quad i \neq j$$

adică σ_{ij} este dat de (4.13'). Simularea vectorului normal in acest caz special se face determinând componentele sale cu formula (4.13").

4.4 Simularea repartiţiei Cauchy multidimensionale

In capitolul precedent s-a introdus repartiţia Cauchy standard a cărei densitate este

$$f(x) = \frac{1}{\pi(1+x^2)} \tag{4.14}$$

şi care se poate simula cu ajutorul metodei inverse (vezi tabelul 3.1). Se poate arătă că dacă Z_1, Z_2 sunt variabile normale $N(0,1)$ independente, atunci variabila

$$X = \frac{Z_1}{Z_2} \tag{4.15}$$

are reparticţia Cauchy standard.

Din această proprietate rezultă deci o metodă simplă de simulare a variabilei X-Cauchy standard unidimensionale ca raport de variabile normale $N(0,1)$ independente.

O variabilă Y are repartiţia $Cauchy(c,\sigma)$ dacă densitatea sa este

$$g(x) = \frac{1}{\sigma}.\frac{1}{1+\frac{(x-c)^2}{\sigma^2}},\ c \in R,\ \sigma > 0. \tag{4.16}$$

Simularea lui Y se va face cu formula $Y = c + \sigma X$.

In mod asemănător se poate introduce repartiţia Cauchy standard multidimensională. Vectorul aleator $\mathbf{X}$ are repartiţia *Cauchy standard multidimensională* dacă densitatea sa de repartiţie este de forma

$$f(\mathbf{x}) = \frac{\Gamma\left(\frac{k+1}{2}\right)}{\pi^{\frac{k+1}{2}}} \frac{1}{(1+x'x)^{\frac{k+1}{2}}}, \quad \mathbf{x} \in R^k. \tag{4.15'}$$

Ca şi in cazul unidimensional se arată că dacă $\mathbf{Z}$ este un vector $k-$dimensional normal $N(\mathbf{0},\mathbf{I})$ şi η este o variabilă normală $N(0,1)$ independentă de $\mathbf{Z}$, atunci $\mathbf{X}$ se poate simula cu formula

$$\mathbf{X} = \frac{\mathbf{Z}}{\eta}. \tag{4.15''}$$

Se poate considera şi repartiţia multidimensională $Cauchy(\mathbf{c},\Sigma)$ care are densitatea de forma

$$g(\mathbf{x}) = \frac{\Gamma(\frac{k+1}{2})}{\pi^{\frac{k+1}{2}}(det\Sigma)^{\frac{1}{2}}} \frac{1}{\left[1+(\mathbf{x}-\mathbf{c})'\Sigma^{-1}(\mathbf{x}-\mathbf{c})\right]^{\frac{k+1}{2}}}. \tag{4.16'}$$

Relaţia dintre vectorul Cauchy standard $\mathbf{X}$ şi vectorul $\mathbf{Y} \rightsquigarrow Cauchy(\mathbf{c}, \Sigma)$ este asemănătoare relaţiei dintre vectorul $\mathbf{Z}$ normal $N(\mathbf{0}, \mathbf{I})$ şi vectorul $\mathbf{Y}$ normal $N(\mathbf{c}, \Sigma)$, adică dacă $\mathbf{X}$ are repartiţie Cauchy standard şi $\mathbf{Y} \rightsquigarrow Cauchy(\mathbf{c}, \Sigma)$ şi considerăm matricea $\mathbf{S}$ inferior triunghiulară astfel ca $\Sigma = \mathbf{SS}'$, atunci $\mathbf{Y}$ se poate genera cu formula

$$\mathbf{Y} = \mathbf{SX} + \mathbf{c}. \tag{4.16''}$$

Combinând (4.15') cu (4.16") rezultă că $\mathbf{Y} \rightsquigarrow Cauchy(\mathbf{c}, \Sigma)$ se simulează cu formula

$$\mathbf{Y} = \frac{\mathbf{Z}^*}{\eta} \tag{4.17}$$

unde $\mathbf{Z} \rightsquigarrow N(\mathbf{0}, \Sigma)$, independent de $\eta \rightsquigarrow N(0, 1)$.

4.5 Simularea repartiţiei multinomiale

Această repartiţie este generalizarea multidimensională a repartiţiei binomiale. Vectorul aleator cu componente intregi nenegative $\mathbf{X} = (X_1, X_2, ..., X_k)'$, $X_1 + X_2 + ... + X_k = n$, are repartiţia *multinomială* $Multinom(n, p_1, p_2, ..., p_k)$ dacă

$$P(X_1 = n_1, X_2 = n_2, ..., X_k = n_k) = \frac{n!}{n_1! n_2! ... n_k!} p_1^{n_1} p_1^{n_2} ... p_k^{n_k},$$

$$n_1 + ... + n_k = n, \quad p_i > 0, 1 \leq i \leq k, \quad p_1 + p_2 + ... p_k = 1. \tag{4.18}$$

Se arată că

$$m_i = E[X_i] = np_i,\ \sigma_i^2 = Var[X_i] = np_i(1 - p_i) = Cov(X_i, X_i),$$

$$\sigma_{ij} = Cov(X_i, X_j) = E[X_i X_j] - E[X_i]E[X_j] = \tag{4.18'}$$

$$= n(n-1)p_i p_j - n^2 p_i p_j = -np_i p_j,\ i \neq j.$$

Vectorul $\mathbf{X}$ are o interpretare asemănătoare variabilei binomiale in cazul unui experiment cu urnă după cum urmează: să presupunem că intr-o urnă se găsesc N_i bile de culoarea i, $1 \leq i \leq k, N = N_1 + N_2 + ... N_k$. Se presupune că se extrag n bile din urnă *cu intoarcere*, din care X_i sunt bile de culoarea i, $n = X_1 + X_2 + ... X_k$. Atunci vectorul aleator $\mathbf{X} = (X_1, X_2, ..., X_k)'$ are repartiţia $Multinom(n, p_1, ..., p_k)$, $p_i = N_i/N$,
$\sum_i p_i = 1$.

Această interpretare ne conduce la următorul algoritm pentru simularea lui $\mathbf{X}$

Algoritmul MULTINOM de simulare a repartiţiei multinomiale

Intrare $n, k, p_1, p_2, ..., p_k$ *şi calculează* $F[\alpha] = \sum_{i=1}^{\alpha} p_i$, $1 \leq i \leq k$; (Acesta este un pas pregătitor);

Iniţializează $X[1] = 0, ..., X[k] = 0$;

for $i := 1$ **to** n **do begin**

Simulează $U := random$; *Ia* $i := 1$;

while $U \geq F[i]$ **do** $i := i + 1$;

Insumează $X[i] := X[i] + 1$;

end.

Testarea algoritmului se face in mod simplu astfel

- Se generează selecţia de volum T, $\mathbf{X}^{(\alpha)} = (X_1^{(\alpha)}, ..., x_k^{(\alpha)})', 1 \leq \alpha \leq T$;
- Se calculează mediile aritmetice şi dispersiile empirice

$$\overline{X}_i = \frac{1}{T}\sum_{\alpha=1}^{T} X_i^{(\alpha)}, \quad s_i^2 = \frac{1}{T}\sum_{i=1}^{T}[X_i^{(\alpha)}]^2 - \overline{X_i}^2.$$

Pentru validarea algoritmului, trebuie, ca pentru un T mare să avem $m_i \approx \overline{X}_i$, $\sigma_i^2 \approx s_i^2$, unde m_i şi σ_i^2 sunt date de (4.18').

4.6 Simularea repartiţiei Dirichlet.

Un vector aleator $\mathbf{X}$ are repartiţia $Dirichlet(\nu_1, \nu_2, ..., \nu_{k+1})$ dacă densitatea sa de repartiţie este

$$f(x_1, x_2, ..., x_k) = \begin{cases} \frac{\Gamma(\nu_1+...+\nu_{k+1})}{\Gamma(\nu_1)...\Gamma(\nu_{k+1})} x_1^{\nu_1}...x_k^{\nu_k}(1 - x_1 - ... - x_k)^{\nu_{k+1}}, \ \mathbf{x} \in S_k \\ 0, \quad daca \quad x \notin S_k \end{cases}$$

$$\mathbf{x} = (x_1, ..., x_k)', \ S_k = \{(x_1, ..., x_k)' \in R^k, x_i \geq 0, \ 1 \leq i \leq k, \ \sum_i x_i \leq 1\}. \tag{4.19}$$

Se arată că dacă $Y_i \rightsquigarrow Gamma(0, 1, \nu_i)$, $1 \leq i \leq k+1$ sunt variabile gama independente, atunci vectorul $\mathbf{X}$ ale cărui componente sunt de forma

$$X_i = \frac{Y_i}{Y_1 + Y_2 + ... + Y_{k+1}}, \quad 1 \leq i \leq k, \tag{4.20}$$

are o repartiţie $Dirichlet(\nu_1, \nu_2, ..., \nu_{k+1})$. Se observă deci că repartiţia Dirichlet este o extensie la cazul multidimensional a repartiţiei Beta. Formulele (4.16) simulează componentele unui vector Dirichlet, presupunându-se desigur că parametri ν_i, $1 \leq i \leq k+1$ sunt cunoscuţi.

Exerciţii

E4.1 Să se prezinte o metodă de simulare a unui vector $\mathbf{V} = (V_1, V_2)'$, uniform pe cercul $C(O, r)$.

Soluţie. Se simulează un vector $\mathbf{W} = (W_1, W_2)'$ uniform pe $[-r, r] \times [-r, r]$ şi se pune condiţia ca el să aparţină cercului. Algoritmul este:

repeat
Generează: $U_1 := random; U_2 := random;$
$W_1 := -r + 2rU_1; W_2 := -r + 2rU_2;$
until $W_1^2 + W_2^2 \leq r^2;$
Ia $V_1 := W_1;\ V_2 := W_2.$

E4.2 Să se prezinte o metodă de simulare a unui vector $\mathbf{W} = (W_1, W_2)'$ uniform pe elipsa de semiaxe $a, b > 0$.

Indicaţie. Se simulează un vector $\mathbf{V} = (V_1, V_2)'$ uniform pe intervalul $[-a, a] \times [-b, b]$ şi se pune condiţia ca $V_1^2/(a^2) + V_2^2/(b^2) \leq 1$.

E4.3 Se consideră vectorul $\mathbf{X} = (X_1, X_2, ..., X_k)'$ unde $X_i \rightsquigarrow Exp(\lambda_i \eta)$ sunt independente iar $\eta \rightsquigarrow Gamma(0, b, a)$. Să se determine densitatea de repartiţie a lui $\mathbf{X}$ şi să se indice un algoritm pentru simularea acestuia.

*Soluţie.*Să notăm

$$g(\eta) = \begin{cases} \frac{b^a}{\Gamma(a)} \eta^{a-1} e^{-\eta b}, & daca \quad \eta > 0 \\ 0, & \eta \geq 0. \end{cases}$$

Atunci densitatea de repartiţie a lui $\mathbf{X}$ este

$$f(x_1, x_2, ..., x_k) = \begin{cases} (\prod_{i=1}^{k} \lambda_i) \int_0^\infty \eta^{k+a-1} \frac{b^a}{\Gamma(a)} e^{-\sum_{i=1}^{k} x_i \lambda_i \eta} e^{-b\eta} d\eta,\ x_i > 0 = \\ o, \quad altfel \end{cases}$$

$$= \begin{cases} \frac{b^a}{\Gamma(a)} (\prod_{i=1}^{k} \lambda_i) \int_0^\infty \eta^{a+b-1} e^{-\eta(\sum_{i=1}^{k} x_i \lambda_i + b)} d\eta,\ daca\ x_i > 0 = \\ 0, \quad altfel \end{cases}$$

$$= \begin{cases} \frac{b^a}{\Gamma(a)} (\prod_{i=1}^{k} \lambda_i) \frac{\Gamma(a+k)}{(b + \sum_{i=1}^{k} \lambda_i x_i)^{a+k}},\ x_i > 0 \\ 0,\ altfel \end{cases} =$$

$$= \begin{cases} \frac{(\prod_{i=1}^{k}\theta_i)a(a+1)...(a+k-1)}{(1+\sum\limits_{i=1}^{k}\theta_i x_i)^{a+k}}, & daca\ x_i > 0 \\ 0, \quad altfel, \end{cases} \qquad \theta_i = \frac{\lambda_i}{b}.$$

Simularea lui $\mathbf{X}$ se poate face prin metoda compunerii.

Se poate arăta (folosind (4.6)) că pentru $x_i > 0$

$$f(x_m|x_1,...,x_{m-1}) = \frac{\theta_m(a+m-1)(1+\sum\limits_{i=1}^{m-1}\theta_i x_i)^{a+m-1}}{(1+\sum\limits_{i=1}^{m-1}\theta_i x_i + \theta_m x_m)^{a+m}}$$

de unde

$$F(x_m|x_1,...,x_{m-1}) = 1 - \left(\frac{1+\sum\limits_{i=1}^{k-1}\theta_i x_i}{1+\sum\limits_{i=1}^{m}\theta_i x_i}\right)^{a+m-1}.$$

Ultima formulă permite simularea lui $\mathbf{X}$ utilizând teorema 4.1.

E4.4 Fie vectorul aleator $(X,Y)'$ cu componente pozitive având densitatea de repartiţie

$$f(x,y) = \begin{cases} \alpha(\alpha+\beta)e^{-(\alpha+\beta)y}, & daca \quad 0 \le x < y \\ \beta(\alpha+\beta)e^{-(\alpha+\beta)x} & daca \quad 0 \le y < x \end{cases}$$

(repartiţia bidimensională a lui Freund). Să se determine $E(X), E(Y)$, $Var(X), Var(Y)$ şi să se găsească o metodă de simulare a vectorului $(X,Y)'$.

Indicaţie. Densităţile marginale şi sunt

$$f_X(x) = [\alpha + x\beta(\alpha+\beta)]e^{-(\alpha+\beta)x},\ x > 0, \quad f_Y(y) = (\alpha+\beta)e^{-(\alpha+\beta)y},\ y > 0$$

iar

$$F_x(x) = 1 - (1+\beta x)e^{-(\alpha+\beta)x}$$

de unde se pot calcula cu uşurinţă mediile şi dispersiile.

Pentru simulare se aplică metoda inversă (teorema 4.1) unde $X \rightsquigarrow F_x(x)$ iar $f(y|x)$ este de forma

$$f(y|x) = \begin{cases} \frac{\beta(\alpha+\beta)}{\alpha+\beta(\alpha+\beta)x} & daca \quad 0 < y < x \\ \frac{\alpha(\alpha+\beta)e^{-(\alpha+\beta)(y-x)}}{\alpha+\beta(\alpha+\beta)x} & daca \quad y > x. \end{cases}$$

Se observă că

$$f(y|x) = p\frac{1}{x}I_{(0,x)} + (1-p)(\alpha+\beta)e^{-(\alpha+\beta)(y-x)}I_{(x,\infty)},\ p = \frac{\beta(\alpha+\beta)x}{\alpha+\beta(\alpha+\beta)x}$$

adica $f(y|x)$ este amestecarea densităţii uniforme pe $[0,x]$ cu densitatea $Exp(\alpha+\beta))$ trunchiată pe (x,∞). ($I_A(x)$ este funcţia indicator a lui A).

E4.5 Enunţaţi şi demonstraţi *teorema de respingere a infăşurătoarei* pentru vectori aleatori. Aplicaţi această metodă pentru simularea vectorului $(X,Y)'$ din exemplul 4.4.

Soluţie. Teorema infăşurătoarei in cazul multidimensional este

Teorema 4. 5 *Fie vectorul aleator* $\mathbf{X} \rightsquigarrow f(\mathbf{x})$ *pe care dorim sa-l simulăm. Fie* $\mathbf{Y} \rightsquigarrow h(\mathbf{x})$ *pe care ştim sa-l simulăm. Presupundem că* $f(\mathbf{x})$ *şi* $h(\mathbf{x})$ *au acelaş suport* $S \subseteq R^k$ *şi că există* α, $0 < \alpha < \infty$ *astfel incât* $f(\mathbf{x}) \leq \alpha h(\mathbf{x})$, $\forall \mathbf{x} \in R^k$. *Dacă* U *este o variabilă uniformă* $0-1$ *independentă de* $\mathbf{Y}$, *atunci densitatea de repartiţie a lui* $\mathbf{Y}$ *condiţionată de* $0 < U \leq \frac{f(\mathbf{Y})}{\alpha h(\mathbf{Y})}$ *este* $f(\mathbf{x})$.

Pentru a aplica această teoremă in cazul exemplului 4.4 să observăm că densitatea de repartiţie este

$$f(x,y) = \begin{cases} e^{-(x+y+\theta xy)}[(1+\theta x)(1+\theta y)-\theta], \, daca \, x,y>0 \\ 0, \, altfel. \end{cases}$$

Pentru a aplica teorema infăşurătoarei vom lua

$$h(x,y) = \begin{cases} e^{-(x+y)} \quad dacax>0, \, y>0 \\ 0 \quad altfel, \end{cases}$$

adică vectorul corespunzând densităţii infăşurătoare are componentele exponenţiale $Exp(1)$ independente.

E4.6. Fie $U_1, U_2, ..., U_n$ numere aleatoare uniforme pe $[0,1]$ independente şi să considerăm aceste numere ordonate (adică statistica de ordine) $U_{(1)} \leq U_{(2)} \leq ... \leq U_{(n)}$. Notăm $U_{(0)} = 0$. $U_{(n+1)} = 1$. Să se arate că vectorul aleator $(S_1, S_2, ..., S_n)$, $S_i = U_{(i)} - U_{(i-1)}, 1 \leq i \leq n+1$, are repartiţie uniformă pe simplexul

$$A_n = \{(x_1, x_2, ..., x_n); x_i \geq 0, \sum_{i=1}^{n} x_i \leq 1\}.$$

Soluţie. Numerele $U_{(1)}, ..., U_{(n)}$ sunt uniform repartizate pe domeniul (simplexul)

$$B_n = \{(x_1, ..., x_n); 0 \leq x_1 \leq ... \leq x_n \leq 1\}.$$

Să facem acum transformarea

$$s_1 = u_1; \; s_2 = u_2 - u_1, \; , ..., s_n = u_n - u_{n-1}$$

a cărei inversă este

$$u_1 = s_1,\ u_2 = s_1 + s_2,\ , ..., u_n = s_1 + s_2 + ... + s_n.$$

Jacobianul acestei transformări este $J = 1$. Deoarece repartiţia lui $(U_{(1)}, ..., U_{(n)})$ este uniformă adică are densitatea

$$f(u_1, ..., u_n) = \begin{cases} \frac{1}{mes\, B_n}, & (u_1, ..., u_n) \in B_n \\ 0, & (u_1, ..., u_n) \notin B_n \end{cases}$$

rezultă că $(S_1, ..., S_n)$ are tot densitate uniformă pe A_n (transformatul lui B_n) adică

$$g(s_1, ..., s_n) = \begin{cases} \frac{1}{mes\, A_n} & (s_1, ..., s_n) \in A_n \\ 0 & (s_1, ..., s_n) \notin A_n, \end{cases} \quad mes\, A_n = mes\, B_n J.$$

Notă. De aici rezultă că simularea unui vector uniform pe domeniul $A_n \in R^n$ se realizează simulând $U_1, ..., U_n$ uniforme pe $[0, 1]$ independente, şi calculând apoi $(S_1, ..., S_n)$ cu ajutorul statisticii de ordine $U_{(1)} \leq U_{(2)} \leq ... \leq U_{(n)}$.

E4.7 Folosind teorema 4.1 construiţi un algoritm pentru simularea vectoruluialeator discret $(X, Y)'$ cu X şi Y luând mulţimi finite de valori.

Indicaţie. Repartiţia discretă a lui $(X, Y)'$ este dată de *tabela de contingenţă*

	$Y = b_1$	.	.	. b_r	$p_{i.}$
$X = a_1$	p_{11}.	.	.	.p_{1r}	$p_{1.}$
a_2	p_{21}	.	.	.p_{2r}	$p_{2.}$
.	.	.	.		
a_s	p_{s1}	.	.	.p_{sr}	$p_{s.}$
	$p_{.1}$	.	.	.$p_{.r}$	

Repartiţia marginală a lui X este

$$p_{1.}\ p_{2.}\ ...\ p_{s.}$$

şi cu ajutorul ei se simulează X, iar repartiţia lui Y condiţionată de $X = a_i$ este

$$\frac{p_{i1}}{p_{i.}}\ \frac{p_{i2}}{p_{i.}}\ ...\ \frac{p_{ir}}{p_{i.}}$$

şi din ea se simulează Y. (In tabela şi in formulele de mai sus, indicii *inlocuiţi cu punct* (.) notează rezultatele insumărilor pe coloane sau pe linii).

Notă. Dacă pentru un vector bidimensional (X, Y) se dă o selecţie $(X_1, Y_1), (X_2, Y_2), ..., (X_n, Y_n)$, cu ajutorul ei se poate construi o *tabelă de frecvenţă bidimensională* asemănătoare histogramei. Ea va fi de forma tabelei de mai sus unde in loc de p_{ij} se vor scrie frecvenţele f_{ij}. Aceasta se numeşte *tabelă de contingenţă.*

Cap. 5

Simularea proceselor stochastice

5.1 Generalităţi

O familie de variabile aleatoare, $\{X_t\}_{t\in T}, T \subset R$, deprinzând de parametrul real t (presupus a fi *timpul*) se numeşte *proces stochastic.* Dacă T este o mulţime discretă (deobicei $T = \{0, 1, 2, ...\}$), atunci procesul se numeşte *lanţ.* Dacă T este un interval $I \subseteq R$ atunci procesul este *cu timp continuu.* Valorile lui X_t se numesc stări. Fie S mulţimea valorilor unui proces stochastic. Dacă mulţimea S este discretă spunem că *procesul este cu stări discrete* iar dacă S este de puterea continuumului, spunem că *procesul este cu stări continue.* Mulţimea $\{(t, X_t)|t \in T\}$ se numeşte *traiectorie a procesului stochastic.*

Procesul stochastic este cunoscut dacă pentru $\forall n$, $t_1, t_2, ..., t_n$ este cunoscută funcţia de repartiţie
$F_{t_1,t_2,...t_n}(x_1, x_2, ..., x_n) = P(X_{t_1} < x_1, ..., X_{t_n} < x_n)$.

Definiţia 5. 1 *Procesul stochastic* $\{X_t\}_{t\in T}$ *se numeşte* **staţionar tare** *dacă* $\forall n$, $h \in R$, $t_1 < t_2 < ... < t_n$ *avem*

$$F_{t_1+h,t_2+h,...,t_n+h}(x_1, x_2, ..., x_n) = F_{t_1,t_2,...,t_n}(x_1, x_2, ..., x_n). \tag{5.1}$$

Procesul $\{X_t\}_{t\in T}$ *se numeşte* **staţionar slab** *sau simplu* **staţionar** *dacă are momente de ordinul doi, adică există şi sunt finite* $m_t = E[X_t]$, $\sigma_{st} = Cov[X_s, X_t]$, $s < t$ *şi* $m_t = m = const$, $\sigma_{st} = \sigma(t - s)$.

Observăm că pentru un proces staţionar tare, avem in particular $F_{t+h}(x) = F_t(x)$, adică procesul are aceeaşi funcţie de repartiţie la orice moment de timp t, dar variabilele X_t pot să nu fie independente pentru diverse valori ale lui t. Dacă variabilele X_t sunt independente atunci suntem in cazul banal clasic, al unei familii de variabile aleatoare independente.

Se poate observa cu uşurinţă că dacă $\{X_t\}_{t\in T}$ este staţionar tare şi are momente de ordinul doi, atunci el este staţionar; reciproc nu este insă adevărat.

In ceeace priveşte simularea proceselor stochastice, ne interesează să producem cu calculatorul *puncte* ale unor traiectorii finite. In cazul unui lanţ de exemplu este necesar să producem un număr finit de puncte ale traiectoriei lanţului. In cazul unui proces cu timp continuu simulăm dintr-o traiectorie numai un număr finit de puncte ale acesteia, deobicei corespunzând valorilor discrete ale timpului $t = 0, 1, 2, ..., n$. De fapt problema construirii unui algoritm de simulare a unui proces revine la următoarele: dându-se o valoare X_0 a procesului (presupusă a se afla pe o traiectorie a sa la momentul $t = 0$), se cere să se simuleze valoarea X_1 aflată pe aceeaşi traiectorie la momentul $t = 1$.

Desigur, problema simulării unui proces stochastic este mult mai complicată decât a variabilelor aleatoare. In cele ce urmează ne vom ocupa de simularea *lanţurilor şi proceselor Markov* şi de simularea unor procese *staţionare* care au repartiţii de probabilitate cunoscute.

5.2 Lanţuri şi procese Markov

Procesele şi lanţurile Markov sunt acelea care satisfac *proprietatea lui Markov.*

Definiţia 5. 2 *Procesul $\{X_t\}_{t\in T}$ discret, este un proces Markov dacă satisface proprietatea*

$$P(X_{t_n} = x_n | X_{t_{n-1}} = x_{n-1}, ..., X_{t_1} = x_1) = P(X_{t_n} = x_n | X_{t_{n-1}} = x_{n-1}) \tag{5.2}$$

adică procesul **nu are memorie.** *Probabilităţile $P(X_{t_n} = x_n | X_{t_{n-1}} = x_{n-1})$ se numesc probabilităţi de tranziţie sau probabilităţi de trecere.*

Dacă $T = \{0, 1, 2, ..., n, \}$ şi $S = \{1, 2, ..., m\}$, atunci avem de-a face cu un lanţ Markov cu un număr finit de stări. Acest lanţ este caracterizat de vectorul $\pi = (\pi_1, \pi_2, ..., \pi_n)'$ care reprezintă repartiţia iniţială a stărilor şi de mulţimea probabilităţilor de tranziţie $P_{ij}(s, t) = P(X_t = j | X_s = i)$, $i, j \in S$, $s < t$, $s, t \in N$.

Se arată că probabilităţile de tranziţie satisfac *relaţia lui Chapman-Kolmogorov,* adică

$$P_{ij}(s,t) = \sum_{k\in S} \sum_{s\le r\le t} P_{ik}(s,r)P_{kj}(r,t). \qquad (5.2')$$

Observăm că $P_{ij}(s,t)$ definesc nişte *matrici de probabilităţi de tranziţie* care au proprietatea

$$\sum_{j\in S} P_{ij}(s,t) = 1. \qquad (5.3)$$

Dacă $t - s = 1$ atunci matricile $||P_{ij}(s,s+1)||, s \in N$ sunt *matrici de probabilităţi de tranziţie intr-un pas* şi matricile $||P_{ij}(s,t)||, s,t \in N$ sunt matrici de tranziţie in $t - s$ paşi. Dacă $P_{ij}(s,s+1) = p_{ij}$ (adică aceste probabilităţi nu depind de momentele s când au loc tranziţiile intr-un pas) spunem că *lanţul Markov este omogen sau staţionar.*

Să notăm cu P matricea probabilităţilor de tranziţie intr-un pas pentru un lanţ Markov omogen şi cu $P^{(n)}$ matricea probabilităţilor de trecere *in n paşi* pentru acelaş lanţ. Atunci din (5.2') se deduce cu uşurinţă că

$$P^{(n+1)} = P^{(n)}P = PP^{(n)} = P^{n+1}. \qquad (5.2'')$$

Dacă repartiţia iniţială $\pi = (\pi_1, \pi_2, ..., \pi_m)'$ este cunoscută atunci folosind *formula probabilităţii totale*

$$P(X_n = s) = \sum_{j=1}^{m} P_{sj}^{(n)}\pi_j \qquad (5.3)$$

deducem, utilizând şi (5.2") repartiţia lanţului omogen la momentul n, $\pi^{(n)}$, astfel

$$\pi^{(n)} = P^n\pi,\ P = ||p_{ij}||,\ \sum_{j=1}^{m} p_{ij} = 1. \qquad (5.4)$$

Dacă o stare i a unui lanţ omogen are proprietatea că există un j, $j \neq i$ astfel incât $p_{ij} > 0$, atunci starea i este *stare de tranziţie,* iar dacă $p_{ii} = 1$ atunci starea i este *stare absorbantă.* (Când lanţul intră intr-o stare absorbantă, el nu mai părăseşte acea stare).

5.3 Simularea unui lanţ Markov

Să presupunem că se cunoaşte repartiţia iniţială $\pi = (\pi_1, \pi_2, ..., \pi_m)'$ şi matricea probabilităţilor de tranziţie $P = ||p_{ij}||$. Atunci starea iniţială $I = i$ se

simulează ca variabila discretă

$$I : \begin{pmatrix} 1, 2, ..., m \\ \pi_1, \pi_2, ..., \pi_m \end{pmatrix}. \tag{5.5}$$

Dacă lanţul se află in starea i atunci starea aleatoare următoare in care trece lanţul se simulează ca variabila discretă

$$J : \begin{pmatrix} 1, 2, ..., m \\ p_{i1}, p_{i2}, ..., p_{im} \end{pmatrix}. \tag{5,5'}$$

Deci, algoritmul de simulare a unei traiectorii $i_1, i_2, ..., i_n$ când se dau π, P este

Algoritmul MARKOVOMOG de simulare a lanţului omogen

Citeşte $\pi, P = ||p_{ij}||$; *Citeţe* n;

calculează $f_i = \sum_{\alpha=1}^{i} \pi_\alpha$, $F_{ij} = \sum_{\alpha=1}^{j} p_{i\alpha}, 1 \leq i \leq m$; (Acestea sunt calcule pregătitoare);

generează $U := random$; $i := 1$: **while** $U \geq f_i$ **do** $i := i + 1$;

for $k := 1$ **to** n **do**

begin

Generează $U := random$; *Ia* $j := 1$;

while $U \geq F_{ij}$ **do** $j := j + 1$;

Ia $i := j, i_k = i$; *Scrie* i_k;

end.

In mod asemănător se poate face şi simularea lanţului neomogen, dacă se cunosc probabilităţile de tranziţie intr-un pas $P_{ij}(s, s + 1)$, $1 \leq s \leq n$. In acest caz, probabilităţile $P_{ij}(s, s + 1)$ sunt citite in cadrul ciclului **for** al algoritmului când se calcululează pe rând şi *probabilităţile cumulate* F_{ij}.

Să mai observăm că dacă lanţul Markov are o stare absorbantă i_0, celelalte fiind de tranziţie, atunci din orice stare iniţială i ar porni procesul, există o traiectorie finită de lungime $N_{(i)}$ astfel incât $X_1 = i, ..., X_{N_{(i)}} = i_0$, adică lanţul se *opreşte* in starea absorbantă i_0.

Pentru un *proces Markov cu o mulţime continuă* S *de stări* $\{X_t\}_{t \in T}$ se presupune dată densitatea de repartiţie iniţială $p(x) > 0$, $x \in S$ şi densitatea de probabilitate *de tranziţie* $q(x, y) > 0$, $x, y \in S$, densităţi care satisfac condiţiile

$$\int_S p(x)dx = 1, \quad \int_S q(x, y)dy = 1. \tag{5.6}$$

Algoritmul in acest caz este

Algoritmul MARKOVCONT de simulare a unui proces Markov continuu

Intrare n; (n =numărul de puncte ale traiectoriei);

generează starea iniţială $X_0 \rightsquigarrow p(x)$;

for $i := 1$ **to** $n-1$ **do**

begin

Generează $Y \rightsquigarrow q(X_0, y)$;

Ia $X_i := Y; X_0 := Y$; *Scrie* X_i;

end.

Punctele traiectoriei generate sunt $X_0, X_1, ..., X_{n-1}$.

5.4 Simularea unor procese gausiene staţionare

Fie procesul staţionar care are *funcţia de autocovarianţă*

$$\phi(t) = Cov(X_s, X_{s+t}),\ t = 0, 1, 2, ... \quad \phi(0) = Var(X_t) = \sigma^2 \tag{4.7}$$

şi fără a pierde generalitatea considerăm $m_t = m = 0$. Dacă notăm $\rho(X_s, X_{t+s}) = Corr(X_s, X_{t+s})$ rezultă că $\rho(X_s, X_{t+s}) = \rho(t)$, $t = 1, 2, ...$ Funcţia $\rho(t)$ este *funcţia de autocorelaţie a procesului.*

Dacă pentru orice t, repartiţia lui X_t este normală $N(0, \sigma)$, atunci procesul $\{X_t\}_{t \in T}$ este un *proces gausian.*

5.4.1 Procesul gausian cu funcţie de autocorelaţie exponenţială

Acest proces are funçtia de autocorelaţie

$$\rho(t) = \frac{\phi(t)}{\phi(0)} = \theta^t,\ t = 1, 2, ...\ 0 < \theta < 1 \tag{5.8}$$

şi să considerăm σ^2 dispersia sa. Acest proces se simulează cu formula de recurenţă

$$Z_0 \rightsquigarrow N(0,1), \quad Z_{t+1} = \theta Z_t + \sqrt{1-\theta^2}\epsilon_t,\ t = 1, 2, ..., X_{t+1} = \sigma Z_{t+1} \tag{5.8'}$$

unde $\epsilon_t, t = 1, 2, ...$ sunt variabile $N(0,1)$ independente şi independente de Z_t. Intr-adevăr, Z_{t+1} are repartiţie normală şi $Var(X_t) = \sigma^2$ şi

$$Cov(Z_t, Z_{t+1}) = Cov(Z_t, Z_{t+1}) = E[Z_t, Z_{t+1}] = E[Z_t(\theta Z_t + \sqrt{1-\theta^2}\epsilon_t)] =$$

$$= E[\theta Z_t^2]E[\sqrt{1-\theta^2}Z_t\epsilon_t] = \theta + \sqrt{1-\theta^2}E[Z_t]E[\epsilon_t] = \theta.$$

Asemănător

$$Cov[Z_t, Z_{t+2}] = E[\theta Z_t Z_{t+1}] + E[Z_t\sqrt{1-\theta^2}\epsilon_t] =$$

$$= \theta\theta + E[Z_{t+1}]E[\epsilon_t]\sqrt{1-\theta^2} = \theta^2.$$

Prin inducţie se deduce că $E[Z_1 Z_{t+1}] = \theta^t$ şi formula (5.8') este justificată. Această formulă iterativă permite deci simularea unei traiectorii $X_1, X_2, ..., X_n$. Se observă că σ nu joacă un rol important şi de aceea putem considera $\sigma = 1$.

5.4.2 Procesul gausian cu funcţie de autocorelaţie liniară

Să presupunem că pentru un intreg m dat funcţia de autocorelaţie a procesului gausian $\{X_t\}$ este de forma

$$\rho(t) = \begin{cases} 1 - \frac{t}{m}, \, daca \, 1 \leq t \leq m \\ o, altfel \end{cases}, \, Cov(X_t, X_{t+k}) = \Phi(k), \qquad (5.9)$$

$$\rho(k) = \frac{\Phi(k)}{\Phi(0)}, \quad Var(X_t) = \sigma_*^2 = \Phi(0)$$

şi dorim să simulăm un proces gausian $\{X_t\}$ cu această funcţie de autocorelaţie şi cu $Var(X_t) = \sigma_*^2$.

Vom simula X_t utilizând *teorema limită centrală.* Pentru aceasta să considerăm variabilele aleatoare V_i,$i \geq 1$, independente şi uniform distribuite pe $[-a, a]$, $a > 0$ cu $a-$dat. Avem deci

$$E[V] = 0, \quad Var[V] = \frac{a^2}{3} = \sigma^2. \qquad (5.10)$$

Pentru a genera procesul gausian $\{X_t\}_{t\in N}$ de medie 0 şi dispersie σ_*^2, alegem mai intâi $N > 10$ (impus de teorema limită centrală) şi un număr natural p, $p > 0$ a cărui valoare va fi precizată mai jos. Simulăm valorile de selecţie $X_t, X_{t+1}, t \in N$, astfel

$$X_t = \sum_{i=1}^{n} V_i, \quad X_{t+1} = \sum_{i=p+1}^{N+p} V_i, \qquad (5.11)$$

şi vom determina a, σ, p astfel incât

$$Corr(X_t, X_{t+k}) = \rho(k). \tag{5.12}$$

Deoarece in expresiile lui X_t, X_{t+1} apar $N - p$ variabile V_i independente, avem

$$Cov[X_t, X_{t+k}] = E\{(X_t - X_{t+k})^2\} = E\{X_t^2 - 2X_tX_{t+k} + X_{t+k}^2\} =$$

$$= E\{(X_t - X_{t+k})^2\} = 2N\sigma^2 - 2\Phi(k). \tag{5.12}$$

Dacă $k < N/p$ atunci in expresia $E[(X_t - X_{t+k})^2]$ există numai $2kp$ variabile de selecţie V_i independente stochastic, de unde

$$E[(X_t - X_{t+k})^2] = 2kp^2 = 2N\sigma^2 - 2\Phi(k) \rightarrow kp^2 = N\sigma^2 - \Phi(k).$$

Din ultima relaţie deducem că

$$\rho(k) = \frac{\Phi(k)}{N\sigma^2} = 1 - k\frac{p}{N}$$

şi identificând cu prima relaţie (5.9) trebuie să luăm

$$p = \frac{N}{m}$$

pentru a obţine expresia cerută a lui $\rho(k)$.

Trebuie să determinăm acum valoarea lui a. Deoarece $Var(X_t) = N\sigma^2 = \sigma_*^2$ şi conform (5.10) $a = \sigma\sqrt{3}$, rezultă că trebuie să avem

$$\sigma = \frac{\sigma_*}{\sqrt{N}} \quad a = \frac{\sigma_*}{\sqrt{N}}\sqrt{3}.$$

In concluzie, dându-se m, σ_* şi N (de ex. $N = 12$,) simularea procesului gausian cu funcţia de autocorelaţie dată de prima formulă (5.9) se realizează cu următorul algoritm

Algoritmul PRGAUSLIN de simulare a procesului gausian cu funcţie de autocorelaţie liniară

Intrare m, σ_*, N; *Calculează* $a := \frac{\sigma_*}{\sqrt{N}}\sqrt{3}$; *Determină* p *intreg astfel incât* $p = \frac{N}{m}$; (Dacă pentru m dat şi N iniţial ales nu există intregul p astfel incât $N = mp$ atunci se alege alt $N > 10$ pentru care să poată fi determinat p. Calculele precedente sunt calcule pregătitoare);

for $i := 1$ **to** N **do begin**

$X_0 := 0$; *Generează* $U := random$;
Calculează $W := -a + 2aU$; $V_i := W$; $X_0 := X_0 + V_i$;
end:
Scrie X_0; $X_1 := 0$; (*)
for $i := 1$ **to** $N - p$ **do begin** $V_i := V_{i+p}$; $X_1 := X_1 + V_i$ **end;**
for $i := N - p + 1$ **to** N **do begin**
Generează $U := random$; $W := -a + 2aU$; $V_i := W$;
$X_1 := X_1 + V_i$; *Scrie* X_1;
end.

Repetând de T ori instrucţiunile algoritmului precedent ce incep cu calculul sumei X_1 (instrucţiunea *) se obţine traiectoria $X_1, X_2, ..., X_T$ a procesului gausian.

• **Simularea procesului de zgomot alb pur.** Procesul *zgomotului alb* $\{X_t\}_{t\in T}$ este descris de următoarele relaţii de recurenţă

$$X_0 = \sqrt{\frac{1-\theta}{1+\theta}}V_0,\ X_t = \theta X_{t-1} + (1-\theta)V_t,\ 0 < \theta < 1, \tag{5.13}$$

unde$V_t, t = 0, 1, 2, ...$ sunt variable aleatoare independente şi identic repartizate având proprietăţile $E(V_t) = 0$, $Var(V_t) = \sigma^2$. Se observă că

$$E(X_t) = 0,\ Var(X_t) = \sigma_*^2 = \frac{1-\theta}{1+\theta}\sigma^2. \tag{5.14}$$

Se deduce apoi că

$$Corr(X_t, X_{t+k}) = \rho(k) = \theta^k. \tag{5.15}$$

Intr-adevăr, deoarece X_{t-1} este independent de V_t rezultă că

$$\Phi(1) = Cov(X_t, X_{t-1}) = E[X_t, X_{t-1})] = \theta E[X_{t-1}^2] = \theta\sigma_*^2.$$

Prin inducţie se arată că

$$\Phi(k) = Cov(X_t, X_{t+k}) = \theta^k\sigma_*^2$$

şi (5.15) este demonnstrată.

Algoritmul pentru simularea zgomotului alb este uşor de construit.

Dacă V_t, $t = 0, 1, ...$ sunt variabile unifirme pe $[-a, a]$ cu a dat, atunci procesul $\{X_t\}_{t\in T}$ se numeşte *zgomot alb pur.* Desigur, asemănător celor de mai sus se deduce că

$$E[V_t] = 0,\ Var(V_t) = \sigma^2 = \frac{a^2}{3},\ Var(X_t) = \sigma_*^2 = \frac{1-\theta}{1+\theta}\sigma^2$$

de unde, dându-se θ şi σ_* avem

$$\sigma = \sigma_* \sqrt{\frac{1+\theta}{1-\theta}},\ a = \sigma\sqrt{3}.$$

Cu aceste elemente se construieşte uşor algoritmul de simulare al zgomotului alb pur pe baza relaţiei (5.13).

5.5 Simularea procesului Poisson

Procesul Poisson este un caz particular de proces de naştere şi deces (vezi \$ 7.1.1), mai precis este un proces de naştere pur cu intensitatea $\lambda_n = \lambda$, a cărui repartiţie satisface eceaţiile diferenţiale

$$P_0'(t) = -\lambda P_0(t), \quad P_n'(t) = -\lambda P_n(t) + \lambda P_{n-1}(t),\ n \geq 1. \tag{5.16}$$

Dacă se dau condiţiile iniţiale naturale $P_0(0) = 1, P_i(0) = 0, i > 0$ atunci soluţia sistemului (5.16) este

$$P_n(t) = \frac{(\lambda t)^n}{n!} e^{-\lambda t} \tag{5.16'}$$

iar dacă λ depinde de $t, (\lambda = \lambda(t))$ atunci soluţia este

$$P_n(t) = \frac{(\Lambda(t))^n}{n!} e^{-\Lambda(t)}, \quad \Lambda(t) = \int_0^t \lambda(u)du. \tag{5.16''}$$

Procesul având repartiţia (5.16') se numeşte *proces Poisson omogen (PPO)* iar procesul având repartiţia (5.16") se numeşte *proces Poisson neomogen (PPNO).*

Simularea unei valori de selecţie a PPO sau PPNO se poate realiza cu ajutorul metodei din \$ 3.6.3. Pentru aceasta reamintim că o valoare de selecţie $N \rightsquigarrow Poisson(\lambda)$ satisface condiţia

$$S_N \leq 1 < S_{N+1}, \quad S_k = \sum_{i=1}^{k} Z_i, \quad Z_i \rightsquigarrow Exp(\lambda).$$

Conceptul de PPO poate fi generalizat astfel: se consideră că procesul are drept parametru tot $t = timpul$, şi ia ca valori puncte $X \in A \subset R^k$; procesul, numit *PPOU uniform pe A de intensitate* λ, ia ca valori $X_i \in A, 1 \leq i \leq N$ dacă $N \rightsquigarrow Poisson(\lambda Vol(A))$ şi X_i sunt puncte uniform repartizate in A. Desigur, PPOU pe A se simulează astfel

Algoritm PPOU
generează $N \rightsquigarrow Poisson(\lambda Vol(A))$;
generează $X_1, X_2, ..., X_k$ *puncte uniform repartizate in* A.

Punctele X_i reprezintă valori aleatoare ale PPOU.

In mod analog se poate considera $PPNOU$ pe A când variabila aleatoare N are repartiţia $N \rightsquigarrow Poisson(\Lambda(t)Vol(A))$.

Cap. 6

Metoda Monte Carlo

6.1 Generalităţi

Sub denumirea de *metode Monte Carlo* sunt cuprinse o serie de tehnici de rezolvare a diverse probleme utilizând *numere aleatoare, variabile aleatoare sau procese stochastice* simulate cu calculatorul. Mai precis se acceptă următoarea definiţie

Definiţia 6. 1 *Fie de rezolvat o problemă numerică $\mathcal{P}$ care are soluţia θ. O metodă Monte Carlo pentru rezolvarea lui $\mathcal{P}$ constă in următoarele:*

- se asociază **in mod adecvat** *un proces stochastic ξ problemei astfel incât cu ajutorul lui ξ să se poată estima θ; (de exemplu $\theta = E[\tau(\xi)]$, sau $E[\tau(\xi)] \to \theta$, sau $\tau(\xi)$ converge intr-un sens probabilist către θ, unde τ este o funcţie dată). Să notăm deci $\tau = \tau(\xi)$ estimatorul lui θ. ($\tau(\xi)$ se numeşte* **estimator primar.***).*

- se simulează o selecţie $\xi_1, \xi_2, ..., \xi_n$ asupra lui ξ şi se calculează un estimator $\overline{\tau}_n$ al lui $\tau(\xi)$; dacă de exemplu $E[\tau(\xi)] = \theta$ atunci estimatorul lui $\tau(\xi)$ este

$$\overline{\tau}_n = \frac{1}{n} \sum_{i=1}^{n} \tau(\xi_i) \tag{6.1}$$

şi el se numeşte **estimator secundar.**

- soluţia problemei $\mathcal{P}$ se aproximează cu $\overline{\tau}_n$.

Dacă estimatorul primar $\tau(\xi)$ este *nedeplasat,* adică $E[\tau(\xi)] = \theta$, atunci estimatorul secundar $\overline{\tau}_n$ satisface de regulă *legea numerelor mari* şi deci $\overline{\tau}_n \to \theta$ in probabilitate.

Desigur, nu există o regulă sau o tehnică matematică bine precizată pentru construcţia unei metode Monte Carlo. Este mai degrabă *o artă* necesară pentru construcţia unei astfel de metode.

De cele mai multe ori, estimatorul primar este $\tau(\xi)$ astfel ca $E[\tau(\xi)] = \theta$. (Deocamdată considerăm că θ este un număr, dar el poate fi şi vector sau o funcţie). Estimaţia Monte Carlo aproximează soluţia θ, aşa cum s-a precizat. Să presupunem că există $\sigma^2 = Var\{\tau(\xi)\} < \infty$ şi că dorim să aproximăm θ cu o anumită eroare dată ϵ. Dar θ este determinist şi $\overline{\tau}_n$ este aleator. De aceea eroarea ϵ trebuie considerată *in probabilitate,* adică

$$P(|\overline{\tau}_n - \theta| < \epsilon) = p = 1 - \delta \tag{6.2}$$

unde δ este o probabilitate mică (un *risc* mic). Deci conform legii numerelor mari, trebuie să alegem n astfel incât pentru ϵ şi δ suficient de mici daţi să fie satisfăcută relaţia (6.2).

Teorema 6. 1 *Volumul n al selecţiei $\xi_1, \xi_2, ..., \xi_n$ necesar pentru a estima θ prin $\overline{\tau}_n$ cu eroarea ϵ dată şi cu riscul δ admis este*

$$n \geq n_0, \quad n_0 = \left[\frac{t_\delta^2 \sigma^2}{\epsilon^2}\right] + 1, \; t_\delta = \frac{1}{\sqrt{\delta}} \tag{6.3}$$

unde $[x]$ este partea intreagă a lui x.

Demonstraţie. Deoarece $Var(\tau(\xi)) = \sigma^2$ rezultă că

$$Var(\overline{\tau}_n) = Var\frac{1}{n}\sum_{i=1}^{n}\tau(\xi_i) = \frac{\sigma^2}{n}.$$

Aplicând *inegalitatea lui Cebâşev* variabilei $\overline{\tau}_n$ avem

$$P(|\overline{\tau}_n - \theta| < t\frac{\sigma}{\sqrt{n}}) \geq 1 - \frac{1}{t^2} = p.$$

Deoarece dorim ca $P(|\overline{\tau}_n - \theta| < \epsilon) \geq p = 1 - \delta$, impunem condiţiile

$$1 - \frac{1}{t^2} = 1 - \delta, \quad t\frac{\sigma}{\sqrt{n}} \leq \epsilon$$

de unde deducem

$$t = t_\delta = \frac{1}{\sqrt{\delta}}, \quad n \geq \frac{t_\delta^2 \sigma^2}{\epsilon^2}$$

de unde rezultă (6.3) şi teorema este demonstrată.

Cu ajutorul metodei Monte Carlo se rezolvă multe tipuri de probleme cum sunt: *calculul integralelor, rezolvarea sistemelor de ecuaţii liniare şi a ecuaţiilor integrale, rezolvarea problemei Dirichlet pentru ecuaţii cu derivate parţiale de tip eliptic, rezolvarea problemelor de oprimizare cu restricţii, etc.* O parte din astfel de probleme vor fi tratate in continuare.

6.1.1 Calculul integralelor

Calculul numeric al integralelor, aşa zisele formule de cuadratură, constituie un domeniu bine studiat in cadrul analizei numerice clasice, mai ales in cazul *unidimensional.* In cazul multidimensional insă, calculul integralelor prin *discretizare,* (aproximare prin sume integrale) este dificil de realizat practic. De aceea in cazul multidimensional mult mai bune sunt procedurile Monte Carlo de calcul al integralelor. Totuşi, pentru uşurarea expunerii, aici vom prezenta unele metode Monte Carlo de calcul al integralelor numai in cazul unidimensional, având ca scop numai simplificarea scrierii. Va trebui insă să avem ca obiectiv utilizarea metodelor mai ales in cazul multidimensional. Desigur, metodele Monte Carlo se aplică şi la calculul integralelor unidimensionale când formulele de cuadratură sunt ne aplicabile (de ex. in cazul integralelor improprii) sau se aplică cu dificultate.

Să presupunem deci, fără a restrânge generalitatea, că avem de calculat integrala

$$\theta = \int_0^1 f(x)dx. \tag{6.4}$$

Dacă am avea de calculat

$$I = \int_a^b g(t)dt, \quad -\infty < a < b < +\infty$$

atunci prin transformarea

$$x = \frac{t-a}{b-a}, \quad dt = (b-a)dx,$$

integrala capătă forma

$$I = (b-a)\int_0^1 f(x)dx = (b-a)\theta,\ f(x) = g(a+(b-a)x),$$

şi deci problema se reduce la calculul integralei de forma (6.4).

6.2 Metoda Monte Carlo brută.

Dacă considetăm U variabila aleatoare uniformă $0-1$, atunci θ se poate scrie

$$\theta = E[f(U)] = \int_0^1 f(u)du$$

adică $\tau = f(U)$ este un estimator primar *nedeplasat* al lui θ de unde rezultă că dacă considerăm selecţia $U_1, U_2, ..., U_n$ atunci θ *se estimează cu media aritmetică,* adică cu estimatorul secundar

$$\overline{\theta}_n = \frac{1}{n}\sum_{i=1}^{n} f(U_i). \tag{6.5}$$

Metoda aceasta bazată pe formula (6.5) se numeşte *metoda Monte Carlo brută* (denumirea derivând din faptul că se foloseşte repartiţia uniformă).

Exemplul 6.1 Calculul lui π. Să presupunem că se dă cercul de rază 1 cu centrul in origine $x^2 + y^2 = 1$, fie D discul mărginit de acest cerc şi să considerăm $\chi_D(x, y)$ funcţia indicator a acestui domeniu D. Atunci

$$a = aria(D) = \int\int_D dxdy = \pi = \int\int \chi_D(x, y)dxdy.$$

Să considerăm acum intervalul bidimensional (pătratul) $I^2 = [-1, 1]\times[-1, 1]$ şi vectorul $V = (V_1, V_2)'$ uniform pe I^2. Atunci

$$p = \frac{aria(D)}{aria(I^2)} = \frac{\pi}{4}$$

se estimează cu

$$\overline{p} = \frac{1}{n}\sum_{i=1}^{n} \chi_D(V_i) = \overline{\chi}_{D,n}$$

unde $V_1, V_2, ..., V_n$ este o selecţie asupra lui V. Din ultima relaţie rezultă că

$$\pi \approx 4\overline{p} = \frac{4}{n}\sum_{i=1}^{n} \chi_D(V_i) = 4\overline{\chi}_{D,n} \tag{6.5'}$$

formulă care dă aproximarea lui π. Dacă notăm $\sigma^2 = Var(\chi_D(V)) = p(1-p)$, rezultă că

$$Var(\overline{p}) = \frac{1}{n}p(1-p) \leq \frac{1}{n4}. \tag{6.5''}$$

Să revenim acum la analiza estimatorului (6.5).

Dacă se cere estimarea (aproximarea) lui θ cu eroarea ϵ dată şi cu coeficientul de risc δ dat, atunci volumul necesar al selecţiei este dat de (6.3). Pentru a aplica formula (6.3) ar trebui să cunoaştem

$$\sigma^2 = Var[f(U)] = \int_0^1 (f(u) - \theta)^2 du = \int_0^1 f^2(u)du - \theta^2. \qquad (6.6)$$

Suntem deci in faţa unui caz special şi oarecum ciudat: pentru a calcula θ ar trebui sa-l cunoaştem mai intâi pe θ şi să cunoaştem şi integrala $\int_0^1 f^2(x)dx$ (presupunând deci că $f \in \mathcal{L}^2[0,1]$). Dar nu putem abandona ideea. Putem determina (estima) pe n cu (6.3) intr-unul din două moduri şi anume:

(1) Presupunem că $f(x) \leq M < \infty$, $x \in [0,1]$. Atunci avem $\sigma^2 < M^2$ si deci in (6.3) putem inlocui $\sigma^2 = M^2$ obţinând un n mai mare dar care asigură *cu atât mai mult* precizia ϵ şi riscul δ. In particular, in cazul exemplului 6.1 putem determina n_0 din (6.3) luând $\sigma^2 = p(1-p)$.

(2) Alt mod de a estima un n constă in a estima mai intâi pe σ^2 folosind o selecţie de volum k, $U_1, U_2, ..., U_k$ şi anume

$$\sigma^2 \approx s^2 = \frac{1}{k}\sum_{i=1}^{k}(f(U_i) - \overline{\theta}_k)^2$$

iar din (6.3) (luând $\sigma = s$) determinăm n.

Din cele analizate, rezultă că estimaţia $\overline{\theta}_n$ a lui θ are dispersia

$$Var(\overline{\theta}_n) = \frac{1}{n}\sigma^2,\ \sigma^2 = Var(f(U)). \qquad (6.7)$$

Deci precizia metodei depinde in mod esenţial de σ. Dar, din prima integrală din (6.6) rezultă că σ^2 depinde direct de *mărimea variaţiei pe* $[0,1]$ *a lui* $f(x) - \theta$. Pentru a mări deci precizia metodei este necesar să *micşorăm dispersia* $\sigma^2 = Var[f(U)]$. De aceea vom prezenta in continuare *metode de reducere a dispersiei.*

6.3 Metode de reducere a dispersiei

Vom prezenta câteva metode de reducere a dispersiei estimatorului primar in calculul integralelor definite.

6.3.1 Metoda Monte Carlo după importanţă.

Această metodă derivă tocmai din cerinţa de a micşora variaţia lui $f(x) - \theta$ pe $[0,1]$. Pentru aceasta să alegem o *densitate de repartiţie* $p(x)$, $x \in [0,1]$ astfel

incât variaţia funcţiei $(f(x)/p(x)-\theta)^2 p(x)$ să fie mai mică decât variaţia lui $(f(x)-\theta)^2$ pe $[0,1]$. (Acest lucru se obţine dacă de exemplu se alege $p(x)$ astfel incât graficele funcţiilor $f(x)$ şi $p(x)$ să fie aproape paralele!).

Cu această alegere a lui $p(x)$ integrala (6.4) se scrie

$$\theta = \int_0^1 \frac{f(x)}{p(x)} p(x)dx = E[\frac{f(X)}{p(X)}] \tag{6.8}$$

unde $X \rightsquigarrow p(x)$. Deci $\tau^{(i)} = f(X)/p(X)$ este un estimator primar pentru θ. Simulând o selecţie $X_1, X_2, ..., X_n$ a lui X, din (6.8) rezultă că θ se estimează cu estimatorul secundar

$$\overline{\theta}_n^{(i)} = \frac{1}{n}\sum_{i=1}^{n} \frac{f(X_i)}{p(X_i)}.$$

Datorită alegerii densitaăţii $p(x)$ ca mai sus rezultă că

$$Var[f(U)] > Var\left[\frac{f(X)}{p(X)}\right], \rightarrow Var[\overline{\theta}_n] > Var[\overline{\theta}_n^{(i)}]$$

şi deci se realizează reducerea dispersiei. Densitatea $p(x)$ se numeşte *funcţie de importanţă*, de unde şi denumirea de *metoda Monte Carlo după importanţă*.

Să observăm că metoda *după importanţă* se poate aplica cu succes la calculul unor integrale pe interval necompact , chiar integrale multiple.

Exemplul 6.2 Să ne propunem să calculăm integrala

$$I = \int_0^\infty \int_0^\infty e^{-x^{5/2}-y^{3/2}} \sqrt{x^2+y^2+1}dxdy.$$

Să alegem funcţia de importanţă

$$p(x,y) = \begin{cases} e^{-x-y}, & daca \quad x>0, y>0 \\ 0, & altfel. \end{cases}$$

Se observă că vectorul aleator $T = (X,Y)'$ care are densitatea $p(x,y)$, are componentele sale X şi Y repartizate $Exp(1)$ şi independente, ele putându-se simula cu uşurinţă. Se poate deci aplica metoda Monte Carlo după importanţă pentru estimarea integralei I.

Exemplul 6.3. Să considerăm integrala

$$\theta = \int_0^1 \frac{g(x)}{\sqrt{x}}dx$$

unde $g(x)$ este cotinuă pe $[0,1]$. Metoda Monte Carlo brută conduce la estimatorul secundar

$$\overline{\theta}_n = \frac{1}{n}\sum_{i=1}^{n}\frac{g(U_i)}{\sqrt{U_i}}, \quad U_i \rightsquigarrow uniform[0,1].$$

Deoarece $Var(\frac{g(U)}{\sqrt{U}}) = \infty$, rezultă că metoda Monte Carlo brută nu este bună. Dar dacă luăm ca *funcţie de importanţă* densitatea pe $[0,1]$

$$p(x) = \frac{1}{2\sqrt{x}}, \quad F(x) = P(X < x) = \begin{cases} 0, \, x < 0 \\ 1 - \sqrt{x}, \, x \in [0,1] \\ 1, \, x > 1 \end{cases}$$

atunci dispersia estimatorului *după importanţă* este finită şi deci această ultimă metodă este mai bună.

•• **Observaţii.** (1) Să observăm că metoda selecţiei după importanţă se poate aplica la calculul oricărei integrale, simplă sau multiplă de forma

$$\theta = \int_D f(x)p(x)dx = \int f(x)p(x)dx, \quad D \subseteq R_k \tag{6.8'}$$

unde $p(x)$ este densitate.Valoarea integralei θ se aproximează cu estimatorul secundar

$$\overline{\theta}_n = \frac{1}{n}\sum_{i=1}^{n} f(Y_i), \quad Y_i \rightsquigarrow p(x).$$

Să alegem acum o altă densitate $q(x)$, $x \in D$ şi fie $X \rightsquigarrow q(x)$. Scriind

$$\theta = \int \frac{f(x)p(x)}{q(x)}q(x)dz = E\left[\frac{f(X)p(X)}{q(X)}\right]$$

putem determina un estimator nedeplasat al lui θ astfel

$$\hat{\theta}_n = \frac{1}{n}\sum_{i=1}^{n}\frac{f(X_i)p(X_i)}{q(X_i)}, \quad X_i \rightsquigarrow q(x).$$

Se poate arăta că dispersia estimatorului $\hat{\theta}_n$ se micşorează faţă de cea a lui $\overline{\theta}_n$ dacă alegem ca densitate $q(x) = q_0(x) \sim |f(x)|p(x)$, adică satisface această relaţie de proporţionalitate. Utilizarea celor două densităţi $p(x), q_0(x)$ pentru calculul unei integrale, (cu $q_0(x)$ ales ca mai sus) poate deci conduce la micşorarea dispersiei faţă de metoda brută.

(2) Să presupunem că avem de calculat mai multe integrale multiple pe acelaş domeniu de forma (6.8'), adică

$$\theta_i = \int_D f_i(x)p_i(x)dx = \int f_i(x)p(x)dx, \quad 1 \leq i \leq s$$

şi pentru fiecare alegem o aceeaşi densitate *de importanţă* $q(x)$ ca la observaţia (1), precedentă. Obţinem atunci estimatorii secundari

$$\hat{\theta}_n^{(i)} = \frac{1}{n}\sum_{i=1}^{n} f_i(X_i)\frac{p(X_i)}{q(X_i)}, \quad X_i \rightsquigarrow q(x).$$

Presupunem că se dau constantele $a_i, 1 \leq i \leq s$, şi dorim să alegem $q(x)$ astfel incât $Var(\sum_i a_i\hat{\theta}_n^{(i)})$ să fie minimă. Avem

$$Var(\hat{\theta}_n) = Var(\sum_i a_i\hat{\theta}_n^{(i)}) = \int \sum_{i=1}^{s} a_i^2 f_i^2(x)\frac{p^2(x)}{q(x)}dx - \sum_{i=1}^{s} a_i^2\theta_i^2$$

iar minimum acestei dispersii se obţine când este minimă expresia

$$S = \int \sum_{i=1}^{s} a_i^2 f_i^2(x)\frac{p^2(x)}{q(x)}dx.$$

Să considerăm variabila aleatoare

$$\eta = \left(\sum_{i=1}^{s} a_i^2 f_i^2(X)\right)^{\frac{1}{2}} \frac{p(X)}{q(X)}, \; X \rightsquigarrow q(x)$$

şi să observăm că $Var(\eta) = S - \overline{I}^2$, unde

$$\overline{I} = \int \left(\sum_{i=1}^{s} a_i^2 f_i^2(x)\right)^{\frac{1}{2}}$$

de unde

$$S \geq \left(\int \left(\sum_{i=1}^{s} a_i^2 f_i^2(x)\right)^{\frac{1}{2}} p(x)dx\right)^2.$$

In concluzie dispersia $Var(\hat{\theta}_n)$ devine minimă când

$$q(x) = q_0(x) = \frac{1}{\overline{I}}\left(\sum_{i=1}^{s} a_i^2 f_i^2(x)\right)^{\frac{1}{2}} p(x), \quad S = \overline{I}$$

şi acest minim se atinge pentru $q_0(x)$ şi el este $\overline{I}^2 - \sum_{i=1}^{s} a_i^2\theta_i^2$.

6.3.2 Metoda variabilei de control

Să presupunem că pentru a calcula integrala (6.6) cunoaştem o funcţie φ pentru care putem calcula uşor integrala

$$\mu = \int_0^1 \varphi(u)du$$

şi presupunem că φ este selectată astfel incât

$$Var[f(U) - \varphi(U)] < Var[f(U)]. \tag{6.9}$$

Deoarece

$$\theta = \mu + \int_0^1 [f(u) - \varphi(u)]du = \mu + E[f(U) - \varphi(U)] \tag{6.10}$$

rezultă că putem estima θ cu

$$\mu + \overline{\theta}_n^{(c)} = \mu + \frac{1}{n}\sum_{i=1}^{n} \psi(U_i), \quad \psi(u) = f(u) - \varphi(u).$$

Dar din (6.9) avem

$$Var[\mu + \psi(U)] = Var[\psi(U)] < Var[f(U)]$$

şi deci dispersia estimatorului $\mu + \overline{\theta}_n^{(c)}$ este mai mică decât cea a estimatorului brut $\overline{\theta}_n$. Funcţia φ se numeşte *funcţie de control* de unde derivă şi denumirea metodei.

6.3.3 Metoda variabilelor corelate

Să presupunem că alegem o funcţie φ (ca in cazul metodei variabilei de control) astfel incât

$$\theta = \int_0^1 \varphi(u)du + \int_0^1 [f(u) - \varphi(u)]du$$

unde integrala $\mu = \int_0^1 \varphi(u)du$ se poate calcula exact şi uşor. Dacă in plus se alege φ astfel incât

$$Var[\varphi(U)] - 2Cov[\varphi(U), f(U)] < 0, \tag{6.11}$$

atunci, de asemenea se poate realiza reducerea dispersiei deoarece

$$Var[\varphi^*(U)] = Var[f(U) - \varphi(U)] =$$
$$= Var[f(U)] + Var[\varphi(U)] - 2Cov[f(U), \varphi(U)] < Var[f(U)].$$

Variabila aleatoare $\varphi(U)$ care satisface (6.11) se numeşte variabilă *corelată* (cu $f(U)$), de unde şi denumirea metodei.

6.3.4 Metoda variabilelor antitetice

Să considerăm integrala (6.4) care se scrie

$$\theta = E[\tau] = E[f(U)],$$

să alegem un alt estimator primar $f^*(U)$, $\theta = E[f^*(U)]$ şi să considerăm estimatorul primar

$$\varphi = f^*(1-U), \quad E[\varphi(U)] = E[f^*(1-U)] = E[f^*(U)] = \theta.$$

Atunci, putem considera estimatorul primar

$$\psi = \frac{1}{2}(\tau + \varphi)$$

pentru care avem

$$Var(\psi) = \frac{1}{4}\{Var(\tau) + Var(\varphi) + 2Cov(\tau, \varphi)\} \, si \, Cov(\tau, \varphi) < 0.$$

Dacă se alege f^* astfel incât

$$Var(\psi) < Var(\tau) \tag{6.12}$$

atunci estimatorul ψ realizează reducerea dispersiei. Uneori puntem alege simplu $\varphi = f(1-U)$ care satisface (6.12).

6.3.5 Metoda selecţiei stratificate

Această metodă derivă tot din ideea de a micşora dispersia estimatorului primar prin reducerea variaţiei funcţiei ce se integrează. Fie $0 = \alpha_0 < \alpha_1 < ... < \alpha_m = 1$ o diviziune a intervalului $[0, 1]$. Atunci integrala (6.4) se scrie

$$\theta = \sum_{i=1}^{m} \int_{\alpha_{j-1}}^{\alpha_j} f(v)dv$$

de unde se deduce că fiecare din cele m integrale ale sumei precedente se poate estima prin metoda Monte Carlo brută, obţinându-se in final estimatorul secundar

$$\overline{\tau}_m^* = \sum_{j=1}^{m} \sum_{i=1}^{k_j} (\alpha_j - \alpha_{j-1}) \frac{1}{k_j} f(\alpha_{j-1} + (\alpha_j - \alpha_{j-1})U_{i_j}) \tag{6.13}$$

unde U_{i_j}, $1 \leq j \leq m$, $1 \leq i \leq k_j$ sunt selecţii de numere aleatoare independente şi uniforme $0-1$ de volume k_j, iar $n = \sum_{i=1}^{m} k_j$. Cu selecţiile de volume k_j de numere aleatoare uniforme $0-1$ se estimează integralele pe intervalele $[\alpha_{j-1}, \alpha_j)$. Cele m intervale se mai numesc (in limbaj statistic) *straturi* iar selecţia formată din cele m selecţii ale straturilor se numeşte *selecţie stratificată.* Făcând schimbări de variabile in integralele (6.13) se deduce că

$$Var(\overline{\tau}_m^*) = \sum_{j=1}^{m} \frac{(\alpha_j - \alpha_{j-1})^2}{k_j} \int_{\alpha_{j-1}}^{\alpha_j} f^2(u)du - \sum_{j=1}^{m} \frac{1}{k_j} \left[\int_{\alpha_{j-1}}^{\alpha_j} f(u)du \right]^2. \quad (6.14)$$

Notând

$$\theta_j = \int_{\alpha_{j-1}}^{\alpha_j} f(u)du,\ 1 \leq j \leq m$$

dispersia lui $\overline{\tau}_m^*$ se micşorează dacă variaţiile funcţiilor $f(u) - \theta_j$ pe intervalele $[\alpha_{j-1}, \alpha_j)$ sunt mult mai mici decât variaţia funcţiei $f(u) - \theta$, $u \in [0,1]$. Din teoria statistică a sondajelor stratificate se ştie că dacă se fixează volumul total al selecţiei n, atunci volumele de selecţie ale straturilor k_j se aleg proporţionale cu valorile

$$(\alpha_j - \alpha_{j-1}) \int_{\alpha_{j-1}}^{\alpha_j} f^2(u)du - \left[\int_{\alpha_{j-1}}^{\alpha_j} f(u)du \right]^2. \quad (6.15)$$

Punctele diviziunii α_j se pot alege astfel incât lungimile intervalelor să fie egale cu $\frac{1}{m}$ (adică $\alpha_j = \frac{j}{m}$) sau se aleg astfel incât variaţia funţiei să fie aceeaşi pe fiecare interval.

Dispersia $Var(\overline{\tau}_m^*)$ se calculează deobicei greu, dar ea se poate estima astfel

$$Var(\overline{\tau}_m^*) \approx s_m^2 = \sum_{j=1}^{m} \frac{(\alpha_j - \alpha_{j-1})^2}{k_j(k_j - 1)} \sum_{i=1}^{k_j} (f_{i_j} - \overline{f}_j)^2 \quad (6.16)$$

unde

$$f_{i_j} = f(\alpha_{j-1} + (\alpha_j - \alpha_{j-1})U_{i_j}),\ \overline{f}_j = \frac{1}{k_j} \sum_{i=1}^{k_j} f_{i_j}. \quad (6.16')$$

6.3.6 Reducerea dimensiunii de integrare

Să presupunem că avem de calculat integrala multiplă

$$\theta = \int\limits_D f(\mathbf{x})p(\mathbf{x})d\mathbf{x}, \quad D \in R^k \tag{6.4'}$$

unde $p(\mathbf{x})$ este densitate de repartiţie. (Oricând, o integrală multiplă se poate scrie aşa). Să notăm $\mathbf{x} = (\mathbf{y}, \mathbf{z})'$, $\mathbf{y} = (x_1, x_2, ..., x_s)'$, $s < k, \mathbf{z} = (x_{s+1}, ..., x_k)'$. Atunci, integrala (6.4') se scrie, folosind notaţii convenabile

$$\theta = \int\limits_{(Y)} \int\limits_{(Z)} f(\mathbf{y}, \mathbf{z})p(\mathbf{y}, \mathbf{z})d\mathbf{y}d\mathbf{z} = \int\limits_{(Y)} \tilde{f}(\mathbf{y})p_1(\mathbf{y})d\mathbf{y}$$

$$p_1(\mathbf{y}) = \int\limits_{(Z)} p(\mathbf{y}, \mathbf{z})d\mathbf{z}, \quad \tilde{f}(\mathbf{y}) = \int\limits_{(Z)} f(\mathbf{y}, \mathbf{z})p(\mathbf{y}, \mathbf{z})\frac{d\mathbf{z}}{p_1(\mathbf{y})}$$

unde $\frac{p(\mathbf{y},\mathbf{z})}{p_1(\mathbf{y})}$ este densitatea condiţionată, iar $p_1(\mathbf{y})$ este densitate marginală.

Să notăm $\tilde{\mathbf{Y}}$ vectorul aleator care are densitatea $p_1(\mathbf{y})$ şi cu $\mathbf{X} = (\mathbf{Y}, \mathbf{Z})'$, $\tilde{\mathbf{Y}} = \mathbf{Y}$ vectorul aleator corespunzător densităţii $p(\mathbf{x})$. Atunci putem estima integrala θ in două moduri şi anume

$$\overline{\theta}_n = \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{Y}_i, \mathbf{Z}_i), \quad \tilde{\theta}_n = \frac{1}{n}\sum_{i=1}^{n} \tilde{f}(\mathbf{Y}_i). \tag{6.4'}$$

Se poate arăta că are loc relaţia

$$Var(\tilde{\theta}_n) \leq Var(\overline{\theta}_n).$$

Intr-adevăr, scriind in detaliu dispersiile avem

$$Var(\overline{\theta}_n) - Var(\tilde{\theta}_n) =$$

$$= \frac{1}{n}\int\limits_{(Y)} \left\{ \int\limits_{(Z)} f^2(\mathbf{y}, \mathbf{z})\frac{p(\mathbf{y}, \mathbf{z})}{p_1(\mathbf{y})}d\mathbf{z} - \left(\int\limits_{(Z)} f(\mathbf{y}, \mathbf{z})\frac{p(\mathbf{y}, \mathbf{z})}{p_1(\mathbf{y})}d\mathbf{z} \right)^2 \right\} \int\limits_{(Z)} p(\mathbf{y}, \mathbf{z})d\mathbf{y} \geq 0.$$

Ultima inegalitate rezultă din faptul că expresia din acolade este dispersia lui $\tilde{\mathbf{Y}}$ (care este pozitivă) iar $p(\mathbf{x}, \mathbf{y}) \geq 0$ fiind densitate de repartiţie.

6.4 Rezolvarea unor ecuaţii operatoriale

In această secţiune ne vom ocupa de rezolvarea sistemelor de ecuţii liniare şi de rezolvarea unor ecuaţii integrale.

6.4.1 Rezolvarea sistemelor de ecuaţii liniare

Să presupunem că sistemul de ecuaţii liniare este scris sub forma matricială

$$\mathbf{x} = \mathbf{a} + H\mathbf{x},\ \mathbf{x} \in R^n,\ \mathbf{a} \in R^n,\ H = (h_{ij}), ||H|| = \max_i(\sum_j |h_{ij}|) < 1. \quad (6.17)$$

(Dacă sistemul este dat sub forma $A\mathbf{x} = \mathbf{a}$ cu $\mathbf{x}, \mathbf{a} \in R^n$, atunci el se scrie sub forma (6.17) dacă $H = (I - A)$, $I-$matricea unitate $n \times n$). Pentru construirea metodei Monte Carlo de rezolvare a sistemului (6.17) asociem problemei un lanţ Markov finit, absorbant şi omegen cu $n + 1$ stări astfel:

- la matricea $H^{n\times n}$ asociem matricea de probabilităţi $P^{n\times n} = (p_{ij})$ cu condiţia

$$p_{ij} \geq 0, \sum_{j=1}^{n} p_{ij} < 1;\ p_{ij} \neq 0 \Leftrightarrow h_{ij} \neq 0 \quad (6.18)$$

şi fie $p_i = 1 - \sum\limits_{j=1}^{n} p_{ij}$. Lanţul Markov omogen şi absorbant va avea matricea probabilităţilor de trecere

$$P^* = \begin{pmatrix} p_{11} & p_{12} & ... & p_{1n} & p_1 \\ p_{21} & p_{22} & ... & p_{2n} & p_2 \\ . & . & ... & . & \\ p_{n1} & p_{n2} & ... & p_{nn} & p_n \\ 0 & 0 & ... & 0 & 1 \end{pmatrix}; \quad (6.18')$$

- in continuare definim *ponderile*

$$v_{ij} = \begin{cases} \frac{h_{ij}}{p_{ij}} & daca\ p_{ij} > 0 \\ 0 & altfel \end{cases}; \quad (6.19)$$

- pentru un i_0, $1 \leq i_0 \leq n$ generăm o traiectorie a lanţului Markov

$$\gamma = (i_0, i_1, ..., i_m, i_{m+1}), \quad i_{m+1} = n + 1$$

adică i_{m+1} este stare absorbantă; din construcţia lanţului rezultă că traiectoria este finită ($m < \infty$ iar $p_{i_m,i_{m+1}} = p_{i_m}$);

- să considerăm acum expresia

$$V_m(\gamma) = v_{i_0 i_1} v_{i_1 i_2} ... v_{i_{m-1} i_m}, \quad X(\gamma) = V_M(\gamma)\frac{a_{i_m}}{p_{i_m}}. \quad (6.20)$$

Pentru statistica $X(\gamma)$ este adevărată teorema

Teorema 6. 2 *Statistica $X(\gamma)$ este un estimator primar pentru componenta x_{i_0} a soluţiei* $\mathbf{x}$ *a sistemului liniar (6.17).*

Demonstraţie. Pentru traiectoria γ dată avem

$$E[X(\gamma)|i_0 = i] = \sum_{\gamma} P(\gamma)X(\gamma) =$$

$$= \sum_{m=0}^{\infty} \sum_{i_1 i_2 ... i_m} p_{ii_1} p_{i_1 i_2} ... p_{i_{m-1} i_m} p_{i_m} v_{ii_1} v_{i_1 i_2} ... v_{i_{m-1} i_m} \frac{a_{i_m}}{p_{i_m}} =$$

$$= \sum_{m=0}^{\infty} \sum_{i_1 i_2 ... i_m} h_{ii_1} h_{i_1 i_2} ... h_{i_{m-1} i_m} a_{i_m} =$$

$$= a_i + (Ha)_i + (H^2 a)_i + ... \qquad (6.20')$$

Ultima formulă reprezintă componenta $x_{i_0} = x_i$ a dezvoltării

$$x = a + Ha + H^2 a + ... \qquad (6.21)$$

adică a soluţiei sistemului (6.17) şi teorema este demonstrată. Deoarece $||H|| < 1$ rezultă că (6.21) deci şi (6.20') converge.

Din cele de mai sus, rezultă că dacă considerăm o selecţie $\gamma_1, \gamma_2, ..., \gamma_N$ de traiectorii independente ale lanţului Markov construit, pornind toate din starea $i_0 = i$ obţinem estimatorul secundar al componentei x_i a sistemului de forma

$$\overline{x}_i = \frac{1}{N} \sum_{i=1}^{N} X(\gamma_i)$$

care este estimator nedeplasat şi consistent pentru x_i. Observăm că metoda Monte Carlo ne permite să determinăm *o singură componentă a soluţiei,* (componenta x_i, $i = i_0$), ceea ce nu putem realiza cu nicio metodă numerică clasică. Acest fapt poate să prezinte un avantaj in cazul unor probleme inginereşti, când de exemplu x_{i_0} reprezintă o componentă importantă a unei structuri de rezistenţă şi numai aceasta este necesar sa fie bine calculată, celelalte putând avea acelaş ordin de mărime.

6.4.2 Rezolvarea ecuaţiilor integrale

Să considerăm ecuaţia integrală *de speţa doua* in funcţia necunoscută $f(x)$ de forma

$$f(x) = g(x) + \int_a^b K(x, y) f(y) dy \qquad (6.22)$$

unde nucleul K şi funcţiile f, g satisfac condiţii astfel incât soluţia $f(x)$ să existe pentru orice $x \in [a, b]$. Considerând K drept un operator liniar ecuaţia (6.22) se poate scrie asemănător lui (6.17) adică

$$f = g + Kf \tag{6.17'}$$

care este *tot o ecuaţie operatorială.* Considerând *normele*

$$||g|| = \int_a^b |g(u)|du, \quad ||K|| = \sup_x \int_a^b |K(x', x)|dx$$

şi iterând nucleul K conform regulei

$$[K^2 f](x) = \int_a^b \int_a^b f(x_0)K(x_0, x_1)K(x_1, x)dx_0 dx_1$$

se arată că dacă $||K|| < 1$ atunci există un n_0 finit astfel incât

$$f = \sum_{i=1}^{n_0} K^i f + g$$

adică in acest caz există soluţie a ecuaţiei integrale.

Un mod direct de rezolvare numerică a ecuaţiei (6.22) constă in a *discretiza ecuaţia.* Adică, alegem o *grilă* de puncte $a = \alpha_0 < \alpha_1 < ... < \alpha_n = b$ suficient de *fină* astfel incât ne interesează să calculăm valorile $f_i = f(\alpha_i)$, $1 \leq i \leq n$. Atunci, aproximând integralele prin sume corespunzând fiecărui punct α_i ajungem la un sistem liniar de forma

$$\mathbf{f} = \mathbf{g} + K\mathbf{f} \tag{6.22'}$$

unde $\mathbf{f} = (f_1, f_2, ..., f_n)'$, $\mathbf{g} = (g_1, g_2, ..., g_n)'$, $g_i = g(\alpha_i), K = (K_{ij})$, $K_{ij} = K(\alpha_i, \alpha_j)$, $1 \leq i, j \leq n$. Acest sistem se rezolvă ca un sistem liniar ca mai sus. Şi aici remarcăm avantajul metodei in a determina soluţia numai a unei componente f_{i_0} a lui $\mathbf{f}$.

Putem insă să construim o metodă Monte Carlo asemănătoare celei privind rezolvarea sistemelor de ecuaţii liniare, folosind un *lanţ Markov cu o mulţime continuă de stări şi absorbant.* Vom presupune că dorim să estimăm o *funţională* I_h (definită de produsul scalar) ce depinde de soluţia f a ecuaţiei integrale (6.22) sau (6.17') dată de relaţia

$$I_h = (f, h) = \int_R f(x)h(x)dx, \; f = Kf + g. \tag{6.23}$$

Lanţul Marcov se presupune că este determinat de o *densitate de repartiţie iniţială* a stărilor $\pi(x)$ şi de o *densitate de tranziţie* $p(x', x)$; vom nota cu N numărul aleator de *stări finale* ale lanţului. (Lanţul este absorbant!). Dacă $x_0, x_1, ..., x_N$ este o traiectorie a lanţului să introducem *ponderile*

$$Q_0 = \frac{g(x_0)}{\pi(x_0)}, \quad Q_n = Q_{n-1}\frac{K(x_{n-1}, x_n)}{p(x_{n-1}, x_n)}. \tag{6.24}$$

Date fiind g, K, h, vrem să alegem densităţile $\pi(x), p(x', x)$ astfel incât estimatorul

$$\xi = \sum_{n=0}^{N} Q_n h(x_n) \tag{6.24'}$$

să fie un estimator primar pentru funcţionala (f, h). Procedăm astfel:

- alegem funcţiile $\pi(x)$, $r(x', x)$ cu condiţiile

$$\pi(x) \neq 0 \Leftrightarrow g(x) \neq 0, \quad \int_a^b \pi(x)dx = 1,$$

$$r(x', x) \neq 0 \Leftrightarrow K(x', x) \neq 0$$

şi construim densitatea de tranziţie a lanţului absorbant de forma

$$p(x', x) = r(x', x)[1 - q(x')], \int_a^b p(x', x)dx = 1 - q(x') \leq 1$$

unde $q(x')$ este probabilitatea ca traiectoria lanţului să se intrerupă (lanţul absorbant!). Cu precizările anterioare se poate demonstra teorema următoare, asemănătoare teoremei 6.2.

Teorema 6. 3 *Dacă nucleul K satisface condiţia că pentru un n_0 finit dat are loc condiţia $||\hat{K}^{n_0}|| < 1$ unde $\hat{K}(x', x) = |K(x', x)|$ şi dacă densităţile π şi p sunt selectate ca mai sus, atunci*

$$E[\xi] = E\left[\sum_{n=0}^{N} Q_n h(x_n)\right] = I_h = (f, h).$$

In plus, lungimea traiectoriei este finită, $E(N) < \infty$.

Demonstraţia teoremei, cu excepţia relaţiei $E(N) < \infty$ este intru totul asemănătoare teoremei precedente 6.2 şi nu o prezentăm aici.

Pentru a estima funcţia $f(x)$ in punctul x este suficient să observăm că

$$f(x) = (f, h_x) + g(x), \quad h_x(x') = K(x', x).$$

Dacă $g(x) \geq 0$ şi

$$\int_a^b g(x)dx = 1,\ K(x',x) \geq 0,\ \int_a^b K(x,x')dx' \leq 1,$$

atunci putem lua

$$\pi(x) = g(x),\ p(x',x) = K(x',x),\ Q_n = 1,\ \xi = \sum_{n=0}^{N} h(x_n)$$

unde N este lungimea traiectoriei până la absorbţie.

6.5 Rezolvarea ecuaţiilor cu derivate parţiale

Să considerăm ecuaţia cu derivate parţiale de ordinul doi

$$\beta_{11}\frac{\partial^2 V}{\partial x^2} + 2\beta_{12}\frac{\partial^2 V}{\partial x \partial y} + \beta_{22}\frac{\partial^2 V}{\partial y^2} + 2\alpha_1\frac{\partial V}{\partial x} + 2\alpha_2\frac{\partial V}{\partial y} = F(x,y) \qquad (6.25)$$

cu condiţia

$$V(x,y) = \varphi(x,y), \quad (x,y) \in C, \qquad (6.25')$$

unde $V(x,y)$ este o funcţie necunoscută pe domeniul mărginit $\mathcal{D} \subset R^2$, coeficienţii β_{ij}, α_i, $i,j = 1,2$ sunt funcţii reale cunoscute definite pe $\mathcal{D}$, funcţia $\varphi(x,y)$ este de asemenea cunoscută, iar $C = \partial\mathcal{D}$ este frontiera lui $\mathcal{D}$. Problema rezolvării ecuaţiei (6.25) cu condiţia *pe contur (6.25')* este cunoscută sub numele de *problema Dirichlet.*

Etapele rezolvării prin metoda Monte Carlo a problemei Dirichlet formulată anterior sunt următoarele:

1) Domeniul D este acoperit de o *reţea dreptunghiulară* de puncte (*sau grilă*), puncte in care se vor calcula valorile funţiei V. Să notăm cu (x_i, y_j) nodurile reţelei definite astfel

$$x_i = x_0 + ih, \quad y_i = y_0 + ih, \quad i = 0, \pm 1, \pm 2, \pm 3, ... \qquad (6,26)$$

unde (x_0, y_0) sunt fixate iar h este o constantă suficient de mică (*pasul grilei*). Orice punct (x,y) al reţelei are partu puncte *vecine*

$$P_1 = (x+h, y),\ P_2 = (x-h, y),\ P_3 = (x, y+h),\ P_4 = (x, y-h) \qquad (6.26')$$

Considerăm de asemenea punctul vecin suplimentar $P_5 = (x+h, y+h)$. Unele puncte ale reţelei sunt pe curba C sau sunt *cele mai apropiate* de curba C; să notăm C_h mulţimea acestor puncte.

2) Discretizăm ecuaţia (6.26) in punctul $P = (x, y)$ in care dorim să estimăm $V(x, y)$. Discretizarea conduce la construirea unor *diferenţe finite* $\Delta_x, \Delta_y, \Delta_{xx}, \Delta_{xy}, \Delta_{yy}$, definite astfel:

$$\Delta_x = \frac{V(x+h,y) - V(x,y)}{h},\ \Delta_y = \frac{V(x,y+h) - V(x,y)}{h}, \Delta_{xx} = \Delta(\Delta_x), etc.$$

$$\Delta_{xx} = \frac{V(x+h,y) + V(x-h,y) - 2V(x,y)}{h^2}, \tag{6.26''}$$

$$\Delta_{xy} = \frac{V(x+h,y+h) - V(x+h,y) - V(x,y+h) + V(x,y)}{h^2}$$

$$\Delta_{yy} = \frac{V(x,y+h) + V(x,y-h) - 2V(x,y)}{h^2}.$$

Folosind aceste diferenţe in (6.25) deducem expresia lui $V(x, y) = V(P)$ astfel

$$V(P) = \sum_{i=1}^{5} p_i(P) V(P_i) - \frac{h^2 F(P)}{D(P)} \tag{6.27}$$

unde

$$p_1(P) = \frac{1}{D}(\beta_{11} - 2\beta_{12}),\ p_2(P) = \frac{\beta_{11}}{D},$$

$$p_3(P) = \frac{1}{D}(\beta_{22} - 2\beta_{12}),\ p_4(P) = \frac{\beta_{22}}{D} \tag{6.27'}$$

$$p_5(P) = \frac{2\beta_{12}}{D}, \quad D(P) = 2\beta_{11} + 2\beta_{22} - 2\beta_{12} + 2h(\alpha_1 + \alpha_2).$$

Se observă că dacă ecuaţia (6.25) este de *tip eliptic* atunci $\beta_{11}\beta_{22} - \beta_{12}^2 > 0$ şi deci $p_i(P)$ din (6.27") sunt probabilităţi (sunt pozitive) şi

$$\sum_{i=1}^{5} p_i(P) = 1 \quad \forall\, \beta_{ij}, \alpha_i\, i = 1, 2.$$

Se arată că dacă in (6.27) luăm $h \to 0$ atunci această expresie converge către $V(x, y)$ care este soluţie a ecuaţiei (6.25). Probabilităţile $p_i(P)$, $1 \le i \le 5$ definesc un lanţ Markov particular numit *mers la intâmplare in plan.* Interpretarea acestui lanţ este următoarea: dacă o particolă porneşte din punctul $P(x, y)$ atunci ea se deplasează pe grilă in punctele vecine prin salturi, cu probabilităţile $p_i(P)$. Dacă $\beta_{12} = 0$ atunci particola nu se poate deplasa in punctul $P_5 = (x+h, y+h)$. Mersul la intâmplare construit este *absorbant pe frontiera C_h;* dacă particola atinge un punct $Q \in C_h$ atunci probabilitatea de a părăsi acel punct este zero.

3) Cu ajutorul mersului la intâmplare construit definim estimatorul primar asociat unei traiectorii

$$\gamma = (P, P^{(1)}, P^{(2)}, ..., P^{(m)}, P^{(m+1)}), \quad P^{(m+1)} = Q_i \in C_h$$

astfel

$$Z_i = -\sum_{j=1}^{m} \frac{h^2 F(P^{(j)})}{D(P^{(j)})} + \varphi(Q_i). \tag{6.28}$$

Cu notaţiile şi construcţiile de mai sus este adevărată teorema

Teorema 6. 4 *Estimatorul primar (6.28) este un estimator asimptotic nedeplasat pentru soluţia $V(P) = V(x, y)$ a ecuaţiei (6.25).*

Demonstraţie. Fără a restrânge generalitatea vom presupune că $F(P) > 0$, $\varphi(Q_i) > 0$. Să notăm $W_m(P) = E[Z^*_{mi}]$ cu

$$Z^*_{mi} = -\sum_{j=1}^{m} \frac{h^2 F(P^{(j)})}{D(P^{(j)})} + \varphi_m(Q_i), \tag{6.28$'$}$$

unde

$$\varphi_m(Q_i) = \begin{cases} \varphi(Q_i), daca\ frontiera\, C_h\ este\ atinsa\ in\ m\ pasi \\ 0,\ in\ rest \end{cases}$$

Ţinând seama de (6.28) atunci deducem că $W_m(P)$ verifică relaţia de recurenţă

$$W_M(P) = \sum_{i=1}^{5} p_i(P) W_{m-1}(P_i) - \frac{h^2}{D(P)} F(P) \tag{6.28$''$}$$

şi de aici se deduce $W_{m+1} \geq W_m$ adică W_m este monoton crescător şi mărginit (conform (6.28)) de

$$\varphi_{max} + \left[-\frac{h^2}{D} f \right]_{max} \overline{k}_m,\ cu\ \varphi_{max} = \max_{Q \in C_h} \varphi(Q).$$

Numărul $\overline{k}_m$ este numărul mediu de paşi ai unei traiectorii, care oricum are cel mult un număr finit de paşi dacă domeniul inconjurat de curba C este mărginit. Deci, pentru un punct P fixat, şirul W_m este convergent, de unde rezultă că estimatorul (6.28) este asimptotic nedeplasat, deci teorema este demonstrată.

4) Un estimator secundar al lui $V(P)$ se obţine dacă se consideră n traiectorii $\gamma_1, \gamma_2, ..., \gamma_n$ ce pornesc toate din punctul $P = (x, y)$ astfel

$$\overline{V} = \frac{1}{n} \sum_{i=1}^{n} Z_i. \qquad (6.28''')$$

Acest estimator este de asemenea asimptotic nedeplasat pentru $V(x, y)$.

Un caz particular de ecuaţie de tip eliptic este *ecuaţia Poisson*, adică

$$\bigtriangledown V(P) = F(P), \; sau \quad \frac{\partial^2 V}{\partial x^2} + \frac{\partial v}{\partial y^2} = F(x, y)$$

cu condiţia pe contur

$$V(P) = \varphi(P), \quad P \in C, \quad C = \mathcal{D} \subset R^2.$$

Aici construcţia este asemănătoare, dar desigur mai simplă. Considerăm ca si in cazul general o grilă de puncte care acoperă $\mathcal{D}$ dar ca puncte vecine lui $P = (x, y)$ considerăm numai punctele P_1, P_2, P_3, P_4, (deoarece $\beta_{12} = 0$). In acest caz prin discretizare obţinem $V(P)$ de forma

$$V(P) = \frac{1}{4} \sum_{i=1}^{4} V(P_i) - \frac{1}{4} h^2 F(P), \quad p_i(P) = p_i = \frac{1}{4}, \quad D = 4. \qquad (6.27')$$

Deci mersul la intâmplare asociat are in acest caz probabilităţi egale de tranziţie din P in punctele vecine $P_i, 1 \leq i \leq 4$. Teorema precedentă este desigur valabilă şi in acest caz şi anume estimatorul primar este

$$Z_i = -\sum_{j=1}^{m} \frac{1}{4} h^2 F(P^{(j)}) + \varphi(Q_i)$$

el fiind calculat cu ajutorul traiectoriei γ ca mai sus, iar estimatorul secundar este tot de forma (6.28"').

In final precizăm un algoritm pentru simularea mersului la intâmplare in cazul general al ecuaţiei (6.25). De fapt este necesar să precizăm cum generăm punctul $P^{(j+1)}$ al traiectoriei când plecăm din punctul $P^{(j)}$. Algoritmul pentru generarea acestui punct este

Algoritmul SALT pentru generarea unui salt de mers la intâmplare.
Intrare $x^{(j)}, y^{(j)}$ şi h; (Acesta este un pas pregătitor);
Calculează $p_i = p_i(P^{(j)}), 1 \leq i \leq 5$ *cu formulele (6.27');*

Calculează $f_i = p_1 + ... + p_i$, $1 \leq i \leq 5, f_5 = 1$;
Generează $U := random$; $i := 1$;
while $U > f_i$ **do** $i := i + 1$;
Calculează coordonatele punctului P_i *cu formulele (6.26') in care* $x = x^{(j)}, y = y^{(j)}$;
Ia $P^{(j+1)} := P_i$ *calculat anterior;*
Dacă $P^{(j+1)} \notin C_h$ *atunci ia* $P^{(j)} := P^{(j+1)}$, *altfel traiectoria se termină.*

Repetând algoritmul precedent de la pasul

$$Genereaza \quad U := random;$$

până când pentru prima dată $P^{(j+1)} \in C_h$ se obţine traiectoria mersului până la absorbţie. In final să observăm că metodele numerice obişnuite pentru rezolvarea problemei Dirichlet se bazează tot pe discretizare. Valorile necunoscute ale funţiei $V(P)$ in puncte ale grilei sunt necunoscute ale unui sistem de ecuaţii liniare de dimensiune foarte mare. Prin rezolvarea acestui sistem liniar se obţin *toate* valorile necunoscute in punctele grilei interioare lui $\mathcal{D}$. Metoda Monte Carlo descrisă aici permite calculul *separat* al valorii funcţiei V in fiecare punct P al grilei, fără a mai fi necesar calculul acesteia in alte puncte. Acest fapt, de asemenea poate constitui un avantaj practic.

Exerciţii

E6.1 Să se calculeze mai intâi direct, iar apoi utilizând metoda Monte Carlo brută integrala

$$I = \int_0^1 \sqrt{1 - x^2} dx.$$

Indicaţie. Facând o schimbare potrivită de variabilă se arată că $I = \frac{\pi}{4}$. Acelaş fapt se constată dacă observăm că I este aria din primul cadran a discului cu centrul in origine şi de rază 1. Cu ajutorul metodei Monte Carlo se obţine

$$\frac{\pi}{4} \approx \overline{I}_n = \frac{1}{n} \sum_{i=1}^{n} \sqrt{1 - U_i^2}$$

unde $U_i, 1 \leq i \leq n$ sunt numere aleatoare uniforme $0 - 1$. Se observă că $\pi \approx 4\overline{I}_n$, ceea ce reprezintă un alt mod de a aproxima constanta π.

E6.2 Să se aplice *metoda variabilelor antitetice* la calculul integralei I din exerciţiul precedent şi dacă $\overline{I}_{a,n}$ este estimaţia lui I in acest caz să se arate că $Var(\overline{I}_{a,n}) < Var(\overline{I}_n)$.

Soluţie. estimatorul Monte Carlo antitetic este

$$\overline{I}_{a,n} = \frac{1}{2n}\sum_{i=1}^{n}\left[\sqrt{1-U^2}+\sqrt{1-(1-U)^2}\right]$$

de unde

$$Var(\overline{I}_{a,n}) = \frac{1}{n}[\frac{1}{4}\{Var(\sqrt{1-U^2}) + Var\sqrt{1-(1-U^2)})+$$

$$+2Cov((\sqrt{1-U^2}), \sqrt{1-(1-U^2)}\}] =$$

$$= \frac{1}{2n}\left\{Var(\sqrt{1-U^2}+Cov(\sqrt{1-U^2}, \sqrt{1-(1-U^2)})\right\} =$$

$$= \frac{1}{2n}\left(\frac{2}{3}-\frac{\pi^2}{16}\right) + \frac{1}{2n}E[\sqrt{(U+1)U(U-1)(U-2)}-\frac{\pi^2}{16}],$$

Se poate arăta prin calcule relativ simple că media expresiei in U din paranteza dreaptă precedentă este $E_U \approx 0.5806$ de unde rezultă că

$$Var(\overline{I}_{a,n}) \approx \frac{0.0052}{n}.$$

In mod direct se poate arăta că

$$Var(\overline{I}_n) = \frac{1}{n}\left(\int_0^1 (1-x^2)dx - \frac{\pi^2}{16}\right) = \frac{1}{n}\left(\frac{2}{3}-\frac{\pi^2}{16}\right) \approx \frac{0.498}{n}$$

de unde rezultă inegalitatea cerută.

E6.3 O problemă interesantă in aplicaţii este calculul funcţiei de repartiţie normală $N(0,1)$ adică

$$L(x) = \int_{-\infty}^{x}\frac{e^{-\frac{y^2}{2}}}{\sqrt{2\pi}}dx = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x}\phi(y)dy.$$

Să se construiască un estimator *după importanţă* pentru $L(x)$ utilizând ca densitate de importanţă o densitate logistică trunchiată.
Soluţie. Să considerăm densitatea logistică

$$f(y) = \frac{\pi e^{-\frac{\pi y}{\sqrt{3}}}}{\sqrt{3}(1+e^{-\frac{\pi y}{\sqrt{3}}})^2},\ y \in (-\infty, +\infty).$$

Trebuie insă să considerăm densitatea logistica (aici non-standard!) trunchiată pe $(-\infty, x)$. Va trebui deci să o normăm cu o constantă $k = k(x)$ in raport cu y (care insă depinde de x), adică să luăm ca densitate de importanţă

$$\tilde{f}(y) = f(y)k^{-1} = \frac{\pi e^{-\frac{\pi y}{\sqrt{3}}}(1 + e^{-\frac{\pi x}{\sqrt{3}}})}{\sqrt{3}(1 + e^{-\frac{\pi y}{\sqrt{3}}})^2},\ y \in (-\infty, x);\ \tilde{f}(y) = 0,\ y < x.$$

In acest caz avem estimatorul primar de importanţă

$$L(x) = \frac{1}{1 + e^{-\frac{\pi x}{\sqrt{3}}}} E\left[\frac{\phi(Y)}{f(Y)}\right], \quad Y \rightsquigarrow \tilde{f}(y).$$

Pentru a construi estimatorul secundar trebuie să găsim un generator pentru Y. Se poate aplica metoda inversă pentru funcţia de repartiţie

$$\tilde{F}(z) = \int_{-\infty}^{z} \tilde{f}(y)dy = \frac{1 + e^{-\frac{\pi x}{\sqrt{3}}}}{1 + e^{-\frac{\pi z}{\sqrt{3}}}},\ z < x$$

adică Y se obţine rezolvând ecuaţia

$$\tilde{F}(Y) = U, \quad U\, uniform\, 0 - 1.$$

E6.4 (Problema lui Buffon).Un ac de lungime l este aruncat pe o duşumea de parchet care are lăţimea lamelelor edală cu d. Lamelele vecine determină deci linii paralele la distanţa d. Să se determine probabilitatea ca acul, după ce a căzut pe parchet să atingă o linie despărţitoare a lamelelor parchetului şi de aici să se deducă un procedeu de estimare a numărului π.
Soluţie. Fie $l \leq d$. Pentru o poziţie dată a acului să notăm cu θ unghiul format de ac cu marginea lamelei. Distanţa de la ac la linie este $l\cos\theta$ iar probabilitatea ca in poziţia dată acul să atinga marginea lamelei este $\frac{2l\cos\theta}{2\pi d}$, iar cum θ poate varia intre $(-\pi/2, +\pi/2)$ rezultă că probabilitatea P ca acul să atingă marginea (indiferent de poziţia sa) este

$$P = 2\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \frac{l\cos\theta}{2\pi d}d\theta = \frac{2l}{\pi d}.$$

Dacă $l > d$ atunci printr-un raţionament asemănător avem

$$P = \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \frac{min(l\cos\theta, d)}{2\pi d}d\theta = \frac{2}{\pi d}\left\{d\cos^{-1}\frac{d}{l} + l(1 - \sqrt{1 - \frac{d^2}{l^2}})\right\}.$$

Pentru estimarea lui π se foloseşte formula lui P (cazul $l \leq d$,) şi se estimează P cu

$$\overline{P}_n = \frac{\alpha}{n}$$

unde α este numărul de cazuri când acul atinge o margine a lamelei din n aruncări independente.Simularea aruncării aleatoare a unui ac de lungine l ($l < d$), presupunând că cele două lamele vecine au ca margine comună axa ox, se realizează astfel: se simulează un punct P aleator intre 2 lamele ($P \in [-d, d]$;) apoi se simulează un θ uniform in $[0, +2\pi]$ şi se determină segmentul cu centrul in P, de lungime l , orientat pe direcţia θ; se verifică apoi prin calcul dacă segmentul construit intâlneşte una din dreptele $y = 0, y = d, y = -d$ care reprezintă delimitările celor două lamele vecine. Determinarea numărului de aruncări ale acului care intersectează liniile ce delimitează cele 2 lamele vecine (din n aruncări) este acum evidentă. Desigur, această cale de determinare a lui π nu este recomandabil a se face cu calculatorul deoarece algoritmul are o complexitate ridicată. Metoda acului lui Buffon are insă importanţa ei istorică. (Simularea lui θ se poate face şi uniform pe $[-\frac{\pi}{2}, +\frac{\pi}{2}]$).

E6.5 Să se precizeze o metodă Monte Carlo de estimare a numărului π folosind simularea de puncte aleatoare intr-un domeniu inconjurat de o elipsă cu centrul in origine şi de semiaxe $a > 0, b > 0$.

Indicaţie. Se foloseşte ideea de la Exemplul 6.1. Se simulează N puncte (V_1, V_2) aleatoare in dreptunghiul $[-a, a] \times [-b, b]$ şi se numără cele ce satisfac relaţia $\frac{V_1^2}{a^2} + \frac{V_2^2}{b^2} \leq 1$.

E6.6 Un lanţ Marcov cu stări continue are densitatea de repartiţie a stărilor iniţiale $\pi(x) \rightsquigarrow N(2, \sqrt{2})$ şi densitatea de tranziţie $p(x, y) \rightsquigarrow N(x, 2)$, unde x este starea de plecare. Să se prezinte un algoritm de simulare a unei stări a lanţului.

Soluţie. Avem deci

$$\pi(x) = \frac{1}{2\sqrt{\pi}} e^{-\frac{(x-2)^2}{4}},\ p(x, y) = \frac{1}{2\sqrt{2\pi}} e^{-\frac{(x-y)^2}{8}}.$$

Algoritmul este următorul

Algoritmul MARKOV de simulare a unui lanţ Marcov gaussian
Generează $X \rightsquigarrow N(2, \sqrt{2})$; ($X$ este starea iniţială);
Generează $Y \rightsquigarrow N(X, 2)$; ($Y$ este starea generată, in care se ajunge).

Acest lanţ se poate utiliza la rezolvarea ecuaţiei integrale

$$f(x) = \int p(x, y) f(y) dy + g(x),\ x \in R.$$

Pentru simularea unei traiectorii de N puncte a lanţului se continuă algoritmul precedent cu instrucţiunile

$X := Y$; $\quad genereaza \quad Y \rightsquigarrow N(X, 2)$.

Cap. 7

Cateva modele de simulare

In acest capitol vom prezenta cateva modele de simulare din domeniul teoriei aştepatării şi teoriei stocurilor, modele implementabile in limbaje evoluate (ca Pascal, C, C++, etc).

7.1 Generalităţi despre modele de aşteptare

Vom incepe cu cateva noţiuni despre modele de aşteptare. Pentru inceput vom introduce o notaţie specifică unui model de aşteptare si anume

$$A/S/c : (L_c; d)$$

unde:

A este o informaţie despre repartiţia de probabilitate a timpului de intersosire AT sau a sosirilor;

S este o informaţie despre repartiţia de probabilitate a duratei serviciului ST;

c precizează numarul de staţii de serviciu şi topologia lor, adică dacă sunt conectate in serie, in paralel, sau in reţea;

L_c este lungimea maximă a cozii, adică un număr natural pozitiv dacă coada este finită, sau ∞ dacă coada poate avea lungime infinită;

d este disciplina de serviciu care precizează reguli de efectuarea serviciului. De exemplu daca clienţii sunt serviţi in ordinea sosirilor, atunci disciplina este $FIFO$ (adica *First In First Out*). Pot exista şi reguli de tipul $LIFO$ (adică *Last In First Out*) sau disciplina de serviciu cu *prioritaţi* in sensul ca unii clienţi, care au prioritaţi pot fi serviţi inaintea altora. Insfarşit, pot exista modele cu *clienţi nedisciplinaţi;* o astfel de siutuaţie se prezintă in

cazul când clienţii care asteaptă mai mult de w_0 unitaţi de timp părăsesc sistemul fără să mai beneficieze de serviciu.

Modelele matematice sunt construite şi rezolvate de regula când repartiţiile lui AT si ST sunt exponenţiale negative, când sistemele au o singură staţie sau staţii paralele, când disciplina este $FIFO$, şi când clienţii sunt disciplinaţi. Toate celelelte tipuri de modele sunt greu de construit in ipoteze generale privind repartiţiile sosirilor şi serviciilor şi de aceea se recurge la simulare. Noi am prezentat (in Cap.1) şi câteva modele de simulare in GPSS, care de regulă au condus la programe scurte dar nu şi flexibile. Modelele ce vor fi construite in acest capitol vor putea fi extinse la diverse repartiţii ale lui AT şi ST.

Exemple de modele matematice de aşteptare sunt:

$$Exp(\lambda)/Exp(\mu)/1 : (\infty; FIFO) \tag{7.1}$$

care este cel mai simplu model cu o staţie de serviciu;

$$Exp(\lambda)/Exp(\mu)/N(paralele); (\infty; FIFO) \tag{7.2}$$

care este un model cu N staţii paralele presupuse identice (adica au aceeaşi repartiţie pentru ST-ul fiecăreia), coada poate fi infinită, iar disciplina este standard (*primul venit primul servit*).

In cazul modelelor de simulare, pentru modelele anterioare, vom folosi notaţiile:

$$\bullet/\bullet/1 : (\infty; FIFO) \tag{7.1$'$}$$

$$\bullet/\bullet/N : (\infty; FIFO) \tag{7.2$'$}$$

semnul $\bullet$ insemnând ca putem folosi orice repartiţii de probabilitate pentru AT şi ST. De asemenea, in descrierea algoritmilor (sub forma de scheme logice!) vom folosi in locul semnului de atribuire := semnul =. Modelele matematice ale teoriei cozilor se bazează pe utilizarea unor procese Markov particulare numite *procese stochastice de naştere şi deces,* procese ce vor fi studiate mai intâi. Numărul de clienţi $N(t)$ in sistemul de aşteptare la un moment dat de timp t este un astfel de proces stochastic. Aceste procese stau la baza modelării şi a altor sisteme naturale cum sunt populaţiile biologice (de unde şi denumirea lor de *procese de naştere şi deces*).

7.1.1 Procese de naştere şi deces

Definiţia 7.1. *Procesul stochastic discret $N(t)$, cu creşteri independente, se numeşte proces de naştere şi deces, dacă satisface următoarele proprietăţi:*

$$P([N(t+\Delta t) = n+1]/[N(t) = n]) = \lambda_n \Delta t + o(\Delta t);$$

$$P([N(t+\Delta t)=n-1]/[N(t)=n])=\mu_n\Delta t+o(\Delta t); \tag{7.3}$$
$$P([N(t+\Delta t)=n\pm i]/[N(t)=n])=o(\Delta t), i>1,$$

unde $P(A/B)$ *inseamnă probabilitatea lui* A *condiţionată de* B, *constantele* $\{\lambda_n, n\geq 0\}, \{\mu_n, n\geq 1\}$, *sunt şiruri de numere pozitive date, iar* $o(\Delta t)$ *este un element al unei clase de funcţii ce satisfac condiţiile*

$$\lim_{\Delta t\to 0} o(\Delta t)=0, \lim_{\Delta t\to 0}\frac{o(\Delta t)}{\Delta t}=0. \tag{7.4}$$

Procesul este cu creşteri indepentente in sensul că oricare ar fi $t_1<t_2<t_3<t_4$, *variabilele* $N(t_2)-N(t_1)$ *şi* $N(t_4)-N(t_3)$ *sunt independente stochastic, iar* $N(t+h)-N(t)=N(h), \forall t, h>0$ *(adică procesul este de* numărare*).*

Constantele λ_n se numesc *intensităţi de natalitate* iar constantele μ_n se numesc *intensităţi de mortalitate*. Aceste constante sunt caracteristice pentru procesul $N(t)$. Funcţiile $o(\Delta t)$ exprimă cantităţi neglijabile in raport cu Δt, deci care tind la zero odată cu Δt, dar mai repede decât acesta şi mulţimea lor este inchisă faţă de operaţiile de adunare şi inmulţire cu alte funcţii sau constante (adică sunt inchise la liniaritate). Relaţiile (7.3) spun că naşterile şi decesele evoluează *rar* in timp. Intensităţile de natalitate şi cele de mortalitate corespund in cazul sistemelor de aşteptare sosirilor, respectiv serviciilor (adică *plecărilor*) din sistem.

•**Teorema de bază.** Cu ajutorul proprietăţilor (7.3) se pot determina probabilităţile $P_n(t)=P[N(t)=n]$, care reprezintă repartiţia procesului de naştere şi deces.

Teorema 7. 1 *Probabilităţile* $P_n(t)$ *satisfac ecuaţiile diferenţiale*

$$P_0'(t)=-\lambda_0 P_0(t)+\mu_1 P_1(t) \tag{7.5}$$

$$P_n'(t)=-(\lambda_n+\mu_n)P_n(t)+\lambda_{n-1}P_{n-1}(t)+\mu_{n+1}P_{n+1}(t), n>0. \tag{7.6}$$

Demonstraţie. Tinând seama de relaţiile (7.3) şi de regulile de calcul cu probabilităţi condiţionate, pentru $n=0$ avem

$$P_0(t+\Delta t)=P_0(t)[1-\lambda_0\Delta t-o(\Delta t)]+P_1(t)[\mu_1\Delta t+o(\Delta t)]+o(\Delta t) \tag{7.5'}$$

unde ultimul termen reprezintă conform (7.3) probabilitatea ca pe intervalul de timp $[t, t+\Delta t]$ să se nască sau să decedeze mai mult de un individ. Din (7.5') rezultă

$$P_0(t+\Delta t)-P_0(t)=-\lambda_0 P_0(t)\Delta t+\mu_1 P_1(t)\Delta t+o(\Delta t).$$

Dacă in ultima relaţie impărţim cu Δt şi facem $\Delta t \to 0$, atunci obţinem ecuaţia (7.5). Procedăm in mod asemănător şi pentru $n > 0$ şi avem

$$P_n(t+\Delta t) = P_n(t)[1-\lambda_n\Delta t - o(\Delta t)][1-\mu_n\Delta t - o(\Delta t)]+$$

$$+P_{n-1}(t)[\lambda_{n-1}\Delta t + o(\Delta t)][1-\mu_{n-1}\Delta t - o(\Delta t)]+$$

$$+P_{n+1}(t)[1-\lambda_{n+1}\Delta t - o(\Delta t)][\mu_{n+1}\Delta t + o(\Delta t)] + o(\Delta t),$$

unde ultimul termen reprezintă de asemenea probabilitatea ca pe intervalul $[t, t+\Delta t]$ să se nască sau să decedeze mai mult de un individ.Din ultima relaţie deducem

$$P_n(t+\Delta t) - P_n(t) = -(\lambda_{n+} + \mu_n)P_n(t)\Delta t) + \lambda_{n-1}P_{n-1}(t)\Delta t+$$

$$+\mu_{n+1}P_{n+1}(t)\Delta t + o(\Delta t).$$

Impărţind ultima relaţie cu Δt şi făcând apoi $\Delta t \to o$ obţinem relaţia (7.6) şi teorema este demonstrată.

Dându-se condiţii iniţiale pentru $P_n(t)$ (de exemplu $P_0(0) = 1, P_i(0) = 0, i > 0$), sistemul (7.5),(7.6) are soluţie unică. Pe noi ne interesează ca soluţia să satisfacă şi condiţia

$$\sum_{n=0}^{\infty} P_n(t) = 1, \forall t, \tag{7.7}$$

adică să reprezinte intr-adevăr repartiţia procesului.Teorema ce urmează dă o condiţie suficientă in acest sens.

Teorema 7. 2 *Dacă este satisfăcută condiţia*

$$\sum_{k=1}^{\infty}\prod_{i=1}^{k}\frac{\mu_i}{\lambda_{i-1}} = \infty. \tag{7.7$'$}$$

atunci are loc relaţia (7.7).

Nu prezentăm aici demonstraţia acestei teoreme.

Procesul de naştere şi deces este staţionar dacă $P_n(t) = p_n = const.$ adică repartiţia sa nu depinde de timp. (Procesul se numeşte *ergodic* dacă $\lim_{t\to\infty} P_n(t) = p_n = const.$). Să vedem care este soluţia sistemului in acest caz (staţionar sau ergodic). Desigur, sistemul (7.5),(7.6) devine sistemul liniar

$$-\lambda_0 p_0 + \mu_1 p_1 = 0 \tag{7.5$''$}$$

$$-(\lambda_n + \mu_n)p_n 1\lambda_{n-1}p_{n-1} + \mu_{n+1}p_{n+1} = 0, n > 0. \qquad (7.6'')$$

Pentru a rezolva sistemul să notăm

$$Z_k = -\lambda_k p_k + \mu_{k+1}p_{k+1}.$$

Atunci din (7.5"),(7.6") rezultă

$$-Z_{n-1} + Z_n = 0$$

adică

$$Z_n = Z_{n-1},$$

iar deoarece din (7.5") avem $Z_0 = 0$, avem şi $Z_n = 0$, de unde

$$p_{n+1} = \frac{\mu_{n+1}}{\lambda_n}p_n, n \geq 0.$$

Prin recurenţă se deduce că

$$p_n = \prod_{\alpha=0}^{\infty} \frac{\lambda_\alpha}{\mu_{\alpha+1}}p_0. \qquad (7.8)$$

Utilizând (7.7) se găseşte şi constanta p_0 şi anume

$$p_0 = \left[1 + \sum_{n=1}^{\infty}\prod_{\alpha=0}^{n-1}\frac{\lambda_\alpha}{\mu_{\alpha+1}}\right]^{-1}. \qquad (7.8')$$

7.1.2 Calculul unor caracteristici ale modelului de aşteptare.

Cunoscând probabilităţile p_n se pot calcula cu uşurinţă câteva caracteristici (parametri de ieşire) pentru sistemul de aşteptare cu c staţii de serviciu şi anume:

Numărul mediu de clienţi in sistem

$$E[N(t)] = \sum_{n=0}^{\infty} np_n;$$

Lungimea medie a cozii

$$E[WL] = \sum_{n=c}^{\infty}(n-c)p_n; \qquad (7.9)$$

Timpul mediu de aşteptare

$$E[WT] = E[WL]E[ST]; \tag{7.9'}$$

Numărul mediu de staţii care lenevesc

$$E[NID] = \sum_{n=0}^{c}(c-n)p_n; \tag{7.10}$$

Timpul mediu de lenevire al celor c staţii

$$E[TID] = E[NID]E[AT]. \tag{7.10'}$$

7.2 Simularea unui sistem cu o statie

Vom studia pentru inceput modele de simulare pentru un sistem de aşteptare cu o singură staţie de serviciu de forma (7.1').

7.2.1 Modelul cu ceas variabil

Prezentăm mai intâi un model de simulare bazat pe ceasul cu creştere variabilă.

Variabilele şi parametri modelului , in afară de cele deja cunoscute, adică, AT, ST, WT (timpul de aşteptare curent al unui client), TID (timpul de lenevire curent al staţiei), sunt:

TWT = timpul total de aşteptare al clienţilor;

$TTID$ = timpul total de lenevire al staţiei;

$ICOUNT$ = contor, care numară clienţii serviţi;

NS = numarul de clienţi ce trebuie serviţi (parametru de intrare);

AWT = timpul mediu de aşteptare (parametru de ieşire);

$ATID$ = timpul mediu de lenevire (parametru de ieşire al modelului).

Schema logică a modelului de simulare este prezentată in Fig.7.1.In blocul (1) se fac iniţializarile:

$$TWT = TTID = 0; WT = TID = 0; ICOUNT = 0 \tag{7.11}$$

care sunt nesesare pentru calculul mediilor empirice $AWT, ATID$; aici se citesc parametri de intrare: NS, şi parametri repartiţiilor variabilelor AT, ST necesari pentru simularea acestor variabile aleatoare. Tot aici se iniţializează contorul $ICOUNT$ care numără serviciile şi care prin comparare cu NS impune condiţia de terminare a simulării. In blocul (2) se simulează sosirea

unui client. Pentru a inţelege mai departe algoritmul menţionăm faptul că in acest model ceasul variabil al simularii nu apare explicit (el este *implicit*), in sensul că de fapt pentru alegerea evenimentului ce trebuie prelucrat se foloseşte *regula primului eveniment urmator,* care este o consecinţă a tehnicii bazată pe ceasul cu creştere variabilă. Blocul (3) realizează o *ajustare* a timpului de sosire, care permite alegerea evenimentului (urmator) ce trebuie prelucrat şi care poate conduce fie la calculul timpului de aşteptare (in blocul (6)), fie la calculul timpului de lenevire a staţiei (in blocul (7)). Este evident ce face blocul (4) (care poate fi situat cu acelaş efect şi inainte de blocul (8)). Blocul (5) este blocul care alege evenimentul următor, ţinând seama şi de disciplina $FIFO$, iar blocul (8) precizează condiţia de terminare a simulării; el conduce la repetarea blocurilor (2)-(8) până când se simulează NS servicii.

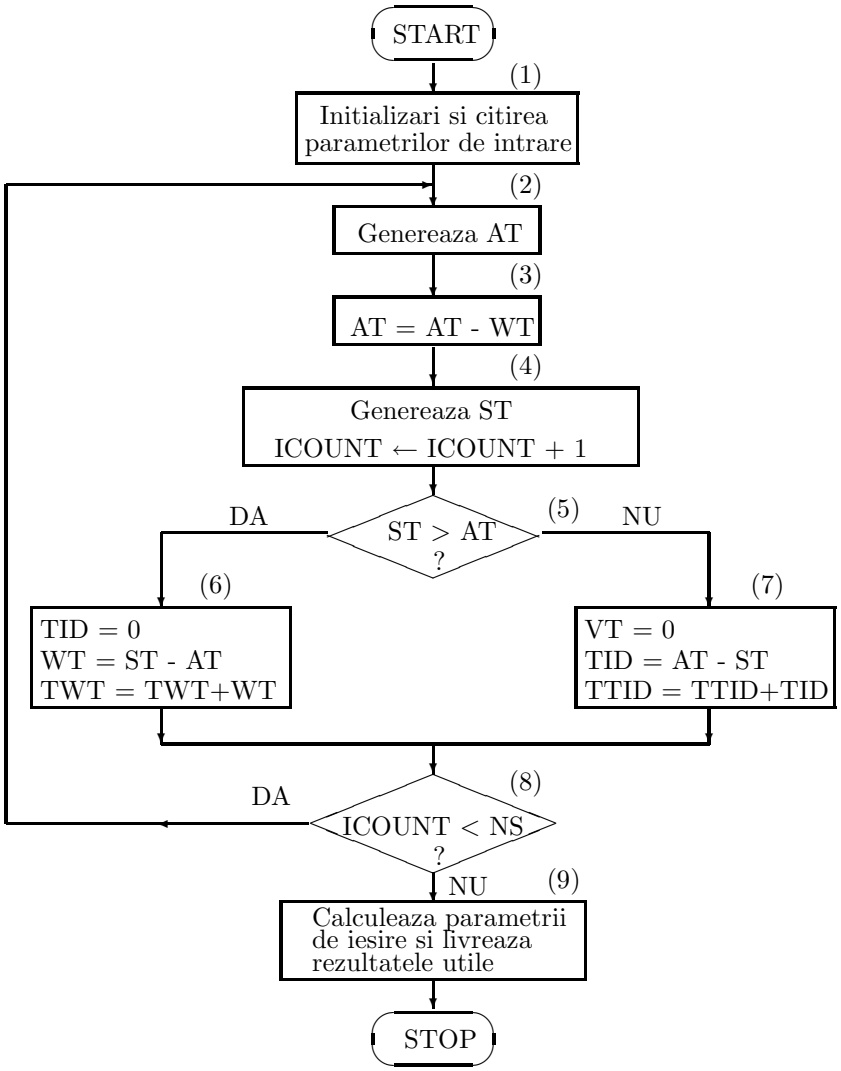

Fig. 7.1. Schema logica a modelului ./1./1: (∞ FIFO), cu ceas variabil

æææææææææææææææææææææææææææææææææææ

Insfârşit, blocul (9) calculează parametri de ieşire după formulele

$$AWT = \frac{TWT}{NS},\ ATID = \frac{TTID}{NS}. \tag{7.12}$$

7.2.2 Modelul cu ceas constant.

Acest model, faţă de modelul precedent, foloseşte in plus variabilele şi parametri:

TAT = momentul ultimei sosiri (i.e. *ceasul sosirilor*);

TST = momentul terminării ultimului serviciu (i.e. *ceasul servirilor*);

$CLOCK$ = ceasul simularii, cu creştere constantă;

C = *ora* ceasului (i.e. o constantă cu care se incrementează ceasul);

I = variabila intreagă care reprezintă lungimea curentă a cozii;

NT = parametru de intrare care reprezintă durata impusa a simularii (adica simularea se termina când $CLOCK \geq NT$).

Iniţializările in acest model sunt:

$$TWT = TTID = TAT = TST = 0,\ CLOCK = 0,\ I = 0. \qquad (7.11')$$

Schema logică a modelului este dată in Fig.7.2. Blocurile (1)-(4) precizează condiţiile initiale şi determină inceperea simulării. In blocul (5) se constată dacă staţia este libera sau ocupată. Dacă staţia este ocupată când soseşte clientul (adică $TAT \leq TST$), atunci se continuă cu blocul (6):se mareşte coada cu o unitate, se calculează timpul curent de aşteptare şi se actualizează timpul total de aşteptare; in continuare blocul (7) decide dacă este necesară creşterea ceasului (in blocul (8), când $CLOCK < TAT$), sau dacă se trece direct la blocul (9) când se generează un nou timp de sosire. Dacă in blocul (5) se decide că staţia este liberă (adică $TAT > TST$), atunci se continuă cu blocul (10) care constată dacă coada este vida sau nu; in caz afirmativ ($I = 0$,) in blocul (11) se determină timpul curent de lenevire TID, se actualizează $TTID$ precum şi ceasul serviciilor TST cu valoarea ceasului sosirilor; dacă in blocul (10) se găseşte coada nevidă, atunci se extrage un client din coadă (conform disciplinei $FIFO$). Mai departe, blocul (13) decide dacă ceasul trebuie actualizat (fapt ce se produce in blocul (14)), după care se continua cu blocul (15) in care se serveşte fie clientul sosit (dacă in blocul (10) coada era vida), fie se serveşte clientul extras din coada (in blocul (12)). Blocul (16) decide dacă simularea continuă (când $CLOCK < NT$) sau dacă simularea se opreşte, caz in care se execută blocul terminal (17) care calculează parametri de ieşire ai modelului şi alte statistici interesante.

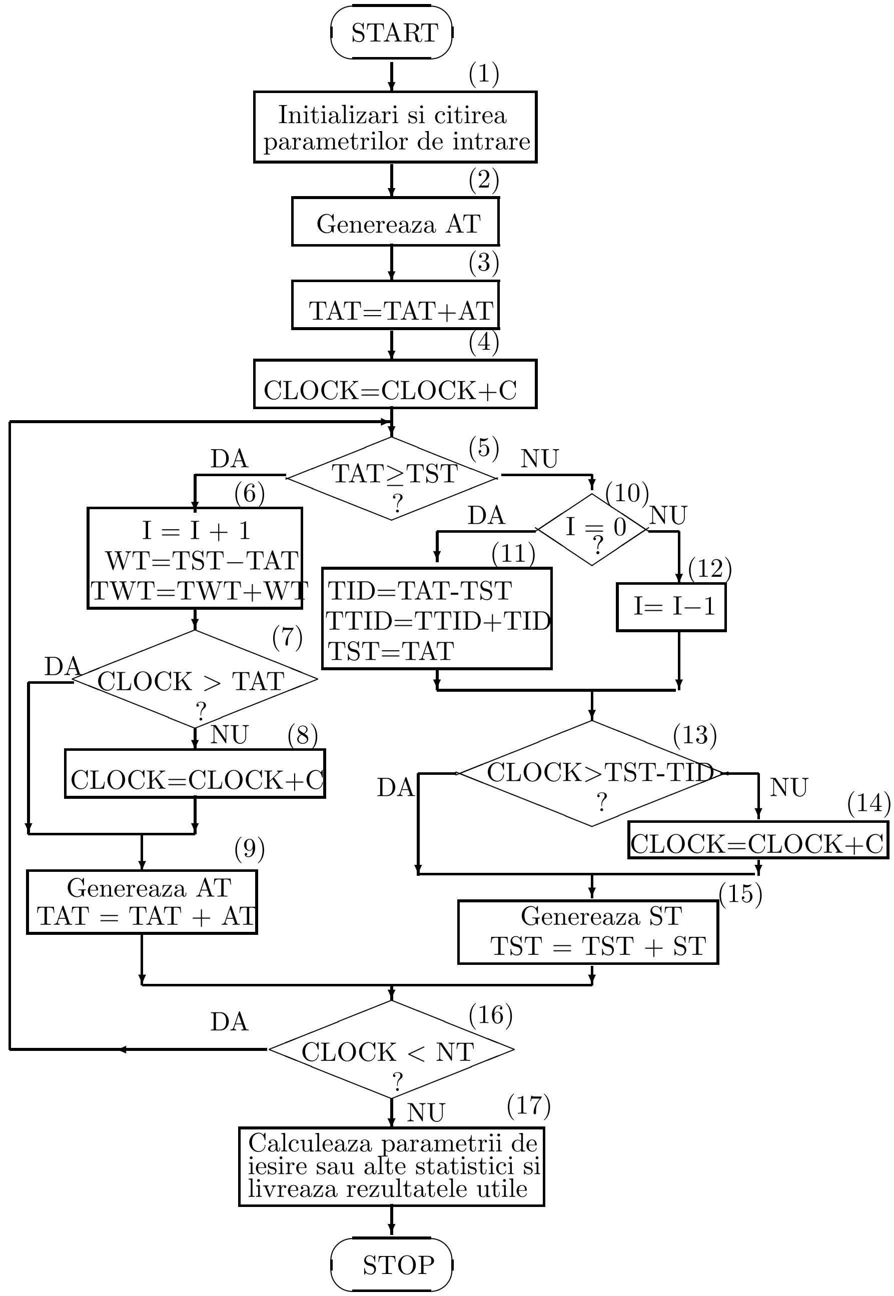

Fig. 7.2. Schema logica a modelului ./1./1: (∞ FIFO);
cu ceas constant.

ææ

Printre altele se poate calcula durata totală a serviciilor efectuate de sistem

$$SST = TST - TTID \tag{7.13}$$

precum şi timpul total petrecut de clienţi in sistem

$$TCST = SST + TWT. \tag{7.14}$$

Pentru a calcula $AWT, ATID$ este necesar să introducem in model un contor $ICOUNT$ ce trebuie initializat cu zero şi care se actualizează (se incrementează cu o unitate) in blocul (15) când se simulează servirea unui client. Calculul parametrilor $AWT, ATID$ se va realiza in acest caz tot cu formulele (7.10) (in cazul ultimului model se ia $NS = ICOUNT$). Aceasta formulă *estimează* de fapt valorile medii teoretice $E(WT), E(TID)$ ale timpilor de aşteptare şi de lenevire. Pentru ca aceste estimări sa fie cât mai bune, trebuie ca *volumele de selecţie* NS sau $ICOUNT$ să fie suficient de mari, fapt ce se realizează fie când NS este mare, in cazul ceasuliui cu creştere variabilă, fie când NT este mare, in cazul ceasului cu creştere constantă.

7.2.3 Validarea modelelor cu o staţie

Cele două modele simulează acelaş sistem de aşteptare, deci validarea lor se face pe baza aceluiaş model matematic de aşteptare şi anume modelul

$$Exp(\lambda)/Exp(\mu)/1 : (\infty, FIFO),$$

adică se consideră că repartiţiile duratelor de sosire şi de serviciu au repartiţii exponenţiale negative.

Trebuie să determinăm mai intâi intensităţile $\lambda_n, n \geq 0$, şi $\mu_n, n > 0$. Intensitatea unei sosiri va satisface deci relaţia

$$P(0 < AT < \Delta t) = 1 - e^{-\lambda \Delta t} = 1 - 1 + \lambda \Delta t + o(\Delta t) = \lambda \Delta t + o(\Delta t). \tag{7.15}$$

Deoarece, conform definiţiei 7.1 trebuie să avem

$$\lambda_n \Delta t + o(\Delta t) = \lambda \Delta t + o(\Delta t),$$

rezultă că

$$\lambda_n = \lambda, \forall n \geq 0. \tag{7.15'}$$

In mod asemănător deducem că

$$\mu_n = \mu. \tag{7.15''}$$

Deci conform formulei (7.8) avem

$$p_n = \rho^n p_0, \quad \rho = \frac{\lambda}{\mu}, \tag{7.16}$$

iar din (7.8') rezultă

$$p_0 = [1 + \sum_{n=1}^{\infty} \rho^n]^{-1} = 1 - \rho, \tag{7.16'}$$

adică

$$p_n = \rho^n(1 - \rho).$$

Conform relaţiei cunoscute dintre repartiţia exponenţială negativa şi repartiţia Poisson, rezultă că λ este numărul mediu de sosiri pe unitatea de timp, iar μ este numărul mediu de clienţi serviţi pe unitatea de timp. De aceea ρ reprezintă *intensitatea de trafic* a clienţilor prin sistem.Pentru a avea sens p_0 trebuie ca $0 < \rho < 1$. (Deci un sistem de aşteptare cu $\lambda > \mu$ *nu are sens!*). Pentru validarea modelelor de simulare cu o staţie putem deci folosi modelul matematic anterior. Formulele (7.9),(7.9'),(7.10),(7.10') devin respectiv

$$E[WL] = \sum_{n=2}^{\infty}(n-1)\rho^n(1-\rho) =$$

$$= \rho^2(1-\rho)\sum_{n=2}^{\infty}(n-1)\rho^{n-2} = \rho^2(1-\rho)\sum_{n=1}^{\infty} n\rho^{n-1} =$$

$$= \frac{\rho^2}{1-\rho},$$

$$E[WT] = E[ST]E[WL] = \frac{1}{\mu}E[WL] = \frac{\rho^2}{\mu(1-\rho)}. \tag{7.17}$$

$$E[NID] = \sum_{n=0}^{1}(1-n)p_n = p_0 = 1 - \rho,$$

$$E[TID] = E[AT]E[NID] = \frac{1-\rho}{\lambda}. \tag{7.18}$$

Validarea modelelor de simulare se realizează când

$$AWT \approx E[WT], \quad ATID \approx E[TID], \tag{7.19}$$

sau dacă

$$|E(WT) - AWT| \leq \epsilon \tag{7.19'}$$

cu $\epsilon > 0$ dat, suficient te mic ($\epsilon = eroare$). Desigur in practică se poate folosi pentru validare numai unul din parametri $E[WT]$ sau $E[TID]$. Dacă (7.19) are loc, atunci inseamnă că in loc de repartiţii exponenţiale se pot folosi orice repartiţii pentru AT şi ST.

O caracteristică importantă a unui sistem de aşteptare este *factorul de eficienţă al sistemului* care se defineşte astfel

$$I_e = \frac{E(W)}{E(ST)}, E(W) = E[N(t)]\frac{1}{\mu} \tag{7.20}$$

care spune de fapt in ce măsură timpul petrecut de client in sistem a fost folositor pentru el.(Clientul este interesat ca I_e să fie cât mai mic!). Factorul de eficienţă se estimează cu ajutorul simulării astfel

$$\overline{I_e} = \frac{TWT + SST}{SST} \tag{7.20'}$$

şi el poate fi utilizat şi pentru validare.

Pentru un model de aşteptare cu o staţie, cu sosiri exponenţiale şi servicii oarecare (adică modelul $Exp(\lambda)/B/1 : (\infty; FIFO), E(B) = E(ST) = \frac{1}{\mu},$) s-a arătat că

$$I_e = \frac{\rho}{2(1-\rho)}(1 + C_s^2) \tag{7.20''}$$

unde C_s este coeficientul de variabilitate al timpului de serviciu definit astfel

$$C_s = \frac{\sqrt{Var(ST)}}{E(ST)}.$$

Formula (7.20"), cunoscută sub numele de *Formula lui Pollaczek*, spune in fapt că factorul de eficienţă al sistemului de aşteptare depinde numai de $\rho = \lambda/\mu$ şi de C_s.

7.3 Simularea unui sistem cu N staţii paralele

Vom prezenta aici modelul de simulare

$$././N : (\infty, FIFO)$$

in care cele N staţii sunt presupuse paralele. In practică un astfel de model se intalneşte la o staţie de *service auto,* la o staţie de benzină sau la o

bancă (*ghişeele care servesc clienţii*), etc. Lista variabilelor şi parametrilor modelului este

AT = intervalul de timp (aleator) dintre două veniri consecutive;

$ST(J)$ = durata unui serviciu (aleator) la staţia $J, 1 \leq J \leq N$, (N este parametru de intrare; tot parametri de intrare sunt parametri repartiţiilor de probabilitate ale variabilelor aleatoare AT şi $ST(J)$);

WT = timpul de aşteptare in coadă al unui client oarecare;

TID = timpul curent de lenevire al staţiei care incepe un serviciu;

TAT = momentul sosirii ultimului client (clientul curent);

$TT(J)$ = timpul staţiei J la plecarea din ea a ultimului client (*ceasul* staţiei J);

$TTMIN$ = ceasul simulării, cu creştere variabilă (definit ca cel mai mic dintre ceasurile $TT(J)$);

L = staţia cu cel mai mic ceas, adică $TTMIN = TT(L)$;

NS = numărul de clienţi ce trebuie serviţi pe parcursul simulării, este parametru de intrare;

$ICOUNT$ = contur care număra serviciile;

$SST(J)$ = suma duratelor de serviciu ale staţiei J;

$SWT(J)$ = suma timpilor de aşteptare ai clienţilor serviţi de staţia J;

$TTID(J)$ = timpul total de lenevire al staţiei J;

$DIF = TAT - TTMIN$ = variabilă de lucru.

Schema logică a modelului de simulare este prezentată in Fig.7.3. In blocul (1) sunt citiţi parametri de intrare şi sunt făcute iniţializările

$$TAT = 0, TT(J) = SST(J) = SWT(J) = TTID(J) = 0, 1 \leq J \leq N. \tag{7.21}$$

In continuare, ciclul definit de blocurile (2) şi (3) generează $N - 1$ sosiri (se presupune că la inceputul simulării un client se află deja in sistem). Ciclul format din blocurile (5) şi (6) simulează servirea primilor N clienţi (specificaţi in blocul (4)) şi actualizează variabilele $TT(J), TTID(J), ST(J), SST(J), 1 \leq J \leq N$. Blocurile (7)-(15) realizează ciclul de simulare al servirii celor NS clienţi. Astfel, blocul (7) determină ceasul (variabil) al simulării alegând staţia L la care se va efectua următorul serviciu, numărat de blocul (8). Blocul (9) controlează şi decide terminarea simulării. Blocul (10) simulează o nouă sosire, actualizează momentul ultimei sosiri TAT, şi calculează variabila de lucru DIF, al cărei semn este testat de blocul (11). Dacă semnul lui DIF este negativ atunci in blocul (12) se calculează timpul de aşteptare al clientului ce urmează să intre in serviciu şi se actualizează timpul de aşteptare al clienţilor serviţi de staţia L, altfel dacă DIF in blocul (11) este pozitiv, atunci in blocul (13) se calculează şi actualizează timpul de lenevire

al staţiei L. In continuare blocul (14) simulează servirea clientului (sosit cel mai de demult conform blocului (12) sau deja sosit conform blocului (10)), conform disciplinei FIFO, de către staţia L. Blocul (15) actualizează ceasul staţiei L care a terminat serviciul si se reia ciclul simulării din blocul (9). Când simularea se termină, se execută blocul (16) care calculează ca deobicei statisticile finale şi alte elemente de interes.

• **Validarea modelului cu N staţii paralele.** Pentru validare folosim modelul matematic de aşteptare

$$Exp(\lambda)/Exp(\mu)/N : (\infty; FIFO) \tag{7.22}$$

adică modelul cu N staţii paralele identice, cu timp de intersosire exponenţial de parametru λ, cu timpul de servire al fiecării staţii repartizat exponenţial de parametru μ, şi cu disciplina de serviciu $FIFO$.

Va trebui şi in acest caz să determinăm intensităţile $\lambda_n, n \geq 0$ şi $\mu_n, n > 0$. Ca şi in cazul modelului cu o staţie, avem

$$\lambda_n = \lambda, \ n \geq 0. \tag{7.23}$$

Deoarece intensitatea servirilor pentru fiecare staţie este μ şi serviciul se desfăşoară in paralel, avem

$$\mu_n = \begin{cases} n\mu, & 1 \leq n \leq N-1, \\ N\mu, & n \geq N. \end{cases} \tag{7.24}$$

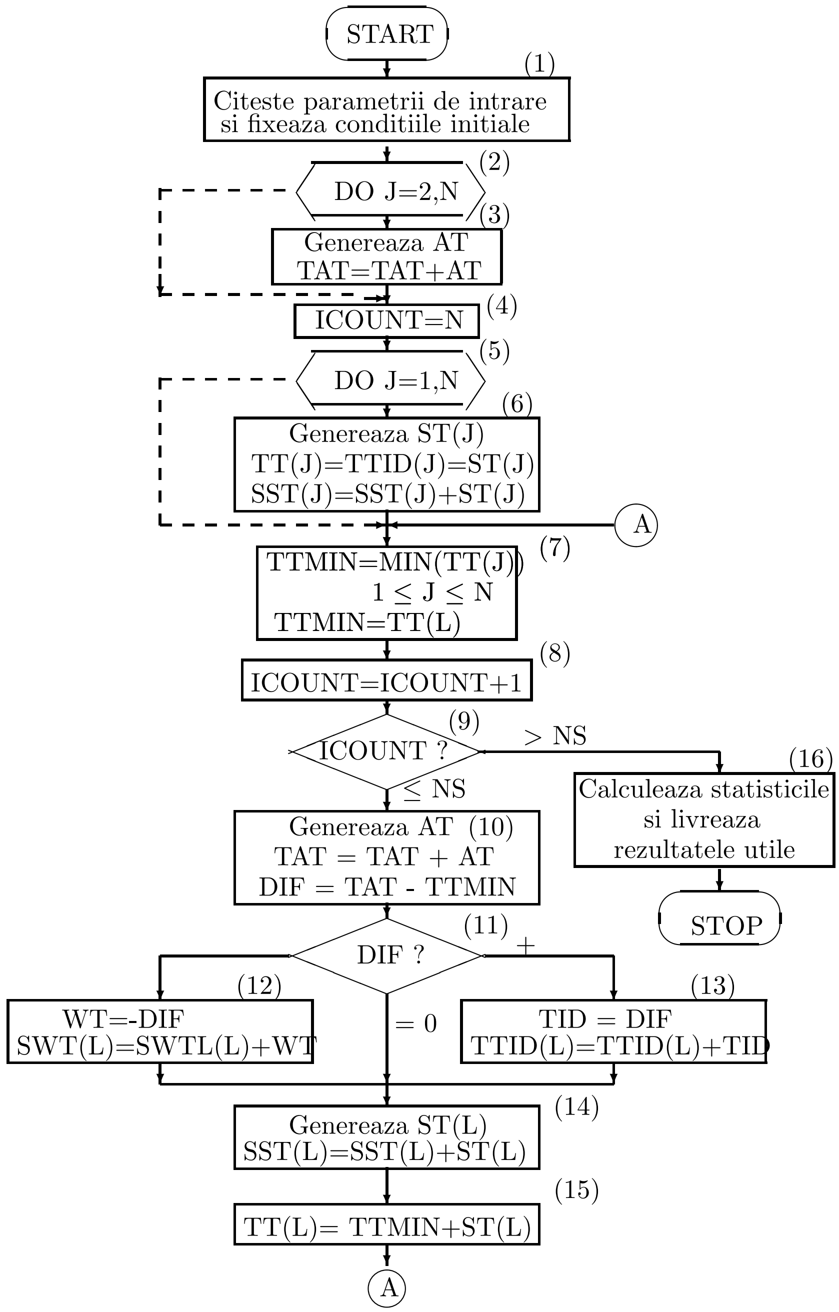

Fig. 7.3. Schema logica pentru simularea unui sistem
cu N staţii paralele.

æææ

Notând ca şi in cazul modelului cu o staţie

$$\rho = \frac{\lambda}{\mu},$$

conform cu (7.8) deducem

$$p_n = \begin{cases} \frac{\rho^n}{n!} p_o, & 1 \leq n \leq N-1, \\ \frac{\rho^n}{N! N^{n-N}} p_0, & N \leq n < \infty, \end{cases} \tag{7.25}$$

iar din (7.8') rezultă

$$p_0 = \left[\sum_{n=0}^{N-1} \frac{\rho^n}{n!} + \frac{\rho^N}{N!} \frac{N}{N-\rho} \right]^{-1}. \tag{7.25'}$$

Lungimea medie a cozii este

$$E(WL) = \sum_{n=N}^{\infty} (n-N) p_n = p_0 \left[\frac{N^n}{N!} \sum_{n=N}^{\infty} n \frac{\rho^n}{N^n} - \frac{N^{N+1}}{N!} \sum_{n=N}^{\infty} \frac{\rho^n}{N^n}. \right].$$

Notând

$$\rho^* = \frac{\rho}{N}$$

se deduce prin calcule că

$$E(WL) = p_0 \left[\frac{N^n}{N!} \rho^* \frac{d}{d\rho} \left(\sum_{n=N}^{\infty} \rho^{*n-1} \right) - \frac{N^{N+1}}{N!} \frac{\rho^{*N+1}}{1-\rho^*} \right] =$$

$$= \frac{N^N}{N!} \frac{\rho^{*N+1}}{(1-\rho^*)^2} = \frac{\lambda \mu \rho^N}{(N-1)(N\mu - \lambda)^2}. \tag{7.26}$$

Mărimea ρ^* se numeşte *intensitatea de trafic a sistemului de aşteptare.*

Timpul mediu de aşteptare este

$$E(WT) = E(WL)E(S), E(S) = E(ST)/N,$$

unde $E(S)$ este timpul mediu de serviciu al sistemului. In final se obţine

$$E(WT) = \frac{\lambda \rho^N}{N(N-1)!(N\mu - \lambda)}. \tag{7.26'}$$

In mod asemănător se calculează numărul mediu de staţii care lenevesc şi timpul mediu de lenevire al staţiilor, adică

$$E(NID) = \sum_{n=0}^{N-1} (N-n) p_n = \left[(N-\rho) \sum_{n=0}^{N-2} \frac{\rho^n}{n!} + \frac{N \rho^{N-1}}{(N-1)!} \right] p_0, \tag{7.27}$$

$$E(TID) = E(NID)E(AT) = \frac{1}{\lambda}\left[(N-\rho)\sum_{n=0}^{N-2}\frac{\rho^n}{n!} + \frac{N\rho^{N-1}}{(N-1)!}\right]p_0. \quad (7.27')$$

Din modelul de simulare se obţin respectiv următoarele estimaţii pentru $E(WT)$ şi $E(TID)$

$$AWT = \frac{\sum_L SWT(L)}{NS}, \quad (7.26'')$$

$$ATID = \frac{\sum_L TTID(L)}{NS}. \quad (7.27'')$$

Ca şi in cazul modelului cu o staţie, condiţiile de validare pentru acest model sunt

$$|E(WT) - AWT| < \epsilon, \quad |E(TID) - ATID| < \epsilon.$$

In caz de validare, inseamnă că modelul de simulare construit poate fi utilizat şi dacă repartiţiile variabilelor aleatoare AT şi ST sunt oarecare (diferite de exponenţiale).

7.4 Modele de simulare pentru stocuri

7.4.1 Introducere in teoria matematică a stocurilor

Un stoc este o resursă de orice fel care are o valoare economică, caracterizată prin *intrări* şi *ieşiri*. Ieşirea din resursă este determinată de regulă de cerere, deşi nu intotdeauna. (Pot fi scoase din stoc şi elemente *expirate* sau deteriorate).

Stocuri (numite uneori şi *inventare*), există in aproape orice activitate economico-socială (de ex. unitate de producţie, unitate comercială, etc). Scopul unui model de stocare este să definească regulile de incărcare optimă a stocului, astfel incât costul (sau profitul) legat de aprovizionarea şi intreţinerea stocului să fie minim (maxim). Stocul se poate măsura in unităţi fizice (de ex. kilograme, metri, bucăţi, etc), sau in unităţi valorice convenţionale (de regulă unităţi monetare).

Variabilele şi parametri unui model general de stocare sunt:

t=timpul;

$I(t)$= nivelul curent al stocului;

$a(t)$= rata intrării in stoc la momentul t;

$b(t)$= rata ieşirii sin stoc la momentul t;

$r(t)$= rata cererii (când aceasta nu coincide cu ieşirea de altă natură).

Rata cererii, ca şi alte elemente ce caracterizează un stoc, pot fi aleatoare.

De regulă, intrarea in stoc se realizează in cantităţi mari (numite *comenzi*) care se introduc in stoc la intervale de timp numite *cicluri de reaprovizionare.* Costurile sunt (parametri!) de tipul:

h= costul de stocare a unei unităţi de stoc intr-o unitate de timp;

d= costul lipsei unei unităţi de stoc intr-o unitate de timp ;

s= costul de lansare a unei comenzi (costul forţei de muncă sau al altor resurse folosite pentru a lansa comanda).

Din cele de mai sus rezultă că *nivelul* stocului la momentul t este

$$I(t) = I_0 + \int_0^t [a(t) - b(t)]dt, \qquad (7.28)$$

unde I_0 este nivelul iniţial al stocului.

In funcţie de elementele de mai sus se defineşte un obiectiv, sa-l notăm $E[a(t), b(t), r(t)]$, care trebuie optimizat. Aşa cum am menţionat, acest obiectiv poate fi un cost (sau un profit) determinat in funţie de cerere şi de costurile h, d, s şi de unele necunoscute (comandă sau allte elemente ce vor fi definite mai jos). Când cererea este aleatoare, funcţia obiectiv este o *medie* (cost mediu sau beneficiu mediu). De regulă, scopul modelului de stocare este de a defini $a(t)$ optim când se dau $b(t), r(t)$. Uneori insă (de exemplu când stocul este un lac de acumulare), scopul modelului este să determine ieşirea optimă (adică $b(t)$ şi/sau $r(t)$) când rata (debitul) intrării $a(t)$ este cunoscut. Pentru a face o alegere ne vom ocupa de primul caz (al stocurilor obişnuite, caz mai frecvent intâlnit),

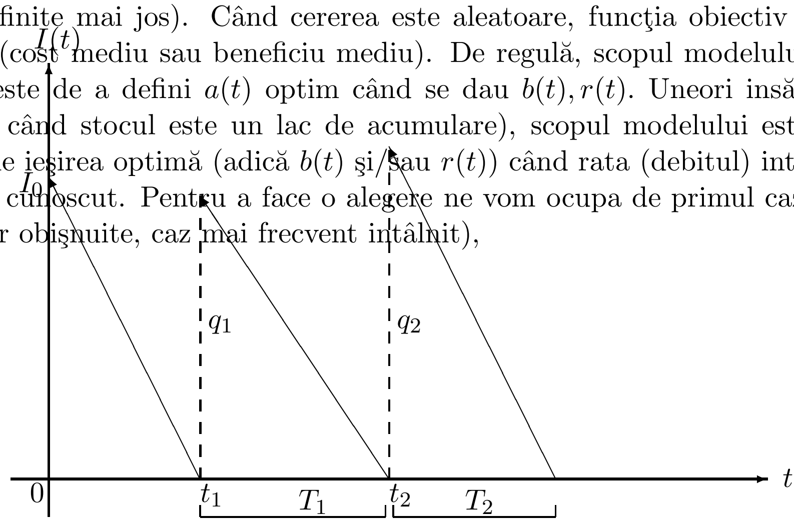

Fig. 7.4. Variaţia stocului $I(t)$

æææ

când se cere intrarea optimă. Am văzut că in acest caz intrarea in stoc se realizează prin comenzi. *Reaprovizionarea* stocului şi variaţia acestuia se prezintă grafic in Fig. 7.4. *Comenzile* $q_1, q_2, ...$ intră in stoc la momentele $t_1, t_2, ...$ Intervalul $T_i = t_{i+1} - t_i$ este *ciclul de reaprovizionare* care paote fi constant sau variabil.

Reaprovizionarea stocului se realizează printr-un *mecanism* de reaprovizionare descris de Fig. 7.5. Se presupune că stocul nu depăşeşte un *nivel maxin* S. In timp stocul scade cu o rată dată (care este constantă in Fig. 7.5, deoarece variaţia stocului este liniară); când stocul scade până la un nivel P, numit *nivel reaprovizionare,* atunci se lansează o comandă q care va intra in stoc după *timpul de avans* L. Comenzile intră in stoc la intervale de timp de reaprovizionare de lungime T. Nivelul de reaprovizionare ar trebui să poată satisface cererea pe perioada timpului de avans L. Dar uneori acceptăm ca stocul de rezervă P să poată satisface cererea pe o perioadă de timp de lungime t', $t' < T$. In acest caz există o perioadă $t'' = T - t'$ când are loc *lipsa de stoc.* Un astfel de model se numeşte *cu lipsă de stoc* şi in el intervine costul d. Conform Fig. 7.5 *cererea ne satisfăcută se păstrează* in sensul că atunci când q intră in stoc se consumă o parte din comandă pentru a satisface cererea care nu a fost satisfăcută pe perioada t''.

In acest caz se admite că nivelul stocului ia şi valori negative, iar nivelul la care se ridică stocul va fi mai mic decât q. Dacă cererea nesatisfăcută *nu se păstrează,* nu se admite stoc negativ şi la intrarea lui q in stoc nivelul maxim al stocului devine $S = q$. Pentru un astfel de model *cererea nestisfăcută se pierde.*

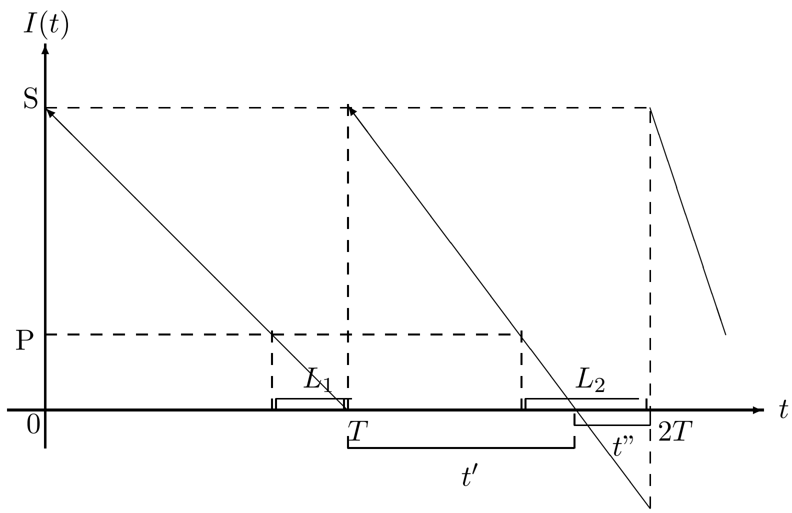

Fig. 7.5. Mecanismul reaprovizionării

ææ

In diverse modele de stocare se pot da costurile h, s şi/sau d, se dă rata cererii r şi timpul de avans L şi se cer q, P optime. O mulţime de elemente ce definesc un mecanism de aprovizionare se spune că determină o *politică de*

reaprovizionare. Când r, L nu sunt aleatoare modelul se numeşte *determinist*; in cazul când cel puţin una din aceste variabile este aleatoare modelul este *stochastic*. Dacă timpul nu intervine in mod explicit in model, spunem că avem de-a face cu un model *static*, altfel, modelul este *dinamic.*

In concluzie, modelele de teoria stocurilor işi propun să determine o politică de reaprovizionare optimă.

7.4.2 Modele simple deterministe de stoccare a unui produs

Vom prezenta două din cele mai simple modele statice deterministe pentru stocarea unui produs.Aceste modele vor constitui scheletul pe baza cărora se vor construi modelele de simulare.

• **Modelul clasic al lotului economic.** Acest model, cunoscut şi sub numele de **modelul lui Wilson,** are la bază următoarele ipoteze:

- rata cererii r este constantă, cunoscută şi cererea este continuă;

- ciclul de reaprovizionare T este constant şi ne cunoscut;

- comanda q este constantă şi necunoscută; intrările cantităţilor q in stoc au loc instantaneu la intervale de timp T;

- timpul de avans L este zero, adică neglijabil; (comenzile q intră in stoc imediat după ce se lansează comanda);

- nivelul de reaprovizionare P este zero (in concordanţă cu ipoteza anterioară);

- costurile de stocare h şi de lansare s sunt constante date;

- nu se admite lipsă de stoc (adică $d = 0$).

Se observă că intre necunoscutele q, T are loc relaţia

$$T = \frac{q}{r}. \tag{7.29}$$

Variaţia stocului in cazul acestui model este prezentată in Fig.7.6.

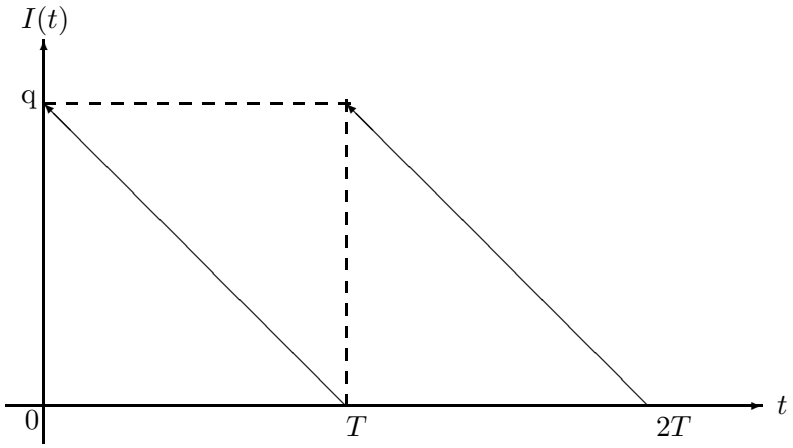

Fig. 7.6. Variaţia stocului pentru modelul clasic al lotului economic
æææææ

Costul total de intreţinere a stocului pe perioada T este

$$C_T = C_{h,T} + s \tag{7.30}$$

unde $C_{h,T}$ este costul de stocare şi s este costul de lansare. Avem

$$C_{h,T} = h\int_0^T I(t)dt = \frac{hq}{2}T = \frac{hq^2}{2r}. \tag{7.30'}$$

Funcţia obiectiv a modelului constă in a minimiza costul asociat menţinerii stocului. Dacă am considera acest cost aşa cum este el definit de (7.30), atunci s-ar deduce că $T = 0$ ceea ce nu are sens. De aceea luăm ca obiectiv **minimizarea costului** $C(q)$ **pe unitatea de timp,** adică, tinând seama şi de (7.29) trebuie să impunem condiţia

$$C(q) = \frac{C_T}{T} = \frac{hq^2}{2r}\frac{1}{T} + \frac{s}{T} = \frac{hq}{2} + \frac{sr}{q} = min. \tag{7.31}$$

Impunând condiţia de mninim lui $C(q)$ obţinem politica optimă de reaprovizionare $(\hat{q_0}, \hat{T_0})$ dată de

$$\hat{q_0} = \sqrt{\frac{2rs}{h}}, \quad \hat{T_0} = \frac{\hat{q_0}}{r} = \sqrt{\frac{2s}{rh}} \tag{7.32}$$

iar costul optim (i.e. minim) este

$$\hat{C_0} = \sqrt{2rsh}. \tag{7.32'}$$

Din formula (7.31) se deduce o regulă simplă de calcul a costului de stocare pe unitatea de timp, anume, el este produsul dintre h şi stocul mediu pe perioada ciclului de reaprovizionare care este $(0+q)/2$; de aici se deduce costul de stocare mediu pe o perioadă de lungime t, adică

$$C_{h,t} = h\frac{q}{2}t. \tag{7.31'}$$

• **Modelul clasic al lipsei de stoc.** In acest caz sunt satisfăcute ipotezele modelului precedent cu excepţia faptului că există lipsă de stoc (adică $d > 0$, d =constantă dată) şi stocul poate creşte până la o valoare S necunoscută. Variaţia stocului in acest caz este ilustrată de Fig. 7.7. Raţionând asemănător modelului precedent se deduce funcţia obiectiv de minimizat de forma

$$C(S,T) = \frac{1}{T}\left[C_{h,t'} + C_{d,t''} + s\right] \tag{7.33}$$

unde $C_{h,t'}$ a fost definit mai sus in (7.31') iar $C_{d,t''}$ se defineşte in mod asemănător. Detaliind costurile rezultă că

$$C(S,T) = \frac{1}{T}(s + h\frac{S}{2}t' + d\frac{q-S}{2}t''), \quad q = rT. \tag{7.33'}$$

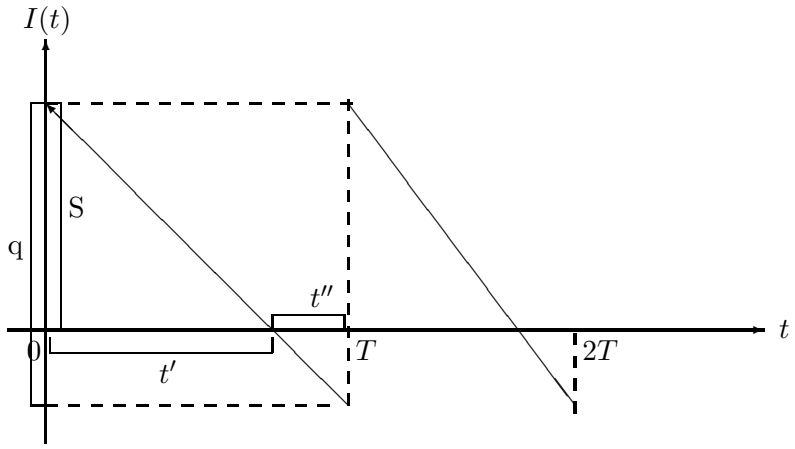

Fig. 7.7. Variaţia stocului pentru modelul lipsei de stoc
æææææææææææææ

Din asemănarea unor triunghiuri din Fig.7.7 rezultă

$$t' = \frac{S}{r}, \quad t'' = T - \frac{S}{r} = \frac{rT - S}{r}$$

de unde se obţine in final

$$C(S,T) = \frac{s}{T} + \frac{hS^2}{2rT} + \frac{d(rT-S)^2}{2rT}. \tag{7.34}$$

Impunând condiţii de minim in (7.34), adică

$$\frac{\partial C}{\partial S} = 0, \quad \frac{\partial S}{\partial T} = 0,$$

deducem in final valorile $\hat{S}_1, \hat{T}_1$ care realizează costul minim $\hat{C}_1$,

$$\hat{T}_1 = \sqrt{\frac{2s}{rh}}\sqrt{\frac{1}{\rho}}, \quad \rho = \frac{d}{h+d},$$

$$\hat{S}_1 = \sqrt{\frac{2rs}{h}}\sqrt{\rho}, \quad \hat{q}_1 = \hat{T}_1 r = \sqrt{\frac{2rs}{h}}\sqrt{\frac{1}{\rho}}. \tag{7.35}$$

Se observă că $0 < \rho < 1$, care conduce la

$$\hat{C}_1 = \sqrt{\rho}\hat{C}_0 < \hat{C}_0,$$

de unde rezultă la prima vedere că modelul clasic al lipsei de stoc este mai bun dacât modelul clasic al lotului economic. Coeficientul ρ satisface relaţia $\rho = \frac{\hat{S}_1}{\hat{q}_1}$, de unde rezultă că $(1-\rho)\%$ poate fi interpretat ca *indice de lipsă.* Deci dacă s-ar da indicele de lipsă $\alpha = 1-\rho$ atunci ar rezulta că $d = \frac{1-\alpha}{\alpha}h$ adică ar rezulta că există o dependenţă strictă intre costurile h şi d, ceea ce nu este intocmai adevărat in practică. Acest fapt constituie deci o *critică* a modelului clasic al lotului economic.

7.4.3 Modele de simulare particulare

Vom prezenta aici două modele de simulare simple pentru un produs, din care vor rezulta planuri de reaprovizionare şi costuri optime. Modelele vor fi construite pe scheletul modelului clasic al lipsei de stoc.

• **Primul model.** In acest model vom folosi următoarele variabile şi parametri

H = costul de stochare;
D = costul lipsei de stoc;
S = costul de lansare;
CH = costul total de stocare pe perioada simulată;
CD = costul total al lipsei de stoc pe perioada simulată;

CS = costul total de lansare pe perioada simulată;

TC = costul total ($TS = CH + CD + CS$);

T = momentul de timp (curent) când intră o comandă in stoc;

R = cerera pe unitatea de timp (rata cererii);

VI = nivelul curent al stocului;

Q = comanda optimă;

P = nivelul de reaprovizionare;

L = timpul de avans;

$CLOCK$ = ceasul simulării;

BI = nivelul iniţial al stocului;

TT = perioada de timp pe care se face simularea.

Parametrii de intrare sunt: H, D, S, P, BI, TT; variabilele R, L sunt variabile aleatoare de intrare (având repartiţii date) iar parametri acestor repartiţii sunt de asemenea parametri de intrare; toate celelalte variabile listate anterior sunt variabile de ieşire.

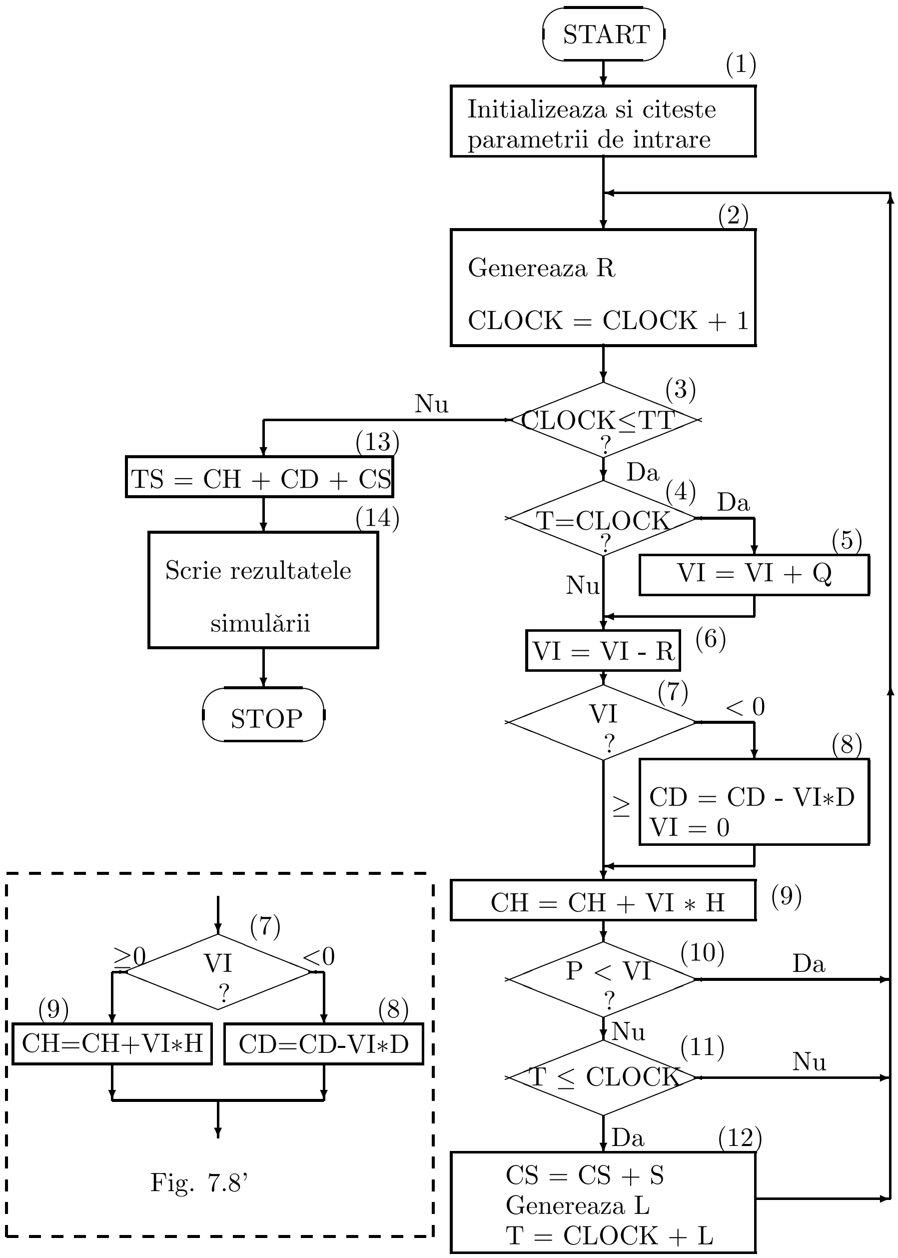

Fig. 7.8'

Fig. 7.8. Schema logica pentru primul model

æææ

Condiţiile iniţiale (inclusiv valorile iniţiale ale variabilelor) sunt

$$TC = CS = CD = CH = 0, \quad VI = BI, \quad T = CLOCK = 0. \tag{7.36}$$

Mărimea comenzii Q se determină pe parcursul simulării cu formula

$$Q = \sqrt{\frac{2RS}{H}}\sqrt{\frac{H+D}{D}}, \quad R \rightsquigarrow N(m,\sigma),\ m >> 3\sigma \tag{7.37}$$

unde R se simulează in prealabil; comanda Q este calculată in ipotezele modelului clasic al lotului economic.

Nivelul de reaprovizionare, in ipoteza $R \rightsquigarrow N(m,\sigma)$ se calculează la inceputul execuţiei simulării folosind proprietatea $Prob(R_{(L)} > P) = \alpha$, unde α este un risc mic dat ($\alpha \approx 0.05$), iar $R_{(L)} = R_1 + ... + R_L$ este *cererea pe timpul aleator de avans* L, (L-intreg) cu media $l = E[L]$. Deoarece $R_{(L)} \rightsquigarrow N(lm, \sqrt{l}\sigma)$ rezultă că

$$Prob(R_{(L)} > P) = Prob(\frac{R_{(L)} - ml}{\sqrt{l}\sigma} > \frac{P - lm}{\sqrt{l}\sigma}) = \alpha,$$

şi deoarece

$$Z = \frac{R_{(L)} - lm}{\sqrt{l}\sigma} \rightsquigarrow N(0,1)$$

rezultă

$$\frac{P - lm}{\sqrt{l}\sigma} = z_\alpha, \quad unde \quad \int_{-\infty}^{z_\alpha} e^{-\frac{t^2}{2}}dt = 1 - \alpha.$$

Din ultima relaţie rezultă nivelul de reaprovizionare de forma

$$P = lm + z_\alpha\sqrt{l}\sigma. \tag{7.38}$$

Schema logică a modelului de simulare este dată de Fig. 7.8 şi ea se autoexplică. Modelul de simulare este construit in ipoteza că *cererea nesatisfăcută se pierde,* fapt ilustrat de blocurile (7) şi (8) din schemă. Printr-o modificare minoră se poate introduce in model ipoteza că *cererea nesatifăcută se păstrează.* Acest fapt este ilustrat prin blocurile (7),(8),(9) din figura incadrată 7.8' care ar trebui să inlocuiască blocurile (7),(8) din Fig. 7.8. Introducând in Fig. 7.8 instrucţiuni corespunzătoare de afişare (scriere), se poate obţine *planul de reaprovizionare* pe perioada simulată TT, constând din momentele intrării in stoc T şi comenzile optime Q.

• **Al doilea model.** Acest model este asemănător primului model, dar el foloseşte unele ipoteze mai generale ce vor fi precizate in continuare.

Astfel, sunt folosite multe din variabilele şi parametri primului model, dar sunt incluse şi următoarele:

AR=media mobilă a cererii calculată pe M unităţi de timp precedente;

RS= abaterea medie pătratică a cererii ca medie mobilă pe M perioade de timp precedente;

AL= media mobilă a timpului de avans calculat din datele pe N perioade precedente;

$K = z_\alpha = \alpha-$ cuantila repartiţiei normale $N(0,1)$ (folosită şi in modelul precedent).

Se presupune deci că rata cererii $R \rightsquigarrow N(\mu(t), \sigma(t))$, are parametri variabili in timp iar L are de asemenea media $E[L] = l(t)$ variabilă in timp. Se presupune că aceste variabile aleatoare au parametri constanţi pe perioade mici de timp de lungimi M respectiv N, parametri ce sunt estimaţi ca medii mobile de forma

$$AR = \frac{\sum_{i=1}^{M} R_i}{M} = \frac{SUMR}{M} \tag{7.39}$$

$$AL = \frac{\sum_{j=1}^{N} L_j}{N} = \frac{SUML}{N} \tag{7.40}$$

$$SR = \sqrt{\frac{\sum_{i=1}^{M} R_i^2}{M} - \left(\frac{\sum_{i=1}^{M} R_i}{M}\right)^2} = \sqrt{\frac{SSUMR}{M} - \left(\frac{SUMR}{M}\right)^2}. \tag{7.41}$$

In ipoteza parametrilor constanţi putem deci aplica formulele de calcul pentru comanda Q şi nivelul de reaprovizionare L, adică

$$Q = \sqrt{\frac{2.AR.S}{H}}\sqrt{\frac{H+D}{D}} \tag{7.42}$$

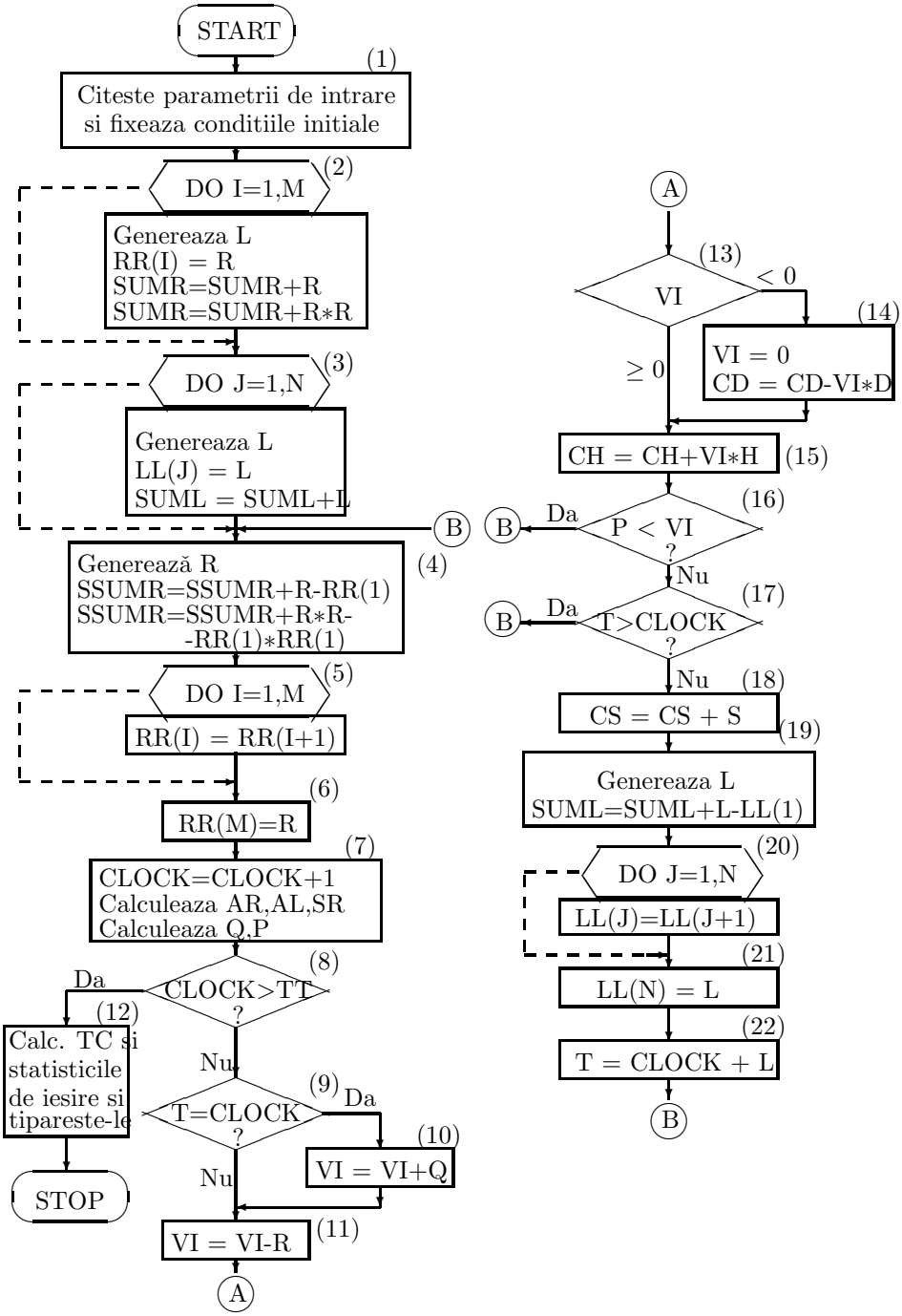

Fig. 7.9. Schema logica pentru al doilea model de stocare

ææ

$$P = AL.AR + K\sqrt{AL}.SR. \quad (7.43)$$

Schema logică a acestui model de simulare este prezentată in Fig. 7. 9.

Condiţiile iniţiale sunt (7.36) precum şi cele ce rezultă din considerarea noilor variabile adică

$$SUMR = SSUMR = SUML = 0. \quad (7.44)$$

Algoritmul simulării foloseşte doi vecroti $RR(I), 1 \leq I \leq M$ (care memorează R_i din (7.39), (7.41)) şi $LL(J), 1 \leq j \leq N$ (care memorează L_j din (7.40)). Blocurile schemei logice se autoexplică. Trebuie sa precizăm insă că printre parametri de intrare se dau şi parametri cu care se calculează funcţiile $\mu(t), \sigma(t), l(t)$. Schema logică este construită in ipoteza că *cererea nesatisfăcută se pierde* (vezi blocurile (13), (14), (15)), dar ca şi in modelul precedent, prin modificări (reamplasări) corespunzătoare ale acestor blocuri, se poate introduce şi ipoteza *că cererea nesatisfăcută se păstrează.* Pentru a obţine din modelul de simulare un plan de reaprovizionare pe perioada TT va trebui să includem in blocul (22) o instrucţiune care să afişeze $CLOCK, Q, P, T$. Repartiţia ratei cererii R poate fi normală $N(\mu(t), \sigma(t))$, caz in care P se calculează cu formula (7.43), sau poate fi $Exp(\lambda(t))$, caz in care P se calculează pe o cale asemănătoare, dar folosind faptul că cererea pe timpul de avans $R_{(L)}$ are o repartiţie $Erlang(0, \lambda, l)$. Repartiţia lui L poate fi de exemplu $Poisson(\lambda(t))$, $l = E[L] = \lambda(t)$. (Vezi exerciţiul **E 7.7**).

Acest ultim model simulează in mod mai realist decât primul, procesul de reaprovizionare al stocului.

Exerciţii.

E 7.1. Precizaţi câteva caracteristici ale modelului de aşteptare

$$Exp(\lambda)/Exp(\mu)/1 : (M, FIFO), \quad 0 < M < \infty$$

adică un model cu o staţie şi cu coadă finită.

Indicaţie. Conform formulelor (7.15'),(7.15") din $ 7.2.3 avem

$$\lambda_n = \lambda,\ n \geq 0, \quad \mu_n = \mu,\ 1 \leq n \leq M + 1.$$

deci după calcule, folosind (7.8') avem

$$p_n = \frac{\rho(1-\rho)}{1-\rho^{M+1}} = \frac{\rho(1-\rho)}{K},\ K = 1 - \rho^{M+1}.$$

De aici se deduce

$$E[N(t)] = \sum_{n=0}^{\infty} \frac{n\rho^n(1-\rho)}{1-\rho^{N+1}} = \frac{\rho(1+(M+1)(\rho-\rho^M)-\rho^{M+1})}{(1+\rho)(1+\rho^{M+1})}$$

$$E[WL] = \sum_{n=1}^{M} \frac{(n-1)\rho^n(1-\rho)}{1-\rho^{M+1}} = \frac{\rho^2[(M-1)\rho^m+1-M\rho^{M-1}]}{(1-\rho^{M+1})(1+\rho)}.$$

$$E[WT] = \frac{E[WL]}{\mu}, \quad E[NID] = p_0,\ E[TID] = \frac{1-\rho}{\lambda(1-\rho^{M+1})}.$$

E 7.2. Fie modelul de aşteptare

$$Exp(\lambda)/Exp(\mu)/N : (M, FIFO)$$

adică un model cu N staţii paralele identice şi cu coada finită $M, M > N$. Să se determine repartiţia numărului de clienţi in sistem şi câteva caracteristici ale modelului.

Indicaţie. Asemănător exerciţiului precedent avem

$$\lambda_n = \lambda, n \geq 0, \quad \mu_n = \begin{cases} n!\mu,\ daca\, 1 \leq n \leq M \\ N!\mu N^{n-N},\ daca\, N < n \leq N+M \end{cases}.$$

Mai departe se continuă calculele ca in exerciţiul precedent.

E 7.3. Modificaţi schema logică din modelul din Secţiunea *7.2* astfel incat clienţii posedă două priorităţi; prioritatea 1 este mai puternică decât prioritatea 2.

Indicaţie. In blocul (2) se consideră repartiţia lui AT ca o amestecare discretă de forma

$$f_{AT}(x) = p_1 f_1(x) + p_2 f_2(x), p_1, p_2 > 0, p_1 + p_2 = 1, f_i - densitati, i = 1, 2.$$

La generarea lui AT clientul sosit se depune in coadă Q_1 sau Q_2 corespunzătoare priorităţii. In blocul (4) se va simula timpul de servire conform priorităţii celei mai mari (adică 1), dacă $Q_1 \neq \emptyset$ sau priorităţii mai mici dacă $Q_1 = \emptyset$. Se pot calcula timpii de aşteptare in funcţie de priorităţi.

E 7.4. La fel ca in exerciţiul precedent, modificaţi schema logică din Fig. 7.3 (a modelului **7.3**) presupunând că sosirile provin dintr-o populaţie cu două priorităţi.

Indicaţie. Se vor folosi ideile din exerciţiul precedent: timpul de intersosire are o repatiţie dată de o amestecare discretă de două repartiţii.

E 7.5. Ce modificări ar trebui făcute in modelul **7.3** dacă duratele de serviciu ale staţiilor au repartiţii diferite.

Indicaţie. Se va folosi un vector $ST(I), 1 \leq I \leq N$ iar in blocurile unde se generează ST se va inlocui această instrucţiune cu $GENEREAZA\ \ ST(I)$.

E 7.6. Ce precauţii trebuie să se ia in modelele **7.3** dacă cererea R este discretă $Poisson(\lambda)$ şi L este $Binom(n,p)$?

Indicaţie. Toate variabilele din model vor fi discrete, inclusiv $CLOCK$. Nivelul de reaprovizionare se va calcula ţinând seama că $R_{(l)} \rightsquigarrow Poisson(l\lambda)$.

E 7.7. Să se determine nivelul de reaprovizionare P când rata cererii r are repartiţia $Exp(\lambda)$.

Indicaţie. Se foloseşte condiţia $Prob(R_{(L)} > P) = \alpha$, $\alpha - risc$. Deci P este soluţia ecuaţiei

$$1 - \alpha = \frac{\lambda^l}{\Gamma(l)} \int_0^P x^{l-1} e^{-\lambda x} dx$$

deoarece $R_{(L)} = r_1 + ... + r_l \rightsquigarrow Erlang(\lambda, l)$, $l = E[L]$.

Bibliography

[1] CHISMAN,J. (1989). *Introduction to Simulation and Modeling using GPSS/PC*, Minuteman Software. (Conţine şi discheta cu versiunea studenţească a GPSS/PC).

[2] ERMAKOV,E.S. (1974). *Metoda Monte Carlo şi probleme inrudite.* Editura Tehnică, Bucureşti. (Traducere din Limba Rusă).

[3] FISHMAN, G. S. (1978). *Principles of Discrete Event Simulation.* Wiley, New York.

[4] GORUNESCU, F., PRODAN, A. (2001). *Modelare stochastică şi simulare.* Editura Albastră, Cluj-Napoca.

[5] MORGAN,BYRON T. (1984). *Elements of Simulation.* Chapman & Hall, New York, London.

[6] ROBERT CHRISTIAN P., CASELLA, GEORGE. (1999). *Monte Carlo Statistical Methods.* Springer Verlag, New York, Berlin.

[7] ROSS, SHELDOM M. (1997) *Simulation. Second Edition.* Academic Press, San Diego, New York, London.

[8] SPRIET, JAN, VANSTEENKISTE GHISLAIN, C. (1982). *Computer ided Modeling and Simulation.* Academic Press, New York.

[9] VĂDUVA, I. (1972). "Metoda Monte Carlo". *Culegere tematică, Matematică-Mecanică, Vol. 1, Nr. 1, CIDI, p.131-188.*

[10] VĂDUVA, I. (1977). *Modele de Simulare cu Calculatorul.* Editura Tehnică, Bucureşti.

[11] VĂDUVA,I., ODĂGESCU,I., STOICA,M. (1983).*Simularea Proceselor Economice.* Editura Tehnică, Bucureşti.

[12] ZEIGLER,B. P. (1976). *Theory of Modeling and Simulation. (First Edition).* John Wiley and Sons, New York.

[13] ZEIGLER, B. P., PRAEHOFER, H. (2000). *Theory of Modeling and Simulation, (Second Edition).* Academic Press, New York.

[14] *** (1977). *Limbajul SIMUB. Manual de utilizare.* Centrul de Calcul Al Universităţii din Bucureşti. (litografiat).