**Modelare și Simulare**

**Proiect**

**Etapa 2**

**Student: Baciu Claudia-Iuliana**

**Grupa: 1310A**

**Profesor îndrumător: Petru Cașcaval**

**Număr proiect: 4**

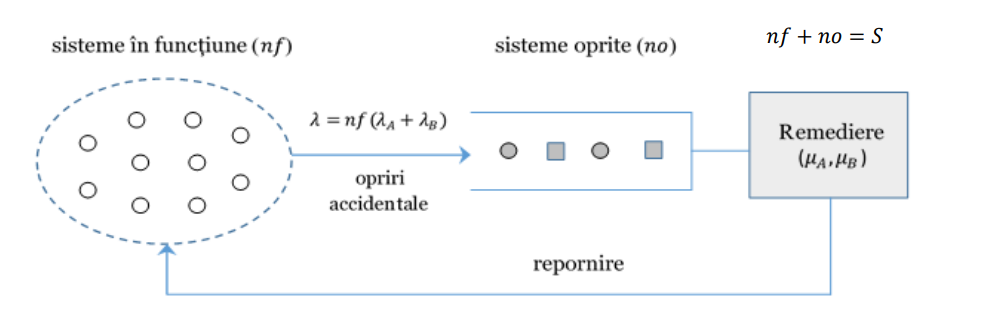
**An universitar: 2021-2022**

**Program de simulare pentru problema de interferenţă în care un muncitor deserveşte mai multe sisteme identice**

1. Analiza detaliată a problemei de interferenţă

➢ 𝑀𝑜𝑑𝑒𝑙𝑎𝑟𝑒𝑎 𝑝𝑟𝑜𝑏𝑙𝑒𝑚𝑒𝑖

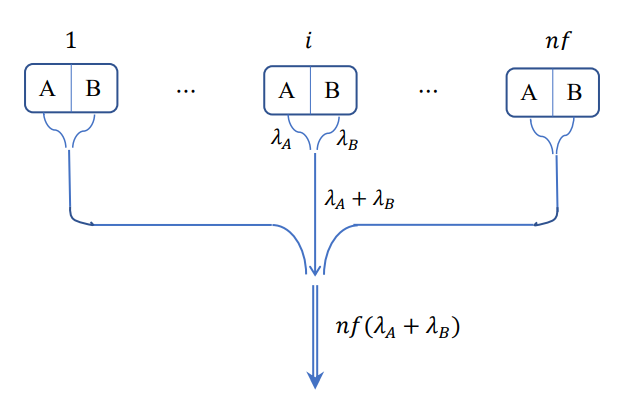
Ca model de simulare, sistemul în ansamblu compus din cele 𝑆 maşini automate şi muncitorul de deservire poate fi privit ca un sistem de servire cu o staţie, aşa cum este ilustrat în figura următoare.



Model de simulare pentru problema de interferenţă studiată.

➢ 𝐹𝑙𝑢𝑥𝑢𝑙 𝑜𝑝𝑟𝑖𝑟𝑖𝑙𝑜𝑟

De la prima etapă a rezultat că cele două variabile aleatoare primare 𝑇𝑓𝐴 şi 𝑇𝑓𝐵 au repartiţii exponenţial negative, de parametru 𝜆𝐴 şi respectiv, 𝜆𝐵. Orice întrerupere accidentală duce la oprirea maşinii pentru remediere. Prin urmare, timpul de funcţionare a unei maşini până apare o oprire accidentală este 𝑇𝑓 = 𝑚𝑖𝑛 {𝑇𝑓𝐴, 𝑇𝑓𝐵}. Pe baza proprietăţii studiate la curs rezultă că şi variabila aleatoare 𝑇𝑓 are tot o repartiţie exponenţial negativă de parametru 𝜆𝐴 + 𝜆𝐵. Cu alte cuvinte, prin suprapunerea efectelor celor două cauze independente de întrerupere accidentală rezultă pentru o maşină un flux al opririlor de tip Poissonian cu o rată medie egală cu 𝜆𝐴 + 𝜆𝐵. Dar, prin reunirea mai multor fluxuri Poissoniene independente rezultă tot un flux Poissonian. Aşadar, atunci când sunt 𝑛𝑓 sisteme în funcţiune rata medie a opririlor este 𝜆 = 𝑛𝑓(𝜆𝐴 + 𝜆𝐵), iar durata dintre două opriri consecutive are o repartiţie exponenţial negativă de acelaşi parametru 𝜆.



Fluxul opririlor – flux Poissonian rezultat din reunirea mai multor fluxuri Poissoniene

De remarcat că cele 𝑛𝑓 maşini nu sunt puse în funcţiune în acelaşi timp. Dacă ţinem cont de proprietatea variabilei aleatoare exponenţial negative 𝑇𝑓 de a fi “fără memorie” acest aspect nu mai are relevanţă. Prin urmare, rezultă că pentru cele 𝑛𝑓 maşini în funcţiune durata dintre două opriri consecutive are într-adevăr o repartiţie exponenţial negativă de parametru 𝜆 = 𝑛𝑓(𝜆𝐴 + 𝜆𝐵). În aceste condiţii fluxul opririlor este uşor de simulat apelând doar funcţia de generate 𝑔𝑒𝑛𝐸𝑥𝑝(𝜆).

➢ 𝐷𝑒𝑠𝑒𝑟𝑣𝑖𝑟𝑒𝑎

Timpul de remediere a unui sistem oprit depinde de tipul modulului afectat de întreruperea accidentală, 𝐴 sau 𝐵. Fie 𝑝𝐴 şi 𝑝𝐵 = 1 − 𝑝𝐴 probabilitatea ca la un sistem oprit modulul care necesită remediere să fie de tip 𝐴 şi respectiv, de tip 𝐵.

𝑝𝐴 = 𝜆𝐴⁄(𝜆𝐴 + 𝜆𝐵) și 𝑝𝐵 = 𝜆𝐵⁄(𝜆𝐴 + 𝜆𝐵)

De observat că

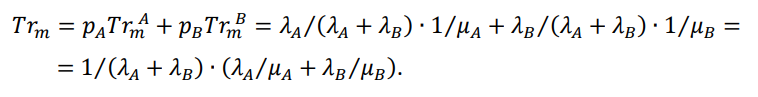
𝑝𝐴⁄𝑝𝐵 = 𝜆𝐴⁄𝜆𝐵

Timpul mediu de remediere a unui sistem oprit se exprimă cu relaţia: 

Dar variabilele aleatoare 𝑇𝑟𝐴 şi 𝑇𝑟𝐵 au repartiţii exponenţial negative, de parametru 𝜇𝐴 şi respectiv, 𝜇𝐵. Ca urmare,



Pentru timpul mediu de remediere a unui sistem oprit rezultă relaţia de calcul:



2. Algoritmul de simulare

Fie 𝑆 numărul de sisteme identice deservite de muncitor. În funcție de valoarea lui 𝑆 trebuie să se determine disponibilitatea sistemelor și gradul de ocupare a muncitorului de deservire. Semnificația variabilelor folosite în algoritmul de simulare este următoarea:

• 𝑐𝑒𝑎𝑠,𝐷𝑆 – ceasul și durata simulării

• 𝑛𝑓 și 𝑛𝑜 – variabile de stare; 𝑛𝑓 + 𝑛𝑜 = 𝑆

• 𝑇𝑝𝑜 – timpul până la o nouă oprire a unui sistem în funcțiune; variabila nu are semnificație când 𝑛𝑓 = 0

• 𝑇𝑟 – timpul necesar pentru remedierea sistemului în curs; variabila nu are semnificație atunci când 𝑛𝑜 = 0.

• 𝑆𝑇𝑓 – statistică cu suma timpilor de funcționare pentru cele 𝑆 sisteme în perioada de monitorizare

• 𝑆𝑇𝑟 – statistică cu suma timpilor de remediere. La sfârșitul simulării mărimile de interes se determină cu relațiile:

𝐷 = 𝑆𝑇𝑓⁄(𝐷𝑆 ∙ S) ∗ 100 (%)

𝑂 = 𝑆𝑇𝑟⁄𝐷𝑆 ∗ 100 (%)

𝜆𝐴, 𝜆𝐵, 𝜇𝐴, 𝜇𝐵, 𝑆, 𝐷𝑆

,

𝑆𝑇𝑓 = 0, 𝑆𝑇𝑟 = 0

c𝑒𝑎𝑠 = 0, 𝑛𝑓 = 𝑆, 𝑛𝑜 = 0

T𝑝𝑜 = 𝑔𝑒𝑛𝐸𝑥𝑝(𝑛𝑓(𝜆𝐴,+𝜆𝐵))

Determinare eveniment următor

(𝑛𝑓, 𝑛𝑜, 𝑇𝑝𝑜, 𝑇𝑟)

𝑇𝑟 = 𝑔𝑒𝑛𝑇𝑟(), S𝑇𝑟 += 𝑇𝑟

𝑛𝑜 = 1

ce𝑎𝑠 += 𝑇𝑝o

ce𝑎𝑠 += 𝑇𝑟

Algoritm de simulare pentru problema de interferenţă a sistemelor 1

DA 1

c𝑒𝑎𝑠 <𝐷S

*i*𝑓(𝑛𝑓 > 0) 𝑇𝑝𝑜 = 𝑔𝑒𝑛𝐸𝑥𝑝(𝑛𝑓(𝜆𝐴 + 𝜆𝐵))

𝑂 = 𝑆𝑇𝑟⁄𝐷𝑆 ∙ 100 (%)

𝐷= 𝑆𝑇𝑓⁄(𝐷𝑆 ∙ 𝑆) ∙ 100 (%)

𝑇𝑟 = 𝑔𝑒𝑛𝑇𝑟(), S𝑇𝑟 += 𝑇𝑟

𝑛𝑓 + +, 𝑛𝑜 − −

𝑛𝑜 > 0

S𝑇𝑓+= 𝑛𝑓 ∙ 𝑇𝑟

𝑛𝑓 − −, 𝑛𝑜 + +

S𝑇𝑓+= 𝑛𝑓 ∙ 𝑇𝑝𝑜

T𝑓(𝑛𝑜 > 0) 𝑇𝑟 -= 𝑇𝑝o

3. Verificarea programului de simulare

Programul de simulare se verifică în mai multe etape, făcând mai întâi verificări paţiale şi apoi verificări privind rezultatul obţinut. Pentru verificarea rezultatului, pot fi avute în vedere teste calitative sau cantitative. În continuare se prezintă modalitatea de verificare a rezultatelor obţinute pentru cazul cel mai simplu cu 𝑆 = 1.

În această situaţie perioadele de funcţionare alternează cu cele de remediere întrucât nu apare fenomenul de interferenţă. Notând cu 𝑇𝑓𝑚 timpul mediu de funcţionare până la o oprire accidentală şi cu 𝑇𝑟𝑚 timpul mediu de remediere, disponibilitatea se exprimă cu relaţia:

𝐷 = 𝑇𝑓𝑚⁄(𝑇𝑓𝑚 + 𝑇𝑟𝑚) ∙ 100 (%)

Dar 𝑇𝑓𝑚=1/(𝜆𝐴 + 𝜆𝐵 ) iar 𝑇𝑟𝑚 = 1/ (𝜆𝐴 + 𝜆𝐵 ) (𝜆𝐴⁄𝜇𝐴 + 𝜆𝐵⁄𝜇𝐵 ).

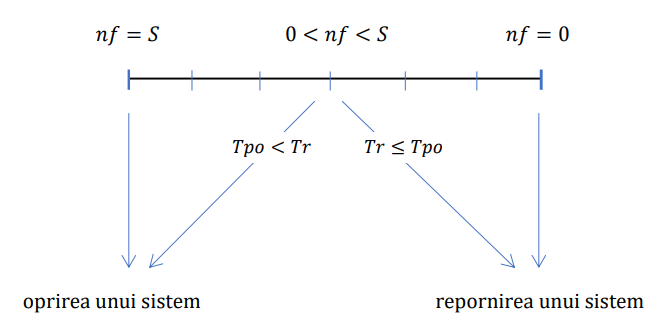
Prin urmare, rezultă relaţia:

𝐷 = 1⁄(1 + 𝜆𝐴⁄𝜇𝐴 + 𝜆𝐵⁄𝜇𝐵) ∙ 100 (%)

Precizări suplimentare

1) Cu privire la determinarea evenimentului următor

Evenimentul următor (care poate fi oprirea a unui sistem în funcţiune sau repornirea sistemului în curs de remediere) se stabileşte pe baza variabilei de stare 𝑛𝑓 şi a variabilelor cu semnificaţie de timp 𝑇𝑝𝑜 şi 𝑇𝑟. Reamintim că variabila 𝑇𝑝𝑜 nu are semnificaţie când 𝑛𝑓 = 0, iar variabila 𝑇𝑟 nu are semnificaţie când 𝑛𝑓 = 𝑆.



Determinarea evenimentului următor

2) Cu privire la durata de simulare

Durata de simulare 𝐷𝑆 trebuie să permită ca numărul de opriri tratate 𝑁𝑜 să fie de ordinul 105 − 107 . Să considerăm deocamdată un număr impus de opriri 𝑁𝑂 = 106 . Numărul de opriri care apar în perioada de simulare [0, 𝐷𝑆] poate fi estimat cu relaţia:

𝑁𝑜 ≃ 𝐷𝑆 ∙ 𝐷 100 ∙ 𝑆 ∙ (𝜆𝐴 + 𝜆𝐵) = 𝑆𝑇𝑓 ∙ (𝜆𝐴 + 𝜆𝐵)

Prin urmare, durata de simulare 𝐷𝑆 care să asigure apariţia numărului impus de opriri 𝑁𝑂 se deduce cu relaţia:



Observaţii:

- Durata de simulare nu este aceeaşi pentru toate experimentele: 𝐷𝑆 are o valoare mai mare pentru pentru un singur sistem şi se reduce pe măsură ce numărul de sisteme 𝑆 (monitorizate în paralel) creşte.

- Durata de simulare depinde de frecvenţa opririlor accidentale (mai precis, de valorile parametrilor 𝜆𝐴 şi 𝜆𝐵).

- Durata de simulare care să asigure precizia impusă prin 𝑁𝑂 nu poate fi determinată de la început întrucât depinde de disponibilitatea 𝐷, tocmai necunoscuta problemei. Prin urmare, 𝐷𝑆 trebuie să se determine în mod adaptiv pe măsură ce se obţin estimări tot mai bune pentru 𝐷.

4.Testarea programului

a) Verificări parţiale, preliminare

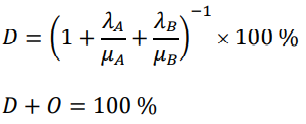
Se completează programul pentru a înregistra şi numărul de remedieri efectuate de muncitor (𝑁𝑟). La sfârşitul simulării trebuie să se verifice:

o 𝑁𝑟 ≃ 𝑁𝑜 – se verifică astfel secvenţa pentru determinarea evenimentului următor

o 𝑆𝑇𝑟⁄𝑁𝑟 ≃ 𝑇𝑟𝑚 = 1 /(𝜆𝐴+𝜆𝐵) ∙ ( (𝜆𝐴/ 𝜇𝐴) +( 𝜆𝐵/ 𝜇𝐵) ) – se verifică astfel funcţia de generare a timpilor de remediere 𝑔𝑒𝑛𝑇𝑟()

b) Verificări cantitative privind rezultatele simulării

o 𝑆 = 1 – se verifică rezultatele simulării cu relaţiile:

****

o 𝑆 ≥ 2 – se verifică indirect disponibilitatea obţinută (𝐷), comparând numărul de opriri înregistrate în perioada de simulare (𝑁𝑂) cu numărul de opriri estimat cu relaţia (16), în care intervine şi 𝐷.

c) Verificări calitative privind gradul de ocupare a muncitorului de deservire

În ceea ce priveşte gradul de ocupare a muncitorului de deservire, ne mulţumim deocamdată cu o verificare calitativă. Să presupunem că pentru 𝑆 = 2 disponibilitatea sistemelor nu scade semnificativ faţă de cazul iniţial, cu 𝑆 = 1. Pentru a obţine cam acelaşi nivel de disponibilitate, muncitorul va interveni la un număr aproape dublu de opriri şi, ca urmare, va munci aproape de două ori mai mult. Generalizând, cât timp disponibilitatea sistemelor nu scade semnificativ faţă de valoarea de la cazul iniţial, gradul de ocupare 𝑂 trebuie să fie aproape de 𝑆 ori mai mare faţă de valoarea de la acel caz de referinţă.

**5.Codul sursă**

Fișierul Header.h

#pragma once

#include<iostream>

#include<cmath>

using namespace std;

#define lim\_max 1000000

//definirea parametrilor variabilelor aleatoare

#define lamA 0.2105

#define lamB 0.1626

#define miuA 3.8533

#define miuB 3.4218

#define pA lamA/(lamA+lamB)

#define pB lamB/(lamA+lamB)

#define D\_t 100/(1+lamA/miuA+lamB/miuB)//val teoretica a gradului de disponibilitate pt un singur sistem

//definitii functii de generare valori de selectie

double genExp(double lambda\_1);

double genGauss(double m, double sigma);

double genTR();//generator timp remediere simulare 1

double genTR2();//generator timp remediere simulare 2

Fișierul Source.cpp

#include"Header.h"

using namespace std;

double genExp(double lambda\_1)

{

double u = 1;

//log(0)=-inf, deci u<1

u = (double)rand() / (RAND\_MAX + 1);

double x = (-1 / lambda\_1) \* log(1 - u);

return x;

}

double genGauss(double m, double sigma)

{

double s = 0;

for (int i = 0; i < 12; i++) {

s += (double)rand() / RAND\_MAX;

}

return m + sigma \* (s - 6);

}

double genTR() // functia de generare a timpilor de remediere

{

double U, Tr;

U = (double)rand() / ((double)RAND\_MAX + 1);

if (U < pA)

{

Tr = genExp(miuA);

}

else

{

Tr = genExp(miuB);

}

return Tr;

}

double genTR2() //geneartor valori aleatorii cu rep normala

{

double U, Tr;

U = (double)rand() / ((double)RAND\_MAX);

if (U < pA)

{

Tr = genGauss(1 / miuA, 1 / (4 \* miuA));

}

else

{

Tr = genGauss(1 / miuB, 1 / (4 \* miuB));

}

return Tr;

}

int main()

{

cout << "Numarul sistemelor (s):";

int s;//nr sisteme

cin >> s;

double DS = 0;//durata simularii

double Tr = 0;//timp remediere

double STf = 0, STr = 0;//suma timp functionare/remediere

double ceas = 0;//ceasul simularii

int nf = s;//nr sisteme fuctionare

int no = 0;//nr sisteme oprite

int No = 0;

int Nr = 0;//contor numar opriri/remedieri

//generare valoare corespunzatoare primei opriri

double TP0 = genExp(nf \* (lamA + lamB));

do {

if ((nf == s) || ((nf > 0) && (TP0 < Tr)))

{

//o oprire accidentala

No++;

ceas += TP0;//actualizare ceas simulare la momentul ultimului eveniment

if (no > 0)

Tr -= TP0;//reducerea timpului de remediere odata cu avansul variabilei ceas

STf += nf \* TP0;//actualizare statistici

nf--;//trecerea la o noua stare a sistemului

no++;//no++ diferit de No++ !!

if (no == 1)

{

//genTR2 pentru repartitia normal

//genTR pentru reparitia exponential negativa

Tr = genTR();//muncitorul e obligat sa intervina imediat la oprirea primului sistem

STr += Tr;//statistica

}

}

else {

//o remediere terminata

Nr++; //incrementam numarul de remedieri

ceas += Tr;//actualizare ceas la momentul ultimului eveniment

if (nf > 0)

TP0 -= Tr;

//determinare exacta a mom opririi relativ la var ceas actualizata

STf += nf \* Tr;

nf++;

no--;

if (no > 0)

{

Tr = genTR();

STr += Tr;//trecerea muncitorului la urmatorul sistem blocat

}

}

if (nf > 0)

{

TP0 = genExp(nf \* (lamA + lamB));

}

} while (No < 9 \* lim\_max);

//verificarea preliminara a determinarii corecte a ev urmator

cout << "Verificari preliminare:\n";

cout << "\tNumarul total al opririlor: " << No << endl;

cout << "\tSuma dintre nr sistemelor remediate si a celor ramase oprite la oprirea simularii: " << Nr + no << endl;

cout << "\tSTr/Nr= " << double(STr / Nr) << " si valoarea teoretica=" << (double)(1 / (lamA + lamB) \* (lamA / miuA + lamB / miuB)) << endl << endl;

DS = ceas;//durata simularii este egala cu var ceas

double D = (double)STf / (s \* DS) \* 100;//(%)-afisat in procente -->gradul de disponibilitate a sistemelor

double O = (double)STr / DS \* 100;//(%)-afisat in procente -->gradul de ocupare al muncitorului

if (s == 1)

{

cout << "Valoare teoretica a gradului de disponibilitate: " << D\_t << " % " << endl;

}

cout << "Gradul de disponibilitate a sistemelor: " << D << " % " << endl;

cout << "Gradul de ocupare al muncitorului: " << O << " % " << endl;

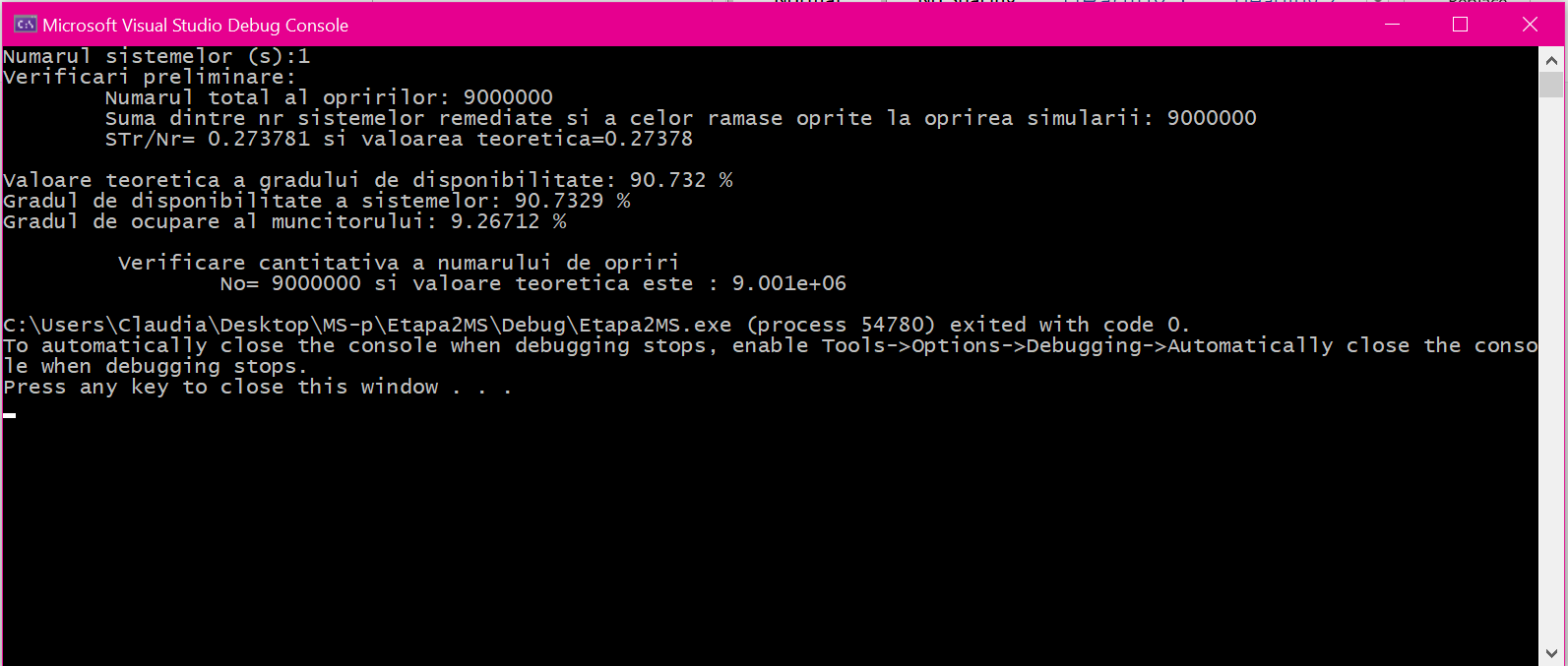
cout << "\n\t Verificare cantitativa a numarului de opriri\n";

cout << "\t\t No= " << No << " si valoare teoretica este : " << DS \* s \* D / 100 \* (lamA + lamB) << endl;

}

➢ *Rezultatele simulării pentru repartiția exponențial negativă*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| D(%) | 90.7329 | 89.9575 | 89.0684 | 88.0638 | 86.879 | 85.5273 | 83.9468 | 82.1922 | 80.1225 | 77.8312 | 75.2591 | 72.4448 | 69.4846 |
| O(%) | 9.26712 | 18.3777 | 27.3011 | 35.9515 | 44.3774 | 52.4244 | 60.0763 | 67.1312 | 73.7069 | 79.492 | 84.5656 | 88.8376 | 92.2108 |



**Temă suplimentară:**

Sa se reia analiza disponibilităţii sistemelor şi a gradului de ocupare a muncitorului de deservire în condiţiile în care timpul de remediere a unui modul afectat de întrerupere ar avea o repartiţie normală. Pentru medie se păstrează aceeaşi valoare, însă pentru dispersie se impune o valoare mult mai mică.

​𝑚𝐴 =1/𝜇𝐴,​ 𝑚𝐵 =1/𝜇𝐵, ​𝜎𝐴 =1/4𝜇𝐴,​ 𝜎𝐵 =1/4𝜇𝐵.

Ţinând cont că valoare medie este 𝑚 =1𝜇, că valorile sunt pozitive şi că repartiţia normală este una simetrică, rezultă că valoarea maximă nu poate depăşi limita de 2/𝜇. Cum marea majoritate a valorilor repartiţiei normale se încadrează în intervalul (𝑚−4𝜎, 𝑚 +4𝜎), impunând condiţia ca 4𝜎 =1/𝜇, rezultă că 𝜎 =1/4𝜇. De remarcat că dispersia valorilor este mult mai redusă în acest caz.

Această modificare se reflectă în programul de simulare folosind pentru variabila Tr următoarea funcție generator:

double genGauss(double m, double sigma)

{

double S = 0;

for (int i = 1; i <= 12; ++i)

S += (double)rand() / RAND\_MAX;

return m + sigma \* (S - 6);

}

double genTR2() //geneartor valori aleatorii cu rep normala

{

double U, Tr;

U = (double)rand() / ((double)RAND\_MAX + 1);

if (U < pA)

{

Tr = genGauss(1 / miuA, 1 / (4 \* miuA));

}

else

{

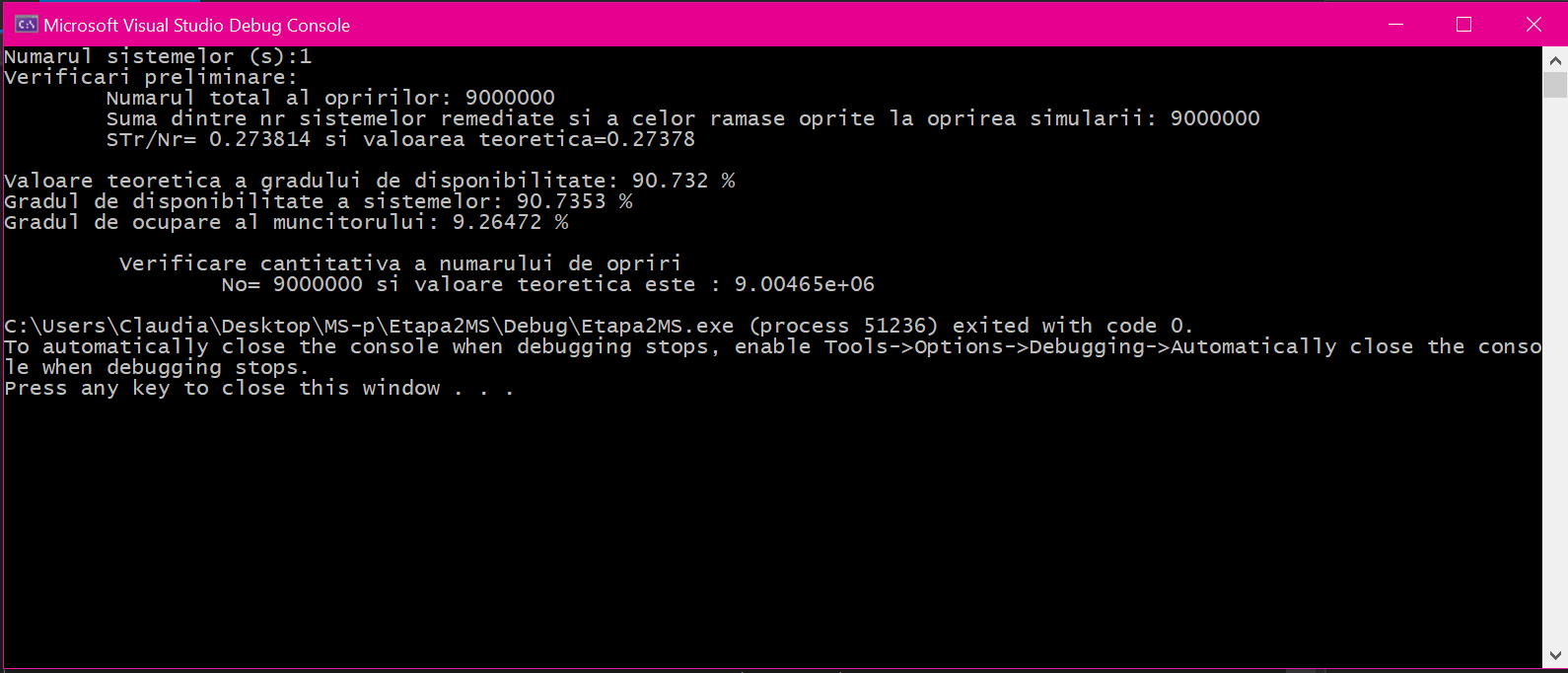
Tr = genGauss(1 / miuB, 1 / (4 \* miuB));

}

return Tr;

}

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| D(%) | 90.7353 | 90.2902 | 89.7716 | 89.1592 | 88.4052 | 87.4849 | 86.3742 | 84.971 | 83.2433 | 81.1189 | 78.5756 | 75.5359 |
| O(%) | 9.26472 | 18.4519 | 27.5161 | 36.4123 | 45.1412 | 53.6365 | 61.7214 | 69.4417 | 76.543 | 82.8854 | 88.2457 | 92.565 |

****

**Concluzii:**

Pe măsură ce numărul de sisteme de servire (S) crește, disponibilitatea acestora scade,iar gradul de ocupare crește, ajungând ca, la un număr de 11 sisteme, să se apropie de 90% .

La presupunerea că timpul de remediere ar fi caracterizat de o repartiție normală, cu parametrii caracterizați mai sus se observă o creștere sensibilă a gradului de disponibilitate al sistemului, cât si al gradului de ocupare al muncitorului de deservire(dacă în cazul anterior O depășea 90% pentru 13 sisteme arondate, în cazul de față acest lucru se întâmplă de la S=12).