KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

MATEMATINĖS SISTEMOTYROS KATEDRA

Giedrius Mickevičius

STOCHASTINIŲ SISTEMŲ FUNKCIONAVIMO APROKSIMAVIMAS MARKOVO MODELIAIS

Magistro darbas

Darbo vadovas
doc. dr. E. Valakevičius

Kaunas, 2007

KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

MATEMATINĖS SISTEMOTYROS KATEDRA

Giedrius Mickevičius

STOCHASTINIŲ SISTEMŲ FUNKCIONAVIMO APROKSIMAVIMAS MARKOVO MODELIAIS

Magistro darbas

Vadovas

Doc. dr. E. Valakevičius

2007-05

Recenzentas

Doc. dr. Vytautas Pilkauskas

Atliko

FMMM-5 gr. stud.

Giedrius Mickevičius

2007-05-25

Kaunas, 2007

KOMISIJA

Pirmininkas: Leonas Saulis, profesorius (VGTU)

Sekretorius: Eimutis Valakevičius, docentas (KTU)

Nariai: Algimantas Jonas Aksomaitis, profesorius (KTU)

Vytautas Janilionis, docentas (KTU)

Vidmantas Povilas Pekarskas, profesorius (KTU)

Rimantas Rudzkis, habil.dr., banko „NORD/LB“ vyriausiasis analitikas

Zenonas Navickas, profesorius (KTU)

Arūnas Barauskas, dr.,UAB „Elsis“ generalinio direktoriaus pavaduotojas

Mickevičius G. Approximation of stochastic systems' dynamics by Markovian models: Master's work in applied mathematics / supervisor dr. assoc. prof. E. Valakevičius; Department of mathematical research in systems, Faculty of Fundamental Sciences, Kaunas University of Technology. – Kaunas, 2007. – 90 p.

SUMMARY

Application of numerical methods with approximation allows to extend a class of systems represented by Markovian processes under investigation compared with analytical methods. So a goal was stated to create algorithms for modeling stochastic systems approximating them by Markovian models. To reach this goal the following tasks were solved:

- ✓ Analyze possibilities to approximate stochastic systems' models by Markovian models;
- ✓ Create a multipurpose software that would calculate stationary probabilities for given system described in an event-based language;
- ✓ Apply created software for models of service systems and stock valuation.

Created software is universal and easy-to-use for anyone that has at least basic knowledge in C++ language. This software was applied for modeling of service systems, for description of share price variability as Markovian process and for option pricing.

SANTRAUKA

Dažnai realių stochastinių sistemų dinamikos negalime aprašyti Markovo procesu, nes operacijų trukmės paprastai nėra pasiskirstę pagal eksponentinį dėsnį. Darbe buvo išnagrinėtas tokių atsitiktinių dydžių aproksimavimas dviejų eksponentinių atsitiktinių dydžių mišiniu.

Paprasčiausioms sistemoms kartais galima gauti analizines formules sistemos būsenų stacionarioms tikimybėms suskaičiuoti, tačiau daugeliui sistemų to padaryti negalima. Būtent tokių sistemų tyrimui, panaudojus aproksimavimo algoritmus, buvo sukurta programinė įranga, kuri leidžia modeliuoti daugelį stochastinių sistemų. Magistro darbo užduotis:

- Sukurti stochastinių sistemų modelių aproksimavimo Markovo modeliais algoritmus ir programinę įrangą.

Buvo iškelti tokie tikslai:

- ✓ Ištirti pasiskirstymo funkcijų aproksimavimo eksponentinių skirstinių mišiniu galimybes;
- ✓ Sukurti universalią programinę priemonę, kuri pagal pateiktą sistemos aprašymą, skaičiuotų jos stacionariąsias tikimybes bei funkcionavimo charakteristikas;
- ✓ Sukurtos programinės priemonės pagalba, sudaryti ir ištirti aptarnavimo sistemų ir vertybinių popierių įkainojimo modelius.

Sukurta programinė įranga pasižymi universalumu ir paprastumu vartotojui. Sistemos funkcionavimą galima aprašyti turint minimalias *C++ Builder* programavimo kalbos žinias.

Magistro darbe sukurta programinė įranga buvo pritaikyta aptarnavimo sistemoms modeliuoti, akcijų kainų dinamikai aprašyti bei opcionams įkainoti.

Buvo daryti pranešimai konferencijose:

- ✓ Šeštoji taikomosios matematikos konferencija – 2006, KTU;
- ✓ Operacijų tyrimas ir taikymai – 2006, VGTU;
- ✓ Matematika ir matematikos dėstymas – 2007, KTU.

Rezultatai paskelbti trijuose straipsniuose, vienas iš jų cituojamas tarptautinėse duomenų bazėse.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

ĮVADAS

Modeliuojant aptarnavimo sistemas, susiduriama su problema, kad labai retai realių sistemų aptarnavimo laikai būna pasiskirstę pagal eksponentinį dėsnį. Darbe (1.17) ir (1.1.8) nagrinėjamas metodas, kuris leidžia sudaryti, šių sistemų funkcionavimo, Markovo modelius, aproksimuojant aptarnavimo trukmes eksponentinių atsitiktinių dydžių mišiniu. Ištiriamos šio metodo galimybės. Buvo patikrintas metodo tinkamumas, programinės priemonės pagalba, modeliuojant paprastą aptarnavimo sistemą, kuriai yra išvestos analitinės formulės (2.2.1). Kituose skyreliuose (2.2.2) nagrinėjamos sudėtingesnės stochastinės sistemos.

Sukurtoji programinė priemonė leidžia modeliuoti finansinių aktyvų rinkas. Buvo siekiama išspręsti vieną iš pagrindinių finansų matematikos problemų – sukurti pakankamai adekvatų matematinį modelį finansų rinkų aktyvų, tokių kaip, akcijų kainų , akcijų indeksų, palūkanų normų bei valiutų kursų dinamikai analizuoti. Investuotojas, turėdamas aktyvų vertės dinamikos modelį, gali:

- teoriškai įkainoti pasirinkimo, išankstinius ir ateities sandorius bei kitus išvestinius vertybinius popierius,
- įvertinti riziką, susietą su rizikingų vertybinių popierių portfelio turėjimu bei ją valdyti.

Klasikinis būdas aprašyti kainų dinamiką yra kiekvienam aktyvui apibrėžti difuzinį procesą, t.y. stochastinį integralą arba stochastinę diferencialinę lygtį, kai atsitiktinumas yra valdomas Brauno judesio procesu. Labiausiai tikėtina, kad Brauno judesio proceso platus panaudojimas, aprašant finansinių aktyvų kainų dinamiką, susijęs su galimybe gauti analizines raiškas. Paprasčiausių išvestinių vertybinių popierių kainos išreiškiamos per Gauso skirstinį ir lengvai skaičiuojamos.

Pastaraisiais metais vis didesnį dėmesį tyrėjai skiria stochastiniams modeliams, besiskiriantiems nuo klasikinių difuzinių modelių. Empiriniai tyrimai parodė, kad Brauno procesas ne visiškai adekvačiai aprašo kainų atsitiktinį procesą. Apžvelgus užsienio autorių publikacijas [6], [7], buvo išskirtos tokios kainų dinamikos empirinės savybės: skirstiniai turi "sunkias uodegas", pasitaiko kainų sklaidos klasteriai, retkarčiais būna dideli kainų šuoliai, dažnai nepasitvirtina prielaidos apie Gauso skirtinį, o modelio parametrai dažniausiai nėra pastovūs. Kai kurie autoriai siūlo normalųjį skirstinį pakeisti kitais, labiau tinkančiais skirstiniais. Tokių skirstinių, kurie atspindi asimetriškumą, ekscesą ir kitas savybes pavyzdžiai yra: Gama, atvirkštinis Gauso, CGMY[1] [8], Hiperbolinis bei Meiksnerio. Naudojant šiuos skirstinius, kuriami Levy procesu pagrįsti modeliai. Tokių modelių pagrindinis trūkumas yra tas, kad jie analitiškai yra pakankamai sudėtingi ir sunkai pritaikomi praktiniams uždaviniams spręsti. Todėl pastaruoju metu

[1] pavadintas mokslininkų Carr, Geman, Madan and Yor pirmosiomis raidėmis

kaip alternatyva tokiems modeliams plačiai naudojamas skaitinis modeliavimas (imitavimas), kuris ženkliai suprastina praktinių uždavinių sprendimą.

Kadangi daugelis procesų, kuriais siūloma aprašyti kainų dinamiką, turi Markovo proceso savybių, tai tikslinga kainų kitimą aprašyti Markovo procesu. Darbe plačiau nagrinėta vertybinių popierių kainos dinamika bei pasirinkimo sandorių įkainojimas. Pasiūlytas skaitmeninis modelis, finansinių aktyvų dinamikai modeliuoti (2.3).

1. TEORINĖ DALIS

1.1 Markovo procesai

1.1.1 Markovo proceso sąvoka

Markovo procesai savo pavadinimą gavo, pagerbiant rusų mokslininką A. A. Markovą (1856-1922), kuris pirmą kartą suformulavo atitinkamus apibrėžimus ir ištyrė tų procesų savybes. Nuo to laiko Markovo procesų teorija buvo plačiai vystoma ir pritaikyta tokiose mokslo srityse, kaip masinio aptarnavimo teorija, branduolinė fizika, ekonomika ar epidemiologija.

Daugelio eksperimentų rezultatus technikoje ar ekonomikoje patogu nagrinėti kaip procesus arba įvykių grandinę, kada kiekviename žingsnyje įvyksta perėjimas iš vienos būsenos į kitą. Aišku, kad iš anksto negalima tiksliai numatyti, kokia kryptimi vystysis procesas kiekviename žingsnyje, o galima tik suskaičiuoti visų galimų pasekmių tikimybes. Toliau apibrėšiu atsitiktinį Markovo procesą.

Sąvoka. **Procesų būsenų erdvė**. Visų galimų proceso būsenų aibė vadinama būsenų erdve.

Apibrėžimas. Atsitiktinis procesas, atvaizduojantis sistemos S funkcionavimą vadinamas **Markovo procesu** (arba procesu be sąveikos), jeigu jis tenkina savybę: tikimybė, kokioje būsenoje sistema bus ateityje ($t>t_0$), priklauso tik nuo to, kokioje būsenoje sistema yra dabartiniu momentu t_0 ir nepriklauso nuo to, kokiu būdu sistema kito iki momento t_0 (praeityje). Trumpai galima pasakyti taip: Markovo procesui „ateitis priklauso nuo praeities tik per dabartį“.

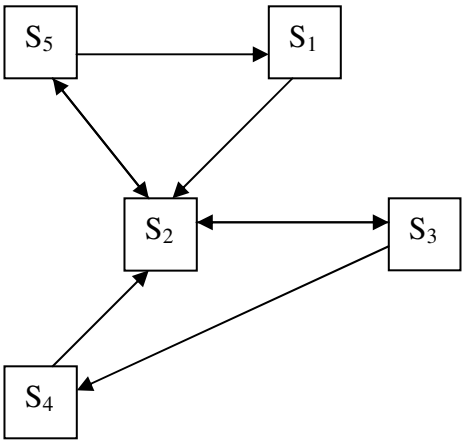
Tarkime, kad sistema S bet kuriuo laiko momentu $t>0$ gali būti vienoje iš būsenų:

$$S_1, S_2, ..., S_i, ..., S_j, ...$$

Tada analitiškai Markovo proceso savybę galima užrašyti 1.1 formule:

$$P\{S(t_0+\tau) = S_j | S(t_0)=S_i,\ S(t),\ 0<t<t_0\} = P\{S(t_0+\tau) = S_j | S(t_0)=S_i\} \ \forall \tau>0. \qquad \textbf{(1.1)}$$

Turint baigtinį būsenų skaičių, sistemą patogu atvaizduoti būsenų grafu (1.1 pav.), kur sistemos S būsenos $S_1, S_2, ..., S_5$ pavaizduotos stačiakampiais, o galimi perėjimai iš būsenos į būseną rodyklėmis.



1.1 pav. Būsenų grafas

Markovo procesą su baigtiniu būsenų skaičiumi N ir begaliniu laiku galima apibrėžti (1.2) sistema:

$$\frac{dq_i(t)}{dt} = \sum_{j=1}^{N} q_j(t)\lambda_{ij}(t) + \sum_{j=1}^{N} q_j(t)\lambda_{ji}(t), \qquad i = \overline{1,N} \tag{1.2}$$

čia $q_i(t)$ – yra tikimybė, kad sistema bus būsenoje x_i laiko momentu t; $\lambda_{ij}(t)$ – srauto intensyvumas pereinant iš būsenos x_i į būseną x_j.

Taigi šiuo atveju Markovo grandinės perėjimo tikimybės yra apibrėžiamos (1.3) išraiška:

$$p_{ij} = \frac{\lambda_{ij}}{\sum_{j=1}^{N} \lambda_{ij}} \quad i, j = \overline{1,N} \tag{1.3}$$

čia p_{ij} – perėjimo tikimybė iš būsenos x_i į būseną x_j.

Markovo grandinės stacionarios tikimybės yra apibrėžiamos tiesine lygčių (1.4) sistema:

$$p_i = \sum_{j=1}^{N} p_j p_{ji}, \quad i, j = \overline{1,N} \tag{1.4}$$

Analogiškai Markovo grandinės stacionarias tikimybes galima užrašyti ir (1.5) sistema:

$$q_i = \frac{\dfrac{p_i}{\sum_{j=1}^{N} \lambda_{ij}}}{\sum_{j=1}^{N} \left(\dfrac{p_j}{\sum_{i=1}^{N} \lambda_{ji}} \right)}, \quad i = \overline{1, N} \tag{1.5}$$

1.1.2 Diskretaus laiko Markovo grandinės

Tarkime, kad sistema S(t) bet kuriuo laiko momentu yra vienoje iš būsenų S_1, S_2,..., S_n ir keičia savo būseną diskrečiais laiko momentais t_1, t_2,..., t_m,... .

Apibrėžimas. Sistemos *S(t)* funkcionavimas yra aprašomas Markovo grandine *{S(t_m),m>0}*, jei visiems $i, j = \overline{1,n}$, *m>0* ir bet kurioms būsenoms *S(t_{m-2}), S(t_{m-3}),...,S(t_1),S(t_0)* teisinga (1.6) lygybė:

$$P(S(t_m)) = S_j \mid S(t_{m-1}) = S_i,\ S(t_{m-2}),\ldots,\ S(t_1),\ S(t_0)) = P(S(t_m) = S_j \mid S(t_{m-1}) = S_i). \tag{1.6}$$

(1.6) tikimybę vadinsime perėjimo iš i-osios būsenos į j-ąją būseną tikimybe ir žymėsime $p_{ij}(t_m)$.

(1.7) matrica

$$P(t_m) = \begin{pmatrix} p_{11}(t_m) & p_{12}(t_m) & \ldots & p_{1n}(t_m) \\ p_{21}(t_m) & p_{22}(t_m) & \ldots & p_{2n}(t_m) \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ p_{n1}(t_m) & p_{n2}(t_m) & \ldots & p_{nn}(t_m) \end{pmatrix} \tag{1.7}$$

vadinama **perėjimo matrica**. Aišku (1.8) lygčių sistema tenkinama

$$\sum_{j=1}^{n} p_{ij}(t_m) = 1. \tag{1.8}$$

Apskritai, kiekviena kvadratinė matrica, sudaryta iš neneigiamų elementų, vadinama **stochastine**, jei kiekvienos jos eilutės elementų suma yra lygi 1.

Apibrėžimas. Markovo grandinė vadinama homogenine, jei perėjimo tikimybės nepriklauso nuo žingsnio numerio, o priklauso tik nuo to, iš kokios į kokią būseną vyksta perėjimas:

$$P\{S(k) = S_j \mid S(k-1) = S_i\} = p_{ij}. \tag{1.9}$$

Jei būsenų skaičius yra baigtinis, tai grandinė vadinama **baigtine**; jei būsenų aibė skaičioji, tai ir grandinė vadinama **skaičiąja**.

Apibrėžimas. Perėjimo iš būsenos S_i į būseną S_j lygiai per n žingsnių tikimybė yra vadinama n-žingsne perėjimo tikimybe ir žymima $p_{ij}^{(n)}$. Atitinkama matrica sudaryta iš $p_{ij}^{(n)}$ elementų yra vadinama n-žingsne perėjimo tikimybių matrica. E_i

1.1.3 Būsenų ir grandinių klasifikacija

Tarkime E – būsenų erdvės S poaibis, o E‘ – poaibio E papildinys iki S, t.y. $S = E \cup E'$. Jei kiekviena aibės E būsena yra pasiekiama iš bet kurios kitos aibės E būsenos, o į būseną iš aibės E‘ negalima patekti nei iš vienos aibės E būsenos, tai aibė E yra vadinama **ergodine būsenų aibe**. Jei kartą patekome į ergodinę būsenų aibę, tai procesas negali iš jos išeiti ir nuo to momento proceso klaidžiojimas vyksta šioje aibėje.

Ergodinė būsena yra ergodinės aibės elementas.

Tegul T yra aibės S poaibis, o $T'=S\backslash T$. Jei bet kuri aibės T būsena yra pasiekiama iš bet kurios kitos aibės T būsenos ir egzistuoja bent viena būsena, priklausanti aibei T, iš kurios galima pereiti į būseną priklausančią aibei T', tai aibė T vadinama **nestabilia būsenų aibe**.

Nestabili būsena yra nestabilios būsenų aibės elementas.

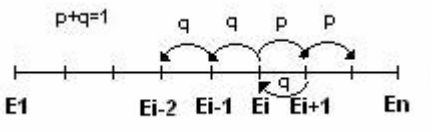
Jei ergodinė būsenų aibė susideda iš vieno elemento, tai ši būsena vadinama **sugeriančia būsena**. Procesas, kartą patekęs į šią būseną, joje ir pasilieka.

Markovo grandinių klasifikacija:

1. **Sugeriančios Markovo grandinės**. Tai grandinės, kurių stabilios būsenos yra sugeriančios.
2. **Ergodinės grandinės**. Grandinė sudaryta iš vienintelės ergodinės būsenų aibės, vadinama ergodine grandine. Galima išskirti du tokių grandinių tipus:
    a. Markovo grandinė vadinama **cikline**, jei kiekviena būsena pasiekiama per tam tikrus periodinius laiko intervalus.
    b. Grandinė vadinama **reguliaria**, jei ji yra neperiodinė ergodinė grandinė.

Pavyzdys.

Atsitiktinis klaidžiojimas sveikaskaitinėmis koordinatėmis su sugeriamaisiais ekranais E_1 ir E_n:



1.2 pav. Klaidžiojimas sveikomis koordinatėmis

Tada perėjimo per vieną žingsnį tikimybių matrica atrodytų taip:

$$P = \begin{pmatrix} 1 & 0 & 0 & ... & 0 & 0 \\ q & 0 & p & ... & 0 & 0 \\ ... & ... & ... & ... & ... & ... \\ 0 & 0 & 0 & ... & 0 & 1 \end{pmatrix}. \quad \textbf{(1.10)}$$

Būsenų aibė: $\{E_1,...,E_i,...E_n\}$;

Esminės būsenos: E_1 ir E_n;

Neesminės būsenos: $E_2,...,E_{n-1}$;

1.1.4 Reguliarių Markovo grandinių fundamentali teorema

Markovo procesą su baigtiniu būsenų skaičiumi N ir begaliniu laiku galima apibrėžti (1.11) sistema:

$$\frac{dq_i(t)}{dt} = \sum_{j=1}^{N} q_j(t)\lambda_{ij}(t) + \sum_{j=1}^{N} q_j(t)\lambda_{ji}(t), \qquad i = \overline{1,N}. \quad \textbf{(1.11)}$$

čia $q_i(t)$ – yra tikimybė, kad sistema bus būsenoje x_i laiko momentu t; $\lambda_{ij}(t)$ – srauto intensyvumas pereinant iš būsenos x_i į būseną x_j.

Taigi šiuo atveju Markovo grandinės perėjimo tikimybės yra apibrėžiamos taip:

$$p_{ij} = \frac{\lambda_{ij}}{\sum_{j=1}^{N} \lambda_{ij}} \quad i, j = \overline{1,N}. \quad \textbf{(1.12)}$$

Čia p_{ij} – perėjimo tikimybė iš būsenos x_i į būseną x_j. Markovo grandinės stacionarios tikimybės yra apibrėžiamos (1.13) tiesine lygčių sistema:

$$p_i = \sum_{j=1}^{N} p_j p_{ji}, \quad i, j = \overline{1,N}. \quad \textbf{(1.13)}$$

Analogiškai Markovo grandinės stacionarias tikimybes galima užrašyti ir taip:

$$q_i = \frac{\dfrac{p_i}{\sum_{j=1}^{N} \lambda_{ij}}}{\sum_{j=1}^{N} \left(\dfrac{p_j}{\sum_{i=1}^{N} \lambda_{ji}} \right)}, \quad i = \overline{1,N}. \quad \textbf{(1.14)}$$

Reguliarių Markovo grandinių fundamentali teorema teigia, kad nepriklausomai nuo pradinės proceso būsenos, tikimybė būti fiksuotoje reguliarios grandinės būsenoje artėja į pastovų dydį, kai žingsnių skaičius didėja. Tie pastovūs dydžiai vadinami ribinėmis tikimybėmis arba Markovo grandinės stacionariuoju pasiskirstymu.

Pavyzdys.

Išnagrinėkime sistemą, kurios perėjimo tikimybių matrica yra:

$$P = \begin{pmatrix} 0 & 0.5 & 0.5 \\ 0.5 & 0 & 0.5 \\ 0.5 & 0.5 & 0 \end{pmatrix}. \qquad \textbf{(1.15)}$$

Suskaičiuokime (1.15) matricos laipsnius:

$$P^2 = \begin{pmatrix} 0.5 & 0.25 & 0.25 \\ 0.25 & 0.5 & 0.25 \\ 0.25 & 0.25 & 0.5 \end{pmatrix}$$
$$P^3 = \begin{pmatrix} 0.375 & 0.375 & 0.25 \\ 0.375 & 0.25 & 0.375 \\ 0.25 & 0.375 & 0.375 \end{pmatrix}$$
$$P^4 = \begin{pmatrix} 0.375 & 0.313 & 0.313 \\ 0.313 & 0.375 & 0.313 \\ 0.313 & 0.313 & 0.375 \end{pmatrix}. \qquad \textbf{(1.16)}$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$P^{11} = \begin{pmatrix} 0.333 & 0.333 & 0.333 \\ 0.333 & 0.333 & 0.333 \\ 0.333 & 0.333 & 0.333 \end{pmatrix}$$

(1.16) skaičiavimuose pastebime, jog tikimybės stabilizuojasi. Jos artėja į ribinę reikšmę 1/3.

Teorema. Jei P – reguliari perėjimo tikimybių matrica ir Π – stacionarus pasiskirstymas, tai Π – vienintelis vektorius, kuris tenkina (1.17) lygčių sistemą:

$$\Pi \cdot P = \Pi. \qquad \textbf{(1.17)}$$

1.1.5 Tolydaus laiko Markovo procesas

Tikimybių teorijoje tolydaus laiko Markovo procesas yra stochastinis procesas, kuris tenkina Markovo savybę, jog proceso tikimybinis skirstinys laiko momentu s ($0<t<s$) priklauso tik nuo to, kurioje būsenoje sistema yra dabar – laiko momentu t ir nepriklauso nuo proceso istorijos.

Apibrėžimas. Jei *S(t)* yra atsitiktinis dydis apibūdinantis sistemos būsena laiko momentu *t*. Per nykstamai mažą laiko tarpą *h*, tikimybė, kad sistema pereis į būsena *S(t+h)=j* iš bet kokios būsenos *S(t)=i* laiko momentu *t* apibrėšime:

$$P(S(t+h)=j \mid S(t)=i) = q_{ij}h + o(h),$$

čia q_{ij} yra perėjimo intensyvumas iš būsenos *i* į būseną *j*, o *o(h)* yra nykstamai mažas dydis atžvilgiu *h*.

1.1.6 Markovo proceso stacionarių tikimybių skaičiavimas

Markovo proceso stacionarių tikimybių skaičiavimas skaitiniu metodu yra suvedamas į tiesinių lygčių sistemos sprendimą. Juos galima klasifikuoti į dvi grupes: tiesioginius ir iteracinius. Iteraciniai metodai tradiciškai buvo labiau naudojami nei tiesioginiai. Bet iteraciniai metodai turi didelį trūkumą, nes jiems dažniausiai reikia daug laiko tam, kad konverguotų į norimą sprendinį. Tam tikroms problemoms spręsti naudojami tiesioginiai skaičiavimai dažniausiai duoda tikslesnį rezultatą per trumpesnį laiką, lyginant su iteraciniais metodais. Tačiau vėlgi - tada susiduriama jau su kitokia problema, t.y. darbinės atminties trūkumu.

Taigi [5] knygoje buvo pasiūlytas tiesioginis skaitinis metodas skaičiuoti Markovo proceso stacionarias tikimybes. Šio metodo esmė yra perėjimo tikimybių matricos elementų interpretavimas kaip tiesinių lygčių sistemos koeficientų. Šios matricos normuoti elementai reiškia Markovo grandinės perėjimo tikimybes ir apibūdina visos sistemos veikimą. Lygčių sistemos mažinimas atliekamas paeiliui eliminuojant Markovo grandinės būsenas ir kiekvieną kartą perskaičiuojant perėjimo tikimybes.

Tarkime $P_N = \left(p_{ij}^{(N)}\right)_{NxN}$ yra Markovo grandinės $\left\{x_m^{(N)}, m \geq 0\right\}$ perėjimo tikimybių matrica, su būsenų aibe $X=\{x_1, x_2,...,x_N\}$. Stacionarios tikimybės apibrėžiamos (1.18) tiesinių lygčių sistema:

$$\overline{p_i} = \sum_{j=1}^{N} \overline{p_j} p_{ji}, i = \overline{1,N} \qquad \textbf{(1.18)}$$

kur

$$p_{ji} = \frac{\lambda_{ji}}{\sum_{i=1}^{N} \lambda_{ji}}, i, j = \overline{1,N} \qquad \textbf{(1.19)}$$

.

Stacionarių nenormuotų tikimybių skaičiavimas $\overline{p_i} = \overline{p_i}^{(N)}, i = \overline{1,N}$, susideda iš dviejų dalių: pirmoje dalyje kiekviename žingsnyje mažinamas lygčių sistemoje esančių lygybių skaičius tol, kol lieka viena būsena, o antroje – stacionarių tikimybių skaičiavimas.

Algoritmo aprašymas:

1. Markovo grandinės perėjimo tikimybės (1.20), kur k-toji būsena yra eliminuojama iš sistemos.

$$p_{ij}^{(k)} = p_{ij}^{(k+1)} + p_{i,k+1}^{(k+1)} \cdot \frac{p_{k+1,j}^{(k+1)}}{1 - p_{k+1,k+1}^{(k+1)}}, \quad i, j = \overline{1,k}, \quad k = \overline{N-1,1} \tag{1.20}$$

2. Stacionarių tikimybių skaičiavimas:

$$\begin{aligned} \overline{p}_1^{(1)} &= 1 \\ \overline{p}_i^{(k+1)} &= \begin{cases} \overline{p}_i^{(k)}, & i = \overline{1,k} \\ \dfrac{\sum_{j=1}^{k} \overline{p}_j^{(k)} p_{ji}^{(k+1)}}{1 - p_{ii}^{(k+1)}}, & i = k+1 \end{cases}, \quad k = \overline{1,N-1} \\ p_i = p_j^{(N)} &= \frac{\overline{p}_j^{(N)}}{\sum_{j=1}^{N} \overline{p}_j^{(N)}}, \quad j = \overline{1,N} \end{aligned} \tag{1.21}$$

čia p_j yra normuotos tikimybės. Ir tada stacionarios tikimybės yra:

$$q_i = \frac{\dfrac{p_i}{\sum_{j=1}^{N} \lambda_{ij}}}{\sum_{j=1}^{N} \left(\dfrac{p_j}{\sum_{i=1}^{N} \lambda_{ji}} \right)}, \quad i = \overline{1,N} \tag{1.22}$$

Pastarąsias formules ((1.20), (1.21) ir (1.22)) galima šiek tiek modifikuoti, įvedant ne perėjimo tikimybes tarp būsenų, o perėjimo intensyvumus. Tada atitinkamai gauname:

$$\lambda_{ij}^{(k)} = \lambda_{ij}^{(k+1)} + \lambda_{i,k+1}^{(k+1)} \cdot \frac{\lambda_{k+1,j}^{(k+1)}}{\overline{S}_{k+1}^{(k+1)}}, \quad i, j = \overline{1,k}, \quad k = \overline{N-1,1} \tag{1.23}$$

kur

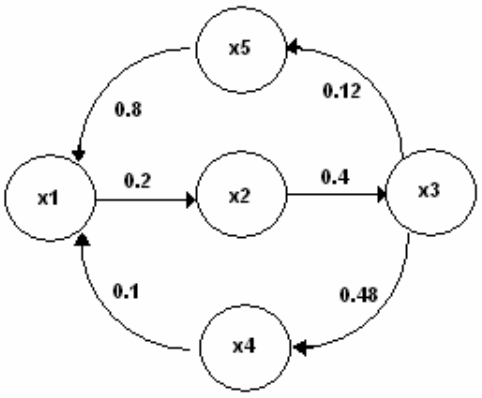
$$\overline{S}_{k+1}^{(k+1)} = \sum_{j=1, j \neq k+1}^{N} \lambda_{k+1,j}^{(k+1)} \tag{1.24}$$

ir

$$\bar{r}_1^{(1)} = 1$$
$$\bar{r}_i^{(k+1)} = \begin{cases} \bar{r}_i^{(k)}, & i = \overline{1,k} \\ \dfrac{\sum_{j=1}^{k} \bar{r}_j^{(k)} \lambda_{ji}^{(k+1)}}{\bar{S}_i}, & i = k+1 \end{cases}, \quad k = \overline{1, N-1} \qquad \textbf{(1.25)}$$
$$q_i = \frac{r_i^{(N)}}{\sum_{j=1}^{N} \left(r_j^{(N)} \right)}, \quad i = \overline{1,N}$$

Pavyzdys.

Duota sistema, pavaizduota grafu (1.3 pav.). Suskaičiuosime visų čia pavaizduotų būsenų stacionarias tikimybes.



1.3 pav. Penkių būsenų grafas.

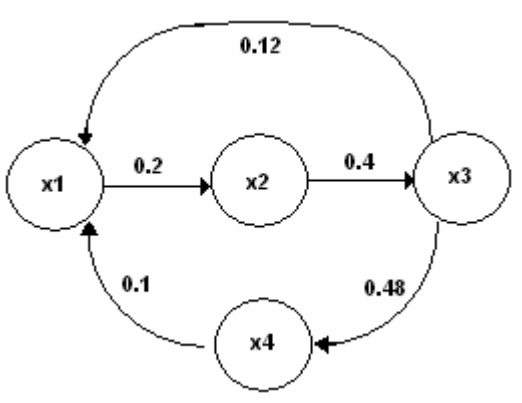
Būsenų aibė $A_5=\{x_1, x_2, x_3, x_4, x_5\}$, intensyvumų masyvas $PER^{(5)}=\{0.2, 0.4, 0.48, 0.12, 0.1, 0.8\}$ ir

$$P = \begin{pmatrix} 0 & 0.2 & 0 & 0 & 0 \\ 0 & 0 & 0.4 & 0 & 0 \\ 0 & 0 & 0 & 0.48 & 0.12 \\ 0.1 & 0 & 0 & 0 & 0 \\ 0.8 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Pirmas žingsnis. Perskaičiuokime perėjimo tikimybes tarę, kad sistemoje nėra būsenos x_5. Tada būsenų aibė $A_4=A_5\setminus\{x_5\}=\{x_1, x_2, x_3, x_4\}$ ir pagal (1.23) formulę gauname:

$$PER_{31}^{(4)} = PER_{31}^{(5)} + \frac{PER_{35}^{(5)} \cdot PER_{53}^{(5)}}{\bar{S}_5^{(5)}} = 0 + \frac{0.12 \cdot 0.8}{0.8} = 0.12;$$

Čia $\overline{S}_i = \sum_i PER_{ij}$. Kiti perėjimai nekinta. Būsenų grafas po tokių pertvarkymų atrodo taip:

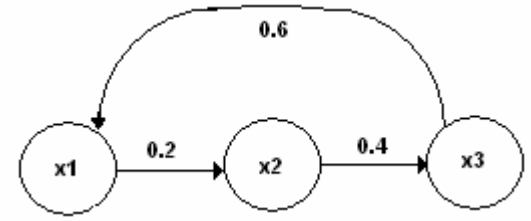


1.4 pav. Būsenų grafas be penktosios būsenos.

Antras žingsnis. Dabar perskaičiuokime perėjimo tikimybes tarę, kad sistemoje nėra būsenos x_4. Tada būsenų aibė $A_3=A_4\setminus\{x_4\}=\{x_1, x_2, x_3\}$ ir gauname:

$$PER_{31}^{(3)} = PER_{31}^{(4)} + \frac{PER_{34}^{(4)} \cdot PER_{43}^{(4)}}{\overline{S}_4^{(4)}} = 0.12 + \frac{0.48 \cdot 0.1}{0.1} = 0.6;$$

Vėl kiti perėjimai nekinta. Būsenų grafas po tokių pertvarkymų atrodo taip:



1.5 pav. Būsenų grafas iš trijų likusių būsenų.

Analogiškai skaičiuodami, po ketvirto žingsnio gauname:

$$PER_{11}^{(1)} = PER_{11}^{(2)} + \frac{PER_{12}^{(2)} \cdot PER_{21}^{(2)}}{\overline{S}_2^{(2)}} = 0 + \frac{0.2 \cdot 0.4}{0.4} = 0.2$$

1.6 pav. Vienos būsenos grafas ir paskutinis algoritmo žingsnis.

Antroji uždavinio dalis sprendžiama remiantis (1.25) formulėmis. Gauname stacionarias tikimybes $q_1,..,q_5$:

$$r_1 = 1$$

$$r_2 = \frac{r_1 \cdot PER_{12}^{(2)}}{\overline{S_2}} = \frac{1 \cdot 0.2}{0.4} = 0.5$$

$$r_3 = \frac{r_1 \cdot PER_{13}^{(3)} + r_2 \cdot PER_{23}^{(3)}}{\overline{S_3}} = \frac{0 + 0.5 \cdot 0.4}{0.6} = \frac{1}{3}$$

$$r_4 = 1.6$$

$$r_5 = 0.05$$

$$q_i = \frac{r_i}{\sum_{i=1}^{5} r_i} = \frac{60}{209} \quad q_2 = \frac{30}{209} \quad q_3 = \frac{20}{209} \quad q_4 = \frac{96}{209} \quad q_5 = \frac{3}{209}$$

1.1.7 Tikimybinio skirstinio aproksimacija, lyginant tris pradinius momentus

Tam, kad procesas būtų Markovo, trukmės tarp būsenų pasikeitimo skirstinys turi būti eksponentinis. Paprastai ši sąlyga nėra išpildoma. Nežinomam skirstiniui aproksimuoti panaudosime fiktyvių eksponentinių fazių metodą [4]. Jis leidžia sistemos perėjimo iš vienos būsenos į kitą intensyvumus, t. y. trukmę tarp būsenų pasikeitimo aprašyti tiesine atsitiktinių dydžių kombinacija. Būsenos kitimas aprašomas nuoseklia *k* fazių, turinčių eksponentinius skirstinius su parametrais $\mu_j, j = 1,2,\cdots,k$, seka. Po *j* fazės su tikimybe $(1 - a_j)$ ši seka yra paliekama. Tik viena fazė iš *j* fazių struktūroje gali būti užimta bet kuriuo laiko momentu.

Tarkime, kad teigiamo atsitiktinio dydžio *Z* (sistemos perėjimo iš vienos būsenos į kitą trukmė) pasiskirstymo funkcija yra *G(z)*. Aproksimuokime *G*(z) eksponentinių skirstinių sąsūkų mišiniu. Tarkime, kad egzistuoja skirstinio *G*(z) pirmieji trys pradiniai momentai $m_k, k = \overline{1,3}$. Jie gali būti žinomi arba gauti jų statistiniai įverčiai iš stebėtų duomenų. Sukonstruokime kitą atsitiktinį dydį *Y* tokiu būdu:

$$Y=\begin{cases} Y_1^{(1)} \quad \text{su} \quad \text{tikimybe} \quad p_2; \\ Y_1^{(1)}+Y_2^{(2)} \quad \text{su} \quad \text{tikimybe} \quad p_1 p_2; \\ \quad \dots \\ Y_1^{(1)}+Y_2^{(2)}+\dots+Y_2^{(n)} \quad \text{su} \quad \text{tikimybe} \quad p_1^{n-1}\cdot p_2; \\ \quad \dots \end{cases} \tag{1.26}$$

čia $Y_i^{(j)} \quad i=1,2\,;\ j=1,n$ yra nepriklausomi atsitiktiniai dydžiai, pasiskirstę pagal eksponentinį dėsnį su tankio funkcijomis $\mu_i e^{-\mu_i t}$; $p_1+p_2=1$. Atsitiktinis dydis *Y* lygus nepriklausomų eksponentiškai pasiskirsčiusių atsitiktinių dydžių sumai su atsitiktiniu dėmenų skaičiumi *N*, pasiskirsčiusiu pagal geometrinį dėsnį. Šio dydžio tankio funkcija yra [2]:

$$f(y)=p_2\mu_1\left(\frac{\mu_2-\mu_1}{\mu_2 p_2-\mu_1}e^{-\mu_1 y}-\frac{\mu_2 p_1}{\mu_2 p_2-\mu_1}e^{-\mu_2 p_2 y}\right) \tag{1.27}$$

Lengva įsitikinti, jei $\mu_1=\mu_2$, tai atsitiktinio dydžio *Y* tankio funkcija:

$$f(y)=p_2\mu_2 e^{-\mu_2 p_2 y} \tag{1.28}$$

Apibrėžkime atsitiktinį dydį *X* tokiu būdu:

$$X=\begin{cases} X_1 \quad \text{su} \quad \text{tikimybe} \quad p_2; \\ X_1+X_2 \quad \text{su} \quad \text{tikimybe} \quad p_1, \end{cases} \tag{1.29}$$

čia X_1 ir X_2 nepriklausomi eksponentiškai pasiskirstę atsitiktiniai dydžiai atitinkamai su parametrais μ_1 ir. $p_2\mu_2$; $p_1+p_2=1$

Teiginys. *Atsitiktinių dydžių X ir Y*, *apibrėžtų* (1.29) *ir* (1.26) *raiškomis, skirstiniai yra vienodi.*

Įrodymas. Parodysime, kad dydžių *X* ir *Y* tankio funkcijos sutampa. Pasinaudoję rezultatu, gautu [3], randame atsitiktinio dydžio *X* tankio funkciją

$$f(x)=\mu_1 e^{-\mu_1 x}+\frac{p_1\mu_1}{p_1\mu_2-\mu_1}\left(\mu_1 e^{-\mu_1 x}-p_2\mu_2 e^{-p_2\mu_2 x}\right) \qquad \textbf{(1.30)}$$

Po algebrinių pertvarkymų gauname tokią raišką:

$$f(x)=p_2\mu_1\left(\frac{\mu_2-\mu_1}{\mu_2 p_2-\mu_1}e^{-\mu_1 x}-\frac{\mu_2 p_1}{\mu_2 p_2-\mu_1}e^{-p_2\mu_2 x}\right) \qquad \textbf{(1.31)}$$

Matome, kad (1.27) ir (1.31) išraiškos sutampa.

*Pastaba***.** Atsitiktinis dydis, apibrėžtas (1.29) išraiška, įgalina automatizuoti skaitmeninio modelio sudarymą. To neleistų atlikti atsitiktinio dydžio *Y* raiška, nes eksponentinių fazių skaičius yra neapibrėžtas.

Norėdami aproksimuoti atsitiktinio dydžio *Z* tankio funkciją dydžio *X* tankiu (1.31), turime įvertinti nežinomus parametrus μ_1,μ_2, p_1 ir p_2. Tam pakanka sulyginti abiejų atsitiktinių dydžių tris pradinius momentus. Prie gautų lygčių prijungę sąlygą $p_1+p_2=1$, gauname netiesinių lygčių sistemą nežinomiems parametrams suskaičiuoti:

$$\begin{cases}\dfrac{1!\cdot p_2\mu_1}{\mu_2 p_2-\mu_1}\left(\dfrac{\mu_2-\mu_1}{\mu_1^2}-\dfrac{\mu_2 p_1}{\mu_2^2 p_2^2}\right)=m_1;\\ \dfrac{2!\cdot p_2\mu_1}{\mu_2 p_2-\mu_1}\left(\dfrac{\mu_2-\mu_1}{\mu_1^3}-\dfrac{\mu_2 p_1}{\mu_2^3 p_2^3}\right)=m_2;\\ \dfrac{3!\cdot p_2\mu_1}{\mu_2 p_2-\mu_1}\left(\dfrac{\mu_2-\mu_1}{\mu_1^4}-\dfrac{\mu_2 p_1}{\mu_2^4 p_2^4}\right)=m_3.\\ p_1+p_2=1.\end{cases} \qquad \textbf{(1.32)}$$

Lygčių sistemos (1.32) sprendinys yra toks:

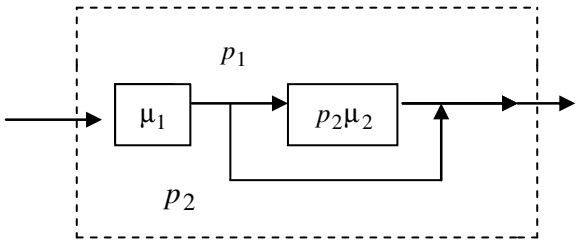
$$\mu_2=\frac{g_2-g_1^2}{g_1^3-2g_1g_2+g_3},\quad g_k=\frac{m_k}{k!},\quad k=\overline{1,3};$$

$$\mu_1 = \frac{1+\mu_2 g_1 \pm \sqrt{(1-\mu_2 g_1)^2 + 4\mu_1{}^2\left(g_2 - g_1^2\right)}}{2g_1 - 2\mu_2\left(g_2 - g_1^2\right)}; \qquad \textbf{(1.33)}$$

$$p_1 = \frac{\mu_2(\mu_1 g_1 - 1)}{\mu_2(\mu_1 g_1 - 1) + \mu_1};$$

$$p_2 = \frac{\mu_1}{\mu_2(\mu_1 g_1 - 1) + \mu_1};$$

Eksponentinių fazių grafinė struktūra pavaizduota 1.7 pav.



1.7 pav. Dviejų eksponentinių fazių diagrama, lyginant tris pradinius momentus

Pastebime, kad aproksimacija grąžins netinkamus fazinio modelio intensyvumus ir tikimybes, jeigu $\mu_2 < 0$ (1.33). Ištirsime šio metodo galimybes. Tad

$$\mu_2 = \frac{\frac{m_2}{2} - m_1^2}{m_1^3 - m_1 m_2 + \frac{m_3}{6}} < 0; \qquad \textbf{(1.34)}$$

$$\frac{m_2}{2} - m_1^2 < 0 \quad arba \quad m_1^3 - m_1 m_2 + \frac{m_3}{6} < 0$$

$$\frac{m_2}{m_1^2} < 2 \quad arba \quad \frac{m_2}{m_1^2} - \frac{m_3}{6m_1^3} > 1$$

Atsitiktinio dydžio, pasiskirsčiusio ne pagal eksponentinį dėsnį, aproksimacijos dviem eksponentinėm fazėm pavyzdys yra nagrinėjamas tiriamojoje dalyje 2.1 skyrelyje.

1.1.8 Tikimybinio skirstinio aproksimacija, lyginant du pradinius momentus

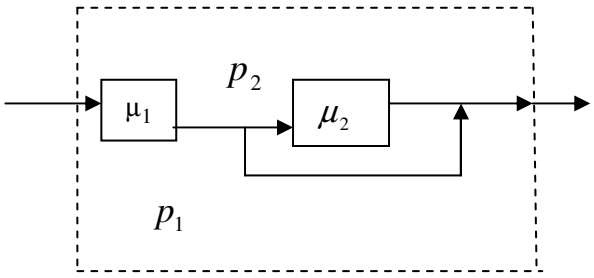
Literatūroje [11] pateikiamas tikimybinio skirstinio aproksimavimas įvertinant tik du pradinius momentus. Tada keturių lygčių sistemą su keturiais nežinomaisiais (1.32) pakeičiame trijų lygčių sistema (1.35).

$$\begin{cases} \dfrac{1!\cdot p_2\mu_1}{\mu_2 p_2-\mu_1}\left(\dfrac{\mu_2-\mu_1}{\mu_1^2}-\dfrac{\mu_2 p_1}{\mu_2^2 p_2^2}\right)=m_1; \\ \dfrac{2!\cdot p_2\mu_1}{\mu_2 p_2-\mu_1}\left(\dfrac{\mu_2-\mu_1}{\mu_1^3}-\dfrac{\mu_2 p_1}{\mu_2^3 p_2^3}\right)=m_2; \\ p_1+p_2=1. \end{cases} \qquad \textbf{(1.35)}$$

Šiuo atveju ieškomų parametrų μ_1, μ_2, p_1 ir p_2 parinkimui turime be galo daug galimų variantų. Buvo pasiriktas variantas:

$$\mu_1=2\mu,\ p_2=(2\cdot c_s^2)^{-1},\ \mu_2=\mu_1 p_2,\ p_1=1-p_2 \qquad \textbf{(1.36)}$$

čia μ ir c_s yra nagrinėjamo skirstinio vidurkio ir vidutinės kvadratinės nuokrypos atvirkštiniai dydžiai. Tada dviejų eksponentinių fazių diagrama (1.7 pav.)



1.8 pav. Dviejų eksponentinių fazių diagrama, lyginant du pradinius momentus

Pastebime, kad aproksimacija grąžins tinkamus fazinio modelio tikimybes, jeigu $p_2 \le 1$ (1.36). Ištirsime šio metodo galimybes. Tad

$$p_2 = \frac{1}{2 \cdot c_s^2} \le 1 \quad \Rightarrow \quad \frac{m_1^2}{2(m_2 - m_1^2)} \le 1$$

$$a)\, jei \quad m_2 - m_1^2 > 0 \quad \Rightarrow \quad \frac{m_2}{m_1^2} > 1,$$

$$tai \quad m_1^2 \le 2(m_2 - m_1^2) \quad \Rightarrow \quad \frac{m_2}{m_1^2} \ge 1.5$$

$$b)\, jei \quad m_2 - m_1^2 < 0 \quad \Rightarrow \quad \frac{m_2}{m_1^2} < 1,$$

$$tai \quad m_1^2 \ge 2(m_2 - m_1^2) \quad \Rightarrow \quad \frac{m_2}{m_1^2} \le 1.5$$

Tikimybės p_1 ir p_2 bus realios, jeigu $\frac{m_2}{m_1^2} \notin \left(1, \frac{3}{2}\right)$.

1.2 Finansų rinkų modeliavimas

1.2.1 Opciono sąvoka

Opcionas yra teisė, bet ne įsipareigojimas, pirkti (arba parduoti) finansinį aktyvą su tam tikromis sąlygomis. Paprastai tai yra iš anksto nustatyta aktyvo kaina ir periodas kurį galioja opcionas. Opcionas, kuris suteikia teisę ką nors pirkti, vadinamas pirkėjo (*call*) arba pirkimo opcionu, o opcionas, kuris suteikia teisę parduoti sandorio objektą, vadinamas pardavėjo (*put*) arba pardavimo opcionu. Jei viena iš kontrakto šalių turi teisę pirkti arba parduoti prekę fiksuota kaina fiksuotu laikotarpiu, tai kita kontrakto šalis įsipareigoja įvykdyti kontrakto sąlygas. Paprastai opcionas turi kainą, kuri vadinama opciono premija (K). Premija paprastai sudaro mažą pasirinkto opciono aktyvo kainos dalį. Jei opciono turėtojas nuperka ar parduoda finansinį aktyvą pagal opciono sąlygas, tai sakoma, kad opciono turėtojas įvykdo opcioną. Bet kuriuo atveju sumokėta premija yra negrąžinama.

Opcionai yra keleto tipų, tačiau populiariausi yra du, priklausomai nuo to, kada yra įvykdomas opcionas jo gyvavimo metu, tai - europietiškas ir amerikietiškas opcionai. Europietiškasis opcionas gali būti įvykdytas tik opciono pabaigoje. Amerikietiškajame opcione jis gali būti įvykdytas bet kuriuo opciono gyvavimo metu. Šiame darbe nagrinėjamas europietiškasis pirkėjo (call) opcionas.

Bendru atveju opciono vertė laiko momentu T suskaičiuojama pagal formulę:

$$C_T = \max\{Y(T) - K, 0\} \qquad \textbf{(1.37)}$$

1.2.2 Opciono įkainojimo Markovo modelis

Imkime Europinį *call* opciono atvejį, ir jo vertė yra

$$C(T)=max\{S(T) - K,0\} \quad \textbf{(1.38)}$$

Čia S(T) yra opciono rinkos vertė momentu T, o K – opciono įvykdymo kaina.

Modeliui apibūdinti tarkime, kad finansinio aktyvo S kaina kinta kaip Markovo grandinė su perėjimo tikimybių matrica

$$P = (p_{ij})$$

ir būsenų erdve:

$$I = \{-n, -(n-1), \dots, 0, 1, \dots, n\}. \quad \textbf{(1.39)}$$

Gauname, jog laiko momentu t finansinio aktyvo rinkos vertė C(t), kai žinoma vertė $S(t) = S_t$, likęs terminas $T - t$ ir įvykdymo kaina $K = k_0\Delta$, turi tokį tikimybinį pasiskirstymą:

$$\left.\begin{aligned} P[C(T) = (j - k_0)\Delta] &= p_{S,j}^{(T-t)}, \quad j > k_0 \\ P[C(T) = 0] &= \sum_{l \le k0} p_{S,l}^{(T-t)}, \quad j \le k_0 \end{aligned}\right\} \quad \textbf{(1.40)}$$

Iš (1.40) išraiškos galime nesunkiai suskaičiuoti dominančius C parametrus. Pavyzdžiui, C(t) vidurkio išraiška:

$$E[C(T) \mid S(t) = S_t] = \sum_{j>k_0} p_{S,j}^{(T-t)} (j - k_0)\Delta \quad \textbf{(1.41)}$$

Žinoma, reikia suskaičiuoti finansinio aktyvo rinkos vertę laiko momentu t, kai palūkanų norma i. Taigi:

$$\begin{aligned} &C(t) = v^{(T-t)} E[C(T) \mid S(t) = S_t] = v^{(T-t)} \sum_{j>k_0} p_{S,j}^{(T-t)} (j - k_0)\Delta; \\ &v = \frac{1}{1+i}; \end{aligned} \quad \textbf{(1.42)}$$

Jeigu matrica P yra ergodinė ir skirtumas T-t yra pakankamai didelis, tai (1.41) ir (1.42) formules galima aproksimuoti atitinkamai taip (1.43) ir (1.44):

$$\left.\begin{aligned} P[C(T) = (j - k_0)\Delta] &= \pi_j, \quad j > k_0 \\ P[C(T) = 0] &= \sum_{l \le k0} \pi_l, \quad j \le k_0 \end{aligned}\right\} \quad \textbf{(1.43)}$$

$$\left.\begin{aligned} E[C(T) \mid S(t) = S_t] &= \sum_{j>k0} \pi_j (j - k_0)\Delta \\ C(t) &= v^{(T-t)} \sum_{j>k0} \pi_l (j - k_0)\Delta \end{aligned}\right\} \quad \textbf{(1.44)}$$

Čia vektorius $\pi = (\pi_{-n}, \dots, \pi_0, \dots, \pi_n)$ yra būsenų (iš būsenų erdvės I (1.39)) stacionarių tikimybių vektorius.

2. TIRIAMOJI DALIS

2.1 Atsitiktinio dydžio aproksimavimas

2.1.1 Log-normaliojo skirstinio aproksimavimas

Tarkime atsitiktinis dydis yra pasiskirstęs pagal log-normalųjį dėsnį. Jį galima aproksimuoti eksponentinių skirstinių mišiniu (metodas aprašytas 1.1.7 skyrelyje). Log-normaliojo skirstinio tankis:

$$f(x,\alpha,\lambda)=\frac{1}{\alpha x\sqrt{2\pi}}e^{\frac{-(\ln x-\lambda)^2}{2\alpha^2}},\quad x>0 \qquad \textbf{(2.1)}$$

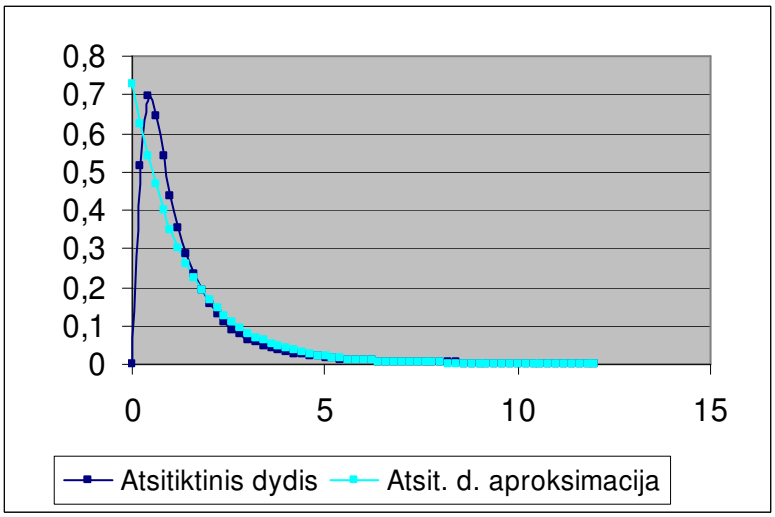
Su skirstinio parametrais $\alpha=0.9,\ \lambda=-0.05$ pirmieji trys pradiniai momentai:

$$m_1=1.42618,\ m_2=4.57225 \text{ ir } m_3=28.3606.$$

Pagal (1.33) formules suskaičiuojame eksponentinių skirstinių parametrus, gauname:

$$\mu_1=0.75935,\ \mu_2=0.141605,\ p_1=0.00559,\quad p_2=0.99441.$$

Tankio funkcijų grafinis palyginimas pateikiamas 2.1 pav.



2.1 pav. Log-normaliojo ir aproksimuoto dydžio tankio funkcijų palyginimas

Tiriant šio modelio apibrėžimo sritį, empiriškai buvo patikrintos įvairios α ir λ kombinacijos. 2.1 lentelėje žaliai pažymėtos celės, kurios netenkina (1.34) nelygybių. Raudonai pažymėtos celės, kurios tenkina šias nelygybes, o log-normaliojo skirstinio su atitinkamais parametrais aproksimuoti šiuo metodu negalime.

2.1 lentelė

Log-normaliojo atsit.d. aproksimavimo dviejų eksponentinių atsit. d. mišiniu tyrimas

λ	α	$\frac{m_2}{m_1^2}$	$\frac{m_2}{m_1^2}-\frac{m_3}{6m_1^3}$	μ_2
-1	1,5	9,487736	-133,855391	0,024683
-0,5	1,5	9,487736	-133,855391	0,014971
1	1,5	9,487736	-133,855391	0,003340
0,5	1,5	9,487736	-133,855391	0,005508
-1	1,4	7,099327	-53,535547	0,048588
1	1,4	7,099327	-53,535547	0,006576
-0,5	1,4	7,099327	-53,535547	0,029470
0,5	1,4	7,099327	-53,535547	0,010841
1	1,3	5,419481	-22,109574	0,012220
-0,5	1,3	5,419481	-22,109574	0,054767
-1	1,3	5,419481	-22,109574	0,090295
0,5	1,3	5,419481	-22,109574	0,020148
-1	1,2	4,220696	-9,310742	0,157789
-0,5	1,2	4,220696	-9,310742	0,095704
1	1,2	4,220696	-9,310742	0,021354
0,5	1,2	4,220696	-9,310742	0,035208
-1	1,1	3,353485	-3,931985	0,255481
1	1,1	3,353485	-3,931985	0,034576
-0,5	1,1	3,353485	-3,931985	0,154957
0,5	1,1	3,353485	-3,931985	0,057005
0,5	1	2,718282	-1,629308	0,081090
1	1	2,718282	-1,629308	0,049184
-1	1	2,718282	-1,629308	0,363420
-0,5	1	2,718282	-1,629308	0,220425
-0,5	0,9	2,247908	-0,645239	0,211251
-1	0,9	2,247908	-0,645239	0,348293
0,5	0,9	2,247908	-0,645239	0,077715
1	0,9	2,247908	-0,645239	0,047136
1	0,8	1,896481	-0,240346	-0,057529
-1	0,8	1,896481	-0,240346	-0,425084
-0,5	0,8	1,896481	-0,240346	-0,257826
0,5	0,8	1,896481	-0,240346	-0,094849
1	0,7	1,632316	-0,092556	-0,571929
-1	0,7	1,632316	-0,092556	-4,226013
-0,5	0,7	1,632316	-0,092556	-2,563206
0,5	0,7	1,632316	-0,092556	-0,942951
1	0,6	1,433329	-0,057451	-1,515442
-1	0,6	1,433329	-0,057451	-11,197685
-0,5	0,6	1,433329	-0,057451	-6,791739
0,5	0,6	1,433329	-0,057451	-2,498541
-1	0,5	1,284025	-0,068808	-12,480640
1	0,5	1,284025	-0,068808	-1,689071
-0,5	0,5	1,284025	-0,068808	-7,569891
0,5	0,5	1,284025	-0,068808	-2,784807

2.1.2 χ^2 skirstinio aproksimavimas

Tarkime atsitiktinis dydis yra pasiskirstęs pagal χ^2 dėsnį. Jį galima aproksimuoti eksponentinių skirstinių mišiniu (metodas aprašytas 1.1.7 skyrelyje). χ^2 skirstinio tankis:

$$f(x,k)=\frac{x^{\frac{k}{2}-1}}{2^{\frac{k}{2}}\cdot\Gamma\left(\frac{k}{2}\right)\cdot e^{\frac{x}{2}}},x\geq 0 \qquad \textbf{(2.2)}$$

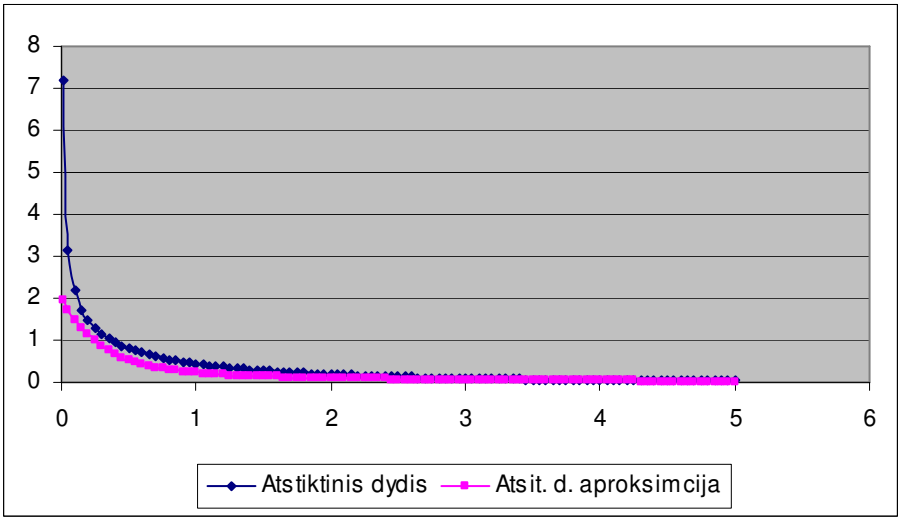
Su laisvės laipsniu $k=1$ pirmieji trys pradiniai momentai:

$$m_1=1, m_2=3 \text{ ir } m_3=15.$$

Pagal (1.33) formules suskaičiuojame eksponentinių skirstinių parametrus ir gauname:

$$\mu_1=3.414214, \mu_2=1,\ p_1=0.414214,\quad p_2=0.585786.$$

Tankio funkcijų grafinis palyginimas pateikiamas 2.2 pav.



2.2 pav. χ^2 ir aproksimuoto dydžio tankio funkcijų palyginimas

Tiriant šio modelio apibrėžimo sritį, empiriškai buvo patikrintos keli galimi laisvės laipsniai *k*. 2.2 lentelėje žaliai pažymėtos celės, kurios netenkina (1.34) nelygybių. Raudonai pažymėtos celės, kurios tenkina šias nelygybes, o χ^2 skirstinio su atitinkamais parametrais aproksimuoti šiuo metodu negalime. Kai *k*=3, abi (1.34) nelygybės yra tenkinamos ir ieškomas parametras μ_2 yra teigiamas.

2.2 lentelė

χ^2 atsit.d. aproksimavimo dviejų eksponentinių atsit. d. mišiniu tyrimas

k	$\frac{m_2}{m_1^2}$	$\frac{m_2}{m_1^2}-\frac{m_3}{6m_1^3}$	μ_2
1	3,00	0,50	1,00
2	2,00	1,00	dalyba iš nulio
3	1,67	1,02	3,00
4	1,50	1,00	dalyba iš nulio
5	1,40	0,98	-3,00
6	1,33	0,96	-1,50

2.2 Aptarnavimo sistemos

2.2.1 Aptarnavimo sistemos modelis su paprastu prioritetu (S_1)

2.2.1.1 Sistemos S_1 aprašymas

Nagrinėsime paprastos sistemos modelį, kuriai yra išvestos analitinės formulės sistemos charakteristikoms skaičiuoti, kad sukalibruotume sukurtą programinę priemonę.

Aptarnavimo sistemoje egzistuoja dviejų tipų paraiškos. Jų aptarnavimo laikai pasiskirstę pagal log-normalųjį dėsnį. Paraiškų pateikimo procesas yra Puasono su parametrais λ_1 ir λ_2. Tarkime, kad pirmojo tipo paraiškos turi paprastą prioritetą prieš antro tipo paraiškas. Sistemoje gali susidaryti dvi eilės abejoms paraiškoms, kurių didžiausi leistini ilgiai l_1 ir l_2 atitinkamai. Suskaičiuosime vidutinį eilės ilgį bei vidutinį laukimo laiką eilėje kiekvienai paraiškų klasei.

Tarkime, kad aptarnavimo laiką aproksimuosime pagal formulę (1.29). Tokios sistemos schema pateikta 2.1 pav.

Nauja paraiška negali būti pradėta aptarnauti, jeigu anksčiau atėjusi paraiška dar nepraėjo visų aptarnavimo fazių.

Tokią sistemą galima apibūdinti Markovo grandine su skaičiąja būsenų aibe ir tolydžiu laiku. Sistemai modeliuoti bus panaudotas sistemos įvykių kalba aprašymas pateiktas prof.Pranevičiaus 1996 m.

Sistemoje gali būti penki įvykiai:

$$E=\{e_1,e_2,e_3,e_4,e_5\},\ \text{čia}$$

e_1 – pirmo tipo paraiška buvo pateikta;

e_2 – antro tipo paraiška buvo pateikta;

e_3 – paraiška baigta aptarnauti pirmoje fazėje su tikimybe p_2;

e_4 – paraiška baigta aptarnauti pirmoje fazėje su tikimybe p_1;

e_5 – paraiška baigta aptarnauti antroje fazėje.

Būsenų aibę apibrėšime taip:

$$N = \{(n_1, n_2, n_3, n_4)\}, n_1 = \overline{0, l_1}; n_2 = \overline{0, l_2}\ ,\ \text{čia}$$

n_1 – pirmo tipo paraiškų skaičius sistemoje;

n_2 – antro tipo paraiškų skaičius sistemoje;

$$n_3 = \begin{cases} 0, \text{ jei sistemoje nera paraisku;} \\ 1, \text{ jei aptarnaujama pirmo tipo paraiska;} \\ 2, \text{ jei aptarnaujama antro tipo paraiska;} \end{cases}$$

$$n_4 = \begin{cases} 0, \text{ jei sistema yra tuscia;} \\ 1, \text{ jei paraiska aptarnaujama pirmoje fazeje;} \\ 2, \text{ jei paraiska aptarnaujama pirmoje fazeje.} \end{cases}$$

Vidutinis paraiškų skaičius eilėje $L^{(1)}$, $L^{(2)}$ ir vidutinis laukimo laikas eilėje $W^{(1)}$, $W^{(2)}$ kiekvienai paraiškos klasei aprašomi formulėmis:

$L^{(j)} = \sum_{n_j=1}^{l_j} \sum_{n_1,n_2,n_3,n_4} n_1 \pi(n_1, n_2, n_3, n_4)$ $W^{(j)} = \frac{L^{(j)}}{\lambda_j}, \quad j = 1,2$	**(2.3)**

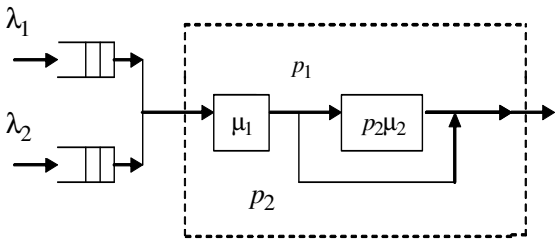
kur $\pi(n_1, n_2, n_3, n_4)$ yra sistemos būsenos stacionari tikimybė. 2.3 pav. pateikiu visų sistemos S_1 įvykių aprašymus.

e_1 : **if** $n_3 = 0$ **then** $n_1 \leftarrow 1$; $n_3 \leftarrow 1$; $n_4 \leftarrow 1$; **else if** $n_1 + n_2 < L$ **then** $n_1 \leftarrow n_1 + 1$; **end if**	e_2 : **if** $n_3 = 0$ **then** $n_2 \leftarrow 1$; $n_3 \leftarrow 2$; $n_4 \leftarrow 1$; **else if** $n_1 + n_2 < L$ **then** $n_2 \leftarrow n_2 + 1$; **end if**

end if **Return** Intens $\leftarrow \lambda_1$ **END** e_1	**end if** **Return** Intens $\leftarrow \lambda_2$ **END** e_2
e_3 : **if** $n_4 = 1$ **if** $n_3 = 1$ **then** $n_1 \leftarrow 0$ **else** $n_2 \leftarrow 0$ **end if** **if** $n_1 > 0$ **then** $n_3 \leftarrow 1$ **else if** $n_2 > 0$ **then** $n_3 \leftarrow 2$ **else** $n_3 \leftarrow 0;\ n_4 \leftarrow 0;$ **end if** **end if** **end if** **Return** Intens $\leftarrow \mu_1 p_2$ **END** e_3	e_4 : **if** $n_4 = 1$ **then** $n_4 \leftarrow 2$; **end if** **Return** Intens $\leftarrow \mu_1 p_1$ **END** e_2 e_5 : **if** $n_4 = 2$ **if** $n_3 = 1$ **then** $n_1 \leftarrow n_1 - 1;$ **else** $n_2 \leftarrow n_2 - 1;$ **end if** **if** $n_1 > 0$ **then** $n_3 \leftarrow 1;\ n_4 \leftarrow 1$ **else if** $n_2 > 0$ **then** $n_3 \leftarrow 2;\ n_4 \leftarrow 1$ **else** $n_3 \leftarrow 0 \quad n_4 \leftarrow 0$ **end if** **end if** **end if** **Return** Intens $\leftarrow \mu_2 p_2$ **END** e_5

2.3 pav. Sistemos S_1 įvykių aprašymai įvykių kalboje

Sukurta programinė priemonė pagal pateiktą įvykių aprašymą generuoja visas galimas sistemos būsenas, perėjimų matricą tarp jų bei būsenų stacionarias tikimybes. Panaudojusi suskaičiuotas tikimybes, programa gali išvesti pageidaujamas sistemos funkcionavimo charakteristikas.



2.4 pav. Aptarnavimo sistema su paprastu prioritetu

2.2.1.2 Sistemos S_1 modeliavimo rezultatai

Jeigu leidžiamas paraiškų eilės ilgis yra begalinis $l_1 = \infty$ ir $l_2 = \infty$, tada $W^{(i)}$ ir $L^{(i)}$ gali būti suskaičiuotos išvestomis analitinėmis formulėmis [10] :

$$L_q^{(1)} = \lambda_1 \cdot W_q^{(1)} = \lambda_1 \cdot \frac{(\lambda_1 + \lambda_2) E^2(X)\left(1 + \nu_X^2\right)}{2\left(1 - \lambda_1 E(X)\right)};$$

$$L_q^{(2)} = \lambda_2 \cdot W_q^{(2)} = \lambda_2 \cdot \frac{(\lambda_1 + \lambda_2) E^2(X)\left(1 + \nu_X^2\right)}{2\left(1 - \lambda_1 E(X)\right)\left(1 - (\lambda_1 + \lambda_2) E(X)\right)}; \quad \textbf{(2.4)}$$

Čia $\nu_X = \frac{\sigma(X)}{E(X)}$, o $E(x)$ ir $\sigma(X)$ yra paraiškos aptarnavimo laiko vidurkis ir vidutinė standartinė nuokrypa.

Analitinio modelio su parametrais

$$\lambda = -0.05, \alpha = 0.9, m_1 = 1.42618, m_2 = 4.57225,\ m_3 = 32.95029, \lambda_1 = 0.2, \lambda_2 = 0.09, \quad l_1 = \infty, l_2 = \infty$$

suskaičiuoti rezultatai:

$$L_q^{(1)} = 0.1855,\ L_q^{(2)} = 0.1423.$$

Skaitinio modelio su parametrais:

$$\lambda_1 = 0.2, \lambda_2 = 0.09,\ \mu_1 = 0.74437, \mu_2 = 0.227796 \quad p_1 = 0.018503, l_1 = 25, l_2 = 25$$

gauti rezultatai:

$$L_q^{(1)} = 0.1855,\ L_q^{(2)} = 0.1424.$$

Kaip matome iš rezultatų, programinės priemonės kalibracija sėkminga, todėl galime ją naudoti modeliuojant ir kitas aptarnavimo sistemas, kurioms nėra išvestų analitinių formulių.

2.2.2 Aptarnavimo sistemos modelis su kokybės tikrinimu (S_2)

2.2.2.1 Sistemos S_2 aprašymas

Tarkime, kad turime aptarnavimo sistemoje S_2 vieną paraiškų srautą (2.5 pav.). Šio srauto intensyvumas yra atsitiktinis dydis ir pasiskirstęs pagal Puasono dėsnį su parametru λ. Paraiškų aptarnavimo laikas taip pat atsitiktinis dydis. Šiame modelyje gali susidaryti dvi paraiškų eilės l_1 ir l_2 prieš kiekvieną iš aptarnavimo įrenginių A ir B. Šioje sistemoje taip pat vykdoma kokybės kontrolė, kuri nukreipia jau aptarnautas paraiškas praeiti abu aptarnavimo įrenginius dar kartą. Suskaičiuokime vidutinį paraiškų skaičių eilėse bei vidutinį laukimo laiką iki aptarnavimo.

Nauja paraiška negali būti pradėta aptarnauti, jeigu prieš tai įėjusi dar nebuvo galutinai aptarnauta aptarnavimo įrenginyje.

Sistemoje egzistuoja tokie įvykiai:

$$E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9\}, \text{ čia}$$

e_1 – paraiška pradėta aptarnauti pirmame aptarnavimo įrenginyje;

e_2 – paraiška baigta aptarnauti A aptarnavimo įrenginyje pirmoje fazėje su tikimybe p_1;

e_3 – paraiška baigta aptarnauti A aptarnavimo įrenginyje pirmoje fazėje su tikimybe p_2;

e_4 – paraiška baigta aptarnauti A aptarnavimo įrenginyje antroje fazėje;

e_5 – paraiška baigta aptarnauti B aptarnavimo įrenginyje pirmoje fazėje su tikimybe p_1;

e_6 – paraiška baigta aptarnauti B aptarnavimo įrenginyje pirmoje fazėje su tikimybe p_2 ir paraiška praėjo kokybės kontrolę;

e_7 – paraiška baigta aptarnauti B aptarnavimo įrenginyje pirmoje fazėje su tikimybe p_2 ir paraiška nepraėjo kokybės kontrolės;

e_8 – paraiška baigta aptarnauti B aptarnavimo įrenginyje antroje fazėje su tikimybe p_2 ir paraiška praėjo kokybės kontrolę;

e_9 – paraiška baigta aptarnauti B aptarnavimo įrenginyje antroje fazėje su tikimybe p_2 ir paraiška nepraėjo kokybės kontrolės.

Galimų būsenų aibė:

$$N = \{(n_1, n_2, n_3, n_4, n_5, n_6)\}, n_1 = \overline{0, l_1}; n_4 = \overline{0, l_2},$$

čia

n_1 – paraiškų skaičius pirmoje eilėje;

$$n_2 = \begin{cases} 0, & \text{jei A irenginio pirma faze yra tuscia;} \\ 1, & \text{jei paraiska yra A irenginio pirmoje fazeje.} \end{cases}$$

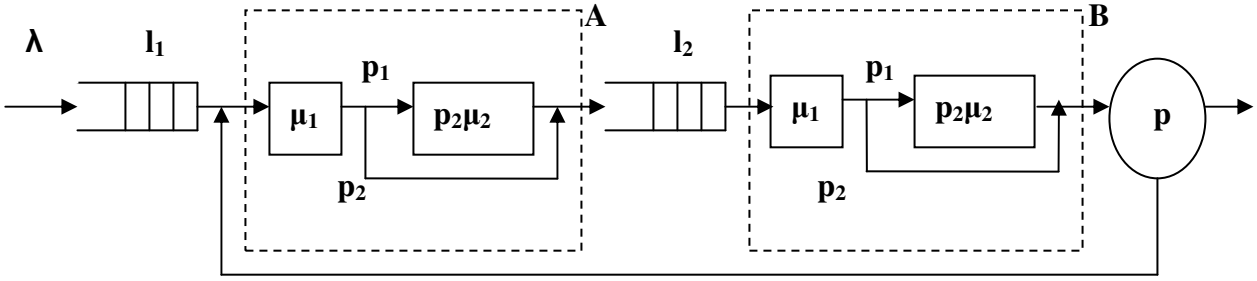
$$n_3 = \begin{cases} 0, & \text{jei A irenginio antra faze yra tuscia;} \\ 1, & \text{jei paraiska yra A irenginio antroje fazeje.} \end{cases}$$

n_4 – paraiškų skaičius antroje eilėje;

$$n_5 = \begin{cases} 0, & \text{jei B irenginio pirma faze yra tuscia;} \\ 1, & \text{jei paraiska yra B irenginio pirmoje fazeje.} \end{cases}$$

$$n_6 = \begin{cases} 0, & \text{jei B irenginio antra faze yra tuscia;} \\ 1, & \text{jei paraiska yra B irenginio antroje fazeje.} \end{cases}$$

Aptarnavimo laiką aproksimuosiu išraiška pateikta teorinėje dalyje (1.29). Sistemos schema pateikta 2.5 pav.



2.5 pav. Aptarnavimo sistema S_2 su kokybės kontrole

Vidutinis eilės ilgis $L^{(1)}$ ir vidutinis paraiškos laukimo laikas $W^{(1)}$ pirmoje eilė yra suskaičiuojamas pagal pateiktas formules:

$$L^{(j)} = \sum_{m_1=1}^{l_1} \sum_{n_1,n_2,n_3,n_4,n_5,n_6} m_1 \pi\left(n_1,n_2,n_3,n_4,n_5,n_6\right)$$
$$W^{(j)} = \frac{L^{(1)}}{\lambda_1}, \quad j = 1,2 \qquad \textbf{(2.5)}$$

čia $\pi\left(n_1,n_2,n_3,n_4,n_5,n_6\right)$ yra sistemos būsenos stacionari tikimybė. 2.6 pav. pateikti visi sistemos įvykių aprašymai įvykiu kalba.

e_1 : **if** $n_2 = 0$ *and* $n_3 = 0$ *and* $\sum_{i=1}^{6} n_i < L$

then $n_2 \leftarrow n_2 + 1$;

else if $\sum_{i=1}^{6} n_i < L$

then $n_1 \leftarrow n_1 + 1$

end if

end if

Return $Intens \leftarrow \lambda_1$

END e_1

e_2 : **if** $n_2 > 0$

then $n_3 \leftarrow n_3 + 1$; $n_2 \leftarrow n_2 - 1$;

end if

Return $Intens \leftarrow \mu_1 p_1$

END e_2

e_3 : **if** $n_2 > 0$

if $n_5 < 1$ *and* $n_6 < 1$

then $n_5 \leftarrow n_5 + 1$

else $n_4 \leftarrow n_4 + 1$

end if

if $n_1 > 0$ **then** $n_1 \leftarrow n_1 - 1$;

else $n_2 \leftarrow n_2 - 1$

end if

end if

Return $Intens \leftarrow \mu_1 p_2$

END e_3

e_4 : **if** $n_3 > 0$

if $n_5 < 1$ *and* $n_6 < 1$

then $n_5 \leftarrow n_5 + 1$

else $n_4 \leftarrow n_4 + 1$

end if

if $n_1 > 0$ **then**

$n_1 \leftarrow n_1 - 1$; $n_2 \leftarrow n_2 + 1$; $n_3 \leftarrow n_3 - 1$;

else $n_3 \leftarrow n_3 - 1$

end if

End if

Return $Intens \leftarrow \mu_2 p_2$

END e_4

e_5 : **if** $n_5 > 0$

then $n_6 \leftarrow n_6 + 1$; $n_5 \leftarrow n_5 - 1$;

end if

Return $Intens \leftarrow \mu_1 p_1$

END e_5

e_6 : **if** $n_5 > 0$

if $n_4 > 0$ **then** $n_4 \leftarrow n_4 - 1$;

else $n_5 \leftarrow n_5 - 1$;

end if

end if

Return $Intens \leftarrow \mu_1 p_2 P$

	END e_6

e_7 : **if** $n_5 > 0$
if $n_4 > 0$ **then** $n_4 \leftarrow n_4 - 1;$
else $n_5 \leftarrow n_5 - 1;$
end if
if $n_2 = 0$ *and* $n_3 = 0$
then $n_2 \leftarrow n_2 + 1;$
else $n_1 \leftarrow n_1 + 1;$
end if
end if
Return $Intens \leftarrow \mu_1 p_2 (1 - P)$
END e_7
	e_8 : **if** $n_6 > 0$
	if $n_4 > 0$ **then**
	$n_4 \leftarrow n_4 - 1;\ n_5 \leftarrow n_5 + 1;\ n_6 \leftarrow n_6 - 1;$
	else $n_6 \leftarrow n_6 - 1;$
	end if
	end if
	Return $Intens \leftarrow \mu_2 p_2 P$
	END e_8

e_9 : **if** $n_6 > 0$
if $n_4 > 0$ **then**
$n_4 \leftarrow n_4 - 1;\ n_5 \leftarrow n_5 + 1;\ n_6 \leftarrow n_6 - 1;$
else $n_6 \leftarrow n_6 - 1;$
end if
if $n_2 = 0$ *and* $n_3 = 0$
then $n_2 \leftarrow n_2 + 1;$
else $n_1 \leftarrow n_1 + 1;$
end if
end if
Return $Intens \leftarrow \mu_2 p_2 (1 - P)$
END e_9	

2.6 pav. Sistemos S_2 įvykių aprašymai įvykių kalboje

Sukurta programinė priemonė pagal pateiktą įvykių aprašymą generuoja visas galimas sistemos būsenas, perėjimų matricą tarp jų bei būsenų stacionarias tikimybes. Naudojant suskaičiuotas tikimybes galima gauti pageidaujamas sistemos funkcionavimo charakteristikas.

2.2.2.2 Sistemos S_2 modeliavimo rezultatai

Naudojant programinę priemonę su pradiniais parametrais:

$$m_1 = 1.42618, m_2 = 4.57225,\ m_3 = 32.95029,\ \lambda = 1, P = 0.9,\ \text{maksimali eilė} - 20.$$

Buvo gauti tokie skaitinio modeliavimo rezultatai:

$$L^{(1)} = 8.3022,\ L^{(2)} = 8.3230;\ W^{(1)} = 8.3022.$$

2.3 Finansų aktyvų dinamikos modeliavimas

Pateiksiu finansinių aktyvų vertės dinamikos modeliavimo algoritmo, pagrįsto Markovo procesu su tolydžiuoju laiku ir skaičiąja būsenų aibe, pagrindinius principus.

2.3.1 Finansų aktyvų dinamikos modelio aprašymas

Panaudosiu įvykių kalbą akcijos kainų būsenoms ir perėjimo intensyvumams tarp jų generuoti. Įvertindami pasirinktos akcijos istorinius kainų duomenimis, aproksimuojame kainų didėjimo ir mažėjimo intensyvumus, remdamiesi gautais fazių intensyvumais pagal (1.33) formules. Metodas vertina kiek dienų aktyvo kainos didėjo ir kiek mažėjo. Pokytis vertinamas tik tada, kai yra ne mažesnis už vidutinį aktyvo dienos pokytį Δ, suskaičiuotą įvertinus pradinius duomenis. Jeigu per dieną aktyvo kaina pakinta daugiau negu po vieną vidutinį dienos pokytį $\frac{|kaina(t+1) - kaina(t)|}{\Delta} = k > 1$, tada fiksuojamas k dydžio kainos pokytis.

Galimų įvykių aibė susideda iš 10 įvykių:

$$E = \{e_1^u, e_2^u, e_3^u, e_4^u, e_5^u, e_1^d, e_2^d, e_3^d, e_4^d, e_5^d\}$$

čia

e_1^u – baigėsi kainos didėjimo etapas pirmojoje fazėje su tikimybe p_1^u;

e_2^u – baigėsi kainos didėjimo etapas pirmojoje fazėje su tikimybe p_2^u ir prasidėjo kainos mažėjimo etapas pirmoje fazėje;

e_3^u – baigėsi kainos didėjimo etapas pirmojoje fazėje su tikimybe p_2^u ir prasidėjo kainos didėjimo etapas pirmoje fazėje;

e_4^u – baigėsi kainos didėjimo etapas antroje fazėje ir prasidėjo kainos mažėjimo etapas pirmoje fazėje. Įvertinant akcijos trendą, antros fazės intensyvumą $p_2^U \mu_2^U$, padauginsime iš papildomo nario: $Ddaugiklis = \frac{m_1^D}{m_1^U + m_1^D}$;

e_5^u – baigėsi kainos didėjimo etapas antroje fazėje ir prasidėjo kainos didėjimo etapas pirmoje fazėje; Įvertinant akcijos trendą, antros fazės intensyvumą $p_2^U \mu_2^U$, padauginsime iš papildomo nario: $Udaugiklis = \frac{m_1^U}{m_1^U + m_1^D}$;

e_1^d – baigėsi kainos mažėjimo etapas pirmojoje fazėje su tikimybe p_1^d;

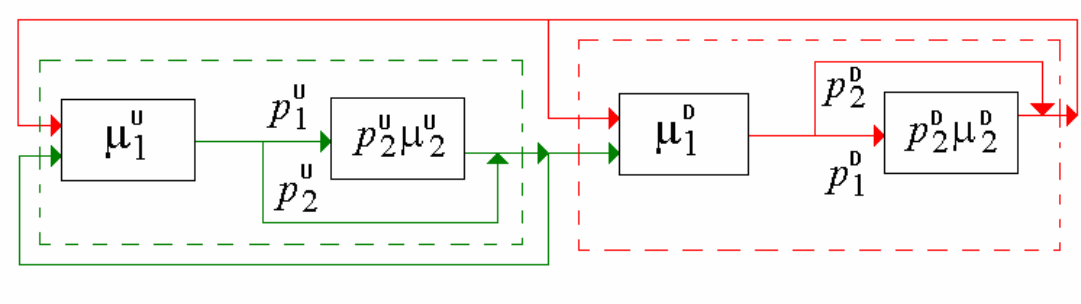
e_2^d – baigėsi kainos mažėjimo etapas pirmojoje fazėje su tikimybe p_2^d ir prasidėjo kainos mažėjimo etapas pirmoje fazėje;

e_3^d – baigėsi kainos mažėjimo etapas pirmojoje fazėje su tikimybe p_2^d ir prasidėjo kainos didėjimo etapas pirmoje fazėje;

e_4^d – baigėsi kainos mažėjimo etapas antroje fazėje ir prasidėjo kainos mažėjimo etapas pirmoje fazėje. Įvertinant akcijos trendą, antros fazės intensyvumą $p_2^D \mu_2^D$, padauginsime iš papildomo nario: $Ddaugiklis = \frac{m_1^D}{m_1^U + m_1^D}$;

e_5^d – baigėsi kainos mažėjimo etapas antroje fazėje ir prasidėjo kainos didėjimo etapas pirmoje fazėje. Įvertinant akcijos trendą, antros fazės intensyvumą $p_2^D \mu_2^D$, padauginsime iš papildomo nario: $Udaugiklis = \frac{m_1^U}{m_1^U + m_1^D}$.

Modelio grafinė struktūra susideda iš dviejų etapų – didėjimo ir mažėjimo, kuri kiekvieną sudaro dvi fazės (2.7 pav.).



2.7 pav. Finansinių aktyvų modelio diagrama

Perėjimo intensyvumų aibė:

$$Intens = \{\mu_1^u, p_2^u\mu_2^u, \mu_1^d, p_2^d\mu_2^d\},$$

čia

μ_1^u - intensyvumas, su kuriuo kainos didėjimas baigėsi pirmojoje fazėje;

$\mu_1^u p_2^u$ - intensyvumas, su kuriuo kainos didėjimas baigėsi antrojoje fazėje;

μ_1^d - intensyvumas, su kuriuo kainos mažėjimas baigėsi pirmojoje fazėje;

$\mu_1^d p_2^d$ - intensyvumas, su kuriuo kainos mažėjimas baigėsi antrojoje fazėje;

Apibrėšime kainų būsenų aibę. Tarkime, kad yra žinomos akcijos didžiausioji ir mažiausioji kainos, t.y. visos galimos kainos priklauso atkarpai $[S_{\min}, S_{\max}]$. Suskaičiuojamas vidutinis aktyvo vertės pokytis per dieną nagrinėjamu laikotarpiu – Δ. Atkarpa $[S_{\min}, S_{\max}]$ suskaidoma i $n = \left[\frac{S_{\max} - S_{\min}}{\Delta}\right] + 1$ intervalų.

Šias kainų būsenas surašysime didėjančia tvarka ir gautąją aibę pažymėsime taip:

$$I = \{S_{\min} + \Delta, S_{\min} + 2\cdot\Delta, \cdots, S_{\min} + n\cdot\Delta = S_{\max}\}.$$

Modeliuojant akcijų kainų procesą nepakanka žinoti tik galimų kainų aibės, bet reikia žinoti ir kurioje fazėje yra modeliuojama kaina. Todėl galimą būsenų erdvę apibrėšime kaip trimačių vektorių erdvę:

$$B = \{(b_1, b_2, b_3)\},$$

čia

$b_1 \in I$

$$b_2 = \begin{cases} 0, \text{ jei akcijos kaina šiuo metu mažėja;} \\ 1, \text{ jei akcijos kaina didėja pirmojoje fazėje;} \\ 2, \text{ jei akcijos kaina didėja antrojoje fazėje.} \end{cases}$$

$$b_3 = \begin{cases} 0, \text{ jei akcijos kaina šiuo metu didėja;} \\ 1, \text{ jei akcijos kaina mažėja pirmojoje fazėje;} \\ 2, \text{ jei akcijos kaina mažėja antrojoje fazėje.} \end{cases}$$

Norint sugeneruoti galimų būsenų erdvę ir perėjimo intensyvumų matricą, kainų dinamikos procesą reikia aprašyti įvykių kalboje [5]. 2.8 pav. pateikiami visų modelio įvykių aprašymai.

e_1^u: **if** $b_2 = 1$ **then** $b_2 \leftarrow 2$ **end if** Intens $\leftarrow \mu_1^u p_1^u$ **end** e_1^u	e_1^d: **if** $b_3 = 1$ **then** $b_3 \leftarrow 2$ **end if** Intens $\leftarrow \mu_1^d p_1^d$ **end** e_1^d
e_2^u: **if** $b_2 = 1$ **then** $b_3 \leftarrow 1;\ b_2 \leftarrow 0;\ b_1 \leftarrow b_1 + \Delta;$ **end if** Intens $\leftarrow \mu_1^u p_2^u \cdot Ddaugiklis$ **end** e_2^u	e_3^u: **if** $b_2 = 1$ **and** $b_1 + 2\Delta \le S_{\max}$ **then** $b_1 \leftarrow b_1 + \Delta;$ **end if** Intens $\leftarrow \mu_1^u p_2^u \cdot Udaugiklis$ **end** e_3^u
e_2^d: **if** $b_3 = 1$ **and** $b_1 - 2 \cdot \Delta \ge S_{\min}$ **then** $b_1 \leftarrow b_1 - \Delta;$ **end if** Intens $\leftarrow \mu_1^d p_2^d \cdot Ddaugiklis$ **end** e_2^d	e_3^d: **if** $b_3 = 1$ **then** $b_2 \leftarrow 1;\ b_3 \leftarrow 0;\ b_1 \leftarrow b_1 - \Delta;$ **end if** Intens $\leftarrow \mu_1^d p_2^d \cdot Udaugiklis$ **end** e_3^d
e_4^u: **if** $b_2 = 2$ **then** $b_3 \leftarrow 1;\ b_2 \leftarrow 0;\ b_1 \leftarrow b_1 + \Delta;$ **end if** Intens $\leftarrow \mu_2^u p_2^u \cdot Ddaugiklis$ **end** e_4^u	e_5^u: **if** $b_2 = 2$ **and** $b_1 + 2\Delta \le S_{\max}$ **then** $b_2 \leftarrow 1;\ b_1 \leftarrow b_1 + \Delta;$ **end if** Intens $\leftarrow \mu_2^u p_2^u \cdot Udaugiklis$ **end** e_5^u
e_4^d: **if** $b_3 = 2$ **and** $b_1 - 2 \cdot \Delta \ge S_{\min}$	e_5^d: **if** $b_3 = 2$

then $b_1 \leftarrow b_1 - \Delta;\ b_3 \leftarrow 1$	**then** $b_2 = 1;\ b_3 = 0;\ b_1 \leftarrow b_1 - \Delta;$
end if	**end if**
Intens $\leftarrow \mu_2^d\, p_2^d \cdot Ddaugiklis$	Intens $\leftarrow \mu_2^d\, p_2^d \cdot Udaugiklis$
end e_4^d	**end** e_5^d

2.8 pav. Finansų aktyvų dinamikos modelio įvykių aprašymas įvykių kalboje

Sukurtoji programinė priemonė pagal įvykių aprašymą generuoja galimų būsenų erdvę, perėjimo intensyvumų matricą, sudaro lygčių sistemą stacionarioms Markovo proceso būsenų tikimybėms suskaičiuoti ir jas suranda. Konkrečios akcijos kainos tikimybė suskaičiuojama pagal formulę (2.6)

$$P(k) = \sum_B \pi\left(S_{\min} + k \cdot \Delta, b_2, b_3\right),\ k = 0, \cdots, n \qquad \textbf{(2.6)}$$

čia $\pi\left(S_{\min} + k \cdot \Delta, b_2, b_3\right)$ yra *k*-tosios būsenos stacionarioji tikimybė.

2.3.2 Finansų aktyvų dinamikos modeliavimo rezultatai

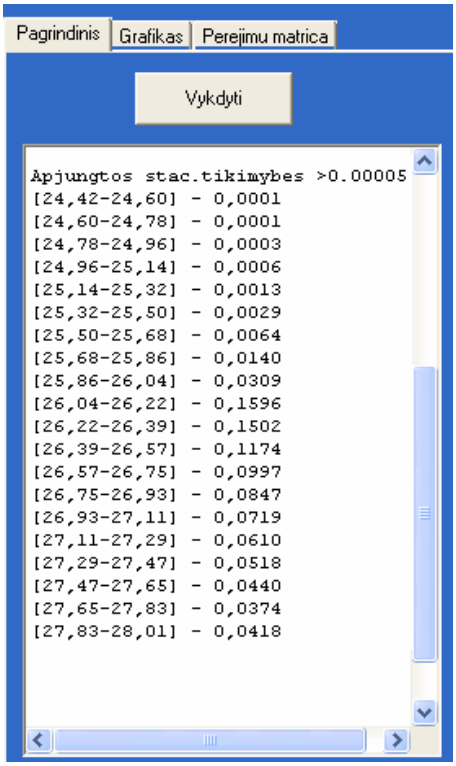
Nagrinėkime keturių mėnesių laikotarpio „*Microsoft Corporation*“ akcijų kainas. Kainų ekstremumai: $S_{\max} = 28{,}10, \mathrm{S}_{\min} = 24{,}15$. Vidutinis akcijos dienos kainos pokytis $\Delta = 0{,}1796$. Tad akcijos kainos suskirstomos į 22 intervalus. Įvertinus akcijos kainos didėjimo ir mažėjimo dienų skaičius 2.2.1 skyrelyje aprašytu metodu, buvo gauti tokie pradiniai momentai:

$$m_1^U = 2{,}2402;\ m_2^U = 11{,}6950;\ m_3^U = 85{,}7250;$$
$$m_1^D = 1{,}8321;\ m_2^D = 6{,}9676;\ m_3^D = 37{,}4190;$$

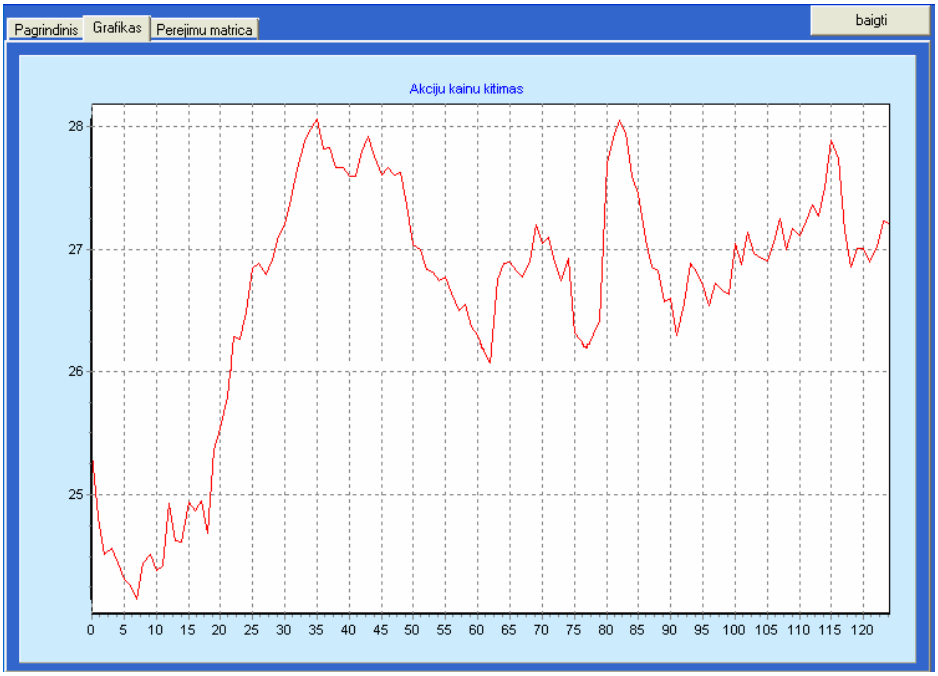
Pasinaudojus (6) formulėmis gauti eksponentinių fazių intensyvumai:

$$\mu_1^U = 0{,}4014;\ \mu_2^U = 1{,}2384;\ \mathrm{p}_1^{\mathrm{U}} = 0{,}2373;$$
$$\mu_1^D = 0{,}5381;\ \mu_2^D = 0{,}3358;\ \mathrm{p}_1^{\mathrm{D}} = 0{,}0088;$$

Programa pagal 2.7 pav. pavaizduotą schemą bei pateiktą įvykių aprašymą, sumodeliuoja 88 būsenų aibę bei perėjimų intensyvumų matricą. Šiais rezultatais remdamasi, suskaičiuoja būsenų stacionarias tikimybes. Į programos langus išvedami modeliavimo rezultatai – 2.9 pav. ir 2.10 pav.



2.9 pav. Pirmasis programos langas



2.10 pav. Antrasis programos langas

2.3.3 Opcionų įkainojimas

Darbe pasiūlytas metodas, kaip akcijų kainų dinamiką aprašyti Markovo procesu. Pateikus pradinius duomenis, nuskaitytus iš tekstinės bylos, programa suskaido galimas aktyvo kainas į intervalus 2.3.1 skyrelyje aprašytu metodu. Kiekvienas intervalas suvokiamas kaip būsena ir suskaičiuojamos jų stacionarios tikimybės. Programos veikimas pavaizduotas (3.1 pav.) ir (3.3 pav.). Suskaičiavus stacionarias tikimybes, yra panaudojamas 1.2.2 skyrelyje pateiktas opcionų įvertinimo algoritmas. Kiekvieno intervalo (būsenos) vidutinė kaina yra traktuojama, kaip opciono įvykdymo kaina ir suskaičiuojama tokio opciono tikėtina vertė bei dabartinė vertė (diskonto palūkanų norma ir nagrinėjamas laikotarpis įvedama programos lange).

Nagrinėkime kompanijos „*McDonalds*“ keturių metų akcijos kainas. Į programos langą įvesti tokie parametrai:

Palūkanų norma – 5%;

Laikotarpis – 10 metų.

Buvo gauti tokie modeliuoto vertybinio popieriaus opciono įvertinimo rezultatai:

Opciono ikainojimas:

Ivykdymo kaina	Tiketina verte	Diskontuota verte
34,71	41,44	25,44
35,34	38,42	23,59
35,97	35,58	21,85
36,59	32,96	20,23
37,22	30,52	18,74
37,85	28,26	17,35
38,47	26,17	16,07
39,10	24,23	14,88
39,73	22,43	13,77
40,35	20,77	12,75
40,98	19,22	11,80
41,61	17,79	10,92
42,23	16,47	10,11
42,86	15,24	9,35
43,49	14,10	8,65
44,11	13,04	8,01
44,74	12,06	7,40
45,37	11,15	6,85
45,99	10,31	6,33
46,62	9,53	5,85
47,25	8,81	5,41
47,87	8,14	5,00
48,50	7,52	4,62
49,13	6,95	4,26
49,75	6,41	3,94
50,38	5,92	3,63
51,01	5,46	3,35
51,63	5,04	3,09
52,26	4,65	2,85
52,89	4,28	2,63
53,51	3,95	2,42
54,14	3,64	2,23
54,77	3,35	2,05
55,39	3,08	1,89
56,02	2,83	1,74
56,65	2,60	1,60
57,27	2,39	1,47
57,90	2,19	1,35
58,53	2,01	1,23
59,16	1,84	1,13
59,78	1,69	1,03
60,41	1,54	0,95
61,04	1,41	0,86
61,66	1,28	0,79
62,29	1,17	0,72
62,92	1,06	0,65
63,54	0,97	0,59
64,17	0,88	0,54
64,80	0,79	0,49
65,42	0,72	0,44
66,05	0,64	0,40
66,68	0,58	0,35
67,30	0,52	0,32
67,93	0,46	0,28
68,56	0,41	0,25
69,18	0,36	0,22
69,81	0,32	0,20
70,44	0,28	0,17
71,06	0,24	0,15

```
71,69          0,21          0,13
72,32          0,17          0,11
72,94          0,15          0,09
73,57          0,12          0,07
74,20          0,09          0,06
74,82          0,07          0,04
75,45          0,05          0,03
76,08          0,03          0,02
76,70          0,02          0,01
77,33          0,00          0,00
-----------------------------------------------
```

2.11 pav. „*McDonalds*“ akcijų opciono įkainojimas

2.3.4 Programinės įrangos veikimo laiko tyrimas

Programos veikimo laiką, modeliuojant finansų rinkų aktyvų dinamiką ir įkainuojant opcionus, labiausiai lemia ne akcijų kainų kiekis pradiniame duomenų faile, o sugeneruotų būsenų (intervalų) skaičius. Kaip pastebime iš 2.3 lentelės, pradinių kainų kiekį didinant sparčiai, programos veikimo laikas keičiasi nežymiai.

2.3 lentelė

Finansinių aktyvų dinamikos modeliavimo trukmė, keičiant pradinių kainų kiekį

MSFT akcijos:		
akcijų kainų skaičius	**Būsenų skaičius**	**veikimo laikas, s**
125	22	0,49
375	22	0,45
625	22	0,42
875	22	0,42
1125	22	0,49

Dow Jones indeksas:		
Akcijų kainų skaičius	**Būsenų skaičius**	**veikimo laikas, s**
250	41	0,95
750	41	0,98
1250	41	0,90
1750	41	0,94
2250	41	0,95

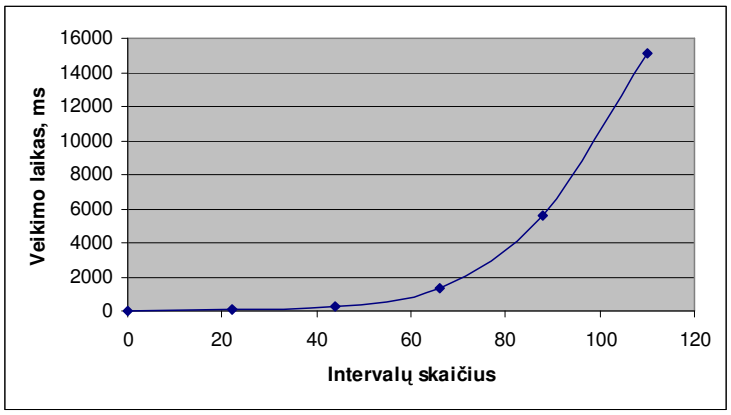
Iš 2.4 lentelėje pateiktų programinės įrangos modeliavimo trukmės duomenų, pastebime, kad programos veikimo laikas auga sparčiai, keičiant generuojamų būsenų (intervalų) skaičių. Būsenų skaičius priklauso nuo minimalios ir maksimalios aktyvo kainos per nagrinėjamą laikotarpį bei vidutinio aktyvo dienos pokyčio Δ (2.3.1 skyrelis). Didinant būsenų skaičių, augančio veikimo laiko grafinis pavaizdavimas 2.12 pav. ir 2.13 pav.

2.4 lentelė

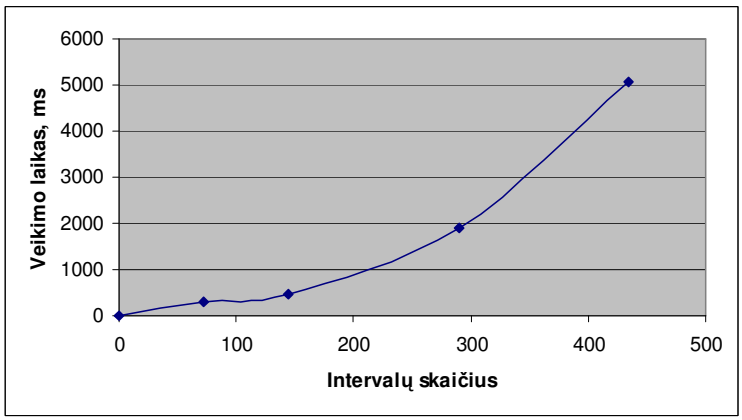
Vertybinių popierių rinkų aktyvų dinamikos modeliavimo trukmė, keičiant būsenų skaičių

MSFT akcijos:		
akcijų kainų skaičius	**Būsenų skaičius**	**veikimo laikas, s**
125	22	0,45
125	44	2,94
125	66	13,63
125	88	55,59
125	110	151,35

Dow Jones indeksas:		
Akcijų kainų skaičius	**Būsenų skaičius**	**veikimo laikas, s**
250	72	2,89
250	145	4,72
250	290	18,87
250	435	50,80
250	544	58,79



2.12 pav. „*Microsoft Corporation*“ akcijų kainų modeliavimo trukmė, keičiant būsenų skaičių

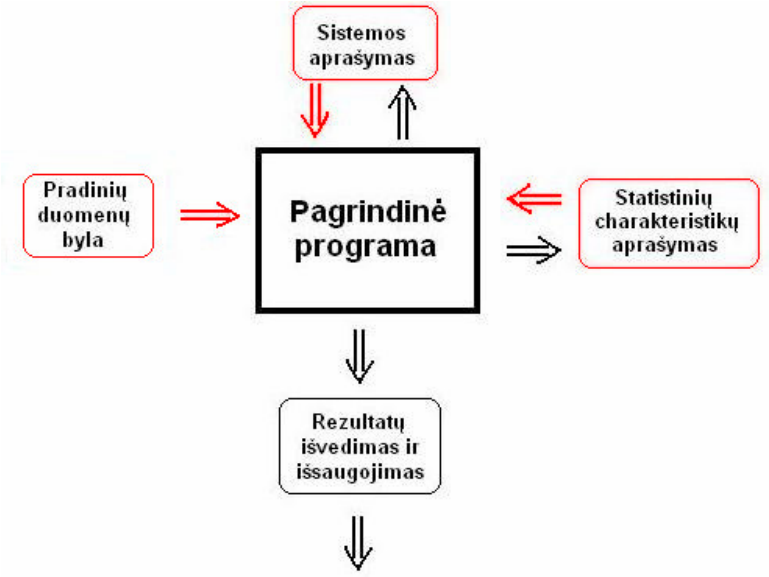


2.13 pav. „*Dow Jones*“ akcijų indekso modeliavimo trukmė, keičiant būsenų skaičių

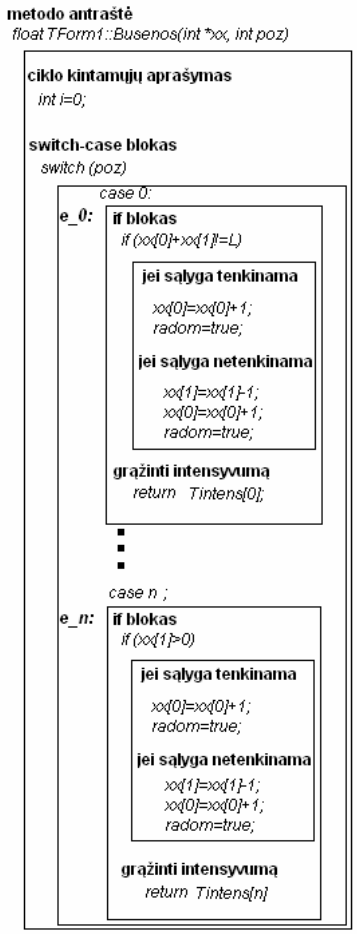
3. PROGRAMINĖ REALIZACIJA

3.1 Programinės įrangos aprašymas

Magistro darbe buvo sukurta universali programinė įranga, skirta modeliuoti aptarnavimo sistemas bei vertybinių popierių rinkas. Jos veikimo schema pateikiama (3.1 pav.) ir (3.3 pav.). Sistemų aprašymą *C++* kalba gali pateikti bet kuris programinės priemonės vartotojas (trys sistemos jau yra sukurtos ir aprašytos 2.2 ir 2.3 skyreliuose), turintis bent minimalias C++ programavimo kalbos žinias. Sistemos funkcionavimas turi būti aprašytas įvykių kalba (3.2 pav.): jei įvyksta įvykis e1 – daromi vieni būsenos atitinkami pakitimai, jei e2 – daromi kiti būsenos pakitimai ir t.t.



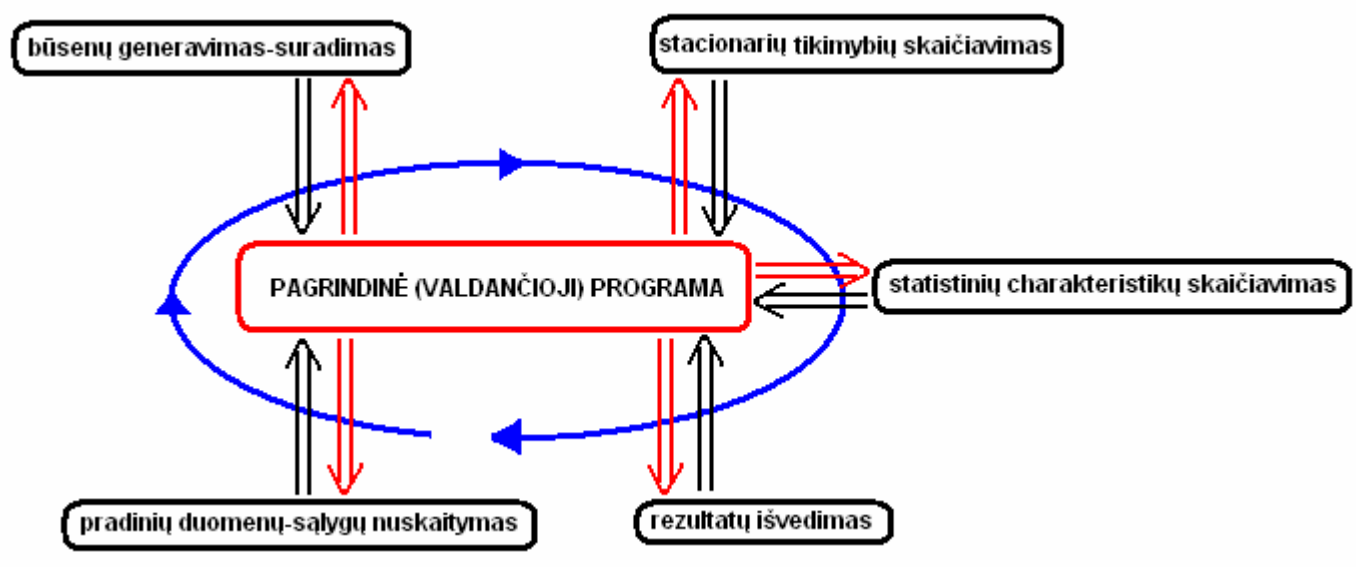
3.1 pav. Programos struktūra



3.2 pav. Sistemos aprašymo metodo struktūra.

Nuskaičiusi pradinių duomenų bylą, programa suskaičiuoja tarpinius, dviejų fazių aproksimacijoje naudojamus, parametrus ir modeliuoja būsenų aibę kartu su perėjimų matrica. Sugeneravusi sistemos modelio būsenas, programa suskaičiuoja stacionarias tikimybes. Statistinių charakteristikų skaičiavimo metodai labai skiriasi įvairioms sistemoms, tad jų sukūrimu turi rūpintis vartotojas. Priede yra pateikiami

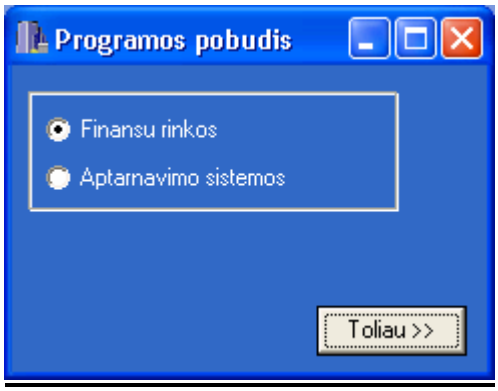
keli statistinių charakteristikų skaičiavimo algoritmai. Paskutinis programos žingsnis – rezultatų išvedimas į ekraną. Vėliau pakeitę pradinius duomenis, programos veikimo ciklą vėl galime pakartoti (3.3 pav.)



3.3 pav. Valdančiosios programos veikimo schema

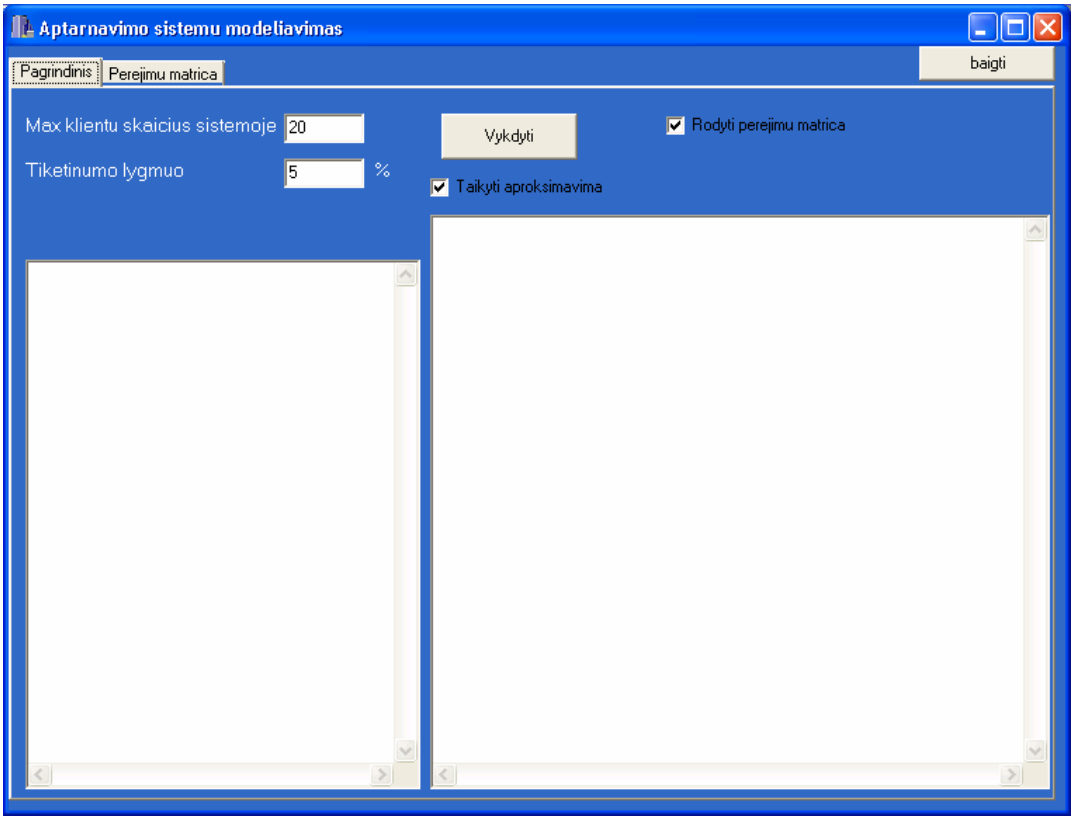
3.2 Programinės įrangos langai

a. Parinkčių langas



3.4 pav. Programinės įrangos pobūdžio parinkties langas

b. Pirmasis programos lango puslapis modeliuojant aptarnavimo sistemas



3.5 pav. Programinės įrangos pagrindinis langas modeliuojant aptarnavimo sistemas

Mytukų paaiškinimai:

„*Vykdyti*“ – programos vykdymo pradžia;

„*Baigti*“ – uždaryti programos langą.

Laukų paaiškinimai:

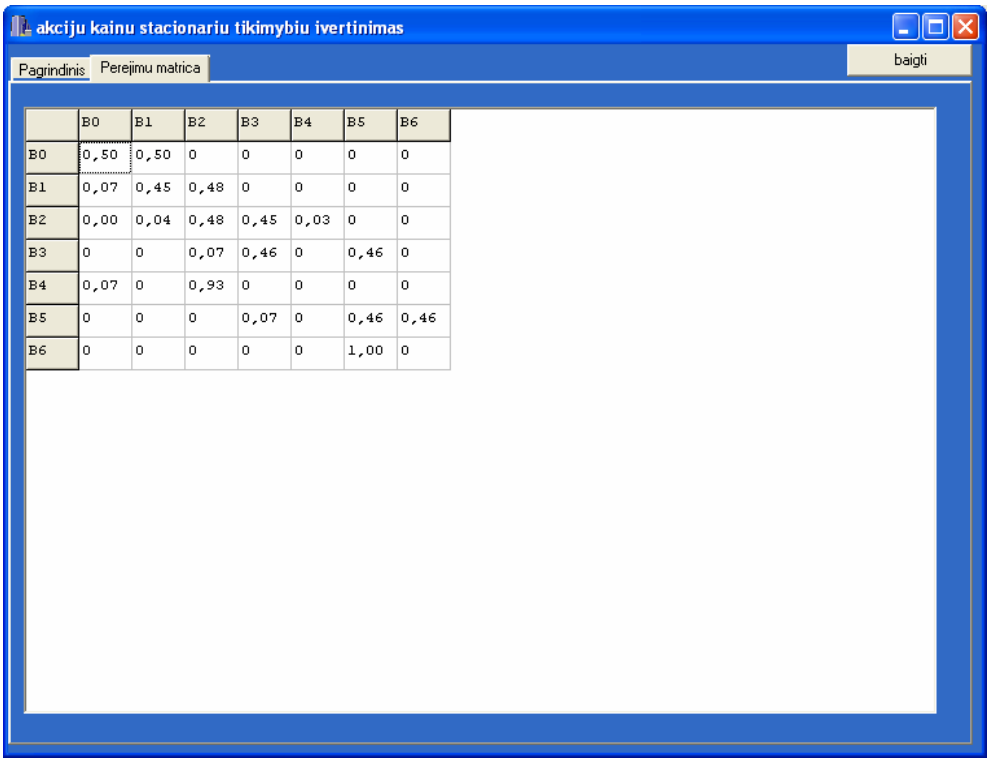
„*Max klientų skaičius sistemoje*“ – bendras leidžiamas paraiškų skaičius sistemoje, įskaitant tuos kurie yra aptarnaujami ar stovi paraiškų eilėje. Įvedamas skaičius, išsaugomas kintamuoju L;

„*Tikėtinumo lygmuo*“ – dešiniajame programos rezultatų lange išvedama charakteristika, išvardinanti būsenas, kurių stacionarių tikimybių suma viršija įvedamą *Tikėtinumo lymenį*;

„*Taikyti aproksimavimą*“ – pažymėjus šią varnelę, bus taikomas aproksimavimas, aptarnavimo trukmės atsitiktiniam dydžiui. Reikia atkrepti dėmesį, kad tuo atveju, pradinių duomenų byloje *Intensyvumų* eilutėje, pirmi trys skaičiai turi būti aproksimuojamo atsitiktinio dydžio pradiniai momentai. Numatytasis aproksimavimo metodas yra „*Trijų momentų lyginimo*“ (1.1.7 skyrelis). Jeigu šio metodo taikyti negalima, dėl (1.34) nelygybių, taikomas „*Dviejų momentų lyginimo*“ metodas;

„*Rodyti perėjimų matricą*“ – pažymėjus šia varnelę, antrajame programos lango puslapyje išvedama perėjimo tarp būsenų matrca.

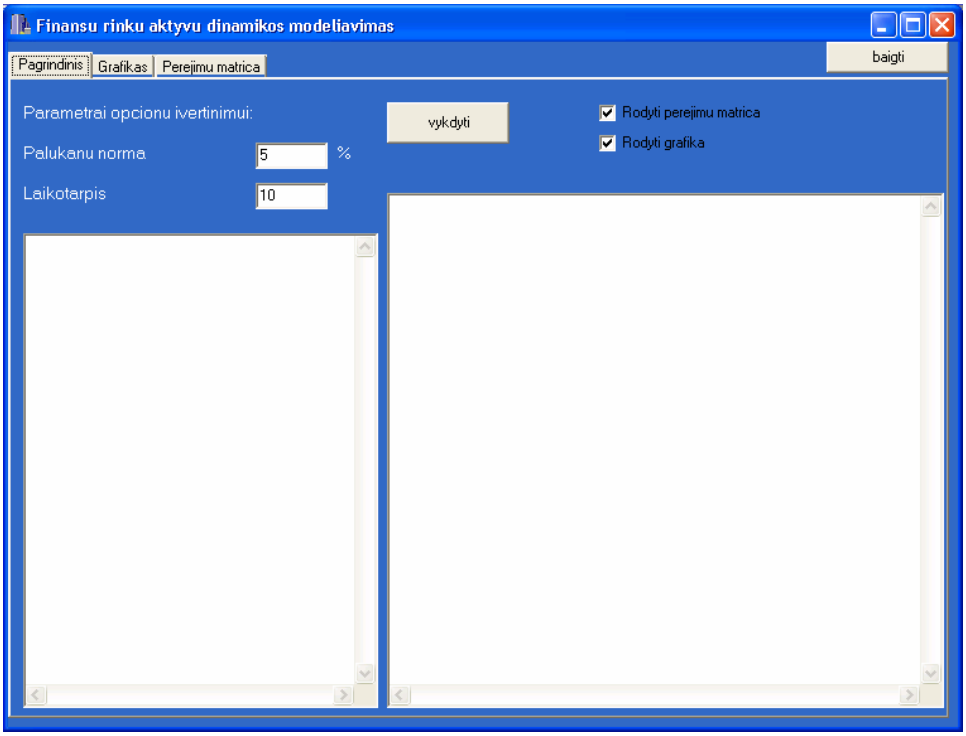
c. Antrasis programos lango puslapis modeliuojant aptarnavimo sistemas



	B0	B1	B2	B3	B4	B5	B6
B0	0,50	0,50	0	0	0	0	0
B1	0,07	0,45	0,48	0	0	0	0
B2	0,00	0,04	0,48	0,45	0,03	0	0
B3	0	0	0,07	0,46	0	0,46	0
B4	0,07	0	0,93	0	0	0	0
B5	0	0	0	0,07	0	0,46	0,46
B6	0	0	0	0	0	1,00	0

3.6 pav. Programinės įrangos antras langas modeliuojant aptarnavimo sistemas

d. Pirmasis programos lango puslapis modeliuojant vertybinių popierių dinamiką



3.7 pav. Programinės įrangos pagrindinis langas modeliuojant vertybinių popierių dinamiką

Mytukų paaiškinimai:

„*Vykdyti*“ – programos vykdymo pradžia;

„*Baigti*“ – uždaryti programos langą.

Laukų paaiškinimai:

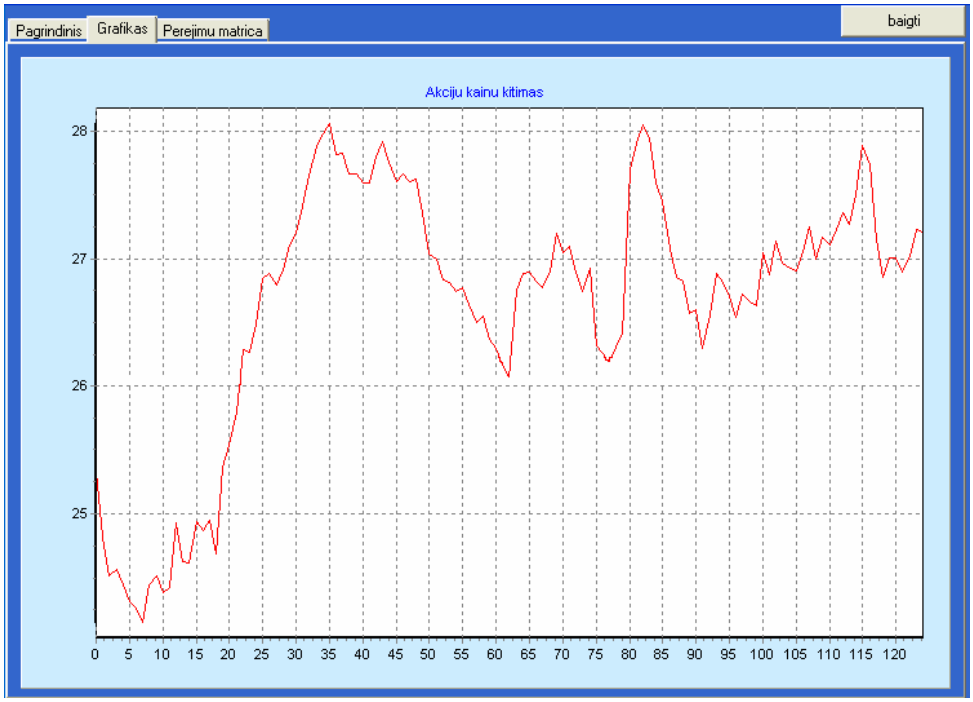
„*Palūkanų norma*“ – palūkanų norma diskontui;

„*Laikotarpis*“ – diskonto laikotarpis;

„*Rodyti grafiką*“ – pažymėjus šia varnelę, antrajame programos lango puslapyje pavaizduojama aktyvo praeities dinamika;

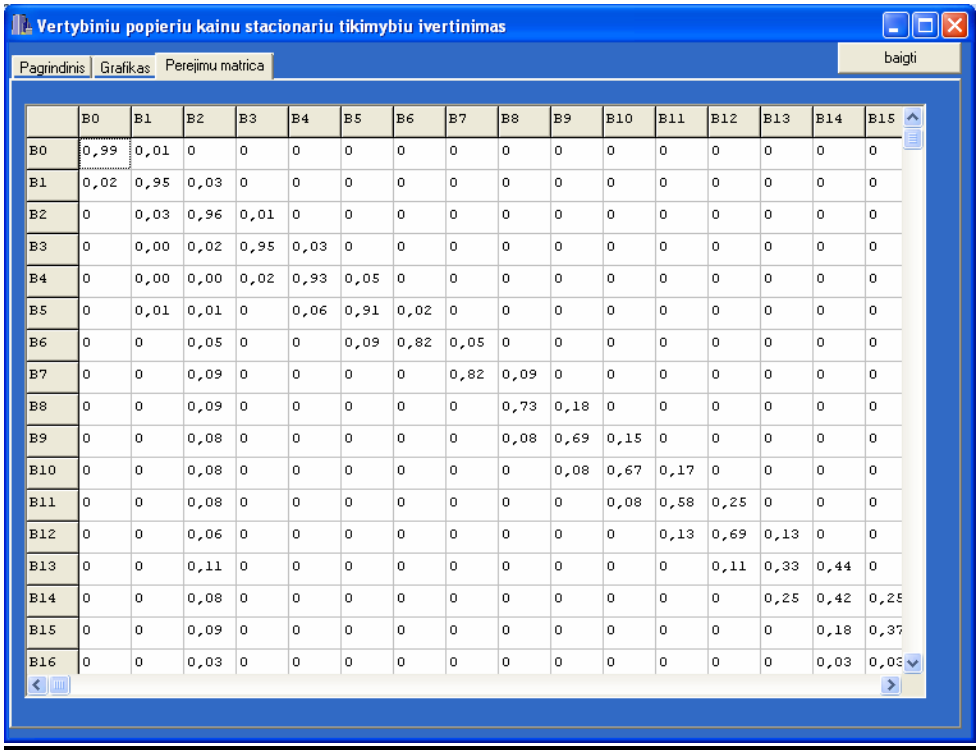
„*Rodyti perėjimų matricą*“ – pažymėjus šia varnelę, trečiajame programos lango puslapyje išvedama perėjimo tarp būsenų matrca.

e. Antrasis programos lango puslapis modeliuojant vertybinių popierių dinamiką



3.8 pav. Programinės įrangos antras langas modeliuojant vertybinių popierių dinamiką

f. Trečiasis programos lango puslapis modeliuojant vertybinių popierių rinkas



	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
B0	0,99	0,01	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B1	0,02	0,95	0,03	0	0	0	0	0	0	0	0	0	0	0	0	0
B2	0	0,03	0,96	0,01	0	0	0	0	0	0	0	0	0	0	0	0
B3	0	0,00	0,02	0,95	0,03	0	0	0	0	0	0	0	0	0	0	0
B4	0	0,00	0,00	0,02	0,93	0,05	0	0	0	0	0	0	0	0	0	0
B5	0	0,01	0,01	0	0,06	0,91	0,02	0	0	0	0	0	0	0	0	0
B6	0	0	0,05	0	0	0,09	0,82	0,05	0	0	0	0	0	0	0	0
B7	0	0	0,09	0	0	0	0	0,82	0,09	0	0	0	0	0	0	0
B8	0	0	0,09	0	0	0	0	0	0,73	0,18	0	0	0	0	0	0
B9	0	0	0,08	0	0	0	0	0	0,08	0,69	0,15	0	0	0	0	0
B10	0	0	0,08	0	0	0	0	0	0	0,08	0,67	0,17	0	0	0	0
B11	0	0	0,08	0	0	0	0	0	0	0	0,08	0,58	0,25	0	0	0
B12	0	0	0,06	0	0	0	0	0	0	0	0	0,13	0,69	0,13	0	0
B13	0	0	0,11	0	0	0	0	0	0	0	0	0	0,11	0,33	0,44	0
B14	0	0	0,08	0	0	0	0	0	0	0	0	0	0	0,25	0,42	0,25
B15	0	0	0,09	0	0	0	0	0	0	0	0	0	0	0	0,18	0,37
B16	0	0	0,03	0	0	0	0	0	0	0	0	0	0	0	0,03	0,03

3.9 pav. Programinės įrangos trečias langas modeliuojant vertybinių popierių dinamiką

3.3 Reikalavimai pradinių duomenų bylai

a) <u>Aptarnavimo sistemų modeliavimas:</u>

- ✓ Pirmoje duomenų failo eilutėje turi būti įvesta pradinė būsena. Ji turi būti rašoma po „=“ ir „ “. gale turi būti „;“ simbolis.

  pradine busena= 0 0 0;

- ✓ Antroje duomenų failo eilutėje turi būti aptarnaujančių įrenginių intensyvumai. Pradžioje eilitės padedami „=“ ir „ “, o gale – „;“.

  intensyvumai= 0.75 1 2;

  Jeigu modeliuojant, norima taikyti aproksimavimą, pirmi trys skaičiai po „=“, turi būti pradiniai aproksimuojamo atsitiktinio dydžio momentai.

- ✓ Trečioje eilutėje įvedamas srautų skaičius. Pradžioje eilutės padedami „=“ ir „ “, o gale – „;“.

  srautu skaicius= 1;

- ✓ Ketvirtoje eilutėje įvedamas aptarnaujančių aparatų skaičius. Pradžioje eilutės padedami „=“ ir „ “, o gale – „;“.

  aptarnaujanciu aparatu skaicius= 1;

b) <u>Vertybinių popierių rinkos modeliavimas:</u>

- ✓ Skirtingose eilutėse pateikiamos akcijos kainos skirtingu laiko momentu. Duomenų faile negali būti tarpų.
- ✓ Keičiant finansų rinkų aktyvų dinamikos modelį, t.y. keičiant įvykių aibę – *BusenosA()* metodą, reikia pakeisti kintamąjį *M* formos *TForm1* metode *Button1Click(TObject *Sender).* Jis nurodo, kiek įvykių sudaro įvykių aibę. Pirminėje programos versijoje *M*=10.

3.4 Reikalavimai sistemos aprašymo metodui

Sistemos aprašymą reikia pateikti formoje *TForm1* esančiame metode *Busenos(int *, int)*. Sistema modeliavimui, turi būti pateikiama *C++ Builder* kalboje naudojamu *switch-case* sakiniu 3.10 pav .

```
switch (poz)
 {
  case 0:           //tikrinam ivyki e{1}
  {
   if(xx[0]+xx[1]<L)
   {
    xx[1]++;
    radom=true;
   }
   else
   {
```

```
    xx[0]++;
    radom=true;
   }
   return Tintens[0];
  }    //ivykio e{1} aprasymo pabaiga
...
...
  case n:           //tikrinam ivyki e{n+1}
  {
   if(...)
   {
    ...;
    radom=true;
   }
   else
   {
    ...;
    radom=true;
   }
   return ...;
  }
 }   //ivykio e{n+1} aprasymo pabaiga
}   //metodo pabaiga
```

3.10 pav. Sistemos aprašymas *Switch-Case* sakiniu

Čia `xx[i]` yra tiriamos būsenos vaktorius (n-mačiam vektoriui $i \in \overline{0,n}$), kur `xx[i]` yra (i+1)-oji vektoriausi koordinatė. `Tintens[0]` yra perėjimo intensyvumų masyvo 1-as narys. Modeliuojant aptarnavimo sistemas, intensyvumai yra įvedami pradinių duomenų byloje (3.3 skyrelis). Modeliuojant finansinių aktyvų dinamiką - $\mu_1^u = \text{Tintens}[1]$; $\mu_2^u = \text{Tintens}[2]$; $p_1^u = \text{Tintens}[3]$; $\mu_1^d = \text{Tintens}[5]$; $\mu_1^d = \text{Tintens}[6]$; $p_1^d = \text{Tintens}[7]$;

Atlikus būsenos koordinatės keitimą, reikalinga pažymėti, kad nauja būsena rasta, sakiniu `radom=true;`. *Case* sakinio pabaigoje reikia gražinti perėjimo į naujai rastą būseną intensyvumą, sakiniu `return Tintens[0];`.

3.5 Reikalavimai sistemos charakteristikų skaičiavimo metodui

Sistemų būna labai įvarių, todėl universalaus charakteristikų skačiavimo metodo sukurti neįmanoma. Pateiksiu rekomendacijas, kuriomis vadovaujantis, galima sėkmingai suskaičiuoti ir išvesti modeliuojamos sistemos charakteristikas.

Sistemos charakteristikų skaičiavimo metodai pateikiami formoje *TForm1* esančiuose metoduose *Statistika_S1(), Statistika_S2(), Statistika_S3(), Statistika_S4(), Statistika_S5(), Statistika_S6()*. Norint sukurti savo metodą, galima redaguoti jau esamą *Statistika_S5()* arba sukurti naują. Tokiu atveju reikia nepamiršti pakoreguoti ir kreipimosi į sukurtą metodą, *Button3Click(TObject *Sender)* metode.

IŠVADOS

- Ištirta, kad atsitiktinį dydį, pasiskirsčiusį pagal log-normalųjį skirstinį, galima aproksimuoti dviejų eksponentinių fazių mišiniu, jeigu log-normaliojo skirstinio $\frac{m_2}{m_1^2} > 2$;
- Ištirta, kad atsitiktinį dydį, pasiskirsčiusį pagal χ^2 skirstinį, galima aproksimuoti dviejų eksponentinių fazių mišiniu, jeigu χ^2 skirstinio laisvės laipsnis $k \in [1;3]$;
- Sukurta universali programinė įranga *C++* kalboje skirta modeliuoti aptarnavimo sistemas, pagal pateiktą modeliuojamo objekto aprašymą, bei vertybinių popierių rinkas;
- Pasiūlyta sistemos funkcionavimo aprašymo metodika *C++* kalboje;
- Eilės ilgį, modeliuojant aptarnavimo sistemas su begalinio ilgio eilėmis, galima apriboti iki 20-30 paraiškų;
- Aproksimavimo eksponentinėm fazėm metodas, lyginant du pradinius momentus, netinkamai aproksimuoja atsitiktinius dydžius, jeigu aproksimuojamo atsitiktinio dydžio skirstinio $\frac{m_2}{m_1^2} \in \left(1, \frac{3}{2}\right)$;
- Nustatyta, kad finansinių aktyvų opcionų įvertinimo modelyje, įvykdymo kainai didėjant, opciono vertė artėja mažėja eksponentiškai;
- Modeliavimo trukmė priklauso nuo vidutinio finansinio aktyvo kainos dienos pokyčio bei skirtumo tarp minimalios ir maksimalios finansinio aktyvo kainos per nagrinėjamą laikotarpį.

LITERATŪROS SĄRAŠAS

1. Pranevičius, H., Valakevičius, E. Markovo procesų teorijos taikymas sistemoms modeliuoti. Kaunas, 1991. p. 5-29.
2. Valakevičius, E. Teigiamo atsitiktinio dydžio skirstinio aproksimavimas eksponentinių skirstinių mišiniu, *Lietuvos matematikų draugijos mokslo darbai*, II tomas, 1998. p. 475-477.
3. Valakevičius, E., Prioritetinės sistemos su eilėmis skaitmeninio modelio aproksimavimas, *Matematika ir matematinis modeliavimas*, Kaunas, Technologija, 32-36, 1999.
4. Whitt, W. On Approximations for Queues, Part III: Mixtures of Exponential Distributions, *AT&T ell Labs Tech Journal*, 63:1, 1984, p. 163-175.
5. Pranevičius, H., Valakevičius, E. Numerical models of systems specified by Markovian processes. Kaunas, 1996.
6. Ahn D., Boudoukh J., Richardson M., Whitelaw R. Implications from stock index and futures return autocorrelations, *Review of Financial Studies,* 16, 2002, p. 655-689.
7. Mayhew, S. Security price dynamics and simulation in financial engineering, *Proceedings of the 2002 Winter Simulation Conference*, p. 1568-1574.
8. Carr, P., Geman, H., Madan, D. H., Yor, M. The fine structure of asset returns: an empirical ivestigation. *Journal of business*, 75, 2002, p. 305- 332.
9. Pranevičius, H., Pranevičienė, I. Masinio aptarnavimo teorijos elementai. Vilnius, 1980.
10. Вентцель, Е. С., Овчаров, Л. А., Прикладные задачи теории вероятностей. Москва,1983, p. 353
11. Yao, D.D.W., Buzacott, J.A. Queueing models for a flexible machining station, Part II, *European journal of operational research,* vol 19, 1985, p.242-251.
12. Janssen, J., Manca, R., Di Biase, G. Markov and semi-Markov option pricing models with arbitrage possibility, *Applied stochastic models and data analysis,* vol 13, 1997, p. 103-108.
13. Mickevičius, G., Valakevičius, E., Modelling of non-Markovian queuing systems, *Technological and economic development of economy*, Vol XII, No 4, 2006, p. 295-300.
14. http://en.wikipedia.org/wiki/Log-normal_distribution .
15. http://en.wikipedia.org/wiki/Chi-square_distribution .

PRIEDAI

2. Priedas. Programos tekstas

Failas Unit1.cpp

```
//---------------------------------------------------------------------------
#include <dos.h>
#include <vcl.h>
#pragma hdrstop
#include <stdio.h>
#include <math.h>
#include <float.h>
#include <stdlib.h>
#include "Unit1.h"
#include "Unit2.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{

TPr=NULL;
TBPr=NULL;
PageControl1->Pages[0]->Show();
PageControl1->Pages[0]->Caption="Pagrindinis";
PageControl1->Pages[1]->Caption="Grafikas";
PageControl1->Pages[2]->Caption="Perejimu matrica";
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
struct  time t1;
 struct  time t2;

 Series1->Clear();
 Series2->Clear();
 Series3->Clear();
 Memo1->Lines->Clear();
 Memo2->Lines->Clear();
 TabSheet2->TabVisible=true;
 K=0;
 M=0;

 //IvestiPradSalygas();
 KK=0;
 NuskaitytiAkcKainas();
 //Apsukti();
 Grafika();

 gettime(&t1);
 Memo1->Lines->Add(FloatToStrF(t1.ti_hour,ffFixed,2,0) +"-"+FloatToStrF(t1.ti_min,ffFixed,2,0)+"-
"+FloatToStrF(t1.ti_sec,ffFixed,2,0)+"-"+FloatToStrF(t1.ti_hund,ffFixed,2,0));
 //"The current time is: %2d:%02d:%02d.%02d\n",
   //       t1.ti_hour, t1.ti_min, t1.ti_sec, t1.ti_hund);

 //AukstynZemynCount();
 AukstynZemynCountSizeBased();
 AukstynZemynStats();

 R=0;
 M=10;
 N=1;
 K=3;
 IK=8;

 if (M>0)
 {
  BusenuRadimasA();
  Apjungimas();
  Normavimas();         //!!!
   //if (ErgodiskumoPatikrinimas()) StacTikRadimas();
   //else ShowMessage("grandine nera ergodine!");
  StacTikRadimas();
  Normavimas();         //!!!
  IsvestiMemoAkc();
  Statistika_S6();
  OpcIkainojimas();
  if (CheckBox1->Checked==true) IsvestiStrGrid();
```

```
  else TabSheet3->TabVisible=false;

  //  Apjungimas();
  /*
  StacTikRadimas();
  IsvestiMemo();
  SurastiMax();

  Statistika_S5();
 // Statistika_S4();   //aproksimacija 1srautas n aparatu
 // Statistika_S3();                    //skaiciuoja tik sistemai S3 (2srautai, 1aparatas)
 // if ((K==2)&&(IK==4)) Statistika_S2();         //skaiciuoti tik sistemai S2
 // else Statistika_S1();
  Memo1->Lines->SaveToFile(BA);
  Memo2->Lines->SaveToFile(ST);
  */
 }
 int trukme;
 gettime(&t2);
 trukme=(t2.ti_hund + t2.ti_sec*100 + t2.ti_min*6000 + t2.ti_hour*360000)
        - (t1.ti_hund + t1.ti_sec*100 + t1.ti_min*6000 + t1.ti_hour*360000);
 Memo1->Lines->Add(FloatToStrF(t2.ti_hour,ffFixed,2,0) +"-"+FloatToStrF(t2.ti_min,ffFixed,2,0)+"-
"+FloatToStrF(t2.ti_sec,ffFixed,2,0)+"-"+FloatToStrF(t2.ti_hund,ffFixed,2,0));
 Memo1->Lines->Add(IntToStr(KK)+"- akciju: "+ FloatToStrF(trukme,ffFixed,10,0));
 TBusA *DD;
 while (TBAPr)
  {DD=TBAPr; TBAPr=TBAPr->kitas; delete DD;}
 TBusIntens *D;
 while (TPr)
  {D=TPr; TPr=TPr->kitas; delete D;}
 while (UPr)
  {DD=UPr; UPr=UPr->kitas; delete DD;}
 while (DPr)
  {DD=DPr; DPr=DPr->kitas; delete DD;}
 //delete r_apj;
 delete r;
 delete Tintens;
 delete[] Akc;

}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button2Click(TObject *Sender)
{
Form2->Close();
}
//---------------------------------------------------------------------------
void TForm1::NuskaitytiAkcKainas()
{
 int i;
 AnsiString DF;
 OpenDialog1->Filter="Text Failai(*.txt)|*.txt";
 if (OpenDialog1->Execute() && FileExists(OpenDialog1->FileName))
  {
   DF=OpenDialog1->FileName;
   KiekKainu(DF);
  }
 else DF="";

 FILE *F;
 if ((F = fopen(DF.c_str(), "r"))==NULL);
 else
 {
  float *AkcX=new float[KK];
  for(i=0; i<KK; i++)
   fscanf(F, "%f", &AkcX[i]);
  //for(i=0; i<KK; i++) Memo1->Lines->Add(AkcX[i]);
  Akc=new double[KK];
  for(i=0; i<KK; i++)
   Akc[i]=AkcX[i];

  //KK--; //nezinau kodel bet paskutine kaina 0

  //for(i=0; i<KK; i++) Memo1->Lines->Add(Akc[i]);
 }
}
//---------------------------------------------------------------------------
void TForm1::Apsukti()
{
 int i;
 float x;
 for (i=0;i<int(KK/2);i++)
 {
  x=Akc[i];
```

```
  Akc[i]=Akc[KK-1-i];
  Akc[KK-1-i]=x;
 }
}
//---------------------------------------------------------------------------

void TForm1::SurastiEkstremumus()
{
 int i;
 AkcMax=0;
 AkcMin=100000;
  for(i=0; i<KK; i++)
   {
    if (Akc[i]>AkcMax)
     AkcMax=Akc[i];
    if (Akc[i]<AkcMin)
     AkcMin=Akc[i];
   }
  //SuskaidytiIntervalais(max);
}
//---------------------------------------------------------------------------
void TForm1::KiekKainu(AnsiString DF)
{
 float elem;
 FILE *F;
 F = fopen(DF.c_str(),"r");
 KK=0;
 while (!feof(F))
 {
  fscanf(F, "%f", &elem);
  KK++;
 }
 fclose(F);
}
//---------------------------------------------------------------------------
void TForm1::SurastiIntensyvumus()
{
//int i=0;
//float PradKaina=0;
//kolkas nezinauka cia rasyt
}
//---------------------------------------------------------------------------

/*
void TForm1::SuskaidytiIntervalais(float max)
{
 N=-1;
 N=StrToInt(Edit1->Text); //nuskaitom intervalu skaiciu
 if (N<=0) ShowMessage("iveskite teigiama intervalu skaiciu");
 else
 {
  IntIlgis=0;
  IntIlgis=(max-AkcMin)/N;
  max=max+IntIlgis/10.;        //pakeiciam
  AkcMin=AkcMin-IntIlgis/10.;  //intervalo
  IntIlgis=(max-AkcMin)/N;    //ilgi
 }
} */
//---------------------------------------------------------------------------
void TForm1::SurastiTiketinaKaina()
{
int i=0;
float Utik=0;     //akc kaina kils (up)
float Dtik=0;     //akc kaina kris (down)
float Stik=0;     //akc kaina nekis (stay)
TBusIntens *D=TPr;
bool yra=false;
while (!yra)    //(i-1) bus tas intervalas kuriame kaina dabr yra
{
 i++;
 if (Akc[KK-1]<AkcMin+i*IntIlgis) yra=true;
}
i=i-1;
//paskaiciuosiu tikimybe kad akc kaina kris/kils
Memo2->Lines->Add("Paskutine akcijos kaina buvo: "+FloatToStrF(Akc[KK-1],ffFixed,5,2));
 while (D)
 {
  if (D->BI.eil==i)
  {
   Memo2->Lines->Add("Akcijos kaina bus "+IntToStr(D->BI.st)+" intervale su tikimybe "+FloatToStrF(D-
>BI.BInt,ffFixed,5,4));
   if (D->BI.st>i)
    Utik=Utik+D->BI.BInt;
   else if (D->BI.st<i)
```

```
         Dtik=Dtik+D->BI.BInt;
        else Stik=Stik+D->BI.BInt;
  }
  D=D->kitas;
 }
Memo2->Lines->Add("");
Memo2->Lines->Add("Tikimybe,kad akcijos kaina kils: " +FloatToStrF(Utik,ffFixed,5,4));
Memo2->Lines->Add("Tikimybe,kad akcijos kaina kris: " +FloatToStrF(Dtik,ffFixed,5,4));
Memo2->Lines->Add("Tikimybe,kad akcijos kaina nekis: " +FloatToStrF(Stik,ffFixed,5,4));

}

//---------------------------------------------------------------------------
void TForm1::PerejimuTikimybes()
{
 int i;
 int poz1,poz2;
 TBusIntens *D;
 for (i=0;i<KK-1;i++)
  {
   poz1=int((Akc[i]-AkcMin)/IntIlgis);
   poz2=int((Akc[i+1]-AkcMin)/IntIlgis);

   D=new TBusIntens;
   D->BI.BInt=1;
   D->BI.eil=poz1;
   D->BI.st=poz2;
   D->kitas=TPr;
   TPr=D;
  }

Apjungimas();
Normavimas();
}
//---------------------------------------------------------------------------
void TForm1::Normavimas()    //sunormuoja TPr sarasa
{
 TBusIntens *D;
 float suma;    //suma visu iseinanciu is tiriamosios busenos
 int i;
 for (i=0;i<N;i++)
 {
  suma=0;
  D=TPr;
  while (D)
  {
   if (D->BI.eil==i)
    suma=suma+D->BI.BInt;
   D=D->kitas;
  }

  D=TPr;
  while (D)
  {
   if (D->BI.eil==i)
    D->BI.BInt=D->BI.BInt/suma;
   D=D->kitas;
  }
 }
}
//---------------------------------------------------------------------------
void TForm1::Apjungimas()    //sudeda pasikartojancius intensyvumus pereinant is vienos busenos i kita
{
 TBusIntens *D0;
 TBusIntens *D1;
 TBusIntens *D;
 TBusIntens *istr;
 D1=TPr;
 D=TPr;
 int i;
 while (D)
 {
  D0=D;
  D1=D->kitas;
  while (D1)
  {
   if(D->BI.eil==D1->BI.eil)
    if(D->BI.st==D1->BI.st)
     {
      D->BI.BInt=D->BI.BInt+D1->BI.BInt;
      istr=D1;
      D0->kitas=D1->kitas;
      D1=D1->kitas;
      delete istr;
```

```
     }
    else {D1=D1->kitas;D0=D0->kitas;}
  else {D1=D1->kitas;D0=D0->kitas;}
  }
 D=D->kitas;
 }
}
//---------------------------------------------------------------------------
bool TForm1::ErgodiskumoPatikrinimas()    //patikrina ar is bet kurios busenos galima patekti i bet kuria kita
{
 int i=0;
 bool yra=false;
 TBusIntens *X,*Y;
 int *A=new int[N];
 for (i=0;i<N;i++) A[i]=i;      //susirasom visus intervalus
 for (i=0;i<N;i++)
 {
  yra=false;
  X=TPr;
  while ((X)&&(!yra))
  {
   if ((X->BI.eil==A[i])||(X->BI.st==A[i])) yra=true;
   X=X->kitas;
  }
  if (!yra) A[i]=-2;  //tokios busenos/intervalo nera
 }

 yra=false;
 int k=-1;
 while (k!=0)
 {
  if (k==-1) A[N-1]=-1; //pradedam eiti nuo galo
  else A[0]=-1;         //nuo pradziu
  while (!yra)
  {
   yra=true;
   for (i=N-1;i>=0;i--)
   {
    if (A[i]==-1)
    {
     X=TPr;
     while (X)
     {
      if (X->BI.eil==i)
       if (A[X->BI.st]!=-1) {A[X->BI.st]=-1; yra=yra*false;}
      X=X->kitas;
     }
    } //if (A[i]==-1) pab
   } //for (i=N-1;i>=0;i--) pab
  } //while (!yra) pab
  k=0;
 }
 yra=true;     //patikrinam ar i visas busenas sugebejom nueiti
 i=0;
 while ((i<N)&&(yra))
 {
  if ((A[i]!=-1)&&(A[i]!=-2)) yra=false;
  i++;
 }
if(yra) return true;  //visos busenos yra griztamos
else return false;
}
//---------------------------------------------------------------------------
void TForm1::StacTikRadimas()    //kilti lygiu galima tik su liambda wadinas
{                                //vadinasi visada bus tik vienas atejimas ir keli isejimai.gal ce ir blogai..
 r=new float[N];
 float *S;         //S yra d elementu sarasas
 S=new float[N];
 int i=N-1;
 float a,b,c,d;              //formules nariai
 TBusIntens *P,*X,*ir;
 int b1,b2;               //busena su kuria trinama busena turi santykiu
 float per;                //pasikeites perejimu intensyvumas suskaiciuotas su a,b,c,d
 bool yra;
 int kzk;               //parodo kelinta "iseinama" intensyvuma pasirinkti
 int kzzk;              //neimam reikiamo intensyvumo kol kzzk!=kzk;

//susikuriu nauja sarasa su tais paciais elementais is TPr
 TBusIntens *xx=TPr,*Pp,*Pg;
 Pg=new TBusIntens;        //pirmas elem
 Pg->BI.BInt=xx->BI.BInt;
 Pg->BI.eil=xx->BI.eil;
 Pg->BI.st=xx->BI.st;
 Pg->kitas=NULL;
```

```
P=Pg;
xx=xx->kitas;

while (xx)
{
 Pp=new TBusIntens;
 Pp->BI.BInt=xx->BI.BInt;
 Pp->BI.eil=xx->BI.eil;
 Pp->BI.st=xx->BI.st;
 Pp->kitas=NULL;
 Pg->kitas=Pp;
 Pg=Pg->kitas;
 xx=xx->kitas;
}
while (i>0)
{
 b1=0;
 b2=0;

 d=0;
 X=P;
 kzk=0;
 while (X)    //surandam paskutini formules elementa
 {
   if (X->BI.eil==i)
    if (X->BI.st!=i)
     d=d+X->BI.BInt;
   X=X->kitas;
 }

 while ((b1==0)&&(b2==0))
 {
   b1=-1;
   b2=-1;
   a=0;
   b=0;
   c=0;

 X=P;
 while ((X)&&(b1==-1))    //busena is kurios galim patekti i trinama busena
 {
   if (X->BI.st==i)
    if (X->BI.eil!=i)   //apsauga nuo rinkiu
     {
      b1=X->BI.eil;
      b=X->BI.BInt;
     }
   X=X->kitas;
 }


 X=P;
 kzzk=0;
 while ((X)&&(b2==-1))   //busena i kuria galim patekti is trinama busena
 {
   if (X->BI.eil==i)
    if (X->BI.st!=i)
    {
     if (kzzk==kzk)
     {
      b2=X->BI.st;
      c=X->BI.BInt;
     }
     kzzk++;
    }
   X=X->kitas;
 }


 X=P;
 while ((X)&&(a==0))    //surandam pirmaji formules elementa
 {
   if ((X->BI.eil==b1)&&(X->BI.st==b2))
   {
    a=X->BI.BInt;
   }
   X=X->kitas;
 }

 if (d!=0) per=a+b*c/d;
 else per=0;

 if ((per>0)&&(b2!=-1))
  {
```

```
  //patikrinam ar tokio dar ner
   X=P;
   yra=false;
   while ((X)&&(!yra))
   {
     if (X->BI.eil==b1)
      if (X->BI.st==b2)
      {
       X->BI.BInt=per;
       yra=true;
      }
     X=X->kitas;
   }
 //irasom nauja gauta perejimu intensyvuma
  if (!yra)
  {
   ir=new TBusIntens;
   ir->BI.BInt=per;
   ir->BI.eil=b1;
   ir->BI.st=b2;
   ir->kitas=P;
   P=ir;
  }

//reik paziuret ar tokio dar intensyvumo dar nera pagrindiniam-pradiniam sarase
//ir irasyti jei nera
   X=TPr;
   yra=false;
   while ((X)&&(!yra))
   {
     if (X->BI.eil==b1)
      if (X->BI.st==b2)
      {
       X->BI.BInt=per;
       yra=true;
      }
     X=X->kitas;
   }
 //irasom nauja gauta perejimu intensyvuma
  if (!yra)
  {
   ir=new TBusIntens;
   ir->BI.BInt=per;
   ir->BI.eil=b1;
   ir->BI.st=b2;
   ir->kitas=TPr;
   TPr=ir;
  }

  b1=0;    //kad dar karta wykdytu cikla
  b2=0;
  kzk++;
 }// if per>0 pabaiga

 //istrinam ta intensyvuma kuriuo atejom i trinama busena
 if ((b2==-1)&&(b1!=-1))
 {
  X=P;
  yra=false;
  TBusIntens *Y=NULL,*Z;
  while ((X)&&(!yra))
  {
    if (X->BI.st==i)
     if (X->BI.eil!=i)
     {
      if (Y)
      {
       Y->kitas=X->kitas;
       Z=X;
       delete Z;
       yra=true;
      }
     else   //jeigu reik trinti pask elementa
     {
      Z=X;
      X=X->kitas;
      P=P->kitas;
      delete Z;
      yra=true;
     }
    }
    Y=X;
    if (X->kitas) X=X->kitas;
  }
```

```
 b1=0;
 b2=0;
 kzk=0;
 } //if b2==-1 pab

 //jeigu neradom jokio "ieinancio" inetnsyvumo istrinam visus "iseinancius" inetnsyvumus
 if(b1==-1)
 {
  X=P;
  TBusIntens *Y=NULL,*Z;
  while (X)
  {
   yra=false;
    if (X->BI.eil==i)
    {
     if (Y)
     {
      Y->kitas=X->kitas;
      Z=X;
      delete Z;
      yra=true;
     }
     else   //jeigu reik trinti pask elementa
     {
      Z=X;
      X=X->kitas;
      P=P->kitas;
      delete Z;
      yra=true;
     }
    }
    if (!yra)
    {
     Y=X;
     X=X->kitas;
    }
    else if (Y) X=Y->kitas;
  }
 }     //if b1==-1 pab

}      //while (b1==0)&&(b2==0) pabaiga

S[i]=d;

i--;
}

//r saraso formavimas
i=1;
int j;
float rr;      //busimas r
r[0]=P->BI.BInt;
while (i<N)
{
 rr=0;
 X=TPr;
  while (X)
  {
    if (X->BI.st==i)
     if (X->BI.eil<i)    //nepriklauso rinkes
     {
      rr=rr+r[X->BI.eil]*X->BI.BInt/S[i];
     }
    X=X->kitas;
  }
 r[i]=rr;
 i++;
}

float suma=0;
for(j=0;j<N;j++)
 suma=suma+r[j];
for(j=0;j<N;j++)
 r[j]=r[j]/suma;
Memo2->Lines->Add("");
Memo2->Lines->Add("Stacionarios tikimybes (>0.005):");
int k=0;
while (k<N)
 {
  if (r[k]>=0.005)
  Memo2->Lines->Add(IntToStr(k)+") "+FloatToStrF(r[k],ffFixed,15,7));
  k++;
 }
Memo2->Lines->Add("");
```

```
}
//---------------------------------------------------------------------

void TForm1::SurastiMax()       //busena kurios stac tik didziausia
{
 int i;
 float max=0;
 int maxI=0;
 for (i=0;i<N;i++)
 {
  if (r[i]>max) {max=r[i];maxI=i;}
 }
 Memo2->Lines->Add("");
 Memo2->Lines->Add("Busena "+IntToStr(maxI)+"-a turi didziausia stac. tikimybe:");
 Memo2->Lines->Add(FloatToStr(max));
 Memo2->Lines->Add("");
 for (i=0;i<KK;i++)
 {
  Series2->AddXY(i,AkcMin+maxI*IntIlgis-IntIlgis/2,"",clGreen);
  Series3->AddXY(i,AkcMin+maxI*IntIlgis+IntIlgis/2,"",clGreen);
 }

 //kuriuose intervaluose kaina gali buti su x% tikimybe
 float x;
 if (Akc==NULL) x=StrToInt(Edit2->Text);  //%
 else x=50;                               //%
if (x<=0) ShowMessage("Iveskite teigiama tiketinumo lygmeni");
else
 {
 x=x/100;
 AnsiString eil1,eil2;
 float *rr=new float[N];

 for(i=0;i<N;i++) rr[i]=r[i];

 float suma=0,maxx;//,xmaxx=2;     //xmax-buves maximumas
 int nr=-1;//,xnr=-1;        //maximumo numeris,xnr-buves maximumo nr
 while (suma<x)
  {
   i=0;
   maxx=0;
   while (i<N)
    {
     if (rr[i]>maxx)//&&(rr[i]<=xmaxx)&&(i!=xnr))
      {
       maxx=rr[i];
       nr=i;
      } //istrinam rasta max
     i++;
    }
   if (suma==0) eil2=IntToStr(nr);
   else eil2=eil2+", "+IntToStr(nr);
   rr[nr]=-1;
   suma=suma+maxx;
  }
 eil1="su "+FloatToStrF(suma,ffFixed,3,2)+" tikimybe galim speti, kad sistema bus: ";
 Memo2->Lines->Add("");
 Memo2->Lines->Add(eil1);
 if (suma==maxx) Memo2->Lines->Add(eil2+" busenoje");
 else Memo2->Lines->Add(eil2+" busenose");
 Memo2->Lines->Add("");
}
}

//---------------------------------------------------------------------------
void TForm1::Grafika()
{
 int i;
 for(i=0;i<KK;i++)
 {
  Series1->AddXY(i,Akc[i],"",clRed);
 }
}
//---------------------------------------------------------------------------
void TForm1::OpcIkainojimas()
{
 int i,k,j;
 float PalNorma=StrToInt(Edit2->Text);
if (PalNorma<=0) ShowMessage("Iveskite teigiama palukanu norma");
else
{
 PalNorma=1/(1+PalNorma/100);       //diskonto daugiklis
 float *Opc=new float[IntSk];
 float *OpcDisk=new float[IntSk];
```

```
 int LaikotarpiuSk=10;//!!!!StrToInt(Edit3->Text);

if (LaikotarpiuSk<=0) ShowMessage("Iveskite teigiama laikotarpiu skaiciu");
else
{
 AnsiString eil;
 for(i=0;i<IntSk;i++)
 {
  k=i;
  j=IntSk-1;
  Opc[i]=0;
  while (j>k)
  {
   Opc[i]=Opc[i]+r_apj[j]*((j-k+0.5)*IntIlgis+AkcMin);
   j--;
  }
 }
 for (i=0;i<IntSk;i++)
  OpcDisk[i]=pow(PalNorma,LaikotarpiuSk)*Opc[i];
 Memo2->Lines->Add("");
 Memo2->Lines->Add("              Opciono ikainojimas:");
 Memo2->Lines->Add("-------------------------------------------------");
 Memo2->Lines->Add("%Ivykdymo kaina   Tiketina verte   Diskontuota verte");
 Memo2->Lines->Add("-------------------------------------------------");
 for (i=0;i<IntSk;i++)
  {
   if (AkcMin+(i+0.5)*IntIlgis>1000)
     eil=FloatToStrF(AkcMin+(i+0.5)*IntIlgis,ffFixed,6,2)+"               ";
   else if (AkcMin+(i+0.5)*IntIlgis>100)
          eil=FloatToStrF(AkcMin+(i+0.5)*IntIlgis,ffFixed,6,2)+"              ";
        else if (AkcMin+(i+0.5)*IntIlgis>10)
               eil=FloatToStrF(AkcMin+(i+0.5)*IntIlgis,ffFixed,6,2)+"               ";
             else eil=FloatToStrF(AkcMin+(i+0.5)*IntIlgis,ffFixed,6,2)+"                ";

   if (Opc[i]>1000)
     eil=eil+FloatToStrF(Opc[i],ffFixed,6,2)+"            ";
   else if (Opc[i]>100)
          eil=eil+FloatToStrF(Opc[i],ffFixed,6,2)+"             ";
        else if (Opc[i]>10)
               eil=eil+FloatToStrF(Opc[i],ffFixed,6,2)+"              ";
             else eil=eil+FloatToStrF(Opc[i],ffFixed,6,2)+"               ";
   eil=eil+FloatToStrF(OpcDisk[i],ffFixed,6,2);
   Memo2->Lines->Add(eil);
          //   Memo2->Lines->Add(FloatToStrF(AkcMin+(i+0.5)*IntIlgis,ffFixed,6,2)+"
"+FloatToStrF(Opc[i],ffFixed,6,2)+"            "+FloatToStrF(OpcDisk[i],ffFixed,6,2));
  }
 Memo2->Lines->Add("-------------------------------------------------");

}
}
}
//---------------------------------------------------------------------------
void TForm1::IvestiPradSalygas()
{
 int i=0;
 AnsiString DF;
 OpenDialog1->Filter="Tekstines bylos (*.txt)|*.txt";
 if (OpenDialog1->Execute() && FileExists(OpenDialog1->FileName))
  {
   DF=OpenDialog1->FileName;
   KasuSk(DF);
  }
 else DF="";

 FILE *F;
 char elem;

 TBus *D;
 D=new TBus;
 D->sk=new int[K];
 Tintens=new float[IK];
  if ((F = fopen(DF.c_str(), "r"))==NULL);
  else
   {
    i=0;
     while (elem!='=') fscanf(F, "%c", &elem);
     fscanf(F, "%c", &elem);
    while (elem!=';')
    {
     fscanf(F, "%d%c", &D->sk[i], &elem);
     i++;
    }
     elem='a';
```

```
    i=0;
     while (elem!='=') fscanf(F, "%c", &elem);
     fscanf(F, "%c", &elem);
    while (elem!=';')
    {
     fscanf(F, "%f%c", &Tintens[i], &elem);
     i++;
    }
     while (elem!='=') fscanf(F, "%c", &elem);
     fscanf(F, "%c", &elem);
    fscanf(F, "%d %c", &R,&elem);      //srautu skaicius

     while (elem!='=') fscanf(F, "%c", &elem);
     fscanf(F, "%c", &elem);
    fscanf(F, "%d %c", &M,&elem);      //ivykiu skaicius

  fclose(F);
  D->kitas=TBPr;
  TBPr=D;
  N=1;                  //busenu sk

  if (CheckBox3->Checked==true) SurastiFaziuIntens();
  }

 L=StrToInt(Edit1->Text); //nuskaitom max parasku skaiciu sistemoje
 if (L<0)
 {
  L=0;
  ShowMessage("Ivedet neigiama maksimalu paraisku kieki sistemoje");
 }

}
//---------------------------------------------------------------------------

void TForm1::IsvestiMemoAkc()
{
 Memo1->Lines->Add("Busenu aibe:");
 AnsiString x;
 TBusA *D;
 D=TBAPr;
 for(int k=0; k<N; k++)
 {
  x="[";
  x=x+FloatToStrF(D->sk[0],ffFixed,7,2);
  for(int j=1; j<K; j++)
  {
   x=x+";"+FloatToStrF(D->sk[j],ffFixed,3,0);
  }
  x=x+"]";
  Memo1->Lines->Add(IntToStr(k)+") "+x);
  D=D->kitas;
 }

 Memo2->Lines->Add("INTENSYVUMAI:");
 for(int j=0; j<IK; j++)
 {
  Memo2->Lines->Add(IntToStr(j)+"-as intensyvumas: "+FloatToStrF(Tintens[j],ffFixed,15,13));
 }
}
//---------------------------------------------------------------------
void TForm1::IsvestiMemo()
{
 Memo1->Lines->Add("Busenu aibe:");
 AnsiString x;
 TBus *D;
 D=TBPr;
 for(int k=0; k<N; k++)
 {
  x="[";
  x=x+IntToStr(D->sk[0]);
  for(int j=1; j<K; j++)
  {
   x=x+";"+IntToStr(D->sk[j]);
  }
  x=x+"]";
  Memo1->Lines->Add(IntToStr(k)+") "+x);
  D=D->kitas;
 }

 Memo2->Lines->Add("INTENSYVUMAI:");
 for(int j=0; j<IK; j++)
 {
  Memo2->Lines->Add(IntToStr(j)+"-as intensyvumas: "+FloatToStrF(Tintens[j],ffFixed,15,13));
 }
```

```
}
//---------------------------------------------------------------------
void TForm1::IsvestiStrGrid()
{
 int i,j;
 TBusIntens *D=TPr;
 StringGrid1->ColCount=N+1;
 StringGrid1->RowCount=N+1;
 for(i=0;i<N;i++)
  StringGrid1->Cells[i+1][0]="B"+IntToStr(i);
 for(i=0;i<N;i++)
  StringGrid1->Cells[0][i+1]="B"+IntToStr(i);
 for(i=1;i<=N;i++)
  for(j=1;j<=N;j++)
   StringGrid1->Cells[i][j]=0;
 float x;
 while (D)
  {
   x=StrToFloat(StringGrid1->Cells[D->BI.st+1][D->BI.eil+1]);
   x=x+D->BI.BInt;
   StringGrid1->Cells[D->BI.st+1][D->BI.eil+1]=FloatToStrF(x,ffFixed,3,2);
   D=D->kitas;
  }
}
//---------------------------------------------------------------------
void TForm1::AukstynZemynCountSizeBased()
{
 int x;
 int y;
 int i=0,j=0;
 double SizeBase=StrToFloat(Edit2->Text);       //kokio dydzio pokytis jau fiksuojamas
 double Base;                                   //kaina nuo kurios prasideda kilimas/kritimas
 KKU=0;
 KKD=0;

 UPr=new TBusA;
 UPr->sk=new double[1];
 UPr->sk[0]=0;             //ar nereik padidinti KKU ir KKD del situ fake nariu??
 UPr->kitas=NULL;
 DPr=new TBusA;
 DPr->sk=new double[1];
 DPr->sk[0]=0;
 DPr->kitas=NULL;

 TBusA *DU1;                //pagalbine rodykle
 TBusA *DU2;                //pagalbine rodykle
 DU2=UPr;
 TBusA *DD1;                //pagalbine rodykle
 TBusA *DD2;                //pagalbine rodykle
 DD2=DPr;

 AkcMax=Akc[0];
 AkcMin=Akc[0];
 for(int ii=1;ii<KK;ii++)
 {
  if (Akc[ii]>AkcMax) AkcMax=Akc[ii];
  else if (Akc[ii]<AkcMin) AkcMin=Akc[ii];
 }

 int yy=0;
 double SumPok=0;
 while (yy+1<KK)
 {
  SumPok=SumPok + fabs(Akc[yy+1]-Akc[yy]);
  yy++;
 }
 IntIlgis=SumPok/(KK-1);
 IntSk=int((AkcMax-AkcMin)/IntIlgis)+1;

 SizeBase= IntIlgis;

 x=0;
 y=1;

 while (y+1<=KK)
 {
  i=0; j=0; Base=Akc[x];
  while ((i==0 || j==0) && (y+1<=KK))
  {
   if (Akc[y]>Akc[x]) i++;
   else if (Akc[y]<Akc[x]) j++;
        //else y++;        //jei kaina nesikeicia; uzkomentuota nes sekantis zingsnis pakeicia y
   if ((i==0) || (j==0)) {x++; y++;}
  } //while (i==0 or j==0) pab
```

```
  if ((i>0) && ((Akc[y]<Akc[x]) || (y==KK)))
  {
      DU1=new TBusA;
      DU1->sk=new double[1];
      DU1->sk[0]=((Akc[x]-Base)/SizeBase);         //kol kas nedalinu is dienu skaiciaus
      DU1->kitas=NULL;
      DU2->kitas=DU1;
      DU2=DU2->kitas;
      KKU++;
  }
  if ((j>0) && ((Akc[y]>Akc[x]) || (y==KK)))
  {
      DD1=new TBusA;
      DD1->sk=new double[1];
      DD1->sk[0]=((Base-Akc[x])/SizeBase);
      DD1->kitas=NULL;
      DD2->kitas=DD1;
      DD2=DD2->kitas;
      KKD++;
  }
 }  //while (y->kitas) pab
}
//---------------------------------------------------------------------
void TForm1::AukstynZemynCount()
{
 int x;
 int y;
 int i=0,j=0;
 KKU=0;
 KKD=0;

 UPr=new TBusA;
 UPr->sk=new double[1];
 UPr->sk[0]=0;              //ar nereik padidinti KKU ir KKD del situ fake nariu??
 UPr->kitas=NULL;
 DPr=new TBusA;
 DPr->sk=new double[1];
 DPr->sk[0]=0;
 DPr->kitas=NULL;

 TBusA *DU1;                //pagalbine rodykle
 TBusA *DU2;                //pagalbine rodykle
 DU2=UPr;
 TBusA *DD1;                //pagalbine rodykle
 TBusA *DD2;                //pagalbine rodykle
 DD2=DPr;

 x=0;
 y=1;

 while (y+1<=KK)
 {
  i=0; j=0;
  while ((i==0 || j==0) && (y+1<=KK))
  {
   if (Akc[y]>Akc[x]) i++;
   else if (Akc[y]<Akc[x]) j++;
        //else y++;        //jei kaina nesikeicia; uzkomentuota nes sekantis zingsnis pakeicia y
   if ((i==0) || (j==0)) {x++; y++;}
  } //while (i==0 or j==0) pab

  if ((i>0) && ((Akc[y]<Akc[x]) || (y==KK)))
  {
      DU1=new TBusA;
      DU1->sk=new double[1];
      DU1->sk[0]=i;
      DU1->kitas=NULL;
      DU2->kitas=DU1;
      DU2=DU2->kitas;
      KKU++;
  }
  if ((j>0) && ((Akc[y]>Akc[x]) || (y==KK)))
  {
      DD1=new TBusA;
      DD1->sk=new double[1];
      DD1->sk[0]=j;
      DD1->kitas=NULL;
      DD2->kitas=DD1;
      DD2=DD2->kitas;
      KKD++;
  }
 }  //while (y->kitas) pab
}
```

```
//---------------------------------------------------------------------
void TForm1::AukstynZemynStats()
{

 //nenaudosiu nes isskaiciuosiu vid kainos pokyti
   //IntSk=StrToFloat(Edit1->Text); //nuskaitom i kiek intervalu skaidysim
 //float max=Akc[0], min=Akc[0];

 TBusA *DU, *DD;
 DU=UPr;
 DD=DPr;
 double SumU=0;      //suma kilimo dienu
 double SumD=0;      //suma kritimo dienu
 double Sum2U=0;      //kvadratu suma kilimo dienu
 double Sum2D=0;      //kvadratu suma kritimo dienu
 double Sum3U=0;      //kubu suma kilimo dienu
 double Sum3D=0;      //kubu suma kritimo dienu

 while (DU)
 {
  SumU=SumU + DU->sk[0];
  Sum2U=Sum2U + powl((DU->sk[0]),2);
  Sum3U=Sum3U + powl((DU->sk[0]),3);
  DU=DU->kitas;
 }
 while (DD)
 {
  SumD=SumD + DD->sk[0];
  Sum2D=Sum2D + powl((DD->sk[0]),2);
  Sum3D=Sum3D + powl((DD->sk[0]),3);
  DD=DD->kitas;
 }
 //if ((DU) || (DD)) ShowMessage("kazkas blogai su U ir D sarasais");

 m1U=(SumU)/KKU;
 m1D=(SumD)/KKD;
 m2U=(Sum2U)/(KKU-1);         //ar tikrai KKU-1 ??
 m2D=(Sum2D)/(KKD-1);
 m3U=(Sum3U)/(KKU-1);
 m3D=(Sum3D)/(KKD-1);
   /*
 m1U=0.994920435;
 m1D=0.995784366;
 m2U=3.156138284;
 m2D=2.607314327;
 m3U=16.91577924;
 m3D=11.15779691;
    */

 Memo1->Lines->Add("m1 U: "+FloatToStrF(m1U,ffFixed,5,4));
 Memo1->Lines->Add("m2 U: "+FloatToStrF(m2U,ffFixed,5,4));
 Memo1->Lines->Add("m3 U: "+FloatToStrF(m3U,ffFixed,5,4));
 Memo1->Lines->Add("m1 D: "+FloatToStrF(m1D,ffFixed,5,4));
 Memo1->Lines->Add("m2 D: "+FloatToStrF(m2D,ffFixed,5,4));
 Memo1->Lines->Add("m3 D: "+FloatToStrF(m3D,ffFixed,5,4));
/* iskeliau auksciau
 AkcMax=Akc[0];
 AkcMin=Akc[0];
 for(int i=1;i<KK;i++)
 {
  if (Akc[i]>AkcMax) AkcMax=Akc[i];
  else if (Akc[i]<AkcMin) AkcMin=Akc[i];
 }

 int yy=0;
 double SumPok=0;
 while (yy+1<KK)
 {
  SumPok=SumPok + fabs(Akc[yy+1]-Akc[yy]);
  yy++;
 }
 IntIlgis=SumPok/(KK-1);
 IntSk=int((AkcMax-AkcMin)/IntIlgis)+1;
*/
 //Pakoreguojam AkcMin ir AkcMax, nes realiai kitimas galimas ir didesnis negu iki siol svyravo
 AkcMax=AkcMin + IntIlgis*float(IntSk);  //!!!
 IntSk=(AkcMax-AkcMin)/IntIlgis+0.1;
 //if (AkcMin-IntIlgis>=0) {AkcMin=AkcMin-IntIlgis; IntSk++;}
 //AkcMax=AkcMax+IntIlgis; IntSk++;

 //IntIlgis=(AkcMax-AkcMin)/IntSk;
 //int xxx=int(IntIlgis*1000)/1000; IntIlgis=xxx;
 Memo1->Lines->Add("Akc max: "+FloatToStrF(AkcMax,ffFixed,5,4));
 Memo1->Lines->Add("Akc min: "+FloatToStrF(AkcMin,ffFixed,5,4));
```

```
 Memo1->Lines->Add("Akc pokytis: "+FloatToStrF(IntIlgis,ffFixed,5,4));

 //m1U=2.0;
 //m1D=3.0;
 //m2U=m2U+2;
 //m3U=30.0;
/*
 m2U=m2U*2;
 m2D=m2D*2;
 m3U=m3U*6;
 m3D=m3D*6;
*/
 //intensyvumai
 Tintens=new float[8];
 //tikrinsim ar parametrai patenka metodo A.S.
 if (((m2U/pow(m1U,2)<2) && (m2U/pow(m1U,2)-m3U/(6*pow(m1U,3))<1)) || ((m2U/pow(m1U,2)>2) && (m2U/pow(m1U,2)-
m3U/(6*pow(m1U,3))>1)))
  {
   Memo1->Lines->Add("Bus taikoma dvieju momentu lyginimo aproksimacija didejimo intensyvumams, nes apskaiciuoti pradiniai
momentai nepatenka i 'Triju momemtu lyginimo' metodo apibrezimo sriti");
   //dvieju momentu lyginimas is papiruso:)
   Tintens[1]=2/m1U;
   Tintens[3]=0.5/((m2U-m1U*m1U)/(m1U*m1U));
   Tintens[2]=Tintens[1]*Tintens[3];
  }
  else
  {
  //--------triju momentu lyginima
  //Tintens[0] ir Tintens[4] reiktu istirnt nes nenaudosiu

   Tintens[0]=m1U;        //?? liamda U?
   Tintens[2]=(m2U/2 - m1U*m1U)/(m1U*m1U*m1U - 2*m1U*m2U/2 + m3U/6);        //miu2 U
   double y=(1-Tintens[2]*m1U)*(1-Tintens[2]*m1U) + 4*(Tintens[2]*Tintens[2])*(m2U/2-m1U*m1U);
    if (y<0) {ShowMessage("posaknis neigiamas"); y=-y;}
   Tintens[1]=(1 + Tintens[2]*m1U + sqrt(y) )/(2*m1U - 2*Tintens[2]*(m2U/2-m1U*m1U));        //miu1 U
   if (Tintens[1]<0)
     Tintens[1]=(1 + Tintens[2]*m1U - sqrt((1-Tintens[2]*m1U)*(1-Tintens[2]*m1U) + 4*(Tintens[2]*Tintens[2])*(m2U/2-
m1U*m1U)) )/(2*m1U - 2*Tintens[2]*(m2U/2-m1U*m1U));        //miu1 U
   Tintens[3]=Tintens[2]*(Tintens[1]*m1U-1) / (Tintens[2]*(Tintens[1]*m1U-1)+Tintens[1]);  //p1 U

  // double p2U=1-Tintens[3];//Tintens[1] / (Tintens[2]*(Tintens[1]*m1U-1)+Tintens[1]);
  }

 if (((m2D/pow(m1D,2)<2) && (m2D/pow(m1D,2)-m3D/(6*pow(m1D,3))<1)) || ((m2D/pow(m1D,2)>2) && (m2D/pow(m1D,2)-
m3D/(6*pow(m1D,3))>1)))
  {
   Memo1->Lines->Add("Bus taikoma dvieju momentu lyginimo aproksimacija mazejimo intensyvumams, nes apskaiciuoti pradiniai
momentai nepatenka i 'Triju momemtu lyginimo' metodo apibrezimo sriti");
   //dvieju momentu lyginimas is papiruso:)
   Tintens[5]=2/m1D;
   Tintens[7]=0.5/((m2D-m1D*m1D)/(m1D*m1D));
   Tintens[6]=Tintens[5]*Tintens[7];
  }
  else
  {
  //--------triju momentu lyginima
  //Tintens[0] ir Tintens[4] reiktu istirnt nes nenaudosiu

   Tintens[4]=m1D;        //?? liamda D?
   Tintens[6]=(m2D/2 - m1D*m1D)/(m1D*m1D*m1D - 2*m1D*m2D/2 + m3D/6);        //miu2 D
   double x=(1-Tintens[6]*m1D)*(1-Tintens[6]*m1D) + 4*(Tintens[6]*Tintens[6])*(m2D/2-m1D*m1D);
    if (x<0) {ShowMessage("posaknis neigiamas"); x=-x;}
   Tintens[5]=(1 + Tintens[6]*m1D + sqrt(x) )/(2*m1D - 2*Tintens[6]*(m2D/2-m1D*m1D));        //miu1 D
   if (Tintens[5]<0)
     Tintens[5]=(1 + Tintens[6]*m1D - sqrt((1-Tintens[6]*m1D)*(1-Tintens[6]*m1D) + 4*(Tintens[6]*Tintens[6])*(m2D/2-
m1D*m1D)) )/(2*m1D - 2*Tintens[6]*(m2D/2-m1D*m1D));        //miu1 D
   Tintens[7]=Tintens[6]*(Tintens[5]*m1D-1) / (Tintens[6]*(Tintens[5]*m1D-1)+Tintens[5]);  //p1 D

   //double p2D=1-Tintens[7];//Tintens[5] / (Tintens[6]*(Tintens[5]*m1D-1)+Tintens[5]);
  }

   /*
//triju momentu lyginimas is papiruso:)
float uU=(6*m1U*m2U-2*m3U)/(3*m2U*m2U-2*m1U*m3U);
float vU=(12*m1U*m1U-6*m2U)/(3*m2U*m2U-2*m1U*m3U);

Tintens[1]=uU-sqrt(uU*uU-4*vU);                              //m1 U
Tintens[2]=uU+sqrt(uU*uU-4*vU);                              //m2 U
Tintens[3]=(1/Tintens[1])*Tintens[2]*(m1U*Tintens[1]-1);  //p2 U

float uD=(6*m1D*m2D-2*m3D)/(3*m2D*m2D-2*m1D*m3D);
float vD=(12*m1D*m1D-6*m2D)/(3*m2D*m2D-2*m1D*m3D);

Tintens[5]=uD-sqrt(uD*uD-4*vD);                              //m1 U
```

```
Tintens[6]=uD+sqrt(uD*uD-4*vD);                              //m2 U
Tintens[7]=(1/Tintens[1])*Tintens[2]*(m1U*Tintens[1]-1);  //p2 U
     */
/*
 Tintens[1]=1/Tintens[1];
 Tintens[2]=1/Tintens[2];
 Tintens[5]=1/Tintens[5];
 Tintens[6]=1/Tintens[6];
*/

 if ((Tintens[1]<0) || (Tintens[5]<0))
   Memo1->Lines->Add("miu1 U arba miu1 D yra neigiamas!!");
 Memo1->Lines->Add("liamda U: "+FloatToStrF(Tintens[0],ffFixed,5,4));
 Memo1->Lines->Add("miu1 U: "+FloatToStrF(Tintens[1],ffFixed,5,4));
 Memo1->Lines->Add("miu2 U: "+FloatToStrF(Tintens[2],ffFixed,5,4));
 Memo1->Lines->Add("p1 U: "+FloatToStrF(Tintens[3],ffFixed,5,4));
 //Memo1->Lines->Add("p2 U: "+FloatToStrF(p2U,ffFixed,5,4));
 Memo1->Lines->Add("liamda D: "+FloatToStrF(Tintens[4],ffFixed,5,4));
 Memo1->Lines->Add("miu1 D: "+FloatToStrF(Tintens[5],ffFixed,5,4));
 Memo1->Lines->Add("miu2 D: "+FloatToStrF(Tintens[6],ffFixed,5,4));
 Memo1->Lines->Add("p1 D: "+FloatToStrF(Tintens[7],ffFixed,5,4));
 //Memo1->Lines->Add("p2 D: "+FloatToStrF(p2D,ffFixed,5,4));


}
//---------------------------------------------------------------------
void TForm1::BusenuRadimas()
{
 int cikl_kint=0;
 bool yra;                  //ziurim ar yra nors wiena laiswa kasa
 int i,j,ii;
 float ints;                //intensyvumas

 int *x;                   //tiriamosios busenos coo
 TBus *D1;                 //pagalbine rodykle
 TBus *D2;                 //pagalbine rodykle
 TBus *Dck;                 //pagalbine rodykle,rodo i ta bsuena kurios numeris cikl_kint
 TBusIntens *D;             //pagalbine rodykle

 x=new int[K];
 D2=TBPr;
 Dck=TBPr;


 while(cikl_kint<N)
  {
   for (j=0;j<K;j++)
    x[j]=Dck->sk[j];
   yra=false;

//tikrinam srautus
 for (ii=0;ii<R;ii++)
  {
   radom=false;
   ints=Busenos(x,ii);
   //ints=0.1;
   if (radom)
   {
    if(!PatikrinimasVienatiskumo(x,cikl_kint,ints))
     {
      D1=new TBus;
      D1->sk=new int[K];
      for(j=0;j<K;j++)
       D1->sk[j]=x[j];
      D1->kitas=NULL;
      D2->kitas=D1;
      D2=D2->kitas;

      D=new TBusIntens;
      D->BI.BInt=ints;
      D->BI.eil=cikl_kint;
      D->BI.st=N;
      D->kitas=TPr;
      TPr=D;
      N++;
     }
    for (j=0;j<K;j++)   //atstatom i Dck->sk busena
     x[j]=Dck->sk[j];
   }
  }

  //tikrinam aptarnavimo aparatus
  for (ii=R;ii<M;ii++)
  {
```

```
   radom=false;
   ints=Busenos(x,ii);
   //ints=0.1;
   if (radom)
   {
    if(!PatikrinimasVienatiskumo(x,cikl_kint,ints))
     {
      D1=new TBus;
      D1->sk=new int[K];
      for(j=0;j<K;j++)
       D1->sk[j]=x[j];
      D1->kitas=NULL;
      D2->kitas=D1;
      D2=D2->kitas;

      D=new TBusIntens;
      D->BI.BInt=ints;
      D->BI.eil=cikl_kint;
      D->BI.st=N;
      D->kitas=TPr;
      TPr=D;
      N++;
     }
    for (j=0;j<K;j++)   //atstatom i Dck->sk busena
     x[j]=Dck->sk[j];
   }
  }
  cikl_kint++;
  Dck=Dck->kitas;
  } //while(cikl_kint<N) pab


}
//---------------------------------------------------------------------
void TForm1::BusenuRadimasA()
{
 int cikl_kint=0;
 bool yra;                  //ziurim ar yra nors wiena laiswa kasa
 int i,j,ii;
 float ints;                //intensyvumas

 double *x;                  //tiriamosios busenos coo
 TBusIntens *D;             //pagalbine rodykle
 TBusA *D1;                //pagalbine rodykle
 TBusA *D2;                //pagalbine rodykle
 TBusA *Dck;                //pagalbine rodykle,rodo i ta bsuena kurios numeris cikl_kint

 x=new double[K];

 Dck=new TBusA;
 Dck->sk=new double[3];
 Dck->kitas=NULL;
 TBAPr=Dck;
 Dck->sk[0]=AkcMin + int(IntSk/2)*IntIlgis;
 if (m1U>=m1D) {Dck->sk[1]=1; Dck->sk[2]=0;}
       else {Dck->sk[1]=0; Dck->sk[2]=1;}
 if (Dck->sk[0]==AkcMax) {Dck->sk[1]=0; Dck->sk[2]=1;}
       else if (Dck->sk[0]==AkcMin) {Dck->sk[1]=1; Dck->sk[2]=0;}
 D2=TBAPr;
 Dck=TBAPr;

 while(cikl_kint<N)
  {
   for (j=0;j<K;j++)
    x[j]=Dck->sk[j];
   yra=false;

//tikrinam srautus
 for (ii=0;ii<R;ii++)
  {
   radom=false;
   ints=BusenosA(x,ii);
   if (radom)
   {
    if(!PatikrinimasVienatiskumoA(x,cikl_kint,ints))
     {
      D1=new TBusA;
      D1->sk=new double[K];
      for(j=0;j<K;j++)
       D1->sk[j]=x[j];
      D1->kitas=NULL;
      D2->kitas=D1;
      D2=D2->kitas;
```

```
      D=new TBusIntens;
      D->BI.BInt=ints;
      D->BI.eil=cikl_kint;
      D->BI.st=N;
      D->kitas=TPr;
      TPr=D;
      N++;
     }
    for (j=0;j<K;j++)   //atstatom i Dck->sk busena
     x[j]=Dck->sk[j];
   }
  }

  //tikrinam aptarnavimo aparatus
  for (ii=R;ii<M;ii++)
  {
   radom=false;
   ints=BusenosA(x,ii);
   if (radom)
   {
    if(!PatikrinimasVienatiskumoA(x,cikl_kint,ints))
     {
      D1=new TBusA;
      D1->sk=new double[K];
      for(j=0;j<K;j++)
       D1->sk[j]=x[j];
      D1->kitas=NULL;
      D2->kitas=D1;
      D2=D2->kitas;

      D=new TBusIntens;
      D->BI.BInt=ints;
      D->BI.eil=cikl_kint;
      D->BI.st=N;
      D->kitas=TPr;
      TPr=D;
      N++;
     }
    for (j=0;j<K;j++)   //atstatom i Dck->sk busena
     x[j]=Dck->sk[j];
   }
  }
  cikl_kint++;
  Dck=Dck->kitas;
  } //while(cikl_kint<N) pab

}
//---------------------------------------------------------------------------

float TForm1::Busenos(int *xx, int poz)
{

//************aproksimacija 1srautas 2 aptarnavimo sistemos su kokybes tikrinimu

switch (poz)
 {
  case 0:           //tikrinam e1-atvyko paraiska
  {
   if (((xx[1]<1)&&(xx[2]<1))&&(xx[0]+xx[1]+xx[2]+xx[3]+xx[4]+xx[5]<L)) //intensyvumu sk-srauto intens / 3 nes yra 1
aptarnavimo sistema+1aptarnavimo sistema su dviem isejimo galimybemis
   {
    xx[1]++;
    radom=true;
   }
   else if(xx[0]+xx[1]+xx[2]+xx[3]+xx[4]+xx[5]<L)
   {
    xx[0]++;
    radom=true;
   }
   return Tintens[3];
  }   //case 0 pab

  case 1:
  {
   if(xx[1]>0)
   {

    xx[2]++;  //pereinam i kita
    xx[1]--;
    //if(xx[0]>0) xx[0]--;
    //else xx[1]--;
    radom=true;
   }
   return Tintens[0]*Tintens[2];        //niu1*p1
```

```
} //case 1 pab

case 2:
{
 if(xx[1]>0)
 {
  if((xx[4]<1)&&(xx[5]<1)) xx[4]++;
  else xx[3]++;  //pereinam i kita
  if(xx[0]>0) xx[0]--;
  else xx[1]--;
  radom=true;
 }
 return Tintens[0]*(1-Tintens[2]);        //niu1*p2
} //case 2 pab

case 3:
{
 if(xx[2]>0)
 {
  if((xx[4]<1)&&(xx[5]<1)) xx[4]++;
  else xx[3]++;  //pereinam i kita
  if(xx[0]>0) {xx[0]--;xx[1]++;xx[2]--;}
  else xx[2]--;
  radom=true;
 }
 return Tintens[1]*(1-Tintens[2]);       //niu2*p2
} //case 3 pab

case 4:
{
 if(xx[4]>0)
 {
  xx[5]++;  //pereinam i kita
  xx[4]--;
  //if(xx[3]>0) xx[3]--;
  //else xx[4]--;
  radom=true;
 }
 return Tintens[0]*Tintens[2];        //niu1*p1
} //case 4 pab

case 5:       //iseinam lauk su P
{
 if(xx[4]>0)
 {
  if(xx[3]>0) xx[3]--;
  else xx[4]--;
  radom=true;
 }
 return Tintens[0]*(1-Tintens[2])*Tintens[4];//niu1*p2*P
} //case 5 pab

case 6:       //iseinam lauk su 1-P
{
 if(xx[4]>0)
 {
  if(xx[3]>0) xx[3]--;
  else xx[4]--;
  if ((xx[1]<1)&&(xx[2]<1)) xx[1]++;       //grazinam paraiska
  else xx[0]++;
  radom=true;
 }
 return Tintens[0]*(1-Tintens[2])*(1-Tintens[4]);        //niu1*p2
} //case 6 pab

case 7:       //iseinam lauk su P
{
 if(xx[5]>0)
 {
  if(xx[3]>0) {xx[3]--;xx[4]++;xx[5]--;}
  else xx[5]--;
  radom=true;
 }
 return Tintens[1]*(1-Tintens[2])*Tintens[4];       //niu2*p2
} //case 7 pab

case 8:           //iseinam lauk su 1-P
{
 if(xx[5]>0)
 {
  if(xx[3]>0) {xx[3]--;xx[4]++;xx[5]--;}
  else xx[5]--;
  if ((xx[1]<1)&&(xx[2]<1)) xx[1]++;       //grazinam paraiska
  else xx[0]++;
```

```
    radom=true;
   }
   return Tintens[1]*(1-Tintens[2])*(1-Tintens[4]);       //niu2*p2
  } //case 8 pab

 } //switch pab

//************NEaproksimacija 1srautas 2 aptarnavimo sistemos su kokybes tikrinimu
 /*
switch (poz)
 {
  case 0:           //tikrinam e1-atvyko paraiska
  {
   if ((xx[1]<(IK-1)/3)&&(xx[0]+xx[1]+xx[2]+xx[3]<L)) //intensyvumu sk-srauto intens / 3 nes yra 1 aptarnavimo
sistema+1aptarnavimo sistema su dviem isejimo galimybemis
   {
    xx[1]++;
    radom=true;
   }
   else if(xx[0]+xx[1]+xx[2]+xx[3]<L)
   {
    xx[0]++;
    radom=true;
   }
   return Tintens[0];
  }   //case 0 pab

  case 1:
  {
   if(xx[1]>0)
   {
    if (xx[3]<(IK-1)/3)  xx[3]++;  //pereinam i kita
    else  xx[2]++;                 //faze.
    if (xx[0]>0) xx[0]--;   //sumazinam eile
    else xx[1]--;           //sumazinam uzimtu aparatu sk
    radom=true;
   }
   return Tintens[1];
  } //case 1 pab

  case 2:
  {
   if(xx[3]>0)
   {
    if (xx[2]>0) xx[2]--;   //sumazinam eile
    else xx[3]--;           //sumazinam uzimtu aparatu sk
    radom=true;
   }
   return Tintens[2]*Tintens[3];
  } //case 2 pab

  case 3:
  {
   if(xx[3]>0)
   {
    if (xx[1]<(IK-1)/3)  xx[1]++;  //sugrazinam
    else  xx[0]++;                 //paraiska.
    if (xx[2]>0) xx[2]--;   //sumazinam eile
    else xx[3]--;           //sumazinam uzimtu aparatu sk
    radom=true;
   }
   return Tintens[2]*(1-Tintens[3]);
  } //case 3 pab
 } //switch pab
*/
//************aproksimacija 1 srautas n aparatu
/*
 switch (poz)
 {
  case 0:           //tikrinam e1-atvyko paraiska
  {
  int i=0;
   while ((i<K)&&(radom==false))
   {
    if (xx[i]==0)
    {
     xx[i]=1;
     xx[i+1]=1;
     radom=true;
    } //if (xx[i]==0) pab
    i=i+2;
   } //while pab
   return Tintens[3];
  }  //case 0 pab
```

```
  case 1:          //1 aptarnavimo aparate paraiska baigta aptarnauti su tikimybe p1
  {
   if (xx[0]==1)
    if (xx[1]==1)
    {
     xx[1]=2;
     radom=true;
    }
   return Tintens[0]*(1-Tintens[2]);//0.74437*(1-0.018503); //niu[0]*p[1];
  }  //case 1 pab
  case 2:       //1 aptarnavimo aparate paraiska baigta aptarnauti su tikimybe p2
  {
   if (xx[0]==1)
    if (xx[1]==1)
    {
     xx[0]=0;
     xx[1]=0;
     radom=true;
    }
   return Tintens[0]*Tintens[2];//0.74437*0.018503; //niu[0]*p[0];
  }  //case 2 pab
  case 3:       //1 aptarnavimo aparate paraiska baigta aptarnauti antroje fazeje
  {
   if (xx[0]==1)
    if (xx[1]==2)
    {
     xx[0]=0;
     xx[1]=0;
     radom=true;
    }
   return Tintens[1]*(1-Tintens[2]);//0.227796*(1-0.018503); //niu[1]*p[1];
  }  //case 3 pab

  case 4:          //2 aptarnavimo aparate paraiska baigta aptarnauti su tikimybe p1
  {
   if (xx[2]==1)
    if (xx[3]==1)
    {
     xx[3]=2;
     radom=true;
    }
   return Tintens[0]*(1-Tintens[2]);//0.74437*(1-0.018503); //niu[0]*p[1];
  }  //case 1 pab
  case 5:       //2 aptarnavimo aparate paraiska baigta aptarnauti su tikimybe p2
  {
   if (xx[2]==1)
    if (xx[3]==1)
    {
     xx[2]=0;
     xx[3]=0;
     radom=true;
    }
   return Tintens[0]*Tintens[2];//0.74437*0.018503; //niu[0]*p[0];
  }  //case 2 pab
  case 6:       //2 aptarnavimo aparate paraiska baigta aptarnauti antroje fazeje
  {
   if (xx[2]==1)
    if (xx[3]==2)
    {
     xx[2]=0;
     xx[3]=0;
     radom=true;
    }
   return Tintens[1]*(1-Tintens[2]);//0.227796*(1-0.018503); //niu[1]*p[1];
  }  //case 6 pab
 } //switch pab
*/

 /*
//************aproksimacija 2 srautai 1 aparatas
 switch (poz)
{
 case 0:           //tikrinam e1
  {
   if (xx[2]==0)
   {
    xx[0]=1;          //ar gb cia >0 bet xx[2]=0?
    xx[2]=1;
    xx[3]=1;
    radom=true;
   }
   else
    if ((xx[0]+xx[1])<L)
    {
```

```
    xx[0]++;
    radom=true;
   }
  return Tintens[3];
 }
case 1:           //tikrinam e2
 {
  if (xx[2]==0)
  {
   xx[1]=1;
   xx[2]=2;
   xx[3]=1;
   radom=true;
  }
  else
   if ((xx[0]+xx[1])<L)
   {
    xx[1]++;
    radom=true;
   }
  return Tintens[4];
 }
case 2:           //tikrinam e3
 {
  if (xx[3]==1)
  {
   if (xx[2]==1)
    {
     xx[0]--;
     radom=true;
    }
   else
    {
     xx[1]--;
     radom=true;
    }
   if (xx[0]>0)
    {
     xx[2]=1;
     radom=true;
    }
   else
    {
    if (xx[1]>0)
     {
      xx[2]=2;
      radom=true;
     }
    else
     {
      xx[2]=0;
      xx[3]=0;
      radom=true;
     }
    }
  }
  return Tintens[0]*(1-Tintens[2]);//0.74437*(1-0.018503); //niu[0]*p[1];
 }
case 3:           //tikrinam e4
 {
  if (xx[3]==1)
  {
   xx[3]=2;
   radom=true;
  }
  return Tintens[0]*Tintens[2];//0.74437*0.018503; //niu[0]*p[0];
 }
case 4:           //tikrinam e5
 {
  if (xx[3]==2)
  {
   if (xx[2]==1)
    {
     xx[0]--;
     radom=true;
    }
   else
    {
     xx[1]--;
     radom=true;
    }
   if (xx[0]>0)
    {
     xx[2]=1;
```

```
      xx[3]=1;
      radom=true;
     }
    else
     {
     if (xx[1]>0)
      {
       xx[2]=2;
       xx[3]=1;
       radom=true;
      }
     else
      {
       xx[2]=0;
       xx[3]=0;
       radom=true;
      }
     }
   }
   return Tintens[1]*(1-Tintens[2]);//0.227796*(1-0.018503); //niu[1]*p[1];
  }
 } //switch pab
 */
}

//---------------------------------------------------------------------
float TForm1::BusenosA(double *xx, int poz)
{
//************aproksimacija su akcijom
float Udaugiklis=(m1U/(m1D+m1U));
float Ddaugiklis=1-Udaugiklis;//=(m1D/(m1D+m1U));
 switch (poz)
{
 case 0:           //tikrinam e1u
  {
   if (xx[1]==1)
   {
    xx[1]=2;
    radom=true;
   }
   return Tintens[1]*Tintens[3]*0.5;
  }
 case 1:           //tikrinam e1d
  {
   if (xx[2]==1)
   {
    xx[2]=2;
    radom=true;
   }
   return Tintens[5]*Tintens[7]*0.5;
  }
 case 2:           //tikrinam e2ud
  {
   if (xx[1]==1)        //nereikia tikrinti ar galima sumazinti kaina nes ka tik padidinom
   {
    xx[2]=1;
    xx[1]=0;
    xx[0]=xx[0]+IntIlgis;
    radom=true;
   }
   return Tintens[1]*(1-Tintens[3])*Ddaugiklis;
  }
 case 3:           //tikrinam e2uu
  {
   if ((xx[1]==1)&&(xx[0]+2*IntIlgis<=AkcMax+IntIlgis/1000))     //paklaida IntIlgis/1000
   {
    xx[0]=xx[0]+IntIlgis;
    radom=true;
   }
   return Tintens[1]*(1-Tintens[3])*Udaugiklis;       //ar tikrai lygus intensyvumai po pirmos fazes kainai didet ar mazet
  }
 case 4:           //tikrinam e2dd
  {
   if ((xx[2]==1)&&(xx[0]-2*IntIlgis>=AkcMin-IntIlgis/1000))
   {
    xx[0]=xx[0]-IntIlgis;
    radom=true;
   }
   return Tintens[5]*(1-Tintens[7])*Ddaugiklis;
  }
 case 5:           //tikrinam e2du
  {
   if (xx[2]==1)  //nereikia tikrinti ar galima didinti kaina nes ka tik sumaiznom
   {
```

```
    xx[1]=1;
    xx[2]=0;
    xx[0]=xx[0]-IntIlgis;
    radom=true;
   }
   return Tintens[5]*(1-Tintens[7])*Udaugiklis;       //ar tikrai lygus intensyvumai po pirmos fazes kainai didet ar mazet
  }
 case 6:           //tikrinam e3ud
  {
   if (xx[1]==2)        //nereikia tikrinti ar galima sumazinti kaina nes ka tik padidinom
   {
    xx[2]=1;
    xx[1]=0;
    xx[0]=xx[0]+IntIlgis;
    radom=true;
   }
   return Tintens[2]*(1-Tintens[3])*Ddaugiklis;
          //Tintens[2]*Ddaugiklis;
  }
 case 7:           //tikrinam e3uu
  {
   if ((xx[1]==2)&&(xx[0]+2*IntIlgis<=AkcMax+IntIlgis/1000))     //paklaida IntIlgis/1000
   //if ((xx[1]==2)&&(int((xx[0]+2*IntIlgis)*100000)<=int(AkcMax*100000)))
   {
    xx[1]=1;
    xx[0]=xx[0]+IntIlgis;
    radom=true;
   }
   return Tintens[2]*(1-Tintens[3])*Udaugiklis;
          //Tintens[2]*Udaugiklis;
  }
 case 8:           //tikrinam e3dd
  {
   if ((xx[2]==2)&&(xx[0]-2*IntIlgis>=AkcMin-IntIlgis/1000))     //paklaida IntIlgis/1000
   //if ((xx[2]==2)&&(int((xx[0]+2*IntIlgis)*100000)<=int(AkcMax*100000)))
   {
    xx[2]=1;
    xx[0]=xx[0]-IntIlgis;
    radom=true;
   }
   return Tintens[6]*(1-Tintens[7])*Ddaugiklis;
          //Tintens[6]*Ddaugiklis;
  }
 case 9:           //tikrinam e3du
  {
   if (xx[2]==2)        //nereikia tikrinti ar galima sumazinti kaina nes ka tik padidinom
   {
    xx[1]=1;
    xx[2]=0;
    xx[0]=xx[0]-IntIlgis;
    radom=true;
   }
   return Tintens[6]*(1-Tintens[7])*Udaugiklis;
          //Tintens[6]*Udaugiklis;
  }
 } //switch pab

}
//---------------------------------------------------------------------
bool TForm1::PatikrinimasVienatiskumo(int s[], int Bnr, double intens) //Bnr busenos numeris is kurios atejom i surasta
busena
{
 bool v=false;
 TBusIntens *D;
 ///TBusA *D1;
 ///D1=TBAPr;
 TBus *D1;
 D1=TBPr;
 int i=0,j;
 while((i<N)&&(!v))
 {
     j=0;
     v=true;
     while((j<K)&&(v))
      {
       ///if (int(s[j]*100000)!=int(D1->sk[j]*100000))
       if (s[j]!=D1->sk[j])
        {
          v=false;
        }
       j++;
      }
    //}
  D1=D1->kitas;
```

```
  i++;
 }

 if (v)
  {
   D=new TBusIntens;
   D->BI.BInt=intens;
   D->BI.eil=Bnr;
   D->BI.st=i-1;
   D->kitas=TPr;
   TPr=D;
  }
 return v;   //jeigu tokia busena jau buwo grazins true
}
//---------------------------------------------------------------------
bool TForm1::PatikrinimasVienatiskumoA(double s[], int Bnr, double intens) //Bnr busenos numeris is kurios atejom i
surasta busena
{
 bool v=false;
 TBusIntens *D;
 TBusA *D1;
 D1=TBAPr;
 int i=0,j;
 while((i<N)&&(!v))
 {
     j=0;
     v=true;
     while((j<K)&&(v))
      {
       if (int(s[j]*100000)!=int(D1->sk[j]*100000))
        {
          v=false;
        }
       j++;
      }
    //}
  D1=D1->kitas;
  i++;
 }

 if (v)
  {
   D=new TBusIntens;
   D->BI.BInt=intens;
   D->BI.eil=Bnr;
   D->BI.st=i-1;
   D->kitas=TPr;
   TPr=D;
  }
 return v;   //jeigu tokia busena jau buwo grazins true
}
//---------------------------------------------------------------------

void TForm1::KasuSk(AnsiString DF)
{
 char elem='a';
 float elem1=0;
 FILE *F;
 F = fopen(DF.c_str(),"r");
 K=0;
 IK=0;

 //nuskaitom kiek bus busenos skaitmenu
 while (elem!='=') fscanf(F, "%c", &elem);
 fscanf(F, "%c", &elem);
 while (elem!=';')
  {
   fscanf(F, "%f%c", &elem1, &elem);
   K++;
  }

 elem='a';
 elem1=0.;
 //nuskaitom kiek bus intensyvumu
 while (elem!='=') fscanf(F, "%c", &elem);
 fscanf(F, "%c", &elem);
 while (elem!=';')
  {
   fscanf(F, "%f%c", &elem1, &elem);
   IK++;
  }
 fclose(F);
}
//---------------------------------------------------------------------------
```

```
void TForm1::SurastiFaziuIntens()       //surasti faziu intensyvumus miu1, miu2, p1
{
 float g1,g2,g3;
 g1=Tintens[0];
 g2=Tintens[1]/2;
 g3=Tintens[2]/6;

 //tikrinsim ar parametrai patenka metodo A.S.
 if (((g2/pow(g1,2)<1) && (g2/pow(g1,2)-g3/(2*pow(g1,3))<0.5)) || ((g2/pow(g1,2)>1) && (g2/pow(g1,2)-
g3/(2*pow(g1,3))>0.5)))
  {
   Memo1->Lines->Add("Bus taikoma dvieju momentu lyginimo aproksimacija , nes pateikti pradiniai momentai nepatenka i
'Triju momemtu lyginimo' metodo apibrezimo sriti");
   Tintens[0]=2/g1;
   Tintens[2]=0.5/((2*g2-pow(g1,2))/pow(g1,2));
   Tintens[1]=Tintens[0]*Tintens[2];
   Tintens[1]=Tintens[1]/(1-Tintens[2]);        //todel kad dvieju ir triju momentu lyginimo atvejais nagrinejamos
skirtingos sistemos schemos - antro aprato intens=miu2
  }
  else
  {
   Tintens[1]=(g2-pow(g1,2))/(pow(g1,3)-2*g1*g2+g3);
   float posaknis=pow((1-Tintens[1]*g1),2)+4*pow(Tintens[1],2)*(g2-pow(g1,2));
   if (posaknis<0)
     {ShowMessage("Posaknis neigiamas! Aproksimacija su tokiais pradiniais momentais negali buti atlikta."); posaknis=-
posaknis;}
   Tintens[0]=(1+Tintens[1]*g1+sqrt(posaknis))/(2*g1-2*Tintens[1]*(g2-pow(g1,2)));
   if (Tintens[0]<0)
     (1+Tintens[1]*g1-sqrt(posaknis))/(2*g1-2*Tintens[1]*(g2-pow(g1,2)));
   Tintens[2]=Tintens[1]*(Tintens[0]*g1-1)/(Tintens[1]*(Tintens[0]*g1-1)+Tintens[0]);
  }
}
//---------------------------------------------------------------------------
void TForm1::Statistika_S1()       //ivairi statistika sistemai S1- 1 eile su ribotu ilgiu ir n aptranavimo aparatu
{
 int i=0,k=0;
 TBus *X=TBPr;

 //surasim kiek yra auksciausio lygio busenu(ty pilnai uzpildytu sisitemu) ir kokie ju nr
 while (X)
 {
  if (X->sk[0]==L-(M-R)) k=i;
  X=X->kitas;
  i++;
 }

float A;          //pralaidumas
A=Tintens[0]*(1-r[k]);

 Memo2->Lines->Add("");
 Memo2->Lines->Add("Sistemos pralaidumas- "+FloatToStrF(A,ffFixed,5,4));
 Memo2->Lines->Add("");

float k_=0;       //kiek vidutiniskai uzimtu AI
float *tik_suma=new float[K-1];
int lygis;

 X=TBPr;
 for(i=0;i<K-1;i++) tik_suma[i]=0;
 i=0;
  while (X)
  {
  k=1;
  lygis=0;
   while (k<K)
   {
    lygis=lygis+X->sk[k];      //surandam kox busenos lygis(kiek uzimtu AI)
    k++;
   }
   if (lygis>0)
    tik_suma[lygis-1]=tik_suma[lygis-1]+r[i];    //ir surandam kokia tikimybe kad bus to lygio busena

   X=X->kitas;
   i++;
  }

 i=1;
 while (i<K)
 {
 k_=k_+i*tik_suma[i-1];
 i++;
 }

 Memo2->Lines->Add("Vidutinis uzimtu aptarnavimo irenginiu skaicius- "+FloatToStrF(k_,ffFixed,5,2));
```

```
 Memo2->Lines->Add("");

float r_=0;      //vidutine eile
 k=0;
 X=TBPr;
 while (X)
 {
  if (X->sk[0]>0) r_=r_+X->sk[0]*r[k];
  X=X->kitas;
  k++;
 }

 Memo2->Lines->Add("Vidutinio ilgio eile- "+FloatToStrF(r_,ffFixed,5,2));
 Memo2->Lines->Add("");

float z_;       //vidutiniskai sistemoje esanciu paraisku sk
z_=r_+k_;

 Memo2->Lines->Add("vidutiniskai sistemoje esanciu paraisku skaicius- "+FloatToStrF(z_,ffFixed,5,2));
 Memo2->Lines->Add("");

float t_sist,t_eil;     //uztrukimas sistemoje ir eileje

 t_eil=r_/Tintens[0];
 t_sist=z_/Tintens[0];

 Memo2->Lines->Add("Pagal Litlio formule apskaiciuotas:");
 Memo2->Lines->Add("Vidutinis uztrukimas eileje- "+FloatToStrF(t_eil,ffFixed,5,2));
 Memo2->Lines->Add("Vidutinis uztrukimas sistemoje- "+FloatToStrF(t_sist,ffFixed,5,2));
 Memo2->Lines->Add("");

}
//---------------------------------------------------------------------
void TForm1::Statistika_S2()       //ivairi statistika sistemai S2- 2 srautai(su absoliuciu pirmumu) ir n aptranavimo
aparatu
{
 int i=0,k=0;
 TBus *X=TBPr;
 int *nrmas;    //masyvas numeriu tu busenu kuriu lygis max
 nrmas=new int[L+2];  //L+1 narys busena (L;0)
 //surasim kiek yra auksciausio lygio busenu(ty pilnai uzpildytu sisitemu) ir kokie ju nr
 while (X)
 {
  if (X->sk[0]+X->sk[1]==L)
  {
   nrmas[i]=k;
   i++;
  }
  if (X->sk[0]==L) nrmas[L+1]=k;
  X=X->kitas;
  k++;
 }

//paraisku atmetimo tikimybes
float P1ref,P2ref=0,Pref;

P1ref=r[nrmas[L+1]];
for(i=0;i<L+1;i++) P2ref=P2ref+r[nrmas[i]];
Pref=Tintens[0]/(Tintens[0]+Tintens[1])*P1ref+Tintens[1]/(Tintens[0]+Tintens[1])*P2ref;

 Memo2->Lines->Add("");
 Memo2->Lines->Add("Tikimybe kad bus atmesta pirmo tipo paraiska- "+FloatToStrF(P1ref,ffFixed,5,4));
 Memo2->Lines->Add("Tikimybe kad bus atmesta antro tipo paraiska- "+FloatToStrF(P2ref,ffFixed,5,4));
 Memo2->Lines->Add("Tikimybe kad paraiska bus atmesta- "+FloatToStrF(Pref,ffFixed,5,4));
 Memo2->Lines->Add("");

float A1,A2=0,A;  //pralaidumas
A1=Tintens[0]*(1-P1ref);
A2=Tintens[1]*(1-P2ref);
A=(A1+A2)/2;

 Memo2->Lines->Add("Pirmo tipo paraisku pralaidumas- "+FloatToStrF(A1,ffFixed,5,4));
 Memo2->Lines->Add("Antro tipo paraisku pralaidumas- "+FloatToStrF(A2,ffFixed,5,4));
 Memo2->Lines->Add("Vidutinis sistemos pralaidumas- "+FloatToStrF(A,ffFixed,5,4));
 Memo2->Lines->Add("");

//uzimtu aptranavimo aparatu skaicius
float kk1,kk2,kk;
kk1=A1/Tintens[2];
kk2=A2/Tintens[3];
kk=kk1+kk2;

 Memo2->Lines->Add("Tiketinas pirmo tipo paraisku uzimamu serveriu skaicius- "+FloatToStrF(kk1,ffFixed,5,4));
 Memo2->Lines->Add("Tiketinas antro tipo paraisku uzimamu serveriu skaicius- "+FloatToStrF(kk2,ffFixed,5,4));
```

```
 Memo2->Lines->Add("Uzimtu serveriu skaicius- "+FloatToStrF(kk,ffFixed,5,2));
 Memo2->Lines->Add("");

}
//---------------------------------------------------------------------------
void TForm1::Statistika_S3()       //ivairi statistika sistemai S3- 2 srautai(su paprastu pirmumu) 1 aptranavimo aparatu
ir 2 kvazi aparatais
{
 float L1=0; //vidutinis 1 klases klientu eiles ilgis
 float L2=0;
 //float W1=0; //vidutinis 1 klases klientu laukimo laikas
 //float W2=0;
 TBus *X=TBPr;
 int i=0;
 while (X)
 {
  if (X->sk[2]==1)
   {
   if (X->sk[0]>1) L1=L1+(X->sk[0]-1)*r[i];
   if (X->sk[1]>0) L2=L2+X->sk[1]*r[i];
   }
  else //tas pats kaip- if (X->sk[2]==2)
   {
   if (X->sk[0]>0) L1=L1+X->sk[0]*r[i];
   if (X->sk[1]>1) L2=L2+(X->sk[1]-1)*r[i];
   }
  //if (X->sk[0]>1) L1=L1+(X->sk[0]-1)*r[i];
  //if (X->sk[1]>1) L2=L2+(X->sk[1]-1)*r[i];
  i++;
  X=X->kitas;
 }
 Memo1->Lines->Add("L1: "+ FloatToStrF(L1,ffFixed,5,4));
 Memo1->Lines->Add("L2: "+ FloatToStrF(L2,ffFixed,5,4));
 Memo1->Lines->Add("W1: "+ FloatToStrF(L1/Tintens[3],ffFixed,5,4));
 Memo1->Lines->Add("W2: "+ FloatToStrF(L2/Tintens[4],ffFixed,5,4));
}
//---------------------------------------------------------------------------
void TForm1::Statistika_S4()       //ivairi statistika sistemai S4- 1 srautas n aptranavimo aparatu
{
 //float L1=0; //vidutinis 1 klases klientu eiles ilgis
 //float L2=0;
 //float W1=0; //vidutinis 1 klases klientu laukimo laikas
 //float W2=0;
 TBus *X=TBPr;
 int i=0;
 float* A=new float[K/2+1];
 for(i=0;i<K/2+1;i++) A[i]=0;
 //for (int i=0;i<K/2;i++)
 //{
 i=0;
  while (X)
  {
   if (X->sk[0]==0) A[0]=A[0]+r[i];
   //if ((X->sk[0]==0)&&(X->sk[2]==0)) A[0]=A[0]+r[i];
   //if (((X->sk[0]==0)&&(X->sk[2]==1))||((X->sk[0]==1)&&(X->sk[2]==0))) A[1]=A[1]+r[i];
   //if ((X->sk[0]==1)&&(X->sk[2]==1)) A[2]=A[2]+r[i];
   X=X->kitas;
   i++;
  }
  A[1]=1-A[0];
 //}
 /*
 float Pn=0;            //tikimybe kad sistema pilnai uzimta
  while (X)
  {
   for (i=0;i<K;i=i+2)
   {
    if (X->sk[i]>0)
   }
  } */

 Memo1->Lines->Add("A0: "+ FloatToStrF(A[0],ffFixed,5,4));
 Memo1->Lines->Add("A1: "+ FloatToStrF(A[1],ffFixed,5,4));
 //Memo1->Lines->Add("A2: "+ FloatToStrF(A[2],ffFixed,5,4));
}
//---------------------------------------------------------------------------
void TForm1::Statistika_S5()       //ivairi statistika sistemai S5- 1 srautas 2 aptranavimo sistemos
{
 //float L1=0; //vidutinis 1 eiles ilgis
 //float L2=0;
 TBus *X=TBPr;
 int i=0;
 float* A=new float[3];
 for(i=0;i<K/2+1;i++) A[i]=0;
```

```
 i=0;
  while (X)
  {
   A[0]=A[0]+X->sk[0]*r[i];
   A[1]=A[1]+X->sk[3]*r[i];
   X=X->kitas;
   i++;
  }
  A[2]=1-A[0]-A[1];

 Memo1->Lines->Add("A0: "+ FloatToStrF(A[0],ffFixed,5,4));
 Memo1->Lines->Add("A1: "+ FloatToStrF(A[1],ffFixed,5,4));
 Memo1->Lines->Add("A2: "+ FloatToStrF(A[2],ffFixed,5,4));
}
//---------------------------------------------------------------------------
void TForm1::Statistika_S6()       //ivairi statistika akciju sistemai su dviem fazem
{
 TBusA *X,*Y=TBAPr;
 TBusA *TBAPr_Apj,*D1,*D2;      //apjungtos akciju kainos, despite faziu
 TBAPr_Apj=new TBusA;
 TBAPr_Apj->sk=new double[3];
 TBAPr_Apj->sk[0]=AkcMin;
 TBAPr_Apj->sk[1]=0;
 TBAPr_Apj->sk[2]=0;
 TBAPr_Apj->kitas=NULL;
 D2=TBAPr_Apj;
 r_apj=new float[IntSk];
 int i,j=0;
 int pirmas;

 while (int((AkcMin+j*IntIlgis)*100000)<=int(AkcMax*100000))
 {
  i=0;
  X=TBAPr;
  r_apj[j]=0;
  pirmas=1;
  while (X)
   {
    if (int(X->sk[0]*1000000)==int((AkcMin+j*IntIlgis)*1000000))
    {/*
     if ((j!=0)&&(pirmas==1))
     {
      D1=new TBusA;
      D1->sk=new double[3];
      D1->sk[0]=X->sk[0];
      D1->sk[1]=0;
      D1->sk[2]=0;
      D1->kitas=NULL;
      D2->kitas=D1;
      D2=D2->kitas;
      pirmas=0;
     }*/
     r_apj[j]=r_apj[j]+r[i];
    }
    X=X->kitas;
    i++;
   }
  j++;
 }
 D1=TBAPr_Apj;
 i=0;
 float maxR=0;
 int maxI=i;
 //////Memo1->Lines->Add("Apjungtos stac. tikimybes > 0.00005");
 //while (D1)
 for(i=0;i<=IntSk;i++)
 {
  if (r_apj[i]>0.00005) //Memo1->Lines->Add("["+FloatToStrF(D1->sk[0]-IntIlgis/2,ffFixed,7,2)+"-"+FloatToStrF(D1-
>sk[0]+IntIlgis/2,ffFixed,7,2)+"]"+" - "+FloatToStrF(r_apj[i],ffFixed,5,4));
        Memo1->Lines->Add("["+FloatToStrF(AkcMin+i*IntIlgis-IntIlgis/2,ffFixed,7,2)+"-
"+FloatToStrF(AkcMin+i*IntIlgis+IntIlgis/2,ffFixed,7,2)+"]"+" - "+FloatToStrF(r_apj[i],ffFixed,5,4));
  //D1=D1->kitas;
  if (r_apj[i]>maxR) {maxR=r_apj[i]; maxI=i;}
  //i++;
  //D1=NULL;     //!!!!!!!!!!!!!!!!!!!!!!!
 }
 //maxI=maxI-5;     //!!!!!!!!!!!!!!!!!!!!!!   maxI+10
 for (i=0;i<KK;i++)
 {
 float aa=AkcMin+maxI*IntIlgis-IntIlgis/2;
  Series2->AddXY(i,aa,"",clGreen);
  Series3->AddXY(i,AkcMin+maxI*IntIlgis+IntIlgis/2,"",clGreen);
 }
}
```

```
//---------------------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
{
 struct  time t1;
 struct  time t2;

 Series1->Clear();
 Series2->Clear();
 Series3->Clear();
 Memo1->Lines->Clear();
 Memo2->Lines->Clear();
 TabSheet2->TabVisible=false;
 K=0;
 M=0;

 IvestiPradSalygas();
 //KK=0;
 //NuskaitytiAkcKainas();
 //Apsukti();
 //Grafika();

 //gettime(&t1);
 //Memo1->Lines->Add(FloatToStrF(t1.ti_hour,ffFixed,2,0) +"-"+FloatToStrF(t1.ti_min,ffFixed,2,0)+"-
"+FloatToStrF(t1.ti_sec,ffFixed,2,0)+"-"+FloatToStrF(t1.ti_hund,ffFixed,2,0));
 //"The current time is: %2d:%02d:%02d.%02d\n",
   //       t1.ti_hour, t1.ti_min, t1.ti_sec, t1.ti_hund);

 //AukstynZemynCount();
 //AukstynZemynCountSizeBased();
 //AukstynZemynStats();
 /*
 R=0;
 M=10;
 N=1;
 K=3;
 IK=8;
 */
 if (M>0)
 {
  BusenuRadimas();
  Apjungimas();
  if (CheckBox1->Checked==true) IsvestiStrGrid();
  else TabSheet3->TabVisible=false;
  StacTikRadimas();
  IsvestiMemo();
  SurastiMax();
  Normavimas();         //!!!
  Statistika_S5();
  //OpcIkainojimas();
  //  Apjungimas();
  /*
  StacTikRadimas();
  IsvestiMemo();
  SurastiMax();

  Statistika_S5();
 // Statistika_S4();   //aproksimacija 1srautas n aparatu
 // Statistika_S3();                    //skaiciuoja tik sistemai S3 (2srautai, 1aparatas)
 // if ((K==2)&&(IK==4)) Statistika_S2();         //skaiciuoti tik sistemai S2
 // else Statistika_S1();
  Memo1->Lines->SaveToFile(BA);
  Memo2->Lines->SaveToFile(ST);
  */
 }
 /*
 int trukme;
 gettime(&t2);
 trukme=(t2.ti_hund + t2.ti_sec*100 + t2.ti_min*6000 + t2.ti_hour*360000)
        - (t1.ti_hund + t1.ti_sec*100 + t1.ti_min*6000 + t1.ti_hour*360000);
 Memo1->Lines->Add(FloatToStrF(t2.ti_hour,ffFixed,2,0) +"-"+FloatToStrF(t2.ti_min,ffFixed,2,0)+"-
"+FloatToStrF(t2.ti_sec,ffFixed,2,0)+"-"+FloatToStrF(t2.ti_hund,ffFixed,2,0));
 Memo1->Lines->Add(IntToStr(KK)+"- akciju: "+ FloatToStrF(trukme,ffFixed,10,0));
 */
 TBusIntens *D;
 while (TPr)
  {D=TPr; TPr=TPr->kitas; delete D;}
 TBus *DD;
 while (TBPr)
  {DD=TBPr; TBPr=TBPr->kitas; delete DD;}

}
//---------------------------------------------------------------------------
```

Failas Unit1.h

```
//---------------------------------------------------------------------------

#ifndef Unit1H
#define Unit1H
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <Grids.hpp>
#include <Dialogs.hpp>
#include <ComCtrls.hpp>
#include <ExtCtrls.hpp>
#include <Chart.hpp>
#include <Series.hpp>
#include <TeEngine.hpp>
#include <TeeProcs.hpp>
#include "Unit2.h"

const char *BA="Busenu aibe.txt";
const char *ST="Stacionarios tikimybes.txt";

struct st
{
float BInt;
int eil;
int st;
};

struct TBusIntens    //uzkoduota perejimu matrica
{
 st BI;
 TBusIntens *kitas;
};

struct TBus
{
 int *sk;
 TBus *kitas;
};

struct TBusA    //~=TBus tik skirtas realiems skaiciams
{
 double *sk;
 TBusA *kitas;
};

//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:     // IDE-managed Components
        TPageControl *PageControl1;
        TTabSheet *TabSheet1;
        TTabSheet *TabSheet2;
        TChart *Chart1;
        TLineSeries *Series1;
        TLineSeries *Series2;
        TLineSeries *Series3;
        TLabel *Label1;
        TEdit *Edit1;
        TMemo *Memo1;
        TMemo *Memo2;
        TOpenDialog *OpenDialog1;
        TButton *Button1;
        TTabSheet *TabSheet3;
        TStringGrid *StringGrid1;
        TButton *Button2;
        TButton *Button3;
        TCheckBox *CheckBox1;
        TCheckBox *CheckBox2;
        TEdit *Edit2;
        TLabel *Label2;
        TLabel *Label3;
        TLabel *Label4;
        TEdit *Edit3;
        TCheckBox *CheckBox3;
        void __fastcall Button2Click(TObject *Sender);
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall Button3Click(TObject *Sender);
private:

        float *Tintens;
```

```
        TBusIntens *TPr; //perejimu tikimybiu koduotas dinaminis sarasas
        TBus *TBPr;
        float *r;      //normuotos stac tikimybes
        float *r_apj;  //apjungtos stac tikimybes, nes vienai busenai dvi fazes

        int L;        //max eiles ilgis
        int M;        //ivykiu skaicius
        int R;          //srautu skaicius
        int K;       //kasu skaicius

        bool radom;     //ar rasta nauja busena;

        double* Akc;
        double AkcMin;     //min akcijos kainos
        double AkcMax;     //max akcijos kainos
        double IntIlgis;   //intervalo i kuriuos suskaidomos akcijos dydis
        int IntSk;      //intervalu skaicius
        int KK;       //akcijos kainu kiekis
        int IK;         //intensyvumu skaicius
        int N;        //intervalu skaicius

        TBusA *TBAPr;           //nereik?
        //float AkcPok;     //max akcijos kainos?? ==IntIlgis
        TBusA *UPr;        //akcijos kainos didejimo dienu sk sarasas
        TBusA *DPr;        //akcijos kainos mazejimo dienu sk sarasas
        int KKU;         //didejimo masyvo elementu skaicius
        int KKD;         //mazejimo masyvo elementu skaicius     jie siep lygus gaunasi
        double m1U;       //pirmas momentas kainos kilimo
        double m2U;
        double m3U;
        double m1D;       //pirmas momentas kainos kritimo
        double m2D;
        double m3D;

        void NuskaitytiAkcKainas();
        void Apsukti();
        void IsvestiMemoAkc();
        void SuskaidytiIntervalais(double);
        void SurastiEkstremumus();
        void KiekKainu(AnsiString);
        bool ErgodiskumoPatikrinimas();
        void Normavimas();
        void Grafika();
        void SurastiTiketinaKaina();
        void OpcIkainojimas();

        void SurastiIntensyvumus();
        void BusenuRadimasA();        //~=BusenuRadimas tik operuosime realiaisiais skaiciais
        float BusenosA(double*,int);      //~=Busenos tik operuosime realiaisiais skaiciais
        bool PatikrinimasVienatiskumoA(double [], int, double);      //~=PatikrinimasVienatiskumo tik operuosime
realiaisiais skaiciais
        void AukstynZemynCount();     //skaiciuosim kiek dienu kainos didejo/mazejo
        void AukstynZemynCountSizeBased();     //skaiciuosim kiek dienu kainos didejo/mazejo atsizvelgiant i ivesta dydi
        void AukstynZemynStats();     //skaiciuosim statistinius parametrus is AukstynZemynCount sarasu

        void SurastiFaziuIntens();
        void PerejimuTikimybes();
        void IvestiPradSalygas();
        void IsvestiMemo();
        void IsvestiStrGrid();
        void BusenuRadimas();
        float Busenos(int*,int);
        void KasuSk(AnsiString);
        bool PatikrinimasVienatiskumo(int [], int, double);
        void Apjungimas();
        void StacTikRadimas();
        void SurastiMax();
        void Statistika_S6();
        void Statistika_S5();
        void Statistika_S4();
        void Statistika_S3();
        void Statistika_S2();
        void Statistika_S1();
         // User declarations
public:          // User declarations
        __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

Failas Unit2.h

```
//---------------------------------------------------------------------------
#ifndef Unit2H
#define Unit2H
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include "Unit2.h"
#include "Unit1.h"
#include <ExtCtrls.hpp>
//---------------------------------------------------------------------------
class TForm2 : public TForm
{
__published:     // IDE-managed Components
        TButton *Button1;
        TRadioGroup *RadioGroup1;
        TRadioButton *RadioButton2;
        TRadioButton *RadioButton1;
        void __fastcall Button1Click(TObject *Sender);
private:
        void Pozymiai();

public:
         __fastcall TForm2(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm2 *Form2;
//---------------------------------------------------------------------------
#endif
```