KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

TAIKOMOSIOS MATEMATIKOS KATEDRA

Andrius Jusas

REGENERACINIO METODO TAIKYMAS APTARNAVIMO SISTEMŲ TEORIJOJE

Magistro darbas

Vadovas
Prof. dr. A. Aksomaitis

KAUNAS, 2009

KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

TAIKOMOSIOS MATEMATIKOS KATEDRA

TVIRTINU
Katedros vedėjas
doc. dr. N. Listopadskis
2009 06 06

REGENERACINIO METODO TAIKYMAS APTARNAVIMO SISTEMŲ TEORIJOJE

Matematikos magistro baigiamasis darbas

Vadovas
Prof. dr. Algimantas Aksomaitis
2009 06 02

Recenzentė
Doc. dr Rūta Jankūnienė
2009 06 04

Atliko
FMMM 7 gr. stud.
A. Jusas
2009 06 02

KAUNAS, 2009

KVALIFIKACINĖ KOMISIJA

Pirmininkas: Leonas Saulis, profesorius (VGTU)

Sekretorius: Eimutis Valakevičius, docentas (KTU)

Nariai: Algimantas Jonas Aksomaitis, profesorius (KTU)

Arūnas Barauskas, dr., vice-prezidentas projektams (UAB „Baltic Amadeus“)

Vytautas Janilionis, docentas (KTU)

Zenonas Navickas, profesorius (KTU)

Vidmantas Povilas Pekarskas, profesorius (KTU)

Rimantas Rudzkis, habil. dr., vyriausiasis analitikas (DnB NORD Bankas)

Jusas A. Regeneration method application in service systems theory: Master's work in applied mathematics / supervisor dr. assoc. prof. A. Aksomaitis; Department of Applied mathematics, Faculty of Fundamental Sciences, Kaunas University of Technology. – Kaunas, 2009. – 79 p.

SUMMARY

While modeling stochastic systems it is very important to examine results using reliable statistics analysis. Estimation methods that can allow user to make conclusions about statistical model from simulation results are necessary. These methods are also used while determining relation between simulation time (iterations count) and precision of estimations.

To complete the task regeneration method was chosen. This method is successfully used in various practical problems solving. The usage of regeneration method is based on the fact that many stochastic systems renovate in a probability sense.

Received results are confidential intervals of the N – channel system characteristics (average usage of channels, average queue size, average waiting time in the queue and maximal length of the queue). From these results we can judge about effectiveness of system work and relation between results' precision and simulation time (iterations count).

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

ĮVADAS

Dauguma užduočių, iškylančių tyrinėjant operacijas ir analizuojant sistemas, per daug sudėtingos, kad jas galima būtų išspręsti, konstruojant ir tiriant matematinius modelius. Dažnai vienintelis praktinis panašaus tipo uždavinių sprendimas yra kompiuterinis modeliavimas. Tai be abejo liečia stochastines sistemas, apimančias įvairias aptarnavimo sistemas, aptarnavimo tinklus, atsargų valdymo sistemas, techninės kontrolės, profilaktinius ir remonto darbus, o taip pat ir kitas daugiamates sistemas.

Stochastinių sistemų modeliavimą reikėtų traktuoti kaip statistinį eksperimentą. Iš pradžių darome sistemos modelį, leidžiantį tyrinėti problemos esmę sistemos struktūros atvaizdavimo keliu ir jo priežastinį – pasekminį mechanizmą. Po to su šiuo modeliu atliekami imitaciniai eksperimentai ir analizuojami gauti duomenys, kuriais remiantis daromos išvados apie sistemos elgesį. Taigi imitaciniai eksperimentai visiškai analogiški paprastiems statistiniams bandymams ir tam, kad būtų gauti reikšmingi rezultatai, jie turi remtis patikimomis statistinėmis procedūromis. Dažnai būtini tokie įvertinimo metodai, kaip pasikliautinųjų intervalų gavimas, padedantys vartotojui daryti pagrįstas statistines išvadas apie modelį, pagal modeliavimo rezultatus. Šie metodai taip pat būtini ir tam, kad būtų įmanoma numatyti svarbią sąveiką tarp modeliavimo iteracijų skaičiaus ir įverčių tikslumo lygio.

Vis dėlto dauguma svarbių statistinių modelių žymiai sudėtingesni nei imitaciniai eksperimentai, kurie analizuojami klasikiniais statistiniais metodais. Todėl statistinių procedūrų rinkinys, naudojamas modeliavimo rezultatams analizuoti, yra labai apribotas. Esant šioms aplinkybės, adekvati gaunamų rezultatų analizė dažnai yra sudėtinga arba tiesiog neįmanoma.

Problemoms spręsti vystomi statistiniai gaunamų duomenų analizės metodai priklausantys regeneracinių modelių klasei. Kaip mes matysime toliau, regeneracinių modelių klasė yra įdomi ir labai plačiai pritaikoma. Regeneracinio metodo taikymas yra sąlygotas to fakto, kad dauguma stochastinių sistemų pasižymi savybe laikas nuo laiko „kartotis pagal tikimybę“. Vartotojui tai suteikia galimybę modeliavimo eigoje stebėti nepriklausomas ir vienodai pasiskirsčiusias duomenų grupes, o kartu ir palengvina statistinę analizę. Taip pat metodas suteikia paprastus sprendimus sunkioms (taktiniu požiūriu) problemoms:

- Kaip pradėti modeliavimą?
- Nuo kokio momento pradėti rinkti duomenis?
- Kaip elgtis su stipriai koreliuotais duomenimis?

Darbo uždaviniai:

- Ištirti regeneracinio metodo taikymo galimybes daugiakanalėse aptarnavimo sistemose
- Esant šioms galimybėms įvertinti aptarnavimo sistemų charakteristikas, leidžiančias įvertinti sistemos darbo našumą.

Šiame darbe bus nagrinėjamas daugiakanalės aptarnavimo sistemos modeliavimo rezultatų apdorojimas naudojant regeneracinį metodą, apskaičiuojamas vidutinis aptarnavimo kanalų užimtumas, vidutinis eilės ilgis, vidutinis laukimo laikas eilėje – surandami šių dydžių įverčiai, apskaičiuojami pasikliautinieji jų intervalai, taip pat randama maksimalus eilės ilgis visose regeneracinėse epochose.

Šia tema skaitytas pranešimas konferencijoje „Matematika ir matematikos dėstymas 2009“ ir pateiktas straipsnis „Regeneracinis metodas eilių teorijoje” spaudai (Priedas Nr.8).

1. TEORINĖ DALIS

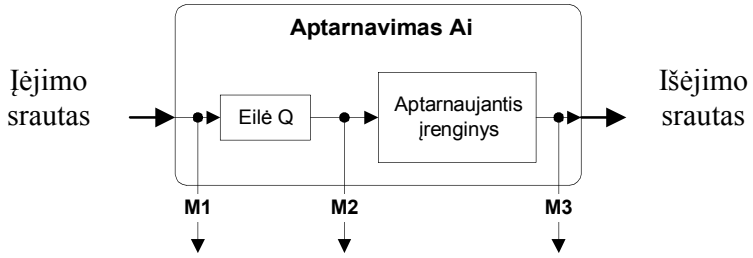
1.1 REGENERACINIAI PROCESAI IR APTARNAVIMO SISTEMŲ ANALIZĖ

1.1.1 PAGRINDINĖS APTARNAVIMO SISTEMŲ TEORIJOS SĄVOKOS

Aptarnavimo sistemų teorija nagrinėja stochastinių sistemų, kurios vadinamos eilių aptarnavimo sistemomis, funkcionavimo efektyvumą. Aptarnavimo sistemos aptarnauja į jas patenkančių paraiškų srautus. Terminą „paraiška“ suprantame kaip tam tikro užsakymo (kliento, duomenų paketo, užduoties ir t.t.) pasirodymą. Paraiškos į sistemą patenka atsitiktiniais laiko momentais, viena paskui kitą. Paraiška aptarnaujama per tam tikrą laiko tarpą, po kurio sistema tampa laisva ir gali aptarnauti kitą atėjusią paraišką. Kiekviena tokia sistema gali būti sudaryta iš kelių tarpusavyje nepriklausomų elementų, kurie dažnai vadinami aptarnavimo kanalais arba aparatais. Jų pavyzdžiais gali būti: bilietų kasos, aerouostai, informacinės sistemos ir kt.

Sistemos aptarnavimo efektyvumą, priklausantį nuo paraiškų srauto charakteristikų, sistemos kanalų skaičiaus ir kanalų našumo, galima įvertinti tam tikru eilių teorijos matematiniu modeliu. Sistemos efektyvumo kriterijumi gali būti įvairūs dydžiai ir funkcijos: paraiškos aptarnavimo tikimybė, vidutinis aptarnautų paraiškų skaičius per laiko vienetą, vidutinis laukimo laikas iki aptarnavimo pradžios, vidutinis kanalų užimtumas ir sistemos prastovų laikas, vidutinis eilės ilgis, sistemos pralaidumo koeficientas ir kt. Šių kriterijų skaitinės charakteristikos tam tikru mastu apibūdina sistemos sugebėjimą įvykdyti jai keliamus uždavinius.

Standartinė vienkanalė aptarnavimo sistemos struktūra schemiškai pavaizduota 1.1.1.1 pav.



1.1.1.1 pav. Aptarnavimo sistemos struktūra

Kiekvieną aptarnavimo sistemą sudaro šie pagrindiniai elementai: įėjimo srautas, išėjimo srautas ir aptarnavimo sistema.

Paraiškų srautas, kuris patenka į aptarnavimo sistemą, vadinamas įėjimo srautu, o išeinantis iš aptarnavimo sistemos srautas – išėjimo srautu. Visuma aptarnavimo aparatų ir taisyklės, pagal kurias aptarnaujamos paraiškos, sudaro aptarnavimo sistemą.

1.1.2 APTARNAVIMO SISTEMŲ KLASIFIKACIJA IR PAGRINDINĖS JŲ CHARAKTERISTIKOS

Klasifikuojant aptarnavimo sistemas, svarbus požymis yra sistemos veikimas, atėjus naujai paraiškai, kai visi aptarnavimo kanalai užimti. Galimi du atvejai:

- paraiškos palieka aptarnavimo sistemą ir tolimesniame procese ši paraiška nebedalyvauja. Tokios sistemos vadinamos sistemomis be eilės;
- paraiška laukia eilėje tol, kol vienas aptarnavimo kanalas tampa laisvas. Kai tik tai įvyksta, aptarnaujama paraiška, esanti eilėje. Tokios sistemos vadinamos sistemomis su eile.

Sistemose su eile paraiškos, kurios yra eilėje, gali būti aptarnaujamos skirtingai. Jos gali būti aptarnaujamos:

- tokia tvarka, kokia jos patenka į sistemą (dažniausiai sutinkama First-In-First-Out tvarka, t.y. pirma atėjusi paraiška bus pirma aptarnauta);
- atsitiktinai imant paraiškas iš eilės;
- pagal paraiškų prioritetus (pradžioje imamos paraiškos su aukštesniu prioritetu, paraiškos kurias galima aptarnauti greičiau, paraiškos kurios negali laukti ilgiau, negu būtina ir t.t.).

Sistemos su eile skirstomos į sistemas:

- su apribojimais,
- be apribojimų.

Apribojimai gali būti eilei, paraiškų laukimo laikui eilėje ir paraiškos buvimo laikui sistemoje. Sistemose su neribota eile kiekviena paraiška, jei nėra laisvų aptarnavimo kanalų, stoja į eilę ir „kantriai“ laukia, kol atsilaisvins kanalas; kiekviena paraiška anksčiau ar vėliau bus aptarnauta.

Sistemose su ribotu laukimo laiku eilėje paraiškos laukimo laikas iki aptarnavimo yra ribojamas. Jei laukimo laikas viršijamas, neaptarnauta paraiška palieka sistemą.

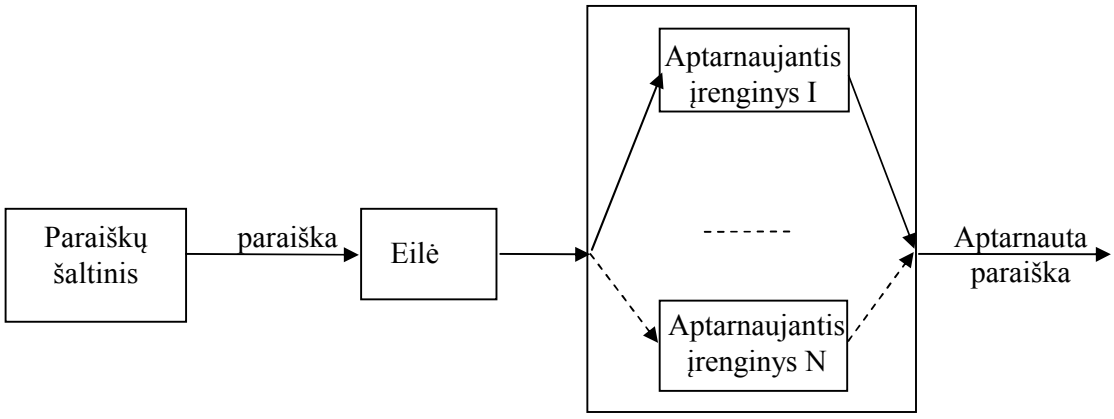
Mišraus tipo aptarnavimo sistemos sujungia sistemų su eile ir be eilės savybes. Tokios sistemos pavyzdys yra sistema su apribojamu eilės dydžiu. Jei atėjusi paraiška randa visus kanalus užimtus, tai ji stoja į eilę tik tuo atveju, jei eilė nedidelė, (kai paraiškų eilės ilgis neviršija tam tikro skaičiaus). Jei eilėje laukia daug paraiškų, tai atėjusi paraiška palieka sistemą.

1.1.3 N-KANALĖ APTARNAVIMO SISTEMA

Šiame darbe, nagrinėjant regeneracinio metodo taikymą aptarnavimo sistemų teorijoje, modeliuojama n-kanalė aptarnavimo sistema su eile be apribojimų. Pasirinkta aptarnavimo sistema pavaizduota 1.1.3.1 pav. ir pasižymi tokiomis savybėmis:

- modeliuojama sistema su eile;
- sistemos eilė be apribojimų;
- eilėje esančios paraiškos aptarnaujamos tokiu principu: pirma atėjo – pirma aptarnaujama.

Paraiškos sistemoje pasirodo po vieną. Jeigu atėjusi paraiška randa bent vieną laisvą aptarnavimo įrenginį, tai tuojau pat pradedamas jos aptarnavimas. Tačiau jeigu visi įrenginiai randami užimti, tai paraiška stoja į eilę ir laukia.



1.1.3.1 pav. N-kanalė aptarnavimo sistema

Sistemos efektyvumo kriterijai – vidutinio kanalų užimtumo, vidutinio paraiškos laukimo laiko eilėje iki aptarnavimo pradžios ir vidutinio bei maksimalaus eilės ilgio pasikliautinieji

intervalai. Jie buvo skaičiuojami, esant skirtingiems paraiškų atėjimo ir apdorojimo pasiskirstymams. Paraiškų atėjimo ir aptarnavimo laiko momentai pasiskirstę pagal eksponentinį, tolygųjį arba Veibulo dėsnį. Taip pat yra galimybė paraiškoms patekti į sistemą determinuotais laiko momentais, tačiau tai labiau buvo naudojama programos kūrimo ir testavimo metu.

Išnagrinėsime pavyzdį, paaiškinantį regeneracinio metodo taikymą modeliuojamos sistemos efektyvumo kriterijų apskaičiavimui.

Tarkime, vienkanalėje sistemoje šaltinis generuoja naujas paraiškas kas 60 sekundžių. Be to, skirtingų paraiškų laukimo eilėje laiko trukmės yra nepriklausomi atsitiktiniai dydžiai, vienodai pasiskirstę pagal tolygųjį skirstinį intervale nuo 10 iki 90 sekundžių. Tarkime, kad sistemos funkcionavimo kokybės kriterijus E{W} – vidutinis paraiškos laukimo eilėje laikas (neįskaitant pačio aptarnavimo laiko). Kadangi skaičiavimo požiūriu paprasti analitiniai rezultatai tiksliai E{W} reikšmei nustatyti duotoje sistemoje neegzistuoja, akivaizdžiu tampa modeliavimo reikalingumas.

Taigi užduotis susideda iš sistemos modeliavimo ir gautų rezultatų analizės, siekiant įvertinti dydį E{W}. Taip pat reikia gauti tam tikrą įverčio E{W} „patikimumo“ matą, pavyzdžiui 90% - tinį pasikliautinąjį intervalą.

Tegu W_1 reiškia pirmosios paraiškos laukimo eilėje laiką, W_2 – antrosios paraiškos laukimo eilėje laiką ir t.t. Tada, jei bendras modeliuojant aptarnautų paraiškų skaičius N, empirinis vidurkis

$$\frac{W_1 + W_2 + \cdots + W_N}{N} \qquad \textbf{(1.1.3.1)}$$

yra suderintasis E{W} įvertis, kadangi akivaizdu, kad empirinis vidurkis su tikimybe 1 artėja prie tikrosios E{W} reikšmės, kai $N \to \infty$. Tačiau empirinis vidurkis (1.1.3.1) priklausys nuo pradinių sąlygų, kitaip tariant bus tikrosios reikšmės paslinktasis įvertis. Pavyzdžiui, jei kaip paprastai, $W_1 = 0$, tai keletas pirmųjų laukimo eilėje laiko reikšmių turi tendenciją būti mažos. Poslinkis išnyks, jei pradinė W_1 reikšmę pasirinksime atsitiktiniu būdu iš pačio W pasiskirstymo. Nelaimei, neaiškus net dydžio W vidurkis, jau nekalbant apie jo skirstinį. Taigi toks „sprendimas“ nėra visiškai priimtinas.

Problemą galime išspręsti tęsdami modeliavimą, nerenkant duomenų tol, kol įsitikinsime, jog modeliuojama sistema pasiekė stacionarų rėžimą, ir tada pradedant tuo momentu, rinksime duomenis. Pavyzdžiui, galima būtų sumodeliuoti sistemą 2000 paraiškų aptarnavimui, atmesti pirmųjų 1000 laukimo eilėje laikų ir naudoti vidurkį:

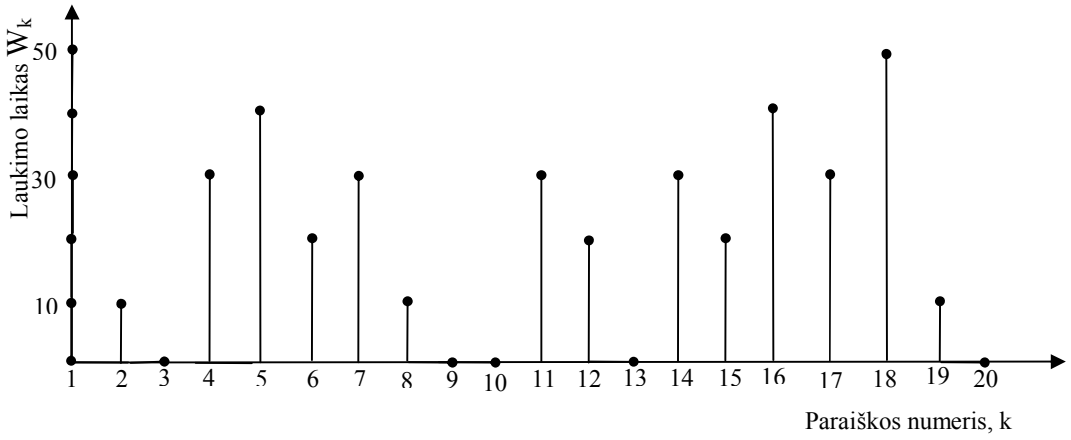
$$\frac{W_{1001} + W_{1002} + \cdots + W_{2000}}{1000}$$

kaip E{W} įvertį. Bet anaiptol neaišku, kokio būtent ilgumo turi būti „stabilizacijos“ periodas. Todėl modeliuojant tokiu būdu, nenašiai sunaudojama daug „mašininio“ laiko, be to neužtikrinamas rezultatų adekvatumas.

Egzistuoja ir kita problema, kylanti naudojant empirinio vidurkio (1.1.3.1) įvertinimo procedūrą. Pagal modeliavimo rezultatus norėtųsi gauti tam tikrą dydžio E{W} įverčio patikimumo matą, nusakantį tikimybę, kad pakartojus modeliavimą gauti įverčiai būtų analogiški. Pavyzdžiui, tarkime, kad norime sudaryti 90% - tinį pasikliautinąjį intervalą *I* tikrajai E{W}reikšmei tokį, jog esant bet kokiam nepriklausomam imitacinio eksperimento pakartojimui tikimybė, kad tikroji E{W} reikšmė priklauso intervalui *I*, būtų 0,90. Tačiau tokiems intervalams konstruoti, naudojant klasikinę statistiką, eksperimento duomenys turi būti nepriklausomi ir vienodai pasiskirstę, pagal tam tikrą tikimybinį dėsnį. Aptarnavimo sistemos modeliavimo rezultatai nusako laukimo eilėje laiko ilgių seką W_1, W_2,..., W_N. Akivaizdu, kad jei W_k – didelis, tai kita (k+1)-moji paraiška, irgi lauks ilgai, tai nepriklauso nuo to, ar modeliavimas prasideda rinkimu iš stacionaraus W pasiskirstymo ar ne. O kadangi laukimų W_1, W_2,..., W_N trukmės nėra nepriklausomi dydžiai, klasikinė statistika mažai naudinga įverčio E{W} patikimumo nustatymui.

Taigi poslinkiai, nulemti pradinių sąlygų, ir stipriai koreliuoti pradiniai duomenys sudaro rimtas kliūtis pasiūlytos įvertinimo procedūros, kuri remiasi empiriniu vidurkiu (1.1.3.1), naudojimui. Tačiau egzistuoja paprastas šių kliūčių pašalinimo metodas.

Išnagrinėsime tokį pavyzdį. Tarkime, kad modeliavimas pradedamas, nustačius $W_1 = 0$, jog galima būtų imituoti tam tikrą neilgą laiko tarpą, ir, kad paraiškų laukimo eilėje laiko stebėjimas atitinka pavaizduotą 1.1.3.2 pav.



1.1.3.2 pav. Paraiškų laukimo eilėje laikai

Matome, jog paraiškos 1, 3, 9, 10, 13 ir 20 – atėjimo momentu randa aptarnavimo įrenginį laisvą, ir joms nereikia laukti eilėje, kai tuo tarpu paraiškos 2, 4, 5, 6, 7, 8, 11, 12, 14, 15, 16, 17, 18, 19 turi laukti , kol jas aptarnaus. Be to, nuo 1-sios paraiškos atėjimo iki 2-osios išėjimo aptarnaujantis įrenginys nuolat užimtas. Po to, kol nepasirodo 3-oji paraiška, jis nuolat laisvas. Ir vėl užimtas, kol vyksta 3, 4, 5, 6, 7, 8 paraiškų aptarnavimas, Jis laisvas nuo to momento, kol

neaptarnaujama pvz. 1000 paraiškų. Ir toliau modeliuojamoji aptarnavimo sistema veiks tokiu pat principu – aptarnaujantis įrenginys užimtas, po to laisvas, vėl užimtas, vėl laisvas ir t.t.

Ciklu vadinsime laiko tarpą, sudarytą iš užimtumo periodo ir po jo einančio prastovos periodo. Tada gauta trajektorija susideda iš penkių pilnų ciklų, kuriems vykstant sistema aptarnauja tokius ateinančius paraiškų rinkinius:

{1, 2}, {3, 4, 5, 6, 7, 8}, {9}, {10, 11, 12}, {13, 14, 15, 16, 17, 18, 19}

Šeštasis ciklas prasideda 20 -tosios paraiškos atėjimu. Tokiu būdu, kiekvieno naujo ciklo pradžią užduoda paraiška, randanti aptarnavimo įrenginį laisvą.

Pastebėkime, kad sistema „atsistato“ šis terminas apibrėžia sistemos grįžimą į tam tikras fiksuotas būsenas kai atėjo 1-oji paraiška, t.y. ciklo pradžios momentu tolimesnis modeliuojamos sistemos elgesys nepriklauso nuo jos elgesio praeityje ir pasiskirsto pagal tą patį tikimybinį dėsnį, kaip ir tuo momentu, kada pirmoji paraiška patenka į laisvą aptarnaujantį įrenginį, t.y. kiekvieno ciklo pradžia neatsiejama nuo momento, kada pati pirmoji paraiška ateina į sistemą. Taigi, natūralu grupuoti modeliavimo rezultatus: pirmoji grupė sudaryta iš laukimo laiko reikšmių pirmajame cikle, antroji – iš laukimo laiko eilėje reikšmių antrajame cikle ir t.t. Tos grupės tokios:

$\{W_1, W_2\}$,

$\{W_3, W_4, W_5, W_6, W_7, W_8\}$,

$\{W_9\}$,

$\{W_{10}, W_{11}, W_{12}\}$,

$\{W_{13}, W_{14}, W_{15}, W_{16}, W_{17}, W_{18}, W_{19}\}$.

Kadangi kiekvienas ciklas prasideda esant toms pačioms sąlygoms – šiais momentais sistema regeneruoja, o ciklų duomenų grupės statistiškai nepriklausomos ir turi vienodus skirstinius. Taigi, jeigu sakysime, kad Y_k reiškia laukimo laikų sumą k-tajame cikle, o α_k – k-tajame cikle aptarnautų paraiškų skaičius, tai poros

$$(Y_1, \alpha_1), (Y_2, \alpha_2), (Y_3, \alpha_3), (Y_4, \alpha_4) \text{ ir } (Y_5, \alpha_5)$$

yra nepriklausomos ir vienodai pasiskirstę. Tačiau Y_k ir α_k priklausomi.

Iš 1.1.3.2 pav. gauname

$$(Y_1, \alpha_1) = (10, 2),$$
$$(Y_2, \alpha_2) = (130, 6),$$
$$(Y_3, \alpha_3) = (0, 1),$$
$$(Y_4, \alpha_4) = (50, 3),$$
$$(Y_5, \alpha_5) = (180, 7).$$

Tokiu būdu stipriai koreliuoti duomenys { W_1, W_2,..., W_{19} } buvo suskirstyti į grupes, kurios yra nepriklausomos ir vienodai pasiskirsčiusios.

Visa tai įtakoja sudėtingų problemų sprendimą, su kuriomis susiduriama, analizuojant modeliavimo rezultatus, t.y. gaunant pagrįstą dydžio E{W} įvertį. Tarkime, atnaujiname modeliavimą nuo 20-osios paraiškos ir tęsiame, kol negausime *n* pilnų ciklų, N – bendras per *n* ciklų aptarnautų paraiškų skaičius, tada

$$\frac{W_1 + W_2 + \cdots + W_N}{N} = \frac{W_1 + W_2 + \cdots + W_n}{a_1 + \alpha_2 + \cdots + \alpha_n} \qquad \textbf{(1.1.3.2)}$$

Dešinę (2) pusę galima užrašyti ir tokiu pavidalu

$$\frac{(Y_1 + Y_2 + \cdots + Y_n)/n}{(\alpha_1 + \alpha_2 + \cdots + \alpha_n)/n}. \qquad \textbf{(1.1.3.3)}$$

Kadangi kiekvienas rinkinys $\{Y_1, Y_2, \ldots, Y_n\}$ ir $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$susideda iš vienodai pasiskirsčiusių atsitiktinių dydžių, tai pagal didžiųjų skaičių dėsnį, kai $n \to \infty$, (1.1.3.3) skaitiklis artėja į E{Y_1}, o vardiklis – į E{α_1} su tikimybe, lygia 1. Pastebėkime, kad $N \geq n$ ir $N \to \infty$, kai $n \to \infty$. Be to, kairioji (1.1.3.2) pusė su tikimybe 1 artėja į E{W}, kai $N \to \infty$.

Taigi

$$E\{W\} = \frac{E\{Y\}}{E\{\alpha\}}$$

Tokiu būdu, E{W} įvertinimo užduotis tampa ekvivalenti santykio $\frac{E\{Y\}}{E\{\alpha\}}$ įvertinimui. Ir kadangi šis santykis gali būti įvertintas pagal nepriklausomas ir vienodai pasiskirsčiusias poras $(Y_1, \alpha_1), \ldots, (Y_n, \alpha_n)$, todėl galima pasinaudoti klasikine statistika ir apskaičiuoti santykio $\frac{E\{Y\}}{E\{\alpha\}}$ pasikliautinąjį intervalą.

Taip gavome paprastą sprendimą, kaip elgtis su stipriai koreliuotais aptarnavimo sistemos modeliavimo rezultatais. Be to, tarę, kad W_1=0, modeliavimą pradedame nuo naujo ciklo, todėl nereikalingas joks stabilizacijos periodas. Tam, kad gauti statistiškai reikšmingą E{W}įvertį, tinkama kiekviena trajektorijos atkarpa.

Taigi, matome, kad šis imitacinis eksperimentas padėjo suprasti, kaip spręsti sunkias „taktines“ problemas, su kuriomis susidūrėme pradžioje, o būtent, kaip pradėti modeliavimą, nuo kurio momento rinkti duomenis, ką daryti su stipriai koreliuotais duomenimis.

1.2 REGENERACINIS METODAS

Regeneracinis metodas tai – stochastinių sistemų, kurios pasižymi savybe laikas nuo laiko regeneruotis, modeliavimo rezultatų analizės būdas. Jei modeliavimo rezultatai traktuojami kaip stochastinis procesas, tai regeneraciniai procesai turi savybę nuolat grįžti į tam tikrą regeneracijos tašką, kuriuo pradedant, tolesnis proceso vystymasis nepriklauso nuo jo elgesio praeityje ir apibrėžiamas tuo pačiu tikimybiniu skirstiniu. Po to, jeigu modeliavimo rezultatai grupuojami pagal sugrįžimus į regeneracijos tašką, tai tos grupės yra nepriklausomos ir vienodai pasiskirsčiusios, kas labai palengvina jų statistinę analizę. Šis metodas taikomas bet kokiai diskretaus modeliavimo schemai, kurią galima traktuoti kaip regeneracinį procesą. Diskrečiu vadiname tokį modeliavimo procesą, kuriame imituojami sistemos būklės pokyčiai tik diskrečiais laiko momentais.

1.2.1 REGENERACINIAI PROCESAI DISKREČIAIS LAIKO MOMENTAIS

Atsitiktinių K išmatavimo vektorių seka $\{X_n,\ n \geq 1\}$ yra regeneracinis procesas, jei egzistuoja didėjanti atsitiktinių diskrečių laiko momentų seka $1 \leq \beta_1 < \beta_2 < \ldots$, tokia, kad proceso vystymasis, pradedant nuo kiekvieno šių momentų, apibrėžiamas tuo pačiu tikimybiniu skirstiniu, kaip ir momentu β_1. Tai reiškia, kad tarp bet kurių dviejų gretimų sekos regeneracijos momentų, β_j ir β_{j+1}, proceso dalis

$$\left\{\mathbf{X}_n, \beta_j \leq n < \beta_{j+1}\right\}$$

yra nepriklausoma „tikimybinė kopija“ kitos proceso dalies tarp bet kurių kitų dviejų gretimų regeneracijos momentų. Proceso dalį $\left\{\mathbf{X}_n, \beta_j \leq n < \beta_{j+1}\right\}$ vadinsime j-tuoju ciklu.

Modeliuojamoje N-kanalėje aptarnavimo sistemoje $\mathbf{X}_n = \mathbf{W}_n$ regeneracijos momentais $\{\beta_j, j \geq 1\}$ laikysime tuos laiko momentus kai atėjusi paraiška randa visus laisvus aptarnavimo įrenginius. Tarkime, kad

$$\alpha_j = \beta_{j+1} - \beta_j, \quad j \geq 1.$$

Pastebėsime, kad *regeneracijos periodai* $\{\alpha_j, j \geq 1\}$ tarp sekos regeneracijos momentų nepriklausomi ir vienodai pasiskirstę.

Toliau tarsime, kad $E\{\alpha_j\} < \infty$. Tai nėra pernelyg apribojanti sąlyga. Ji teisinga beveik kiekvienoje eilių teorijos sistemoje.

Regeneracijos savybė – galingas aukščiausio lygio instrumentas analitiniams proceso $\{X_n,\ n \geq 1\}$ rezultatams gauti. Bet kuris regeneracinis procesas diskrečiais laiko momentais, tam tikra prasme pasiskirstęs stacionariai ir dažniausiai tokia įprastine prasme: egzistuoja K-matis atsitiktinis vektorius **X** toks, kad seka $\mathbf{X_n}$ konverguoja į **X**, kai $n \to \infty$, t.y.

$$\lim_{n\to\infty} P\{X_n \le x\} = P\{X \le x\}. \qquad \textbf{(1.2.1.1)}$$

Kompaktiškai tai pateiksime tokiu būdu:

$$\mathbf{X}_n \Rightarrow \mathbf{X}, \text{ kai } n \to \infty.$$

Kadangi dabar žinome, kad regeneracinių modelių atveju skirstinys stacionarus, galime įvertinti jų charakteristikas. Tarkime, kad modeliavimo tikslas yra statistikos $r \equiv E\{f(X)\}$ įvertinimas. (Čia f – išmatuojama funkcija).

Tam iliustruoti, tarkime, kad **X** – tikrinis dydis. Jeigu *f(x)* = *x*, $\forall x$, tai

$$r \equiv E\{f(X)\} = E\{X\};$$

Tokiu būdu, r įvertinimas ekvivalentus $E\{X\}$ įvertinimui.

Pateiksime tuos regeneracijos tyrimus, kurie bus naudojami gaunant pasikliautinąjį intervalą dydžiui *r*. Sakykime, kad

$$Y_j = \sum_{i=\beta_j}^{\beta_{j+1}-1} f(X_i)$$

čia β_j – regeneracijos momentai, Y_j – dydžių $f(X_j)$ suma j-ajame cikle, $\alpha_j = \beta_{j+1} - \beta_j$ yra j-tojo ciklo ilgis. Apibūdinsime fundamentalią regeneracinio ciklo savybę **(1.2.1.3)**.

Teiginys. Tarkime, kad seka $\{(\mathbf{Y}_j, \alpha_j), j \ge 1\}$ susideda iš nepriklausomai ir vienodai pasiskirsčiusių atsitiktinių vektorių.

Jeigu $$E\{|f(X)|\} < \infty, \qquad \textbf{(1.2.1.2)}$$

tai $$r \equiv E\{f(X)\} = \frac{E\{Y_1\}}{E\{\alpha_1\}}. \qquad \textbf{(1.2.1.3)}$$

Žinome, kad (1.2.1.2) teisingas, kadangi vektoriai $\{(\mathbf{Y}_j, \alpha_j), j \ge 1\}$ charakterizuoja regeneracijos sekos ciklus, o tie ciklai nepriklausomi ir vienodai elgiasi tikimybine prasme. Tiriant (1.2.1.3), aišku, kad, esant pakankamai bendroms sąlygoms, su tikimybe lygia 1

$$\frac{f(X_1) + \cdots + f(X_N)}{N} \to E\{f(X)\}, \text{ kai } N \to \infty. \qquad \textbf{(1.2.1.4)}$$

Tarkime, $\beta_1 = 1$. Taigi pirmasis regeneracijos ciklas prasideda nuo pačios modeliavimo pradžios. Jeigu *n*-tasis ciklas baigiasi *N* momentu, tada santykį (1.2.1.4) galime užrašyti taip:

$$\frac{(Y_1 + \ldots Y_n)/n}{(\alpha_1 + \ldots + \alpha_n)/n}, \qquad \textbf{(1.2.1.5)}$$

kadangi Y_j reikšmių $f(X_i)$ suma j-tajame cikle, o α_j j-tojo ciklo ilgis. Santykis (1.2.1.5) su tikimybe, lygia 1, konverguoja į $\frac{E\{Y_1\}}{E\{\alpha_1\}}$, kai $n \to \infty$ (dėl (1.2.1.2) ir didžiųjų skaičių dėsnio [1]). Kadangi $n \to \infty$, kai $N \to \infty$, gauname esminį santykį

$$E\{f(X)\} = \frac{E\{Y_1\}}{E\{\alpha_1\}}.$$

Tuo atveju, kai $\beta_1 > 1$ (tai yra ciklas neprasideda iš karto), santykį (1.2.1.5) būtina pakeisti, taip:

$$\frac{(Y_0 + Y_1 + \ldots Y_n)/n}{(\alpha_0 + \alpha_1 + \ldots + \alpha_n)/n}, \qquad \textbf{(1.2.1.5`)}$$

čia α_0 ir Y_0 atitinka pradinį „pereinamąjį periodą“ modeliuojant ir yra tokie

$$\alpha_0 = \beta_1 - 1 \text{ ir } Y_0 = f(X_1) + \ldots + f(X_{\beta_1 - 1}).$$

Atsitiktinis dydis α_0 pasiskirstęs kitaip nei α_j, $j \geq 1$, o Y_0 kitaip nei Y_j, $j \geq 1$. Tačiau esant dideliems *n, Y_0* bus mažas, lyginant su $Y_1 + Y_2 + \ldots + Y_n$, o α_0 bus mažas, lyginant su $\alpha_1 + \alpha_2 + \ldots + \alpha_n$, todėl santykis (1.2.1.5`) vis dėlto konverguoja į $\frac{E\{Y_1\}}{E\{\alpha_1\}}$, ir (1.2.1.3) teisinga.

Prielaida $E\{|f(X)|\} < \infty$ nėra labai apribojanti ir nesudaro kliūčių regeneracinio metodo naudojimui.

Taigi, dėl nepriklausomų ir vienodai pasiskirsčiusių stebėjimų santykio $\frac{E\{Y_1\}}{E\{\alpha_1\}}$ įvertinimui galima naudoti klasikinės statistikos rezultatus. Šiam dydžiui galima gauti pasikliautinąjį intervalą.

Regeneracinio metodo teorija vienkanalės sistemos atveju išdėstyta [3] knygoje.

1.2.2 PASIKLIAUTINIEJI INTERVALAI

Užduotis tokia: turint nepriklausomas ir vienodai pasiskirsčiusias poras $(Y_1, \alpha_1), (Y_2, \alpha_2), \ldots, (Y_n, \alpha_n)$ apskaičiuoti $100(1-\delta)\%$-tinį pasikliautinąjį intervalą dydžiui $\frac{E\{Y_1\}}{E\{\alpha_1\}}$, kai n pakankamai didelis. Vienas iš tokių metodų tokiam tikimybiniam intervalui gauti remiasi centrine ribine teorema [3].

Žymime

$$V_j = Y_j - r\alpha_j$$

Dydžiai V_j – nepriklausomi ir vienodai pasiskirstę ir $E\{V_j\} = E\{Y_j\} - rE\{\alpha_j\} = 0$. Dydžiai $\overline{Y}, \overline{\alpha}, \overline{V}$ reiškia empirinius vidurkius:

$$\overline{Y} = \frac{1}{n}\sum_{j=1}^{n} Y_j,$$

$$\overline{\alpha} = \frac{1}{n}\sum_{j=1}^{n} \alpha_j,$$

$$\overline{V} = \frac{1}{n}\sum_{j=1}^{n} V_j.$$

Pastebėsime, kad $\overline{V} = \overline{Y} - r\overline{\alpha}$.

Žymime $\sigma^2 = E\{V_j^2\}$

Tada, tarus, kad $0 < \sigma^2 < \infty$, iš centrinės ribinės teoremos išplaukia, kad kiekvienam baigtiniam x

$$\lim_{n\to\infty} P\left\{\frac{\sqrt{n}\overline{V}}{\sigma} \le x\right\} = \Phi(x); \qquad \textbf{(1.2.2.1)}$$

čia Φ – standartinė normaliojo skirstinio funkcija:

$$\Phi(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{\frac{-y^2}{2}}\,dy$$

Prielaida, kad $0 < \sigma^2 < \infty$, neapriboja modeliavimo taikymo.

Toliau iš (1.2.2.1) turime

$$\lim_{n\to\infty} P\left\{\frac{\sqrt{n}\left|\overline{Y} - r\overline{\alpha}\right|}{\sigma} \le x\right\} = \Phi(x),$$

arba

$$\lim_{n\to\infty} P\left\{\frac{\sqrt{n}\left|\hat{r} - r\right|}{\sigma/\overline{\alpha}} \le x\right\} = \Phi(x); \qquad \textbf{(1.2.2.2)}$$

čia $\hat{r} = \frac{\overline{Y}}{\overline{\alpha}}$.

Tačiau betarpiškai iš (1.2.2.2) pasikliautinojo intervalo gauti negalime, lygiai, kaip ir negalime sužinoti σ reikšmės. Tačiau σ galime įvertinti tam tikru būdu:

$$\sigma^2 = E\left\{\left(Y_1 - r\alpha_1\right)^2\right\} = DY_1 - 2r\,\mathrm{cov}\left(Y_1, \alpha_1\right) + r^2 D\alpha_1.$$

Tegu s_{11}, s_{22}, s_{12} reiškia, atitinkamai, Y_j empirinę dispersiją, α_j empirinę dispersiją ir $\left(Y_j, \alpha_j\right)$ koreliacinį momentą, t.y.

$$s_{11} = \frac{1}{n}\sum_{j=1}^{n}\left(Y_j - \overline{Y}\right)^2 = \frac{1}{n-1}\sum_{j=1}^{n} Y_j^2 - \frac{1}{n(n-1)}\left(\sum_{j=1}^{n} Y_j\right)^2,$$

$$s_{22} = \frac{1}{n}\sum_{j=1}^{n}\left(\alpha_j - \overline{\alpha}\right)^2 = \frac{1}{n-1}\sum_{j=1}^{n} \alpha_j^2 - \frac{1}{n(n-1)}\left(\sum_{j=1}^{n} \alpha_j\right)^2,$$

$$s_{12} = \frac{1}{n}\sum_{j=1}^{n}\left(Y_j - \overline{Y}\right)\left(\alpha_j - \overline{\alpha}\right) = \frac{1}{n-1}\sum_{j=1}^{n} Y_j\alpha_j - \frac{1}{n(n-1)}\left(\sum_{j=1}^{n} Y_j\right)\left(\sum_{j=1}^{n} \alpha_j\right).$$

Tegu $s^2 = s_{11} - 2\hat{r}s_{12} + \hat{r}^2 s_{22}$. Tada su tikimybe 1 $s^2 \to \sigma^2$, kai $n \to \infty$.

Todėl (1.2.2.2) pakeičiame σ į s, t.y.

$$\lim_{n\to\infty} P\left\{\frac{\sqrt{n}|\hat{r} - r|}{s/\overline{\alpha}} \le x\right\} = \Phi(x), \qquad \textbf{(1.2.2.3)}$$

Žymime $z_\delta^* = \Phi^{-1}\left(1 - \frac{\delta}{2}\right)$, t.y. $\Phi\left(z_\delta^*\right) = 1 - \frac{\delta}{2}$.

Iš (1.2.2.3) dideliems n turime:

$$P\left\{-z_\delta^* \le \frac{\sqrt{n}|\hat{r} - r|}{s/\overline{\alpha}} \le z_\delta^*\right\} \cong 1 - \delta$$

Tai galima perrašyti taip:

$$P\left\{\hat{r} - \frac{z_\delta^* s}{\overline{\alpha}\sqrt{n}} \le r \le \hat{r} + \frac{z_\delta^* s}{\overline{\alpha}\sqrt{n}}\right\} \cong 1 - \delta \qquad \textbf{(1.2.2.3`)}$$

Tokiu būdu gaunamas $100(1-\delta)\%$-tinis pasikliautinasis intervalas dydžiui $r \equiv E\{f(X)\}$:

$$\hat{I} = \left[\hat{r} - \frac{z_\delta^* s}{\overline{\alpha}\sqrt{n}}, \hat{r} + \frac{z_\delta^* s}{\overline{\alpha}\sqrt{n}}\right] \qquad \textbf{(1.2.2.4)}$$

Pastebėkime, kad jeigu dideliems n per $\hat{J}$ išreikštume intervalo $\hat{I}$ ilgį, tai su didele tikimybe

$$\hat{J} \cong \frac{2z_\delta^*\sigma}{E\{\alpha_1\}\sqrt{n}}$$

Todėl tam, kad sumažinti pasikliautinojo intervalo $\hat{I}$ ilgį du kartus (esant tokiam pačiam patikimumo lygiui), būtina keturis kartus padidinti modeliuojamų regeneracinių ciklų skaičių. Deja tai yra atsitiktinis dydis, kurio neįmanoma tiksliai koreguoti, tačiau modeliavimo metu buvo pastebėta tam tikra, artima tiesinei priklausomybė tarp modeliavimo laiko ir regeneracinių ciklų skaičiaus.

1.3 MAKSIMUMŲ RIBINĖS TEOREMOS

Pritaikant ribines maksimumų teoremas, galime skaičiuoti pasikliautinuosius intervalus ne tik stebimų dydžių vidurkiams, bet ir jų maksimalioms reikšmėms.

Pateikiame teoremas, kurios įrodytos [2] monografijoje.

1.3.1 teorema. Jei egzistuoja tokios normalizavimo konstantos a_n ir $b_n > 0$, su kuriomis

$$n\left(1 - F\left(a_n + b_n \cdot x\right)\right) \to z\left(x\right), \tag{1.3.1}$$

tai

$$P\left(\frac{Z_n - a_n}{b_n} < x\right) \to e^{-z(x)}. \tag{1.3.2}$$

1.3.2 teorema. Tarkime, kad $\omega\left(F\right) = +\infty$ ir egzistuoja tokia konstanta $\gamma > 0$, kad

$$\lim_{t \to +\infty} \frac{1 - F\left(t \cdot x\right)}{1 - F\left(t\right)} = x^{-\gamma}. \tag{1.3.3}$$

Tada egzistuoja tokia konstantų $b_n > 0$ seka, kad

$$\lim_{t \to +\infty} P\left(b_n \cdot Z_n < x\right) = H_{1,\gamma}\left(x\right). \tag{1.3.4}$$

Čia

$$H_{1,\gamma}\left(x\right) = \begin{cases} e^{-x^{-\gamma}}, x > 0 \\ 0, x \le 0 \end{cases}. \tag{1.3.5}$$

Normavimo konstantas b_n galima parinkti taip:

$$b_n = \inf\left\{x : 1 - F\left(x\right) \le \frac{1}{n}\right\}. \tag{1.3.6}$$

1.3.3 teorema. Tarkime, kad $\omega\left(F\right) < +\infty$ ir skirstinio funkcija $F^*\left(x\right) = F\left(\omega\left(F\right) - \frac{1}{x}\right)$ tenkina **(1.3.3)** sąlygą. Tada galima rasti tokias konstantų a_n ir $b_n > 0$ sekas, kad

$$\lim_{n \to \infty} P\left(Z_n < b_n \cdot x + a_n\right) = H_{2,\gamma}\left(x\right), \tag{1.3.7}$$

čia

$$H_{2,\gamma}\left(x\right) = \begin{cases} 1, x \ge 0 \\ e^{-(-x)^{\gamma}}, x < 0 \end{cases}. \tag{1.3.8}$$

Normalizavimo konstantas a_n ir $b_n > 0$ galima parinkti tokiu būdu:

$$\begin{aligned} a_n &= \omega\left(F\right) \\ b_n &= \omega\left(F\right) - \inf\left\{x : 1 - F\left(x\right) \le \frac{1}{n}\right\}. \end{aligned} \tag{1.3.9}$$

1.3.4 teorema. Tarkime, kad kiekvienam α teisingas sąryšis

$$\int_{\alpha}^{\omega(F)} (1 - F(y))dy < \infty \qquad \textbf{(1.3.10)}$$

ir visiems $t : \alpha(F) < t < \omega(F)$ apibrėžiame funkciją $R(t)$:

$$R(t) = \frac{\int_{t}^{\omega(F)} (1 - F(y))dy}{1 - F(t)}. \qquad \textbf{(1.3.11)}$$

Sakykime, kad visiems $x \in R$ egzistuoja riba

$$\lim_{t \to \omega(F)} \frac{1 - F(t + x \cdot R(t))}{1 - F(t)} = e^{-x} \qquad \textbf{(1.3.12)}$$

Tada egzistuoja tokios konstantų a_n ir $b_n > 0$ sekos, su kuriomis

$$\lim_{n \to \infty} P(Z_n < b_n \cdot x + a_n) = H_{3,0}(x), \qquad \textbf{(1.3.13)}$$

čia

$$H_{3,0}(x) = e^{-e^{-x}}, x \in R. \qquad \textbf{(1.3.14)}$$

Normalizavimo konstantas a_n ir $b_n > 0$ galima parinkti tokiu būdu:

$$a_n = \inf\left\{x : 1 - F(x) \le \frac{1}{n}\right\}.$$
$$b_n = R(a_n)$$

1.3.1 MAKSIMUMŲ PERKĖLIMO TEOREMA

Pateikiame teoremą, kuri įrodyta [2] monografijoje.

1.3.1.1 teorema. Tegul

$$\lim_{n \to \infty} u_n(x) \to u(x) \qquad \textbf{(1.3.1.1)}$$

ir

$$\lim_{n \to \infty} P\left(\frac{N_n}{n} \le x\right) = \lim_{n \to \infty} A_n(n \cdot x) = A(x) \qquad \textbf{(1.3.1.2)}$$

ir

$$\lim_{n \to \infty} P\left(\frac{Z_{N_n} - a_n}{b_n} \le x\right) = \Psi(x), \qquad \textbf{(1.3.1.3)}$$

kur

$$\Psi(x) = \int_{0}^{\infty} e^{-z \cdot u(x)} dA(z). \qquad \textbf{(1.3.1.4)}$$

1.3.2 ATSITIKTINIŲ DYDŽIŲ MAKSIMUMŲ RIBINIS SKIRSTINYS, KAI n – ATSITIKTINIS

Tarkime, kad ribinė maksimumų skirstinio funkcija yra

$$H_{1,\alpha\cdot\tau}(x)=e^{-x^{-\alpha\cdot\tau}},\ x>0. \qquad \textbf{(1.3.2.1)}$$

Tegul $N_1,N_2,...,N_n$ yra teigiami nepriklausomi diskretūs dydžiai, pasiskirstę pagal skirstinio funkciją:

$$A(x)=x,\ 0\le x\le 1. \qquad \textbf{(1.3.2.2)}$$

Nagrinėjame struktūrą $Z_{N_n}=\max(X_1,...,X_{N_n})$.

Pritaikę perkėlimo 1.3.1.1 teoremą randame ribinį skirstinį:

$$\Psi(x)=\int_0^{\infty}H^z(x)dA(z);$$

Gauname ribinį skirstinį:

$$\Psi(x)=\int_0^{1}H^z(x)dz=\frac{1}{1-\ln H(x)}. \qquad \textbf{(1.3.2.3)}$$

Ribinę maksimumų skirstinio funkciją (1.3.2.1) įstatome į funkciją (1.3.2.3), gauname ribinį skirstinį, kai komponenčių skaičius n yra atsitiktinis:

$$\psi(x)=\frac{1}{1+x^{-\alpha\cdot\tau}}. \qquad \textbf{(1.3.2.4)}$$

1.3.3 ATSITIKTINIŲ DYDŽIŲ, PASISKIRSČIUSIŲ PAGAL EKSPONENTINĮ SKIRSTINĮ, MAKSIMUMO RIBINIS SKIRSTINYS

Tarkime, kad $X_1,X_2,...,X_n$ yra nepriklausomi vienodai pasiskirstę atsitiktiniai dydžiai su skirstinio funkcija

$$F(x)=1-e^{-\lambda\cdot x},\ x\ge 0,\ \lambda>0 \qquad \textbf{(1.3.3.1)}$$

Nagrinėjame struktūrą $Z_n=\max(X_1,...,X_n)$.

Rasime dydžio Z_n ribinį skirstinį. Pasinaudodami 1.3.4 teorema randame funkciją $R(t)$:

$$R(t)=\frac{\int_t^{\infty}(1-F(y))dy}{1-F(t)}=\frac{\int_t^{\infty}e^{-\lambda\cdot y}dy}{e^{-\lambda\cdot t}}=\frac{1}{\lambda},$$

Apskaičiuojame ribą:

$$\lim_{t\to\infty}\frac{1-F(t+x\cdot R(t))}{1-F(t)}=\lim_{t\to\infty}\frac{e^{-\lambda\cdot\left(t+\frac{1}{\lambda}\cdot x\right)}}{e^{-\lambda\cdot t}}=e^{-x}.$$

Kadangi $\varpi(F)=+\infty$ ir $\lim\limits_{t\to\infty}\dfrac{1-F(t+x\cdot R(t))}{1-F(t)}=e^{-x}$, tai dydis $\dfrac{Z_n-a_n}{b_n}$ silpnai konverguoja į $H_{3,0}(x)$.

Rasime konstantą a_n:

$1-F(a_n)=\dfrac{1}{n}$; $e^{-\lambda\cdot a_n}=\dfrac{1}{n}$; $a_n=\dfrac{\ln n}{\lambda}$.

Rasime konstantą b_n:

$b_n=R(a_n)=\dfrac{1}{\lambda}$.

Atsitiktinių dydžių, pasiskirsčiusių pagal eksponentinį skirstinį, maksimumo ribinis skirstinys:

$$P\left(\frac{Z_n-\dfrac{\ln n}{\lambda}}{\dfrac{1}{\lambda}}<x\right)\xrightarrow[n\to+\infty]{}e^{-e^{-x}},\ x\in R. \qquad \textbf{(1.3.3.2)}$$

1.3.4 ATSITIKTINIŲ DYDŽIŲ, PASISKIRSČIUSIŲ PAGAL VEIBULO SKIRSTINĮ, MAKSIMUMO RIBINIS SKIRSTINYS

Tarkime, kad $X_1,X_2,...,X_n$ yra nepriklausomi vienodai pasiskirstę atsitiktiniai dydžiai su skirstinio funkcija

$$F(x)=1-e^{-\lambda\cdot x^{\alpha}},\ x\ge 0,\ \lambda>0,\ \alpha>0 \qquad \textbf{(1.3.4.1)}$$

Kai $\alpha=1$, gauname eksponentinį skirstinį.

Nagrinėjame struktūrą $Z_n=\max(X_1,...,X_n)$.

Rasime dydžio Z_n ribinį skirstinį. Randame funkciją $R(t)$:

$$R(t)=\frac{\int\limits_t^{\infty}(1-F(y))dy}{1-F(t)}=\frac{\int\limits_t^{\infty}e^{-\lambda\cdot x^{\alpha}}dy}{e^{-\lambda\cdot t^{\alpha}}}.$$

Apskaičiuojame integralą:

$$\int\limits_t^{\infty}e^{-\lambda\cdot x^{\alpha}}dy=-\int\limits_t^{\infty}\frac{1}{\lambda\cdot\alpha\cdot y^{\alpha-1}}de^{-\lambda\cdot y^{\alpha}}\sim-\frac{1}{\lambda\cdot\alpha\cdot t^{\alpha-1}}\cdot e^{-\lambda\cdot t^{\alpha}}\Bigg|_t^{\infty}=\frac{1}{\lambda\cdot\alpha\cdot t^{\alpha-1}}\cdot e^{-\lambda\cdot t^{\alpha}}.$$

$$R(t)\sim\frac{1}{\lambda\cdot\alpha\cdot t^{\alpha-1}},\ \text{kai}\ t\to\infty.$$

Apskaičiuojame ribą:

$$\lim_{t\to\infty}\frac{1-F(t+x\cdot R(t))}{1-F(t)}=\lim_{t\to\infty}\frac{1-F\left(t+x\cdot\dfrac{1}{\lambda\cdot\alpha\cdot t^{\alpha-1}}\right)}{e^{-\lambda\cdot t^{\alpha}}}=\lim_{t\to\infty}e^{-\lambda\cdot t^{\alpha}\cdot\left(\left(1+\frac{x}{\lambda\cdot\alpha\cdot t^{\alpha}}\right)^{\alpha}-1\right)}=e^{-x}.$$

Kadangi $\varpi(F)=+\infty$ ir $\lim_{t\to\infty}\frac{1-F(t+x\cdot R(t))}{1-F(t)}=e^{-x}$, tai dydis $\frac{Z_n-a_n}{b_n}$ silpnai konverguoja į $H_{3,0}(x)$.

Rasime konstantą a_n:

$$1-F(a_n)=\frac{1}{n};\ e^{-\lambda\cdot a_n{}^{\alpha}}=\frac{1}{n};\ a_n=\left(\frac{\ln n}{\lambda}\right)^{\frac{1}{\alpha}}.$$

Rasime konstantą b_n:

$$b_n=R(a_n)=\frac{\left(\frac{\ln n}{\lambda}\right)^{\frac{1}{\alpha}}}{\alpha\cdot\ln n}.$$

Atsitiktinių dydžių, pasiskirsčiusių pagal Veibulo skirstinį, maksimumo ribinis skirstinys:

$$P\left(\frac{Z_n-\left(\frac{\ln n}{\lambda}\right)^{\frac{1}{\alpha}}}{\frac{\left(\frac{\ln n}{\lambda}\right)^{\frac{1}{\alpha}}}{\alpha\cdot\ln n}}<x\right)\xrightarrow[n\to+\infty]{}e^{-e^{-x}},\ x\in R. \qquad \textbf{(1.3.4.2)}$$

2. TIRIAMOJI DALIS IR REZULTATAI

2.1 REGENERACINIO METODO TAIKYMO DAUGIAKANALĖJE SISTEMOJE KOREKTIŠKUMO PATIKRINIMAS

Geriausiai išvystyta teorija, nusakanti aptarnavimo sistemos darbą be modeliavimo, naudoja eksponentinius laikus (tiek paraiškų atėjimo, tiek jų aptarnavimo). Taigi su tokiais atsitiktinių dydžių skirstiniais buvo testuotas sukurtos programos darbo korektiškumas, skaičiuojant šias, sistemos efektyvumą parodančias, charakteristikas:

- Vidutinio kanalų užimtumo įvertį;
- Vidutinio eilės ilgio įvertį;
- Vidutinio laukimo eilėje laiko įvertį.

Taip pat, naudojant regeneracinį metodą, buvo rasti ne tik ieškomų dydžių vidurkiai, bet ir jų pasikliautinieji intervalai (su 90% tikimybe).

Buvo skaičiuojamos teorinės charakteristikos ir lyginamos su modeliavimo rezultatais, gaunamais po skirtingų iteracijų skaičiaus.

Žinome, kad tiriant paprasčiausių srautų variantus ir turėdami atėjimo srauto intensyvumą λ, apdorojimo intensyvumą μ ir apdorojimo kanalų skaičių n, galime ieškoti sistemos darbo efektyvumą nusakančių charakteristikų, naudojant šias formules [4]:

$\rho = \frac{\lambda}{\mu}$, santykis tarp atėjimo ir aptarnavimo laikų skirstinių intensyvumų;

$p_0 = \frac{1}{\sum_{k=0}^{n} \frac{\rho^k}{k!} + \frac{\rho^{n+1}}{n!}}$, tikimybė, kad visi kanalai laisvi;

$p_n = \frac{\rho^n}{n!} \cdot p_0$, tikimybė, kad visi kanalai užimti, bet eilės nėra;

$t_{vid} = \frac{n}{\mu \cdot (n-\rho)^2} \cdot p_n$, vidutinis paraiškos laukimo laikas eilėje;

$M_{eil} = p_n \cdot \frac{\rho}{n \cdot \left(1 - \frac{\rho}{n}\right)^2}$, vidutinis eilės ilgis;

$N_u = n - \sum_{k=0}^{n-1} \left[(n-k) \cdot \frac{\rho^k}{k!} \cdot p_0 \right]$, vidutinis užimtų kanalų skaičius.

Modeliavimas buvo atliekamas naudojantis programa, veikiančia regeneraciniu metodu. Modeliavimo adekvatumo tikrinimui, buvo modeliuota sistema su paprasčiausiais paraiškų atėjimo srautais ir eksponentiniais aptarnavimo laikais, esant neribotai eilei ir n aptarnavimo kanalų.

Patikrinimas buvo atliktas, naudojant tokius duomenis:

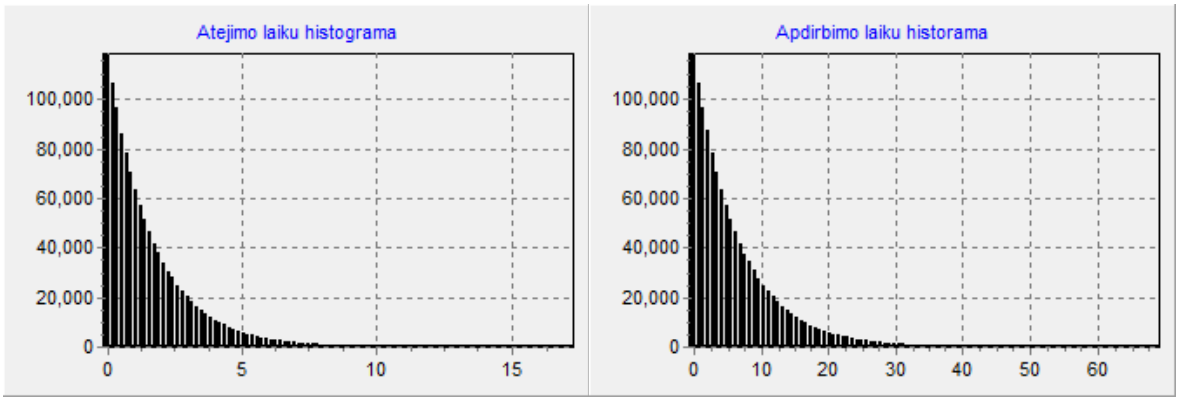
$$\lambda = 0.60,\ \mu = 0.15,\ n=5.$$

Čia λ – atėjimo intensyvumas, μ – apdorojimo intensyvumas (vieno kanalo), n – kanalų skaičius.

Ieškomų charakteristikų reikšmės buvo apskaičiuojamos teoriškai, atliekami 10^6 ir $2x10^6$ laiko vienetų trukmės modeliavimai (žr. 2.1.1 lent.).

$\lambda = 0.60$, $\mu = 0.15$, n=5	Teorinė reikšmė	10^6 laiko vienetų modeliavimas	$2x10^6$ laiko vienetų modeliavimas
Vidutinis kanalų užimtumas	80%	79.7969%	80.0619%
Vidutinis eilės ilgis	2.216	2.1688	2.2130
Vidutinis laukimo eilėje laikas	3.694	3.6233	3.6862
Regeracinių ciklų skaičius		7952	15656

2.1.1 lent. modeliavimo duomenys



2.1.1 pav. Modeliavimo metu generuotų laikų histogramos $2x10^6$ atveju

2.2 PASIKLIAUTINŲJŲ INTERVALŲ KOREKTIŠKUMO PATIKRINIMAS

Iš 2.1. skyrelio jau žinome, kad programa korektiškai modeliuoja aptarnavimo sistemos charakteristikas, tačiau nežinome apie pasikliautinųjų intervalų patikimumą.

Buvo sukurta programos posistemė, kuri įgalina vartotoją atlikti norimą kiekį atskirų modeliavimų su vienodais pradiniais parametrais ir gavus visų modeliavimų rezultatus palyginti, kiek iš jų pakliuvo į numatytus pasikliautinuosius intervalus, gautus kituose eksperimentuose. Tokiu būdu buvo patikrinti visi gauti vidurkiai su visais pasikliautinaisiais intervalais ir surastas procentas, kiek modeliavimo rezultatų pateko į numatytus rėžius.

Dėl posistemės reiklumo kompiuterinio darbo laikui buvo apsiribota 100 modeliavimų su 10^6 ir $2x10^6$ laiko vienetų ir 20 modeliavimų su $5x10^6$ laiko vienetų. Naudojant paprasčiausią atėjimo ir eksponentinį apdorojimo srautus gauti šie rezultatai (žr. 2.2.1 lent.).

Patekimai i pasikliautinuosius intervalus	10^6 laiko vienetų modeliavimas	$2x10^6$ laiko vienetų modeliavimas	$5x10^6$ laiko vienetų modeliavimas
Kanalų užimtumas	84.99%	90.64%	90.50%
Eilės ilgis	84.04%	86.94%	88.25%
Laukimo eilėje laikas	84.08%	86.71%	88.50%

2.2.1 lent. pasikliautinųjų intervalų testavimo rezultatai

Programa naudoja 90% duodantį normaliojo pasiskirstymo kvantilį lygų $z_\delta^* = 1{,}645$.

Pilnesni modeliavimo rezultatai pateikiami prieduose, atitinkamai: Priedas Nr.1 – 10^6, Priedas Nr.2 – $2x10^6$, Priedas Nr.3 – $5x10^6$ laiko vienetų modeliavimo rezultatai.

2.3 EKSPERIMENTINIAI MODELIAVIMAI

Bandant programos darbą buvo atlikti eksperimentiniai modeliavimai su skirtingais atėjimo ir apdorojimo laikų pasiskirstymais, bet išlaikant tą patį laiko momentų skaičių ir pradinius duomenis. Visais bandymų atvejais buvo naudoti tokie pradiniai nustatymai: atėjimo ir apdorojimo laikų pasiskirstymai vienodi (eksponentinis – eksponentinis, tolygusis – tolygusis ir Veibulo – Veibulo), atėjimo intensyvumas 0.6, apdorojimo intensyvumas 0.15 ir 5 aptarnavimo kanalai. Atėjimo ir apdorojimo intensyvumas, bei kanalų skaičius buvo palikti tie patys, kad būtų galima palyginti sistemos charakteristikų priklausomybę tik nuo pasiskirstymo pasirinkimo.

Pateikiami eksperimentai, kurie buvo pakartoti su tikslu patikrinti pasikliovimo intervalų korektiškumą.

2.3.1 EKSPONENTINIS SKIRSTINYS ($5x10^6$ MODELIAVIMAS)

Vidutinio kanalų užimtumo pasikliautinasis intervalas:

(3.9996; 4.0104)

Užimtų kanalų vidurkio įvertis :

4.005

Kanalų užimtumas simuliacijos metu : 80.1%

Vidutinio eilės ilgio pasikliautinasis intervalas:

(2.1968; 2.263)

Vidutinio eilės ilgio įvertis :

2.2299

Vidutinio laukimo eilėje laiko pasikliautinasis intervalas:

(3.6638; 3.7707)

Laukimo eilėje vidurkio įvertis :

3.7172

Maksimali eilė simuliacijos metu: 48

Regeneracinių ciklų skaičius: 39123

Atlikus 20 analogiškų eksperimentų, gauti tokie pasikliautinųjų intervalų empiriniai įverčiai:

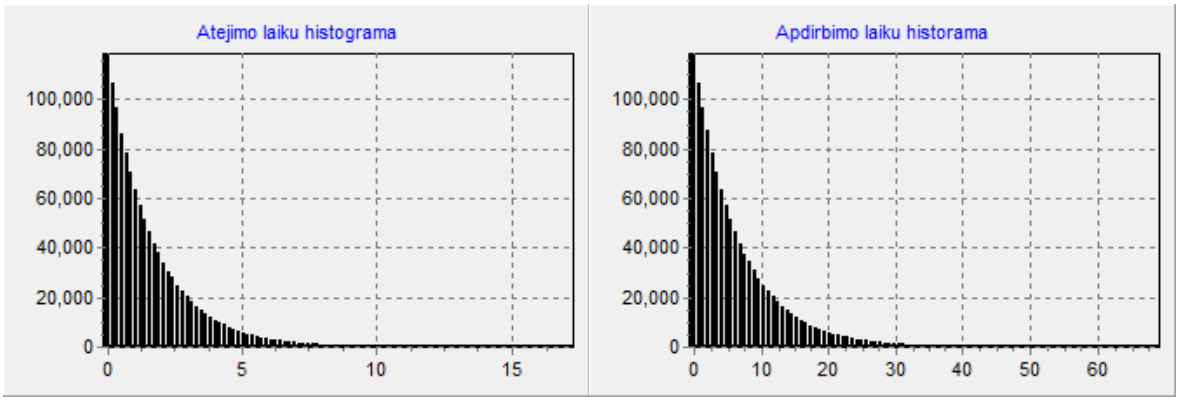
Patekimas į pasikliautinuosius intervalus:

Užimti kanalai: 90.50%

Eiles ilgis: 88.25%

Laukimo laikas: 88.50%

Pilnesni modeliavimo duomenys pateikiami Priede Nr.3.



2.3.1.1 pav. Modeliavimo metu generuotų laikų histogramos

2.3.2 TOLYGUSIS SKIRSTINYS (10^6 MODELIAVIMAS)

Vidutinio kanalų užimtumo pasikliautinasis intervalas:

(3.9938; 4.0079)

Užimtų kanalų vidurkio įvertis :

4.0009

Kanalų užimtumas simuliacijos metu : 80.018%

Vidutinio eilės ilgio pasikliautinasis intervalas:

(0.6243; 0.6538)

Vidutinio eilės ilgio įvertis :

0.6390

Vidutinio laukimo eilėje laiko pasikliautinasis intervalas:

(1.0410; 1.0889)

Laukimo eilėje vidurkio įvertis :

1.0649

Maksimali eilė simuliacijos metu: 19

Regeneracinių ciklų skaičius: 2795

Atlikus 100 analogiškų eksperimentų, gauti tokie pasikliautinųjų intervalų empiriniai įverčiai:

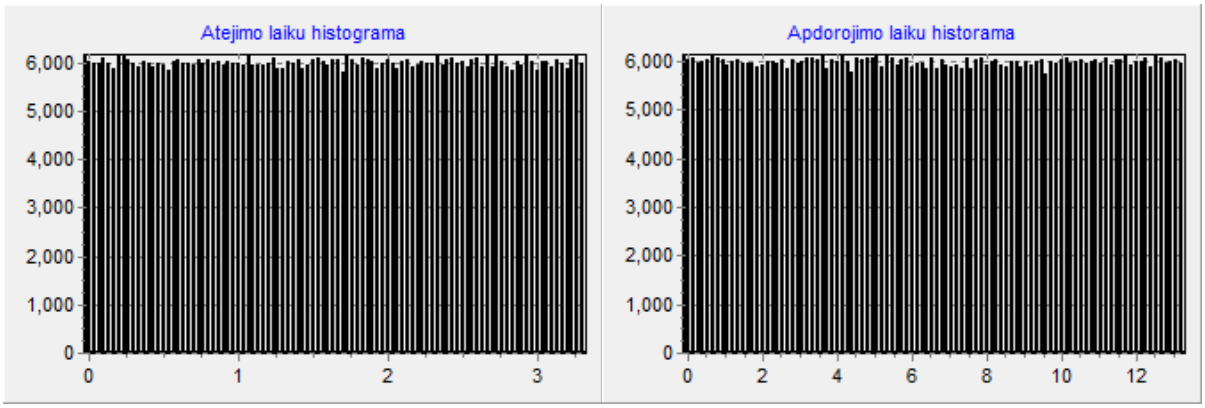
Patekimas į pasikliautinuosius intervalus:

Užimti kanalai: 87.81%

Eiles ilgis: 89.16%

Laukimo laikas: 89.10%

Pilnesni modeliavimo duomenys pateikiami Priede Nr.4.



2.3.2.1 pav. Modeliavimo metu generuotų laikų histogramos

2.3.3 VEIBULO SKIRSTINYS

(10^6 MODELIAVIMAS, PARAMETRAS = 1.5)

Vidutinio kanalų užimtumo pasikliautinasis intervalas:

(3.9898; 4.0052)

Užimtų kanalų vidurkio įvertis :

3.9975

Kanalų užimtumas simuliacijos metu : 79.95%

Vidutinio eilės ilgio pasikliautinasis intervalas:

(0.8724; 0.9155)

Vidutinio eilės ilgio įvertis :

0.8940

Vidutinio laukimo eilėje laiko pasikliautinasis intervalas:

(1.3133; 1.3765)

Laukimo eilėje vidurkio įvertis :

1.3449

Maksimali eilė simuliacijos metu: 22

Regeneracinių ciklų skaičius: 4723

Atlikus 100 analogiškų eksperimentų, gauti tokie pasikliautinųjų intervalų empiriniai įverčiai:

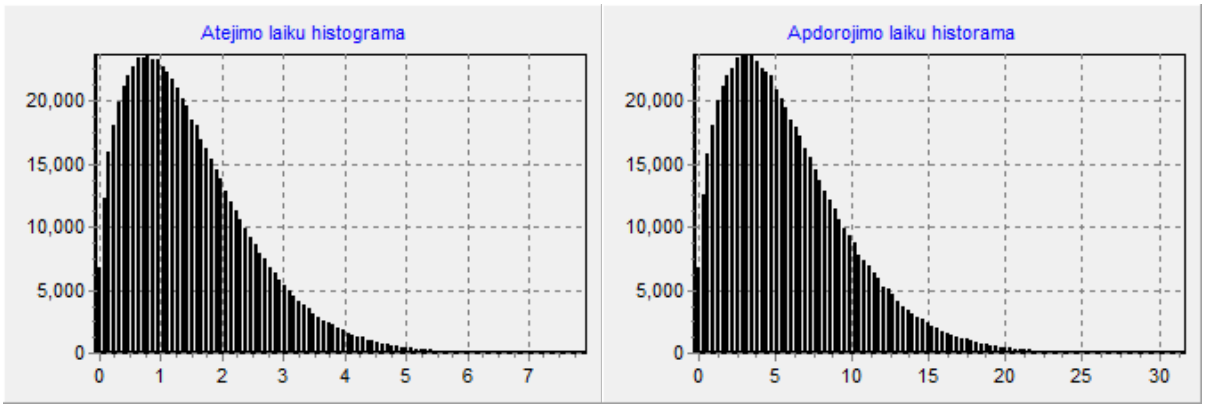
Patekimas į pasikliautinuosius intervalus:

Užimti kanalai: 88.03%

Eiles ilgis: 87.44%

Laukimo laikas: 87.28%

Pilnesni modeliavimo duomenys pateikiami Priede Nr.5.



2.3.3.1 pav. Modeliavimo metu generuotų laikų histogramos

3. PROGRAMINĖ REALIZACIJA IR INSTRUKCIJA VARTOTOJUI

3.1 PASIKLIAUTINOJO INTERVALO SKAIČIAVIMO ALGORITMAS

Apytikslio 100(1-δ)%-tinio pasikliautinojo intervalo dydžiams $r_i \equiv E\{f_i(X)\}$ gavimo procedūra:

1. Sumodeliuojame *n* regeneracinių ciklų.
2. Apskaičiuojame $Y^i{}_j$ ir α_j kiekvienam *j*-tajam ciklui, kur $Y^i{}_j$ – dydžių $f(\mathbf{X}_i)$ (užimtų kanalų, eilės ilgio, laukimo eilėje laiko) suma *j*-tame cikle ir α_j – *j*-tojo ciklo ilgis.
3. Apskaičiuojame statistikas:

$$\overline{Y}^i = \frac{1}{n}\sum_{j=1}^{n} Y^i{}_j, \qquad \overline{\alpha} = \frac{1}{n}\sum_{j=1}^{n}\alpha_j, \qquad \hat{r}_i = \frac{\overline{Y}^i}{\overline{\alpha}},$$

$$s_{11}^i = \frac{1}{n-1}\sum_{j=1}^{n}(Y^i{}_j)^2 - \frac{1}{n(n-1)}\left(\sum_{j=1}^{n} Y^i{}_j\right)^2,$$

$$s_{22} = \frac{1}{n-1}\sum_{j=1}^{n}\alpha_j^2 - \frac{1}{n(n-1)}\left(\sum_{j=1}^{n}\alpha_j\right)^2,$$

$$s_{12}^i = \frac{1}{n-1}\sum_{j=1}^{n} Y^i{}_j\alpha_j - \frac{1}{n(n-1)}\left(\sum_{j=1}^{n} Y^i{}_j\right)\left(\sum_{j=1}^{n}\alpha_j\right),$$

$$(s^i)^2 = s_{11}^i - 2\hat{r}s_{12}^i + \hat{r}^2 s_{22}$$

4. Suformuojame pasikliautinuosius intervalus.

$$\hat{r}_i \pm \frac{z_\delta^* s^i}{\overline{\alpha}\sqrt{n}}$$

čia $z_{\delta}^{*} = \Phi^{-1}\left(1 - \frac{\delta}{2}\right)$ ir Φ – standartinė normalioji pasiskirstymo funkcija.

Naudojame $z_{\delta}^{*} = 1{,}645$, kad gautume 90% pasikliautinąjį intervalą.

3.2 ATSITIKTINIŲ SKAIČIŲ GENERAVIMAS

Programa generuoja pseudo atsitiktinius skaičius, remiantis šiomis taisyklėmis:

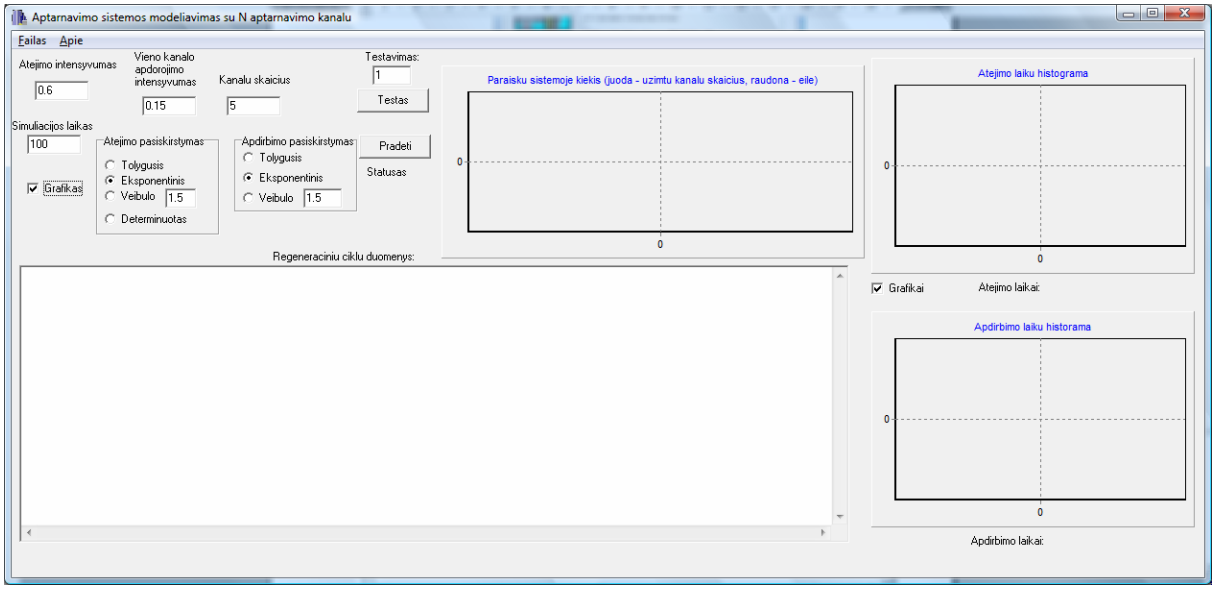
- Tolygusis atvejis – C++ Builder programinis paketas pateikia tolygaus atsitiktinio dydžio gavimo metodą rand(). Po to gautą pseudo atsitiktinį dydį dauginame iš 2, daliname iš intensyvumo ir normuojame, dalindami iš RAND_MAX skaičiaus, tokiu būdu gaudami tolygiai pasiskirsčiusį atsitiktinį dydį, priklausantį intervalui $(0, \frac{2}{\lambda})$, čia λ - pasiskirstymo intensyvumo parametras.
- Eksponentinis atvejis [5] – naudodami tolygius pseudo atsitiktinius skaičius x iš intervalo (0,1), eksponentiškai pasiskirsčiusius skaičius T gauname, naudodami formulę $T = \frac{-\ln(x)}{\lambda}$, nes $F^{-1}(x) = \frac{-\ln(1-x)}{\lambda}$, čia λ - pasiskirstymo intensyvumo parametras. Formulėje 1-x galime pakeisti į x, nes jų pasiskirstymas yra vienodas.
- Veibulo atvejis [6] – naudodami tolygius pseudo atsitiktinius skaičius x iš intervalo (0,1), pagal Veibulo pasiskirstymą pasiskirsčiusius skaičius T gauname, naudodami formulę $T = \lambda(-\ln(x))^{1/k}$, čia k – formos parametras, λ - skalės parametras. Skalės parametras yra dydis atvirkščiai proporcingas intensyvumo parametrui [7], taigi skaičiavimo metu naudojama formulė $T = \frac{(-\ln(x))^{1/k}}{\lambda}$, kur λ - pasiskirstymo intensyvumo parametras, kaip ir kitais atvejais.
- Determinuotas atvejis – pats paprasčiausias. Determinuotas laiko intervalas gaunamas naudojant formulę $\frac{1}{\lambda}$, čia λ - pasiskirstymo intensyvumo parametras.

3.3 PROGRAMA

Aptarnavimo sistemos su N aptarnavimo kanalų modeliavimui buvo sukurta programa, kuri proceso simuliacijos metu atkartoja tikėtiną realios sistemos proceso vystymąsi, naudodama regeneracijos ciklus, kas įgalina nesunkiai surasti ieškomus dydžius, naudojant klasikinius statistinius metodus. Programa sumodeliuoja ir pateikia aptarnavimo kanalų užimtumo, eilės ilgio ir laukimo eilėje laiko vidurkių įverčius ir vidurkių pasikliautinuosius intervalus, taip pat randama maksimali eilė, susidaranti proceso metu.

3.4 ATMINTINĖ VARTOTOJUI

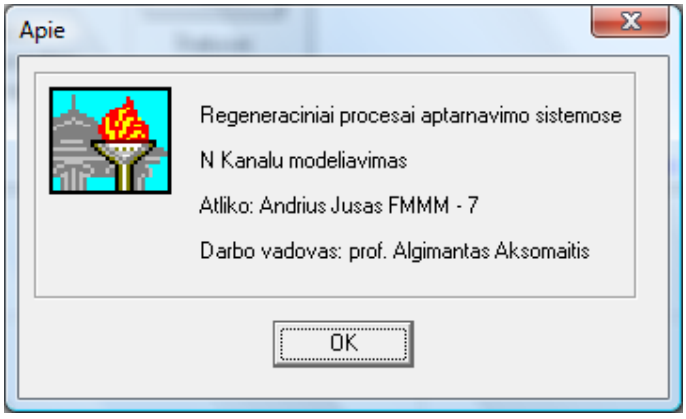
N-kanalė aptarnavimo sistema modeliuojama ir visi skaičiavimai atliekami C++ Builder 5.0 programiniu paketu sukurta programa. Rezultatai išvedami į failus „iteracijos.txt“, „regeneracijos ciklai.txt“ ir pačiame programos lange. Paleidus failą Project1.exe įsijungia pagrindinis langas 3.3.1 pav.



3.4.1 pav. Pagrindinis programos langas

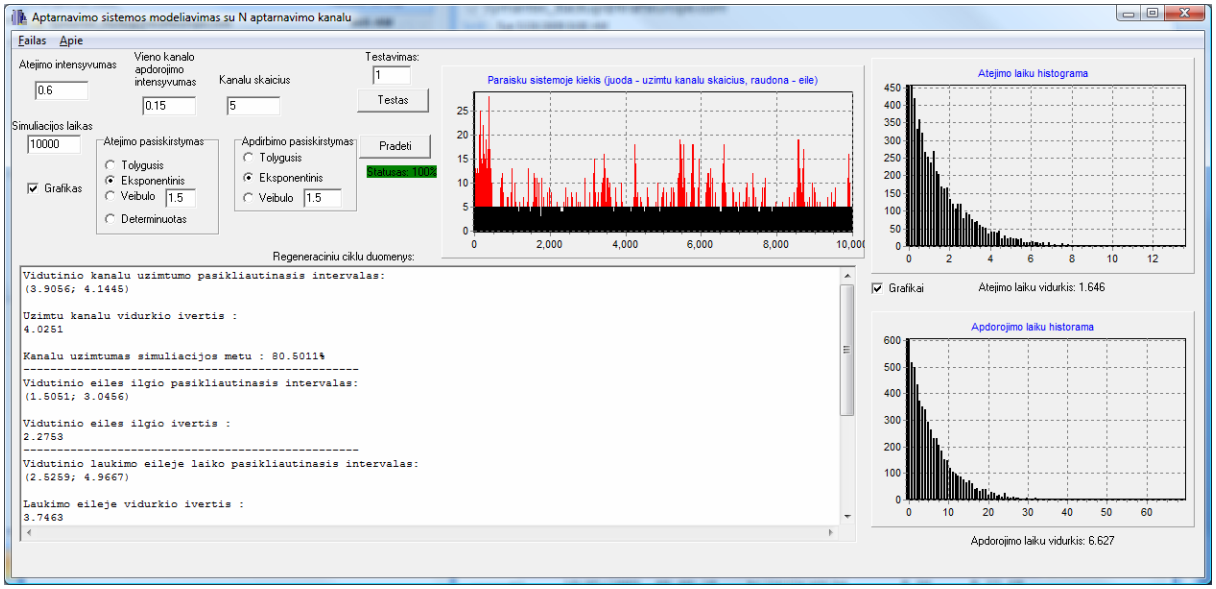
Jame vartotojas gali pasirinkti paraiškų atėjimo ir apdorojimo laikų pasiskirstymo tipą ir intensyvumą (Veibulo pasiskirstymo atveju – dar ir formos parametrus), simuliacijos laiką. Taip pat dėl grafinės optimizacijos leidžiama valdyti realaus laiko grafiko braižymą, kas esant dideliems iteracijų skaičiams sumažina mašininio laiko sunaudojimą. Pasiskirstymų grafikus taip pat galima išjungti norint pagreitinti programos darbą.

Paspaudus mygtuką „Apie“ pasirodys pranešimas su informacija apie programos kūrėją ir darbo vadovą 3.4.2 pav.



3.4.2 pav. „Apie“ langas

Mygtuku „Pradėti“ paleidus programos vykdymą, pagal nurodytus kintamuosius, programa (pasibaigus nurodytam modeliavimo laikui) išves rezultatus į ekraną 3.4.3 pav.



3.4.3 pav. Programos langas modeliavimo pabaigoje

Modeliavimo pabaigoje programa į ekraną išveda šiuos duomenis:

- Vidutinio kanalų užimtumo pasikliautinąjį intervalą;
- Vidutinio kanalų užimtumo įvertį;
- Vidutinį kanalų užimtumą simuliacijos metu procentais;
- Vidutinio eilės ilgio pasikliautinąjį intervalą;
- Vidutinio eilės ilgio įvertį;
- Vidutinio laukimo eilėje pasikliautinąjį intervalą;
- Vidutinio laukimo eilėje laiko įvertį;
- Maksimalios eilės ilgį viso modeliavimo metu;
- Regeneracinių ciklų skaičių.

Programa modeliavimo metu į failus išveda tokius duomenis:

- Konkrečios iteracijos duomenis: iteracijos numeris, ar per laiko tarpą tarp praeitos ir šios iteracijos atėjo paraiška (0 arba 1), koks dabartinis eilės ilgis ir kiek yra užimtų kanalų (failas „iteracijos.txt“);
- Regeneracinių ciklų duomenis: regeneracinio ciklo numerį, jo pabaigos laiko momentą, ciklo metu užimtų kanalų sumą, ciklo metu buvusių eilių sumą, ciklo metu atėjusių paraiškų sumą, ciklo trukmę ir maksimalios eilės dydį ciklo metu (failas „regeneracijos ciklai.txt“);

- Į failą „Statistika.txt“ išvedamos tos pačios sistemos charakteristikos, kurios išvedamos ir į ekraną.
- Grafiniuose failuose „Atejimo laikai.bmp“, „Apdorojimo laikai.bmp“ ir „Proceso_eiga.bmp“ išsaugomi grafikai, jeigu vartotojas programos darbo pradžioje pasirinko, kad jam jų reikės.

Proceso eigos grafike vaizduojamas paraiškų sistemoje kiekis (juoda spalva žymimos aptarnaujamos paraiškos, raudona – laukiančios eilėje). Kiekvieną regeneracinio ciklo pradžią žymi raudonas apskritimas ant laiko ašies. Grafikas gali būti pritraukiamas pažymint norimą vietą stačiakampiu (pelės žymekliu vedant iš kaires viršuje į dešinę apačioje). Laikant dešinį pelės mygtuką galima stumdyti grafiką norima kryptimi (aktualu pritraukus vaizdą). Norint atstatyti buvusį vaizdą, priešinga tvarka vykdomi pritraukimo veiksmai.

Kitų dviejų grafikų paskirtį nusako jų pavadinimai: „Atėjimo laikų histograma“ ir „Apdorojimo laikų histograma“. Pabaigus programos darbą, juose išvedamos histogramos, pagal kurias vartotojas gali vizualiai matyti, kiek jo sugeneruoti atsitiktiniai skaičiai atitinka jo poreikius. Pasirinkus per mažą laiko iteracijų skaičių, akivaizdžiai matosi netolydūs grafikai. Prie tų pačių histogramų išvedami atėjimo ir apdorojimo laikų vidurkiai – jų pagalba taip pat galime pasitikrinti, ar programa veikia korektiškai su duotais duomenimis.

Paspaudus mygtuką ”**X**“ arba pasirinkus meniu punktą Failas → Baigti išeinama iš programos.

DISKUSIJA

Magistro darbe yra tiriamos regeneracinio metodo pritaikymo daugiakanalėse aptarnavimo sistemose galimybės. Sukurta programa, modeliuojanti daugiakanalę sistemą ir gaunamus modeliavimo rezultatus apdorojanti pagal regeneracinio metodo principus.

Regeneracinio metodo pagalba gauti vienodai pasiskirstę tarpusavyje nepriklausomi duomenų segmentai. Jiems pritaikius klasikinės teorijos metodus, gauti sistemos efektyvumo kriterijų įverčiai ir jų pasikliautinieji intervalai.

Patikrinus programos darbo korektiškumą su teoriškai gaunamais rezultatais eksponentinio pasiskirstymo atveju gauti rezultatai pateikiami 2.1.1 lentelėje. Matome, kad modeliavimo rezultatai su eksponentiniais laikų pasiskirstymais yra gana artimi teoriniams rezultatams, be to akivaizdu, kad didėjant modeliavimo iteracijų kiekiui, tikslumas didėja ir sumodeliuotos reikšmės artėja link apskaičiuotų teoriškai. Iš gautų atėjimo ir apdorojimo laikų histogramų galime matyti, kad

generuojami atsitiktiniai dydžiai tikrai pasiskirstę pagal eksponentinius skirstinius. Iš įvykdyto bandymo galime spręsti, kad metodas veikia korektiškai.

Dėl regeneracinio metodo savybės žinomą kiekį priklausomų dydžių versti atsitiktiniu skaičiumi, tačiau nepriklausomų dydžių, mes galime naudoti bet kokį pasiskirstymą ir klasikinės statistikos pagalba skaičiuoti ne tik vidurkių įverčius, bet ir stebimų dydžių pasikliautinuosius intervalus modeliavimo pagalba.

2.2 skyriuje, patikrinus programos darbo metu apskaičiuojamus pasikliautinuosius intervalus empiriniu metodu, gauti rezultatai pateikiami 2.2.1 lentelėje (eksponentinis atvejis). Gauti rezultatai leidžia teigti, kad didinant modeliavimo laiką pasikliautinieji intervalai darosi tikslesni – kartojant eksperimentą, gaunami rezultatai su artima 90% tikimybe pakliūna į buvusių eksperimentų pasikliautinuosius intervalus. Nuokrypiai nuo 90% pasikliautinojo intervalo, „gadinantys“ bendrą statistiką, nėra dideli, jie gali atsirasti dėl pseudo atsitiktinių skaičių generavimo apvalinimų (nėra idealiai tolygus pasiskirstymas).

Taip pat buvo atliktas bandymas, parodęs, kad regeneraciniai momentai negali būti pasirenkami su sąlyga: „sistemoje yra bent vienas laisvas kanalas“. Programai nurodžius veikti su tokia regeneracinio momento sąlyga pasikliautinųjų intervalų adekvatumas pradėjo priklausyti ne tik nuo to, kiek laiko vienetų modeliuojama sistema, bet ir nuo to, kokie yra atėjimo ir apdorojimo laikų intensyvumai. Tai galima nesunkiai paaiškinti tuo, kad nurodžius regeneracinę sąlygą, kai keletas kanalų yra vis dar užimti – pažeidžiama regeneracinio metodo taikomumo sąlyga. Atskiri regeneraciniai ciklai turi skirtingas pradines sąlygas, nes užimtuose kanaluose aptarnaujamos paraiškos gali būti skirtingai aptarnautos – taigi vienų aptarnavimas baigsis tuoj pat, o kitos ką tik pradėtos aptarnauti. Taigi, jeigu laikų pasiskirstymai yra tokie, kad vienos paraiškos aptarnavimas trunka sąlyginai ilgai – neteisingai parinkta regeneracinio taško sąlyga gali smarkiai padidinti gaunamų rezultatų dispersiją.

Remiantis tuo regeneracinis momentas turi būti pasirinktas su sąlyga: „visi aptarnavimo kanalai yra laisvi“.

Bandant įgyvendinti ekstremumų modeliavimą (apdorojimo laikų maksimumų ieškojimas) buvo pastebėtas pasirinkto programinio paketo trūkumas, generuojant atsitiktinius dydžius, dėl kurio maksimumų modeliavimas yra nekorektiškas. Problema iškilo, nes atsitiktinių skaičių generatorius C++ terpėje generuoja RAND_MAX=32767 konstantos rėžiuose (0; RAND_MAX). Taigi, artimiausias nuliui skaičius, pasiskirstęs pagal tolygųjį pasiskirstymą, yra 1/32767. Generuojant eksponentinius (ir Veibulo) dydžius, naudojama formulė $T = \frac{-\ln(x)}{\lambda}$, kur x – tolygiai intervale (0;1) pasiskirstęs dydis. Kadangi artimiausias nuliui dydis pagal tolygųjį pasiskirstymą yra

1/32767, tai maksimalus eksponentinis dydis, kurį galima gauti yra $T = \frac{-\ln(\frac{1}{32767})}{\lambda} \approx \frac{10.3971}{\lambda}$, kur λ - pasiskirstymo intensyvumo parametras. Generuojant pakankamą kiekį laiko momentų, gaunama ta pati maksimali reikšmė, kuri, didinant modeliavimo laiką, nebedidėja. Teoriškai, atsitiktinių dydžių maksimumas turėtų didėti bandymų skaičiui didėjant. Taigi gaunamos „lubos“ padaro tokį eksperimentinį modeliavimą nekorektišku. Dėl šios priežasties buvo atsisakyta maksimumų modeliavimo dalies programoje.

IŠVADOS

- Regeneracinio metodo taikymas įgalina smarkiai koreliuotus vienodai pasiskirsčiusius duomenis, paversti nepriklausomų duomenų grupėmis, kurioms galima taikyti klasikinius statistikos metodus.
- Šio metodo pagalba išsprendžiamos modeliavimo pradžios ir atskaitos taško parinkimo problemos (nelieka pradinių sąlygų nulemtų poslinkių ir nėra problemų dėl stabilizacijos periodo).
- Modelis naudojantis regeneracinį metodą gali dirbti su įvairiais duomenų atėjimo laikų ir aptarnavimo laikų pasiskirstymais (bent kokį pasiskirstymą regeneracinis procesas verčia Markovo procesu, kai praeitis neturi įtakos ateičiai).
- Regeneraciniu principu modeliavimo rezultatus apdorojanti programa, pakankamai gerai apskaičiuoja pasikliautinuosius intervalus (Nuokrypis nuo modeliuoto 90% pasikliautinojo intervalo ~2-3%).
- Regeneracinis metodas verčia duomenų grupių skaičių atsitiktiniu dydžiu, tai apriboja tam tikras galimybes (pavyzdžiui, neįmanoma numatyti, koks iteracijų skaičius duos tam tikrą regeneracinių ciklų kiekį).

REKOMENDACIJOS

Darbas galėtų būti patobulintas, pridėjus ekstremalių reikšmių charakteristikų modeliavimus (Ribiniai ekstremumų skirstiniai pateikiami teorijos skyriuje – 1.3). Tam reikia išspręsti pseudo atsitiktinių dydžių generavimo problemą. Tikslių rekomendacijų šios problemos sprendimui neturiu. Kito kompiliatoriaus naudojimas, su didesne RAND_MAX reikšme, neišspręstų problemos, o tiesiog ją atitolintų – „lubos“ vis tiek liks.

ŠALTINIAI IR LITERATŪRA

1. Aksomaitis A. Tikimybių teorija ir statistika. Technologija, Kaunas (2002).
2. Galambos J. The Asymptotic Theory of Extreme Order Statistics, John Willey and Sons. New York, 1984.
3. Крейн М., Лемуан О. Введение в регенеративный метод анализа моделей. – Москва: Наука, (1982).
4. Pranevičius H., Pranevičienė I. Masinio aptarnavimo teorijos elementai. Raidės. Kaunas (1980).
5. Nuoroda internete: http://en.wikipedia.org/wiki/Exponential_distribution
6. Nuoroda internete: http://www.wikidoc.org/index.php/Weibull_distribution
7. Nuoroda internete: http://en.wikipedia.org/wiki/Scale_parameter

PRIEDAS Nr.1 10^6 LAIKO VIENETŲ MODELIAVIMO REZULTATAI

(EKSPONENTINIS PASISKIRSTYMAS)

minpasikl	Kanalu uzimtumas	maxpasikl	minpasikl	Eiles ilgis	maxpasikl	minpasikl	Laukimas eileje	maxpasikl	Max eile	Regeneraciniai ciklai
3.9942	4.0025	4.0108	2.1282	2.1757	2.2232	3.5503	3.627	3.7036	46	7665
3.9852	3.9938	4.0024	2.1493	2.2042	2.2591	3.5889	3.6779	3.7669	39	8064
4.001	4.0095	4.0179	2.2031	2.2612	2.3193	3.6688	3.7628	3.8569	48	7893
3.9916	4.0001	4.0085	2.1329	2.1831	2.2333	3.5616	3.6428	3.724	43	7777
3.9976	4.0061	4.0146	2.1807	2.2329	2.2852	3.6334	3.7178	3.8022	40	7664
3.986	3.9945	4.0029	2.1184	2.1665	2.2146	3.5385	3.6162	3.6939	39	7929
3.9937	4.0023	4.0108	2.2063	2.2585	2.3106	3.6787	3.7628	3.8468	41	7754
3.993	4.0015	4.0101	2.2332	2.2864	2.3397	3.725	3.8109	3.8969	41	7706
3.9944	4.0029	4.0114	2.1603	2.2126	2.2649	3.6001	3.6844	3.7687	43	7959
3.9859	3.9945	4.003	2.1482	2.1995	2.2508	3.5864	3.6693	3.7521	40	7889
3.9912	3.9996	4.0079	2.1661	2.2153	2.2645	3.6126	3.692	3.7714	48	7842
3.992	4.0006	4.0091	2.2057	2.2606	2.3155	3.6822	3.7711	3.8599	38	7666
3.993	4.0016	4.0102	2.1798	2.2305	2.2813	3.6326	3.7145	3.7964	43	7808
3.9948	4.0033	4.0118	2.1739	2.2283	2.2826	3.6219	3.7097	3.7975	55	7778
3.986	3.9945	4.0029	2.124	2.1744	2.2249	3.5468	3.6284	3.71	49	7846
3.9909	3.9993	4.0078	2.1771	2.2302	2.2833	3.6344	3.7204	3.8064	39	7849
4.0004	4.0088	4.0173	2.2062	2.2591	2.312	3.6776	3.7629	3.8482	37	7653
3.9891	3.9974	4.0057	2.1161	2.1658	2.2154	3.5273	3.6075	3.6877	41	7798
3.9884	3.9968	4.0052	2.1711	2.2213	2.2715	3.6219	3.7028	3.7838	43	7810
3.9821	3.9908	3.9994	2.1509	2.205	2.2592	3.5924	3.6801	3.7679	52	7700
3.9959	4.0045	4.0132	2.2234	2.2794	2.3355	3.7075	3.7981	3.8888	45	7698
3.9847	3.9931	4.0014	2.1074	2.1572	2.2069	3.5178	3.5984	3.679	49	7716
3.9911	3.9996	4.008	2.1601	2.213	2.2659	3.6066	3.6923	3.7779	40	7853
3.9911	3.9996	4.0081	2.1548	2.2039	2.253	3.5987	3.678	3.7573	41	7790
3.9974	4.0059	4.0144	2.1797	2.2315	2.2832	3.6354	3.719	3.8025	40	7915
3.9763	3.9848	3.9934	2.1072	2.157	2.2068	3.52	3.6005	3.681	42	7682
3.9922	4.0009	4.0096	2.1964	2.252	2.3076	3.6628	3.7528	3.8428	46	7809
4.0029	4.0114	4.0199	2.228	2.2825	2.3369	3.7149	3.803	3.891	39	7842
3.9839	3.9922	4.0006	2.1041	2.1527	2.2012	3.5115	3.5899	3.6683	37	7631
3.997	4.0055	4.014	2.1677	2.2213	2.2749	3.6103	3.6969	3.7834	40	7764
3.9791	3.9873	3.9955	2.0748	2.1222	2.1695	3.4683	3.545	3.6216	44	8063
3.9804	3.9888	3.9971	2.0963	2.1444	2.1924	3.5	3.5777	3.6554	39	7522

3.9847	3.9932	4.0017	2.1125	2.1644	2.2163	3.5269	3.611	3.695	47	7652
3.989	3.9975	4.006	2.1606	2.2119	2.2632	3.6037	3.6865	3.7692	40	7820
3.9947	4.003	4.0114	2.1723	2.2225	2.2726	3.6219	3.703	3.784	40	7846
3.9892	3.9979	4.0065	2.1807	2.2341	2.2875	3.6377	3.7239	3.8101	51	7652
3.9842	3.9928	4.0013	2.1498	2.2148	2.2797	3.5863	3.6923	3.7983	64	7731
3.994	4.0025	4.0111	2.1916	2.2459	2.3002	3.6558	3.7436	3.8313	41	7678
3.9967	4.0052	4.0136	2.1862	2.2408	2.2954	3.6423	3.7307	3.8191	40	7880
3.9916	4.0001	4.0086	2.1791	2.2306	2.2821	3.6386	3.7219	3.8052	38	7826
3.9956	4.0041	4.0126	2.2053	2.2585	2.3116	3.6759	3.7617	3.8475	46	7854
3.9887	3.9972	4.0057	2.1541	2.2078	2.2614	3.5935	3.6805	3.7675	45	7805
3.9909	3.9994	4.0079	2.1539	2.2035	2.2531	3.591	3.671	3.7509	45	7665
3.9958	4.0042	4.0126	2.1846	2.236	2.2875	3.6392	3.7222	3.8051	44	7981
4.0035	4.0119	4.0203	2.2154	2.2671	2.3188	3.6891	3.7726	3.8561	52	7885
3.9942	4.0026	4.0111	2.16	2.2119	2.2638	3.5982	3.682	3.7658	40	7955
3.9962	4.0048	4.0135	2.1656	2.2215	2.2774	3.61	3.7006	3.7912	62	7690
3.9885	3.997	4.0056	2.1488	2.2043	2.2599	3.5876	3.6776	3.7676	47	7849
3.986	3.9945	4.0029	2.1256	2.1742	2.2228	3.5511	3.6298	3.7085	49	7838
3.9884	3.9969	4.0054	2.1496	2.2013	2.253	3.5887	3.6723	3.7559	39	7719
3.9835	3.992	4.0006	2.1176	2.1669	2.2163	3.5359	3.6157	3.6954	42	7586
3.9839	3.9926	4.0012	2.1745	2.2254	2.2763	3.6296	3.7117	3.7938	44	7825
4.0057	4.0141	4.0226	2.2347	2.2923	2.3499	3.7226	3.8158	3.909	49	7646
3.9972	4.0057	4.0141	2.1755	2.2264	2.2774	3.6213	3.7034	3.7854	41	7867
3.9866	3.9953	4.0039	2.1413	2.1923	2.2432	3.5736	3.6559	3.7383	49	7800
3.9914	3.9999	4.0083	2.2038	2.2604	2.3171	3.6789	3.7707	3.8626	45	7687
3.9949	4.0034	4.0119	2.1932	2.2452	2.2972	3.6586	3.7425	3.8265	47	7948
3.9868	3.9954	4.004	2.1487	2.1983	2.2479	3.5872	3.6671	3.747	50	8125
3.9935	4.002	4.0106	2.1814	2.2329	2.2844	3.6375	3.7208	3.804	45	7620
3.9935	4.002	4.0106	2.1892	2.2434	2.2976	3.6523	3.7399	3.8275	40	7870
3.9985	4.0069	4.0154	2.2006	2.2518	2.303	3.6696	3.7521	3.8347	52	7701
3.9872	3.9958	4.0043	2.1637	2.2178	2.272	3.6086	3.696	3.7834	45	7797
3.9936	4.002	4.0103	2.1368	2.1873	2.2377	3.561	3.6424	3.7239	45	7734
4.0002	4.0087	4.0173	2.2007	2.2533	2.3059	3.6667	3.7515	3.8363	39	7834
3.9836	3.9919	4.0003	2.1514	2.2032	2.255	3.5877	3.6714	3.7552	46	7977
3.9984	4.007	4.0156	2.2164	2.2674	2.3183	3.694	3.776	3.8581	56	7770
3.9851	3.9934	4.0016	2.1261	2.1738	2.2215	3.5567	3.6339	3.7112	51	7881
3.9873	3.9958	4.0044	2.1474	2.2013	2.2552	3.5824	3.6696	3.7567	44	7735
3.9913	3.9998	4.0083	2.173	2.2252	2.2774	3.623	3.7073	3.7915	46	7720
3.9888	3.9974	4.0059	2.1817	2.2366	2.2916	3.6432	3.7321	3.821	46	7819
3.9956	4.004	4.0123	2.1918	2.2416	2.2915	3.6539	3.7343	3.8146	46	7745

3.9931	4.0017	4.0102	2.163	2.2136	2.2643	3.609	3.6908	3.7726	41	7749
3.9828	3.9913	3.9997	2.0957	2.1421	2.1885	3.499	3.5736	3.6482	42	7796
3.9939	4.0024	4.0109	2.1934	2.2431	2.2929	3.661	3.7412	3.8215	42	7808
3.9934	4.0019	4.0104	2.1773	2.2293	2.2813	3.6313	3.7151	3.799	45	7937
3.9913	3.9997	4.0082	2.1296	2.1778	2.2259	3.5515	3.6291	3.7067	49	7858
3.9883	3.9967	4.0051	2.1696	2.2223	2.2749	3.6199	3.705	3.7902	47	7807
3.9929	4.0014	4.0099	2.1457	2.1951	2.2445	3.578	3.6577	3.7375	46	7736
3.9948	4.0034	4.0119	2.2023	2.2582	2.3142	3.669	3.7596	3.8501	42	7618
4.0007	4.0093	4.018	2.2146	2.2695	2.3245	3.6926	3.7813	3.87	37	7723
3.9854	3.9939	4.0024	2.1267	2.1756	2.2245	3.5486	3.6277	3.7067	48	7818
3.9947	4.0032	4.0117	2.146	2.1956	2.2452	3.5784	3.6584	3.7384	45	7662
3.9977	4.0061	4.0146	2.2296	2.2855	2.3415	3.7191	3.8095	3.8998	47	7747
3.9945	4.003	4.0115	2.2063	2.2619	2.3174	3.6803	3.7703	3.8603	53	7675
3.9867	3.9953	4.004	2.1594	2.2111	2.2628	3.6019	3.6854	3.7689	46	7815
3.9776	3.986	3.9944	2.0806	2.1294	2.1782	3.4737	3.5526	3.6314	43	7897
3.9921	4.0007	4.0093	2.1541	2.2051	2.2561	3.5933	3.6755	3.7577	37	7792
3.9856	3.994	4.0024	2.1558	2.2063	2.2568	3.5976	3.6792	3.7608	50	7570
3.9955	4.0038	4.0121	2.1944	2.2465	2.2986	3.6563	3.7403	3.8243	38	7958
3.9874	3.9959	4.0044	2.1735	2.2267	2.2799	3.6279	3.714	3.8001	42	7866
3.9902	3.9988	4.0073	2.1876	2.24	2.2923	3.648	3.7326	3.8171	46	7970
3.9961	4.0046	4.0132	2.1862	2.2418	2.2975	3.6445	3.7345	3.8245	50	7817
3.9951	4.0034	4.0118	2.1694	2.224	2.2785	3.6179	3.7064	3.7948	48	7725
3.9858	3.9941	4.0025	2.1509	2.2013	2.2517	3.5906	3.6721	3.7536	55	8002
3.9902	3.9988	4.0073	2.1583	2.2118	2.2653	3.6007	3.6873	3.7739	60	7868
3.9869	3.9954	4.004	2.1604	2.208	2.2556	3.6104	3.687	3.7637	43	7930
3.9946	4.0032	4.0117	2.211	2.268	2.325	3.6829	3.7752	3.8675	41	7675
3.9865	3.9951	4.0038	2.1627	2.2134	2.2641	3.6098	3.6916	3.7734	45	7655
3.9865	3.9949	4.0033	2.1567	2.2115	2.2663	3.6042	3.6932	3.7822	54	7779
3.9907	3.9992	4.0077	2.1607	2.2121	2.2635	3.6	3.6828	3.7657	44	7503

Patekimas i pasikliautinuosius intervalus:
Kanalai: 84.99%
Eiles ilgis: 84.04%
Laukimo laikas: 84.08%

PRIEDAS Nr.2 $2x10^6$ LAIKO VIENETŲ MODELIAVIMO REZULTATAI

(EKSPONENTINIS PASISKIRSTYMAS)

minpasikl	Kanalu uzimtumas	maxpasikl	minpasikl	Eiles ilgis	maxpasikl	minpasikl	Laukimas eileje	maxpasikl	Max eile	Regeneraciniai ciklai
3.9922	4.004	4.0158	2.147	2.2171	2.2872	3.5832	3.6965	3.8097	47	15290
3.9843	3.9964	4.0084	2.1722	2.2454	2.3187	3.6224	3.7403	3.8582	60	15488
3.9885	4.0002	4.0119	2.123	2.1933	2.2637	3.5425	3.6564	3.7702	64	15480
3.996	4.008	4.0199	2.1998	2.2727	2.3455	3.6631	3.7805	3.8978	43	15440
3.9942	4.0062	4.0182	2.1549	2.2303	2.3056	3.5944	3.7166	3.8387	49	15296
3.989	4.0012	4.0135	2.1755	2.2512	2.3269	3.6295	3.7517	3.8738	39	15698
3.9878	3.9995	4.0112	2.1191	2.1831	2.247	3.5303	3.6333	3.7362	48	15478
3.9904	4.0026	4.0148	2.1858	2.2581	2.3304	3.6534	3.77	3.8865	46	15707
3.9739	3.9858	3.9978	2.113	2.1823	2.2517	3.537	3.6492	3.7613	46	15505
3.9831	3.9951	4.0072	2.1319	2.2029	2.274	3.5589	3.6738	3.7886	50	15647
3.9893	4.0016	4.0138	2.1642	2.2398	2.3153	3.6095	3.7318	3.8541	57	15780
3.987	3.9986	4.0103	2.1022	2.1716	2.241	3.5095	3.6218	3.7341	62	15682
3.9958	4.0078	4.0199	2.2161	2.2925	2.3688	3.6902	3.8137	3.9371	40	15532
3.9892	4.0012	4.0131	2.2152	2.2894	2.3636	3.6982	3.8181	3.938	43	15422
3.9922	4.004	4.0157	2.1575	2.2296	2.3016	3.5898	3.7058	3.8218	57	15790
3.9836	3.9957	4.0078	2.1236	2.1925	2.2613	3.5466	3.6575	3.7683	43	15637
3.9941	4.0061	4.018	2.1283	2.1979	2.2676	3.5464	3.6585	3.7707	54	15356
3.9816	3.9933	4.0051	2.1075	2.1694	2.2314	3.5209	3.6209	3.7208	47	15799
3.9862	3.9981	4.0099	2.1394	2.2125	2.2856	3.5761	3.6944	3.8127	51	15737
3.9876	3.9995	4.0115	2.1064	2.1779	2.2495	3.5198	3.6357	3.7516	46	15700
3.9988	4.0105	4.0221	2.1575	2.229	2.3005	3.5917	3.7069	3.8221	46	15694
4.004	4.0163	4.0286	2.2034	2.2782	2.353	3.6757	3.7966	3.9176	51	15664
3.9914	4.0034	4.0154	2.1859	2.2595	2.3331	3.6505	3.7692	3.888	50	15577
3.9869	3.9989	4.0109	2.1115	2.1794	2.2472	3.5222	3.6313	3.7403	39	15434
3.9868	3.9988	4.0107	2.1515	2.2204	2.2894	3.5903	3.7018	3.8133	45	15519
3.9977	4.0094	4.0212	2.2102	2.2807	2.3513	3.6772	3.7907	3.9043	53	15750
3.9944	4.0066	4.0187	2.1454	2.2208	2.2962	3.5772	3.6989	3.8206	50	15628
3.9808	3.9928	4.0048	2.081	2.1469	2.2128	3.4814	3.5879	3.6944	46	15597
3.9928	4.005	4.0171	2.1531	2.2271	2.3011	3.5912	3.7105	3.8299	56	15962
3.986	3.9981	4.0101	2.1001	2.1669	2.2338	3.5009	3.6084	3.7158	54	15315

3.9745	3.9864	3.9983	2.0301	2.0961	2.1621	3.3949	3.5015	3.608	53	15690
3.9995	4.0113	4.0231	2.1842	2.253	2.3218	3.6401	3.7509	3.8618	53	15639
3.9861	3.9981	4.0102	2.0953	2.1589	2.2225	3.4931	3.5951	3.6971	47	15644
3.9751	3.987	3.9988	2.0965	2.1675	2.2384	3.5011	3.6156	3.7301	56	15380
3.9842	3.9963	4.0084	2.1104	2.1726	2.2348	3.5246	3.6247	3.7248	45	15395
3.9936	4.0053	4.017	2.1674	2.2387	2.31	3.6161	3.7312	3.8463	48	15962
3.9955	4.0077	4.0199	2.1832	2.272	2.3608	3.6357	3.7798	3.9239	83	15920
3.987	3.9994	4.0117	2.1951	2.2719	2.3487	3.658	3.7818	3.9056	45	15481
3.9788	3.9908	4.0028	2.1029	2.1753	2.2476	3.5137	3.6305	3.7472	54	15489
3.9855	3.9977	4.0099	2.1177	2.1848	2.252	3.5346	3.6427	3.7507	44	15537
3.9839	3.9958	4.0078	2.1326	2.2071	2.2815	3.5644	3.6852	3.806	46	15409
3.9912	4.0029	4.0146	2.1697	2.2414	2.3132	3.6223	3.7382	3.8541	59	15886
3.9885	4.0005	4.0125	2.1284	2.1976	2.2667	3.5536	3.6649	3.7763	40	15252
3.968	3.9798	3.9917	2.0479	2.117	2.1861	3.4245	3.5364	3.6482	53	15434
3.9814	3.9931	4.0048	2.094	2.1608	2.2276	3.495	3.603	3.711	65	15555
3.9813	3.9935	4.0058	2.1131	2.1874	2.2617	3.5322	3.6526	3.773	45	15437
3.9917	4.0037	4.0157	2.1583	2.2486	2.3389	3.604	3.7512	3.8984	60	15474
3.9804	3.9923	4.0041	2.0806	2.1491	2.2176	3.4738	3.5845	3.6951	48	15743
3.9925	4.0044	4.0164	2.1462	2.2144	2.2826	3.5803	3.6905	3.8007	48	15702
3.994	4.0059	4.0178	2.1755	2.2466	2.3176	3.6258	3.7403	3.8548	50	15517
3.9948	4.0068	4.0188	2.1805	2.2503	2.3201	3.634	3.7461	3.8581	42	15868
3.9843	3.996	4.0078	2.1018	2.176	2.2503	3.5097	3.6301	3.7505	45	15871
3.9834	3.9956	4.0077	2.1418	2.2219	2.302	3.5732	3.7031	3.833	60	15165
3.9889	4.0004	4.0119	2.1392	2.211	2.2829	3.5705	3.6868	3.8031	43	15527
3.9911	4.0032	4.0153	2.1542	2.2301	2.3059	3.591	3.7135	3.836	45	15920
3.9948	4.007	4.0191	2.2364	2.3146	2.3929	3.7313	3.8578	3.9844	57	15606
3.9922	4.0042	4.0162	2.1736	2.2516	2.3295	3.6267	3.7528	3.8789	43	15310
3.9789	3.9912	4.0035	2.1277	2.2031	2.2784	3.5494	3.6714	3.7933	49	15633
3.9937	4.0057	4.0177	2.1565	2.2371	2.3178	3.5978	3.7283	3.8589	49	15681
3.994	4.0061	4.0183	2.1485	2.2197	2.2908	3.5821	3.6967	3.8112	51	15503
3.9953	4.0077	4.02	2.1561	2.2399	2.3236	3.5913	3.727	3.8627	46	15639
3.9893	4.0012	4.0131	2.144	2.219	2.2941	3.5783	3.6999	3.8214	49	15734
3.9992	4.0114	4.0237	2.222	2.2961	2.3701	3.6956	3.8146	3.9336	40	15767
3.9884	4.0002	4.012	2.1319	2.2009	2.2699	3.5589	3.6705	3.7821	49	15622
3.9781	3.9903	4.0024	2.1125	2.1825	2.2525	3.5298	3.6426	3.7553	48	15874
3.9912	4.0033	4.0154	2.1734	2.2515	2.3296	3.6182	3.7437	3.8693	47	15306
3.9924	4.0047	4.0169	2.1682	2.2457	2.3231	3.6115	3.7365	3.8616	48	15690
3.9969	4.0088	4.0208	2.1854	2.2614	2.3373	3.6439	3.7667	3.8895	45	15369
3.9868	3.9986	4.0104	2.106	2.1699	2.2338	3.5153	3.618	3.7208	46	15516

3.9936	4.0057	4.0179	2.1915	2.2716	2.3516	3.6551	3.7846	3.9142	49	15513
3.9931	4.0051	4.0172	2.1609	2.2323	2.3038	3.6099	3.7253	3.8406	47	15388
3.9876	3.9995	4.0114	2.1193	2.1908	2.2623	3.5348	3.6503	3.7658	39	15745
3.9903	4.0024	4.0145	2.1472	2.2241	2.3011	3.5847	3.7092	3.8337	45	15774
3.9889	4.0008	4.0127	2.1233	2.1906	2.2579	3.5449	3.6534	3.7619	54	15506
3.9914	4.004	4.0166	2.2731	2.3605	2.448	3.7928	3.9346	4.0763	43	15575
3.9821	3.9943	4.0065	2.1068	2.1821	2.2574	3.5173	3.6392	3.7612	46	15497
3.9843	3.9963	4.0083	2.1658	2.2427	2.3197	3.6197	3.7442	3.8687	43	15887
3.991	4.0031	4.0152	2.1495	2.2262	2.3029	3.5834	3.7072	3.8311	43	15598
3.9941	4.0062	4.0183	2.1427	2.2098	2.2769	3.574	3.6819	3.7898	59	15473
3.9882	3.9998	4.0115	2.1193	2.1893	2.2592	3.5363	3.6494	3.7624	46	15027
3.9907	4.0027	4.0146	2.1428	2.2159	2.289	3.5757	3.6941	3.8126	43	15645
3.9986	4.0106	4.0226	2.16	2.2308	2.3015	3.5974	3.7115	3.8257	43	15416
3.9867	3.9992	4.0117	2.1307	2.2076	2.2844	3.5561	3.6803	3.8046	43	15507
3.9973	4.0093	4.0213	2.1905	2.2743	2.3581	3.6497	3.7857	3.9218	47	15664
3.9836	3.9957	4.0078	2.1472	2.2254	2.3037	3.5857	3.7125	3.8392	46	15643
3.9894	4.0015	4.0137	2.1666	2.2448	2.3231	3.6081	3.7345	3.8608	44	15919
3.9828	3.9946	4.0065	2.087	2.1519	2.2168	3.4862	3.5909	3.6955	47	15473
3.9956	4.0076	4.0197	2.1837	2.2637	2.3437	3.6453	3.7751	3.905	43	15743
3.9812	3.9932	4.0052	2.0607	2.1265	2.1924	3.4446	3.5508	3.6569	54	15283
3.9901	4.0024	4.0146	2.1813	2.2622	2.3431	3.6414	3.7726	3.9037	54	15785
3.9797	3.9914	4.0032	2.0762	2.1465	2.2168	3.4655	3.579	3.6924	54	15707
3.988	3.9998	4.0115	2.1229	2.1925	2.2622	3.5473	3.6602	3.7731	65	15317
3.9919	4.0039	4.016	2.1445	2.2213	2.298	3.5765	3.7007	3.8249	47	15472
3.9786	3.9904	4.0022	2.1336	2.2063	2.2791	3.5584	3.6761	3.7939	53	15736
3.9817	3.9938	4.0058	2.1747	2.2604	2.3462	3.6335	3.7732	3.9129	51	15593
3.9862	3.9983	4.0105	2.15	2.2224	2.2948	3.585	3.7017	3.8184	42	15635
3.9929	4.0051	4.0173	2.163	2.2342	2.3055	3.6127	3.7278	3.843	65	15451
4.0015	4.0133	4.025	2.2246	2.2964	2.3683	3.7035	3.8195	3.9355	43	15691
3.987	3.999	4.0109	2.1107	2.1798	2.2489	3.5229	3.6345	3.7462	58	15444
3.9919	4.0037	4.0156	2.1715	2.2487	2.3258	3.6251	3.7502	3.8752	46	15545

Patekimas i pasikliautinuosius intervalus:
Kanalai: 90.64%
Eiles ilgis: 86.94%
Laukimo laikas: 86.71%

PRIEDAS Nr.3 $5x10^6$ LAIKO VIENETŲ MODELIAVIMO REZULTATAI

(EKSPONENTINIS PASISKIRSTYMAS)

minpasikl	Kanalu uzimtumas	maxpasikl	minpasikl	Eiles ilgis	maxpasikl	minpasikl	Laukimas eileje	maxpasikl	Max eile	Regeneraciniai ciklai
3.9923	3.9977	4.0031	2.1677	2.199	2.2302	3.62	3.6705	3.7209	42	38947
3.9913	3.9967	4.0021	2.1821	2.2147	2.2473	3.6409	3.6935	3.7462	53	38856
3.9967	4.002	4.0074	2.1863	2.2183	2.2503	3.6457	3.6975	3.7492	52	38812
3.9957	4.0011	4.0065	2.1978	2.2302	2.2625	3.6656	3.7178	3.77	50	38945
3.9958	4.0011	4.0064	2.1925	2.2251	2.2578	3.6568	3.7096	3.7624	75	38948
3.991	3.9963	4.0017	2.1437	2.1758	2.2079	3.5759	3.6277	3.6795	53	38968
3.9869	3.9923	3.9978	2.1565	2.1889	2.2213	3.6005	3.6529	3.7053	50	39569
3.9996	4.005	4.0104	2.1968	2.2299	2.263	3.6638	3.7172	3.7707	48	39123
3.9935	3.9988	4.0041	2.1727	2.2044	2.2362	3.6247	3.676	3.7272	52	39433
3.9938	3.9991	4.0045	2.1871	2.2216	2.2562	3.6473	3.7033	3.7594	57	38496
3.9918	3.9971	4.0025	2.1623	2.1935	2.2247	3.6064	3.6568	3.7072	57	39368
3.9905	3.9958	4.0012	2.1668	2.1991	2.2314	3.6162	3.6685	3.7208	49	38924
3.9945	3.9998	4.0052	2.1684	2.2	2.2317	3.6155	3.6666	3.7177	52	38835
3.992	3.9973	4.0026	2.1703	2.2023	2.2343	3.6208	3.6725	3.7242	48	39210
3.9949	4.0003	4.0057	2.2201	2.2537	2.2874	3.7017	3.7562	3.8106	50	39105
3.9958	4.0012	4.0066	2.1861	2.2189	2.2517	3.642	3.6949	3.7477	48	38775
3.9979	4.0032	4.0085	2.1967	2.2304	2.2641	3.6609	3.7154	3.7699	52	38609
3.9936	3.999	4.0044	2.1497	2.184	2.2183	3.5851	3.6407	3.6963	48	39250
3.9898	3.9951	4.0005	2.1547	2.1861	2.2176	3.595	3.6459	3.6967	54	39129
3.9961	4.0014	4.0068	2.1771	2.2098	2.2426	3.6321	3.6851	3.7381	54	38591

Patekimas i pasikliautinuosius intervalus:

Kanalai: 90.50%

Eiles ilgis: 88.25%

Laukimo laikas: 88.50%

PRIEDAS Nr.4 10^6 LAIKO VIENETŲ MODELIAVIMO REZULTATAI

(TOLYGUSIS PASISKIRSTYMAS)

	Kanalu			Eiles			Laukimas				
minpasikl	uzimtumas	maxpasikl	minpasikl	ilgis	maxpasikl	minpasikl	eileje	maxpasikl	Max eile	Regeneraciniai ciklai
3.9938	4.0009	4.0079	0.6243	0.639	0.6538	1.041	1.0649	1.0889	18	2739
3.9896	3.9965	4.0034	0.6072	0.6203	0.6334	1.0131	1.0343	1.0555	16	2654
3.989	3.996	4.0029	0.6122	0.6258	0.6395	1.0217	1.0438	1.0658	19	2713
3.9948	4.0016	4.0084	0.6177	0.6306	0.6435	1.03	1.0507	1.0715	18	2736
3.9945	4.0016	4.0088	0.6202	0.6349	0.6495	1.0345	1.0582	1.082	20	2709
3.9977	4.0046	4.0114	0.6233	0.6376	0.652	1.0396	1.0628	1.086	20	2693
3.9948	4.0015	4.0081	0.6267	0.64	0.6534	1.0442	1.0658	1.0873	16	2688
3.9983	4.0051	4.0118	0.6205	0.6347	0.6489	1.0337	1.0567	1.0798	23	2724
3.9936	4.0005	4.0075	0.6279	0.6425	0.6571	1.0481	1.0716	1.0952	21	2838
3.9908	3.9977	4.0047	0.6299	0.6453	0.6606	1.0508	1.0757	1.1006	18	2855
3.9939	4.0009	4.0079	0.6079	0.6209	0.6339	1.0132	1.034	1.0549	17	2714
3.9943	4.0011	4.008	0.6104	0.6244	0.6385	1.0185	1.0412	1.064	24	2698
3.9924	3.9991	4.0057	0.6106	0.6241	0.6376	1.0182	1.0401	1.062	16	2684
3.9959	4.0028	4.0097	0.6198	0.6347	0.6496	1.0338	1.0579	1.082	18	2757
3.9944	4.0014	4.0084	0.6186	0.6328	0.6471	1.0314	1.0545	1.0776	18	2789
3.9943	4.0013	4.0083	0.6162	0.63	0.6438	1.0269	1.0493	1.0716	16	2814
3.9935	4.0004	4.0073	0.616	0.6288	0.6415	1.028	1.0486	1.0691	17	2698
3.9929	4	4.007	0.6096	0.623	0.6364	1.017	1.0386	1.0602	16	2755
3.9913	3.9981	4.005	0.6095	0.623	0.6366	1.0164	1.0383	1.0602	23	2761
3.9863	3.9931	3.9999	0.604	0.6182	0.6324	1.0088	1.0319	1.0549	17	2733
3.9968	4.004	4.0112	0.621	0.636	0.651	1.0356	1.0598	1.0841	18	2772
3.9963	4.0032	4.0101	0.6124	0.6257	0.639	1.0215	1.0431	1.0646	16	2717
3.9966	4.0032	4.0099	0.6233	0.6372	0.6511	1.0398	1.0624	1.085	17	2682
3.9982	4.0049	4.0115	0.6266	0.6408	0.655	1.0442	1.0671	1.09	16	2686
3.9953	4.0022	4.009	0.6146	0.6282	0.6419	1.0252	1.0473	1.0694	17	2754
3.9993	4.0059	4.0124	0.6294	0.6436	0.6578	1.049	1.072	1.095	18	2656
3.9886	3.9956	4.0026	0.6174	0.6318	0.6462	1.0298	1.0531	1.0764	19	2794
3.9888	3.996	4.0032	0.6138	0.6279	0.6419	1.0249	1.0477	1.0704	16	2882
3.9977	4.0044	4.0112	0.619	0.6319	0.6448	1.0317	1.0525	1.0733	15	2728
3.9894	3.9964	4.0034	0.6134	0.6273	0.6412	1.023	1.0455	1.068	15	2861
3.9903	3.9972	4.004	0.6223	0.6364	0.6505	1.0383	1.0611	1.084	16	2836

3.9864	3.9933	4.0002	0.6092	0.623	0.6367	1.0179	1.0401	1.0624	16	2745
3.9936	4.0003	4.0071	0.6139	0.6269	0.64	1.0234	1.0445	1.0656	18	2794
3.9928	3.9996	4.0065	0.6288	0.6435	0.6581	1.0485	1.0721	1.0958	17	2786
3.9969	4.0037	4.0104	0.6138	0.6275	0.6411	1.0235	1.0456	1.0676	17	2735
3.9869	3.9941	4.0012	0.6096	0.6234	0.6372	1.0175	1.0397	1.062	18	2780
3.9867	3.994	4.0013	0.6098	0.6236	0.6375	1.0175	1.0399	1.0624	16	2737
3.9937	4.0007	4.0078	0.6046	0.6179	0.6312	1.0087	1.0301	1.0516	17	2674
3.9984	4.0053	4.0123	0.6229	0.6377	0.6525	1.0377	1.0617	1.0857	17	2611
3.9875	3.9944	4.0012	0.6092	0.6229	0.6367	1.0152	1.0374	1.0595	18	2779
3.998	4.0047	4.0113	0.6169	0.63	0.6431	1.0283	1.0493	1.0704	17	2771
3.9992	4.0064	4.0135	0.6284	0.6433	0.6583	1.0475	1.0716	1.0958	17	2830
3.9933	4.0001	4.0069	0.6135	0.6275	0.6415	1.0229	1.0454	1.068	19	2777
3.9956	4.0025	4.0093	0.6177	0.6319	0.646	1.0299	1.0527	1.0756	16	2781
3.9887	3.9957	4.0026	0.6126	0.6264	0.6402	1.0219	1.0442	1.0665	18	2699
3.9881	3.9951	4.0022	0.6035	0.6169	0.6302	1.0074	1.029	1.0505	20	2762
3.9976	4.0046	4.0116	0.6274	0.643	0.6585	1.0461	1.0713	1.0966	18	2741
4	4.0069	4.0138	0.6254	0.6396	0.6538	1.041	1.0639	1.0868	17	2714
3.9983	4.0051	4.0119	0.6242	0.6381	0.6521	1.0406	1.0631	1.0855	17	2607
3.9942	4.0011	4.008	0.621	0.6351	0.6492	1.0353	1.0582	1.0811	21	2757
3.993	4.0001	4.0071	0.627	0.6412	0.6555	1.0447	1.0677	1.0906	16	2629
3.9929	3.9999	4.0068	0.6151	0.6292	0.6433	1.025	1.0477	1.0704	18	2827
3.9887	3.9954	4.002	0.6036	0.6169	0.6303	1.0072	1.0289	1.0506	17	2821
3.9953	4.0024	4.0095	0.6294	0.6448	0.6601	1.0502	1.0751	1.1001	19	2753
3.9968	4.0035	4.0103	0.6221	0.6354	0.6488	1.037	1.0585	1.08	17	2700
3.9926	3.9995	4.0064	0.6095	0.6233	0.6371	1.0167	1.039	1.0614	15	2676
3.9964	4.0038	4.0112	0.6257	0.6409	0.6562	1.043	1.0677	1.0923	15	2787
3.9934	4.0004	4.0074	0.6165	0.6302	0.644	1.0282	1.0503	1.0724	14	2810
3.9896	3.9966	4.0037	0.6163	0.6302	0.6442	1.0287	1.0513	1.0738	17	2767
3.999	4.0056	4.0122	0.6161	0.6304	0.6447	1.0265	1.0496	1.0727	16	2700
3.9961	4.003	4.0099	0.6184	0.6329	0.6474	1.0309	1.0544	1.0779	18	2852
3.9939	4.0007	4.0075	0.6038	0.6167	0.6295	1.0073	1.0281	1.0489	16	2775
3.9978	4.0047	4.0117	0.6228	0.6369	0.651	1.0382	1.0609	1.0836	16	2718
4.0019	4.0087	4.0156	0.6245	0.6387	0.6529	1.0409	1.0639	1.0868	17	2754
3.9971	4.0039	4.0108	0.6197	0.6329	0.646	1.0335	1.0547	1.0759	16	2653
3.987	3.9942	4.0015	0.6078	0.6215	0.6352	1.0141	1.0363	1.0585	17	2785
3.9976	4.0046	4.0116	0.6148	0.6289	0.643	1.0244	1.0472	1.07	21	2718
3.9886	3.9955	4.0023	0.6055	0.62	0.6346	1.0106	1.0342	1.0578	19	2731
3.991	3.9977	4.0045	0.6131	0.6267	0.6403	1.0223	1.0443	1.0662	18	2817
3.9907	3.9975	4.0043	0.6083	0.622	0.6357	1.0154	1.0376	1.0598	17	2905

3.9872	3.9942	4.0011	0.6143	0.6285	0.6427	1.0247	1.0477	1.0707	20	2800
4.0024	4.0094	4.0165	0.6387	0.6545	0.6703	1.0636	1.0892	1.1148	23	2758
3.9976	4.0047	4.0118	0.6195	0.6328	0.646	1.033	1.0544	1.0757	15	2734
3.999	4.0057	4.0125	0.6214	0.6358	0.6501	1.0366	1.0599	1.0832	21	2697
3.9924	3.9996	4.0067	0.6128	0.6267	0.6405	1.022	1.0445	1.067	17	2757
3.994	4.0009	4.0078	0.6147	0.6284	0.642	1.0246	1.0466	1.0687	16	2659
3.9969	4.0036	4.0103	0.6255	0.6405	0.6556	1.0427	1.0672	1.0918	24	2589
3.9923	3.9995	4.0067	0.6198	0.6341	0.6485	1.0333	1.0564	1.0796	16	2752
3.9948	4.0018	4.0088	0.6209	0.6357	0.6506	1.0356	1.0595	1.0835	19	2743
3.9949	4.0019	4.0089	0.6163	0.6304	0.6445	1.0276	1.0505	1.0734	17	2830
3.9994	4.0064	4.0135	0.6222	0.6371	0.652	1.0364	1.0605	1.0847	19	2815
3.9858	3.9929	3.9999	0.6106	0.6246	0.6385	1.0204	1.0431	1.0657	16	2720
3.99	3.9971	4.0042	0.616	0.6301	0.6442	1.0275	1.0504	1.0733	20	2785
3.9948	4.0017	4.0087	0.6206	0.6348	0.649	1.0353	1.0583	1.0813	21	2757
4.0071	4.014	4.0209	0.638	0.6526	0.6672	1.0633	1.087	1.1107	16	2705
3.9929	3.9998	4.0068	0.6051	0.6187	0.6324	1.0083	1.0304	1.0524	19	2667
3.9985	4.0055	4.0125	0.6217	0.6355	0.6494	1.0361	1.0585	1.081	16	2663
3.9936	4.0005	4.0075	0.6153	0.6297	0.644	1.0257	1.0489	1.0721	19	2739
3.9964	4.0033	4.0103	0.6144	0.628	0.6416	1.0242	1.0461	1.068	16	2774
4.0027	4.0095	4.0163	0.6356	0.6503	0.6651	1.0584	1.0823	1.1061	17	2715
3.9932	4	4.0068	0.6168	0.6303	0.6437	1.0284	1.0501	1.0718	15	2746
3.9917	3.9988	4.0059	0.625	0.6398	0.6546	1.0434	1.0674	1.0914	17	2762
3.988	3.9949	4.0019	0.6127	0.6261	0.6395	1.0215	1.0431	1.0647	16	2812
3.9931	4.0004	4.0076	0.6188	0.6329	0.6471	1.0319	1.0547	1.0775	16	2778
3.9886	3.9957	4.0028	0.6168	0.6317	0.6466	1.0302	1.0543	1.0785	20	2807
3.9998	4.0067	4.0136	0.6195	0.6333	0.6471	1.0324	1.0546	1.0769	17	2682
3.9963	4.0032	4.0101	0.6172	0.6308	0.6445	1.0286	1.0506	1.0727	19	2729
3.9882	3.9951	4.002	0.618	0.6325	0.6469	1.0312	1.0546	1.078	17	2765
3.9926	3.9994	4.0062	0.6155	0.6289	0.6423	1.0271	1.0487	1.0704	16	2850
3.9914	3.9984	4.0053	0.6128	0.6277	0.6426	1.0214	1.0455	1.0697	26	2699

Patekimas i pasikliautinuosius intervalus:
Kanalai: 87.81%
Eiles ilgis: 89.16%
Laukimo laikas: 89.10%

PRIEDAS Nr.5 10^6 LAIKO VIENETŲ MODELIAVIMO REZULTATAI

(VEIBULO PASISKIRSTYMAS, FORMOS PARAMETRAS 1.5)

	Kanalu			Eiles			Laukimas				Regeneraciniai
minpasikl	uzimtumas	maxpasikl	minpasikl	ilgis	maxpasikl	minpasikl	eileje	maxpasikl	Max eile	ciklai
3.9898	3.9975	4.0052	0.8724	0.894	0.9155	1.3133	1.3449	1.3765	22	4723
3.9897	3.9973	4.0049	0.867	0.888	0.9089	1.3063	1.337	1.3677	26	4726
3.9906	3.9984	4.0062	0.8802	0.9028	0.9253	1.3265	1.3595	1.3926	22	4791
3.9995	4.0073	4.0151	0.8878	0.911	0.9343	1.3346	1.3686	1.4026	32	4625
3.9945	4.0021	4.0098	0.8651	0.8861	0.9071	1.3013	1.332	1.3627	23	4724
3.9899	3.9978	4.0056	0.8798	0.9011	0.9224	1.3249	1.356	1.3872	23	4789
3.9933	4.0009	4.0085	0.8655	0.886	0.9066	1.3032	1.3332	1.3632	21	4687
3.9942	4.0019	4.0096	0.8688	0.8906	0.9124	1.3081	1.34	1.3719	25	4734
3.9916	3.9995	4.0074	0.881	0.9029	0.9247	1.3268	1.3587	1.3906	22	4773
3.9903	3.9982	4.0061	0.8831	0.9047	0.9262	1.3304	1.3619	1.3934	25	4685
4.0005	4.0085	4.0165	0.8907	0.9136	0.9365	1.3398	1.3733	1.4068	26	4711
3.9976	4.0053	4.013	0.8738	0.8943	0.9148	1.3145	1.3444	1.3742	19	4644
3.9968	4.0044	4.0121	0.8858	0.907	0.9282	1.3328	1.3637	1.3945	21	4671
3.9924	4.0003	4.0082	0.8645	0.8852	0.9058	1.3026	1.3327	1.3629	20	4765
3.9922	3.9998	4.0075	0.8755	0.8962	0.917	1.3184	1.3487	1.379	21	4733
3.9959	4.0036	4.0113	0.8859	0.9079	0.9298	1.3324	1.3645	1.3966	26	4736
3.9942	4.0017	4.0093	0.8613	0.8811	0.9009	1.2966	1.3255	1.3545	26	4644
3.9874	3.9952	4.003	0.8742	0.8953	0.9165	1.3174	1.3483	1.3792	22	4731
3.9929	4.0006	4.0082	0.8663	0.8873	0.9083	1.3043	1.3351	1.3658	25	4626
3.9972	4.0049	4.0127	0.8787	0.9022	0.9257	1.3226	1.3571	1.3916	25	4762
3.9962	4.0038	4.0113	0.8693	0.8899	0.9104	1.3095	1.3395	1.3695	24	4612
3.9915	3.9992	4.0068	0.873	0.8936	0.9142	1.3151	1.3453	1.3755	21	4733
3.9949	4.0025	4.01	0.8739	0.8956	0.9174	1.316	1.3479	1.3798	33	4631
3.9955	4.0032	4.0108	0.8705	0.8922	0.914	1.3109	1.3428	1.3747	21	4590
3.9962	4.004	4.0118	0.8767	0.8983	0.9198	1.3202	1.3517	1.3832	21	4653
3.9953	4.0031	4.0109	0.897	0.9189	0.9408	1.3506	1.3827	1.4148	24	4666
3.9905	3.9984	4.0062	0.8583	0.8786	0.8988	1.2921	1.3217	1.3512	22	4594
3.9946	4.0026	4.0107	0.8992	0.9212	0.9433	1.3543	1.3865	1.4187	22	4777
3.9928	4.0005	4.0081	0.8564	0.8764	0.8963	1.289	1.3182	1.3473	21	4703
3.989	3.9965	4.0041	0.871	0.8926	0.9142	1.3118	1.3434	1.3751	23	4635
3.9919	3.9997	4.0076	0.894	0.9171	0.9403	1.3471	1.3811	1.4151	24	4730

3.9875	3.9951	4.0028	0.8607	0.8834	0.906	1.2965	1.3297	1.3629	25	4846
3.9959	4.0036	4.0114	0.8722	0.8946	0.9169	1.3131	1.3458	1.3786	26	4567
3.9983	4.0063	4.0142	0.8925	0.9141	0.9357	1.3419	1.3734	1.4049	26	4763
3.9823	3.9897	3.9971	0.859	0.8795	0.9001	1.2939	1.324	1.3542	23	4825
3.988	3.9956	4.0032	0.8659	0.8871	0.9083	1.304	1.335	1.366	28	4707
3.9919	3.9997	4.0075	0.8583	0.8785	0.8986	1.2922	1.3217	1.3511	22	4792
3.991	3.9989	4.0068	0.8623	0.8833	0.9043	1.299	1.3298	1.3606	25	4748
3.9881	3.9958	4.0034	0.8623	0.8826	0.903	1.2993	1.329	1.3588	23	4674
3.9959	4.0035	4.0112	0.8824	0.9035	0.9247	1.3274	1.3583	1.3892	24	4805
3.9912	3.9992	4.0071	0.8747	0.8975	0.9204	1.3174	1.3508	1.3843	22	4651
4.0032	4.0111	4.0189	0.9056	0.9275	0.9494	1.3619	1.3938	1.4257	22	4662
3.9937	4.0013	4.0088	0.8745	0.8946	0.9148	1.3166	1.3461	1.3755	20	4704
3.9932	4.0013	4.0094	0.8904	0.9127	0.9351	1.3406	1.3732	1.4057	23	4722
3.9829	3.9905	3.9982	0.8592	0.8797	0.9002	1.2939	1.3239	1.3539	24	4810
3.9908	3.9988	4.0069	0.8868	0.91	0.9332	1.3353	1.3692	1.4032	31	4794
3.9855	3.9933	4.001	0.8494	0.8693	0.8891	1.2809	1.3099	1.3389	24	4699
3.9893	3.9972	4.005	0.865	0.8862	0.9073	1.304	1.335	1.366	25	4758
3.9939	4.0017	4.0094	0.8676	0.8888	0.9099	1.3063	1.3373	1.3683	25	4662
3.99	3.9978	4.0057	0.8623	0.8827	0.9032	1.2995	1.3294	1.3592	20	4702
3.9932	4.0011	4.0089	0.8886	0.9123	0.9359	1.3382	1.3729	1.4076	25	4687
3.9947	4.0026	4.0105	0.8817	0.903	0.9243	1.3272	1.3583	1.3894	21	4737
4.0048	4.0124	4.02	0.8831	0.9031	0.9232	1.3285	1.3577	1.3869	19	4671
3.9945	4.0021	4.0097	0.8706	0.8907	0.9108	1.3109	1.3403	1.3697	19	4626
3.9885	3.9964	4.0043	0.8685	0.8906	0.9128	1.3084	1.3408	1.3733	22	4717
3.9905	3.9982	4.0059	0.8728	0.8947	0.9167	1.316	1.3481	1.3803	22	4698
3.9887	3.9963	4.004	0.866	0.8861	0.9063	1.3046	1.334	1.3635	19	4752
3.9897	3.9974	4.0051	0.8647	0.8854	0.9062	1.3015	1.3318	1.3622	22	4733
3.9881	3.996	4.0039	0.8746	0.8946	0.9147	1.3177	1.3469	1.3761	21	4859
3.9927	4.0004	4.0081	0.8814	0.9048	0.9282	1.3264	1.3607	1.3949	35	4702
3.9856	3.9933	4.0009	0.8506	0.8711	0.8916	1.281	1.3111	1.3412	25	4762
3.9955	4.0032	4.0109	0.8732	0.8935	0.9137	1.3143	1.3439	1.3736	24	4571
4.0022	4.0099	4.0177	0.8829	0.9048	0.9268	1.3272	1.3593	1.3913	20	4669
3.9982	4.006	4.0138	0.8906	0.9121	0.9336	1.3398	1.3712	1.4025	20	4680
4.0008	4.0084	4.0159	0.9127	0.9348	0.9569	1.3735	1.4059	1.4382	25	4727
3.9882	3.9959	4.0037	0.8773	0.8997	0.9221	1.3201	1.353	1.3859	30	4720
3.991	3.9988	4.0066	0.8735	0.895	0.9165	1.3137	1.3452	1.3766	23	4826
3.9899	3.9976	4.0054	0.8749	0.8962	0.9175	1.3175	1.3487	1.3799	27	4762
3.9916	3.9994	4.0072	0.8688	0.8901	0.9113	1.3066	1.3377	1.3688	22	4720
3.9923	4.0001	4.0079	0.8671	0.8873	0.9075	1.3061	1.3356	1.3651	21	4648

3.9849	3.9927	4.0006	0.8803	0.9011	0.9218	1.3248	1.3551	1.3854	20	4746
3.9936	4.0014	4.0091	0.8726	0.8944	0.9161	1.3127	1.3445	1.3763	22	4620
3.9875	3.995	4.0026	0.8451	0.865	0.885	1.2717	1.3008	1.33	23	4691
4.0025	4.0101	4.0178	0.8886	0.9091	0.9297	1.3365	1.3666	1.3967	19	4652
3.9952	4.0031	4.0109	0.8868	0.9089	0.931	1.3344	1.3667	1.399	21	4701
3.9887	3.9965	4.0043	0.8796	0.9016	0.9236	1.3254	1.3575	1.3896	25	4781
3.9942	4.0017	4.0092	0.8676	0.8873	0.9071	1.3058	1.3346	1.3634	20	4738
3.9895	3.9973	4.0051	0.8643	0.8858	0.9073	1.3027	1.3341	1.3656	21	4723
3.9924	4.0003	4.0081	0.8799	0.9023	0.9246	1.3244	1.3572	1.39	22	4681
3.9928	4.0005	4.0081	0.8701	0.8897	0.9094	1.3109	1.3397	1.3684	21	4657
3.9928	4.0008	4.0088	0.8869	0.9099	0.9328	1.3349	1.3684	1.402	23	4711
3.9872	3.9948	4.0025	0.8569	0.8775	0.8981	1.2907	1.3208	1.3509	21	4735
3.99	3.9977	4.0054	0.8609	0.88	0.8992	1.2963	1.3242	1.3521	19	4747
3.9924	3.9999	4.0074	0.8578	0.8777	0.8976	1.2912	1.3202	1.3493	23	4740
4.004	4.0118	4.0196	0.8929	0.9152	0.9376	1.3435	1.3761	1.4088	22	4726
4.0061	4.0141	4.0222	0.9122	0.935	0.9578	1.3728	1.4061	1.4394	28	4608
3.9922	3.9997	4.0072	0.8716	0.8919	0.9121	1.3114	1.3409	1.3704	20	4769
3.9962	4.0039	4.0116	0.8761	0.8968	0.9175	1.3186	1.3488	1.379	25	4543
3.9827	3.9906	3.9986	0.876	0.8979	0.9198	1.32	1.352	1.3839	25	4838
3.988	3.996	4.0039	0.8793	0.9017	0.9241	1.3249	1.3576	1.3903	22	4758
3.9929	4.0004	4.008	0.8762	0.8981	0.9201	1.3196	1.3517	1.3839	22	4601
3.997	4.0045	4.012	0.8728	0.8931	0.9134	1.313	1.3426	1.3723	22	4606
3.9882	3.9958	4.0034	0.8564	0.8774	0.8984	1.2906	1.3214	1.3521	20	4910
3.9921	4	4.0079	0.8767	0.8975	0.9183	1.3208	1.3511	1.3814	21	4746
3.9901	3.9978	4.0056	0.8737	0.8956	0.9176	1.3156	1.3478	1.3799	25	4760
3.989	3.9968	4.0047	0.8646	0.8853	0.9059	1.3015	1.3316	1.3617	21	4781
3.991	3.9989	4.0067	0.8695	0.8911	0.9126	1.3102	1.3417	1.3732	20	4636
3.9997	4.0073	4.0148	0.8753	0.8951	0.9149	1.3158	1.3447	1.3736	22	4674
3.9974	4.0051	4.0129	0.8909	0.912	0.933	1.3387	1.3694	1.4	22	4666
3.9986	4.0066	4.0147	0.8778	0.8999	0.9221	1.3209	1.3533	1.3857	26	4680

Patekimas i pasikliautinuosius intervalus:
Kanalai: 88.03%
Eiles ilgis: 87.44%
Laukimo laikas: 87.28%

PRIEDAS Nr.6 PROGRAMOS TEKSTAS Unit1.cpp FAILAS

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
#include "Unit2.h"
#include "Math.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"

TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
void __fastcall TForm1::Button1Click(TObject *Sender)
{
 ncikl=-1;
 Laikas=0;    //pradzia modeliavimo
 Eilute="";
 Natejimo=0;
 Napdorojimo=0;
 Label4->Caption="Atejimo laikai:";
 Label7->Caption="Apdorojimo laikai:";
 Memo2->Lines->Clear();
 Series1->Clear();
 Series2->Clear();
 Series3->Clear();
 Series4->Clear();
 Series5->Clear();
 Chart1->Refresh();
 Chart2->Refresh();
 Chart3->Refresh();
 Form1->Refresh();
 try
 {
  LAIKAS=StrToFloat(SutvarkoKabl(Edit5->Text)); //galutinis laikas kai reikes nutraukti viska
  MAX=0;
  lambda=StrToFloat(SutvarkoKabl(Edit1->Text)); //nuskaito atejimo intensyvuma
  if (lambda>0)
  {
   miu=StrToFloat(SutvarkoKabl(Edit2->Text));   //nuskaito vieno kanalo apdirbimo intensyvuma
   if (RadioButton4->Checked)
   {
    forma1=StrToFloat(SutvarkoKabl(Edit4->Text));   //nuskaito veibulo formos parametra
    if (forma1<=0) throw 1;
   }
```

```
   if (RadioGroup2->ItemIndex==2)
   {
    forma2=StrToFloat(SutvarkoKabl(Edit6->Text));   //nuskaito veibulo formos parametra
    if (forma2<=0) throw 1;
   }
   Kanalai=StrToInt(Edit3->Text);               //nuskaito veikianciu kanalu skaiciu sistemoje

   for (int i=0; i<Kanalai; i++) laikasNR[i]=-1; //laisvo kanalo indikacija
     //---------------------------------------------------
    //kada ateis kita paraishka
    if (RadioButton1->Checked)//tolygusis atvejis
    {
     laikinas = RandTolyg(lambda);
     laikasa=Laikas + laikinas;
    }
    else
    if (RadioButton2->Checked)//eksponentinis atvejis
    {
     laikinas=RandExp(lambda);
     laikasa=Laikas + laikinas;
    }
    else
    if (RadioButton4->Checked)//Veibulo atvejis
    {
     laikinas=RandVeibul(lambda, forma1);
     laikasa=Laikas + laikinas;
    }
    else //determinuotas atvejis
     laikasa=Laikas + ceil(LAIKAS-1/lambda);//siuo atveju lamda
     //---------------------------------------------------
   atejimas[Natejimo]=laikinas;
   Natejimo++;

   Eile=0;

   Label8->Color = clRed;
   Label8->Caption="";
   Form1->Refresh();
   Daro();//Pradeda modeliavima
  }else ShowMessage("Blogi duomenys!");

 }catch(int param)
 {
  if (param ==1) ShowMessage("Bloga Veibulo forma!");
  if (param ==2) ShowMessage("Patikrinkite duomenis!");
 }
 catch(...){ShowMessage("Blogi duomenys!");};
}
//---------------------------------------------------------------------------
void TForm1::Daro()
{
 double procentai=-1;
```

```
 ofstream rezultatai("iteracijos.txt");
 ofstream rezciklai("regeneracijos ciklai.txt");
 rezultatai <<"Laikas    | Atejo paraiska | Eiles ilgis | Uzimti kanalai |"<<endl;
 rezciklai << "Numeris |     Laikas | Ciklo metu uzimtu kanalu suma | Ciklo metu buvusiu eiliu suma |
Ciklo atejusiu paraisku suma | Ciklo trukme | Maksimali eilė |" <<endl;
 int ivykis;
 int atejo;    //ar atejo paraiska fiksuojamu laiko momentu
 Isveda(0, -1, rezultatai, rezciklai);//statistika ir ciklu tikrinimai pirmas kartas

 ivykis = -1; //pradzhioj ateina pirma paraiska i sistema
 while (Laikas < LAIKAS)    //kol nesibaige duotas laikas
 {
  if (int(Laikas*100/LAIKAS) > procentai)
  {
   procentai = int(Laikas*100/LAIKAS);
   Label8->Caption = "Statusas:"+AnsiString(procentai)+"%";
   Label8->Refresh();
  }
  atejo=0;
  //---------------------------------------------------------------------
  if (ivykis < 0) //atejo paraiska
  {
   Laikas = laikasa; //praslenkam laika iki ivykusio ivykio (atejo paraiska)
   atejo=1;
   Eile++; //atejusi paraiska stoja i eile
   UzimtiKanalai();//randa kiek kanalu yra uzimti

   if (uzimti<Kanalai) //jeigu ne visi kanalai uzimti - paraiska is karto aptarnaus
    for (int i=0; i<Kanalai; i++)
     if (laikasNR[i] < 0) //i-tasis kanalas laisvas
     {
      ivykis = i; //radom laisva kanala, kuris aptarnaus paraiska
      break;      //nutraukiam cikla
     }

    //kada ateis kita paraishka
    if (RadioButton1->Checked)//tolygusis atvejis
    {
     laikinas = RandTolyg(lambda);
     laikasa=Laikas + laikinas;
    }
    else
    if (RadioButton2->Checked)//eksponentinis atvejis
    {
     laikinas=RandExp(lambda);
     laikasa=Laikas + laikinas;
    }
    else
    if (RadioButton4->Checked)//Veibulo atvejis
    {
     laikinas=RandVeibul(lambda, forma1);
     laikasa=Laikas + laikinas;
```

```
   }
   else //determinuotas atvejis
   {
    laikinas = ceil(LAIKAS-1/lambda);
    laikasa=Laikas + laikinas;//siuo atveju lamda
   }
   if (CheckBox2->Checked) //jeigu reikia duomenu
    if (Natejimo < 2000000) //taupom vieta, grafikui duomenu pakanka
    {
     atejimas[Natejimo]=laikinas;
     Natejimo++;
    }
 }
 //---------------------------------------------------------------------
 if (ivykis >= 0) //buvo aptarnauta paraiska "laikasNR[ivykis]" kanale
 {
  if (!atejo)     //jeigu laikas dar nepakeistas del atejusios paraiskos
   Laikas = laikasNR[ivykis]; //praslenkam laika iki ivykusio ivykio (aptarnauta paraiska)
  if (Eile > 0) //jeigu yra eile
  {
   Eile--;   //kanalas paima aptarnavimui paraiska is eiles
   if (RadioGroup2->ItemIndex == 0)//tolygusis atvejis
    {
     laikinas=RandTolyg(miu);
     laikasNR[ivykis] = Laikas + laikinas;//kada baigs apdoroti paraiska tolygiuoju atveju
    }
    else if (RadioGroup2->ItemIndex == 1)//eksponentinis atvejis
    {
     laikinas=RandExp(miu);
     laikasNR[ivykis] = Laikas + laikinas;//kada baigs apdoroti paraiska exponentiniu atveju
    }
    else if (RadioGroup2->ItemIndex == 2)//Veibulo atvejis
    {
     laikinas=RandVeibul(miu, forma2);
     laikasNR[ivykis] = Laikas + laikinas;//kada baigs apdoroti paraiska exponentiniu atveju
    }
    if (CheckBox2->Checked) //jeigu reikia duomenu
     if (Napdorojimo < 2000000) //taupom vieta, grafikui duomenu pakanka
     {
      apdorojimas[Napdorojimo]=laikinas;
      Napdorojimo++;
     }
  }else         //jeigu nera eiles - kanalas lieka tuscias
   {
    laikasNR[ivykis] = -1; //laisvo kanalo indikacija
   }
 }
 //---------------------------------------------------------------------
 if (Eile > MAX) MAX=Eile;//visu generaciju
 ivykis = RandaMin();
 Isveda(atejo, ivykis, rezultatai, rezciklai);//statistika ir ciklu tikrinimai
}//pasibaigia laikas.
```

```
 ncikl--;//paskutini nebaigta cikla istriname
 //Chart1->Visible = true;
// CheckBox1->State=cbChecked;
 rezultatai.close();
 rezciklai.close();
 if (CheckBox2->Checked) Histogramos();  //jeigu reikia duomenu
 PasiklIntervalasKanalu();
 PasiklIntervalasEiliu();
 PasiklIntervalasLaiku();
 PasiklIntervalasMAX();
 Label8->Color = clGreen;
 Label8->Caption = "Statusas: 100%";
 Memo2 -> Lines -> Add("");
 Memo2->Lines->Add("Maksimali eilė simuliacijos metu: " + AnsiString(MAX));
 Memo2->Lines->Add("Regeneracinių ciklų skaicius: "+AnsiString(ncikl));
 Memo2->Lines->SaveToFile("Statistika.txt");

 //Memo1->Lines->AddStrings(Memo2->Lines);

 if (Eilute!="")
 {
  Eilute+=AnsiString(MAX)+"\t"+AnsiString(ncikl);
  npasikl++;
  Memo1->Lines->Add(Eilute);
  Memo1->Lines->SaveToFile("Statistika_Kaupimui.txt");
 }
 Eilute="";

 Chart1->SaveToBitmapFile("Proceso eiga.bmp");
 Chart2->SaveToBitmapFile("Atejimo laikai.bmp");
 Chart3->SaveToBitmapFile("Apdorojimo laikai.bmp");
}
//---------------------------------------------------------------------------
//randa koks veiksmas bus kitas x>=0 - aptarnavimas x kanale, -1 - paraishkos atejimas i sistema
int TForm1::RandaMin()
{
 int minindex = -1;
 double reiksme = laikasa; //pradzioj tarkim kad turi ateiti nauja paraiska
 for (int i=0; i<Kanalai; i++)
  if ((laikasNR[i] > 0) && (laikasNR[i] < reiksme)) //jeigu kanale yra paraishka ir jeigu ji jau
aptarnauta
  {
   minindex = i;
   reiksme = laikasNR[i];
  }
 return minindex;
}
//---------------------------------------------------------------------------
void TForm1::Isveda(int atejo, int ivykis, ofstream& failas, ofstream& ciklufailas)
{
 double laiks;
```

```
 UzimtiKanalai();//suranda kiek yra uzimtu kanalu

 if (ivykis < 0) laiks = laikasa;
 else
  laiks=laikasNR[ivykis];    //kiek laiko dar sistema nepasikeis

 if(CheckBox1->Checked)
 {
  Series1->AddXY(Laikas, Eile+uzimti, "", clRed);//idedam ne eiles ilgi o sistemoje esanciu paraisku
skaiciu
  Series1->AddXY(laiks, Eile+uzimti, "", clRed); //idedam ne eiles ilgi o sistemoje esanciu paraisku
skaiciu
  Series3->AddXY(Laikas, uzimti, "", clBlack);   //uzpiesiam ant virsaus juodai uzimtus kanalus
  Series3->AddXY(laiks, uzimti, "", clBlack);    //uzpiesiam ant virsaus juodai uzimtus kanalus
 }
 if (uzimti==0)// regeracines epochos salygos tikrinimas "jeigu yra visi laisvi kanalai"
 // if (uzimti==Kanalai-1)//bent vienas laisvas
 {
  ///
  if (ncikl>=0)
  {
   ciklufailas  <<  (Lygiuoja  (IntToStr(ncikl+1),  9)  +  Lygiuoja(FloatToStrF(Laikas,  ffFixed,  6,
2),12)+Lygiuoja(FloatToStrF(Ciklai[ncikl].Kanalai,                    ffFixed,                    6,
0),32)+Lygiuoja(FloatToStrF(Ciklai[ncikl].EilesIlgis,                   ffFixed,                    6,
0),32)+Lygiuoja(FloatToStrF(Ciklai[ncikl].ParaiskuSk,                   ffFixed,                    6,
0),31)+Lygiuoja(FloatToStrF(Ciklai[ncikl].a,                        ffFixed,                        6,
2),15)+Lygiuoja(FloatToStrF(Ciklai[ncikl].maxas, ffFixed, 6, 0),17)).c_str()<<endl;
   ///
   failas << "--------------Regeneracinio ciklo riba--------------------" << endl;
  }

  ncikl++;
  Ciklai[ncikl].Kanalai=0;
  Ciklai[ncikl].EilesIlgis=0;
  Ciklai[ncikl].ParaiskuSk=0;
  Ciklai[ncikl].maxas=0;
  Ciklai[ncikl].a=0;

  if (Chart1->Visible) Series2->AddBubble(Laikas, 0, 0.1, "", clRed);
  if (Chart1->Visible) Chart1->Refresh();
 }
 failas             <<             (Lygiuoja(FloatToStrF(Laikas,             ffFixed,             6,
2),11)+Lygiuoja(IntToStr(atejo),17)+Lygiuoja(IntToStr(Eile),14)+Lygiuoja(IntToStr(uzimti),17)).c_str()
<< endl;
 if (ncikl>=0)
 {
  Ciklai[ncikl].Kanalai+=uzimti*(laiks-Laikas);   //dauginu ish intervalo, kurio
  Ciklai[ncikl].EilesIlgis+=Eile*(laiks-Laikas);  //metu nesikeite sistemos busena, ilgio
  Ciklai[ncikl].ParaiskuSk+=atejo;
  Ciklai[ncikl].a+=laiks-Laikas;
  if (Eile > Ciklai[ncikl].maxas) Ciklai[ncikl].maxas=Eile;//sios generacijos
 }
```

```
}
//---------------------------------------------------------------------------
void TForm1::UzimtiKanalai()
{
 uzimti=0;
 for (int i=0; i<Kanalai; i++)
 if (laikasNR[i] > 0) uzimti++;//kanalas uzimtas
}
//---------------------------------------------------------------------------
double TForm1::RandTolyg(double tikimybe)  //grazina atsitiktini dydi paiskirst. pagal tolyguji desni
{
  double p;
  p=rand();
  p=(p*2.0/tikimybe)/RAND_MAX;  //dauginta is dvieju del vidurkiu sutapatinimo su eksponentiniu atveju
 return p;
}
//--------Generuojamas eksponentinis atsitiktinis dydis----------------------
double TForm1::RandExp(double lambda){
  double p;
  p=rand();
  if (p==0) p++;//kad nesigautu log(0)  p++
  p=p*1.0/RAND_MAX;
  p=((-1)*log(p)/lambda);
 return p;
}
//--------Generuojamas Veibulo atsitiktinis dydis----------------------
double TForm1::RandVeibul(double lambda, double forma){
  double p;
  p=rand();
  if (p==0) p++;//kad nesigautu log(0)  p++
  p=p*1.0/RAND_MAX; //nuo 0 iki 1 tolygusis
  //p=((-1)*log(p)/lambda);
  //p=lambda*pow((-1)*log(p),1.0/forma);
  p=pow((-1)*log(p),1.0/forma)/lambda;
 return p;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::baigti1Click(TObject *Sender)
{
 Form1->Close();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Apie1Click(TObject *Sender)
{
 AboutBox->ShowModal();
}
//---------------------------------------------------------------------------
void TForm1::PasiklIntervalasMAX()
{
 double sumaY=0,        //Y suma
        sumaa=0,        //alfa suma
        Y_kv=0,         //Y kvadratu suma
```

```
       alfa_kv=0,      //alfa kvadratu suma
       Y_alfa=0,       //Y padauginta is alfa suma
       Y_alfa_sum=0,   //laikinas kintamasis
       s11=0,          //Yj empirine dispersija
       s22=0,          //aj empirine dispersija
       s12=0,          //(Yj ir aj)  koreliacinis momentas
       s=0,            //Dispersija
       interval=0,     //tiketinas nuokrypis
       prad=0,         //intervalo pradzia
       pab=0;          //intervalo pabaiga
 for (int i=0; i<=ncikl; i++){
    sumaY += Ciklai[i].maxas;
    sumaa += 1;
    Y_kv += (Ciklai[i].maxas*Ciklai[i].maxas);
    alfa_kv += 1;//1*1=1
    Y_alfa += (Ciklai[i].maxas*1);
 }
 ncikl++;//kad formules veiktu su standartine ishraishka (ncikl yra vienu didesnis negu atrodo, nes
isiskaito 0 reiksme)
 if (ncikl>1)
 {
  Ivertis = sumaY/sumaa;
  Y_alfa_sum = sumaY*sumaa;
  s11 = Y_kv/(ncikl-1)-sumaY*sumaY/(ncikl*(ncikl-1));
  s12 = Y_alfa/(ncikl-1)- Y_alfa_sum/(ncikl*(ncikl-1));
  s22 = alfa_kv/(ncikl-1)-sumaa*sumaa/(ncikl*(ncikl-1));
  s = s11 - 2*Ivertis*s12 + Ivertis*Ivertis*s22;
 }
 if (s < 0 || ncikl<2) Memo2->Lines->Add("Neracionalus duomenys");
   else
   {
    s = sqrt(s);//kvantilis 1.645 duoda 90% pasikliautinaji intervala
    interval = ((1.645)*s)/((sumaa/ncikl)*sqrt(ncikl));
    prad = Ivertis - interval;
    if (prad < 0) prad = 0;
    pab = Ivertis + interval;
    Memo2 -> Lines -> Add("-------------------------------------------------");
    Memo2 -> Lines -> Add("Vidutines maksimalios eiles pasikliautinasis intervalas:");
//    Memo2 -> Lines -> Add("");
    Memo2  ->  Lines  ->  Add("("  +  FloatToStrF(prad,  ffFixed,  6,4)+";  "+FloatToStrF(pab,  ffFixed,
6,4)+")");
    Memo2 -> Lines -> Add("");
    Memo2 -> Lines -> Add("Maksimalių eilės ilgių vidurkio ivertis :");
//    Memo2 -> Lines -> Add("");
    Memo2 -> Lines -> Add(FloatToStrF(Ivertis,ffFixed, 6,4));

    //Eilute+=FloatToStrF(prad,             ffFixed,             6,4)+"\t"+FloatToStrF(Ivertis,ffFixed,
6,4)+"\t"+FloatToStrF(pab, ffFixed, 6,4)+"\t";
    //Pasikliautinas[npasikl].maxeile[0]=prad;
    //Pasikliautinas[npasikl].maxeile[1]=Ivertis;
    //Pasikliautinas[npasikl].maxeile[2]=pab;
   }
```

```
 ncikl--;//sugraziname i buvusia reiksme
}
//---------------------------------------------------------------------------
void TForm1::PasiklIntervalasKanalu()
{
 double sumaY=0,        //Y suma
        sumaa=0,        //alfa suma
        Y_kv=0,         //Y kvadratu suma
        alfa_kv=0,      //alfa kvadratu suma
        Y_alfa=0,       //Y padauginta is alfa suma
        Y_alfa_sum=0,   //laikinas kintamasis
        s11=0,          //Yj empirine dispersija
        s22=0,          //aj empirine dispersija
        s12=0,          //(Yj ir aj)  koreliacinis momentas
        s=0,            //Dispersija
        interval=0,     //tiketinas nuokrypis
        prad=0,         //intervalo pradzia
        pab=0;          //intervalo pabaiga
 for (int i=0; i<=ncikl; i++){
    sumaY += Ciklai[i].Kanalai;
    sumaa += Ciklai[i].a;
    Y_kv += (Ciklai[i].Kanalai*Ciklai[i].Kanalai);
    alfa_kv += (Ciklai[i].a*Ciklai[i].a);
    Y_alfa += (Ciklai[i].Kanalai*Ciklai[i].a);
 }
 ncikl++;//kad formules veiktu su standartine ishraishka (ncikl yra vienu didesnis negu atrodo, nes
isiskaito 0 reiksme)
 if (ncikl>1)
 {
  Ivertis = sumaY/sumaa;
  Y_alfa_sum = sumaY*sumaa;
  s11 = Y_kv/(ncikl-1)-sumaY*sumaY/(ncikl*(ncikl-1));
  s12 = Y_alfa/(ncikl-1)- Y_alfa_sum/(ncikl*(ncikl-1));
  s22 = alfa_kv/(ncikl-1)-sumaa*sumaa/(ncikl*(ncikl-1));
  s = s11 - 2*Ivertis*s12 + Ivertis*Ivertis*s22;
 }
 if (s < 0 || ncikl<2) Memo2->Lines->Add("Neracionalus duomenys");
   else
   {
    s = sqrt(s);//kvantilis 1.645 duoda 90% pasikliautinaji intervala
    interval = ((1.645)*s)/((sumaa/ncikl)*sqrt(ncikl));
    prad = Ivertis - interval;
    if (prad < 0) prad = 0;
    pab = Ivertis + interval;
    Memo2 -> Lines -> Add("Vidutinio kanalu uzimtumo pasikliautinasis intervalas:");
//    Memo2 -> Lines -> Add("");
    Memo2  ->  Lines  ->  Add("("  +  FloatToStrF(prad,  ffFixed,  6,4)+";  "+FloatToStrF(pab,  ffFixed,
6,4)+")");
    Memo2 -> Lines -> Add("");
    Memo2 -> Lines -> Add("Uzimtu kanalu vidurkio ivertis :");
//    Memo2 -> Lines -> Add("");
    Memo2 -> Lines -> Add(FloatToStrF(Ivertis,ffFixed, 6,4));
```

```
    Memo2 -> Lines -> Add("");
    Memo2      ->      Lines      ->      Add("Kanalu      uzimtumas      simuliacijos      metu      :
"+FloatToStrF((Ivertis/Kanalai)*100,ffFixed, 6,4)+"%");

    Eilute+=FloatToStrF(prad,              ffFixed,              6,4)+"\t"+FloatToStrF(Ivertis,ffFixed,
6,4)+"\t"+FloatToStrF(pab, ffFixed, 6,4)+"\t";
    Pasikliautinas[npasikl].Kanalai[0]=prad;
    Pasikliautinas[npasikl].Kanalai[1]=Ivertis;
    Pasikliautinas[npasikl].Kanalai[2]=pab;
   }
 ncikl--;//sugraziname i buvusia reiksme
}
//---------------------------------------------------------------------------
void TForm1::PasiklIntervalasEiliu()
{
 double sumaY=0,        //Y suma
       sumaa=0,        //alfa suma
       Y_kv=0,         //Y kvadratu suma
       alfa_kv=0,      //alfa kvadratu suma
       Y_alfa=0,       //Y padauginta is alfa suma
       Y_alfa_sum=0,   //laikinas kintamasis
       s11=0,          //Yj empirine dispersija
       s22=0,          //aj empirine dispersija
       s12=0,          //(Yj ir aj)  koreliacinis momentas
       s=0,            //Dispersija
       interval=0,     //tiketinas nuokrypis
       prad=0,         //intervalo pradzia
       pab=0;          //intervalo pabaiga
 for (int i=0; i<=ncikl; i++){
    sumaY += Ciklai[i].EilesIlgis;
    sumaa += Ciklai[i].a;
    Y_kv += (Ciklai[i].EilesIlgis*Ciklai[i].EilesIlgis);
    alfa_kv += (Ciklai[i].a*Ciklai[i].a);
    Y_alfa += (Ciklai[i].EilesIlgis*Ciklai[i].a);
 }
 ncikl++;//kad formules veiktu su standartine ishraiska (ncikl yra vienu didesnis negu atrodo, nes
isiskaito 0 reiksme)
 if (ncikl>1)
 {
  Ivertis = sumaY/sumaa;
  Y_alfa_sum = sumaY*sumaa;
  s11 = Y_kv/(ncikl-1)-sumaY*sumaY/(ncikl*(ncikl-1));
  s12 = Y_alfa/(ncikl-1)- Y_alfa_sum/(ncikl*(ncikl-1));
  s22 = alfa_kv/(ncikl-1)-sumaa*sumaa/(ncikl*(ncikl-1));
  s = s11 - 2*Ivertis*s12 + Ivertis*Ivertis*s22;
 }
 if (s < 0 || ncikl<2) Memo2->Lines->Add("Neracionalus duomenys");
   else
   {
    s = sqrt(s);//kvantilis 1.645 duoda 90% pasikliautinaji intervala
    interval = ((1.645)*s)/((sumaa/ncikl)*sqrt(ncikl));
    prad = Ivertis - interval;
```

```
    if (prad < 0) prad = 0;
    pab = Ivertis + interval;

    Memo2 -> Lines -> Add("-------------------------------------------------");
    Memo2 -> Lines -> Add("Vidutinio eiles ilgio pasikliautinasis intervalas:");
//    Memo2 -> Lines -> Add("");
    Memo2  ->  Lines  ->  Add("("  +  FloatToStrF(prad,  ffFixed,  6,4)+";  "+FloatToStrF(pab,  ffFixed,
6,4)+")");
    Memo2 -> Lines -> Add("");
    Memo2 -> Lines -> Add("Vidutinio eiles ilgio ivertis :");
//    Memo2 -> Lines -> Add("");
    Memo2 -> Lines -> Add(FloatToStrF(Ivertis,ffFixed, 6,4));

    Eilute+=FloatToStrF(prad,              ffFixed,              6,4)+"\t"+FloatToStrF(Ivertis,ffFixed,
6,4)+"\t"+FloatToStrF(pab, ffFixed, 6,4)+"\t";
    Pasikliautinas[npasikl].Eile[0]=prad;
    Pasikliautinas[npasikl].Eile[1]=Ivertis;
    Pasikliautinas[npasikl].Eile[2]=pab;
   }
 ncikl--;//sugraziname i buvusia reiksme
}
//---------------------------------------------------------------------------
void TForm1::PasiklIntervalasLaiku()
{
 double sumaY=0,        //Y suma
       sumaa=0,        //alfa suma
       Y_kv=0,         //Y kvadratu suma
       alfa_kv=0,      //alfa kvadratu suma
       Y_alfa=0,       //Y padauginta is alfa suma
       Y_alfa_sum=0,   //laikinas kintamasis
       s11=0,          //Yj empirine dispersija
       s22=0,          //aj empirine dispersija
       s12=0,          //(Yj ir aj)  koreliacinis momentas
       s=0,            //Dispersija
       interval=0,     //tiketinas nuokrypis
       prad=0,         //intervalo pradzia
       pab=0;          //intervalo pabaiga
 for (int i=0; i<=ncikl; i++){
   // if (Ciklai[i].ParaiskuSk==0)Ciklai[i].ParaiskuSk=1;// kad nebutu dalybos is nulio
                                //atejusiu paraishku sk. per cikla
    sumaY += Ciklai[i].EilesIlgis;//*1.0/Ciklai[i].ParaiskuSk;//vidutine eile tenkanti vienai atejusiai
paraiskai
    sumaa += Ciklai[i].ParaiskuSk;
    Y_kv += Ciklai[i].EilesIlgis*Ciklai[i].EilesIlgis;
    alfa_kv += (Ciklai[i].ParaiskuSk*Ciklai[i].ParaiskuSk);
    Y_alfa += Ciklai[i].EilesIlgis*1.0*Ciklai[i].ParaiskuSk;
 }
 ncikl++;//kad formules veiktu su standartine ishraishka (ncikl yra vienu didesnis negu atrodo, nes
isiskaito 0 reiksme)
 if (ncikl>1)
 {
  Ivertis = sumaY/sumaa;
```

```
  Y_alfa_sum = sumaY*sumaa;
  s11 = Y_kv/(ncikl-1)-sumaY*sumaY/(ncikl*(ncikl-1));
  s12 = Y_alfa/(ncikl-1)- Y_alfa_sum/(ncikl*(ncikl-1));
  s22 = alfa_kv/(ncikl-1)-sumaa*sumaa/(ncikl*(ncikl-1));
  s = s11 - 2*Ivertis*s12 + Ivertis*Ivertis*s22;
 }
 if (s < 0 || ncikl<2) Memo2->Lines->Add("Neracionalus duomenys");
   else
   {
    s = sqrt(s);//kvantilis 1.645 duoda 90% pasikliautinaji intervala
    interval = ((1.645)*s)/((sumaa/ncikl)*sqrt(ncikl));
    prad = Ivertis - interval;
    if (prad < 0) prad = 0;
    pab = Ivertis + interval;
    Memo2 -> Lines -> Add("--------------------------------------------------");
    Memo2 -> Lines -> Add("Vidutinio laukimo eileje laiko pasikliautinasis intervalas:");
//    Memo2 -> Lines -> Add("");
    Memo2  ->  Lines  ->  Add("("  +  FloatToStrF(prad,  ffFixed,  6,4)+";  "+FloatToStrF(pab,  ffFixed,
6,4)+")");
    Memo2 -> Lines -> Add("");
    Memo2 -> Lines -> Add("Laukimo eileje vidurkio ivertis :");
//    Memo2 -> Lines -> Add("");
    Memo2 -> Lines -> Add(FloatToStrF(Ivertis,ffFixed, 6,4));

    Eilute+=FloatToStrF(prad,              ffFixed,              6,4)+"\t"+FloatToStrF(Ivertis,ffFixed,
6,4)+"\t"+FloatToStrF(pab, ffFixed, 6,4)+"\t";
    Pasikliautinas[npasikl].Laukimas[0]=prad;
    Pasikliautinas[npasikl].Laukimas[1]=Ivertis;
    Pasikliautinas[npasikl].Laukimas[2]=pab;
   }
 ncikl--;//sugraziname i buvusia reiksme
}
//---------------------------------------------------------------------------
AnsiString TForm1::SutvarkoKabl(AnsiString eil)
{
 int ilgis = eil.Length();
 for (int i=1; i<=ilgis; i++)
 {
  if (eil[i]==',' || eil[i]=='.')
   eil[i]=kabl;
 }
 return eil;
}
void __fastcall TForm1::FormCreate(TObject *Sender)
{
 Label5->Caption=AnsiString(1.5);
 kabl=Label5->Caption[2];  //pasigauna sistemos kablelio simboli.
 randomize();

}
//---------------------------------------------------------------------------
void __fastcall TForm1::CheckBox1Click(TObject *Sender)
```

```
{
 if (CheckBox1->Checked)
 {
  Chart1->  Visible=true;
 /* Chart2->  Visible=true;
  Chart3->  Visible=true; */
 }
 else
 {
  Chart1->  Visible=false;
 /* Chart2->  Visible=false;
  Chart3->  Visible=false;   */
 }
}
//---------------------------------------------------------------------------
// Formuoja viena lenteles eilutes elementa "eil" prailgindamas iki
// nurodyto ilgio "ilgis" ir pridedamas jo gale simbolius " |"
AnsiString TForm1::Lygiuoja(AnsiString eil, int ilgis)
{
 int temp = ilgis - eil.Length() - 2;
 AnsiString eila;
 eila = "";
 for (int i = 0; i < temp; i++)
  eila+=" ";
 eila+=eil;
 eila+=" |";
 return eila;
}
//Histogramu piesimas-------------------------------------------------------


void TForm1::Histogramos()
{
 ofstream atejimai("atejimo laikai.txt");
 ofstream apdirbimai("apdirbimo laikai.txt");

 double min=atejimas[0], max=atejimas[0];
 double intervalas=0;
 double vid=0;
 for (int i=0; i<Natejimo; i++)
 {
  if (atejimas[i] < min) min = atejimas[i];
  if (atejimas[i] > max) max = atejimas[i];
  atejimai << atejimas[i] << endl;
  vid+=atejimas[i]/Natejimo;
 }
 intervalas = (max-min)/100;
// if (intervalas<0.000000001) throw 2;
 Label4->Caption="Atejimo laiku vidurkis: " + FloatToStrF(vid, ffFixed, 6, 3);

 for (int i=0; i<100; i++)
 hist[i]=0;
```

```
 for (int i=0; i<Natejimo; i++)
  hist[int(atejimas[i]/intervalas)]++;

 for (int i=0; i<100; i++)
  Series4->AddXY(i*intervalas, hist[i], "", clBlue);
 //---------------------------------------
 min=apdorojimas[0], max=apdorojimas[0];
 intervalas=0;
 vid=0;
 for (int i=0; i<Napdorojimo; i++)
 {
  if (apdorojimas[i] < min)   min = apdorojimas[i];
  if (apdorojimas[i] > max)   max = apdorojimas[i];
  apdirbimai << apdorojimas[i] << endl;
  vid+=apdorojimas[i]/Napdorojimo;
 }
 intervalas = (max-min)/100;
// if (intervalas<0.000000001) throw 2;
 Label7->Caption="Apdorojimo laiku vidurkis: " + FloatToStrF(vid, ffFixed, 6, 3);

 for (int i=0; i<100; i++)
 hist[i]=0;

 for (int i=0; i<Napdorojimo; i++)
  hist[int(apdorojimas[i]/intervalas)]++;

 for (int i=0; i<100; i++)
  Series5->AddXY(i*intervalas, hist[i], "", clBlue);

 atejimai.close();
 apdirbimai.close();
}
void __fastcall TForm1::CheckBox2Click(TObject *Sender)
{
 if (CheckBox2->Checked)
 {
  Chart2->  Visible=true;
  Chart3->  Visible=true;
 }
 else
 {
  Chart2->  Visible=false;
  Chart3->  Visible=false;
 }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button2Click(TObject *Sender)
{
 npasikl=0;
 Memo1->Lines->Clear();
```

```
 Memo1->Lines->Add("minpasikl\tKanalu                              uzimtumas\tmaxpasikl\tminpasikl\tEiles
ilgis\tmaxpasikl\tminpasikl\tLaukimas eileje\tmaxpasikl\tMax eile\tRegeneraciniai ciklai");
 Memo1->Lines->SaveToFile("Statistika_Kaupimui.txt");
 try
 {
  Kartoti=StrToInt(Edit7->Text);
  for (int i=0; i<Kartoti; i++)
  {
   Label9->Caption="Testavimas:"+AnsiString(i+1);
   Form1->Refresh();
   Button1->Click();
  }
  Tikrinti();
 }
 catch(...){ShowMessage("Blogi duomenys!");};
}
//---------------------------------------------------------------------------
void TForm1::Tikrinti()
{
 int Nuokrypis[4];//[0]-kanalai, [1]-eile, [2]-laukimas, [3]-max eile

 for (int i=0; i<4; i++)
 {
  Nuokrypis[i]=0;
 }
 Memo2->Lines->LoadFromFile("Statistika_Kaupimui.txt");

 for (int i=0; i<npasikl; i++)
  for (int j=0; j<npasikl; j++)
  {   //kai patenka i pasikliautinaji
   if ((Pasikliautinas[i].Kanalai[0] < Pasikliautinas[j].Kanalai[1]) && (Pasikliautinas[i].Kanalai[1] <
Pasikliautinas[j].Kanalai[2]))
    Nuokrypis[0]++;

   if   ((Pasikliautinas[i].Eile[0]   <   Pasikliautinas[j].Eile[1])   &&   (Pasikliautinas[i].Eile[1]   <
Pasikliautinas[j].Eile[2]))
    Nuokrypis[1]++;

   if         ((Pasikliautinas[i].Laukimas[0]         <         Pasikliautinas[j].Laukimas[1])         &&
(Pasikliautinas[i].Laukimas[1] < Pasikliautinas[j].Laukimas[2]))
    Nuokrypis[2]++;

   //if ((Pasikliautinas[i].maxeile[0] < Pasikliautinas[j].maxeile[1]) && (Pasikliautinas[i].maxeile[1]
< Pasikliautinas[j].maxeile[2]))
    //Nuokrypis[3]++;
  }

 Memo2->Lines->Add("Patekimas i pasikliautinuosius intervalus:");
 if (npasikl==0) npasikl++;
 Memo2->Lines->Add("Kanalai: "+FloatToStrF(Nuokrypis[0]*100.0/(npasikl*npasikl), ffFixed, 6,2)+"%");
 Memo2->Lines->Add("Eiles     ilgis:     "+FloatToStrF(Nuokrypis[1]*100.0/(npasikl*npasikl),     ffFixed,
6,2)+"%");
```

```
 Memo2->Lines->Add("Laukimo    laikas:    "+FloatToStrF(Nuokrypis[2]*100.0/(npasikl*npasikl),    ffFixed,
6,2)+"%");
 //Memo2->Lines->Add("Maksimalios  eiles:  "+FloatToStrF(Nuokrypis[3]*100.0/(npasikl*npasikl),  ffFixed,
6,2)+"%");
 Memo2->Lines->Add(npasikl);
 Memo2->Lines->SaveToFile("Statistika_Kaupimui.txt");
}
```

PRIEDAS Nr.7 PROGRAMOS TEKSTAS Unit1.h FAILAS

```
//---------------------------------------------------------------------------
#ifndef Unit1H
#define Unit1H
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <ExtCtrls.hpp>
#include <ComCtrls.hpp>
#include <Chart.hpp>
#include <Series.hpp>
#include <TeEngine.hpp>
#include <TeeProcs.hpp>
#include <BubbleCh.hpp>
#include <Menus.hpp>
#include <TeEngine.hpp>
#include <iostream>
#include <fstream>
using namespace std;
//---------------------------------------------------------------------------
struct RegeneracinisCiklas
{
 double Kanalai;      //sumuojamas uzimtu kanalu skaicius kiekvienu laiko momentu ciklo metu
 double EilesIlgis;   //sumuojamas eiles ilgis kiekvienu laiko momentu ciklo metu
 double ParaiskuSk;   //ciklo metu atejusiu paraisku skaicius
 double maxas;        //dabartines generacijos maksimalus eiles ilgis
 double a;  //kiek laiko tesesi ciklas
}Ciklai[1000000];
//pasikliautinuju intervalu patikrinimui
struct Pasikliautinieji
{
 double Kanalai[3];  //[0]-minpasikl, [1]-vidurkis, [2]-maxpasikl
 double Eile[3];     //[0]-minpasikl, [1]-vidurkis, [2]-maxpasikl
 double Laukimas[3]; //[0]-minpasikl, [1]-vidurkis, [2]-maxpasikl
 //int maxeile[3];     //[0]-minpasikl, [1]-vidurkis, [2]-maxpasikl
}Pasikliautinas[1000];

class TForm1 : public TForm
{
__published:   // IDE-managed Components
```

```
        TButton *Button1;
        TEdit *Edit1;
        TLabel *Label1;
        TEdit *Edit2;
        TLabel *Label2;
        TEdit *Edit3;
        TLabel *Label5;
        TEdit *Edit5;
        TLabel *Label6;
        TLabel *Label8;
        TChart *Chart1;
        TMemo *Memo2;
        TBubbleSeries *Series2;
        TLabel *Label10;
        TMainMenu *MainMenu1;
        TMenuItem *Failas1;
        TMenuItem *Apie1;
        TMenuItem *baigti1;
        TRadioGroup *RadioGroup1;
        TRadioButton *RadioButton1;
        TRadioButton *RadioButton2;
        TRadioButton *RadioButton3;
        TCheckBox *CheckBox1;
        TLabel *Label3;
        TRadioGroup *RadioGroup2;
        TAreaSeries *Series3;
        TAreaSeries *Series1;
        TChart *Chart2;
        TBarSeries *Series4;
        TChart *Chart3;
        TBarSeries *Series5;
        TLabel *Label4;
        TLabel *Label7;
        TRadioButton *RadioButton4;
        TEdit *Edit4;
        TEdit *Edit6;
        TEdit *Edit7;
        TLabel *Label9;
        TCheckBox *CheckBox2;
        TMemo *Memo1;
        TButton *Button2;
        void __fastcall FormCreate(TObject *Sender);
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall baigti1Click(TObject *Sender);
        void __fastcall Apie1Click(TObject *Sender);
        void __fastcall CheckBox1Click(TObject *Sender);
        void __fastcall CheckBox2Click(TObject *Sender);
        void __fastcall Button2Click(TObject *Sender);
private:       // User declarations
public:        // User declarations
        double laikinas;
```

```
        char kabl;
        AnsiString Eilute; //eilute kaupiamajai statistikai saugoti
        int ncikl;   //ciklu skaitiklis
        int npasikl; //pagalbinis skaitiklis
        int uzimti;//uzimtu kanalu skaicius
        void UzimtiKanalai();//nustato kiek kanalu yra uzimtu
        double Laikas; //praejes laikas
        double laikasa; //laikas kada ateis kita paraiska
        double LAIKAS; //visos simuliacijos laikas - konstanta pradine
        int Eile; //eileje laukianciu paraishku skaicius
        int Kartoti; //kiek kartu reikia kartoti modeliavima (pasikliautinuju intervalu tikrinimui)
        int MAX; //visu laiku max eiles ilgis
        double laikasNR[1000];//kada ivyks ivykis: a-ateis paraishka, 1..N x serveris apdoros paraiska
ir priims kita skaiciuojant nuo galo "Laikas"
        double atejimas[2000000], apdorojimas[2000000], hist[110];
        int Natejimo, Napdorojimo;
        int Kanalai;//kiek yra apdorojanciu kanalu sistemoje
        double lambda, miu;//atejimo ir vieno serverio apdirbimo koeficientai
        double forma1, forma2; //formos parametrai Veibulo skirstiniui (1-atejimas, 2-apdorojimas)
        __fastcall TForm1(TComponent* Owner);
        AnsiString SutvarkoKabl(AnsiString eil);
        double RandTolyg(double tikimybe);
        double RandExp(double lambda);
        double RandVeibul(double lambda, double forma);
        double Ivertis; //uzimtu kanalu vidurkio ivertis
        void Isveda(int atejo, int ivykis, ofstream& failas, ofstream& ciklufailas);
        AnsiString Lygiuoja(AnsiString eil, int ilgis);
        void  PasiklIntervalasKanalu();//randa  iverti  ir  pasikliautinaji  intervala  vidutiniam  kanalu
uzimtumui
        void  PasiklIntervalasEiliu();//randa  iverti  ir  pasikliautinaji  intervala  vidutiniam  eiles
ilgiui
        void  PasiklIntervalasLaiku();//randa  iverti  ir  pasikliautinaji  intervala  vidutiniam  laukimo
laikui
        void PasiklIntervalasMAX();  //randa iverti ir pasikliautinaji intervala eiliu maksimumams
        void Daro();               //atlieka veiksmus
        int RandaMin();              //randa koks veiksmas bus kitas x>=0 - aptarnavimas x kanale, -1 -
paraishkos atejimas i sistema
        void Histogramos();        //nubrezia histogramas
        void Tikrinti();           //patikrina pasikliautinuju intervalu adekvatuma
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

PRIEDAS Nr.8 STRAIPSNIS

REGENERATIVE METHOD IN QUEUEING THEORY

Andrius Jusas, Algimantas Aksomaitis

Kaunas University of Technology

Most problems encountered in studying real-life systems and processes are too complex to solve them by constructing mathematical models. In many cases, the computer simulation is used. This is undoubtedly true when talking about various service systems, inventory management systems, technical control, and preventative maintenance work, as well as other multidimensional systems.

However, computer simulation has certain problems, which complicates its application:

- Indeterminate process of the stabilization period
- Initial conditions can lead to shifted characteristics
- The data is generally interdependent, and this prevents the effective use of classical statistical methods.

We provide a regenerative approach to the theory of queueing systems, the method to process system characteristics obtained from the simulation. This method helps to solve these problems inherent in computer simulation and enable the calculation of such characteristics as the average estimate of the observed size and its confidence interval, regardless of what interarrival time distribution for coming and serving applications are used.

The service system containing a single entry flow and the N service channels, with the queue without restrictions is being analyzed (see Figure 1).

The system features:

- system is modeled with a queue;
- system queue is unlimited;

- applications are served by the following principle: first come - first served.

Applications arrive to the system one by one. If the application came to find at least one free service channel, it service starts immediately. However, if all service channels are found to be occupied, then the application is joining the queue and waiting.

The time intervals between entry and processing applications are random variables, distributed, according to the chosen distribution with the selected intensity.

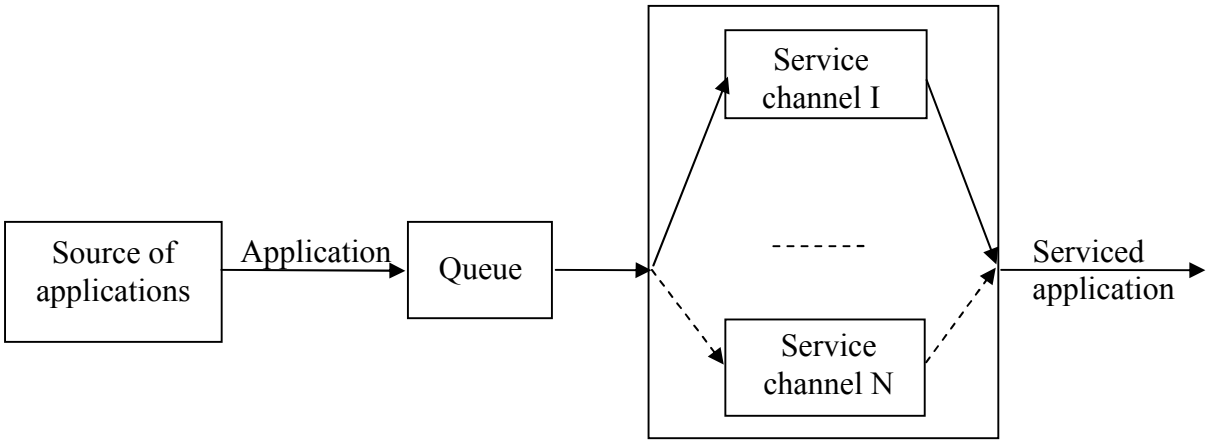


Figure 1. N-channel service system

The best-developed theory, describing this type of service systems without a simulation is using exponential times (both for coming and service times). That why the correctness of the modeling system was tested using exponentially distributed random variables, while calculating these characteristics that are used to determine system effectiveness:

- The average channel usage estimate;
- The average queue length estimate;
- The average waiting time in the queue estimate.

Also, using the regenerative method not only wanted average sizes were found, but also their confidence intervals (with 90% probability).

Theoretical characteristics were calculated and compared with the simulation results, obtained after different number of iterations.

It is known that while testing systems with simple arrival and service times, with the arrival flow intensity λ, the processing intensity μ and number of service channels n, the system can be evaluated by finding the performance characteristics with the use of the following formulas [3]:

$\rho = \frac{\lambda}{\mu}$, the ratio between the entry times and service times;

$p_0 = \frac{1}{\sum_{k=0}^{n} \frac{\rho^k}{k!} + \frac{\rho^{n+1}}{n!}}$, the probability that all the channels available;

$p_n = \frac{\rho^n}{n!} \cdot p_0$, the probability that all channels occupied, but there is no queue;

$t_{vid} = \frac{n}{\mu \cdot (n-\rho)^2} \cdot p_n$, the average waiting time in queue;

$M_{eil} = p_n \cdot \frac{\rho}{n \cdot \left(1 - \frac{\rho}{n}\right)^2}$, the average queue length;

$N_u = n - \sum_{k=0}^{n-1} \left[(n-k) \cdot \frac{\rho^k}{k!} \cdot p_0 \right]$, the average number of occupied channels.

Modeling was performed using the regenerative method. (The single channel system case is described in [1] book).

Any regenerative process with discrete time points, in some sense is distributed in steady-state, usually in this common sense: there is a K-dimensional random vector X, such that the sequence X_n converges to X, as $n \to \infty$, i.e.

$$\lim_{n \to \infty} P\{X_n \le x\} = P\{X \le x\}. \qquad (1.1)$$

We will present it in this compact way:

$$\mathbf{X}_n \Rightarrow \mathbf{X}, \text{ when } n \to \infty .$$

Since we know that in regenerative models case, the distribution is stationary, we can assess certain characteristics. Suppose that the goal of the modeling is statistic $r \equiv E\{f(X)\}$ evaluation. (Here f is measurable function).

To illustrate, suppose that X - proper size. If f(x) = x, $\forall$ x, then

$$r \equiv E\{f(X)\} = E\{X\};$$

thus, evaluation of r equivalents to the $E\{X\}$ assessment.

We will introduce the regeneration studies, which will be used to obtain confidence interval for size r. Let say that

$$Y_j = \sum_{i=\beta_j}^{\beta_{j+1}-1} f(X_i)$$

Here β_j – regenerative moments, Y_j – sum of $f(X_j)$values in j–th cycle and $\alpha_j = \beta_{j+1} - \beta_j$ is j–th cycle length. We will provide the fundamental feature of regenerative cycle (1.3).

Suppose that a sequence $\{(\mathbf{Y}_j, \alpha_j), j \geq 1\}$ consists of independent and equal distributed random vectors.

If $$E\{|f(X)|\} < \infty, \qquad (1.2)$$

then $$r \equiv E\{f(X)\} = \frac{E\{Y_1\}}{E\{\alpha_1\}}. \qquad (1.3)$$

We know that (1.2) is correct, whereas the vectors $\{(\mathbf{Y}_j, \alpha_j), j \geq 1\}$ are characterizing the regeneration cycles of the sequence, and cycles are independent and are equally behaving in probabilistic terms. In the case of (1.3), it is clear that, under fairly general conditions, with a probability equal to 1

$$\frac{f(X_1)+\cdots+f(X_N)}{N} \to E\{f(X)\}, \text{ when } N \to \infty. \qquad (1.4)$$

Suppose $\beta_1 = 1$. The first regeneration cycle starts from the beginning of the simulation. If the n-th cycle ends at the moment of N, then the ratio (1.4) we can write in the following form:

$$\frac{(Y_1 + \ldots Y_n)/n}{(\alpha_1 + \ldots + \alpha_n)/n}, \qquad (1.5)$$

Y_j is the sum of the values $f(X_i)$ in the j-th cycle, and α_j is j-th cycle length. Equation (1.5) with a probability equal to 1 converge to $\frac{E\{Y_1\}}{E\{\alpha_1\}}$, when $n \to \infty$ ([2]). Since $n \to \infty$ when $N \to \infty$, we receive a substantial relationship, which we will use to find the averages of interest values:

$$E\{f(X)\} = \frac{E\{Y_1\}}{E\{\alpha_1\}}.$$

Adequacy testing of the modeling was performed using the system with the simplest applications arrival flow times and exponential service times, with the unlimited queue and n-service channels.

The verification was carried out using the following data:

$$\lambda = 0.60,\ \mu = 0.15,\ n=5.$$

Seeking characteristics values were calculated theoretically [3], and then by means of modeling with 10^6 and $2x10^6$ time units simulations (see Table 1).

$\lambda = 0.60,\ \mu = 0.15,\ n=5$	Theoretical value	10^6 time units simulation	$2x10^6$ time units simulation
Average number of occupied channels	80%	79.7969%	80.0619%
Average queue length	2.216	2.1688	2.2130
Average waiting time in queue	3.694	3.6233	3.6862

Regenerative cycles count	7952	15656

Table 1. Modeling Data

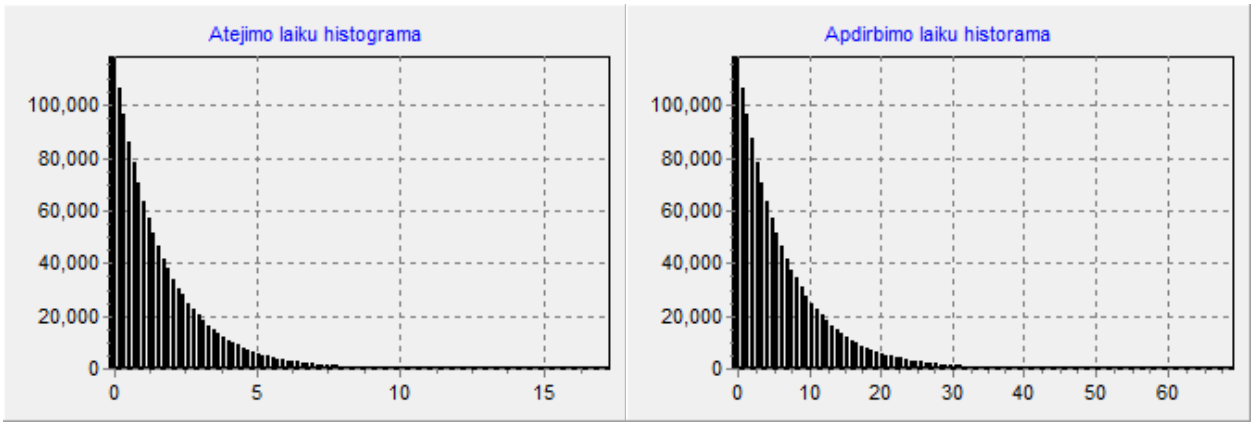


Figure 2. Generated times histograms $2x10^6$ case

As you can see, the modeling results are fairly similar to the theoretical results. In addition, it is clear that increasing the simulation iterations count is increasing the accuracy. The calculated values during simulation are getting closer to theoretical values as the iterations count increases. From the arrival and processing time histograms can be seen that model generates random values distributed according to exponential distribution.

Concluding the test we can state that the method works correctly and it is suitable for modeling systems using not only the simplest flows, but also otherwise distributed arrival and service flows.

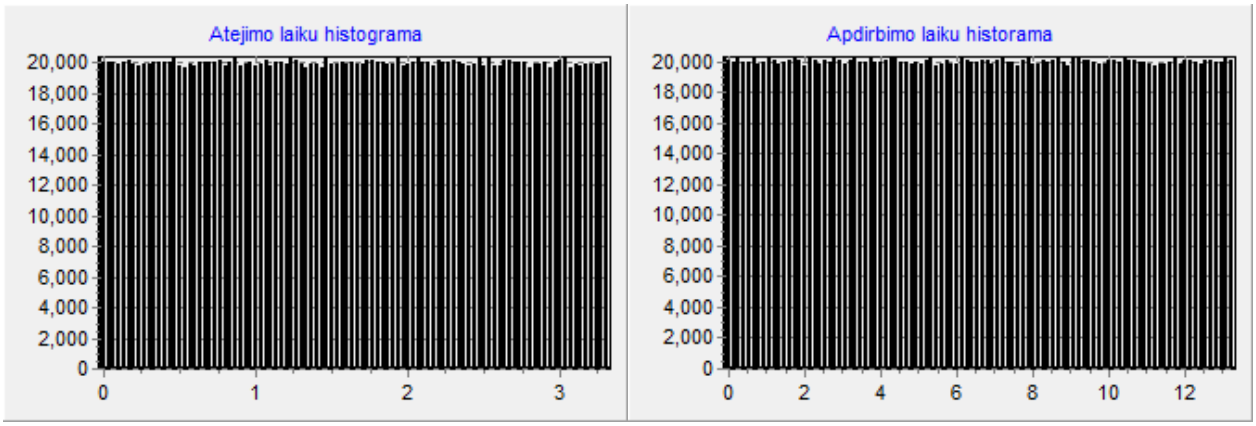
Regenerative method properties translate the known quantity dependent values to random number, but independent values. We can use any distribution and with aid of classical statistical methods calculate not only the average estimates, but also the observed values confidence intervals using computer modeling.

We took uniformly distributed arrival and service times and after applying regenerative method in simulation program we found the characteristics, indicating model performance, and their confidence intervals. We used the same flow of entry and processing intensities and the same number of processing

channels as in previous example, as well we generated 10^6, $2x10^6$ and $5x10^6$ time units simulations (see Table 2).

$\lambda = 0.60$, $\mu = 0.15$, n=5	10^6 time units simulation	$2x10^6$ time units simulation	$5x10^6$ time units simulation
Average number of occupied channels	79.9852%	80.0214%	79.9988%
Confidence interval (90%)	(3.9727; 4.0066)	(3.9962; 4.0060)	(3.9959; 4.0040)
Average queue length	0.6427	0.6351	0.6329
Confidence interval (90%)	(0.6243; 0.6572)	(0.6252; 0.6449)	(0.6264; 0.6393)
Average waiting time in queue	1.0725	1.0580	1.0548
Confidence interval (90%)	(1.0490; 1.0959)	(1.0421; 1.0740)	(1.0443; 1.0652)
Regenerative cycles count	2771	5515	13894

Table 2. Modeling Data



3 pav. Generated times histograms $5x10^6$ case

As we increase the number of iterations the modeling accuracy is increasing (decreasing confidence interval strips). Theoretically, under given arrival and service intensities and the number of channels the average channel occupation rate should be 80%, the approximation to the theoretical value

can be observed by increasing the number of iterations. Simple averages estimation may be a reference to the modeler, how accurately the program is working with a given simulation time. Naturally, it is necessary to simulate a number of times with the same number of iterations, to avoid errors due to random number generation.

The current program which is using regenerative method to process simulation results can be easily adapted to work with any arrival and service time distributions and their parameters. In this way, the methodology can be applied in practice, for any system performance and load evaluation.

References:

1. Крейн М., Лемуан О. Введение в регенеративный метод анализа моделей. – Москва: Наука, (1982).
2. Aksomaitis A. Tikimybių teorija ir statistika. Technologija, Kaunas (2002).
3. Pranevičius H., Pranevičienė I. Masinio aptarnavimo teorijos elementai. Raidės. Kaunas (1980).

Regenerative method in queueing theory

Andrius Jusas, Algimantas Aksomaitis (*Kaunas University of Technology)*

While modeling stochastic systems it is very important to examine results using reliable statistics analysis. Estimation methods that can allow user to make conclusions about statistical model from simulation results are necessary. These methods are also used while determining relation between simulation time (iterations count) and precision of estimations.

To complete the task regeneration method was chosen. This method is successfully used in various practical problems solving. The usage of regeneration method is based on the fact that many stochastic systems renovate in a probability sense.

Received results are confidential intervals of the N – channel system characteristics (average usage of channels, average queue size, average waiting time in the queue). From these results we can

judge about effectiveness of system work and relation between results' precision and simulation time (iterations count).

Regeneracinis metodas eilių teorijoje

Andrius Jusas, Algimantas Aksomaitis (*Kauno Technologijos Universitetas)*

Modeliuojant stochastines sistemas yra labai svarbu gautų rezultatų tyrimui naudoti patikimą statistinę analizę. Reikalingi metodai, kuriuos naudodamas vartotojas galėtų padaryti pagrįstas išvadas apie statistinį modelį. Tie patys metodai turi pagelbėti nustatyti sąryšį tarp modeliavimo laiko ir gaunamų įverčių tikslumo.

Šios užduoties įgyvendinimui buvo pasirinktas regeneracinis metodas. Šis metodas yra sėkmingai taikomas įvairių praktinių problemų sprendimui. Jo naudojimas pagrįstas faktu, kad dauguma stochastinių sistemų pasižymi savybe „kartotis pagal tikimybę".

Naudojant regeneracinį metodą gaunami N-kanalės sistemos charakteristikų pasikliautinieji intervalai (vidutinio kanalų užimtumo, vidutinės eilės ilgio, vidutinio laukimo eilėje laiko). Iš gautų rezultatų galime daryti išvadas apie sistemos darbo efektyvumą ir modeliavimo tikslumo priklausomybę nuo modeliavimo laiko.