KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

MATEMATINĖS SISTEMOTYROS KATEDRA

Tadas Dosinas

HURST INDEKSO SKAIČIAVIMO METODŲ TYRIMAS SIMETRINIAMS SAVIPANAŠIEMS PROCESAMS

Magistro darbas

Vadovas
doc. dr. A. Kabašinskas

KAUNAS, 2012

SANTRAUKA

Darbas skirtas savipanašiųjų α-stabilių simetrinių procesų tyrimui. Tokie procesai vyksta finansų rinkose, gamtoje, įvairiose fizinėse sistemose. Procesai buvo modeliuojami, generuojant duomenis su įvairiomis stabilumo parametro α vertėmis, kintančiomis nuo 1,1 iki 2. Sumodeliuoti procesai buvo tirti keturiais sekų analizės metodais: R/S statistikos, absoliutinių momentų, agreguotos dispersijos ir liekanų dispersijos metodais. Tam buvo skaičiuojamas Hurst indeksas, vertinamas jo skaičiavimų vidutinis standartinis nuokrypis. Tyrimams buvo naudojami skirtingo ilgio (200, 5000 ir 10000) duomenys. Buvo tiriama pakartotinai, naudojant 100 ir 1000 duomenų imčių. Tyrimams atlikti buvo sudaryta programa, parašyta C++ kalba. Buvo vertinamas apskaičiuotų Hurst indekso verčių atitikimas teoriniams rezultatams, skaičiuojama vidutinė absoliutinė procentinė paklaida. Tyrimų rezultatai parodė, kad liekanų dispersijos ir absoliutinių momentų metodais buvo gauti žymiai geresni, artimesni teoriniams, rezultatai, nei kitais dviem metodais.

SUMMARY

Dosinas T. Study of Hurst index calculation for symmetric self-similar processes : Master's work in applied mathematics / supervisor dr. assoc. prof. A.Kabašinskas; Department of Mathematical Research in Systems, Faculty of Fundamental Sciences, Kaunas University of Technology. – Kaunas, 2012. – 45 p.

The aim of this paper is to research self-similar α-stable symmetric processes. These processes accure in finance market, nature, various physic systems. Processes were modeled by generating data with different stability parameter α, which was changed from 1,1 to 2. Modeled processes were researched with four different methods: R/S statistic, absoliute moment, aggregated variance and variance of residuals method. Hurst index and its standart deviation were estimated for that purpose. Data of different lengths (2000, 5000, 10000) were researched. This research was made repeatedly with data samples of 100 and 1000. To make this research program was written with C++ programming language. Results of Hurst index values were compared with theorical values also mean absolute pecentage error was calculated for each method. Results of this research showed, that results calculated with variance of residuals method and absolute moments method were much closer to theorical values than results with other two methods.

TURINYS

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

ĮVADAS

Daugelyje statistinių ir stochastinių modelių fundamentaliosios prielaidos yra paremtos nepriklausomais stebėjimais. Modeliai, kurie nesiremia šiomis prielaidomis, turi Markovo savybę, pagal kurią sistemos ateities būsena nepriklauso nuo sistemos praeities būsenos, o priklauso tik nuo dabar esančios sistemos būsenos.

Ilgos atminties fenomenas gamtoje, tokiose įvairiose srityse, kaip hidrologija, ekonomika, chemija, matematika, fizika, gamtos mokslai, aplinkos mokslai, buvo žinomas seniai, dar prieš atsirandant tinkamiems stochastiniams modeliams. Taip pat yra ganėtinai įprasta, jog stebėjimai, atlikti įvairiose laiko arba kitokio mato erdvėse, būtų netrivialiai susiję (koreliuoti).

Jau nuo seno Nilo upė buvo žinoma savo ilgais sausros periodais, besikeičiančiais į ilgus potvynių periodus. Hidrologas Hurst (Hurst, 1951) buvo pirmasis, kuris aprašė šias charakteristikas, kai bandė išspręsti Nilo upės tėkmės reguliavimo problemą. Matematikoje mokslas apie procesus su ilga atmintimi buvo pradėtas Mandelbroto darbu (Mandelbrot, 1975) apie savipanašius ir kitus stacionariuosius stochastinius procesus, kurie pasižymi tolima priklausomybe (angl. *long-range dependence* (*LRD*)). Jis sukūrė mokslo apie šiuos procesus pradmenis, o Hurst matematiškai aprašė fraktalinį Brauno judesį (angliškai *Fractional Brown motion (FBM)*), kuris yra laikomas savipanašių procesų ir procesų su tolima priklausomybe prototipu. Vėliau literatūroje buvo aprašyta dar keletas matematinių ir statistinių klausimų, tokių kaip centrinės ir kitų ribinių teoremų išvedimas, parametrų apskaičiavimo būdai ir metodų modeliavimas.

Savipanašaus arba ilgos atminties proceso parametro H (Hurst parametro) statistinio įvertinimo problema yra labai svarbi ir šiandien. Šis parametras lemia naudojamo modelio matematines ypatybes ir apibūdina pagrindinių fizinių sistemų elgesį.

Hurst pasiūlė žymiąją R/S statistiką (Hurst, 1951) ir sudarė grafinę metodiką, kaip įvertinti ilgos proceso atminties parametrą H. Jis atrado, jog duomenims, gautiems iš Nilo upės, R/S statistika elgiasi kaip konstanta su daugikliu kH, kai k yra laiko intervalas. Vėliau Mandelbrot tai pavadino Hurst efektu, kuris buvo sumodeliuotas fraktaliniu Gauso triukšmu (angl. *Fractional Gaussian noise* (*FGN*)) (Chronopulou, A. ir Viens, F.G., 2009).

Literatūroje galima rasti keletą būdų, susijusių su Hurst parametro (indekso) įvertinimo uždaviniu. Yra nemažai grafinių metodų, įskaitant ir R/S statistiką, autokoreliacijos ir dalinės koreliacijos grafikus, dispersijos grafiką ir variogramą, kurie dažnai naudojami gamtos moksluose ir hidrologijoje. Kadangi metodai, paremti grafiniu sprendimu, nėra labai tikslūs, tai reikalingi griežtesni ir sudėtingesni metodai, pvz., maksimumo tikėtinumo metodas. Fox ir Taqqu (1986) aprašė Whittle apytikslį maksimumo tikėtinumo metodą Gauso proceso atveju, kuris vėliau buvo

apibendrintas ir kai kuriems kitiems procesams. Tačiau šiems metodams trūko skaičiavimų efektyvumo, kas lėmė atsiradimą įvertinimų, paremtų diskrečiąja dispersija ir dažninėmis (banginėmis) procesų savybėmis.

Šiame darbe buvo tiriami stabilūs procesai, juos modeliuojant su skirtingu stabilumo parametru, ir vertinami, naudojant įvairius sekų analizės metodus, skaičiuojant Hurst indeksą.

1. SAVIPANAŠUMAS IR JO SAVYBĖS

1.1 PAGRINDINĖS SĄVOKOS IR APIBRĖŽIMAI

Pradžioje apibrėžkime pagrindines sąvokas, kurios bus naudojamos tolimesnėje analizėje.

Stabilus stochastinis procesas. Stochastinis procesas $\{X(t), \forall t \in T\}$ yra stabilus, jei visi jo baigtiniamačiai skirstiniai yra stabilūs. $\{X(t), \forall t \in T\}$ yra α-stabilus tada ir tik tada, jei visos tiesinės kombinacijos

$$\sum_{k=1}^{d} b_k X(t_k), \qquad \textbf{(1.1)}$$

yra α-stabilios. Čia $d \geq 1,\ t_1, t_2, \dots, t_d \in T, b_1, b_2, \dots, b_d$ - realūs skaičiai. α-stabiliojo proceso pavyzdžiu galėtų būti α-stabilusis Levy procesas.

Stabilusis atsitiktinis dydis žymimas $S_\alpha(\sigma, \beta, \mu)$, kai β – asimetrija, $-1 \leq \beta \leq 1$, $\mu \in R$ – poslinkis, σ yra mastelio parametras, $\sigma > 0$. Kiekvienas stabilusis skirstinys turi stabilumo parametrą α, kuris yra esminis charakterizuojant duomenis. Modeliuojant, pavyzdžiui, finansines sekas paprastai laikoma, kad parametras $\alpha \in (1; 2]$. Kuo mažesnis skirstinio stabilumo parametras, tuo šio skirstinio tankio funkcija bus aukštesnio maksimumo ir sunkesne uodega, jei asimetriškumo parametras β yra lygus nuliui, kaip Gauso atveju, tai tuomet skirstinys yra simetriškas. Jei $\beta > 0$, skirstinys yra pasviręs į dešinę, jei $\beta < 0$, – į kairę. Mastelio parametras apibūdina standartinio nuokrypio apibrėžimą. Stabilusis dispersijos analogas yra variacija v_α, kuris apibūdinamas σ^α (Kabašinskas, 2007).

Levy stochastinis procesas. Stochastinis procesas $\{X(t), \forall t \in T\}$ yra vadinamas Levy procesu (judesiu) jei:

$X(0) = 0$ (beveik tikrai) ir X turi nepriklausomus pokyčius. Pokyčiai $X(t) - X(s)$ yra stacionarūs visiems $0 \leq s < t < \infty$.

Jei pokyčiai yra pasiskirstę pagal α-stabilųjį dėsnį, tai laikoma, kad toks procesas yra α-stabilus Levy procesas (1.1 pav.).

Stacionarus stochastinis procesas. Stochastinis procesas $\{X_n, \forall n \in N\}$ yra laikomas stacionariu, jei vektoriai $(X_{n_1}, \dots, X_{n_d})$ ir $(X_{n_1+m}, \dots, X_{n_d+m})$ yra pasiskirstę pagal tą patį pasiskirstymo dėsnį visiems sveikiesiems $d,\ m \geq 1$ ir $n_1, \dots, n_d \geq 0$.

Gauso procesas. Gauso procesas yra stochastinis procesas $\{X_t; t \in T\}$, kuriam bet kokia baigtinė imties tiesinė kombinacija yra pasiskirsčiusi pagal normalųjį skirstinį.

1.2 SAVIPANAŠUMO PARAMETRAI IR CHARAKTERISTIKOS

Kaip buvo minėta įvade savybė, kuri buvo pastebėta renkant duomenis apie Nilo upę, buvo savipanašumas.

Savipanašumo savybė reiškia, kad duomenų suglaudinimas skirtingais masteliais nekeičia arba mažai keičia duomenų struktūrą. Tarkime, kad $X = \{X(t), t \geq 0\}$ yra Levi procesas. Tada X procesas yra savipanašus tada ir tik tada, jei kiekvienas X(t) yra griežtai stabilus. Stabilumo parametras α ir Hurst indeksas H savipanašiems procesams tenkina sąlygą $\alpha = 1/H$ (Kabašinskas, 2007).

Yra keletas neekvivalenčių savipanašumo apibrėžimų. Populiariausias iš jų teigia, jog tolydaus laiko procecas $Y = \{Y(t), t \in T\}$ yra savipanašus, jei tenkina sąlygą:

$$Y(at) \overset{d}{=} a^H Y(t), \quad \forall t \in T, \forall a > 0, \tag{1.2}$$

čia H-savipanašumo parametras (Hurst indeksas), o $\overset{d}{=}$ yra lygybė pasiskirstymo funkcijos prasme.

Šis savipanašumo rodiklis (Hurst indeksas) kaip savipanašumo charakteristika ir yra dažniausiai naudojamas. Kai rodiklis $0{,}5 \leq H < 1$ sakoma, kad procesas yra su ilga atmintimi, o kai $0 \leq H < 0{,}5$ laikoma, kad procesas yra ergodinis (Kabašinskas, 2007).

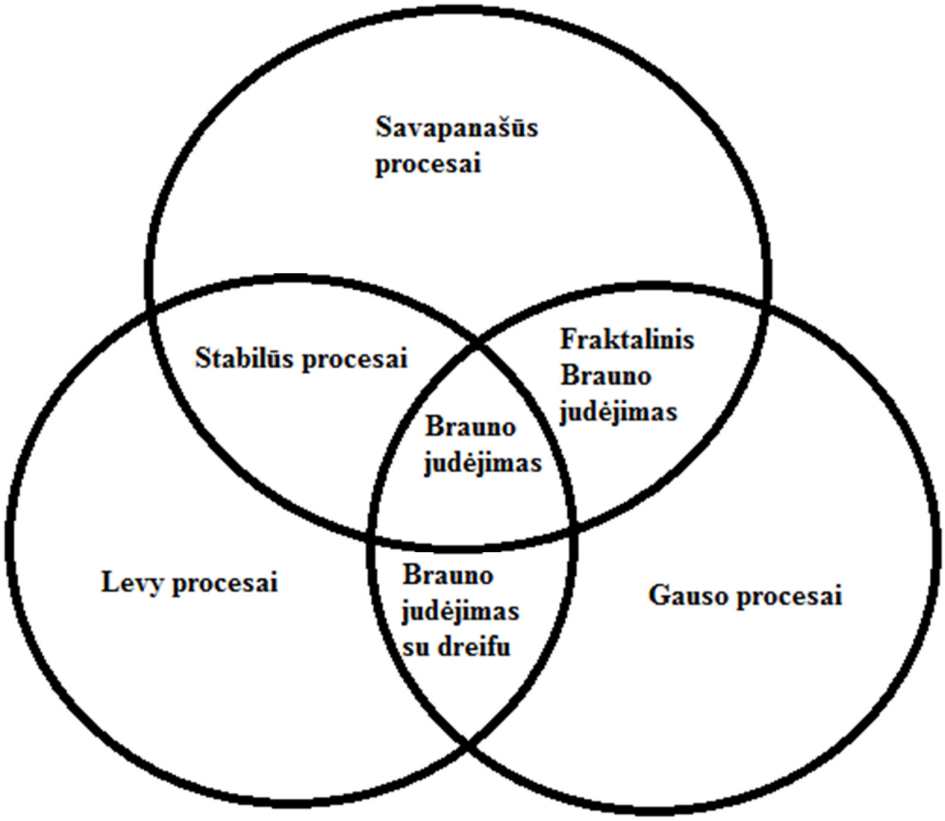
Mandelbrot ir Van Ness (1968) pateikė ryšį kaip fraktalinį Gauso triukšmą tarp savipanašių procesų ir tolimos priklausomybės stacionariose laiko eilutėse. Jie taip pat pateikė Gauso triukšmo spektrinį tankį $f(\lambda) \sim c_H |\lambda|^{1-2H} \left(\frac{1}{2} < H < 1\right)$ su integruojamu poliumu pradžioje, kas privedė prie sąvokos "$1/f$-triukšmas". Trupmeninis Brauno judėjimas yra tipinis savipanašiojo proceso, kurio priaugiai parodo tolimą priklausomybę, pavyzdys. Fraktalinis Brauno judesys su savipanašumo eksponente (Hurst indeksu) $H \in [0,1]$ yra centruotas Gauso procesas su stacionariu prieaugiu $(B_t^H)_{t \geq 0}$ ir su kovariacijos funkcija:

$$Cov(B_t^H, B_s^H) = \frac{1}{2}(|t|^{2H} + |s|^{2H} - |t-s|^{2H}). \tag{1.3}$$

Jei $H = \frac{1}{2}$, tai gauname Brauno judėjimą.

Tačiau savipanašumas nereiškia, jog procesas yra su ilgąja atmintimi: α-stabilus Levy procesas rodo ir savipanašiojo proceso su nepriklausomais teigiamais pokyčiais pavyzdžių. Taip pat, jei procesas yra su ilga atmintimi, nereiškia, jog jis yra savipanašus: yra keletas Gauso proceso pavyzdžių, kurie turi tokias pačias ilgos atminties ypatybes kaip ir fraktalinis Brauno triukšmas, tačiau neturi savipanašumo.

1.1 paveiksle parodytas ryšys tarp savipanašaus, Levy ir Gauso procesų. Jeigu procesas tenktina visų šių trijų procesų sąlygas, tai gauname Brauno judesį, jeigu galioja Gauso ir Levy procesų apibrėžimai, gauname Brauno judėjimą su dreifu. Fraktalinis Brauno judėjimas tenkina Gauso ir savipanašaus procesų sąlygas, o stabilus procesas tenkina Levy bei savipanašaus procesų sąlygas (Kabašinskas, 2007).



1.1 pav. Savipanašių procesų ryšys su Levy ir Gauso procesais

Fraktalinių Brauno judesių ir α-stabilų Levy procesų lyginimas parodo, jog savipanašumo kilmė gali būti labai skirtinga. Savipanašumas gali atsirasti dėl didelio kintamumo, kai proceso

teigiami pokyčiai yra nepriklausomi ir procesas yra su sunkiomis uodegomis (angl. *heavy-tails*), pavyzdžiui, stabilūs Levy procesai. Taip pat savipanašumas gali atsirasti dėl stiprios priklausomybės tarp proceso teigiamų pokyčių, net jeigu pokyčiai yra nedideli. Šiuos savipanašumo mechanizmus Mondelbrot pavadino atitinkamai "Noah efektu" ir "Joseph efektu". Iš šių efektų galima sukonstruoti savipanašius procesus su ilga atmintimi ir sunkiomis uodegomis (fraktaliniai stabilūs procesai) (Chronopulou, A. ir Viens, F.G., 2009).

1.3 ATSITIKTINIŲ α-STABILIŲJŲ DYDŽIŲ GENERAVIMAS

Sakykime, kad norime sugeneruoti atsitiktinį standartinį stabilųjį dydį A, pasiskirsčiusį pagal α-stabilųjį dėsnį $A \sim S_\alpha(1,\beta)$, nusakomą parametru α, asimetrija β ir dispersija $\sigma^2 = 1$. Galima išvesti (http://www.ics.forth.gr), kad šiuo atveju

$$S(\alpha,\beta,1) = D_{\alpha,\beta}\frac{\sin(\alpha(U-U_0))}{(\cos U)^{\frac{1}{\alpha}}}\left(\frac{\cos\left(U-\alpha(U-U_0)\right)}{W}\right)^{\frac{1-\alpha}{\alpha}}, \text{ kai } \alpha \neq 1, \qquad \textbf{(1.4)}$$

o

$$S(1,\beta,1) = \frac{2}{\pi}\left[\left(\frac{\pi}{2}+\beta U\right)\tan U - \beta \ln\left(\frac{\frac{\pi}{2}W\cos U|}{\frac{\pi}{2}+\beta U}\right)\right]. \qquad \textbf{(1.5)}$$

Čia W yra dydis, kurio tikimybė $P\{W > \omega\} = e^{-\omega}, \omega > 0$, U yra tolygaus pasiskirstymo atsitiktinis dydis intervale $(-\frac{\pi}{2},\frac{\pi}{2})$, $D_{\alpha,\beta} = \left[\cos\left(arctan\left(\beta\tan(\pi\frac{\alpha}{2})\right)\right)\right]^{-\frac{1}{\alpha}}$, $o\ U_0 = -\frac{\pi}{2}\beta\left(\frac{k(\alpha)}{\alpha}\right)$, kai $k(\alpha) = 1-|1-\alpha|$. Iš šio α-stabiliojo dydžio A galima gauti α-stabilųjį dydį A_1 su kitokia dispersija σ^2:

$$A_1 = \sigma^{\frac{2}{\alpha}}A. \qquad \textbf{(1.6)}$$

2. HURST INDEKSO ĮVERTINIMO METODAI

Hurst indeksui įvertinti yra nemažai metodų, tačiau dažniausiai literatūroje yra minimi metodai, kuriuos galima suskirstyti į dvi grupes: vertinimo metodai, grįsti sekų analize ir metodai, grįsti dažninėmis (banginėmis) procesų savybėmis.

Vertinimo metodai grįsti sekų analize yra šie:

1. absoliutinių reikšmių metodas (angl. *Absolute Value*) arba absoliutinių momentų (angl. *Absolute Moments*);
2. dispersijos metodas(angl. *Variance*) arba agreguotos dispersijos (angl. *Aggregate Variance*);
3. R/S metodas;
4. modelio liekanų dispersijos metodas (angl. *Variance of Residuals*).

Pagrindiniai metodai, grįsti dažninėmis(banginėmis) procesų savybėmis yra:

1. periodogramos metodas;
2. Whittle metodas;
3. Abry ir Veitch metodas.

Toliau trumpai apžvelgsime šiuos Hurst indekso įvertinimo metodus.

2.1 METODAI PAGRĮSTI SEKŲ ANALIZE

2.1.1 R/S METODAS

Šis metodas yra vienas žinomiausių. Jis yra aprašytas Mandelbroto ir Wallis, taip pat jį yra aprašęs Taqqu. Laiko eilutėms $X = \{X_i,\ i \geq 1\}$ su daline suma $Y(n) = \sum_{i=1}^{n} X_i$ ir imties daline dispersija

$$S^2(n) = \frac{1}{n}\sum_{i=1}^{n} X_i^2 - \left(\frac{1}{n}\right)^2 Y(n)^2, \qquad \textbf{(2.1)}$$

R/S statistika (angl.- *rescaled adjusted range*) aprašoma taip:

$$\frac{R}{S}(n) = \frac{1}{S(n)}\left[\max_{0\leq t\leq n}\left(Y(t) - \frac{t}{n}Y(n)\right) - \min_{0\leq t\leq n}\left(Y(t) - \frac{t}{n}Y(n)\right)\right]. \qquad \textbf{(2.2)}$$

Fraktaliniam Gauso triukšmui arba fraktaliniams ARIMA procesams

$$E\left[\frac{R}{S(n)}\right] \sim C_H n^H, \tag{2.3}$$

kai $n \to \infty$, o C_H yra teigiama baigtinė konstanta, nepriklausanti nuo *n*.

Toliau aptarsime, kaip naudojant R/S statistiką yra nustatomas Hurst indeksas *H*. Eilutes, kurių ilgis *N*, padalinkime į *K* blokų, kurių ilgis *N/K*. Tuomet kiekvienam *n* sudarykime santykius $\frac{R(k_i,n)}{S(k_i,n)}$, pradedant nuo taškų $k_i = i\frac{N}{K} + 1,\ i = 1,2,\ldots,$ tokius, kad $k_i + n \leq N$. Visoms *n* vertėms, kurios yra mažesnės už *N/K*, gauname *K* skirtingų $R(n)/S(n)$ įverčių. Kai *n* artėja prie *N*, gauname vis mažiau įverčių ir gauname tik vieną vertę, kai $n \geq N - N/K$.

Tada brėžiama grafinė $\log\left[\frac{R(k_i,n)}{S(k_i,n)}\right]$ priklausomybė nuo $\log n$ ir, parenkant logaritminiu masteliu išsidėstytas *n* reikšmes, kiekvienam *n* gauname taškus tame grafike. Šis grafikas kartais yra vadinamas pox („raupsų“) grafiku *R/S* statistikai. Parametras *H* gali būti įvertinamas sudarant tiesę iš „pox“ grafiko taškų. Kadangi eilutės su „trumpa atmintimi“ dažniausiai įtakoja grafiko pradžią, todėl nustatomas taškas, nuo kurio naudojama tiesė *H* įvertinimui, neįtraukiant grafiko dalies, esančios prieš tą tašką. Dažniausiai grafiko pabaiga taip pat neįtraukiama, nes joje yra per mažai taškų grafike, ir todėl negalime atlikti patikimų H įvertinimų. Taigi, *n* reikšmės tarp grafiko pradžios ir pabaigos (nuo nustatyto pradinio iki nustatyto galinio taško) yra naudojamos įvertinant Hurst indeksą *H* (Taqqu, Teverovsky, 1997).

Šiandien R/S analizė dažniausiai naudojama hidrologiniams tyrinėjimams, kur *H* vertė yra įvairiai interpretuojama, kaip priklausomybės proceso atminties ilgio arba nereguliarumo indikatorius. Kai kurie autoriai R/S analizę naudojo norėdami analizuoti hidrologinius duomenų parametrus, tokius kaip upės tėkmė, krituliai, temperatūra. R/S analizė buvo pritaikyta norint įvertinti kasmetinę Logano upės tėkmę 70 metų periodui. Hurst indeksas apytiksliai buvo lygus 0,8, o koreliacija tarp gretimų duomenų elementų buvo lygi 0,4. Abdibi ir Collins R/S statistika buvo naudojama prognozuojant ilgalaikius sausros bei potvynių reiškinius. J.Jablonskis surinko bei tyrinėjo Nemuno upės tėkmes kasmetinius duomenis, surinktuos nuo 1812 iki 1991m.

2.1.2 ABSOLIUTINIŲ MOMENTŲ METODAS

Tarkime, kad agreguotos eilutės (2.4) yra gaunamos padalinus duotas *N* ilgio eilutes į *m* ilgio blokus, ir vidurkinant kiekvieną eilutę pagal kiekvieną bloką:

$$X^{(m)} = \frac{1}{m}\sum_{i=(k-1)m+1}^{km} X_i,\ k = 1,2 \dots, \left[\frac{N}{m}\right]. \quad \textbf{(2.4)}$$

Paimame šios eilutės absoliutinį *n*-tąjį momentą

$$AM_n^{(m)} = \frac{1}{N/m}\sum_{k=1}^{N/m}\left|X^{(m)}(k) - \overline{X}\right|^n, \quad \textbf{(2.5)}$$

kai $X^{(m)}$, yra aprašomas (2.4) išraiška, o $\overline{X}$ yra bendras eilutės vidurkis. Tokia agreguota eilutė $X^{(m)}$ asimptotiškai elgiasi kaip ir $Cm^{n(H-1)}$ dideliems *m* ir todėl $AM_n^{(m)}$ yra proporcingas $m^{n(H-1)}$.

Šis metodas dažniausiai naudojamas, kai $n = 1$ (absoliutinio vidurkio atvejis). Kai $n = 2$, šis metodas tampa agreguotos dispersijos metodu.

m reikšmėms gauti, duotos imties agreguotos eilutės absoliutinis momentas yra brėžiamas kaip priklausomybė nuo m, logaritminu masteliu. Gautas rezultatas turėtų būti tiesi linija su $n(H-1)$ nuolydžiu. Praktikoje ši linija ir jos nuolydis yra įvertinamas mažiausių kvadratų metodu. Daroma prielaida, jog abu, tiek *N* tiek ir *N/m* yra dideli. Tai užtikrina, jog ir blokų ilgis, ir jų skaičius yra didelis. Jeigu eilutės nepasižymi ilga atmintimi ir turi baigtinę dispersiją, tuomet Hurst indeksas $H = 0.5$ ir sudarytos kreivės nuolydis turėtų būti $-n/2$. Praktikoje, taikant mažiausių kvadratų metodą, pati grafiko pradžia ir pabaiga nėra naudojamos, nes "trumpa atmintis" (angl. *short-range*) gali iškreipti indekso *H* įvertinimą grafiko pradžioje, o grafiko pabaigoje yra per mažai blokų, nes jeigu blokų ilgis *m* yra didelis, tai jų skaičius tampa per mažas, kad gauti patikimas $AM_n^{(m)}$ vertes. (Taqqu, Teverovsky, 1997).

2.1.3 AGREGUOTOS DISPERSIJOS METODAS

Tarkime, kad agreguotos eilutės (2.6), kaip ir absoliutaus momento metodo atveju, yra gaunamos, padalinus N ilgio duomenis į m ilgio blokus ir suvidurkinus pagal kiekvieną bloką:

$$X^{(m)} = \frac{1}{m}\sum_{i=(k-1)m+1}^{km} X_i, k = 1,2 \dots, \left[\frac{N}{m}\right]. \quad \textbf{(2.6)}$$

Tuomet apskaičiuojama dispersija

$$\widehat{Var}X^{(m)} = \frac{1}{N/m}\sum_{k=1}^{N/m}\left(X^{(m)}(k) - \overline{X}\right)^2, \quad \textbf{(2.7)}$$

Eilutės blokų vidurkiai $X^{(m)}$ yra proporcingi dydžiui m^{H-1}. Jeigu eilutės yra pasiskirsčiusios pagal Gauso dėsnį arba bent jau turi baigtinę dispersiją, tai gautoji dispersija bus asimptotiškai proporcinga m^{2H-2} dideliems *N/m* ir *m*.

Teisingom *m* reikšmėms gauti, agreguotos dispersijos (2.7) logaritmas yra brėžiama spriklausomai nuo *m* logaritmo. Gautas rezultatas turėtų būti tiesi linija su $2H-2$ nuolydžiu. Praktikoje nuolydis dažniausiai yra įvertinamas mažiausių kvadratų metodu, brėžiant tiesę per gautuosius taškus. Daroma prielaida, jog abu *N* ir *N/m* yra dideli. Tai užtikrina, jog blokų ilgis, ir jų skaičius yra didelis. Jeigu eilutės nepasižymi ilga atmintimi ir turi baigtinę dispersiją, tuomet $H = 0.5$ ir sudarytos tiesės nuolydis turėtų būti -1. Praktikoje pati grafiko pradžia ir pabaiga, konstruojant mažiausių kvadratų metodu kreivę, nėra naudojamos, nes „trumpa atmintis“ gali iškreipti *H* įvertinimą, jeigu naudojama grafiko pradžia, o grafiko pabaigoje yra per mažai blokų.

2.1.4 LIEKANŲ DISPERSIJOS METODAS

Šiuo atveju N ilgio duomenys pirmiausiai yra padalinami į m dydžio blokus. Tuomet suskaičiuojamos kiekvieno bloko dalinės sumos $Y(t) = \sum_{i=1}^{t} X_i$. Naudojant mažiausių kvadratų ar kitą panašų metodą, šioms dalinėms sumoms yra sudaroma tiesė $a + bt$ ir surandamos jų liekanų dispersijos

$$\frac{1}{m}\sum_{t=1}^{m}(Y(t) - a - bt)^2. \quad \textbf{(2.8)}$$

Taip gaunama $\frac{N}{m}$ liekanų dispersijos įverčių. Tuomet skaičiuojamos šių liekanų dispersijos įverčių medianos (arba vidurkiai). Šių įverčių medianos yra proporcingos m^{2H} (vidurkiai proporcingi m^{2d+1}, kai $d = H - \frac{1}{\alpha}$). Šias medianas (vidurkius) logaritmuojant ir brėžiant grafiką nuo $\log m$ gaunama tiesė su nuolydžiu 2H (vidurkių atveju – su nuolydžiu $2d + 1$) . Iš šio nuolydžio surandamas Hurst indeksas. Stabiliems arba Gauso procesams mažiausias blokų ilgis gali būti ganėtinai mažas ($m = 5$).

2.2 METODAI PAREMTI DAŽNINĖMIS PROCESŲ SAVYBĖMIS

2.2.1 PERIODOGRAMOS METODAS

Periodograma yra aprašoma kaip

$$I(v) = \frac{1}{2\pi N}\left|\sum_{j=1}^{N} X(j)e^{ijv}\right|, \quad \textbf{(2.9)}$$

kai v yra dažnis, N – eilutės ilgis, o X yra laiko eilutė. Kai dispersija yra baigtinė, $I(v)$ yra spektrinio tankio įvertis, o eilutės su ilga atmintimi turės spektrinį tankį proporcingą $|v|^{1-2H}$ dažniams, kurie yra artimi nuliui. Log-log plokštumoje periodogramos priklausomybė nuo dažnio sudaro tiesią liniją, turinčią $1-2H$ nuolydį. Praktikoje skaičiavimams naudojama tik 10% žemiausių dažnių , nes šie skaičiavimai tinka tik dažniams artimiems nuliui.

2.2.2 WHITTLE IR KITI METODAI

Whittle įvertis yra vertė vektoriaus η, kuris minimizuoja funkciją

$$Q(\eta) = \int_{-\pi}^{\pi} \frac{I(v)}{f(v;\eta)}dv + \int_{-\pi}^{\pi} \log f(v;\eta)\, dv. \quad \textbf{(2.10)}$$

Šios išraiškos dėmuo $\int_{-\pi}^{\pi} \log f(v;\eta)\, dv$ gali būti prilygintas 0 normalizuojant $f(v;\eta)$. Normalizavimas priklauso tiktai nuo mastelio parametro ir nepriklauso nuo kitų η komponentų. Taigi, pakeičiamas f į f^* taip, kad $f^* = \beta f$ ir $\int_{-\pi}^{\pi} \log f^*(v;\eta)\, dv = 0$. Pirmasis (2.10) išraiškos dėmuo lieka nepakeistas, išskyrus dalybą iš β. Parametras β yra laikomas papildomu parametru, kurio nereikia įvertinti. Kadangi β nepriklauso nuo kitų parametrų, tai jis neturės įtakos $Q(\eta)$ funkcijos minimizavimui. Taip pat vietoje integralų sudaroma Furjė dažnių suma $v_j = \frac{2\pi j}{N}$, $j = 1,2,\dots,[(N-1)/2]$, kai N yra eilutės duomenų ilgis. Todėl tikra funkcija, kurią algoritmas minimizuoja yra

$$Q^*(\eta) = \sum_{j=1}^{[(N-1)/2]} \frac{I(v_j)}{f^*(v_j;\eta)}. \quad \textbf{(2.11)}$$

Jei turime fraktalinį Gauso triukšmą, tai η, remiantis (Fox ir Taqqu, 1996), yra parametras *H*.

Abry ir Veitch tipo metodai operuoja dažninėmis vilnelių (angl. *wavelet*) savybėmis.

2.3 METODŲ TIKSLUMO VERTINIMAS

2.3.1 KOLMOGOROVO SMIRNOVO TESTAS

Norint ištirti, ar gautieji vienu ar kitu metodu rezultatai yra pasiskirstę pagal tam tikrą dėsnį, vienas iš būdų yra Kolmogorovo Smirnovo testas. Šiuo atveju yra tiriamos hipotezės:

H_0: Duomenys, pasiskirstę pagal tiriamą dėsnį,

H_a: Duomenys nėra pasiskirstę pagal šį dėsnį.

Tuomet apskaičiuojama Kolmogorovo Smirnovo statistika D:

$$D = \max_{1 \le i \le N}(F(Y_i) - \frac{i-1}{N}, \frac{i}{N} - F(Y_i)), \quad \textbf{(2.12)}$$

kai F yra teorinė tiriamo skirstinio pasiskirstymo funkcija, N – tiriamų duomenų kiekis, o Y – surikiuoti pagal dydį tiriami duomenys (Engineering statistics handbook).

Hipotezė H_0 yra priimama, jeigu gautoji statistika $D < d$, kai kritinė vertė d su tam tikru reikšmingumo lygmeniu yra paimama iš žinyno lentelių.

2.3.2 T-TESTAS

T-testas dažniausiai yra taikomas duomenims, pasiskirsčiusiems pagal normalųjį dėsnį. Jis tikrina, ar tiriamų duomenų vidurkis statistiškai nesiskiria nuo nurodyto dydžio μ_0:

$$t = \frac{\bar{x}-\mu_0}{s}\sqrt{n}, \quad \textbf{(2.13)}$$

kai $\bar{x}$ yra tiriamų duomenų vidurkis, s – duomenų standartinis nuokrypis, o n – laisvės laipsnių skaičius.

Tiriant t-testą taip pat sudaromos hipotezės:

H_0: Duomenų vidurkis, statistiškai nesiskiria nuo μ_0,

H_a: Gautųjų Hurst indekso įverčių vidurkis statistiškai skiriasi nuo μ_0.

Hipotezė H_0 yra priimama, jei gautoji statistika t (su tam tikru reikšmingumo lygmeniu.) patenka į intervalą, kurio kraštinės reikšmės yra paimamos iš žinyno lentelių.

3. SAVIPANAŠIŲJŲ PROCESŲ TYRIMAS

Šių tyrimų tikslas buvo ištirti α-stabiliuosius procesus, naudojantis metodais, paremtais sekų analize. Tiriant šiuos procesus buvo naudoti RS, absoliučių momentų, agreguotos dispersijos ir liekanų dispersijos metodai. Tyrimui buvo generuojamos α-stabiliojo skirstinio duomenų imtys. Duomenų skaičius imtyje buvo keičiamas didėjimo kryptimi. Imčių tyrimai taip pat buvo daromi pakartotinai su skirtingo dydžio imtimis. Be to, kad būtų galima įsitikinti sudarytų apskaičiuoto Hurst indekso įvertinimo programų darbo teisingumu, buvo generuojami ir duomenys pasiskirstę pagal normalųjį skirstinį.

Tyrimo skaičiavimams realizuoti buvo parašytos programos C++ kalba. Šių programų kodai pateikti darbo prieduose.

3.1 α-STABILIOJO PROCESO DUOMENŲ GENERAVIMAS

Savipanašiųjų procesų tyrimui buvo generuojami atsitiktiniai duomenys, pasiskirstę pagal α-stabilųjį dėsnį. Tam buvo realizuojama (1.4) išraiška ir buvo naudojamas atvejis, kai asimetrijos koeficientas $\beta = 0$, o dispersija standartinė ($\sigma^2 = 1$).

3.2 PROCESO HURST INDEKSO SKAIČIAVIMAS R/S METODU

Tiriant savipanašųjį procesą R/S metodu buvo realizuotos (2.1) ir (2.2) išraiškos.

Pradžioje buvo sudarytas skaičiavimo metodas dispersijai $S^2(n)$ skaičiuoti. Tam buvo sugeneruoti α-stabiliojo skirstinio duomenys

$X = \{X_i, i \in [1, N]\}$, kai *N* yra duomenų kiekis.

Po to buvo sudarytas skaičiavimo metodas $\frac{R}{S}(n)$ išraiškai (2.2) apskaičiuoti. Tam pradžioje buvo apskaičiuoti $Y(t) - \frac{t}{n}Y(n)$ masyvai, kai $t \in [0; n]$. Naudojantis šių masyvų rezultatais bei jau anksčiau sudarytu metodu dispersijai $S^2(n)$ surasti, buvo sudarytas metodas apskaičiuoti santykiams $\frac{R}{S}(n)$.

Duomenys, kurių ilgis *N* buvo sudalinami į *K* blokų, kurių ilgis $\frac{N}{K}$. Tada kiekvienam *n* buvo sudaroma $\frac{R(k_i,n)}{S(k_i,n)}$, pradedant nuo taškų $k_i = i\frac{N}{K} + 1, i = 1,2, \ldots,$ imant k_i ir n tokius, kad $k_i + n \leq N$. Vertės $\frac{R(k_i,n)}{S(k_i,n)}$ buvo suvidurkinamos, kad būtų gaututa vienas įvertis kiekvienam n. Tuomet buvo sudaromi du masyvai, masyvas $\log\frac{R(n)}{S(n)}$ ir masyvas $\log n$. Įvertinant tai, jog $\log\frac{R,n)}{S(,n)}$ priklausomybė nuo $\log n$ yra su nuolydžiu H, buvo sudaroma regresinė lygtis $\log\frac{R(,n)}{S(,n)} = H\log n + a$. Kadangi a neturi įtakos nuolydžiui, tai Hurst indeksas buvo surandamas koviariaciją tarp šių masyvų $Cov(\log n, \log\frac{R(,n)}{S(n)})$ padalinant iš masyvo $\log n$ dispersijos:

$$H = \frac{Cov(\log n, \log\frac{R(n)}{S(,n)})}{\mathrm{Var}(\log n)}. \quad \textbf{(3.1)}$$

3.3 HURST INDEKSO SKAIČIAVIMAS ABSOLIUTINIŲ MOMENTŲ METODU

Absoliutinių momentų metodas Hurst indekso skaičiavimui buvo remtasi 2.1.2 skyriuje aprašytomis (2.4) ir (2.5) išraiškomis. Skaičiuojant šiuo metodu, pradžioje duomenys buvo padalinami į m ilgio blokus, ir po to vidurkinami pagal kiekvieną bloką.

Kitame skaičiavimo etape buvo surandamas duomenų imties vidurkis $\bar{X}$ ir buvo sudarytas metodas sumai (2.5) surasti, kai $n = 1$. Gautoji suma ir yra pirmasis absoliutinis momentas $AM_1^{(m)}$– absoliutinis vidurkis.

Pradinis blokų ilgis buvo imamas $m = 2$ ir didinamas eksponentiniu žingsniu $\frac{e^i}{i+1}$, kai $i = 1,2,3, \ldots$, tol, kol blokų kiekis $\frac{N}{m} \geq 20$. Čia *N* – duomenų kiekis imtyje. Taip buvo daroma todėl, kad ilgų blokų yra mažai ir jie iškraipo rezultatus.

Kadangi $AM_1^{(m)}$ absoliutinis momentas yra proporcingas m^{H-1}, tai *H* indeksui gauti tiek m reikšmės, tiek ir absoliutinis momentas buvo logaritmuojami, surandama kovariacija $Cov(\log m, \log AM_1^{(m)})$ bei dispersija $Var(m)$ ir iš tiesinės regresinės lygties surandamas Hurst indeksas

$$H = \frac{Cov(\log m, \log AM_1^{(m)})}{Var(\log m)} + 1. \quad \textbf{(3.2)}$$

3.4 PROCESO HURST INDEKSO SKAIČIAVIMAS AGREGUOTOS DISPERSIJOS METODU

Savipanašaus proceso Hurst indekso skaičiavimo agreguotos dispersijos metodas yra panašus į absoliutinių momentų metodą. Kaip ir absoliutinių momentų metodo atveju buvo duomenys padalinti į *m* ilgio blokus ir vidurkinami pagal kiekvieną bloką. Tik šiuo atveju buvo skaičiuojama ne pirmasis absoliutinis momentas, bet agreguota dispersija $\widehat{Var}X^{(m)}$ (2.7).

Pradinis blokų ilgis buvo imamas $m = 2$ ir didinamas eksponentiniu žingsniu $\frac{e^i}{i+1}$, kai $i = 1,2,3, \dots$, tol, kol blokų kiekis $\frac{N}{m} \geq 20$. Čia *N* – duomenų kiekis imtyje. Taip buvo daroma todėl, nes mažas kiekis ilgų blokų iškraipo rezultatus.

Kadangi šiuo atveju agreguota dispersija yra proporcinga m^{2H-2}, tai Hurst indeksui rasti m reikšmės ir agreguota dispersija taip pat buvo logaritmuojami. Analogiškai kaip ir absoliutinių momentų metodu, pasinaudojus tiesine regresijos lygtimi buvo surastas Hurst indeksas

$$H = 0,5\frac{Cov(\log m, \log \widehat{Var}X^{(m)})}{Var(\log m)} + 1. \quad \textbf{(3.3)}$$

3.5 HURST INDEKSO SKAIČIAVIMAS LIEKANŲ DISPERSIJOS METODU

Kaip ir prieš tai aprašytuose metoduose, sudarytame liekanų dispersijos skaičiavimo metode tiriami duomenys pirmiausia buvo padalinami į m ilgio blokus. Tuomet buvo ieškomos dalinės sumos $Y(t)$ atskiruose blokuose. Turint šias dalines sumas, pasinaudojus regresijos lygtimi $Y = a + bt$, kiekvienam blokui buvo surandami nuolydžio tiesės koeficientai a ir b. Koeficientų b suradimui buvo suskaičiuota kovariacijos $cov(t, Y)$ ir dispersijos $Var(t)$ ir vidurkiai $\bar{Y}$ bei $\bar{t}$. Kiekvieno bloko koeficientai b buvo surandami naudojant išraišką $b = \frac{Cov(t,Y)}{Var(t)}$, o koeficientui a – išraišką $a = \bar{Y} - b\bar{t}$. Suradus regresijos lygtį $Y = a + bt$, buvo surandama liekanų dispersija (2.8).

Turint liekanų dispersijas kiekvienam blokui, buvo apskaičiuojama jų mediana M. Blokų ilgis m buvo didinamas eksponentiniu žingsniu $\frac{e^i}{i+1}$, kai $i = 1,2,3, \dots$, tol, kol bloko ilgis m neviršija $\frac{N}{2}$, kai *N* – duomenų kiekis imtyje. Skaičiavimai buvo atliekami pradedant nuo pradinio blokų ilgio $m = 5$.

Turint medianas skirtingiems blokų ilgiams m, visam procesui buvo surandama priklausomybė $\log M$ nuo $\log m$, kai M – liekanų dispersijų mediana. Turint šią priklausomybę ir žinant, jog jos nuolydis yra $2H$, vėl buvo sudaryta tiesinės regresijos lygtis $\log M = 2H \log m + a$. Įvertinus, kad koeficientas a tiesės nuolydžiui įtakos neturi, iš šios lygties buvo randamas Hurst indeksas H:

$$H = 0{,}5 \frac{Cov(\log m, \log M)}{Var(\log m)}. \quad \textbf{(3.4)}$$

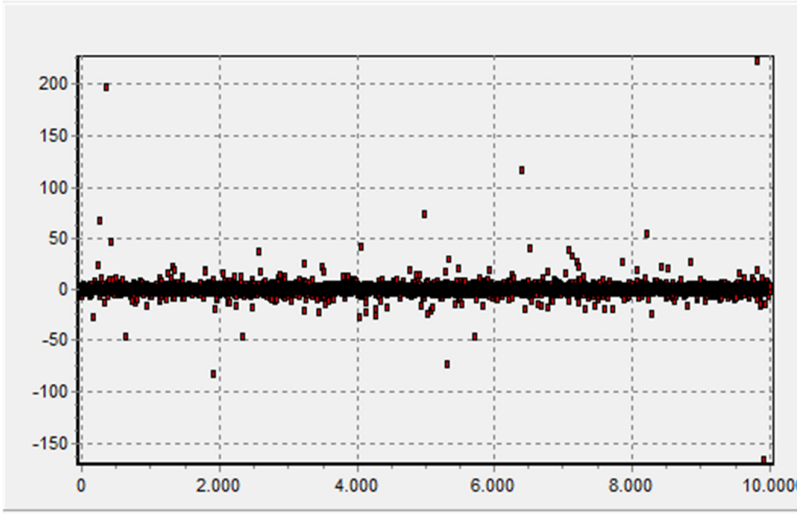
4. SAVIPANAŠIŲJŲ PROCESŲ TYRIMO REZULTATAI

Tiriant savipanašiuosius procesus buvo generuojamos α-stabiliojo skirstinio duomenų imtys. Generuojant imčių duomenis, buvo keičiamas α-stabiliojo skirstinio parametras α ($1 < \alpha \leq 2$) žingsniu 0,1 ir buvo tiriama šio parametro įtaka Hurst indeksui.

Kiekvienoje imtyje buvo naudojama nuo 2000 iki 10000 duomenų, kurių skaičius tyrimo eigoje buvo keičiamas. Tyrimo eigoje taip pat buvo naudojamas ir skirtingas imčių skaičius – 100 ir 1000. Visiems metodams tirti buvo naudojami tie patys duomenys.

4.1 GENERUOJAMI DUOMENYS

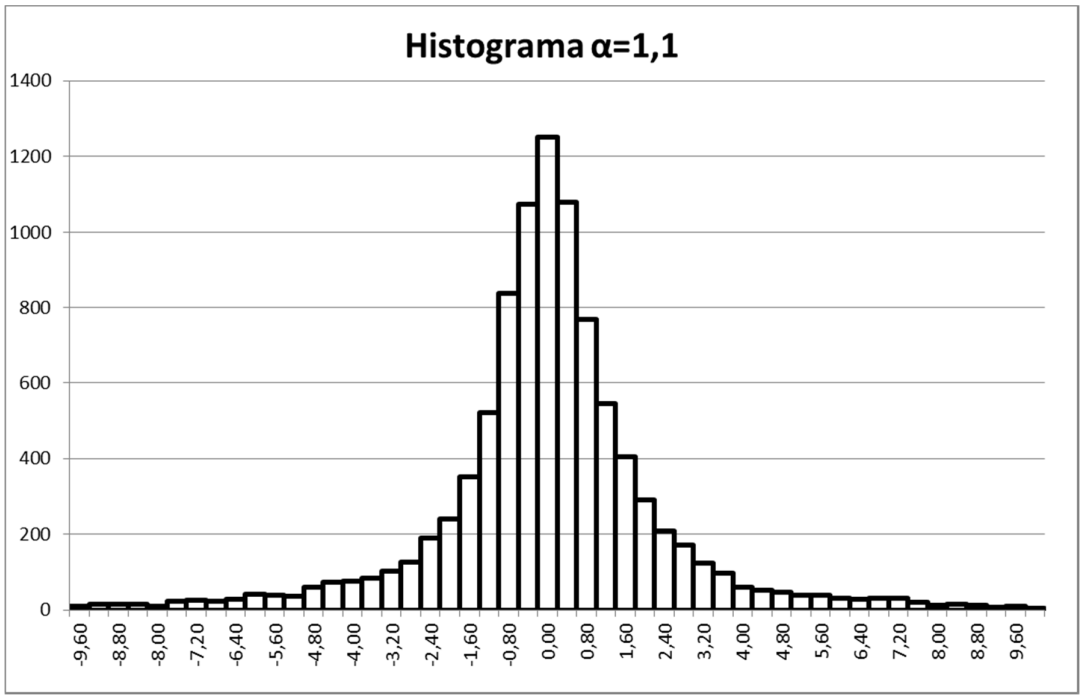
Duomenys tyrimams buvo generuojami naudojantis 3.1 skyriuje aprašyta metodika. 4.1 pav. parodytas tokių sugeneruotų α-stabilaus proceso duomenų pavzdys, kai duomenų skaičius $N = 10000$, o stabilumo parametras $\alpha = 1{,}5$.



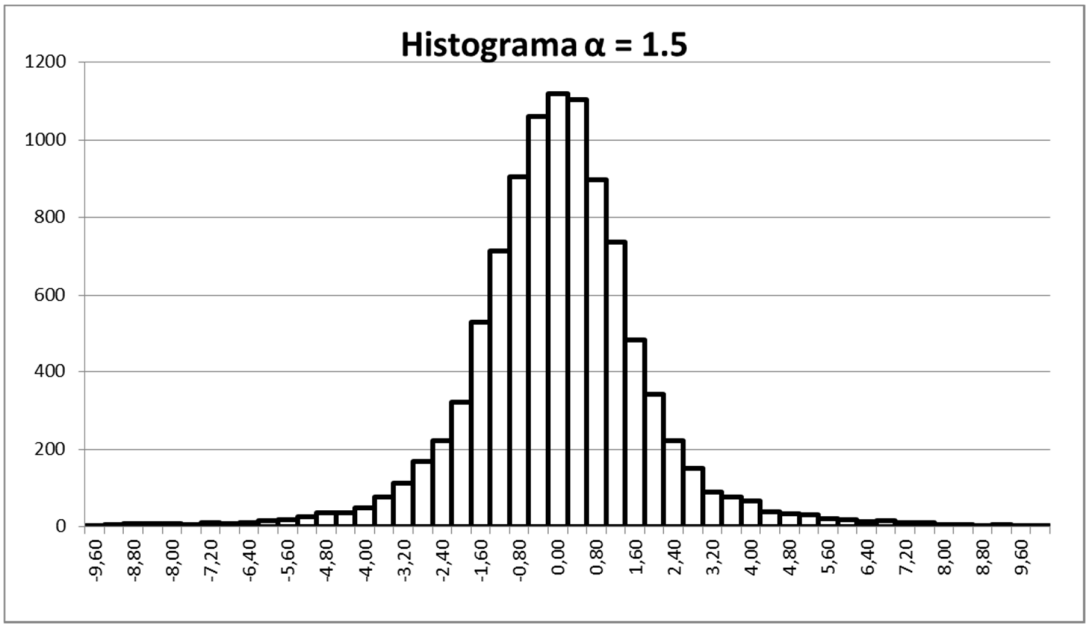
4.1pav. Sugeneruoti 10000 ilgio α-stabiliejieji duomenys, kai $\alpha = 1,5$

Sugeneruotų duomenų histogramų pavyzdžiai prie skirtingų α parametro verčių parodyti 4.2, 4.3 ir 4.4 paveiksluose. Kiekvienu atveju buvo generuojama po 10000 atsitiktinių dydžių, pasiskirsčiusių pagal α-stabilųjį dėsnį. 4.2 pav. pateiktas gautas α-stabilusis skirstinys, kai parametras $\alpha = 1{,}1$, 4.3 pav. – kai $\alpha = 1{,}5$, o 4.4 pav. – kai $\alpha = 2$. Kai parametras α lygus 1,1 ir 1,5, matosi ilgosi pasiskirstymo uodegos. Gautų duomenų histogramos atvaizduotos naudojantis

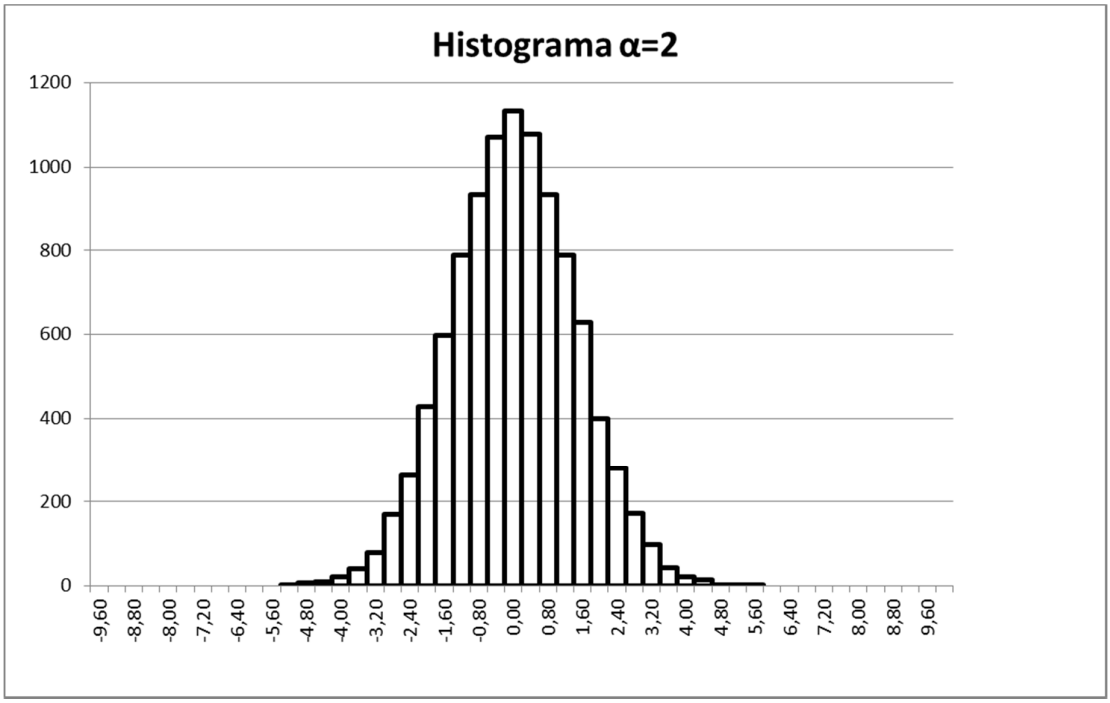
Visual basic kalba parašyta atvaizdavimo programa Mircosoft excel aplinkoje. Programos kodas pateiktas 3 priede.



4.2pav. α-stabiliojo dėsnio duomenų histograma, kai $\alpha = 1,1$



4.3 pav. α-stabiliojo dėsnio duomenų histograma, kai $\alpha = 1,5$



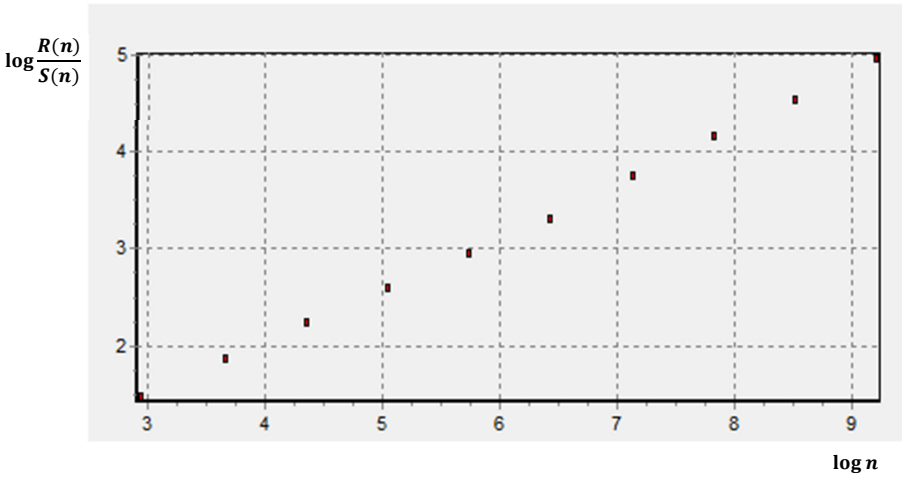
4.4 pav. α-stabiliojo dėsnio duomenų histograma, kai $\alpha = 2$

Kaip matyti iš pateiktų pavyzdžių, didinant parametrą α, duomenų pasiskirstymo uodegos trumpėja, ir kai $\alpha = 2$ ilgos uodegos dingsta ir procesas virsta Gausiniu (Taqqu ir Teverovsky, 1996), gauname normalųjį skirstinį.

4.2 TYRIMO R/S METODU REZULTATAI

Tiriant buvo naudojami sugeneruoti atsitiktiniai dydžiai, pasiskirstę pagal α-stabilųjį dėsnį, kai parametras α buvo keičiamas nuo 1,1 iki 2. Buvo tiriamos imtys su 2000, 5000 ir 10000 duomenų. Skaičiavimai buvo atlikti su 100 ir su 1000 imčių.

Buvo skaičiuojamos $\frac{R}{S}(n)$ statistikos (2.6), skirtingiems dalinių sumų ilgiams n. Šių statistikų logaritmo priklausomybės nuo dalinių sumų ilgių n logaritmo pavyzdys, kai $\alpha = 1{,}5$, parodytas 4.5 paveiksle. Parodytam atvejui buvo generuojama 10000 duomenų. Iš šių rezultatų buvo sudaroma regresinė lygtis, iš kurios gaunamas posvyrio kampo koeficientas, atitinkantis Hurst indeksą. Tam, kad rezultatai nebūtų iškraipomi prie mažų n, skaičiavimai buvo atlikti, kai $n > 8$.



4.5 pav. Funkcijos $\log\frac{R(n)}{S(n)}$ priklausomybės nuo $\log n$ pavyzdys, kai sugeneruota 10000 atsitiktinių dydžių, pasiskirsčiusių pagal α-stabilųjį dėsnį

Gautieji suvidurkinto pagal imtis Hurst indekso įvertinimo rezultatai prie skirtingų parametrų pateikti 4.1 lentelėje. Čia *N* – duomenų skaičius imtyje, *k* – imčių kiekis. 4.2 lentelėje pateikti Hurst indekso skaičiavimo standartiniai nuokrypiai prie tų pačių parametrų α, *N* ir *k*.

4.1 lentelė

Suvidurkintos Hurst indekso vertės prie skirtingų parametrų *k, N ir* α (R/S metodas)

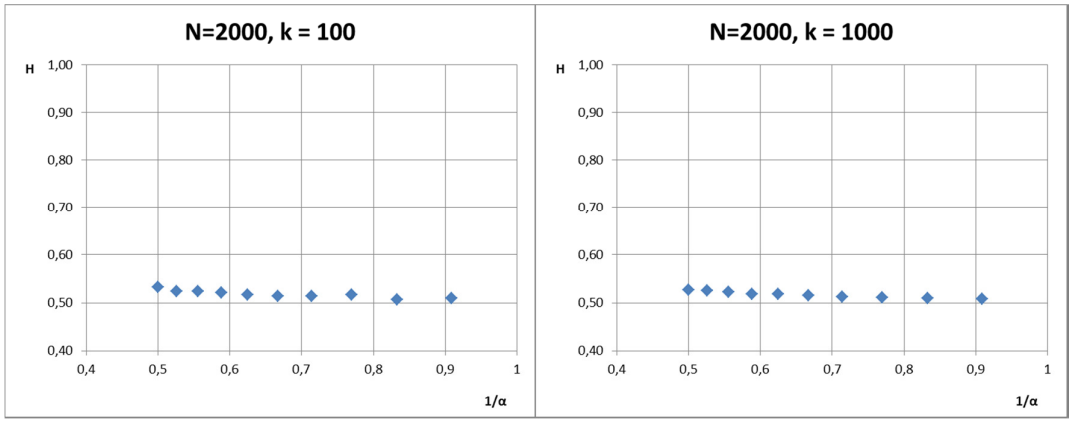
α	1/α	N=2000, k=100	N=5000, k=100	N=10000, k=100	N=2000, k=1000	N=5000, k=1000	N=10000, k=1000
1,1	0,9091	0,509694	0,505177	0,508014	0,507849	0,50766	0,507997
1,2	0,8333	0,507122	0,513007	0,510057	0,510049	0,511892	0,508512
1,3	0,7692	0,516093	0,510839	0,51036	0,510406	0,51331	0,511563
1,4	0,7143	0,514321	0,512852	0,512196	0,511965	0,515483	0,512414
1,5	0,6667	0,513892	0,514781	0,515565	0,515391	0,51642	0,514226
1,6	0,6250	0,517376	0,522821	0,517508	0,518559	0,519555	0,516083
1,7	0,5882	0,52121	0,516891	0,518787	0,518489	0,520787	0,519617
1,8	0,5556	0,524309	0,523793	0,518992	0,521824	0,522739	0,521036
1,9	0,5263	0,523874	0,523334	0,523948	0,524683	0,524232	0,523548
2	0,5000	0,532627	0,528042	0,52257	0,526963	0,52772	0,526387

4.2 lentelė

Skaičiavimų standartiniai nuokrypiai prie skirtingų *k, N* ir α (R/S metodas)

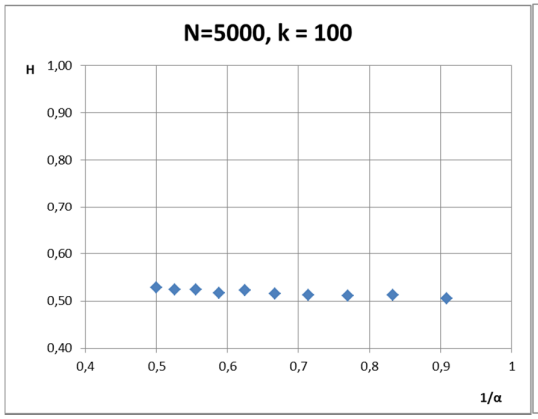
α	N=2000, k=100	N=5000, k=100	N=10000, k=100	N=2000, k=1000	N=5000, k=1000	N=10000, k=1000
1,1	0,024399	0,017533	0,034883	0,022216	0,019195	0,027923
1,2	0,024605	0,021118	0,036372	0,023779	0,020039	0,030233
1,3	0,022419	0,021350	0,037962	0,025100	0,021232	0,028810
1,4	0,024211	0,022227	0,037837	0,026239	0,022700	0,029787
1,5	0,027620	0,025344	0,041746	0,028057	0,023567	0,029801
1,6	0,029273	0,021103	0,042510	0,027816	0,024275	0,030852
1,7	0,026023	0,022432	0,045380	0,028835	0,025282	0,030655
1,8	0,027260	0,026006	0,044356	0,031158	0,026407	0,032540
1,9	0,031123	0,026624	0,045393	0,030496	0,025841	0,033128
2,0	0,031632	0,026876	0,047879	0,031726	0,026907	0,032192

4.6 paveiksle parodytos gautos Hurst indekso priklausomybės nuo $\frac{1}{\alpha}$ su skirtingais duomenų kiekiais, naudojant 100 (duomenys kairėje paveikslo pusėje) ir 1000(duomenys dešinėje paveikslo pusėje) imčių.

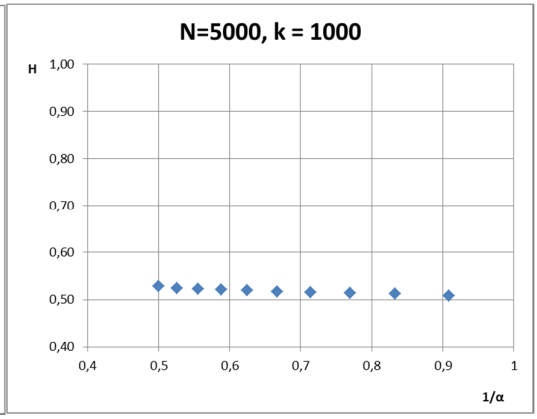


a

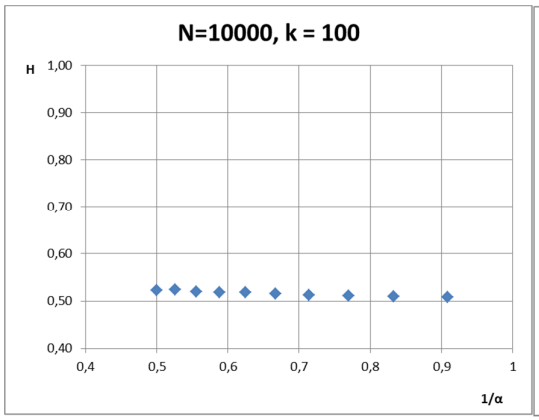
d



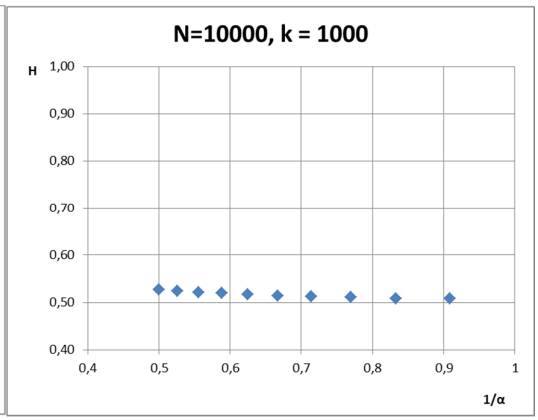
b



e



c



f

4.6 pav. Gautosios R/S metodu Hurst indekso vertės priklausomai nuo $\frac{1}{\alpha}$, kai keičiamas duomenų skaičius *N* imtyje ir imčių skaičius *k*

Kaip matyti iš pateiktų rezultatų, Hurst indekso vertės yra netoli 0,5 ir beveik nepriklauso nuo parametro α (turi tendenciją didėti, didėjant α) ir praktiškai nepriklauso nei nuo duomenų kiekio *N*, nei nuo imčių skaičiaus *k*.

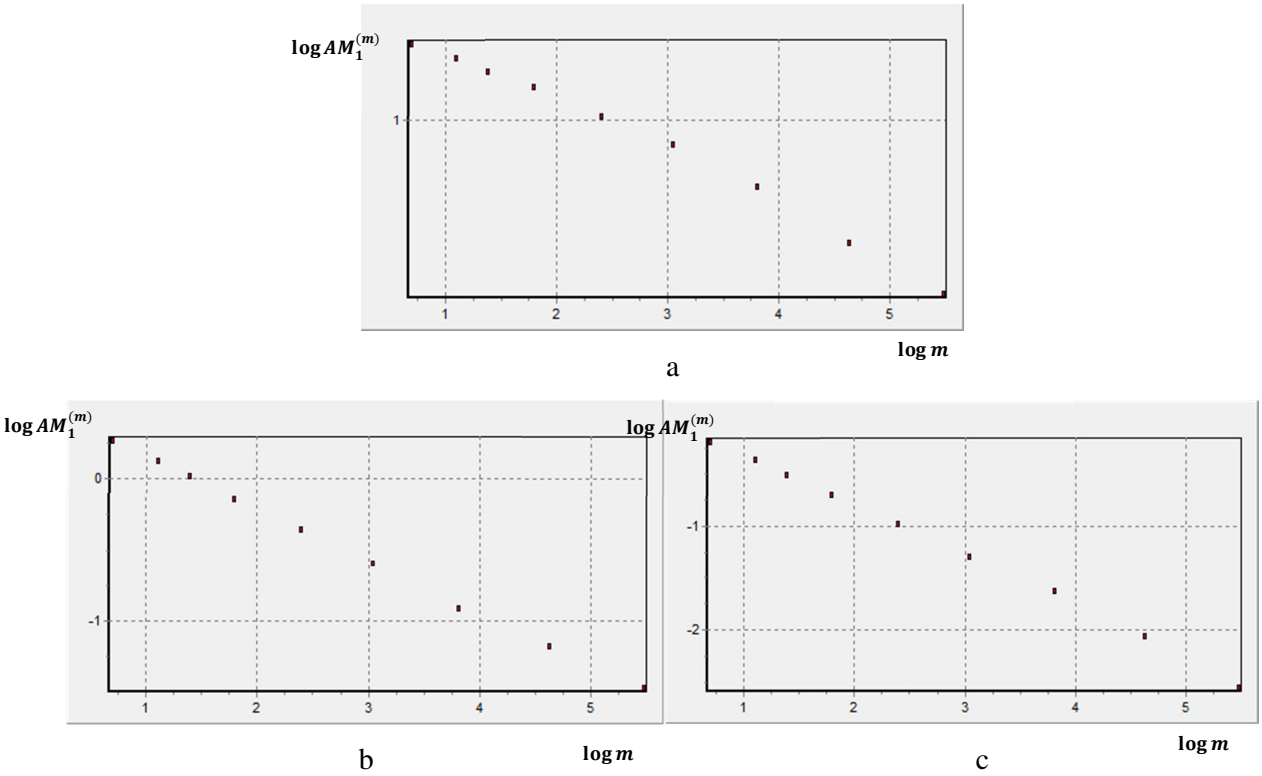
Didėjant parametrui α, H indekso standartiniai nuokrypiai taip pat mažai keičiasi (pradžioje šiek tiek padidėja, o po to vėl mažėja). Didinant duomenų kiekį, šis maksimumas slenka α parametro didėjimo kryptimi.

Taigi, šis tyrimas rodo, kad šis metodas tirtiems duomenims nėra tikslus. Šis metodas neparodo ilgų pasiskirstymo funkcijos uodegų.

4.3 TYRIMO ABSOLIUTINIŲ MOMENTŲ METODU REZULTATAI

Tiriant α-stabiliuosius procesus šiuo metodu, generuojant tyrimui reikalingus duomenis, buvo naudojami tie patys jų parametrai *α, N* ir *k* kaip ir R/S metodu.

Skaičiavimo metu gauto tarpinio rezultato – sumos (2.5) logaritmo, kai $n = 1$, priklausomybės nuo blokų ilgio m logaritmo vienai duomenų imčiai prie skirtingų parametro α verčių parodytos 4.7 paveiksle.



4.7 pav. Funkcijos $\log AM_1^{(m)}$ priklausomybės nuo $\log m$ pavyzdžiai, kai sugeneruota 10000 atsitiktinių dydžių, pasiskirsčiusių pagal α-stabilųjį dėsnį ir parametras α lygus 1,1 (a), 1,5 (b) ir 1,9 (c)

Kaip buvo minėta ankstesniame skyriuje, turint šiuos rezultatus, kiekvienai imčiai buvo sudaromos regresinės lygtys ir naudojant (3.3) išraišką surandama viena Hurst indekso vertė. Taip gaunama Hurst indekso k įverčių, kurie vidurkinami pagal imtis.

Suskaičiuotos Hurst indekso vertės, kintant parametrui α prie skirtingų duomenų skaičiaus N ir imčių kiekio k, pateiktos 4.3 lentelėje, o skaičiavimų standartiniai nuokrypiai – 4.4 lentelėje.

4.3 lentelė

Suvidurkintos Hurst indekso vertės prie skirtingų parametrų *k, N* ir α (absoliutinių momentų metodas)

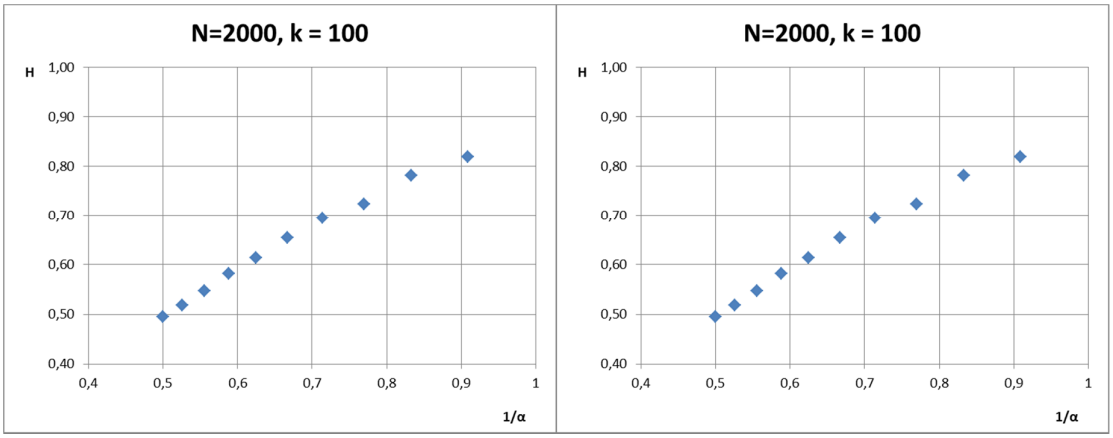
α	1/α	N=2000, k=100	N=5000, k=100	N=10000, k=100	N=2000, k=1000	N=5000, k=1000	N=10000, k=1000
1,1	0,9091	0,81903	0,8326	0,8433	0,815544	0,823414	0,831097
1,2	0,8333	0,782083	0,782193	0,789027	0,775823	0,780282	0,787415
1,3	0,7692	0,723249	0,735495	0,750373	0,733713	0,737154	0,740312
1,4	0,7143	0,694174	0,692511	0,696695	0,694849	0,693219	0,700399
1,5	0,6667	0,653693	0,66172	0,640727	0,648414	0,653348	0,656618
1,6	0,6250	0,613616	0,62555	0,61497	0,61155	0,616346	0,616802
1,7	0,5882	0,58155	0,581522	0,583542	0,578466	0,584279	0,582776
1,8	0,5556	0,547357	0,549769	0,557383	0,549248	0,550634	0,551024
1,9	0,5263	0,518309	0,521634	0,524109	0,519642	0,521874	0,522885
2	0,5000	0,495023	0,497918	0,494358	0,495897	0,496225	0,497709

4.4 lentelė

Skaičiavimų standartiniai nuokrypiai prie skirtingų *k, N* ir α (absoliutinių momentų metodas)

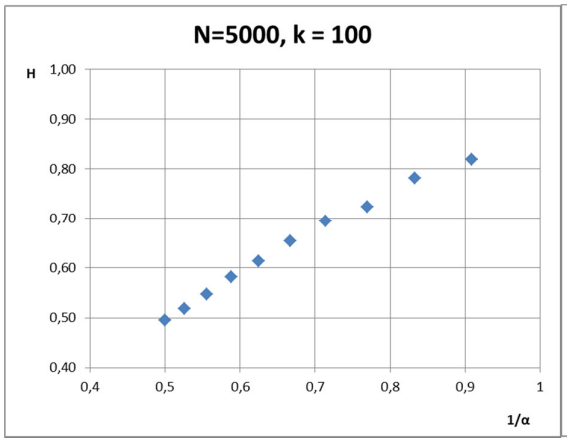
α	n=2000, k=100	n=5000, k=100	n=10000, k=100	n=2000, k=1000	n=5000, k=1000	n=10000, k=1000
1,1	0,076480	0,071487	0,064899	0,074992	0,066091	0,064257
1,2	0,077869	0,078723	0,065848	0,082371	0,070725	0,067553
1,3	0,084792	0,063690	0,074032	0,081112	0,067919	0,065593
1,4	0,079547	0,056083	0,063101	0,078043	0,064740	0,064451
1,5	0,074692	0,073522	0,051612	0,070649	0,060665	0,058251
1,6	0,063936	0,070648	0,044812	0,062952	0,054279	0,049454
1,7	0,052654	0,054975	0,052934	0,054745	0,049397	0,039001
1,8	0,045068	0,042768	0,043540	0,047746	0,037518	0,035991
1,9	0,035843	0,021985	0,041559	0,038823	0,030475	0,030872
2	0,033884	0,026273	0,022109	0,033389	0,022773	0,019759

Vaizdumo dėlei 4.3 lentelės rezultatai pateikti grafikais 4.8 paveiksle.

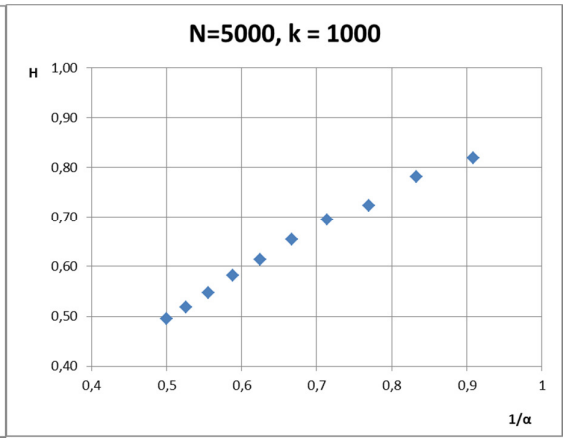


a

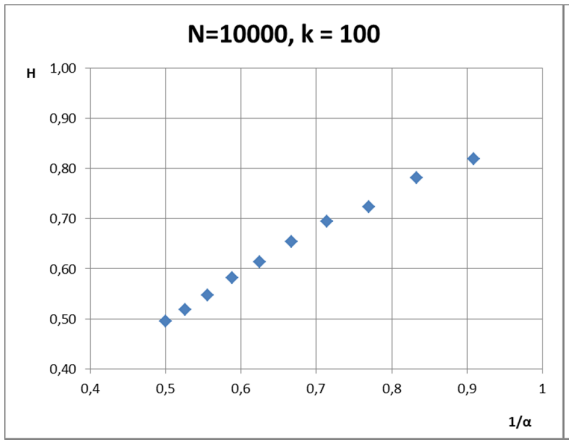
d



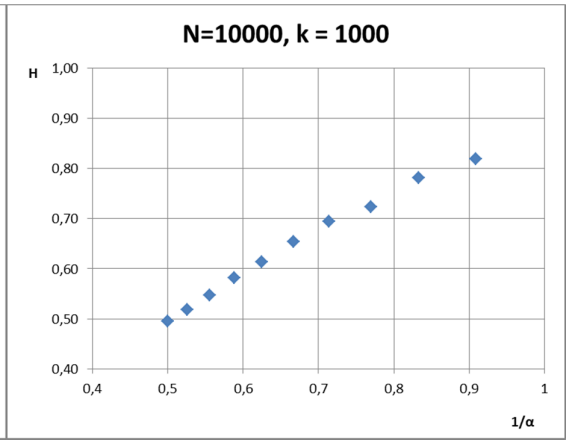
b



e



c



f

4.8 pav. Gautosios absoliutinių momentu metodu Hurst indekso vertės priklausomai nuo $\frac{1}{\alpha}$, kai keičiamas duomenų skaičius *N* imtyje ir imčių skaičius *k* lygus100 (a, b,c) ir 1000 (d,e,f)

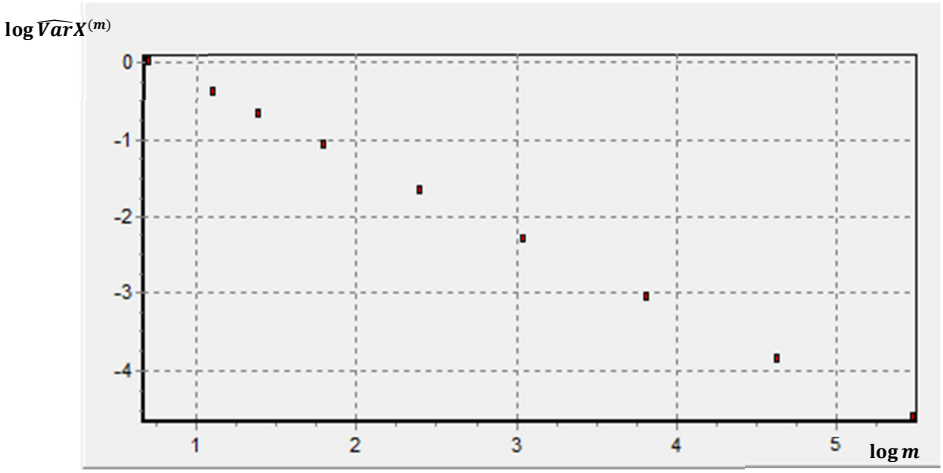
Iš gautų rezultatų galime pastebėti, kad gautasis absoliutinių momentų metodu, kai vertinamas pirmasis momentas, *H* indeksas reaguoja į proceso ilgas uodegas. Keičiantis parametrui α nuo 1,1

iki 2, Hurst indeksas kito nuo 0,83...0,84 iki 0,495...0,498. Tai rodo, kad *H* vertės apytiksliai yra lygios $\frac{1}{\alpha}$. Keičiant duomenų skaičių nuo 2000 iki 10000, vidutinės Hurst indekso vertės mažai keičiasi. Palyginus nedidelę įtaką turi ir tiriamų duomenų imčių kiekis.

Proceso parametrui α keičiantis nuo 1 iki 2, standartiniai nuokrypiai pradžioje šiek tiek padidėja, o po to sumažėja kelis kartus.

4.4 TYRIMO AGREGUOTOS DISPERSIJOS METODU REZULTATAI

Kaip buvo minėta 3.4 skyriuje, šis metodas panašus į absoliutinių momentų metodą, tik skaičiuojant Hurst indeksą šiuo atveju buvo ieškoma ne pirmasis absoliutinis momentas, o antrasis momentas, t. y. agreguota dispersija $\widehat{Var}X^{(m)}$. Kaip ir kituose metoduose, buvo naudojami tie patys parametrai α, *N* ir *k*. Agreguotos dispersijos logaritmo priklausomybės nuo blokų ilgio parametro m logaritmo pavyzdys, kai parametras $\alpha = 1{,}5$, pateiktas 4.9 paveiksle. Kaip jau buvo minėta anksčiau, šios priklausomybės nuolydis yra proporcingas $2H - 2$ dydžiui. Analogiškai kaip ir kituose metoduose, sudarius regresijos lygtis, buvo gautos Hurst indekso (3.4) vertės (4.5 lentelė) ir jo įvertinimų standartiniai nuokrypiai (4.6 lentelė). Hurst indekso verčių priklausomybės nuo $\frac{1}{\alpha}$ grafiškai pateiktos 4.10 paveiksle.



4.9 pav. Funkcijos log $\widehat{Var}X^{(m)}$ priklausomybės nuo log m pavyzdys, kai α lygus 1,5

4.5 lentelė

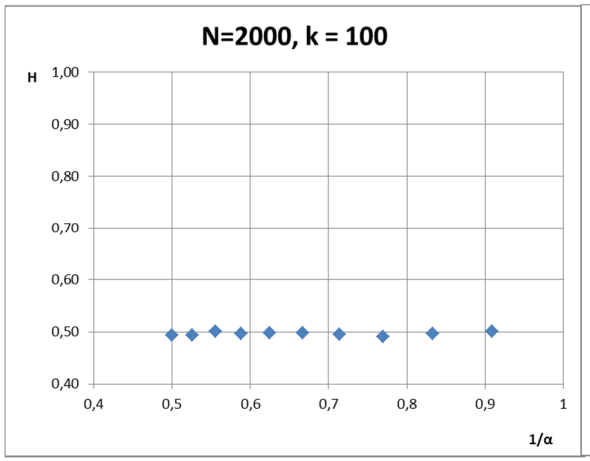
Hurst indekso vertės prie skirtingų parametrų *k*, *N* ir α (agreguotos dispersijos metodas)

α	1/α	N=2000, k=100	N=5000, k=100	N=10000, k=100	N=2000, k=1000	N=5000, k=1000	N=10000, k=1000
1,1	0,9091	0,500591	0,493994	0,498297	0,496193	0,495733	0,498267
1,2	0,8333	0,496263	0,496558	0,498156	0,496967	0,49688	0,497958
1,3	0,7692	0,490002	0,493898	0,498523	0,494873	0,496648	0,498204
1,4	0,7143	0,494955	0,494511	0,499336	0,497197	0,495781	0,498623
1,5	0,6667	0,497662	0,498355	0,498653	0,494912	0,495179	0,497309
1,6	0,6250	0,498487	0,495554	0,499272	0,498344	0,4952	0,49785
1,7	0,5882	0,496675	0,493351	0,498369	0,495767	0,49599	0,498333
1,8	0,5556	0,501232	0,491892	0,496913	0,495909	0,4953	0,498246
1,9	0,5263	0,493466	0,496341	0,495802	0,495712	0,496177	0,497415
2	0,5000	0,493856	0,496342	0,495693	0,496586	0,495295	0,498358

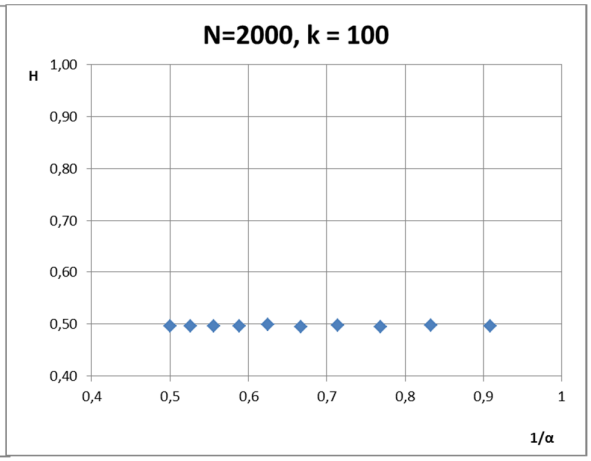
4.6 lentelė

Skaičiavimų standartiniai nuokrypiai prie skirtingų *k*, *N* ir α (agreguotos dispersijos metodas)

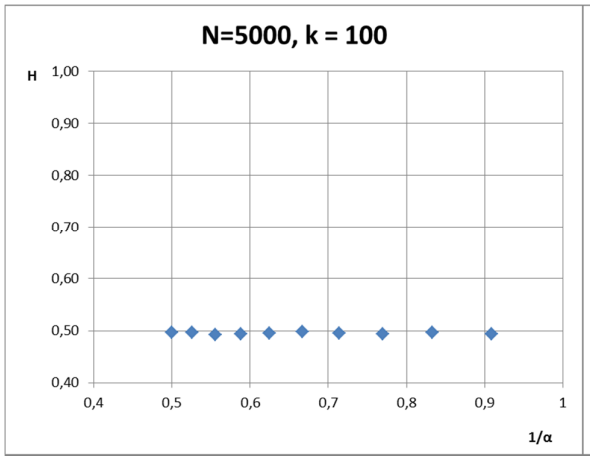
α	N=2000, k=100	N=5000, k=100	N=10000, k=100	N=2000, k=1000	N=5000, k=1000	N=10000, k=1000
1,1	0,028882	0,019794	0,012313	0,030547	0,019771	0,011905
1,2	0,033654	0,014471	0,013040	0,024234	0,016832	0,016297
1,3	0,024806	0,017792	0,020643	0,034532	0,018840	0,011398
1,4	0,022512	0,017706	0,013084	0,023458	0,019856	0,014707
1,5	0,028278	0,019866	0,014962	0,029624	0,020631	0,014259
1,6	0,022305	0,022111	0,012568	0,027025	0,021982	0,014428
1,7	0,027599	0,018883	0,015579	0,027096	0,021391	0,015045
1,8	0,026046	0,019714	0,016445	0,026327	0,021427	0,015014
1,9	0,031050	0,021454	0,013060	0,028847	0,022277	0,016182



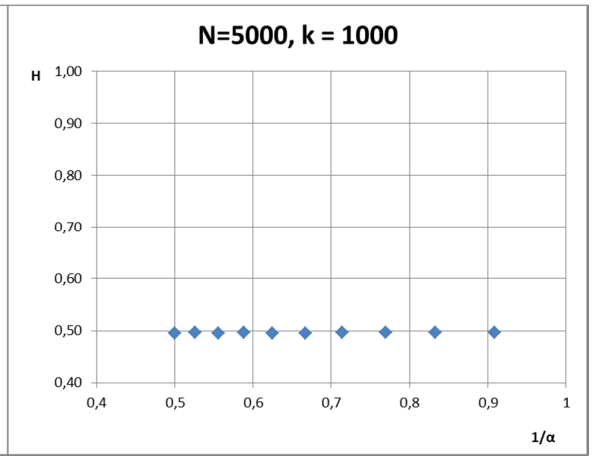
a



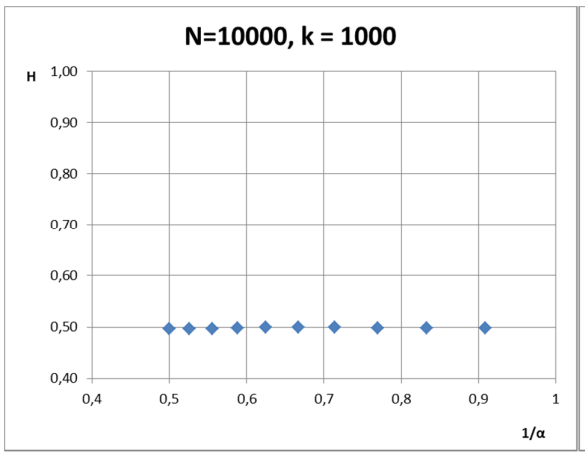
d



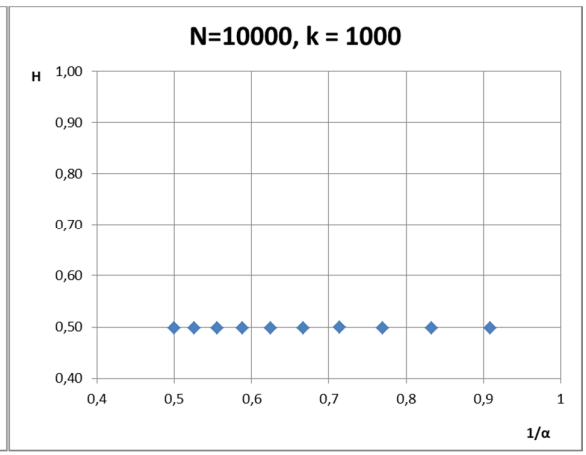
b



e



c



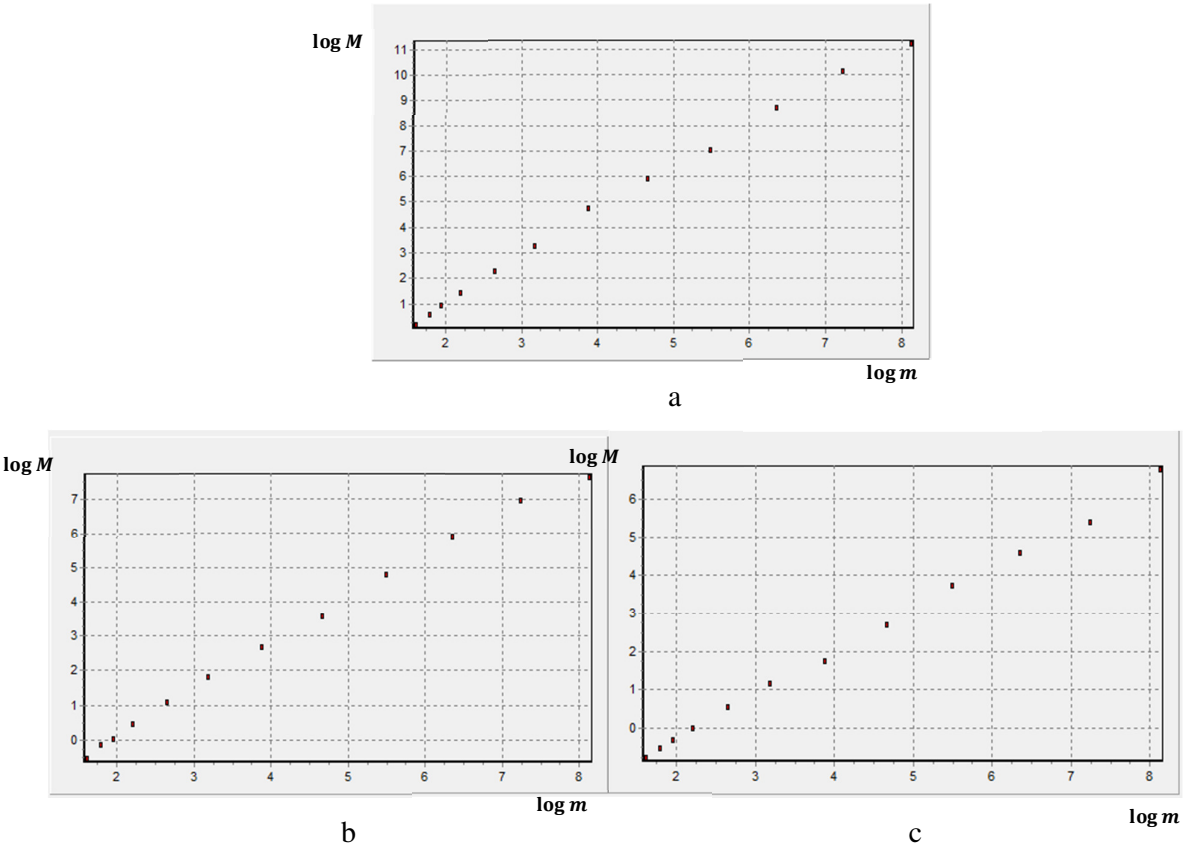
f

4.10 pav. Agreguotos dispersijos metodu gautos Hurst indekso verčių priklausomybės nuo $\frac{1}{\alpha}$, kai keičiamas duomenų skaičius *N* imtyje ir imčių skaičius *k* lygus 100 (a, b,c) ir 1000 (d,e,f)

Šiuo atveju, kaip ir tyrimo R/S metodu atveju, Hurst indekso vertė, ypač kai imčių skaičius didelis ($k = 1000$) praktiškai nepriklauso nuo parametro α. Todėl, kai α yra netoli vieneto (t.y. procesas turi ilgas uodegas) šio metodo rezultatams jos įtakos neturi. Hurst idenkso skaičiavimo vidutinis standartinis nuokrypis, ypač prie didesnio duomenų skaičiaus *N*, yra mažesnis nei R/S metodu, tačiau mažesnis nei absoliutinių momentų metodu. Galima teigti, kad tiek R/S metodas, tiek ir agreguotos dispersijos metodas tiriamiems duomenims nėra tikslus.

4.5 TYRIMO LIEKANŲ DISPERSIJOS METODU REZULTATAI

Kaip ir tiriant procesus kitais metodais, šiuo atveju taip pat buvo naudojami tie patys parametrai α, *N* ir *k*. 4.11 paveiksle pateikti liekanų dispersijų medianos logaritmo priklausomybės nuo blokų ilgio (parametro m) logaritmo pavyzdžiai, kai parametras α lygus 1,1 (a), 1,5 (b) ir 1,9 (c). Nesunku pastebėti, kad didėjant parametro α vertei, duomenų išsidėstymo nuolydis mažėja.



4.11 pav. Apdorotų duomenų medianos logarítmo priklausomybės nuo blokų skaičiaus m logaritmo pavyzdžiai, kai parametras $\alpha = 1,1$ (a) , $\alpha = 1,5$ (b) ir $\alpha = 1,9$ (c)

Analogiškai kaip ir kituose metoduose, sudarius regresijos lygtis, buvo gautos Hurst indekso vertės (4.7 lentelė) ir jo įvertinimų standartiniai nuokrypiai (4.8 lentelė). Šio indekso verčių grafinės priklausomybės nuo $\frac{1}{\alpha}$ pateiktos 4.12 paveiksle.

4.7 lentelė

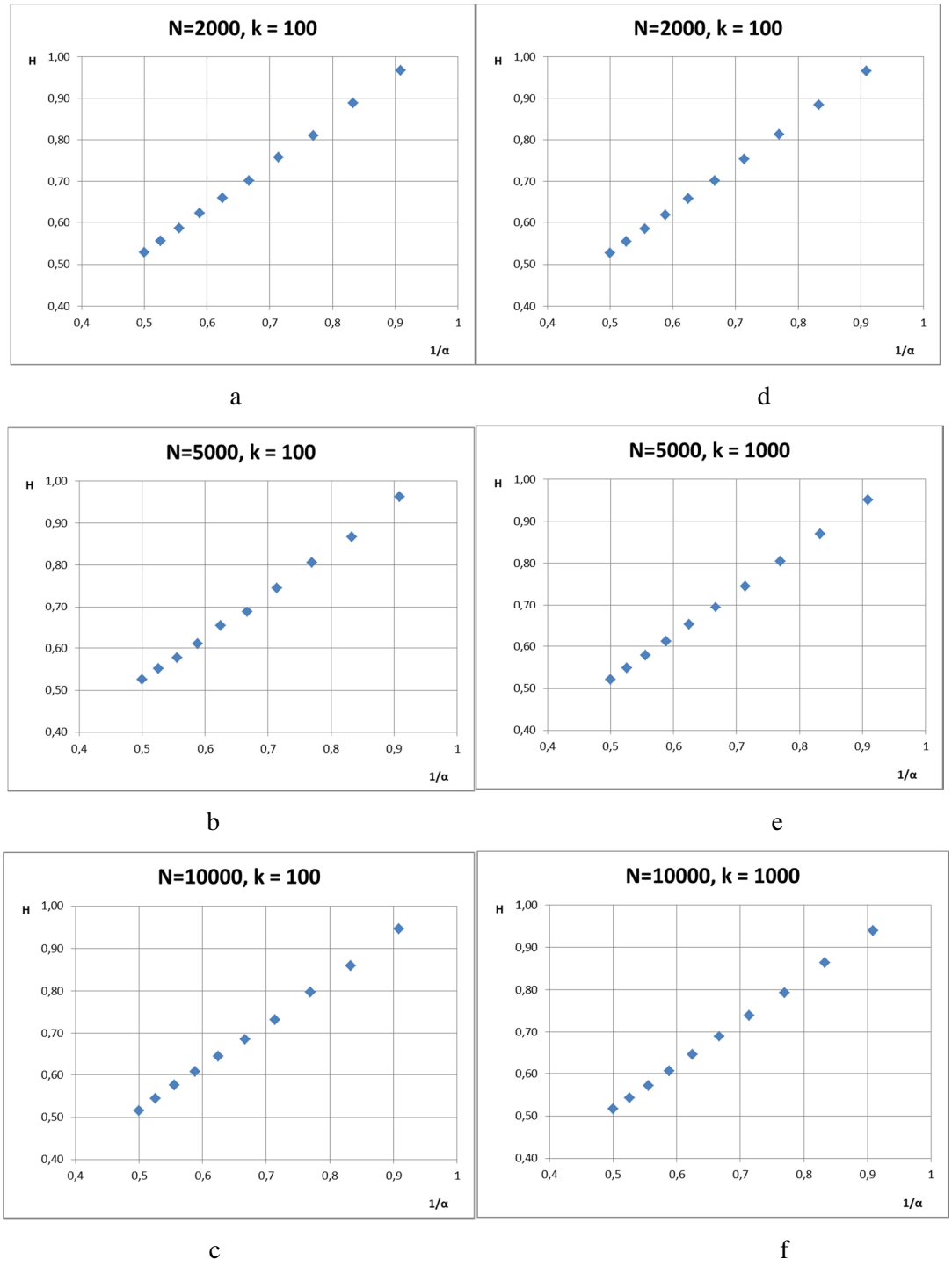
Hurst indekso vertės prie skirtingų parametrų *k, N* ir α (liekanų dispersijos metodas)

α	1/α	N=2000, k=100	N=5000, k=100	N=10000, k=100	N=2000, k=1000	N=5000, k=1000	N=10000, k=1000
1,1	0,909091	0,967442	0,962975	0,94696	0,964938	0,951391	0,938788
1,2	0,833333	0,888943	0,867377	0,859243	0,883978	0,869666	0,863995
1,3	0,769231	0,810274	0,806797	0,797118	0,813903	0,805307	0,79306
1,4	0,714286	0,758108	0,745574	0,732087	0,754486	0,745549	0,739078
1,5	0,666667	0,702222	0,687183	0,68495	0,7024	0,694295	0,688665
1,6	0,625	0,659065	0,654587	0,644298	0,657232	0,652041	0,64511
1,7	0,588235	0,622505	0,610662	0,607771	0,618679	0,612748	0,606978
1,8	0,555556	0,586178	0,577592	0,575883	0,584668	0,579008	0,572222
1,9	0,526316	0,555294	0,551282	0,54346	0,554686	0,5485	0,543326
2	0,5	0,527843	0,525646	0,51457	0,526974	0,520392	0,517111

4.8 lentelė

Hurst indekso verčių standartiniai nuokrypiai prie skirtingų parametrų *k, N* ir α (liekanų dispersijos metodas)

α	N=2000, k=100	N=5000, k=100	N=10000, k=100	N=2000, k=1000	N=5000, k=1000	N=10000, k=1000
1,1	0,094483	0,065422	0,051376	0,080251	0,060967	0,055394
1,2	0,069706	0,05006	0,044263	0,072938	0,055797	0,047174
1,3	0,056966	0,049145	0,039229	0,063198	0,046644	0,04064
1,4	0,059182	0,039888	0,033471	0,052707	0,04119	0,035329
1,5	0,044841	0,032795	0,030397	0,047592	0,036594	0,030834
1,6	0,039278	0,033735	0,025871	0,040172	0,031889	0,026825
1,7	0,035587	0,027749	0,023321	0,037359	0,02741	0,02337
1,8	0,030377	0,024241	0,021877	0,031533	0,023546	0,019558
1,9	0,027949	0,01872	0,018333	0,028472	0,020294	0,018019
2	0,025634	0,019248	0,016696	0,025094	0,017983	0,015426



4.12 pav. Liekanų dispersijos metodu gautos Hurst indekso vertės priklausomai nuo parametro $\frac{1}{\alpha}$, kai keičiamas duomenų skaičius *N* imtyje ir imčių skaičius *k* lygus 100 (a, b,c) ir 1000 (d,e,f)

Gautieji rezultatai rodo, kad liekanų dispersijos metodu randamas Hurst indeksas, didėjant α parametrui, mažėja, kas rodo, kad šiuo metodu α-stabilius savipanašius procesus su ilgomis

uodegomis galima atskirti nuo Gausinių procesų, kai α kitos nuo 1,1 iki 2, Hurst indeksas keitėsi nuo 0,93...0,96 iki 0,51...0,52. Tai rodo, kad šiuo metodu gautas Hurst indeksas $H \approx \frac{1}{\alpha}$. Tyrimo rezultatai taip pat rodo, kad kai duomenų kiekis $N > 2000$, tai nei tolesnis duomenų skaičiaus didinimas, nei imčių kiekio didinimas rezultatams įtakos beveik neturi. Hurst idenkso skaičiavimo standartiniai nuokrypiai mažėja, didėjant parametrui α (parametrui α padidėjus nuo 1,1 iki 2, sumažėja apie 3 kartus). Didėjant duomenų kiekiui N, jie taip pat mažėja.

Lyginant su kitais tirtais metodais, tiriant α-stabiliuosius procesus liekamosios dispersijos metodu, buvo gauti Hurst indekso įvertinimų standartiniai nuokrypiai, panašūs į absoliutinių momentų metodu gautus standartinius nuokrypius.

4.6 GAUTŲ REZULTATŲ ĮVERTINIMAS

Vertinant gautuosius rezultatus, pirmiausiai buvo ištirta, ar gautieji Hurst indekso rezultatai prie tų pačių parametrų yra pasiskirstę pagal normalųjį dėsnį. Tam buvo naudojamas Kolmogorovo Smirnovo testas. Buvo sudaromos hipotezės:

H_0: Gautieji Hurst indekso įverčiai, pasiskirstę pagal normalųjį skirstinį,

H_a: Duomenys nėra pasiskirstę pagal normalųjį skirstinį.

Tuomet buvo apskaičiuojamos Kolmogorovo Smirnovo statistikos D (2.12) prie skirtingų parametrų *k, N* ir α.

Gautųjų D statistikų reikšmės R/S metodu pateiktos 4.9 lentelėje, o kitų skaičiavimo metodų statistikos pateiktos 1 priede.

4.9 lentelė

Statistika D, su skirtingais parametrais k, N ir α (R/S metodas)

α	N=2000, k=100	N=5000, k=100	N=10000, k=100	N=2000, k=1000	N=5000, k=1000	N=10000, k=1000
1,1	0,06281	0,115131	0,09458	0,03532	0,037487	0,042256
1,2	0,063534	0,073901	0,090458	0,02209	0,031839	0,038339
1,3	0,056006	0,059408	0,093195	0,038481	0,044197	0,044917
1,4	0,053133	0,077401	0,050351	0,038198	0,032146	0,044772
1,5	0,061406	0,061923	0,067496	0,049197	0,05033	0,032567
1,6	0,077887	0,097137	0,072838	0,038995	0,033595	0,037691
1,7	0,06352	0,053164	0,067185	0,021678	0,039653	0,034189
1,8	0,058706	0,064993	0,07848	0,035856	0,02501	0,025072
1,9	0,070632	0,088729	0,046449	0,030272	0,020231	0,018511
2	0,042814	0,064765	0,064117	0,019616	0,025604	0,02503

Kai $k = 100$, tai hipotezė H_0 yra priimama, jei gautoji statistika $D < 0{,}163$, kai reikšmingumo lygmuo $\alpha = 0.01$, jeigu $D > 0{,}163$, tai Hipotezė H_0 atmetama. Kai $k = 1000$, tai hipotezė H_0 yra priimama, jei gautoji statistika $D < 0{,}05154$, kai reikšmingumo lygmuo $\alpha = 0.01$, jeigu $D > 0{,}05154$, tai Hipotezė H_0 atmetama. Kritinės statistikos D vertės yra paimtos iš lentelės (Table 7: Kolmogorov-Smirnov test).

Taigi, gauname, kad visais metodais gautieji Hurst įverčiai prie visų tirtų parametrų *k, N* ir α verčių yra pasiskirstę pagal normalųjį dėsnį.

Ištyrus, kad gautosios Hurst indeksų vertės yra pasiskirsčiusios pagal normalųjį skirstinį, buvo tikrinama ar gautosios Hurst indekso verčių vidurkiai statistiškai nesiskiria nuo teorinių verčių. Tam buvo naudotas t-testas. Tiriant t-testą buvo sudarytos tokios hipotezės:

H_0: Gautųjų Hurst indekso įverčių vidurkiai, statistiškai nesiskiria nuo $\frac{1}{\alpha}$,

H_a: Gautųjų Hurst indekso įverčių vidurkiai, statistiškai skiriasi nuo $\frac{1}{\alpha}$.

Naudojantis išraiška (2.13), buvo apskaičiuotos statistikos t. pilni skaičiavimų rezultatai pateikti 2 priede.

Kai $k = 100$, tai Hipotezė H_0 yra priimama, jei gautoji statistika $t \in (-2.626;\ 2.626)$, kai reikšmingumo lygmuo $\alpha = 0.01$, kitu atveju Hipotezė H_0 atmetama ir priimama alternatyvioji hipotezė H_a. Kai Kai $k = 1000$, tai Hipotezė H_0 yra priimama, jei gautoji statistika $t \in (-2.576;\ 2.576)$, kai reikšmingumo lygmuo $\alpha = 0.01$, kitu atveju Hipotezė H_0 yra atmetama ir priimama alternatyvioji hipotezė H_a. Kritinės t statistikos vertės paimtos iš lentelės (Table of Critical Values for T).

Lentelėse 4.10 ir 4.11 pateiktos tos t statistikos, su kuriomis hipotezė H_0 yra priimama (paryškinta storesniu šriftu).

Gautieji rezultatai parodė, kad tik R/S metodui ir absoliutinių momentų metodui, ir tik esant mažesniam imčių skaičiui ($k = 100$), kai kurios apskaičiuotos Hurst indekso vertės statistiškai nesiskiria nuo teorinių reikšmių (reikšmingumo lygmuo 0,01). R/S metodui, – kai stabilumo parametras $\alpha = 1{,}9$, ir absoliutinių momentų metodui, – kai $\alpha = 1{,}4 \ldots 2$. Tai rodo, kad prie didelių imčių ($k = 1000$) naudoti t-testą nėra visiškai korektiška. Taip yra todėl, kad duomenų yra labai daug, o standartiniai skaičiavimų nuokrypiai yra labai maži.

4.10 lentelė

T-testo rezultatai, su skirtingais parametrais k, N ir α (absoliutinių momentų metodas)

α	N=2000, k=100	N=5000, k=100	N=10000, k=100
1,1	-11,775685	-10,699930	-10,137430
1,2	-6,581626	-6,496279	-6,728577
1,3	-5,422864	-5,296846	**-2,547246**
1,4	**-2,528297**	-3,882608	-2,787708
1,5	**-1,736948**	**-0,672819**	-5,025898
1,6	**-1,780536**	**0,077851**	**-2,238235**
1,7	**-1,269660**	**-1,221163**	**-0,886633**
1,8	**-1,819136**	**-1,353023**	**0,419716**
1,9	**-2,233869**	**-2,129557**	**-0,531000**
2,0	**-1,468844**	**-0,792449**	**-2,551902**

4.11 lentelė

T-testo rezultatai, su skirtingais parametrais k, N ir α (R/S metodas)

α	N=2000, k=100	N=5000, k=100	N=10000, k=100	N=2000, k=1000
1,9	**-0,784551**	**-1,119959**	**-0,521623**	**-1,693102**

Todėl vertinant bendrą šių metodų tikslumą, buvo skaičiuojama vidutinė absoliutinė procentinė paklaida *M* (ang. *mean absolute percentage* error, *MAPE*):

$$M = \frac{100\%}{k}\sum_{t=1}^{k}\left|\frac{A_t - F_t}{A_t}\right|, \qquad \textbf{(4.1)}$$

kai k – imčių kiekis, A_t – teorinė Hurst indekso vertė ($\frac{1}{\alpha}$), F_t – gautosios Hurst indekso vertės. Gautosios vidutinės absoliutinės paklaidos pateiktos 4.12 lentelėje.

4.12 lentelė

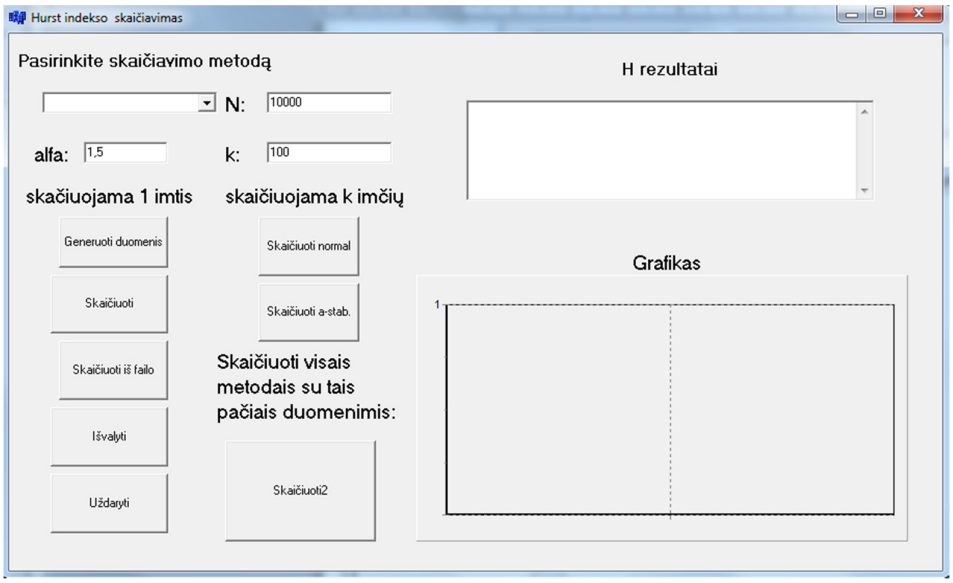
Metodų skaičiavimo rezultatų vidutinės absoliutinės procentinės paklaidos

α	N=2000, k=100	N=5000, k=100	N=10000, k=100	N=2000, k=1000	N=5000, k=1000	N=10000, k=1000
R/S	20,81261	20,78136	20,80729	20,82561	20,65538	20,88325
Absoliutinių momentų	3,374643	2,630373	2,563821	3,430451	2,944977	2,54659
Agreguotos dispersijos	23,0796	23,26101	22,85041	23,08233	23,15564	22,80282
Liekanų dispersijos	5,775857	4,473971	3,237387	5,53739	4,323251	3,287251

Apskaičiuota vidutinė absoliutinė procentinė paklaida parodė, jog Hurst indekso skaičiavimai absoliutinių momentų metodu ir liekanų dispersijos metodu yra kur kas artimesni teoriniams rezultatams, nei apskaičiuoti R/S metodu ir agreguotos dispersijos metodu.

5. PROGRAMINĖ ĮRANGA

Tyrimams atlikti, buvo parašyta programa C++ kalboje ir sukompiliuota su *Borland C++ builder 6*. Programos langas pateiktas 5.1 paveiksle

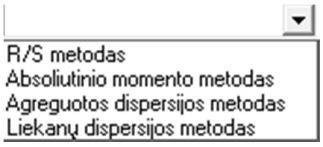


5.1 pav. Programinės įrangos langas

5.1 PROGRAMINĖS ĮRANGOS NAUDOJIMAS

Sukurtos programos langas (5.1 pav.) sudarytas iš tokių elementų.

Skaičiavimo metodo pasirinkimo langelis. Šiuo langeliu galima pasirinkti Hurst indekso skaičiavimo metodą (5.2 pav.).



5.2 pav. Skaičiavimo metodo pasirinkimo langelis

Parametrų įvesties langeliai. Šiuose langeliuose įvedami duomenų generavimo parametrai: ***alfa*** – stabilumo parametras α, ***k*** – imčių skaičius ir ***N*** – duomenų kiekis imtyje.

Rezultatų išvesties langelis *H rezultatai*. Šiame langelyje yra išvedami Hurst indekso skaičiavimo pasirinktu metodu rezultatai.

Duomenų generavimo mygtukas *Generuoti duomenis*. Jį aktyvavus, yra generuojami ***N*** ilgio α-stabilieji duomenys, kurių parametras α nurodytas parametrų įvesties langelyje ***alfa***. Sugeneruoti duomenys yra atvaizduojami grafiko langelyje ***Grafikas*.**

Hurst indekso skaičiavimo mygtukas *Skaičiuoti iš failo*. Aktyvavus šį mygtuką yra nuskaitomi duomenys iš failo „Data.txt“ ir apskaičiuojamas Hurst indeksas.

Hurst indekso skaičiavimo mygtukas *Skaičiuoti*. Aktyvavus šį mygtuką, yra suskaičiuojamas Hurst indeksas pasirinktu metodu. Skaičiavimo rezultatai išvedami rezultatų išvesties langelyje, o tarpiniai skaičiavimų rezultatai išvedami grafiko langelyje ***Grafikas*** logaritminiu masteliu.

Hurst indekso skaičiavimo mygtukas *Skaičiuoti normal*. Aktyvavus šį mygtuką yra sugeneruojama ***k*** imčių, kuriose yra ***N*** duomenų, pasiskirsčiusių pagal normalųjį pasiskirstymą, ir apskaičiuojama k Hurst indekso verčių pasirinktu metodu. Rezultatų faile „Rezultatai.txt“ yra pateikiamas šių verčių vidurkis bei standartinis nuokrypis.

Hurst indekso skaičiavimo mygtukas *Skaičiuoti a-stab*. Aktyvavus šį mygtuką yra sugeneruojama ***k*** imčių, kuriose yra ***N*** duomenų, pasiskirsčiusių pagal α-stabilųjį pasiskirstymą, ir apskaičiuojama *k* Hurst indekso verčių pasirinktu metodu. Šios vertės suvidurkinamos. Šie skaičiavimai atliekami keičiant parametrą α nuo 1.1 iki 2 žingsniu 0,1. Rezultatai išvedami į failą „Rezultatai.txt“.

Hurst indekso skaičiavimo mygtukas *Skaičiuoti2*. Aktyvavus šį mygtuką yra sugeneruojami α-stabilieji dydžiai su nurodytais parametras, ir su tais pačiais duomenimis apskaičiuojamas Hurst indeksas visais metodais.

Mygtukas *Išvalyti*. Aktyvavus šį mygtuką yra išvalomi rezultato išvedimo langai bei failai.

Mygtukas *Uždaryti*. Paspaudis šį mygtuką programa yra uždaroma.

IŠVADOS

1. Modeliuojant α-stabilius savipanašiuosius procesus, esant skirtingom stabilumo parametro α vertėms ($\alpha \in (1; 2]$) ir vertinant šiems procesams Hurst indeksą keturiais sekų analizės metodais, R/S statistikos metodu ir agreguotos dispersijos metodu, keičiantis parametrui α, mažai kito, kai tuo tarpu absoliutinių momentų metodu ir liekanų dispersijos metodu žymiai keitėsi. Tai reiškia, kad R/S ir agreguotos dispersijos metodais α-stabilių savipanašių procesų su ilgomis uodegomis negalima atskirti nuo Gausinių procesų.

2. Keičiantis α-stabilaus proceso stabilumo parametrui α nuo 1,1 iki 2, Hurst indeksas, įvertintas absoliutinių momentų metodu, kito nuo 0,83...0,84 iki 0,495...0,498, o įvertintas liekanų dispersijos metodu – nuo 0,93...0,96 iki 0,51...0,52. Tai rodo, kad gautos šiais metodais Hurst indekso vertės apytiksliai yra lygios $\frac{1}{\alpha}$.

3. Absoliutinių momentų metodu įvertintos H indekso vertės turėjo apie 1,3...2 kartus didesnį standartinį nuokrypį, nei įvertintos liekanų dispersijos metodu.

4. Tikrinant hipotezę apie apskaičiuotų Hurst indekso verčių atitikimą teoriniams rezultatams t-testu, gauta, kad tik R/S metodui ir absoliutinių momentų metodui ir tik esant mažesniam imčių skaičiui ($k = 100$) kai kurios apskaičiuotos Hurst indekso vertės statistiškai nesiskiria nuo teorinių reikšmių (reikšmingumo lygmuo 0,01). R/S metodui, – kai stabilumo parametras $\alpha = 1{,}9$, ir absoliutinių momentų metodui, – kai $\alpha = 1{,}4 \dots 2$. Tai rodo, kad prie didelių imčių ($k = 1000$) naudoti t-testą nėra visiškai korektiška.

5. Apskaičiuota vidutinė absoliutinė procentinė paklaida parodė, jog Hurst indekso skaičiavimai absoliutinių momentų metodu ir liekanų dispersijos metodu yra kur kas artimesni teoriniams rezultatams, nei apskaičiuoti R/S metodu ir agreguotos dispersijos metodu.

LITERATŪRA

1. Fox, R. and Taqqu, M.S. Large-sample properties of parameter estimates for strongly dependent stationary Gaussian time series/ *The Annals of Statistics,* 1986, 14. p. 517-532.
2. Kabašinskas, A. Finansinių rinkų statistinė analizė ir statistinio modeliavimo metodai. Daktaro disertacija. Vilnius, 2007.
3. Taqqu, M.S. and Teverovsky, V. Estimating long-range dependence in finite and infinite variance series. A Practical Guide to Heavy Tails: Statistical Techniques for Analyzing Heavy-Tailed Distributions. Boston, 1996.
4. Chronopulou, A. and Viens, F.G. Hurst Index Estimation for Self-similar processes with Long-Memory/ *World Scientific Review Volume,* 2009, p. 1-27.
5. Hurst, H. Long Term Storage Capacity of Reservoirs /*Transactions of the American Society of Civil Engineers,* 1951, p. 770-799.
6. Mandelbrot, B.B. Limit theorems of the self-normalized ranged for weakly and strongly dependent processes / *Z. Wahrscheinlichkeitstheorie verw. Gebiete, 31, 1975, p. 271-285*
7. Mandelbrot, B.B and Van Ness J. Fractional Brownian motion, fractional noises and aplications / *SIAM Review, 10,* 1968, p. 422-437
8. Engineering statistics handbook / Kolmogorov-Smirnov Goodness-of-Fit Test. http://www.itl.nist.gov/div898/handbook/eda/section3/eda35g.htm
9. http://www.ics.forth.gr/~tsakalid/THESIS/chapter3.pdf
10. Table 7: Kolmogorov-Smirnov test. http://www.eridlc.com/onlinetextbook/appendix/table7.htm
11. Table of Critical Values for T. http://www.jeremymiles.co.uk/misc/tables/t-test.html

1 priedas. Kolmogorovo Smirnovo testo rezultatai

Statistika D, su skirtingais parametrais k, N ir α (R/S metodas)

α	N=2000, k=100	N=5000, k=100	N=10000, k=100	N=2000, k=1000	N=5000, k=1000	N=10000, k=1000
1,1	0,06281	0,115131	0,09458	0,03532	0,037487	0,042256
1,2	0,063534	0,073901	0,090458	0,02209	0,031839	0,038339
1,3	0,056006	0,059408	0,093195	0,038481	0,044197	0,044917
1,4	0,053133	0,077401	0,050351	0,038198	0,032146	0,044772
1,5	0,061406	0,061923	0,067496	0,049197	0,05033	0,032567
1,6	0,077887	0,097137	0,072838	0,038995	0,033595	0,037691
1,7	0,06352	0,053164	0,067185	0,021678	0,039653	0,034189
1,8	0,058706	0,064993	0,07848	0,035856	0,02501	0,025072
1,9	0,070632	0,088729	0,046449	0,030272	0,020231	0,018511
2	0,042814	0,064765	0,064117	0,019616	0,025604	0,02503

Statistika D, su skirtingais parametrais k, N ir α (absoliutinių momentų metodas)

α	N=2000, k=100	N=5000, k=100	N=10000, k=100	N=2000, k=1000	N=5000, k=1000	N=10000, k=1000
1,1	0,06281	0,115131	0,09458	0,03532	0,037487	0,042256
1,2	0,063534	0,073901	0,090458	0,02209	0,031839	0,038339
1,3	0,056006	0,059408	0,093195	0,038481	0,044197	0,044917
1,4	0,053133	0,077401	0,050351	0,038198	0,032146	0,044772
1,5	0,061406	0,061923	0,067496	0,049197	0,05033	0,032567
1,6	0,077887	0,097137	0,072838	0,038995	0,033595	0,037691
1,7	0,06352	0,053164	0,067185	0,021678	0,039653	0,034189
1,8	0,058706	0,064993	0,07848	0,035856	0,02501	0,025072
1,9	0,070632	0,088729	0,046449	0,030272	0,020231	0,018511
2	0,042814	0,064765	0,064117	0,019616	0,025604	0,02503

Statistika D, su skirtingais parametrais k, N ir α (agreguotos dispersijos metodas)

α	N=2000, k=100	N=5000, k=100	N=10000, k=100	N=2000, k=1000	N=5000, k=1000	N=10000, k=1000
1,1	0,0628096	0,115131	0,0945801	0,0353203	0,0374866	0,0422555
1,2	0,0635337	0,073901	0,0904582	0,0220903	0,0318389	0,0383394
1,3	0,0560057	0,059408	0,0931952	0,0384805	0,0441974	0,0449169
1,4	0,0531327	0,077401	0,0503505	0,0381982	0,032146	0,0447716
1,5	0,0614061	0,061923	0,0674957	0,0491974	0,0503304	0,0325668
1,6	0,0778872	0,097137	0,0728375	0,0389946	0,0335952	0,0376908
1,7	0,0635198	0,053164	0,067185	0,0216775	0,0396528	0,0341891
1,8	0,058706	0,064993	0,07848	0,0358563	0,0250102	0,0250716
1,9	0,0706318	0,088729	0,0464491	0,0302718	0,0202307	0,0185107
2	0,0428142	0,064765	0,0641169	0,0196155	0,0256041	0,0250301

Statistika D, su skirtingais parametrais k, N ir α (liekanų dispersijos metodas)

α	N=2000, k=100	N=5000, k=100	N=10000, k=100	N=2000, k=1000	N=5000, k=1000	N=10000, k=1000
1,1	0,0628096	0,115131	0,0945801	0,0353203	0,0374866	0,0422555
1,2	0,0635337	0,0739005	0,0904582	0,0220903	0,0318389	0,0383394
1,3	0,0560057	0,059408	0,0931952	0,0384805	0,0441974	0,0449169
1,4	0,0531327	0,0774013	0,0503505	0,0381982	0,032146	0,0447716
1,5	0,0614061	0,0619229	0,0674957	0,0491974	0,0503304	0,0325668
1,6	0,0778872	0,0971371	0,0728375	0,0389946	0,0335952	0,0376908
1,7	0,0635198	0,0531642	0,067185	0,0216775	0,0396528	0,0341891
1,8	0,058706	0,0649931	0,07848	0,0358563	0,0250102	0,0250716
1,9	0,0706318	0,0887285	0,0464491	0,0302718	0,0202307	0,0185107
2	0,0428142	0,0647652	0,0641169	0,0196155	0,0256041	0,0250301

2 priedas. Gautieji t-testo rezultatai

Statistika t, su skirtingais parametrais k, N ir α (R/S metodas)

α	N=2000, k=100	N=5000, k=100	N=10000, k=100	N=2000, k=1000	N=5000, k=1000	N=10000, k=1000
1,1	-163,694638	-230,368274	-114,976768	-571,144877	-661,354032	-454,233672
1,2	-132,579286	-151,681875	-88,8798161	-429,925196	-507,264354	-339,757496
1,3	-112,914174	-121,027724	-68,1922584	-326,082073	-381,175391	-282,822414
1,4	-82,5925052	-90,6277256	-53,4108895	-243,837977	-276,951745	-214,313094
1,5	-55,3138592	-59,9289255	-36,1956572	-170,500751	-201,601228	-161,761205
1,6	-36,7661226	-48,4203293	-25,2862856	-121,008487	-137,364876	-111,637715
1,7	-25,756076	-31,8042707	-15,303824	-76,4888566	-84,3657971	-70,7850316
1,8	-11,4625456	-12,213549	-8,24320398	-34,2351608	-39,2981662	-33,5461209
1,9	-0,784551	-1,11995879	-0,52162341	-1,69310168	-2,55004524	-2,64203055
2	10,3144245	10,4339608	4,71393692	26,875697	32,5781436	25,9205825

Statistika t, su skirtingais parametrais k, N ir α (absoliutinių momentų metodas)

α	N=2000, k=100	N=5000, k=100	N=10000, k=100	N=2000, k=1000	N=5000, k=1000	N=10000, k=1000
1,1	-11,7756849	-10,6999298	-10,1374303	-39,4469952	-40,9942966	-38,3831172
1,2	-6,58162619	-6,49627912	-6,72857692	-22,0784935	-23,7204396	-21,495072
1,3	-5,42286446	-5,29684571	-2,54724568	-13,8470873	-14,9348194	-13,9419537
1,4	-2,52829657	-3,88260806	-2,78770769	-7,8757153	-10,2902388	-6,81350398
1,5	-1,73694834	-0,67281906	-5,02589837	-8,16999015	-6,9426179	-5,45512001
1,6	-1,7805361	0,07785075	-2,23823476	-6,75640246	-5,04179349	-5,24211434
1,7	-1,26966018	-1,22116289	-0,88663297	-5,64314364	-2,53270425	-4,42649152
1,8	-1,81913615	-1,35302321	0,41971623	-4,17757341	-4,14819573	-3,98157226
1,9	-2,23386895	-2,12955746	-0,5310003	-5,43611961	-4,6091406	-3,51421151
2	-1,46884352	-0,79244852	-2,55190194	-3,8859813	-5,24206535	-3,66660836

Statistika t, su skirtingais parametrais k, N ir α (agreguotos dispersijos metodas)

α	N=2000, k=100	N=5000, k=100	N=10000, k=100	N=2000, k=1000	N=5000, k=1000	N=10000, k=1000
1,1	-141,439501	-209,706333	-333,639723	-427,441773	-661,149711	-1091,27349
1,2	-100,157584	-232,729124	-257,039803	-438,92388	-632,116102	-650,751919
1,3	-112,565012	-154,749142	-131,136534	-251,242447	-457,537528	-751,947199
1,4	-97,4305196	-124,122326	-164,286915	-292,653465	-347,988324	-463,702207
1,5	-59,7652129	-84,7256131	-112,294339	-183,344455	-262,859173	-375,58363
1,6	-56,7208411	-58,542917	-100,035804	-148,202951	-186,72461	-278,688637
1,7	-33,1753418	-50,2474629	-57,6827696	-107,916851	-136,370758	-188,958762
1,8	-20,857015	-32,2930845	-35,6606762	-71,6457766	-88,9270117	-120,708904
1,9	-10,5796764	-13,9716554	-23,3639527	-33,5487279	-42,7829818	-56,477417
2	-2,08563175	-1,51064638	-2,52070348	-3,65755867	-6,83544882	-3,29709302

Statistika t, su skirtingais parametrais k, N ir α (liekanų dispersijos metodas)

α	N=2000, k=100	N=5000, k=100	N=10000, k=100	N=2000, k=1000	N=5000, k=1000	N=10000, k=1000
1,1	6,17585576	8,23638698	7,37099781	22,0063461	21,9405337	16,9531803
1,2	7,97780185	6,80062699	5,85355898	21,9573772	20,5912785	20,5536697
1,3	7,20483914	7,64394228	7,10890288	22,3528514	24,458197	18,5422173
1,4	7,40462701	7,84407406	5,31850009	24,1189485	24,0019204	22,1915523
1,5	7,9292017	6,25593332	6,01486774	23,7433175	23,875274	22,560935
1,6	8,67290433	8,77049448	7,45940388	25,372342	26,8152498	23,7067675
1,7	9,62994663	8,08198706	8,37698091	25,7692104	28,2805959	25,3612036
1,8	10,0809322	9,0904924	9,2918664	29,1953296	31,4972633	26,9475023
1,9	10,3683201	13,3370071	9,35145557	31,5099442	34,567824	29,8525486
2	10,8615766	13,3242586	8,72648431	33,9921724	35,8597567	35,0781064

3 priedas. Programų kodas hurst indeksui apskaičiuoti

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

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{

}
//---------------------------------------------------------------------------
double TForm1::maxmax(double*x,int n){
double ma=-100000000;
int i;
for (i=0;i<n;i++)
      //            if (ma<fabs(x[i])) ma=fabs(x[i]);
        if (ma<x[i]) ma=x[i];
return ma;
}
//---------------------------------------------------------------------------
double TForm1::minmin(double*x,int n){
double minn=1000000000000000.000;
int i;
for (i=0;i<n;i++)
                  if (x[i]<minn) minn=x[i];

return minn;

}
//---------------------------------------------------------------------------
void TForm1::isvesti(char* textas,int n, double *x){

ofstream myfile;
myfile.open (textas);
for (int i=0;i<n;i++){
myfile << x[i] << endl;
}
myfile.close();
}
//---------------------------------------------------------------------------

double TForm1::paklaidos(double*x5,double m,int nn1){
int i;
double s=0;
```

```
		for(i=0;i<nn1;i++)
		s+=fabs(x5[i]-m);

return s/nn1;
}
//---------------------------------------------------------------------------

double TForm1::disp(double*x5,double m,int nn1){
int i;
double s=0;
		for(i=0;i<nn1;i++)
		s+=(x5[i]-m)*(x5[i]-m);

return s/nn1;
}
//---------------------------------------------------------------------------

double TForm1::vid(double *x5,int nn1){
double s=0.0;
int i;
		for(i=0;i<nn1;i++)
		{
			s+=x5[i];
		}
return (s/nn1);
}
//---------------------------------------------------------------------------
double *TForm1::agregavimas(double *X, int nn,int k){
		double *z=new double[nn];
		int i;
		for (i=0;i<nn;i++){
			z[i]=X[k*nn+i];
		}
return z;
}
//---------------------------------------------------------------------------
double TForm1::kovarijacia(double *x5, double *x6, int nn1){
		double sum=0;
		double vid5=vid(x5,nn1);
		double vid6=vid(x6,nn1);
		int i;
		for ( i=0;i<nn1;i++){
			sum+=(x5[i]-vid5)*(x6[i]+vid6);
		}

return sum/nn1;
}
//---------------------------------------------------------------------------
double *TForm1::zsuma(double *X, int m){
		double *z=new double [m];
		int i=0, k=0;
		for (i=0;i<m;i++){
```

```
                                z[i]=0;
                                for (k=0;k<=i;k++){
                                        z[i]+=X[k];
                                }
                }
return z;
}
//---------------------------------------------------------------------------
double *TForm1::keisti(double *x1,int nn){
double *x3=new double[nn];
int i;

for (i=0;i<nn;i++)
                x3[i]=x1[i];

return x3;
}
//---------------------------------------------------------------------------
double TForm1::gaus(){

double GaussNum = 0.0;
int NumInSum = 10;
for(int i = 0; i < NumInSum; i++)
{
GaussNum += ((double)rand()/(double)RAND_MAX - 0.5);
}
GaussNum = GaussNum*sqrt((double)12/(double)NumInSum);
return GaussNum;
}
//---------------------------------------------------------------------------
double TForm1::gaussp(double y){


                double x,a,r,p,fi;
  int i;

 x = y*y;
 p=exp(-x*0.5);
 if (x > 12.25) {

    r = (x*x*x+6.0*x*x-14.0*x-28.0)/((x+2)*(x*x*x+5.0*x*x-20.0*x-4.0));
    if (y > 0.0)
                                fi = 1.0-0.3989422804*y*p*r;
                 else fi = -0.3989422804*y*p*r;

  }
  else
                {
                r = 1.0;
                a = 1.0;
                i = 1;
                  while (a >= 1.e-6) {
```

```
                    i = i+2;
                    a = a*x/i;
                    r = r+a;
                   }
                fi = 0.5+0.3989422804*r*y*p;
                }

if (fi<1e-307)
                fi=1e-20;
if (fi>=1)
                fi=0.9999999999999999;


return fi;
 }
//---------------------------------------------------------------------------
double *TForm1::bubbleSort ( double *arr, int size )
{
  int last = size - 2;
  int isChanged = 1;

  while ( last >= 0 && isChanged )
  {
        isChanged = 0;
        for (int  k = 0; k <= last; k++ )
           if ( arr[k] > arr[k+1] )
           {
              swap ( arr[k], arr[k+1] ) ;
              isChanged = 1;

           }
        last--;
  }
  return arr;
}

//---------------------------------------------------------------
double TForm1::med(double *X,int nn1){
                double mediana=0;
                double *x;
                int xm=0;
                x=keisti(X,nn1);
                x=bubbleSort(x,nn1);
if (nn1==2){
                                        mediana=(X[0]+X[1])/2;
                }
                else {
                        if (nn1 %2){
                                        mediana=x[nn1/2];
                        }
                        else
```

```
            mediana=(x[(nn1/2-1)]+x[nn1/2])/2;
                }
                free(x);

return mediana;
}
//---------------------------------------------------------------------------
double *TForm1::Xas(int n){
       double *Z=new double [n];
      for (int i=0;i<n;i++)
          Z[i] = gaus();
          return Z;
          free (Z);
          delete (Z);
}
//---------------------------------------------------------------------------
double *TForm1::generavimas(double *x,const int r)
{
double *y=new double[r];
int mk=0;
double a,b,m,s,c,d,tre,v,w,*temp,t11,t21,t31,t41,a1;
int i;
a=x[0];
b=x[1];
m=x[2];
s=x[3];
     MTRand mtrand1;

  c = atan(b * tan(pi * a / 2)) / a;
  d = pow(1+pow(b * tan(pi * a / 2),2) , 1 / (2*a));

for (i = 0;i< r;i++)
     {
          start1:
     v = (mtrand1()* pi) - (pi / 2);
          a1=mtrand1();
     w = -log(a1);

          if (w<=0.0000000000000000001){
               i--;
               //delete temp;
               mk++;
               goto start1;
          }
          t11=d * sin(a * (v + c));
          t21=pow(fabs(cos(v)) , (1 / a));
          t31=cos(v - a * (v + c) )/ w ;
          t41=pow(fabs(t31), (1 - a) / a);
     tre = t41*t11 / ((t21!=0) ? t21: 1);
          temp=new double;

          *temp= tre * s + m;
```

```
            y[i]=*temp;

            delete temp;
      }
temp=NULL;
return y;
free (y);
}

//---------------Absoliutiniu momentu metodas---------------------------------
double TForm1::mom_X(double *X, int m, int k){
                  double s=0;
                  int i=0;
                  for (i=k*m;i<(k+1)*m;i++){
               s+=X[i];
                  }
return s/m;
}

//----------------------------------------------------------------------------
double *TForm1::mom_agreg(double *X, int m, int K){
                  int k;
                  double z=0;
                  double *Y=new double[K];
                  for (k=0;k<K;k++){
                                    Y[k]=mom_X(X,m,k);
                  }
return Y;
free (Y);
}
//----------------------------------------------------------------------------
double TForm1::mom_agr(double *X, int n, int m,double M){

                  int K=static_cast<int>(floor((double)n/m));
                  int i;
                  double z=0;
                  double *Z;
                  Z=mom_agreg(X,m,K);
                  for (i=0;i<K;i++){
                                    z+=fabs(Z[i]-M);
                  }
free(Z);
return z/K;
}
//----------------------------------------------------------------------------
double TForm1::hurst_absmom(double *X,int n, int m){
                  double H;
                  double M=vid(X,n);
                  int i=0;


      int hh=static_cast<int>((double)floor((double)n/m))+2;
```

```
		double *Z1=new double [hh];
		double *Z2=new double [hh];


		while (m<=0.1*n)
		{
				Z1[i]=log((double)m);
				Z2[i]=log(mom_agr(X,n,m,M));

				i++;
				m=static_cast<int>(floor(exp((double)i-1)/(double)i))+m;
		}
int k=i-1;
		double vidX=vid(Z1,k);
		H=kovarijacia(Z1, Z2, k)/disp(Z1,vidX,k)+1;
free(Z1);
free(Z2);
return H;
}
//--------------------Dispersijos-----------------------
double TForm1::var_agr(double *X, int n, int m,double M){

		int K=static_cast<int>((double)floor((double)n/m));
		int i;
		double z=0;
		double *Z;
		Z=mom_agreg(X,m,K);
		for (i=0;i<K;i++){
	//z+=fabs(Z[i]-M);
				z+=pow(Z[i]-M,2.0);
		}
free(Z);
return z/K;;
}
//---------------------------------------------------------------------------
double TForm1::hurst_var(double *X,int n, int m){
		double H;
		double M=vid(X,n);
		int i=0;


	int hh=static_cast<int>((double)floor((double)n/m))+2;
		double *Z1=new double [hh];
		double *Z2=new double [hh];


		while (m<=0.05*n)
		{

				Z1[i]=log((double)m);
				Z2[i]=log(var_agr(X,n,m,M));
```

```
                        i++;
                        m=static_cast<int>(floor(exp((double)i-1)/(double)i))+m;
                }
int k=i-1;
                double vidX=vid(Z1,k);
                H=0.5*kovarijacia(Z1, Z2, k)/disp(Z1,vidX,k)+1;
free(Z1);
free(Z2);
return H;
}
//-------------------------RS-----------------------------------------------
double TForm1::RS(double *X,int n){
double rs=0;
double Yn=vid(X,n);
double *z=new double [n];
z[0]=0;
for (int i=1;i<n;i++){
                z[i]=z[i-1]+X[i-1]-Yn;
}
rs=(maxmax(z,n)-minmin(z,n))/sqrt(disp(X,Yn,n));
//free(z);
delete [] z;
return rs;
}
//---------------------------------------------------------------------------
double TForm1::agr_RS(double *X,int n,int K){
                int i;
                double rs=0;
                int nn=static_cast<int>((double)floor((double)n/K));
                double *Y1;
                for (i=0;i<K;i++){
                        Y1=agregavimas(X,nn,i);
                        rs+=RS(Y1,nn);
                        free(Y1);
                }
return rs/K;
}
//---------------------------------------------------------------------------
double TForm1::hurst_RS(double *X,int n){
        Series1->Clear();
                int i=n,K=1,j=0;
                while (i>8){
                        i=static_cast<int>((double)floor((double)n/K));
                        K=2*K;
                        j++;
                }
                double *Z1=new double [j+1];
                double *Z2=new double [j+1];
                double H;

                Z1[0]=log((double)n);
                Z2[0]=log(RS(X,n));
```

```
		int m=0,k=0;
		K=2;
		m=static_cast<int>((double)floor((double)n/K));
		i=1;
	while (m>8)
		{

				Z1[i]=log((double)m);
				Z2[i]=log(agr_RS(X,n,K));


				i++;
				K=2*K;
				m=static_cast<int>((double)floor((double)n/K));
		}

		k=i-1;
		double vidX=vid(Z1,k);
		H=kovarijacia(Z1, Z2, k)/disp(Z1,vidX,k);
free(Z1);
free(Z2);
return H;

}

//---------------Var. res.---------------------------------
double TForm1::varrez(double *X, double *Y, int n){
		double vidX=vid(X,n);
		double vidY=vid(Y,n);
		double b=kovarijacia(X, Y, n)/disp(X,vidX,n);
		double a=vidY-vidX*b;
		double s=0;
		int i=0;
		for(i=0;i<n;i++){
				s+=pow(Y[i]-a-b*(i+1),2.0);//

		}
return s/n;
}
//-------------------------------------------------
double TForm1::varrez_agr(double *X, int n, int m){
		double r=0;
		int nn=static_cast<int>(floor((double)n/m));
		int i;
		double *Y;//=new double[m];
		double *Y1;
		double *Z=new double[nn];
		double *T=new double[m];

		for (i=0;i<m;i++) T[i]=i+1;//
```

```
                for (i=0;i<nn;i++){
                                Y1=agregavimas(X,m,i);
                                Y=zsuma(Y1,m);
                                free(Y1);
                                Z[i]=varrez(T,Y,m);
                                free(Y);
                }
                r=med(Z,nn);
                free(Z);
                free(T);

return r;
}

//---------------------------------------------------------------
double TForm1::hurst_varres(double *X,int n, int K){
                double H;
                int i=0;
                int m=5;
                int m0=static_cast<int>(floor((double)n/K));
                int hh=static_cast<int>(floor((double)0.5*K))+2;
                double *Z1=new double [hh];
                double *Z2=new double [hh];
                int h=static_cast<int>(floor(0.5*n));
                while (m<=h)
                {
                                Z1[i]=log((double)m);
                                Z2[i]=log(varrez_agr(X,n,m));
            i++;
                        //          m=m0*(++i);
          m=static_cast<int>(floor(exp((double)i-1)/(double)i))+m;
                }
int k=i-1;
                double vidX=vid(Z1,k);
                H=0.5*kovarijacia(Z1, Z2, k)/disp(Z1,vidX,k);
free(Z1);
free(Z2);
return H;
}
//-----------------------Mygtukai-----------------------------------------
void __fastcall TForm1::Button4Click(TObject *Sender)
{
double nn = StrToFloat(Edit2->Text);
srand(time(0));
double a = Edit1->Text.ToDouble();
double *Z=new double [4];
      Z[0]=a;
      Z[1]=0.0;
      Z[2]=0.0;
      Z[3]=1.0;
```

```
            X = generavimas(Z,nn);
          // X=Xas(nn);
Series1->Clear();
for(int i=0;i<nn;i++)
Series1->AddXY(i, X[i]);
int g = 0;
isvesti("Duomenys.txt",nn, X);

}

//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
        Series1->Clear();

        double m = 2;
double l = 50;
double n = StrToFloat(Edit2->Text);


if (ComboBox1->ItemIndex==0)
{ double RS1;
RS1 = hurst_RS(X,n);
RS_graf(X, n);

Memo1->Lines->Add("RS:");
Memo1->Lines->Add(RS1);

}
if (ComboBox1->ItemIndex==1)
{
double absmom;
absmom = hurst_absmom(X,n, m);
absmom_graf(X, n,m, absmom);
Memo1->Lines->Add("Absoliutinio momento metodas:");
Memo1->Lines->Add(absmom);
}
if (ComboBox1->ItemIndex==2)
{
double hvar;
hvar = hurst_var(X, n,  m);

Memo1->Lines->Add("Agreguotos dispersijos metodas:");
Memo1->Lines->Add(hvar);
var_graf(X, n, m, hvar);
}
if (ComboBox1->ItemIndex==3)
{
double varrez;
varrez = hurst_varres(X, n, l);
varres_graf(X, n, l);
```

```
Memo1->Lines->Add("Var rez:");
Memo1->Lines->Add(varrez);
}

}

//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
Memo1->Lines->Clear();
Series1->Clear();


ofstream myfile;
myfile.open ("RS.txt");
myfile.close();
myfile.open ("var.txt");
myfile.close();
myfile.open ("mom.txt");
myfile.close();
myfile.open ("RSstab.txt");
myfile.close();
myfile.open ("varstab.txt");
myfile.close();
myfile.open ("momstab.txt");
myfile.close();
myfile.open ("varrezstab.txt");
myfile.close();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button3Click(TObject *Sender)
{
Close();
}
//---------------------------------------------------------------------------
void TForm1::RS_graf(double *X,int n){
	Series1->Clear();
		int i=n,K=1,j=0;
		while (i>8){
				i=static_cast<int>((double)floor((double)n/K));
				K=2*K;
				j++;
		}
		double *Z1=new double [j+1];
		double *Z2=new double [j+1];
		double H;

		Z1[0]=log((double)n);
		Z2[0]=log(RS(X,n));
```

```
                int m=0,k=0;
                K=2;
                m=static_cast<int>((double)floor((double)n/K));
                i=1;
        while (m>8)
                {

                                Z1[i]=log((double)m);
                                Z2[i]=log(agr_RS(X,n,K));

                                i++;
                                K=2*K;
                                m=static_cast<int>((double)floor((double)n/K));
                }
        int p=i-1;

          for(int i=0;i<p;i++)   {
          Series1->AddXY(Z1[i], Z2[i]);
          }


free(Z1);
free(Z2);


}
//---------------------------------------------------------------------------
void TForm1::varres_graf(double *X,int n, int K){
                double H;
                int i=0;
                int m=5;
                int m0=static_cast<int>(floor((double)n/K));
                int hh=static_cast<int>(floor((double)0.5*K))+2;
                double *Z1=new double [hh];
                double *Z2=new double [hh];
                int h=static_cast<int>(floor(0.5*n));
                while (m<=h)
                {
                                Z1[i]=log((double)m);
                                Z2[i]=log(varrez_agr(X,n,m));

            Series1->AddXY(Z1[i], Z2[i]);
            i++;
             m=static_cast<int>(floor(exp((double)i-1)/(double)i))+m;
                                //m=m0*(++i);

                }

        }
//---------------------------------------------------------------------------
void TForm1::var_graf(double *X,int n, int m, double H){
```

```
                double M=vid(X,n);
                int i=0;
       //double m = 3;
      int hh=static_cast<int>((double)floor((double)n/m))+2;

                double *Z1=new double [hh];
       double *Z2=new double [hh];


       while (m<=0.05*n)
                {
                                Z1[i]=log((double)m);
                                Z2[i]=log(var_agr(X,n,m,M));
            Series1->AddXY(Z1[i], Z2[i]);
             i++;
                                m=static_cast<int>(floor(exp((double)i-1)/(double)i))+m;
                }

}
//---------------------------------------------------------------------------
void TForm1::absmom_graf(double *X,int n, int m, double H){

                double M=vid(X,n);
                int i=0;

                int hh=static_cast<int>((double)floor((double)n/m))+2;

                double *Z1=new double [hh];
                double *Z2=new double [hh];



                while (m<=0.05*n)
                {

                                Z1[i]=log((double)m);
                                Z2[i]=log(mom_agr(X,n,m,M));
             Series1->AddXY(Z1[i], Z2[i]);
                                i++;
                                m=static_cast<int>(floor(exp((double)i-1)/(double)i))+m;
                }

}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button5Click(TObject *Sender)
{

int v = StrToFloat(Edit1->Text);
double m = 2;
int nn = StrToFloat(Edit2->Text);
int k = StrToFloat(Edit3->Text);
double absmomd;
```

```
double hvard;
double RS1d;
double varrezd;
double absmomvid=0;
double hvarvid=0;
double RS1vid=0;
double varrezvid=0;
double *RS1=new double [k];
double *absmom=new double [k];
double *hvar=new double [k];
double *varrez=new double [k];


 ofstream myfile;
  myfile.open ("Matricastab.txt", ios::out);




      for (int j=0;j<k;j++){
      X = Xas(nn);
      isvesti("Duomenys.txt",nn, X);
            if (ComboBox1->ItemIndex==0)
            RS1[j] = hurst_RS(X,nn);
            if (ComboBox1->ItemIndex==1)
            absmom[j] = hurst_absmom(X,nn, m);
            if (ComboBox1->ItemIndex==2)
            hvar[j] = hurst_var(X, nn,  m);
            if (ComboBox1->ItemIndex==3)
            varrez[j]= hurst_varres(X, nn, v);
            free(X);

        }
if (ComboBox1->ItemIndex==1){
absmomvid=vid(absmom,k);
absmomd=pow(disp(absmom,absmomvid,k),0.5);
}
if (ComboBox1->ItemIndex==2){
hvarvid=vid(hvar,k);
hvard=pow(disp(hvar,hvarvid,k),0.5);
}
if (ComboBox1->ItemIndex==3){
varrezvid=vid(varrez,k);
varrezd=pow(disp(varrez,varrezvid,k),0.5);
}
if (ComboBox1->ItemIndex==0){
RS1vid=vid(RS1,k);
RS1d=pow(disp(RS1,RS1vid,k),0.5);
}

myfile << "Narmalusis skirstinys" << endl;
myfile << "Vidurkiai" << endl;
```

```
if (ComboBox1->ItemIndex==0)
myfile <<RS1vid<< endl;
if (ComboBox1->ItemIndex==1)
myfile <<absmomvid<< endl;
if (ComboBox1->ItemIndex==2)
myfile <<hvarvid<< endl;
if (ComboBox1->ItemIndex==3)
myfile <<varrezvid<< endl;
myfile << "Standartiniai nuokrypiai" << endl;

if (ComboBox1->ItemIndex==0)
myfile <<RS1d<< endl;
if (ComboBox1->ItemIndex==1)
myfile <<absmomd<< endl;
if (ComboBox1->ItemIndex==2)
myfile <<hvard<< endl;
if (ComboBox1->ItemIndex==3)
myfile << varrezd<< endl;
myfile << endl;
myfile.close();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button6Click(TObject *Sender)
{

double m = 2;
int nn = StrToFloat(Edit2->Text);
int k = StrToFloat(Edit3->Text);
double absmomd;
double hvard;
double RS1d;
double varrezd;
double absmomvid=0;
double hvarvid=0;
double RS1vid=0;
double varrezvid=0;
double *RS1=new double [k];
double *absmom=new double [k];
double *hvar=new double [k];
double *varrez=new double [k];
double K = 500;

 ofstream myfile;
  myfile.open ("Rezultatai.txt", ios::out);

for (double i=1.1;i<2.1;i=i+0.1){

        double *Z=new double [4];
        Z[0]=i;
        Z[1]=0.0;
        Z[2]=0.0;
```

```
        Z[3]=1.0;

       for (int j=0;j<k;j++){
              X = generavimas(Z,nn);
              if (ComboBox1->ItemIndex==0)
              RS1[j] = hurst_RS(X,nn);
              if (ComboBox1->ItemIndex==1)
              absmom[j] = hurst_absmom(X,nn, m);
              if (ComboBox1->ItemIndex==2)
              hvar[j] = hurst_var(X, nn,  m);
              if (ComboBox1->ItemIndex==3)
              varrez[j]= hurst_varres(X, nn, K);


         }
if (ComboBox1->ItemIndex==1){
absmomvid=vid(absmom,k);
absmomd=pow(disp(absmom,absmomvid,k),0.5);
}
if (ComboBox1->ItemIndex==2){
hvarvid=vid(hvar,k);
hvard=pow(disp(hvar,hvarvid,k),0.5);
}
if (ComboBox1->ItemIndex==3){
varrezvid=vid(varrez,k);
varrezd=pow(disp(varrez,varrezvid,k),0.5);
}
if (ComboBox1->ItemIndex==0){
RS1vid=vid(RS1,k);
RS1d=pow(disp(RS1,RS1vid,k),0.5);
}

myfile << "alfa=" << Z[0] << endl;
myfile << "Vidurkiai" << endl;
if (ComboBox1->ItemIndex==0)
myfile <<RS1vid<< endl;
if (ComboBox1->ItemIndex==1)
myfile <<absmomvid<< endl;
if (ComboBox1->ItemIndex==2)
myfile <<hvarvid<< endl;
if (ComboBox1->ItemIndex==3)
myfile <<varrezvid<< endl;
myfile << "Standartiniai nuokrypiai" << endl;

if (ComboBox1->ItemIndex==0)
myfile <<RS1d<< endl;
if (ComboBox1->ItemIndex==1)
myfile <<absmomd<< endl;
if (ComboBox1->ItemIndex==2)
myfile <<hvard<< endl;
if (ComboBox1->ItemIndex==3)
myfile << varrezd<< endl;
```

```
myfile << endl;

        if (ComboBox1->ItemIndex==0)
        isvesti("RSstab.txt",k, RS1);
        if (ComboBox1->ItemIndex==1)
        isvesti("momstab.txt",k, absmom);
        if (ComboBox1->ItemIndex==2)
       isvesti("varstab.txt",k, hvar);
       if (ComboBox1->ItemIndex==3)
        isvesti("varrezstab.txt",k, varrez);



free(X);



        }

myfile.close();
}
//---------------------------------------------------------------------------
double TForm1::ztest(double disp, double vid, double u, int n){
double se;
double z;
se = sqrt(disp/n);
z = (vid-u)/se;
return z;

}
//---------------------------------------------------------------------------
double *TForm1::ivesti(){


double *Z=new double [eilutesfaile()];
int i = 0;
                double d=0;
       failoilgis =0;
        string ss;
                ifstream file("Data.txt");


                while(!file.eof())
                {
         //file << d;

         //double g = atof(s);

         getline ( file, ss );
        std::stringstream s(ss);
```

```
        s >>  d;

        Z[i] = d;

        i++;

    }

                file.close();
       return Z;
       }
//---------------------------------------------------------------------------
double TForm1::eilutesfaile(){
        double numLines = 0;
        double r;
        string s;
                ifstream file("Data.txt");

                while(!file.eof())
                {
         getline ( file, s );
         numLines++;
         }

  file.close();

   return numLines;
  }
  //---------------------------------------------------------------------------
void __fastcall TForm1::Button7Click(TObject *Sender)
{
double n = eilutesfaile();
free (X);
double *Z=new double [n];
X = ivesti();

        Series1->Clear();
        double m = 2;


double l = 50;


Memo1->Lines->Add(n);
if (ComboBox1->ItemIndex==0)
{ double RS1;
RS1 = hurst_RS(X,n);
RS_graf(X, n);

Memo1->Lines->Add("RS:");
Memo1->Lines->Add(RS1);
}
```

```
if (ComboBox1->ItemIndex==1)
{
double absmom;
absmom = hurst_absmom(X,n, m);
absmom_graf(X, n,m, absmom);
Memo1->Lines->Add("Absoliutinio momento metodas:");
Memo1->Lines->Add(absmom);
}
if (ComboBox1->ItemIndex==2)
{
double hvar;
hvar = hurst_var(X, n,  m);

Memo1->Lines->Add("Agreguotos dispersijos metodas:");
Memo1->Lines->Add(hvar);
var_graf(X, n, m, hvar);
}
if (ComboBox1->ItemIndex==3)
{
double varrez;
varrez = hurst_varres(X, n, l);
varres_graf(X, n, l);

Memo1->Lines->Add("Var rez:");
Memo1->Lines->Add(varrez);
}

}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button8Click(TObject *Sender)
{


char *textas1 =   "10k1000RS.txt";
char *textas2 =   "10k1000absmom.txt";
char *textas3 =   "10k1000disp.txt";
char *textas4 =   "10k1000liekdisp.txt";

ofstream myfile;
myfile.open (textas1);myfile.close();
myfile.open (textas2);myfile.close();
myfile.open (textas3);myfile.close();
myfile.open (textas4);myfile.close();

double m = 2;
int nn = StrToFloat(Edit2->Text);
int k = StrToFloat(Edit3->Text);
double absmomd;
double hvard;
double RS1d;
double varrezd;
double absmomvid=0;
```

```
double hvarvid=0;
double RS1vid=0;
double varrezvid=0;
double *RS1=new double [k];
double *absmom=new double [k];
double *hvar=new double [k];
double *varrez=new double [k];
double K = 500;


for (double i=1.1;i<2.1;i=i+0.1){

        double *Z=new double [4];
        Z[0]=i;
        Z[1]=0.0;
        Z[2]=0.0;
        Z[3]=1.0;

       for (int j=0;j<k;j++){
              X = generavimas(Z,nn);
              RS1[j] = hurst_RS(X,nn);

              absmom[j] = hurst_absmom(X,nn, m);

              hvar[j] = hurst_var(X, nn,  m);
              varrez[j]= hurst_varres(X, nn, K);


         }




RS1vid=vid(RS1,k);
double RS1disp=disp(RS1,RS1vid,k);
RS1d=sqrt(RS1disp);
double Dr1 = normalityt (k, RS1, RS1vid, RS1d);
double z1 =  ztest(RS1disp,  RS1vid, 1/i, k);


absmomvid=vid(absmom,k);
double absmomdisp= disp(absmom,absmomvid,k);
absmomd=sqrt(absmomdisp);
double Dr2 = normalityt (k, absmom, absmomvid, absmomd);
double z2 =  ztest(absmomdisp,  absmomvid, 1/i, k);

hvarvid=vid(hvar,k);
double hvardisp= disp(hvar,hvarvid,k);
hvard=sqrt(hvardisp);
double Dr3 = normalityt (k, hvar, hvarvid, hvard);
double z3 =  ztest(hvardisp,  hvarvid, 1/i, k);
```

```
varrezvid=vid(varrez,k);
double varrezdisp= disp(varrez,varrezvid,k);
varrezd=sqrt(varrezdisp);
double Dr4 = normalityt (k, varrez, varrezvid, varrezd);
double z4 =  ztest(varrezdisp,  varrezvid, 1/i, k);


isvesti2(textas1, RS1vid, RS1d, i, Dr1, z1);
isvesti2(textas2, absmomvid, absmomd, i, Dr2, z2);
isvesti2(textas3, hvarvid, hvard, i, Dr3, z3);
isvesti2(textas4, varrezvid, varrezd, i, Dr4, z4);

       /*isvesti("RSstab1.txt",k, RS1);
       isvesti("momstab1.txt",k, absmom);
       isvesti("varstab1.txt",k, hvar);
       isvesti("varrezstab1.txt",k, varrez); */


     /*if (ComboBox1->ItemIndex==0){
      double t = ttest(RS1d, RS1vid, i,k);
        Memo1->Lines->Add(t); }  */

free(X);
}



}

//---------------------------------------------------------------------------
void TForm1::isvesti2(char* textas, double a, double b, double c, double d,
 double z)
{
ofstream myfile;
myfile.open (textas, ios::app);


myfile << "alfa=" << c << endl;
myfile << "Vidurkiai" << endl;
myfile <<a<< endl;
myfile << "Standartiniai nuokrypiai" << endl;
myfile <<b<< endl;
myfile << "Kolmogorovo statistika:" << endl;
myfile <<d<< endl;
myfile << "z-test:" << endl;
myfile <<z<< endl;
myfile.close();

}
//---------------------------------------------------------------------------
double TForm1::normalityt( int n, double *x5,double vi,double di){
```

```
double Dp,Dp1m,Dp2m, p=0.05,f1,c2,c3;
double Dna,Dng,Dnmin=10000000;

double *Dp1=new double [n];
double *Dp2=new double [n];
int i=0;
double rez=0.0, rezmin=0.0;

double k = n;
Dp1=new double [n];
Dp2=new double [n];
x5 = bubbleSort(x5,n);
       double ec;
       for (i=0;i<n;i++){
         ec = i/k ;

              }
                 for (i=0;i<n;i++){

                                  f1=gaussp((x5[i]-vi)/di);

                                  c2=((double)i+1)/k;
                                  c3=((double)i)/k;
                                  Dp1[i]= c2- f1;
                                  Dp2[i]=f1-c3;
                 }
                 Dp1m=maxmax(Dp1,k);
                 Dp2m=maxmax(Dp2,k);
                 Dng=max(Dp1m,Dp2m);









                 free (Dp1);
       free (Dp2);
return Dng;

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



//---------------------------------------------------------------------------

#ifndef Unit1H
#define Unit1H
//---------------------------------------------------------------------------
```

```
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <Chart.hpp>
#include <ExtCtrls.hpp>
#include <TeEngine.hpp>
#include <TeeProcs.hpp>
#include <Series.hpp>
#include <strstream>
#include <math.h>
#include <DbChart.hpp>
#include <iostream>
#include <fstream>
#include <stdio.h>
#include <stdlib.h>
#include <algorithm>
#include <vector>
#include <sstream>
#include <string>


#include "MersenneTwister.h"#invlude <stdlib.h>
using namespace std;
const int n=5000;
const double pi = 3.141592654;
//const int m = 2;
//int mk =0;
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published: // IDE-managed Components
        TMemo *Memo1;
        TButton *Button1;
        TButton *Button2;
        TButton *Button3;
        TComboBox *ComboBox1;
        TLabel *Label1;
        TButton *Button4;
        TLabel *Label2;
        TChart *Grafikas;
        TPointSeries *Series1;
        TEdit *Edit1;
        TLabel *Label3;
        TButton *Button5;
        TButton *Button6;
        TEdit *Edit2;
        TEdit *Edit3;
        TLabel *Label4;
        TLabel *Label5;
        TLabel *Label6;
        TLabel *Label7;
```

```
        TLabel *Label8;
        TButton *Button7;
        TButton *Button8;
        TLabel *Label9;
        void __fastcall Button4Click(TObject *Sender);
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall Button2Click(TObject *Sender);
        void __fastcall Button3Click(TObject *Sender);
        void __fastcall Button5Click(TObject *Sender);
        void __fastcall Button6Click(TObject *Sender);
        void __fastcall Button7Click(TObject *Sender);
        void __fastcall Button8Click(TObject *Sender);



private:
double *X;
double failoilgis;
public:
double RS(double *X,int n);
double vid(double *x5,int nn1);
double minmin(double*x,int n);
double maxmax(double*x,int n);
double disp(double*x5,double m,int nn1);
double gaussp(double y);
double gaus();
double *Xas(int n);
double mom_X(double *X, int m, int k);
double *mom_agreg(double *X, int m, int K);
double mom_agr(double *X, int n, int m,double M);
double hurst_absmom(double *X,int n, int m);
double kovarijacia(double *x5, double *x6, int nn1);
double var_agr(double *X, int n, int m,double M);
double hurst_var(double *X,int n, int m);
double agr_RS(double *X,int n,int K);
double hurst_RS(double *X,int n);
double *agregavimas(double *X, int nn,int k);
double whit1(double *X, int nn, int j, double v);
double *bubbleSort ( double *arr, int size );
void var_graf(double *X,int n, int m, double H);
void absmom_graf(double *X,int n, int m, double H);
double per_X(double *X, int N, double v);
double varrez(double *X, double *Y, int n);
double varrez_agr(double *X, int n, int m);
double hurst_varres(double *X,int n, int K);
double *zsuma(double *X, int m);
double med(double *X,int nn1);
double *keisti(double *x1,int nn);
double paklaidos(double*x5,double m,int nn1);
void isvesti(char* textas, int n, double *x);
```

```
double *generavimas(double *x,const int r);
double ztest(double disp, double vid, double u, int n);
void varres_graf(double *X,int n, int K);
void RS_graf(double *X,int n);
double *ivesti();
double eilutesfaile();
void isvesti2(char* textas, double a, double b, double c, double d, double z);
double normalityt( int n, double *x5,double vi,double di);
    __fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif
```

4 priedas. Programos kodas duomenims atvaizduoti

```
Sub Trinti()
    Sheet2.Select
    Range("A6").Select
    Range(Selection, Selection.End(xlToRight)).Select
    Range(Selection, Selection.End(xlDown)).Select
    Selection.ClearContents
    Range("A6").Select
End Sub

Sub histograma()
imtis = 10000
ReDim M(1 To imtis)
For i = 1 To imtis
        M(i) = Sheet1.Cells(i, 1).Value
    Next i

 ReDim w(1 To imtis)
For j = 1 To imtis
        w(j) = Sheet1.Cells(j, 2).Value
    Next j
    ReDim p(1 To imtis)
For i = 1 To imtis
        p(i) = Sheet1.Cells(i, 3).Value
    Next i
Call Hist(M(), 5, 1)
Call Hist(w(), 5, 4)
Call Hist(p(), 5, 7)

End Sub
Sub Hist(M() As Variant, z As Integer, c As Integer)


Dim nbbin As Double
Dim min, max As Variant
Dim binsize As Variant
Dim bin() As Variant
Dim binlabel() As Variant

imtis = 10000
nbbin = Sheet2.Cells(1, 2).Value
ReDim frequency(1 To nbbin)
ReDim binlabel(1 To nbbin)
min = -10
max = 10
binsize = (max - min) / nbbin

For i = 1 To nbbin
binlabel(i) = min + (i) * binsize
```

```
Next

For i = 1 To nbbin - 1
For j = 1 To UBound(M)
If M(j) >= binlabel(i) - 0.5 * binsize And M(j) < binlabel(i + 1) - 0.5 * binsize Then
frequency(i) = frequency(i) + 1
End If
Next
Next
For j = 1 To UBound(M)
If M(j) >= binlabel(nbbin) - 0.5 * binsize And M(j) < max Then
frequency(nbbin) = frequency(nbbin) + 1
End If
Next

For i = 1 To nbbin
      Sheet2.Cells(z + i, c).Value = binlabel(i)
      Sheet2.Cells(z + i, c + 1).Value = frequency(i)
Next
End Sub

Sub Button3_Click()

Call Trinti

End Sub

Sub Button4_Click()

   Call histograma

End Sub
```