Proiect modelare și simulare

Diaconu Rareș-George

1306A

Etapa 1:

1. Etapele prelucrării numerice pentru estimarea repartiţiei variabilelor aleatoare primare:

În cadrul acestui proiect se studiază problema unui sistem a cărui funcționare este afectată de întreruperi accidentale specifice procesului realizat. În caz de oprire, este necesară intervenția unui muncitor pentru remediere și repunere în funcțiune. Ca indicator de performanță se folosește disponibilitatea sistemului, care se impune a fi cât mai mare din considerente economice. Pentru muncitor, se impune un grad de ocupare corespunzător, care să nu ducă la suprasolicitare. Astfel, pentru un muncitor sau un grup de muncitori, se alocă mai multe sisteme. Atunci când numărul de sisteme oprite este mai mare decât numărul muncitorilor, intervine un timp de așteptare până la începerea remedierii, numit timp de interferență a sistemelor.

În această etapă, s-a identificat repartiția pentru variabilele aleatorii Tf (timpul de funcționare până la întrerupere și timpul de remediere în caz de întrerupere) pentru două module, A și B, care pot fi afectate în mod independent de întreruperi accidentale.

Etapa implică prelucrarea statistică a acestor eşantioane de valori în vederea determinării funcţiei de repartiţie pentru aceste variabile aleatoare primare. Fiecare variabilă se studiază separat. Prelucrarea unui eşantion de valori în vederea determinării repartiţiei variabilei aleatoare studiate implică parcurgerea următorilor paşi:

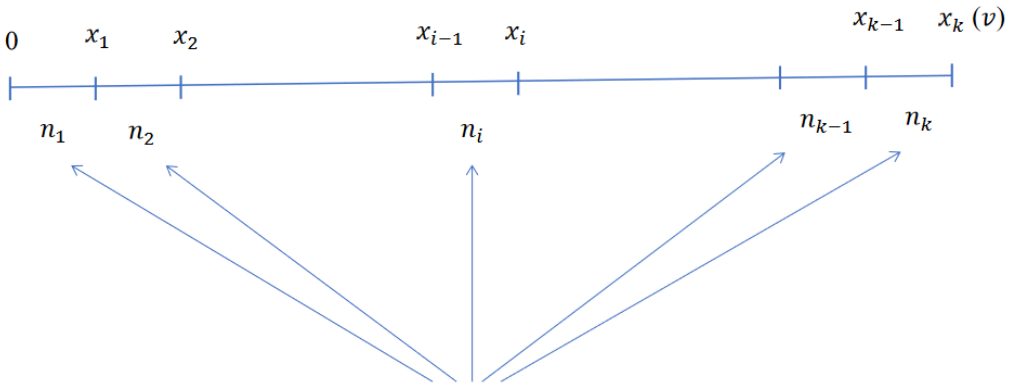
***Pasul* 1**. Valorile din eşantion se ordonează crescător şi se determină media aritmetică a lor (ma). Asta permite o analiză preliminară mai uşoară a valorilor de care dispunem; de exemplu, care este valoarea cea mai mică şi cea mai mare, dacă este o oarecare simetrie faţă de valoarea medie sau, din contra, dacă se remarcă o asimetrie accentuată. Aceste elemente ne ajută să ne orientăm către un anumit tip de repartiţie.

Pasul 2. Se adoptă un interval de analiză potrivit [0, v] care să cuprindă valorile din eşantion sau marea majoritate a lor (între 98% şi 100% din valori).

Pasul 3. Se împarte intervalul de analiză [0, v] în k diviziuni egale şi se determină repartizarea valorilor din eşantion pe aceste subintervale. Fie N numărul de valori din eşantion. În funcţie de mărimea eşantionului se adoptă un număr potrivit de subintervale. Ca referinţă în adoptarea lui k, poate fi aplicată relaţia propusă de $Sturgers$, şi anume:

$$k = 1 + 3.222 * ln(N)$$

Dacă $\Delta = v/k$ este mărimea unei diviziuni (a unui subinterval) atunci capătul din dreapta al subintervaluilui cu numărul de ordine i este $xi = i \cdot \Delta$ (i = 1, 2, … . , k). Să notăm cu ni numărul de valori din eşantion care se regăsesc în subintervalul cu numărul de ordine i, aşa cum este ilustrat în figura următoare.



N valori independente

Pasul 4. Se adoptă o valoare $\lambda 0$ ca primă estimare a parametrului variabilei aleatoare studiate, pentru care s-a presupus o repartiţie exponenţial-negativă. Pentru asta, se au în vedere două proprietăţi privind variabilele de acest tip, şi anume:

- valoarea medie este $1/\lambda$;

- peste 99% din valori se încadrează în intervalul [0, $5/\lambda$].

Prin urmare, $\lambda 0$ se poate adopta în două moduri: $\lambda 0 = 1/ma$, sau $\lambda 0 = 5/v$. În ambele cazuri acurateţea aproximării poate fi relativ scăzută, dar permite totuşi o verificare a ipotezei cu privire la tipul repartiţiei adoptate.

Pasul 5. Se compară grafic funcţia de repartiţie teoretică $F(\lambda_0, x) = 1 - e^{-\lambda_0 x}$ sau funcţia densitate de probabilitate $f(\lambda_0, x) = \lambda_0 e^{-\lambda_0 x}$

cu funcţia empirică corespunzătoare (Fe sau fe), calculată în k puncte echidistante din intervalul de analiză [0, 5/λ0]. Pentru determinarea valorilor funcţiei expirice se pleacă de la semnificaţia funcţiei teoretice. În cazul lucrului cu funcţia densitate de probabilitate, aşa cum se arată în figura următoare se au în Vedere următoarelor relaţii:

$$\mathcal{A}=\int_{x_{i-1}}^{x_i} f(\lambda,x)\,dx \approx f(y_i)\times\Delta$$

$$\mathcal{A} = Prob(x_{i-1} \le X < x_i) \approx n_i/N$$

Pe baza lor se deduce următoarea relaţie de calcul pentru funcţia empirică:

$$f_e(y_i) = n_i/N/\Delta, i = 1 \div k.$$

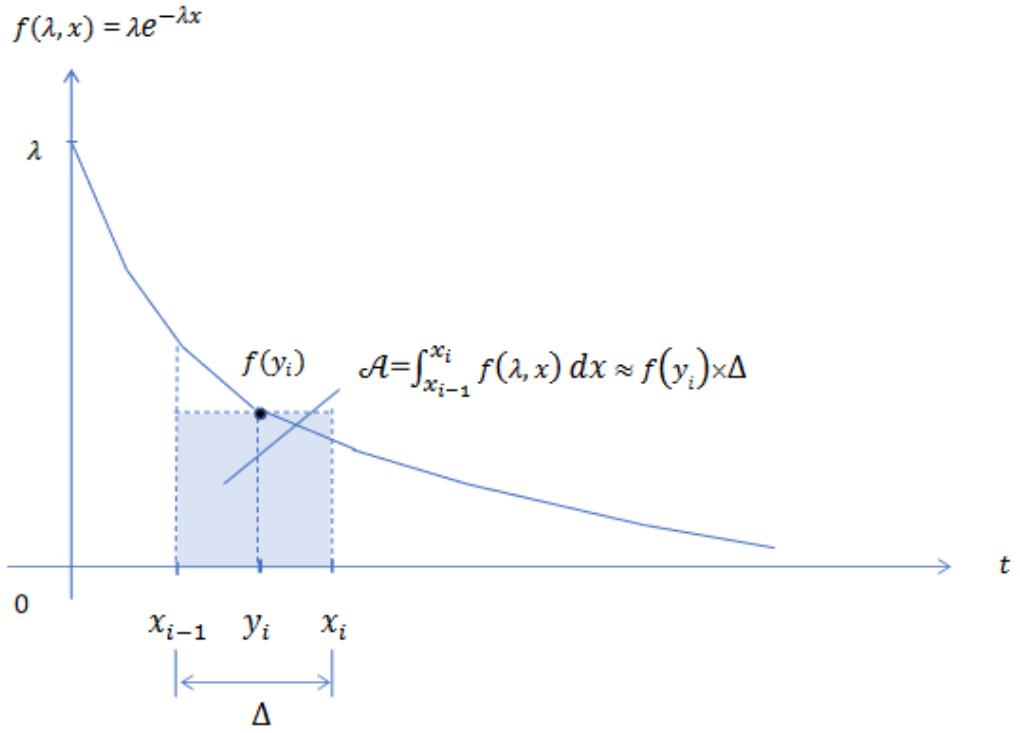


Fig. 3 – Estimarea valorilor funcţiei densitate de probabilitate.

Pasul 6. Se realizează un proces de căutare într-o vecinătate a valorii λ0 care să permită găsirea celei mai potrivite valori pentru parametrul λ. Drept criteriu folosit în acest proces se adoptă suma pătratelor diferenţelor dintre funcţia teorerită şi funcţia empirică, diferenţe calculate în cele k puncte din intervalul de analiză. Aşadar, funcţia criteriu care trebuie minimizată în cadrul procesului de căutare poate fi:

$$F_c(\lambda) = \sum_{i=1}^{k}(F(\lambda, x_i) - F_e(x_i))^2,$$

sau

$$F_c(\lambda) = \sum_{i=1}^{k}(f(\lambda, y_i) - f_e(y_i))^2,$$

în care $x_i = i \cdot \Delta$, iar $y_i = (x_{i-1} + x_i)/2 = x_i - \Delta/2, i = 1, 2, \ldots, k.$

***Pasul* 7**. Se verifică corectitudinea soluţiei adoptate privind parametrul λ (valoare notată λF), comparând grafic funcţia teoretică cu funcţia empirică corespunzătoare (Fe sau fe), calculată în k puncte echidistante din intervalul de analiză [0, 5/λF].

2. Rezultate intermediare şi finale de la prelucarea fiecărui eşantion de valori:

a. La primul eșantion:

- Eşantionul de valori direct sortat (nu cel iniţial) : 0.0020 0.0249

0.0307	0.0443	0.0528	0.0760	0.0802	0.0982
0.1069	0.1098	0.1202	0.1632	0.1674	0.1832
0.1930	0.2071	0.2122	0.2163	0.2174	0.2549
0.2562	0.2797	0.2970	0.3113	0.3473	0.3543
0.3727	0.3902	0.4569	0.4719	0.4869	0.4926
0.4977	0.5324	0.5578	0.5622	0.5758	0.5791
0.5891	0.6063	0.6137	0.6195	0.6297	0.6353
0.6499	0.6666	0.6737	0.6757	0.6967	0.7273
0.7543	0.7589	0.7710	0.7827	0.7949	0.7958
0.8661	0.9282	0.9358	0.9367	0.9377	1.0358
1.0400	1.0476	1.0638	1.0667	1.0730	1.0793
1.0800	1.1420	1.1498	1.1591	1.1738	1.1739
1.1856	1.1980	1.2029	1.2204	1.2251	1.2559
1.2607	1.2714	1.2805	1.2872	1.2920	1.3129
1.3179	1.3247	1.3247	1.3583	1.3694	1.3749
1.3790	1.3847	1.3866	1.3955	1.4126	1.4444
1.4493	1.4602	1.4689	1.5267	1.5302	1.5450
1.6081	1.6429	1.6531	1.6566	1.7094	1.7417
1.7517	1.7602	1.7623	1.7721	1.8172	1.8602
1.8874	1.8908	1.9489	1.9593	1.9781	2.0337
2.0960	2.1383	2.1764	2.2213	2.2260	2.3443
2.3619	2.4074	2.4546	2.4867	2.4976	2.4976
2.5028	2.5030	2.6062	2.6292	2.6356	2.6733

2.7098	2.7182	2.7307	2.7384	2.8361	2.8521
2.9443	3.1221	3.1293	3.1367	3.1678	3.1855
3.2061	3.2108	3.2486	3.2527	3.3085	3.3316
3.3964	3.4614	3.4835	3.5035	3.5375	3.5574
3.5883	3.6099	3.6752	3.8403	3.9501	3.9668
3.9740	4.0053	4.0438	4.0439	4.1204	4.2181
4.2345	4.2883	4.2954	4.3535	4.3576	4.3621
4.3685	4.4591	4.4922	4.5428	4.6025	4.6412
4.6916	4.6978	4.8083	4.8091	4.8484	4.8518
4.9379	4.9559	4.9678	4.9790	4.9838	4.9931
5.0732	5.1889	5.2266	5.3104	5.3278	5.3664
5.3688	5.3735	5.3941	5.4197	5.5163	5.5229
5.5845	5.6856	5.7714	5.8528	5.8936	5.9156
5.9256	5.9627	5.9650	5.9783	6.0062	6.1117
6.1908	6.2087	6.2249	6.2556	6.2723	6.3655
6.4177	6.5450	6.6170	6.7113	6.7480	6.9407
6.9767	7.1890	7.1998	7.2062	7.2277	7.2482
7.3705	7.4629	7.4712	7.4718	7.4789	7.5370
7.5836	7.5918	7.6439	7.6680	7.6849	7.7195
7.7580	7.7678	7.7701	7.8312	7.9427	8.0283
8.5051	8.5554	8.6063	8.6117	8.7343	9.0266
9.1448	9.5524	9.5701	9.5735	9.6155	9.7570
9.8965	9.9291	10.1056	10.1348	10.1396	10.2594
10.3244	10.4746	10.5573	11.2015	11.4795	11.6461
12.4411	12.9942	13.5685	13.7095	13.9225	13.9251
13.9372	14.3050	14.4415	16.2122	20.8130	22.1897
23.1281	23.3795	24.1381	25.0364;		

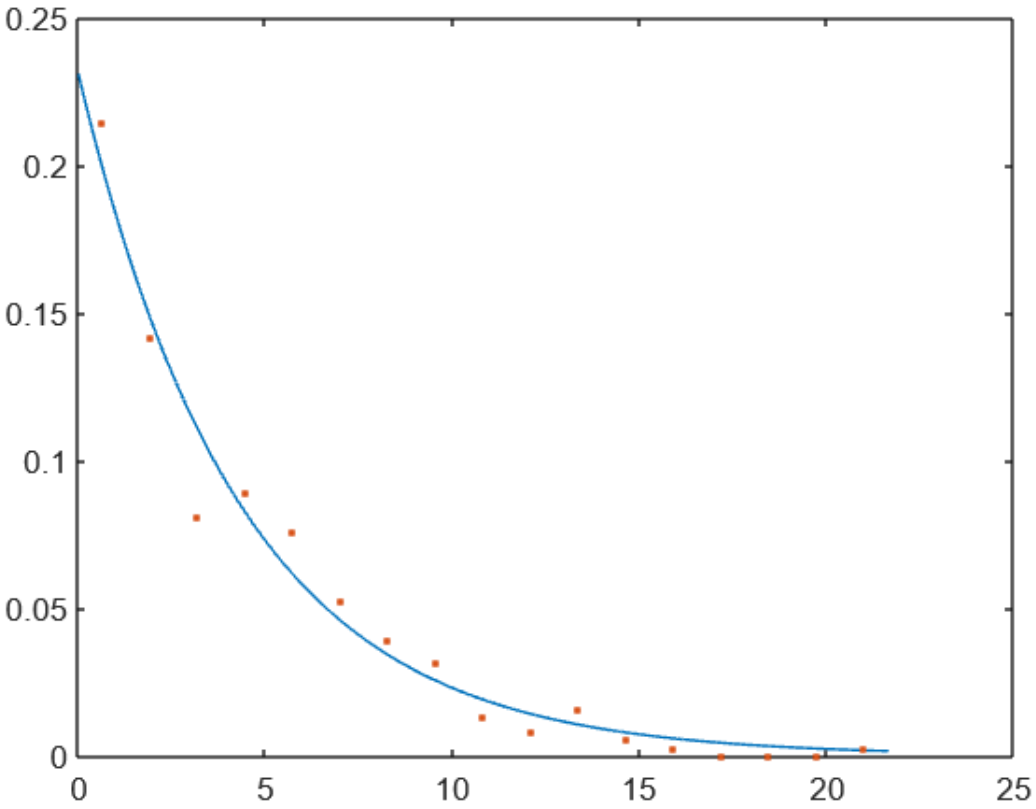
- Media aritmetică a eşantionului de valori (ma): 4.33;

- Valoarea maximă a intervalului de analiză (v) pentru prelucrarea preliminară: 16.22;

- Numărul de subintervale cu care se lucrează: 17;

- Estimarea iniţială (lambda0) a parametrului funcţiei de repartiţie: 0.2309;

- Compararea grafică a funcţiei teoretice cu cea empirică în intervalul de analiză [0, 5/lambda0]:

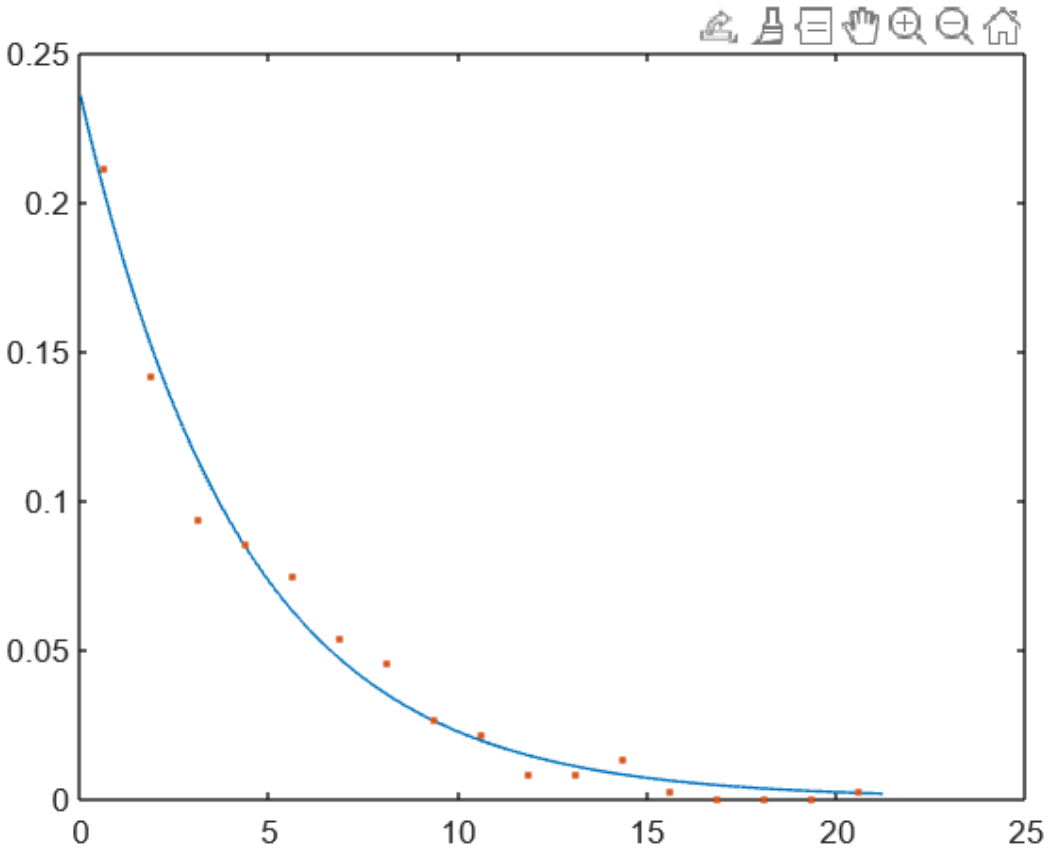


- Valoarea finală (lambdaF) adoptată în urma procesului de căutare: 0.2358;

- Valoarea funcţiei criteriu pentru lambdaF: 9.7213e-04;

- Valorile minimă şi maximă pentru intervalul final de căutare: 0.1155 și 0.3464;

- Compararea grafică a funcţiei teoretice adoptate cu cea empirică în intervalul de analiză [0, 5/lambdaF]:



b. Al doilea eșantion:

- Eşantionul de valori direct sortat (nu cel iniţial) : 0.0198 0.0422

0.0452	0.0484	0.0492	0.0532	0.0574	0.0649
0.0652	0.0697	0.0825	0.1017	0.1283	0.1639
0.1704	0.1903	0.1926	0.1962	0.2017	0.2034
0.2178	0.2216	0.2240	0.2389	0.2581	0.2983
0.3346	0.3579	0.3599	0.3722	0.3747	0.3813
0.3827	0.3961	0.4213	0.4309	0.4436	0.4554
0.4572	0.4721	0.5359	0.5361	0.5376	0.5478
0.5551	0.6046	0.6378	0.6486	0.6595	0.7002
0.7220	0.7396	0.7575	0.7606	0.8285	0.8298
0.8630	0.8643	0.8673	0.8791	0.9308	0.9388
0.9539	0.9539	0.9996	1.0044	1.0073	1.0124

1.0202	1.0496	1.0582	1.1319	1.1443	1.1489
1.1569	1.1627	1.1754	1.1785	1.2014	1.2022
1.2308	1.2388	1.2493	1.2563	1.2689	1.2913
1.3259	1.3276	1.3362	1.3430	1.3646	1.3654
1.3771	1.3931	1.4062	1.4357	1.4788	1.4794
1.4861	1.4996	1.5456	1.5533	1.5566	1.5757
1.6801	1.7194	1.7327	1.7580	1.7583	1.7607
1.7657	1.7704	1.7904	1.7932	1.7993	1.8264
1.8440	1.8517	1.8602	1.9002	1.9047	1.9048
1.9132	2.0016	2.0160	2.0628	2.1025	2.1099
2.1741	2.2546	2.2649	2.2814	2.3110	2.3193
2.3283	2.4026	2.4069	2.4592	2.4610	2.4678
2.4773	2.5103	2.5885	2.5929	2.6135	2.6269
2.6349	2.6397	2.7122	2.7719	2.8180	2.8243
2.8337	2.8537	2.9194	2.9262	2.9441	2.9730
2.9785	3.0622	3.1214	3.1402	3.1611	3.1885
3.1895	3.2159	3.2427	3.2604	3.2861	3.2897
3.3483	3.4114	3.4301	3.4536	3.5097	3.5234
3.5660	3.6621	3.6727	3.7056	3.7237	3.7875
3.8114	3.8427	3.9055	3.9645	3.9970	4.0151
4.0815	4.0840	4.1171	4.1604	4.2177	4.2339
4.2474	4.3834	4.4023	4.4142	4.4870	4.4892
4.5658	4.6095	4.6512	4.6526	4.7158	4.7193
4.7991	4.8118	4.8319	4.8347	4.9742	5.0726
5.1414	5.1575	5.1807	5.1927	5.2337	5.2433
5.3384	5.3434	5.3699	5.4048	5.4208	5.6220
6.0081	6.0309	6.0486	6.0794	6.0796	6.1087
6.1718	6.1766	6.2191	6.2934	6.3154	6.3205
6.3218	6.4946	6.4985	6.5181	6.5289	6.5451
6.5635	6.6146	6.6715	6.7984	6.8058	6.8439
6.8557	6.8933	6.9338	6.9996	7.0698	7.3293
7.3769	7.4412	7.6118	7.6887	7.7326	7.7442
7.8553	7.8784	8.0926	8.1827	8.1932	8.2775
8.4454	8.4894	8.6078	8.6925	8.7101	9.0273
9.0779	9.1976	9.3417	9.3909	9.6900	9.8674
9.9442	10.0284	10.0348	10.4456	10.5376	10.7216
10.7933	11.0792	11.0992	11.4072	11.5631	11.8625
11.9468	12.5584	13.2094	13.4397	13.9425	15.0877
15.8588	16.8544	17.8580	19.6582		

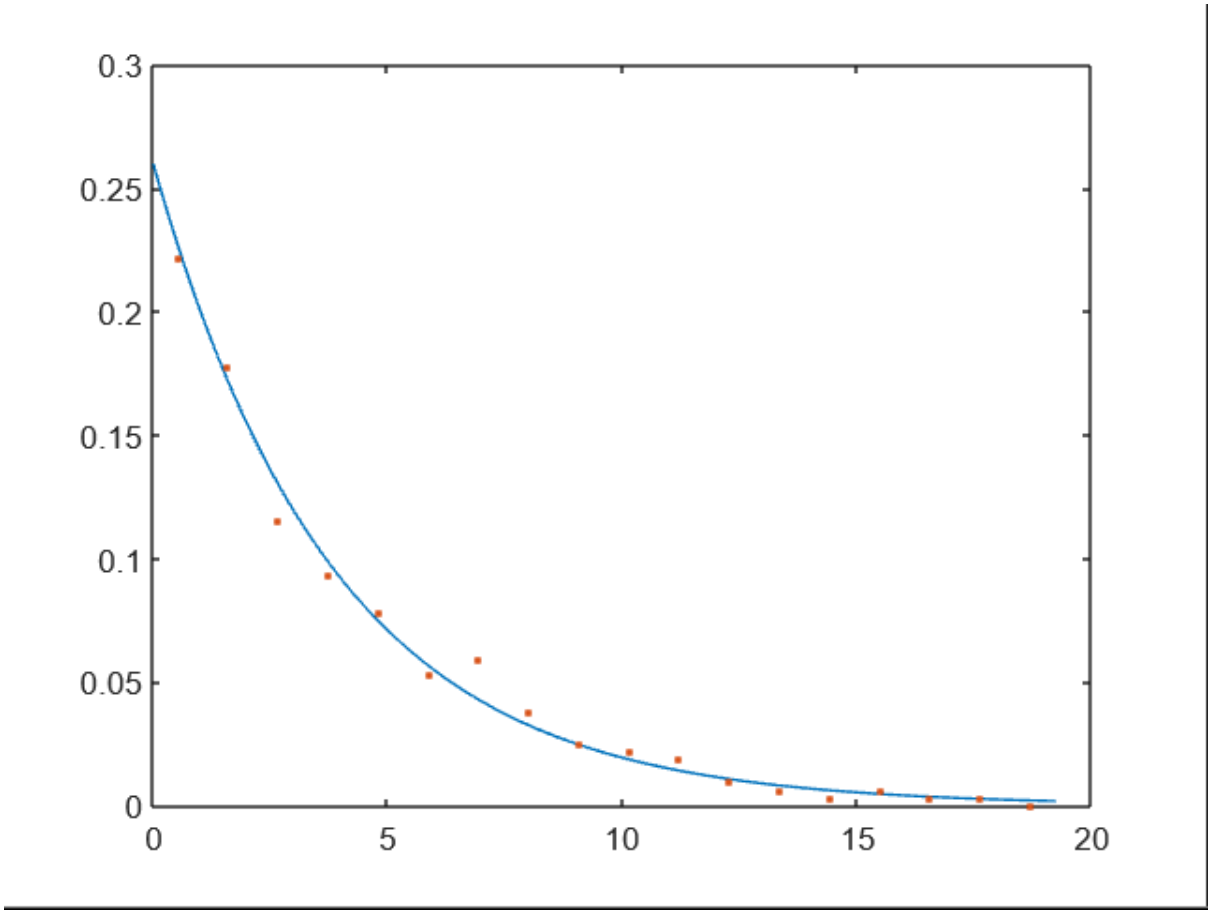
- Media aritmetică a eşantionului de valori (ma): 3.8484;

- Valoarea maximă a intervalului de analiză (v) pentru prelucrarea preliminară: 15.86;

- Numărul de subintervale cu care se lucrează: 18;

- Estimarea iniţială (lambda0) a parametrului funcţiei de repartiţie: 0.2599;

- Compararea grafică a funcţiei teoretice cu cea empirică în intervalul de analiză [0, 5/lambda0]:

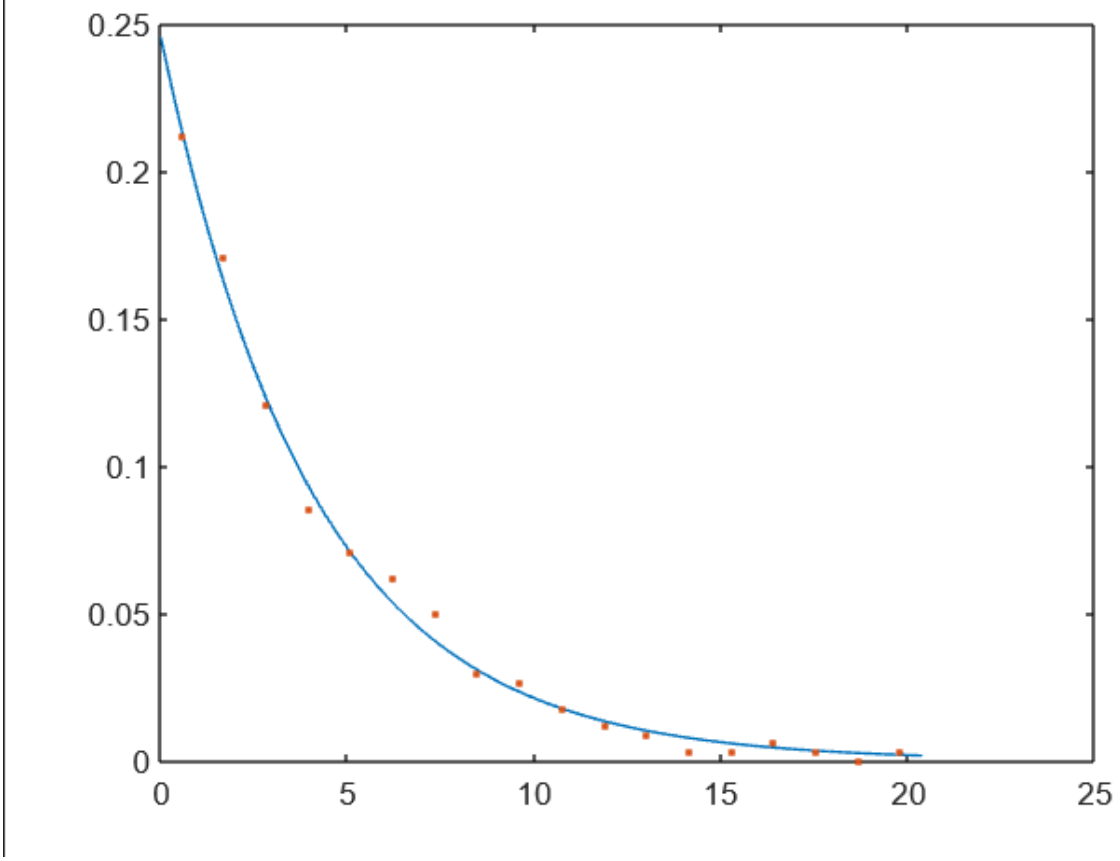


- Valoarea finală (lambdaF) adoptată în urma procesului de căutare: 0.2453;

- Valoarea funcţiei criteriu pentru lambdaF: 3.6610e-04;

- Valorile minimă şi maximă pentru intervalul final de căutare: 0.1299 și 0.3898;

- Compararea grafică a funcţiei teoretice adoptate cu cea empirică în intervalul de analiză [0, 5/lambdaF]:



c. Al treilea eșantion:

- Eşantionul de valori direct sortat (nu cel iniţial) : 0.0013 0.0022

0.0059	0.0064	0.0067	0.0070	0.0074	0.0077
0.0118	0.0124	0.0124	0.0128	0.0129	0.0157
0.0208	0.0216	0.0217	0.0219	0.0227	0.0229
0.0229	0.0247	0.0256	0.0274	0.0292	0.0325
0.0347	0.0352	0.0357	0.0378	0.0393	0.0395
0.0415	0.0428	0.0434	0.0469	0.0486	0.0490
0.0495	0.0498	0.0513	0.0527	0.0537	0.0597
0.0609	0.0614	0.0616	0.0617	0.0624	0.0645
0.0651	0.0653	0.0666	0.0679	0.0686	0.0718
0.0732	0.0749	0.0783	0.0787	0.0810	0.0813
0.0815	0.0819	0.0829	0.0829	0.0850	0.0852
0.0877	0.0889	0.0902	0.0903	0.0916	0.0950
0.0954	0.0956	0.0983	0.0984	0.1020	0.1023
0.1025	0.1039	0.1045	0.1045	0.1057	0.1070

0.1071	0.1092	0.1132	0.1152	0.1157	0.1162
0.1167	0.1185	0.1204	0.1206	0.1215	0.1218
0.1281	0.1319	0.1325	0.1336	0.1338	0.1341
0.1386	0.1387	0.1435	0.1467	0.1482	0.1482
0.1501	0.1506	0.1550	0.1596	0.1602	0.1618
0.1624	0.1637	0.1645	0.1649	0.1675	0.1680
0.1685	0.1713	0.1743	0.1756	0.1765	0.1780
0.1807	0.1830	0.1869	0.1875	0.1900	0.1902
0.1936	0.1955	0.1960	0.1969	0.1975	0.2023
0.2028	0.2044	0.2059	0.2114	0.2155	0.2158
0.2211	0.2257	0.2266	0.2280	0.2284	0.2289
0.2309	0.2310	0.2397	0.2402	0.2411	0.2432
0.2443	0.2456	0.2486	0.2498	0.2502	0.2520
0.2560	0.2568	0.2650	0.2656	0.2706	0.2734
0.2736	0.2756	0.2793	0.2815	0.2821	0.2828
0.2876	0.2910	0.2916	0.2944	0.2950	0.2961
0.2961	0.2981	0.3008	0.3020	0.3056	0.3093
0.3124	0.3139	0.3160	0.3165	0.3181	0.3212
0.3234	0.3239	0.3251	0.3253	0.3279	0.3313
0.3319	0.3368	0.3371	0.3392	0.3407	0.3439
0.3462	0.3497	0.3536	0.3621	0.3651	0.3692
0.3697	0.3715	0.3765	0.3825	0.3981	0.4110
0.4196	0.4214	0.4217	0.4360	0.4366	0.4384
0.4396	0.4429	0.4463	0.4477	0.4533	0.4665
0.4715	0.4752	0.4754	0.4787	0.4800	0.4862
0.4930	0.4976	0.4993	0.5035	0.5147	0.5164
0.5308	0.5359	0.5383	0.5409	0.5533	0.5540
0.5558	0.5605	0.5644	0.5754	0.5782	0.5804
0.5817	0.5837	0.6214	0.6400	0.6572	0.6719
0.6859	0.6911	0.7039	0.7057	0.7109	0.7176
0.7325	0.7800	0.7904	0.8054	0.8248	0.8264
0.8286	0.8315	0.8337	0.8485	0.8782	0.8817
0.8983	0.9030	0.9094	0.9183	0.9466	0.9565
0.9699	0.9704	0.9886	1.0233	1.0286	1.0779
1.0784	1.0847	1.1651	1.1757	1.2010	1.3757
1.4161	1.4190	1.4813	1.7352		

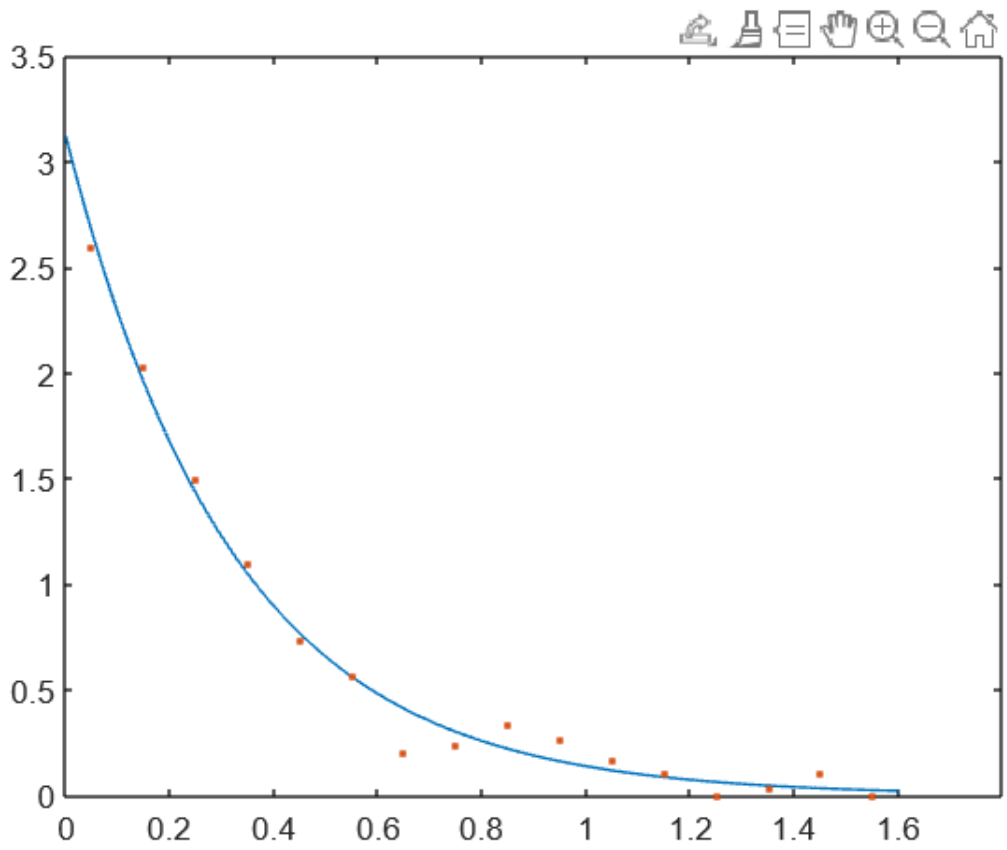
- Media aritmetică a eşantionului de valori (ma): 0.3205;

- Valoarea maximă a intervalului de analiză (v) pentru prelucrarea preliminară:1.0848;

- Numărul de subintervale cu care se lucrează: 16;

- Estimarea inițială (lambda0) a parametrului funcţiei de repartiţie: 3.1202;

- Compararea grafică a funcţiei teoretice cu cea empirică în intervalul de analiză [0, 5/lambda0]:

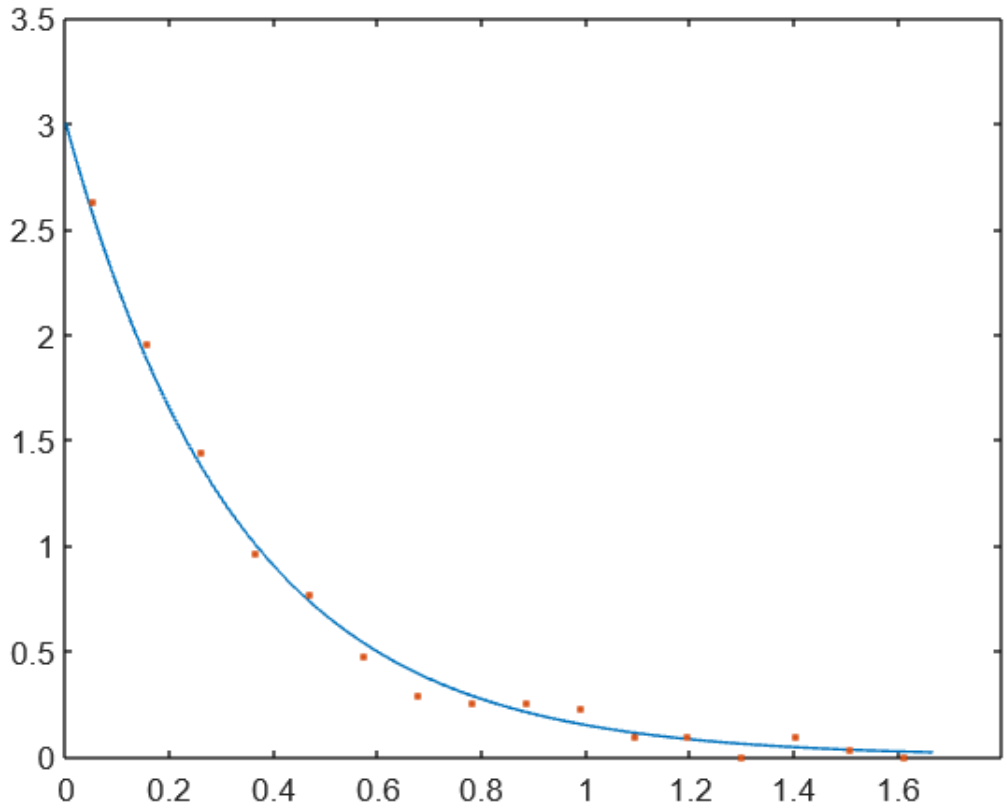


- Valoarea finală (lambdaF) adoptată în urma procesului de căutare: 3.0016;

- Valoarea funcţiei criteriu pentru lambdaF: 0.0463;

- Valorile minimă şi maximă pentru intervalul final de căutare: 1.5601 și 4.6803;

- Compararea grafică a funcţiei teoretice adoptate cu cea empirică în intervalul de analiză [0, 5/lambdaF]:



---

d. Al patrulea eșantion:

- Eşantionul de valori direct sortat (nu cel iniţial) : 0.0003 0.0018

0.0019	0.0028	0.0039	0.0044	0.0048	0.0073
0.0080	0.0122	0.0127	0.0144	0.0149	0.0157
0.0184	0.0190	0.0193	0.0209	0.0246	0.0262
0.0263	0.0267	0.0274	0.0300	0.0303	0.0307
0.0319	0.0330	0.0341	0.0346	0.0347	0.0348
0.0362	0.0384	0.0386	0.0397	0.0406	0.0434
0.0448	0.0453	0.0481	0.0492	0.0515	0.0530
0.0532	0.0553	0.0554	0.0581	0.0589	0.0595
0.0621	0.0626	0.0630	0.0642	0.0652	0.0656
0.0695	0.0718	0.0720	0.0725	0.0732	0.0756
0.0763	0.0770	0.0779	0.0781	0.0790	0.0797
0.0811	0.0819	0.0840	0.0848	0.0900	0.0914
0.0923	0.0925	0.0934	0.0951	0.0982	0.0989
0.0995	0.1017	0.1026	0.1038	0.1053	0.1055
0.1057	0.1099	0.1108	0.1114	0.1147	0.1150
0.1222	0.1227	0.1229	0.1259	0.1268	0.1281
0.1310	0.1311	0.1316	0.1319	0.1350	0.1366

0.1381	0.1391	0.1393	0.1413	0.1413	0.1422
0.1469	0.1477	0.1531	0.1555	0.1563	0.1572
0.1576	0.1578	0.1582	0.1600	0.1624	0.1629
0.1633	0.1654	0.1665	0.1768	0.1775	0.1781
0.1816	0.1840	0.1841	0.1842	0.1890	0.1912
0.1918	0.1952	0.1955	0.1986	0.1993	0.1997
0.1999	0.2035	0.2045	0.2094	0.2097	0.2102
0.2122	0.2177	0.2190	0.2247	0.2249	0.2270
0.2273	0.2274	0.2275	0.2317	0.2351	0.2358
0.2395	0.2418	0.2456	0.2491	0.2495	0.2522
0.2584	0.2588	0.2622	0.2629	0.2662	0.2666
0.2719	0.2770	0.2784	0.2794	0.2801	0.2856
0.2881	0.2891	0.2962	0.3035	0.3192	0.3266
0.3276	0.3367	0.3409	0.3421	0.3423	0.3427
0.3436	0.3448	0.3498	0.3500	0.3505	0.3537
0.3650	0.3682	0.3706	0.3719	0.3728	0.3755
0.3791	0.3856	0.3865	0.3867	0.3949	0.4011
0.4092	0.4127	0.4248	0.4296	0.4304	0.4326
0.4474	0.4509	0.4521	0.4734	0.4762	0.4864
0.4865	0.4925	0.4982	0.4992	0.5124	0.5140
0.5170	0.5173	0.5175	0.5217	0.5239	0.5298
0.5449	0.5468	0.5501	0.5503	0.5588	0.5618
0.5645	0.5681	0.5738	0.5745	0.5749	0.5820
0.5914	0.5919	0.5980	0.6006	0.6051	0.6078
0.6098	0.6098	0.6100	0.6139	0.6278	0.6364
0.6429	0.6484	0.6503	0.6623	0.6677	0.6723
0.6738	0.6972	0.7124	0.7166	0.7365	0.7423
0.7431	0.7436	0.7445	0.7622	0.7711	0.7724
0.7787	0.7992	0.8195	0.8344	0.8377	0.8387
0.8435	0.8525	0.8800	0.9306	0.9391	0.9425
0.9803	1.0044	1.0543	1.0647	1.0707	1.0713
1.0937	1.1630	1.1703	1.1813	1.2419	1.3449
1.4111	1.5240	1.6176	2.1687		

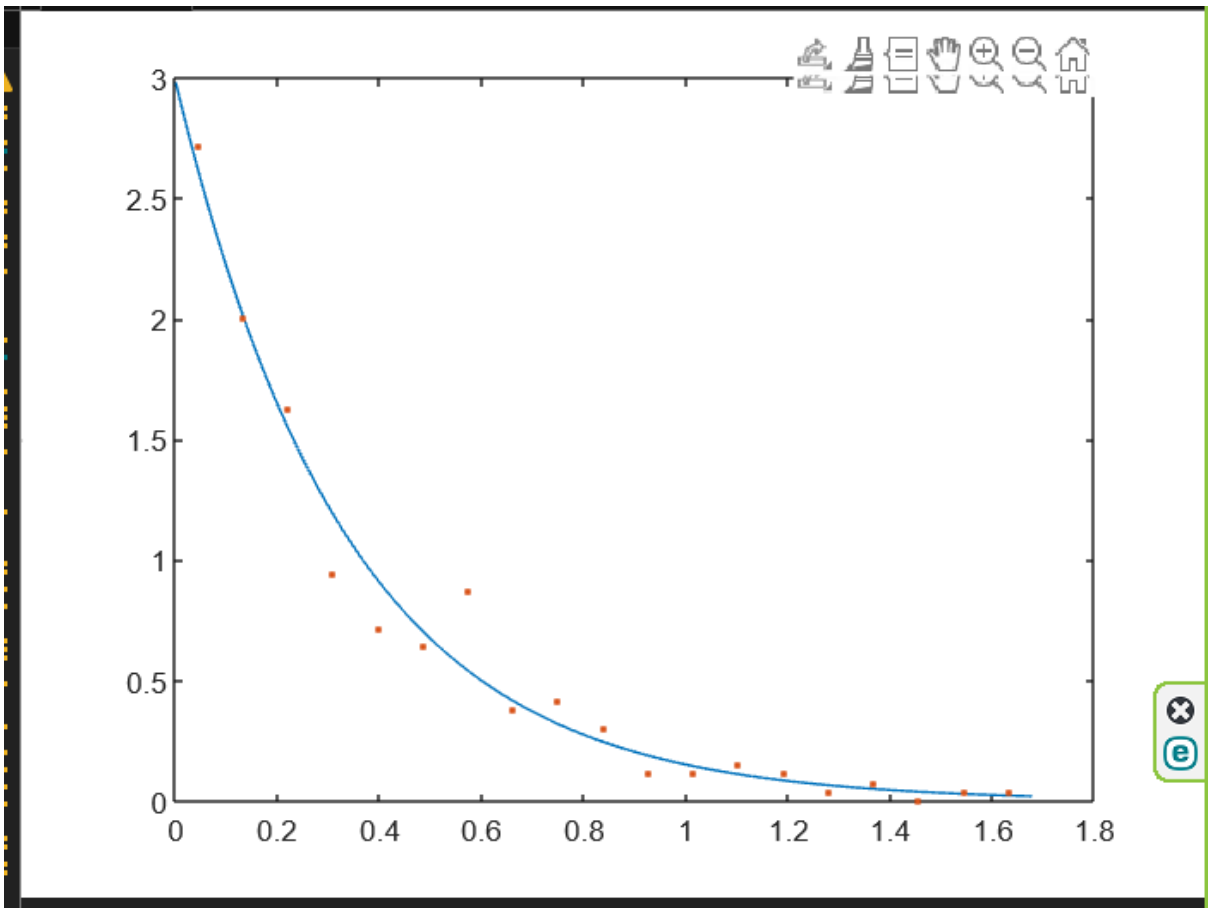
- Media aritmetică a eşantionului de valori (ma): 0.3354;

- Valoarea maximă a intervalului de analiză (v) pentru prelucrarea preliminară:1.0938;

- Numărul de subintervale cu care se lucrează: 19;

- Estimarea iniţială (lambda0) a parametrului funcţiei de repartiţie: 2.9815;

- Compararea grafică a funcţiei teoretice cu cea empirică în intervalul de analiză [0, 5/lambda0]:

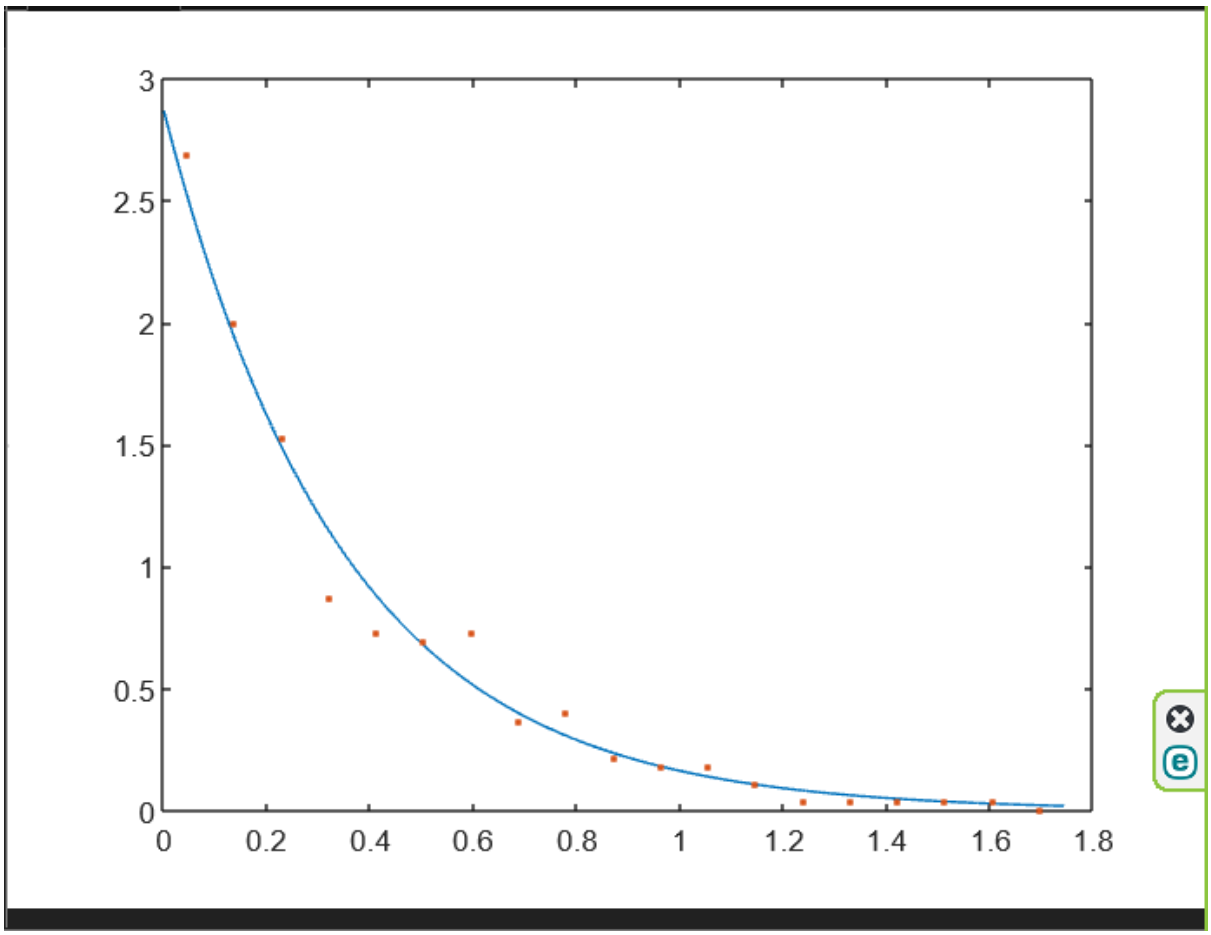


- Valoarea finală (lambdaF) adoptată în urma procesului de căutare: 2.8682;

- Valoarea funcţiei criteriu pentru lambdaF: 0.1909;

- Valorile minimă şi maximă pentru intervalul final de căutare: 1.4907 și 4.4722;

- Compararea grafică a funcţiei teoretice adoptate cu cea empirică în intervalul de analiză [0, 5/lambdaF]:



3. Codul sursă folosit la prelucrarea eşantioanelor de valori:

```
E = []
N = length(E)

E_sort = sort(E)
ma = mean(E)

E_sort

v = 16.22
k = floor(1+3.222*log(N))

delta = v/k;
x = delta:delta:v
n = zeros(1,k)
for i = 1:N
    j = ceil(E_sort(i)/delta)
    if(j <= k)
        n(j) = n(j)+1;
    end
end
```

```
sum(n)
n

lam0 = 1/ma;
d = 5/lam0/1000;
z = 0:d:5/lam0;
f = lam0 * exp(-lam0*z)

delta = 5/lam0/k
x = delta:delta:5/lam0
n = zeros(1,k)
for i = 1:N
    j = ceil(E_sort(i)/delta)
    if(j <= k)
        n(j) = n(j)+1;
    end
end

y = x - delta/2
fe = n/N/delta;

plot(z,f,'-',y,fe,'.')

st = 0.5 * lam0
dr = 1.5 * lam0

pc = (dr-st)/1000

Spdm = 1000

for lam = st:pc:dr
    delta = 5/lam/k
    x = delta:delta:5/lam
    n = zeros(1,k)
    for i = 1:N
        j = ceil(E_sort(i)/delta)
        if(j <= k)
            n(j) = n(j)+1;
        end
    end
    y = x-delta/2
    fe = n/N/delta;
    f = lam*exp(-lam*y)

    Spd = sum((f-fe).^2)
    if Spd<Spdm
        Spdm = Spd
        lamF = lam
    end
end

st
dr
lamF
Spdm
```

4. Concluzii de etapă:

Pentru toate cele 4 variabile aleatoare TfA, TfB, TrA și TrB s-a adoptat o lege de distribuție exponențial-negativă, valorile parametrilor fiind:

- lambda_a = 0.2358;
- lambda_b = 0.2453;
- mu_a = 3.0016;
- mu_b = 2.8682.

Etapa 2:

1. **Prezentarea modelului de simulare**

a.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. Modelul evidenţiază un sistem de servire cu cereri de tip diferit. Principala diferenţă faţă de modelele studiate anterior apare însă la fluxul de alimentare cu cereri.

b.fluxul opririlor:

Faţă de toate sistemele studiate anterior, în acest caz rata medie a cererilor pentru muncitorul de deservire nu mai este constantă în timp întrucât depinde de numărul de sisteme în funcţiune (nf). Rata este maximă atunci când toate sistemele funcţionează şi devine 0 când toate sistemele sunt oprite. Acest aspect al unui flux de intrare variabil în timp nu a mai fost studiat până acum.De la prima etapă a rezultat că cele două variabile aleatoare primare TfA şi TfB au repartiţiiexponenţial 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. 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 $genExp(\lambda)$.

c.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. Se deduc uşor relațiile de calcul:

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

-de observat că

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

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

$Trm = pATrmA + pBTrmB$

-dar variabilele aleatoare TrA şi TrB au repartiţii exponenţial negative, de parametru μA şi respectiv, μB. Ca urmare, $TrmA = 1\ \mu A$ şi $TrmB = 1\ \mu B$. Pentru timpul mediu de remediere a unui sistem oprit rezultă relaţia de calcul:

$Trm = pATrmA + pBTrmB = \lambda A\ (\lambda A + \lambda B)/ \cdot 1\ \mu A + \lambda B\ (\lambda A + \lambda B) \cdot 1/\ \mu B$
$=1\ (\lambda A + \lambda B) \cdot (\lambda A\ \mu A + \lambda B\ \mu B)$

Acest rezultat este foarte util la validarea programului de simulare. Pentru generarea valorilor de selecţie privind timpul de remediere, care să reflecte amestecul celor două tipuri de operaţii care se succed în mod aleatoriu, cu respectarea proporţiei date de parametrii λA și λB, se procedează așa cum este prezentat în cursul 4, care tratează sistemele de servire cu cereri de tip diferit.

2. **Programul de simulare**:

```
#include <iostream>
#include <cmath>
using namespace std;

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

```
void simulare(double lambdaA, double lambdaB, double miuA, double miuB, int s,
double& ds, double& d, double &o, double&trm ,int& no1,int& nr1)
{
	double str = 0;
	double stf = 0;
	double ceas = 0;
	int nf = s;
	int NO = 1000000;
	int no = 0;
	int No = 0;
	int Nr = 0;
	double tr = 0;
	double raport = 1 / (lambdaA + lambdaB) * ((lambdaA / miuA) + (lambdaB /
miuB));
	double tpo = genExp(nf * (lambdaA + lambdaB));
	do
	{
		if ((nf!=0 && tpo < tr) || (nf==s))
		{
			No++;
			ceas += tpo;
			if (no > 0)
			{
				tr -= tpo;
			}
			stf += nf * tpo;
			nf--;
			no++;
			if (no == 1)
			{
				tr = genExp(1/raport);
				str += tr;
			}

		}
		else
		{
			Nr++;
			ceas += tr;
			stf += nf * tr;
			nf++;
			no--;
			if (no > 0)
			{
				tr = genExp(1/ raport);
				str += tr;
			}
		}
		if (nf > 0)
		{
			tpo = genExp(nf * (lambdaA + lambdaB));
		}

	} while (No < NO);
	ds = ceas;
	d = stf / ds / s * 100;
	o = str / ds * 100;
	trm = str / Nr;
	nr1 = Nr;
	no1 = No;
}

int main()
```

```
{
	double lambdaA = 0.2358;
	double lambdaB = 0.2453;
	double miuA = 3.0016;
	double miuB = 2.8682;
	double ds = 0;
	double d1 = 0;
	double d[11] = { 0 };
	double o1 = 0;
	double o[11] = { 0 };
	double trm = 0;
	int nr1 = 0;
	int no1 = 0;
	cout << fixed;

	for (int s = 1; s <= 9; s++)
	{
		simulare(lambdaA, lambdaB, miuA, miuB, s, ds, d1, o1,trm,no1,nr1);
		o[s] = o1;
		d[s] = d1;
		cout << "====sisteme: " << s << "====" << endl;
		cout << "====verificari====" << endl;
		cout << "Verificari preliminare:" << endl;
		cout << "     " << nr1 << " ~= " << no1 << endl;
		cout << "     " << trm << " ~= " << "0.3492" << endl;
		cout << "Verificari cantitative:" << endl;
		if (s == 1)
		{
			cout << "    d+o=100" << d[1] + o[1] << endl;
			cout << "  d: " << d[1] << " ~= " << "86.55" << endl;
		}
		cout << "     " << no1 << " ~= " << ds * (d[s] / 100) * s * (lambdaA
+ lambdaB) << endl;
		cout << "====rezultate====" << endl;
		cout << "durata_simularii: " << ds << endl;
		cout << "disponibilitate: " << d[s] << endl;
		cout << "ocupare: " << o[s] << endl;
		cout << "Verificare calitativa: " << o[s] << " ~= " << s * o[1] <<
endl;
	}
	return 0;
}
```

3. **Verificarea 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:

- $Nr \simeq No$ – se verifică astfel secvenţa pentru determinarea evenimentului următor;
- $STr/Nr \simeq Trm = 1/(\lambda A+\lambda B)\cdot (\lambda A/\mu A+ \lambda B/\mu B)$ – se verifică astfel funcţia de generare a timpilor de remediere $genTr()$.

b) Verificări cantitative privind rezultatele simulării

- $S = 1$ – se verifică rezultatele simulării cu relaţiile:

  - $D = (1 + \lambda A/\mu A + \lambda B/\mu B)$ la puterea minus 1 ori 100%;

  - $D + O = 100$ %

- $S \geq 2$ – 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.

Acestea sunt rezultatele verificarilor pana cand gradul de ocupare depaseste 90%:

====sisteme: 1====

====verificari====

Verificari preliminare:

999999 ~= 1000000

0.3414 ~= 0.3492

Verificari cantitative:

d+o=100.000001

d: 85.9403 ~= 86.55

1000000 ~= 1001075.7243

====rezultate====

durata_simularii: 2422275.0203

disponibilitate: 85.9403

ocupare: 14.0598

Verificare calitativa: 14.0598 ~= 14.0598

====sisteme: 2====

====verificari====

Verificari preliminare:

    999999 ~= 1000000

    0.3412 ~= 0.3492

Verificari cantitative:

    1000000 ~= 999852.2144

====rezultate====

durata_simularii: 1233649.1015

disponibilitate: 84.2001

ocupare: 27.6932

Verificare calitativa: 27.6604 ~= 28.1940

====sisteme: 3====

====verificari====

Verificari preliminare:

    999999 ~= 1000000

    0.3409 ~= 0.3492

Verificari cantitative:

    1000000 ~= 999231.5470

====rezultate====

durata_simularii: 841860.1567

disponibilitate: 82.2722

ocupare: 40.4671

Verificare calitativa: 40.5037 ~= 42.2911

====sisteme: 4====

====verificari====

Verificari preliminare:

999999 ~= 1000000

0.3405 ~= 0.3492

Verificari cantitative:

1000000 ~= 998826.3346

====rezultate====

durata_simularii: 649460.7575

disponibilitate: 79.9342

ocupare: 52.4043

Verificare calitativa: 52.4391 ~= 56.3881

====sisteme: 5====

====verificari====

Verificari preliminare:

999999 ~= 1000000

0.3410~= 0.3492

Verificari cantitative:

1000000 ~= 1000644.3670

====rezultate====

durata_simularii: 539087.0092

disponibilitate: 77.1588

ocupare: 63.3918

Verificare calitativa: 63.2690 ~= 70.4852

====sisteme: 6====

====verificari====

Verificari preliminare:

999999 ~= 1000000

0.3406 ~= 0.3492

Verificari cantitative:

1000000 ~= 999502.8847

====rezultate====

durata_simularii: 467810.4969

disponibilitate: 74.0163

ocupare: 72.9478

Verificare calitativa: 72.8095 ~= 84.5822

====sisteme: 7====

====verificari====

Verificari preliminare:

999999 ~= 1000000

0.3412 ~= 0.3492

Verificari cantitative:

1000000 ~= 1000096.9870

====rezultate====

durata_simularii: 421652.0254

disponibilitate: 70.4415

ocupare: 80.9174

Verificare calitativa: 80.9295 ~= 98.6792

====sisteme: 8====

====verificari====

Verificari preliminare:

999999 ~= 1000000

0.341033 ~= 0.3492

Verificari cantitative:

1000000 ~= 1000829.1990

====rezultate====

durata_simularii: 390643.2700

disponibilitate: 66.5052

ocupare: 87.356

Verificare calitativa: 87.3001 ~= 112.7763

====sisteme: 9====

====verificari====

Verificari preliminare:

999999 ~= 1000000

0.3408 ~= 0.3492

Verificari cantitative:

1000000 ~= 1000563.4788

====rezultate====

durata_simularii: 370278.3175

disponibilitate: 62.3994

ocupare: 92.1179

Verificare calitativa: 92.0614 ~= 126.8733

4. **Tabele cu rezultatele simulării:**

S	1	2	3	4	5	6	7	8	9
D(%)	85.94	84.20	82.27	79.93	77.15	74.01	70.44	66.50	62.40
O(%)	14.05	27.69	40.46	52.40	63.39	72.94	80.92	87.35	92.11

5. **Concluzii**:

În concluzie, în această etapă am implementat un algoritm de simulare pentru un sistem de servire cu o stație unde fluxul de cereri depinde de numărul de sisteme în funcțiune. Prin simulare, am determinat disponibilitatea sistemelor și ocuparea muncitorului când unei stații îi sunt atribuite un anumit număr de sisteme. Am încheiat simularea când gradul de ocupare a depășit 90%.

Etapa 3:

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 λ/μ este mai mare.

2. **Programul de simulare:**

```
3. import random

   ND = 10_000  # Numar defectari impus
   lambda_a = 0.2358
   lambda_b = 0.2453
   mu_a = 3.0016
   mu_b = 2.8682

   A = 0
   B = 1

   pasiva: float = 0.0
   semiactiva: float = 0.5
   activa: float = 1.0

   MFR = A
   MCR = B


   def gen_exp(lambda_: float) -> float:
       return random.expovariate(lambda_)


   def gen_tr(tip: int) -> float:
       if tip == A:
           return gen_exp(mu_a)
       elif tip == B:
           return gen_exp(mu_b)
```

```
    else:
        raise ValueError


def determinare_modul_defectat() -> int:
    return A if random.random() < lambda_a / (lambda_a + lambda_b)
else B


def urm_remediere_fara_prioritate(
    nmf: tuple[list[int], list[int]], S: int, St: list[bool],
fara_rezerva: bool = False
) -> tuple[int, int]:
    for i in range(S):
        if not St[i]:
            if nmf[A][i] < 1:
                return i, A
            elif nmf[B][i] < (1 if fara_rezerva else 2):
                return i, B
    for i in range(S):
        if nmf[A][i] < 1:
            return i, A
        elif nmf[B][i] < (1 if fara_rezerva else 2):
            return i, B
    raise ValueError


def urm_remediere_cu_prioritate(
    nmf: tuple[list[int], list[int]], S: int
) -> tuple[int, int]:
    prioritate = -1
    sr = 0
    mr = 0
    for i in range(S):
        if nmf[MFR][i] == 0:
            return i, MFR
        if nmf[MCR][i] == 0 and prioritate != 2:
            prioritate = 2
            sr = i
            mr = MCR
        elif nmf[MCR][i] == 0 and prioritate == 1:
            prioritate = 3
            sr = i
            mr = MCR
    return sr, mr


def det_tpd(Tf: list[float], St: list[bool]) -> float:
    val_min = None
    for i in range(len(Tf)):
        if St[i] and (val_min is None or Tf[i] < val_min):
            val_min = Tf[i]
    if val_min:
        return val_min
    else:
        raise ValueError


def simulare_fara_rezerva(S: int):
    nmf = ([1] * S, [1] * S)  # numar module functionale
    STf: float = 0  # Suma timpi functionare
```

```
    STr: float = 0  # Suma timpi remediere
    Nd = 0  # Numar defectari
    Nr = 0  # Numar remedieri
    ceas: float = 0
    Tf = [gen_exp(lambda_a + lambda_b) for _ in range(S)]  # Timp
functionare
    nf = S  # Numar sisteme functionale
    nmd = 0  # Numar module defecte
    St = [True] * S  # Stare sisteme, true -> functional
    Tpd: float = det_tpd(Tf, St)  # Timp pana la prima defectare
    sd = Tf.index(Tpd)  # Sistemul defectat
    md: int = determinare_modul_defectat()  # Modul defectat
    Tr: float = 0  # Timp remediere
    sr = 0  # Sistemul remediat
    mr = 0  # Modul remediat

    while Nd < ND:
        if nmd == 0 or (nf > 0 and Tpd < Tr):
            # aparitie intrerupere
            ceas += Tpd
            if nmd > 0:
                Tr -= Tpd
                STr += Tpd
            for i in range(S):
                if St[i]:
                    Tf[i] -= Tpd
            STf += nf * Tpd
            nmf[md][sd] -= 1
            nmd += 1
            Nd += 1

            if nmf[md][sd] == 0:
                nf -= 1
                St[sd] = False
            else:
                Tf[sd] = gen_exp(lambda_a + lambda_b)

            if nmd == 1:
                sr = sd
                mr = md
                Tr = gen_tr(mr)
        else:
            # terminare remediere in curs
            ceas += Tr
            for i in range(S):
                if St[i]:
                    Tf[i] -= Tr
            STr += Tr
            if nf > 0:
                STf += nf * Tr

            nmf[mr][sr] += 1
            nmd -= 1
            Tr = 0

            if St[sr] is False and nmf[A][sr] >= 1 and nmf[B][sr]
>= 1:
                Tf[sr] = gen_exp(lambda_a + lambda_b)
                St[sr] = True
                nf += 1
```

```
            if nmd >= 1:
                sr, mr = urm_remediere_fara_prioritate(nmf, S, St,
fara_rezerva=True)
                Tr = gen_tr(mr)

            Nr += 1

        if nf > 0:
            Tpd = det_tpd(Tf, St)
            sd = Tf.index(Tpd)
            md = determinare_modul_defectat()
    D = STf / (ceas * S) * 100
    O = STr / ceas * 100
    return D, O


def simulare_cu_rezerva(
    S: int,
    alfa: float,
    remediere_cu_prioritate: bool = False,
    intrerupere_remed_in_curs: bool = False,
):
    def lambda_m():
        return lambda_b if MCR == B else lambda_a

    def lambda_r():
        return lambda_m() * alfa

    def lambda_total(i: int):
        return lambda_a + lambda_b + lambda_r() * (nmf[MCR][i] -
1)

    def prob_defectiune_la_mfr(nr_sistem: int) -> float:
        return 1 - lambda_m() / lambda_total(nr_sistem)

    def determ_modul_defectat(nr_sistem: int):
        return MFR if random.random() <
prob_defectiune_la_mfr(nr_sistem) else MCR

    nmf = ([1] * S, [2] * S)  # numar module functionale
    STf: float = 0  # Suma timpi functionare
    STr: float = 0  # Suma timpi remediere
    Nd = 0  # Numar defectari
    Nr = 0  # Numar remedieri
    ceas: float = 0
    Tf = [gen_exp(lambda_total(i)) for i in range(S)]  # Timp
functionare
    nf = S  # Numar sisteme functionale
    nmd = 0  # Numar module defecte
    St = [True] * S  # Stare sisteme, true -> functional
    Tpd: float = det_tpd(Tf, St)  # Timp pana la prima defectare
    sd = Tf.index(Tpd)  # Sistemul defectat
    md: int = determ_modul_defectat(sd)  # Modul defectat
    Tr: float = 0  # Timp remediere
    sr = 0  # Sistemul remediat
    mr = 0  # Modul remediat

    while Nd < ND:
        if nmd == 0 or (nf > 0 and Tpd < Tr):
            # aparitie intrerupere
            ceas += Tpd
```

```
            if nmd > 0:
                Tr -= Tpd
                STr += Tpd
            for i in range(S):
                if St[i]:
                    Tf[i] -= Tpd
            STf += nf * Tpd
            nmf[md][sd] -= 1
            nmd += 1
            Nd += 1

            if nmf[md][sd] == 0:
                nf -= 1
                St[sd] = False
            else:
                Tf[sd] = gen_exp(lambda_total(sd))

            cond = nmd == 1
            if intrerupere_remed_in_curs:
                cond = (
                    cond
                    or (St[sr] is True and St[sd] is False)
                    or (sd == sr and md == MFR)
                )
            if cond:
                sr = sd
                mr = md
                Tr = gen_tr(mr)
        else:
            # terminare remediere in curs
            ceas += Tr
            for i in range(S):
                if St[i]:
                    Tf[i] -= Tr
            STr += Tr
            if nf > 0:
                STf += nf * Tr

            nmf[mr][sr] += 1
            nmd -= 1
            Tr = 0

            if nmf[A][sr] >= 1 and nmf[B][sr] >= 1:
                Tf[sr] = gen_exp(lambda_total(sr))
                if not St[sr]:
                    St[sr] = True
                    nf += 1

            if nmd >= 1:
                sr, mr = (
                    urm_remediere_fara_prioritate(nmf, S, St)
                    if not remediere_cu_prioritate
                    else urm_remediere_cu_prioritate(nmf, S)
                )
                Tr = gen_tr(mr)

            Nr += 1

        if nf > 0:
            Tpd = det_tpd(Tf, St)
            sd = Tf.index(Tpd)
```

```
            md = determ_modul_defectat(sd)
        D = STf / (ceas * S) * 100
        O = STr / ceas * 100
        return D, O


    def main():
        Ds = [[], [], [], []]
        Os = [[], [], [], []]
        print("============== Fara rezerva ==============")
        for i in range(1, 11):
            D, O = simulare_fara_rezerva(i)
            print(f"S = {i}\nD = {D:.2f}\nO = {O:.2f}\n")
            Ds[0].append(D)
            Os[0].append(O)

        aux = 0
        for val, name in [
            (activa, "Activa"),
            (semiactiva, "Semi-activa"),
            (pasiva, "Pasiva"),
        ]:
            print(f"\n============== Rezerva {name} ==============")
            for i in range(1, 11):
                D, O = simulare_cu_rezerva(i, val)
                print(f"S = {i}\nD = {D:.2f}\nO = {O:.2f}")
                Ds[aux + 1].append(D)
                Os[aux + 1].append(O)
                D, O = simulare_cu_rezerva(i, val,
    remediere_cu_prioritate=True)
                print(f"Remediere cu prioritate:\n\tD = {D:.2f}\n\tO =
    {O:.2f}")
                D, O = simulare_cu_rezerva(
                    i, val, remediere_cu_prioritate=True,
    intrerupere_remed_in_curs=True
                )
                print(f"Intrerupere remediere in curs:\n\tD =
    {D:.2f}\n\tO = {O:.2f}\n")
            aux += 1


    main()
```

4. Tabele cu rezultatele simulării:

Disponibilitate	1	2	3	4	5	6	7	8	9
Fără rezervă	85.94	84.20	82.27	79.93	77.15	74.01	70.44	66.50	62.40
Rezervă activă	85.48	83.35	80.71	77.33	73.03	70.68	66.64	63.36	60.31
Rezervă semi-activă	88.59	87.09	84.85	81.79	79.14	76.29	72.54	67.63	64.04
Rezervă pasivă	91.81	90.71	88.87	87.11	85.98	82.76	79.45	72.90	65.36

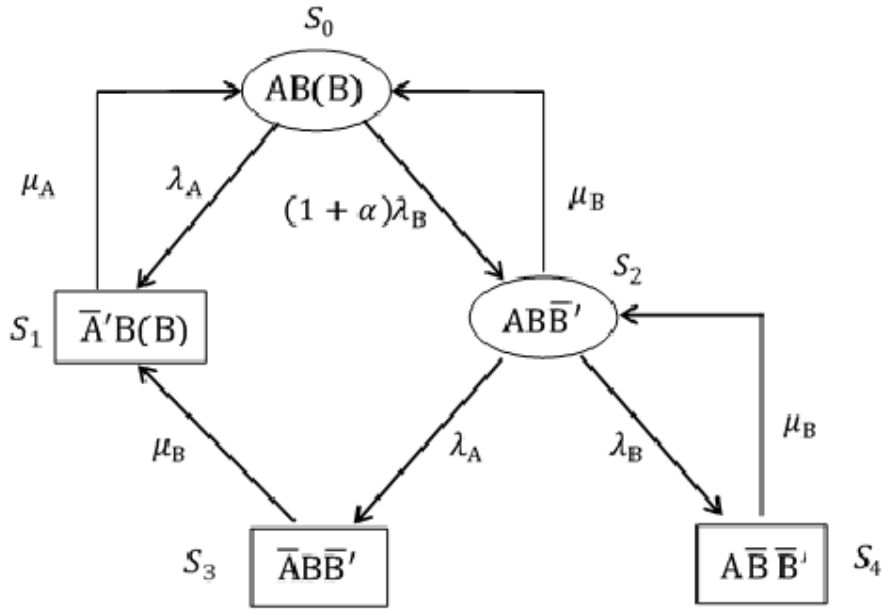
Ocupare	1	2	3	4	5	6	7	8	9
Fără rezervă	14.05	27.69	40.46	52.40	63.39	72.94	80.92	87.35	92.11
Rezervă activă	20.63	39.36	56.65	71.87	85.59	92.25	97.68	99.48	99.94
Rezervă semi-activă	17.75	34.42	50.13	65.73	78.49	88.06	94.40	98.34	99.73
Rezervă pasivă	14.75	29.19	43.93	56.86	68.57	81.21	88.63	96.11	99.40

Din aceste rezultate, se observă o îmbunătățire a disponibilității odată cu introducerea rezervei care crește odată cu pasivizarea acesteia.

Etapa 4:

1. Cazul S = 1, cu rezervă la modulul B

1.1 Modelul Markov pentru cazul S = 1, fără întreruperea remedierii în curs:



Matricea de adiacență a grafului obținut după modelul Markov:

	0	1	2	3	4
0	$-[\lambda_A + (1+\alpha)\cdot\lambda_B]$	μ_A	μ_B	0	0
1	λ_A	$-\mu_A$	0	μ_B	0
2	$(1+\alpha)\cdot\lambda_B$	0	$-(\mu_B+\lambda_A+\lambda_B)$	0	μ_B
3	0	0	λ_A	$-\mu_B$	0
4	0	0	λ_B	0	$-\mu_B$

Pentru toate cazurile analizate, stările în care sistemul este în funcţiune sunt $S0$ şi $S2$. Singura stare în care muncitorul este liber este $S0$. Prin urmare, disponibilitatea sistemului şi gradul de ocupare a muncitorului de deservire se determină cu relaţiile:

$$D = (p_0 + p_2) \cdot 100\ (\%)$$

$$O = (1 - p_0) \cdot 100\ (\%)$$

1.2 Programul de simulare:

```
clc
clear all

l_a = 0.2358
l_b = 0.2453
miu_a = 3.0016
miu_b = 2.8682
alfa = 0

A = [-(l_a+(1+alfa)*l_b),miu_a,miu_b,0,0;
    l_a,-miu_a,0,miu_b,0;
    (1+alfa)*l_b,0,-(miu_b+l_a+l_b),0,miu_b;
    0,0,l_a,-miu_b,0;
    0,0,l_b,0,-miu_b]

A_tilda = A;
A_tilda(1, :) = ones(1, 5);

B_tilda = zeros(5, 1);
B_tilda(1) = 1;

P = A_tilda \ B_tilda;

if all(P >= 0) && all(P <= 1) && abs(sum(P) - 1) < eps
    disp('Vectorul rezultat este valid.');
    disp('Vectorul de probabilități de stare:');
    disp(P);
else
    disp('Vectorul rezultat NU este valid.');
```

```
end

D = (P(1)+P(3))*100
O = (1-P(1))*100
```

1.3 Compararea rezultatelor dintre simulat și analitic pentru disponibilitate și gradul de ocupare:

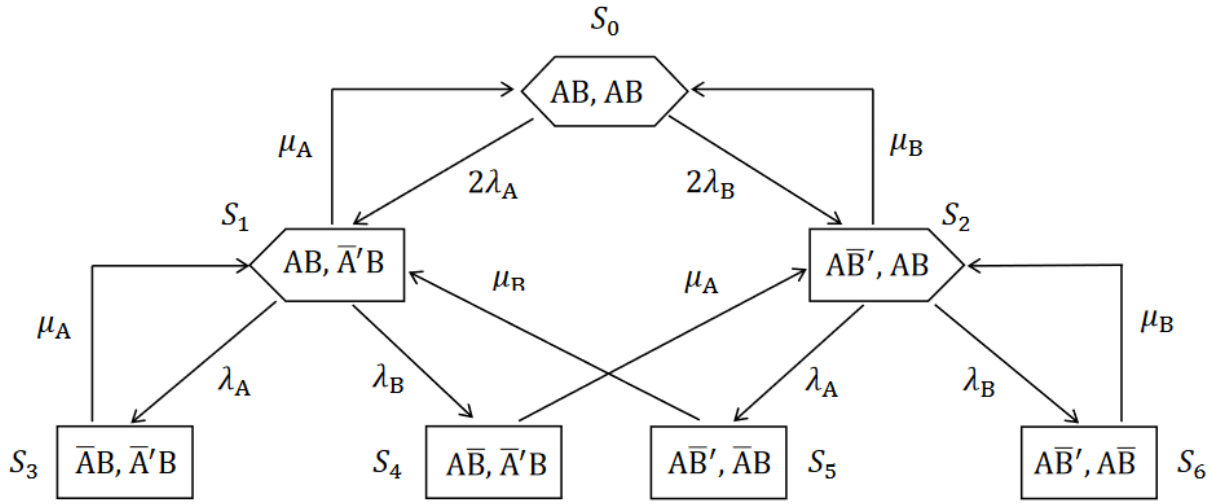
- alfa = 0, rezervă pasivă:
  - D simulat: 91.81;
  - D analitic: 91.6722;
  - O simulat: 14.75;
  - O analitic: 15.0418.

-alfa = 0.5, rezervă semi-activă:
  - D simulat: 88.59;
  - D analitic: 88.7120;
  - O simulat: 17.75;
  - O analitic: 18.4535.

-alfa = 1, rezervă activă:
  - D simulat: 85.48;
  - D analitic: 85.5304;
  - O simulat: 20.63;
  - O analitic: 21.6018.

2. Cazul S = 2, fără rezervă

2.1 Cum cele două sisteme sunt identice, nu se face distincţie explicită între ele. Ca urmare, stările în care cele două grupări ce descriu starea sistemelor sunt inversate se consideră echivalente; de exemplu, $(AB, \overline{A}'B)$ şi $(\overline{A}'B, AB)$. Acestea vor apărea în graf o singură dată. În felul acesta, rezultă un model Markov mai redus, cu mai puţine stări.



Matricea de adiacență a grafului obținut după modelul Markov:

	0	1	2	3	4	5	6
0	$-2\cdot(\lambda_A+\lambda_B)$	μ_A	μ_B	0	0	0	0
1	$2\lambda_A$	$-(\lambda_A+\lambda_B+\mu_A)$	0	μ_A	0	μ_B	0
2	$2\lambda_B$	0	$-(\lambda_A+\lambda_B+\mu_B)$	0	μ_A	0	μ_B
3	0	λ_A	0	$-\mu_A$	0	0	0
4	0	λ_B	0	0	$-\mu_A$	0	0
5	0	0	λ_A	0	0	$-\mu_B$	0
6	0	0	λ_B	0	0	0	$-\mu_B$

În starea $S0$ ambele sisteme sunt în funcţiune, în timp ce în stările $S1$ şi $S2$ un sistem este în funcţiune iar celălalt este oprit. Ca urmare, disponibilitatea sistemelor se exprimă cu relaţia:

$$D = \left(p_0 + \frac{p_1+p_2}{2}\right)\cdot 100\ (\%)$$

În ceea ce priveşte gradul de ocupare a muncitorului de deservire, rămâne valablă relaţia

$$O = (1 - p_0) \cdot 100\ (\%)$$

întrucât şi aici singura stare în care muncitorul nu lucrează este $S0$.

2.2 Programul de simulare:

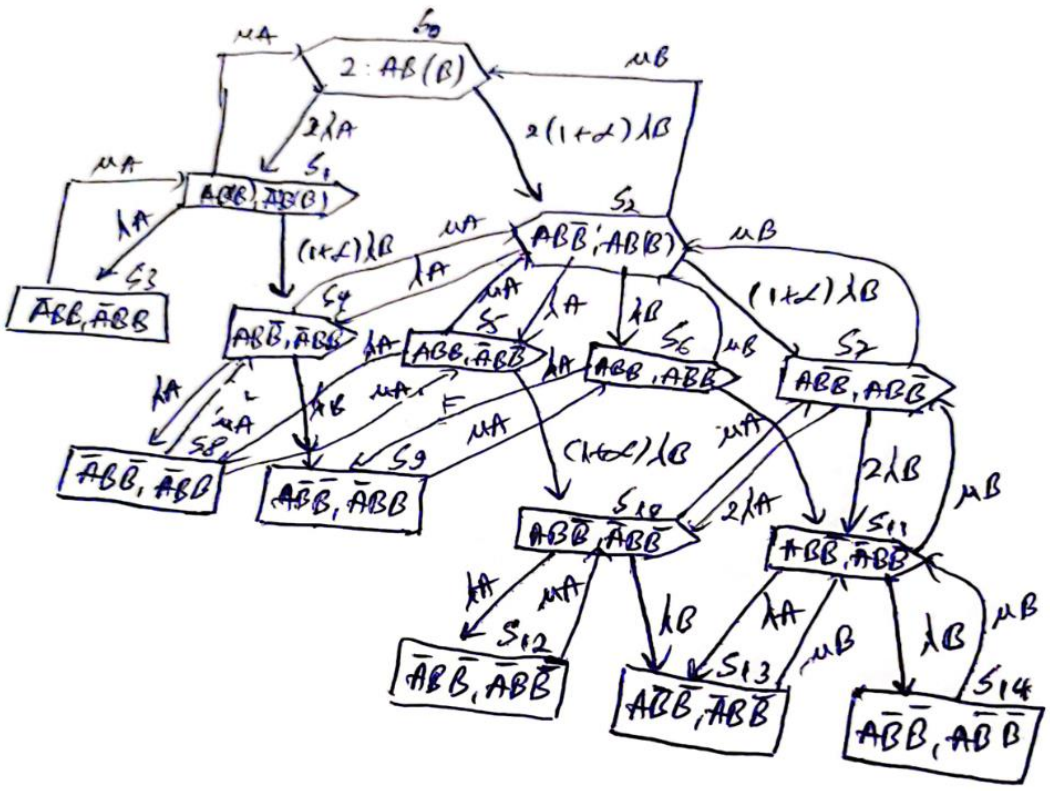
```
clc
clear all

l_a = 0.2358
l_b = 0.2453
miu_a = 3.0016
miu_b = 2.8682

A = [-2*(l_a + l_b),miu_a,miu_b,0,0,0,0;
                        2*l_a,-(l_a+l_b+miu_a),0,miu_a,0,miu_b,0;
                        2*l_b,0,-(l_a+l_b+miu_b),0,miu_a,0,miu_b;
                        0,l_a,0,-miu_a,0,0,0;
                        0,l_b,0,0,-miu_a,0,0;
                        0,0,l_a,0,0,-miu_b,0;
                        0,0,l_b,0,0,0,-miu_b]

A_tilda = A;
A_tilda(1, :) = ones(1, 7);

B_tilda = zeros(7, 1);
B_tilda(1) = 1;

P = A_tilda \ B_tilda;

if all(P >= 0) && all(P <= 1) && abs(sum(P) - 1) < eps
    disp('Vectorul rezultat este valid.');
    disp('Vectorul de probabilități de stare:');
    disp(P);
else
    disp('Vectorul rezultat NU este valid.');
end

D = (P(1) + (P(2)+P(3))/2)*100
O = (1-P(1))*100
```

2.3 Compararea rezultatelor dintre simulat și analitic pentru disponibilitate și gradul de ocupare:

- D simulat: 84.20;
- D analitic: 84.2305;
- O simulat: 27.69;
- O analitic: 27.6414.

3. Cazul $S = 2$, cu rezervă

3.1 Ca şi la modelele anterioare, pentru reducerea complexităţii modelului Markov se ţine cont de faptul că cele două sisteme sunt identice şi, de asemenea, de faptul că rezerva este identică cu modulul de bază. Prin urmare, nu se face distincţie explicită între cele două sisteme sau între modulul de bază şi rezervă. Modelul Markov pentru cazul S = 2, cu rezervă:



Matricea de adiacență a grafului obținut după modelul Markov:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	-(2*lambda_a+2*(1+alfa)*lambda_b	miu_a	miu_a	0	0	0	0	0	0	0	0	0	0	0	0
1	2*lambda_a	-(miu_a+(1+alfa)*lambda_b+lambda_a)	0	miu_a	0	0	0	0	0	0	0	0	0	0	0
2	2*(1+alfa)*lambda_b	0	-(2*lambda_a + (2+alfa)*lambda_b)	0	miu_a	miu_a	miu_b	miu_b	0	0	0	0	0	0	0
3	0	lambda_a	0	-miu_a	0	0	0	0	0	0	0	0	0	0	0
4	0	(1+alfa)*lambda_b	lambda_a	0	-(lambda_a+lambda_b+miu_a)	0	0	0	miu_a	0	0	0	0	0	0
5	0	0	lambda_a	0	0	-((1+alfa)*lambda_b+lambda_a+miu_a)	0	0	miu_a	0	0	0	0	0	0
6	0	0	lambda_b	0	0	0	-((1+alfa)*lambda_b+lambda_a+miu_b)	0	0	miu_a	0	0	0	0	0
7	0	0	(1+alfa)*lambda_b	0	0	0	0	-(2*(lambda_a+lambda_b)+miu_b)	0	0	miu_a	miu_b	0	0	0
8	0	0	0	0	lambda_a	lambda_a	0	0	-2*miu_a	0	0	0	0	0	0
9	0	0	0	0	lambda_b	0	lambda_a	0	0	-miu_a	0	0	0	0	0
10	0	0	0	0	0	(1+alfa)*lambda_b	0	2*lambda_a	0	0	-(lambda_a+lambda_b+miu_a)	0	miu_a	0	0
11	0	0	0	0	0	0	(1+alfa)*lambda_b	2*lambda_b	0	0	0	-(lambda_a+lambda_b+miu_b)	0	miu_b	miu_b
12	0	0	0	0	0	0	0	0	0	0	lambda_a	0	-miu_a	0	0
13	0	0	0	0	0	0	0	0	0	0	lambda_b	lambda_a	0	-miu_b	0
14	0	0	0	0	0	0	0	0	0	0	0	lambda_b	0	0	-miu_b

După determinarea probabilităţilor de stare, disponibilitatea sistemelor se determină cu relaţie de forma:

$$D = \left(p_0 + p_2 + p_7 + \frac{1}{2}(p_1 + p_4 + p_5 + p_6 + \cdots)\right) \cdot 100\ (\%)$$

3.2 Programul de simulare:

```
clc
clear all

l_a = 0.2358
l_b = 0.2453
miu_a = 3.0016
```

```
miu_b = 2.8682
alfa = 0



A = [-(2*lambda_a+2*(1+alfa)*lambda_b),miu_a,miu_a,0,0,0,0,0,0,0,0,0,0,0,0
;
2*lambda_a,-(miu_a+(1+alfa)*lambda_b+lambda_a),0,miu_a,0,0,0,0,0,0,0,0,0,0,0
;
2*(1+alfa)*lambda_b,0,-(2*lambda_a + (2+alfa)*lambda_b +
miu_a),0,miu_a,miu_a,miu_b,miu_b,0,0,0,0,0,0,0                ;
0,lambda_a,0,-miu_a,0,0,0,0,0,0,0,0,0,0,0
;
0,(1+alfa)*lambda_b,lambda_a,0,-(lambda_a+lambda_b+miu_a),0,0,0,miu_a,0,0,0,0,0,0
;
0,0,lambda_a,0,0,-((1+alfa)*lambda_b+lambda_a+miu_a),0,0,miu_a,0,0,0,0,0,0
;
0,0,lambda_b,0,0,0,-((1+alfa)*lambda_b+lambda_a+miu_b),0,0,miu_a,0,0,0,0,0
;
0,0,(1+alfa)*lambda_b,0,0,0,0,-(2*(lambda_a+lambda_b)+miu_b),0,0,miu_a,miu_b,0,0,0
;
0,0,0,0,lambda_a,lambda_a,0,0,-2*miu_a,0,0,0,0,0,0
;
0,0,0,0,lambda_b,0,lambda_a,0,0,-miu_a,0,0,0,0,0
;
0,0,0,0,0,(1+alfa)*lambda_b,0,2*lambda_a,0,0,-
(lambda_a+lambda_b+miu_a),0,miu_a,0,0                                    ;
0,0,0,0,0,0,(1+alfa)*lambda_b,2*lambda_b,0,0,0,-
(lambda_a+lambda_b+miu_b),0,miu_b,miu_b                                ;
0,0,0,0,0,0,0,0,0,0,lambda_a,0,-miu_a,0,0
;
0,0,0,0,0,0,0,0,0,0,lambda_b,lambda_a,0,-miu_b,0
;
0,0,0,0,0,0,0,0,0,0,0,lambda_b,0,0,-miu_b
;]

A_tilda = A;
A_tilda(1, :) = ones(1, 15);

B_tilda = zeros(15, 1);
B_tilda(1) = 1;

P = A_tilda \ B_tilda;

if all(P >= 0) && all(P <= 1) && abs(sum(P) - 1) < eps
    disp('Vectorul rezultat este valid.');
    disp('Vectorul de probabilități de stare:');
    disp(P);
else
    disp('Vectorul rezultat NU este valid.');
end

D =
(P(1)+P(3)+P(7)+(P(2)+P(4)+P(5)+P(6)+P(8)+P(9)+P(10)+P(11)+P(12)+P(13)+P(14)+P(15)
)/2)*100
```

3.3 Compararea rezultatelor dintre simulat și analitic pentru disponibilitate:

- alfa = 0, rezervă pasivă:

- D simulat: 90.71;
- D analitic: 92.0662;

-alfa = 0.5, rezervă semi-activă:
- D simulat: 87.09;
- D analitic: 91.2949;

-alfa = 1, rezervă activă:
- D simulat: 83.35;
- D analitic: 90.3601.

.

4.Concluzii

Proiectul a abordat o problemă complexă de interferență în sistemele cu evenimente discrete, concentrându-se asupra interferențelor accidentale care afectează funcționarea unui sistem. Această problemă este relevantă în contextul organizării optime a producției în întreprinderi mari și în gestionarea muncii în companiile de service. Pentru evaluarea performanțelor sistemului, s-a ales disponibilitatea ca indicator principal, exprimată ca procentul perioadelor de funcționare. Importanța maximizării disponibilității sistemului a fost subliniată în contextul economic, dar s-a recunoscut și necesitatea unui grad de ocupare corespunzător al muncitorilor de deservire pentru a evita suprasolicitarea. În concluzie, proiectul a adus în atenție o problemă importantă din domeniul cercetării operaționale, evidențiind complexitatea și interconexiunile dintre disponibilitatea sistemelor, capacitatea de deservire și fenomenul de interferență.