KAUNO TECHNOLOGIJOS UNIVERSITETAS
MATEMATIKOS IR GAMTOS MOKSLŲ FAKULTETAS

Gabrielė Kaltanaitė

DALELIŲ SPIEČIAUS INTELEKTO PANAUDOJIMAS ĮKAINOJANT PASIRINKIMO SANDORIUS

Baigiamasis magistro projektas

Vadovas
lekt. Rita Palivonaitė

KAUNAS, 2015

KAUNO TECHNOLOGIJOS UNIVERSITETAS
MATEMATIKOS IR GAMTOS MOKSLŲ FAKULTETAS

DALELIŲ SPIEČIAUS INTELEKTO PANAUDOJIMAS ĮKAINOJANT PASIRINKIMO SANDORIUS

Baigiamasis magistro projektas
Taikomoji matematika (kodas 621G10003)

Vadovas
lekt. Rita Palivonaitė

Recenzentas
doc. dr. Audrius Kabašinskas

Projektą atliko
Gabrielė Kaltanaitė

KAUNAS, 2015

KAUNO TECHNOLOGIJOS UNIVERSITETAS
MATEMATIKOS IR GAMTOS MOKSLŲ FAKULTETAS
Gabrielė Kaltanaitė
Taikomoji matematika (kodas 621G10003)

Baigiamojo projekto „Dalelių spiečiaus intelekto panaudojimas įkainojant pasirinkimo sandorius“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

2015 m. birželio mėn. 5 d.
Kaunas

Patvirtinu, kad mano, **Gabrielė Kaltanaitė,** baigiamasis darbas tema „Dalelių spiečiaus intelekto panaudojimas įkainojant pasirinkimo sandorius“ yra parašytas visiškai savarankiškai, o visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena darbo dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymu nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

_____ _____

(studento vardas ir pavardė, įrašyti ranka) (parašas)

Kaltanaitė G. Particle Swarm Optimization in Option Pricing: Master‘s work in applied mathematics/ supervisor lekt. Rita Palivonaitė; Kaunas University Of Technology, The Faculty of Mathematics and Natural Science, department of Mathematical Modelling.

Kaunas, 2015. - 67 p.

SUMMARY

Swarm intelligence is a well-known collective behavior of self-organized systems, natural or artificial, which is widely used in solving complex non-linear optimization problems. One of the most challenging and popular problem in computational finance is option pricing. In this research we investigate one of the artificial intelligence based techniques – particle swarm optimization (PSO) algorithm. According to foreign and Lithuanian literature review, in this paper two stochastic volatility option pricing models are used: the Heston and Bates models which leads to optimization problems. The goal of this paper is to implement particle swarm optimization algorithm and evaluate its efficiency in finding the best sets of parameters for these two models. Optimization procedure for obtaining the optimal sets of parameters is performed on simulated data and on historical data from *https://www.historicaloptiondata.com*.

Turinys

Lentelių sąrašas

Paveikslų sąrašas

Įžanga

Išvestinės finansinės priemonės vaidina svarbų vaidmenį investuotojo gyvenime. Viena tokių priemonių yra pasirinkimo sandoriai. Kiekybinėje finansų inžinerijoje pasirinkimo sandorių įkainojimas vienas iš sudėtingiausių ir svarbiausių uždavinių. Dėl stipriai kintančių ir dinamiškų rinkos sąlygų nėra jokių uždaros formos sprendinių, kurie būtų priimtini paprastos rūšies pasirinkimo sandoriams, tokiems kaip Europietiški pasirinkimo sandoriai.

Biologijos ir gamtos įkvėpti algoritmai įgijo didelį populiariumą sprendžiant realius sudėtingus mokslinius ir inžinerinius optimizavimo uždavinius, tokius kaip mobiliųjų socialinių tinklų sistemos. Šie algoritmai yra pagrįsti gyvūnų socialiniu elgesiu, tokiu kaip žuvų ar paukščių būrių, bičių spiečių judėjimu. Pasirinkimo sandorio įkainojimas atitinka tokio pobūdžio optimizavimo uždavinį. Dalelių spiečiaus optimizavimas yra vienas iš euristinių paieškos algoritmų paremtas spiečiaus intelektu [16].

Šiame darbe tiriami du pasirinkimo sandorio įkainojimo modeliai: Heston stochastinio kintamumo modelis ir Bates modelis. Aptariame pasirinkimo sandorių įkainojimą šiais modeliais, kai optimalūs jų parametrai gaunami panaudojant dalelių spiečiaus optimizavimo algoritmą ir istorinius duomenis.

Tyriamojoje dalyje pateikiami duomenys ir rezultatai, gauti atliekant simuliacijas ir panaudojant realius rinkos duomenis. Tyrimui naudojama MATLAB R2012b programinė įranga.

1. Teorinė dalis

Šioje dalyje apžvelgti su magistro baigiamojo darbo tikslais susiję užsienio ir Lietuvos mokslininkų paskelbti straipsniai finansinių aktyvų ir dirbtinio intelekto tematika.

1.1. Pasirinkimo sandoriai

Pasirinkimo sandoris (angl. *options*) yra toks sandoris, kuris jo pirkėjui suteikia teisę, bet neįpareigoja, už sutartą kainą pirkti arba parduoti finansinį aktyvą iki sutartos datos ateityje. Pasirinkimo sandoris kaip ir akcija ar obligacija yra vertybinis popierius. Į jį galima žiūrėti kaip į sutartį, su detaliai ir tiksliai apibrėžtomis sąlygomis bei savybėmis. Opcionai gali būti dviejų tipų:

1. Pirkimo (angl. *call*). Šis opcionas suteikia teisę jo turėtojui pirkti susietą turtą už tam tikrą kainą iki tam tikro laiko. Pirkimo opcionai panašūs į ilgalaikę (angl. *long*) akcijų poziciją - perkantysis pasirinkimo sandorį, tikisi susietosios akcijos ženklaus kainos augimo iki opciono galiojimo pabaigos datos.

2. Pardavimo (angl. *put*). Šis pasirinkimo sandoris jo turėtojui suteikia teisę parduoti susietą turtą už tam tikrą kainą iki tam tikro laiko (pasirinkimo sandorio galiojimo pabaigos datos). Pardavimo pasirinkimo sandoriai panašūs į trumpąją (angl. *short*) akcijų poziciją - pardavimo pirkėjas tikisi, kad iki pasirinkimo sandorio galiojimo pabaigos datos susieto turto vertė ženkliai kris.

Pasirinkimo sandorio kaina vadinama premija. Premija paprastai sudaro mažą pasirinkto opciono aktyvo kainos dalį. Pasirinkimo sandoriai yra išvestiniai vertybiniai popieriai, nes jų vertė priklauso nuo susieto turto vertės.

Pasirinkimo sandorio sutartyje turi būti nurodyta [10]:

1. Kas yra perkama (pirkimo atveju) ar parduodama (pardavimo atveju). Dažniausiai akcijų pasirinkimo sandoriuose perkama arba parduodama 100 vienetų akcijų.

2. Įvykdymo (ceremonijos) kaina X (angl. *strike* ar *exercise price*). Tai tokia kaina, už kurią yra perkamas arba parduodamas su pasirinkimo sandoriu susietas turtas. Įvykdymo kaina rodo, iki kiek turi pakilti (pirkimo pasirinkimo sandoriams) arba nukristi (pardavimo pasirinkimo sandoriams) susieto turto kaina, kad pasirinkimo sandorį apsimokėtų vykdyti ir būtų gautas pelnas.

3. Pasirinkimo sandorio galiojimo (gyvavimo) terminas T (angl. *expiration* ar *maturity time*).

Investuotojai, kurie perka pasirinkimo sandorius, yra vadinami pasirinkimo sandorių laikytojais arba turėjais (angl. *holders*), o tie, kurie juos parduoda, vadinami pasirinkimo sandorių leidėjais arba išrašytojais (angl. *writers*).

Kadangi pasirinkimo sandoris yra sandoris tarp dviejų šalių ir jie būna dviejų tipų, todėl rinkoje dalyvauja keturių tipų dalyviai:

1. Pirkimo opcionų pirkėjai;
2. Pirkimo opcionų pardavėjai;
3. Pardavimo opcionų pirkėjai;
4. Pardavimo opcionų pardavėjai;

Esminis skirtumas tarp pirkėjų ir pardavėjų yra toks:

- Pirkėjai (tiek pirkimo tiek pardavimo) nėra įpareigoti pirkti ar parduoti finansinį aktyvą. Jie gali savo nuožiūra arba pasinaudoti pasirinkimo sandorio suteikiamomis teisėmis arba ne. Todėl pirkėjams nereikalingas joks išankstinis susieto turto ar lėšų užstatas, garantuojantis pasirinkimo sandorio įvykdymą.
- Pardavėjai (tiek pirkimo tiek pardavimo) privalo pirkti finansinį aktyvą, todėl dažniausiai jie pasirinkimo sandorio vykdymo garantui turi turėti finansinį aktyvą ar piniginį užstatą [6].

Yra tokių rūšių pasirinkimo sandoriai:

Europietiškasis pasirinkimo sandoris (angl. *European option*). Sandoris gali būti įvykdytas tik pasirinkimo sandorio pabaigoje.

Amerikietiškasis pasirinkimo sandoris (angl. *American option*). Sandoris gali būti realizuotas bet kuriuo laiku iki pasirinkimo sandorio pabaigos.

1.1.1. Pasirinkimo sandorio vertė

Pasirinkimo sandorio turėtojas jį realizuos, jei sandorio termino pabaigoje T jis turės naudą. Pirkimo pasirinkimo sandoris bus realizuotas tik tada, kai akcijos rinkos kaina S_T termino pabaigoje yra didesnė už įvykdymo kainą X. Tokio pasirinkimo sandorio vertė lygi $S_T - X$. Todėl pirkimo pasirinkimo sandorio vertė C_T termino pabaigoje yra:

$$\boldsymbol{C_T = \begin{cases} S_T - X, & S_T > X, \\ 0, & S_T \leq X. \end{cases}} \qquad \textbf{(1.1)}$$

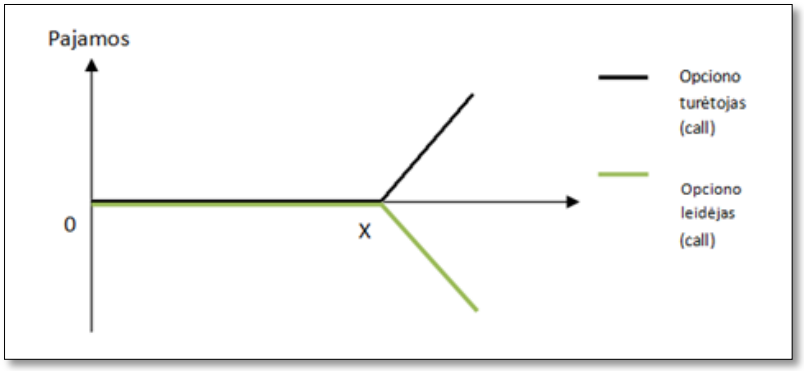
arba kitaip galima užrašyti: $C_T = \max(S_T - X)$

Analogiškai pardavimo pasirinkimo sandorio vertė lygi:

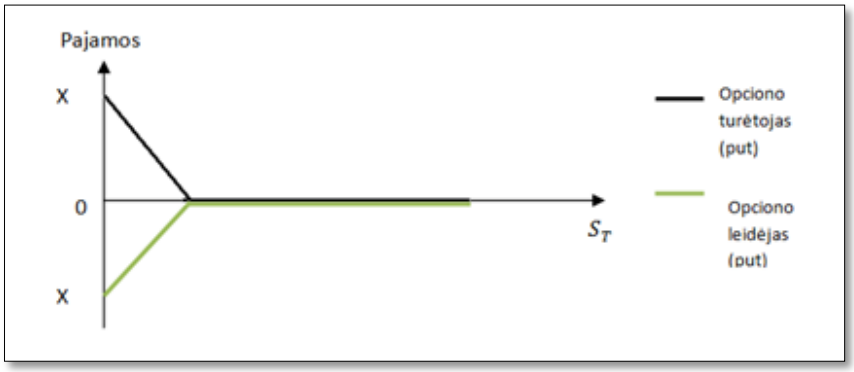
$$\boldsymbol{P_T = \begin{cases} X - S_T, & S_T < X, \\ 0, & S_T \geq X. \end{cases}} \qquad \textbf{(1.2)}$$

arba $P_T = \max(X - S_T)$.

Žemiau pavaizduota pirkimo ir pardavimo pasirinkimo sandorių vertės (1.1) ir (1.2) grafikai. Matome, kad pirkimo pasirinkimo sandorio turėtojas gauna pelną, kai akcijos pasiekia įvykdymo kainą. Ir atvirkščiai, pirkimo pasirinkimo sandorio leidėjas, neturi nuostolių iki kol akcijos kaina nepasiekė įvykdymo kainos. Čia nevertiname kainos, už kurią buvo pirktas pasirinkimo sandoris.



1.1 pav. Pirkimo pasirinkimo sandorio turėtojo ir leidėjo pelnas

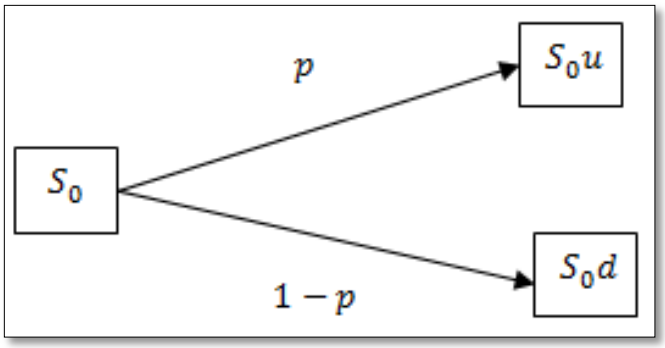


1.2 pav. Pardavimo pasirinkimo sandorio turėtojo ir leidėjo pelnas

Matome, kad pardavimo pasirinkimo sandorio turėtojas gauna pelną iki kol akcijos kaina pasiekia įvykdymo kainą. Ir atvirkščiai, pardavimo pasirinkimo sandorio leidėjas, neturi nuostolių nuo tada, kai akcijos kaina pasiekia įvykdymo kainą. Čia taip pat nevertiname kainos, už kurią buvo pirktas pasirinkimo sandoris [18].

1.2. Binominis modelis pasirinkimo sandorių įkainojimui

Binominis modelis yra vienas iš plačiausiai naudojamų vertybinių popierių įkainojimo modelių. Šis modelis gali įgyti tik dvi reikšmes periodo (intervalo) pabaigoje. Tarkime, kad akcijos kaina šiandieną lygi S_0. Taikant binominį modelį, akcijos kainos mažame intervale kinta. Ji pakyla iki S_0u su tikimybe p ir nukrenta iki S_0d su tikimybe $1-p$. Vadinasi periodo pabaigoje akcijos kaina gali įgyti tik dvi reikšmes: arba S_0u su tikimybe p, arba S_0d su tikimybe $1-p$ (kaip pavaizduota (1.3) pav.). Ši prielaida nėra reali, tik tikimybinė, nes akcijos kaina yra visiškai atsitiktinis dydis. Be abejo, kad kainos kilimų ir kritimų per periodą gali būti daug [11].



1.3 pav. Binominis modelis

Pažymėkime S_t akcijos kainą diskrečiuoju momentu t, $t = 0, 1, 2, \ldots$ Tuomet kiekvieno periodo pabaigoje akcijos kaina gali įgyti dvi reikšmes:

$$S_t = \begin{cases} u \cdot S_t, & jei\ kaina\ pakyla\ su\ tikimybe\ p, \\ d \cdot S_t, & jei\ kaina\ pakyla\ su\ tikimybe\ 1-p. \end{cases} \quad \textbf{(1.3)}$$

iš čia u ir d yra atitinkamai didinimo ir mažinimo konstantos bei $d < 1 < u$.

Bendruoju atveju po n periodų akcija gali įgyti vieną iš reikšmių

$$S_n = S_0 \cdot u^k d^{n-k} \quad \textbf{(1.4)}$$

Turėdami binominio modelio parametrų įverčius:

$$\hat{u} = e^{\sigma\sqrt{\Delta t}} \quad \text{ir} \quad \hat{d} = e^{-\sigma\sqrt{\Delta t}}$$

galime užpildyti kainų gardelę, rasti akcijos kainas po N periodų ir apskaičiuoti pasirinkimo sandorio vertę (gaunamas pajamas) jo pabaigoje. Pažymėjimai ir reikšmės:

S_0 - akcijos kaina nuliniu (esamuoju) momentu; X- pasirinkimo sandorio įvykdymo kaina; u- daugiklis, kuris naudojamas akcijos kainai padidinti; d- daugiklis, kuris naudojamas akcijos kainai sumažinti; σ- svyravimo koeficientas arba standartinis nuokrypis; q- rizikai neutrali tikimybė (angl. *risk-free probability*), kuri išreiškiama tokia formule:

$$q = \frac{R-d}{u-d}$$

iš čia, R- vieno periodo nerizikingoji grąžos norma; C- pirkimo pasirinkimo sandorio kaina; P - pardavimo pasirinkimo sandorio kaina;

C_u- pirkimo pasirinkimo sandorio išmoka, jei akcijos kaina padidėja, t.y.

$$C_u = \begin{cases} uS_0 - X, & jei\ uS_0 > X, \\ 0, & jei\ uS_0 \leq X. \end{cases}$$

C_d - pirkimo pasirinkimo sandorio išmoka, jei akcijos kaina sumažėja, t.y.

$$C_d = \begin{cases} dS_0 - X, & jei\ dS_0 > X, \\ 0, & jei\ dS_0 \leq X. \end{cases}$$

Sandorio vertę vienu periodu ankščiau randame ieškodami diskontuoto verčių vidurkio pagal tokią bendrą formulę:

$$C = \frac{1}{R}(qC_u + (1-q)C_d) \quad \textbf{(1.5)}$$

Diskontuojame naudodami nerizikingosios palūkanų normos grąžą $R = 1 + r$, o kad apskaičiuoti vidurkį, naudojame rizikai neutralią tikimybę q. Tokius skaičiavimus atliekame tol, kol apskaičiuojame dabartinę pasirinkimo sandorio vertę C pagal nurodytą formulę (1.5).

1.3. Black-Scholes-Merton modelis

Black–Scholes-Merton arba Black-Scholes (BSM) modelis yra binominio modelio tolydžioji aproksimacija. Jis buvo sukurtas 1973 m. Fischer Black ir Myron Scholes. Šis modelis turi didelę reikšmę įkainojant ir draudžiant europietiškus pasirinkimo sandorius. Black-Scholes modelis nuo binominio modelio skiriasi dviem pagrindinėmis prielaidomis:

- Akcijos kaina ateityje pereinant nuo vieno laiko momento iki kito gali įgyti ne dvi galimas reikšmes, tačiau bet kurią reikšmę iš tam tikro baigtinio realiųjų skaičių intervalo;
- Akcijos kainos kinta tolydžiu laiku, o ne diskrečiais laiko momentais (mėnesiais, dienomis, valandomis ir pan.), t.y. akcijų kaina gali kisti per bet kurį nykstamai mažą laiko intervalą;

Black-Scholes europietiškojo pirkimo nuliniu laiko momentu pasirinkimo sandorio įkainojimo formulė:

$$C = S_0 \cdot \Phi(h_1) - X \cdot e^{-r \cdot T} \cdot \Phi(h_2) \quad \textbf{(1.6)}$$

Black-Scholes europietiškojo pardavimo nuliniu laiko momentu pasirinkimo sandorio įkainojimo formulė:

$$P = X \cdot e^{-r \cdot T} \cdot \Phi(-h_2) - S_0 \cdot \Phi(-h_1) \quad \textbf{(1.7)}$$

čia,

$$h_1 = \frac{\ln\left(\frac{S_0}{X}\right) + \left(r + \frac{\sigma^2}{2}\right) \cdot T}{\sigma\sqrt{T}} \quad \textbf{(1.8)}$$

$$h_2 = \frac{\ln\left(\frac{S_0}{X}\right) + \left(r - \frac{\sigma^2}{2}\right) \cdot T}{\sigma\sqrt{T}} = h_1 - \sigma\sqrt{T} \quad \textbf{(1.9)}$$

Normaliojo standartinio skirstinio pasiskirstymo funkcija, $y \sim N(0,1)$:

$$\Phi(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\omega} e^{-\frac{y^2}{2}} dy \quad \textbf{(1.10)}$$

čia S_0 - pradinė akcijos kaina, X- įvykdymo kaina, r - nerizikingoji palūkanų norma, σ- akcijos svyravimo koeficientas, T- opciono gyvavimo laiko periodas, $C\ ir\ P$ - europietiškojo pirkimo ir pardavimo pasirinkimo sandorių kainos.

Be galo mažam laiko intervalui Δt, kai $N \to \infty$, t.y. be galo didinant žingsnių skaičių, binominis įkainojimo modelis turi konverguoti į Back-Scholes modelį. Todėl galutinė binominio modelio aktyvo kaina turi konverguoti į lognormalųjį skirstinį, kai $\Delta t \to 0$.

1.4. Įkainojimas charakteristine funkcija

Yra keli apibendrinti metodai pasirinkimo sandoriams įkainoti. Vienas jų - Black-Scholes-Merton metodas, kurio pagrindas yra neginčytinas argumentas; tai veda į dalinių diferencialų lygtį, kuri gali būti išsprendžiama skaitiniu arba tam tikrais atvejais ir analitiniu būdu. Naujesnis būdas yra konstruojamas pasinaudojant logaritmine akcijos kainos charakteristine funkcija. Taigi, europietiški pasirinkimo sandoriai gali būti įkainoti tokia lygtimi

$$C_0 = e^{-q\tau} S_0 \Pi_1 - e^{-r\tau} X \Pi_2 \quad \textbf{(1.11)}$$

čia C_0 – pirkimo kaina šiandien (nuliniu momentu), S_0 – pradinė akcijos kaina, X – įvykdymo kaina, r – nerizikingoji palūkanų norma, q – dividendų pajamos, τ – pasirinkimo sandorio gyvavimo laiko periodas. Nežinomasis Π_j apskaičiuojami pagal formules:

$$\Pi_1 = \frac{1}{2} + \frac{1}{\pi} \int_0^{\infty} \mathrm{Re}\left(\frac{e^{-i\omega \log(X)} \phi(\omega - i)}{i\omega\phi(-i)}\right) d\omega \quad \textbf{(1.12)}$$

$$\Pi_2 = \frac{1}{2} + \frac{1}{\pi} \int_0^{\infty} \mathrm{Re}\left(\frac{e^{-i\omega \log(X)} \phi(\omega)}{i\omega}\right) d\omega \quad \textbf{(1.13)}$$

Šiose lygybėse esantiems integralams apibrėžiame $\Pi_j^* \equiv \pi(\Pi_j - 1/2)$. Simbolis ϕ yra logaritminės akcijos kaina charakteristinei funkcijai. Funkcija Re(·) grąžina kompleksinio skaičiaus realiąją dalį. Turint ϕ, skaitiniu metodu galime apskaičiuoti $\Pi_1$ir Π_2 reikšmes, taigi pasirinkimo sandorio kainos apskaičiuojamos iš lygties (1.11).

1.4.1. Merton modelis

Sakykime, jeigu kompanijos kokia nors labai svarbi informacija išeina į viešumą, tai ji gali įtakoti staigius pasikeitimus kompanijos akcijų kainai. Tokia informacija dažniausiai pasitaiko atsitiktinai ir šio poveikio dydis akcijų kainai traktuojamas kaip atsitiktinis kintamasis. Tam, kad susidoroti su tokia informacija Mertonas (1976) pasiūlė modelį, kuris sudaro sąlygas netolydžioms turto kainų trajektorijoms. Žinodami, jog Black-Scholes modelio bazinė akcijos kaina tariamai išsiveda iš geometrinio Brauno judėjimo [20] dinamikos:

$$dS_t = rS_t dt + \sigma S_t dW_t \quad \textbf{(1.14)}$$

Tuomet Merton modelis yra (1.14) formulės plėtinys su akcijos kainų dinaminiais šuoliukais:

$$\frac{dS_t}{S_t} = rdt + \sigma dW_t + dZ_t \quad \textbf{(1.15)}$$

Čia Z_t – sudėtinis Puasono procesas su lognormaliuoju šuoliukų dydžių pasiskirstymu. Šuoliukai laikosi homogeninio Puasono proceso N_t su intensyvumu λ, kuris yra nepriklausomas nuo W_t. Logaritminių šuoliukų dydžiai $Y_i \sim N(\mu, \delta^2)$ yra nepriklausomi ir vienodai pasiskirstę atsitiktiniai kintamieji su vidurkiu μ ir variacija δ^2, kurie yra nepriklausomi nuo N_t ir W_t.

Egzistuoja daug variantų kaip pasirinkti rizikai neutralų matą šiam modeliui, tokį kad diskontuotos kainos procesas būtų martingalas (pastatytos sumos padvigubinimas pralošus). Mertonas pasiūlė pakeisti Vynerio proceso trendą (angl. *drift*) ir palikti kitus faktorius nepakeistus. Tuomet įkainojimo formulė yra:

$$S_t = S_0 \exp\left(\mu^M t + \sigma W_t + \sum_{i=1}^{N_t} Y_i\right) \quad \textbf{(1.16)}$$

čia $\mu^M = r - \sigma^2 - \lambda\{\exp\left(\mu + \frac{\delta^2}{2}\right) - 1\}$. Šuoliuko komponentės prideda svorio grąžų pasiskirstymo uodegoms. Didėjantis parametras δ prideda svorio prie abiejų uodegų. Neigiamas ar teigiamas parametras μ reiškia santykinai daugiau svorio atitinkamai kairei ar dešinei uodegai.

Merton modelio charakteristinė funkcija aprašoma lygybe:

$$\phi_{Merton} = e^{A+B} \quad \textbf{(1.17)}$$

čia turime:

$$A = i\omega s_0 + i\omega\tau\left(r - q - \frac{1}{2}\upsilon - \mu_J\right) + \frac{1}{2}i^2\omega^2\upsilon\tau \quad \textbf{(1.18)}$$

$$B = \lambda\tau\left(exp\left(i\omega\log(1+\mu_J) - \frac{1}{2}i\omega\sigma_J^2 - \omega^2\sigma_J^2\right) - 1\right) \quad \textbf{(1.19)}$$

A narys charaketristinėje funkcijoje atitinka Black-Scholes-Merton dinamiką su trendo korekcija skaičiuoti šuoliukus. B narys prideda prie šuoliuko komponentę.

1.4.2. Heston modelis

Šiame skyrelyje aprašome Heston opcionų įkainojimo modelį ir jo parametrus.

Tarkime, kad akcijos kaina S_0 laiko momentu t apibrėžta stochastiniu procesu:

$$dS_t = rS_t dt + \sqrt{\upsilon_t} S_t dW_t^{(1)} \tag{1.20}$$

kur υ_t yra aktyvo grąžos variacija ir išreiškiama:

$$d\upsilon_t = \text{к}(\theta - \upsilon_t)dt + \sigma\sqrt{\upsilon_t} dW_t^{(2)} \tag{1.21}$$

iš čia $W_t^{(1)}$ ir $W_t^{(2)}$ yra Vynerio procesai (angl. *Wiener processes*), kurie koreliuoja vienas su kitu ir gali būti užrašomas tokia lygybe [13]:

$$dW_t^{(1)} dW_t^{(2)} = \rho dt \tag{1.22}$$

Žymime $(S_t)_{t\geq 0}$ ir $(\upsilon_t)_{t\geq 0}$ atitinkamai akcijos kainos ir sklaidos parametro procesai.

Brauno procesas (angl. *The Brownian motion*) su koreliacijos koeficientu ρ:

$$\rho = corr(dW_t^{(1)}, dW_t^{(2)})$$

Formulėse esantys parametrai ir jų reikšmės:

θ – sklaidos parametro vidurkis (angl. *the long run variance*); к – grįžimo prie vidurkio intensyvumas (angl. *mean reversion*); σ – kintamumo sklaidos parametras (angl. *volatility of volatility*);

Kai $\sigma \to 0$, tuomet Hestono modelio dinamika priartėja į Black-Scholes-Merton modelį.

Heston modelio logaritminės kainos charakteristinė funkcija aprašoma tokia lygybe [1]:

$$\phi_{Heston} = e^{A+B+C} \tag{1.23}$$

iš čia turime

$$A = i\omega s_0 + i\omega(r-q)\tau \tag{1.24}$$

$$B = \frac{\theta\text{к}}{\sigma^2}\left((\text{к} - \rho\sigma i\omega - d)\tau - 2\log\left(\frac{1-ge^{-d\tau}}{1-g}\right)\right) \tag{1.25}$$

$$C = \frac{\left(\frac{\upsilon_0}{\sigma^2}\text{к} - \rho\sigma i\omega - d\right)\left(1-e^{-d\tau}\right)}{1-ge^{-d\tau}} \tag{1.26}$$

$$d = \sqrt{(\rho\sigma i\omega - \text{к})^2 + \sigma^2(i\omega + \omega^2)} \tag{1.27}$$

$$g = \frac{\text{к} - \rho\sigma i\omega - d}{\text{к} - \rho\sigma i\omega + d} \tag{1.28}$$

Tik su penkiais parametrais su rizikos neutralia tikimybe (angl. risk neutral probability), Heston modelis gali pavaizduoti kintamumo šypsenos kreivę (angl. *volatility smile*).

Tuomet dabartinė Europietiško pirkimo pasirinkimo sandorio vertė gali būti apskaičiuojama naudojantis tikimybiniu modeliu:

$$C_0 = S_0\Pi_1 - e^{rT}K\Pi_2 \quad \textbf{(1.29)}$$

kur Π_1 ir Π_1 yra dvi tikimybinės funkcijos (1.12 ir 1.13).

1.4.3. Bates modelis

Bates modelis prideda šuoliukų procesą (atsitiktinius šuoliukų įvykių laikus ir atsitiktinių šuoliukų dydžius). Heston modelio dinamikoje. Akcijos kaina S ir jos variacija v aprašoma tokiomis lygtimis:

$$dS_t = (r - q - \lambda\mu_J)S_t dt + \sqrt{v_t}S_t dW_t^{(1)} + J_t S_t dN_t \quad \textbf{(1.30)}$$

$$dv_t = \kappa(\theta - v_t)dt + \sigma\sqrt{v_t}dz^{(2)} \quad \textbf{(1.31)}$$

N_t – Puasono procesas su intensyvumu λ, vadinasi tikimybė turėti šuoliuką su dydžiu vienas yra λdt. Vadinasi, šuoliuko dydžio J_t logaritmas yra Gauso skirstinys, pavyzdžiui:

$$\log(1 + J_t) = \mathcal{N}\left(\log(1 + \mu_J) - \frac{\sigma_J^2}{2}, \sigma_J^2\right). \quad \textbf{(1.32)}$$

Charakteristinė funkcija aprašoma [12]:

$$\phi_{Bates} = e^{A+B+C+D} \quad \textbf{(1.33)}$$

iš čia turime:

$$A = i\omega s_0 + i\omega(r - q)\tau \quad \textbf{(1.34)}$$

$$B = \frac{\theta\kappa}{\sigma^2}\left((\kappa - \rho\sigma i\omega - d)\tau - 2\log\left(\frac{1 - ge^{-d\tau}}{1 - g}\right)\right) \quad \textbf{(1.35)}$$

$$C = \frac{\left(\frac{v_0}{\sigma^2}\kappa - \rho\sigma i\omega - d\right)\left(1 - e^{-d\tau}\right)}{1 - ge^{-d\tau}} \quad \textbf{(1.36)}$$

$$D = -\lambda\mu_J i\omega\tau + \lambda\tau\left(\left(1 + \mu_J\right)^{i\omega} e^{\frac{1}{2}\sigma_J^2 i\omega(i\omega - 1)} - 1\right) \quad \textbf{(1.37)}$$

$$d = \sqrt{(\rho\sigma i\omega - \kappa)^2 + \sigma^2(i\omega + \omega^2)} \quad \textbf{(1.38)}$$

$$g = \frac{\kappa - \rho\sigma i\omega - d}{\kappa - \rho\sigma i\omega + d} \quad \textbf{(1.39)}$$

Kai daroma prielaida, kad šuoliukai yra nepriklausomi, charakteristinė funkcija išplaukia iš Heston modelio charakteristinės funkcijos pridedant šuoliukų dalį D.

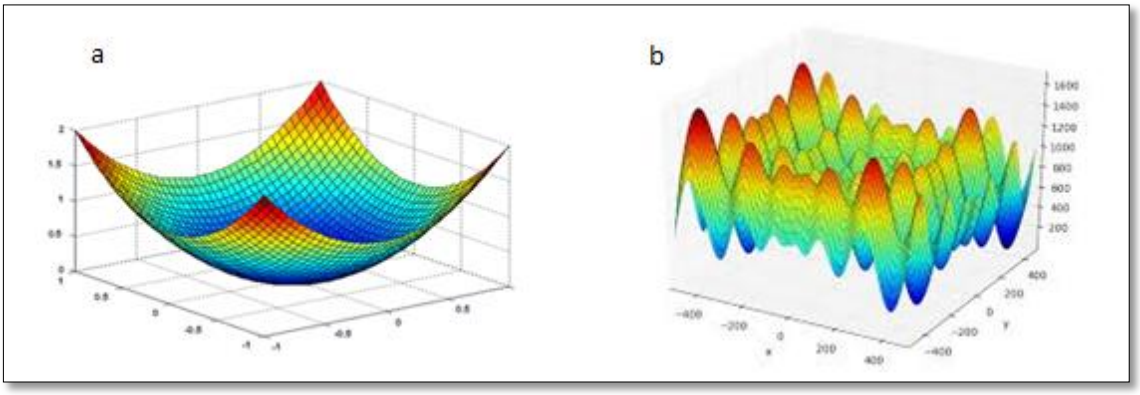
1.5. Parametrų parinkimas

Prieš naudojant įkainojimo modelį, turime būti tikri, kad modelis pateikia tikslius rezultatus pasirinkimo sandoriams, kurie jau yra prekiaujami rinkoje. Pirmiausia turime suprasti kai kiekvienas parametras įtakoja opciono kainą ir parenka svarbiausius. Taip pat apibrėšime tokias parametrų ribas [3]:

- Sklaidos parametro vidurkis θ ir pradinis kintamumas v_0. Priimtinos kintamumo lygių reikšmės turi būti leistinos. Tačiau, duotam jų grįžimo prie vidurkio intensyvumui κ, finansinio aktyvo sklaidos parametro σ lygis retai kada pasiekia 100%. Dėl šios priežasties tyrime naudojame ribas nuo 0 iki 1 abiems reikšmėms θ ir v_0.
- Koreliacija ρ. Statistinė koreliacija ima reikšmes nuo -1 iki 1. Žinoma, jog koreliacija tarp sklaidos parametro ir akcijos kainų krypsta į neigiamą reikšmę. Tačiau, teigiamos koreliacijos reikšmės taip pat gali būti įmanomos kai kuriais atvejais. Todėl naudojame visą intervalą priimtinų reikšmių.
- Kintamumo sklaidos parametras σ. Atsižvelgiant į sklaidos parametrą, šis parametras turėtų rodyti teigiamas reikšmes. Tačiau bazinio aktyvo sklaidos parametras gali stipriai keistis esant trumpiems laiko periodams (sklaidos parametras iš savęs yra labai kintantis). Todėl, šiam parametrui yra reikalaujamos aukštesnės viršutinės ribos. Tam kad išvengti galimų apribojimų, gana platus sprendimų rinkinys, t.y. nuo 0 iki 5 bus naudojamas kalibravime.
- Grįžimas prie vidurkio intensyvumo κ. Šio parametro užtikrinimui turėtume paimti teigiamas reikšmes (neigiamos reikšmės sukels atvirkštinį efektą). Tačiau, neradome aiškių įrodymų apie tai, kuri viršutinio apribojimo reikšmė galėtų būti tinkama. Dėl to, vietoje viršutinio rėžio fiksuotos reikšmės nustatomos atsitiktinai maksimalios κ reikšmės kaip neneigiamas apribojimas.
- Neneigiamas apribojimas. Dar viena sąlyga yra reikalaujama, tam kad kintamumo procesas Heston modelyje nepasiektų 0 ar neigiamos reikšmės. Šiuo atžvilgiu, garsus matematikas Feller W. (1951) įrodė, jog sąlyga $2\kappa\theta - \sigma^2 > 0$ (žinoma kaip Feller sąlyga) garantuoja, jog kintamumas CIR (angl. *Cox-Ingersoll-Ross*) modelyje [5] visada griežtai teigiamas.
- Šuoliukų intensyvumas λ. Neneigiamas skaičius, ribos nuo 0 iki 1.
- Šuoliukų vidurkis μ_J. Gali svyruoti nuo -1 iki 1.
- Šuoliukų variacija σ_J. Neneigiamas skaičius, apribojame intervale nuo 0 iki 1.

1.6. Meta-euristiniai metodai

Tradiciniai optimizavimo metodai labiau tinka tyrinėjant tolydžiąsias ir diferencialines funkcijas, ieškant neapriboto optimalaus sprendinio. Tačiau dauguma sudėtingų realaus pasaulio inžinerinių uždavinių yra netiesiniai, kelių objektų dinaminės sistemos, kurios linkusios užstrigti kaip lokalusis optimalus sprendinys duotoje erdvėje. Šiems uždaviniams ne visada būtina ieškoti optimalaus sprendinio [8] (1.4 pav.).



1.4 pav. (a) Galima lokalaus sprendinio paieška (b) Globalaus sprendinio paieška.

Viena iš tokių optimizavimo metodų grupių yra meta-euristiniai modeliai. Meta-euristiniai metodai priklauso stochastinių algoritmų grupei, kuri per ilgą praktikavimo laikotarpį pasiteisino kaip itin efektyvi ir greita priemonė, uždavinių sprendime dideliu mastu. Skaičiavimo intelektas (angl. *Computational Intelligence* - CI) priklauso šiai meta euristinių paieškos metodikų grupei. Meta-euristiniai metodai taip pat gali būti klasifikuojami pagal gamtos populiacijų tyrimų pagrindu paremtas paieškos metodikas, kurios apima skaičiavimo intelektą (CI) ir vienu sprendiniu paremtą paieškos metodiką kaip atkaitinimo modelis (angl. *Simulated Annealing*). Skaičiavimo pasaulis susiduria su netiesinio programavimo (angl. *Nonlinear Programming* - NP) sudėtingų uždavinių sprendimų būdų permainomis. Šios permainos, netiesinių dinaminių problemų apskaičiavimo metodologijoje, tokiose srityse kaip inžinerija, bioinformatika ir skaičiavimas yra skatinamos dėl efektyvesnių, pigesnių ir greitesnių, didelio masto problemų spendimų būdų poreikio. Mokslininkai semiasi įkvėpimo iš to kaip gamtoje esantys organizmai evoliucijos metu prisitaikė sprendžiant sudėtingas adaptacijos problemas. Remiantis biologinės sistemos metafora šie objektai buvo sumodeliuoti veikti sistemose, kurios tvarkytų daugiamačius kintamuosius su keliais galimais sprendiniais sprendžiant netiesinius uždavinius.

Skaičiavimo intelektas (angl. *Computational Intelligence* - CI) yra kylanti dirbtinio intelekto sritis, pastaruoju metu sulaukusi daug dėmesio iš daugelio mokslo technologijos ir

valdymo sričių. Tai labai pasiteisinusi meta-euristinė optimizavimo technika, įkvėpta natūralios evoliucijos, stebint biologines ir neuro biologines elgsenos sistemas. CI aprėpia žemiau išvardintas atsitiktinumo technikas, tokias kaip:

1. Dirbtinės imuninės sistemos (angl. *Artificial Immune System*)
2. Dirbtiniai neuroniniai tinklai (angl. *Artificial Neural Network*)
3. Neraiškioji logika (angl. *Fuzzy Logic*)
4. Evoliuciniai skaičiavimai (angl. *Evolutionary Computation*)
    - Evoliucinės strategijos
    - Evoliucinis programavimas
    - Genetiniai algoritmai
    - Genetinis programavimas
5. Spiečiaus intelektas (angl. *Swarm Intelligence* - SI)

    Dalelių spiečiaus optimizavimas (PSO)

    Skruzdžių kolonijos optimizavimas (ACO)

Dirbtiniai neuroniniai tinklai (angl. *Artificial Neural Network*), finansuose, naudojami apsidraudimui (angl. *hedging*) ir laiko eilučių prognozei. Genetinio programavimo (angl. *Genetic Programming* - GP) metodai naudojami įkainoti pasirinkimo sandorius, aprašomi literatūroje kurioje išvystytos idėjos apie GP ir skruzdžių sistemas vadinamos apibendrintu skruzdžių programavimu (angl. *Generalized Ant Programming* - GAP). Iš čia išvedamos formulės amerikietiško pardavimo pasirinkimo sandorio pagrindinio turto numanomam sklaidos parametrui (angl. *implied volatility*) apskaičiuoti. Numanomas sklaidos parametras yra toks turto kintamumas, kuris apskaičiuojamas naudojant pasirinkimo sandorio vertę. Nėra uždaros formos sprendimo (vieno konkretaus sprendimo būdo), apskaičiuojant numanomą kintamumą, todėl yra dirbama su apytiksliais analitiniais vertinimais. Keber ir Schuster (2003) [7] Vienos universitete aprašė savo eksperimentus, kuriuose jų formulė pateikia tikslius aproksimacijos rezultatus, stipriai pranokstančius ankščiau aprašytų formulių pasiekimus.

Tyrėjai naudojo GP, tam kad palygintų Black-Scholes-Merton modelio prognozes su tikromis rinkos kainomis. Tikimybių mutacijos (angl. *mutation*) ir kryžminimo (angl. *crossover*) koeficientai GP modelyje paprastai yra fiksuoti. Tyrėjai dinamiškai kaitaliojo mutacijos ir kryžminimo koeficientus kiekvieno GP modelio realizacijoje. Tyrinėtojai teigė, jog šis, algoritmas užfiksuoja rinkos kainas realiu laiku ir pateikia geresnį numanomo sklaidos parametro apytikslį vertinimą.

Spiečiaus sąvoka išvesta iš socialinių organizmų sąveikos su jų dinamine biologinio proceso aplinka. Tyrinėjimai tebevyksta įtraukiant gamtos inspiruotus, kaip skaičiavimo įrankius, sudėtingiems uždaviniams spręsti. Šie sumanūs veikėjai bendradarbiauja sinchronizuotame elgesyje tarp vienas kito su numanomų taisyklių sąryšiu, atsiskyrimo ir suderinimo, veda į problemos sprendimą, kuri vadovaujasi sprendimo metrika žinoma kaip tinkamumas (angl. *fitness*). Tai euristinis įvertinimas vadinamas objektine funkcija, kuri yra maksimizuojama arba minimizuojama priklausomai nuo uždavinio problemos.

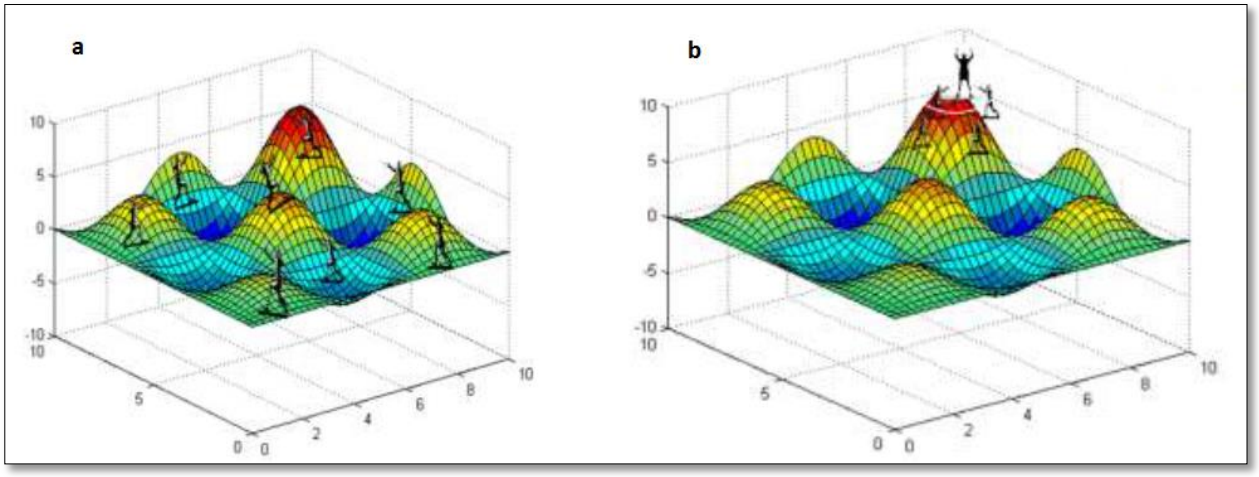
Spiečiaus intelekto globalaus optimizavimo (angl. *Global Optimization* - GO) problema nukreipiama nagrinėjant prisitaikančias sistemas, kurios palengvina sudėtingą sumanų elgesį ir dinamines aplinkas, kur sumanių organizmų skaičius žinomas kaip „veikėjai“ pradėję spręsti optimizavimo problemą esančią čia pat. Paveiksle (1.5 pav.) parodo problemos sprendimo metodologiją ieškomojoje srityje paremtoje meta euristinės populiacijos algoritmu, kur šie sumanūs veikėjai juda įvairiais lokaliais optimaliais taškais tam, kad pasiektų globalų optimalų tašką [8].

Dalelių spiečiaus optimizavimo metodas iš pradžių buvo pateiktas J. Kennedy [9] kaip socialinio elgesio simuliacija ir pristatytas kaip optimizavimo galimybė. Spiečius yra būrelis daugybės homogeninių individų, kurie rodo sumanų elgesį kontroliuodami save, pasikeisdami informacija vieni su kitais. Ši dirbtinio intelekto šaka yra taikoma daugumai optimizavimo problemų spręsti. PSO ir ACO yra dvi gerai žinomos klasės, priklausančios gamtos inspiruotai globalaus optimizavimo metodikai. Stulbinanti žuvų, bičių ir paukščių būrių choreografija, ar net mikroorganizmai, tokie kaip bakterijos, demonstruoja besiformuojantį elgesį. Jų sinchronizuoti nesusiduriantys kitų individų judesiai yra spiečiaus narių, palaikančių optimalų atstumą nuo jų kaimyninių individų, rezultatas. Ši idėja, prieš suformuluojant PSO, buvo visuotinis tikėjimas, jog socialinis informacijos dalinimasis tarp populiacijos individų, jų lankstumas ir prisitaikymas prieš pasikeitimus jų aplinkoje, gali duoti naudos evoliucijai. PSO buvo pagrįstas kaip efektyvus metodas daugeliui GO problemų ir kai kuriais atvejais buvo išvengiama sunkumų susiduriant su kitomis evoliucinio skaičiavimo metodikomis.

1.6.1. Dalelių spiečiaus optimizavimas

Dalelių spiečiaus optimizavimo algoritmas aprašytas 1995 metais Eberhart ir Kennedy [8, 20] darbuose. PSO priklauso spiečiaus intelekto algoritmų grupei ir yra skirtas vieno kriterijaus funkcijų globaliajam optimizavimui. Šis algoritmas yra grįstas socialiniu gyvūnų elgesiu, tokių kaip paukščių ar žuvų migracija: *spiečius* (aibė) *dalelių* (galimų sprendinių) keičia savo pozicijas

uždavinio leistinoje srityje, atsižvelgiant į geriausią žinomą asmeninę poziciją ir geriausią poziciją, aptiktą visų spiečiaus dalelių.



1.5 pav. (a) Kelių dalelių optimalaus taško paieška (b) Globalaus sprendinio radimas

Paimkime pradinį spiečių iš n_p dalelių:

$$S^{(0)} = \left(x_1^{(0)}, x_2^{(0)}, \dots, x_{n_p}^{(0)}\right) \quad \textbf{(1.40)}$$

Spiečiaus dalelių pozicijos gali būti parenkamos atsitiktinai leistinoje srityje arba sistemingai, remiantis ankstesnių uždavinio sprendimų rezultatais ar kita informacija. Kiekvienai spiečiaus dalelei

$$x_i^{(0)} = \left(x_{i1}^{(0)}, x_{i2}^{(0)}, \dots, x_{id}^{(0)}\right), \quad \textbf{(1.41)}$$

yra žinomas jos judėjimo greitis

$$v_i^{(0)} = \left(v_{i1}^{(0)}, v_{i2}^{(0)}, \dots, v_{id}^{(0)}\right), \quad \textbf{(1.42)}$$

iš čia geriausia asmeninė pozicija $Pbest_i$ ir geriausia visų spiečiaus dalelių pozicija $Pbest_{gbest}$ $(i = 1,2, \dots, N)$.

k-toji spiečiaus būsena S^k sudaroma keičiant dalelių pozicijas $k-1$-joje spiečiaus būsenoje $S^{(k-1)}$:

$$v_i^{(k)} = \omega v_i^{(k-1)} + c_1 r_1 \left(Pbest_i - x_i^{(k-1)}\right) + c_2 r_2 (Pbest_{gbest} - x_i^{(k-1)}) \quad \textbf{(1.43)}$$

$$x_i^{(k)} = x_i^{(k-1)} + v_i^{(k)} \quad \textbf{(1.44)}$$

$$i = 1, \dots, N$$

Čia ω – inercijos momentas, r_1 ir r_2 – atsitiktiniai skaičiai iš intervalo [0,1]. Konstantos c_1 ir c_2 yra atitinkamai pasikliautinieji (angl. *self-confidence*) koeficientai – didesnė c_1 reikšmė reiškia didesnį pasikliovimą geriausia asmenine pozicija, o didesnė c_2 reikšmė – didesnį pasikliautinumą geriausia pozicija, rasta visų spiečiaus dalelių [10]. Literatūroje [10,15] siūloma naudoti $c_1 = c_2 = 2$, kad sandaugų $c_1 r_1$ ir $c_2 r_2$ vidurkiai būtų lygūs 1.

Kai nauja spiečiaus būsena $S^{(k)}$ sudaryta, apskaičiuojamos visų dalelių tikslo funkcijos reikšmės ir atnaujinama informacija apie geriausias asmenines pozicijas $Pbest_i$ ir geriausią visų spiečiaus dalelių poziciją $Pbest_{gbest}$.

Analogiškai keičiant spiečiaus $S^{(k)}$ dalelių pozicijas sudaroma kita spiečiaus būsena $S^{(k+1)}$. Taip gaunamas iteracinis procesas, kuris tęsiamas kol gaunamas optimalus sprendinys.

Dalelių spiečiaus optimizavimo privalumas – lengva realizacija ir mažai keičiamų parametrų, funkcijų, be to šis algoritmas geba optimizuoti funkcijas su daug lokalių minimumų.

Dalelių spiečiaus algoritmas [12]:

1. Nustatome parametrus $n_p, n_G, \delta, c_1\ ir\ c_2$
2. Paimame daleles $P_i^{(0)}$ ir greitį $v_i^{(0)}$, $i = 1, \dots, n_p$
3. Įvertiname tikslo funkciją $F_i = (P_i^{(0)})$, $i = 1, \dots, n_p$
4. $Pbest = P^{(0)}$, $Fbest = F$, $Gbest = \min_i(F_i)$, $gbest = \text{argmin}_i(F_i)$
5. **for** $k = 1$ iki n_G **do**
6. **for** $i = 1$ iki n_p **do**
7. $\Delta v_i = c_1 \times \zeta_1 \times \left(Pbest_i - P_i^{(k-1)}\right) + c_2 \times \zeta_2 \times \left(Pbest_{gbest} - P_i^{(k-1)}\right)$
8. $v_i^{(k)} = \delta v^{(k-1)} + \Delta v_i$
9. $P_i^{(k)} = P_i^{(k-1)} + v_i^{(k)}$
10. **end for**
11. Įvertiname tikslo funkciją $F_i = F\left(P_i^{(k)}\right)$, $i = 1, \dots, n_p$
12. **for** $i = 1$ iki n_p **do**
13. **if** $F_i < Fbest_i$ **then** $Pbest_i = P_i^{(k)}$ ir $Fbest_i = F_i$
14. **if** $F_i < Gbest$ **then** $Gbest = F_i$ ir $gbest = i$
15. **end for**
16. **end for**
17. sprendinys $P_{.,gbest}^{n_G}$

n_G – generavimų skaičius, P – populiacija ($p \times n_p$ dydžio matrica), F – tikslo funkcija, F_i – tikslo funkcijos reikšmė susijus su i-tuoju sprendiniu, ζ – atsitiktinis kintamasis su vienodu pasiskirstymu [0 1].

1.6.2. PSO metodo taikymas pasirinkimo sandorių įkainojimui

Viena iš svarbiausių mokslinių tyrimų užduočių finansų skaičiavime yra atpažinti geriausią duoto pasirinkimo sandorio įvykdymo laiką, maksimizuojant jo pelną. Metodai, kurie aprašyti anksčiau, gali pasiekti šį tikslą. Šiame darbe siekiame išnagrinėti algoritmą, kuris yra aprašytas remiantis iš natūralios gamtos, tokios kaip žuvų būrys, paukščių pulkas ar kitaip tariant spiečiumi. Šiuo algoritmu tikimasi apskaičiuoti optimalias pasirinkimo sandorio reikšmes duotoje sprendimų aibėje su priimtu laiku, kuris yra geresnis negu kiti ankstesni metodai.

1.7. Modelio parametrų kalibravimas

Kalibruojant opciono įkainojimo modelius į rinkos kainas dažnai susiduriama su optimizavimo problemomis, kurios negali būti sprendžiamos taikant standartinius modelius, paremtus gradientu. Šiame darbe nagrinėjame du modelius: Heston stochastinio kintamumo modelį ir Bates modelį, kuris taip pat turi šuoliukus. Aptariama kaip įkainoti opcionus remiantis šiais modeliais ir kaip kalibruoti šių modelių parametrus su euristiniu PSO metodu.

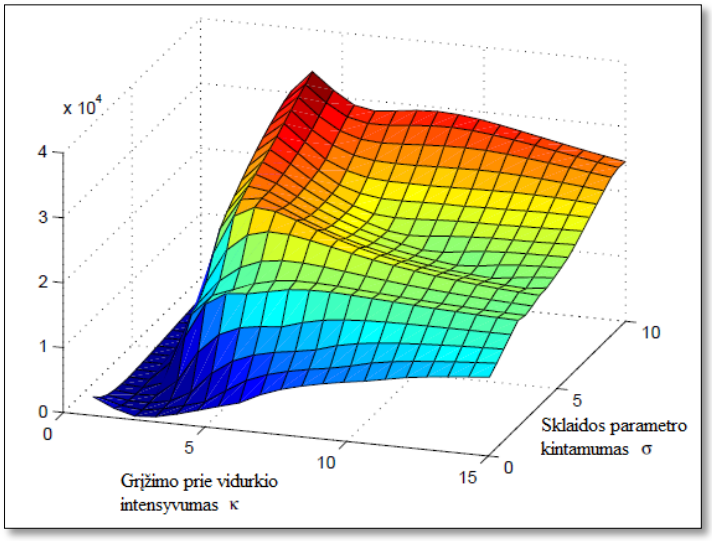
Pasirinkimo sandorio įkainojimo modelio kalibravimas reiškia, kad reikia surasti tokius parametrus, su kuriais modelio kainos atitinka rinkos kainas, vadovaujantis tokia optimizavimo uždavinio forma:

$$\min \sum_{i=1}^{M} \frac{\left|C_i^{modelio} - C_i^{rinkos}\right|}{C_i^{rinkos}} \qquad \textbf{(1.45)}$$

čia M – rinkos (akcijų) kainų skaičius. Taip pat galime patikslinti absoliučias paklaidas, naudojant kvadratų vertes vietoje absoliučių verčių arba įvedant svorinę sistemą. Tikslo funkcijos parinkimas priklauso nuo pritaikymo. Galiausiai, tai praktinis klausimas, kokią geriausiai parinkti tikslo funkciją. Čia mus domina skaitiniai aspektai, todėl naudojame tikslo funkciją (1.45). Grafike (1.6 pav.) yra vaizduojama Heston modelio tikslo funkcijos pavyzdys (ant logaritminės skalės), kai keičiami du parametrai – grįžimo prie vidurkio intensyvumas κ ir sklaidos parametras σ, o kiti parametrai yra fiksuoti.

Uždavinys nėra iškilasis ir standartiniai metodai, paremti tikslo funkcijos išvestinėmis, gali būti neveiksmingi. Todėl šiame darbe, kad išspręstume šį uždavinį, įgyvendinamas euristinis metodas – dalelių spiečiaus optimizavimo metodas.

Taigi kalibravimo tikslas yra rasti tokį parametrų rinkinį, kuris minimizuoja atstumą tarp modelio prognozuojamų ir stebėtų rinkos kainų. Iš ankstesnio skyrelio žinome, jog naudojant rizikos neutralios tikimybės įvertį, Heston modelis turi penkis nežinomus parametrus $\Omega = \{\upsilon_0, \theta, \rho, \kappa, \sigma\}$.



1.6 pav. Hestono modelio paklaidų paviršius keičiant du parametrus.

Įvertindami duomenis pagal formulę (1.45), įkainojame ne tik vieną pasirinkimo sandorį, tačiau visą matricą skirtingų pasirinkimo sandorių įvykdymo kainų su skirtingais sandorio laikotarpiais. Tačiau duotiems parametrų rinkiniams, kurie aprašo pagrindinį modelio procesą, charakteristinė funkcija ϕ priklauso tik nuo sandorio laikotarpio, o ne nuo įvykdymo kainos. Tai rodo, kad pirminiu apdorojimu gali būti pasiektas greičio pagerinimas šiomis ϕ sąlygomis, kurios yra konstantos duotiems laikotarpiams. Tuomet suskaičiuojamos pasirinkimo sandorių kainos su tam tikru laikotarpiu visoms įvykdymo kainoms. Algoritmas pateikiamas žemiau [12].

Kainų skaičiavimo duotam paviršiui algoritmas:

1. Nustatome parametrus, T = įvykdymo laikus, X = įvykdymo kainas
2. **for** $t \in T$ **do**
3. apskaičiuojama charakteristinė funkcija ϕ
4. **for** $x \in X$ **do**
5. apskaičiuojama įvykdymo kaina x, įvykdymo laikas t
6. **end for**
7. **end for**
8. apskaičiuojama tikslo funkcija.

2. Tiriamoji dalis

Šioje dalyje pateikiami pasirinkimo sandorių įkainojimo, dvejais skirtingais modeliais ir jų optimizavimo PSO algoritmu, tyrimo duomenys su imitaciniais ir realiais duomenimis, rezultatai ir jų analizė.

2.1. Tyrimo duomenys

Norint patikrinti atliekamą dalelių spiečiaus optimizavimo efektyvumą pasirinkimo sandorių taikyme šiame darbe, reikia turėti tam tikrus pradinius duomenis. Tam tikslui, sugeneruosime dirbtinius duomenų rinkinius ir juos palyginsime su modelio duomenimis.

Parenkame dabartinę akcijos kainą $S_0 = 100$ su nerizikinga pelno norma $r = 2\%$, taip pat sakykime, jog nėra dividendų $q = 0\%$. Keičiant įvykdymo kainas intervale nuo 80 iki 120 (žingsnis kas du) su skirtinga sandorio gyvavimo trukme metais $\tau = \{{}^{1}/_{12}, {}^{3}/_{12}, {}^{6}/_{12}, 1\}$, turėsime $21 \times 4 = 84$ skirtingų duomenų kombinacijų. Naudodamiesi Heston modeliu apskaičiuojame S_t pasirinkimo sandorių kainas parinktoms duomenų kombinacijoms su kiekvienu parametrų rinkiniu (2.1 lentelė). Taigi, paėmę pirmą rinkinį iš lentelės, t.y. $\Omega = (\upsilon_0, \theta, \rho, \text{к}, \sigma) = (0.4, 0.8, -0.4, 3.1, 0.5)$, gausime 84-ias skirtingas pasirinkimo sandorio kainas. Vadinasi, galime sakyti, jog vienos pasirinkimo sandorio kainos sugeneravimui pagal Heston modelį reikia turėti duomenų matricą dydžio 1×10, iš kurių pirmi penki stulpeliai yra finansiniai duomenys $f = (S_0, X, \tau, r, q)$, o kiti 5 stulpeliai yra Heston modelio parametrų rinkinys Ω parinktas pagal tam tikrus apribojimus (1.5 skyrelis). Šias sugeneruotas S_t kainas laikome tikromis kainomis (C_i^{rinkos}).

2.1 lentelė

Heston modelio parametrų rinkiniai

$\sqrt{\upsilon_0}$	0.4	0.4	0.4	0.5	0.3	0.7	0.8	0.9
$\sqrt{\theta}$	0.8	0.2	0.3	0.3	0.5	0.4	0.4	0.4
ρ	-0.4	-0.8	0.1	-0.6	-0.6	-0.6	-0.2	-0.6
к	3.1	3.1	3.1	0.3	2.0	3.1	2.1	1.1
σ	0.5	1.6	1.0	0.4	0.9	1.1	1.1	1.0

Tuomet apibrėžiame tikslo funkciją (1.45) pagal duotus duomenis:

$$\boldsymbol{f(\upsilon_0, \theta, \rho, \text{к}, \sigma) = \min \sum_{i=1}^{M} \frac{\left| Heston(S_i, X_i, \tau_i, r_i, q_i, \upsilon_0, \theta, \rho, \text{к}, \sigma) \; - C_i^{rinkos} \right|}{C_i^{rinkos}}} \quad \textbf{(2.1)}$$

Turime sukonstruotą tikslo funkcijos modelį, iš kurio gauname mažiausią tikslo funkcijos reikšmę arba vidutinę procentinę paklaidą.

Toliau detalizuosime kaip veikia dalelių spiečiaus algoritmas su šia tikslo funkcija.

Tarkime turime N dalelių, kurios yra pasiskirstę 5-matėje erdvėje. Kiekvienos dalelės pozicija ($Pbest_i$) yra atsitiktinis Heston parametrų rinkinys $\Omega = (\upsilon_0, \theta, \rho, \kappa, \sigma)$. Taigi turime 25-is pradinius skirtingus parametrų rinkinius arba spiečiaus daleles sprendimų erdvėje. Kiekviena dalelė turi informaciją apie dabartinę poziciją $\overrightarrow{X^i}$ ir skridimo greitį $\overrightarrow{V^i}$, t.y. kryptį į kurią skrenda dalelė. Turimus kiekvienos dalelės duomenis įstatome į (1.45) funkciją ir taip gauname 25-ias vidutines procentines paklaidas, t.y. rasta geriausia viso spiečiaus dalelės pozicija $Pbest_{gbest}$ pirmojoje iteracijoje.

Antrojoje iteracijoje, dalelė p_i skriejanti greičiu, aprašytu formulėje (1.42), į naują poziciją (tašką 5-matėje erdvėje) tikrina ar nauja jos asmeninė pozicija yra geresnė už buvusią jos poziciją, jeigu taip – tai atnaujiname jos esamą asmeninę poziciją $Pbest_i$. Taip pat atnaujiname $Pbest_{gbest}$ jeigu geriausia asmeninė pozicija yra didesnė už geriausią visų spiečiaus dalelių poziciją.

Antrame priede pateiktas pavyzdys, kaip 25-ios dalelės yra atsitiktinai pasiskirsčiusios pirmos iteracijos atveju ir 50-tos iteracijos atveju. Po kiekvienos iteracijoms gaunama 10 grafikų (C_5^2) su skirtingais dalelės koordinačių pjūviais erdvėje.

Tokiu būdu kiekvienos iteracijos metu yra atnaujinami kiekvienos dalelės duomenys – dalelės pozicija $\overrightarrow{X^i}$ (5-matis vektorius $\Omega = (\upsilon_0, \theta, \rho, \kappa, \sigma)$), $Pbest_i$ ir $Pbest_{gbest}$. Taip tikriname kokiu tikslumu dalelių spiečius atkuria dirbtinai susikurtus duomenis.

Šiame darbe, pasinaudojus Matlab programine įranga, sugeneruojame 25, 50 ir 100 dalelių spiečius po 50 iteracijų. Taip pat vienam parametrų Ω rinkiniui sugeneruojame po 10 kartų visą evoliuciją, dėl galimų tikslesnių duomenų.

Apribojame savo tyrimo duomenis, tam kad būtų palankios interpretuojamos reikšmės (1.5 skyrius). Vadinasi, reikalaujame neneigiamų nuokrypių, koreliacija tarp -1 ir 1, o kiti parametrai turi būti neneigiami. Šios sąlygos yra įgyvendinamos baudos sąlygose. Tai reiškia, jog nurodome parametrams jų ribas:

υ_0 – [0 0.9] – pradinis kintamumas

θ – [0 0.9] – sklaidos parametro vidurkis

ρ – [-1 1] – koreliacijos koeficientas

κ – [0 5] - grįžimo prie vidurkio intensyvumas

σ – [0 5] – sklaidos parametro kintamumas

λ – [0 1] – šuoliukų intensyvumas

μ_J – [-1 1] – šuoliukų vidurkis

σ_J – [0 1] – šuoliukų variacija

Tyrimas atliktas su 8-iais parametrų rinkiniais (2.1) ir (2.2) lentelės Heston ir Bates modeliams. Taip pat sugeneruojame po 10 kartų su tuo pačiu parametrų rinkiniu. Kiekvienam parametrų rinkiniui turime $8 \times 10 = 80$ rezultatų optimizavimo metodui. Kiekvienam paleidimui yra atnaujinama tikslo funkcijos reikšmė (vidutinė procentinė paklaida (1.45)) ir atitinkami parametrų įverčiai. Pastarajam apskaičiuojame absoliutines paklaidas, pavyzdžiui:

$$\boldsymbol{paklaida} = |\, \boldsymbol{įvertintas\ parametras - tikrasis\ parametras}\, | \qquad \textbf{(2.2)}$$

Europietiškojo pirkimo pasirinkimo sandorio įkainojimo modelių Matlab programos kodai yra parsisiųsti [12] (*callBatescf.m* ir *HestonCall.m*) ir perdaryti pagal reikiamas sąlygas.

Generuojame preliminarių eksperimentų skaičių, tam kad rasti tinkamas parametrų reikšmes PSO algoritmui. Šiam algoritmui svarbiausias uždavinys yra pagreitinti konvergavimą. Greitis $\overrightarrow{V^i}$ neturi būti nustatomas per didelis, tačiau inertiškumo koeficientas turi būti mažiau vieneto [15]. Tiriamiems duomenims nustatėme inercijos koeficientą $\omega = 0.7$ [15]. Maksimalus absoliutus greitis nustatytas 0.2 su korekcijos faktoriumi 2. Taigi generuojant tris rinkinius ($populiacijos\ dydis \times iteracij$ų $skaičius$) su (Intel(R) Core(TM) i5-2450M CPU at 2.50GHz with 6 GB of RAM) viena iteracija trunka apie 16 sekundžių rinkiniui (25×50), apie 24 sekundes (50×50) ir apie vieną minutę (100×50). Kuo didesnis populiacijos skaičius, tuo ilgesnis iteracijos generavimo laikas.

2.2 lentelė

Bates modelio parametrų rinkiniai

$\sqrt{v_0}$	0,4	0,4	0,4	0,5	0,3	0,7	0,8	0,9
$\sqrt{\theta}$	0,8	0,2	0,3	0,3	0,5	0,4	0,4	0,4
ρ	-0,4	-0,8	0,1	-0,6	-0,6	-0,6	-0,2	-0,6
κ	3,1	3,1	3,1	0,3	2	3,1	2,1	1,1
σ	0,5	1,6	1	0,4	0,9	1,1	1,1	1
λ	0,1	0,2	0,3	0,2	0,1	0,2	0,3	0,1
μ_J	-0,1	-0,2	-0,1	-0,2	0,1	0,1	0,2	-0,1
σ_J	0,1	0,2	0,2	0,1	0,1	0,2	0,1	0,3

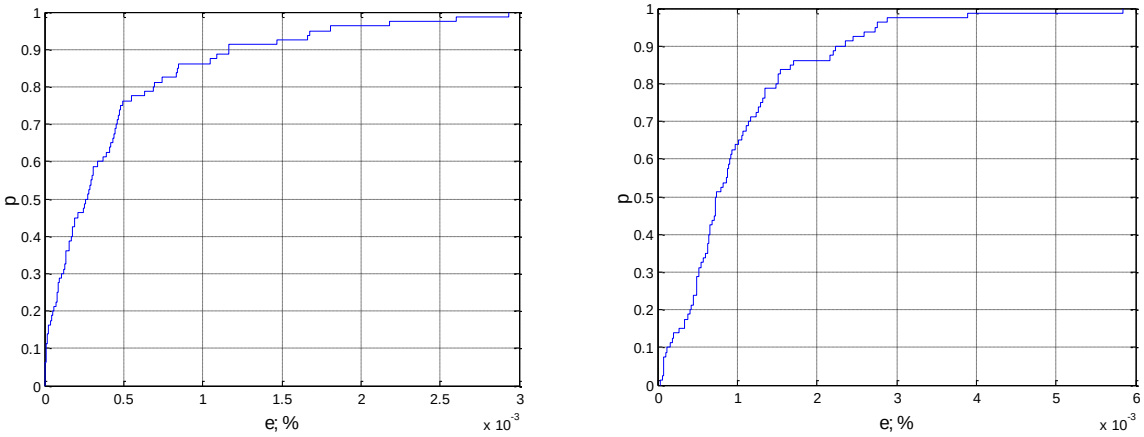
2.2. Tyrimo rezultatai

Pirmiausia aptarsime gautas tikslo funkcijos reikšmes, t.y. vidutinę procentinę įkainojimo paklaidą, Heston ir Bates modeliams. Tyrimus atlikome su visomis keturiomis pasirinkimo sandorio trukmėmis ir su kiekviena atskirai, kad galėtume palyginti dalelių spiečiaus modelio efektyvumą ir tinkamumą pasirinkimo sandorių įkainojimo modeliams.

Žemiau grafikuose vaizduojami empiriniai įkainojimo paklaidų pasiskirstymai ir jų grafikai. Heston modelio paklaidos vaizduojamos (2.1-2.8) grafikuose, o Bates modelio paklaidos (2.9-2.15) grafikuose.

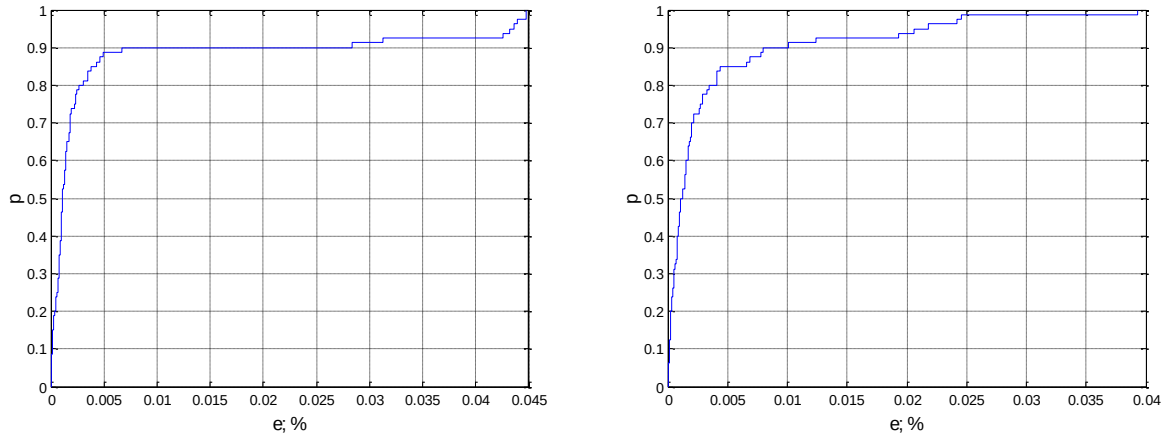
25-ių dalelių spiečiaus pasiskirstymai pavaizduoti (2.1-2.3) grafikuose, kur kiekviena iš jų pajuda 50 kartų per vieną generavimą (generavimų yra po 10) ir taip su 8-iais Heston parametrų rinkiniais (2.1 lentelė), t.y. 8-mis skirtingomis sugeneruotomis "tikromis" pasirinkimo sandorio kainomis. Galime pastebėti, jog imant kiekvieną laikotarpį atskirai, didžiausios paklaidos gaunamos, esant ilgesniam laikotarpiui. Atitinkamai, kai $T = 6/12$, paklaidos neviršija 4.48%, o kai $T = 1$, ne daugiau 3.93%.

Mažesnės paklaidos gaunamos, kai $T = 3/12$, tuomet jos neviršija 0,58%. Matome, kai $T = 1/12$ gaunamos didžiausias tikslumas lyginant su kitais laikotarpiais. Paklaidos neviršija 0,29%.



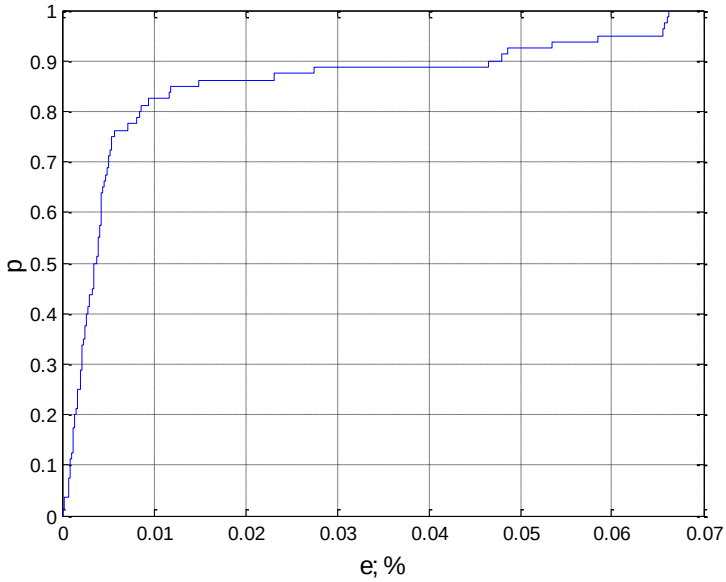
2.1 pav. Heston modelis: PSO paklaidų pasiskirstymas, kai $n_p = 25, iter = 50, T = \frac{1}{12}$ kairėje ir $T = \frac{3}{12}$ dešinėje.

Taip pat galime matyti, jog pirmame (2.1) grafike su 90% tikimybe, paklaidos yra nedidesnės nei 0.1%. O antrame grafike, 90% paklaidų sudaro iki 0.22% paklaidos.



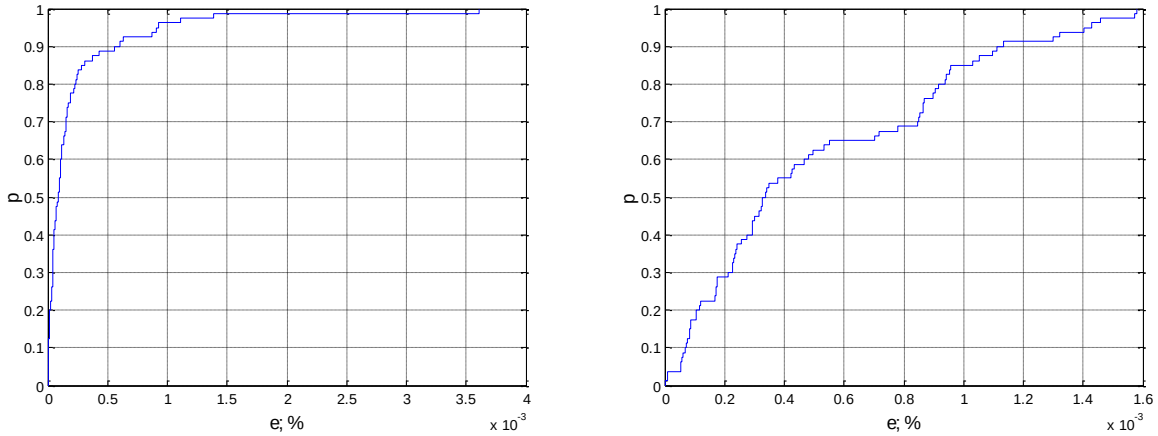
2.2 pav. Heston modelis: PSO paklaidų pasiskirstymas, kai $n_p = 25, iter = 50, T = \frac{6}{12}$ kairėje ir $T = 1$ dešinėje.

Imant visus keturis pasirinkimo sandorių laikotarpius ir generuojant „tikrų“ kainų ir modelio kainų paklaidas, gauname, jog paklaidos neviršija 6,4%.



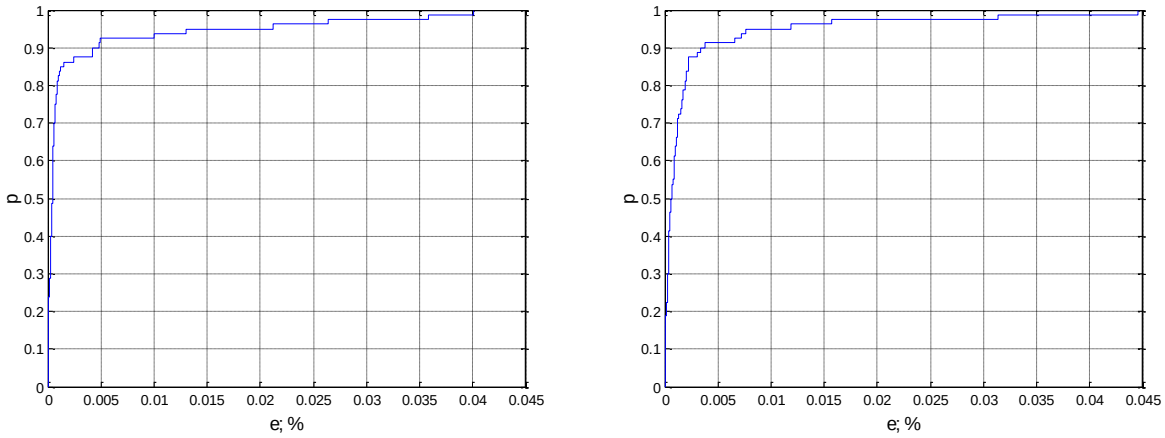
2.3 pav. Heston modelis: PSO paklaidų pasiskirstymas, kai $n_p = 25, iter = 50, T = \{\frac{1}{12}, \frac{3}{12}, \frac{6}{12}, 1\}$.

Parinkus didesnį dalelių kiekį, kai $n_p = 50$, o iteracijų skaičių paliekant tą patį, galime pastebėti, jog, kai $T = 1/12$ ir $T = 3/12$ paklaidos neviršija atitinkamai 0,36% ir 0.16% ribos (2.4 grafikas). Šiuose grafikuose taip pat galime matyti, kai $T = 1/12$, priartėjus beveik vieneto ribą, paklaida neviršija ~0,16%. Tačiau, kai $T = 3/12$ paklaidos “lėčiau“ užkyla prie vieneto ribos negu esant $T = 1/12$, o tai reiškia, jog pirmame (2.4) grafike 90% paklaidų sudaro iki 0.5%, o antrame grafike gerai matyti, jog 90% paklaidų sudaro iki 1.2% paklaidos.



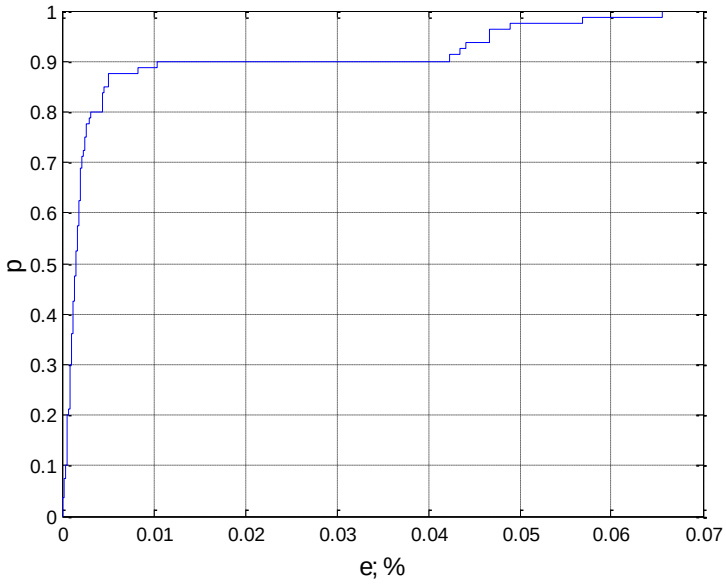
2.4 pav. Heston modelis: PSO paklaidų pasiskirstymas, kai $n_p = 50, iter = 50, T = 1/12$ kairėje ir $T = 3/12$ dešinėje.

Grafikuose (2.5) nesunku pastebėti, jog su ilgesniu pasirinkimo sandoriu laikotarpiu, kai $T = 1$, gauname 4.45% didžiausią paklaidą, o $T = 6/12$ paklaidos neviršija 4,01%. Galime nesunkiai pastebėti, jog paklaidos yra labai panašios, nes kreivės ties 90% riba turi iki 0.5% paklaidų.



2.5 pav. Heston modelis: PSO paklaidų pasiskirstymas, kai $n_p = 50, iter = 50, T = 6/12$ kairėje ir $T = 1$ dešinėje.

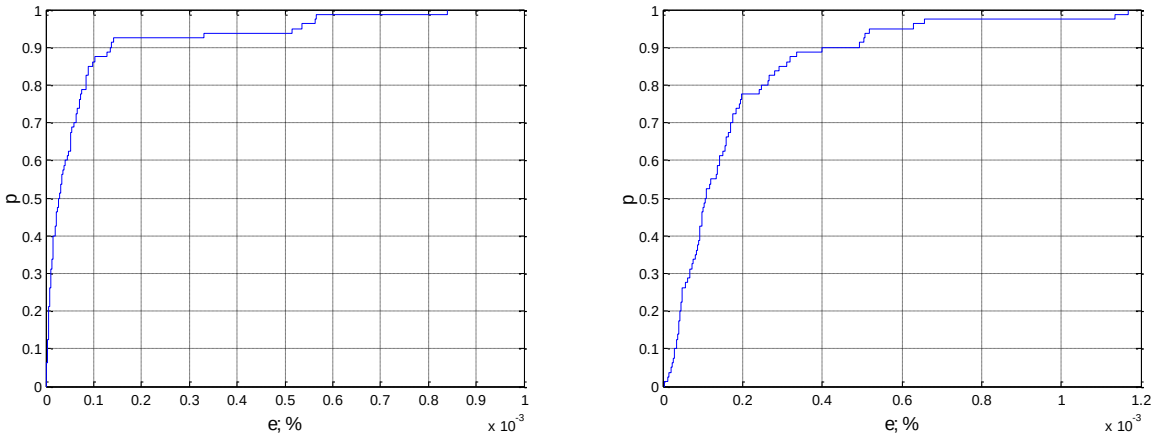
Nubraižius paklaidų grafiką generuojant duomenis su visais keturiais pasirinkimo sandorio laikotarpiais, grafike (2.6) paklaidos neviršija 6.55%. Taip svarbu paminėti, jog 90% paklaidų sudaro iki 1% esančios paklaidos.



2.6 pav. Heston modelis: PSO paklaidų pasiskirstymas, kai $n_p = 50, iter = 50, T = \{\frac{1}{12}, \frac{3}{12}, \frac{6}{12}, 1\}$.

Taip pat lyginant grafikuose (2.3) ir (2.6) esančias paklaidų kreives, matome, jog paklaidų kreivė greičiau „užkyla“ grafike (2.6) (generuojant su 50 dalelių), negu (2.3) grafike (25 dalelės).

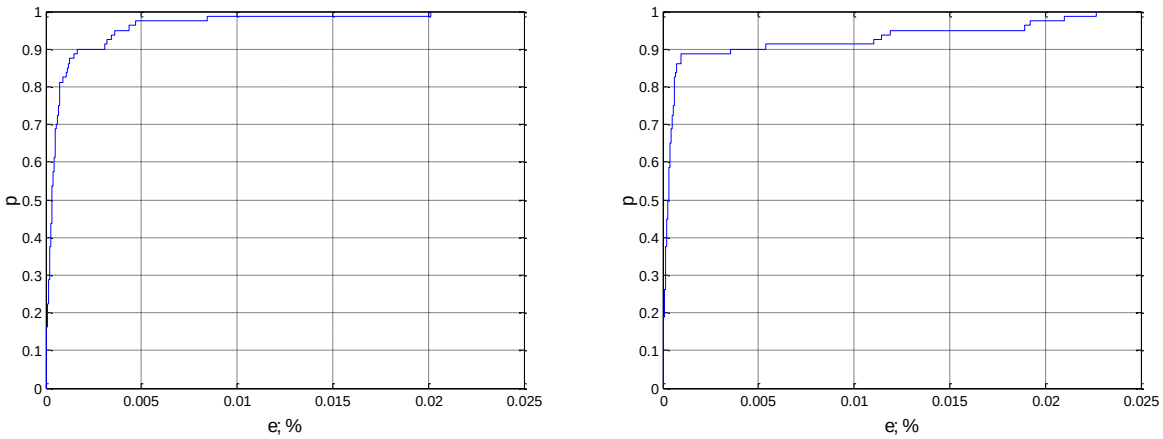
Generuojant duomenis su dar daugiau padidintu dalelių skaičiumi, kai $n_p = 100$, gauname mažesnes modelio paklaidas (2.7-2.8 grafikai).



2.7 pav. Heston modelis: PSO paklaidų pasiskirstymas, kai $n_p = 100, iter = 50, T = 1/12$ ir $T = 3/12$.

Grafike (2.7), kai $T = 1/12$, gauname didžiausią paklaidą 0,08%, o kai $T = 3/12$ paklaida neviršija 0,12%. Matome, jog pirmame (2.7) grafike apie 90% paklaidų sudaro iki 0,01% paklaidos dydžio.

Grafike (2.8) matome, kai $T = 6/12$, tai paklaida nedidesnė už 2,27%, o kai $T = 1$, tai paklaida neviršija 2.01%. Taip pat galime pastebėti, jog pusės metų duomenims kreivė "užkyla" link 90% ribos greičiau, nei metų duomenims.



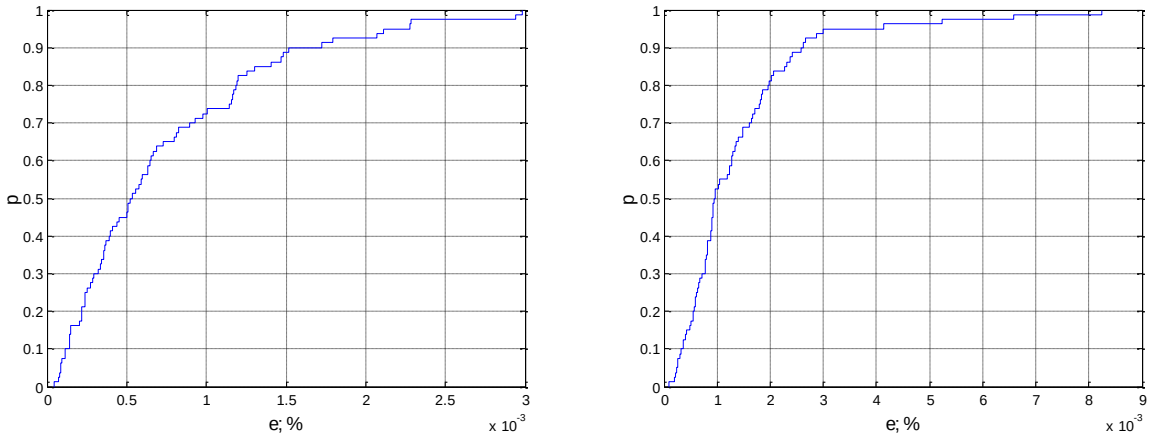
2.8 pav. Heston modelis: PSO paklaidų pasiskirstymas, kai $n_p = 100, iter = 50, T = 6/12$ kairėje ir $T = 1$ dešinėje.

Optimizuojant Heston įkainojimo modelį su skirtingu dalelių n_p kiekiu, galime pastebėti tendenciją – kuo parinktas didesnis dalelių skaičius, tuo modelio paklaidos yra mažesnės. Taip pat galime pastebėti, jog atlikus skaičiavimus su mažesniu pasirinkimo sandorio įvykdymo laiku, gaunamas tikslesnis modelis, t.y. mažesnės modelio paklaidos negu skaičiuojant ilgesniems sandorio laikotarpiams.

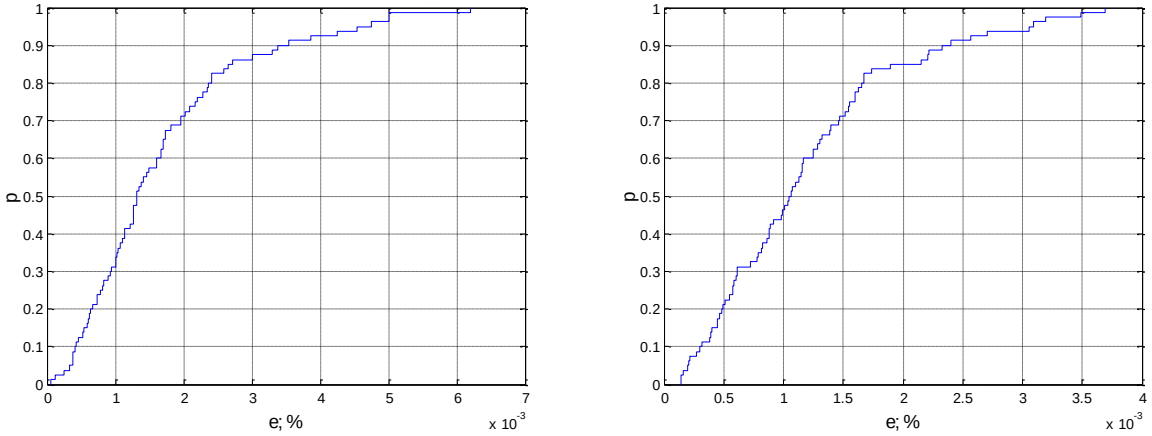
Įvertinant dalelių spiečiaus optimizacijos tinkamumą su Bates pasirinkimo sandorio įkainojimo modeliu, apibrėžiame tikslo funkciją taip pat kaip Heston įkainojimo modeliui (2.1 formulė), tiktais su Bates modelio parametrais:

$$f(\upsilon_0, \theta, \rho, \kappa, \sigma, \lambda, \mu_J, \sigma_J) = \min \sum_{i=1}^{M} \frac{\left| Bates(S_i, X_i, \tau_i, r_i, q_i, \upsilon_0, \theta, \rho, \kappa, \sigma, \lambda, \mu_J, \sigma_J) \; - C_i^{rinkos} \right|}{C_i^{rinkos}}$$

Turime sukonstruotą tikslo funkcijos modelį, su kuriuo minimizuojame Bates pasirinkimo sandorio įkainojimo modelio paklaidas dalelių spiečiaus algoritmu.



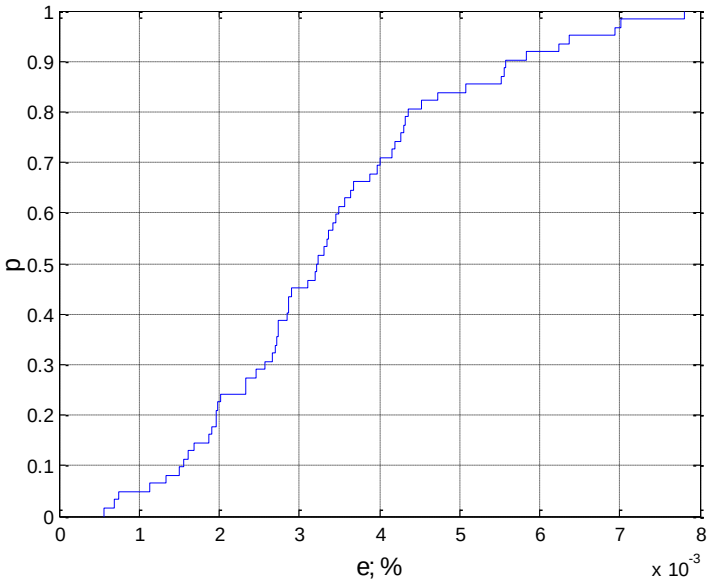
2.9 pav. Bates modelis: PSO paklaidų pasiskirstymas, kai $n_p = 25, iter = 50, T = \frac{1}{12}$ kairėje ir $T = \frac{3}{12}$ dešinėje.



2.10 pav. Bates modelis: PSO paklaidų pasiskirstymas, kai $n_p = 25, iter = 50, T = \frac{6}{12}$ kairėje ir $T = 1$ dešinėje.

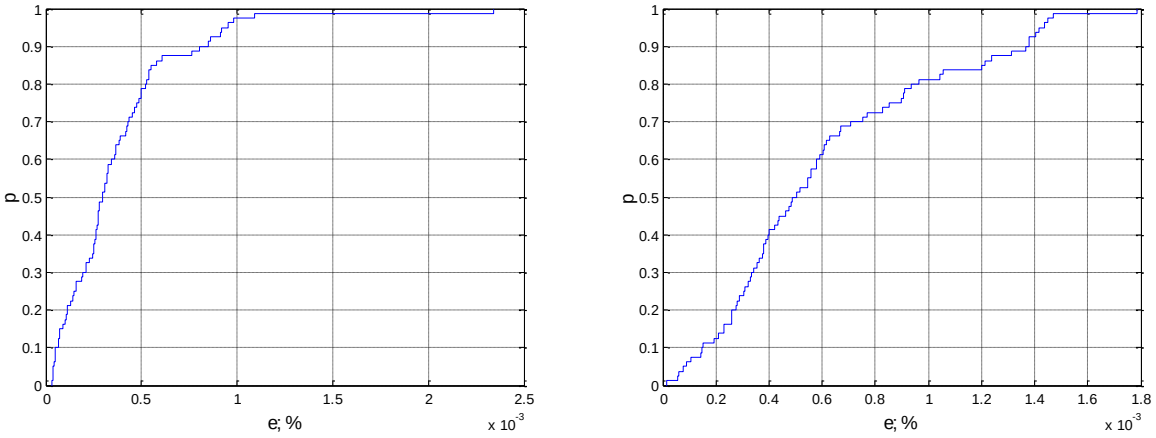
Generuojant 25-ių dalelių spiečių, gaunami įdomūs rezultatai (2.9-2.10 grafikai). Kai $T = 1/12$ ir $T = 3/12$, gauname paklaidas ne didesnes nei atitinkamai 0.29% ir 0.82%. Tačiau matome, jog kreivės gan tolygiai pasiskirsčiusios, tai reiškia, jog pirmame grafike (2.9) apie 90% paklaidų siekia 0.17% paklaidos dydžio (apie puse visų paklaidų dydžio), tuo tarpu antrame grafike 90% riba pasiekiama su nedidesnėmis nei 0.28% paklaidos.

Su ilgesniais pasirinkimo sandorio laikotarpiais gaunamos taip pat nedidelės paklaidos. Kai $T = 6/12$, tai 0.62%, o kai $T = 1$, tuomet paklaidos neviršija 0.37%. Tačiau 90% paklaidų sudaro atitinkamai pirmamame (2.10) grafike iki 0.4% paklaidos, antrame iki 0.25% paklaidos.



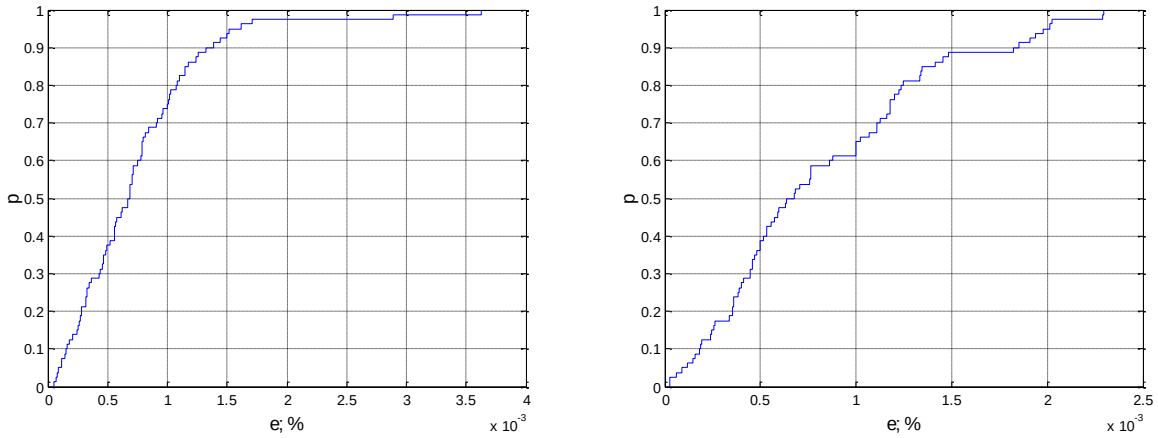
2.11 pav. Bates modelis: PSO paklaidų pasiskirstymas, kai $n_p = 25, iter = 50, T = \{\frac{1}{12}, \frac{3}{12}, \frac{6}{12}, 1\}$

Su visais keturiais laikotarpiais grafike (2.11), paklaidos neviršija 0.78% ribos. Galime pastebėti, jog paklaidų kreivė negreitai kyla link vieneto ribos, gana tolygiai pasiskirsčiusios, tačiau paklaidos yra nedidelės.



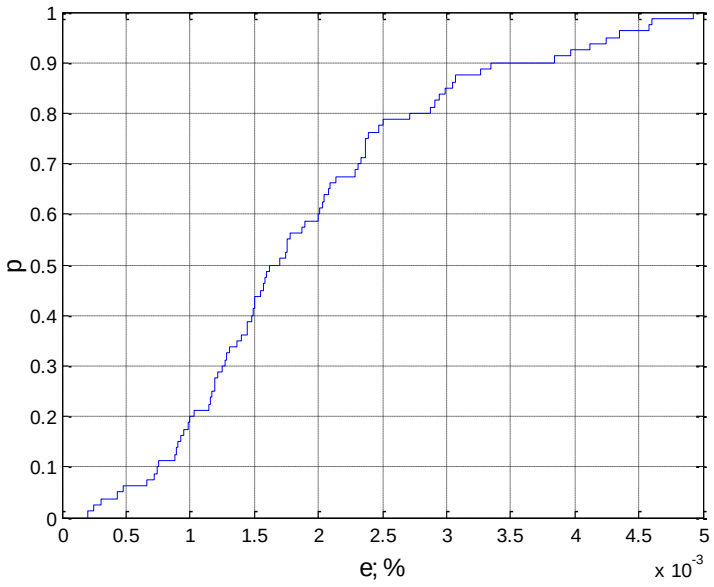
2.12 pav. Bates modelis: PSO paklaidų pasiskirstymas, kai $n_p = 50, iter = 50, T = \frac{1}{12}$ kairėje ir $T = \frac{3}{12}$ dešinėje.

Padidinus dalelių spiečiaus kiekį iki 50-ties dalelių, pirmame (2.12) grafike gauname ne didesnes nei 0.23% paklaidas, o antrame grafike paklaidos neviršija 0.18% ribos. Aiškiai matome, jog 90% paklaidų riba pasiekiama greičiau pirmame grafike negu antrame.



2.13 pav. Bates modelis: PSO paklaidų pasiskirstymas, kai $n_p = 50, iter = 50, T = \frac{6}{12}$ kairėje ir $T = 1$ dešinėje.

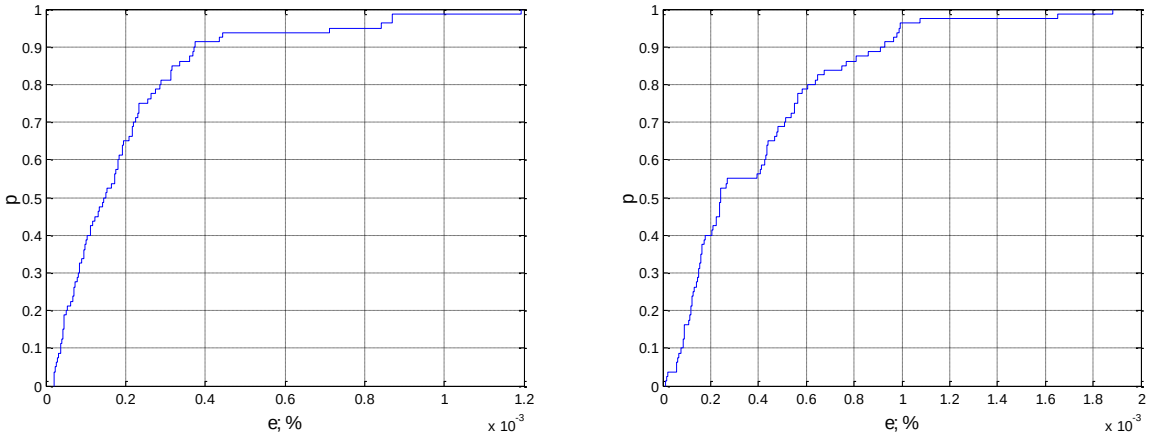
Grafike (2.13) matome, jog paklaidų kreivės pamažu „užkyla“ iki vieneto. Kai $T = 6/12$, tai paklaidos ne didesnės nei 0.36%, o esant laikotarpiui $T = 1$, neviršija 0.23%. Galime pastebėti, jog abiejuose grafikuose 90% paklaidų sudaro iki 0.15% paklaidos dydžio.



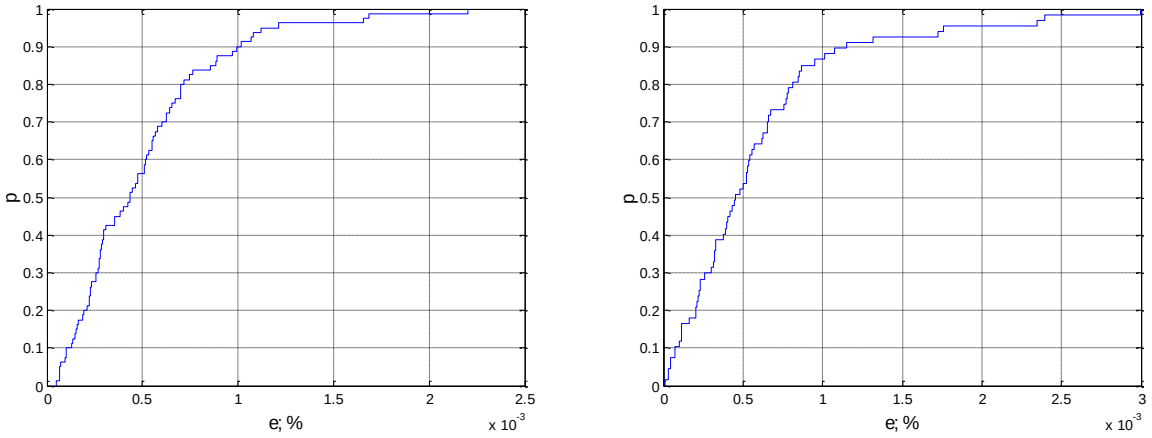
2.14 pav. Bates modelis: PSO paklaidų pasiskirstymas, kai $n_p = 50, iter = 50, T = \{\frac{1}{12}, \frac{3}{12}, \frac{6}{12}, 1\}$

Optimizuojant Bates modelį turint visus keturis pasirinkimo sandorio laikotarpius, grafike (2.14) matome, jog gaunamos taip pat nedidelės paklaidos, t.y. nedidesnės už 0.49%, su kuriomis paklaidos pasiskirsčiusios gana tolygiai.

Padidinus dalelių spiečiaus skaičių iki $n_p = 100$, grafikuose (2.15) pastebime, jog paklaidos yra didesnės antrame grafike, kai $T = 1/12$ iki 0.19%, o pirmame grafike neviršija 0.11%. Tačiau 90% paklaidų sudaro iki 0.04% pirmame, o antrame, iki 0.08%.



2.15 pav. Bates modelis: PSO paklaidų pasiskirstymas, kai $n_p = 100, iter = 50, T = \frac{1}{12}$ kairėje ir $T = \frac{3}{12}$ dešinėje.



2.16 pav. Bates modelis: PSO paklaidų pasiskirstymas, kai $n_p = 100, iter = 50, T = \frac{6}{12}$ kairėje ir $T = 1$ dešinėje.

Optimizuojant Bates įkainojimo modelį su skirtingu dalelių n_p kiekiu, galime įžvelgti panašią tendenciją kaip ir Heston įkainojimo modeliui – kuo parinktas didesnis dalelių skaičius, tuo modelio paklaidos yra mažesnės. Taip pat, priešingai negu Heston modeliui, Bates modeliui su ilgesniais pasirinkimo sandorio laikotarpiais gauname panašiai mažas paklaidas kaip ir su trumpesniais laikotarpiais. Tačiau Bates modelio apie 90% paklaidų dažniausiai yra tolygiai pasiskirsčiusios, o Heston modelis turi tendenciją išlaikyti 90% savo mažas paklaidas, o likusieji 10% paklaidų reikšmių yra artimos didžiausiai paklaidai.

2.3. Tiriamasis pavyzdys

Pasirinkimo sandorio duomenų rinkinys, kurį naudojame dalelių spiečiaus optimizavimo tinkamumui Heston ir Bates įkainojimo modeliams patikrinti, pateiktas 1-ame priede. Naudojant apribojimus, nurodytus 1.5 skyrelyje, dalelių spiečiaus optimizavimo algoritmo įgyvendinimas yra pateiktas 3 priede (*run_optim_market_bates.m* ir *run_optim_market_heston.m*).

Atlikus 10 paleidimų kiekvienam įkainojimo modeliui, gauname tikslo funkcijos reikšmes. Duomenų rinkiniui (1 Priedas, 1 lentelė) optimizuojant dalelių spiečiaus algoritmu, gauname tokias geriausias Heston ir Bates tikslo funkcijos reikšmes, atitinkamai suapvalinus yra $0{,}00332$ ir $0{,}00323$ su parametrų rinkiniais, nurodytais (2.3) lentelėje.

2.3 lentelė

Geriausi modelių parametrų rinkiniai (25 dalelės, 50 iteracijų)

	υ_0	θ	ρ	к	σ
Heston	0.0673321588	0.0943401902	-0.284301909	3.12828376	0.855432714
Bates	0.0634827081	0.0877208988	-0.333980186	3.16895001	0.843968335
	λ	μ_J	σ_J		
	0.0848809892	0.127174772	0.0674759899		

2.4 lentelė

Geriausi modelių parametrų rinkiniai (50 dalelių, 50 iteracijų)

	υ_0	θ	ρ	к	σ
Heston	0.0658587674	0.0917934733	-0.301483613	3.8420175	0.90596313
Bates	0.0589022448	0.0817863129	-0.363542039	3.67943684	0.805595773
	λ	μ_J	σ_J		
	0.168597707	0.02352433	0.0576512076		

Naudojant šiuos gautus geriausius parametrų rinkinius (2.3 lentelė), Heston ir Bates modelių prognozuojamos reikšmės ir jų palyginimas su rinkos kainomis parodytas lentelėje (2.5). Kaip matome iš šios lentelės, Heston ir Bates modelių reikšmės gana gerai atkuria rinkos kainas. Spredas (angl. *spread*) – tai mokestis už galimybę prekiauti. Vadinasi, galimybė bus suteikiama, jeigu paklaidos pateks į šį intervalą (spredą). Heston modelio 2 iš 24 reikšmės iškrenta iš stebimo spredo pasiskirstymo. Bates modelio tik 1 iš 24 reikšmių iškrenta iš spredo intervalo. Taip pat, kada vertinamas priėmimo/atmetimo sąlygos kriterijus [3], Heston modelio vidutinis paklaidų atstumas yra $0{,}00741$, kuris turi būti mažesnis už spredo vidutinį absoliutų nuokrypį, kuris šiuo atveju yra $0{,}02448$. Bates modelio vidutinis paklaidų atstumas yra $0{,}00720 < 0{,}02448$. Vadinasi, dalelių spiečiaus sprendimas yra priimamas pagal šį kriterijų.

2.5 lentelė

Realių rinkos kainų ir modelio kainų palyginimas (25 dalelės, 50 iteracijų)

Pasirinkimo sandorio id.	Rinkos kaina	Heston kaina	$\lvert C^{modelio} - C^{rinkos} \rvert$	Ar patenka į spredą?	Bates kaina	$\lvert C^{modelio} - C^{rinkos} \rvert$	Ar patenka į spredą?
1	3.150	3.154	0.00449	TAIP	3.153	0.00331	TAIP
2	2.170	2.169	0.00139	TAIP	2.168	0.00158	TAIP
3	1.240	1.240	0.00033	TAIP	1.241	0.00084	TAIP
4	0.475	0.508	0.03316	ne	0.505	0.02969	TAIP
5	0.115	0.130	0.01522	ne	0.130	0.01468	ne
6	0.035	0.023	0.01157	TAIP	0.030	0.00453	TAIP
7	3.200	3.176	0.02432	TAIP	3.175	0.02482	TAIP
8	2.230	2.220	0.01034	TAIP	2.220	0.00972	TAIP
9	1.355	1.348	0.00741	TAIP	1.348	0.00735	TAIP
10	0.655	0.659	0.00403	TAIP	0.656	0.00121	TAIP
11	0.250	0.250	0.00049	TAIP	0.249	0.00075	TAIP
12	0.085	0.081	0.00449	TAIP	0.087	0.00244	TAIP
13	3.250	3.250	0.00007	TAIP	3.250	0.00001	TAIP
14	2.350	2.353	0.00279	TAIP	2.354	0.00353	TAIP
15	1.555	1.555	0.00003	TAIP	1.555	0.00005	TAIP
16	0.910	0.916	0.00602	TAIP	0.915	0.00483	TAIP
17	0.475	0.481	0.00586	TAIP	0.482	0.00691	TAIP
18	0.225	0.235	0.00967	TAIP	0.242	0.01661	TAIP
19	3.300	3.303	0.00339	TAIP	3.303	0.00290	TAIP
20	2.455	2.432	0.02275	TAIP	2.433	0.02214	TAIP
21	1.670	1.662	0.00769	TAIP	1.662	0.00757	TAIP
22	1.040	1.040	0.00039	TAIP	1.040	0.00002	TAIP
23	0.600	0.598	0.00175	TAIP	0.600	0.00010	TAIP
24	0.325	0.325	0.00021	TAIP	0.332	0.00727	TAIP

Kai generuojame duomenis su 50 dalelių, gauname tokias geriausias Heston ir Bates tikslo funkcijos reikšmes, atitinkamai suapvalinus jos įgyja reikšmes 0,00322 ir 0,00307 su parametrų rinkiniais, nurodytais (2.4) lentelėje.

2.6 lentelė

Geriausi modelių parametrų rinkiniai (100 dalelių, 50 iteracijų)

	υ_0	θ	ρ	к	σ
Heston	0.0654021035	0.0886365179	-0.312295213	4.76401076	0.979769763
Bates	0.0628497487	0.084056307	-0.218453637	3.94400571	0.979358895
	λ	μ_J	σ_J		
	0.149627913	-0.166824313	0.0091575523		

Generuojame duomenis su 100 dalelių, gauname modelių Heston ir Bates geriausias tikslo funkcijos reikšmes, atitinkamai jos įgyja reikšmes 0,00313 ir 0,00308 su parametrų rinkiniais, nurodytais (2.6) lentelėje.

2.7 lentelė

Realių pasirinkimo sandorių kainų ir modelio kainų palyginimas (50 dalelių, 50 iteracijų)

Pasirinkimo sandorio id.	Rinkos kaina	Heston kaina	$\lvert C^{modelio} - C^{rinkos} \rvert$	Ar patenka į spredą?	Bates kaina	$\lvert C^{modelio} - C^{rinkos} \rvert$	Ar patenka į spredą?
1	3.150	3.155	0.00474	TAIP	3.155	0.00487	TAIP
2	2.170	2.169	0.00093	TAIP	2.170	0.00028	TAIP
3	1.240	1.240	0.00006	TAIP	1.240	0.00015	TAIP
4	0.475	0.505	0.02989	TAIP	0.498	0.02343	TAIP
5	0.115	0.126	0.01130	ne	0.122	0.00723	TAIP
6	0.035	0.022	0.01303	TAIP	0.027	0.00776	TAIP
7	3.200	3.176	0.02389	TAIP	3.177	0.02332	TAIP
8	2.230	2.220	0.00992	TAIP	2.221	0.00860	TAIP
9	1.355	1.347	0.00800	TAIP	1.347	0.00841	TAIP
10	0.655	0.656	0.00111	TAIP	0.651	0.00378	TAIP
11	0.250	0.245	0.00490	TAIP	0.241	0.00914	TAIP
12	0.085	0.077	0.00761	TAIP	0.080	0.00476	TAIP
13	3.250	3.250	0.00001	TAIP	3.250	0.00000	TAIP
14	2.350	2.353	0.00271	TAIP	2.354	0.00378	TAIP
15	1.555	1.555	0.00005	TAIP	1.555	0.00024	TAIP
16	0.910	0.916	0.00557	TAIP	0.913	0.00341	TAIP
17	0.475	0.479	0.00379	TAIP	0.476	0.00098	TAIP
18	0.225	0.231	0.00599	TAIP	0.232	0.00673	TAIP
19	3.300	3.303	0.00283	TAIP	3.303	0.00256	TAIP
20	2.455	2.432	0.02299	TAIP	2.433	0.02196	TAIP
21	1.670	1.663	0.00719	TAIP	1.663	0.00651	TAIP
22	1.040	1.041	0.00114	TAIP	1.040	0.00001	TAIP
23	0.600	0.598	0.00239	TAIP	0.596	0.00426	TAIP
24	0.325	0.322	0.00267	TAIP	0.323	0.00218	TAIP

Priėmimo kriterijus yra labai panašus abejoms lentelėms (2.7, 2.8) kaip ir lentelei (2.6). Heston modelio tik **1** iš **24** reikšmių iškrenta iš stebimo spredo pasiskirstymo. Bates modelio nei viena reikšmė neiškrenta iš spredo intervalo, vadinasi yra galimybė prekiauti su visomis duotomis pasirinkimo sandorio kombinacijomis.

2.8 lentelė

Realių pasirinkimo sandorių kainų ir modelio kainų palyginimas (100 dalelių, 50 iteracijų)

Pasirinkimo sandorio id.	Rinkos kaina	Heston kaina	$\lvert C^{modelio} - C^{rinkos} \rvert$	Ar patenka į spredą?	Bates kaina	$\lvert C^{modelio} - C^{rinkos} \rvert$	Ar patenka į spredą?
1	3.150	3.155	0.00510	TAIP	3.155	0.00482	TAIP
2	2.170	2.170	0.00001	TAIP	2.173	0.00270	TAIP
3	1.240	1.241	0.00075	TAIP	1.245	0.00531	TAIP
4	0.475	0.504	0.02853	TAIP	0.503	0.02813	TAIP
5	0.115	0.124	0.00920	TAIP	0.127	0.01151	ne
6	0.035	0.021	0.01366	TAIP	0.024	0.01062	TAIP
7	3.200	3.177	0.02316	TAIP	3.177	0.02264	TAIP
8	2.230	2.221	0.00882	TAIP	2.225	0.00546	TAIP
9	1.355	1.348	0.00739	TAIP	1.351	0.00409	TAIP
10	0.655	0.655	0.00002	TAIP	0.655	0.00037	TAIP
11	0.250	0.243	0.00724	TAIP	0.246	0.00435	TAIP
12	0.085	0.076	0.00921	TAIP	0.081	0.00372	TAIP
13	3.250	3.250	0.00000	TAIP	3.250	0.00000	TAIP
14	2.350	2.353	0.00266	TAIP	2.354	0.00434	TAIP
15	1.555	1.555	0.00000	TAIP	1.556	0.00089	TAIP
16	0.910	0.916	0.00554	TAIP	0.914	0.00444	TAIP
17	0.475	0.478	0.00279	TAIP	0.479	0.00427	TAIP
18	0.225	0.229	0.00377	TAIP	0.236	0.01060	TAIP
19	3.300	3.302	0.00191	TAIP	3.301	0.00133	TAIP
20	2.455	2.431	0.02384	TAIP	2.432	0.02323	TAIP
21	1.670	1.663	0.00750	TAIP	1.662	0.00763	TAIP
22	1.040	1.041	0.00126	TAIP	1.040	0.00012	TAIP
23	0.600	0.597	0.00295	TAIP	0.598	0.00190	TAIP
24	0.325	0.320	0.00470	TAIP	0.327	0.00175	TAIP

Generuojant spiečių su 100 dalelių, iš Heston modelio tikrintų paklaidų visos patenka į stebimo spredo pasiskirstymą. Bates modelio tik viena reikšmė iškrenta iš spredo intervalo.

Optimizuojant įvairių dydžių dalelių spiečius $(25, 50, 100)$ imant 50 iteracijų kiekvienu kartu, panaudodami realius duomenis galime pastebėti, jog abiems pasirinkimo sandorių įkainojimo modeliams (Heston ir Bates), šis optimizavimo algoritmas pateikia labai neblogus rezultatus – minimalias tikslo funkcijos reikšmes. Galime sakyti, jog reikėtų generuoti su didesniu dalelių kiekiu (mūsų atveju nuo 100 dalelių) norint gauti patikimesnius rezultatus. Įgyvendinant Matlab programinėje įrangoje, tai atima dideles laiko sąnaudas.

Išvados

- Atlikta užsienio ir Lietuvos literatūros apžvalga parodė, kad praktikoje įkainojant pasirinkimo sandorius dažnai modelių optimizavimui taikomi gradientiniai metodai. Heston ir Bates modeliams šie metodai netinka dėl lokalių tikslo funkcijos minimumo taškų, todėl vienas iš šio uždavinio sprendimo būdų yra dirbtinio intelekto globalios paieškos dalelių spiečiaus optimizavimas.
- Panaudojus Matlab programines įrangos priemones, realizuotas dalelių spiečiaus optimizavimo algoritmas, kurį panaudojame Heston ir Bates pasirinkimo sandorių įkainojimo modeliams optimizuoti. Atliktas dalelių spiečiaus optimizavimo algoritmo efektyvumo tyrimas šio uždavinio sprendimui naudojant simuliuotus ir realius rinkos duomenis.
- Svarbus aspektas yra tinkamas pasirinkimo sandorio įkainojimo modelio optimizacijai parinkimas. Iš tyrimo galime sakyti, jog norint įkainoti pasirinkimo sandorius trumpesniems laikotarpiams (iki 4 mėnesių), geresnis dalelių spiečiaus optimizacijai yra Heston modelis, o ilgesniems laikotarpiams (nuo 5 mėnesių), tinkamesnis yra Bates modelis.
- Įvertinus dalelių spiečiaus optimizavimo efektyvumą rinkos kainai prognozuoti, galime teigti, jog dalelių spiečiaus optimizavimas yra nebloga euristinė technika pasirinkimo sandorių modeliams įvertinti. Atlikus tyrimą, su simuliuotais ir realiais rinkos duomenimis, pastebėta, jog generuojant su 100 dalelių gaunamos mažesnės paklaidos negu generuojant 25-ių dalelių spiečių. Tačiau skaičiavimai užima nemažas laiko sąnaudas.

Literatūros sąrašas

1. Albrecher H., Runggaldier W. J., Schachermayer W. *Advanced Financial Modelling*, 2009.
2. Alizadeh S., Brandt M. W., Diebold F. X. *Range-Based Estimation of Stochastic Volatility Models of Exchange Rate Dynamics are More Interesting Than You Think*, 2002.
3. Crisostomo R. *An Analysis of the heston Stochastic Volatility Model: Implementation and Calibration using Matlab*, 2014.
4. Das S., Abraham A. and Konar A. *Particle Swarm Optimization and Differential Evolution Algorithms: Technical Analysis, Applications and Hybridization Perspectives*. Studies in Computational Intelligence, 2008.
5. Feller W. *Diffusion Processes in Genetics*. Princeton University, 1951.
6. Investopedijos personalas. Internetinė svetainė *http://www.traders.lt/page.php?id=309*
7. Keber C. And Schuster M. G. *Generalized ant programming in option pricing*, 2003.
8. Kennedy J. and Russell Eberhart. *Particle Swarm Optimization*. Purdue School of Engineering and technology, Indianapolis, 1995.
9. Kennedy J. And Eberthart R.C. and Shi Y. *Swarm intelligence*. San Francisco, CA, Kaufman M., 2001.
10. Kimontas J. *Opcionų įkainojimas Monte-Karlo metodu ir tyrimas/Bakalauro darbas*. Kaunas, 2007.
11. Landauskas M. ir Valakevičius E. *Pasirinkimo sandorių įkainojimo modelių tyrimas*. Kaunas, KTU, 2009.
12. M. Gilli, E. Schumann. *Calibrating Option Pricing Models with Heuristics*. University of Geneva, Department of Econometrics, and Swiss Finance Institute, 2010. Prieiga per internetą: http://comisef.eu/?q=calibrating_option_pricing_models_heuristics
13. Maslova M. *Calibration of parameters for the Heston model in the high volatility period of market*. Master‘s Thesis in Financial Mathematics. Halmstad University, 2008.
14. Mukhopadhyay S., Banerjee S. Cooperating Swarms: *A Paradigm for Collective Intelligence and its Application in Finance*. Department of Computer Science Army Institute of Management and Department of Mathematics Politecnico di Torino, 2010.
15. Prasain H. *A Parallel Particle Swarm Optimization Algorithm for Option Pricing*. Master Thesis. The University of Manitoba, Computer Science, 2010.

16. Shi, Y, Eberhart, R.C. *A modified particle swarm optimizer*. Proceedings of IEEE International Conference on Evolutionary Computation, 1998.
17. Vaidelys M. *Skaitmeninio optimizavimo metodai*. Referatas, 2014.
18. Valakevičius E. *Investicijų matematika*. Kaunas, 2002.
19. Wilmott P. *Frequently Asked Questions In Quantitatve Finance*. WIley, 2007.
20. Yang Z., Aldous D. *Geometric Brownian Motion Modeli n Finance*.

1 Priedas. Duomenų rinkiniai

1 lentelė

Duomenų rinkinys: 24 pasirinkimo sandoriai (4 laikotarpiai, 6 įvykdymo kainos). Pirkimo pasirinkimo sandoriai American Capital, Ltd (Nasdaq: ACAS). Rinkos duomenys nuo rugsėjo 16, 2013.

S_0	X	tau	r	Rinkos kaina	Bid	Ask	Spredas
13.14	10	0.099206	0.010590	3.15	3.1	3.2	0.1
13.14	11	0.099206	0.010590	2.17	2.15	2.19	0.04
13.14	12	0.099206	0.010590	1.24	1.21	1.27	0.06
13.14	13	0.099206	0.010590	0.475	0.46	0.49	0.03
13.14	14	0.099206	0.010590	0.115	0.11	0.12	0.01
13.14	15	0.099206	0.010590	0.035	0.02	0.05	0.03
13.14	10	0.178571	0.010346	3.2	3.15	3.25	0.1
13.14	11	0.178571	0.010346	2.23	2.21	2.25	0.04
13.14	12	0.178571	0.010346	1.355	1.34	1.37	0.03
13.14	13	0.178571	0.010346	0.655	0.64	0.67	0.03
13.14	14	0.178571	0.010346	0.25	0.24	0.26	0.02
13.14	15	0.178571	0.010346	0.085	0.08	0.09	0.01
13.14	10	0.357143	0.009516	3.25	3.2	3.3	0.1
13.14	11	0.357143	0.009516	2.35	2.32	2.38	0.06
13.14	12	0.357143	0.009516	1.555	1.53	1.58	0.05
13.14	13	0.357143	0.009516	0.91	0.89	0.93	0.04
13.14	14	0.357143	0.009516	0.475	0.46	0.49	0.03
13.14	15	0.357143	0.009516	0.225	0.21	0.24	0.03
13.14	10	0.456349	0.010686	3.3	3.25	3.35	0.1
13.14	11	0.456349	0.010686	2.455	2.4	2.51	0.11
13.14	12	0.456349	0.010686	1.67	1.64	1.7	0.06
13.14	13	0.456349	0.010686	1.04	1.02	1.06	0.04
13.14	14	0.456349	0.010686	0.6	0.58	0.62	0.04
13.14	15	0.456349	0.010686	0.325	0.31	0.34	0.03

2 lentelė

Modelių geriausios, blogiausios ir vidutinės paklaidos iš Gbest

	25_50		50_50		100_50	
	Heston (id)	Bates (id)	Heston (id)	Bates (id)	Heston (id)	Bates (id)
Min	0.00332 (9)	0.00323 (3)	0.00322 (9)	0.00307 (8)	0.00313 (4)	0.00308 (6)
Max	0.00808 (5)	0.01002 (7)	0.00440 (10)	0.00441 (6)	0.00379 (9)	0.00433 (7)
Vidurkis	0.00473	0.00484	0.00357	0.00356	0.00341	0.00358

3 lentelė

Modelių geriausios tikslo funkcijos reikšmės. Pilkai pažymėti Gbest minimumai ir maksimumai.

Gbest id.	25_50		50_50		100_50	
	Heston	Bates	Heston	Bates	Heston	Bates
1	0.00334	0.00486	0.00349	0.00315	0.00347	0.00374
2	0.00381	0.00406	0.00344	0.00357	0.00334	0.00327
3	0.00608	0.00323	0.00335	0.00331	0.00336	0.00426
4	0.00399	0.00539	0.00379	0.00327	0.00313	0.00319
5	0.00808	0.00406	0.00332	0.00394	0.00343	0.00325
6	0.00364	0.00340	0.00365	0.00441	0.00318	0.00308
7	0.00391	0.01002	0.00325	0.00369	0.00334	0.00433
8	0.00383	0.00347	0.00381	0.00307	0.00336	0.00387
9	0.00332	0.00390	0.00322	0.00401	0.00379	0.00313
10	0.00733	0.00599	0.00440	0.00315	0.00374	0.00365

4 lentelė

Modelių parametrų rinkiniai su didžiausia paklaida (25 dalelės, 50 iteracijų)

	υ_0	θ	ρ	к	σ
Heston	0.0700269001	0.313199287	0.113938072	0.882323347	1.83916646
Bates	0.0678463807	0.423023426	0.377548141	0.75391956	3.35281887
	λ	μ_J	σ_J		
	0.68569479	-0.01021924	0.027137782		

5 lentelė

Modelių parametrų rinkiniai su didžiausia paklaida (50 dalelių, 50 iteracijų)

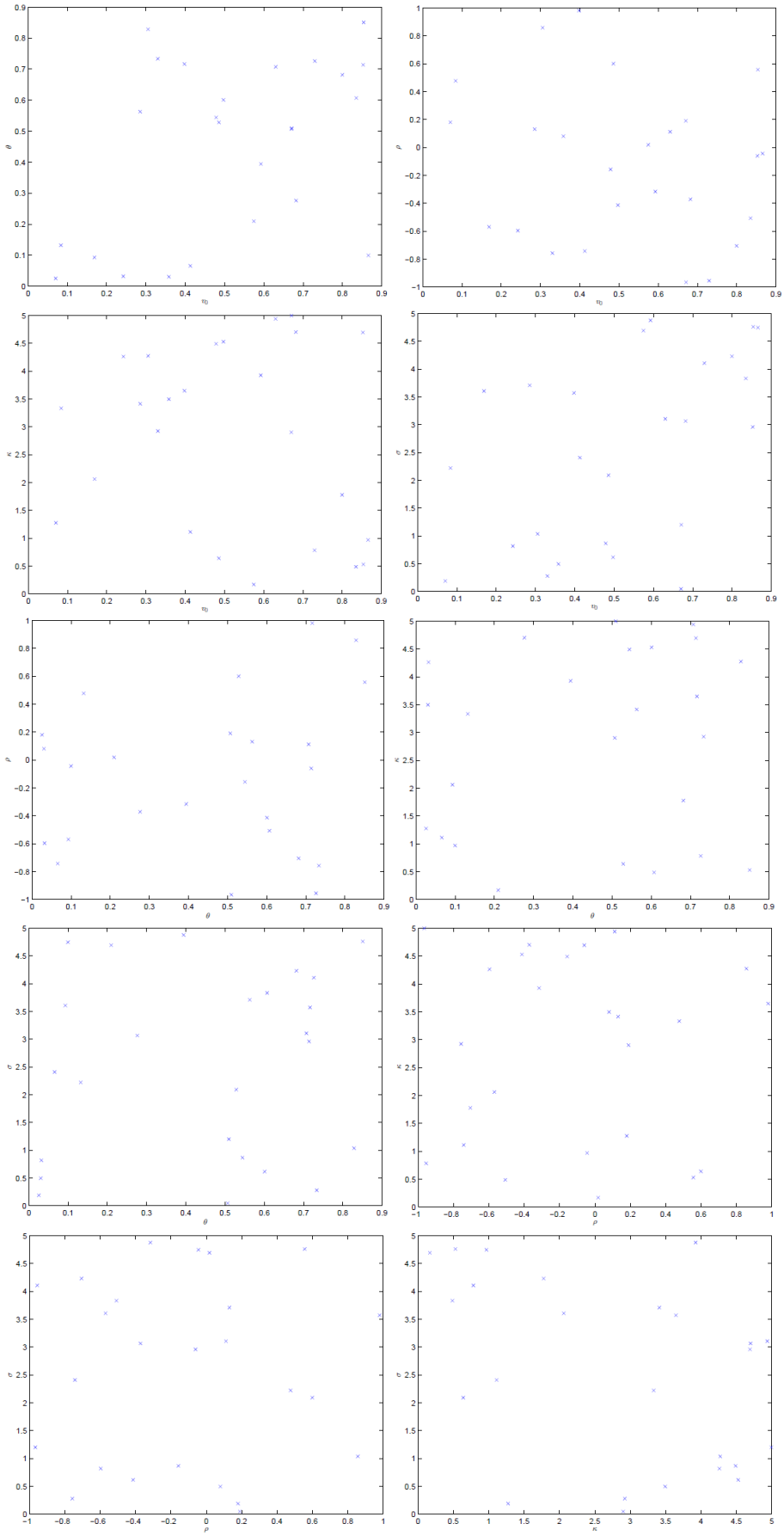
	υ_0	θ	ρ	к	σ
Heston	0.0669779678	0.476095529	-0.19662996	0.207145768	0.758703321
Bates	0.101128346	0.314598388	-0.18996499	2.42395096	3.06260645
	λ	μ_J	σ_J		
	0.140277499	-0.05179388	1.01053077		

6 lentelė

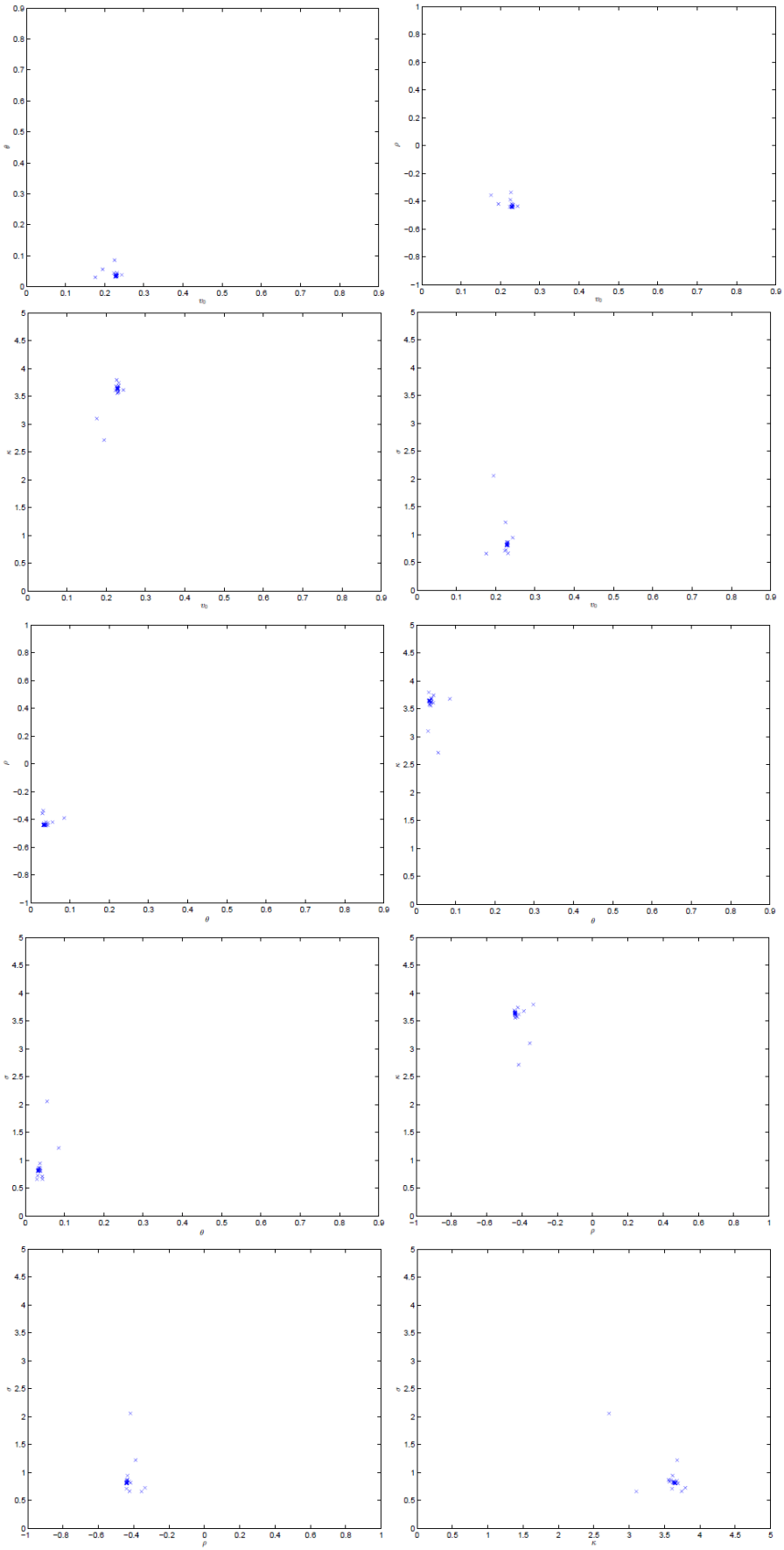
Modelių parametrų rinkiniai su didžiausia paklaida (100 dalelių, 50 iteracijų)

	υ_0	θ	ρ	к	σ
Heston	0.0646718731	0.325736541	-0.310091714	0.28618625	0.597511206
Bates	0.0693766259	0.627641102	-0.252471092	0.220409865	0.878416746
	λ	μ_J	σ_J		
	-0.034366948	0.133933083	0.492136789		

2 Priedas. Dalelės pozicijos vaizdavimas skirtingose pjūviuose



1 pav. 1-tos iteracijos 25-ių dalelių atsitiktinis pasiskirstymas apibrėžimo srityje.



2 pav. 50-tos iteracijos 25-ių dalelių konvergavims apibrėžimo srityje.

3 Priedas. Programos kodai

prices_parameters.m

```
% spot prices
Ss = 100;
% strike prices
Xs = 80:2:120;
% maturities
taus = [1/12,3/12,6/12,1];
% risk-free rates
rs = 0.2;
% dividend yields
qs = 0;
data_prices = combvec(Ss,Xs,taus,rs,qs)';
```

run_optim_heston.m

```
clc;clear all;close all;
run('prices_parameters');
idxs_params_heston = 1:8;
idxs_runs_heston = 1:10;
opt_np_niter_pairs = [
   10 3];
% v0, vT, rho, k, sigma
parameters_heston = [
    0.4^2, 0.8^2, -0.4, 3.1, 0.5;
    0.4^2, 0.2^2, -0.8, 3.1, 1.6;
    0.4^2, 0.3^2,  0.1, 3.1, 1.0;
    0.5^2, 0.3^2, -0.6, 0.3, 0.4;
    0.3^2, 0.5^2, -0.6, 2.0, 0.9;
    0.7^2, 0.4^2, -0.6, 3.1, 1.1;
    0.8^2, 0.4^2, -0.2, 2.1, 1.1;
    0.9^2, 0.4^2, -0.6, 1.1, 1.0;
    ];
init_limits_heston = [
    % v0    = initial variance
     0.0  0.9;
    % vT    = long run variance (theta in Heston's paper)
     0.0  0.9;
    % rho   = correlation
    -1.0  1.0;
    % k     = speed of mean reversion (kappa in Heston's paper)
     0.0    5;
    % sigma = vol of vol
     0.0    5;
    ];

heston_parameter_names =
{'$v_0$','$\theta$','$\rho$','$\kappa$','$\sigma$'};

idxs_params = idxs_params_heston;
```

```
model_name = 'heston';
fn_model_prices = @v_call_hestoncf;
model_parameters = parameters_heston;
model_obj_fn = @v_objfn_heston;
model_parameter_names = heston_parameter_names;
model_limits = init_limits_heston;
idxs_runs = idxs_runs_heston;

run('run_optim');
```

run_optim_bates.m

```
clc;clear all;close all;
run('prices_parameters');
idxs_params_bates = 7:8;
idxs_runs_bates = 1:7;
opt_np_niter_pairs = [
   25 50];

% v0, vT, rho, k, sigma, lambda, muJ, vJ
parameters_bates = [
    0.4^2, 0.8^2, -0.4, 3.1, 0.5, 0.1, -0.1, 0.1;
    0.4^2, 0.2^2, -0.8, 3.1, 1.6, 0.2, -0.2, 0.2;
    0.4^2, 0.3^2,  0.1, 3.1, 1.0, 0.3, -0.1, 0.2;
    0.5^2, 0.3^2, -0.6, 0.3, 0.4, 0.2, -0.2, 0.1;
    0.3^2, 0.5^2, -0.6, 2.0, 0.9, 0.1, 0.1, 0.1;
    0.7^2, 0.4^2, -0.6, 3.1, 1.1, 0.2, 0.1, 0.2;
    0.8^2, 0.4^2, -0.2, 2.1, 1.1, 0.3, 0.2, 0.1;
    0.9^2, 0.4^2, -0.6, 1.1, 1.0, 0.1, -0.1, 0.3;
];
    init_limits_bates = [
    % v0    = initial variance
     0.0  0.9;
    % vT    = long run variance (theta in Heston's paper)
     0.0  0.9;
    % rho   = correlation
    -1.0  1.0;
    % k     = speed of mean reversion (kappa in Heston's paper)
     0.0    5;
    % sigma = vol of vol
     0.0    5;
    % lambda= intensity of jumps;
     0.0  1.0;
    % muJ   = mean of jumps;
    -1.0  1.0;
    % vJ    = variance of jumps;
     0.0  1.0;
    ];

bates_parameter_names =
{'$v_0$','$\theta$','$\rho$','$\kappa$',...
    '$\sigma$','$\lambda$','$\mu_J$','$\sigma_J$'};
```

```
idxs_params = idxs_params_bates;
model_name = 'bates';
fn_model_prices = @v_call_batescf;
model_parameters = parameters_bates;
model_obj_fn = @v_objfn_bates;
model_parameter_names = bates_parameter_names;
model_limits = init_limits_bates;
idxs_runs = idxs_runs_bates;
run('run_optim');
```

pso.m

```
function [Gbest,p_gbest] = pso( ...
    objfn, limits, n_iter, n_p, ...
    inertia, correction_factor, max_abs_velocity, ...
    swarm_plots_axis_names, output_dir, ouput_fname_base)

    dlm_prec = 10;

    dim_p = size(limits,1);
    % zeros needed for initial velocity
    swarm = zeros(n_p,4,dim_p);
    for i = 1:dim_p
        % initialize particles from uniform distribution with
limits
        swarm(:,1,i) = unifrnd(limits(i,1),limits(i, 2),n_p,1);
    end
    % set the best value so far
    swarm(:,4,1) = 1000;

    tic_run = tic;
    %% The main loop of PSO
    for iter = 1:n_iter
        tic_iter = tic;
        % contrain maximum absolute velocity
        velocity = sign(swarm(:,2,:)).*...
            min(abs(swarm(:,2,:)),max_abs_velocity);
        % update particle location
        swarm(:,1,:) = swarm(:,1,:) + velocity/1.3;

        iter_p = reshape(swarm(:,1,:),[],dim_p);
        fvals = objfn(iter_p);

        % get indices of particles with better objective
function value
        idxs_better_p = fvals < swarm(:,4,1);
        % update the best value so far (Fbest)
        swarm(idxs_better_p,4,1) = fvals(idxs_better_p,1);
        % update best positions (Pbest)
        swarm(idxs_better_p,3,:) = swarm(idxs_better_p,1,:);
```

```
        % find the best function value and best particle so far
        [Gbest,gbest] = min(swarm(:,4,1));

        % velocity components
        swarm(:,2,:) = inertia*(rand(n_p,1,dim_p).*swarm(:,2,:))
+ ...
            correction_factor*(rand(n_p,1,dim_p).*...
            (swarm(:,3,:) - swarm(:,1,:))) + ...
            correction_factor*(rand(n_p,1,dim_p).*...
            (repmat(swarm(gbest,3,:),n_p,1) - swarm(:,1,:)));

        % plot the particles on each parameter axis pair

        % n_subplot_cols = 4
        % TODO: dynamic subplots n_subplot_cols x
ceil(size(axis_pairs, 2) / n_subplot_cols)

        % save particle locations
        s_iter = sprintf('%s_iter%03i', ouput_fname_base, iter);
        dir_iter = dirjoin({output_dir s_iter});
        mkdir_if_not_exist(dir_iter);

        fname_particles = dirjoin({dir_iter, [s_iter
'_swarm.csv']});
        dlmwrite(fname_particles, iter_p, 'precision',
dlm_prec);

        fname_Gbest_iter = dirjoin({dir_iter, [s_iter
'_Gbest.csv']});
        dlmwrite(fname_Gbest_iter, Gbest, 'precision',
dlm_prec);

        fname_pgbest_iter = dirjoin({dir_iter, [s_iter
'_pgbest.csv']});
        dlmwrite(fname_pgbest_iter, ...
            reshape(swarm(gbest,3,:),[],dim_p), 'precision',
dlm_prec);
     %{
        if size(swarm_plots_axis_names, 2) == size(limits, 1)
            axis_pairs = nchoosek(1:dim_p, 2);
            for i_pair = 1:size(axis_pairs,1)
                i_x = axis_pairs(i_pair,1);
                i_y = axis_pairs(i_pair,2);
                name_x = swarm_plots_axis_names(i_x);
                name_y = swarm_plots_axis_names(i_y);
                axis_ranges = reshape(limits([i_x i_y],:)', 1,
[]);
                fig = figure(i_pair);
                clf;

                plot(swarm(:, 1, i_x), swarm(:, 1, i_y), 'bx');
                axis(axis_ranges);
```

```
                xlabel(name_x, 'interpreter', 'latex');
                ylabel(name_y, 'interpreter', 'latex');

                % save particle plots
                fname_plot = dirjoin({dir_iter, ...
                    sprintf('%s_swarm_x%i_y%i.eps',s_iter, i_x,
i_y)});

                print(fig,'-dpdf',fname_plot);

                pause(0.01);
            end
        end
   %}
            disp(['iteration: ' num2str(iter)]);
         toc(tic_iter)
    end

    p_gbest = reshape(swarm(gbest,3,:),[],dim_p);
    toc(tic_run)
end
```

plot_errors.m

```
n_p = 100;
n_i = 50;
model_name = 'heston';
i_maturity = 'all';
% ------------------------------------------------------------------
dir_res = dirjoin({'..' sprintf('%s maturity %i_%i %s', ...
    i_maturity, 100, 50, 'heston')})

s_run = sprintf('np%03i_it%03i', n_p, n_iter);
dir_run = dirjoin({dir_res s_run});
mkdir_if_not_exist(dir_run);

% results\np025_it050\np025_it050_pso\
s_pso = [s_run '_pso'];
dir_run_pso = dirjoin({dir_run s_pso});
mkdir_if_not_exist(dir_run_pso);

% results\np025_it050\np025_it050_pso\np025_it050_pso_heston\
s_model = [s_pso '_' model_name];
dir_run_pso_model = dirjoin({dir_run_pso, s_model});
mkdir_if_not_exist(dir_run_pso_model);

% results\np025_it050\np025_it050_pso\np025_it050_pso_heston\...
%   np025_it050_pso_heston_Gbest.csv
fname_run_pso_model_Gbest = dirjoin(...
    {dir_run_pso_model [s_model '_Gbest.csv']});

data_Gbest = csvread(fname_run_pso_model_Gbest, 1);
```

```
errors = data_Gbest(:,3);

f_e_Gbest=figure(3);
set(f_e_Gbest,'defaulttextinterpreter','latex');
cdfplot(errors)
%xlabel('$\frac{1}{M}\sum_{i=1}^{M}\frac{|C_i^{model} -
C_i^{market}|}{C_i^{market}}$','FontSize', 13);
xlabel('$e, \%$','FontSize', 13);
ylabel('$p$','FontSize', 13);
title('');
set(f_e_Gbest,'PaperUnits','inches','PaperPosition',[0 0 8 4])
print(f_e_Gbest,'-
dmeta',dirjoin({'..','grafikai',sprintf('%s_errors_m%s.emf',s_mo
del,i_maturity)}))
```

objfn_surface.m

```
function [x_vals,y_vals,z] = objfn_surface(...
    objfn,fixed_pars,limits,i_x,i_y,n)
    lims_x = limits(i_x,:);
    lims_y = limits(i_y,:);
    x_vals = linspace(lims_x(1), lims_x(2), n)';
    y_vals = linspace(lims_y(1), lims_y(2), n)';
    x_y_vals = combvec(x_vals',y_vals')';
    pars = repmat(fixed_pars, size(x_y_vals,1),1);
    pars(:,i_x) = x_y_vals(:,1);
    pars(:,i_y) = x_y_vals(:,2);
    z_vals = objfn(pars);
    z = reshape(z_vals,size(x_vals,1),size(y_vals,1));
end
```

metric1.m

```
function error = metric1(price_market, price_model)
    error = mean((abs(price_market - price_model) /
price_market));
end
```

run_optim.m

```
dir_res = 'results';
dlm_prec = 9;

inertia = 0.7;
correction_factor = 2.0;
max_abs_velocity = 0.2;

tic_all = tic;
for i_opt = 1:size(opt_np_niter_pairs,1)
    tic_i_opt = tic;
```

```
    n_p = opt_np_niter_pairs(i_opt,1);
    n_iter = opt_np_niter_pairs(i_opt,2);

    % results\np025_it050\
    s_run = sprintf('np%03i_it%03i', n_p, n_iter);
    dir_run = dirjoin({dir_res s_run});
    mkdir_if_not_exist(dir_run);

    % results\np025_it050\np025_it050_pso\
    s_pso = [s_run '_pso'];
    dir_run_pso = dirjoin({dir_run s_pso});
    mkdir_if_not_exist(dir_run_pso);

    %
results\np025_it050\np025_it050_pso\np025_it050_pso_heston\
    s_model = [s_pso '_' model_name];
    dir_run_pso_model = dirjoin({dir_run_pso, s_model});
    mkdir_if_not_exist(dir_run_pso_model);

    %
results\np025_it050\np025_it050_pso\np025_it050_pso_heston\...
    %   np025_it050_pso_heston_Gbest.csv
    fname_run_pso_model_Gbest = dirjoin(...
        {dir_run_pso_model [s_model '_Gbest.csv']});
    if exist(fname_run_pso_model_Gbest, 'file') == 0
        csvheader(fname_run_pso_model_Gbest, {'p' 'r' 'Gbest'});
    end

    for i_par = idxs_params
        tic_i_par = tic;

        %
results\np025_it050\np025_it050_pso\np025_it050_pso_heston\...
        %   np025_it050_pso_heston_p01\
        s_model_p = sprintf('%s_p%02i', s_model, i_par);
        dir_run_pso_model_p = dirjoin({dir_run_pso_model,
s_model_p});
        mkdir_if_not_exist(dir_run_pso_model_p);

        %
results\np025_it050\np025_it050_pso\np025_it050_pso_heston\...
        %   np025_it050_pso_heston_p01\...
        %   np025_it050_pso_heston_p01_Gbest.csv
        fname_run_pso_model_p_Gbest = dirjoin(...
            {dir_run_pso_model_p [s_model_p '_Gbest.csv']});
        if exist(fname_run_pso_model_p_Gbest, 'file') == 0
            csvheader(fname_run_pso_model_p_Gbest, {'r'
'Gbest'});
        end

        params_model = model_parameters(i_par,:);
```

```
                %
results\np025_it050\np025_it050_pso\np025_it050_pso_heston\...
        %   np025_it050_pso_heston_p01\...
        %   np025_it050_pso_heston_p01_mprices.csv
        fname_run_pso_model_p_mprices = dirjoin(...
            {dir_run_pso_model_p [s_model_p '_mprices.csv']});
        if exist(fname_run_pso_model_p_mprices, 'file') == 0
            csvheader(fname_run_pso_model_p_mprices,
{'market_price'});
            market_prices =
fn_model_prices(data_prices,params_model);
            dlmwrite(fname_run_pso_model_p_mprices,
market_prices,...
                'precision',dlm_prec);
        else
            market_prices =
csvread(fname_run_pso_model_p_mprices);
        end

        objfn = @(p)
model_obj_fn(p,data_prices,market_prices,@metric1);

        for i_run = idxs_runs
            tic_i_run = tic;
            % results\np025_it050\np025_it050_pso\...
            %   np025_it050_pso_heston\...
            %   np025_it050_pso_heston_p01\...
            %   np025_it50_pso_heston_p01_r01\
            s_model_p_r = sprintf('%s_r%02i', s_model_p, i_run);
            dir_run_pso_model_p_r = dirjoin(...
                {dir_run_pso_model_p, s_model_p_r});
            mkdir_if_not_exist(dir_run_pso_model_p_r);

            % results\np025_it050\np025_it050_pso\...
            %   np025_it050_pso_heston\...
            %   np025_it050_pso_heston_p01\...
            %   np025_it050_pso_heston_p01_r01\...
            %   np025_it050_pso_heston_p01_r01_Gbest.csv
            fname_run_pso_model_p_r_Gbest = dirjoin(...
                {dir_run_pso_model_p_r [s_model_p_r
'_Gbest.csv']});

            % results\np025_it050\np025_it050_pso\...
            %   np025_it050_pso_heston\...
            %   np025_it050_pso_heston_p01\...
            %   np025_it050_pso_heston_p01_r01\...
            %   np025_it050_pso_heston_p01_r01_pgbest.csv
            fname_run_pso_model_p_r_pgbest = dirjoin(...
                {dir_run_pso_model_p_r [s_model_p_r
'_pgbest.csv']});

            [Gbest,p_gbest] = pso(...
```

```
                objfn,model_limits,n_iter,n_p,...
                inertia,correction_factor,max_abs_velocity,...

model_parameter_names,dir_run_pso_model_p_r,s_model_p_r);

            dlmwrite(fname_run_pso_model_p_r_Gbest,Gbest,...
                'precision',dlm_prec);
            dlmwrite(fname_run_pso_model_p_r_pgbest,p_gbest,...
                'precision',dlm_prec);

            dlmwrite(fname_run_pso_model_p_Gbest,[i_run
Gbest],...
                '-append','precision',dlm_prec);
            dlmwrite(fname_run_pso_model_Gbest,[i_par i_run
Gbest],...
                '-append','precision',dlm_prec);

            disp(['done running: ' s_model_p_r]);
            toc(tic_i_run)
        end
        disp(['done running: ' s_model_p]);
        toc(tic_i_par)
    end
    disp(['done running: ' s_run]);
    toc(tic_i_opt)
end
disp('done running all');
toc(tic_all);
```

run_optim_market.m

```
file_market_data = 'd1.csv';

tmp_s_market_data = strsplit(file_market_data, '.');
s_market_data = cell2mat(tmp_s_market_data(1));

dir_res = 'results_market';

dlm_prec = 9;

inertia = 0.7;
correction_factor = 2.0;
max_abs_velocity = 0.2;

tic_all = tic;
for i_opt = 1:size(opt_np_niter_pairs,1)
    tic_i_opt = tic;

    n_p = opt_np_niter_pairs(i_opt,1);
    n_iter = opt_np_niter_pairs(i_opt,2);

    % results_market\np025_it050\
```

```
    s_run = sprintf('np%03i_it%03i', n_p, n_iter);
    dir_run = dirjoin({dir_res s_run});
    mkdir_if_not_exist(dir_run);

    % results_market\np025_it050\np025_it050_pso\
    s_pso = [s_run '_pso'];
    dir_run_pso = dirjoin({dir_run s_pso});
    mkdir_if_not_exist(dir_run_pso);

    %
results_market\np025_it050\np025_it050_pso\np025_it050_pso_hesto
n\
    s_model = [s_pso '_' model_name];
    dir_run_pso_model = dirjoin({dir_run_pso, s_model});
    mkdir_if_not_exist(dir_run_pso_model);

    %
results_market\np025_it050\np025_it050_pso\np025_it050_pso_hesto
n\...
    %   np025_it050_pso_heston_Gbest.csv
    fname_run_pso_model_Gbest = dirjoin(...
        {dir_run_pso_model [s_model '_Gbest.csv']});
    if exist(fname_run_pso_model_Gbest, 'file') == 0
        csvheader(fname_run_pso_model_Gbest, {'p' 'r' 'Gbest'});
    end


    tic_i_par = tic;

    %
results_market\np025_it050\np025_it050_pso\np025_it050_pso_hesto
n\...
    %   np025_it050_pso_heston_duomenys1\
    s_model_m = sprintf('%s_%s', s_model, s_market_data);
    dir_run_pso_model_p = dirjoin({dir_run_pso_model,
s_model_m});
    mkdir_if_not_exist(dir_run_pso_model_p);

    %
results_market\np025_it050\np025_it050_pso\np025_it050_pso_hesto
n\...
    %   np025_it050_pso_heston_duomenys1\...
    %   np025_it050_pso_heston_duomenys1_Gbest.csv
    fname_run_pso_model_p_Gbest = dirjoin(...
        {dir_run_pso_model_p [s_model_m '_Gbest.csv']});
    if exist(fname_run_pso_model_p_Gbest, 'file') == 0
        csvheader(fname_run_pso_model_p_Gbest, {'r' 'Gbest'});
    end

    market_data = csvread(file_market_data);
    market_prices = market_data(:,6);
    data_prices = market_data(:,1:5);
```

```
    objfn = @(p)
model_obj_fn(p,data_prices,market_prices,@metric1);

    for i_run = idxs_runs
        tic_i_run = tic;
        % results_market\np025_it050\np025_it050_pso\...
        %   np025_it050_pso_heston\...
        %   np025_it050_pso_heston_duomenys1\...
        %   np025_it50_pso_heston_duomenys1_r01\
        s_model_p_r = sprintf('%s_r%02i', s_model_m, i_run);
        dir_run_pso_model_p_r = dirjoin(...
            {dir_run_pso_model_p, s_model_p_r});
        mkdir_if_not_exist(dir_run_pso_model_p_r);

        % results_market\np025_it050\np025_it050_pso\...
        %   np025_it050_pso_heston\...
        %   np025_it050_pso_heston_duomenys1\...
        %   np025_it050_pso_heston_duomenys1_r01\...
        %   np025_it050_pso_heston_duomenys1_r01_Gbest.csv
        fname_run_pso_model_p_r_Gbest = dirjoin(...
            {dir_run_pso_model_p_r [s_model_p_r '_Gbest.csv']});

        % results_market\np025_it050\np025_it050_pso\...
        %   np025_it050_pso_heston\...
        %   np025_it050_pso_heston_duomenys1\...
        %   np025_it050_pso_heston_duomenys1_r01\...
        %   np025_it050_pso_heston_duomenys1_r01_pgbest.csv
        fname_run_pso_model_p_r_pgbest = dirjoin(...
            {dir_run_pso_model_p_r [s_model_p_r
'_pgbest.csv']});

        [Gbest,p_gbest] = pso(...
            objfn,model_limits,n_iter,n_p,...
            inertia,correction_factor,max_abs_velocity,...

model_parameter_names,dir_run_pso_model_p_r,s_model_p_r);

        dlmwrite(fname_run_pso_model_p_r_Gbest,Gbest,...
            'precision',dlm_prec);
        dlmwrite(fname_run_pso_model_p_r_pgbest,p_gbest,...
            'precision',dlm_prec);

        fname_run_pso_model_p_r_mprices = dirjoin(...
            {dir_run_pso_model_p_r [s_model_p_r
'_mprices.csv']});
        model_prices = fn_model_prices(data_prices,p_gbest);
        dlmwrite(fname_run_pso_model_p_r_mprices,...
            [data_prices market_prices model_prices],...
            'precision',dlm_prec);

        dlmwrite(fname_run_pso_model_p_Gbest,[i_run Gbest],...
```

```
            '-append','precision',dlm_prec);
        dlmwrite(fname_run_pso_model_Gbest,[s_market_data i_run
Gbest],...
            '-append','precision',dlm_prec);

        disp(['done running: ' s_model_p_r]);
        toc(tic_i_run)
    end
    disp(['done running: ' s_model_m]);
    toc(tic_i_par)
end
disp(['done running: ' s_run]);
toc(tic_i_opt)

disp('done running all');
toc(tic_all)
```

run_optim_market_bates.m

```
clc;clear all;close all;

idxs_runs_bates = 1:10;

opt_np_niter_pairs = [
   50 50];
init_limits_bates = [
    % v0    = initial variance
     0.0  0.9;
    % vT    = long run variance (theta in Heston's paper)
     0.0  0.9;
    % rho   = correlation
    -1.0  1.0;
    % k     = speed of mean reversion (kappa in Heston's paper)
     0.0    5;
    % sigma = vol of vol
     0.0    5;
    % lambda= intensity of jumps;
     0.0  1.0;
    % muJ   = mean of jumps;
    -1.0  1.0;
    % vJ    = variance of jumps;
     0.0  1.0;
    ];

bates_parameter_names =
{'$v_0$','$\theta$','$\rho$','$\kappa$',...
    '$\sigma$','$\lambda$','$\mu_J$','$\sigma_J$'};

model_name = 'bates';
fn_model_prices = @v_call_batescf;

model_obj_fn = @v_objfn_bates;
```

```
model_parameter_names = bates_parameter_names;
model_limits = init_limits_bates;
idxs_runs = idxs_runs_bates;

run('run_optim_market');
```

run_optim_market_heston.m

```
clc;clear all;close all;

idxs_runs_heston = 1:10;

opt_np_niter_pairs = [
   50 50];

init_limits_heston = [
    % v0    = initial variance
     0.0  0.9;
    % vT    = long run variance (theta in Heston's paper)
     0.0  0.9;
    % rho   = correlation
    -1.0  1.0;
    % k     = speed of mean reversion (kappa in Heston's paper)
     0.0    5;
    % sigma = vol of vol
     0.0    5;
    ];

heston_parameter_names =
{'$v_0$','$\theta$','$\rho$','$\kappa$','$\sigma$'};

model_name = 'heston';
fn_model_prices = @v_call_hestoncf;

model_obj_fn = @v_objfn_heston;
model_parameter_names = heston_parameter_names;
model_limits = init_limits_heston;
idxs_runs = idxs_runs_heston;

run('run_optim_market');
```

run_optim_market_both.m

```
run('run_optim_market_heston');
run('run_optim_market_bates');
```

strjoin.m

```
function joined_str = strjoin(s, separator)
    joined_str = [sprintf(['%s' separator],s{1:end-1}),s{end}];
end
```

strsplit.m

```
function terms = strsplit(s, delimiter)
assert(ischar(s) && ndims(s) == 2 && size(s,1) <= 1, ...
    'strsplit:invalidarg', ...
    'The first input argument should be a char string.');

if nargin < 2
    by_space = true;
else
    d = delimiter;
    assert(ischar(d) && ndims(d) == 2 && size(d,1) == 1 &&
~isempty(d), ...
        'strsplit:invalidarg', ...
        'The delimiter should be a non-empty char string.');

    d = strtrim(d);
    by_space = isempty(d);
end

%% main
s = strtrim(s);

if by_space
    w = isspace(s);
    if any(w)
        % decide the positions of terms
        dw = diff(w);
        sp = [1, find(dw == -1) + 1];     % start positions of
terms
        ep = [find(dw == 1), length(s)];  % end positions of
terms

        % extract the terms
        nt = numel(sp);
        terms = cell(1, nt);
        for i = 1 : nt
            terms{i} = s(sp(i):ep(i));
        end
    else
        terms = {s};
    end

else
    p = strfind(s, d);
    if ~isempty(p)
        % extract the terms
        nt = numel(p) + 1;
        terms = cell(1, nt);
        sp = 1;
        dl = length(delimiter);
```

```
        for i = 1 : nt-1
            terms{i} = strtrim(s(sp:p(i)-1));
            sp = p(i) + dl;
        end
        terms{nt} = strtrim(s(sp:end));
    else
        terms = {s};
    end
end
```

dirjoin.m

```
function joined_dirs = dirjoin(dirs)
    joined_dirs = strjoin(dirs, '\\');
end
```

config.m

```
idxs_params_heston = 1:2;
idxs_params_bates = 1:2;

idxs_runs_heston = 1:1;
idxs_runs_bates = 1:1;

pso_np_niter_pairs = [
    5 2;
    5 3];

inertia = 0.7;
correction_factor = 0.2;
max_abs_velocity = 0.2;
```

csvheader.m

```
function csvheader(fname, header)
    fid = fopen(fname, 'w') ;
    fprintf(fid, '%s,', header{1,1:end-1}) ;
    fprintf(fid, '%s\n', header{1,end}) ;
    fclose(fid) ;
end
```

mkdir_if_not_exist.m

```
function mkdir_if_not_exist(dirpath)
    if dirpath(end) ~= '/', dirpath = [dirpath '/']; end
    if (exist(dirpath, 'dir') == 0), mkdir(dirpath); end
end
```

v_call_batescf.m

```
function calls = v_call_batescf(x,p)
```

```
% x = option parameters: [S,X,tau,r,q] each of size [n 1]
% p = bates parameters: [v0,vT,rho,k,sigma,lambda,muJ,vJ]
%     each of size [1 1]
    n_rows = size(x,1);
    calls = zeros(n_rows,1);
    for ii = 1:n_rows
        calls(ii, 1) = callBatescf(...
            x(ii,1),x(ii,2),x(ii,3),x(ii,4),x(ii,5),...
            p(1),p(2),p(3),p(4),p(5),p(6),p(7),p(8));
    end
end
```

v_call_hestoncf.m

```
function calls = v_call_hestoncf(x,p)
% x = option parameters: [S,X,tau,r,q] each of size [n 1]
% p = heston parameters: [v0,vT,rho,k,sigma] each of size [1 1]
    n_rows = size(x,1);
    calls = zeros(n_rows,1);
    Ss = x(:,1);
    Xs = x(:,2);
    taus = x(:,3);
    rs = x(:,4);
    qs = x(:,5);
    v0s = p(:,1);
    vTs = p(:,2);
    rhos = p(:,3);
    ks = p(:,4);
    sigmas = p(:,5);
    parfor i = 1:n_rows
        calls(i, 1) = HestonCall(...

Ss(i),Xs(i),rs(i),taus(i),v0s,ks,vTs,sigmas,rhos,qs(i));
        %calls(ii, 1) = HestonCall(...
        %    x(ii,1),x(ii,2),x(ii,4),x(ii,3),...
        %    p(1),p(4),p(2),p(5),p(3),x(ii,5));
        %calls(ii, 1) =
callHestoncf(x(ii,1),x(ii,2),x(ii,3),x(ii,4),x(ii,5),p(1),p(2),p
(3),p(4),p(5));
    end
end
```

v_objfn_heston.m

```
function fvals = v_objfn_heston(ps, x, y, metric_fn)
% ps = heston parameters: [v0,vT,rho,k,sigma] each of size [n_p
1]
% x = option parameters: [S,X,tau,r,q] each of size [n 1]
% y = true prices: size [n 1]
% vectorized error metric function, f(y_true, y_forecast)
    n_p = size(ps, 1);
    fvals = zeros(n_p, 1);
```

```
    parfor i = 1:n_p
        fvals(i) = objfn_heston(ps(i,:), x, y, metric_fn);
    end
end
```

v_objfn_bates.m

```
function fvals = v_objfn_bates(ps, x, y, metric_fn)
% ps = bates parameters: [v0,vT,rho,k,sigma,lambda,muJ,vJ]
%     each of size [n_p 1]
% x = option parameters: [S,X,tau,r,q] each of size [n 1]
% y = true prices: size [n 1]
% vectorized error metric function, f(y_true, y_forecast)
    n_p = size(ps, 1);
    fvals = zeros(n_p, 1);
    for i = 1:n_p
        fvals(i) = objfn_bates(ps(i,:), x, y, metric_fn);
    end
end
```

HestonCall.m

```
function C=HestonCall(St,K,r,T,vt,kap,th,sig,rho,lda)
dphi=0.01;
maxphi=50;
phi=(eps:dphi:maxphi)';

f1 = CF_SVj(log(St),vt,T,0,kap*th,0.5,kap+lda-
rho*sig,rho,sig,phi);
P1 = 0.5+(1/pi)*sum(real(exp(-
1i*phi*log(K)).*f1./(1i*phi))*dphi);
f2 = CF_SVj(log(St),vt,T,0,kap*th,-0.5,kap+lda,rho,sig,phi);
P2 = 0.5+(1/pi)*sum(real(exp(-
1i*phi*log(K)).*f2./(1i*phi))*dphi);
C = St*P1 -K*exp(-r*T)*P2;
```

vCallHeston.m

```
function C=vCallHeston(St,K,r,T,vt,kap,th,sig,rho,lda)
dphi=0.01;
maxphi=50;
phi=(eps:dphi:maxphi);

phi = repmat(phi, size(St, 1), 1);

f1 = vCF_SVj(log(St),vt,T,0,kap.*th,0.5,kap+lda-
rho.*sig,rho,sig,phi);
P1 = 0.5+(1/pi)*sum(real(exp(-1i*phi.*repmat(log(K), 1,
size(phi,2))).*f1./(1i*phi))*dphi, 2);
f2 = vCF_SVj(log(St),vt,T,0,kap.*th,-0.5,kap+lda,rho,sig,phi);
```

```
P2 = 0.5+(1/pi)*sum(real(exp(-1i*phi.*repmat(log(K), 1,
size(phi,2))).*f2./(1i*phi))*dphi, 2);
C = St.*P1 -K.*exp(-r.*T).*P2;
```

callBatescf.m

```
function call =
callBatescf(S,X,tau,r,q,v0,vT,rho,k,sigma,lambda,muJ,vJ)
vP1 = 0.5 + 1/pi *
quadl(@P1,0,200,[],[],S,X,tau,r,q,v0,vT,rho,k,sigma,lambda,muJ,v
J);
vP2 = 0.5 + 1/pi *
quadl(@P2,0,200,[],[],S,X,tau,r,q,v0,vT,rho,k,sigma,lambda,muJ,v
J);
call = exp(-q * tau) * S * vP1 - exp(-r * tau) * X * vP2;
end

function p = P1(om,S,X,tau,r,q,v0,vT,rho,k,sigma,lambda,muJ,vJ)
i=1i;
p = real(exp(-i*log(X)*om) .* cfBates(om-
i,S,tau,r,q,v0,vT,rho,k,sigma,lambda,muJ,vJ) ./ (i * om * S *
exp((r-q) * tau)));
end

function p = P2(om,S,X,tau,r,q,v0,vT,rho,k,sigma,lambda,muJ,vJ)
i=1i;
p = real(exp(-i*log(X)*om) .* cfBates(om
,S,tau,r,q,v0,vT,rho,k,sigma,lambda,muJ,vJ) ./ (i * om));
end

function cf =
cfBates(om,S,tau,r,q,v0,vT,rho,k,sigma,lambda,muJ,vJ)
d   = sqrt((rho * sigma * 1i*om - k).^2 + sigma^2 * (1i*om + om
.^ 2));

g2  = (k - rho*sigma*1i*om - d) ./ (k - rho*sigma*1i*om + d);

cf1 = 1i*om .* (log(S) + (r - q) * tau);
cf2 = vT * k / (sigma^2) * ((k - rho*sigma*1i*om - d) * tau - 2
* log((1 - g2 .* exp(-d * tau)) ./ (1 - g2)));
cf3 = v0 / sigma^2 * (k - rho*sigma*1i*om - d) .* (1 - exp(-d *
tau)) ./ (1 - g2 .* exp(-d * tau));
% jump
cf4 = -lambda*muJ*1i*tau*om + lambda*tau*(  (1+muJ).^(1i*om) .*
exp( vJ*(1i*om/2) .* (1i*om-1) )-1  );
cf  = exp(cf1 + cf2 + cf3 + cf4);
end
```

objfn_bates.m

```
function fval = objfn_bates(p,x,price_market,metric_fn)
    penalty_terms(1) = min(p(1), 0);
```

```
    penalty_terms(2) = min(p(2), 0);
    penalty_terms(3) = max(abs(p(3)) - 1, 0);
    penalty_terms(4) = min(p(4), 0);
    penalty_terms(5) = min(p(5), 0);
    penalty_terms(6) = min(p(5), 0);
    penalty_terms(7) = max(abs(p(3)) - 1, 0);
    penalty_terms(8) = min(p(5), 0);
    penalty = sum(abs(penalty_terms));
    price_model = v_call_batescf(x,p);
    fval = metric_fn(price_market', price_model') + penalty;
end
```

objfn_heston.m

```
function fval = objfn_heston(p,x,price_market,metric_fn)
% p = heston parameters: [v0,vT,rho,k,sigma] each of size [1 1]
% x = option parameters: [S,X,tau,r,q] each of size [n 1]
% price_market = true market prices: size [n 1]
% vectorized error metric function, f(y_true, y_forecast)
    penalty_terms(1) = min(p(1), 0);
    penalty_terms(2) = min(p(2), 0);
    penalty_terms(3) = max(abs(p(3)) - 1, 0);
    penalty_terms(4) = min(p(4), 0);
    penalty_terms(5) = min(p(5), 0);
    penalty = sum(abs(penalty_terms));
    price_model = v_call_hestoncf(x,p);
    fval = metric_fn(price_market', price_model') + penalty;
end
```

CF_SVj.m

```
function fj=CF_SVj(xt,vt,tau,mu,a,uj,bj,rho,sig,phi)
%-------------------------------------------------------------------
%PURPOSE: implements the CF fj of Heston. Uses Heston's
notations.
%-------------------------------------------------------------------
xj = bj - rho.*sig.*phi.*1i;
dj = sqrt( xj.^2 - (sig.^2).*( 2.*uj.*phi.*1i - phi.^2 ) );
gj = ( xj+dj )./( xj-dj );
D  = ( xj+dj )./(sig.^2).* ( 1-exp(dj.*tau) )./( 1-
gj.*exp(dj.*tau)  ) ;
xx = ( 1-gj.*exp(dj.*tau) )./( 1-gj );
C  = mu.*phi.*1i.*tau + a./( sig.^2 ) .* ( (xj+dj) .* tau -
2.*log(xx) );
fj = exp( C + D.*vt + 1i.*phi.*xt );
```