KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

TAIKOMOSIOS MATEMATIKOS KATEDRA

Reda Isodaitė

NIUTONO METODO REALIZACIJA IR TYRIMAS TAIKANT ŽULIJA AIBES

Magistro darbas

Vadovas
prof. dr. J. Valantinas

KAUNAS, 2007

KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

TAIKOMOSIOS MATEMATIKOS KATEDRA

TVIRTINU
Katedros vedėjas
prof. dr. J.Rimas
2007 06 06

NIUTONO METODO REALIZACIJA IR TYRIMAS TAIKANT ŽULIJA AIBES

Taikomosios matematikos magistro baigiamasis darbas

Vadovas
( ) prof. dr. J. Valantinas
2007 06 03

Recenzentas
( ) doc.dr. K. Plukas
2007 06 01

Atliko
FMMM 5 gr. stud.
( ) R. Isodaitė
2007 05 25

KAUNAS, 2007

KVALIFIKCINĖ KOMISIJA

Pirmininkas: Leonas Saulis, habil. dr.,Vilniaus Gedimino technikos universiteto profesorius

Sekretorius: Eimutis Valakevičius, docentas

Nariai:

- Algimantas Jonas Aksomaitis, profesorius (KTU)
- Arūnas Barauskas, dr.,UAB „Elsis“ generalinio direktoriaus pavaduotojas
- Vytautas Janilionis, docentas (KTU)
- Zenonas Navickas, profesorius (KTU)
- Vidmantas Povilas Pekarskas, profesorius (KTU)
- Rimantas Rudzkis, habil.dr., banko „NORD/LB“ vyriausiasis analitikas.

Isodaite R. Implementation and analysis of Newton's method using Julia sets: Master's work in applied mathematics / supervisor dr. assoc. prof. J. Valantinas; Department of Applied mathematics, Faculty of Fundamental Sciences, Kaunas University of Technology. – Kaunas, 2007. – 45 p.

SUMMARY

Julia sets and filled Julia sets of Newton's fractals are analyzed in this work. The Escape Time Algorithm provides us with a means for „seeing" the filled Julia sets of Newton's fractals, but roots, (zeros) of the polynomial under investigation should be known. The Newton's method for finding roots of an algebraic equation $f(x)=0$ is well known. Here in the paper the complex Newton method for finding roots of a complex polynomial is presented. The main difficulties, associated with implementation of this method in practice, are discussed, namely: construction of the set of initial points (first approximations of the roots), finding the basin of attraction for a particular root and so forth.

It was found out that accuracy of the roots of a complex polynomial and clearness of Julia sets depended on the number of iterations in the search process. The step size used in scanning selected areas of the complex plane also influenced the said accuracy and clearness. The proper choice of both the number of iterations and the step size made it possible to detect multiple (and neighboring) roots of a complex polynomial and to visualize their basins of attraction.

Some experimental results are presented.

TURINYS

Lentelių sąrašas

Paveikslų sąrašas

ĮVADAS

Terminą „fraktalas“, kuriuo apibūdinamos geometrinės struktūros, gaunamos rekurentinių procedūrų pagalba, 1975 m. pasiūlė amerikiečių matematikas Benua Mandelbrotas (Benoit Mandelbrot). Šios rekurentinės procedūros sukuria geometrines formas, kurios atrodo panašiai vaizduojant jas įvairiais masteliais. Beje, panašumas gali būti ne tik vizualinis, bet ir statistinis.

Fraktalas – tai geometrinis objektas, kurio atskirose dalyse galima pamatyti jį patį ar bent jau kažką labai panašaus. Fraktalas – tai tarsi žaidimas su veidrodžiu, prieš kurį atsistojame, paėmę kitą (mažesnį) veidrodį. Didžiajame veidrodyje matome mažesnįjį, kuriame, savo ruožtu atsispindi didesnysis su visu jame matomu vaizdu. Šitaip išgauname be galo mažėjančių (smulkėjančių) to paties objekto kopijų seką. Taigi, fraktalas – dalinai savipanašus darinys. Dalinį savipanašumą galime įžvelgti medžiuose, snaigėse, kalnuose, jūros kranto linijoje, netgi architektūroje, mene, transporto srautuose, finansinių rinkų elgsenoje ir panašiai, [10].

Fraktalais vadinami (dalinai) savipanašūs objektai, kurie dar yra ir nereguliarūs, t. y. tokie, kurių fragmentuose matome arba be galo “susigarbanojusią” kreivę (pavyzdžiui, Mandelbroto aibės kontūrą), arba be galo “padrikus” taškus (pavyzdžiui Sierpinskio trikampyje). Būtent dėl šio nereguliarumo fraktalai turi tiek daug analogijų su mus supančiu realiu pasauliu.

Mandelbroto aibė - tai vienas geriausiai pasaulyje ištyrinėtų (algebrinių) fraktalų. Pirmieji šią aibę, XX amžiaus pradžioje, nagrinėjo prancūzų matematikai P. Fatou ir G. Julia, tačiau pirmieji Mandelbroto aibės "paveikslai" buvo gauti tik 1978 metais. Kartu buvo tiriama ir Žulija aibė, labai tampriai susijusi su Mandelbroto aibe.

Šiame darbe nagrinėjama Žulija aibė Niutono fraktaluose.

Niutono-Rafsono (kartais, tiesiog, Niutono) metodas yra matematinės analizės metodas, skirtas algebrinio daugianario šaknims rasti. Ieškant kompleksinio daugianario šaknų, reikia analizuoti tam tikras kompleksinės plokštumos sritis: nustatant rėžius realioje ir menamoje ašyse ir tam tikru pasirinktu žingsniu pereinant visus taškus, kurie patenka į rėžiais apribotą sritį (daugianario šaknų lokalizacijos srritį). Tokiu būdu ieškant kompleksinio daugianario šaknų, gauname žemėlapius, kurie atskleidžia pasirinktų pradinių taškų artėjimo į šaknis tendencijas. Tokie žemėlapiai turi fraktalinę struktūrą ir yra vadinami Niutono-Rafsono fraktalais. Juose akivaizdžiai matosi, link kurių daugianrio šaknų artėja vienos ar kitos kompleksinės plokštumos taškų aibės, įvardijamos kaip atitinkamų šaknų pritraukimo baseinai. Pastarieji baseinai – tai užpildytos Žulija aibės. Beje, racionaliosios aukštesnio nei pirmojo laipsnio polinominės transformacijos Žulija aibę galima interpretuoti kaip aibę taškų, kurių orbitos nekonverguoja nė į vieną iš daugianario šaknų.

Darbo tikslas – algoritmizuoti, realizuoti ir ištirti Niutono metodą dvimačiu atveju, Žulija aibių pagalba vizualizuoti kompleksinio daugianario šaknų pritraukimo baseinus.

1. BENDROJI DALIS

1.1. DINAMINĖS SISTEMOS, FRAKTALAI BEI JŲ GENERAVIMO ALGORITMAI

Sąvokos „fraktalas“ ir „fraktalinė geometrija“ atsirado aštuntojo dešimtmečio pabaigoje, o nuo devintojo dešimtmečio šios sąvokos ilgam įsitvirtino matematikų ir programuotojų žodyne. Terminas „fraktalas“ paimtas iš lotynų kalbos – „fractus“ reiškia nereguliarų fragmentą. Šį terminą 1975m. pasiūlė amerikiečių matematikas Benua Mandelbrotas (Benoit Mandelbrot) nereguliarioms, panašioms į save struktūroms žymėti, t. y. žiūrint pro padidinamąjį stiklą į atskirą figūros fragmentą, pastarasis „primena“ visą figūrą. Kita fraktalams būdinga sąvybė – juos sudaro be galo daug taškų, kurių tarpusavio išsidėstymas toks sudėtingas, kad neįmanoma aprašyti jų „geometrijos“. Fraktalų įvairovę patogu pateikti, atsižvelgiant į jų klasifikaciją: geometriniai fraktalai, algebriniai fraktalai, stochastiniai fraktalai. Geometriniai fraktalai siejami su iteruotųjų funkcijų sistemomis (IFS) ir jų atraktoriais, algebriniai fraktalai paprastai yra netiesinių diskrečiųjų dinaminių sistemų, veikiančių kompleksinėje ploštumoje, padarinys, pagaliau, stochastiniai fraktalai – tai tie patys geometriniai fraktalai, tik jų sintezės (generavimo) metu atsititinai keičiami IFS sudarančių transformacijų parametrai.

Žemiau detaliau apibrėžiama fraktalo sąvoka bei pateikiamas vienas iš fraktalų generavimo algoritmų.

1.1.1. FRAKTALAI IR JŲ KLASIFIKAVIMAS

Tarkime, turime metrinę erdvę (X,d) ir joje apibrėžtą transformaciją $f: X \to X$. Pastebėsime, jog $f(S) = \{f(x) \mid x \in S\}$, kai $S \subset X$. Transformacija f yra apgręžiama, jeigu ji yra abipusiškai vienareikšmė ir $f(X) = X$. Šiuo atveju galima apibrėžti atvirkštinę transformaciją $f^{-1}: X \to X$ tokią, kad $f^{-1}(y) = x$ ($x \in X$) yra vienintelis taškas, kuriam $f(x) = y$.

Transformacijos $f: X \to X$ iteracijomis pirmyn vadinamos transformacijos $f^{0n}: X \to X$, apibrėžiamos lygybėmis:

$$f^{00}(x) = x, \quad f^{01}(x) = f(x), \quad f^{0(n+1)}(x) = f(f^{0n}(x)), \text{ su visais } n = 0,1,2,\ldots.$$

Jeigu f yra apgręžiama, tai transformacijos f iteracijomis atgal vadinamos transformacijos $f^{0(-m)}: X \to X$, apibrėžiamos lygybėmis:

$$f^{0(-1)}(x) = f^{-1}(x), \quad f^{0(-m)}(x) = \left(f^{0m}\right)^{-1}(x), \text{ su visais } m = 1,2,3,\ldots.$$

Praktiniuose taikymuose svarbiausia yra akcentuoti ryšį tarp transformacijas apibūdinančių formulių ir jų poveikyje atsirandančių geometrinių pasikeitimų (ištempimų, poslinkių, lenkimų ir pan.) nagrinėjamose erdvėse. Be to, svarbu ir tai, kaip transformacijos veikia ne atskirus erdvės *X* taškus, o jos poaibius.

Kalbant apie geometrinius fraktalus, jų prigimtį, paprastai imamos Euklido metrinės erdvės $(\mathbf{R}, d)$, $(\mathbf{R}^2, d)$ arba $(\mathbf{R}^3, d)$ ir joje veikiančios afiniosios transformacijos.

Bendru atveju afinioji transformacija $\omega : \mathbf{R}^2 \to \mathbf{R}^2$, veikianti Euklido erdvėje $(\mathbf{R}^2, d)$, apibrėžiama lygybe $\omega(x) = \omega(x_1, x_2) = (ax_1 + bx_2 + e,\ cx_1 + dx_2 + g)$, su visais $x = (x_1, x_2) \in \mathbf{R}^2$; čia *a, b, c, d, e, g* yra realieji skaičiai (afiniosios transformacijos ω parametrai). Afiniosios transformacijos turi daug svarbių geometrinių ir algebrinių savybių. Jų pagalba galima realizuoti posūkio, atspindžio, panašumo, pražulniąsias ir kitas transformacijas Euklido erdvėje $(\mathbf{R}^2, d)$, [9].

Jeigu su visais $x, y \in \mathbf{R}^2$ teisinga nelygybė $d(\omega(x), \omega(y)) \le s \cdot d(x, y)$ $(0 \le s < 1)$, tai afinioji transformacija $\omega : \mathbf{R}^2 \to \mathbf{R}^2$ vadinama suspaudžiančiąja, o realusis skaičius s, tenkinantis šią sąlygą, - afiniosios transformacijos suspaudimo koeficientu.

Taškas $x_\omega \in \mathbf{R}^2$ vadinamas nejudamuoju transformacijos ω tašku, jeigu $\omega(x_\omega) = x_\omega$.

Imkime afiniųjų suspaudžiančiųjų transformacijų $\omega_i : \mathbf{R}^2 \to \mathbf{R}^2$, $i = 1,2,...,N$, rinkinį; atskirų afiniųjų transformacijų suspaudimo koeficientus pažymėkime s_i, $i = 1,2,...,N$. Tada Euklido erdvė $(\mathbf{R}^2, d)$ su joje veikiančių suspaudžiančiųjų afiniųjų transformacijų rinkiniu vadinama iteruotųjų funkcijų sistema ir žymima $\mathrm{IFS}\{\mathbf{R}^2; \omega_1, \omega_2, ..., \omega_N\}$. IFS suspaudimo koeficientu laikomas skaičius $s = \max\{s_1, s_2, ..., s_N\}$.

Apibrėžkime dar vieną metrinę erdvę $(\mathcal{H}(\mathbf{R}^2), h)$ tokiu būdu: $\mathcal{H}(\mathbf{R}^2)$ - visų netuščių uždarųjų aibės $\mathbf{R}^2$ poaibių aibė; *h* — metrika, nusakanti atstumą tarp bet kurių dviejų aibės $\mathcal{H}(\mathbf{R}^2)$ elementų (aibės $\mathbf{R}^2$ poaibių), būtent:

$$h(A, B) = \max\{d(A, B), d(B, A)\};$$

čia $d(A, B) = \max\limits_{x \in A}\{\min\limits_{y \in B}\{d(x, y)\}\}$; $d(B, A) = \max\limits_{y \in A}\{\min\limits_{x \in B}\{d(y, x)\}\}$.

Įvestoji erdvė $(\mathcal{H}(\mathbf{R}^2), h)$ vadinama „fraktalų“ erdve arba, tiesiog, fraktaline erdve.

Perkelkime IFS sudarančias afiniąsias transformacijas į erdvę $(\mathcal{H}(\mathbf{R}^2), h)$. Tada $\omega_i : \mathcal{H}(\mathbf{R}^2) \to \mathcal{H}(\mathbf{R}^2)$, apibrėžiama lygybe $\omega_i(B) = \{\omega_i(y) \mid y \in B\}$, $\forall B \in \mathcal{H}(\mathbf{R}^2)$, yra suspaudžiančioji transformacija erdvėje $(\mathcal{H}(\mathbf{R}^2), h)$, ir jos suspaudimo koeficientas s_i, $i = 1,2,...,N$.

Apibrėžkime dar vieną transformaciją fraktalinėje erdvėje $(\mathcal{H}(\mathbf{R}^2), h)$, būtent:

$W:\mathcal{H}(\mathbf{R}^2)\to\mathcal{H}(\mathbf{R}^2)$, čia $W(B)=\omega_1(B)\cup\omega_2(B)\cup\ldots\cup\omega_N(B)=\bigcup_{i=1}^{N}\omega_i(B),\ \forall B\in\mathcal{H}(\mathbf{R}^2)$.

Galima įsitikinti, jog pastaroji transformacija irgi yra suspaudžiančioji, t.y.

$$h(W(B),W(C))\le s\cdot h(B,C) \text{ su visais } B,C\in\mathcal{H}(\mathbf{R}^2);$$

be to $s=\max\{s_1,s_2,\ldots,s_N\}$.

Vienintelis nejudamasis transformacijos $W:\mathcal{H}(\mathbf{R}^2)\to\mathcal{H}(\mathbf{R}^2)$ taškas A $(A\in\mathcal{H}(\mathbf{R}^2))$ toks, kad $A=W(A)=\bigcup_{i=1}^{N}\omega_i(A)=\lim_{n\to\infty}W^{0n}(B),\ \forall B\in\mathcal{H}(\mathbf{R}^2)$, vadinamas IFS atraktoriumi (arba fraktalu), [9].

Įvesime dinaminės sistemos, kuriai tenka svarbus vaidmuo vizualizuojant kompleksinių daugianrių šaknų pritraukimo baseinus, sąvoką.

Dinamine sistema vadinama transformacija $f:\mathbf{R}^2\to\mathbf{R}^2$ metrinėje erdvėje $(\mathbf{R}^2,d)$; paprastai dinaminė sistema žymima $\{\mathbf{R}^2;f\}$.

Taškas $x\in\mathbf{R}^2$ vadinamas periodiniu *f* tašku, jeigu yra skaičius $n\in\{1,2,\ldots\}$ toks, kad $f^{0n}(x)=x$. Skaičius *n* vadinamas taško *x* periodu. Mažiausias iš tokių skaičių vadinamas minimaliu periodinio taško *x* periodu.

Periodinio *f* taško orbita vadinama *f* ciklu; *f* ciklo periodas yra taško cikle periodas; minimalus *f* ciklo periodas – tai skirtingų ciklo taškų skaičius.

Imkime dinaminę sistemą $\{X;f\}$. Tarkime, kad $x_f\in X$ yra nejudamasis *f* taškas, t.y. $f(x_f)=x_f$. Taškas x_f vadinamas pritraukiančiuoju nejudamuoju *f* tašku, jeigu egzistuoja teigiamas skaičius $\varepsilon>0$ toks, kad $f:B(x_f,\varepsilon)\to B(x_f,\varepsilon)$, t.y. uždarasis rutulys *B* yra atvaizduojamas į save; be to, *f* yra suspaudžiantysis atvaizdis aibėje $B(x_f,\varepsilon)$ (čia $B(x_f,\varepsilon)=\{y\in X\mid d(x_f,y)\le\varepsilon\}$).

Taškas x_f vadinamas atstumiančiuoju nejudamuoju *f* tašku, jeigu yra skaičiai $\varepsilon>0$ ir $c>1$ tokie, kad $d(f(x_f),f(y))\ge c\cdot d(x_f,y)$ su visais $y\in B(x_f,\varepsilon)$.

Šias sąvokas naudosime nagrinėdami fraktalus bei jų sintezės (generavimo) algoritmą.

Fraktalinėje geometrijoje, kaip jau buvo užsiminta aksčiau, išskiriami šie fraktalų tipai – geometriniai fraktalai, algebriniai fraktalai, stochastiniai fraktalai.

Geometriniai fraktalai paprastai siejami su ankstyvuoju fraktalinės geometrijos (kaip mokslo šakos) vystymosi periodu. Faktiškai tai iteruotųjų funkcijų sistemų, charakterizuojamų afiniųjų transformacijų, veikiančių įvairaus matavimo Euklido erdvėse, rinkiniais, atraktorių aibė. Šio tipo fraktalai yra vaizdžiausi ir, galima sakyti, geriausiai išanalizuoti. Dvimačiu atveju jie gaunami laužtės (trimačiu atveju – paviršiaus), vadinamos generatoriumi, pagalba. Kiekviename algoritmo žingsnyje

(atitinkamai parinkus mastelį) laužtę sudaranti atkarpa keičiama laužte – generatoriumi. Daugkartinio šios procedūros taikymo rezultatas – geometrinis fraktalas.

Kompiuterinėje grafikoje geometriniai fraktalai yra būtina priemonė medžių, krūmų, kalnų linijos ir kitiems vaizdams gauti. Dvimačiai geometriniai fraktalai naudojami erdvinėms tekstūroms išgauti.

Algebriniai fraktalai priklauso pačiai didžiausiai fraktalų grupei. Jie gaunami netiesinių iteracinių procesų *n*-matėse erdvėse pagalba. Geriausia išanalizuoti yra dvimačiai iteraciniai procesai. Žinoma, jog netiesinės dinaminės sistemos turi kelias stabilias būsenas. Būsena, kurioje atsidūrė dinaminė sistema po baigtinio iteracijų skaičiaus, priklauso nuo jos pradinės padėties. Todėl kiekviena stabili būsena (dar vadinama atraktoriumi) turi fiksuotą aibę pradinių padėčių, iš kurių sistema būtinai pereina į analizuojamą galutinę stabilią būseną. Tokiu būdu fazinė sistemos erdvė gali būti suskaidyta į atraktorių pritraukimo zonas. Jei fazinė erdvė yra dvimatė, tai pritraukimo zonoms priskirdami skirtingas spalvas, galime nesunkiai gauti spalvinį fazinį šios sistemos (iteracinio proceso) portretą.

Dar vieną fraktalų klasę sudaro stochastiniai fraktalai. Juos galima būtų priskirti geometriniams fraktalams. Skirtumas tas, jog vietoj įprastinių iteruotųjų funkcijų sistemų imamos parametrizuotos sistemos. Iteracinio proceso metu keičiamos parametrų reikšmės, ir tai daroma atsitiktine tvarka. Gaunami fraktalai yra labai panašūs į realaus pasaulio objektus – vietovės reljefą bei jūros paviršių.

1.1.2 ALGEBRINIŲ FRAKTALŲ SINTEZĖ, TAIKANT PL-ALGORITMĄ

Yra žinomi keli IFS atraktorių sintezės (generavimo) algoritmai. Tai determinuotasis algoritmas, atsitiktinių iteracijų algoritmas bei „pabėgimo laiko“ (PL) algoritmas. Pirmieji du algoritmai sėkmingai taikomi geometrinių (kartais stochastinių) fraktalų sintezei. Algebrinių fraktalų generavimui paprastai taikomas taikant PL-algoritmas.

Geometrinių fraktalų sintezės kontekste, PL-algoritmas žinomas daugiau kaip teorinis rezultatas (idėja), jo praktinė realizacija iki šiol dar kelia tam tikrų problemų. Nepaisant to, PL-algoritmo esmę atskleisime kalbėdami būtent apie geometrinius fraktalus (IFS atraktorius).

Tarkime, IFS$\{\mathbf{R}^2;\omega_1,\omega_2,...,\omega_N\}$ atraktorius yra aibė A ($A=\omega_1(A)\cup\omega_2(A)\cup...\cup\omega_N(A)$). Be to tarkime, jog visos afiniosios transformacijos yra abipusiškai vienareikšmės, t. y. apgręžiamos.

Duotajai IFS konstruojama dinaminė poslinkių sistema $\{A;S\}$, kai poslinkio transformacija $S:A\to A$ apibrėžiama (kiekvienam taškui $a\in A$) taip: $S(a)=\omega_i^{-1}(a)$, jeigu $a\in\omega_i(A)$ $(i\in\{1,2,...,N\})$ ir $a\notin\bigcup_{j=1(j\neq i)}^{N}\omega_j(A)$; sakoma, jog taškas a yra transformacijos ω_i^{-1} veikimo zonoje;

$S(a)=\omega_t^{-1}(a)$, ($t\in\{i_1,i_2,...,i_r\}\subset\{1,2,...,N\}$), jeigu $a\in\bigcap_{s=1}^{r}\omega_{i_s}(A)$; sakoma, jog taškas *a* yra bet kurios iš transformacijų ω_t^{-1} veikimo zonoje.

Su duotąja IFS susijusi dinaminė poslinkių sistema $\{A;S\}$ praplečiama į visą erdvę $\mathbf{R}^2$, t.y. konstruojama nauja dinaminė sistema $\{\mathbf{R}^2;\hat{S}\}$. Naujoji poslinkio transformacija $\hat{S}:\mathbf{R}^2\to\mathbf{R}^2$ apibrėžiama lygybe $\hat{S}(x)=\omega_i^{-1}(x)$, kai taškas $x\in\mathbf{R}^2$ patenka į *i*-tosios atvirkštinės afiniosios transformacijos $\omega_i^{-1}(x)(i\in\{1,2,...,N\})$ veikimo zoną. Aišku, kad $\hat{S}$ sutampa su *S*, kai taškas $x\in A$.

Toliau tarkime, kad IFS atraktorius *A* patenka į stačiakampį $\mathcal{W}$, t.y. $A\subset\mathcal{W}\subset\mathbf{R}^2$. Kiekvienam stačiakampio $\mathcal{W}$ taškui *x* konstruojama orbita $\{\hat{S}^{0n}(x)\}_{n=1}^{\tau}$; čia τ - iš anksto apibrėžtas iteracijų skaičius. Fiksuokime skritulį su centru stačiakampio įstrižainių susikirtimo taške (pažymėkime *O*) ir spinduliu *R*; beje, skritulys talpina savyje stačiakampį $\mathcal{W}$. Taip pat apibrėžkime aibę $\mathcal{V}=\{(x,y)\in\mathbf{R}^2\mid x^2+y^2>R\}$.

Jeigu $d(\hat{S}^{0\tau}(x),O)\le R$, t.y. po τ iteracijų taško $x\in\mathcal{W}$ orbita nepalieka skritulio, t.y. nepasiekia srities $\mathcal{V}$, daroma išvada, jog taškas *x* priklauso IFS atraktoriui ($x\in A$); priešingu atveju ($d(\hat{S}^{0\tau}(x),O)>R$), $x\notin A$ [9].

„Pabėgimo laiko“ algoritmo „veikimas“ remiasi tuo, jog IFS atraktorius (aibė *A*) yra transformacijos $W:\mathcal{H}(\mathbf{R}^2)\to\mathcal{H}(\mathbf{R}^2)$ atstumiantysis nejudamasis taškas. Kitaip tariant, taškų, esančių arčiau (metrikos *h* prasme) atraktoriaus *A*, orbitos ilgiau „užsibūna“ skritulyje, negu orbitos taškų, labiau nutolusių nuo atraktoriaus.

Pagrindinė kliūtis, dėl kurios šis algoritmas sunkokai pritaikomas praktikoje (geometriniams fraktalams generuoti), - tai afiniųjų transformacijų veikimo zonų atskyrimo kriterijaus nebuvimas. Akivaizdu, jog šios kliūties išvengiame, kai kalbama apie algebrinių fraktalų sintezę arba problematiškai orientuotą Žulija aibių analizę.

1.1.3. ŽULIJA AIBĖS

„Pabėgimo laiko“ algoritmas gali būti taikomas bet kuriai iš dinaminių sistemų $\{\mathbf{R}^2;f\}$, $\{\mathbf{C};f\}$ arba $\{\hat{\mathbf{C}};f\}$. Tereikia nurodyti stačiakampę sritį $\mathcal{W}$ ir aibę $\mathcal{V}$, į kurią gali „pabėgti“ srities $\mathcal{W}$ taškų orbitos. Algoritmo veikimo rezultatas bus $\mathcal{W}$ „paveiksliukas“, kuriame pikselis, (atitinkantis tašką *z*), yra nuspalvinamas, atsižvelgiant į mažiausią sveikąją reikšmę *n* reikšmę, su kuria $f^{on}(z)\in\mathcal{V}$.

Kas nutiktų, jei „pabėgimo laiko“ algoritmą taikytume dinaminei sistemai $f:\hat{\mathbf{C}}\to\hat{\mathbf{C}}$, kur $f(z)=z^2$? Šią transformaciją galime išreikšti $f(z)=f(x_1,x_2)=(x_1^2-x_2^2,2x_1x_2)$. Jau žinome, kad taškų, nepriklausančių vienetiniam skrituliui $F=\{z\in\mathbf{C}:|z|\le 1\}$, orbitos tolsta į begalybę. Taškų, priklausančių *F*, orbitos artėja į skritulio centrą. Taigi, jei $\mathcal{W}$ yra stačiakampė sritis, talpinanti *F*, ir apibrėžiantis aibę $\mathcal{V}$ spindulys *R* yra pakankamai didelis, tai galime tikėtis, kad „pabėgimo laiko“ algoritmas duos paveiksliuką, kuriame matysime *F*, apsuptą spalvotų koncentrinių žiedų.

Aibė *F* vadinama užpildyta Žulija aibe, susieta su polinomine transformacija $f(z)=z^2$. *F* kraštas vadinamas transformacijos *f* Žulija aibe; ją žymėsime *J*. Minėtu atveju, tai vienetinis apskritimas, kurio centras taške $z=0$. Ši Žulija aibė atskiria tuos taškus, kurių orbitos konverguoja į begalybę, nuo tų, kurių orbitos artėja į tašką $z=0$. Pačios Žulija aibės taškų orbitos, nekonverguoja nei į begalybę, nei į centrą. Be to $J\in\mathcal{H}(\hat{\mathbf{C}})$ ir $f(J)=J=f^{-1}(J)$, [9].

Tegul $f:\hat{\mathbf{C}}\to\hat{\mathbf{C}}$ yra aukštesnio nei pirmojo laipsnio polinominė transformacija (daugianaris). Tegul F_f – aibė taškų, priklausančių **C**, kurių orbitos netolsta į begalybę, t. y.

$$F_f=\{z\in\mathbf{C}\mid\{|f^{on}(z)|\}_{n=0}^{\infty}\text{ yra aprėžta}\}.$$

Tokia aibė vadinama užpildyta Žulija aibe, susieta su daugianariu *f*. Aibės F_f kraštas vadinamas daugianario *f* Žulija aibe, ir yra žymima J_f.

Teorema. Tegul $f:\hat{\mathbf{C}}\to\hat{\mathbf{C}}$ yra daugianaris, kurio laipsnis didesnis nei 1. Tegul F_f – užpildyta Žulija aibė, o J_f – Žulija aibė. Tada F_f ir J_f yra netušti uždarieji *C* poaibiai, t.y.

$$F_f\in\mathcal{H}(\mathbf{C})\text{ ir }J_f\in\mathcal{H}(\mathbf{C}).\text{ Be to }f(J_f)=J_f=f^{-1}(J_f)\text{ ir }f(F_f)=F_f=f^{-1}(F_f).$$

Aibė $\mathcal{V}_\infty=\hat{\mathbf{C}}\setminus F_f$ yra jungi lanku.

Šių teiginių įrodymus galima rasti literatūroje, [9].

Apibendrinant pabrėšime, kad užpildyta Žulija aibė F_f - tai mažėjančios kompaktinių aibių sekos $\{K_n\}$ riba, kur K_n yra aibė taškų, kurių orbitos po pirmųjų *n* itercijų nepasiekia $\mathcal{V}$; jos papildymas (pažymėkime $\mathcal{V}_\infty$) yra didėjančios atvirųjų jungiųjų lanku poaibių sekos $\{\mathcal{V}_n\}$ riba, t.y.

$$\mathcal{V}_\infty=\lim_{n\to\infty}\mathcal{V}_n=\bigcup_{n=0}^{\infty}\mathcal{V}_n.$$

Ši aibė vadinama be galo nutolusio taško traukos (veikiant polinominei transformacijai *f*) baseinu. Ji yra jungi, kadangi $\mathcal{V}_n$ yra jungiosios aibės. Taigi

$$\hat{\mathbf{C}}=F_\lambda\cup\mathcal{V}_\infty.$$

$\mathcal{V}_\infty$ yra jungus, atviras ir netuščias. F_f yra kompaktinis ir netuščias.

Galima teigti, jog PL-algoritmas – tai priemonė stebėti bei „fiksuoti“ užpildytas Žulija aibes, o taip pat aibių sekas $\{\mathcal{V}_n\}$ ir $\{K_n\}$.

PL-algoritmas fiksuoja, kaip skirtingi srities $\mathcal{W}$ taškai, veikiami dinaminės sistemos, pasiekia sritį $\mathcal{V}$. Kuri aibė „atstumia“ taškų orbitas? Kur prasideda „pabėgančios“ orbitos? Dinaminių sistemų atveju orbitos „pabėga“ nuo IFS atraktoriaus.

Imkime dinaminę sistemą $\{\hat{\mathbf{C}}; f_\lambda(z) = z^2 - \lambda\}, \lambda \in \mathbf{C}$. Norėdami išsiaiškinti, kokia aibė „atstumia“ šios dinaminės sistemos orbitas, panagrinėkime atvirkštinę $f_\lambda(z)$ transformaciją: $f^{-1}(z) = \left\{+\sqrt{z+\lambda}, -\sqrt{z+\lambda}\right\}$; tai transformacijų pora; čia ‚+‘ ženklu pažymėta kompleksinės šaknies reikšmė, atitinkanti tašką ant neneigiamos realiosios ašies, t.y.

$$\sqrt{z} = \sqrt{x_1 + ix_2} = (a(x_1, x_2), b(x_1, x_2));$$

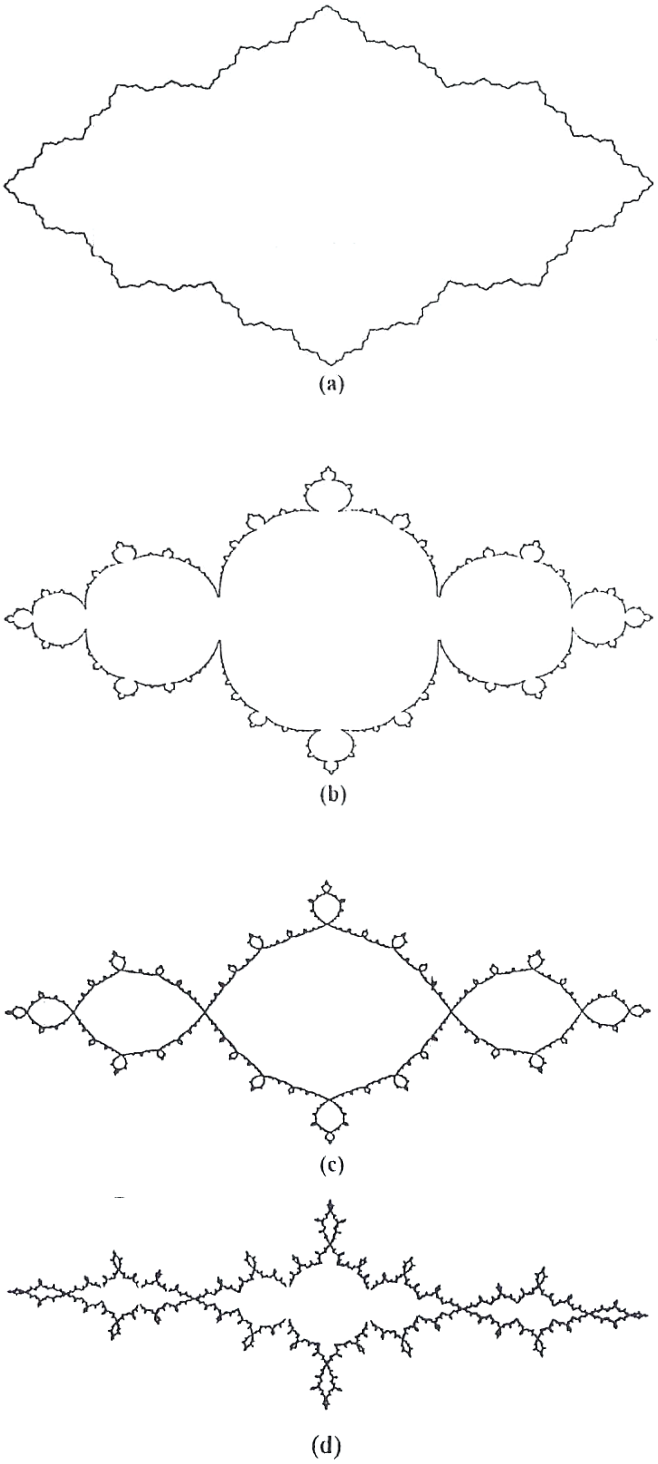
$$a(x_1, x_2) = \sqrt{\frac{\sqrt{x_1^2 + x_2^2} + x_1}{2}}, \text{ kai } x_2 \geq 0, \quad a(x_1, x_2) = -\sqrt{\frac{\sqrt{x_1^2 + x_2^2} + x_1}{2}}, \text{ kai } x_2 < 0,$$

$$b(x_1, x_2) = -\sqrt{\frac{\sqrt{x_1^2 + x_2^2} - x_1}{2}}.$$

Norėdami rasti atstumiančiąją aibę, paleiskime dinaminę sistemą priešinga kryptimi. Tai veda prie $\text{IFS}\{\hat{\mathbf{C}}; \omega_1(z) = \sqrt{z+\lambda}, \omega_2(z) = -\sqrt{z+\lambda}\}$.

Ši IFS turi atraktorių, nuo kurio dinaminės sistemos $\{\hat{\mathbf{C}}; z^2 - \lambda\}$ veikiamos taškų orbitos stengiasi „pasprukti“. Eksperimentiškai šį atraktorių galima „užčiuopti“ pasinaudojant atsitiktinių iteracijų algoritmu. Dalis rezultatų, gautų taikant ši algoritmą sistemai $\text{IFS}\{\hat{\mathbf{C}}; \omega_1(z) = \sqrt{z+\lambda}, \omega_2(z) = -\sqrt{z+\lambda}\}, \lambda \in [0,3]$, pateikti **1.1 pav.** Visai atvejais IFS atraktorius sutampa su Žulija aibe J_f.

Iš pirmo žvilgsnio peršasi mintis, kad $\text{IFS}\{\hat{\mathbf{C}}; \omega_1(z) = \sqrt{z+\lambda}, \ \omega_2(z) = -\sqrt{z+\lambda}\}$ yra suspaudžiančioji IFS, kurios atraktorius yra J_f. Deja, ne visiškai taip, kadangi $\hat{\mathbf{C}} = \omega_1(\hat{\mathbf{C}}) \cup \omega_2(\hat{\mathbf{C}})$; taigi, IFS nėra susijusi su vieninteliu nejudamuoju erdvės $\mathcal{H}(\hat{\mathbf{C}})$ tašku. Norint, kad aibė J_λ būtų vienintelis minėtosios IFS atraktorius, iš erdvės $\hat{\mathbf{C}}$, kurioje veikia ši IFS, reikėtų pašalinti kai kurias zonas.



1.1 pav. IFS$\{\mathbf{C}; \sqrt{z+\lambda}, -\sqrt{z+\lambda}\}$, $\lambda \in [0,3]$ **atraktoriai, gauti taikant atsitiktinių iteracijų algoritmą**

Teorema. Tarkime, kad $\lambda \in \mathbf{C}$. Tarkime, kad dinaminė sistema $\left\{\hat{\mathbf{C}}; f_\lambda(z) = z^2 - \lambda\right\}$ turi pritraukiantįjį ciklą $\left\{z_1, z_2, ..., z_p\right\} \subset \mathbf{C}$. Tarkime, kad ε yra pakankamai mažas teigiamas skaičius, ir X žymi Rymano sferą $\hat{\mathbf{C}}$ su (p+1) pašalintais atviraisiais (spindulio ε) rutuliais; vieno rutulio centras

sutampa su šiauriniu poliumi, likusių rutulių – su ciklo taškais. Nagrinėkime IFS $\{X; \omega_1(z) = \sqrt{z+\lambda}, \omega_2(z) = -\sqrt{z+\lambda}\}$. Tada transformacija W erdvėje $\mathcal{H}(X)$, apibrėžta lygybe

$$W(B) = \omega_1(B) \cup \omega_2(B), \forall B \in \mathcal{H}(X),$$

tolydžiai (Hausdorfo metrikos prasme) atvaizduoja erdvę $\mathcal{H}(X)$ į ją pačią; be to $W : \mathcal{H}(X) \to \mathcal{H}(X)$ turi vienintelį nejudamąjį tašką – transformacijos $f_\lambda(z) = z^2 - \lambda$ Žulija aibę J_λ. Taip pat

$$\lim_{n\to\infty} W^{0n}(B) = J_\lambda, \forall B \in \mathcal{H}(X).$$

Šios išvados išlieka teisingos ir tuo atveju, kai orbita $\{f^{0n}(O)\}$ tolsta į begalybę, ir $X = \hat{\mathbf{C}} \setminus B(\infty, \varepsilon)$.

Ši teorema gali būti apibendrinta imant kitokias aukštesnio nei pirmojo laipsnio polinomines transformacijas $f : \hat{\mathbf{C}} \to \hat{\mathbf{C}}$. Sunkumai iškyla, kai reikia gauti atvirkštinių transformacijų išraiškas. Tai ne visada įmanoma; tuo pačiu, atsitiktinių iteracijų algoritmo taikymas tampa problematišku. Dažniausiai Žulija ir užpildytų Žulija aibių vaizdai gaunami panaudojant PL-algoritmą.

Šiame darbe Žulija aibių teoriją taikysime daugianario šaknų paieškos metodui, literatūroje dažnai vadinamam Niutono – Rafsono (arba Niutono) metodu.

1.2. NIUTONO METODAS IR JO GEOMETRINĖ INTERPRETACIJA

Įvairiuose matematikos bei taikomojo pobūdžio uždaviniuose dažnai tenka susidurti su algebrinių lygčių $f(z) = 0$ ($z \in \mathbf{R}$ arba $z \in \mathbf{C}$) šaknų paieškos problema. Šį uždavinį galima spręsti įvairiais metodais: pusiaukirtos, stygų, liestinių ir kt. Nagrinėsime liestinių (arba Niutono) metodą.

1.2.1. VIENMATIS ATVEJIS

Bet kokiai diferencijuojamai funkcijai $f : \mathbf{R} \to \mathbf{R}$ Niutono iteracinė funkcija nusakoma išraiška

$$N(x) = x - f(x)/f'(x). \qquad \textbf{(1.1)}$$

Pasirinkę spėjamą šaknį (pirmąją šaknies aproksimaciją) x_0, gauname iteracinę seką $\{x_n\}$; čia

$$x_n = N(x_{n-1}) = N^{0n}(x_0) = \underbrace{(N \circ N \circ \ldots \circ N)}_{n \text{ kartų}}(x_0).$$

Jei pradinis taškas x_0 parinktas sėkmingai, tuomet riba $N^{0\infty}(x_0) = \lim_{n\to\infty} N^{0n}(x_0)$ egzistuoja ir sutampa su daugianario $f(x)$ šaknimi.

Geometrinė Niutono metodo ((**1.1**) išraiška) interpretacija gana paprasta. Norint gauti $N(x_0)$, reikia funkcijos $f(x)$ grafiką pakeisti liestine T taške $(x_0, f(x_0))$. Tuomet $N(x_0)$ bus ne kas kita kaip liestinės T ir *Ox* ašies susikirtimo taškas. Niutono metodas remiasi tuo, kad net jeigu $N(x_0)$ ir nėra labai arti x_0, taškas $N(x_0)$ yra tikslesnė $f(x)$ šaknies aproksimacija nei x_0, be to tikėtina, kad seka $\{N^{0n}(x_0)\}$ konverguoja į šaknį $N^{0\infty}(x_0)$.

Liestinę T sudaro taškai (x, y), kurie tenkina lygtį

$$y - f(x_0) = f'(x_0)(x - x_0),$$

ir $(N(x_0),0)$ yra šios lygties šaknis.

1.2.2. DVIMATIS ATVEJIS

Nagrinėkime funkciją $g : \mathbf{R}^2 \to \mathbf{R}$. Šiuo atveju funkcijos *g* grafikas yra paviršius erdvėje $\mathbf{R}^3$. Tarkim x_0 (dabar jau dvimatis vektorius) yra spėjama funkcijos $g(x)$ šaknis (pirmoji aproksimacija). Taške $(x_0, g(x_0))$ pakeičiame $g(x)$ grafiką jo liečiamąja plokštuma T. Plokštumą sudaro taškai (x, y) (y - skaliaras) tokie, kad $y - g(x_0) = \nabla g(x_0)(x - x_0)$; čia ∇g yra funkcijos $g(x)$ gradientas taške x_0. Taigi, dešinėje lygybės pusėje turime skaliarinę vektorių sandaugą. Ieškome sprendinio $(x,0)$ tokio, kad

$$-g(x_0) = \nabla g(x_0)(x - x_0). \qquad \textbf{(1.2)}$$

Tačiau šiuo atveju neegzistuoja sprendinys, kurį galėtume vadinti $N(x_0)$. Sprendinių aibė susideda iš tiesės L, kuria T kerta *x*-plokštumą. Tarkim, kad x^* ir x^{**} yra du L taškai. Įstatę juos į **(1.2)** lygtį ir, atlikę elementarius pertvarkius, gauname, kad $\nabla g(x_0)(x^* - x^{**}) = 0$. Kitaip tarus L ir $\nabla g(x_0)$ yra statmeni vienas kitam, [3].

Bet kurį tašką $x \in L$ galime laikyti $N(x_0)$. Parinkus artimiausią tašką, gauname $N(x_0) - x_0 = \alpha \nabla g(x_0)$, bet kuriam α. Įstatę į **(1.2)** lygtį gauname

$$\alpha = -g(x_0) / \|\nabla g(x_0)\|^2 \text{ arba } N(x_0) - x_0 = -g(x_0)\nabla g(x_0) / \|\nabla g(x_0)\|^2.$$

Pastebėsime, kad $\nabla g(x_0)$ nukreiptas greičiausio nusileidimo kryptimi, ir norma $\|N(x_0) - x_0\| = g(x_0) / \|\nabla g(x_0)\|$ yra minimali. Ši interpretacija teisinga ir didesnių matavimų duomenims, [3].

Tarkime $g : \mathbf{R}^k \to \mathbf{R}$. Tuomet Niutono iteracinė funkcija yra

$$N(x_0) = x_0 - \frac{g(x_0)}{\|\nabla g(x_0)\|}\,\text{kryptis}(\nabla g(x_0)) \qquad \textbf{(1.3)}$$

Išnagrinėkime dar vieną atvejį. Imkime funkciją $f:\mathbf{R}^2 \to \mathbf{R}^2$ (arba $f:\mathbf{C}\to\mathbf{C}$). Kintamąjį žymime z, o realiąją ir kompleksinę f dalis atitinkamai $u = u(z) = u(x+iy)$ ir $v = v(z) = v(x+iy)$. Ieškome bendros šių lygčių šaknies. Naudodami Jakobiano matricą

$$Df(z_0) = \begin{pmatrix} \partial u/\partial x & \partial u/\partial y \\ \partial v/\partial x & \partial v/\partial y \end{pmatrix}$$

du atskirus **(1.2)** lygties atvejus galime apjungti:

$$-f(z_0) = Df(z_0)(N(z) - z_0);$$

čia z_0, $N(z_0)$ ir $f(z_0)$ yra dvimačiai vektoriai-stulpeliai, o dešinėje lygties pusėje turime matricų daugybą. Jei $Df(z_0)$yra nesinguliari matrica, sprendinys bus vienintelis. Šiuos rezultatus nesunkiai galima apibendrinti. Tegu $f:\mathbf{R}^k \to \mathbf{R}^k$, o $Df = \left(\partial f_i/\partial x_j\right)_{i,j}$ yra šios funkcijos Jakobiano matrica. Tuomet Niutono iteracinė funkcija

$$N(x_0) = x_0 - (Df(x_0))^{-1} f(x_0). \qquad \textbf{(1.4)}$$

(Jakobiano matrica turi būti nesinguliari!).

Pastebėsime, kad Jakobiano matrica Dg skaliarinei funkcijai $g:\mathbf{R}^k \to \mathbf{R}$ yra matrica-eilutė – transponuotas vektorinis stulpelis ∇g. Analogišką apibendrinimą galime gauti ir iš kitos pusės (pritaikydami **(1.3)** išraiškai normą $\|f\|$). Tęsdami iki $f:\mathbf{R}^k \to \mathbf{R}^k$, lengvai įsitikinsime, kad

$$\|f\|\nabla\|f\| = \left(f^T Df\right)^T = (Df)^T f\,. \qquad \textbf{(1.5)}$$

Tarkime $f:\mathbf{R}^k \to \mathbf{R}^k$, o $Df = \left(\partial f_i/\partial x_j\right)_{i,j}$ yra šios funkcijos Jakobiano matrica . Tuomet Niutono iteracinė funkcija

$$N(x_0) = x_0 - \frac{\|f\|^2}{\left\|(Df)^T f\right\|^2}(Df)^T f(x_0).$$

1.2.2. KONVERGAVIMO GREITIS

Algebrinės lygtys dažnai sprendžiamos Niutono metodu, nes jis pakankamai greitai konverguoja. Šio metodo konvergavimą aptarkime plačiau.

Sakykime, Niutono iteracinė funkcija generuoja skaičių $x_0, x_1, x_2, \ldots, x_n, \ldots$ seką, kurią žymime $\{x_n\}$.

Tarkime, kad $a \in \mathbf{R}$, $x_n \in \mathbf{R}$, $n = 0, 1, \ldots$. Seka $\{x_n\}$ konverguoja prie a, jei $\lim_{n\to\infty} |x_n - a| = 0$.

Jei egzistuoja konstanta $c \in [0; 1)$ ir sveikasis skaičius $k \geq 0$ tokie, kad su visais $n \geq k$ galioja nelygybė $|x_{n+1} - a| \leq c|x_n - a|$, tai sakoma, kad seka $\{x_n\}$ konverguoja prie a tiesiškai.

Jei egzistuoja seka $\{c_n\}$, konverguojanti prie 0, ir galioja nelygybė $|x_{n+1} - a| \leq c_n |x_n - a|$, tai sakoma, kad seka $\{x_n\}$ konverguoja prie a greičiau nei tiesiškai.

Jei $\{x_n\}$ konverguoja prie a ir egzistuoja $p > 1$, $c \geq 0$ ir $k \geq 0$ tokie, kad nelygybė $|x_{n+1} - a| \leq c \cdot |x_n - a|^p$ galioja su visais $n \geq k$, tai sakoma, kad seka $\{x_n\}$ konverguoja prie a eile, ne mažesne nei p. Kai $p = 2$, sakoma, kad konvergavimo greitis yra kvadratinis.

Funkcija $g(x)$ vadinama tolydžiąja pagal Lipšicą su konstanta c aibėje X (rašoma $g(x) \in Lip_c(x)$), jei $|g(x) - g(y)| \leq c|x - y|$, su visais $x, y \in X$.

Lema. Tarkime, kad duota funkcija $y = f(x)$, ($x \in D$, D - atvirasis intervalas) ir $f'(x) \in Lip_\gamma(D)$. Tada su visais $x, y \in D$

$$|f(y) - f(x) - f'(x)(y - x)| \leq \frac{\gamma (y - x)^2}{2}.$$

Įrodymas. Iš matematinės analizės žinome, kad

$$\int_x^y f'(z)dz = f(y) - f(x).$$

Vadinasi,

$$f(y) - f(x) - f'(x)(x - y) = \int_x^y (f'(z) - f'(x))\, dz.$$

Pakeitę $z = x + t(y - x)$; $dz = (y - x)dt$; $z = x$; $t = 0$; $z = y$; $t = 1$, gauname:

$$f(y) - f(x) - f'(x)(y - x) = \int_0^1 (f'(x + t(y - x)) - f'(x))(y - x)dt.$$

Remdamiesi tuo, kad $f'(x) \in Lip_\gamma(D)$, gauname:

$$|f(y) - f(x) - f'(x)(y - x)| \leq |y - x| \int_0^1 \gamma |x + t(x - y) - x|\, dt =$$

$$= \gamma\, |y - x|^2 \int_0^1 t\, dt = \frac{\gamma (y - x)^2}{2}$$

Lema įrodyta, [5].

Teorema. Tarkime, kad duota funkcija $y = f(x);\ x \in D$ — atvirasis intervalas, $f'(x) \in Lip_\gamma(D)$. Sakykime, egzistuoja toks $\rho > 0$, kad $|f'(x)| > \rho$, su visais $x \in D$. Jeigu $f(x) = 0$ turi šaknį $s \in D$, tai egzistuoja $\eta > 0$ ir, kai $|s - x_0| < \eta$, seka $\{x_n\}$, nusakoma formule

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)},\ n = 0, 1, 2, \ldots$$

konverguoja prie *s*; be to, kai $n \geq 0$, galioja nelygybė

$$|x_{n+1} - s| \leq \frac{\gamma}{2\rho} |x_n - s|^2. \qquad \textbf{(1.6)}$$

Įrodymas. Tarkime, kad $\tau \in (0; 1)$, o *r* - intervale D esančio atvirojo intervalo, kurio centras *s*, spindulys. Pažymėkime

$$\eta = \min\left(r, \tau\left(\frac{2\rho}{\gamma}\right)\right).$$

Kai $n = 0, 1, 2, \ldots$, teoremos **(1.6)** sąlyga yra tenkinama ir $|x_{n+1} - s| \leq \tau |x_n - s| < \eta$.

Kai $n = 0$, tai

$$|x_1 - s| = x_0 - \frac{f(x_0)}{f'(x_0)} - s = x_0 - s - \frac{f(x_0) - f(s)}{f'(x_0)} = \frac{1}{f'(x_0)}\left(f(s) - f(x_0) - f'(x_0)(s - x_0)\right).$$

Pagal aukščiau įrodytą lemą, skliaustuose esantis reiškinys yra mažesnis už $\frac{\gamma (x_0 - s)^2}{2}$. Vadinasi,

$$|x_1 - s| \leq \frac{\gamma}{2|f'(x)|} |x_0 - s|^2 \leq \frac{\gamma}{2\rho} |x_0 - s|^2.$$

Kadangi $|x_0 - s| \leq \eta \leq \tau \frac{2\rho}{\gamma}$, tai $|x_1 - s| \leq \frac{\gamma}{2\rho} \tau \frac{2\rho}{\gamma} |x_0 - s| \leq \tau |x_0 - s| \leq \tau \cdot \eta < \eta$.

Analogiškai įrodome, kai $n = 1, 2, \ldots$.

Teorema įrodyta, [5].

Reikalavimas, jog $|f'(x)| > \rho > 0$, reikalingas tam, kad konvergavimo greitis būtų kvadratinis. Jei $f'(s) = 0$, tai *s* yra kartotinė šaknis ir Niutono metodas konverguoja tik tiesiškai.

Pavyzdžiui, **1.1** ir **1.2 lentelėse** pateikti dviejų lygčių sprendimo Niutono metodu rezultatai, iliustruojantys metodo konvergavimo greitį. Iš jų matyti, kad, ieškant lygties $x^2 - 1 = 0$ šaknies $x = 1$, konvergavimo greitis yra kvadratinis, nes $f'(1) \neq 0$. Antrajai lygčiai ($x^2 - 2x + 1 = 0$) $y'(1) = 0$, todėl, ieškant šaknies $x = 1$, sekos konvergavimo greitis yra tiesinis.

Kadangi Niutono metodas turi kvadratinį konvergavimo greitį, tai jis dažniausiai taikomas ieškant daugianario šaknų. Pagrindinis šio metodo trūkumas yra tas, kad reikia nurodyti pradinę šaknies reikšmę. Šios problemos sprendimą pateikiame **2.1.1.** skyrelyje.

1.1 lentelė

Lygties $f(x) = x^2 - 1 = 0$ sprendimo Niutono metodu rezultatai

n	x_n
0	2,0
1	1,25
2	1,025
3	1,000304878
4	1,0000000465
5	1,0

1.2 lentelė

Lygties $y(x) = x^2 - 2x + 1 = 0$ sprendimo Niutono metodu rezultatai

n	x_n
0	2,0
1	1,5
2	1,25
3	1,125
4	1,0625
5	1,03125

2. TIRIAMOJI DALIS

2.1. NIUTONO METODO REALIZACIJA IR TYRIMAS TAIKANT ŽULIJA AIBES

Algebrinių lygčių $f(z)=0$ ($z \in \mathbf{R}$ arba $z \in \mathbf{C}$) šaknų paieškos problemos sprendimas paprastai skaidomas į du etapus:

- lygties šaknų atskyrimas (lokalizavimas),
- šaknų apskaičiavimas norimu tikslumu.

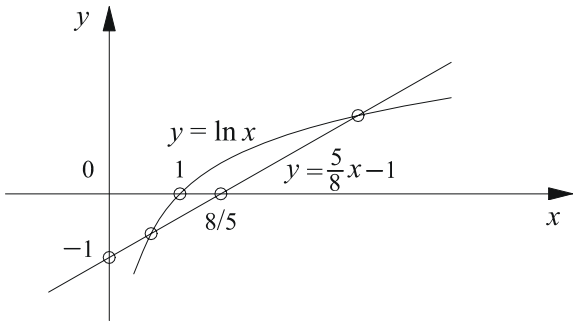
Šaknų apskaičiavimui naudosime kompleksinį Niutono metodą (**1.2** skyrius). Aptarkime su šaknų lokalizavimu susijusias problemas ir jų sprendimo būdus.

2.1.1. DAUGIANARIO ŠAKNŲ LOKALIZAVIMAS

Bendrų šio uždavinio sprendimo metodų nėra, todėl aptarkime dažniausiai taikomus jo sprendimo būdus:

- grafinis būdas,
- fizikinis būdas,
- specialūs būdai,

Ieškant daugianario šaknų lokalizavimo intervalų grafiniu būdu lygtis $f(x)=0$ perrašoma $f_1(x)=f_2(x)$. Tarkime, kad nesunku nubraižyti funkcijų $f_1(x)$ bei $f_2(x)$ grafikus. Akivaizdu, kad tų grafikų sankirtos taškų abscisės bus lygties $f(x)=0$ šaknys. Braižant $f_1(x)$ ir $f_2(x)$ grafikus, dažnai pakanka iš pradžių perteikti tik tų funkcijų kitimo pobūdį, paskui analiziškai nustatyti tikslesnę šaknų buvimo vietą (**2.1 pav.**).



2.1 pav. Lygties $5x-8\ln x=8$ **šaknų lokalizavimas**

Kaip matyti iš **2.1 pav**., lygtis $5x - 8\ln x = 8$ turi dvi šaknis, kurių izoliacijos intervalai (patikrinti analiziškai) yra $(0{,}1;\ 0{,}5)$ ir $(3;\ 4)$.

Jei sprendžiama lygtis $f(x) = 0$ apibūdina kurį nors fizikinį reiškinį, remdamiesi juo, galime nustatyti arba šaknų izoliacijos intervalus, arba intervalą, kuriame yra sprendžiamos lygties šaknys.

Teorema. Tarkime, kad $f(x)$ yra tolydžioji funkcija. Jei $f(a)\cdot f(b) < 0$ ir $f'(x)$ turi pastovų ženklą, kai $x \in [a;b]$, tai $(a;b)$ yra šaknies izoliacijos intervalas.

Kai žinomas intervalas, kuriame yra lygties šaknys, remiantis pateikta teorema, galima rasti šaknų izoliacijos intervalus.

Specialios priemonės (sprendimo būdai) taikomi atskiroms lygčių klasėms. Kadangi, sprendžiant įvairius matematikos bei inžinerinius uždavinius, dažnai tenka ieškoti daugianario šaknų, detaliau panagrinėsime atvejį, kai algebrinės lygties $f(z) = 0$ kairioji pusė yra aukštesnio nei pirmojo laipsnio daugianaris.

Tarkime, kad

$$f(z) = a_d z^d + a_{d-1} z^{d-1} + \ldots + a_1 z + a_0 \qquad \textbf{(2.1)}$$

yra *d*-tojo laipsnio kompleksinis daugianaris; $a_i \in \mathbf{C}$, $i = \overline{0,d}$, $d > 1$.

Suformuluosime ir įrodysime vieną su daugianario šaknų lokalizavimu susijusią teoremą.

Teorema. Jei $R \ge 1 + \dfrac{A}{|a_d|}$ ($A = \max\limits_{0 \le i \le d-1} |a_i|$), tai $|a_d| \cdot R^d > \left|a_{d-1}R^{d-1} + \ldots + a_1 R + a_0\right|$.

Įrodymas. Iš nelygybės $R \ge 1 + \dfrac{A}{|a_d|}$ betarpiškai gauname, kad $|a_d| \cdot R^d \ge \dfrac{A \cdot R^d}{R-1}$.

Kita vertus,

$$\left|a_{d-1}R^{d-1} + \ldots + a_1 R + a_0\right| \le |a_{d-1}| \cdot R^{d-1} + \ldots + |a_1| \cdot R + |a_0| \le A\left(R^{d-1} + \ldots + 1\right) = \frac{A\left(1 - R^d\right)}{1 - R} < \frac{A \cdot R^d}{R-1}.$$

Teorema įrodyta.

Taigi, jei $R \ge 1 + \dfrac{A}{|a_d|}$, tai $|f(z)| > 0$ su visais *z*, tenkinančiais sąlygą $|z| \ge R$, t.y. visos **(2.1)** daugianario šaknys patenka į skritulį, kurio spindulys *R*, o centras sutampa su koordinačių pradžios tašku.

Šis faktas gana svarbus, nustatinėjant (Niutono metodo pagalba) atskiras daugianario šaknis.

Niutono metodas dažnai naudojamas dėl savo paprastumo ir efektyvumo, tačiau turi ir tam tikrų ribotumų: egzistuoja tokie pradiniai taškai, kurie nekonverguoja nė į vieną iš daugianario šaknų (tai visi šaknų pritraukimo baseinų krašto taškai).

„Gerų“ pradinių taškų atrankai organizuoti tikslinga naudotis žemiau pateiktu teiginiu.

Teorema. Egzistuoja skaičius $n = n(d, \varepsilon)$ ($\varepsilon > 0$) toks, kad kiekvienai *d*-tojo laipsnio daugianario *f*, kurio visos šaknys yra vienetiniame skritulyje *D*, šakniai z_i galima parinkti tašką *z* $(|z| = 2)$, kuris po ne daugiau kaip

$$n = n(d, \varepsilon) \le \frac{9\pi d^4 f_d^2}{\varepsilon^2 \log 2} + 2 + \frac{|\log \varepsilon| + \log 13}{\log 2}$$

(Niutono funkcijos) iteracijų bus nutolęs nuo šaknies z_i atstumu, ne didesniu nei ε; čia $f_d = \frac{d^2(d-1)}{2(2d-1)}\binom{2d}{d}$.

Įrodymą galima rasti literatūroje [8].

Taigi, ieškant **(2.1)** daugianario šaknų, pakanka Niutono iteracinę funkciją (**(1.4)** išraišką) taikyti taškams *z* $(|z| = 2)$. Tai gerokai pagreitina pradinių taškų (pirmųjų šaknų aproksimacijų), o kartu ir pačių šaknų paiešką.

Tam, kad galėtume remtis pastarąja teorema, nagrinėjamo daugianario šaknis reikia „suspausti“ į vienetinį skritulį *D*. Tai nėra apribojimas, nes lokalizavus daugianario šaknis, t.y. suradus šaknis apimančio skritulio spindulį *R*, pakanka pereiti prie naujo kintamojo, būtent:

$$z = R\omega, \ (\omega, z \in \mathbf{C}). \qquad \textbf{(2.2)}$$

Tada daugianaris $f(z)$ (**(2.1)** išraiška) įgyja pavidalą

$$f(R\omega) = a_d R^d \omega^d + a_{d-1} R^{d-1} \omega^{d-1} + \ldots + a_1 R\omega + a_0. \qquad \textbf{(2.3)}$$

Akivaizdu, jog šio daugianario šaknys yra vienetiniame skritulyje *D*. Pasinaudoję Niutono metodu, randame **(2.3)** daugianario šaknis ω_i, $i = \overline{1,d}$. Tuo pačiu, pirminio daugianario šaknys bus lygios $z_i = R\omega_i$.

2.1.2. DAUGIANARIO ŠAKNŲ PRITRAUKIMO BASEINŲ VIZUALIZAVIMAS, TAIKANT PL-ALGORITMĄ

Nagrinėkime *d*-tojo laipsnio daugianarį $f(z)$ (**(2.1)** išraiška). Egzistuoja *d* kompleksinių skaičių z_i ($i = \overline{1,d}$) tokių, kad $f(z_i) = 0$, t.y. daugianaris $f(z)$ turi *d* šaknų (nulių). Kaip matėme, Niutono metodas leidžia visas jas apskaičiuoti. Pabandysime vizualizuoti šaknų pritraukimo baseinus.

Tirkime netiesinę dinaminę sistemą, veikiančią kompleksinėje plokštumoje, t.y.

$$\left\{\hat{\mathbf{C}}; F(z) = z - \frac{f(z)}{f'(z)}\right\};$$

čia $F(z)$ yra Niutono transformacija, susieta su funkcija $f(z)$. Tikėtina, kad orbita $\left\{f^{0n}(z_0)\right\}$, užgimstanti taške (spėjamoje šaknyje) $z_0 \in \mathbf{C}$, konverguos į vieną iš $f(z)$ šaknų z_i $\left(i \in \{1,2,...,d\}\right)$:

$$\lim_{n\to\infty} F^{on}(z_0) = z_i .$$

Tačiau, kas nutinka, jei z_0 yra toli nuo visų šaknų z_i? Gal z_0 orbita konverguoja į arčiausiai esančią $f(z)$ šaknį? O gal orbita klaidžioja be galo (tolsta į begalybę)?

Atlikime eksperimentą. Kiekvienai Niutono metodu rastai daugianario šakniai z_i $(i = \overline{1,d})$ taikome universalų PL („pabėgimo laiko“) - algoritmą, kurio veikimo rezultatas yra aibė taškų $z_0 \in \hat{\mathbf{C}}$, kurių orbitos konverguoja į z_i. Apibrėžiame stačiakampį $\mathcal{W} = \{(x,y) \in \mathbf{C} \mid -R \le x \le R, -R \le y \le R\}$ ir skritulį su centru taške z_i - $\mathcal{V} = \{z \in \mathbf{C} : |z - z_i| \le \varepsilon\}$. Srities $\mathcal{W}$ taškai, kurių orbitos pasiekia $\mathcal{V}$ per mažesnį nei fiksuotas iteracijų skaičių, yra spalvinami.

Pritaikykime PL-algoritmą lygčiai $z^4 - 1 = 0$ (**2.2 pav.**), kai $\mathcal{V}_i = \{z \in \mathbf{C} : |z - z_i| \le 0.0001\}$, $\mathcal{W} = \{(x,y) \in \mathbf{C} \mid -1 \le x \le 1, -1 \le y \le 1\}$, su visais $i = \overline{1,d}$, o tikrinamų iteracijų skaičius lygus 20.

Kaip matome (**2.2 pav.**), kiekvienos daugianario šaknies pritraukimo baseino krašto taškų aibės geometrinė struktūra yra gana sudėtinga. Detalesnė analizė leidžia teigti, jog šios aibės turi fraktalinį pobūdį. Dar daugiau, tai Žulija aibės. Pastebėsime, kad racionaliosios aukštesnio nei pirmojo laipsnio polinominės transformacijos $f : \hat{\mathbf{C}} \to \hat{\mathbf{C}}$ Žulija aibė yra aibė taškų, kurių orbitos nekonverguoja nė į vieną iš taškų (lygties $f(z) = 0$ šaknų) z_i, $i = \overline{1,d}$. Užpildytos Žulija aibės – tai (šiuo atveju) keturių nejudamųjų taškų (šaknų) z_i pritraukimo baseinai. Nenuspalvinti taškai atsiranda ir dėl apvalinimo paklaidų, per reto tinklelio bei fakto, kad, tikrinant konvergavimą į vieną iš šaknų, imama tik (šiuo atveju) 20 taškų iš kiekvienos orbitos.

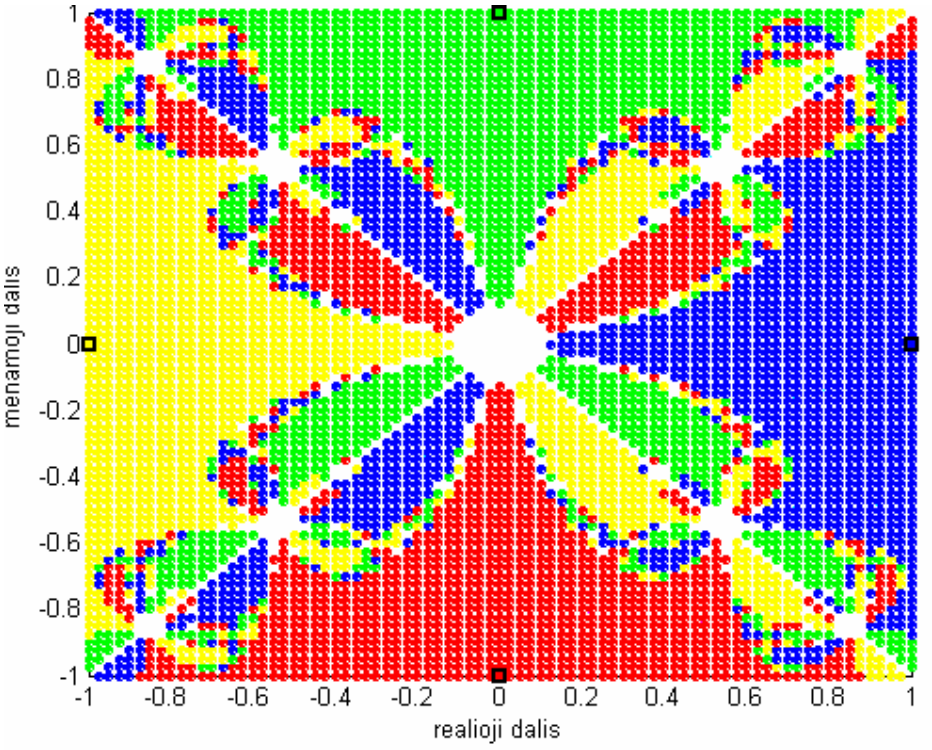
Parodysime, kad IFS teorija leidžia į problemą pažvelgti iš kitos pusės. Apibrėžkime atvirkštinį atvaizdį, susietą su $f(z) = z^4 - 1$. Tarkime $z \in \hat{\mathbf{C}}$ yra duotas taškas. Išspręskime ω atžvilgiu lygtį

$$z = \frac{3\omega^4 + 1}{4\omega^3},$$

t.y. išreiškime ω per z. Gauname lygtį

$$3\omega^4 - 4z\omega^3 + 1 = 0 ,$$

turinčią keturis sprendinius. Iš jų randame keturias funkcijas: parašykime $f^{-1}(z) = \{\omega_1(z), \omega_2(z), \omega_3(z), \omega_4(z)\}$. Tuomet Žulija aibė yra $IFS\left\{\hat{\mathbf{C}}; \omega_i, i = 1,2,3,4\right\}$ „atraktorius“.



2.2 pav. Lygties $z^4 - 1 = 0$ šaknų pritraukimo baseinai

Teorema. Tegul $f : \hat{\mathbf{C}} \to \hat{\mathbf{C}}$ yra Niutono transformacija, susieta su daugianariu $z^4 - 1$. Tegul $\varepsilon > 0$. Tegul $X = \hat{\mathbf{C}} \setminus \bigcup_{i=1}^{4} B(a_i, \varepsilon)$, kur $a_1 = 1,\ a_2 = -1,\ a_3 = i,$ ir $a_4 = -i$. Kaip ir anksčiau, apibrėžkime $W : \mathcal{H}(X) \to \mathcal{H}(X)$ tokiu būdu:

$$W(B) = \bigcup_{i=1}^{4} \omega_i(B) = f^{-1}(B) \text{ visiems } B \in \mathcal{H}(X).$$

Tuomet W yra tolydi, turi vienintelį nejudamąjį tašką J (Žulija aibė) ir

$$\lim_{n\to\infty} W^{on}(B) = J \text{ visiems } B \in \mathcal{H}(X), [1].$$

2.2. EKSPERIMENTO REZULTATŲ ANALIZĖ

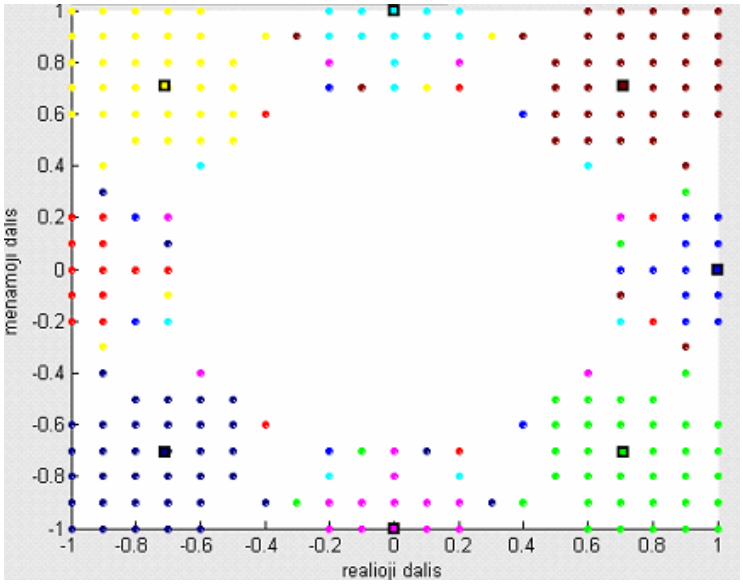
2.2.1. ŠAKNŲ PRITRAUKIMO BASEINŲ PRIKLAUSOMYBĖ NUO KAI KURIŲ PARAMETRŲ

Nagrinėkime daugianarį $f(z) = z^8 - 1,\ z \in \mathbf{C}$. Kompleksiniu Niutono metodu randame $z_1 = 1$, $z_2 = -1 + 6.587 \cdot 10^{-9} i$, $z_3 = 0.70711 - 0.70711i$, $z_4 = -0.70711 + 0.70711i$, $z_5 = 2.5274 \cdot 10^{-23} + i$,

$z_6 = -i$, $z_7 = -0.70711 - 0.70711i$ ir $z_8 = 0.70711 + 0.70711i$ tokius, kad $|f(z_i)| \le \varepsilon$, $i = \overline{1,8}$, t.y. nurodytu tikslumu ε apskaičiuojame daugianario $f(z)$ šaknis. Tuomet tiriame dinaminę sistemą

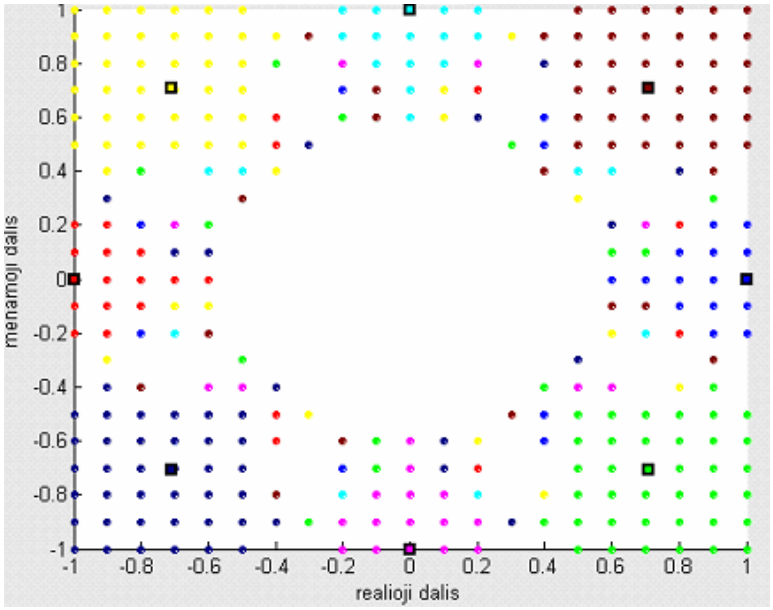
$$\left\{\hat{\mathbf{C}}; F(z) = \frac{7z^8 + 1}{8z^7}\right\};$$

čia $F(z)$ yra Niutono transformacija, susieta su daugianariu $f(z)$. Apibrėžiame stačiakampį $\mathcal{W} = \{(x, y) \in \mathbf{C} \mid -1 \le x \le 1, -1 \le y \le 1\}$ ir $\mathcal{V} = \{z \in \mathbf{C} : |z - z_i| \le \varepsilon\}$ - skritulį su centru taške z_i, visiems $i = \overline{1,8}$. Tikimės, kad parinkus pradinį tašką (pirmąją šaknies aproksimaciją) $z_0 \in \mathcal{W}$ orbita konverguos į vieną iš skaičių z_i, $i = \overline{1,8}$. Taikykime PL-algoritmą daugianario $f(z) = z^8 - 1$ ($z \in \mathbf{C}$) šaknų pritraukimo baseinams vizualizuoti.

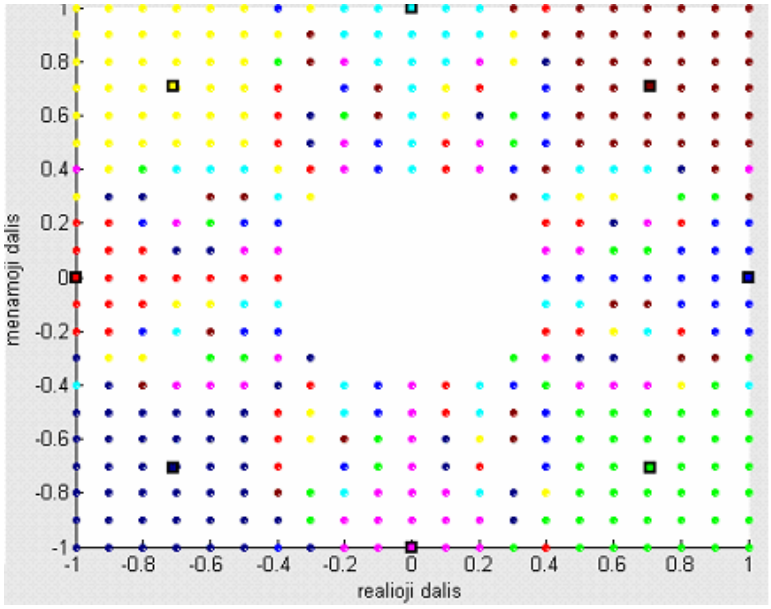


2.3 pav. Lygties $z^8 - 1 = 0$ šaknų pritraukimo baseinai, kai iteracijų skaičius 10, tikslumas $\varepsilon = 0.0001$, algoritmo žingsnis 0,1

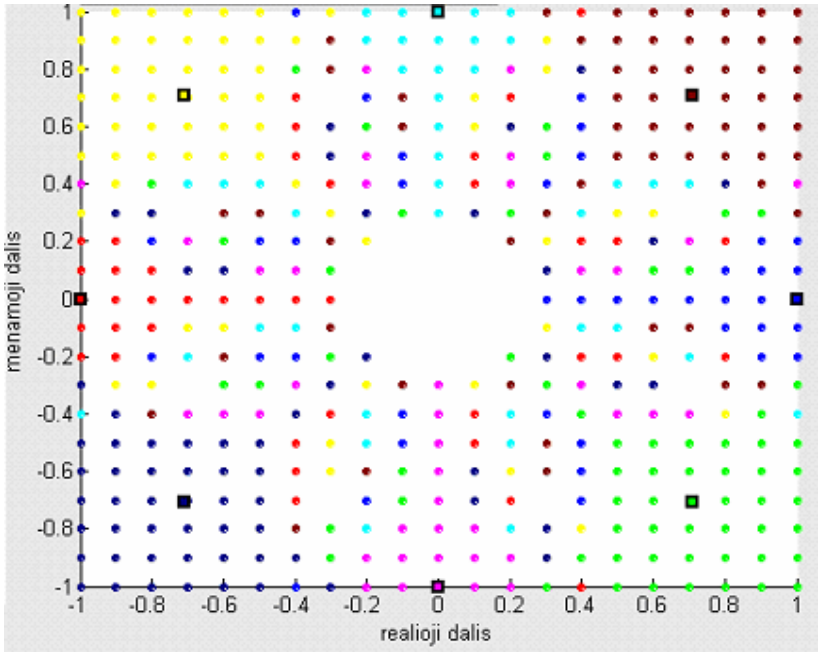
2.3 pav. matome užpildytas Žulija aibes – aštuonių nejudamųjų taškų (šaknų) z_i pritraukimo baseinus, kai $\varepsilon = 0.0001$, algoritmo žingsnis 0.1, o iteracijų skaičius lygus 10. Tikėtina, kad nenuspalvinti taškai atsiranda ne tik dėl apvalinimo paklaidų, bet ir per mažo iteracijų skaičiaus.



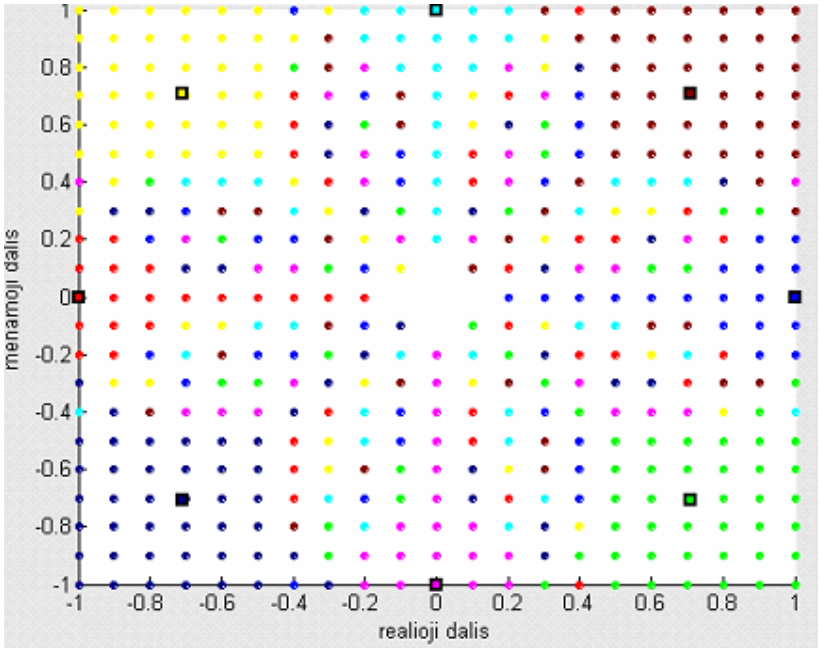
2.4 pav. Lygties $z^8 - 1 = 0$ šaknų pritraukimo baseinai, kai iteracijų skaičius 20, tikslumas $\varepsilon = 0.0001$, algoritmo žingsnis 0,1



2.5 pav. Lygties $z^8 - 1 = 0$ šaknų pritraukimo baseinai, kai iteracijų skaičius 40, tikslumas $\varepsilon = 0.0001$, algoritmo žingsnis 0,1



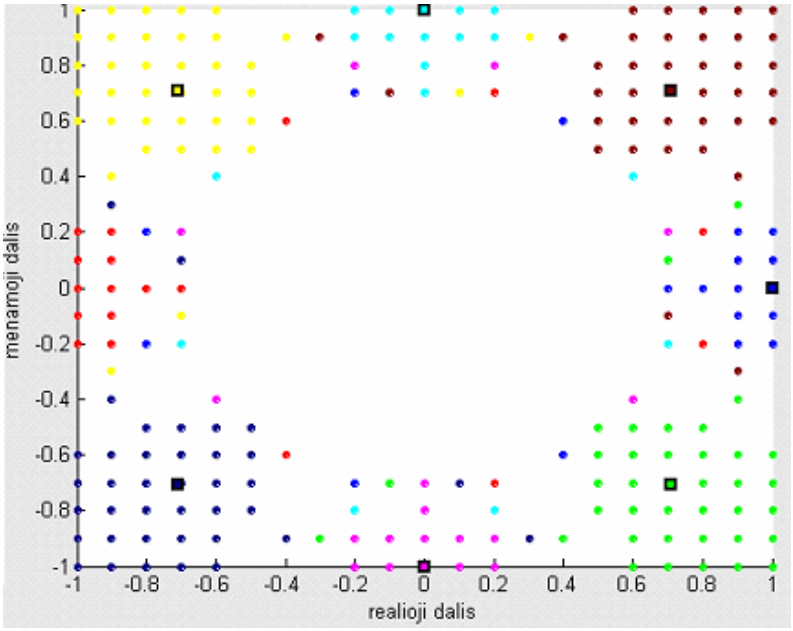
2.6 pav. Lygties $z^8 - 1 = 0$ šaknų pritraukimo baseinai, kai iteracijų skaičius 60, tikslumas $\varepsilon = 0.0001$, algoritmo žingsnis 0,1



2.7 pav. Lygties $z^8 - 1 = 0$ šaknų pritraukimo baseinai, kai iteracijų skaičius 100, tikslumas $\varepsilon = 0.0001$, algoritmo žingsnis 0,1

2.4 pav., 2.5 pav., 2.6 pav. ir 2.7 pav. matome daugianario $f(z) = z^8 - 1$ šaknų z_i pritraukimo baseinus, kai iteracijų skaičius atitinkamai lygus 20, 40, 60 ir 100, o kiti parametrai – algoritmo žingsnis ir tikslumas – nesikeičia. Žulija aibė yra aibė taškų, kurių orbitos nekonverguoja nė į vieną iš

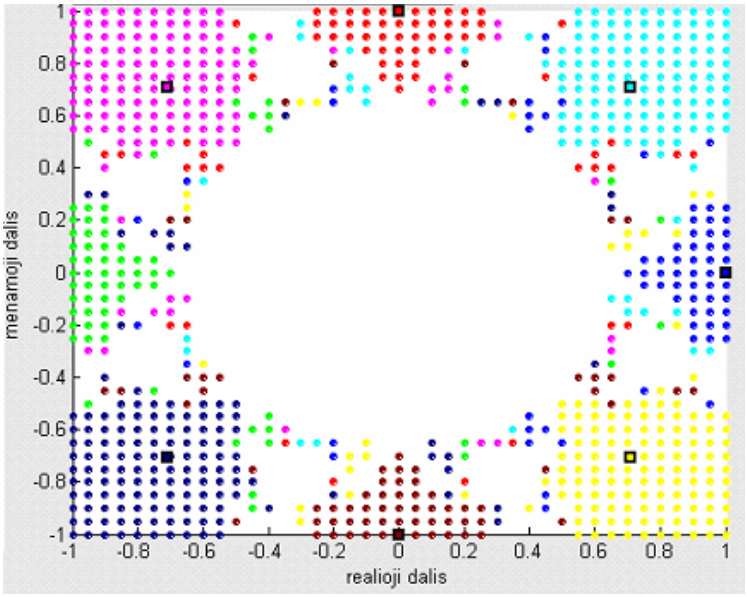
taškų (lygties $f(z)=0$ šaknų) z_i, $i=\overline{1,8}$; tačiau dabar galime teigti, kad taikant PL-algoritmą nenuspalvinti plotai atsiranda ir dėl per mažo iteracijų skaičiaus.



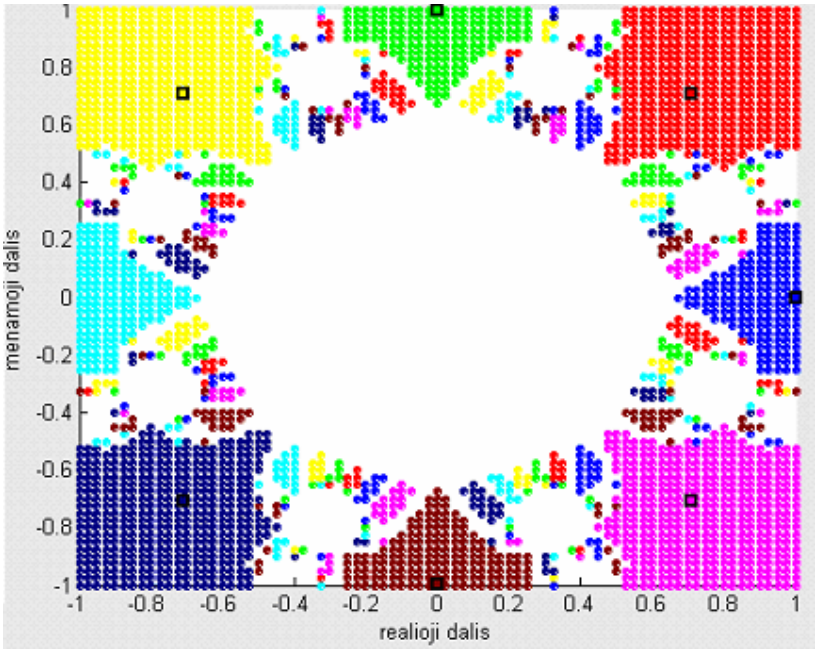
2.8 pav. Lygties $z^8-1=0$ šaknų pritraukimo baseinai, kai iteracijų skaičius 10, tikslumas $\varepsilon=0.00001$, algoritmo žingsnis 0,1

2.8 pav. pateikiami daugianario $f(z)=z^8-1$ šaknų z_i pritraukimo baseinai, padidinus tikslumą iki $\varepsilon=0.00001$. Matome, kad bent vizualiai Žulija aibė nesumažėjo, t. y. šiuo atveju tikslumas akivaizdžios įtakos neturi.

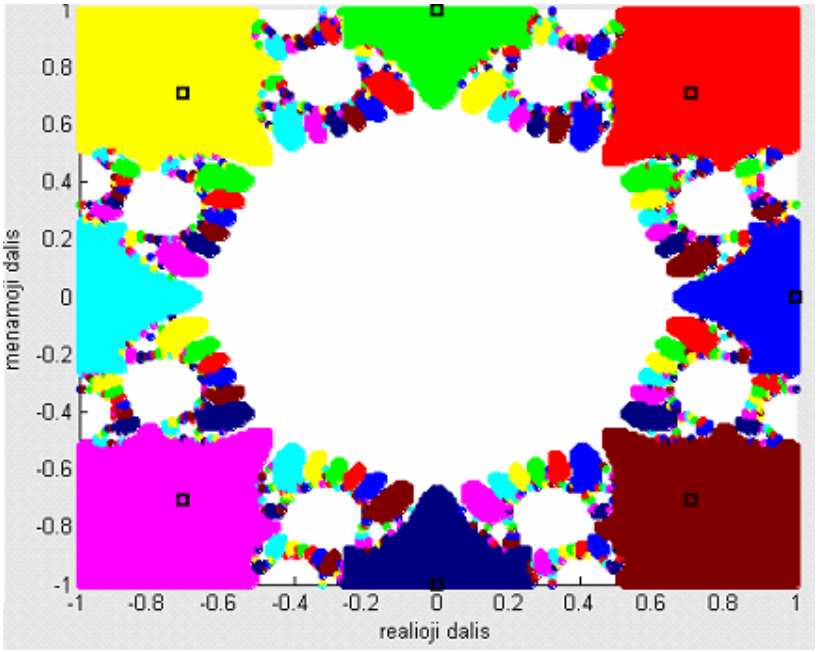
Sumažinkime algoritmo žingsnį iki 0,05 (**2.9 pav.**), 0,025 (**2.10pav.**) ir 0,01 (**2.11 pav.**). Matome, kad daugianrio šaknų pritraukimo baseinai mažinant žingsnį „tankėja“, tačiau, jei iteracijų skaičius yra nepakankamas (šiuo atveju jis nekito – tikrinant konvergavimą į vieną iš šaknų buvo imama 10 kiekvienos orbitos narių), negalime gauti tikslaus pritraukimo baseinų vaizdo koordinačių pradžios taško aplinkoje. Akivaizdu, kad mažinant algoritmo žingsnį, didėja pradinių taškų (pirmųjų šaknų) aproksimacijų aibė ir todėl skaičiavimai užtrunka ilgiau.



2.9 pav. Lygties $z^8 - 1 = 0$ šaknų pritraukimo baseinai, kai iteracijų skaičius 10, tikslumas $\varepsilon = 0.0001$, algoritmo žingsnis 0,05

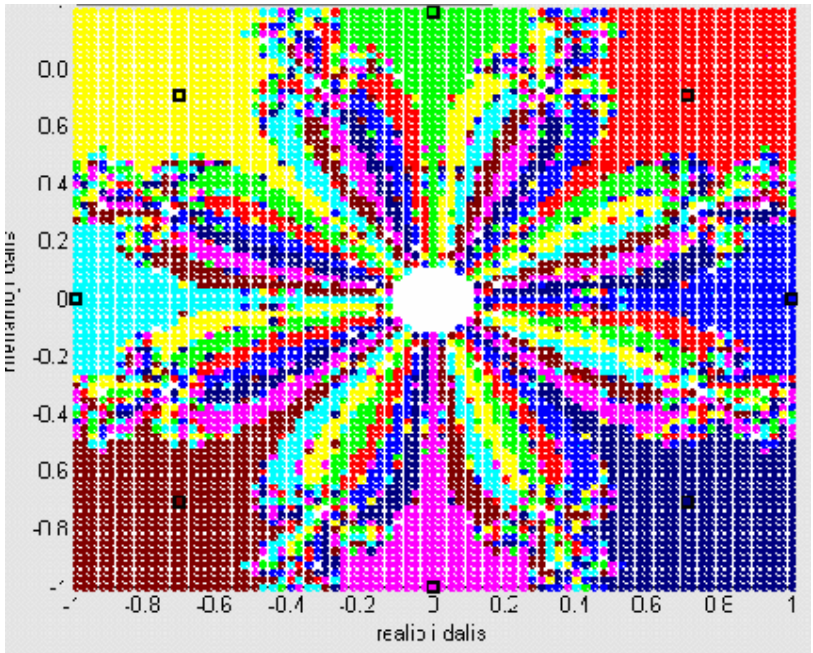


2.10 pav. Lygties $z^8 - 1 = 0$ šaknų pritraukimo baseinai, kai iteracijų skaičius 10, tikslumas $\varepsilon = 0.0001$, algoritmo žingsnis 0,025



2.11 pav. Lygties $z^8 - 1 = 0$ šaknų pritraukimo baseinai, kai iteracijų skaičius 10, tikslumas $\varepsilon = 0.0001$, algoritmo žingsnis 0,01

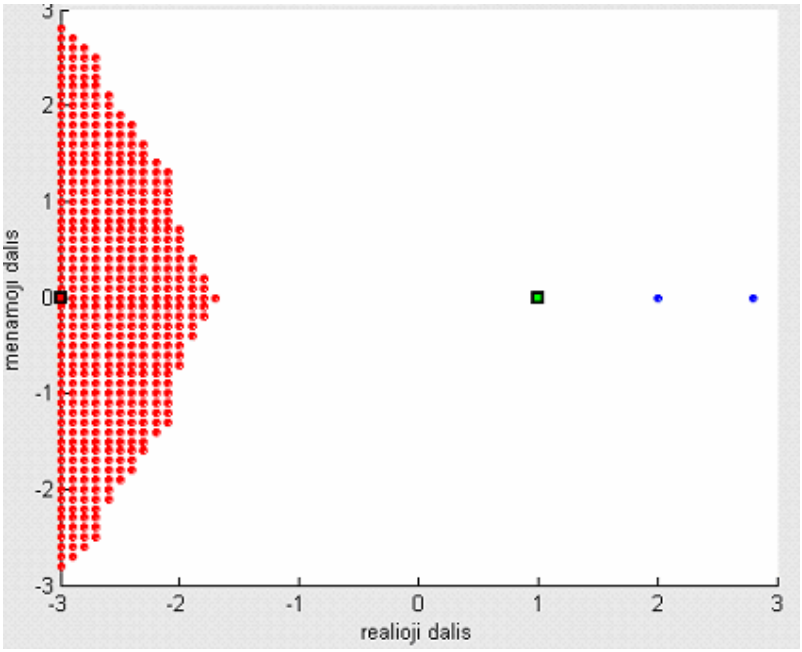
Pabandykime gauti kiek galima tikslesnį daugianario $f(z) = z^8 - 1$ šaknų pritraukimo baseinų vaizdą. Tikrinant konvergavimą į vieną iš šaknų, imkime 100 taškų iš kiekvienos orbitos, per šaknų lokalizacijos sritį „eikime“ 0,025 žingsniu, kai tikslumas $\varepsilon = 0.0001$ (**2.12 pav.**)



2.12 pav. Lygties $z^8 - 1 = 0$ šaknų pritraukimo baseinai, kai iteracijų skaičius 100, tikslumas $\varepsilon = 0.0001$, algoritmo žingsnis 0,025

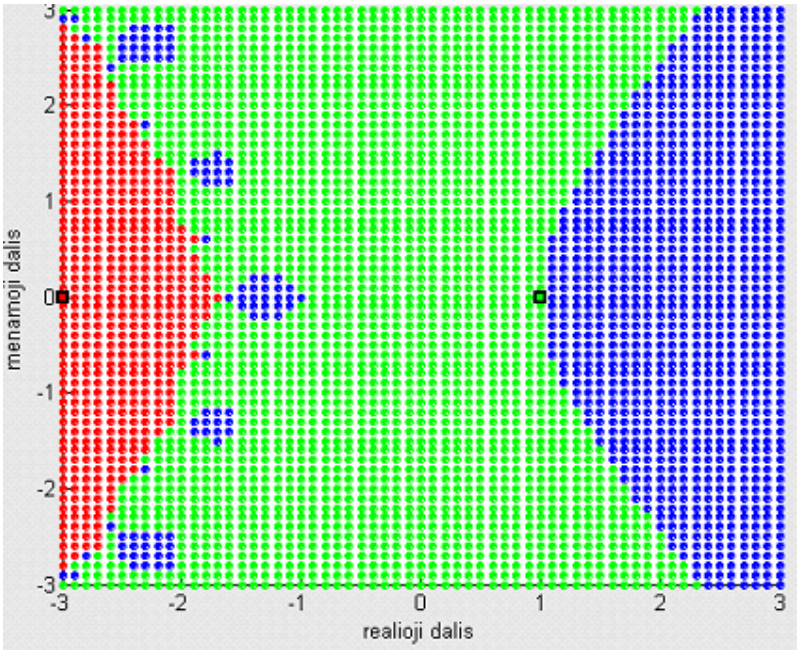
2.2.2. GRETIMŲ AR KARTOTINIŲ DAUGIANARIO ŠAKNŲ PRITRAUKIMO BASEINŲ VIZUALIZAVIMAS

Nagrinėkime daugianarį $f(z)=(z-1.001)(z-1)(z+3)=z^3+0.999z^2-5.002z+3.003,\ z\in\mathbf{C}$. Akivaizdu, kad šis daugianaris turi gretimas šaknis $z_1=1$ ir $z_2=1.001$, o trečioji šaknis lygi $z_3=-3$. Taikykime PL-algoritmą šių šaknų pritraukimo baseinams vizualizuoti (**2.13 pav.**).

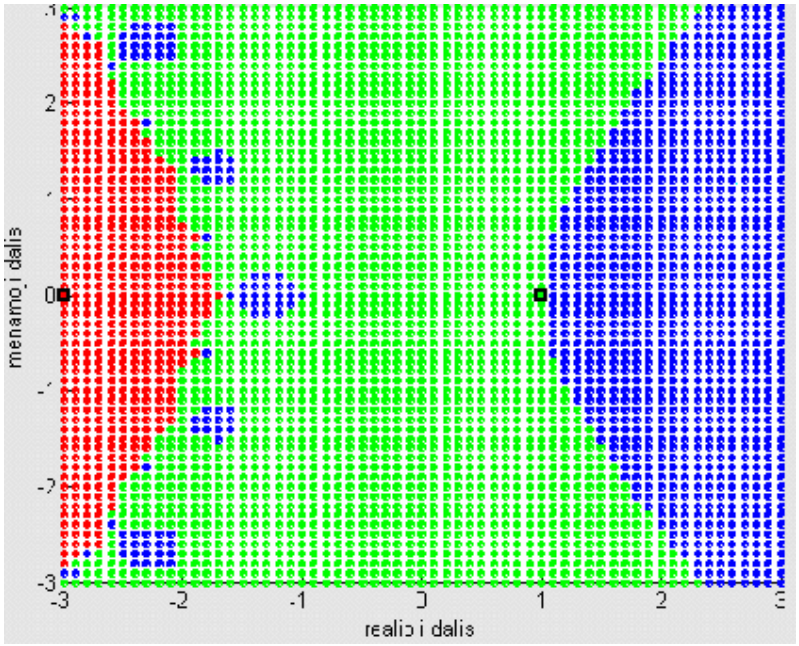


2.13 pav. Lygties $z^3+0.999z^2-5.002z+3.003=0$ šaknų pritraukimo baseinai, kai iteracijų skaičius 10, tikslumas $\varepsilon=0.0001$, algoritmo žingsnis 0,1

2.13 pav. matyti, kad šiuo atveju vizualizuojamas tik trečiosios šaknies $z_3=-3$ pritraukimo baseinas, tuo tarpu šaknų $z_1=1$ ir $z_2=1.001$ baseinai neatskiriami. Kaip pastebėjome aukščiau, algoritmo žingsnis įtakoja tik „piešinio“ tankumą, todėl norit rasti visų trijų šaknų pritraukimo baseinus, tikslinga būtų didinti iteracijų skaičių arba algoritmo tikslumą. Padidinkime iteracijų skaičių iki 40 (**2.14 pav.**), o tikslumą – iki $\varepsilon=0.0000001$ (**2.15 pav.**).



2.14 pav. Lygties $z^3 + 0.999z^2 - 5.002z + 3.003 = 0$ šaknų pritraukimo baseinai, kai iteracijų skaičius 40, tikslumas $\varepsilon = 0.0001$, algoritmo žingsnis 0,1



2.15 pav. Lygties $z^3 + 0.999z^2 - 5.002z + 3.003 = 0$ šaknų pritraukimo baseinai, kai iteracijų skaičius 40, tikslumas $\varepsilon = 0.0000001$, algoritmo žingsnis 0,1

Iš **2.15 pav.** matome, kad padidinus algoritmo tikslumą daugianario šaknų pritraukimo baseinų vaizdas nepasikeičia. Detaliau panagrinėkime, kaip nuo iteracijų skaičiaus priklauso lygties $f(z) = (z - 1.001)(z - 1)(z + 3) = z^3 + 0.999z^2 - 5.002z + 3.003 = 0$ šaknų tikslumas (**2.1 lentelė**).

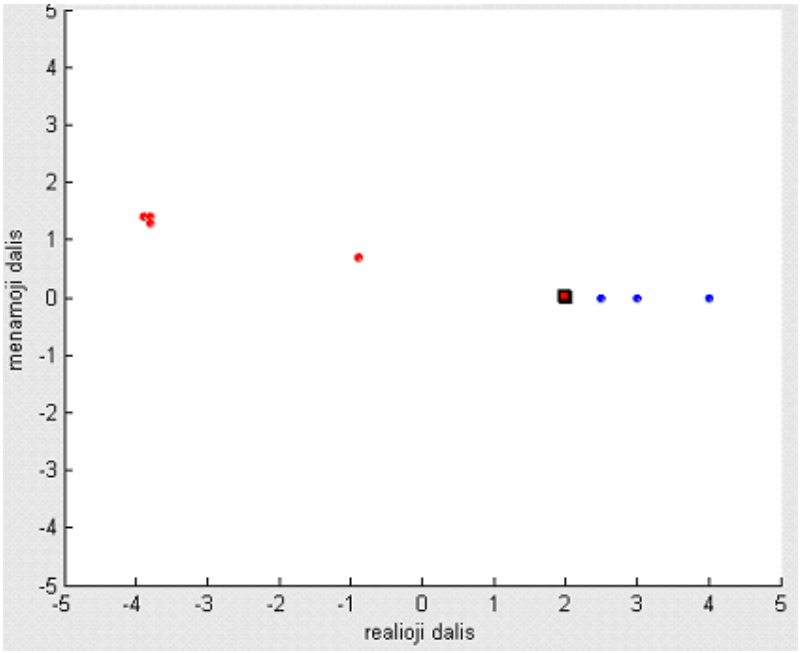
Matyti, kad, didinant iteracijų skaičių, galima gauti tikslesnę daugianario šaknies aproksimaciją. Tas ypač akivaizdu šiuo atveju, kai iš anksto žinome, kokias šaknis turime gauti.

2.1 lentelė

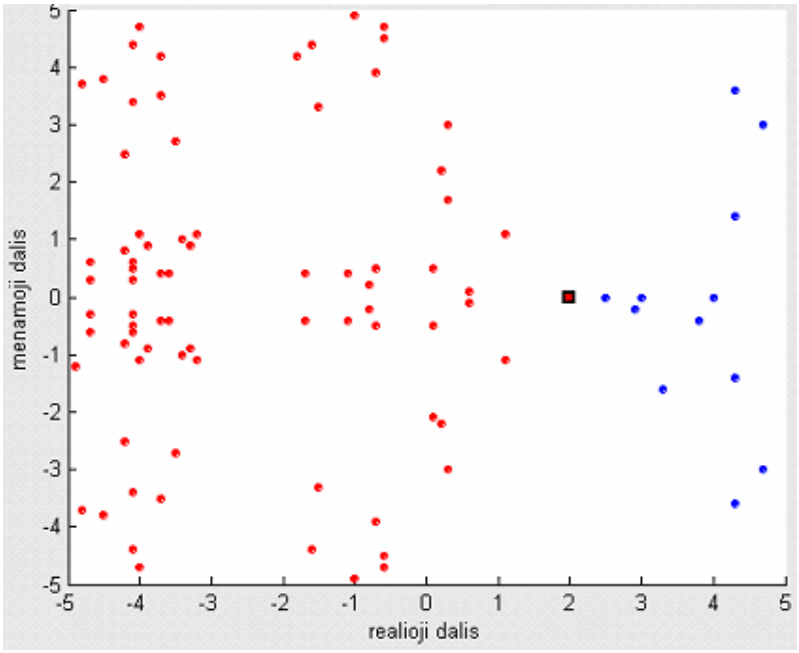
Daugianario šaknų tikslumo priklausomybė nuo iteracijų skaičiaus, kai algoritmo tikslumas $\varepsilon = 0.0001$, žingsnis 0,1

Iteracijų skaičius	$z_1 = 1$	$z_2 = 1.001$	$z_3 = -3$
10	$1.0034 - 0.0040276i$	1.0054	-3
20	$1 + 9.5124 \cdot 10^{-26} i$	1.001	-3
30	$1 - 2.557 \cdot 10^{-132} i$	1.001	-3
40	$1 + 4.2682 \cdot 10^{-264} i$	1.001	-3
60	1	1.001	-3

Nagrinėkime daugianarį $f(z) = (z-2)(z-2) = z^2 - 4z + 4,\ z \in \mathbf{C}$. Pastarasis turi kartotinę šaknį $z_{1,2} = 2$. Jos pritraukimo baseinas (arba dviejų gretimų – labai arti esančių – šaknų pritraukimo baseinai), gautas naudojant PL-algoritmą su skirtingais parametrais, matyti **2.16 pav.** ir **2.17 pav.** Pastebėsime, kad rastos šaknys abiem atvejais yra atitinkamai $z_1 = 2.0039$, $z_2 = 1.9943 + 0.0013344i$ ir $z_1 = 2$, $z_2 = 2 - 1.1981 \cdot 10^{-88} i$, t. y. kompleksiniu Niutono metodu randamos dvi gretimos šaknys.



2.16 pav. Lygties $z^2 - 4z + 4 = 0$ šaknų pritraukimo baseinai, kai iteracijų skaičius 10, tikslumas $\varepsilon = 0.0001$, algoritmo žingsnis 0,1



2.17 pav. Lygties $z^2 - 4z + 4 = 0$ šaknų pritraukimo baseinai, kai iteracijų skaičius 40, tikslumas $\varepsilon = 0.0000000001$, algoritmo žingsnis 0,1

Galime teigti, kad didinant iteracijų skaičių ir tikslumą, mažinant algoritmo žingsnį, galima gauti tikslesnius daugianario šaknų pritraukimo baseinus, net jei tarp šaknų yra kartotiniu ar gretimų.

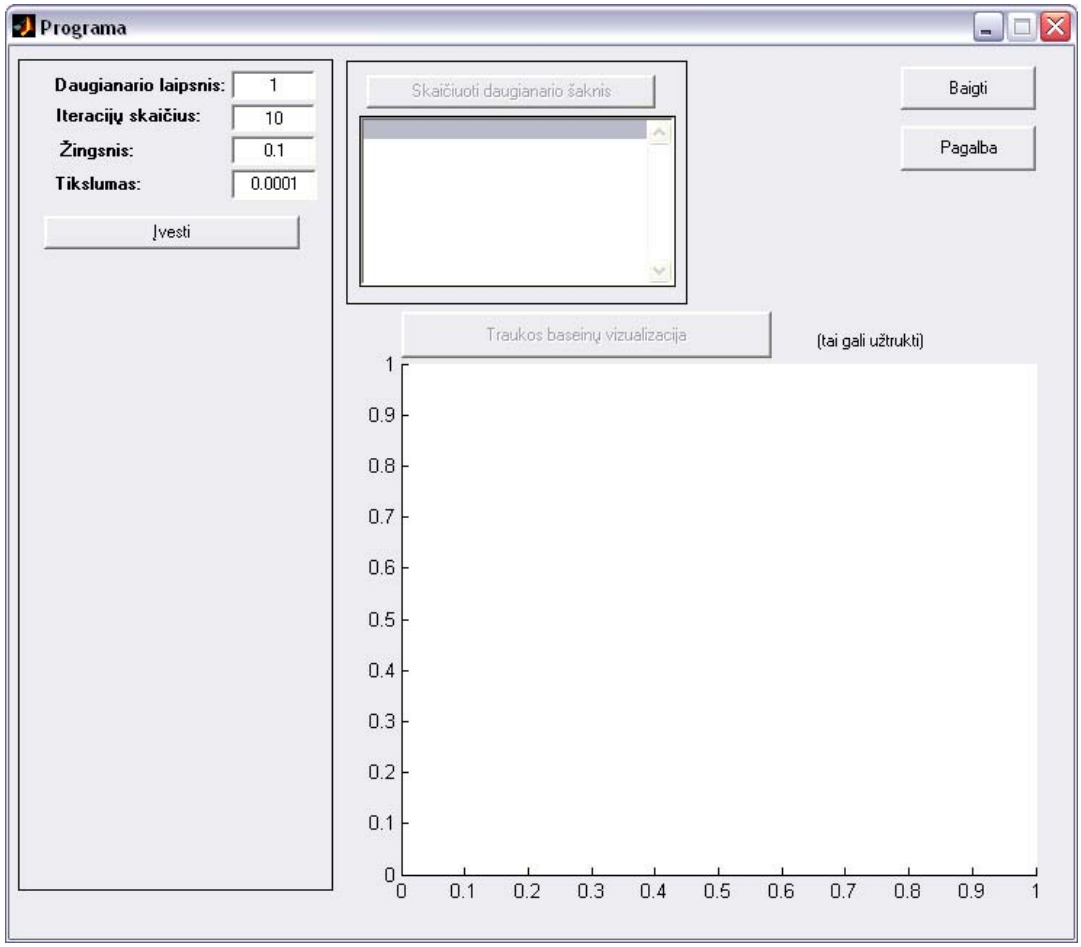
Daugiau Niutono fraktalų pavyzdžių pateikiama **1 priede**.

3. PROGRAMINĖ REALIZACIJA IR INSTRUKCIJA VARTOTOJUI

Nagrinėjamų Niutono fraktalų programinė realizacija atlikta matematiniu paketu MATLAB 6.5.1. Sukurta grafinė vartotojo sąsaja susidedanti iš dviejų pagrindinių dalių:

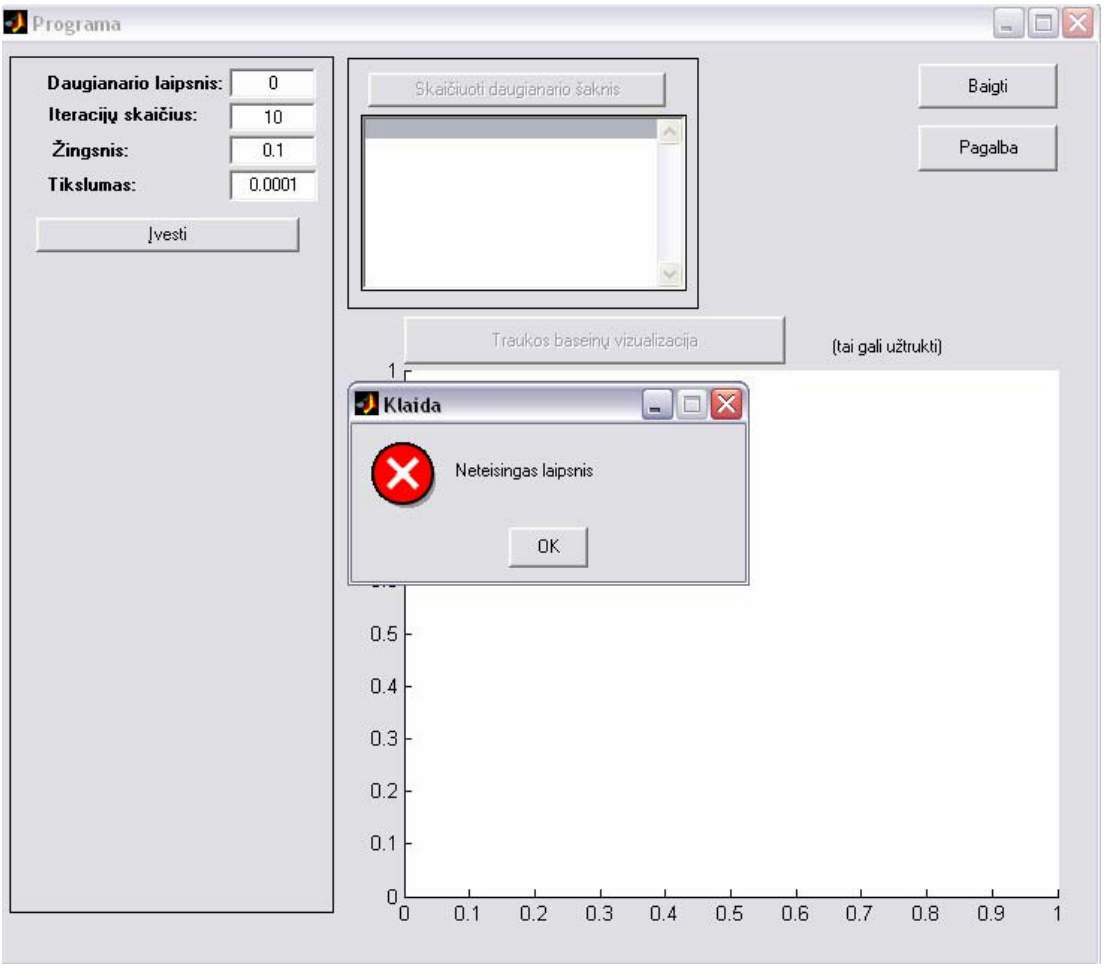
- daugianario šaknų radimas kompleksiniu Niutono metodu,
- šaknų pritraukimo baseinų braižymas PL-algoritmu.

Pagrindiniai programos parametrai yra daugianario laipsnis, iteracijų skaičius, algoritmo žingsnis ir tikslumas (**3.1 pav.**). Daugianario laipsnis yra sveikasis skaičius iš intervalo [1; 12]. Iteracijų skaičius nurodo, kiek iteracijų bus tikrinama iš kiekvieno pradinio taško orbitos. Šaknų lokalizacijos sritis „pereinama“ vartotojo pasirinktu algoritmo žingsniu. Algoritmo tikslumas nusako, kokiu didžiausiu atstumu turi būti daugianario nuliai, kad jie būtų laikomi dviem skirtingomis šaknimis, t. y. apibrėžia aplinką, kuriai priklausantys taškai laikomi ta pačia šaknimi, o tuo pačiu ir šaknų tikslumą.



3.1 pav. Programos langas

Jei bet kuris iš šių parametrų nenurodomas ar nurodomas klaidingai, programa išveda pranešimą apie klaidą (**3.2 pav.**)

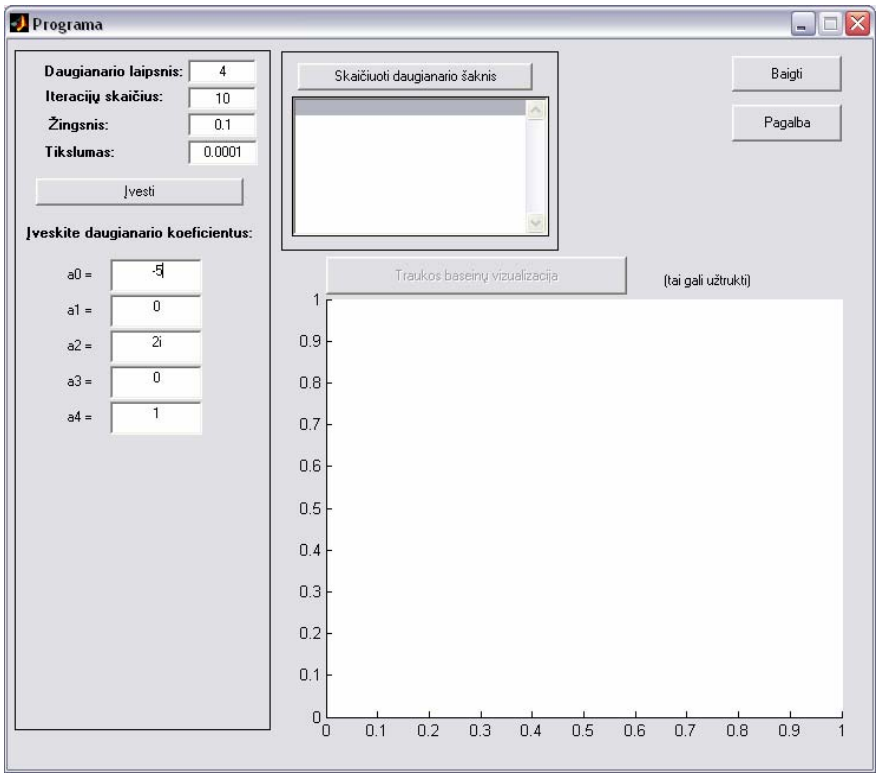


3.2 pav. Pranešimas apie klaidą

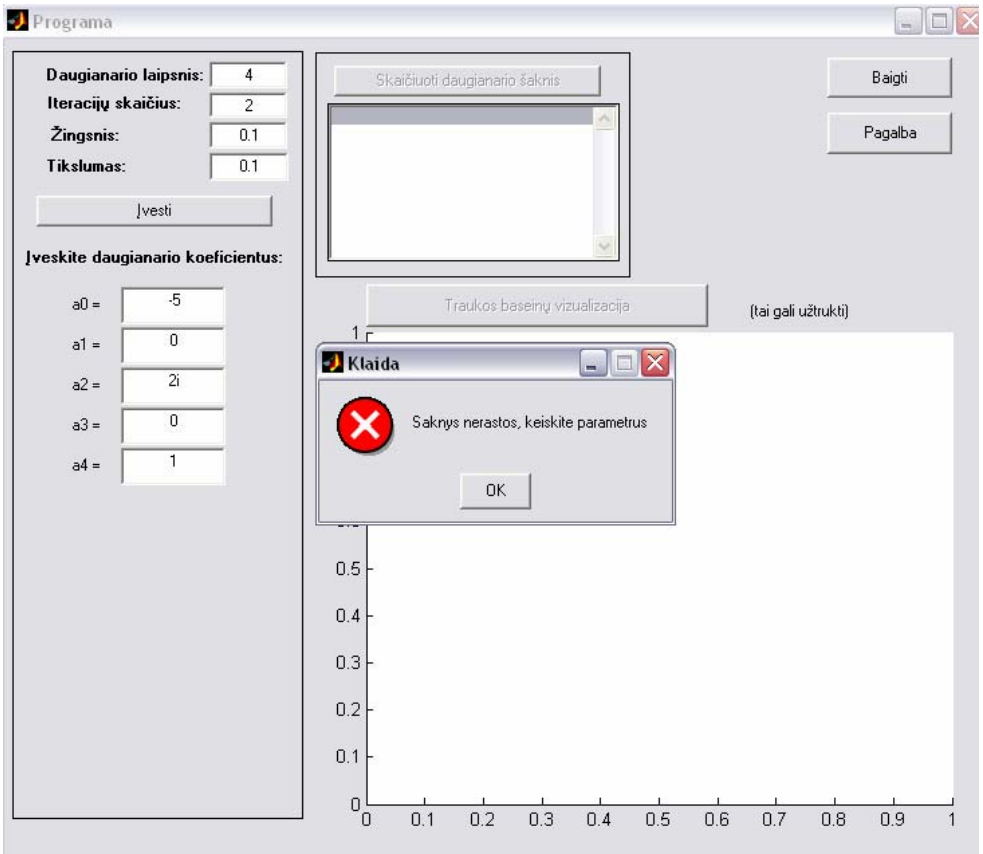
Paspaudus mygtuką „Įvesti“ nuskaitomi pasirinkti parametrai ir parodomi laukai daugianario šaknų koeficientų įvedimui (**3.3 pav.**). Koeficientai yra realūs arba kompleksiniai skaičiai. Kitu atveju programa išves pranešimą apie klaidą. Pranešimas bus išvedamas ir tuo atveju, jei koeficientas prie kintamojo su didžiausiu laipsniu bus lygus nuliui. Koeficientai nuskaitomi ir daugianario šaknys randamos paspaudus mygtuką „Skaičiuoti daugianario šaknis“.

Jei šaknys nerandamos, rekomenduojama keisti parametrus: didinti iteracijų skaičių, mažinti algoritmo žingsnį ir/arba mažinti tikslumą (**3.4 pav.**). Jei netinkama šaknų traukos baseinų grafiko kokybė, rekomenduojama keisti parametrus: didinti iteracijų skaičių, mažinti algoritmo žingsnį ir/arba didinti tikslumą. Be to, realaus skaičiaus trupmeninė dalis skiriama "." ženklu (tašku).

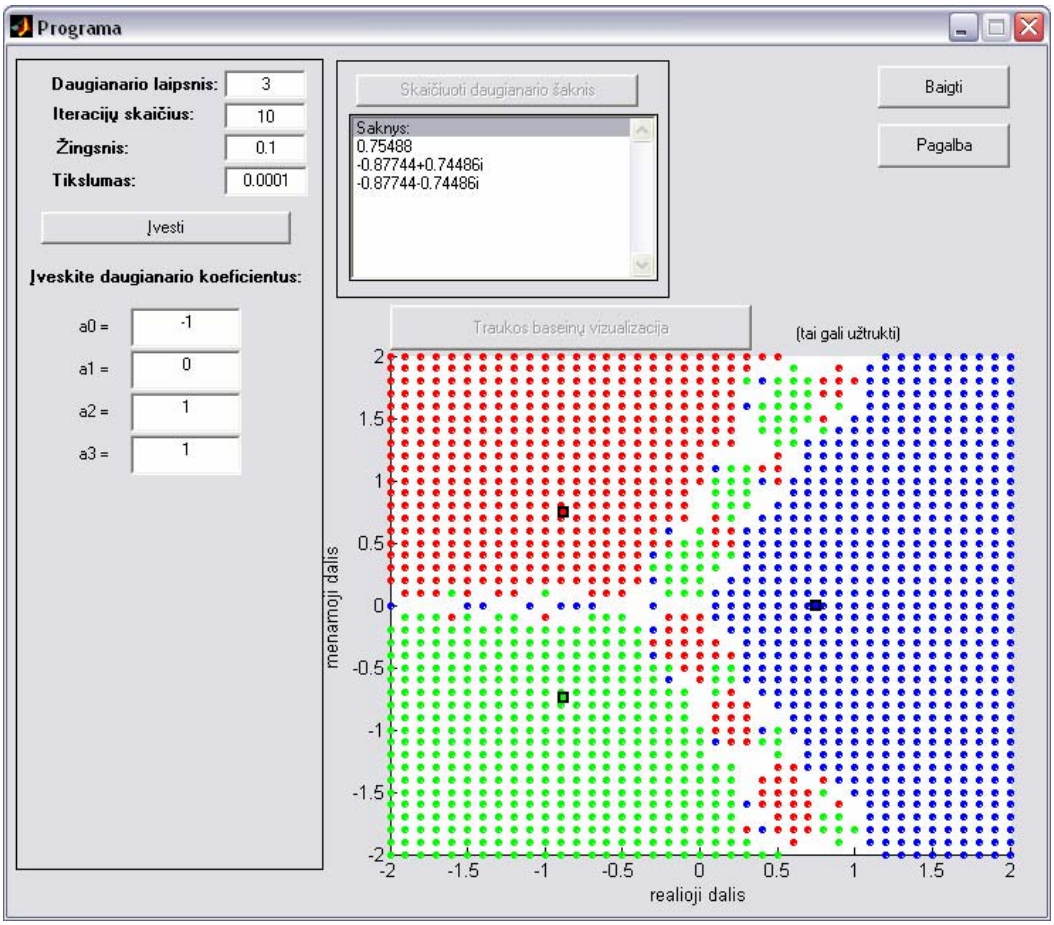
Antroje dalyje – apskaičiavus daugianario šaknis – PL-algoritmu vizualizuojami pritraukimo baseinai. Priklausomai nuo daugianario koeficientų, iteracijų skaičiaus ar algoritmo žingsnio tai gali užtrukti nuo kelių minučių iki kelių valandų (**3.5 pav.**).



3.3 pav. Daugianario koeficientų įvedimas



3.4 pav. Pranešimas apie klaidą



3.5 pav. Pritraukimo baseinų vizualizacija

Trumpą vartotojo instrukciją galima rasti paspaudus mygtuką „Pagalba“.

DISKUSIJA

Šiame darbe buvo analizuojama Niutono fraktalų Žulija aibės. Dažniausiai Žulija ir užpildytų Žulija aibių vaizdai gaunami, panaudojant „pabėgimo laiko“ algoritmą. Norėdami šį algoritmą naudoti kompleksinio daugianario šaknų vizualizacijai, turime nurodyti iteracijų skaičių, algoritmo tikslumą, žingsnį bei kompleksiniu Niutono metodu rasti daugianario šaknis.

Taikant Niutono metodą, buvo susidurta su pradinių taškų parinkimo problema. Tyrimo metu patvirtinta, kad pakanka Niutono iteracinę funkciją (**(1.4)** išraišką) taikyti taškams z $\left(|z| = 2\right)$. Tai gerokai pagreitina pradinių taškų (pirmųjų šaknų aproksimacijų), o kartu ir pačių šaknų paiešką. Tačiau daugianaris turi būti modifikuotas taip, kad jo šaknys būtų „suspaustos“ į vienetinį skritulį *D*. Tai nėra apribojimas, kai žinoma šaknų lokalizacijos sritis.

Darbe buvo pasiūlytas šios srities nustatymo būdas. Jei $R \geq 1 + \frac{A}{|a_d|}$, kur $A = \max\limits_{0 \leq i \leq d-1} |a_i|$, $a_i \in \mathbf{C}$, $i = \overline{0, d}$, $d > 1$, tai visos **(2.1)** daugianario šaknys patenka į skritulį, kurio spindulys *R*, o centras sutampa su koordinačių pradžios tašku. Šį skritulį apibūdinantis spindulys *R* nustato rėžius realioje ir menamoje ašyse. Tuomet, naudojant PL-algoritmą, pasirinktu žingsniu pereiname visus taškus, kurie patenka į rėžiais apribotą plotą. Taip gauname Niutono-Rafsono fraktalus ir lygiagrečiai galime analizuoti Žulija aibes bei užpildytas Žulija aibes.

Buvo tiriama šaknų pritraukimo baseinų (užpildytų Žulija aibių) priklausomybė nuo:

- iteracijų skaičiaus,
- žingsnio,
- algoritmo tikslumo.

Matematinio modeliavimo būdu nustatyta, kad kompleksinio daugianario šaknų (tuo pačiu ir Niutono fraktalo Žulija aibės) tikslumas priklauso nuo iteracijų skaičiaus, o algoritmo žingsnis turi įtakos fraktalo „tankumui“.

Nustatyta, kad, parinkus parametrus, įmanoma rasti gretimas kompleksinio daugianario šaknis ir vizualizuoti jų pritraukimo baseinus. Kai daugianaris turi kartotines šaknis, Niutono metodu jos randamos kaip gretimos – arti viena kitos esančios šaknys. Algoritmo tikslumas nurodo, kokiu didžiausiu atstumu turi būti daugianario nuliai, kad jie būtų laikomi skirtingomis šaknimis, t. y. apibrėžia aplinką, kuriai priklausantys taškai laikomi ta pačia šaknimi. Tokiu būdu, šis parametras įtakoja gretimų (kartotinių) šaknų paiešką.

Pastebėta, kad mažinant žingsnį ar didinant iteracijų skaičių, ilgėja skaičiavimų atlikimo laikas. Jis priklauso ir nuo daugianario koeficientų, nes būtent jais remiantis randama šaknų lokalizacijos sritis, t. y. plokštumos dalis, kuri analizuojama, taikant PL-algoritmą.

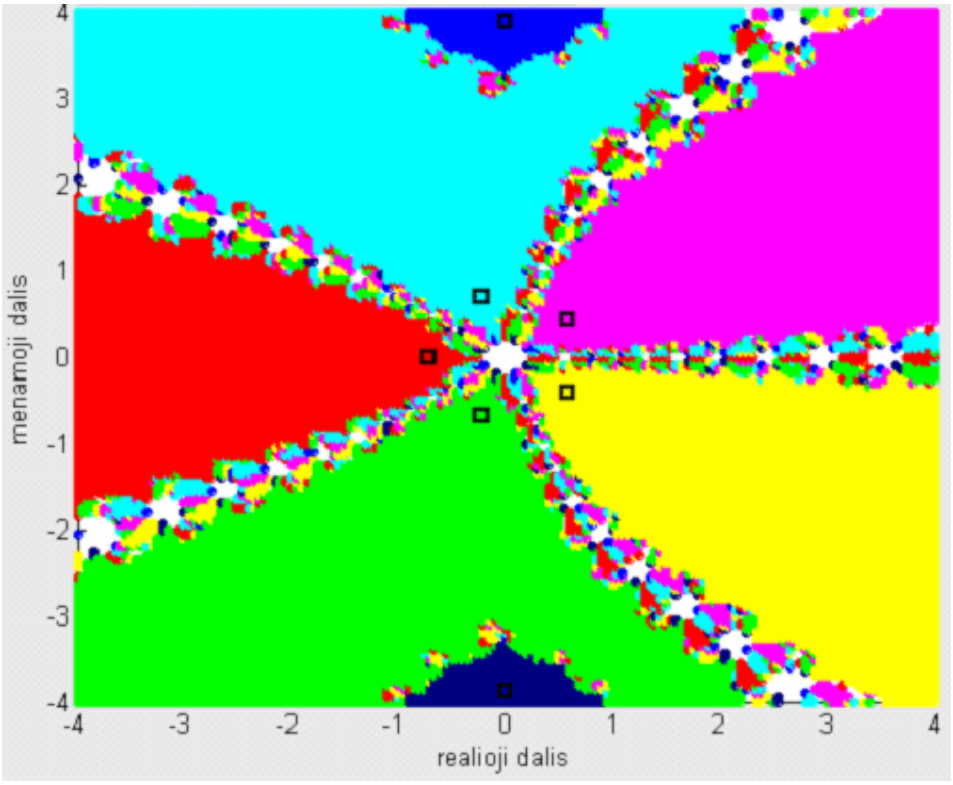
IŠVADOS

1. Pasiūlytas kompleksinio daugianario šaknų lokalizavimo būdas.
2. Realizuota ir ištirta Niutono algoritmo pradinių taškų (pirmųjų šaknų aproksimacijų) radimo metodika.
3. Matematinio modeliavimo būdu nustatyta, kad kompleksinio daugianario šaknų (tuo pačiu ir Niutono fraktalo Žulija aibės) tikslumas priklauso nuo pasirinkto iteracijų skaičiaus.
4. Pastebėta, kad naudojamas algoritmo žingsnis (gardelė) turi lemiamą įtaką šaknų pritraukimo baseinų (užpildytų Žulija aibių) geometrijai.
5. Nustatyta, kad atitinkamai fiksavus tikslumo parametro reikšmę $\varepsilon > 0$, pasiūlytų priemonių pagalba galima nustatyti ne tik gretimas, bet ir kartotines daugianario šaknis.

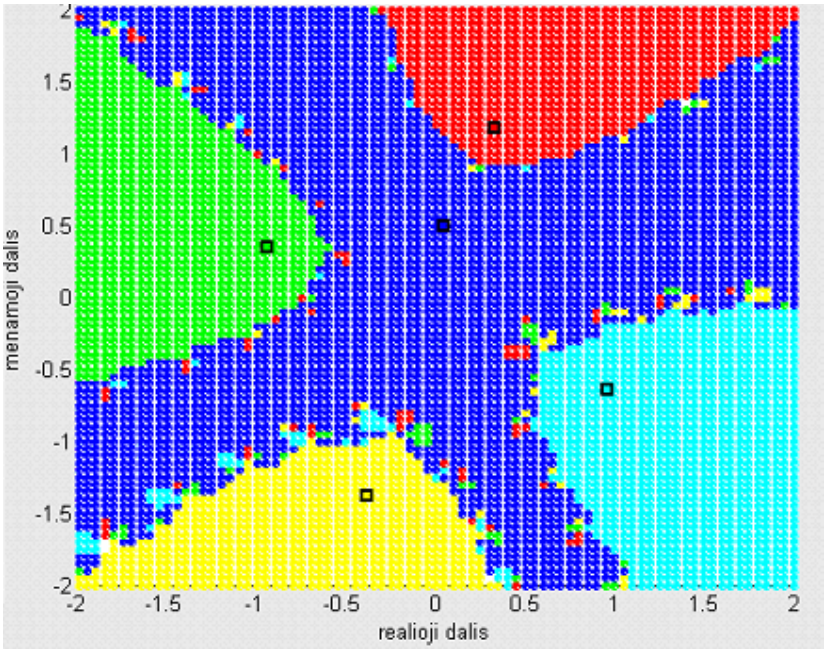
LITERATŪRA

1. Barnsley M. Fractals Everywhere.- San Diego, CA, USA: Academic Press Professional, 1988. – 394 p.
2. Dosinas G., Matiukienė I., Rimas J. Skaitiniai metodai. – Kaunas: Technologija, 2000. – 137 p.
3. Pergler M. The Complex Newton Method – 3 different ways, 1999. users.arczip.com/pergler /mp/documents/3newton.pdf
4. Pergler M. Newton's method and Newton basin fractals. http://users.arczip.com/pergler/mp /documents/newton/index.html
5. Plukas K. Skaitiniai metodai ir algoritmai. – Kaunas: Naujasis lankas, 2001. – 548 p.
6. Schleicher D. On The Number of Iteration for Newton's Method, 2000. journals.cambridge.org/ article_S0143385702000482
7. The Mandelbrot Set and Julia Sets. http://classes.yale.edu/fractals/MandelSet/ welcome.htm
8. The Escape - Time Algorithm. http://members.tripod.com/vismath1/javier/b2.htm
9. Valantinas J. Fraktalinė geometrija. – Kaunas: Technologija, 1999. – 185 p.
10. http://mokslasplius.lt/rizikos-fizika/node/180
11. http://mathworld.wolfram.com/NewtonsMethod.html
12. http://numbers.computation.free.fr/Constants/Algorithms/newton.html
13. http://aleph0.clarku.edu/~djoyce/newton/newton.html

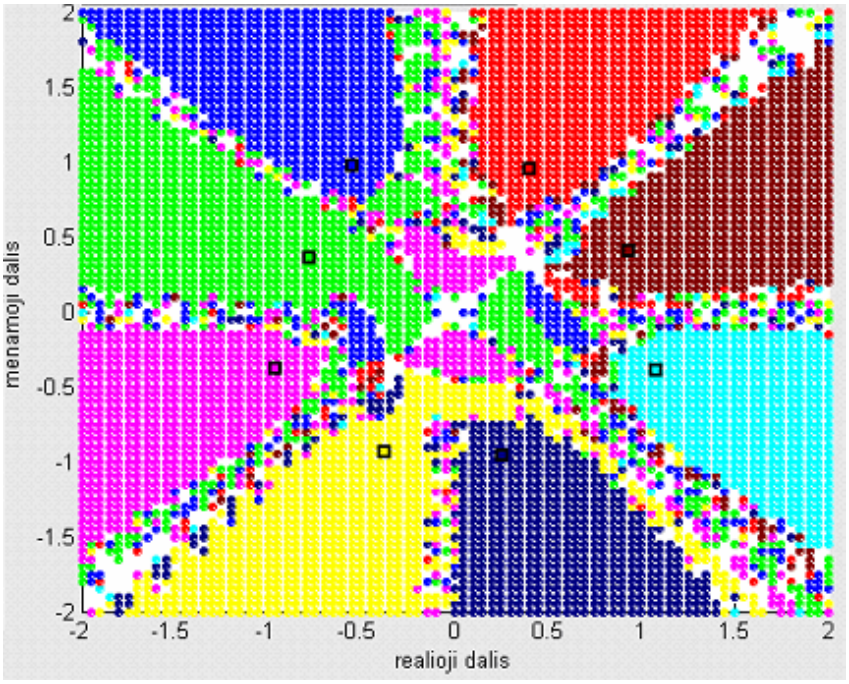
1 PRIEDAS. NIUTONO FRAKTALŲ PAVYZDŽIAI



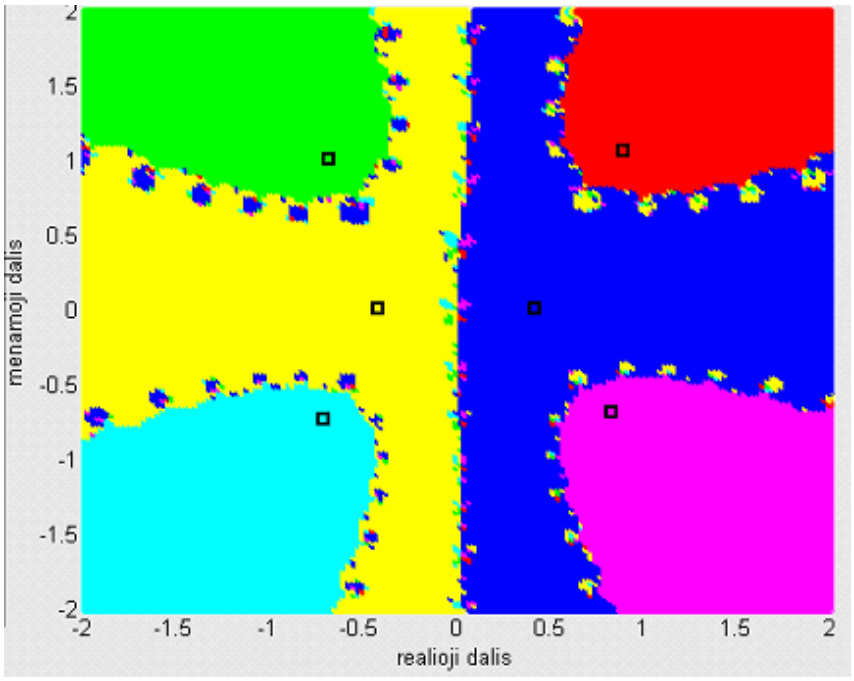
1 pav. Lygties $z^7 + 15z^5 + 3 = 0$ šaknų pritraukimo baseinai, kai iteracijų skaičius 20, tikslumas $\varepsilon = 0.0001$, algoritmo žingsnis 0,05



2 pav. Lygties $z^5 + z^3 + 2iz + 1 = 0$ šaknų pritraukimo baseinai, kai iteracijų skaičius 20, tikslumas $\varepsilon = 0.0001$, algoritmo žingsnis 0,05



3 pav. Lygties $4.2z^8 - 1.5z^5 + 3iz^3 + 2z + 5 = 0$ **šaknų pritraukimo baseinai, kai iteracijų skaičius 20, tikslumas** $\varepsilon = 0.00001$**, algoritmo žingsnis 0,05**



4 pav. Lygties $3z^6 - (2i+1)z^5 + 5.5z^2 - 1 = 0$ **šaknų pritraukimo baseinai, kai iteracijų skaičius 20, tikslumas** $\varepsilon = 0.0001$**, algoritmo žingsnis 0,025**

2 PRIEDAS. PROGRAMOS TEKSTAS

failas *Programa.m*

```
function varargout = Programa(varargin)
% PROGRAMA M-file for Programa.fig
%      PROGRAMA, by itself, creates a new PROGRAMA or raises the existing
%      singleton*.
%
%      H = PROGRAMA returns the handle to a new PROGRAMA or the handle to
%      the existing singleton*.
%
%      PROGRAMA('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in PROGRAMA.M with the given input arguments.
%
%      PROGRAMA('Property','Value',...) creates a new PROGRAMA or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Programa_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Programa_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Programa

% Last Modified by GUIDE v2.5 31-Mar-2007 21:20:26

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Programa_OpeningFcn, ...
                   'gui_OutputFcn',  @Programa_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin & isstr(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Programa is made visible.
function Programa_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Programa (see VARARGIN)

% Choose default command line output for Programa
handles.output = hObject;
guidata(hObject, handles);

%Pradines reiksmes
```

```
set(handles.text2,'Visible','off');

set(handles.edit13,'Visible','off');
set(handles.text6,'Visible','off');

set(handles.edit1,'Visible','off');
set(handles.text14,'Visible','off');

set(handles.edit2,'Visible','off');
set(handles.text5,'Visible','off');

set(handles.edit3,'Visible','off');
set(handles.text7,'Visible','off');

set(handles.edit4,'Visible','off');
set(handles.text8,'Visible','off');

set(handles.edit5,'Visible','off');
set(handles.text9,'Visible','off');

set(handles.edit6,'Visible','off');
set(handles.text11,'Visible','off');

set(handles.edit7,'Visible','off');
set(handles.text12,'Visible','off');

set(handles.edit8,'Visible','off');
set(handles.text13,'Visible','off');

set(handles.edit9,'Visible','off');
set(handles.text10,'Visible','off');

set(handles.edit10,'Visible','off');
set(handles.text15,'Visible','off');

set(handles.edit11,'Visible','off');
set(handles.text16,'Visible','off');

set(handles.edit12,'Visible','off');
set(handles.text17,'Visible','off');

set(handles.niuton_button,'Enable','off');
set(handles.algoritmas_button,'Enable','off');

% --- Outputs from this function are returned to the command line.
function varargout = Programa_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

% --- Executes during object creation, after setting all properties.
function edit13_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function edit13_Callback(hObject, eventdata, handles)
a0= str2double(get(hObject, 'String'));
if isnan(a0)
    set(hObject, 'String', 0);
    errordlg('Įveskite a0 koeficientą','Klaida');
end

% --- Executes during object creation, after setting all properties.
```

```
function edit1_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function edit1_Callback(hObject, eventdata, handles)
a1= str2double(get(hObject, 'String'));
if isnan(a1)
    set(hObject, 'String', 0);
    errordlg('Įveskite a1 koeficientą','Klaida');
end

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function edit2_Callback(hObject, eventdata, handles)
a2= str2double(get(hObject, 'String'));
if isnan(a2)
    set(hObject, 'String', 0);
    errordlg('Įveskite a2 koeficientą','Klaida');
end

% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function edit3_Callback(hObject, eventdata, handles)
a3= str2double(get(hObject, 'String'));
if isnan(a3)
    set(hObject, 'String', 0);
    errordlg('Įveskite a3 koeficientą','Klaida');
end


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function edit4_Callback(hObject, eventdata, handles)
a4= str2double(get(hObject, 'String'));
if isnan(a4)
    set(hObject, 'String', 0);
    errordlg('Įveskite a4 koeficientą','Klaida');
end

% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
```

```
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function edit5_Callback(hObject, eventdata, handles)
a5= str2double(get(hObject, 'String'));
if isnan(a5)
    set(hObject, 'String', 0);
    errordlg('Įveskite a5 koeficientą','Klaida');
end

% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function edit6_Callback(hObject, eventdata, handles)
a6= str2double(get(hObject, 'String'));
if isnan(a6)
    set(hObject, 'String', 0);
    errordlg('Įveskite a6 koeficientą','Klaida');
end

% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function edit7_Callback(hObject, eventdata, handles)
a7= str2double(get(hObject, 'String'));
if isnan(a7)
    set(hObject, 'String', 0);
    errordlg('Įveskite a7 koeficientą','Klaida');
end


% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function edit8_Callback(hObject, eventdata, handles)
a8= str2double(get(hObject, 'String'));
if isnan(a8)
    set(hObject, 'String', 0);
    errordlg('Įveskite a8 koeficientą','Klaida');
end

% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function edit9_Callback(hObject, eventdata, handles)
a9= str2double(get(hObject, 'String'));
if isnan(a9)
    set(hObject, 'String', 0);
    errordlg('Įveskite a9 koeficientą','Klaida');
end

% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function edit10_Callback(hObject, eventdata, handles)
a10= str2double(get(hObject, 'String'));
if isnan(a10)
    set(hObject, 'String', 0);
    errordlg('Įveskite a10 koeficientą','Klaida');
end

% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function edit11_Callback(hObject, eventdata, handles)
a11= str2double(get(hObject, 'String'));
if isnan(a11)
    set(hObject, 'String', 0);
    errordlg('Įveskite a11 koeficientą','Klaida');
end

% --- Executes during object creation, after setting all properties.
function edit12_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function edit12_Callback(hObject, eventdata, handles)
a12= str2double(get(hObject, 'String'));
if isnan(a12)
    set(hObject, 'String', 0);
    errordlg('Įveskite a12 koeficientą','Klaida');
end

% --- Executes during object creation, after setting all properties.
function laipsnis_edit_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function laipsnis_edit_Callback(hObject, eventdata, handles)
laipsnis = str2double(get(hObject, 'String'));
if isnan(laipsnis)
```

```
    set(hObject, 'String', 1);
    errordlg('Įveskite laipsnį','Klaida');
end

if ((round(laipsnis)~=laipsnis)||(laipsnis<=0))
    set(hObject, 'String', 1);
    errordlg('Įveskite teisingą laipsnį','Klaida');
end
% --- Executes during object creation, after setting all properties.
function spindulys_edit_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function spindulys_edit_Callback(hObject, eventdata, handles)
spindulys = str2double(get(hObject, 'String'));
if isnan(spindulys)
    set(hObject, 'String', 0.0001);
    errordlg('Įveskite tikslumą','Klaida');
end
if (spindulys<=0)
    set(hObject, 'String', 0.0001);
    errordlg('Įveskite teisingą tikslumą','Klaida');
end

% --- Executes during object creation, after setting all properties.
function iteracijos_edit_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function iteracijos_edit_Callback(hObject, eventdata, handles)
iteracijos = str2double(get(hObject, 'String'));
if isnan(iteracijos)
    set(hObject, 'String', 10);
    errordlg('Įveskite iteracijų skaičių','Klaida');
end
if ((round(iteracijos)~=iteracijos)||(iteracijos<=0))
    set(hObject, 'String', 10);
    errordlg('Įveskite teisingą iteracijų skaičių','Klaida');
end
% --- Executes during object creation, after setting all properties.
function epsilon_edit_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end

function epsilon_edit_Callback(hObject, eventdata, handles)
epsilon = str2double(get(hObject, 'String'));
if isnan(epsilon)
    set(hObject, 'String', 0.1);
    errordlg('Įveskite žingsnį','Klaida');
end
if (epsilon<=0)
    set(hObject, 'String', 0.1);
    errordlg('Įveskite teisingą žingsnį','Klaida');
end
% --- Executes on button press in Baigti_button.
```

```
function Baigti_button_Callback(hObject, eventdata, handles)
delete(handles.figure1);

% --- Executes on button press in algoritmas_button.
function algoritmas_button_Callback(hObject, eventdata, handles)
% hObject    handle to algoritmas_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global S a laipsnis;
set(handles.text25, 'String',{['Palaukite...']});

epsilon = str2double(get(handles.epsilon_edit, 'String'));
spindulys = str2double(get(handles.spindulys_edit, 'String'));
iteracijos = str2double(get(handles.iteracijos_edit, 'String'));

PL(iteracijos, epsilon, spindulys);
set(handles.text25, 'String',{['(tai gali užtrukti)']});
set(handles.algoritmas_button,'Enable','off');

% --- Executes on button press in ivesti_button.
function ivesti_button_Callback(hObject, eventdata, handles)
% hObject    handle to ivesti_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
laipsnis = str2num(get(handles.laipsnis_edit, 'String'));
global S;
S = [];
switch laipsnis
     case 1
set(handles.edit13,'Visible','on');
set(handles.text6,'Visible','on');

set(handles.edit1,'Visible','on');
set(handles.text14,'Visible','on');

set(handles.edit2,'Visible','off');
set(handles.text5,'Visible','off');

set(handles.edit3,'Visible','off');
set(handles.text7,'Visible','off');

set(handles.edit4,'Visible','off');
set(handles.text8,'Visible','off');

set(handles.edit5,'Visible','off');
set(handles.text9,'Visible','off');

set(handles.edit6,'Visible','off');
set(handles.text11,'Visible','off');

set(handles.edit7,'Visible','off');
set(handles.text12,'Visible','off');

set(handles.edit8,'Visible','off');
set(handles.text13,'Visible','off');

set(handles.edit9,'Visible','off');
set(handles.text10,'Visible','off');

set(handles.edit10,'Visible','off');
set(handles.text15,'Visible','off');

set(handles.edit11,'Visible','off');
set(handles.text16,'Visible','off');
```

```
set(handles.edit12,'Visible','off');
set(handles.text17,'Visible','off');

set(handles.text2,'Visible','on');
set(handles.niuton_button,'Enable','on');
    case 2
set(handles.edit13,'Visible','on');
set(handles.text6,'Visible','on');

set(handles.edit1,'Visible','on');
set(handles.text14,'Visible','on');

set(handles.edit2,'Visible','on');
set(handles.text5,'Visible','on');

set(handles.edit3,'Visible','off');
set(handles.text7,'Visible','off');

set(handles.edit4,'Visible','off');
set(handles.text8,'Visible','off');

set(handles.edit5,'Visible','off');
set(handles.text9,'Visible','off');

set(handles.edit6,'Visible','off');
set(handles.text11,'Visible','off');

set(handles.edit7,'Visible','off');
set(handles.text12,'Visible','off');

set(handles.edit8,'Visible','off');
set(handles.text13,'Visible','off');

set(handles.edit9,'Visible','off');
set(handles.text10,'Visible','off');

set(handles.edit10,'Visible','off');
set(handles.text15,'Visible','off');

set(handles.edit11,'Visible','off');
set(handles.text16,'Visible','off');

set(handles.edit12,'Visible','off');
set(handles.text17,'Visible','off');

set(handles.text2,'Visible','on');
set(handles.niuton_button,'Enable','on');
    case 3
set(handles.edit13,'Visible','on');
set(handles.text6,'Visible','on');

set(handles.edit1,'Visible','on');
set(handles.text14,'Visible','on');

set(handles.edit2,'Visible','on');
set(handles.text5,'Visible','on');

set(handles.edit3,'Visible','on');
set(handles.text7,'Visible','on');

set(handles.edit4,'Visible','off');
set(handles.text8,'Visible','off');
```

```
set(handles.edit5,'Visible','off');
set(handles.text9,'Visible','off');

set(handles.edit6,'Visible','off');
set(handles.text11,'Visible','off');

set(handles.edit7,'Visible','off');
set(handles.text12,'Visible','off');

set(handles.edit8,'Visible','off');
set(handles.text13,'Visible','off');

set(handles.edit9,'Visible','off');
set(handles.text10,'Visible','off');

set(handles.edit10,'Visible','off');
set(handles.text15,'Visible','off');

set(handles.edit11,'Visible','off');
set(handles.text16,'Visible','off');

set(handles.edit12,'Visible','off');
set(handles.text17,'Visible','off');

set(handles.text2,'Visible','on');
set(handles.niuton_button,'Enable','on');
    case 4
set(handles.edit13,'Visible','on');
set(handles.text6,'Visible','on');

set(handles.edit1,'Visible','on');
set(handles.text14,'Visible','on');

set(handles.edit2,'Visible','on');
set(handles.text5,'Visible','on');

set(handles.edit3,'Visible','on');
set(handles.text7,'Visible','on');

set(handles.edit4,'Visible','on');
set(handles.text8,'Visible','on');

set(handles.edit5,'Visible','off');
set(handles.text9,'Visible','off');

set(handles.edit6,'Visible','off');
set(handles.text11,'Visible','off');

set(handles.edit7,'Visible','off');
set(handles.text12,'Visible','off');

set(handles.edit8,'Visible','off');
set(handles.text13,'Visible','off');

set(handles.edit9,'Visible','off');
set(handles.text10,'Visible','off');

set(handles.edit10,'Visible','off');
set(handles.text15,'Visible','off');

set(handles.edit11,'Visible','off');
set(handles.text16,'Visible','off');

set(handles.edit12,'Visible','off');
```

```
set(handles.text17,'Visible','off');

set(handles.text2,'Visible','on');
set(handles.niuton_button,'Enable','on');
    case 5
set(handles.edit13,'Visible','on');
set(handles.text6,'Visible','on');

set(handles.edit1,'Visible','on');
set(handles.text14,'Visible','on');

set(handles.edit2,'Visible','on');
set(handles.text5,'Visible','on');

set(handles.edit3,'Visible','on');
set(handles.text7,'Visible','on');

set(handles.edit4,'Visible','on');
set(handles.text8,'Visible','on');

set(handles.edit5,'Visible','on');
set(handles.text9,'Visible','on');

set(handles.edit6,'Visible','off');
set(handles.text11,'Visible','off');

set(handles.edit7,'Visible','off');
set(handles.text12,'Visible','off');

set(handles.edit8,'Visible','off');
set(handles.text13,'Visible','off');

set(handles.edit9,'Visible','off');
set(handles.text10,'Visible','off');

set(handles.edit10,'Visible','off');
set(handles.text15,'Visible','off');

set(handles.edit11,'Visible','off');
set(handles.text16,'Visible','off');

set(handles.edit12,'Visible','off');
set(handles.text17,'Visible','off');

set(handles.text2,'Visible','on');
set(handles.niuton_button,'Enable','on');

    case 6
set(handles.edit13,'Visible','on');
set(handles.text6,'Visible','on');

set(handles.edit1,'Visible','on');
set(handles.text14,'Visible','on');

set(handles.edit2,'Visible','on');
set(handles.text5,'Visible','on');

set(handles.edit3,'Visible','on');
set(handles.text7,'Visible','on');

set(handles.edit4,'Visible','on');
set(handles.text8,'Visible','on');

set(handles.edit5,'Visible','on');
```

```
set(handles.text9,'Visible','on');

set(handles.edit6,'Visible','on');
set(handles.text11,'Visible','on');

set(handles.edit7,'Visible','off');
set(handles.text12,'Visible','off');

set(handles.edit8,'Visible','off');
set(handles.text13,'Visible','off');

set(handles.edit9,'Visible','off');
set(handles.text10,'Visible','off');

set(handles.edit10,'Visible','off');
set(handles.text15,'Visible','off');

set(handles.edit11,'Visible','off');
set(handles.text16,'Visible','off');

set(handles.edit12,'Visible','off');
set(handles.text17,'Visible','off');

set(handles.text2,'Visible','on');
set(handles.niuton_button,'Enable','on');
    case 7
set(handles.edit13,'Visible','on');
set(handles.text6,'Visible','on');

set(handles.edit1,'Visible','on');
set(handles.text14,'Visible','on');

set(handles.edit2,'Visible','on');
set(handles.text5,'Visible','on');

set(handles.edit3,'Visible','on');
set(handles.text7,'Visible','on');

set(handles.edit4,'Visible','on');
set(handles.text8,'Visible','on');

set(handles.edit5,'Visible','on');
set(handles.text9,'Visible','on');

set(handles.edit6,'Visible','on');
set(handles.text11,'Visible','on');

set(handles.edit7,'Visible','on');
set(handles.text12,'Visible','on');

set(handles.edit8,'Visible','off');
set(handles.text13,'Visible','off');

set(handles.edit9,'Visible','off');
set(handles.text10,'Visible','off');

set(handles.edit10,'Visible','off');
set(handles.text15,'Visible','off');

set(handles.edit11,'Visible','off');
set(handles.text16,'Visible','off');

set(handles.edit12,'Visible','off');
set(handles.text17,'Visible','off');
```

```
set(handles.text2,'Visible','on');
set(handles.niuton_button,'Enable','on');
    case 8
set(handles.edit13,'Visible','on');
set(handles.text6,'Visible','on');

set(handles.edit1,'Visible','on');
set(handles.text14,'Visible','on');

set(handles.edit2,'Visible','on');
set(handles.text5,'Visible','on');

set(handles.edit3,'Visible','on');
set(handles.text7,'Visible','on');

set(handles.edit4,'Visible','on');
set(handles.text8,'Visible','on');

set(handles.edit5,'Visible','on');
set(handles.text9,'Visible','on');

set(handles.edit6,'Visible','on');
set(handles.text11,'Visible','on');

set(handles.edit7,'Visible','on');
set(handles.text12,'Visible','on');

set(handles.edit8,'Visible','on');
set(handles.text13,'Visible','on');

set(handles.edit9,'Visible','off');
set(handles.text10,'Visible','off');

set(handles.edit10,'Visible','off');
set(handles.text15,'Visible','off');

set(handles.edit11,'Visible','off');
set(handles.text16,'Visible','off');

set(handles.edit12,'Visible','off');
set(handles.text17,'Visible','off');

set(handles.text2,'Visible','on');
set(handles.niuton_button,'Enable','on');
    case 9
set(handles.edit13,'Visible','on');
set(handles.text6,'Visible','on');

set(handles.edit1,'Visible','on');
set(handles.text14,'Visible','on');

set(handles.edit2,'Visible','on');
set(handles.text5,'Visible','on');

set(handles.edit3,'Visible','on');
set(handles.text7,'Visible','on');

set(handles.edit4,'Visible','on');
set(handles.text8,'Visible','on');

set(handles.edit5,'Visible','on');
set(handles.text9,'Visible','on');
```

```
set(handles.edit6,'Visible','on');
set(handles.text11,'Visible','on');

set(handles.edit7,'Visible','on');
set(handles.text12,'Visible','on');

set(handles.edit8,'Visible','on');
set(handles.text13,'Visible','on');

set(handles.edit9,'Visible','on');
set(handles.text10,'Visible','on');

set(handles.edit10,'Visible','off');
set(handles.text15,'Visible','off');

set(handles.edit11,'Visible','off');
set(handles.text16,'Visible','off');

set(handles.edit12,'Visible','off');
set(handles.text17,'Visible','off');

set(handles.text2,'Visible','on');
set(handles.niuton_button,'Enable','on');
    case 10
set(handles.edit13,'Visible','on');
set(handles.text6,'Visible','on');

set(handles.edit1,'Visible','on');
set(handles.text14,'Visible','on');

set(handles.edit2,'Visible','on');
set(handles.text5,'Visible','on');

set(handles.edit3,'Visible','on');
set(handles.text7,'Visible','on');

set(handles.edit4,'Visible','on');
set(handles.text8,'Visible','on');

set(handles.edit5,'Visible','on');
set(handles.text9,'Visible','on');

set(handles.edit6,'Visible','on');
set(handles.text11,'Visible','on');

set(handles.edit7,'Visible','on');
set(handles.text12,'Visible','on');

set(handles.edit8,'Visible','on');
set(handles.text13,'Visible','on');

set(handles.edit9,'Visible','on');
set(handles.text10,'Visible','on');

set(handles.edit10,'Visible','on');
set(handles.text15,'Visible','on');

set(handles.edit11,'Visible','off');
set(handles.text16,'Visible','off');

set(handles.edit12,'Visible','off');
set(handles.text17,'Visible','off');

set(handles.text2,'Visible','on');
```

```
set(handles.niuton_button,'Enable','on');
    case 11
set(handles.edit13,'Visible','on');
set(handles.text6,'Visible','on');

set(handles.edit1,'Visible','on');
set(handles.text14,'Visible','on');

set(handles.edit2,'Visible','on');
set(handles.text5,'Visible','on');

set(handles.edit3,'Visible','on');
set(handles.text7,'Visible','on');

set(handles.edit4,'Visible','on');
set(handles.text8,'Visible','on');

set(handles.edit5,'Visible','on');
set(handles.text9,'Visible','on');

set(handles.edit6,'Visible','on');
set(handles.text11,'Visible','on');

set(handles.edit7,'Visible','on');
set(handles.text12,'Visible','on');

set(handles.edit8,'Visible','on');
set(handles.text13,'Visible','on');

set(handles.edit9,'Visible','on');
set(handles.text10,'Visible','on');

set(handles.edit10,'Visible','on');
set(handles.text15,'Visible','on');

set(handles.edit11,'Visible','on');
set(handles.text16,'Visible','on');

set(handles.edit12,'Visible','off');
set(handles.text17,'Visible','off');

set(handles.text2,'Visible','on');
set(handles.niuton_button,'Enable','on');
    case 12
set(handles.edit13,'Visible','on');
set(handles.text6,'Visible','on');

set(handles.edit1,'Visible','on');
set(handles.text14,'Visible','on');

set(handles.edit2,'Visible','on');
set(handles.text5,'Visible','on');

set(handles.edit3,'Visible','on');
set(handles.text7,'Visible','on');

set(handles.edit4,'Visible','on');
set(handles.text8,'Visible','on');

set(handles.edit5,'Visible','on');
set(handles.text9,'Visible','on');

set(handles.edit6,'Visible','on');
set(handles.text11,'Visible','on');
```

```
set(handles.edit7,'Visible','on');
set(handles.text12,'Visible','on');

set(handles.edit8,'Visible','on');
set(handles.text13,'Visible','on');

set(handles.edit9,'Visible','on');
set(handles.text10,'Visible','on');

set(handles.edit10,'Visible','on');
set(handles.text15,'Visible','on');

set(handles.edit11,'Visible','on');
set(handles.text16,'Visible','on');

set(handles.edit12,'Visible','on');
set(handles.text17,'Visible','on');

set(handles.text2,'Visible','on');
set(handles.niuton_button,'Enable','on');
    otherwise
        errordlg('Neteisingas laipsnis','Klaida');
end

% --- Executes during object creation, after setting all properties.
function listbox2_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes on selection change in listbox2.
function listbox2_Callback(hObject, eventdata, handles)




% --- Executes on button press in niuton_button.
function niuton_button_Callback(hObject, eventdata, handles)
global S a laipsnis
% hObject    handle to niuton_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
laipsnis = str2num(get(handles.laipsnis_edit, 'String'));
epsilon = str2num(get(handles.epsilon_edit, 'String'));
spindulys = str2num(get(handles.spindulys_edit, 'String'));
iteracijos = str2num(get(handles.iteracijos_edit, 'String'));
a0 = str2num(get(handles.edit13, 'String'));
a1 = str2num(get(handles.edit1, 'String'));
a2 = str2num(get(handles.edit2, 'String'));
a3 = str2num(get(handles.edit3, 'String'));
a4 = str2num(get(handles.edit4, 'String'));
a5 = str2num(get(handles.edit5, 'String'));
a6 = str2num(get(handles.edit6, 'String'));
a7 = str2num(get(handles.edit7, 'String'));
a8 = str2num(get(handles.edit8, 'String'));
a9 = str2num(get(handles.edit9, 'String'));
a10 = str2num(get(handles.edit10, 'String'));
a11 = str2num(get(handles.edit11, 'String'));
a12 = str2num(get(handles.edit12, 'String'));
```

```
switch laipsnis
    case 1
        a(1) = a0;
        a(2) = a1;
    case 2
        a(1) = a0;
        a(2) = a1;
        a(3) = a2;
    case 3
        a(1) = a0;
        a(2) = a1;
        a(3) = a2;
        a(4) = a3;
    case 4
        a(1) = a0;
        a(2) = a1;
        a(3) = a2;
        a(4) = a3;
        a(5) = a4;
    case 5
        a(1) = a0;
        a(2) = a1;
        a(3) = a2;
        a(4) = a3;
        a(5) = a4;
        a(6) = a5;
    case 6
        a(1) = a0;
        a(2) = a1;
        a(3) = a2;
        a(4) = a3;
        a(5) = a4;
        a(6) = a5;
        a(7) = a6;
    case 7
        a(1) = a0;
        a(2) = a1;
        a(3) = a2;
        a(4) = a3;
        a(5) = a4;
        a(6) = a5;
        a(7) = a6;
        a(8) = a7;
    case 8
        a(1) = a0;
        a(2) = a1;
        a(3) = a2;
        a(4) = a3;
        a(5) = a4;
        a(6) = a5;
        a(7) = a6;
        a(8) = a7;
        a(9) = a8;
    case 9
        a(1) = a0;
        a(2) = a1;
        a(3) = a2;
        a(4) = a3;
        a(5) = a4;
        a(6) = a5;
        a(7) = a6;
        a(8) = a7;
        a(9) = a8;
        a(10) = a9;
```

```
    case 10
        a(1) = a0;
        a(2) = a1;
        a(3) = a2;
        a(4) = a3;
        a(5) = a4;
        a(6) = a5;
        a(7) = a6;
        a(8) = a7;
        a(9) = a8;
        a(10) = a9;
        a(11) = a10;
    case 11
        a(1) = a0;
        a(2) = a1;
        a(3) = a2;
        a(4) = a3;
        a(5) = a4;
        a(6) = a5;
        a(7) = a6;
        a(8) = a7;
        a(9) = a8;
        a(10) = a9;
        a(11) = a10;
        a(12) = a11;
    case 12
        a(1) = a0;
        a(2) = a1;
        a(3) = a2;
        a(4) = a3;
        a(5) = a4;
        a(6) = a5;
        a(7) = a6;
        a(8) = a7;
        a(9) = a8;
        a(10) = a9;
        a(11) = a10;
        a(12) = a11;
        a(13) = a12;
    otherwise
        errordlg('Neteisingas laipsnis','Klaida');
end
if (a(laipsnis+1)==0)
    errordlg('Neteisingas koeficientas prie kintamojo su auksciausiu
laipsniu','Klaida');
else
set(handles.niuton_button,'String',{['Palaukite...']});
Newton(iteracijos, epsilon, spindulys);
[kk,saknys] = size(S);
if (saknys==laipsnis)
for ii=1:laipsnis
    SS(ii) = S(1,ii)+S(2,ii)*i;
end;

switch laipsnis
    case 1
        set(handles.listbox2,'string',{['Saknys:'],num2str(SS(1))});
    case 2
        set(handles.listbox2,'string',{['Saknys:'],num2str(SS(1)),
num2str(SS(2))});
    case 3

set(handles.listbox2,'string',{['Saknys:'],num2str(SS(1)),num2str(SS(2)),num2str(S
S(3))});
```

```
    case 4

set(handles.listbox2,'string',{['Saknys:'],num2str(SS(1)),num2str(SS(2)),num2str(S
S(3)),num2str(SS(4))});
    case 5

set(handles.listbox2,'string',{['Saknys:'],num2str(SS(1)),num2str(SS(2)),num2str(S
S(3)),num2str(SS(4)),num2str(SS(5))});
    case 6

set(handles.listbox2,'string',{['Saknys:'],num2str(SS(1)),num2str(SS(2)),num2str(S
S(3)),num2str(SS(4)),num2str(SS(5)),num2str(SS(6))});
    case 7

set(handles.listbox2,'string',{['Saknys:'],num2str(SS(1)),num2str(SS(2)),num2str(S
S(3)),num2str(SS(4)),num2str(SS(5)),num2str(SS(6)),num2str(SS(7))});
    case 8

set(handles.listbox2,'string',{['Saknys:'],num2str(SS(1)),num2str(SS(2)),num2str(S
S(3)),num2str(SS(4)),num2str(SS(5)),num2str(SS(6)),num2str(SS(7)),num2str(SS(8))})
;
    case 9

set(handles.listbox2,'string',{['Saknys:'],num2str(SS(1)),num2str(SS(2)),num2str(S
S(3)),num2str(SS(4)),num2str(SS(5)),num2str(SS(6)),num2str(SS(7)),num2str(SS(8)),n
um2str(SS(9))});
    case 10

set(handles.listbox2,'string',{['Saknys:'],num2str(SS(1)),num2str(SS(2)),num2str(S
S(3)),num2str(SS(4)),num2str(SS(5)),num2str(SS(6)),num2str(SS(7)),num2str(SS(8)),n
um2str(SS(9)),num2str(SS(10))});
    case 11

set(handles.listbox2,'string',{['Saknys:'],num2str(SS(1)),num2str(SS(2)),num2str(S
S(3)),num2str(SS(4)),num2str(SS(5)),num2str(SS(6)),num2str(SS(7)),num2str(SS(8)),n
um2str(SS(9)),num2str(SS(10)),num2str(SS(11))});
    case 12

set(handles.listbox2,'string',{['Saknys:'],num2str(SS(1)),num2str(SS(2)),num2str(S
S(3)),num2str(SS(4)),num2str(SS(5)),num2str(SS(6)),num2str(SS(7)),num2str(SS(8)),n
um2str(SS(9)),num2str(SS(10)),num2str(SS(11)),num2str(SS(12))});
    otherwise
        errordlg('Neteisingas laipsnis','Klaida');
end;
set(handles.algoritmas_button,'Enable','on');
else
    errordlg('Saknys nerastos, keiskite parametrus','Klaida');
end;
end;
set(handles.niuton_button,'String',{['Skaičiuoti daugianario šaknis']});
set(handles.niuton_button,'Enable','off');

% --- Executes on button press in pagalba_button.
function pagalba_button_Callback(hObject, eventdata, handles)
% hObject    handle to pagalba_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
info(0);
```

failas *PL.m*

```
unction PL(iteracijos, zing, r)
global S laipsnis sritis a;
```

```
%Simbolinis funkcijos ir Niutono transformacijos radimas
syms z;
n = laipsnis;
%Funkcijos israiska
j = a(1);
for h = 2:n+1
    j = j + a(h)*z^(h-1);
end;
% disp ('F(z) = ');
% disp(j);
%Niutono transformacija
N = z - (j/diff(j,'z'));
%-----------------------------------------------------------------
%Grafiko antrastes
ax = sritis;
xlabel('realioji dalis');
ylabel('menamoji dalis');
axis ([-ax ax -ax ax]);
hold on;
%-----------------------------------------------------------------
%********************Pabegimo laiko algoritmas**********************
%-----------------------------------------------------------------

for x = -sritis:zing:sritis
   for y = -sritis:zing:sritis
        z = x + y*i;
        iter = 0;      %iteraciju skaicius
    while ((iter<=iteracijos)&(eval(diff(j,'z'))~=0))
          iter = iter + 1;
          z_g = eval(N); %Niutono transformacija
          for d=1:laipsnis
              if (((real(z_g)-S(1,d))^2 + (imag(z_g)-S(2,d))^2)<=r^2)
                  spalva = colors(d);

plot(x,y,'o','MarkerEdgeColor',spalva,'MarkerFaceColor',spalva,'MarkerSize',2);
                  hold on;
                  break;
               end;
          end;
          z = z_g;
    end;
  end;
end;

%-----------------------------------------------------------------
%*****************************************************************
%-----------------------------------------------------------------
 %saknys atidedamos grafike
 for d=1:laipsnis
     spalva = colors(d);

plot(S(1,d),S(2,d),'s','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor',spalv
a,'MarkerSize',5);
     hold on;
end
```

failas *Newton.m*

```
function niutonas(iteracijos, zingsnis, epsilon)
format long;
global u v matrix S a laipsnis sritis; %paskelbiami globalus kintamieji
%--------------------Srities nustatymas-----------------------------
syms r x y real;
z = x + i*y;
```

```
n = laipsnis;
%Nelygybes kairioji puse
nelyg1 = abs((r)^n);
%Nelygybes desinioji puse
nelyg2 = abs(a(1)/a(n+1));
for h = 1:(n-1)
    nelyg2 = nelyg2 + abs(a(h+1)/a(n+1)*(r)^h);
end;
%Srities radimas
r = 1;
dr = 1;
if (n==1)
    while ((eval(nelyg1)) < (abs(nelyg2)))
    r = r+dr;
end;
else
while ((eval(nelyg1)) < (eval(nelyg2)))
    r = r+dr;
end;
end;
sritis = r;
%-------------------------------------------------------------------------
%Simbolinis funkcijos realios ir menamos daliu, bei jakobiano matricos
%radimas
syms x y real;
w = x + i*y;
z= sritis * w;
%Funkcijos israiska
j = a(1);
for h = 1:n
    j = j + a(h+1)*z^h;
end;
%Funkcijos menamoji ir realioji dalys
u = real(j);
v = imag(j);
dudx = diff(u,'x');
dudy = diff(u,'y');
dvdx = diff(v,'x');
dvdy = diff(v,'y');
%Jakobiano matrica
matrix = [dudx dudy; dvdx dvdy];
%-------------------------------------------------------------------------
%********************Lygties saknu radimas*************************
%-------------------------------------------------------------------------
delta = 90*zingsnis/pi;
k = 1; %saknu skaicius
for fi = 0:delta:360
    x = 2*cos(fi);
    y = 2*sin(fi);
       x_pr = [x; y]; %pradinis taskas
         iter = 0;      %iteraciju skaicius
         while ((det(Jacob(x_pr))~=0)&(iter<=iteracijos))
           x_g = N(x_pr); %Niutono iteracine funkcija
           x_pr = x_g;
          iter = iter + 1;
         end;
         taip = 0;
         if (k<=n) %einama per masyva, sauganti saknis
           for w=1:k-1
             if ((abs((S(1,w)-x_pr(1)))<=epsilon)&&((abs(S(2,w)-
x_pr(2)))<=epsilon))
               taip = 1; %jei tokia saknis jau irasyta (jei r spindulio skritulyje
nuo kazkurios saknies yra - tikslinam)
                 if ((f(x_pr)-0)<(f([S(1,w) S(2,w)])-0))
```

```
                S(1,w) = x_pr(1);
                S(2,w) = x_pr(2);
               end;
            end;
         end;
       end;

%jei tokios saknies dar nebuvo ir jei tai tikrai saknis
   if ((taip == 0)&&(k<=n)&&(patikrinimas(x_pr, epsilon)))
       %saknis irasoma i masyva
      S(1,k) = x_pr(1);
      S(2,k) = x_pr(2);
      k = k+1;
    end;
end;
%---------------------------------------------------------------------
%*********************************************************************
%---------------------------------------------------------------------
[kk,saknys] = size(S);
if ((kk~=0)||(saknys~=0))
    S = sritis * S;
end;
```

failas *colors.m*

```
function spalva = colors(d)
spalvos =[0 0 1;
          1 0 0;
          0 1 0;
          1 1 0;
          0 1 1;
          1 0 1;
          0 0 0.5;
          0.5 0 0;
          0 0.5 0;
          0.5 0.5 0;
          0 0.5 0.5;
          0.5 0 0.5
          0 0.5 1
          0 1 0.5
          0.5 0 1
          0.5 0.5 1
          0.5 1 0
          0.5 1 0.5
          0.5 1 1
          1 0 0.5
          1 0.5 0
          1 0.5 0.5
          1 0.5 1
          1 1 0.5];
spalva = spalvos(d,:);
```

failas *f.m*

```
function g = f(r)
global u v;
x = r(1);
y = r(2);
f_r = eval(u);
f_m = eval(v);
g = [f_r; f_m];
```

failas *Jacob.m*

```
function mat = Jacob(r)
global matrix;
```

```
x = r(1);
y = r(2);
mat = eval(matrix);
```

failas *N.m*

```
function  y = N(x);
%Niutono iteracine funkcija
   j = [x]- inv(Jacob(x))*f(x);
   y=[real(j(1)); real(j(2))];
```

failas *patikrinimas.m*

```
function g = patikrinimas(x,r)
%patikrinama, ar r tikslumu taskas yra saknis
j = f(x);
if ((abs(j(1)-0)<=r)&&(abs(j(2)-0)<=r))
    g = 1;
else
    g = 0;
end;
```