PROIECT

**MODELARE ȘI SIMULARE**

**BACIU CLAUDIA-IULIANA**

**GRUPA 1310A**

**PROF. COORDONATOR:**

**PETRU CAȘCAVAL**

**Proiect nr. 4**

**TEMA PROIECTULUI**:

**STUDIUL UNEI PROBLEME DE INTERFERENȚĂ A SISTEMELOR**

CUPRINS:

Prezentarea problemei studiate...................................................................................................3

Etapele proiectului......................................................................................................................4

Etapa I - Identificarea repartiţiei pentru variabilele aleatoare primare pe baza unor eşantioane de valori independente................................................................................................................5

Pașii parcurși pentru prelucrarea unui eșantion...............................................................5

Codul script-ului MATLAB cu care s-au obținut rezultatele.......................................12

Concluzie......................................................................................................................14

Etapa II - Program de simulare pentru problema de interferenţă în care un muncitor deserveşte mai multe sisteme identice.......................................................................................................15

Analiza detaliată a problemei de interferenţă...............................................................15

Algoritmul de simulare.................................................................................................16

Verificarea programului de simulare............................................................................18

Testarea programului....................................................................................................19

Codul sursă...................................................................................................................19

Temă suplimentară........................................................................................................22

Concluzie......................................................................................................................24

Etapa III - Program de simulare pentru problema de interferenţă în condiţiile în care sistemele sunt prevăzute cu modul de rezervă..........................................................................................25

Stabilirea modulului la care se adaugă rezervă.............................................................25

Algoritmul de simulare.................................................................................................25

Verificarea programului de simulare............................................................................28

Modulul A nu este echipat cu modul de rezervă...............................................28

Modulul A este echipat cu modul de rezervă....................................................29

Codul sursă...................................................................................................................30

Concluzie..........................................................................................................34

Completarea programului de simulare pentru a acoperi şi alte aspect.........................34

Codul sursă...................................................................................................................36

Concluzie..........................................................................................................41

Alegerea următorului modul pentru remediere............................................................41

Modificările efectuate în programul de simulare.........................................................42

Concluzie..........................................................................................................42

Întreruperea remedierii în curs.....................................................................................43

Concluzie..........................................................................................................43

Etapa IV - Rezolvarea analitică a problemei de interferenţă studiate pe baza modelelor Markov.....................................................................................................................................44

Prezentarea metodei de studio......................................................................................44

Cazuri studiate..............................................................................................................45

Cazul S = 1, cu rezervă la modulul A, cu întreruperea remedierii în curs........45

Codul sursă............................................................................................46

Cazul 𝑆 = 2, fără rezervă...................................................................................47

Codul sursă............................................................................................47

Cazul S = 2, cu rezervă la modulul A, cu întreruperea remedierii în curs........48

Codul sursă............................................................................................48

Concluzii.......................................................................................................................49

**Prezentarea problemei studiate**

Să considerăm un sistem a cărui funcţionare este afectată de întreruperi accidentale specifice procesului pe care acesta îl realizează. În caz de oprire este necesară intervenţia unui muncitor de deservire pentru remediere şi repunere în funcţiune. Ca indicator de performanţă se foloseşte disponibilitatea sistemului ce exprimă procentul perioadelor de funcţionare. Evident, din considerente economice, pentru sistem se impune o disponibilitate cât mai mare. Pentru muncitorul de deservire se impune un grad de ocupare corespunzător, care să nu ducă însă la suprasolicitare. Astfel, pentru un muncitor sau un grup de muncitori se alocă spre deservire, de regulă, mai multe sisteme. Să notăm cu 𝑆 numărul lor. Atunci când numărul de sisteme oprite este mai mare decât numărul muncitorilor de deservire intervine un timp de aşteptare până la începerea remedierii care se mai numeşte şi timp de interferenţă a sistemelor.

Fenomenul de interferenţă afectează disponibilitatea sistemelor şi trebuie redus pe cât posibil. Pentru reducerea timpului de interferenţă este nevoie de o capacitatea de deservire mai mare. În aceste condiţii însă gradul de ocupare a muncitorilor va scădea. Prin urmare, trebuie găsită o soluţie optimă, de compromis, între cele două tendinţe contradictorii.

Pentru simplificarea problemei de interferenţă, în cadrul acestui proiect vom considera doar cazul cu un singur muncitor care deserveşte sisteme identice.

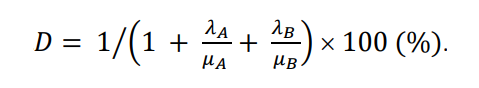
Să admitem că pentru procesul studiat sunt două cauze independente de întrerupere accidentală, notate cu 𝐴 şi 𝐵, iar modulele afectate sunt identificate prin aceleaşi simboluri. Modelul stohastic primar care descrie acest proces afectat de întreruperi accidentale cuprinde patru variabile aleatoare, şi anume:

*•* 𝑇𝑓𝐴 / 𝑇𝑓B− timpul de funcţionare până la prima întrerupere accidentală (prima defectare) care afectează modulul 𝐴 / 𝐵;

• 𝑇𝑟𝐴 / 𝑇𝑟𝐵 − timpul de remediere în caz de întrerupere a funcţionării (defectare) la modul 𝐴 / B.

Notăm cu 𝜆𝐴 şi 𝜆𝐵 ratele medii de înrerupere care afectează modulul 𝐴 şi respectiv, 𝐵. Intensităţile medii de remediere pentru cele două tipuri de întrerupere sunt notate cu 𝜇𝐴 şi respectiv, 𝜇𝐵.

Pentru acest model stohastic primar, la care nu intervine fenomenul de interferenţă, disponibilitatea sistemului şi gradul de ocupare a muncitorului de deservire (care sunt de fapt mărimi complementare) sunt uşor de determinat, chiar şi pe baza unui raţionament intuitiv. Anticipând, disponibilitatea sistemului în acest caz se determină cu relaţia:



Proiectul îşi propune ca pe baza acestui model stohastic primar, ce cuprinde 4 variabilele aleatoare (𝑇𝑓𝐴, 𝑇𝑓𝐵, 𝑇𝑟𝐴, 𝑇𝑟𝐵) pentru care trebuie să se cunoască funcţia de repartiţie, să se rezolve o problemă de predicţie în sensul estimării disponibilităţii atunci când muncitorului îi sunt alocate spre deservire mai multe sisteme. Mai precis, ne propunem să estimăm disponibilitatea care s-ar putea obţine în funcţie de numărul 𝑆 de sisteme deservite de muncitor.

În vererea creşterii disponibilităţii sistemelor, pentru unul sau pentru ambele module s-ar putea prevedea căte o rezervă care să înlocuiască în caz de defectare modulul de bază. Astfel, nu orice întrerupere accidentală ar provoca şi căderea sistemului. În aceste condiţii, problema de predicţie studiată este şi mai greu de rezolvat.

**ETAPELE PROIECTULUI**

**Etapa I.**

**Identificarea repartiţiei pentru variabilele aleatoare primare pe baza unor eşantioane de valori independente**

**Etapa II.**

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

**Etapa III.**

**Program de simulare pentru problema de interferenţă în condiţiile în care sistemele sunt prevăzute cu modul de rezervă**

**Etapa IV.**

**Rezolvarea analitică a problemei de interferenţă studiate pe baza modelelor Markov**

**Etapa I.**

**Identificarea repartiţiei pentru variabilele aleatoare primare pe baza unor eşantioane de valori independente**

**Pași de rezolvare ai etapei 1 din proiect:**

**Pasul 1**. Valorile din eşantion se ordonează crescător şi se determină media aritmetică a lor (𝑚𝑎).

**𝑇𝑓𝐴:**

0.0018 0.0033 0.0077 0.0155 0.0246 0.0276 0.0639 0.0713 0.0931 0.1003 0.1006 0.1160 0.1227 0.1495 0.1610 0.1768 0.1798 0.1952 0.1996 0.1997 0.2193 0.2224 0.2268 0.2394 0.2473 0.2479 0.2480 0.3310 0.3383 0.3703 0.3774 0.4020 0.4114 0.4146 0.4214 0.4393 0.4725 0.4988 0.5083 0.5100 0.5135 0.5512 0.5578 0.5585 0.6020 0.6205 0.6462 0.6547 0.7064 0.7317 0.7369 0.7850 0.7970 0.8175 0.8402 0.8469 0.8510 0.8957 0.9126 0.9274 0.9295 0.9313 0.9864 0.9892 1.0051 1.0074 1.0234 1.0269 1.0720 1.0864 1.0878 1.0897 1.0907 1.1062 1.1133 1.1424 1.1489 1.1501 1.1556 1.1605 1.1840 1.2088 1.2098 1.2114 1.2149 1.2383 1.2844 1.3110 1.3410 1.3801 1.3901 1.4419 1.4448 1.4555 1.4601 1.4736 1.4774 1.5340 1.5367 1.5598 1.5635 1.5959 1.5963 1.6173 1.6281 1.6345 1.6393 1.6903 1.6916 1.6998 1.7119 1.7167 1.7196 1.7200 1.7481 1.7709 1.7746 1.7798 1.7971 1.8013 1.8040 1.8267 1.8337 1.8344 1.8439 1.8564 1.8959 1.9028 1.9197 1.9333 1.9550 1.9592 1.9597 1.9681 1.9682 1.9836 1.9956 2.0049 2.0147 2.0369 2.0479 2.0798 2.0818 2.0926 2.1170 2.1385 2.1416 2.1467 2.1527 2.1856 2.2078 2.2861 2.3353 2.3514 2.3598 2.3855 2.4050 2.4338 2.4519 2.4557 2.4866 2.4921 2.5134 2.5213 2.5224 2.5307 2.5350 2.5381 2.5556 2.5778 2.5980 2.6133 2.6342 2.6563 2.6846 2.7014 2.7246 2.7504 2.7607 2.7702 2.7779 2.8310 2.8333 2.8449 2.8579 2.8759 2.8825 2.8957 2.9131 2.9150 2.9195 2.9591 2.9994 3.0015 3.0270 3.0615 3.0696 3.0748 3.0941 3.0982 3.1254 3.1580 3.1720 3.1752 3.2233 3.2444 3.2641 3.2716 3.2776 3.3294 3.3363 3.3590 3.3826 3.4126 3.4143 3.4209 3.4537 3.4601 3.4657 3.4773 3.4807 3.4937 3.4962 3.5092 3.5210 3.5482 3.5648 3.5679 3.6077 3.6896 3.6997 3.7058 3.7279 3.7798 3.7862 3.8358 3.8426 3.8852 3.9272 3.9410 3.9748 3.9983 4.0210 4.0527 4.0791 4.0919 4.1042 4.1078 4.1156 4.1251 4.1377 4.1472 4.2015 4.2594 4.2888 4.3139 4.3184 4.3345 4.3861 4.3922 4.4170 4.4359 4.5226 4.5750 4.6251 4.6656 4.6981 4.7185 4.7938 4.8026 4.8752 4.9077 4.9830 5.0024 5.0161 5.0385 5.0591 5.0819 5.1078 5.1118 5.1495 5.1759 5.3115 5.3444 5.5343 5.5832 5.5897 5.6067 5.6599 5.6753 5.6758 5.7954 5.8433 5.8947 5.9428 5.9908 6.0122 6.0230 6.0578 6.0694 6.1101 6.1878 6.3599 6.3749 6.4442 6.4543 6.4742 6.5427 6.5491 6.5620 6.6147 6.7083 6.7153 6.7766 6.7906 6.8005 6.8400 6.8532 6.8579 6.8697 6.9524 7.0503 7.0522 7.1053 7.1190 7.1722 7.1911 7.1999 7.2433 7.2471 7.2666 7.3731 7.5428 7.5951 7.6159 7.6800 7.8374 7.9191 7.9571 7.9618 7.9657 7.9662 8.0278 8.0698 8.1125 8.1371 8.2390 8.4033 8.4975 8.5049 8.5087 9.0304 9.3808 9.4129 9.4722 9.6179 9.6339 9.6665 9.8407 9.8501 10.0057 10.0323 10.1034 10.1125 10.3614 10.4334 10.5988 10.6446 10.9627 11.0146 11.1356 11.2923 11.3127 11.3163 11.3722 11.4841 11.6040 11.6995 12.3461 12.5148 12.6814 13.1470 13.3662 13.7470 14.2303 14.5987 14.6437 15.1554 15.4903 16.5188 16.8035 16.9880 17.2769 18.1431 19.4375 19.7226 19.9089 20.6400 23.2124 40.0231

Media aritmetica

ma = 4.4174

**𝑇𝑓B:**

0.0047 0.0317 0.0325 0.0368 0.0678 0.0691 0.0962 0.1761 0.1812 0.2066 0.2143 0.2195 0.2415 0.2911 0.3067 0.3380 0.3438 0.3447 0.3907 0.4037 0.4658 0.4831 0.5103 0.5123 0.5125 0.5296 0.5303 0.5451 0.5562 0.5737 0.5789 0.5942 0.5947 0.5962 0.5979 0.6250 0.6736 0.6957 0.7224 0.7247 0.7441 0.7647 0.7711 0.7766 0.7972 0.8050 0.8064 0.8111 0.8507 0.8651 0.8997 0.9075 0.9089 0.9360 0.9384 0.9723 0.9871 0.9900 1.0318 1.0459 1.0537 1.0604 1.0706 1.0767 1.0931 1.1148 1.1156 1.1625 1.2138 1.2514 1.2738 1.2871 1.2946 1.2960 1.3041 1.3137 1.3265 1.3281 1.3330 1.3593 1.3921 1.4292 1.4295 1.4552 1.4866 1.5162 1.5228 1.5333 1.5576 1.6009 1.6081 1.6239 1.6964 1.7012 1.7477 1.7831 1.7995 1.8054 1.8175 1.8306 1.8349 1.8906 1.8909 1.9123 1.9406 1.9548 1.9598 2.0224 2.0871 2.0937 2.1088 2.1314 2.1578 2.1855 2.1975 2.2200 2.2467 2.2627 2.2705 2.3165 2.3477 2.3947 2.4241 2.4300 2.4330 2.4540 2.5179 2.5208 2.5243 2.5268 2.5455 2.6071 2.6174 2.6315 2.6653 2.6730 2.6882 2.7185 2.7610 2.7707 2.7999 2.8096 2.8192 2.8453 2.8574 2.8587 2.8751 2.9282 2.9528 2.9640 2.9697 2.9751 2.9962 3.0120 3.0185 3.0329 3.0970 3.1247 3.1372 3.1474 3.1551 3.1614 3.1963 3.2491 3.2622 3.2969 3.3173 3.3651 3.3764 3.3979 3.4101 3.4474 3.4870 3.5232 3.5561 3.5585 3.5873 3.6034 3.6395 3.6737 3.7084 3.7337 3.7756 3.8226 3.8340 3.8457 3.8652 3.9351 3.9374 3.9596 3.9600 3.9847 4.0450 4.0527 4.0547 4.0754 4.1034 4.1368 4.1506 4.1610 4.2898 4.3362 4.3452 4.3699 4.3951 4.4876 4.5341 4.5438 4.5479 4.5594 4.5638 4.5976 4.6275 4.6309 4.6611 4.7411 4.7593 4.7616 4.8053 4.8374 4.9916 4.9917 5.0057 5.0082 5.0129 5.0581 5.0797 5.1138 5.1201 5.1269 5.1327 5.1475 5.2558 5.2811 5.3477 5.3886 5.3981 5.3985 5.4635 5.4902 5.5189 5.5486 5.6086 5.6176 5.6405 5.6882 5.7027 5.7033 5.7343 5.7473 5.7544 5.7987 5.8032 5.8305 5.9231 5.9354 5.9547 5.9739 5.9877 6.0167 6.0237 6.0472 6.0593 6.2353 6.2880 6.3380 6.3488 6.5086 6.5125 6.5415 6.5528 6.5636 6.9226 7.0574 7.0905 7.2513 7.2795 7.2942 7.3135 7.3401 7.3576 7.4572 7.5622 7.5768 7.6220 7.6338 7.6829 7.7204 7.8044 7.9288 7.9292 7.9585 7.9958 8.0644 8.1628 8.2279 8.3073 8.3847 8.4058 8.4353 8.5498 8.5939 8.5977 8.6124 8.6193 8.6305 8.6481 8.6954 8.9167 8.9718 9.0054 9.0174 9.0930 9.1603 9.1603 9.1691 9.3062 9.4237 9.4696 9.4867 9.7754 9.9430 10.1706 10.2344 10.2477 10.2614 10.2697 10.2908 10.3870 10.3951 10.4166 10.4832 10.7560 10.7862 10.8479 10.9597 11.2521 11.3552 11.4944 11.5260 11.5339 11.8346 11.8393 11.8742 12.0903 12.1264 12.1553 12.1770 12.4360 12.4734 12.6310 12.7953 12.9558 13.2112 13.2882 13.3633 13.3883 13.4775 13.5431 13.5548 13.7132 14.1746 14.1798 14.2682 14.3911 14.4101 14.4223 14.4939 15.3135 15.4243 15.5925 15.5936 15.6996 15.8653 16.2707 16.2777 16.3482 16.7588 17.0344 17.5250 18.0385 18.1691 18.7901 19.1817 19.4522 19.8744 19.9663 21.0899 21.5959 23.3610 24.3366 24.4709 24.8687 25.8254 26.6846 27.1954 27.3985 27.7168 37.0421 47.4302

Media aritmetica

ma = 6.0898

**𝑇𝑟𝐴:**

0 0.0007 0.0007 0.0015 0.0023 0.0023 0.0025 0.0027 0.0029 0.0038 0.0041 0.0048 0.0048 0.0056 0.0060 0.0068 0.0072 0.0076 0.0085 0.0094 0.0103 0.0113 0.0115 0.0120 0.0157 0.0159 0.0162 0.0175 0.0181 0.0182 0.0186 0.0191 0.0191 0.0207 0.0215 0.0219 0.0225 0.0236 0.0238 0.0252 0.0259 0.0262 0.0268 0.0268 0.0274 0.0278 0.0280 0.0286 0.0292 0.0296 0.0297 0.0297 0.0306 0.0309 0.0322 0.0343 0.0352 0.0354 0.0374 0.0381 0.0383 0.0390 0.0391 0.0394 0.0402 0.0402 0.0405 0.0411 0.0416 0.0421 0.0432 0.0436 0.0468 0.0489 0.0501 0.0505 0.0510 0.0511 0.0519 0.0532 0.0533 0.0538 0.0546 0.0554 0.0555 0.0587 0.0605 0.0621 0.0626 0.0628 0.0630 0.0632 0.0632 0.0639 0.0646 0.0653 0.0669 0.0672 0.0688 0.0699 0.0701 0.0724 0.0725 0.0729 0.0740 0.0761 0.0771 0.0771 0.0791 0.0808 0.0816 0.0836 0.0845 0.0860 0.0864 0.0870 0.0890 0.0896 0.0901 0.0930 0.0941 0.0948 0.0961 0.0961 0.0963 0.0974 0.0981 0.0994 0.1000 0.1030 0.1036 0.1040 0.1048 0.1048 0.1060 0.1068 0.1071 0.1074 0.1077 0.1078 0.1088 0.1094 0.1111 0.1129 0.1131 0.1135 0.1148 0.1153 0.1155 0.1161 0.1169 0.1185 0.1186 0.1190 0.1204 0.1221 0.1223 0.1235 0.1242 0.1252 0.1257 0.1263 0.1271 0.1278 0.1283 0.1290 0.1295 0.1301 0.1311 0.1313 0.1313 0.1318 0.1332 0.1337 0.1367 0.1382 0.1386 0.1402 0.1403 0.1433 0.1450 0.1453 0.1464 0.1467 0.1469 0.1487 0.1503 0.1508 0.1518 0.1527 0.1558 0.1559 0.1624 0.1645 0.1685 0.1694 0.1712 0.1716 0.1742 0.1745 0.1752 0.1767 0.1768 0.1775 0.1789 0.1797 0.1819 0.1821 0.1828 0.1828 0.1864 0.1868 0.1869 0.1874 0.1888 0.1908 0.1930 0.2001 0.2004 0.2005 0.2032 0.2041 0.2046 0.2054 0.2055 0.2084 0.2098 0.2150 0.2155 0.2161 0.2196 0.2199 0.2220 0.2229 0.2255 0.2255 0.2260 0.2286 0.2331 0.2333 0.2355 0.2363 0.2379 0.2410 0.2415 0.2420 0.2424 0.2438 0.2464 0.2502 0.2509 0.2546 0.2554 0.2614 0.2622 0.2649 0.2652 0.2654 0.2655 0.2668 0.2677 0.2683 0.2684 0.2698 0.2702 0.2704 0.2711 0.2715 0.2724 0.2786 0.2851 0.2854 0.2862 0.2895 0.2955 0.2960 0.2970 0.3013 0.3030 0.3035 0.3050 0.3051 0.3055 0.3148 0.3188 0.3195 0.3211 0.3225 0.3229 0.3233 0.3237 0.3239 0.3272 0.3275 0.3366 0.3376 0.3385 0.3408 0.3427 0.3441 0.3507 0.3541 0.3576 0.3617 0.3639 0.3643 0.3683 0.3686 0.3785 0.3791 0.3822 0.3828 0.3912 0.4016 0.4080 0.4119 0.4188 0.4254 0.4266 0.4267 0.4298 0.4375 0.4395 0.4428 0.4441 0.4441 0.4491 0.4493 0.4509 0.4553 0.4604 0.4614 0.4615 0.4635 0.4687 0.4763 0.4796 0.4805 0.4997 0.5040 0.5053 0.5195 0.5247 0.5271 0.5319 0.5326 0.5430 0.5465 0.5471 0.5502 0.5522 0.5561 0.5631 0.5690 0.5729 0.5788 0.5802 0.5912 0.5914 0.6029 0.6143 0.6308 0.6399 0.6429 0.6486 0.6597 0.6715 0.6753 0.6785 0.6795 0.6808 0.6852 0.6909 0.6948 0.6983 0.7105 0.7119 0.7212 0.7350 0.7556 0.7716 0.8324 0.8476 0.8608 0.9062 0.9093 0.9187 0.9211 0.9296 0.9849 0.9937 1.0147 1.0177 1.0728 1.1769 1.1992 1.2503 1.2746 1.4462 1.7870

Media aritmetica

ma = 0.2574

**𝑇𝑟𝐵:**

0.0002 0.0018 0.0026 0.0030 0.0034 0.0042 0.0043 0.0044 0.0075 0.0075 0.0076 0.0078 0.0078 0.0090 0.0096 0.0106 0.0116 0.0123 0.0125 0.0134 0.0135 0.0144 0.0167 0.0170 0.0173 0.0179 0.0194 0.0207 0.0213 0.0217 0.0221 0.0224 0.0224 0.0227 0.0236 0.0240 0.0242 0.0259 0.0259 0.0262 0.0272 0.0281 0.0281 0.0285 0.0290 0.0293 0.0294 0.0297 0.0305 0.0313 0.0314 0.0316 0.0318 0.0321 0.0322 0.0328 0.0329 0.0345 0.0349 0.0374 0.0378 0.0399 0.0406 0.0423 0.0423 0.0468 0.0487 0.0490 0.0491 0.0494 0.0499 0.0508 0.0511 0.0517 0.0520 0.0521 0.0523 0.0524 0.0528 0.0548 0.0554 0.0559 0.0574 0.0579 0.0581 0.0583 0.0602 0.0623 0.0642 0.0644 0.0647 0.0654 0.0657 0.0668 0.0679 0.0698 0.0698 0.0699 0.0711 0.0713 0.0738 0.0758 0.0772 0.0776 0.0785 0.0814 0.0816 0.0823 0.0833 0.0837 0.0840 0.0859 0.0862 0.0878 0.0879 0.0885 0.0903 0.0921 0.0923 0.0929 0.0931 0.0935 0.0937 0.0944 0.0950 0.0964 0.0974 0.0989 0.0990 0.1014 0.1021 0.1023 0.1048 0.1054 0.1058 0.1067 0.1068 0.1081 0.1093 0.1094 0.1108 0.1118 0.1135 0.1146 0.1159 0.1183 0.1192 0.1209 0.1211 0.1226 0.1248 0.1254 0.1262 0.1265 0.1330 0.1353 0.1365 0.1372 0.1372 0.1392 0.1393 0.1400 0.1424 0.1438 0.1458 0.1464 0.1493 0.1494 0.1504 0.1505 0.1516 0.1539 0.1545 0.1604 0.1658 0.1664 0.1676 0.1690 0.1710 0.1730 0.1743 0.1769 0.1771 0.1782 0.1805 0.1808 0.1816 0.1816 0.1820 0.1825 0.1845 0.1852 0.1876 0.1890 0.1909 0.1914 0.1917 0.1933 0.1952 0.1962 0.1965 0.1983 0.1987 0.1992 0.1993 0.2003 0.2018 0.2025 0.2039 0.2063 0.2067 0.2091 0.2103 0.2126 0.2136 0.2137 0.2143 0.2192 0.2246 0.2249 0.2300 0.2320 0.2321 0.2334 0.2348 0.2379 0.2431 0.2458 0.2458 0.2465 0.2465 0.2471 0.2489 0.2506 0.2515 0.2515 0.2520 0.2535 0.2543 0.2544 0.2581 0.2604 0.2609 0.2626 0.2663 0.2673 0.2699 0.2723 0.2746 0.2750 0.2783 0.2801 0.2813 0.2825 0.2952 0.2998 0.3014 0.3051 0.3062 0.3071 0.3077 0.3101 0.3121 0.3122 0.3130 0.3159 0.3213 0.3298 0.3315 0.3326 0.3340 0.3364 0.3401 0.3432 0.3439 0.3496 0.3525 0.3541 0.3542 0.3607 0.3616 0.3622 0.3660 0.3697 0.3756 0.3756 0.3763 0.3783 0.3788 0.3810 0.3851 0.3865 0.3905 0.3958 0.3964 0.3984 0.4006 0.4008 0.4027 0.4035 0.4036 0.4039 0.4069 0.4070 0.4078 0.4089 0.4111 0.4114 0.4221 0.4239 0.4269 0.4289 0.4334 0.4392 0.4438 0.4450 0.4459 0.4463 0.4477 0.4484 0.4501 0.4575 0.4589 0.4600 0.4731 0.4772 0.4781 0.4794 0.4813 0.4884 0.4896 0.4913 0.4915 0.4977 0.5056 0.5076 0.5153 0.5164 0.5247 0.5280 0.5316 0.5342 0.5410 0.5450 0.5508 0.5557 0.5622 0.5637 0.5641 0.5712 0.5759 0.5785 0.5855 0.5929 0.5949 0.6007 0.6042 0.6043 0.6094 0.6101 0.6166 0.6293 0.6329 0.6564 0.6576 0.6613 0.6617 0.6755 0.6888 0.7093 0.7141 0.7239 0.7439 0.7520 0.7543 0.7563 0.7626 0.7645 0.8224 0.8226 0.8647 0.8678 0.8970 0.9204 0.9604 0.9690 0.9761 1.0020 1.0180 1.0697 1.2168 1.2398 1.2591 1.2817 1.2847 1.3220 1.3404 1.4395 1.6470 2.6759

Media aritmetica

ma = 0.2841

**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).

|  |  |
| --- | --- |
| **Variabila aleatoare:** | **Valoarea lui v:** |
| **TfA** | **24** |
| **TfB** | **38** |
| **TrA** | **1.5** |
| **TrB** | **1.7** |

**Pasul 3**. Se împart cele N valori ale eșantionului în k intervale, dupa relația:

k=1+3.222\*ln(N)

Vom avea următoarele valori pentru eșantioanele date:

|  |  |  |
| --- | --- | --- |
| **Variabila aleatoare:** | **Valoarea lui k:** | **Calculat cu formula:** |
| **TfA** | **19** | k=1+3.222\*ln(N) |
| **TfB** | **18** | k=1+3.222\*ln(N)-1 |
| **TrA** | **18** | k=1+3.222\*ln(N)-1 |
| **TrB** | **17** | k=1+3.222\*ln(N)-2 |

**Pasul 4**. Afișarea unei functii empirice notată cu fe, în cele k puncte corespunzătoare vectorului N. Se va lucra cu funcția densitate de repartiție.

**Pasul 5.** Se adoptă o primă estimare a parametrului λ ( ca 1/ma sau ca 5/v) și se compară grafic funcția empirică fe cu cea teoretică f, caracterizată de parametrul λ0.

|  |  |
| --- | --- |
| **Variabila aleatoare:** | **Valoarea estimatorului λ0:** |
| **TfA** | **0.2263** |
| **TfB** | **0.1642** |
| **TrA** | **3.8850** |
| **TrB** | **3.5198** |

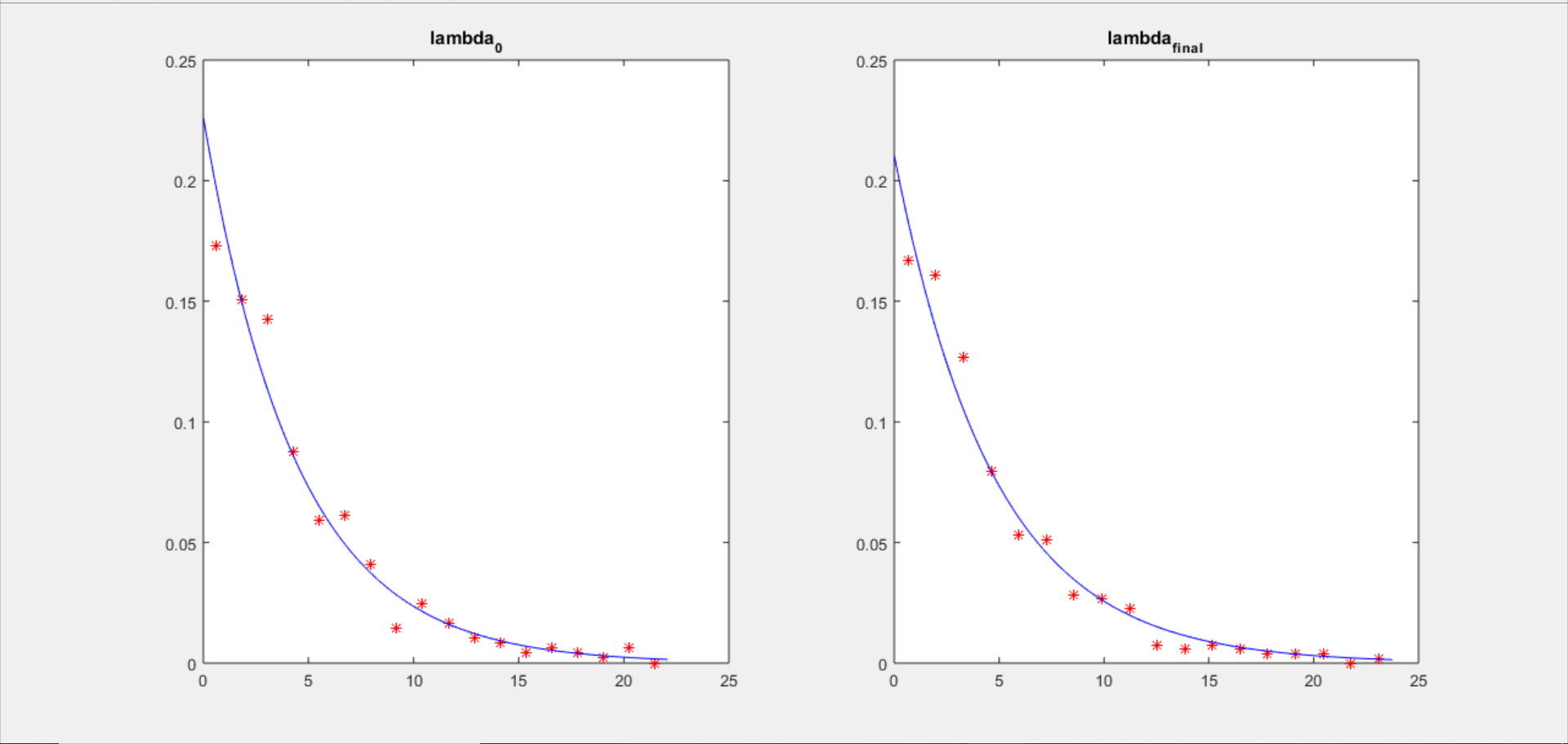
Graficele cu funcțiile empirice si funcțiile teoretice sunt afișate în paginile următoare pentru a se putea compara direct cu graficele obținute pentru valoarea λfinal.

**Pasul 6.** Se caută într-o vecinatate al estimatorului λ0, intervalul [ 0.5\*λ0 , 1.5\*λ0], valoarea care minimizează funcția criteriu:

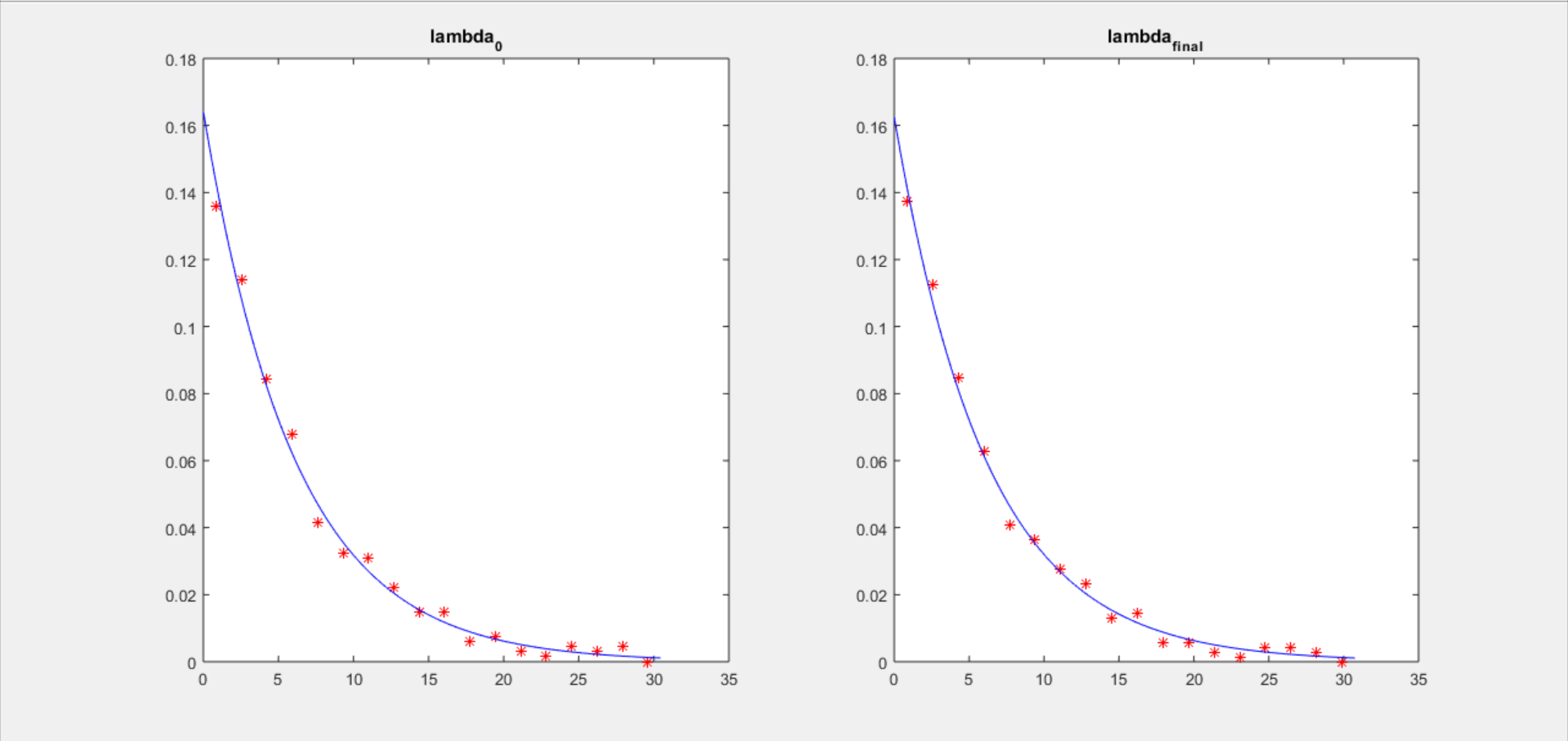
|  |  |  |
| --- | --- | --- |
| **Variabila aleatoare:** | **Valoarea parametrului λfinal:** | **Vecinătatea:** |
| **TfA** | **0.2105** | **[0.11315 , 0.33945]** |
| **TfB** | **0.1626** | **[0.0821 , 0.2463]** |
| **TrA** | **3.8533** | **[1.9425 , 5.8275 ]** |
| **TrB** | **3.4218** | **[1.7585 , 5.2785]** |

**În continuare arătam graficele funcțiilor empirice și teoretice pentru parametrii λ0 și λfinal:**

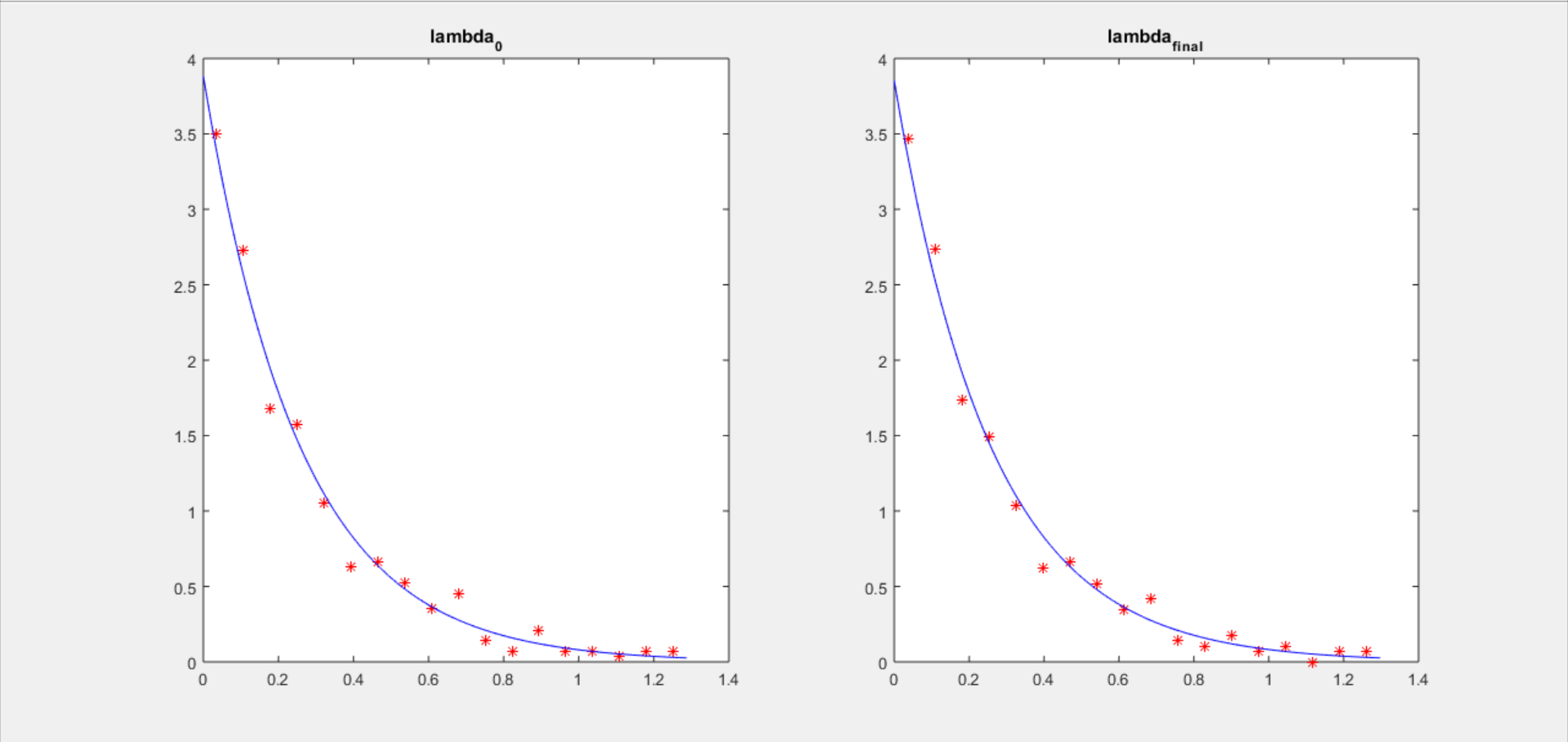
**TfA:**

****

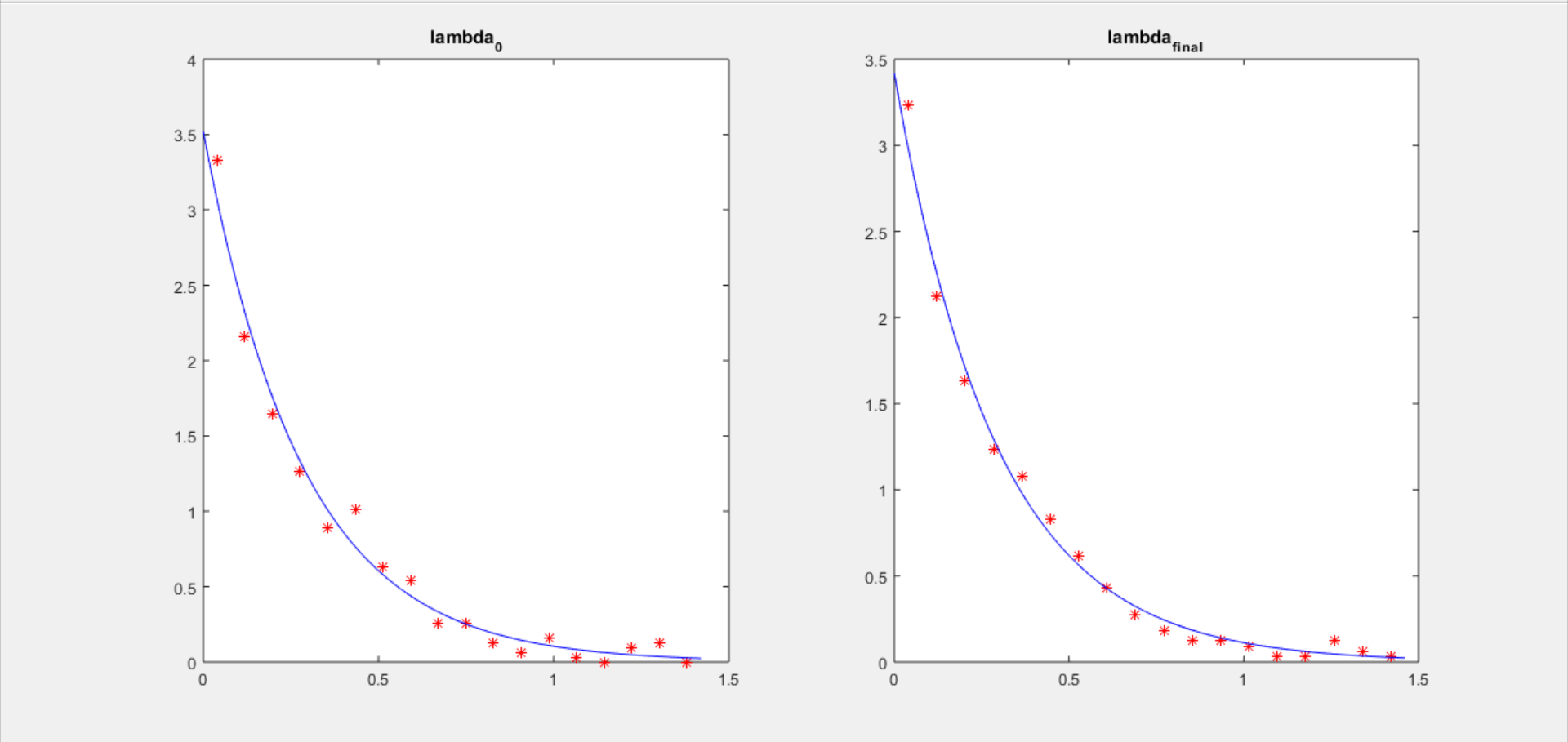
**TfB:**

****

**TrA:**

****

**TrB:**

****

**Codul sursă:**

E=…

E=sort(E);

ma=mean(E);

ma

N=400;

k=round(1+3.222\*log(N))-2;

k

lam0=1/ma;

D=5/lam0/k;

x=D:D:5/lam0;

y=x-D/2;

n=zeros(1,k);

for i=1:N

if E(i)<5/lam0

j=ceil(E(i)/D);

n(j)=n(j)+1;

end;

end;

n

sum(n)

fe=n/N/D;

d=5/lam0/1000;

z=0:d:5/lam0;

f=lam0\*exp(-lam0\*z);

subplot(1,2,1);

plot(y,fe,'\*r');

title('lambda\_0');

hold on;

plot(z,f,'-b');

spdm=30;

pc=lam0/500;

for lam=0.5\*lam0:pc:1.5\*lam0;

D=5/lam/k;

x=D:D:5/lam;

y=x-D/2;

n=zeros(1,k);

for i=1:N

if E(i)<5/lam

j=ceil(E(i)/D);

n(j)=n(j)+1;

end;

end;

fe=n/N/D;

f=lam\*exp(-lam\*y);

spd=sum((fe-f).^2);

if spd<spdm

lamF=lam;

spdm=spd;

end;

end;

lamF

0.5\*lam0

1.5\*lam0

spdm

D=5/lamF/k;

x=D:D:5/lamF;

y=x-D/2;

n=zeros(1,k);

for i=1:N

if E(i)<5/lamF

j=ceil(E(i)/D);

n(j)=n(j)+1;

end;

end;

fe=n/N/D;

d=5/lamF/1000;

z=0:d:5/lamF;

f=lamF\*exp(-lamF\*z);

subplot(1,2,2);

plot(y,fe,'\*r',z,f,'-b');

title('lambda\_f\_i\_n\_a\_l');

**Concluzii:**

S-a folosit, pentru estimarea inițiala a parametrului funcției de repartiție valoarea 1/ma. După inspecția grafică se observă necesitatea rulării unei proceduri iterative în vecinătatea estimării inițiale a parametrului cu scopul determinării mai exacte a valorii lambda. Întrucât funcția criteriu are o evoluție impredictibilă a fost nevoie de afișarea capetelor intervalului reprezentând vecinatatea. În cazul în care valoarea optimă a parametrului ar fi coincis cu unul dintre capetele intervalului, ajustarea intervalului de căutare s-ar fi făcut corespunzător în scopul determinării parametrului optim. Afișăm încă un tabel cu valoarea sumei pătratelor diferențelor minime.

|  |  |
| --- | --- |
| **Variabila aleatoare:** | **Valoarea lui SPDM:** |
| **TfA** | **0.0015** |
| **TfB** | **3.0072e-04** |
| **TrA** | **0.1791** |
| **TrB** | **0.1325** |

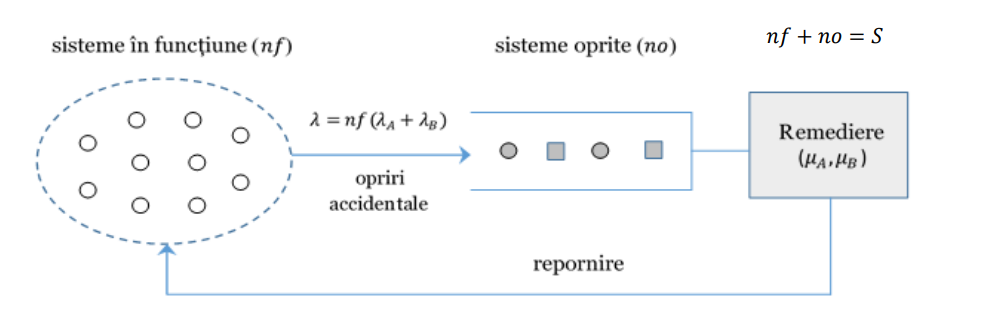
**Etapa II.**

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

**Analiza detaliată a problemei de interferenţă**

➢ 𝑀𝑜𝑑𝑒𝑙𝑎𝑟𝑒𝑎 𝑝𝑟𝑜𝑏𝑙𝑒𝑚𝑒𝑖

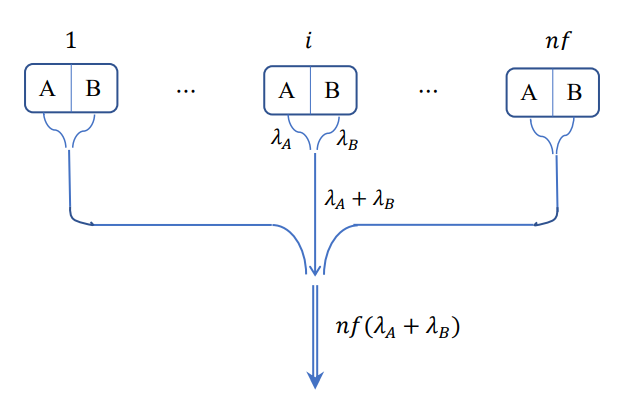
Ca model de simulare, sistemul în ansamblu compus din cele 𝑆 maşini automate şi muncitorul de deservire poate fi privit ca un sistem de servire cu o staţie, aşa cum este ilustrat în figura următoare.



Model de simulare pentru problema de interferenţă studiată.

➢ 𝐹𝑙𝑢𝑥𝑢𝑙 𝑜𝑝𝑟𝑖𝑟𝑖𝑙𝑜𝑟

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



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

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

➢ 𝐷𝑒𝑠𝑒𝑟𝑣𝑖𝑟𝑒𝑎

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

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

De observat că

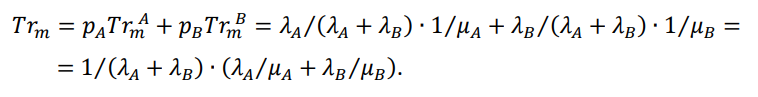
𝑝𝐴⁄𝑝𝐵 = 𝜆𝐴⁄𝜆𝐵

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

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



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



**Algoritmul de simulare**

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

• 𝑐𝑒𝑎𝑠,𝐷𝑆 – ceasul și durata simulării

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

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

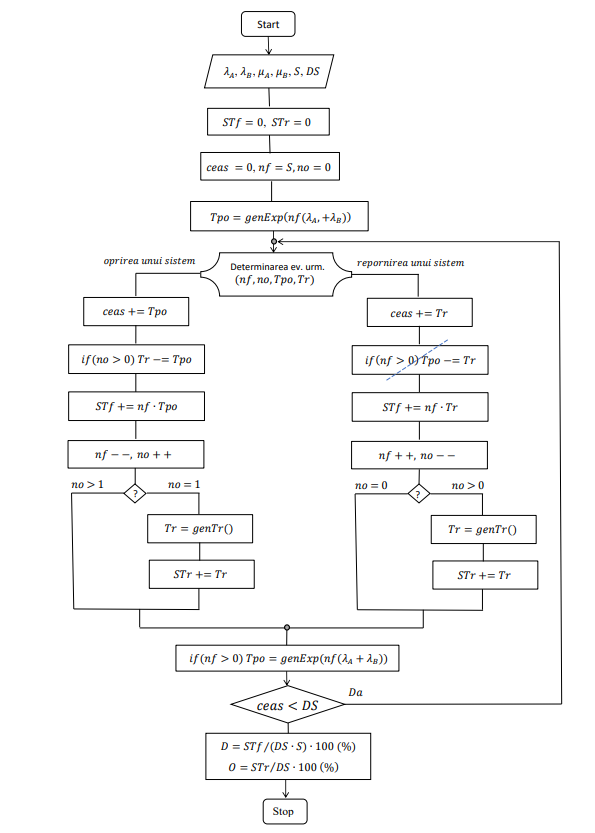
• 𝑇𝑟 – timpul necesar pentru remedierea sistemului în curs; variabila nu are semnificație atunci când 𝑛𝑜 = 0.

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

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

𝐷 = 𝑆𝑇𝑓⁄(𝐷𝑆 ∙ S) ∗ 100 (%)

𝑂 = 𝑆𝑇𝑟⁄𝐷𝑆 ∗ 100 (%)



**Verificarea programului de simulare**

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

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

𝐷 = 𝑇𝑓𝑚⁄(𝑇𝑓𝑚 + 𝑇𝑟𝑚) ∙ 100 (%)

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

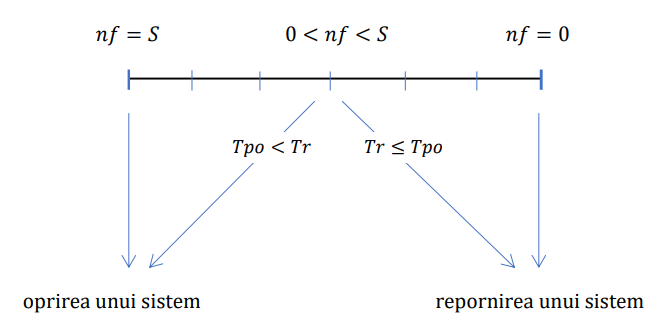
Prin urmare, rezultă relaţia:

𝐷 = 1⁄(1 + 𝜆𝐴⁄𝜇𝐴 + 𝜆𝐵⁄𝜇𝐵) ∙ 100 (%)

Precizări suplimentare

1) Cu privire la determinarea evenimentului următor

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



Determinarea evenimentului următor

2) Cu privire la durata de simulare

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

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

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



Observaţii:

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

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

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

**Testarea programului**

a) Verificări parţiale, preliminare

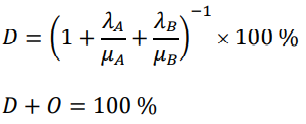
Se completează programul pentru a înregistra şi numărul de remedieri efectuate de muncitor (𝑁𝑟). La sfârşitul simulării trebuie să se verifice:

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

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

b) Verificări cantitative privind rezultatele simulării

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

****

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

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

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

**Codul sursă**

**Fișierul Header.h**

#pragma once

#include<iostream>

#include<cmath>

using namespace std;

#define lim\_max 1000000

//definirea parametrilor variabilelor aleatoare

#define lamA 0.2105

#define lamB 0.1626

#define miuA 3.8533

#define miuB 3.4218

#define pA lamA/(lamA+lamB)

#define pB lamB/(lamA+lamB)

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

//definitii functii de generare valori de selectie

double genExp(double lambda\_1);

double genGauss(double m, double sigma);

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

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

**Fișierul Source.cpp**

#include"Header.h"

using namespace std;

double genExp(double lambda\_1)

{

double u = 1;

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

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

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

return x;

}

double genGauss(double m, double sigma)

{

double s = 0;

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

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

}

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

}

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

{

double U, Tr;

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

if (U < pA)

{

Tr = genExp(miuA);

}

else

{

Tr = genExp(miuB);

}

return Tr;

}

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

{

double U, Tr;

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

if (U < pA)

{

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

}

else

{

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

}

return Tr;

}

int main()

{

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

int s;//nr sisteme

cin >> s;

double DS = 0;//durata simularii

double Tr = 0;//timp remediere

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

double ceas = 0;//ceasul simularii

int nf = s;//nr sisteme fuctionare

int no = 0;//nr sisteme oprite

int No = 0;

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

//generare valoare corespunzatoare primei opriri

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

do {

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

{

//o oprire accidentala

No++;

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

if (no > 0)

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

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

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

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

if (no == 1)

{

//genTR2 pentru repartitia normal

//genTR pentru reparitia exponential negativa

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

STr += Tr;//statistica

}

}

else {

//o remediere terminata

Nr++; //incrementam numarul de remedieri

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

if (nf > 0)

TP0 -= Tr;

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

STf += nf \* Tr;

nf++;

no--;

if (no > 0)

{

Tr = genTR();

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

}

}

if (nf > 0)

{

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

}

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

//verificarea preliminara a determinarii corecte a ev urmator

cout << "Verificari preliminare:\n";

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

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

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

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

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

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

if (s == 1)

{

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

}

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

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

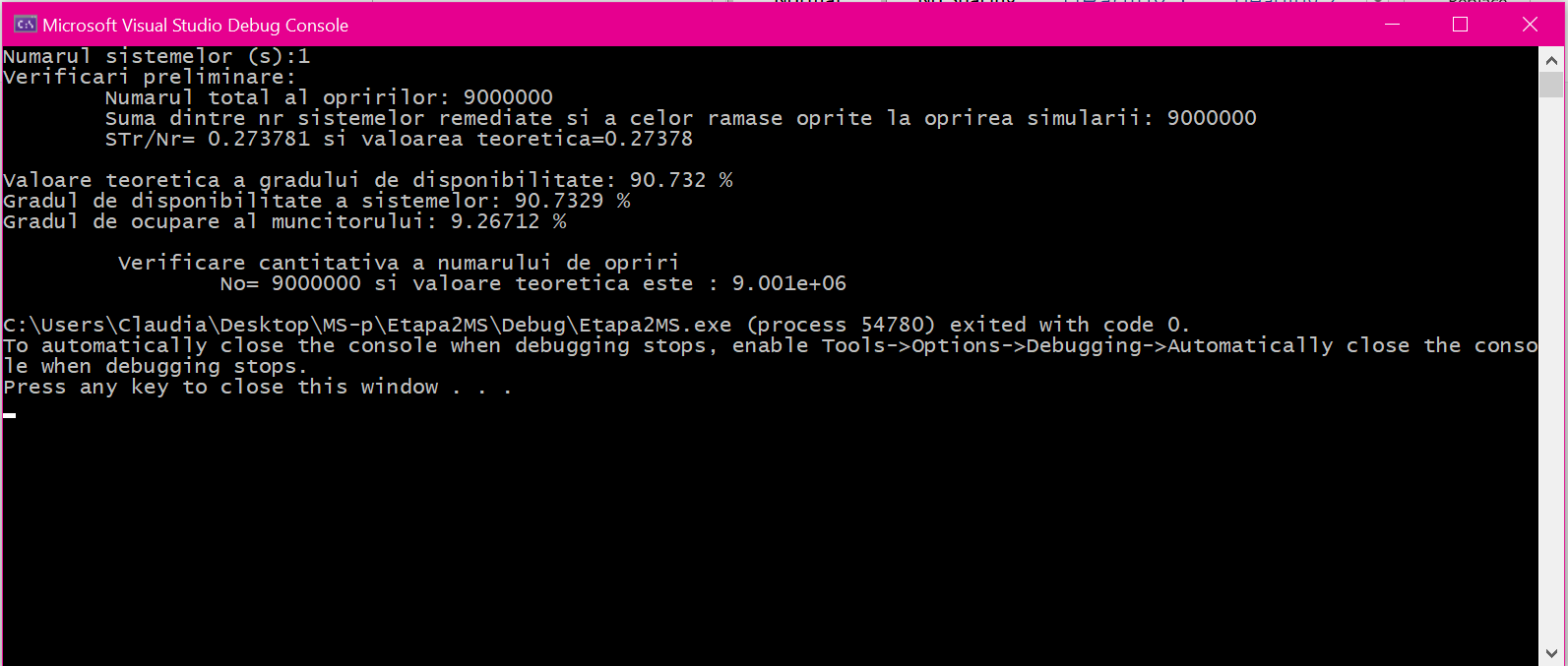
cout << "\n\t Verificare cantitativa a numarului de opriri\n";

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

}

➢ *Rezultatele simulării pentru repartiția exponențial negativă*

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



**Temă suplimentară:**

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

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

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

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

double genGauss(double m, double sigma)

{

double S = 0;

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

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

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

}

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

{

double U, Tr;

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

if (U < pA)

{

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

}

else

{

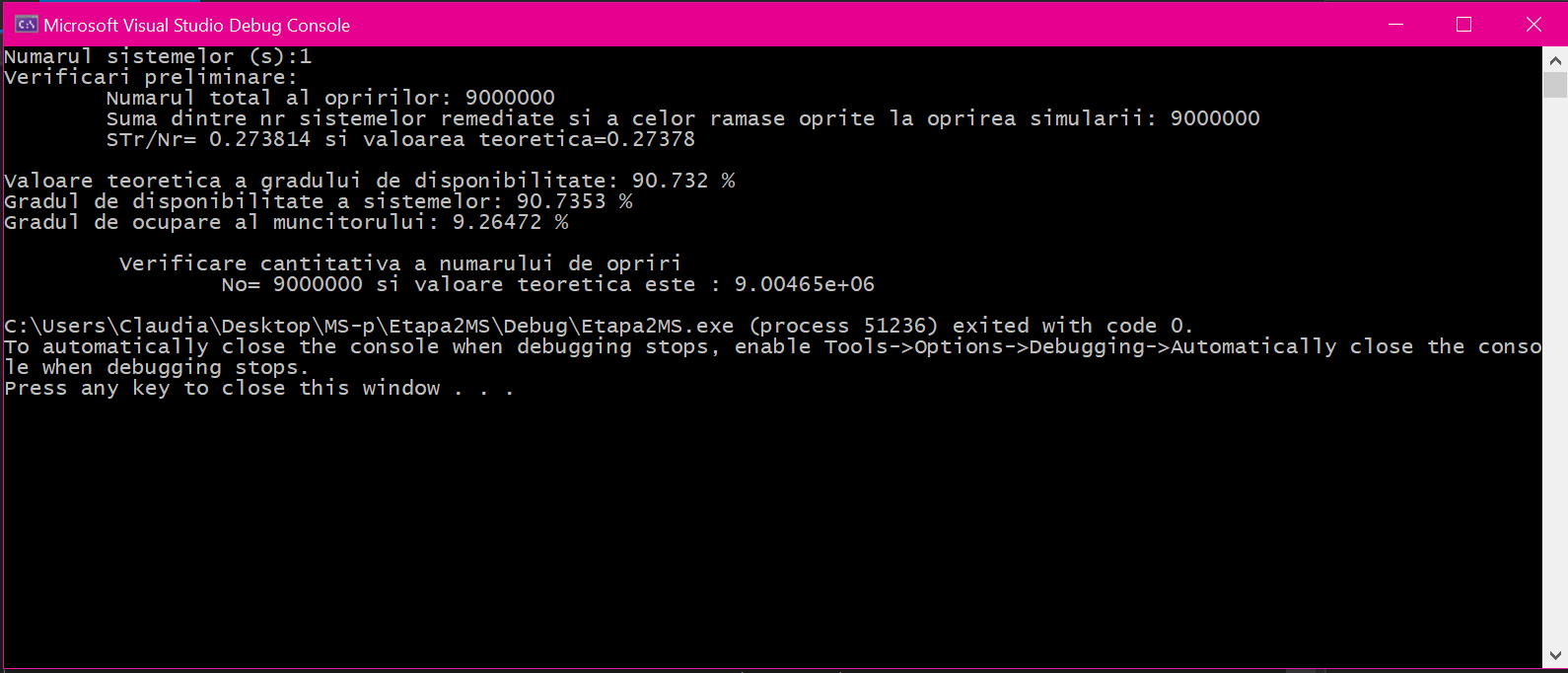
Tr = genGauss(1 / miuB, 1 / (4 \* miuB));

}

return Tr;

}

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

****

**Concluzii:**

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

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

**Etapa III.**

**Program de simulare pentru problema de interferenţă în condiţiile în care sistemele sunt prevăzute cu modul de rezervă**

**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 + 𝜆𝐴⁄𝜇𝐴 + 𝜆𝐵⁄𝜇𝐵) ∙ 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.

**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 ÷ 𝑆 – starea sistemului 𝑖: în funcţiune (𝐹) sau oprit (𝑂);

• 𝑛𝑚𝑓[𝑚][𝑖], 𝑚 = 𝐴 sau 𝐵, 𝑖 = 1 ÷ 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 ÷ 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;

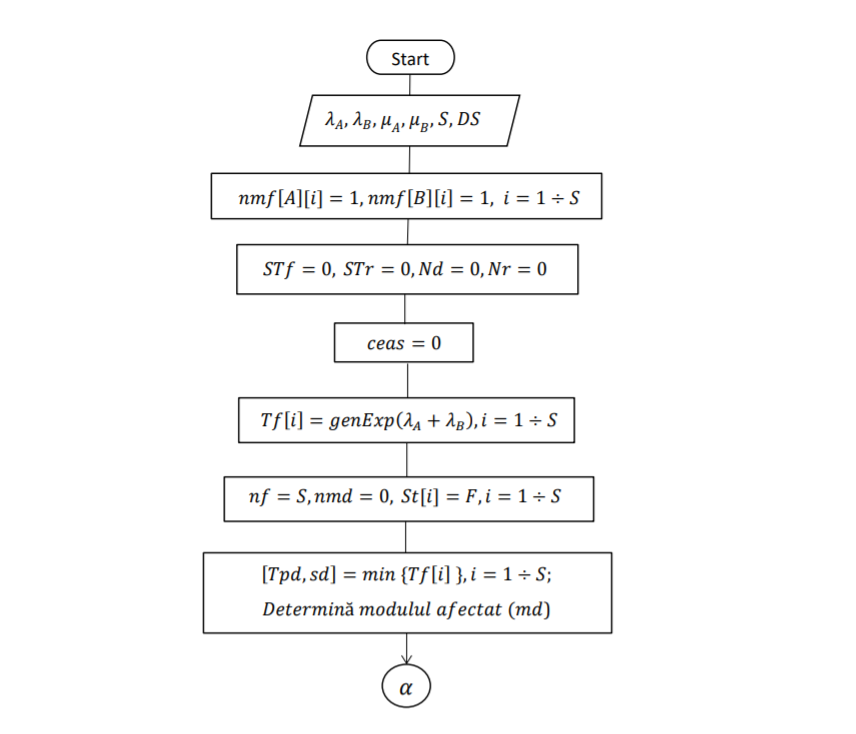
• 𝑆𝑇𝑟 − Suma timpilor de lucru pentru operaţiile de remediere efectuate de muncitor în perioada simulată;

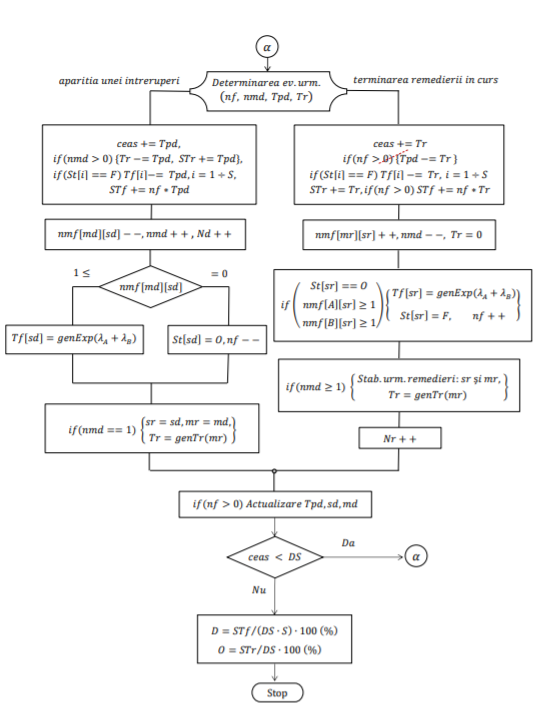
• 𝑁𝑑 − Numărul de întreruperi accidentale (defectări) produse în perioada de monitorizare;

• 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

**Verificarea programului de simulare**

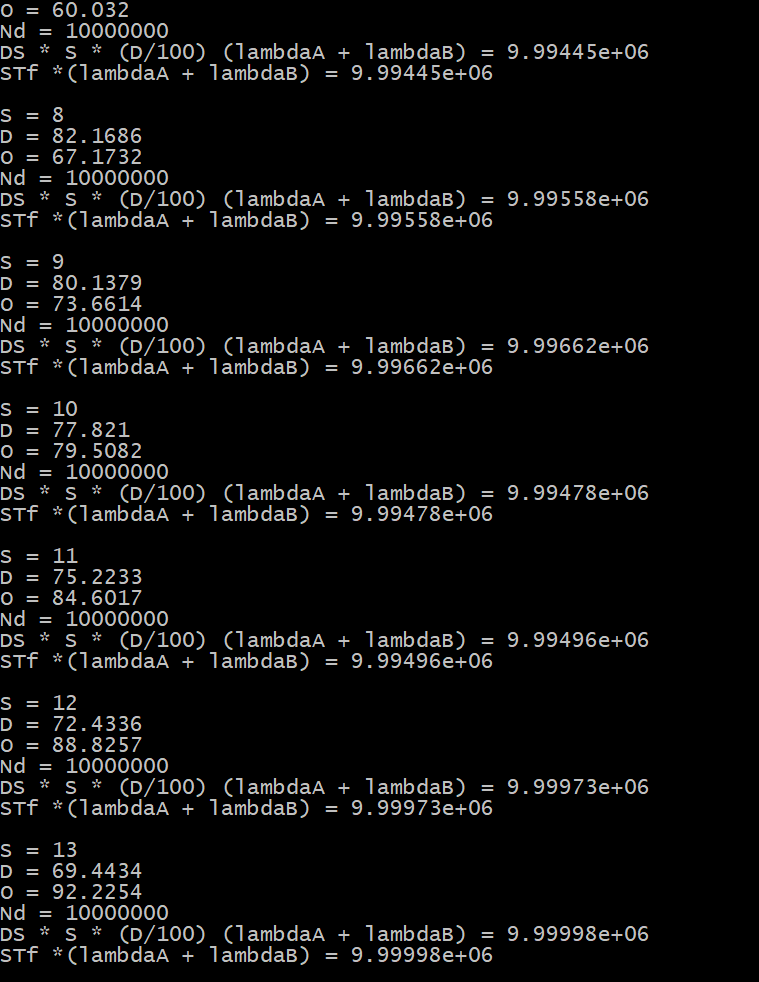
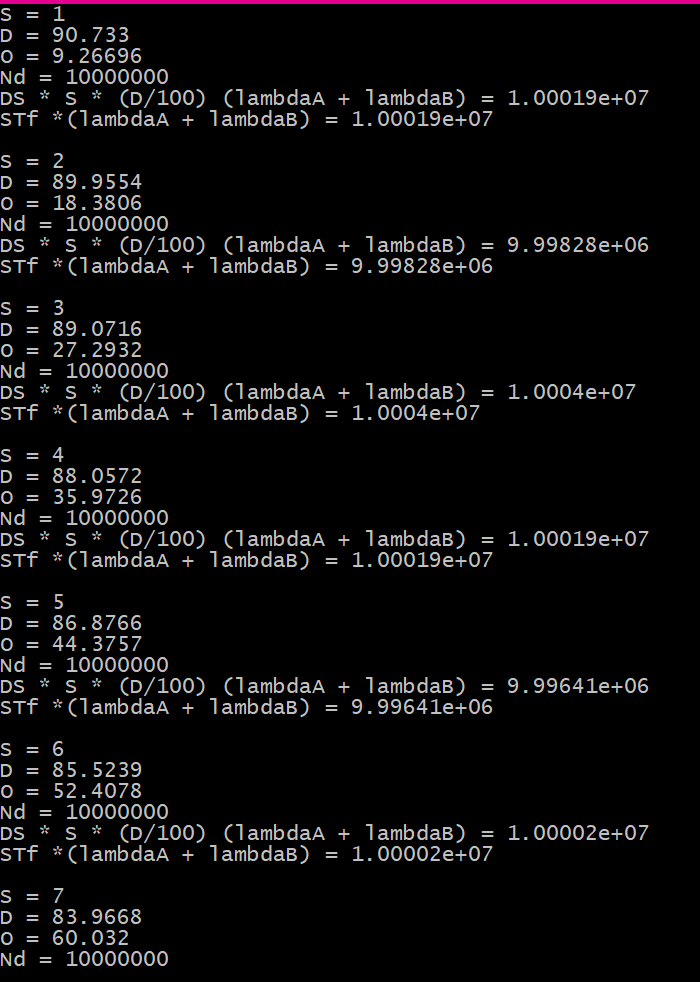
➢ Cu iniţializarea 𝑛𝑚𝑓[𝐴][𝑖] = 1, 𝑛𝑚𝑓[𝐵][𝑖] = 1 , 𝑖 = 1 ÷ 𝑆, se obţine o altă implementare pentru problema de interferenţă a sistemelor fără modul de rezervă, studiată la Etapa 2.

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 𝜆𝐴 + 𝜆𝐵 cât era şi la etapa anterioară.

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

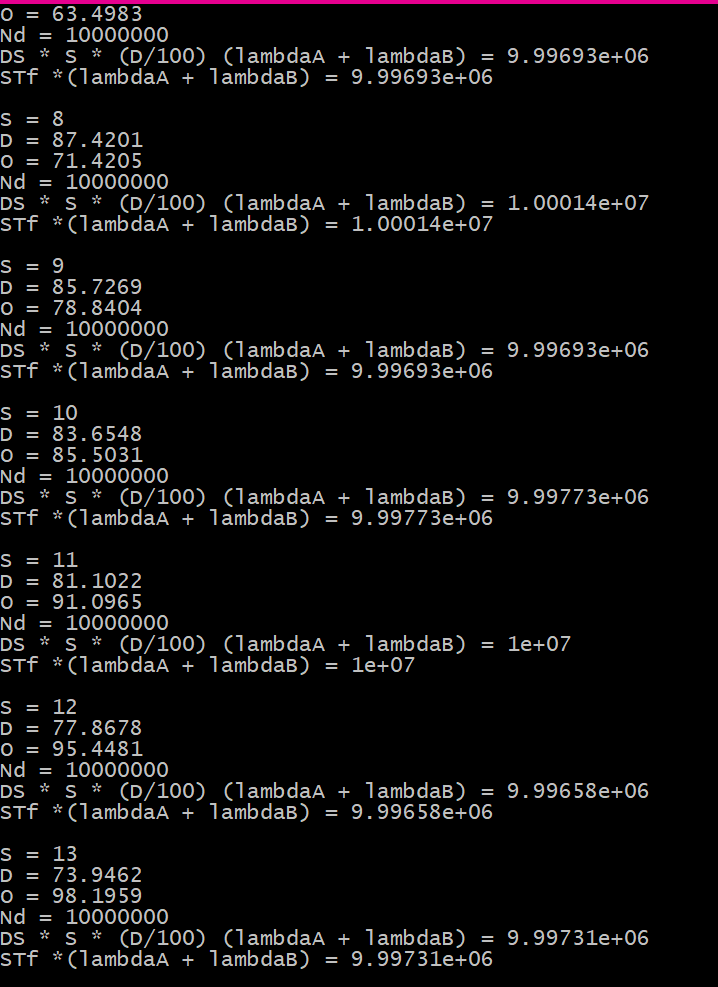
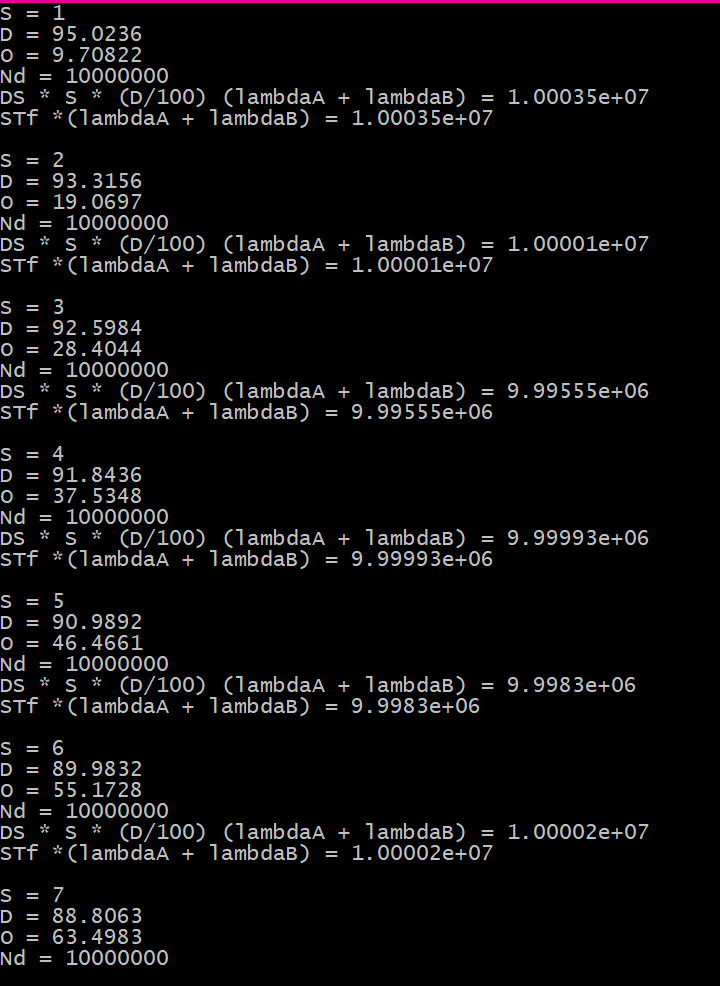
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| S | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| D(%)etapa 2 | 90.7329 | 89.9575 | 89.0684 | 88.0638 | 86.879 | 85.5273 | 83.9468 | 82.1922 | 80.1225 | 77.8312 | 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 | S | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| 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 | 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(%) | 9.26696 | 18.3806 | 27.2932 | 35.9726 | 44.3757 | 52.4078 | 60.032 | 67.1732 | 73.6614 | 79.5082 | 84.6017 | 88.8257 | 92.2254 | 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 |



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 | 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(%)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 | 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 |
| 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 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

****

**Codul 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);

}

}

**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.

**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: 𝜆𝑅=𝛼𝜆𝑀, în care 𝑀 = 𝐴 sau 𝐵, iar 𝛼 ∈ [0, 1]. (1)

• 𝛼 = 0 → rezervă pasivă (nesolicitată cât timp nu este în funcţiune);

• 𝛼 = 1 → rezervă activă (solicitată în aceeaşi măsură cu modulul de bază);

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

➢ Rezervă la modulul 𝐴

𝜆 = 𝜆𝐵 + 𝜆𝐴(1 + 𝛼(𝑛𝑚𝑓[𝐴][𝑖] − 1))

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

𝑝𝐵 = 𝜆𝐵/ (𝜆𝐵 + 𝜆𝐴(1 + 𝛼(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] ≥ 1 && 𝑛𝑚𝑓[𝐵][𝑠𝑟] ≥ 1)

{

𝑇𝑓[𝑠𝑟] = 𝑔𝑒𝑛𝐸𝑥𝑝 (𝜆𝐵 + 𝜆𝐴(1 + 𝛼(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  α = 1 | | Rezervă semiactivă  α = 0.5 | | Rezervă pasivă  α = 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 |

**Codul 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);

}

}

**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.

**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:

S1 = A B

Sistem în funcțiune

S2 = AA

S3 =

Sistem oprit

S4 =  B

Dacă 𝜇𝐴 > 𝜇𝐵 ordinea de prioritate la remediere este: 𝑆4, 𝑆2 sau 𝑆3 (se repară 𝐵̅), 𝑆1;

Dacă 𝜇𝐵 > 𝜇𝐴 ordinea de prioritate la remediere este: 𝑆2 sau 𝑆3 (se repară 𝐵̅), 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 ∥ 𝑆𝑡[𝑠𝑟] == 𝐹 && 𝑆𝑡[𝑠𝑑] == 𝑂 ∥ 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.

**Etapa IV.**

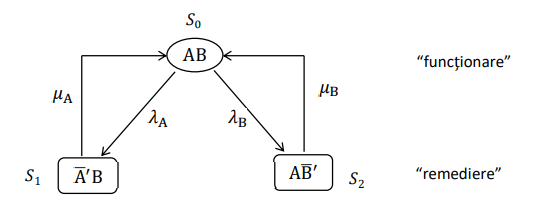
**Rezolvarea analitică a problemei de interferenţă studiate pe baza modelelor Markov**

**Prezentarea metodei de studiu**

În această etapă se va rezolva problema de interferenţă pentru câteva cazuri mai simple, pe baza unor modele Markov. Scopul acestui studiu analitic este acela de a verifica rezultatele simulării, în special pentru cazul cu modul de rezervă unde testarea programului este deficitară.

Întrucât toate cele patru variabilele aleatoare primare au o repartiţie exponenţial negativă, problema de interferenţă poate fi studiată şi analitic pe baza unor modele Markov.

Metoda analitică bazată pe lanţuri Markov este exemplificată în continuare pentru modelul primar cu 𝑆 = 1, fără rezervă, la care nu intervine fenomenul de interferenţă. Evoluţia sistemului afectat de întreruperi accidentale este ilustrată de următorul graf de tranziţie între stări. Un simbol negat reflectă o stare de defectare, iar cu apostrof se indică modulul defect în curs de remediere.



Graful stărilor pentru cazul primar cu 𝑆 = 1, fără rezervă

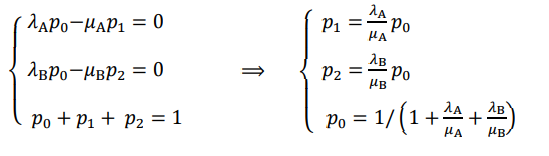
Matricea intensităţilor de tranziţie care reflectă graful stărilor este următoarea:

De precizat că o locaţie de pe diagonala principală se completează cu suma valorilor de pe coloană luată cu minus. În acest fel, pentru determinarea probabilităţilor de stare (𝑝0, 𝑝1, 𝑝2) rezultă următorul sistem de ecuaţii lineare:

𝐀 × 𝐏 = 𝟎

𝑝0 + 𝑝1 + 𝑝2 = 1

În sistemul de ecuaţii de mai sus sunt 4 ecuaţii şi 3 necunosute. Primele 3 ecuaţii nu sunt însă independente datorită modului în care s-au completat elementele de pe diagonala principală. Prin urmare, se poate renunţa la una din aceste ecuaţii (de exemplu, la prima) şi rezultă un sistem omogen, cu soluţie unică, aşa cum se prezintă în continuare.



Disponibilitatea se exprimă în funcție de stările de succes.

**Cazuri studiate**

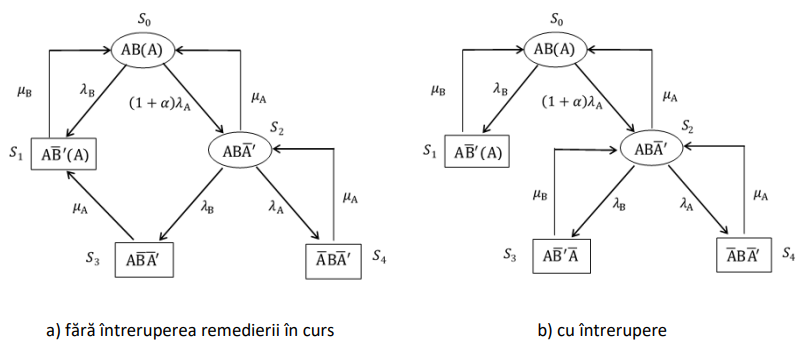
Pentru verificarea rezultatelor simulării de la etapa 3 se vor rezolva analitic următoarele cazuri: a) 𝑆 = 1, cu rezervă;

b) 𝑆 = 2, fără rezervă şi cu rezervă.

Pentru fiecare caz se calculează atât disponibilitatea cât şi gradul de ocupare a muncitorului de deservire. În continuare se prezintă modelele Markov pentru cazul cu 𝑆 = 1, în funcţie de modulul prevăzut cu rezervă. Se au în vedere cele două variante: (a) fără întreruperea remedierii în curs şi (b) cu posibilitatea de întrerupere a remedeierii în curs dacă situaţia o impune. Cum rezerva este identică cu modulul de bază nu se face distincţie explicită între cele două module. În felul acesta, rezultă un model Markov cu un număr mai mic de stări.

Deoarece toate cele 4 variabile aleatoare primare: **TfA, TfB, TrA** și **TrB** au repartiție exponențial negativă, pentru rezolvarea problemei de interferență ne putem folosi de **modelele Markov**.

**➢ Cazul S = 1, cu rezervă la modulul A, cu întreruperea remedierii în curs**



Model Markov pentru cazul 𝑆 = 1, cu rezervă la modulul A

Stările de succes în care sistemul este în funcțiune sunt *S1*și *S2* , ca urmare

disponibilitatea este dată de relația:

**D = (p1 + p2) · 100 (%)**

**O = (1 − 𝑝0 ) ∙ 100 (%)**

**Rezultate obținute**

|  |  |  |  |
| --- | --- | --- | --- |
| **S=1 cu rezervă la modulul A, cu întreruperea remedierii în curs** | **Cu rezervă activă**  **(α = 1)** | **Cu rezervă semiactivă**  **(α = 0.5)** | **Cu rezervă pasivă**  **(α = 0)** |
| **D** | 94.9758 | 95.0881 | 95.2065 |
| **O** | 14.3789 | 12.1136 | 9.7251 |

**Codul sursă:**

lamA = 0.2105;

lamB = 0.1626;

miuA = 3.8533;

miuB = 3.4218;

for alfa = 0:0.5:1

A = [1 1 1 1 1;

(1+alfa)\*lamA -(miuA+lamA+lamB) 0 miuA miuB ;

lamB 0 -miuB 0 0;

0 lamA 0 -miuA 0;

0 lamB 0 0 -miuB];

B = [1 0 0 0 0]';

P = inv(A)\*B;

sum(P)

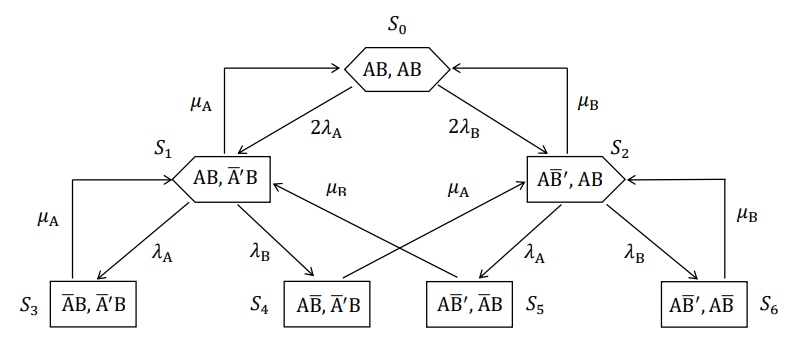
alfa

D = (P(1)+P(2))\*100

O = (1 - P(1))\*100

end

**➢ Cazul 𝑆 = 2, fără rezervă**

****

Model Markov pentru cazul 𝑆 = 2, fără rezervă

Matricea intensităților de tranziție:

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:

**Rezultate obținute**

|  |  |  |
| --- | --- | --- |
| **S=2, fără rezervă** | **D** | **O** |
| 89.9570 | 18.3777 |

**Codul sursă:**

lamA = 0.2105;

lamB = 0.1626;

miuA = 3.8533;

miuB = 3.4218;

A = [1 1 1 1 1 1 1;

2\*lamA -(lamB+lamA+miuA) 0 miuA 0 miuB 0;

2\*lamB 0 -(lamA+lamB+miuB) 0 miuA 0 miuB;

0 lamA 0 -miuA 0 0 0;

0 lamB 0 0 -miuA 0 0;

0 0 lamA 0 0 -miuB 0;

0 0 lamB 0 0 0 -miuB];

B = [1 0 0 0 0 0 0 ]';

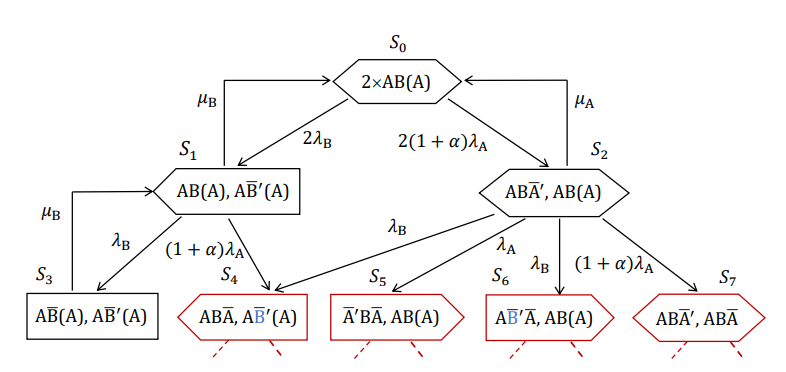
P = inv(A)\*B;

sum(P)

D = (P(1)+1/2\*(P(2)+P(3)))\*100

O = (1 - P(1))\*100

**➢ Cazul S = 2, cu rezervă la modulul A, cu întreruperea remedierii în curs**

****

Porţiune din graful stărilor pentru cazul 𝑆 = 2, cu rezervă la modulul A, cu întrerupere.

Disponibilitatea sistemelor se determină cu relaţie de forma:

**Rezultate obținute**

|  |  |  |  |
| --- | --- | --- | --- |
| **S=2 cu rezervă la modulul A, cu întreruperea remedierii în curs** | **Cu rezervă activă**  **(α = 1)** | **Cu rezervă semiactivă**  **(α = 0.5)** | **Cu rezervă pasivă (α = 0)** |
| **D** | 99.2171 | 97.9513 | 96.2068 |
| **O** | 28.4519 | 24.0785 | 19.3945 |

**Codul sursă:**

lamA = 0.2105;

lamB = 0.1626;

miuA = 3.8533;

miuB = 3.4218;

for alfa = 0:0.5:1

A = [ -2\*(1+alfa)\*lamA-2\*lamB miuA miuB 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0;

2\*(1+alfa)\*lamA -(1+alfa)\*lamA-miuA-2\*lamB 0 miuA miuA miuB miuB 0 0 0 0 0 0 0 0 0 0 0;

2\*lamB 0 -(1+alfa)\*lamA-lamB-miuB 0 0 0 0 miuB 0 0 0 0 0 0 0 0 0 0;

0 (1+alfa)\*lamA 0 -miuA-2\*lamA-2\*lamB 0 0 0 0 miuA miuB 0 0 0 0 0 0 0 0;

0 lamA 0 0 -miuA-(1+alfa)\*lamA-lamB 0 0 0 0 0 0 0 miuB 0 0 0 0 0;

0 lamB 0 0 0 -miuB-lamB-(1+alfa)\*lamA 0 0 0 0 0 0 0 miuB 0 0 0 0;

0 lamB (1+alfa)\*lamA 0 0 0 -miuB-lamA-lamB 0 0 0 miuA miuB 0 0 0 0 0 0;

0 0 lamB 0 0 0 0 -miuB 0 0 0 0 0 0 0 0 0 0;

0 0 0 2\*lamA (1+alfa)\*lamA 0 0 0 -miuA-lamA-lamB 0 0 0 0 0 miuA 0 miuB 0;

0 0 0 2\*lamB 0 (1+alfa)\*lamA 0 0 0 -miuB-lamA-lamB 0 0 0 0 0 miuA 0 miuB;

0 0 0 0 lamB 0 0 0 0 0 -miuA 0 0 0 0 0 0 0;

0 0 0 0 0 lamB 0 0 0 0 0 -miuB 0 0 0 0 0 0;

0 0 0 0 0 0 lamA 0 0 0 0 0 -miuB 0 0 0 0 0;

0 0 0 0 0 0 lamB 0 0 0 0 0 0 -miuB 0 0 0 0;

0 0 0 0 0 0 0 0 lamA 0 0 0 0 0 -miuA 0 0 0;

0 0 0 0 0 0 0 0 lamB 0 0 0 0 0 0 -miuA 0 0;

0 0 0 0 0 0 0 0 0 lamA 0 0 0 0 0 0 -miuB 0;

0 0 0 0 0 0 0 0 0 lamB 0 0 0 0 0 0 0 -miuB];

%renuntam la ecuatia 2

A(2, :) = 1;

B = [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]';

P = inv(A) \* B;

s = sum(P);

D =(P(1)+P(2)+P(7)+1/2\*(P(2)+ P(5)+ P(6)+ P(7)+ P(9)+ P(10)))\*100;

O = (1 - P(1)) \* 100;

alfa

s

D

O

end

**Concluzii:**

Folosirea sistemelor Markov pentru a determina gradul de corectitudine al programului de simulare pentru diferite rulări ale acestuia ne-a demonstrat direcția corectă în implementarea programelor, cât si eficiența acestui aparat matematic în studiul sistemelor cu evenimente discrete. O analiză analitică a disponibilității și a gradului de ocupare făcându-se mult mai simplu și necostisitor din punct de vedere computațional față de rularea algoritmului de simulare, însă dezavantajul aici este creșterea imensă a numărului de stări necesare analizei, și astfel dimensiunea matricei A, odată cu creșterea numărului de sisteme.

Prin creșterea numărului de ecuații necesare analizei analitice, aduce cu sine un cost mai mare de implementare și o putere de calcul necesară sporită.