Modelare și Simulare

Proiect

Etapa 2

Student: Baciu Claudia-Iuliana

Grupa: 1310A

Profesor îndrumător: Petru Cașcaval

Număr proiect: 4

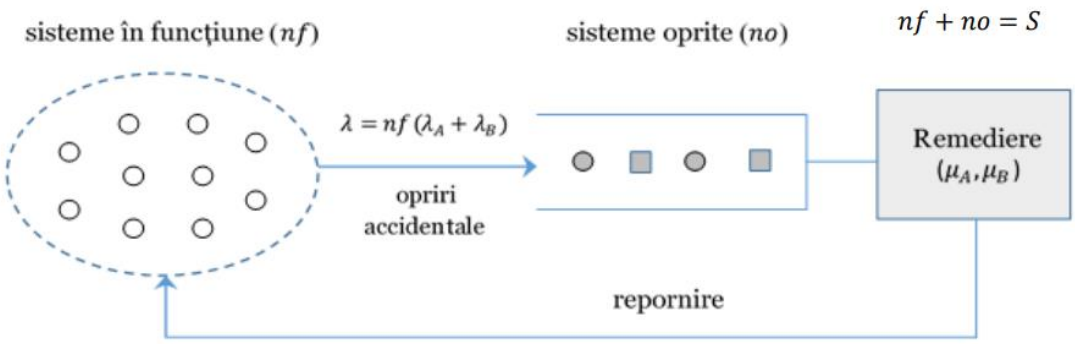
An universitar: 2021-2022

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*

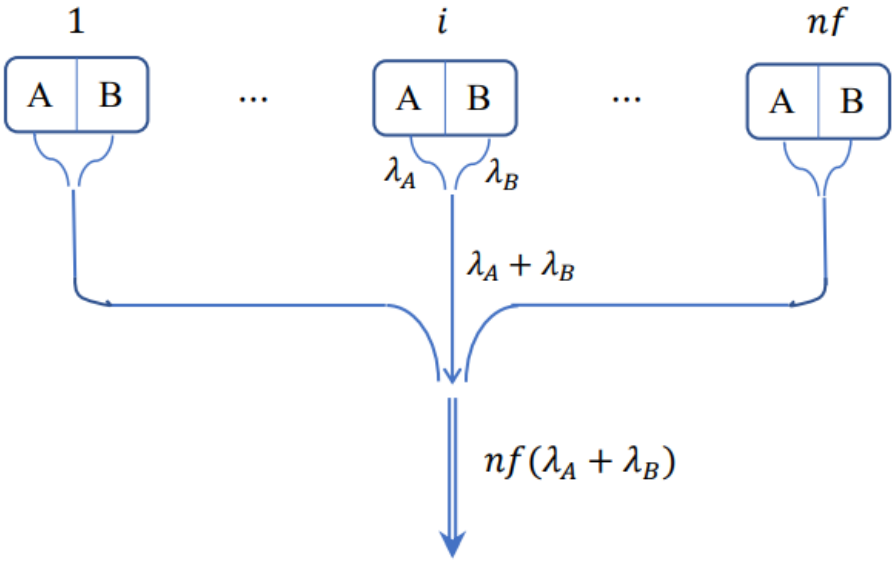
Ca model de simulare, sistemul în ansamblu compus din cele S maşini automate şi muncitorul de deservire poate fi privit ca un sistem de servire cu o staţie, aşa cum este ilustrat în figura următoare.



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 $\lambda A + \lambda B$. 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 $\lambda A + \lambda B$. 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 $\lambda = nf(\lambda A + \lambda B)$, 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 $\lambda = nf(\lambda A + \lambda B)$. În aceste condiţii fluxul opririlor este uşor de simulat apelând doar funcţia de generate $genExp(\lambda)$.

➢ $Deservirea$

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.

$pA = \lambda A/(\lambda A + \lambda B)$ şi $pB = \lambda B/(\lambda A + \lambda B)$

De observat că

$pA/pB = \lambda A/\lambda B$

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

$$Tr_m = p_A Tr_m^A + p_B Tr_m^B$$

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

$$Tr_m^A = 1/\mu_A \text{ şi } Tr_m^B = 1/\mu_B$$

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

$$Tr_m = p_A Tr_m^A + p_B Tr_m^B = \lambda_A/(\lambda_A + \lambda_B) \cdot 1/\mu_A + \lambda_B/(\lambda_A + \lambda_B) \cdot 1/\mu_B =$$
$$= 1/(\lambda_A + \lambda_B) \cdot (\lambda_A/\mu_A + \lambda_B/\mu_B).$$

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 $nf = 0$

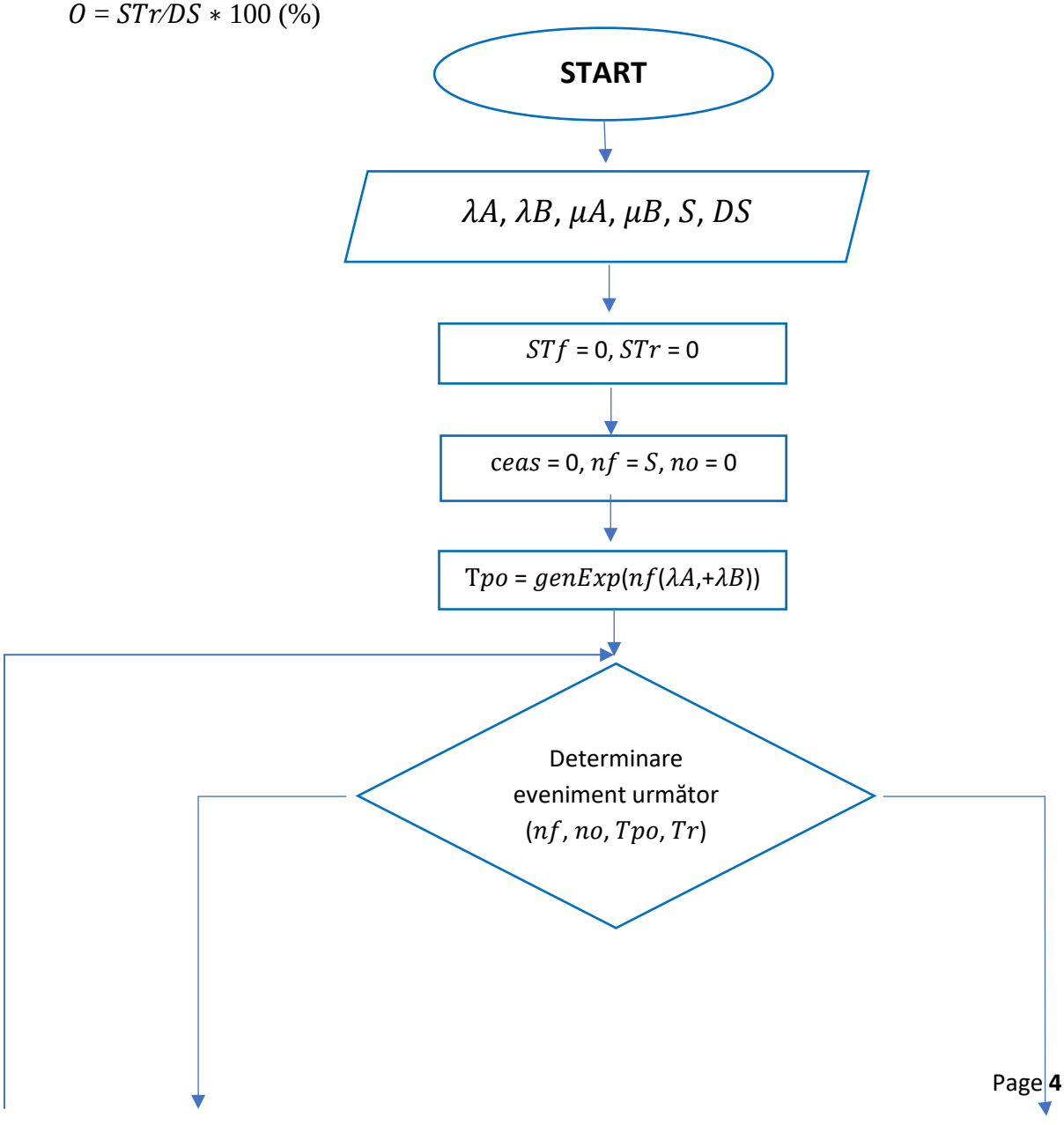
• Tr – timpul necesar pentru remedierea sistemului în curs; variabila nu are semnificație atunci când $no = 0$.

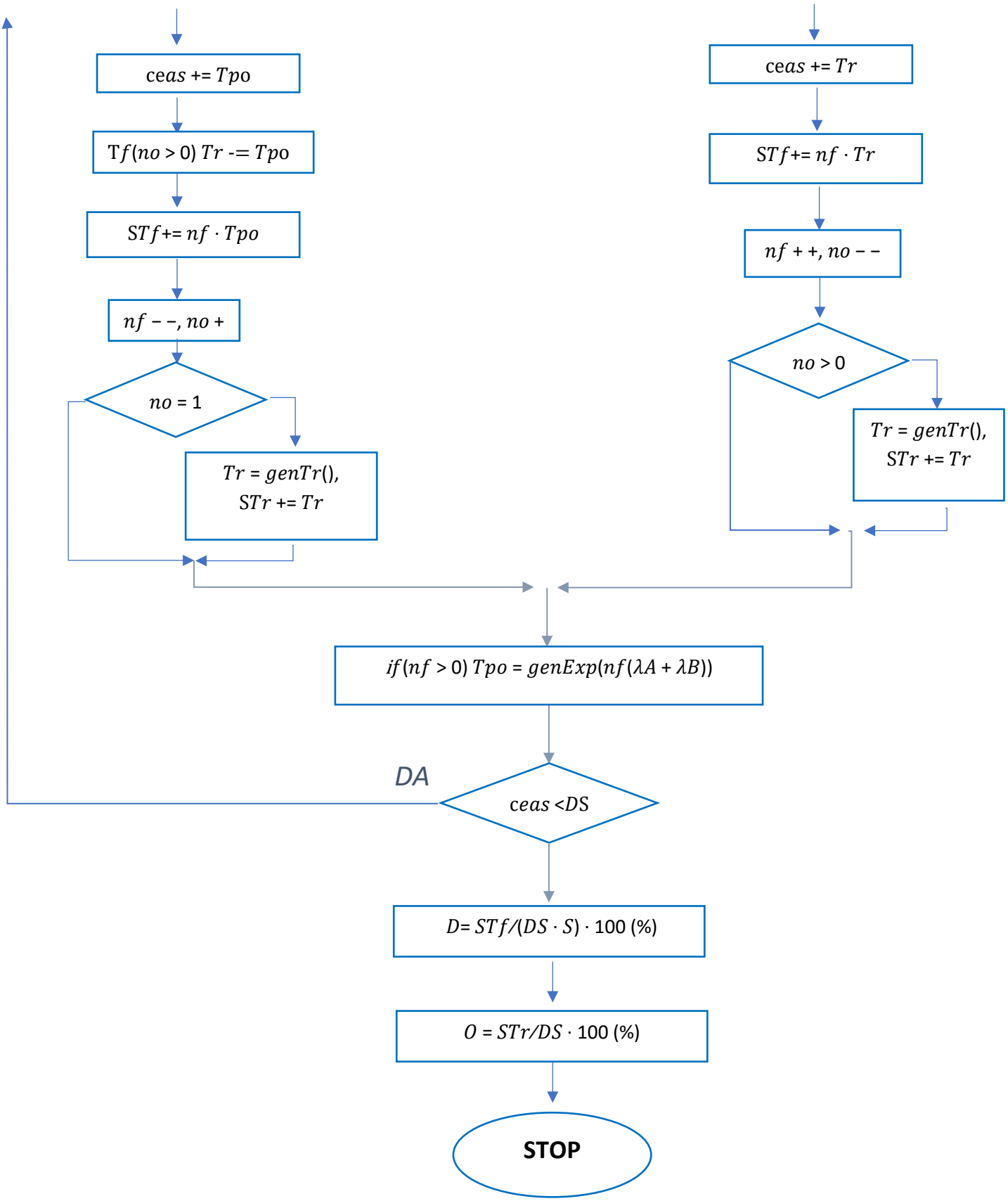
• STf – statistică cu suma timpilor de funcționare pentru cele S sisteme în perioada de monitorizare

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

$D = STf/(DS \cdot S) * 100$ (%)

$O = STr/DS * 100$ (%)





Algoritm de simulare pentru problema de interferenţă a sistemelor

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:

$D = Tfm/(Tfm + Trm) \cdot 100$ (%)

Dar $Tfm = 1/(\lambda A + \lambda B)$ iar $Trm = 1/(\lambda A + \lambda B)(\lambda A/\mu A + \lambda B/\mu B)$.

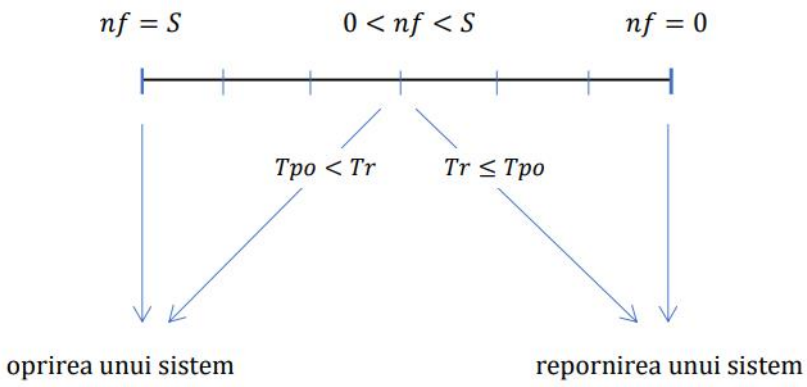
Prin urmare, rezultă relaţia:

$D = 1/(1 + \lambda A/\mu A + \lambda B/\mu B) \cdot 100$ (%)

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:

$No \simeq DS \cdot D\ 100 \cdot S \cdot (\lambda A + \lambda B) = STf \cdot (\lambda A + \lambda B)$

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

$$DS \simeq NO \cdot \frac{100}{D} \cdot \frac{1}{S} \cdot \frac{1}{\lambda_A + \lambda_B}$$

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 $Nr \simeq No$ – se verifică astfel secvenţa pentru determinarea evenimentului următor

o $STr/Nr \simeq Trm = 1/(\lambda A+\lambda B) \cdot ((\lambda A/\mu A) + (\lambda B/\mu B))$ – 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:

$$D = \left(1 + \frac{\lambda_A}{\mu_A} + \frac{\lambda_B}{\mu_B}\right)^{-1} \times 100\ \%$$

$$D + O = 100\ \%$$

o $S \geq 2$ – 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
```

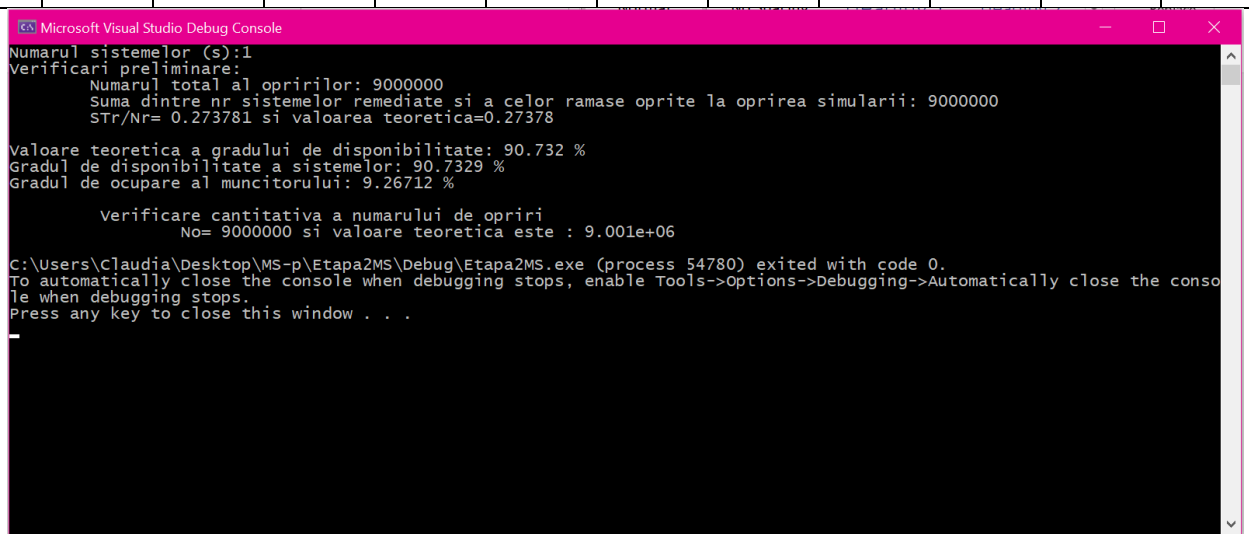
```
{
	double U, Tr;
	U = (double)rand() / ((double)RAND_MAX);
	if (U < pA)
	{
		Tr = genGauss(1 / miuA, 1 / (4 * miuA));
	}
	else
	{
		Tr = genGauss(1 / miuB, 1 / (4 * miuB));
	}
	return Tr;
}
int main()
{
	cout << "Numarul sistemelor (s):";
	int s;//nr sisteme
	cin >> s;
	double DS = 0;//durata simularii
	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 repartiţia exponenţial 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.

$mA = 1/\mu A$, $mB = 1/\mu B$, $\sigma A = 1/4\mu A$, $\sigma B = 1/4\mu B$.

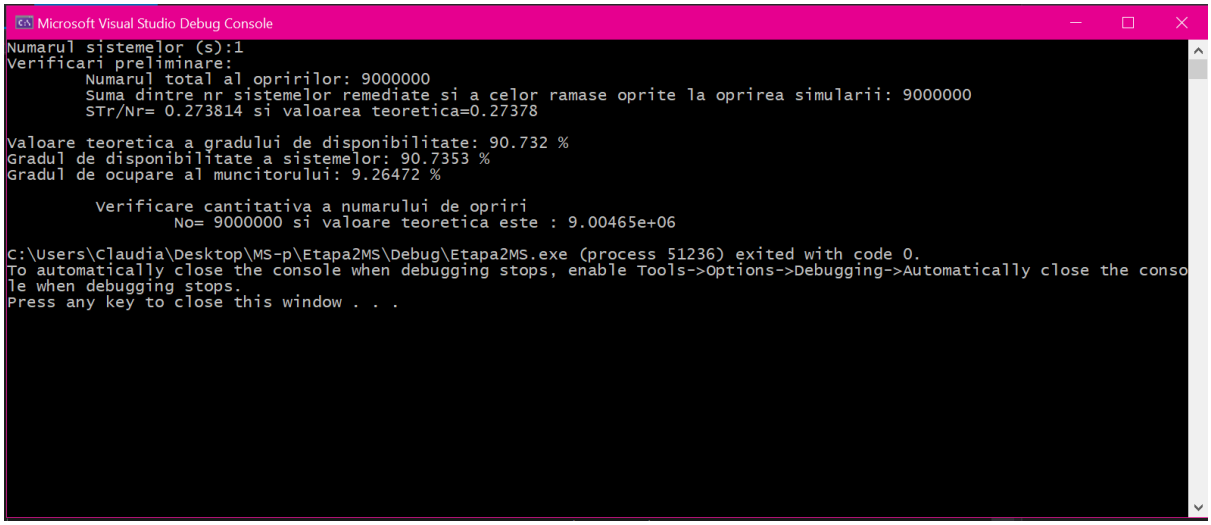
Ţinând cont că valoare medie este $m = 1\mu$, că valorile sunt pozitive şi că repartiţia normală este una simetrică, rezultă că valoarea maximă nu poate depăşi limita de $2/\mu$. Cum marea majoritate a valorilor repartiţiei normale se încadrează în intervalul $(m-4\sigma, m+4\sigma)$, impunând condiţia ca $4\sigma = 1/\mu$, rezultă că $\sigma = 1/4\mu$. 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).