# Modelare și Simulare

# Proiect

# Etapa 3

**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 condiţiile în care sistemele sunt prevăzute cu modul de rezervă

## 1.Stabilirea modulului la care se adaugă rezerva

Scăderea disponibilităţii este o consecinţă a suprapunerii efectelor celor două cauze independente de întrerupere accidentală. Pentru a obţine o disponibilitate mai ridicată, se adaugă o rezervă identică cu modulul de bază acolo unde întreruperile accidentale afectează într-o măsură mai mare disponibilitatea sistemului. Pentru stabilirea modulului la care se adaugă o rezervă trebuie avută în vedere atât frecvenţa întreruperilor cât şi timpul mediu de remediere. Aşa cum se cunoaşte, pentru modelul primar în care nu intervine fenomenul de interferenţă, disponibilitatea este dată de relaţia

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

Relaţia de calcul evidenţiază explicit influenţa fiecărei cauze de întrerupere accidentală asupra disponibilităţii sistemului. Pe baza acestei relaţii se deduce că în ceea ce priveşte disponibilitatea este mai bine ca rezerva să se adauge la modulul pentru care raportul $\lambda/\mu$ este mai mare.

## 2. Algoritmul de simulare

➢ O analiză preliminară

La un modul prevăzut cu rezervă, în cazul unei întreruperi accidentale rezerva înlocuieşte modulul afectat asigurându-se astfel continuarea funcţionării sistemului. Este de dorit ca remedierea modulului defect să se facă înaintea apariţiei unei noi întreruperi pentru a readuce sistemul la starea de toleranţă de la început. Ca urmare, la această problemă de interferenţă nu orice întrerupere duce şi la oprirea sistemului. De asemenea, nu orice remediere permite repornirea sistemului. În aceste condiţii, algoritmul de simulare trebuie să urmărească explicit schimbările de stare din sistem, pentru ca la apariţia unei întreruperi să se poată stabili dacă sistemul se opreşte sau îşi continuă funcţionarea cu modulul de rezervă, şi dacă după remedierea unui modul afectat de întrerupere sistemul poate fi repus în funcţiune sau nu.

➢ Semnificaţia mărimilor folosie în algoritmul de simulare:

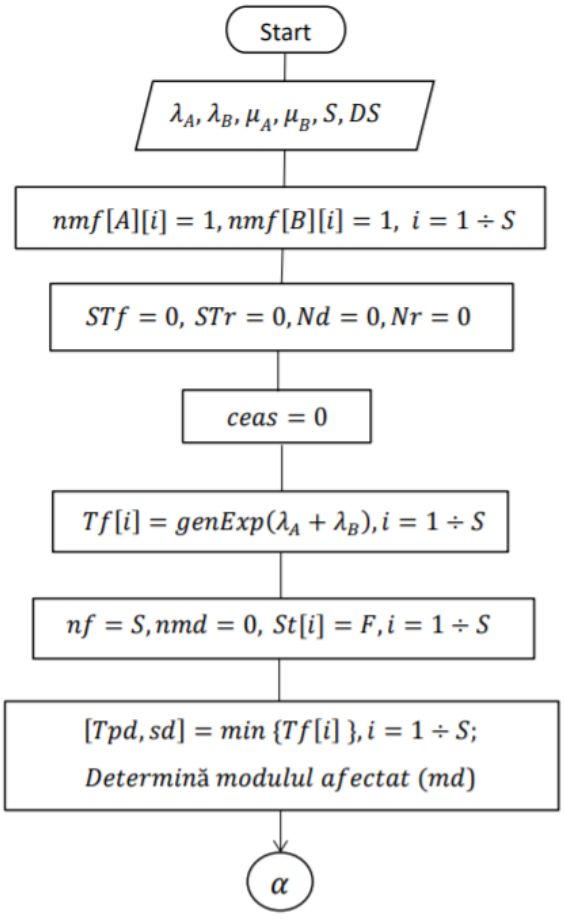
• $St[i]$, $i = 1 \div S$ – starea sistemului $i$: în funcţiune ($F$) sau oprit ($O$);
• $nmf[m][i]$, $m = A$ sau $B$, $i = 1 \div S$ – numărul modulelor funcţionale de tip $A$ sau $B$ la sistemul $i$ (la iniţializare se specifică astfel modulul care este prevăzut cu rezervă);
• $nf$ − numărul de sisteme în funcţiune la un moment dat;
• $nmd$ − numărul de module afectate de întrerupere care necesită remediere, indiferent de tipul lor sau de sistemele de care aparţin;
• $Tf[i]$ − timpul de funcţionare până la prima întrerupere accidentală la sistemul $i$, $i = 1 \div S$; locaţia $Tf[i]$ nu are semnificaţie când sistemul $i$ este oprit ($St[i] = O$);

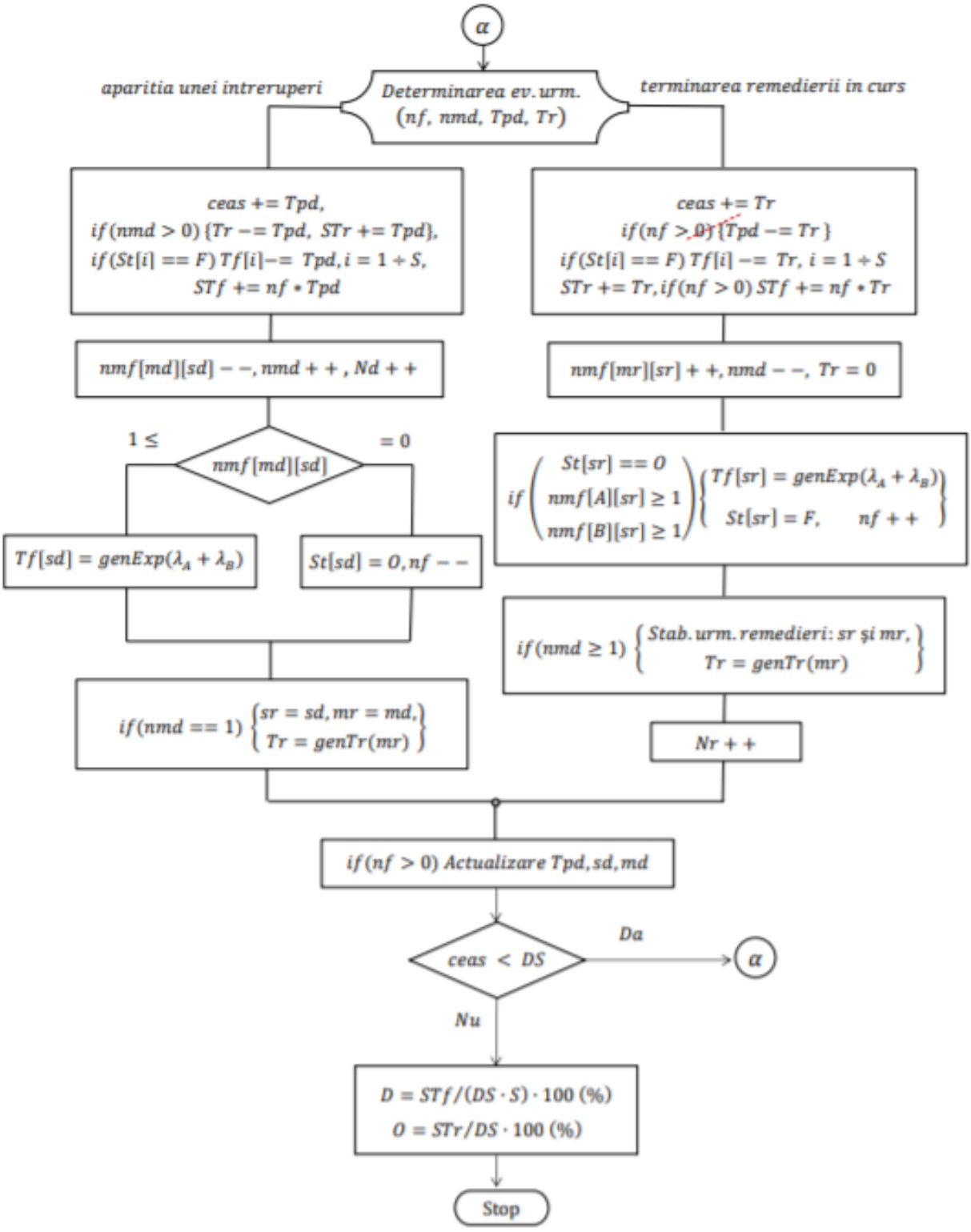
• $Tpd$ − timpul până la prima întrerupere accidentală (defectare) care se va produce, indiferent de tipul modulului afectat sau de sistemul de care aparţine; variabila nu are semnificaţie când $nf$ = 0;
• $Tr$ − timpul până la terminarea remedierii în curs; variabila nu are semnificaţie când $nmd$ = 0;
• $sd, md$ − sistemul la care va apărea prima defectare şi tipul modulului afectat;
• $sr, mr$ − sistemul la care se efectuează o operaţie de remediere şi tipul modulului în curs de remediere.

Statistici:
• $STf$ − Suma timpilor de funcționare pentru cele $S$ sisteme în perioada simulată;
• $STr$ − Suma timpilor de lucru pentru operaţiile de remediere efectuate de muncitor în perioada simulată;
• $Nd$ − Numărul de întreruperi accidentale (defectări) produse în perioada de monitorizare;
• $Nr$ − Numărul de remedieri efectuate de muncitor în perioada de simulare.

➢ $Precizare$:

Pentru început vom considera că modulul de rezervă se menţine în stare pasivă, nefiind aşadar solicitat cât timp nu este folosit.






Algoritm de simulare pentru problema de interferenţă în care sistemele sunt prevăzute cu modul de rezervă

## 3. Verificarea programului de simulare

➢ Cu iniţializarea $nmf[A][i] = 1$, $nmf[B][i] = 1$ , $i = 1 \div S$, se obţine o altă implementare pentru problema de interferenţă a sistemelor fără modul de rezervă, studiată la Etapa 2.

Observaţie:

Modulele de rezervă sunt menţinute în stare pasivă şi nu sunt afectate de întreruperi cât timp nu sunt folosite. Prin urmare, rata medie de întrerupere accidentală pentru un sistem în funcţiune este tot $\lambda A + \lambda B$ cât era şi la etapa anterioară.

### 1)Modulul A nu este echipat cu modul de rezervă

| S | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D(%) etapa 2 | 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(%) etapa 2 | 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 |
| D(%) | 90.733 | 89.9554 | 89.0716 | 88.0572 | 86.8766 | 85.5239 | 83.9668 | 82.1686 | 80.1379 | 77.821 | 75.2233 | 72.4336 | 69.4434 |
| O(%) | 9.26696 | 18.3806 | 27.2932 | 35.9726 | 44.3757 | 52.4078 | 60.032 | 67.1732 | 73.6614 | 79.5082 | 84.6017 | 88.8257 | 92.2254 |

```
S = 1
D = 90.733
O = 9.26696
Nd = 10000000
DS * S * (D/100) (lambdaA + lambdaB) = 1.00019e+07
STf *(lambdaA + lambdaB) = 1.00019e+07

S = 2
D = 89.9554
O = 18.3806
Nd = 10000000
DS * S * (D/100) (lambdaA + lambdaB) = 9.99828e+06
STf *(lambdaA + lambdaB) = 9.99828e+06

S = 3
D = 89.0716
O = 27.2932
Nd = 10000000
DS * S * (D/100) (lambdaA + lambdaB) = 1.0004e+07
STf *(lambdaA + lambdaB) = 1.0004e+07

S = 4
D = 88.0572
O = 35.9726
Nd = 10000000
DS * S * (D/100) (lambdaA + lambdaB) = 1.00019e+07
STf *(lambdaA + lambdaB) = 1.00019e+07

S = 5
D = 86.8766
O = 44.3757
Nd = 10000000
DS * S * (D/100) (lambdaA + lambdaB) = 9.99641e+06
STf *(lambdaA + lambdaB) = 9.99641e+06

S = 6
D = 85.5239
O = 52.4078
Nd = 10000000
DS * S * (D/100) (lambdaA + lambdaB) = 1.00002e+07
STf *(lambdaA + lambdaB) = 1.00002e+07

S = 7
D = 83.9668
O = 60.032
Nd = 10000000
```

```
O = 60.032
Nd = 10000000
DS * S * (D/100) (lambdaA + lambdaB) = 9.99445e+06
STf *(lambdaA + lambdaB) = 9.99445e+06

S = 8
D = 82.1686
O = 67.1732
Nd = 10000000
DS * S * (D/100) (lambdaA + lambdaB) = 9.99558e+06
STf *(lambdaA + lambdaB) = 9.99558e+06

S = 9
D = 80.1379
O = 73.6614
Nd = 10000000
DS * S * (D/100) (lambdaA + lambdaB) = 9.99662e+06
STf *(lambdaA + lambdaB) = 9.99662e+06

S = 10
D = 77.821
O = 79.5082
Nd = 10000000
DS * S * (D/100) (lambdaA + lambdaB) = 9.99478e+06
STf *(lambdaA + lambdaB) = 9.99478e+06

S = 11
D = 75.2233
O = 84.6017
Nd = 10000000
DS * S * (D/100) (lambdaA + lambdaB) = 9.99496e+06
STf *(lambdaA + lambdaB) = 9.99496e+06

S = 12
D = 72.4336
O = 88.8257
Nd = 10000000
DS * S * (D/100) (lambdaA + lambdaB) = 9.99973e+06
STf *(lambdaA + lambdaB) = 9.99973e+06

S = 13
D = 69.4434
O = 92.2254
Nd = 10000000
DS * S * (D/100) (lambdaA + lambdaB) = 9.99998e+06
STf *(lambdaA + lambdaB) = 9.99998e+06
```

Prin similaritatea rezultatelor se demostrează,deci, corectitudinea programului.

## 2)Modulul A este echipat cu modul de rezervă

| S | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D(%) Fara rezerva | 90.733 | 89.9554 | 89.0716 | 88.0572 | 86.8766 | 85.5239 | 83.9668 | 82.1686 | 80.1379 | 77.821 | 75.2233 | 72.4336 | 69.4434 |
| O(%) Fara rezerva | 9.26696 | 18.3806 | 27.2932 | 35.9726 | 44.3757 | 52.4078 | 60.032 | 67.1732 | 73.6614 | 79.5082 | 84.6017 | 88.8257 | 92.2254 |
| D(%) | 95.0236 | 93.3156 | 92.5984 | 91.8436 | 90.9892 | 89.9832 | 88.8063 | 87.4201 | 85.7269 | 83.6548 | 81.1022 | 77.8678 | 73.9462 |
| O(%) | 9.70822 | 19.0697 | 28.4044 | 37.5348 | 46.4661 | 55.1728 | 63.4983 | 71.4205 | 78.8404 | 85.5031 | 91.0965 | 95.4481 | 98.1959 |

```
S = 1
D = 95.0236
O = 9.70822
Nd = 10000000
DS * S * (D/100) (lambdaA + lambdaB) = 1.00035e+07
STf *(lambdaA + lambdaB) = 1.00035e+07

S = 2
D = 93.3156
O = 19.0697
Nd = 10000000
DS * S * (D/100) (lambdaA + lambdaB) = 1.00001e+07
STf *(lambdaA + lambdaB) = 1.00001e+07

S = 3
D = 92.5984
O = 28.4044
Nd = 10000000
DS * S * (D/100) (lambdaA + lambdaB) = 9.99555e+06
STf *(lambdaA + lambdaB) = 9.99555e+06

S = 4
D = 91.8436
O = 37.5348
Nd = 10000000
DS * S * (D/100) (lambdaA + lambdaB) = 9.99993e+06
STf *(lambdaA + lambdaB) = 9.99993e+06

S = 5
D = 90.9892
O = 46.4661
Nd = 10000000
DS * S * (D/100) (lambdaA + lambdaB) = 9.9983e+06
STf *(lambdaA + lambdaB) = 9.9983e+06

S = 6
D = 89.9832
O = 55.1728
Nd = 10000000
DS * S * (D/100) (lambdaA + lambdaB) = 1.00002e+07
STf *(lambdaA + lambdaB) = 1.00002e+07

S = 7
D = 88.8063
O = 63.4983
Nd = 10000000
```

```
O = 63.4983
Nd = 10000000
DS * S * (D/100) (lambdaA + lambdaB) = 9.99693e+06
STf *(lambdaA + lambdaB) = 9.99693e+06

S = 8
D = 87.4201
O = 71.4205
Nd = 10000000
DS * S * (D/100) (lambdaA + lambdaB) = 1.00014e+07
STf *(lambdaA + lambdaB) = 1.00014e+07

S = 9
D = 85.7269
O = 78.8404
Nd = 10000000
DS * S * (D/100) (lambdaA + lambdaB) = 9.99693e+06
STf *(lambdaA + lambdaB) = 9.99693e+06

S = 10
D = 83.6548
O = 85.5031
Nd = 10000000
DS * S * (D/100) (lambdaA + lambdaB) = 9.99773e+06
STf *(lambdaA + lambdaB) = 9.99773e+06

S = 11
D = 81.1022
O = 91.0965
Nd = 10000000
DS * S * (D/100) (lambdaA + lambdaB) = 1e+07
STf *(lambdaA + lambdaB) = 1e+07

S = 12
D = 77.8678
O = 95.4481
Nd = 10000000
DS * S * (D/100) (lambdaA + lambdaB) = 9.99658e+06
STf *(lambdaA + lambdaB) = 9.99658e+06

S = 13
D = 73.9462
O = 98.1959
Nd = 10000000
DS * S * (D/100) (lambdaA + lambdaB) = 9.99731e+06
STf *(lambdaA + lambdaB) = 9.99731e+06
```

**Concluzie:**

În cea de-a doua simulare, în varianta cu rezervă, se observă o creștere a disponibilității sistemului cât și a gradului de ocupare al muncitorului de deservire.

## COD SURSĂ:

```
#include <iostream>
#include <stdlib.h>
#include <math.h>

using namespace std;

#define lambdaA 0.2105
#define lambdaB 0.1626
#define miuA 3.8533
#define miuB 3.4218

#define pA lambdaA/(lambdaA + lambdaB)
#define pB 1-pA

#define mA 1/miuA
#define mB 1/miuB

#define sigmaA 1/(3.5*miuA)
#define sigmaB 1/(3.5*miuB)

enum Module { A, B };
enum Stare { O, F };


double genExp(double lambda)
{
	double u, x;
	u = (double)rand() / (RAND_MAX + 1);
	x = -1 / lambda * log(1 - u);
	return x;
}

double genTr(Module m)
{
	double Tr;
	if (m == A)
	{
		Tr = genExp(miuA);
	}
	else
	{
		Tr = genExp(miuB);
	}

	return Tr;
}

void Simulare(int& S, double& D, double& Oc)
{
	double NS = 1e+7; //numarul de simulari
	double DS;
	double STr = 0; //suma timpilor de remediere
	double STf = 0; //suma timpilor de functionare
	double ceas = 0; //ceasul simularii
	int Nd = 0; //numarul sistemelor decfectate pe parcursul simularii
	int Nr = 0; //numarul remedierilor efectuate
	int nf = S; //numarul de sisteme in functiune
	int nmd = 0; // numarul modulelor defecte, indiferent de tipul lor si de
sistemele de care apartin
```

```
	int nmf[2][100];

	double Tf[100]; //pentru un sistem in functiune, timpul pana la aparitia primei intreruperi accidentale
	Stare St[100]; //starea sistemului i

	double Tpd; //timpul pana la prima intrerupere accidentala(defectare) la sistemele aflate in functiune
	int sd;//sistemul la care va aparea prima defectare
	int sr; //sistemul la care se face remedierea;

	Module md; //modulul afectat de aceasta intrerpere accidentala
	Module mr; //modulul de remediat in curs

	double Tr = 0; //timpul pana la terminarea remedierii in curs
	int No = 0; //numarul de sisteme oprite

	for (int i = 1; i <= S; i++)
	{
		//nmf[A][i] = 1;///modulul A este fara modul de rezerva(pentru verificare etapa II)
		nmf[A][i] = 2;//modulul A este prevazut cu modul de rezerva
		nmf[B][i] = 1;
	}
	for (int i = 1; i <= S; i++)
	{
		St[i] = F;
		Tf[i] = genExp(lambdaA + lambdaB);
	}

	double min = 1e+7;
	int ind;

	for (int i = 1; i <= S; i++)
	{
		if (St[i] == F && Tf[i] < min)
		{
			min = Tf[i];
			ind = i;
		}
	}
	Tpd = min;
	sd = ind;

	//determina modulul afectat de intrerupere
	double u = (double)rand() / RAND_MAX;
	if (u < pA)
	{
		md = A;
	}
	else
	{
		if (u < (pA + pB))
			md = B;
	}
	do {
		//Determinarea evenimentului urmator
		if (nmd == 0 || ((nf > 0) && Tpd < Tr))
		{
			//defectare
			Nd++;
			ceas += Tpd;
```

```
		if (nmd > 0)
		{
			Tr -= Tpd;
		}
		for (int i = 1; i <= S; i++)
		{
			if (St[i] == F)
			{
				Tf[i] -= Tpd;
			}
		}
		STf += nf * Tpd;
		nmf[md][sd]--;
		nmd++;
		if (nmf[md][sd] == 0) //oprire sistem
		{
			St[sd] = O;
			nf--;
			No++;
		}
		else
		{
			if (nmf[md][sd] > 0)
				Tf[sd] = genExp(lambdaA + lambdaB);
		}
		if (nmd == 1)
		{
			sr = sd;
			mr = md;
			Tr = genTr(mr);
			STr += Tr;
		}

	}
	else
	{
		//remediere
		Nr++;
		ceas += Tr;
		for (int i = 1; i <= S; i++)
		{
			if (St[i] == F)
			{
				Tf[i] -= Tr;
			}
		}
		STf += nf * Tr;
		nmf[mr][sr]++;
		nmd--;
		if (St[sr] == O && nmf[A][sr] >= 1 && nmf[B][sr] >= 1)
		{
			St[sr] = F;
			nf++;
			Tf[sr] = genExp(lambdaA + lambdaB);
		}
		if (nmd > 0)
		{
			//inceputul unei noi remedieri
			//Actualizare sr, mr
			for (int i = 1; i <= S; i++)
			{
				if (St[i] == O)
```

```
                                {
                                        if (nmf[B][i] < 1)
                                        {
                                                mr = B;
                                                sr = i;
                                                break;
                                        }
                                        else
                                        {
                                                //if (nmf[A][i] < 1)//modulul A fara
rezerva, pentru verificare etapa II
                                                if (nmf[A][i] < 2)
                                                {
                                                        mr = A;
                                                        sr = i;
                                                        break;
                                                }
                                        }
                                }
                        }
                        Tr = genTr(mr);
                        STr += Tr;
                }

        }

        if (nf > 0)
        { //Actualizare Tpd,sd
                min = 1e+6;
                for (int i = 1; i <= S; i++)
                {
                        if (Tf[i] < min && St[i] == F)
                        {
                                min = Tf[i];
                                ind = i;
                        }
                }
                Tpd = min;
                sd = ind;
                //Actualizare md
                double u = (double)rand() / RAND_MAX;
                if (u < pA)
                {
                        md = A;
                }
                else
                {
                        if (u < (pA + pB))
                                md = B;
                }
        }

} while (Nd < NS);
DS = ceas;

//calcul statistici
D = (STf / (DS * S)) * 100;
Oc = (STr / DS) * 100;

cout << "D = " << D << endl;
cout << "O = " << Oc << endl;
```

```
	//Verificarea programului de simulare (Modulul B cu rezerva)
	cout << "Nd = " << Nd << endl;
	cout << "DS * S * (D/100) (lambdaA + lambdaB) = " << DS * S * (D / 100) *
(lambdaA + lambdaB) << endl;
	cout << "STf *(lambdaA + lambdaB) = " << STf * (lambdaA + lambdaB) << endl <<
endl;

	S++;

}

int main(void)
{
	double Oc = 0, D;
	int S = 1;

	while (Oc <= 99)
	{
		cout << "S = " << S << endl;
		Simulare(S, D, Oc);
	}
}
```

# 4.Completarea programului de simulare pentru a acoperi şi alte aspecte

## Generalizare privind regimul de lucru pentru modulul de rezervă

În funcţie de gradul de solicitare a rezervei (în sensul de modul cu regim de rezervă), aceasta poate fi pasivă, activă sau parţial activă. Pentru o rezervă identică cu modulul de bază, rata întreruperilor accidentale (rata de defectare) se exprimă cu relaţia: $\lambda R = \alpha \lambda M$, în care $M = A$ sau $B$, iar $\alpha \in [0, 1]$. (1)

• $\alpha = 0 \rightarrow$ rezervă pasivă (nesolicitată cât timp nu este în funcţiune);

• $\alpha = 1 \rightarrow$ rezervă activă (solicitată în aceeaşi măsură cu modulul de bază);

• $\alpha \in (0, 1) \rightarrow$ rezervă parţial activă (solicitată într-o măsură mai mică decât modulul de bază); în studiul nostru vom lucra cu $\alpha = 0.5$ şi vom numi rezerva ca fiind semi-activă.

➢ Rezervă la modulul $A$

$\lambda = \lambda B + \lambda A(1 + \alpha(nmf[A][i] - 1))$

Probabilitatea ca la sistemul $i$ modulul care se va defecta mai întâi să fie de tip $B$ este:

$pB = \lambda B / (\lambda B + \lambda A(1 + \alpha(nmf[A][i] - 1)))$

Programul de simulare trebuie modificat ţinând cont de aceste relaţii care reflectă faptul că rata defectărilor la un sistem în funcţiune nu mai este constantă în timp. În plus, pe ramura care tratează terminarea unei remedieri la sistemul $sr$, secvenţa de program trebuie modificată astfel:

⋮

$if(nmf[A][sr] \geq 1 \;\&\&\; nmf[B][sr] \geq 1)$

{
$Tf[sr] = genExp\ (\lambda B + \lambda A(1 + \alpha(nmf[A][sr] - 1)))$;
// actualizarea se impune chiar dacă sistemul era deja în funcţiune pentru că în urma
// remedierii s-a modificat configuraţia sistemului
$if(St[sr] == O)$
{
$St[sr] = F$;
$nf++$;
}
}
⋮

| S | Fără rezervă | | Rezervă activă $\alpha = 1$ | | Rezervă semiactivă $\alpha = 0.5$ | | Rezervă pasivă $\alpha = 0$ | |
|---|---|---|---|---|---|---|---|---|
| | D(%) | O(%) | D(%) | O(%) | D(%) | O(%) | D(%) | O(%) |
| 1 | 90.733 | 9.26696 | 93.9203 | 9.53029 | 93.9203 | 9.53029 | 93.9203 | 9.53029 |
| 2 | 89.9554 | 18.3806 | 93.5494 | 18.9785 | 93.5494 | 18.9785 | 93.5494 | 18.9785 |
| 3 | 89.0716 | 27.2932 | 93.1256 | 28.3306 | 93.1256 | 28.3306 | 93.1256 | 28.3305 |
| 4 | 88.0572 | 35.9726 | 92.656 | 37.5567 | 92.6561 | 37.5564 | 92.656 | 37.5567 |
| 5 | 86.8766 | 44.3757 | 92.1039 | 46.6361 | 92.1048 | 46.6323 | 92.104 | 46.6339 |
| 6 | 85.5239 | 52.4078 | 91.4255 | 55.6266 | 91.426 | 55.625 | 91.4254 | 55.6285 |
| 7 | 83.9668 | 60.032 | 90.6191 | 64.3163 | 90.6221 | 64.3096 | 90.6199 | 64.3089 |
| 8 | 82.1686 | 67.1732 | 89.594 | 72.7184 | 89.5919 | 72.7236 | 89.6045 | 72.6994 |
| 9 | 80.1379 | 73.6614 | 88.2575 | 80.577 | 88.2608 | 80.5854 | 88.2632 | 80.5579 |
| 10 | 77.821 | 79.5082 | 86.405 | 87.5539 | 86.3965 | 87.5618 | 86.3967 | 87.5563 |
| 11 | 75.2233 | 84.6017 | 83.6837 | 93.2888 | 83.6739 | 93.3034 | 83.7194 | 93.2577 |
| 12 | 72.4336 | 88.8257 | 79.9071 | 97.1788 | 79.9309 | 97.1765 | 79.8916 | 97.1796 |
| 13 | 69.4434 | 92.2254 | 75.219 | 99.1244 | 75.1673 | 99.1439 | 75.2439 | 99.1305 |

**Concluzie:**

Se observă o creștere a disponibilității sistemelor, începând cu sistemele fără rezervă până la sistemele ce au o rezervă pasivă. Disponibilitatea sistemelor cu rezervă pasivă este cea mai mare urmată de disponibilitatea sistemelor cu rezervă semi-activă și rezervă activă ( în această ordine), cea mai mică fiind pentru sistemele care nu dețin niciun modul de rezervă.

# COD SURSĂ:

```
#include <iostream>
#include <stdlib.h>
#include <math.h>

using namespace std;

#define lambdaA 0.2105
#define lambdaB 0.1626
#define miuA 3.8533
#define miuB 3.4218

#define alfa 0.5 //alfa=0 sau alfa=1

//#define pA lambdaA/(lambdaA + lambdaB)
//#define pB 1-pA

#define MFR B

#define mA 1/miuA
#define mB 1/miuB

#define sigmaA 1/(3.5*miuA)
#define sigmaB 1/(3.5*miuB)

enum Module { A, B };
enum Stare { O, F };


double genExp(double lambda)
{
	double u, x;
	u = (double)rand() / (RAND_MAX + 1);
	x = -1 / lambda * log(1 - u);
	return x;
}

double genTr(Module m)
{
	double Tr;
	if (m == A)
	{
		Tr = genExp(miuA);
	}
	else
	{
		Tr = genExp(miuB);
	}
```

```
	return Tr;
}

void Simulare(int& S, double& D, double& Oc)
{
	double NS = 1e+7; //numarul de simulari
	double DS;
	double STr = 0; //suma timpilor de remediere
	double STf = 0; //suma timpilor de functionare
	double ceas = 0; //ceasul simularii
	int Nd = 0; //numarul sistemelor decfectate pe parcursul simularii
	int Nr = 0; //numarul remedierilor efectuate
	int nf = S; //numarul de sisteme in functiune
	int nmd = 0; // numarul modulelor defecte, indiferent de tipul lor si de
sistemele de care apartin
	int nmf[2][100];

	double Tf[100]; //pentru un sistem in functiune, timpul pana la aparitia primei
intreruperi accidentale
	Stare St[100]; //starea sistemului i

	double Tpd; //timpul pana la prima intrerupere accidentala(defectare) la
sistemele aflate in functiune
	int sd;//sistemul la care va aparea prima defectare
	int sr; //sistemul la care se face remedierea;

	Module md; //modulul afectat de aceasta intrerpere accidentala
	Module mr; //modulul de remediat in curs

	double Tr = 0; //timpul pana la terminarea remedierii in curs
	int No = 0; //numarul de sisteme oprite

	for (int i = 1; i <= S; i++)
	{
		//nmf[A][i] = 1;///modulul A este fara modul de rezerva(pentru
verificare etapa II)
		nmf[A][i] = 2;//modulul A este prevazut cu modul de rezerva
		nmf[B][i] = 1;
	}
	for (int i = 1; i <= S; i++)
	{
		St[i] = F;
		Tf[i] = genExp(lambdaB + lambdaA * (1 + alfa * (nmf[A][i] - 1)));
	}

	double min = 1e+7;
	int ind;

	for (int i = 1; i <= S; i++)
	{
		if (St[i] == F && Tf[i] < min)
		{
			min = Tf[i];
			ind = i;
		}
	}
	Tpd = min;
	sd = ind;

	//determina modulul afectat de intrerupere
	double u = (double)rand() / RAND_MAX;
	double pB = lambdaB + lambdaA * (1 + alfa * (nmf[A][sd] - 1));
```

```
	double pA = 1 - pB;
	if (u < pA)
	{
		md = A;
	}
	else
	{
		if (u < (pA + pB))
			md = B;
	}
	do {
		//Determinarea evenimentului urmator
		if (nmd == 0 || ((nf > 0) && Tpd < Tr))
		{
			//defectare
			Nd++;
			ceas += Tpd;
			if (nmd > 0)
			{
				Tr -= Tpd;
			}
			for (int i = 1; i <= S; i++)
			{
				if (St[i] == F)
				{
					Tf[i] -= Tpd;
				}
			}
			STf += nf * Tpd;
			nmf[md][sd]--;
			nmd++;
			if (nmf[md][sd] == 0) //oprire sistem
			{
				St[sd] = O;
				nf--;
				No++;
			}
			else
			{
				if (nmf[md][sd] > 0)
					Tf[sd] = genExp(lambdaB + lambdaA * (1 + alfa *
(nmf[A][sd] - 1)));
				else
				{
					St[sd] = O;
					--nf;
					++No;
				}
			}
			if ((nmd == 1) || (St[sr] == F && St[sd] == O) || (sd == sr && md
== MFR))
			{
				sr = sd;
				mr = md;

				if (nmd > 1)
				{
					STr -= Tr;
				}

				Tr = genTr(mr);
				STr += Tr;
```

```
                }

            }
            else
            {
                //remediere
                Nr++;
                ceas += Tr;

                for (int i = 1; i <= S; i++)
                {
                    if (St[i] == F)
                    {
                        Tf[i] -= Tr;
                    }
                }
                STf += nf * Tr;
                nmf[mr][sr]++;
                nmd--;
                if (nmf[A][sr] >= 1 && nmf[B][sr] >= 1)
                {
                    if (St[sr] == O)
                    {
                        St[sr] = F;
                        nf++;
                    }
                    Tf[sr] = genExp(lambdaB + lambdaA * (1 + alfa * (nmf[A][sr]
- 1)));
                }
                if (nmd > 0)
                {
                    //inceputul unei noi remedieri
                    //Actualizare sr, mr
                    bool gasit = false;

                    for (int i = 1; i <= S; i++)
                    {
                        if (nmf[B][i] == 0) //starea S2 sau S3(se remediaza
B)
                        {
                            mr = B;
                            sr = i;
                            gasit = true;
                            break;
                        }
                    }
                    if (gasit == false) //nu s a gasit modulul remediat
                    {
                        for (int i = 1; i <= S; i++)
                        {
                            if (nmf[A][i] == 0) //starea 4 (ambele module
A defecte)
                            {
                                mr = A;
                                sr = i;
                                gasit = true;
                                break;
                            }
                        }
                    }
                    if (gasit == false)
                    {
```

```
                                for (int i = 1; i <= S; i++)
                                {
                                        if (nmf[A][i] == 1 && nmf[B][i] == 1)
//starea 1
                                        {
                                                mr = B;
                                                sr = i;
                                                gasit = true;
                                        }
                                }
                        }
                        Tr = genTr(mr);
                        STr += Tr;
                }

        }

        if (nf > 0)
        { //Actualizare Tpd,sd
                min = 1e+6;
                for (int i = 1; i <= S; i++)
                {
                        if (Tf[i] < min && St[i] == F)
                        {
                                min = Tf[i];
                                ind = i;
                        }
                }
                Tpd = min;
                sd = ind;
                //Actualizare md
                double u = (double)rand() / RAND_MAX;
                double pB = lambdaB + lambdaA * (1 + alfa * (nmf[A][sd] - 1));
                double pA = 1 - pB;

                if (u < pA)
                {
                        md = A;
                }
                else
                {
                        if (u < (pA + pB))
                                md = B;
                }
        }

    } while (Nd < NS);
    DS = ceas;

    //calcul statistici
    D = (STf / (DS * S)) * 100;
    Oc = (STr / DS) * 100;

    cout << "D = " << D << endl;
    cout << "O = " << Oc << endl;

    //Verificarea programului de simulare (Modulul B cu rezerva)
    cout << "Nd = " << Nd << endl;
    cout << "DS * S * (D/100) (lambdaA + lambdaB) = " << DS * S * (D / 100) *
(lambdaA + lambdaB) << endl;
    cout << "STf *(lambdaA + lambdaB) = " << STf * (lambdaA + lambdaB) << endl <<
endl;
```

```
        S++;

}

int main(void)
{
        double Oc = 0, D;
        int S = 1;

        while (Oc <= 99)
        {
                cout << "S = " << S << endl;
                Simulare(S, D, Oc);
        }
}
```

## Alegerea următorului modul pentru remediere

La terminarea remedierii în curs, dacă mai sunt module defecte, muncitorul trebuie să înceapă imediat o nouă remediere. La alegerea următorului modul care să fie remediat, muncitorul trebuie să urmărească repunerea cât mai rapidă în funcţiune a unui sistem oprit. Modulele de rezervă defecte de la sistemele în funcţiune trebuie remediate ulterior. Prin urmare, la alegerea următorului modul pentru remediere trebuie să se ţină cont de starea sistemelor şi de intensităţile medii de remediere pentru cele două tipuri de module, $A$ şi $B$. Să analizăm mai întâi stările posibile pentru un sistem, în funcţie de modulul la care este prevăzută rezerva.

**Rezervă la modulul A**

Stări posibile:

$S_1 = A\ B\ \overline{A}$ } Sistem în funcțiune

$S_2 = A\ \overline{B}\ A$
$S_3 = A\ \overline{A}\ \overline{B}$
$S_4 = \overline{A}\ \overline{A}\ B$ } Sistem oprit

Dacă $\mu A > \mu B$ ordinea de prioritate la remediere este: $S4$, $S2$ sau $S3$ (se repară $\overline{B}$), $S1$;

Dacă $\mu B > \mu A$ ordinea de prioritate la remediere este: $S2$ sau $S3$ (se repară $\overline{B}$), $S4$, $S1$.

Verificăm creșterea disponibilității sistemelor prin impunere la remediere a acestor priorități.

| S | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D(%) | 95.0236 | 93.3156 | 92.5984 | 91.8436 | 90.9892 | 89.9832 | 88.8063 | 87.4201 | 85.7269 | 83.6548 | 81.1022 | 77.8678 | 73.9462 |
| O(%) | 9.70822 | 19.0697 | 28.4044 | 37.5348 | 46.4661 | 55.1728 | 63.4983 | 71.4205 | 78.8404 | 85.5031 | 91.0965 | 95.4481 | 98.1959 |
| D(%) Prioritate | 94.3954 | 93.8494 | 93.252 | 92.5644 | 91.8103 | 90.9401 | 89.9126 | 88.6873 | 87.1814 | 85.1917 | 82.503 | 78.9429 | 74.5046 |
| O(%) Prioritate | 9.6513 | 19.1837 | 28.5642 | 37.8456 | 46.9036 | 55.7531 | 64.322 | 72.5135 | 80.1314 | 87.0408 | 92.7048 | 96.7155 | 98.8961 |

Modificările efectuate în programul de simulare:

```
if (nmd > 0)
{
	//inceputul unei noi remedieri
	//Actualizare sr, mr
	bool gasit = false;

	for (int i = 1; i <= S; i++)
	{
		if (nmf[B][i] == 0) //starea S2 sau S3(se remediaza
B)
		{
			mr = B;
			sr = i;
			gasit = true;
			break;
		}
	}
	if (gasit == false) //nu s a gasit modulul remediat
	{
		for (int i = 1; i <= S; i++)
		{
			if (nmf[A][i] == 0) //starea 4 (ambele module
A defecte)
			{
				mr = A;
				sr = i;
				gasit = true;
				break;
			}
		}
	}
	if (gasit == false)
	{
		for (int i = 1; i <= S; i++)
		{
			if (nmf[A][i] == 1 && nmf[B][i] == 1)
//starea 1
			{
				mr = B;
				sr = i;
				gasit = true;
			}
		}
	}
	Tr = genTr(mr);
	STr += Tr;
```

}

**Concluzie:**

Atunci când nu alegem modulul defect la întâmplare, ci se stabilesc anumite priorități în funcție de starea sistemului și de rata de defectare a modulelor, se observă o creștere a disponibilității care este mai accentuată atunci când creștem și numărul de sisteme.

## Întreruperea remedierii în curs

Dacă în timp ce muncitorul remediază o rezervă de la un sistem în funcţiune se produce o întrerupere accidentală care conduce la oprirea sistemului, se poate pune problema întreruperii remedierii în curs pentru a interveni cu prioritate asupra modulului afectat, astfel încât sistemul oprit să fie repus în funcţiune cât mai rapede.

Pentru implementarea acestei facilităţi în programul de simulare, pe ramura care tratează apariţia unei întreruperi accidentale, se impune următoarea modificare:

⋮
$if(nmd == 1 \parallel St[sr] == F \;\&\&\; St[sd] == O \parallel sd == sr \;\&\&\; md == MFR)$
{
$sr = sd;$
$mr = md;$
$Tr = genTr(mr);$
}
⋮
în care $MFR$ reprezintă modulul fără rezervă.

Verificăm creșterea disponibilității sistemelor prin impunere la remediere a acestor priorități.

| S | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| D(%) | 95.0236 | 93.3156 | 92.5984 | 91.8436 | 90.9892 | 89.9832 | 88.8063 | 87.4201 | 85.7269 | 83.6548 | 81.1022 | 77.8678 | 73.9462 |
| O(%) | 9.70822 | 19.0697 | 28.4044 | 37.5348 | 46.4661 | 55.1728 | 63.4983 | 71.4205 | 78.8404 | 85.5031 | 91.0965 | 95.4481 | 98.1959 |
| D(%) Întrerupere | 94.3949 | 94.0699 | 93.6954 | 93.2569 | 92.7609 | 92.1374 | 91.3723 | 90.3684 | 88.9706 | 86.9208 | 83.8957 | 79.7012 | 74.8414 |
| O(%) Întrerupere | 9.65287 | 19.2277 | 28.7184 | 38.1164 | 47.3606 | 56.4731 | 65.3252 | 73.8196 | 81.7623 | 88.7624 | 94.2707 | 97.7555 | 99.347 |

**Concluzie:**

Întreruperea remedierii în curs a crescut și mai mult disponibilitatea sistemelor și prin urmare rezultatele obținute, adăugând programului anterior și această facilitate, sunt mai bune, sistemul având capacitatea mai mare de a-și îndeplini funcțiile la un moment dat.