VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
TIKIMYBIŲ TEORIJOS IR SKAIČIŲ TEORIJOS KATEDRA

Mindaugas Venckevičius

Rekursinė formulė baigtinio laiko bankroto tikimybei trijų žalų rizikos modelyje

Recurrent formula of finite time ruin probability in three claims risk model

Magistro baigiamasis darbas

Leidžiu ginti _____

Darbo vadovas lekt. dr. A. Grigutis

Vilnius 2016

Turinys

Iliustracijų sąrašas

Lentelių sąrašas

Santrauka

Mindaugo Venckevičiaus magistro darbe nagrinėjamas diskretaus laiko homogeninį rizikos modelis su trimis skirtingai pasiskirsčiusiomis žalomis:

$$U(t) = u + t - \sum_{i=1}^{t} Z_{1i} - \sum_{i=1}^{\lfloor \frac{t}{2} \rfloor} Z_{2i} - \sum_{i=1}^{\lfloor \frac{t}{3} \rfloor} Z_{3i}.$$

Naudojantis pagalbiniais apibrėžimais išvedama ir įrodoma baigtinio laiko bankroto tikimybės rekursinė formulė. Be to, pateikiamas $C\#$ programa parašytas algoritmas bankroto tikimybės apskaičiavimui bei pateikiami bankroto tikimybės pavyzdžiai skirtingiems skirstiniams: Puasono, geometriniam ir trijų (realių draudimo įmonės) žalų skirstiniui.

Raktiniai žodžiai: Homogeninis rizikos atstatymo modelis, Bankroto tikimybė, Atsitiktiniai dydžiai.

Abstract

Mindaugas Venckevičius in his master thesis investigates the discrete time homogeneous risk model with three claims.

$$U(t) = u + t - \sum_{i=1}^{t} Z_{1i} - \sum_{i=1}^{\lfloor \frac{t}{2} \rfloor} Z_{2i} - \sum_{i=1}^{\lfloor \frac{t}{3} \rfloor} Z_{3i}.$$

The recurrent formula for finite time ruin probability is derived and special cases with Poisson and geometrical distributions are analyzed. In addition, the case with a distribution derived from actual insurance claims is also considered. The results are tested by algorithm written in $C\#$ language, which enables to calculate the finite time ruin probability in the analyzed cases.

Keywords: Homogeneous renewal risk model, Ruin probability, Random variables.

1 skyrius

Įvadas

Magistro baigiamajame darbe toliau plėtojama ir gilinamasi į bakalauro darbe pradėtą nagrinėti homogeninį rizikos atstatymo modelį. Klasikinį rizikos modelį pirmieji apibrėžė švedų aktuarai- matematikai F. Lundberg ir H. Cramer (žiūrėti [4] ir [5]). Jų aprašytame modelyje žalų skaičius $\Theta(t)$ baigtiniame laiko intervale $t \in \mathbb{N}$ yra homogeninis Puasono procesas su intensyvumo parametru λ. Nors šis modelis yra paprastas, nes žinoma, kad laiko tarpas tarp žalų yra eksponentinis dydis, bet jį pritaikyti draudimo kompanijoms yra ganėtinai sunku. Todėl buvo nutarta atsisakyti eksponentinio laiko tarpo tarp žalų ir apibrėžti nepriklausomus, vienodai pasiskirsčiusius, neneigiamus atsitiktinio dydžio laiko tarpus tarp žalų. Tokį modelį pirmasis aprašė E. Sparre Andersen (žiūrėti [6]). Iš vienos pusės, atsisakius eksponentinio laiko tarpo tarp žalų buvo supaprastintas modelis ir lengviau pritaikomas praktikoje, iš kitos pusės, matematiškai modelis tapo komplikuotas, nes žalų skaičius tam tikrame laiko momente yra nusakomas atstatymo procesu. Bakalauro darbe buvo gilinamasi į dviejų žalų rizikos modelį, tuo tarpu magistriniame darbe yra nagrinėjama ir išvedama rekursinė baigtinio laiko bankroto tikimybės trijų žalų rizikos formulė. Žemiau pateikiamas rizikos modelio apibrėžimas.

1.0.1 Apibrėžimas. *Sakysime, jog draudiko turtas $U(t)$ kinta pagal diskretaus laiko kelių rizikų modelį*

$$U(t) = u + ct - \sum_{i=1}^{\Theta(t)} Z_i, \tag{1.1}$$

čia:

- *draudiko turtas pradiniu laiko momentu $u = U(0) \in \mathbb{N}_0 = \mathbb{N} \cup \{0\}$,*

• $c > 0$ *pasirašytų įmokų dydis,*

• *žalos* $Z_1, Z_2, ..., Z_K$ *yra nepriklausomi neneigiami sveikareikšmiai atsitiktiniai dydžiai, kurie sveikąsias reikšmes įgyja su lokaliomis tikimybėmis*

$$\mathbb{P}(Z_i = k) = h_{ik}, k \in \mathbb{N}_0, i = 1, 2, ..., K, \tag{1.2}$$

ir pasiskirstymo funkcijomis

$$H_i(x) = \sum_{i=1} h_{ik}, x \in \mathbb{R}, i = 1, 2, ..., K, \tag{1.3}$$

• *žalų sumos* $Z_1, Z_2, ..., Z_K$ *laiko intervale* $[0, t]$ *pasirodo pagal atstymo procesą*

$$\Theta(t) = \sum_{n=1}^{\infty} I\{\Theta_1 + \Theta_2 + ... + \Theta_n \leqslant t\}, \tag{1.4}$$

čia $\Theta_1, \Theta_2, ..., \Theta_n$ *yra nepriklausomos neneigiamos kopijos atsitiktinio dydžio* Θ*, kuris nėra išsigimęs taške, t.y.* $P(\Theta = 0) < 1$*. Taip pat reikia pabrėžti, kad žalos* $Z_1, Z_2, ...$ *ir žalų įvykio laikas* $\Theta_1, \Theta_2, ...$ *yra vienas nuo kito nepriklausomi įvykiai.*

Iš apibrėžimo matome, kad kai modelyje yra keičiami c ar u dydžiai ar atsitiktiniai skirstiniai, gauname visai kitą modelį. Tarkim, jei atstatymo procese Θ yra eksponentinis a.d., tai toks modelis vadinamas klasikiniu rizikos modeliu. Kai parenkame tokias konsantas: $u \in \mathbb{N}_0$, $c = 1$, $\Theta = 1$ ir žalos yra sveikareikšmės, tai modelis vadinamas *diskretaus laiko rizikos modeliu.* Šiuo atveju iš formulės (1.1) gauname:

$$U(t) = u + t - \sum_{i=1}^{t} Z_i, \ t \in \mathbb{N}_0 \tag{1.5}$$

Suprantame, jog draudiko turtas $U(t)$ kinta dėl įvykusių žalų $\{Z_{j1}, Z_{j2}...\}$, $j \in \{1, 2...K\}$, žalų įvykimo laiko $\{\Theta_{j1}, \Theta_{j2}...\}$ kapitalo u ir gaunamų įmokų dydžio c. Jei pasirenkame tokius dydžius: $K = 3$, $c = 1$, $\Theta_{11} = 1$, $\Theta_{21} = 2$, $\Theta_{31} = 3$, tada iš lygties (1.1) gauname trijų žalų bankroto tikimybės formulę:

$$U(t) = u + t - \sum_{i=1}^{t} Z_{1i} - \sum_{i=1}^{\lfloor \frac{t}{2} \rfloor} Z_{2i} - \sum_{i=1}^{\lfloor \frac{t}{3} \rfloor} Z_{3i}, \tag{1.6}$$

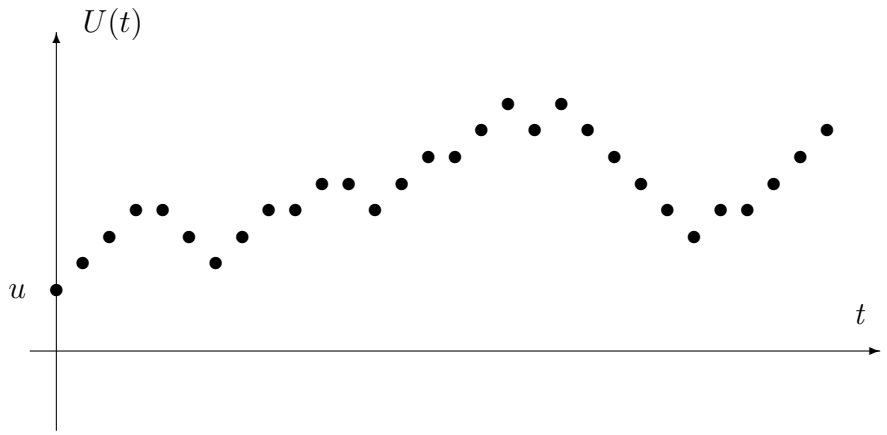
visiems $t \geqslant 0$.

1.0.2 Apibrėžimas. *Laikas, kai draudiko turtas pirmą kartą tapo mažesnis arba lygus nuliui, yra vadinamas bankroto laiku ir žymimas* T_u*:*

$$T_u = \begin{cases} \min\{t \in \mathbb{N} : U(t) \leqslant 0\}, \\ \infty, \ \textit{jeigu} \ \ U(t) > 0 \ \textit{, visiems} \ \ t \in \mathbb{N}. \end{cases}$$

1.0.3 Apibrėžimas. *Baigtinio laiko bankroto tikimybė - tai tikimybė draudikui bankrutuoti iki laiko $T \in \mathbb{N}$, kai pradinis draudiko kapitalas lygus $u \in \mathbb{N}_0$. Šią tikimybę žymėsime $\psi(u,T)$ ir ji lygi*

$$\begin{aligned}\psi(u,T) =& \mathbb{P}(U(t) \leqslant 0 \text{ bent vienas } 1 \leqslant t \leqslant T) \\ =& \mathbb{P}\left(\bigcup_{t=1}^{T}\{U(t) \leqslant 0\}\right).\end{aligned}$$



1.1 pav.: Proceso $U(t)$ vystymosi grafiko pavyzdys

Paveikslėlyje 1.1 pavaizduotas, galimas draudiko turto kitimas laike. Pagal 1.0.1 apibrėžimą, draudiko turtas kiekvienu laiko momentu pasikeičia po 1 piniginį vienetą. Suprantame, kad kol neįvyko žala, tol draudiko turtas $U(t)$ kyla, jei įvyko žala, kuri lygi 1 piniginiui vienetui, tai kapitalas nepasikeičia, jei įvyksta didesnė žala, kapitalas sumažėja. Kai draudiko turtas sumažėja iki $U(t) \leqslant 0$, sakysime, kad įvyksta bankrotas.

Sekančiame skyriuje, pasinaudojus šiame skyriuje suformuotais apibrėžimais, bus aprašomas diskretaus laiko trijų žalų rizikos modelis ir įrodoma rekursinė formulė baigtinio laiko bankroto tikimybei apskaičiuoti. Trečiame skyriuje pasinaudojus $C\#$ programa apskaičiuoti ir pateikti pavyzdžiai pagal Puasono skirstinį, geometrinį skirstinį ir trijų (realių draudimo įmonės) žalų skirstinį.

2 skyrius

Diskretaus laiko trijų žalų rizikos modelis

Šiame skyriuje remsimės A. Grigučio, A. Korvel ir J. Šaulio (žiūrėti [1]) straipsnyje nagrinėjamu diskretaus laiko kelių žalų rizikos modeliu. Šiame straipsnyje teorema yra įrodyta rizikos modeliui su dviem žalomis. Bendru atveju, kai žalų skaičius yra bet koks baigtinis dydis, teorema nėra įrodyta.

2.0.1 Teorema. *Tegu turime atsitikines nepriklausomas žalas* $Z_1, Z_2, ..., Z_K$, $K \geqslant 1$, *o visiems* $u, l \in \mathbb{N}_0$ *tegu*

$$D_{lu}^K = \{k_{ij} \in \mathbb{N}_0 : i \in \{1, 2, ..., K\}, j \in \mathbb{N}, B_{Kl} \leqslant u + l\}, \tag{2.1}$$

čia

$$B_{Kl} = \sum_{i=1}^{K} \sum_{j=1}^{\lfloor \frac{l+1}{i} \rfloor} k_{ij} = \sum_{j=1}^{\lfloor \frac{l+1}{1} \rfloor} k_{1j} + \sum_{j=1}^{\lfloor \frac{l+1}{2} \rfloor} k_{2j} + ... + \sum_{j=1}^{\lfloor \frac{l+1}{K} \rfloor} k_{Kj}. \tag{2.2}$$

Tada visiems $u \in \mathbb{N}_0$

$$\psi(u, 1) = \sum_{k_{11} > u} h_{1k_{11}}, \ \ kai\ T = 1, \tag{2.3}$$

$$\psi(u, 2) = \sum_{k_{11} > u} h_{1k_{11}} + \sum_{\substack{k_{11} \leqslant u \\ k_{11} + k_{12} + k_{21} > u + 1}} h_{1_{k_{11}}} h_{1_{k_{12}}} h_{2_{k_{21}}}, \ \ kai\ T = 2. \tag{2.4}$$

Toliau iki $T \in \{3, 4, ..., M\}$, *kur* M *yra žalų skaičiaus* $1, 2, ..., K$ *mažiausias bendras kartotinis (MBK) formulė tokia:*

$$\psi(u, T) = \psi(u, T - 1) + \sum_{D \subseteq D_{0u}^K \cap D_{1u}^K \cap D_{T-2u}^K ... \cap \overline{D_{T-1u}^K}} \prod_{k_{ij} \in D} h_{ik_{ij}}, \tag{2.5}$$

kai $T \geqslant M+1$, *tai*

$$\psi(u,T) = \psi(u,M) + \sum_{D \subseteq D^K_{0u} \cap D^K_{1u} \ldots \cap D^K_{M-2u} \cap D^K_{M-1u}} \prod_{k_{ij} \in D} h_{ik_{ij}} \psi(u+M-B_{K(M-1)}, T-M). \tag{2.6}$$

Magistro darbe nagrinėjamas atvejis, kai yra trys nepriklausomos žalos, $K = 3$, iš (2.2) formulės gauname žemiau aprašytas nelygybes, pagal kurias vyks žalų sumavimas (2.5) ir (2.6) formulėse. Tegu

$$\begin{aligned}
B^3_{0u} &= \sum_{j=1}^{\lfloor \frac{1}{1} \rfloor} k_{1j} + \sum_{j=1}^{\lfloor \frac{1}{2} \rfloor} k_{2j} + \sum_{j=1}^{\lfloor \frac{1}{3} \rfloor} k_{3j} \leqslant u+0, \\
B^3_{1u} &= \sum_{j=1}^{\lfloor \frac{2}{1} \rfloor} k_{1j} + \sum_{j=1}^{\lfloor \frac{2}{2} \rfloor} k_{2j} + \sum_{j=1}^{\lfloor \frac{2}{3} \rfloor} k_{3j} \leqslant u+1, \\
B^3_{2u} &= \sum_{j=1}^{\lfloor \frac{3}{1} \rfloor} k_{1j} + \sum_{j=1}^{\lfloor \frac{3}{2} \rfloor} k_{2j} + \sum_{j=1}^{\lfloor \frac{3}{3} \rfloor} k_{3j} \leqslant u+2, \\
B^3_{3u} &= \sum_{j=1}^{\lfloor \frac{4}{1} \rfloor} k_{1j} + \sum_{j=1}^{\lfloor \frac{4}{2} \rfloor} k_{2j} + \sum_{j=1}^{\lfloor \frac{4}{3} \rfloor} k_{3j} \leqslant u+3, \\
B^3_{4u} &= \sum_{j=1}^{\lfloor \frac{5}{1} \rfloor} k_{1j} + \sum_{j=1}^{\lfloor \frac{5}{2} \rfloor} k_{2j} + \sum_{j=1}^{\lfloor \frac{5}{3} \rfloor} k_{3j} \leqslant u+4, \\
B^3_{5u} &= \sum_{j=1}^{\lfloor \frac{6}{1} \rfloor} k_{1j} + \sum_{j=1}^{\lfloor \frac{6}{2} \rfloor} k_{2j} + \sum_{j=1}^{\lfloor \frac{6}{3} \rfloor} k_{3j} \leqslant u+5.
\end{aligned}$$

Teoremoje 2.0.1, kai $T = 0$ ir $T = 1$, formulės (2.3) ir (2.4) jau yra išvestos ir jos nekinta didėjant žalų skaičiui, kai $K \geqslant 2$. Nuo $T = 2$ iki M, kai M yra žalų skaičiaus $1, 2, \ldots K$ - MBK, formulėje (2.5) didėja žalų skaičius ir sumavimo aibė. O nuo $T \geqslant M+1$ yra rekurentinė formulė (2.6).

Kaip matome iš teoremos 2.0.1, trijų žalų atveju, kai laikas kinta nuo $M = 3$ iki $M = 6$, tai formulėje (2.5) žalų pasiskirstymo tikimybės sumuojamos šioje aibėje: kai $T = 3$, tai $D^3_{0u} \cap D^3_{1u} \cap \overline{D^3_{2u}}$; kai $T = 4$, tai $D^3_{0u} \cap D^3_{1u} \ldots \cap \overline{D^3_{3u}}$; kai $T = 5$, tai $D^3_{0u} \cap D^3_{1u} \ldots \cap \overline{D^3_{4u}}$; kai $T = 6$, tai $D^3_{0u} \cap D^3_{1u} \ldots \cap \overline{D^3_{5u}}$. Kai laiko momentas yra $T \geqslant 7$, tai formulėje (2.6) sumuojama šioje aibėje: $D^3_{0u} \cap D^3_{1u} \ldots \cap D^3_{5u}$. Kadangi aprašyti po sumavimo simbolio visas nelygybes būtų per daug sudėtinga, darbe paliksime tik aibių sąjungą neišrašant visų sumų.

Toliau darbe įrodoma teoremos 2.0.1 bankroto tikimybės formulė, kai turime tris nepriklausomas vienodai laiko tarpuose pasiskirsčiusias žalas.

2.0.1 teoremos įrodymas. Kai $T = 1$:

$$\begin{aligned}\psi(u,1) &= \mathbb{P}(U(1) \leqslant 0) = \mathbb{P}(u + 1 - Z_{11} \leqslant 0) = \mathbb{P}(Z_{11} \geqslant u + 1) \\ &= \mathbb{P}(Z_{11} > u) = \sum_{k_{11} > u} h_{1k_{11}}.\end{aligned}$$

Kai $T = 2$:

$$\begin{aligned}\psi(u,2) &= \mathbb{P}\{U(1) \leqslant 0\} \cup \{U(2) \leqslant 0\} \\ &= \mathbb{P}\{U(1) \leqslant 0\} + \mathbb{P}(\{U(2) \leqslant 0\} \cap \{U(1) > 0\}) \\ &= \psi(u,1) + \mathbb{P}(\{u + 2 - Z_{11} - Z_{21} - Z_{12} \leqslant 0\} \cap \{u + 1 - Z_{11} > 0\}) \\ &= \psi(u,1) + \mathbb{P}(Z_{11} + Z_{12} + Z_{21} \geqslant u + 2, Z_{11} \leqslant u) \\ &= \psi(u,1) + \sum_{k=0}^{u} \mathbb{P}(Z_{11} + Z_{12} + Z_{21} \geqslant u + 2, Z_{11} = k_{11}) \\ &= \psi(u,1) + \sum_{k=0}^{u} \mathbb{P}(Z_{12} + Z_{21} \geqslant u + 2 - k_{11}) h_{1k_{11}} \\ &= \sum_{k_{11} > u} h_{1k_{11}} + \sum_{\substack{k_{11} \leqslant u \\ k_{11} + k_{12} + k_{21} > u + 1}} h_{1k_{11}} h_{1k_{12}} h_{2k_{21}}.\end{aligned}$$

Kai $T = 3$:

$$\begin{aligned}\psi(u,3) &= \mathbb{P}\left(\bigcup_{t=1}^{3} \{u + 3 - \sum_{i=1}^{3} Z_{1i} - \sum_{i=1}^{\lfloor \frac{3}{2} \rfloor} Z_{2i} - \sum_{i=1}^{\lfloor \frac{3}{3} \rfloor} Z_{3i} \leqslant 0\}\right) \\ &= \sum_{k_{11}=0}^{\infty} \sum_{k_{12}=0}^{\infty} \sum_{k_{13}=0}^{\infty} \sum_{k_{21}=0}^{\infty} \sum_{k_{31}=0}^{\infty} \mathbb{P}\left(\bigcup_{t=1}^{3} \{u + 3 - \sum_{j=1}^{3} \sum_{i=1}^{\lfloor \frac{3}{j} \rfloor} Z_{ji} \leqslant 0\},\right. \\ &\qquad \left. Z_{11} = k_{11}, Z_{12} = k_{12}, Z_{13} = k_{13}, Z_{21} = k_{21}, Z_{31} = k_{31}\right).\end{aligned}$$

Toliau atskirsime pirmo laiko bankroto tikimybės formulę:

$$\begin{aligned}\psi(u,3) = &\sum_{k_{11} > u} \sum_{k_{12}=0}^{\infty} \sum_{k_{13}=0}^{\infty} \sum_{k_{21}=0}^{\infty} \sum_{k_{31}=0}^{\infty} \mathbb{P}\left(\bigcup_{t=1}^{3} \{u + 3 - \sum_{j=1}^{3} \sum_{i=1}^{\lfloor \frac{3}{j} \rfloor} Z_{ji} \leqslant 0\},\right. \\ &\left. Z_{11} = k_{11}, Z_{12} = k_{12}, Z_{13} = k_{13}, Z_{21} = k_{21}, Z_{31} = k_{31}\right) \\ &+ \sum_{k_{11} \leqslant u} \sum_{k_{12}=0}^{\infty} \sum_{k_{13}=0}^{\infty} \sum_{k_{21}=0}^{\infty} \sum_{k_{31}=0}^{\infty} \mathbb{P}\left(\bigcup_{t=1}^{3} \{u + 3 - \sum_{j=1}^{3} \sum_{i=1}^{\lfloor \frac{3}{j} \rfloor} Z_{ji} \leqslant 0\},\right. \\ &\left. Z_{11} = k_{11}, Z_{12} = k_{12}, Z_{13} = k_{13}, Z_{21} = k_{21}, Z_{31} = k_{31}\right)\end{aligned}$$

$$= \overbrace{\mathbb{P}\left(u+1-Z_{11}\leqslant 0\right)}^{\psi(u,1)}$$
$$+\sum_{k_{11}\leqslant u} h_{1k_{11}} \sum_{k_{12}=0}^{\infty}\sum_{k_{13}=0}^{\infty}\sum_{k_{21}=0}^{\infty}\sum_{k_{31}=0}^{\infty} \mathbb{P}\left(\bigcup_{t=2}^{3}\{u+3-\sum_{i=2}^{3} Z_{1i}\sum_{i=1}^{\lfloor\frac{3}{2}\rfloor} Z_{2i}\sum_{i=1}^{\lfloor\frac{3}{3}\rfloor} Z_{3i}\leqslant 0\},\right.$$
$$\left. Z_{12}=k_{12}, Z_{13}=k_{13}, Z_{21}=k_{21}, Z_{31}=k_{31}\right).$$

Dabar galime atskirti antro laiko bankroto tikimybės formulę:

$$\psi(u,3) = \overbrace{\psi(u,1)+\sum_{k_{11}\leqslant u} h_{1k_{11}} \sum_{\substack{k_{12},k_{21}\\ k_{22}+k_{21}+k_{11}>u+1}} h_{1k_{12}}h_{2k_{21}}}^{\psi(u,2)}$$
$$+\sum_{\substack{k_{11}\leqslant u\\ k_{11}+k_{22}+k_{21}\leqslant u+1}} h_{1k_{11}}h_{1k_{12}}h_{2k_{21}} \sum_{\substack{k_{13},k_{31}\\ k_{11}+k_{11}+k_{13}+k_{21}+k_{31}>u+2}} \times$$
$$\times \mathbb{P}\left(u+3-\sum_{i=3}^{3} Z_{1i}-\sum_{i=2}^{\lfloor\frac{3}{2}\rfloor} Z_{2i}-\sum_{i=1}^{\lfloor\frac{3}{3}\rfloor} Z_{3i}\leqslant k_{11}+k_{12}+k_{13}+k_{21}+k_{31}\right)$$
$$= \psi(u,2)+\sum_{\substack{k_{11}\leqslant u\\ k_{11}+k_{22}+k_{21}\leqslant u+1}} h_{1k_{11}}h_{1k_{12}}h_{2k_{21}}\times$$
$$\times \sum_{\substack{k_{13},k_{31}\\ k_{11}+k_{12}+k_{13}+k_{21}+k_{31}>u+2}} \mathbb{P}\left(Z_{13}+Z_{31}\geqslant u+3-k_{11}-k_{12}-k_{13}-k_{21}-k_{31}\right)$$
$$= \psi(u,2)+\sum_{D^3_{0u}\cap D^3_{1u}\cap\overline{D^3_{2u}}} h_{1k_{11}}h_{1k_{12}}h_{1k_{13}}h_{2k_{21}}h_{3k_{31}}.$$

Formulėje (2.5), kai $T=4$, prisideda naujai įvykusios Z_{14} ir Z_{22} žalos, kai $T=5$, prisideda Z_{15} naujai įvykusi žala, kai $T=6$, prisideda naujai įvykusios Z_{16}, Z_{23} ir Z_{32} žalos. Kaip matome, kadangi žalos pasirodo tik vėlyvesniais laiko momentais, tai jau įrodytos bankroto tikimybės formulės bus pateikiamos be įrodymų.

Kai $T=4$:

$$\psi(u,4) = \mathbb{P}\left(\bigcup_{t=1}^{4}\{u+4-\sum_{i=1}^{4} Z_{1i}-\sum_{i=1}^{\lfloor\frac{4}{2}\rfloor} Z_{2i}-\sum_{i=1}^{\lfloor\frac{4}{3}\rfloor} Z_{3i}\leqslant 0\}\right)$$
$$= \psi(u,2)+\sum_{\substack{k_{11}\leqslant u\\ k_{11}+k_{22}+k_{21}\leqslant u+1\\ k_{11}+k_{12}+k_{13}+k_{21}+k_{31}>u+2}} \prod_{k_{ij}\in D} h_{ik_{ij}}$$
$$+\sum_{\substack{k_{11}\leqslant u\\ k_{11}+k_{22}+k_{21}\leqslant u+1\\ k_{11}+k_{12}+k_{13}+k_{21}+k_{31}\leqslant u+2\\ k_{11}+k_{12}+k_{13}+k_{14}+k_{21}+k_{22}+k_{31}>u+3}} \mathbb{P}\left(Z_{14}+Z_{22}\geqslant u+4-\sum_{i=1}^{4} k_{1i}-k_{21}-k_{22}-k_{31}\right)$$
$$= \psi(u,3)+\sum_{D^3_{0u}\cap D^3_{1u}\cap D^3_{2u}\cap\overline{D^3_{3u}}} \prod_{k_{ij}\in D} h_{ik_{ij}}.$$

Kai $T = 5$:

$$\psi(u,5) = \mathbb{P}\left(\bigcup_{t=1}^{5}\{u+5-\sum_{i=1}^{5}Z_{1i}-\sum_{i=1}^{\lfloor\frac{5}{2}\rfloor}Z_{2i}-\sum_{i=1}^{\lfloor\frac{5}{3}\rfloor}Z_{3i}\leqslant 0\}\right)$$
$$= \psi(u,4) + \sum_{D^3_{0u}\cap D^3_{1u}\cap D^3_{2u}\cap D^3_{3u}}\prod_{k_{ij}\in D} h_{ik_{ij}}\times$$
$$\times \sum_{\overline{D^3_{4u}}}\mathbb{P}\left(Z_{15}\geqslant u+5-\sum_{i=1}^{5}k_{1i}-k_{21}-k_{22}-k_{31}\right)$$
$$= \psi(u,4) + \sum_{D^3_{0u}\cap D^3_{1u}\cap D^3_{2u}\cap D^3_{3u}\cap\overline{D^3_{4u}}}\prod_{k_{ij}\in D} h_{ik_{ij}}.$$

Kai $T = 6$:

$$\psi(u,6) = \mathbb{P}\left(\bigcup_{t=1}^{6}\{u+6-\sum_{i=1}^{6}Z_{1i}-\sum_{i=1}^{\lfloor\frac{6}{2}\rfloor}Z_{2i}-\sum_{i=1}^{\lfloor\frac{6}{3}\rfloor}Z_{3i}\leqslant 0\}\right)$$
$$= \psi(u,5) + \sum_{D^3_{0u}\cap D^3_{1u}\cap D^3_{2u}\cap D^3_{3u}\cap D^3_{4u}}\prod_{k_{ij}\in D} h_{ik_{ij}}\times$$
$$\times \sum_{\overline{D^3_{5u}}}\mathbb{P}\left(Z_{16}+Z_{23}+Z_{32}\geqslant u+6-\sum_{i=1}^{6}k_{1i}-\sum_{i=1}^{3}k_{2i}-k_{31}-k_{32}\right)$$
$$= \psi(u,5) + \sum_{D^3_{0u}\cap D^3_{1u}\cap D^3_{2u}\cap D^3_{3u}\cap D^3_{4u}\cap\overline{D_{5u}}}\prod_{k_{ij}\in D} h_{ik_{ij}}.$$

Kai $T \geqslant 7$:

$$\psi(u,T) = \mathbb{P}\left(\bigcup_{t=1}^{T}\{u+t-\sum_{i=1}^{t}Z_{1i}-\sum_{i=1}^{\lfloor\frac{t}{2}\rfloor}Z_{2i}-\sum_{i=1}^{\lfloor\frac{t}{3}\rfloor}Z_{3i}\leqslant 0\}\right)$$
$$= \psi(u,6) + \sum_{D^3_{0u}\cap D^3_{1u}\cap D^3_{2u}\cap D^3_{3u}\cap D^3_{4u}\cap D^3_{5u}}\prod_{k_{ij}\in D} h_{ik_{ij}}\times$$
$$\times \mathbb{P}\left(\bigcup_{t=6}^{T}\{\sum_{i=6}^{t}Z_{1i}+\sum_{i=5}^{\lfloor\frac{t}{2}\rfloor}Z_{2i}+\sum_{i=4}^{\lfloor\frac{t}{3}\rfloor}Z_{3i}\geqslant u+t-\sum_{i=1}^{6}k_{1i}-\sum_{i=1}^{3}k_{2i}-k_{31}-k_{32}\}\right).$$

Kadangi žalos Z_{11}, Z_{12},..., Z_{21},Z_{22},... ir Z_{31}, Z_{32}... yra nepriklausomos ir atsitiktinai pasiskirsčiusios, tai

$$\sum_{i=6}^{t}Z_{1i}\overset{d}{=}\sum_{i=1}^{t-5}Z_{1i},\quad \sum_{i=5}^{t}Z_{2i}\overset{d}{=}\sum_{i=1}^{t-4}Z_{2i},\quad \sum_{i=4}^{t}Z_{3i}\overset{d}{=}\sum_{i=1}^{t-3}Z_{3i},\text{ visiems } t\in\{6,7...T\}$$

Tada paskutinę lygtį galime perrašyti taip:

$$\psi(u,T) = \psi(u,6) + \sum_{D^3_{0u}\cap D^3_{1u}\cap D^3_{2u}\cap D^3_{3u}\cap D^3_{4u}\cap D^3_{5u}}\prod_{k_{ij}\in D} h_{ik_{ij}}\times$$
$$\times \mathbb{P}\left(\bigcup_{\tau=1}^{T-5}\{\sum_{i=1}^{\tau}Z_{1i}+\sum_{i=1}^{\lfloor\frac{\tau}{2}\rfloor}Z_{2i}+\sum_{i=1}^{\lfloor\frac{\tau}{3}\rfloor}Z_{3i}\geqslant u+6+\tau-\sum_{i=1}^{6}k_{1i}-\sum_{i=1}^{3}k_{2i}-k_{31}-k_{32}\}\right)$$

$$= \psi(u,6) + \sum_{D^3_{0u}\cap D^3_{1u}\cap D^3_{2u}\cap D^3_{3u}\cap D^3_{4u}\cap D^3_{5u}} \prod_{k_{ij}\in D} h_{ik_{ij}} \times$$
$$\times \psi(u+6-\sum_{j=1}^{\lfloor \frac{6}{1} \rfloor} k_{1j} - \sum_{j=1}^{\lfloor \frac{6}{2} \rfloor} k_{2j} - \sum_{j=1}^{\lfloor \frac{6}{3} \rfloor} k_{3j}, T-6).$$

Teorema 2.0.1 įrodyta. □

Kaip matome, kuo didesnis žalų skaičius, tuo formulė teoremoje 2.0.1 tampa sudėtingesnė: tiek jos išvedimas, tiek programavimas.

Sekančiame skyriuje pateikiami trys skirtingi pavyzdžiai, parodantys kaip kinta draudiko tikimybė bankrutuoti, jei draudimo įmonės žalų tikimybės pasiskirsčiusios pagal Puasono skirstinį arba pagal geometrinį žalų skirstinį. Taip pat pateikiami vienos Lietuvoje veikiančios draudimo įmonės trijų skirtingų žalų skirstiniai. Draudiko žalų rūšys parinktos tokios, kad formulėje (1.6) atitiktų apibrėžtą žalų pasirodymo laiką.

3 skyrius

Skaičiavimai

3.1 Bankroto tikimybė pagal Puasono žalų skirstinį

1 pavyzdys. Sakykime, baigtiniame diskretaus laiko trijų žalų rizikos modelyje, žalos Z_1, Z_2 ir Z_3 pasiskirsčiusios pagal Puasono skirstinį, kai žalai Z_1 priskiriame parametrą $\lambda = 4/5$, žalai Z_2 priskiriame parametrą $\lambda = 1/8$, o žalai Z_3 - parametrą $\lambda = 1/5$. Tada

$$\mathbb{P}(Z_1 = k) = e^{-\frac{4}{5}} \frac{(\frac{4}{5})^k}{k!}, k = 0, 1, 2...$$
$$\mathbb{P}(Z_2 = k) = e^{-\frac{1}{8}} \frac{(\frac{1}{8})^k}{k!}, k = 0, 1, 2...$$
$$\mathbb{P}(Z_3 = k) = e^{-\frac{1}{5}} \frac{(\frac{1}{5})^k}{k!}, k = 0, 1, 2...$$

Darbe panaudojus Puasono skirstinį gauname retai pasirodančių (pvz., žemės drebėjimai, gaisrai, potvyniai) žalų bankroto tikimybes. Puasono parametrui λ artėjant prie 1, tikimybė patirti žalą didėja, atitinkamai parametrui mažėjant, tikimybė bankrutuoti mažėja. Kaip matome iš 3.1 lentelės, pagal parinktus parametrus, didžiausia tikimybė patirti nuostolių yra pirmaisiais 10 metų, vėliau bankroto tikimybė beveik nesikeičia.

$T \setminus u$	0	1	2	3	4	5	6	7	8	9	10	$\geqslant 20$
1	0, 551	0, 191	0, 047	0, 009	0, 001	0	0	0	0	0	0	0
2	0, 661	0, 312	0, 116	0, 036	0, 010	0, 002	0	0	0	0	0	0
3	0, 748	0, 441	0, 221	0, 097	0, 038	0, 0140	0, 004	0, 001	0	0	0	0
4	0, 780	0, 498	0, 277	0, 138	0, 062	0, 025	0, 009	0, 003	0, 001	0	0, 0001	0
5	0, 796	0, 527	0, 308	0, 162	0, 078	0, 034	0, 014	0, 005	0, 002	0, 001	0, 0002	0
6	0, 810	0, 555	0, 340	0, 190	0, 097	0, 046	0, 020	0, 008	0, 003	0, 001	0, 0004	0
7	0, 810	0, 555	0, 340	0, 190	0, 097	0, 046	0, 020	0, 008	0, 003	0, 001	0, 0004	0
8	0, 811	0, 556	0, 340	0, 190	0, 097	0, 046	0, 020	0, 008	0, 003	0, 001	0, 0004	0
9	0, 813	0, 558	0, 341	0, 190	0, 097	0, 046	0, 020	0, 008	0, 003	0, 001	0, 0004	0
10	0, 815	0, 56	0, 343	0, 191	0, 098	0, 046	0, 020	0, 008	0, 003	0, 001	0, 0004	0
20	0, 818	0, 564	0, 346	0, 193	0, 099	0, 047	0, 020	0, 008	0, 003	0, 001	0, 0004	0
30	0, 818	0, 564	0, 346	0, 193	0, 099	0, 047	0, 020	0, 008	0, 003	0, 001	0, 0004	0
40	0, 818	0, 564	0, 346	0, 193	0, 099	0, 047	0, 020	0, 008	0, 003	0, 001	0, 0004	0
50	0, 818	0, 564	0, 346	0, 193	0, 099	0, 047	0, 020	0, 008	0, 003	0, 001	0, 0004	0
100	0, 818	0, 564	0, 346	0, 193	0, 099	0, 047	0, 020	0, 008	0, 003	0, 001	0, 0004	0
150	0, 818	0, 564	0, 346	0, 193	0, 099	0, 047	0, 020	0, 008	0, 003	0, 001	0, 0004	0

3.1 lentelė: Bankroto tikimybė pagal Puasono žalų skirstinius.

3.2 Bankroto tikimybė pagal geometrinį žalų skirstinį

2 pavyzdys. Sakykime, baigtiniame diskretaus laiko trijų žalų rizikos modelyje žalos Z_1, Z_2 ir Z_3 pasiskirsčiusios pagal geometrinį dėsnį. Žalai Z_1 priskiriame parametrą $p = 1/10$, žalai Z_2 parametrą $p = 1/2$, o žalai Z_3 priskiriame parametrą $p = 1/8$, tai

$$\mathbb{P}(Z_1 = k) = \frac{1}{10}(1 - \frac{1}{10})^k, k = 0, 1, 2...$$
$$\mathbb{P}(Z_2 = k) = \frac{1}{2}(1 - \frac{1}{2})^k, k = 0, 1, 2...$$
$$\mathbb{P}(Z_3 = k) = \frac{1}{8}(1 - \frac{1}{8})^k, k = 0, 1, 2...$$

Darbe panaudojus geometrinį skirstinį matome, kaip ir Puasono skirstinio atveju, tikimybė bankrutuoti nuo 40-tų metų išlieka tokia pati. Paskutiniame pavyzdyje yra pavaizduotas draudimo įmonės žalų skirstinys.

$T \setminus u$	0	1	2	3	4	5	6	7	8	9	10	20	$\geqslant 30$
1	0,100	0,010	0,001	0	0	0	0	0	0	0	0	0	0
2	0,271	0,096	0,037	0,015	0,006	0,002	0,001	0	0	0	0	0	0
3	0,479	0,281	0,170	0,103	0,062	0,038	0,023	0,014	0,008	0,005	0,003	0	0
4	0,526	0,328	0,208	0,130	0,081	0,050	0,030	0,018	0,011	0,007	0,004	0	0
5	0,527	0,329	0,208	0,131	0,081	0,050	0,030	0,018	0,011	0,007	0,004	0	0
6	0,622	0,441	0,313	0,220	0,151	0,103	0,069	0,046	0,031	0,020	0,013	0,0001	0
7	0,622	0,441	0,314	0,220	0,151	0,103	0,069	0,046	0,031	0,020	0,013	0,0001	0
8	0,625	0,443	0,315	0,221	0,152	0,104	0,070	0,046	0,031	0,020	0,013	0,0001	0
9	0,644	0,463	0,332	0,234	0,162	0,111	0,075	0,050	0,033	0,022	0,014	0,0001	0
10	0,650	0,469	0,338	0,238	0,166	0,113	0,077	0,051	0,034	0,022	0,015	0,0001	0
20	0,684	0,510	0,376	0,272	0,193	0,135	0,093	0,064	0,043	0,029	0,019	0,0002	0
30	0,688	0,515	0,381	0,277	0,197	0,138	0,096	0,066	0,045	0,030	0,020	0,0002	0
40	0,689	0,515	0,382	0,277	0,198	0,139	0,096	0,066	0,045	0,030	0,020	0,0002	0
50	0,689	0,515	0,382	0,277	0,198	0,139	0,096	0,066	0,045	0,030	0,020	0,0002	0
60	0,689	0,515	0,382	0,277	0,198	0,139	0,096	0,066	0,045	0,030	0,020	0,0002	0
70	0,689	0,515	0,382	0,277	0,198	0,139	0,096	0,066	0,045	0,030	0,020	0,0002	0
80	0,689	0,515	0,382	0,277	0,198	0,139	0,096	0,066	0,045	0,030	0,020	0,0002	0
90	0,689	0,515	0,382	0,277	0,198	0,139	0,096	0,066	0,045	0,030	0,020	0,0002	0
100	0,689	0,515	0,382	0,277	0,198	0,139	0,096	0,066	0,045	0,030	0,020	0,0002	0
150	0,689	0,515	0,382	0,277	0,198	0,139	0,096	0,066	0,045	0,030	0,020	0,0002	0
200	0,689	0,515	0,382	0,277	0,198	0,139	0,096	0,066	0,045	0,030	0,020	0,0002	0

3.2 lentelė: Bankroto tikimybė pagal geometrinius žalų skirstinius.

3.3 Bankroto tikimybė pagal draudimo įmonės žalų skirstinius

3 pavyzdys. Šiame pavyzdyje pateikiamas Lietuvoje veikiančios gyvybės draudimo įmonės vieno iš produkto žalų Z_1, Z_2 ir Z_3 skirstiniai. Žalos yra pasiskirsčiusios pagal tokį skirstinį:

Z_1	0	1	2	3
$\mathbb{P}$	0,70	0,17	0,09	0,04

Z_2	0	1	2
$\mathbb{P}$	0,18	0,42	0,40

Z_3	0	1	2	3	4	5
$\mathbb{P}$	0,30	0,25	0,15	0,12	0,10	0,08

3.3 lentelė: Draudimo įmonės žalų Z_1, Z_2 ir Z_3 skirstiniai.

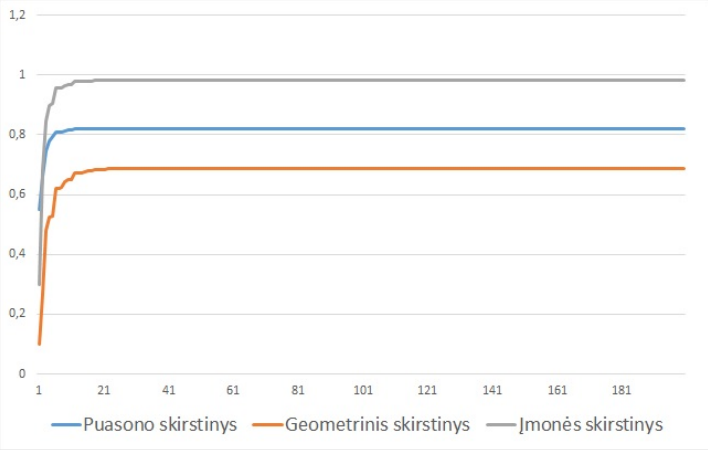
Kadangi žalų vidurkių suma didesnė už vienetą, t.y. $EZ_1 + EZ_2/2 + EZ_3/3 = 1,68 > 1$ ir netenkina grynojo pelno sąlygos $EZ_1 + EZ_2/2 + EZ_3/3 < 1$, tai draudimo įmonė, pagal žalų lentelės 3.3 duomenis, patiria nuostolius, kai turtas auga po 1 piniginį vienetą. Reikia paminėti, kad šiame pavyzdyje nėra įvertinamos draudimo įmonės premijos (pasirašytos įmokos) dydis. Kadangi pavyzdys paruoštas su trim skirtingomis žalomis ir jų rizikos kaina taip pat yra skirtinga, tai c dydis formulėje (1.6) turėtų būti didesnis už vienetą. Draudimo kompanijoms norint sužinot tiks-

$T \setminus u$	0	1	2	3	4	5	6	7	8	9	10	$\geqslant 20$
1	0, 300	0, 130	0, 040	0	0	0	0	0	0	0	0	0
2	0, 685	0, 351	0, 169	0, 061	0, 016	0, 004	0, 001	0	0	0	0	0
3	0, 847	0, 645	0, 464	0, 303	0, 179	0, 094	0, 045	0, 018	0, 007	0, 002	0, 001	0
4	0, 897	0, 745	0, 582	0, 422	0, 283	0, 173	0, 097	0, 049	0, 023	0, 009	0, 003	0
5	0, 905	0, 762	0, 604	0, 445	0, 304	0, 191	0, 110	0, 058	0, 028	0, 012	0, 005	0
6	0, 957	0, 884	0, 787	0, 671	0, 546	0, 422	0, 310	0, 216	0, 143	0, 089	0, 053	0
7	0, 957	0, 884	0, 787	0, 671	0, 546	0, 422	0, 310	0, 216	0, 143	0, 089	0, 053	0
8	0, 958	0, 885	0, 788	0, 672	0, 546	0, 423	0, 311	0, 216	0, 143	0, 089	0, 053	0
9	0, 965	0, 897	0, 802	0, 685	0, 557	0, 431	0, 316	0, 219	0, 144	0, 090	0, 053	0
10	0, 969	0, 906	0, 814	0, 698	0, 570	0, 441	0, 324	0, 225	0, 148	0, 092	0, 054	0
20	0, 983	0, 942	0, 873	0, 775	0, 657	0, 528	0, 402	0, 290	0, 197	0, 126	0, 076	0
30	0, 984	0, 945	0, 878	0, 783	0, 665	0, 537	0, 410	0, 297	0, 203	0, 131	0, 079	0
40	0, 985	0, 946	0, 879	0, 783	0, 666	0, 538	0, 411	0, 297	0, 203	0, 131	0, 080	0
50	0, 985	0, 946	0, 879	0, 784	0, 666	0, 538	0, 412	0, 298	0, 203	0, 131	0, 080	0
100	0, 985	0, 946	0, 879	0, 784	0, 666	0, 538	0, 412	0, 298	0, 203	0, 131	0, 080	0
150	0, 985	0, 946	0, 879	0, 784	0, 666	0, 538	0, 412	0, 298	0, 203	0, 131	0, 080	0
200	0, 985	0, 946	0, 879	0, 784	0, 666	0, 538	0, 412	0, 298	0, 203	0, 131	0, 080	0

3.4 lentelė: Bankroto tikimybė pagal draudimo įmonės žalų skirstinius.

lų savo produktų nuostolingumą reikėtų atsižvelgti ne tik į žalų skirstinius, bet ir įmokos dydį c ar pradinį kapitalą u. Bet kaip jau buvo minėta įvade, toks modelis taptų visai kitoks nei šiame darbe nagrinėjamas.

Kaip matome iš lentelės 3.4 jau trečiais metais, kai turtas yra lygus 0 ar 1, nuostolingumas (bankrotas) yra didesnis nei 0,5 proc. Draudimo įmonėms palankiausia, kai grynasis nuostolingumas yra apie 0,5 proc., o nuostolingumas su sąnaudomis neviršytų 0,85 proc. Taigi įmonei su nurodytu žalų skirstiniu norint išlaikyti žemesnį nuostolingumą veiklos pradžioje reikia turėti didesnį kapitalą arba gauti didesnes įmokas. Žemiau pateiktas skaičiavimuose naudotų skirstinių bankroto tikimybės pasiskirstymas, kai kapitalas u lygus nuliui, o laikas kinta nuo $t = 1$ iki $t = 200$ grafikas.



3.1 pav.: Bankroto tikimybės pasiskirstymo funkcijų grafikas

4 skyrius

Išvados

Magistro darbe aprašytas atstatymo rizikos modelis ir gauta trijų žalų bankroto tikimybės formulė (1.6). Pasinaudojus teorema įrodytos $T = 3$, $T = 4$, $T = 5$ ir $T = 6$ laiko bankroto tikimybei apskaičiuoti formulės, taip pat nuo $T \geqslant 7$ momento įrodyta rekursinė bankroto tikimybės formulė. Išvestos formulės įrodytos 2 skyriuje. Su $C\#$ programa parašytas algoritmas skaičiuoti bankroto tikimybę $\psi(u, T)$. Programa galima skaičiuoti tikimybes bet kokiems u ir T dydžiams, kai turime tris nepriklausomas žalas, pasiskirsčiusias pagal Puasono skirstinį, geometrinio dėsnio skirstinį ar kai turime trijų žalų pasirodymo tikimybes. Draudimo įmonių praktikoje išvestą formulę galima skaičiuoti produktų pelningumus (bankroto tikimybes) įvedus savo esamas ar prognozuojamas žalų tikimybes.

Ateityje šis darbas galėtų būti plėtojamas įvairiomis kryptimis:

- Diskretaus laiko trijų žalų rizikų modelis begaliniame laike, kai žalos laiko tarpuose yra nepriklausomos ir pasiskirsčiusios vienodai.
- Diskretaus laiko didesnio baigtinio žalų skaičiaus rizikų modelis baigtiniame ir/ar begaliniame laike, kai žalos laiko tarpuose yra nepriklausomos ir pasiskirsčiusios vienodai.
- Kelių žalų rizikų modelis baigtiniame ir/ar begaliniame laike, kai žalos ir laikais yra nebūtinai vienodai pasiskirstę ir vienas nuo kito priklausomi.

Literatūra

[1] A. Grigutis, A. Korvel, J. Šiaulys *Ruin probabilities of the discrete time multi-risk model, ISSN 2335-884X.*

[2] E. Bieliauskienė, *disertacinis darbas: http* : $//vddb.laba.lt/fedora/get/LT-eLABa-0001 : E.02\ 2012\ D_20120629_152603-71070/DS.005.0.01.ETD$.

[3] Chen, Y., Yuen, K., and Ng, K. *(2011). Assymptotics for the ruin probabilities of a two dimensional renewal risk model with heavy-tailed claims. Applied Stochastic Models in Bussines and Industry, 27(3):290–300..*

[4] F. Lundberg, *Approximerad framställning av annolikhetsfunktionen. Å terförsäkring av kollektivrisker. Almqvist och Wiksell, Uppsala, 1903.*

[5] H. Cramer, *On the mathematical theory of risk, Skandia Jubilee volume, Stockholm, 1930.*

[6] E. Sparre Andersen, *On the cpllective theory of risk in the case of contagion between the claims, Transactions XVth International Congress of Actuaries, 2(6):219-229, 1957.*

[7] J. Šiaulys, *paskaitų konspektas: http* : $//mif.vu.lt/lt2/mak/darbuotojai/jonas-siaulys/failai/JS_Pirmas_skyrius.pdf$.

[8] De Vylder, F. E. and Goovaerts, M. J. *(1988). Recursive calculation of finite-time ruin probabilities. Insurance: Mathematics and Economics, 7:1–8..*

5 skyrius

Priedas: Programa C#

```
void Main()
{
const int laikas = 150;
const int turtas = 100;
var matrica = suskaiciuoti_laiko_matrica(laikas, turtas);
foreach(double[] eilute in matrica)
{
Console.WriteLine(string.Join(";", eilute));
}}
// LAIKAS
static double[][] suskaiciuoti_laiko_matrica(int laikas, int turtas)
{
double[][] matrica = new double[laikas][];
  for (int laiko_indeksas = 0; laiko_indeksas < laikas; laiko_indeksas++)
{
matrica[laiko_indeksas] = new double[turtas + 1];
for (int turto_indeksas = 0; turto_indeksas <= turtas; turto_indeksas++)
{
if (laiko_indeksas == 0)//pirmas laiko momentas
{
matrica[laiko_indeksas][turto_indeksas] = zalos_suma_formule1(turto_indeksas);
}
```

```
if (laiko_indeksas == 1)//antras laiko momentas
{
matrica[laiko_indeksas][turto_indeksas] =
zalos_suma_formule2(turto_indeksas)
+ matrica[0][turto_indeksas];
}
if (laiko_indeksas ==2)//trecias laiko momentas
{
matrica[laiko_indeksas][turto_indeksas] =
zalos_suma_formule3(turto_indeksas)
+ matrica[1][turto_indeksas];
}
if (laiko_indeksas ==3)//ketvirtas laiko momentas
{
matrica[laiko_indeksas][turto_indeksas] =
zalos_suma_formule4(turto_indeksas)
+ matrica[2][turto_indeksas];
}
if (laiko_indeksas ==4)//penktas laiko momentas
{
matrica[laiko_indeksas][turto_indeksas] =
zalos_suma_formule5(turto_indeksas)
 + matrica[3][turto_indeksas];
}
if (laiko_indeksas ==5)//sestas laiko momentas
{
matrica[laiko_indeksas][turto_indeksas] =
zalos_suma_formule6(turto_indeksas)
+ matrica[4][turto_indeksas];
}
if (laiko_indeksas > 5)//septintas laiko momentas
{
matrica[laiko_indeksas][turto_indeksas] =
zalos_suma_formule7(turto_indeksas, matrica[laiko_indeksas - 6])
```

```
+ matrica[5][turto_indeksas];
}}}
return matrica;
}
//FORMULE, kai t=1
static double[] atvejis_2_formule1(int maxTurtas)
{
double[] rezultatas = new double [maxTurtas + 1];

for (int indeksas = 0; indeksas <= maxTurtas; indeksas++)
{
rezultatas[indeksas] = zalos_suma_formule1(indeksas);
}
return rezultatas;
}
static double zalos_suma_formule1(int turtas)
{
return zalos_suma_rekursija_formule1(0, turtas);
}
static double zalos_suma_rekursija_formule1(int zalos_dydis_indeksas, int turtas)
{
if (zalos_dydis_indeksas > turtas)
{
return 0;
}
double zalos_dydis_reiksme;
if (zalos_dydis_indeksas < zalos_dydis.Length)
{
zalos_dydis_reiksme = zalos_dydis[zalos_dydis_indeksas];
}
else
{
zalos_dydis_reiksme = 0;
}
```

```
double sumos_demuo =  tikimybe_laiko_momentu_formule1(zalos_dydis_indeksas,
 turtas);
return sumos_demuo;
}
static double tikimybe_laiko_momentu_formule1(int zalos_dydis_indeksas,
 int turtas)
{
double suma = 0;
int pradzios_indeksas = turtas + 1 - zalos_dydis_indeksas;
for(int pradzia = pradzios_indeksas; pradzia < zalos_dydis.Length; pradzia++)
{
suma = suma + zalos_dydis[pradzia];
}
return suma;
}
//FORMULE; kai t=2
static double[] atvejis_2_formule2(int maxTurtas)
{
double[] rezultatas = new double[maxTurtas + 1];
for (int indeksas = 0; indeksas <= maxTurtas; indeksas++)
{
rezultatas[indeksas] = zalos_suma_formule2(indeksas);
}
return rezultatas;
}
static double zalos_suma_formule2(int turtas)
{
return zalos_suma_rekursija_formule2(0, turtas);
}
static double zalos_suma_rekursija_formule2(int zalos_dydis_indeksas,
 int turtas)
{
if (zalos_dydis_indeksas > turtas)
{
```

```
return 0;
}
double zalos_dydis_reiksme;
if (zalos_dydis_indeksas < zalos_dydis.Length)
{
zalos_dydis_reiksme = zalos_dydis[zalos_dydis_indeksas];
}
else
{
zalos_dydis_reiksme = 0;
}
double sumos_demuo =  zalos_dydis_reiksme
* tikimybe_laiko_momentu_formule2(zalos_dydis_indeksas, turtas);
return sumos_demuo
+ zalos_suma_rekursija_formule2(zalos_dydis_indeksas + 1, turtas);
}
static double tikimybe_laiko_momentu_formule2(int zalos_dydis_indeksas,
 int turtas)
{
double suma = 0;
const int perslenkam_nes_daugiau = 1;
int pradzios_indeksas = turtas + 1
- zalos_dydis_indeksas + perslenkam_nes_daugiau;
for(int pradzia = pradzios_indeksas; pradzia
< zalos_dydis_antru_laiko_momentu.Length; pradzia++)
{
suma = suma + zalos_dydis_antru_laiko_momentu[pradzia];
}
return suma;
}
//FORMULE, kai t=3
static double[] atvejis_2_formule3(int maxTurtas)
{
double[] rezultatas = new double[maxTurtas + 1];
```

```
for (int indeksas = 0; indeksas <= maxTurtas; indeksas++)
{
rezultatas[indeksas] = zalos_suma_formule3(indeksas);
}
return rezultatas;
}
static double zalos_suma_formule3(int turtas)
{
return zalos_suma_rekursija_formule3(0, turtas);
}
static double zalos_suma_rekursija_formule3(int zalos_dydis_indeksas,
 int turtas)
{
if (zalos_dydis_indeksas > turtas)
{
return 0;
}
double zalos_dydis_reiksme;
if (zalos_dydis_indeksas < zalos_dydis.Length)
{
zalos_dydis_reiksme = zalos_dydis[zalos_dydis_indeksas];
}
else
{
zalos_dydis_reiksme = 0;
}
double sumos_demuo =
  tikimybe_laiko_momentu_formule33(zalos_dydis_indeksas, turtas);

return sumos_demuo;
}
static double tikimybe_laiko_momentu_formule33(int zalos_dydis_indeksas,
 int turtas)
{
```

```
double suma = 0;
const int perslenkam_nes_daugiau = 1;
int zalos_dydis_indeksas2 = zalos_dydis_antru_laiko_momentu.Length;

int pradzios_indeksas = turtas + 2  + perslenkam_nes_daugiau;
int pabaigos_indeksas = turtas + 1  + perslenkam_nes_daugiau;
int pabaigos_indeksas1 = turtas + 0  + perslenkam_nes_daugiau;

for(int pradzia2 = 0; pradzia2 < Math.Min(pabaigos_indeksas1,zalos_dydis.Length);
pradzia2++)
{
for(int pradzia1 = 0; pradzia1+pradzia2
< Math.Min(pabaigos_indeksas,zalos_dydis_antru_laiko_momentu.Length); pradzia1++)
{
for(int pradzia = pradzios_indeksas-pradzia1-pradzia2;
pradzia <zalos_dydis_treciu_laiko_momentu.Length; pradzia++)
{
suma = suma + zalos_dydis[pradzia2] * zalos_dydis_antru_laiko_momentu[pradzia1]
* zalos_dydis_treciu_laiko_momentu[pradzia];
}}}
return suma;
}
//FORMULE, kai t=4
static double[] atvejis_2_formule4(int maxTurtas)
{
double[] rezultatas = new double[maxTurtas + 1];

for (int indeksas = 0; indeksas <= maxTurtas; indeksas++)
{
rezultatas[indeksas] = zalos_suma_formule4(indeksas);
}
return rezultatas;
}
static double zalos_suma_formule4(int turtas)
```

```
{
return zalos_suma_rekursija_formule4(0, turtas);
}
static double zalos_suma_rekursija_formule4(int zalos_dydis_indeksas,
 int turtas)
{
if (zalos_dydis_indeksas > turtas)
{
return 0;
}
double zalos_dydis_reiksme;
if (zalos_dydis_indeksas < zalos_dydis.Length)
{
zalos_dydis_reiksme = zalos_dydis[zalos_dydis_indeksas];
}
else
{
zalos_dydis_reiksme = 0;
}
double sumos_demuo =
tikimybe_laiko_momentu_formule41(zalos_dydis_indeksas, turtas);
return sumos_demuo;
}
static double tikimybe_laiko_momentu_formule41(int zalos_dydis_indeksas,
 int turtas)
{
double suma = 0;
const int perslenkam_nes_daugiau = 1;
int pradzios_indeksas = turtas + 3 + perslenkam_nes_daugiau;
int pabaigos_indeksas2 = turtas + 2  + perslenkam_nes_daugiau;
int pabaigos_indeksas1 = turtas + 1 + perslenkam_nes_daugiau;
int pabaigos_indeksas = turtas + 0  + perslenkam_nes_daugiau;
for(int pradzia2 = 0; pradzia2
< Math.Min(pabaigos_indeksas,zalos_dydis.Length); pradzia2++)
```

```
{
for(int pradzia4 = 0; pradzia4+pradzia2
< Math.Min(pabaigos_indeksas1, zalos_dydis_antru_laiko_momentu.Length); pradzia4++)
{
for(int pradzia3 = 0; pradzia3+pradzia4+pradzia2 < Math.Min(pabaigos_indeksas2,zalo
{
for(int pradzia = Math.Max(0,pradzios_indeksas-pradzia2-pradzia4-pradzia3);
pradzia < zalos_dydis_antru_laiko_momentu.Length; pradzia++)
{
suma = suma + zalos_dydis[pradzia2] * zalos_dydis_treciu_laiko_momentu[pradzia3]
* zalos_dydis_antru_laiko_momentu[pradzia4]
* zalos_dydis_antru_laiko_momentu[pradzia];
}}}}
return suma;
}
static double tikimybe_laiko_momentu_formule41x(int zalos_dydis_indeksas,
 int turtas)
{
double suma = 0;
const int perslenkam_nes_daugiau = 1;
int pradzios_indeksas = turtas + 3 + perslenkam_nes_daugiau;
int pabaigos_indeksas2 = turtas + 2 + perslenkam_nes_daugiau;
int pabaigos_indeksas1 = turtas + 1 + perslenkam_nes_daugiau;
for(int pradzia1 = 0; pradzia1
< Math.Min(pabaigos_indeksas1, zalos_dydis_antru_laiko_momentu.Length);
pradzia1++)
{
for(int pradzia2 = 0; pradzia2
< Math.Min(pabaigos_indeksas2, zalos_dydis_treciu_laiko_momentu.Length);
pradzia2++)
{
for(int pradzia = pradzios_indeksas; pradzia
< zalos_dydis_antru_laiko_momentu.Length; pradzia++)
if (pradzios_indeksas < turtas+3)
```

```
{
return 0;
}
else
suma = suma + zalos_dydis_antru_laiko_momentu[pradzia1] *zalos_dydis_treciu_laiko_m
*zalos_dydis_antru_laiko_momentu[pradzia];
}}
return suma;
}

//FORMULE, kai t=5
static double[] atvejis_2_formule5(int maxTurtas)
{
double[] rezultatas = new double[maxTurtas + 1];

for (int indeksas = 0; indeksas <= maxTurtas; indeksas++)
{
rezultatas[indeksas] = zalos_suma_formule4(indeksas);
}
return rezultatas;
}
static double zalos_suma_formule5(int turtas)
{
return zalos_suma_rekursija_formule5(0, turtas);
}
static double zalos_suma_rekursija_formule5(int zalos_dydis_indeksas,
 int turtas)
{
if (zalos_dydis_indeksas > turtas)
{
return 0;
}
double zalos_dydis_reiksme;
if (zalos_dydis_indeksas < zalos_dydis.Length)
```

```
{
zalos_dydis_reiksme = zalos_dydis[zalos_dydis_indeksas];
}
else
{
zalos_dydis_reiksme = 0;
}
double sumos_demuo =
 tikimybe_laiko_momentu_formule51(zalos_dydis_indeksas, turtas);
return sumos_demuo;
}
static double tikimybe_laiko_momentu_formule51(int zalos_dydis_indeksas,
 int turtas)
{
double suma = 0;
const int perslenkam_nes_daugiau = 1;
int pradzios_indeksas = turtas + 4  + perslenkam_nes_daugiau;
int pabaigos_indeksas3 = turtas + 3  + perslenkam_nes_daugiau;
int pabaigos_indeksas2 = turtas + 2  + perslenkam_nes_daugiau;
int pabaigos_indeksas1 = turtas + 1  + perslenkam_nes_daugiau;
int pabaigos_indeksas0 = turtas + 0  + perslenkam_nes_daugiau;
for(int pradzia0 = 0; pradzia0 < Math.Min(pabaigos_indeksas0,
zalos_dydis.Length);
 pradzia0++)
{
for(int pradzia1 = 0; pradzia1+pradzia0
< Math.Min(pabaigos_indeksas1, zalos_dydis_antru_laiko_momentu.Length);
 pradzia1++)
{
for(int pradzia2 = 0; pradzia2+pradzia0+pradzia1
< Math.Min(pabaigos_indeksas2, zalos_dydis_treciu_laiko_momentu.Length);
pradzia2++)
{
for(int pradzia3 = 0; pradzia3+pradzia2+pradzia1+pradzia0
```

```
< Math.Min(pabaigos_indeksas3, zalos_dydis_antru_laiko_momentu.Length);
 pradzia3++)
{
for(int pradzia =
Math.Max(0, pradzios_indeksas-pradzia3-pradzia2-pradzia1-pradzia0);
pradzia < zalos_dydis.Length; pradzia++)
{
suma = suma + zalos_dydis[pradzia0]
* zalos_dydis_antru_laiko_momentu[pradzia1]
*zalos_dydis_treciu_laiko_momentu[pradzia2]
*zalos_dydis_antru_laiko_momentu[pradzia3]*zalos_dydis[pradzia];
}}}}}
return suma;
}
//FORMULE, kai t=6
static double[] atvejis_2_formule6(int maxTurtas)
{
double[] rezultatas = new double[maxTurtas + 1];
for (int indeksas = 0; indeksas <= maxTurtas; indeksas++)
{
rezultatas[indeksas] = zalos_suma_formule6(indeksas);
}
return rezultatas;
}
static double zalos_suma_formule6(int turtas)
{
return zalos_suma_rekursija_formule6(0, turtas);
}
static double zalos_suma_rekursija_formule6(int zalos_dydis_indeksas,
 int turtas)
{
if (zalos_dydis_indeksas > turtas)
{
return 0;
```

```
}
double zalos_dydis_reiksme;
if (zalos_dydis_indeksas < zalos_dydis.Length)
{
zalos_dydis_reiksme = zalos_dydis[zalos_dydis_indeksas];
}
else
{
zalos_dydis_reiksme = 0;
}
double sumos_demuo =  tikimybe_laiko_momentu_formule61(zalos_dydis_indeksas,
 turtas);
return sumos_demuo;
}
static double tikimybe_laiko_momentu_formule61(int zalos_dydis_indeksas,
 int turtas)
{
double suma = 0;
const int perslenkam_nes_daugiau = 1;
int pradzios_indeksas = turtas + 5  + perslenkam_nes_daugiau;
int pabaigos_indeksas4 = turtas + 4  + perslenkam_nes_daugiau;
int pabaigos_indeksas3 = turtas + 3  + perslenkam_nes_daugiau;
int pabaigos_indeksas2 = turtas + 2 + perslenkam_nes_daugiau;
int pabaigos_indeksas1 = turtas + 1  + perslenkam_nes_daugiau;
int pabaigos_indeksas0 = turtas + 0  + perslenkam_nes_daugiau;
for(int pradzia0 = 0; pradzia0
< Math.Min(pabaigos_indeksas0, zalos_dydis.Length); pradzia0++)
{
for(int pradzia1 = 0; pradzia1 + pradzia0
< Math.Min(pabaigos_indeksas1, zalos_dydis_antru_laiko_momentu.Length);
pradzia1++)
{
for(int pradzia2 = 0; pradzia2 + pradzia1 + pradzia0
< Math.Min(pabaigos_indeksas2, zalos_dydis_treciu_laiko_momentu.Length);
```

```
pradzia2++)
{
for(int pradzia3 = 0; pradzia3 + pradzia2 + pradzia1 + pradzia0
< Math.Min(pabaigos_indeksas3, zalos_dydis_antru_laiko_momentu.Length);
pradzia3++)
{
for(int pradzia4 = 0; pradzia4 + pradzia3 + pradzia2 + pradzia1 + pradzia0
< Math.Min(pabaigos_indeksas4, zalos_dydis.Length); pradzia4++)
{
for(int pradzia =
Math.Max(0,pradzios_indeksas-pradzia4-pradzia3-pradzia2-pradzia1-pradzia0);
 pradzia < zalos_dydis_sestu_laiko_momentu.Length; pradzia++)
{
suma = suma + zalos_dydis[pradzia0]
 * zalos_dydis_antru_laiko_momentu[pradzia1]
* zalos_dydis_treciu_laiko_momentu[pradzia2]
*  zalos_dydis_sestu_laiko_momentu[pradzia]
*zalos_dydis_antru_laiko_momentu[pradzia3]* zalos_dydis[pradzia4];
}}}}}}
return suma;
}
//FORMULE, kai t>=7
static double[] atvejis_2_formule7(int maxTurtas)
{
double[] rezultatas = new double[maxTurtas + 1];

for (int indeksas = 0; indeksas <= maxTurtas; indeksas++)
{
rezultatas[indeksas] = zalos_suma_formule7(indeksas, new double[0]);
}
return rezultatas;
}
static double zalos_suma_formule7(int turtas, double[] eiluteT)
{
```

```
return zalos_suma_rekursija_formule7(0, turtas, eiluteT);
}
static double zalos_suma_rekursija_formule7(int zalos_dydis_indeksas, int turtas, d
{
if (zalos_dydis_indeksas > turtas)
{
return 0;
}
double zalos_dydis_reiksme;
if (zalos_dydis_indeksas < zalos_dydis.Length)
{
zalos_dydis_reiksme = zalos_dydis[zalos_dydis_indeksas];
}
else
{
zalos_dydis_reiksme = 0;
}
double sumos_demuo =
 tikimybe_laiko_momentu_formule71(zalos_dydis_indeksas, turtas, eiluteT);
return sumos_demuo;
}
static double tikimybe_laiko_momentu_formule71(int zalos_dydis_indeksas,
int turtas, double[] eiluteT)
{
double suma = 0;
const int perslenkam_nes_daugiau = 1;
int pabaigos_indeksas5 = turtas + 5  + perslenkam_nes_daugiau;
int pabaigos_indeksas4 = turtas + 4 + perslenkam_nes_daugiau;
int pabaigos_indeksas3 = turtas + 3  + perslenkam_nes_daugiau;
int pabaigos_indeksas2 = turtas + 2  + perslenkam_nes_daugiau;
int pabaigos_indeksas1 = turtas + 1 + perslenkam_nes_daugiau;
int pabaigos_indeksas0 = turtas + 0  + perslenkam_nes_daugiau;
for(int pradzia0 = 0; pradzia0
< Math.Min(pabaigos_indeksas0, zalos_dydis.Length); pradzia0++)
```

```
{
for(int pradzia1 = 0; pradzia1 + pradzia0
< Math.Min(pabaigos_indeksas1, zalos_dydis_antru_laiko_momentu.Length);
 pradzia1++)
{
for(int pradzia2 = 0; pradzia2 + pradzia1 + pradzia0
< Math.Min(pabaigos_indeksas2, zalos_dydis_treciu_laiko_momentu.Length);
 pradzia2++)
{
for(int pradzia3 = 0; pradzia3 + pradzia2 + pradzia1 + pradzia0
< Math.Min(pabaigos_indeksas3, zalos_dydis_antru_laiko_momentu.Length);
 pradzia3++)
{
for(int pradzia4 = 0; pradzia4 + pradzia3 + pradzia2 + pradzia1 + pradzia0
< Math.Min(pabaigos_indeksas4, zalos_dydis.Length); pradzia4++)
{
for(int pradzia5 = 0;
pradzia5 + pradzia4 + pradzia3 + pradzia2 + pradzia1 + pradzia0
< Math.Min(pabaigos_indeksas5,  zalos_dydis_sestu_laiko_momentu.Length);
pradzia5++)
{
double eilutesT_reiksme = 0;
int eilutesT_indeksas = pabaigos_indeksas5-pradzia5-0;
if (eilutesT_indeksas >= 0 && eilutesT_indeksas < eiluteT.Length)
{
eilutesT_reiksme = eiluteT[eilutesT_indeksas];
}
suma = suma + zalos_dydis[pradzia0]
* zalos_dydis_antru_laiko_momentu[pradzia1]
*zalos_dydis_treciu_laiko_momentu[pradzia2]

*zalos_dydis_antru_laiko_momentu[pradzia3]
*zalos_dydis[pradzia4]
*zalos_dydis_sestu_laiko_momentu[pradzia5]
```

```
* eilutesT_reiksme;
}}}}}}} return suma;
}
// Puasono funkcija

static double[] generuoti_zalos_dydi()
{
const double l = 0.8;
const int k = 30;
return generuoti_zalos_dydi_bendras(l, k);
}
static double[] generuoti_zalos_dydi_du()
{
const double l = 0.15;
const int k = 30;
return generuoti_zalos_dydi_bendras(l, k);
}
static double[] generuoti_zalos_dydi_trys()
{
const double l = 0.5;
const int k = 30;
return generuoti_zalos_dydi_bendras(l, k);
}
static double[] generuoti_zalos_dydi_antru_laiko_momentu(double[] lentele1,
double[] lentele2)
{
int max = lentele1.Length + lentele2.Length - 1;
double[] rezultatas = new double[max];
for (int indeksas = 0; indeksas < rezultatas.Length; indeksas++)
{
double suma = 0;
for (int lentele1Idx = 0; lentele1Idx <= indeksas && lentele1Idx
< lentele1.Length; lentele1Idx++)
{
```

```
for (int lentele2Idx = 0; lentele2Idx <= indeksas && lentele2Idx
< lentele2.Length; lentele2Idx++)
{
if (lentele1Idx + lentele2Idx == indeksas)
{
var sandauga = lentele1[lentele1Idx] * lentele2[lentele2Idx];
suma = suma + sandauga ;
}}}
rezultatas[indeksas] = suma ;
}
return rezultatas;
}
static double[] generuoti_zalos_dydi_treciu_laiko_momentu(double[] lentele1,
double[] lentele2)
{
int max = lentele1.Length + lentele2.Length - 1;
double[] rezultatas = new double[max];
for (int indeksas = 0; indeksas < rezultatas.Length; indeksas++)
{
double suma = 0;
for (int lentele1Idx = 0; lentele1Idx <= indeksas && lentele1Idx
< lentele1.Length; lentele1Idx++)
{
for (int lentele2Idx = 0; lentele2Idx <= indeksas && lentele2Idx
< lentele2.Length; lentele2Idx++)
{
if (lentele1Idx + lentele2Idx == indeksas)
{
var sandauga = lentele1[lentele1Idx] * lentele2[lentele2Idx];
suma = suma + sandauga ;
}}}
rezultatas[indeksas] = suma ;
}
return rezultatas;
```

```
}
static double[] generuoti_zalos_dydi_sestu_laiko_momentu(double[] lentele1,
double[] lentele2, double[] lentele3)
{
int max = lentele1.Length + lentele2.Length+ lentele3.Length - 1;
double[] rezultatas = new double[max];
for (int indeksas = 0; indeksas < rezultatas.Length; indeksas++)
{
double suma = 0;
for (int lentele1Idx = 0; lentele1Idx <= indeksas && lentele1Idx
< lentele1.Length; lentele1Idx++)
{
for (int lentele2Idx = 0; lentele2Idx <= indeksas && lentele2Idx
< lentele2.Length; lentele2Idx++)
{
for (int lentele3Idx = 0; lentele3Idx <= indeksas && lentele3Idx
< lentele3.Length; lentele3Idx++)
{
if (lentele1Idx + lentele2Idx == indeksas)
{
var sandauga = lentele1[lentele1Idx] * lentele2[lentele2Idx]
* lentele3[lentele3Idx];
suma = suma + sandauga ;
}}}}
rezultatas[indeksas] = suma ;
}
return rezultatas;
}
static double[] generuoti_zalos_dydi_bendras(double l, int k)
{
double eksp = Math.Exp(-l);
double[] rezultatas = new double[k + 2];
double suma = 0;
for (int indeksas = 0; indeksas < k + 1; indeksas++)
```

```
{
double f = eksp * (Math.Pow(l, indeksas) / fact(indeksas));
rezultatas[indeksas] = f;
suma = suma + f;
}
rezultatas[k + 1] = 1 - suma;
return rezultatas;
}
static double[] zalos_dydis = generuoti_zalos_dydi();
static double[] zalos_dydis_du = generuoti_zalos_dydi_du();
static double[] zalos_dydis_trys = generuoti_zalos_dydi_trys();
static double[] zalos_dydis_antru_laiko_momentu =
generuoti_zalos_dydi_antru_laiko_momentu(zalos_dydis, zalos_dydis_du);
static double[] zalos_dydis_treciu_laiko_momentu =
generuoti_zalos_dydi_treciu_laiko_momentu(zalos_dydis, zalos_dydis_trys);
static double[] zalos_dydis_sestu_laiko_momentu =
 generuoti_zalos_dydi_sestu_laiko_momentu(zalos_dydis,
zalos_dydis_du, zalos_dydis_trys);
// Geometrinės progresijos funkcija
/*
static double[] generuoti_zalos_dydi()
{
const double p = 0.80;
const int k = 200;
return generuoti_zalos_dydi_bendras(p, k);
}
static double[] generuoti_zalos_dydi_du()
{
const double p = 0.5;
const int k = 200;
return generuoti_zalos_dydi_bendras(p, k);
}
static double[] generuoti_zalos_dydi_trys()
{
```

```
const double p = 0.2;
const int k = 200;
return generuoti_zalos_dydi_bendras(p, k);
}
static double[] generuoti_zalos_dydi_antru_laiko_momentu(double[] lentele1,
double[] lentele2)
{
int max = lentele1.Length + lentele2.Length - 1;
double[] rezultatas = new double[max];
for (int indeksas = 0; indeksas < rezultatas.Length; indeksas++)
{
double suma = 0;
// double atimti = 0;
for (int lentele1Idx = 0; lentele1Idx <= indeksas && lentele1Idx
< lentele1.Length; lentele1Idx++)
{
for (int lentele2Idx = 0; lentele2Idx <= indeksas && lentele2Idx
< lentele2.Length; lentele2Idx++)
{
if (lentele1Idx + lentele2Idx == indeksas)
{
var sandauga = lentele1[lentele1Idx] * lentele2[lentele2Idx];
suma = suma + sandauga ;
}}}
rezultatas[indeksas] = suma ;
}
return rezultatas;
}
static double[] generuoti_zalos_dydi_treciu_laiko_momentu(double[] lentele1,
double[] lentele3)
{
int max = lentele1.Length + lentele3.Length - 1;
double[] rezultatas = new double[max];
for (int indeksas = 0; indeksas < rezultatas.Length; indeksas++)
```

```
{
double suma = 0;
// double atimti = 0;
for (int lentele1Idx = 0; lentele1Idx <= indeksas && lentele1Idx
< lentele1.Length; lentele1Idx++)
{
for (int lentele3Idx = 0; lentele3Idx <= indeksas && lentele3Idx
< lentele3.Length; lentele3Idx++)
{
if (lentele1Idx + lentele3Idx == indeksas)
{
var sandauga = lentele1[lentele1Idx] * lentele3[lentele3Idx];
suma = suma + sandauga ;
}}}
rezultatas[indeksas] = suma ;
}
return rezultatas;
}
static double[] generuoti_zalos_dydi_sestu_laiko_momentu(double[] lentele1,
double[] lentele2, double[] lentele3)
{
int max = lentele1.Length + lentele2.Length+ lentele3.Length - 1;
double[] rezultatas = new double[max];
for (int indeksas = 0; indeksas < rezultatas.Length; indeksas++)
{
double suma = 0;
// double atimti = 0;
for (int lentele1Idx = 0; lentele1Idx <= indeksas && lentele1Idx
< lentele1.Length; lentele1Idx++)
{
for (int lentele2Idx = 0; lentele2Idx <= indeksas && lentele2Idx
< lentele2.Length; lentele2Idx++)
{
for (int lentele3Idx = 0; lentele3Idx <= indeksas && lentele3Idx
```

```
< lentele3.Length; lentele3Idx++)
{
if (lentele1Idx + lentele2Idx+ lentele3Idx == indeksas)
{
var sandauga = lentele1[lentele1Idx]
* lentele2[lentele2Idx]* lentele3[lentele3Idx];
suma = suma + sandauga ;
}}}}}
rezultatas[indeksas] = suma ;
}
return rezultatas;
}
static double[] generuoti_zalos_dydi_bendras(double p, int k)
{
double pp = 1-p;
double[] rezultatas = new double[k + 2];
double suma = 0;
for (int indeksas = 0; indeksas < k + 1; indeksas++)
{
double f = Math.Pow(pp, indeksas) * p;
rezultatas[indeksas] = f;
suma = suma + f;
}
rezultatas[k + 1] = 1 - suma;
return rezultatas;
}
static double[] zalos_dydis = generuoti_zalos_dydi();
static double[] zalos_dydis_du = generuoti_zalos_dydi_du();
static double[] zalos_dydis_trys = generuoti_zalos_dydi_trys();
static double[] zalos_dydis_antru_laiko_momentu =
generuoti_zalos_dydi_antru_laiko_momentu(zalos_dydis, zalos_dydis_du);
static double[] zalos_dydis_treciu_laiko_momentu =
generuoti_zalos_dydi_treciu_laiko_momentu(zalos_dydis, zalos_dydis_trys);
static double[] zalos_dydis_sestu_laiko_momentu =
```

```
generuoti_zalos_dydi_sestu_laiko_momentu(zalos_dydis,
zalos_dydis_du, zalos_dydis_trys);
*/
// Bet toks žalų sirstinys
/*
static double[] generuoti_zalos_dydi = new []
{
0.99,
0.01,
0.00
};
static double[] generuoti_zalos_dydi_du = new []
{
0.00,
0.99,
0.01,
0.00
};
static double[] generuoti_zalos_dydi_trys = new []
{
0.00,
0.99,
0.01
};
//pirmos ir antros zalu sandaugos
static double[] generuoti_zalos_dydi_antru_laiko_momentu(double[] lentele1,
double[] lentele2)
{
int max = lentele1.Length + lentele2.Length - 1;
double[] rezultatas = new double[max];
for (int indeksas = 0; indeksas < rezultatas.Length; indeksas++)
{
double suma = 0;
for (int lentele1Idx = 0; lentele1Idx <= indeksas && lentele1Idx
```

```
< lentele1.Length; lentele1Idx++)
{
for (int lentele2Idx = 0; lentele2Idx <= indeksas && lentele2Idx
< lentele2.Length; lentele2Idx++)
{
if (lentele1Idx + lentele2Idx == indeksas)
{
var sandauga = lentele1[lentele1Idx] * lentele2[lentele2Idx];

suma = suma + sandauga ;
}}}
rezultatas[indeksas] = suma ;
}
return rezultatas;
}
//antros ir trecios zalos sandauga
static double[] generuoti_zalos_dydi_treciu_laiko_momentu
(double[] lentele2, double[] lentele3)
{
int max = lentele2.Length + lentele3.Length - 1;
double[] rezultatas = new double[max];
for (int indeksas = 0; indeksas < rezultatas.Length; indeksas++)
{
double suma = 0;
for (int lentele2Idx = 0; lentele2Idx <= indeksas && lentele2Idx
< lentele2.Length; lentele2Idx++)
{
for (int lentele3Idx = 0; lentele3Idx <= indeksas && lentele3Idx
< lentele3.Length; lentele3Idx++)
{
if (lentele2Idx + lentele3Idx == indeksas)
{
var sandauga = lentele2[lentele2Idx] * lentele3[lentele3Idx];
```

```
suma = suma + sandauga ;
}}}
rezultatas[indeksas] = suma ;
}
return rezultatas;
}
//pirmos, antros ir trecios zalos sandauga
static double[] generuoti_zalos_dydi_sestu_laiko_momentu
(double[] lentele1, double[] lentele2, double[] lentele3)
{
int max = lentele1.Length + lentele2.Length + lentele3.Length- 1;
double[] rezultatas = new double[max];
for (int indeksas = 0; indeksas < rezultatas.Length; indeksas++)
{
double suma = 0;
for (int lentele1Idx = 0; lentele1Idx <= indeksas && lentele1Idx
< lentele1.Length; lentele1Idx++)
{
for (int lentele2Idx = 0; lentele2Idx <= indeksas && lentele2Idx
< lentele2.Length; lentele2Idx++)
{
for (int lentele3Idx = 0; lentele3Idx <= indeksas && lentele3Idx
< lentele3.Length; lentele3Idx++)
{
if (lentele1Idx + lentele2Idx + lentele3Idx == indeksas)
{
var sandauga = lentele1[lentele1Idx] * lentele2[lentele2Idx]
* lentele3[lentele3Idx];
suma = suma + sandauga ;
}}}}
rezultatas[indeksas] = suma ;
}
return rezultatas;
}
```

```
static double[] zalos_dydis = generuoti_zalos_dydi;
static double[] zalos_dydis_du = generuoti_zalos_dydi_du;
static double[] zalos_dydis_trys = generuoti_zalos_dydi_trys;
static double[] zalos_dydis_antru_laiko_momentu =
 generuoti_zalos_dydi_antru_laiko_momentu(zalos_dydis, zalos_dydis_du);
static double[] zalos_dydis_treciu_laiko_momentu =
generuoti_zalos_dydi_treciu_laiko_momentu(zalos_dydis, zalos_dydis_trys);
static double[] zalos_dydis_sestu_laiko_momentu =
 generuoti_zalos_dydi_sestu_laiko_momentu(zalos_dydis,
zalos_dydis_du, zalos_dydis_trys);
*/
// Define other methods and classes here
static int fact(int x)
{
int fact=1;
int i=1;
while (i<=x)
{
fact = fact * i;
i++;
}
return fact;
}
```