KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

TAIKOMOSIOS MATEMATIKOS KATEDRA

Raimondas Katinas

DAUGIAMAČIŲ DUOMENŲ APROKSIMAVIMAS

Magistro darbas

Vadovas

doc.dr. N.Listopadskis

KAUNAS, 2008

KAUNO TECHNOLOGIJOS UNIVERSITETAS

FUNDAMENTALIŲJŲ MOKSLŲ FAKULTETAS

TAIKOMOSIOS MATEMATIKOS KATEDRA

TVIRTINU
Katedros vedėjas
doc. dr. N. Listopadskis
2008 06 09

DAUGIAMAČIŲ DUOMENŲ APROKSIMAVIMAS

Taikomosios matematikos magistro baigiamasis darbas

Vadovas
( ) doc. dr. N. Listopadskis
2008 06 09

Recenzentas
( ) doc.dr. D. Rubliauskas
2008 06 09

Atliko
FMMM-6 gr. stud.
( ) R. Katinas
2008 06 09

KAUNAS, 2008

KVALIFIKACINĖ KOMISIJA

Pirmininkas:	Leonas Saulis, profesorius (VGTU)
Sekretorius:	Eimutis Valakevičius, docentas (KTU)
Nariai:	Algimantas Jonas Aksomaitis, profesorius (KTU)
	Vytautas Janilionis, docentas (KTU)
	Vidmantas Povilas Pekarskas, profesorius (KTU)
	Rimantas Rudzkis, habil. dr., valdybos pirmininko pavaduotojas (DnB NORD Bankas)
	Zenonas Navickas, profesorius (KTU)
	Arūnas Barauskas, dr., vice-prezidentas projektams (UAB „Baltic Amadeus“)

Katinas R. Approximation of multi-dimensional data: Master‘s work in applied mathematics / supervisor assoc. prof. dr. N. Listopadskis; Department of Applied mathematics, Faculty of Fundamental Sciences, Kaunas University of Technology. – Kaunas, 2008. – 82 p.

SUMMARY

This Master‘s work covers a mathematical analysis system which can visualize multivariate data layers, approximate multi-dimensional functions by polynomials, estimate approximation accuracy and present few the most effective aproximation models. Multivariate approximation theory is an increasingly active research area today. It encompasses a wide range of tools for multivariate approximation such as multi-dimensional splines and finite elements, shift-invariant spaces and radial-basis functions. Approximation theory in the multivariate setting has many applications including numerical analysis, wavelet analysis, signal processing, geographic information systems, computer aided geometric design and computer graphics. The field is fascinating since much of the mathematics of the classical univariate theory does not straightforwardly generalize to the multivariate setting, so new tools are required. Graphs of one variable functions are frequantly displayed as curves, bivariate functions - as contour plots. In generally it is very hard to display or realize function in the multivariate setting. However, some efforts have been made to render functions of precisely three variables. Two obvious approaches suggest themselves: 1. Display a number of cross sections where one of the variables is held constant, or, 2. display contour surfaces where the value of function equals some constant. We will use the first method modification in this Master‘s work. All function variables except one will be taken as constants. Changing the constants values we will be given certain data layers. Other approximation method is based on multivariate plynomial interpolation. Instead of interpolant we will get smooth function while decreasing each variable maximum degree. Approximation accuracy will be estimated using some standard errors characteristics and three quality coefficients. Also it is possible to compare suitability of two polynomial models using F-test. The mathematical analysis system covered in this paper realizes previously mentioned approximation methods, also allows fast data layers visualization and these layers approximation. This paper includes technical documentation of the system and theory of mathematical part. Also, there is an example demonstrating work and capabilities of the system.

This work has been reported at the three conferences (7th Student‘s Conference, 2008; Mathematics and Mathematical Modeling, 2007 & 2008. In additions, two articles have been published based on this topic [31], [32] and one more article is going to be presented at the 49th Conference of Lithuanian Mathematician Association.

TURINYS

PAVEIKSLŲ SĄRAŠAS

ĮVADAS

Šiais laikais vis daugiau domimasi daugiamačių duomenų aproksimavimo teorija. Daugiamatėje erdvėje aproksimavimo teorija palčiai kur taikoma, pavyzdžiui, skaitinių metodų analizėje, bangų analizėje, signalų apdorojime, įvairiose informacinių technologijų sistemose, kompiuterių grafikoje, astronomijoje, naftos klodų tyrinėjime. Ši sritis viliojanti, nes didelė dalis klasikinės matematikos sunkiai pritaikoma daugiamačiams uždaviniams analizuoti. Taigi senoms problemoms spręsti reikalingi nauji įrankiai.

Funkcijų aproksimavimo uždavinių gausu įvairiose matematikos, fizikos ir technikos srityse. Gausu ir jų sprendimų būdų bei metodų. Nesunkiai šie uždaviniai sprendžiami, kai funkcija priklauso nuo vieno ar dviejų kintamųjų. Tačiau realiame gyvenime naudojamos funkcijos turi daug daugiau nežinomųjų. Didėjant kintamųjų skaičiui uždavinio sudėtingumas taip pat auga. Pavyzdžiui, kai funkcija priklauso nuo vieno kintamojo, ją galima pavaizduoti plokštumoje kaip kreivę. Dviejų kintamųjų funkciją atitinka paviršius, nubrėžtas trimatėje erdvėje. Funkcijų, kurios priklauso nuo trijų ir daugiau kintamųjų, vaizdavimas jau sukelia problemų, nes žmogus nebegali suvokti didesnio matumo erdvės. Kadangi trimatę erdvę galima pavaizduoti plokštumoje, manoma, kad panašiu principu keturmatę erdvę galima pavaizduoti trimatėje, o šią vėl plokštumoje. Jei pavyktų sugalvoti tokį metodą, erdvės matumas nebesukeltų problemų. Visgi trijų kintamųjų funkciją bandoma vaizduoti dviem būdais: 1. pateikti paviršių aibę, kai aproksimuojamos funkcijos reikšmė lygi tam tikrai konstantai; 2. pateikti duomenų pjūvių aibę, kai vienas iš kintamųjų yra laikomas konstanta. Tokių būdu, galima susidaryti vaizdą, kaip funkcija priklauso nuo vieno ar kito kintamojo. Šiame darbe kaip tik ir taikysime antrą funkcijų vaizdavimo metodą – jo pagalba konstruosime aproksimuojantį daugianarį. Kitas analizuojamas aproksimavimo metodas šiame darbe yra pagrįstas daugiamačiu polinominiu interpoliavimu. Daugiamačio polinominio interpoliavimo metodo esmė yra rasti daugianarį, kuris eitų per visus duomenų tinklelio taškus, t.y. juos interpoliuotų. Tačiau toks daugianaris paprastai būna labai sudėtingas. Jį suprastinti galima mažinant kintamųjų maksimalius laipsnius, arba pašalinant kai kuriuos jo narius. Tokiu būdu, nedaug pabloginus aproksimacijos tikslumą, galima daug paprasčiau gauti rezultatą. Darbe naudojami trys aproksimacijos kokybės koeficientai, kurie leidžia įvertinti aproksimacijos tikslumą bei atlikti aproksimacijos metodų lyginamąją analizę.

1. BENDROJI DALIS

1.1. FUNKCIJŲ APROKSIMAVIMAS

Literatūroje nagrinėjama daugybė funkcijų aproksimavimo uždavinio sprendimo metodų. Tai paaiškinama tuo, kad praktikoje susiduriama su daugeliu skirtingų šio uždavinio formuluočių.

Apibendrintai funkcijų aproksimavimo uždavinį galima suformuluoti taip: turint funkcijos $f = f(x, y, \dots z)$ reikšmių lentelę $(x_i, y_i, \dots, z_i, f_i)$ (čia $i = \overline{0,m}$, $f_i = f(x_i, y_i, \dots z_i)$ ir $(x_i, y_i, \dots, z_i) \neq (x_j, y_j, \dots, z_j)$, kai $i \neq j$), reikia rasti funkcijos $f(x, y, \dots, z)$ reikšmę, kai $(x, y, \dots, z) = (s, t, \dots, v)$, o $(s, t, \dots, v) \in [x_0, y_0, \dots, z_0; x_m, y_m, \dots, z_m]$.

Funkcijos $f(x, y, \dots z)$ analizinė išraiška paprastai yra nežinoma arba pernelyg sudėtinga.

Atsižvelgiant į f_i reikšmių tikslumą, taikomi du šio uždavinio sprendimo metodai:

- interpoliavimo metodas,
- suglodinimo metodas.

1.1.1. INTERPOLIAVIMAS

Interpoliavimo uždavinio formulavimas. Duota funkcijos $f(x, y, \dots z)$ reikšmių lentelė $(x_i, y_i, \dots, z_i, f_i)$, čia $i = \overline{0,m}$, f_i reikšmės yra tikslios arba paklaidos tokios mažos, kad praktiškai jų galima nepaisyti. Reikia rasti aproksimuojančiąją funkciją $F(x, y, \dots z)$, priklausančią funkcijų klasei K ir tenkinančią sąlygas

$$F(x_i, y_i, \dots z_i) = f_i,\ i = \overline{0,m}. \tag{1}$$

Šios sąlygos vadinamos Lagranžo interpoliavimo sąlygomis, o pati funkcija $F(x, y, \dots z)$ — Lagranžo interpoliacine funkcija, arba tiesiog interpoliacine funkcija.

Jei, be funkcijos $f(x, y, \dots z)$ reikšmių, taške $(x_i, y_i, \dots z_i)$ yra žinomos funkcijos $f(x, y, \dots z)$ mišrių išvestinių iki $(m_i - 1)$-osios eilės imtinai reikšmės, tai galima reikalauti, kad aproksimuojančioji funkcija $F(x, y, \dots z)$ tenkintų sąlygas

$$\frac{\partial^{(l)} F(x_i, y_i, \dots, z_i)}{\partial x^{(p)} \partial y^{(t)} \dots \partial z^{(k)}} = \frac{\partial^{(l)} f(x_i, y_i, \dots, z_i)}{\partial x^{(p)} \partial y^{(t)} \dots \partial z^{(k)}},\ i = \overline{0,n},\ p + t + \dots + k = l,\ l = \overline{0, m_i - 1},.$$

Simbolis (l) žymi *l*-tosios eilės išvestinę. Šios sąlygos vadinamos Ermito interpoliavimo sąlygomis, o funkcija $F(x, y, \dots z)$ — Ermito interpoliacine funkcija.

Kaip matyti iš interpoliavimo uždavinio formuluotės, keičiant aproksimuojančiųjų funkcijų klasę *K* bei interpoliavimo sąlygas, galima rasti įvairias interpoliacines funkcijas. Išsamesnė informacija šaltinyje [2].

Istoriškai pirmoji aproksimuojančiųjų funkcijų klasė buvo *n*-tosios eilės daugianarių klasė. Pastaruoju metu plačiai naudojama splainų klasė.

1.1.1.1. INTERPOLIACINŲ DAUGIANARIŲ KONSTRAVIMAS IR JŲ REIKŠMIŲ APSKAIČIAVIMAS

Uždavinio formulavimas. Duota funkcijos $f(x, y, \dots z)$ reikšmių lentelė $(x_i, y_i, \dots, z_i, f_i)$, čia $i = \overline{0, m}$. Reikia rasti tokį daugianarį $F(x, y, \dots z)$, kuris tenkintų Lagranžo interpoliavimo sąlygas.

Pirmasis klausimas, į kurį reikia atsakyti, yra toks: ar interpoliacinio daugianario konstravimo ir jo reikšmės apskaičiavimo uždavinys turi sprendinį ir ar šis sprendinys yra vienintelis?

Tarkime, kad taškai $(x_i, y_i, \dots z_i)$ $\left(i = \overline{0, m}\right)$ yra skirtingi, t. y. $(x_i, y_i, \dots, z_i) \neq (x_j, y_j, \dots, z_j)$, kai $i \neq j$. Tada interpoliacinis daugianaris egzistuoja ir yra vienintelis.

Sakykime,

$$F(x, y, \dots, z) = a_{00\dots0} + a_{10\dots0}x + \dots + a_{n0\dots0}x^n + a_{01\dots0}y + \dots + a_{0u\dots0}y^u + a_{11\dots a}xy + \dots + a_{nu\dots t}x^n y^u \dots z^t$$

yra ieškomasis interpoliacinis daugianaris. Jis turi tenkinti (1) sąlygas, t. y.

$$a_{00\dots0} + a_{10\dots0}x_i + \dots + a_{n0\dots0}x_i^n + a_{01\dots0}y_i + \dots + a_{0u\dots0}y_i^u + a_{11\dots a}x_i y_i + \dots + a_{nu\dots t}x_i^n y_i^u \dots z_i^t = f_i,\ i = \overline{0, m}$$

bei koeficientų *a* skaičius turi būti lygus lygiai *m*, o kintamųjų maksimalūs laipsniai vienetu mažesni, negu jų įgyjamų skirtingų reikšmių skaičiai.

Iš šios tiesinių lygčių sistemos apskaičiuojami interpoliacinio daugianario koeficientai. Kadangi sistemos determinantas yra Vandermondo determinantas, kuris nelygus nuliui, kai $(x_i, y_i, \dots, z_i) \neq (x_j, y_j, \dots, z_j)$ $(i \neq j)$, tai sistemos sprendinys egzistuoja ir yra vienintelis.

Yra gana daug interpoliacinio daugianario konstravimo metodų. Čia pateiksime keletą iš jų.

1.1.1.1.1. TIESINĖ INTERPOLIACIJA

Vienas iš paprasčiausių metodų yra tiesinė interpoliacija. Pradėkime nuo vienmatės funkcijos. Paprastai šiuo atveju naudojami du gretimi reikšmių lentelės taškai (x_i, y_i) ir (x_{i+1}, y_{i+1}), tuomet interpoliacinis daugianaris tarp šių taškų turės išraišką

$$f(x) = f_1 + \frac{(x - x_1)(f_2 - f_1)}{(x_2 - x_1)},$$

čia $f_i = f(x_i)$, $i = \overline{0,m}$. Tiesinė interpoliacija yra greita ir paprasta, tačiau nelabai efektyvi. Šiuo atveju interpoliuojanti kreivė yra laužtė, gauta gretimus reikšmių lentelės taškus sujungus tiesių atkarpomis [9]. Trūkumai yra tie, kad būtent dėl to šiuose taškuose kreivė nėra diferencijuojama.

Dvimatės funkcijos tiesinis interpoliacinis daugianaris turės daug sudėtingesnę išraišką [10], [13]

$$f(x,y) = \frac{f_{11}}{(x_2 - x_1)(y_2 - y_1)}(x_2 - x)(y_2 - y) + \frac{f_{21}}{(x_2 - x_1)(y_2 - y_1)}(x - x_1)(y_2 - y) +$$

$$+ \frac{f_{12}}{(x_2 - x_1)(y_2 - y_1)}(x_2 - x)(y - y_1) + \frac{f_{22}}{(x_2 - x_1)(y_2 - y_1)}(x - x_1)(y - y_1),$$

čia $f_{ij} = f(x_i, y_j)$ ir $x_1 < x < x_2$, $y_1 < y < y_2$. Šiam užrašui galima suteikti paprastesnį pavidalą [14].

Įveskime pažymėjimus: $q_1 = \frac{(x - x_1)}{(x_2 - x_1)}$, $q_2 = \frac{(y - y_1)}{(y_2 - y_1)}$, $p_1 = 1 - q_1$, $p_2 = 1 - q_2$ ir tada

$$f(x,y) = p_1 p_2 f_{11} + q_1 p_2 f_{21} + p_1 q_2 f_{12} + q_1 q_2 f_{22}.$$

Tokį pat pavidalą galima suteikti ir vieno kintamojo funkcijai:

$$f(x) = p_1 f_1 + q_1 f_2.$$

Šiuo principu galime sukonstruoti daugianarį ir trimatei funkcijai:

$$f(x,y,z) = p_1 p_2 p_3 f_{111} + q_1 p_2 p_3 f_{211} + p_1 q_2 p_3 f_{121} + q_1 q_2 p_3 f_{221} +$$

$$+ p_1 p_2 q_3 f_{112} + q_1 p_2 q_3 f_{212} + p_1 q_2 q_3 f_{122} + q_1 q_2 q_3 f_{222},$$

kai $q_3 = \frac{(z - z_1)}{(z_2 - z_1)}$, $p_3 = 1 - q_3$, $z_1 < z < z_2$ ir $f_{ijk} = f(x_i, y_j, z_k)$. Matome, kaip keičiantis dimensijai auga funkcijos išraiškoje sumuojamų narių skaičius. Bendru atveju *n*-matėje erdvėje sumuosime 2^n narius. Todėl jau 5-matėje erdvėje reiktų sumuoti 32 narius, viename duomenų tinklelio intervale [14].

1.1.1.1.2. POLINOMINĖ INTERPOLIACIJA

Polinominė interpoliacija yra tiesinės interpoliacijos apibendrinimas. Anksčiau duomenų tinklelio taškus jungėme tiese, t.y. pirmo laipsnio daugianariu. Dabar interpoliacijai naudosime aukštesnio laipsnio daugianarį. Jei paimtumėme *m* taškų, tai vienmatės funkcijos atveju tiktai vienas *m*-1-ojo laipsnio daugianaris (arba daugianaris turintis lygiai *m* koeficientų) eis tiksliai per tuos taškus. Daugiamačiu atveju keliamas tas pats reikalavimas: interpoliacino daugianario maksimalūs kintamųjų laipsniai turi būti vienetu mažesni negu kintamųjų įgyjamų skirtingų reikšmių skaičius. Interpoliacinio *n*-tojo laipsnio daugianario bendra išraiška vienmačiu atveju yra tokia

$$f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

arba

$$f(x) = \sum_{i=0}^{N_x} a_i x^i ,$$

čia a_i - tam tikri koeficientai. Daugiamačiu atveju formulė atrodys taip

$$f(x, y, \ldots, z) = \sum_{i=0}^{N_x} \sum_{j=0}^{N_y} \ldots \sum_{k=0}^{N_k} a_{ij\ldots k} x^i y^j \ldots z^k .$$

Tarkime, kad kintamųjų skirtingų įgyjamų reikšmių skaičiai atitinkamai lygūs $k_x, k_y, \ldots, k_z$. Tada tokio daugianario maksimalūs laipsniai turi tenkinti sąlygą

$$N_x - 1 = k_x \cap N_y - 1 = k_y \cap \ldots \cap N_z - 1 = k_z .$$

Kaip ir vienmačiu atveju, tik vienas toks daugiamatis daugianaris eis tiksliai per visus taškus [3], [4], [12].

1.1.1.1.3. INTERPOLIACIJA SPLAINAIS

Tiesinės interpoliacijos atveju, du duomenų tinklelio taškai sujungiami tiesė, šiuo atveju kiekviename intervale naudojami žemo laipsnio daugianariai. Jų visuma vadinama splainu. Sudaryta kreivė yra daug glodesnė negu tiesinės interpoliacijos atveju ir paprasčiau apskaičiuojama negu interpoliacinis daugianaris.

Vienmačiu atveju interpoliacino splaino apibrėžimas skamba taip: duotas $m+1$ duomenų tinklelio taškas x_i toks, kad

$$x_0 < x_1 < \ldots < x_{m-1} < x_m$$

taip pat žinoma $m+1$ interpoliuojmos funkcijos reikšmė f_i minėtuose taškuose. Reikia rasti tokią laipsnio *m* splaino funkciją

$$S(x) = \begin{cases} S_0(x) & x \in [x_0, x_1] \\ S_1(x) & x \in [x_1, x_2] \\ \vdots & \vdots \\ S_{m-1}(x) & x \in [x_{m-1}, x_m] \end{cases},$$

kur kiekviena funkcija $S_i(x)$ yra laipsnio *k* daugianaris. Pavyzdžiui kvadratinio splaino išraiška gali būti tokia

$$S_i(x) = f_i + z_i(x - x_i) + \frac{z_{i+1} - z_i}{2(x_{i+1} - x_i)}(x - x_i)^2 .$$

Koeficientai gali būti randami laisvai pasirenkant z_0 bei naudojant rekuriančią formulę

$$z_{i+1} = -z_i + 2\frac{f_{i+1} - f_i}{x_{i+1} - x_i}.$$

Polinominės interpoliacijos atveju, *m*-tojo laipsnio daugianaris yra vienareikšmiškai apibrėžiamas duomenų tinklelio taškų. Tuo tarpu *m*-tojo laipsnio splainas, interpoliuojantis tuos pačius taškus nėra vienareikšmiškai nusakomas [5], [16].

1.1.1.1.4. IDW INTERPOLIACIJOS METODAS

IDW (Inverse Distance Weighting) yra daugiamatis interpoliavimo metodas. Tai toks procesas, kurio metu apskaičiuojama nežinoma funkcijos reikšmė, naudojant visus duomenų tinklelio taškus. Bendru atveju, turint tašką *x*, funkcijos reikšmė f IDW metodu apskaičiuojama pagal formulę

$$f(x) = \frac{\sum_{k=0}^{m} w_k(x) f_k}{\sum_{k=0}^{m} w_k(x)},$$

kur

$$w_k(x) = \frac{1}{d(x, x_k)^p},$$

paprasta IDW svorio funkcija, *x* – taškas, kuriame norime apskaičiuoti funkcijos reikšmę *f*, x_k ir f_k atitinkamai duomenų tinklelio taškas ir funkcijos reikšmė jame, *d* – metrika, kitaip tariant atstumas tarp taškų *x* ir x_k, *m* yra duomenų tinklelio taškų skaičius, *p* – realus teigiamas skaičius, laipsnio kriterijus. Šiuo atveju svoris mažėja, didėjant atstumui tarp taškų. Kuo didesnis *p*, tuo didesne stebimų taškų įtaka interpoliuojamam taškui. Kai $0 < p < 1$, tai $f(x)$ turi smailias viršūnes, o kai *p* yra didesnis už vienetą, tai viršūnės tampa glodesnės. Dažniausiai pasirenkama *p* reikšmė yra 2. Metodą lengvai galima pritaikyti aukštesnio matumo erdvėms, tiesiog tada *x* reikia laikyti vektoriumi. Iš tikrųjų, tai yra Lagranžo aproksimacijos apibendrinimas daugiamatėms erdvėms [17].

1.1.1.1.5. KRIGINGO INTERPOLIACIJA

Kriging‘o (Kriging) metodas priskiriamas tiesinių mažiausių kvadratų metodų šeimai. Metodo tikslas yra rasti nežinomos realios funkcijos f reikšmę kažkokiame tarpiniame taške x^*, kai žinomos funkcijos reikšmės kituose taškuose $x_1, x_2, \ldots, x_n$. Kriging‘o metodas vadinamas tiesiniu, nes prognozuojama $\overline{f}(x^*)$ reikšmė yra tiesinė kombinacija, kuri gali būti užrašyta kaip

$\overline{f}(x^*) = \sum_{i=1}^{n} \lambda_i f(x_i)$. Svoriai λ_i randami iš tiesinių lygčių sistemos, kuri sudaroma remiantis prielaida, kad f yra atsitiktinio proceso $F(x)$ reikšmės ir kad prognozės paklaida $\varepsilon(x) = F(x) - \sum_{i=1}^{n} \lambda_i F(x_i)$ tam tikru būdu bus minimizuota. Pavyzdžiui, taip vadinama paprastoji Kriging'o prielaida yra tokia, kai tariama, kad žinomas $F(x)$ vidurkis ir dispersija, tuomet Kriging'o prognozė bus kaip tik tokia, kuri minimizuoja prognozės paklaidos dispersiją [18].

1.1.1.2. FUNKCIJOS REIKŠMĖS APSKAIČIAVIMO UŽDAVINYS

Praktikoje dažniausiai tenka ieškoti funkcijos reikšmės norimu tikslumu.

Duota: funkcijos $f(x, y, \ldots z)$ reikšmių lentelė $(x_i, y_i, \ldots, z_i, f_i)$, čia $i = \overline{0,m}$.;

$(s,t,\ldots,v)$ — argumento reikšmė; $(s,t,\ldots,v) \in [x_0, y_0, \ldots, z_0; x_m, y_m, \ldots, z_m]$.

Reikia rasti: $f(s,t,\ldots,v)$ tikslumu ε.

Sprendžiant šį uždavinį, reikia išsiaiškinti:

- kurios eilės interpoliacinį dugianarį naudoti;
- kuriuos interpoliavimo taškus pasirinkti.

Žinoma, kad liekamojo nario reikšmė yra mažiausia, kai argumentas $(s,t,\ldots,v)$ artimas hyperintervalo $[x_0, y_0, \ldots, z_0; x_m, y_m, \ldots, z_m]$ vidurio taškui.

Remdamiesi šiomis žiniomis, siūlome uždavinio sprendimo metodą, kuris leidžia apskaičiuoti funkcijos reikšmę $f(s,t,\ldots,v)$ tikslumu ε, jeigu tai įmanoma, arba tiksliausią galimą $f(s,t,\ldots,v)$ reikšmę ir jos paklaidą.

Simboliu $L_k(s,t,\ldots,v)$ pažymėkime *k*-tosios eilės interpoliacinio daugianario, einančio per lentelės $k+1$ tašką, reikšmę, o simboliu min — $L_k(s,t,\ldots,v)$ liekamojo nario reikšmę.

$f(s,t,\ldots,v)$ reikšmės skaičiavimą baigiame, kai galioja bent viena iš šių sąlygų:

1) min reikšmė lygi ε arba mažesnė už ε;

2) didinant interpoliacinio daugianario eilę, liekamojo nario įvertis pradeda didėti;

3) interpoliacinio daugianario eilė lygi $m-1$.

1.1.2. SUGLODINIMAS

Nagrinėdami interpoliavimo uždavinį, funkciją $f(x, y, \dots z)$, apibrėžtą reikšmių lentele, stengėmės pakeisti tokia aproksimuojančia funkcija $F(x, y, \dots z)$ (*n*-tos eilės daugianariu, splainu ir pan.), kad taškuose $(x_i, y_i, \dots z_i)$ $(i = \overline{0, m})$ funkcijos $F(x_i, y_i, \dots z_i)$ reikšmės būtų lygios $f(x_i, y_i, \dots z_i)$ reikšmėms.

Labai dažnai $f(x_i, y_i, \dots z_i)$ reikšmės yra eksperimento rezultatai ir turi matavimo bei metodo paklaidų. Todėl reikalauti, kad aproksimuojančioji funkcija $F(x, y, \dots z)$ tenkintų sąlygą $F(x_i, y_i, \dots z_i) = f(x_i, y_i, \dots z_i)$ $(i = \overline{0, m})$, būtų neprotinga. Geriau rasti tokią funkciją $F(x, y, \dots z)$, kuri pagal pasirinktą kriterijų geriausiai aproksimuotų $f(x, y, \dots z)$. Toks funkcijos $F(x, y, \dots z)$ apskaičiavimo metodas vadinamas suglodinimu. Atsižvelgiant į suglodinimo kriterijų, galima gauti įvairias $F(x, y, \dots z)$ išraiškas.

Suglodinimo uždavinio formulavimas. Sakykime, funkcija $f = f(x, y, \dots z)$ nusakyta reikšmių lentele $(x_i, y_i, \dots, z_i, \tilde{f}_i)$; čia $(i = \overline{0, m})$. Simbolis $\tilde{f}_i$ rodo, kad $f(x, y, \dots z)$ reikšmės taškuose $(x_i, y_i, \dots z_i)$ yra apytikslės. Taip pat žinoma aproksimuojančios funkcijos $F(x, y, \dots z)$ analizinė išraiška: $F(x, y, \dots z, a_0, \dots, a_n)$; čia a_i $(i = \overline{0, n})$ - nežinomi parametrai ir $n << m$. Reikia rasti tokias parametrų a_i reikšmes, su kuriomis $F(x, y, \dots z)$ geriausiai aproksimuotų funkciją $f(x, y, \dots z)$.

Sprendžiant šį uždavinį, taikomi įvairūs suglodinimo kriterijai, o kartu ir įvairūs suglodinimo metodai.

Pasirinktų taškų metodas. Taikant šį metodą, iš lentelės $(x_i, y_i, \dots, z_i, \tilde{f}_i)$ $(i = \overline{0, m})$ pasirenkamas $n+1$ taškas $(x_{i_k}, y_{i_k}, \dots, z_{i_k}, \tilde{f}_{i_k})$ $(k = \overline{0, n})$, kurių $\tilde{f}_{i_k}$ reikšmės yra tiksliausios, ir parametrai a_k $(k = \overline{0, n})$ apskaičiuojami atsižvelgiant į sąlygą $F(x_{i_k}, y_{i_k}, \dots, z_{i_k}) = \tilde{f}_{i_k}$, $k = \overline{0, n}$.

Aišku, kad taip apskaičiuota funkcija $F(x, y, \dots z)$ sutampa su interpoliuojančiąja funkcija, einančia per taškus $(x_{i_k}, y_{i_k}, \dots, z_{i_k}, \tilde{f}_{i_k}), (k = \overline{0, n})$.

Vidurkių metodas. Šiuo metodu parametrai a_k $(k = \overline{0, n})$ apskaičiuojami taip, kad $f(x_i, y_i, \dots z_i)$ ir $F(x_i, y_i, \dots z_i)$ skirtumų suma būtų lygi nuliui, t.y. $z = \sum_{i=0}^{m} (f(x_i, y_i, \dots z_i) - F(x_i, y_i, \dots z_i)) = 0$.

Mažiausių kvadratų metodas. Tai dažniausiai taikomas suglodinimo metodas. Jis formuluojamas taip: koeficientus a_k $\left(k=\overline{0,n}\right)$ reikia apskaičiuoti taip, kad $f\left(x_i, y_i, \ldots z_i\right)$ ir $F\left(x_i, y_i, \ldots z_i\right)$ skirtumų kvadratų suma būtų pati mažiausia, t.y. reikia minimizuoti

$$z=\sum_{i=0}^{m}\left(F\left(x_i, y_i, \ldots z_i, a_0, \ldots, a_n\right)-\tilde{f}_i\right)^2 \quad . \tag{2}$$

Toliau nagrinėsime mažiausių kvadratų metodą.

1.1.2.1. MAŽIAUSIŲ KVADRATŲ METODAS

(2) formule nusakyta tikslo funkcija turi vienintelį ekstremumą, kuris apskaičiuojamas iš lygčių sistemos

$$\frac{\partial z}{\partial a_k}=0,\ k=\overline{0,n}. \tag{3}$$

Bendruoju atveju ši sistema yra netiesinė, taigi ją galima spręsti taikant netiesinių lygčių sprendimo metodus.

(3) sistemos formavimą ir sprendimą galima palengvinti dvejopai:

1) Jei $F\left(x, y, \ldots z, a_0, \ldots, a_n\right)$ yra *n*-tos eilės daugianaris, tai (3) lygčių sistema yra tiesinė ir jos formavimas bei sprendimas nesudaro sunkumų.

2) Vienmačiam atvejui galima taikyti vadinamąjį ištiesinimo metodą, kurio esmė yra tokia: atitinkamai parinktoje koordinačių sistemoje (X,Y) taškai $\left(x_i, \tilde{y}_i\right)$ $\left(i=\overline{1,m}\right)$ apytiksliai tenkina tiesės $Y=kX+b$ lygtį, tada mažiausių kvadratų metodas toje sistemoje realizuojamas labai paprastai.

Pavyzdžiui, tarkime, kad taškai $\left(x_i, \tilde{y}_i\right)$ tenkina dėsnį

$$y=ae^{cx}. \tag{4}$$

Logaritmuodami abi šios lygybės puses, gauname: $\ln y=\ln a+cx$. Pažymėkime: $X=x$, $Y=\ln y$, $b=\ln a$. Tada (4) lygtį koordinačių sistemoje (X,Y) galėsime užrašyti taip: $Y=cX+b$.

Ištiesinimo metodas gali būti taikomas, norint nustatyti funkcijos $F\left(x,a_0,\ldots,a_n\right)$ analizinę išraišką, t.y. sužinoti, kokį dėsnį tenkina taškai $\left(x_i, \tilde{y}_i\right)$. Šio metodo esmė yra ta, kad, turėdami tęstinį koordinačių sistemų (X,Y) rinkinį, ieškome, kurioje koordinačių sistemoje taškai $\left(x_i, \tilde{y}_i\right)$ apytiksliai tenkina tiesės $Y=kX+b$ lygtį. Tačiau šio metodo taikymas yra daugiau menas negu mokslas, todėl detaliau jo nenagrinėsime [2], [15].

1.1.2.2. MAŽIAUSIŲ KVADRATŲ METODAS, TAIKOMAS TADA, KAI $F(x,a_0,\ldots,a_n)$ YRA n-TOSIOS EILĖS DAUGIANARIS

Duoti taškai $\left(x_i, \tilde{f}_i\right)$, $i=\overline{0,m}$. Reikia rasti tokį *n*-tosios eilės $(n<<m)$ daugianarį

$$F(x,a_0,\ldots,a_n)=a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0,$$

su kuriuo

$$z=\sum_{i=1}^{m}\left(a_n x_i^n + a_{n-1}x_i^{n-1} + \ldots + a_k x_i^k + \ldots + a_1 x_i + a_0 - \tilde{f}_i\right)^2$$

turėtų mažiausią reikšmę.

Nesunku pastebėti, kad šiuo atveju (3) lygčių sistema įgis išraišką

$$a_0\sum_{i=1}^{m}x_i^k + a_1\sum_{i=1}^{m}x_i^{k+1} + \ldots + a_n\sum_{i=1}^{m}x_i^{k+n} = \sum_{i=1}^{m}\tilde{y}_i x_i^k,\ k=\overline{0,n}. \qquad (5)$$

Pavyzdžiui, kai $n=2$, tai (5) lygčių sistema bus tokia:

$$a_0 m + a_1\sum_{i=1}^{m}x_i + a_2\sum_{i=1}^{m}x_i^2 = \sum_{i=1}^{m}\tilde{y}_i,$$

$$a_0\sum_{i=1}^{m}x_i + a_1\sum_{i=1}^{m}x_i^2 + a_2\sum_{i=1}^{m}x_i^3 = \sum_{i=1}^{m}\tilde{y}_i x_i,$$

$$a_0\sum_{i=1}^{m}x_i^2 + a_1\sum_{i=1}^{m}x_i^3 + a_2\sum_{i=1}^{m}x_i^4 = \sum_{i=1}^{m}\tilde{y}_i x_i^2.$$

(5) lygčių sistema yra tiesinė, tačiau labai dažnai ji būna blogai sąlygota, t.y. mažos paklaidos apskaičiuojant sumas sukelia dideles parametrų a_k paklaidas.

1.1.2.3. SVORINIS MAŽIAUSIŲ KVADRATŲ METODAS

Naudojant mažiausių kvadratų metodą, visų taškų „svoriai“ vienodi, t.y. į visus taškus atsižvelgiama vienodai. Pabandykime kiekvienam taškui suteikti svorį, t.y. minimizuokime

$$z=\sum_{i=0}^{m}w_i\left(F(x_i,y_i,\ldots z_i,a_0,\ldots,a_n)-\tilde{f}_i\right)^2.$$

Dažnai w_i apibrėžiamas kaip atvirkščias dispersijai dydis

$$w_i=\frac{1}{\sigma_i^2}.$$

Tokiu būdu nutolusiems taškams suteikiamas mažesnis svoris ir jų įtaka glodžiajai kreivei sumažėja. Taškai, kurie yra arčiau vidurkio, turės didesnę įtaką, todėl arčiau jų bus ir aproksimuojanti kreivė. Šis

metodas taikytinas tada, kai duomenų tinklelio taškai yra plačiai išsibarstę arba yra didelių taškų nukrypimų nuo vidurkio [2].

1.1.2.4. MAŽIAUSIŲ KVADRATŲ METODAS, TAIKOMAS TADA, KAI $F(x, y, \ldots, z, a_{00\ldots0}, a_{10\ldots0}, \ldots, a_{nu\ldots t})$ YRA DAUGIAMATIS DAUGIANARIS

Duoti taškai $(x_i, y_i, \ldots, z_i, \tilde{f}_i)$, $i = \overline{0,m}$. Reikia rasti tokį daugianarį

$$F(x, y, \ldots, z, a_{00\ldots0}, a_{10\ldots0}, \ldots, a_{nu\ldots t}) = a_{00\ldots0} + a_{10\ldots0}x + \ldots + a_{n0\ldots0}x^n + a_{01\ldots0}y + \ldots + a_{0u\ldots0}y^u + \\ + a_{11\ldots a}xy + \ldots + a_{nu\ldots t}x^n y^u \ldots z^t,$$

su kuriuo

$$z = \sum_{i=1}^{m}\left(a_{00\ldots0} + a_{10\ldots0}x + \ldots + a_{n0\ldots0}x^n + a_{01\ldots0}y + \ldots + a_{0u\ldots0}y^u + a_{11\ldots a}xy + \ldots + a_{nu\ldots t}x^n y^u \ldots z^t - \tilde{f}_i\right)^2$$

turėtų mažiausią reikšmę. Tokio daugianario kintamųjų maksimalūs laipsniai atitinkamai lygūs *n*, *u*, ..., *t*, ir tokie, kad $((n+1)(u+1)\cdot\ldots\cdot(t+1) < m)$.

Daugiamačiu atveju (3) išraiška atrodys taip

$$\frac{\partial z}{\partial a_{kl\ldots p}} = 0,\ k = \overline{0,n} \cap l = \overline{0,u} \cap \ldots \cap p = \overline{0,t}, \tag{6}$$

o lygčių sistema turės pavidalą

$$a_{00\ldots0}\sum_{i=1}^{m} x_i^k y_i^l \ldots z_i^p + a_{10\ldots0}\sum_{i=1}^{m} x_i^{k+1} y_i^l \ldots z_i^p + \ldots + a_{n0\ldots0}\sum_{i=1}^{m} x_i^{k+n} y_i^l \ldots z_i^p + \\ + a_{01\ldots0}\sum_{i=1}^{m} x_i^k y_i^{l+1} \ldots z_i^p + \ldots + a_{0u\ldots0}\sum_{i=1}^{m} x_i^k y_i^{l+u} \ldots z_i^p + a_{11\ldots0}\sum_{i=1}^{m} x_i^{k+1} y_i^{l+1} \ldots z_i^p + \ldots + \\ + a_{nu\ldots0}\sum_{i=1}^{m} x_i^{k+n} y_i^{l+u} \ldots z_i^p + \ldots + a_{nu\ldots t}\sum_{i=1}^{m} x_i^{k+n} y_i^{l+u} \ldots z_i^{p+t} = \sum_{i=1}^{m} \tilde{f}_i x_i^k y_i^l \ldots z_i^p, \\ k = \overline{0,n} \cap l = \overline{0,u} \cap \ldots \cap p = \overline{0,t}. \tag{7}$$

Pavyzdžiui, dvimačiu atveju, kai $n = 1$ ir $u = 1$ (7) lygčių sistema bus tokia:

$$a_{00}m + a_{10}\sum_{i=1}^{m} x_i + a_{01}\sum_{i=1}^{m} y_i + a_{11}\sum_{i=1}^{m} x_i y_i = \sum_{i=1}^{m} \tilde{f}_i,$$

$$a_{00}\sum_{i=1}^{m} x_i + a_{10}\sum_{i=1}^{m} x_i^2 + a_{01}\sum_{i=1}^{m} x_i y_i + a_{11}\sum_{i=1}^{m} x_i^2 y_i = \sum_{i=1}^{m} \tilde{f}_i x_i,$$

$$a_{00}\sum_{i=1}^{m} y_i + a_{10}\sum_{i=1}^{m} x_i y_i + a_{01}\sum_{i=1}^{m} y_i^2 + a_{11}\sum_{i=1}^{m} x_i y_i^2 = \sum_{i=1}^{m} \tilde{f}_i y_i,$$

$$a_{00}\sum_{i=1}^{m}x_i y_i + a_{10}\sum_{i=1}^{m}x_i^2 y_i + a_{01}\sum_{i=1}^{m}x_i y_i^2 + a_{11}\sum_{i=1}^{m}x_i^2 y_i^2 = \sum_{i=1}^{m}\tilde{f}_i x_i y_i \ .$$

1.1.2.5. SUGLODINIMAS NAUDOJANT BANGELES

Bangelės (wavelets) yra funkcijos, kurios tenkina tam tikrus matematinius reikalavimus ir kurių pagalba galima išreikšti kitas funkcijas. Toks aproksimavimo metodas nėra naujas, tiesiog XXI amžiuje pradėta juo daugiau domėtis. Aproksimacija naudojant funkcijų superpoziciją jau buvo žinoma XIX amžiaus pradžioje, kai Furje (Joseph Fourier) pastebėjo, jog naudojant sinusus ir kosinusus galima išreikšti įvairias funkcijas.

Metodo teorija sudėtinga, kadangi bendru atveju naudojama bangos funkcija bei superpozicijos operacija. Algoritmo sudėtingumo eilė apibrėžiama skaičiumi N^{3d-2}, kur N - tinklelio taškų skaičius, o d - erdvės dimensija [27].

1.2. APROKSIMAVIMO METODŲ APŽVALGA

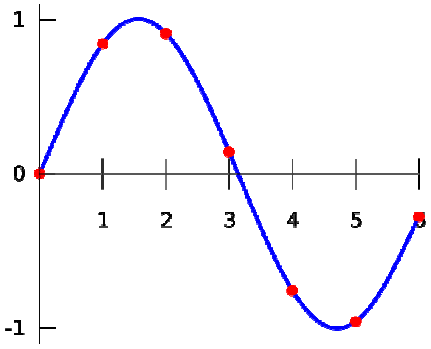
Aproksimacijos metodų yra gana daug, keletą iš jų paminėjome skyreliuose aukščiau. Tačiau nepaminėjome jų privalumų ir trūkumų. Pradėkime nuo paprasčiausio atvejo - tiesinės interpoliacijos. Teisinė interpoliacija bloga tuo, kad tokiu būdu sukonstruota funkcija neturi išvestinių taškuose, per kuriuos eina tiesinės interpoliacijos laužtė. Be to, naudojamos formulės struktūra bei dėsningumas tampa labai sudėtingas augant erdvės dimensijai. Polinominės interpoliacijos atveju diferencijavimo problema nebeiškyla, nes interpoliuojanti kreivė (hyperpaviršius) yra tam tikro laipsnio daugianaris, ir todėl yra be galo daug kartų diferencijuojama. Tačiau šis metodas turi kitų trūkumų. Jis atima santykinai daug skaičiavimo laiko arba kartais jis gali ne tiksliai interpoliuoti taškus, ypač galiniuose duomenų tinklelio taškuose. Šis trūkumas gali būti pašalintas naudojant splainų interpoliaciją. Tačiau interpoliacinio splaino struktūra yra žymiai sudėtingesnė negu interpoliacinio daugianario, ypač jeigu yra daug vingio taškų arba jeigu nagrinėjamas intervalas suskaidomas į daug duomenų langelių. Pavyzdžiui, tam tikrus duomenis aproksimuojančio interpoliacinio daugianario išraiška

$$f(x) = -0.0001521x^6 - 0.003130x^5 0.07321x^4 - 0.3577x^3 + 0.2255x^2 + 0.9038x$$

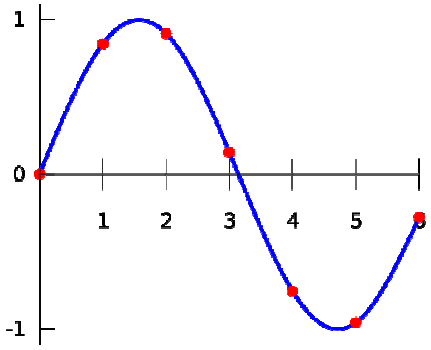
nusakoma 7 koeficientais bei daugianario laipsniu. Tuo tarpu tuos pačius duomenis aproksimuojantis kubinis interpoliacinis splainas atrodo taip

$$f(x)=\begin{cases} -0.1522x^3+0.9937x, & x\in[0,1], \\ -0.01258x^3-0.4189x^2+1.4126x-0.1396 & x\in[1,2], \\ 0.1403x^3-1.3359x^2+3.2467x-1.3623 & x\in[2,3], \\ 0.1579x^3-1.4945x^2+3.7225x-1.8381 & x\in[3,4], \\ 0.05375x^3-0.2450x^2-1.2756x+4.8259 & x\in[4,5], \\ -0.1871x^3+3.3673x^2-19.3370x+34.9282 & x\in[5,6], \end{cases}$$

o norint nusakyti tokį splainą, reikia žinoti 24 koeficientus, splaino laipsnį, bei 6 duomenų langelius. Vizualiai palyginus rezultatus reikšmingo skirtumo nematyti (pav. 1.1 ir pav. 1.2)



1.1 pav Polinominė interpoliacija



1.2 pav Splaino interpoliacija

Jei IDW metodu norėsime rasti funkcijos reikšmę labai arti duomenų tinklelio taško, tai apskaičiuodami $w_k(x)$ galime gauti neapibrėžtumą (dalybą iš nulio) arba iššaukti labai dideles paklaidas. Be to šiuo metodu apskaičiuojant funkcijos reikšmę, įvertinama visų taškų įtaka, t.y. į sumas įeina visi duomenų tinklelio taškai, kas yra labai neracionalu. Kriging‘o interpoliacijos metu kiekvienam taškui priskiriamas svoris. Jie randami iš tiesinės lygčių sistemos. Apskaičiuojant funkcijos reikšmę norimuose taškuose, kiekvieną kartą reikia perskaičiuoti tuos svorius. Vadinasi, jei turime tarkime 1000 duomenų langelių taškų, tai kiekvieną kartą apskaičiuojant funkcijos reikšmę, teks išspręsti 1000 lygčių. Pagrindinis Kriging‘o metodo pliusas yra tas, kad jis pateikia interpoliacijos kreivės pasikliovimo intervalus. Svorinio mažiausių kvadratų metodo trūkumas yra tas, kad reikia papildomai nagrinėti taškus, t.y. apskaičiuoti jų vidurkį bei dispersiją, ir po to kiekvienam taškui priskirti svorį. Nagrinėjant tarkime tūkstantį taškų, reikia atlikti jau gana daug papildomų skaičiavimų, o rezultatas dažnai būna mažai geresnis už paprasto mažiausių kvadratų metodo. Suglodinimas naudojant bangeles yra labai sudėtingas metodas, paremtas daugiamačių bangų funkcijų teorija, sudėtingais daugiamačių bazės funkcijų parinkimo metodais. Taigi atsižvelgiant į paminėtus faktus, buvo nuspręsta šiame darbe daugiamates funkcijas aproksimuoti daugianariais, o jų koeficientus apskaičiuoti paprastu mažiausių kvadratų metodu.

1.3. FUNKCIJŲ APROKSIMAVIMO TIKSLUMAS

Tarkime, kad tam tikru metodu gavome nagrinėjamus duomenis aproksimuojančią funkciją. Tačiau kyla klausimas: kaip tiksliai ta funkcija aproksimuoja duomenis? Arba tarkime, kad įvairiais metodais gavome kelias aproksimuojančias funkcijas. Kaip iš jų išrinkti tinkamiausią? Kad atsakytume į šiuos klausimus turime įvesti tam tikrus aproksimacijos tikslumo koeficientus – kokybės kriterijus arba panaudoti tam tikrus aproksimacijos modelių palyginimo metodus.

1.3.1. APROKSIMACIJOS KOKYBĖS KRITERIJAI

Kvadratinių nuokrypių suma SS_{reg}. Kvadratinių nuokrypių suma SS_{reg} (Sum-of-Squares) vadinama vertikalių atstumų nuo taško iki kreivės kvadratų suma:

$$SS_{reg} = \sum_i \left(y_i - \hat{y}_i\right)^2 ,$$

čia y_i - stebimos funkcijos reikšmės, $\hat{y}_i$ - sumodeliuotos funkcijos reikšmės. Siekiama, kad ji būtų kuo mažesnė. Ši suma išreiškiama funkcijos reikšmių kvadratiniais matavimo vienetais.

Standartinis nuokrypis S_{xy} ***arba*** S_e. Dydis S_{xy} yra vertikalių atstumų nuo taškų iki kreivės standartinis nuokrypis. Kadangi minėti atstumai nuo taško iki kreivės dažnai vadinami liekanomis, tai dydis S_{xy} dar vadinamas liekanų standartiniu nuokrypiu arba tiesiog standartiniu nuokrypiu ir taip pat žymimas S_e. Jo matavimo vienetai yra tokie patys kaip ir funkcijos reikšmių.

Šiame darbe S_e apskaičiuosime naudodami SS_{reg} ir laisvės laipsnius - *df* (degrees of freedom), jie lygūs nagrinėjamų taškų skaičiui, minus aproksimuojančio daugianario parametrų skaičiui. Formulė tokia:

$$S_e = \sqrt{\frac{SS_{reg}}{df}}$$

Apibrėžtumo koeficientas R^2. R^2 įvertina aproksimacijos kokybę. Jo reikšmės yra tarp 0 ir 1, be to jis neturi dimensijos. Kuo jo reikšmė aukštesnė, tuo parinkta kreivė eina arčiau taškų. Vien tik R^2 nepatartina naudoti kaip pagrindinio kriterijaus, renkant tinkamą aproksimacijos modelį. Aukštas R^2 parodo, kad parinkta kreivė eina labai arti taškų, bet tai dar nereiškia, kad aproksimacija yra pakankamai „gera“. Kai R^2 lygus nuliui, parinktas modelis aproksimuoja taškus ne geriau negu horizontali linija, einanti per funkcijos reikšmių vidurkį. Tokiu atveju funkcijos argumento reikšmės žinojimas nepadeda nuspėti funkcijos reikšmės. Kai R^2 lygus vienam, sumodeliuota kreivė eina

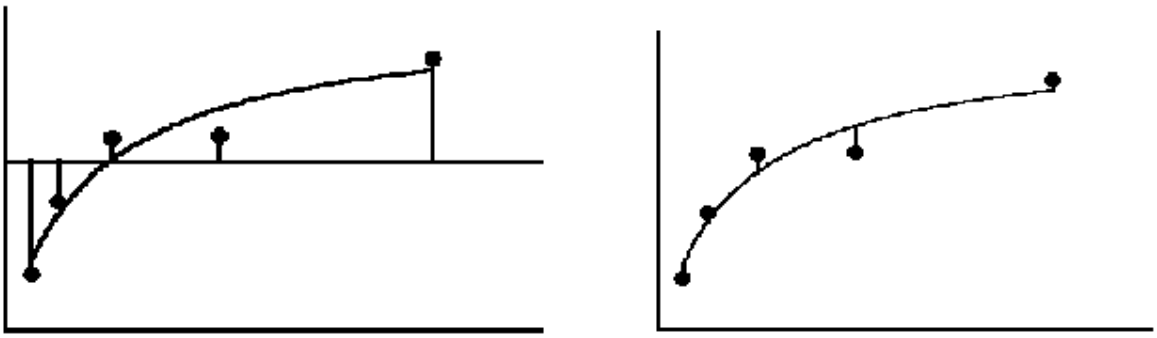
tiksliai per visus taškus. Vadinasi, žinodami funkcijos argumento reikšmę, galime tiksliai nustatyti funkcijos reikšmę. Todėl R^2 gali būti interpretuojamas kaip funkcijos reikšmių dalis, kurią paaiškina modelis. R^2 yra apskaičiuojamas naudojant kvadratinių nuokrypių sumą SS_{reg}, kuri yra lygi taškų vertikalių nuokrypių nuo parinktos kreivės kvadratų sumai. SS_{reg} normuojama, padalinant iš kvadratinių nuokrypių sumos SS_{tot}, kuri yra lygi taškų vertikalių nuokrypių nuo funkcijos vidurkio kvadratų sumai:

$$SS_{tot} = \sum_i \left(y_i - \bar{y}\right)^2 ,$$

čia y_i - stebimos funkcijos reikšmės, $\bar{y}_i$ - stebimos funkcijos reikšmių vidurkis. Jei parinkta kreivė tiksliai aproksimuoja taškus, tai SS_{reg} bus daug mažesnė už SS_{tot}. R^2 išraiška:

$$R^2 = 1 - \frac{SS_{reg}}{SS_{tot}} .$$

Paveikslėliai žemiau iliustruoja R^2 apskaičiavimą (1.3 pav.). Abiejuose paveikslėliuose pavaizduoti tie patys taškai bei parinkta ta pati aproksimuojanti kreivė. Kairiajame paveikslėlyje yra pavaizduota horizontali linija, nubrėžta per funkcijos reikšmių vidurkį, ir vertikalios linijos, kurios parodo, kaip toli kiekvienas taškas yra nutolęs nuo funkcijos reikšmių vidurkio. Dešinėje pavaizduoti vertikalūs taškų atstumai nuo parinktos kreivės. Teoriškai R^2 gali įgyti ir neigiamas reikšmes. Jei SS_{reg} bus didesnė negu SS_{tot}, tai R^2 bus neigiamas. Kitaip tariant, R^2 bus neigiamas, kai parinktas modelis taškus aproksimuos prasčiau negu horizontali linija, nubrėžta per funkcijos reikšmių vidurkį. Taip gali atsitikti tada, kai parenkamas netinkamas modelis arba netinkami modelio parametrai. Norint palyginti dviejų modelių aproksimacijos kokybę, nepatartina lyginti vien tik R^2 reikšmių [19], [28].



1.3 pav. Nuokrypių atstumai

1.3.2. DVIEJŲ MODELIŲ PALYGINIMAS NAUDOJANT F KRITERIJŲ

Jei nežinote, kurį iš dviejų modelių pasirinkti, reikia pritaikyti abu modelius ir palyginti rezultatus. Pirmiausia lyginant du modelius reikia įsitikinti, kad abu jie turi matematinę prasmę. Pavyzdžiui, jei gautas R^2 koeficientas yra neigiamas ar labai mažas arba jei SS_{reg} yra labai didelė, tai tokius modelius galima iš karto atmesti ir apsieiti be statistinės analizės. Jei abu modeliai tinkami analizei, pirmiausia reiktų naudoti SS_{reg} kaip kokybės kriterijų. Jei sudėtingesnis modelis turi aukštesnę SS_{reg} negu paprastesnis modelis, tai reiktų pasirinkti paprastesnį. Taip atsitiks gana retai, kadangi sudėtingesnis modelis beveik visada turi mažesne SS_{reg}, nes tokio modelio kreivė turi daugiau vingių ir todėl eina arčiau taškų. Jei abu modeliai yra tinkami analizei ir sudėtingesnio modelio SS_{reg} yra mažesnė, tai tenka atlikti statistinę analizę, norint sužinoti, kuris modelis yra tinkamesnis. Kad tai nuspręstume, naudosime F kriterijų.

F kriterijus lygina du aproksimacijos modelius, kai sudėtingesnio modelio kvadratinių nuokrypių suma yra mažesnė negu paprastesnio modelio. Keliamas klausimas, ar kvadratinių nuokrypių sumos sumažėjimas yra vertas papildomų kintamųjų "kainos" (laisvės laipsnių skaičiaus praradimo). F kriterijus pateikia *p* reikšmę, kuri ir atsako į klausimą: ar paprastesnis modelis iš tikrųjų yra tinkamesnis? Kokia tikimybė, kad nagrinėjami taškai yra atsitiktinai daug geriau aproksimuojami sudėtingesnio modelio? Jei *p* reikšmė gaunama maža, tai daroma išvada, kad sudėtingesnis modelis yra tinkamesnis už paprastesnį. Dauguma tyrinėtojų pasirenka *p* slenkstį 0.05. Paprastesnis modelis būna tinkamesnis tik tada, kai jis yra specialus atvejis sudėtingesniojo modelio. Kitaip tariant du modeliai turi būti susiję. Parenkant tam tikrų parametrų tam tikras reikšmes (sakykime 0 ar 1) iš sudėtingesnio modelio galime gauti paprastesnį.

Kad suvoktume, kaip F kriterijus veikia, reiktų palyginti dviejų modelių kvadratinių nuokrypių sumas, bei laisvės laipsnius. Panagrinėkime pavyzdį pateiktą 1 lentelėje.

Lentelė nr. 1

	Sudėtingesnis modelis	Paprastesnis modelis	Padidėjimas, %
Laisvės laipsnių skaičius	7	9	28.57%
Kvadratinių nuokrypių suma	52330	221100	322.51%

Pereidami nuo sudėtingesnio modelio prie paprastesnio, mes įgyjame du laisvės laipsnius, nes paprastesnis modelis turi dviem kintamaisiais mažiau negu sudėtingesnis. Vadinasi, laisvės laipsnių skaičius padidėjo 28.6%. Jei paprastesnis modelis būtų tinkamesnis, tikėtumėmės, kad ir kvadratinių nuokrypių suma pasikeistų 28.6%. Tačiau iš tikrųjų, pereinant nuo sudėtingesnio modelio prie

paprastesnio, kvadratinių nuokrypių suma padidėjo 322%. Vadinasi, kvadratinių nuokrypių sumos padidėjimas yra 11.29 karto didesnis negu laisvės laipsnių skaičiaus padidėjimas. Taigi F koeficientas 11.29. Kitaip tariant, jei paprastesnis modelis tinkamesnis, tai tikimasi, kad santykinis kvadratinių nuokrypių sumų padidėjimas bus lygus santykiniam laisvės laipsnių skaičiaus padidėjimui, t.y.

$$\frac{SS1_{reg} - SS2_{reg}}{SS2_{reg}} \approx \frac{df1 - df2}{df2}$$

Koeficientas F įvertina ryšį tarp santykinio laisvės laipsnių skaičiaus padidėjimo ir santykinio kvadratinių nuokrypių sumų padidėjimo:

$$F = \frac{\frac{SS1_{reg} - SS2_{reg}}{SS2_{reg}}}{\frac{df1 - df2}{df2}}$$

arba

$$F = \frac{\frac{SS1_{reg} - SS2_{reg}}{df1 - df2}}{\frac{SS2_{reg}}{df2}}.$$

F koeficientas visada yra susietas su tam tikru skaitiklio laisvės laipsnių skaičiumi ir su tam tikru vardiklio laisvės laipsnių skaičiumi. Šiuo atveju skaitiklio laisvės laipsnių skaičius $df1 - df2$, o vardiklio laisvės laipsnių skaičius $df2$. Jei paprastesnis modelis tinkamesnis, tai tikimasi F reikšmę gauti artimą 1. Jei F koeficientas daug didesnis negu 1 tai yra dvi galimybės:

1) sudėtingesnis modelis yra tinkamesnis;

2) paprastesnis modelis yra tinkamesnis, tačiau atsitiktinai taškai išsidėstę taip, jog sudėtingesnis modelis geriau aproksimuoja taškus. p reikšmė parodo kaip retai būna toks sutapimas. Jei p reikšmė maža, rekomenduojama pasirinkti sudėtingesni atvejį. Jei p reikšmė didesnė negu 0.05, rekomenduojama pasirinkti paprastesnį modelį [20].

1.3.2.1. p REIKŠMĖS APSKAIČIAVIMAS

p reikšmė apskaičiuojama naudojant F koeficiento skirstinio funkciją

$$G(x) = I_{\frac{d1 \cdot x}{d1 \cdot x + d2}}\left(\frac{d1}{2}, \frac{d2}{2}\right),$$

čia $d1$, $d2$ su F koeficientu susieti laisvės laipsniai, $I_x(a,b)$ suderinta nepilnoji Beta funkcija (regularized incomplete Beta function)

$$I_x(a,b) = \frac{B(x,a,b)}{B(a,b)},$$

čia $B(x,a,b)$ nepilnoji Beta funkcija (incomplete Beta function)

$$B(x,a,b) = \int_0^x t^{a-1}(1-t)^{b-1}\,dt,$$

o $B(a,b)$ - Beta funkcija

$$B(a,b) = \int_0^1 t^{a-1}(1-t)^{b-1}\,dt.$$

Taigi žinant *F*, randama p reikšmė

$$p = 1 - G(F).$$

Papildoma informacija šaltiniuose [21], [22], [24], [25]. Turint *F* koeficientą bei su juo susietus laisvės laipsnius, *p* reikšmę galima apskaičiuoti naudojant įrankį, pateiktą šaltinyje [23].

1.3.3. JARQUE-BERA KRITERIJUS

Jarque-Bera kriterijus skirtas nustatyti, ar imties dydžiai pasiskirstę pagal normalųjį dėsnį. Jis apskaičiuojamas naudojant imties asimetrijos bei smailiaviršūniškumo koeficientą. Kriterijaus statistika *JB* apibrėžiama

$$JB = \frac{n}{6}\left(S^2 + \frac{(K-3)^2}{4}\right),$$

kur *n* imties plotis, *S* – asimetrijos koeficientas, *K* – smailiaviršūniškumo koeficientas. Jie apibrėžiami formulėmis

$$S = \frac{\mu_3}{\sigma^3} = \frac{\mu_3}{\left(\sigma^2\right)^{3/2}} = \frac{\frac{1}{n}\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^3}{\left(\frac{1}{n}\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2\right)^{3/2}},$$

$$K = \frac{\mu_4}{\sigma^4} = \frac{\mu_4}{\left(\sigma^2\right)^2} = \frac{\frac{1}{n}\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^4}{\left(\frac{1}{n}\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2\right)^2}.$$

Čia μ_3 ir μ_4 trečias ir ketvirtas centrinis momentas, $\bar{x}$ - imties vidurkis, σ^2 - antras centrinis momentas arba dispersija. Kai imties plotis pakankamai didelis (daugiau negu 400), statistika *JB* turi χ^2 skirstinį su dviem laisvės laipsniais ir gali būti naudojama nulinės hipotezės, kad duomenys

pasiskirstę pagal normalųjį dėsnį, tikrinimui. Jei gauta JB statistikos reikšmė absoliučiai didesnė už kritinę χ^2 reikšmę 5.991, nulinė hipotezė atmetama [29].

1.3.4. VOLDO IR VOLFOVICO RUNS KRITERIJUS

„Runs“ kriterijus skirtas hipotezės, apie atsitiktinį dydžių išsidėstymą dviejose grupėse, tikrinimui. Pažymėkime taškų, esančių virš aproksimuojančios kreivės, skaičių N_a, o po kreive - N_b. Visų taškų skaičius N ir $N = N_a + N_b$. „Run“ yra nenulinė elementų seka, priklausanti vienai elementų grupei. Jų skaičių imtyje žymėsime *R*. Pavyzdžiui, sekoje +++---++---+++-- „run“ skaičius yra $R = 6$. Pažymėkime vidurkį

$$\mu = \frac{2N_a N_b}{N} + 1$$

ir dispersiją

$$\sigma^2 = \frac{2N_a N_b \left(2N_a N_b - N\right)}{N^2 \left(N - 1\right)} = \frac{\left(\mu - 1\right)\left(\mu - 2\right)}{N - 1}.$$

Kai imties plotis pakankamai didelis (daugiau negu 100), dydis

$$Z = \frac{R - \mu}{\sqrt{\sigma^2}}$$

turi standartinį normalųjį skirstinį ir gali būti naudojamas minėtos hipotezės tikrinimui. Jei gauta statistikos Z reikšmė absoliučiai didesnė už kritinę 1.960, tai hipotezė, kad dydžiai pasiskirstę dviejose grupėse atsitiktinai, atmetama [30].

2 TIRIAMOJI DALIS

2.1. DARBE SPRENDŽIAMI UŽDAVINIAI

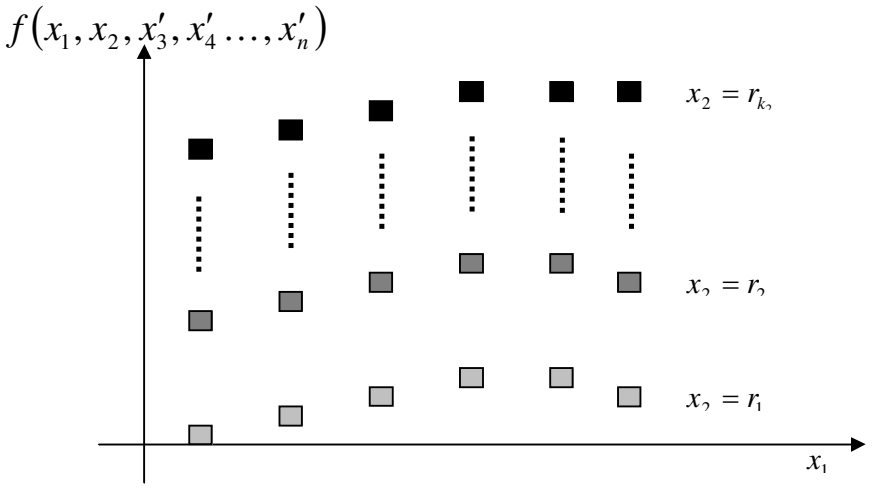
Šiuo metu daugiamačių duomenų aproksimavimo uždaviniai tampa vis aktualesni. Viena iš problemų sprendžiant šiuos uždavinius yra daugiamatė erdvė. Nėra galimybės joje pavaizduoti nagrinėjamą funkciją. Vienas iš būdų, norint sužinoti daugiamatės funkcijos „elgesį“, yra duomenų pjūvių (funkcijos priklausomybių nuo laisvai pasirenkamo kintamojo) sudarymas. Nagrinėjant tokias priklausomybes, galima įžvelgti, kaip funkcija priklauso nuo to, laisvai pasirinkto kintamojo. Matant priklausomybę, galima parinkti tinkamą aproksimuojančią funkciją, mūsų atveju norimo laipsnio daugianarį. Naudojant tokį metodą, duomenys nagrinėjami pažingsniui, tam tikromis dalimis. Todėl toks duomenų pjūvių sudarymas atima daug laiko. Norėtųsi turėti greitesnį metodą ir tokį, kuris skaičiavimams iškarto panaudotų visus duomenis. Taip pat yra reikalingi aproksimacijos kokybės kriterijai, kurių pagalba būtų galima įvertinti aproksimacijos tikslumą ar palyginti du aproksimacijos modelius. Tokių kokybės kriterjų pagalba būtų galima įvertinti abu, aukščiau minėtus, aproksimacijos metodus, atlikti jų lyginamąją analizę. Pagrindinis darbo tikslas yra surasti efektyviausią (arba kelis) aproksimacijos modelius, t.y. tokius, kurie būtų kuo paprastesni ir kiek įmanoma tikslesni. Būtų labai patogu, jei visus išvardintus uždavinius galima būtų spęsti naudojant programinį įrankį. Taigi reikalinga sistema:

- kuri leistų sudaryti duomenų pjūvius laisvai pasirenkamo kintamojo atžvilgiu;
- kuri duomenų pjūvius vaizduotų grafiškai;
- kuri leistų greitai peržiūrėti norimus duomenų pjūvius;
- kuri leistų pasirinkti norimo laipsnio daugianarį pjūvių aproksimacijai;
- kuri leistų grafiškai palyginti keleto funkcijų duomenų pjūvius;
- kuri leistų greitai peržvelgti duomenų pjūvių aproksimacijas;
- kuri realizuotų polinominės aproksimacijos metodą;
- kuri pateiktų aproksimacijos kokybės kriterijus;
- kuri sudarytų galimybę palyginti kelis modelius kokybės kriterijų atžvilgiu;
- kuri pasiūlytų vartotojui kelis efektyviausius modelius;
- kuri leistų apdoroti kuo didesnio matumo funkcijas;
- kuri vėliau galėtų buti papildoma kitais aproksimacijos metodais;
- kurią būtų nesudėtina naudoti;
- kuri pateiktų aproksimuojančio daugianario koeficientus;
- kuri keltų kompiuteriui kuo mažesnius reikalavimus;

Tokios sistemos pagalba būtų galima greitai ir patogiai išspręsti rūpimus klausimus.

2.2. PIRMAS DAUGIAMATĖS FUNKCIJOS APROKSIMAVIMO METODAS

Tarkime turime funkciją, kuri priklauso nuo n kintamųjų $f(x_1, x_2, \ldots, x_n)$. Kintamųjų $x_1, x_2, \ldots, x_n$ įgyjamų reikšmių skaičius yra baigtinis ir lygus atitinkamai $k_1, k_2, \ldots, k_n$. Fiksuokime kintamųjų $x_3, x_4, \ldots, x_n$ reikšmes tokiu būdu: kintamajam x_3 priskirkime bet kurią jo reikšmę iš k_3 galimų; x_4 priskirkime bet kurią jo reikšmę iš k_4 galimų; ir t.t. Gausime fiksuotų kintamųjų rinkinį $x_3', x_4', \ldots, x_n'$. Tarkime, kad kintamojo x_2 reikšmės yra $r_1, r_2, \ldots, r_{k_2}$. Jei koordinačių sistemoje y ašyje pateiktume funkcijos $f(x_1, r_1, x_3', x_4' \ldots, x_n')$ reikšmes, o x ašyje x_1 reikšmes ir atidėsime taškus, gautume duomenų pjūvį. Tokių pjūvių skaičius lygus $m = k_2 \cdot k_3 \cdot \ldots \cdot k_n$. Ta patį galime atlikti ir kai x_2 reikšmė lygi $r_2, r_3, \ldots, r_{k_2}$. Tokiu būdu galime gauti k_2 kintamojo x_2 priklausomybių. 2.1 paveiksle pavaizduotos šios priklausomybės.



2.1 pav. Duomenų pjūviai.

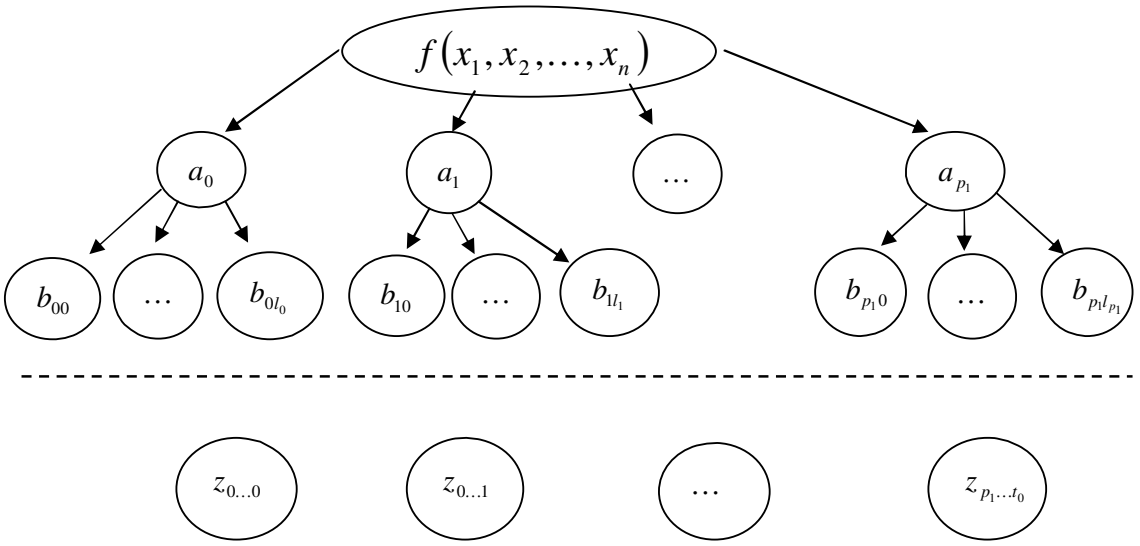
Šiame darbe kintamąjį x_1 vadinsime x ašies kintamuoju, o x_2 - grupavimo kintamuoju. Iš taškų išsidėstymo galime nuspręsti, kokio laipsnio daugianariu galime aproksimuoti funkciją $f(x_1, x_2, x_3', x_4' \ldots, x_n')$. Tarkime aproksimuosime p-tojo laipsnio daugianariu $a_0 + a_1 x + a_2 x^2 + \ldots + a_p x^p$. Tuomet funkciją $f(x_1, r_1, x_3', x_4' \ldots, x_n')$ atitiks daugianaris $a_0^{r_1} + a_1^{r_1} x_1 + a_2^{r_1} x_1^2 + \ldots + a_p^{r_1} x_1^p$, funkciją $f(x_1, r_2, x_3', x_4' \ldots, x_n')$ atitiks daugianaris $a_0^{r_2} + a_1^{r_2} x_1 + a_2^{r_2} x_1^2 + \ldots + a_p^{r_2} x_1^p$, ..., funkciją $f(x_1, r_{k_2}, x_3', x_4' \ldots, x_n')$ - $a_0^{r_{k_2}} + a_1^{r_{k_2}} x_1 + a_2^{r_{k_2}} x_1^2 + \ldots + a_p^{r_{k_2}} x_1^p$.

Pakeitę fiksuotų kintamųjų $x_3, x_4, \ldots, x_n$ reikšmes, gausime kitą aproksimuojančių daugianarių rinkinį. Tokių daugianarių bus lygiai $m_1 = k_2 \cdot k_3 \cdot \ldots \cdot k_n$. Norėdami rasti daugianarių koeficientus, remdamiesi mažiausių kvadratų metodu, sudarysime (5) lygčių sistemą. Ją spręsime Gauso metodu. Taigi gausime daugianario koeficientų aibes $a_0 = \left\{a_0^{r_1}, a_0^{r_2}, \ldots, a_0^{r_{k_2}}, \ldots\right\}$, $a_1 = \left\{a_1^{r_1}, a_1^{r_2}, \ldots, a_1^{r_{k_2}}, \ldots\right\}$, ..., $a_p = \left\{a_p^{r_1}, a_p^{r_2}, \ldots, a_p^{r_{k_2}}, \ldots\right\}$. Šių aibių elementų skaičius vienodas ir lygus m_1. Visi šie koeficientai jau nebepriklauso nuo x ašies kintamojo x_1, bet vis dar priklauso nuo kintamųjų $x_2, x_3, \ldots, x_n$. Todėl juos galime laikyti kaip funkcijas. Vietoj nagrinėjamos funkcijos pasirinkime bet kurią koeficientų aibę, tarkime a_0. x ašies kintamuoju tegu būna prieš tai buvęs grupavimo kintamasis x_2, o naują grupavimo kintamąjį pasirinkime iš fiksuotų kintamųjų sąrašo. Tarkim jis bus x_3. Likusių fiksuotų kintamųjų reikšmes pasirenkame laisvai. Vėl atidėję duomenų pjūvio taškus, pasirenkame aproksimuojančio daugianario laipsnį, tarkime l, t.y. $b_0 + b_1 x + b_2 x^2 + \ldots + b_l x^l$. Tokiais daugianariais aproksimuojame visus galimus duomenų pjūvius. Remdamiesi mažiausių kvadratų metodu, kiekvienam pjūviui sudarome (5) lygčių sistemą, ją išsprendžiame Gauso metodu. Gausime koeficientų $b_{00}, b_{01}, \ldots, b_{0l}$ masyvus, kurių elementų skaičius irgi vienodas ir lygus $m_2 = k_3 \cdot k_4 \cdot \ldots \cdot k_n$. Šių masyvų elementai nepriklauso jau nuo dviejų kintamųjų x_1 ir x_2, bet vis dar priklauso nuo likusiųjų $x_3, x_4, \ldots, x_n$. Tęsdami šį procesą galiausiai gausime koeficientų aibes, tarkime $z_{0\ldots0}, z_{0\ldots1}, \ldots, z_{p\ldots t}$, kurios sudarytos iš $m_n = 1$ elementų ir nebepriklauso nuo nei vieno funkcijos $f(x_1, x_2, \ldots, x_n)$ kintamojo. Toks algoritmas 2.2 paveiksle pavaizduotas kaip medis.

Imkime kaip pavyzdį funkciją $f(x_1, x_2, x_3)$. Tarkime, kad x ašies kintamasis x_1, o grupavimo x_2. Trečiam kintamajam priskirkime vieną iš jo įgyjamų reikšmių x_3'. Tarkime, kad funkcijos $f(x_1, x_2, x_3')$ nuo kintamojo x_1 priklauso tiesiškai, t.y. galime aproksimuoti daugianariu $a_0 + a_1 x_1$. Kitame žingsnyje x ašies kintamuoju tampa x_2, grupavimo - x_3, o fiksuotu kintamųjų nebėra. Tarkime, kad funkcijos $a_0(x_2, x_3)$ ir $a_1(x_2, x_3)$ nuo kintamojo x_2 priklauso irgi tiesiškai ir jas atitinkamai galima aproksimuoti daugianariais $b_{00} + b_{01} x_2$ ir $b_{10} + b_{11} x_2$. Paskutiniame žingsnyje x ašies kintamuoju tampa x_3, o grupavimo kintamojo nebėra. Sakykime, kad funkcijas $b_{00}(x_3)$ ir $b_{01}(x_3)$ galime aproksimuoti kvadratiniais trinariais $c_{000} + c_{001} x_3 + c_{002} x_3^2$ ir $c_{010} + c_{011} x_3 + c_{012} x_3^2$, o $b_{10}(x_3)$ ir $b_{11}(x_3)$ tiesėmis $c_{100} + c_{101} x_3$ ir $c_{110} + c_{111} x_3$. Kadangi nebėra nei fiksuotų kintamųjų, nei grupavimo kintamojo, tai koeficientų aibės $c_{000}, c_{001}, c_{002}, c_{010}, c_{011}, c_{012}, c_{100}, c_{101}, c_{110}, c_{111}$ sudarytos iš vieno elemento ir nepriklauso nei nuo vieno kintamojo, t.y. yra konstantos. Taigi aproksimuojanti funkcija turės pavidalą:

$$F(x_1, x_2, x_3) = a_0 + a_1 x_1 = (b_{00} + b_{01} x_2) + (b_{10} + b_{11} x_2) x_1 =$$

$$= \left((c_{000} + c_{001} x_3 + c_{002} x_3^2) + (c_{010} + c_{011} x_3 + c_{012} x_3^2) x_2 \right) + \left((c_{100} + c_{101} x_3) + (c_{110} + c_{111} x_3) x_2 \right) x_1 .$$



2.2 pav. Koeficientų aibių grafas

2.3. ANTRAS DAUGIAMATĖS FUNKCIJOS APROKSIMAVIMO METODAS

Antras daugiamatės funkcijos aproksimacijos metodas paremtas polinominės interpoliacijos metodu. Tarkime, kad duoti taškai $(x_i, y_i, \ldots, z_i, \tilde{f}_i)$, $i = \overline{0, m}$. Kiekvieno kintamojo skirtingų įgyjamų reikšmių skaičių pažymėkime atitinkamai $k_x, k_y, \ldots, k_z$. Polinominės interpoliacijos atveju tiksliai per visus duomenų tinklelio taškus eis tik vienas daugianaris

$$F(x, y, \ldots, z, a_{00\ldots0}, a_{10\ldots0}, \ldots, a_{nu\ldots t}) = a_{00\ldots0} + a_{10\ldots0} x + \ldots + a_{n0\ldots0} x^n + a_{01\ldots0} y + \ldots + a_{0u\ldots0} y^u +$$

$$+ a_{11\ldots a} xy + \ldots + a_{nu\ldots t} x^n y^u \ldots z^t ,$$

kurio visų kintamųjų maksimalūs laipsniai $n, u, \ldots, t$ bus vienetu mažesni, negu jų įgyjamų skirtingų reikšmių skaičius, t.y. toks daugianaris, kad

$$n - 1 = k_x \cap u - 1 = k_y \cap \ldots \cap t - 1 = k_z .$$

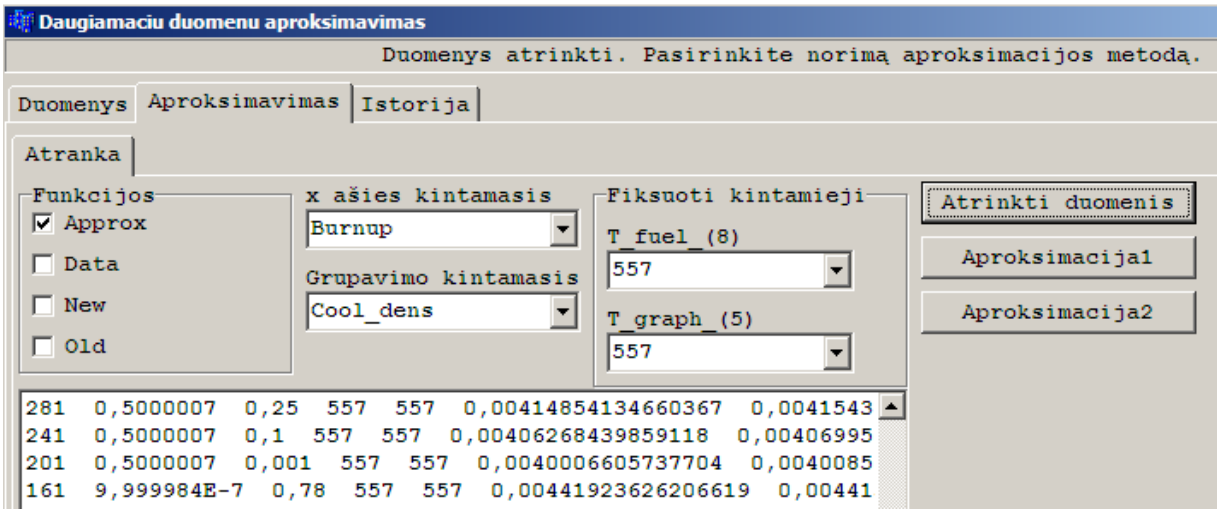
Tačiau mūsų darbo tikslas rasti efektyviausią daugianarį, t.y. tokį, kuris būtų kuo paprastesnis (turėtų kuo mažiau narių) ir tokį, kuris būtų kuo tikslesnis. Taigi iš tikrųjų reikia spręsti ne interpoliavimo, o suglodinimo uždavinį, parenkant maksimalius kintamųjų laipsnius tokius, kad

$$n < k_x \cap u < k_y \cap \ldots \cap t < k_z .$$

Nagrinėjant įvairias maksimalių kintamųjų kombinacijas, galima rasti tokį daugianarį, kurio pagalba duomenys aproksimuojami efektyviausiai.

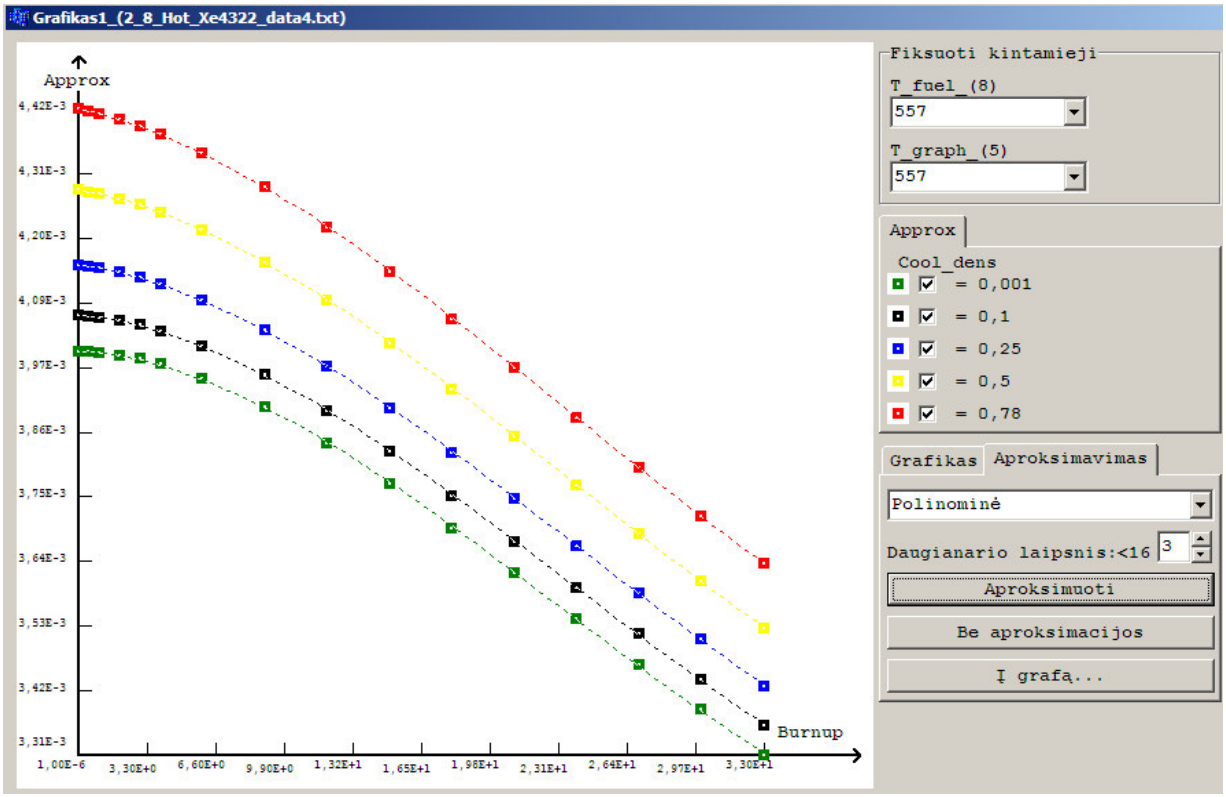
2.4. VARTOTOJO SĄSAJA

Vartotojo sąsaja sukurta naudojantis C++ Builder 6 enterprise programavimo įrankiu, C++ kalba. Darbo patirtis su šiuo įrankiu yra gana gera, todėl jis ir buvo pasirinktas. Vartotojo sąsaja susideda iš keleto langų. 2.3 paveiksle pavaizduotas pagrindinis langas. Patogumo dėlei jis yra suskirstytas į puslapius. Puslapyje „Duomenys“ galima atlikti įvairias operacijas su pradiniais duomenimis: įvesti, pakeisti, pašalinti nereikalingus vardus, atskirti funkcijas nuo kintamųjų. Sudarius ir patvirtinus funkcijų bei kintamųjų duomenų struktūrą, atverčiamas puslapis „Aproksimavimas“. Čia pasirinkus x ašies kintamąjį, grupavimo kintamąjį, funkcijos vardą bei fiksuotų kintamųjų reikšmes, paspaudus mygtuką „Atrinkti duomenis“, tikrinama, ar yra duomenų, tenkinančių sudarytą užklausą. Jei taip, galima pasirinkti vieną iš dviejų aproksimacijų metodų - „Aproksimacija1“ arba „Aproksimcija2“. Kadangi šis langas yra pagrindinis, jį uždarius, automatiškai užsidaro visi kiti langai (jei jie buvo atidaryti) ir sąsaja baigia darbą. Viršutinėje jo dalyje yra pranešimams skirtas komponentas, naudojamas įvairiems pranešimams išvesti. Pavyzdžiui, apie įvestą/neįvestą failą, kurį veiksmą toliau



2.3 pav. Vartotojo sąsajos pagrindinis langas

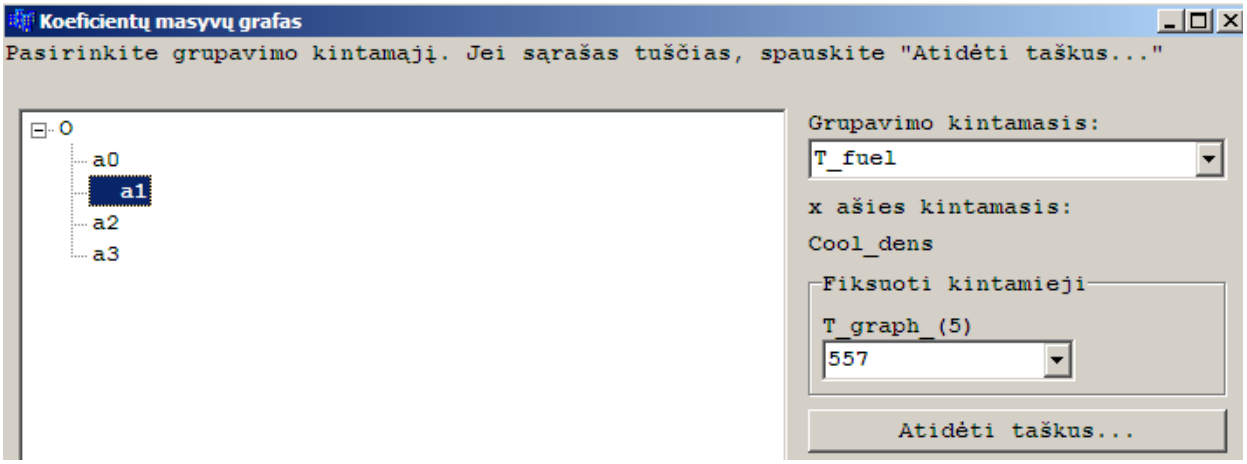
atlikti ir panašiai. Jis naudingas tada, kai vartotojas nėra gerai susipažinęs su įrankiu. Atvertus puslapį „Istorija“, galima atidaryti anksčiau išsaugotą grafiką, arba peržiūrėti glodinimo procedūrą, jei ji jau yra baigta. Paspaudus mygtuką „Aproksimacija1“, atsiranda suformuoto duomenų pjūvio langas, kuris pavaizduotas 2.4 paveiksle. Duomenų pjūvio lange pavaizduoti taškai, kurie atitiko



2.4 pav. Duomenų pjūvio langas.

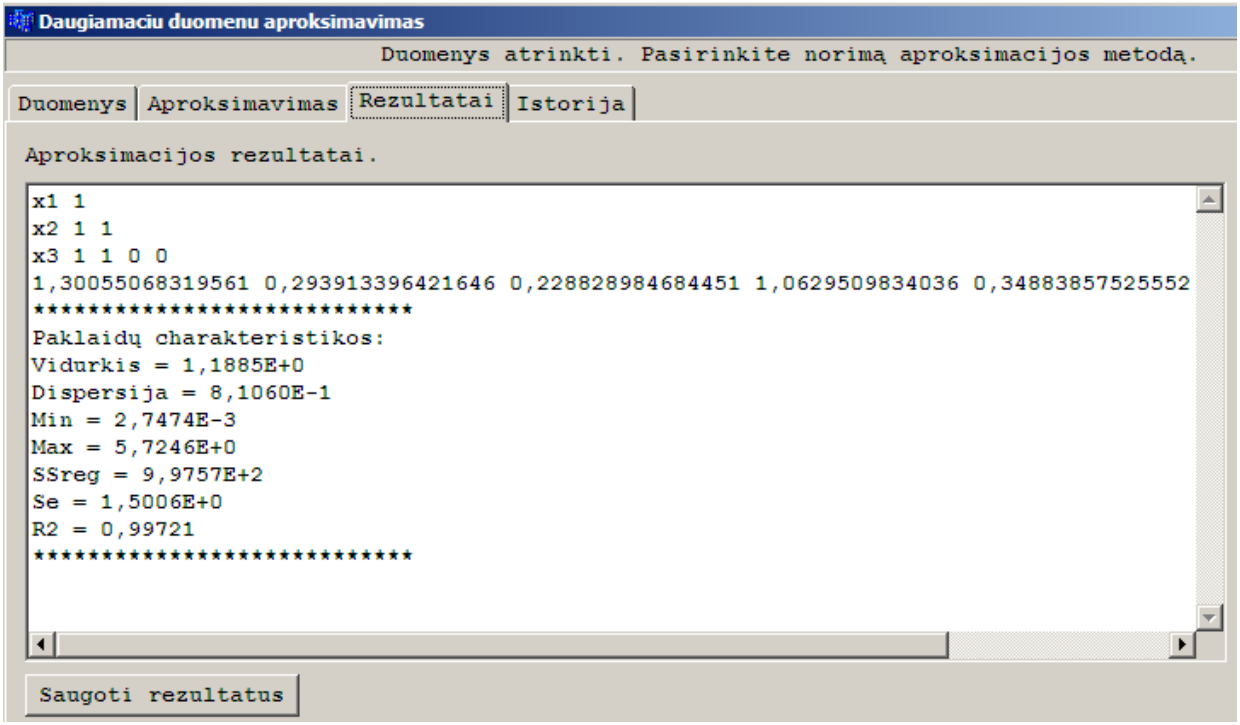
pagrindiniame lange sudarytą užklausą. Skirtingas grupavimo kintamojo reikšmes atitinka skirtingos taškų spalvos. Jei pagrindiniame lange parenkamos kelios funkcijos, tai kiekvienai funkcijai vaizduoti parenkami skirtingi taško simboliai. Šiame lange galima keisti fiksuotų kintamųjų reikšmes. Jas keičiant, automatiškai keičiasi taškų pozicijos. Kiekvienai parinktai funkcijai skirtas atskiras puslapis. Jis pavadinamas funkcijos vardu, o pačiame puslapyje atidedamos grupavimo kintamojo reikšmės su indikatorių komponentais bei legendomis. Nuėmus žymę prie atitinkamos grupavimo kintamojo reikšmės, pradingsta tą grupavimo kintamojo reikšmę atitinkantys taškai. Tai naudinga tada, kai atidėti taškai persidengia. Šiame lange yra dar vienas puslapių valdymo komponentas. Pirmame jo puslapyje „Grafikas“ yra trys mygtukai, skirti grafiko išsaugojimui, atidarymui bei uždarymui. Antrame puslapyje „Aproksimavimas“ yra išskleidžiamo sąrašo komponentas, skirtas aproksimacijos metodų sąrašui. Šiame sąraše pasirinkus įrašą „Polinominė“, sukuriamas redagavimo komponentas, skirtas daugianario laipsniui įvesti, bei mygtukas „Aproksimuoti“. Paspaudus minėtą mygtuką, per atidėtus taškus nubrėžiamos aproksimuojančios kreivės. Mygtukas „Be aproksimacijos“ skirtas panaikinti aproksimuojančioms kreivėms, o mygtukas „Į grafą“ skirtas sukurti dar vienam langui, kuris pavaizduotas 2.5 paveiksle. Lange pavaizduotame 2.4 paveiksle, keičiant fiksuotų kintamųjų reikšmes, keičiasi ir aproksimavimo kreivės. Keičiant minėtas reikšmes bei aproksimuojančio daugianario laipsnį, vartotojas gali parinkti jo nuomone optimaliausią aproksimuojančio daugianario laipsnį. Paspaudus mygtuką „Aproksimuoti“ iš tikrųjų aproksimuojami visi galimi duomenų pjūviai, o keičiant

fiksuotų kintamųjų reikšmes, aproksimuojančios kreivės tik perbraižomos ir jokie kiti skaičiavimai



2.5 pav. Koeficientų aibių grafas.

nebeatliekami. Lange, pavaizduotame 2.5 paveiksle, galima pasirinkti norimą toliau nagrinėti koeficientų aibių grafo mazgą, kuris vėliau bus laikomas kaip funkcija. Visi kiti šio lango komponentai bei jų funkcijos analogiški lango, pavaizduoto 2.3 paveiksle, komponentams bei funkcijoms. Paspaudus mygtuką „Atidėti taškus“, sukuriamas langas labai panašus į 2.4 paveiksle pavaizduotą langą. Skirtųsi tik parinktos funkcijos vardas - jos vietoje įrašomas parinkto grafo mazgo vardas, x ašies kintamuoju automatiškai tampa prieš tai buvęs grupavimo kintamasis, o fiksuotų kintamųjų sąrašas sutrumpėja vienu įrašu. Vėl galima pasirinkti norimo laipsnio aproksimuojantį daugianarį, keisti fiksuotų kintamųjų reikšmes. Pasirinkus optimaliausią vartotojo nuomone variantą, grafas papildomas naujais mazgais, t.y. koeficientų aibėmis. Jei pažymimas grafo mazgas, kuris jau turi šakas, vartotojui pateikiamas pranešimas: „Pasirinkote mazgą, kuris jau turi šakas. Ar tikrai norite trinti šakas?“. Tokiu atveju vartotojas gali ištrinti mazgą (tai naudinga, kai vartotojas mano, kad pasirinko blogą aproksimuojantį daugianarį) arba netrinti mazgo (tai naudinga, kai vartotojas netyčia pasirenka ne tą mazgą, kurį norėjo). Jei mazgas yra ištrinamas, sunaikinami visi žemiau esantys mazgai, t.y. koeficientų aibės. Kai grafas išnagrinėjamas iki galo, apie tai informuojamas vartotojas ir gauti koeficientai tam tikra struktūra (ją aptarsime vėliau) atspausdinami pagrindinės formos puslapyje „Rezultatai“. Tai pavaizduota 2.6 paveiksle. Pirmose eilutėse pateikiama sudaryto grafo struktūra. Pirma eilutė reiškia, kad pirmame žingsnyje x ašies kintamasis buvo $x1$, o sudaryti pjūviai aproksimuoti pirmo laipsnio daugianariu, tarkime $a_0 + a_1 x1$. Antra eilutė reiškia, kad antrame žingsnyje x ašies kintamasis buvo $x2$, koeficientai a_0 buvo aproksimuojami pirmo laipsnio daugianariu, tarkime $b_{00} + b_{01} x2$, koeficientai a_1 irgi pirmo laipsnio daugianariu , tarkime $b_{10} + b_{11} x2$. Analogiškai su trečia eilute. Ketvirtoje eilutėje atspausdinamos koeficientų reikšmės. Dar žemiau



2.6 pav. Pateikiamas rezultatas.

pateikiamos paklaidų charakteristikos bei aproksimacijos kokybės kriterijai. Paspaudus mygtuką „Saugoti rezultatus“, visa lange pateikta informacija išsaugoma pageidaujamame tekstiniame faile.

2.5. REIKALAVIMAI VARTOTOJO SĄSAJAI

Norint paleisti vartotojo sąsają kompiuteryje reikia suinstaliuoti programinę įrangą *C++ Builder 6 enterprise*. Rekomenduotina, kad kompiuterio skiriamoji geba būtų ne mažesnė negu 1024x768 taškai. Mažesnės gebos kompiuteriuose bus nepatogu dirbti, nes pavyzdžiui lyginant du grafikus, ekrane atsiranda du langai, todėl reikalinga didesnė erdvė.

Griežti reikalavimai keliami duomenų failams. Juose neturi būti tuščių eilučių. Duomenų failo pirmoje eilutėje turi būti surašyti pirmiausia visų kintamųjų vardai, atskirti bent vienu tarpų, po to visų funkcijų vardai, taip pat atskirti bent vienu tarpu. Kintamųjų ir funkcijų vardai turi būti žodžio struktūros, t.y. juose negali būti tarpų, ar kitų skiriamųjų ženklų. Nepatartina naudoti vardų ilgesnių negu penkiolika simbolių, nes kiti simboliai programos metu bus nematomi. Kitose duomenų failo eilutėse turi būti surašytos kintamųjų ir funkcijų reikšmės, tokia tvarka, kokia surašyti kintamųjų ir funkcijų vardai. Vienoje eilutėje gali būti tik po vieną kiekvieno kintamojo ir kiekvienos funkcijos reikšmę. Kiekvieno kintamojo ir kiekvienos funkcijos reikšmių skaičius turi būti vienodas. Jei faile yra kažkokie papildomi duomenų stulpeliai, programos metu jie galės būti pašalinti. Duomenų failo pavyzdys:

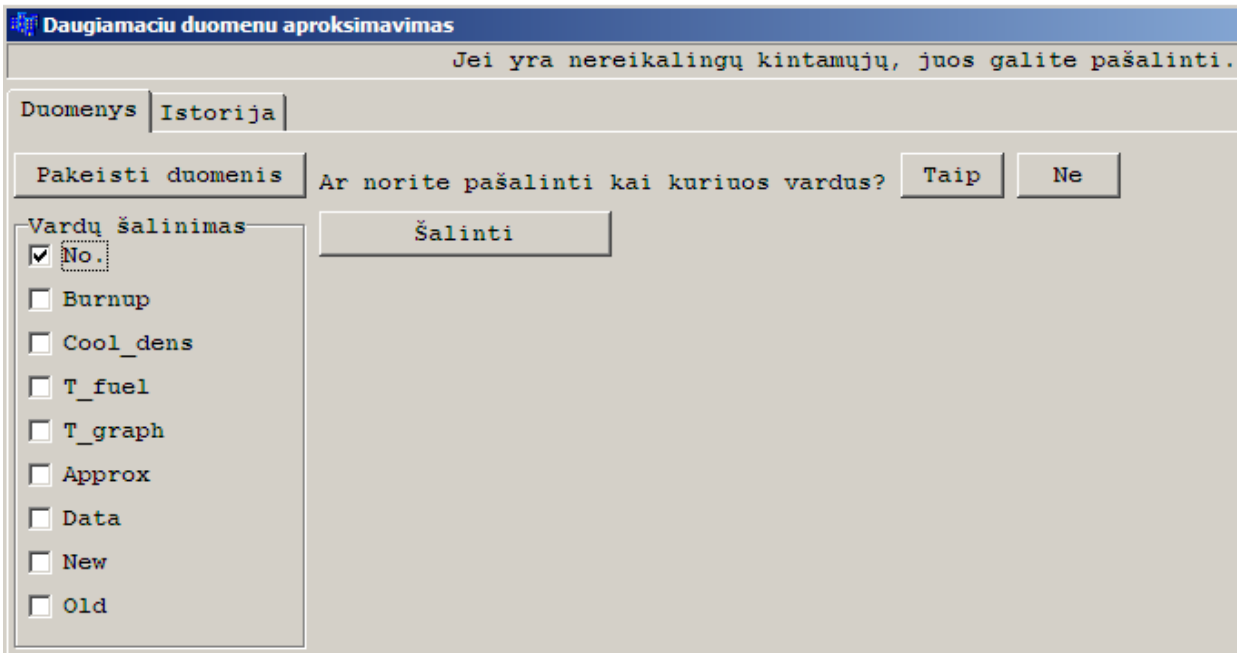
x1	x2	x3	funkcija
1	0	2.145	.8310624319189982
2	0	2.145	1.5753609531622907
2	0	7.55	2.8914364667446675
2	0	6.75	2.7425306734535124
2	0	5	2.527011697313694
2	0	8.333	3.0813843374308223
2	0	1.45	1.0839606264237893
2	0	4.001	2.189943775765926
2	1.55	2.145	6.835916556946847
2	1.55	7.55	17.351437780716434
2	1.55	6.75	15.387422966732274
2	1.55	5	12.272247831499396
2	1.55	8.333	19.03115039422907
2	1.55	1.45	5.124100495046722
2	1.55	4.001	10.512350347035094
2	2.66	2.145	10.392264815983737
2	2.66	7.55	26.864992081426603
2	2.66	6.75	25.038564196404295
2	2.66	5	18.061504585982448
2	2.66	8.333	27.58686487118742
...	...	...	... ...

Paleidus vartotojo sąsają, prašoma įvestų vardų sąraše pažymėti pirmąją iš funkcijų. Pažymėtas vardas ir žemiau esantys vardai bus traktuojami kaip funkcijos, o aukščiau esantys vardai – kaip kintamieji.

Grafikai turi būti saugomi faile su plėtiniu **.graf*. Jei grafikas išsaugomas be plėtinio arba su kitokiu plėtiniu, jis negalės būti atidarytas.

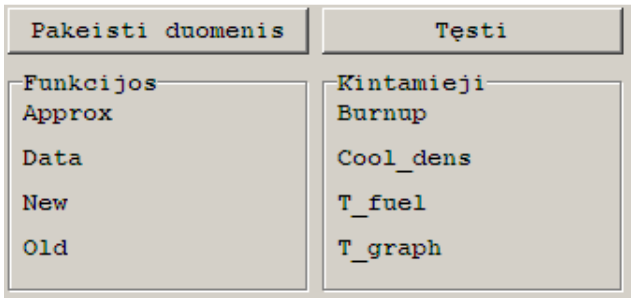
2.6. TAIKYMO PAVYZDYS

Pabandykime išspręsti daugiamačio aproksimavimo uždavinį, t.y. gauti aproksimuojančio daugianario koeficientus, naudodami vartotojo sąsają, aprašytą 2.4 skyriuje. Nagrinėsime failą *2_8_Hot_Xe4322_data4.txt*. Ten pateikti devyni duomenų stulpeliai. Pirmame iš jų yra duomenų eilučių numeriai, kiti keturi stulpeliai yra kintamųjų reikšmės, o likę keturi – funkcijų reikšmės. Paleidžiame programą ir įvedame duomenis, paspausdami mygtuką „Įvesti duomenis“. Įrankis pateikia iš failo nuskaitytų vardų sąrašą ir pasiūlo galimybę kai kuriuos iš jų pašalinti. Duomenų failo



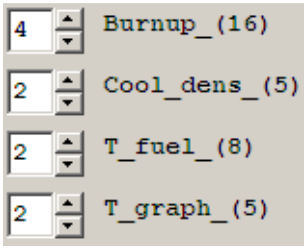
2.7 pav. Vardų šalinimas

pirmame stulpelyje yra surašyti eilučių numeriai, kurie tolimesnei analizei yra nereikalingi. Todėl paspausdami mygtuką „Taip“ patvirtiname, kad norime pašalinti kai kuriuos vardus. Tuomet programos langas turėtų atrodyti taip, kaip parodyta 2.7 paveiksle. Pažymime vardą „No.“ Ir spaudžiame mygtuką „Šalinti“. Atsiradusiame vardų sąraše pasirenkame vardą „Approx“, t.y. pirmąjį funkcijos vardą. Įrankis aukščiau sąraše esančius vardus traktuos kaip kintamųjų vardus, o pažymėtąjį vardą bei žemiau esančius vardus laikys funkcijų vardais. Po šio veiksmo, sąsaja vartotojui pateikia sudarytą kintamųjų vardų bei funkcijų vardų struktūrą (pav. 2.8). Jeigu tokia vardų struktūra vartotojo netenkina (taip gali atsitikti, jei vardų



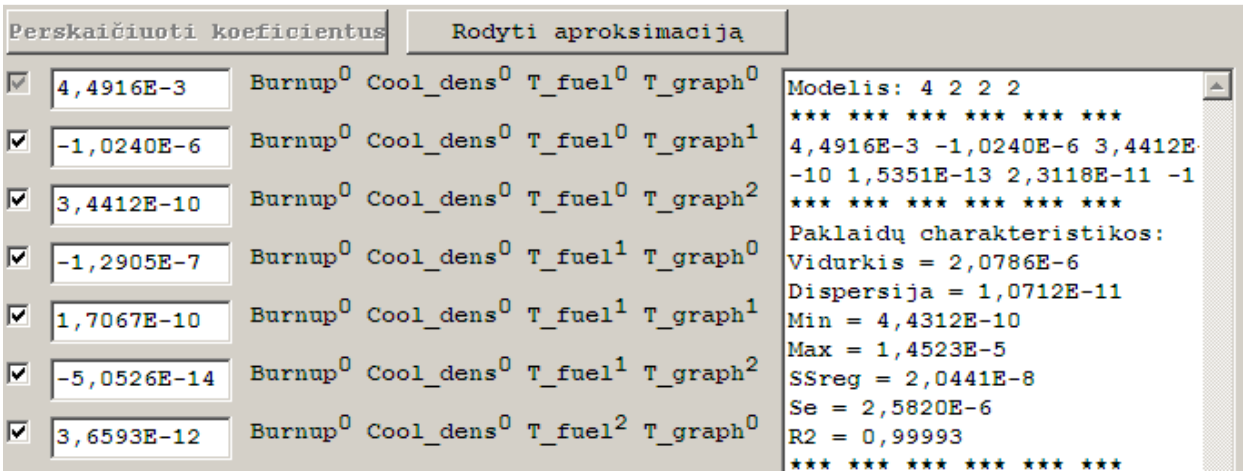
2.8 pav. Kintamųjų bei funkcijų vardai

sąraše pažymimas ne pirmosios funkcijos vardas), jis gali pakartoti duomenų įvedimo procedūrą, paspausdamas mygtuką „Pakeisti duomenis“. Kitu atveju, kai tokia vardų struktūra vartotoją tenkina, jis tai turi patvirtinti paspausdamas mygtuką „Tęsti“. Atidaromas puslapis „Atranka“. Pasirinkus funkcijos vardą „*Approx*“, *x* ašies kintamojo vardą „*Burnup*“, grupavimo kintamąjį „*Cool_dens*“ ir paspaudus mygtuką „Atrinkti duomenis“ sąsajos langas turėtų atrodyti taip, kaip pavaizduota 2.3 paveiksle. Pirmiausia aproksimuojantį daugianarį pabandykime gauti polinominės aproksimacijos metodu, aprašytu skyrelyje 2.3. t.y., spaudžiame mygtuką „Aproksimacija2“. Atsivertusiame puslapyje pateikiamas kintamųjų sąrašas, su šalia esančiais įvedimo langeliais, maksimaliems kintamųjų laipsniams parinkti (2.9 pav.). Kintamųjų vardų gale, lenktiniuose skliaustuose, pateikiami kintamųjų skirtingų įgyjamų reikšmių skaičiai. Pavyzdžiui, kintamasis „Burnup“ įgyja 16 skirtingų reikšmių, vadinasi jo maksimalus laipsnis daugianaryje gali būti 15. Parinkus maksimalius kintamųjų laipsnius, procesą galima tęsti dvejopai.



2.9 pav. Maksimalūs kintamųjų laipsniai

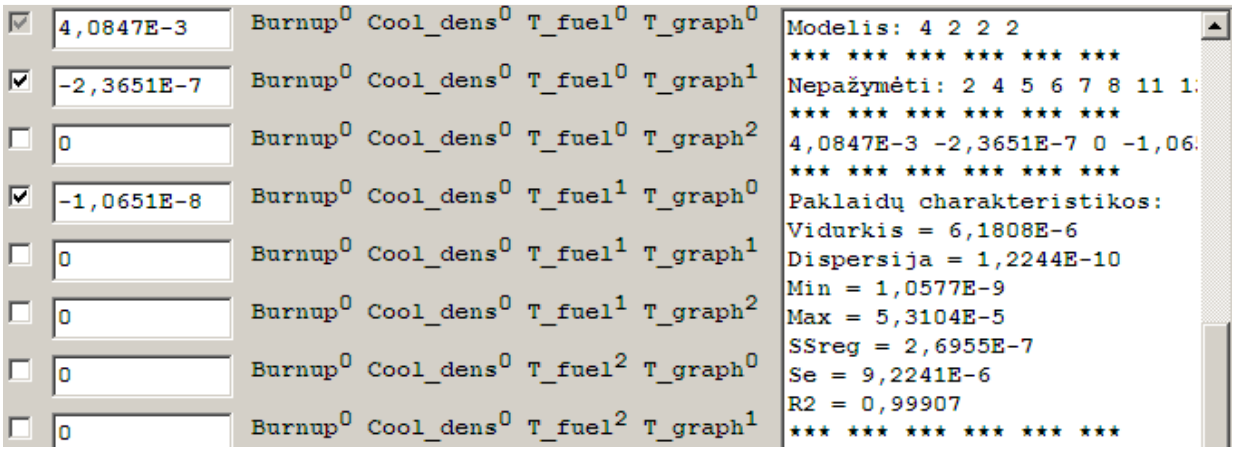
Pirmiausia nagrinėkime vieną daugianarį - paspauskime mygtuką „Atskiras atvejis“. Atidaromas langas su sugeneruotomis visomis daugianario kintamųjų sandaugų kombinacijomis bei su laukeliais, daugianario koeficientams pateikti. Paskutinės kintamųjų sandaugų kombinacijos laipsniai bus tokie, kokie buvo parinkti anksčiau (2.9 pav.). Paspaudus mygtuką „Apskaičiuoti koeficientus“, minėti laukeliai užpildomi daugianario koeficientais. Tai gali užimti šiek tiek laiko, priklausomai nuo parinktų maksimalių kintamųjų laipsnių didumo. Koeficientai, kintamųjų sandaugų kombinacijos, bei aproksimacijos tikslumo kriterijai pavaizduoti 2.10 paveiksle. Kaip matyti,



2.10 pav. Kintamųjų sandaugos bei koeficientai

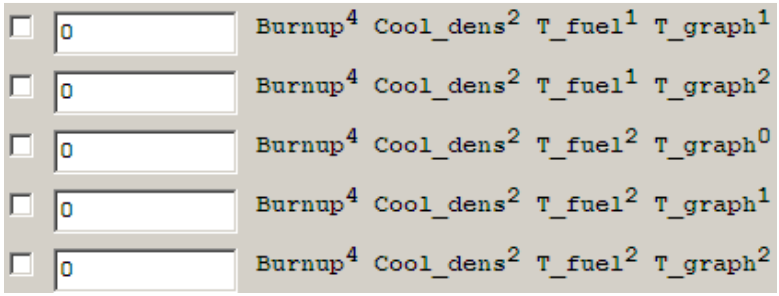
apibrėžtumo koeficientas yra labai artimas vienetui. Vadinasi, sumodeliuoto daugianario reikšmės yra labai artimos nagrinėjamos funkcijos reikšmėms. Taip pat 2.10 paveiksle matyti, kad kai kurios

kintamųjų reikšmės yra labai mažos, t.y. 10^{-12} ar net 10^{-14} eilės. Sąsajos pagalba galima norimus koeficientus pašalinti iš daugianario. Tiesiog reikia nuimti „varneles“ nuo šalia koeficientų esančių indikatorių ir paspausti mygtuką „Perskaičiuoti koeficientus“. Pašalinkime visus koeficientus, kurių eilė yra mažesnė ar lygi 10^{-10} ir perskaičiuokime koeficientus. Rezultatas pateiktas 2.11 paveiksle. Iš

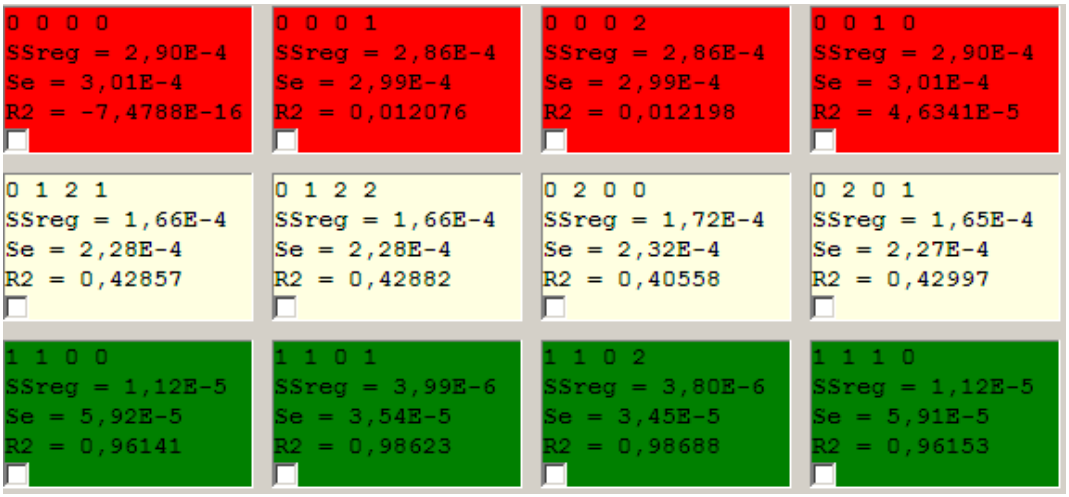


2.11 pav. Suprastinta daugianario struktūra

viso daugianarį sudarė 135 nariai, nes $(4+1)\cdot(2+1)\cdot(2+1)\cdot(2+1)=135$. Koeficientų, kurių eilė mažesnė ar lygi 10^{-10}, buvo 102. Palyginkime aproksimacijos tikslumo kriterijus (pav. 2.10 ir pav. 2.11). Matome, kad jie tik nežymiai pablogėjo. Tačiau, daugianario struktūrą suprastinome 4 kartus (dabar daugianarį sudaro tik 33 nariai). Tęsdami procesą, t.y., atmetinėdami mažiausius koeficientus, galime dar žymiai suprastinti daugianarį. Pavyzdžiui šiuo atveju, gautume dar pakankamai tikslius rezultatus, jei daugianaryje paliktume tik 15 narių. 2.12 paveiksle pavaizduoti koeficientai šalia kintamųjų sandaugų su aukščiausiais laipsniais. Matyti, kad jie visi buvo atmesti. Vadinasi, buvo neverta pasirinkti tokius aukštus maksimalius kintamųjų laipsnius. Būtų labai patogu, jei parinkus kintamųjų maksimalius laipsnius, iš karto būtų įmanoma išanalizuoti ne tik vieną daugianarį (išnagrinėtą aukščiau), tačiau ir visus kitus, žemesnės eilės modelius. Tokia galimybė taip pat yra realizuota. Tiesiog pasirinkus maksimalius kintamųjų laipsnius reikia paspausti mygtuką „Visų atvejų analizė“. Skaičiavimai taip pat gali šiek tiek užtrukti, priklausomai nuo parinktų laipsnių didumo. 2.13 paveiksle pavaizduota visų modelių žemėlapio dalis. Visą žemėlapį galima rasti priede. Paaiškinsime modelio langelio struktūrą. Pirmoje eilutėje pateikiami



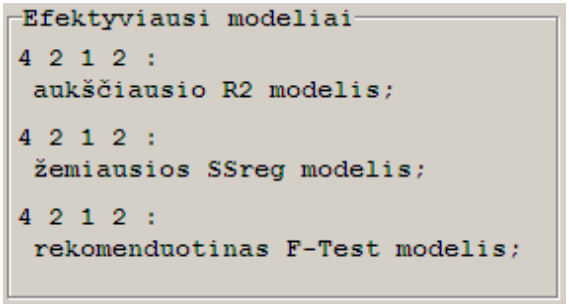
2.12 pav. Paskutiniai daugianario koeficientai



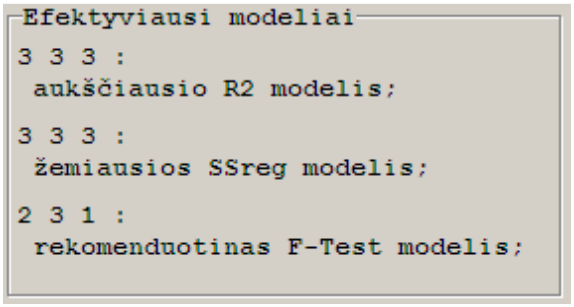
2.13 pav. Modelių žemėlapis

daugianario maksimalūs kintamųjų laipsniai. Kitose trijose eilutėse – aproksimacijos tikslumo koeficientai. Modelio langelis būna raudonas, kai modelio apibrėžtumo koeficientas neviršija 0.33. Kitaip tariant raudona spalva pažymėti tie daugianarių modeliai, kurie prasčiausiai aproksimuoja duomenų tinklelio taškus. Modelio langelis būna žalias, jei apibrėžtumo koeficientas būna didesnis už nustatytą reikšmę, o kvadratinių nuokrypių suma būna mažesnė už nustatytą reikšmę. Kitaip tariant taip žymimi modeliai, kurie tiksliausiai aproksimuoja duomenis. Minėtus kokybės kriterijų slenksčius gali nustatyti vartotojas. Apibrėžtumo koeficiento slenksčio kitimo intervalas nuo 0 iki 1. Kvadratinių nuokrypių suma kinta nuo iš visų modelių išrinktos minimalios reikšmės iki iš visų modelių išrinktos maksimalios reikšmės. Pradinė apibrėžtumo koeficiento slenksčio reikšmė 0.66, o kvadratinių nuokrypių sumos pradinis slenkstis lygus jos kitimo diapazono trečdaliui. Daugianario modelio langelis būna baltas visais kitais atvejais, t.y. tuomet, kai apibrėžtumo koeficientas būna didesnis už 0.33, tačiau modelis netenkina bent vieno nustatyto kokybės kriterijų slenksčio. Kiekviename daugianario modelio langelyje, kairiajame apatiniame kampe, yra po indikatoriaus komponentą. Norint patikrinti, kuris iš dviejų modelių yra statistiškai tinkamesnis, tereikia pažymėti tų modelių indikatorius ir paspausti mygtuką „F-test“. 1.3.2 skyriuje aprašyta, kokiu būdu nustatoma, kuris modelis yra tinkamesnis. Koeficientų žemėlapis yra labai informatyvus, kadangi vienu metu tarpusavyje galime palyginti labai daug modelių. Šiuo atveju galime pastebėti, kad pakankamai tikslūs daugianarių modeliai gaunami jau tada, kai maksimalūs kintamųjų laipsniai tėra vienetai. F kriterijaus pagalba galima nustatyti, ar aproksimacijos tikslumas vertas papildomų kintamųjų įvedimo. Kitaip tariant, ar apsimoka pasirinkti sudėtingesnį modelį, norint gauti tikslesnį rezultatą. Įrankis taip pat vartotojui pasiūlo pačius tiksliausius modelius, t.y. modelį, kurio aukščiausias apibrėžtumo koeficientas ir modelį, kurio žemiausia kvadratinių nuokrypių suma. Taip pat yra galimybė išrinkti patį efektyviausią modelį naudojant F kriterijų, tereikia paspausti mygtuką „Visų modelių F-test“. Sąsaja

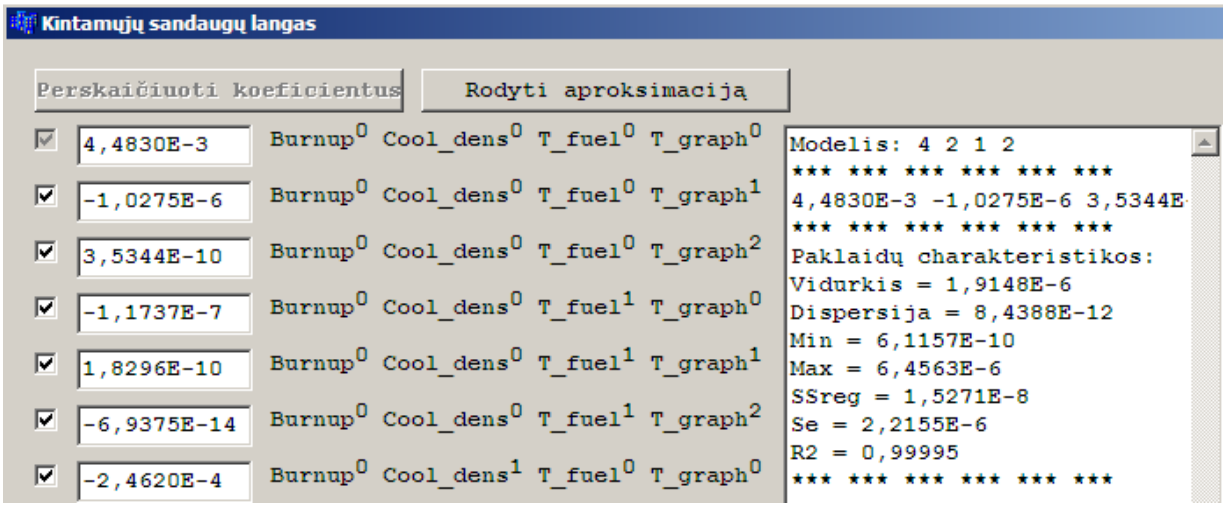
tarpusvyje palygins visus modelius ir pateiks vartotojui atsakymą. 2.14 paveiksle matyti, kokie modeliai šiuo konkrečiu atveju yra naudingiausi. Šį kartą vienas ir tas pats daugianario modelis turi aukščiausią apibrėžtumo koeficientą, žemiausią nuokrypių kvadratų sumą bei yra statistiškai tinkamiausias. Paprastai pagal šiuos kriterijus atrenkami skirtingi modeliai. Pavyzdžiui, 2.15 paveiksle pateikti kito uždavinio efektyviausi modeliai. Matome, kad rekomenduotinas F kriterijaus modelis nesutampa su aukščiausio apibrėžtumo koeficiento bei žemiausios kvadratinių nuokrypių sumos modeliais. Vadinasi, vertinant modelio tikslumą ir efektyvumą nepakanka naudoti vieno kriterijaus. Atlikus visų modelių analizę ir vartotojui išsirinkus jo nuomone tinkamiausią daugianarį, jo struktūrą galima suprastinti anksčiau aprašytu būdu, atskiro atvejo analizės metu. Taigi, pasirinkime modelį „4 2 1 2“ ir apskaičiuokime jo koeficientus. Rezultatai pateikti 2.16 paveiksle.



2.14 pav. Efektyviausi modeliai 1



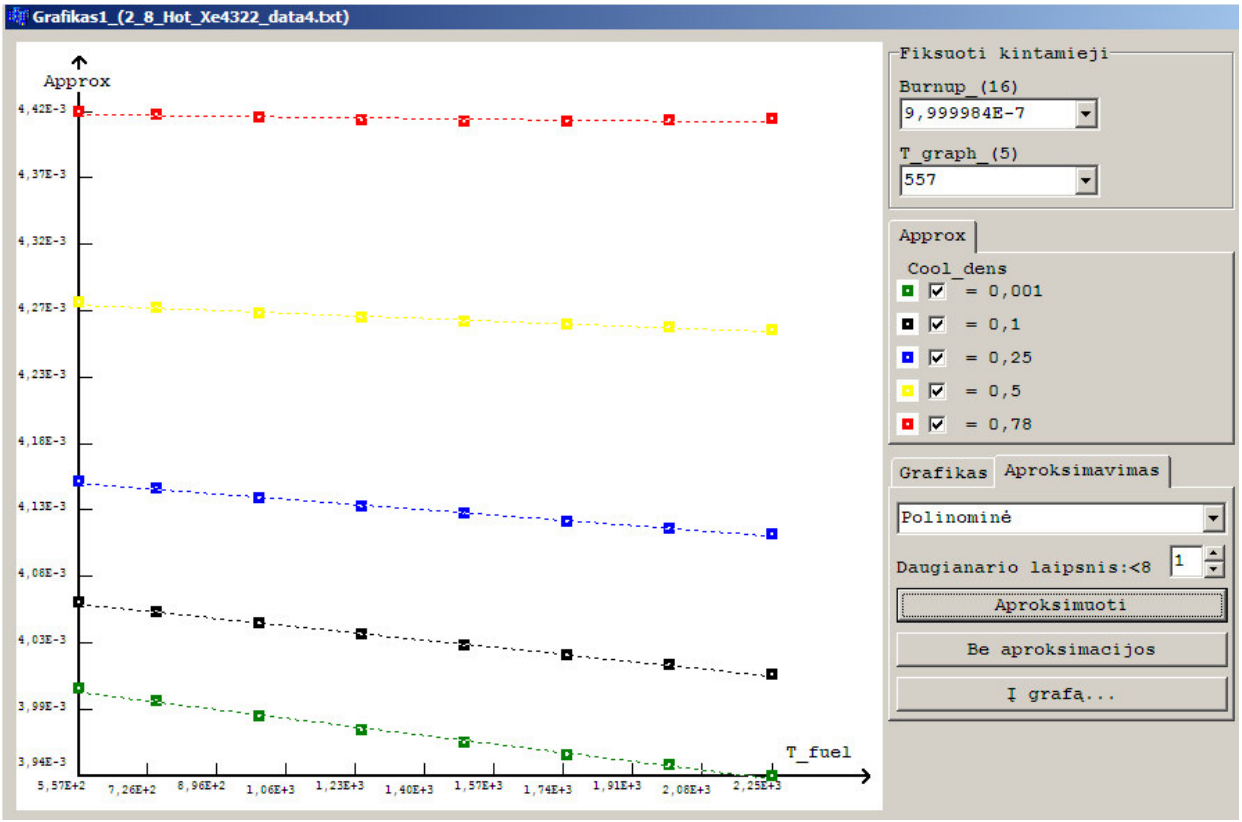
2.15 pav. Efektyviausi modeliai 2



2.16 pav. Antro aproksimacijos metodo rezultatai

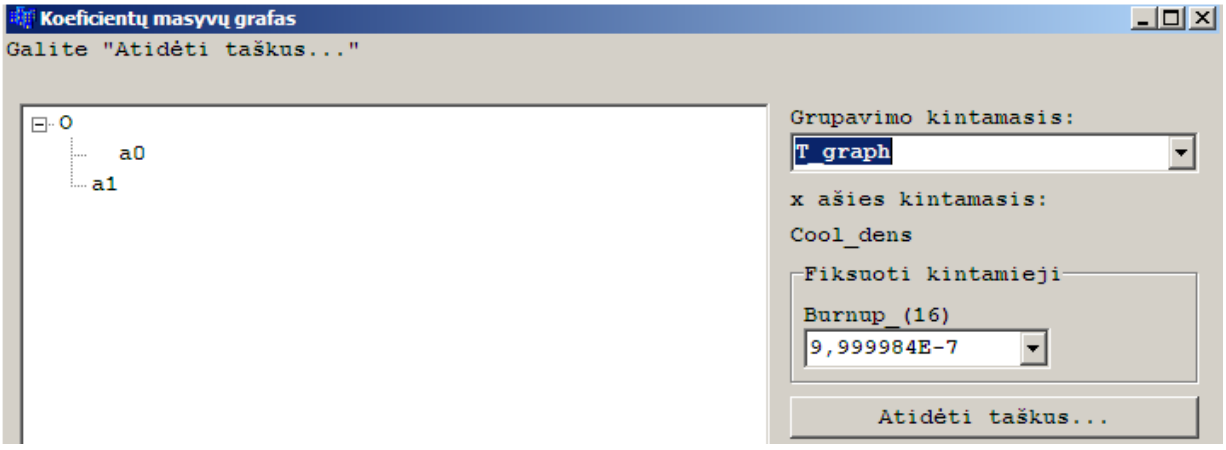
Pabandykime tą patį uždavinį išspręsti pirmuoju aproksimacijos metodu, aprašytu skyrelyje 2.2. Instrukcija kaip naudotis vartotojo sąsaja sprendžiant aproksimavimo uždavinį šiuo metodu yra pateikta šaltinyje [1]. Dabar galime paaiškinti, kodėl analizę pradėjome nuo atrojo metodo. Turėdami maksimalius kintamųjų laipsnius, iš anksto žinosime, kokio laipsnio daugianariu aproksimuoti duomenų pjūvį. Taigi, atliekame anksčiau aprašytus duomenų redagavimo veiksmus taip, kad gautume

2.3 paveiksle pavaizduotą langą, tik *x* ašies kintamojo vardą pasirinkime ne „*Burnup*“, o „*T_fuel*“ ir grupavimo kintamojo vardą pakeiskime į „*Cool_dens*“. Paspauskime mygtuką „Atrinkti duomenis“ ir po to „Aproksimcija1“. Gautus duomenų pjūvius aproksimuokime pirmo laipsnio daugianariais. Rezultatas pateiktas 2.17 paveiksle. Spaudžiame mygtuką „Į grafą“ – pradėjome formuoti koeficientų



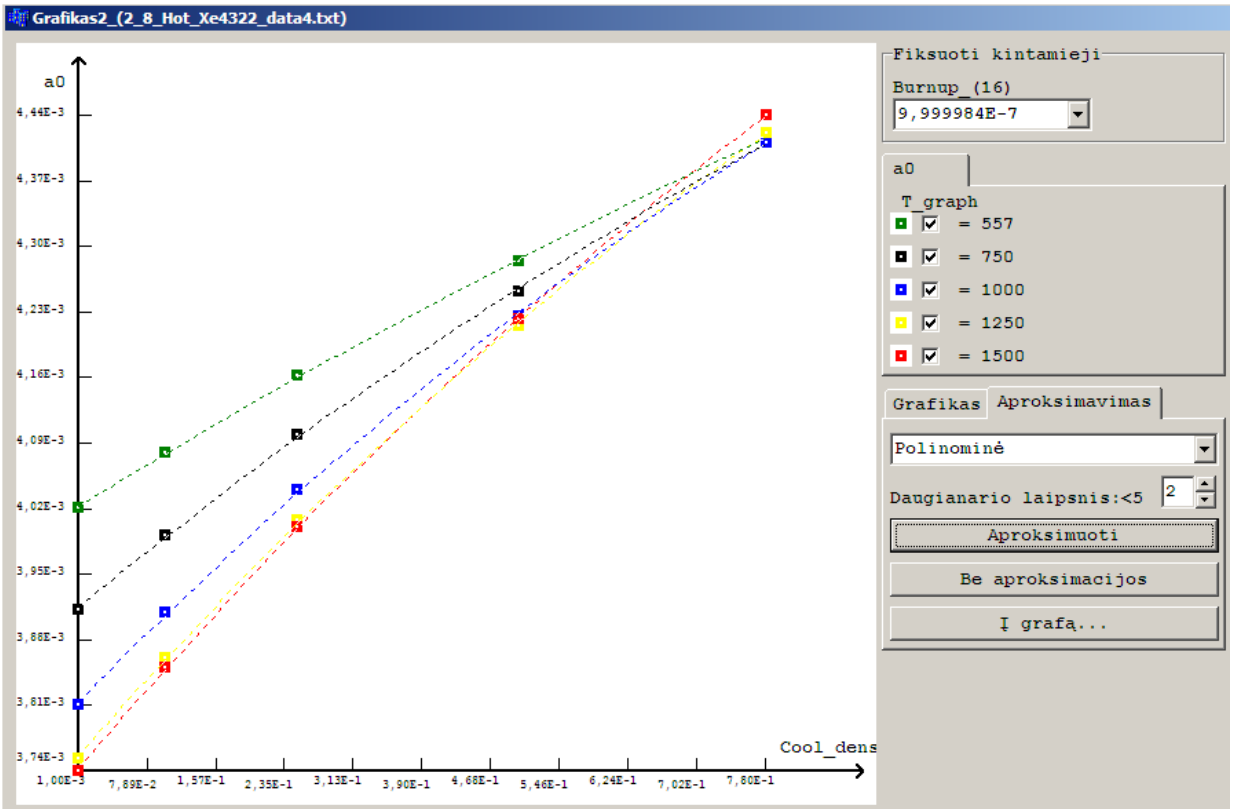
2.17 pav. Funkcijos priklausomybės nuo kintamojo *T_fuel*

aibių grafą. Medyje pasirinkime koeficientą *a0*, grupavimo kintamąjį *T_graph*. Viskas turėtų atrodyti taip, kaip pavaizduota 2.18 paveiksle. Atidėkime taškus – gausime koeficiento *a0* priklausomybes nuo



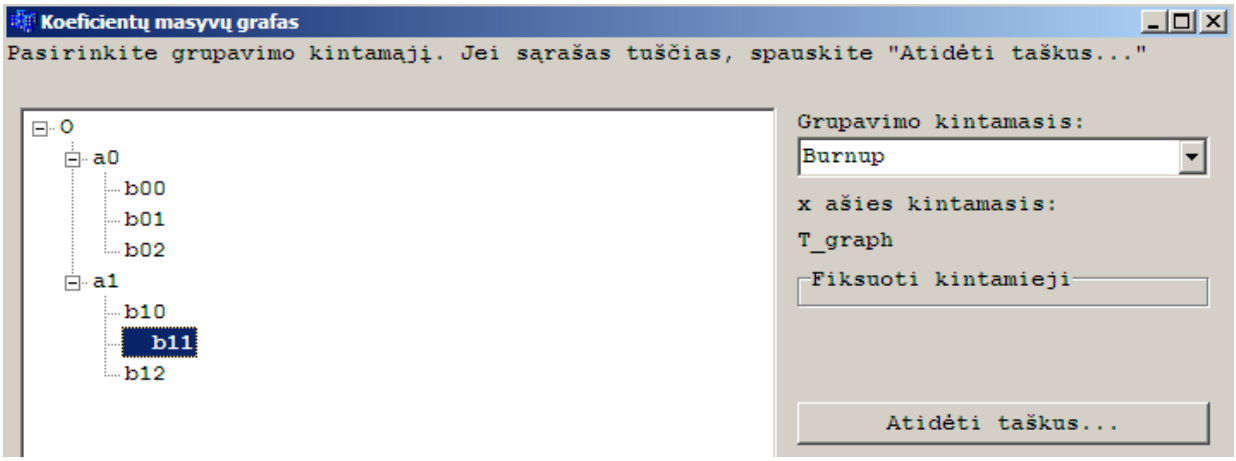
2.18 pav. Pradinis koeficientų aibių grafas

kintamojo *T_fuel*. Gautus duomenų pjūvius, kaip jau žinome, aproksimuokime antro laipsnio daugianariais. Rezultatas pateiktas 2.19 paveiksle.



2.19 pav. Koeficientų *a0* priklausomybės nuo kintamojo *Cool_dens*

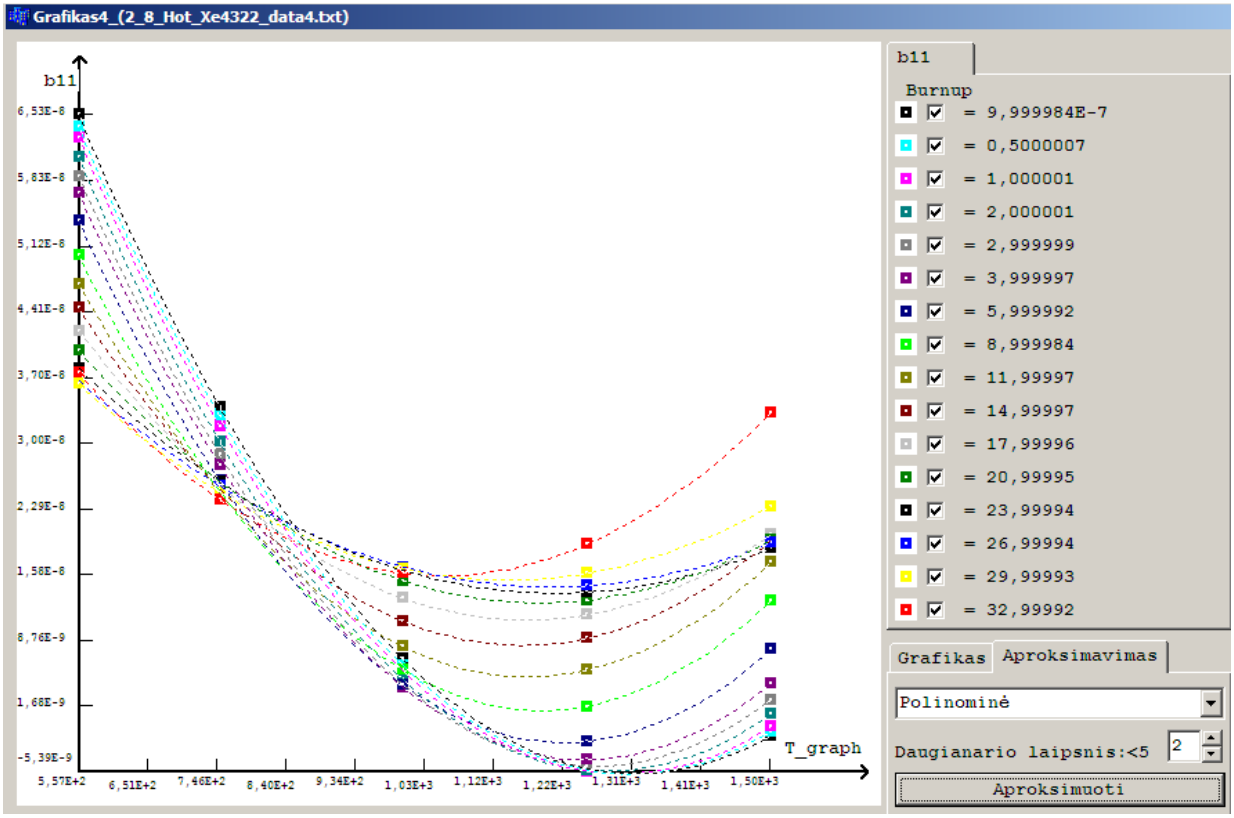
Koeficientų medyje pasirinkę *a1*, atliekame analogiškus veiksmus kaip ir pasirinkę koeficientą *a0*. Kitame lygyje, aproksimuosime koeficientus *b***. Koeficientų medyje pasirinkime aibę *b11*, grupavimo kintamasis lieka *Burnup*. Programos sąsaja šiame žingsnyje pavaizduota 2.20 paveiksle.



2.20 pav. Papildytas koeficientų aibių grafas 1

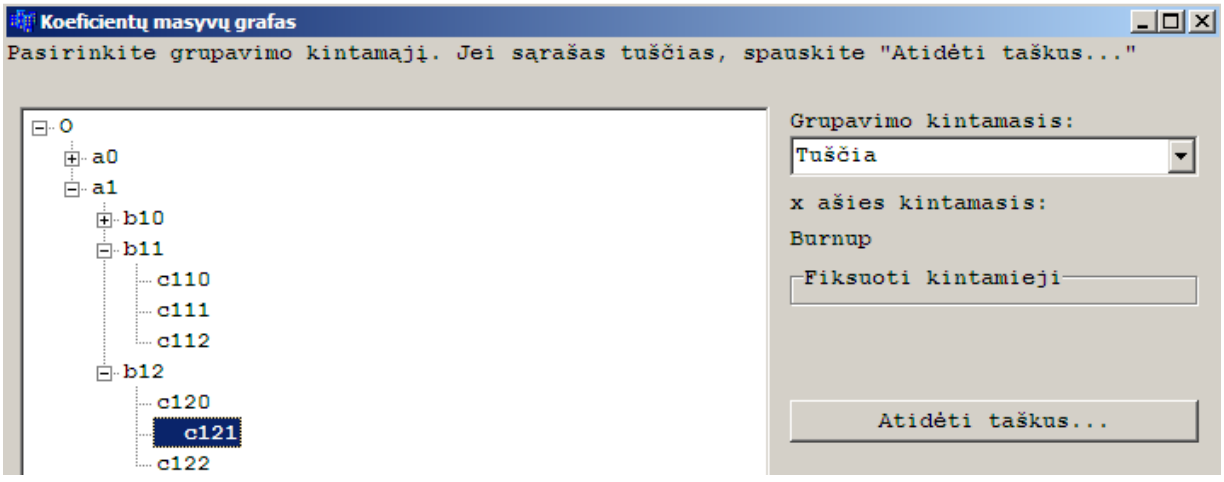
Paspaudę mygtuką „Atidėti taškus...“ gausime koeficientų *b11* priklausomybes nuo kintamojo *T_graph*. Taškus aproksimuojame vėlgi antro laipsnio daugianariu, kaip tai atrodo pavaizduota 2.21

paveiksle. Tą pačią procedūrą pakartojame su visais *b*** koeficientais. Kitame lygyje pasirinkime aibę



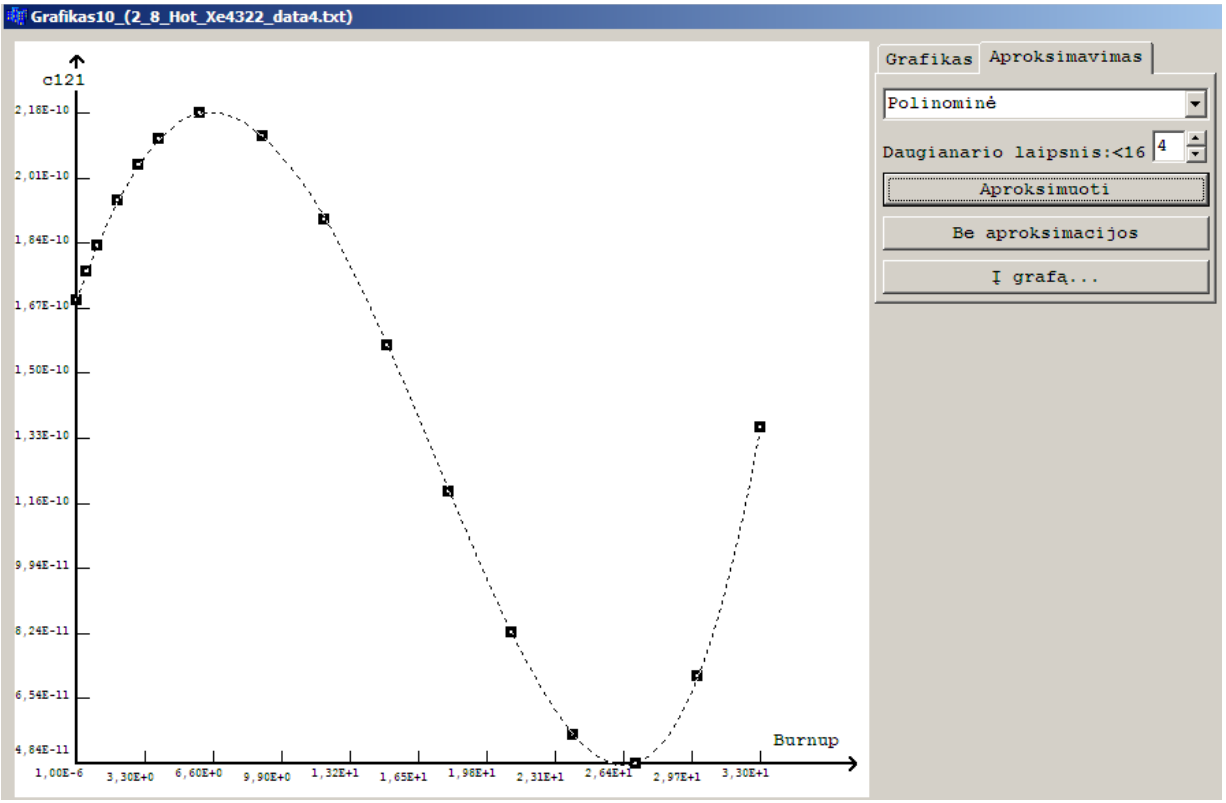
2.21 pav. Koeficientų *b11* priklausomybė nuo kintamojo *T_graph*

c121. Grupavimo kintamojo jau nebelieka. Kaip turėtų atrodyti koeficientų aibių grafas parodyta 2.22 paveiksle. Atidedame taškus – gausime koeficientų *c121* priklausomybes nuo kintamojo *Burnup*. Šį



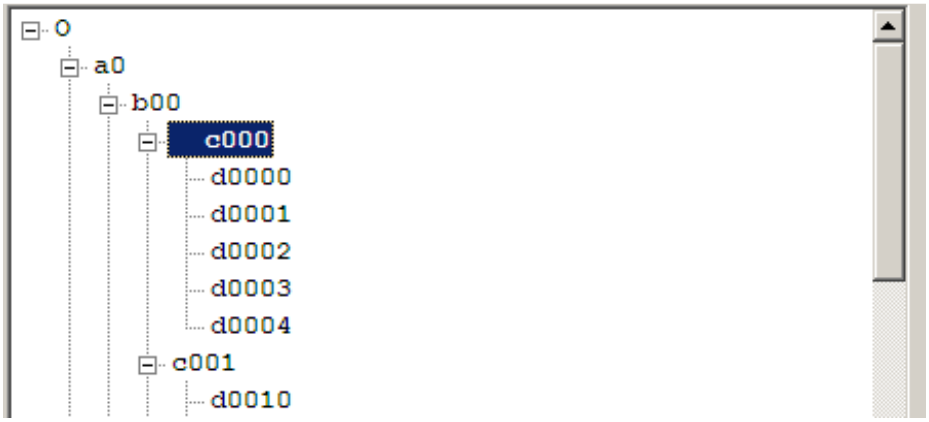
2.22 pav. Papildytas koeficientų aibių grafas 2

kartą gautus duomenų pjūvius aproksimuojame ketvirto laipsnio daugianariais. Rezultatas pateiktas 2.23 paveiksle. Matome kaip tiksliai ketvirto laipsnio daugianaris aproksimuoja taškus. Tokio



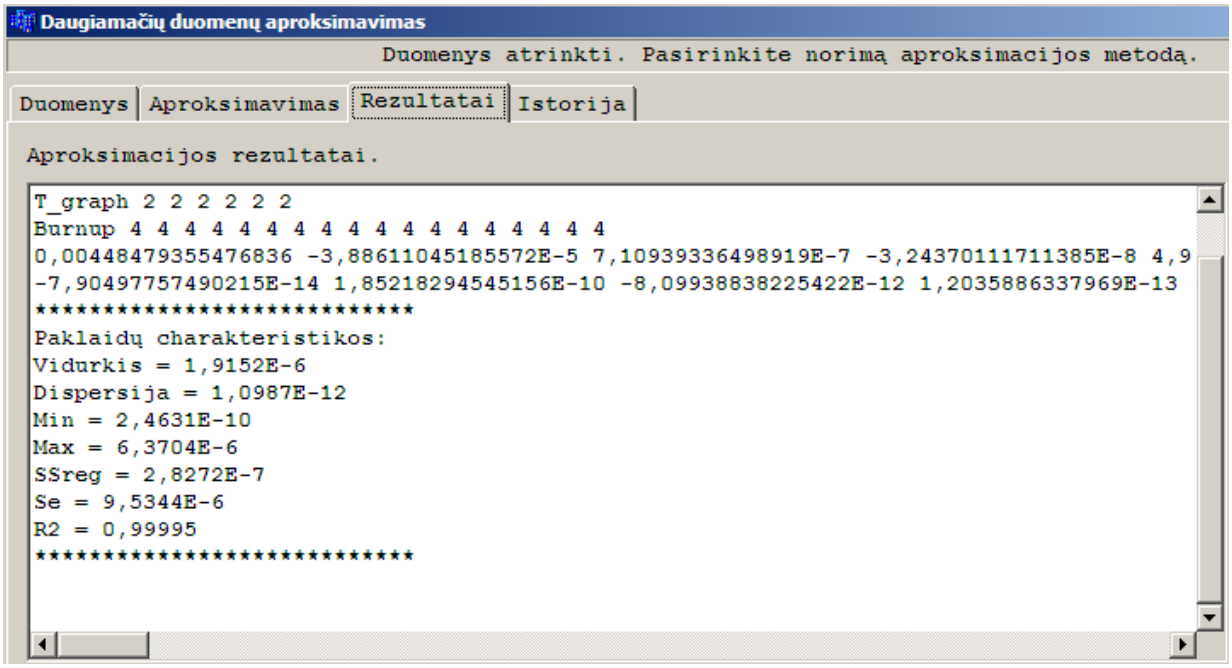
2.23 pav. Koeficientų *c121* priklausomybė nuo kintamojo *Burnup*

tikslumo nepasiektume, jei aproksimuotume 3 laipsnio daugianariu. O jei pasirinktume penkto laipsnio daugianarį, tai aproksimacijos kokybės praktiškai nepagerintume. Likusius koeficientus *c**** taip pat aproksimuojame ketvirto laipsnio daugianariais. Galiausiai suformuosime koeficientų aibių grafą, kurio fragmentas pavaizduotas 2.24 paveiksle. Baigus nagrinėti visus koeficientus, sąsaja atverčia



2.24 pav. Sudaryto koeficientų aibių grafo dalis

rezultatų puslapį, kuriame, kaip ir antro metodo atveju, pateikiami aproksimacijos kokybės kriterijai, kurie leidžia įvertinti aproksimacijos kokybę. Pirmo metodo rezultatai pateikti 2.25 paveiksle.



2.25 pav. Pirmo aproksimacijos metodo rezultatai

Palyginkime rezultatus, gautus pirmu aproksimacijos metodu, su rezultatais, gautais antru aproksimacijos metodu. Matome, kad paklaidų vidurkiai, dispersijos, minimalios bei maksimalios paklaidų reikšmės yra labai panašios. Šiek tiek didesnė nuokrypių kvadratų suma, gauta antruoju metodu, o apibrėžtumo koeficientai lygūs. Vadinasi, abu aprašyti aproksimacijos metodai yra lygiaverčiai.

IŠVADOS

1. Taigi, sukurtos sistemos pagalba išsiaiškinome, kad abu aproksimacijos metodai, aprašyti skyriuose 2.2 ir 2.3 yra lygiaverčiai, pateiktų aproksimacijos kokybės kriterijų atžvilgiu. Nors šių metodų pateikiami rezultatai yra labai panašūs, nepatartina pasirinkti tik vieno iš jų. Atvirkščiai – rekomenduotina juos naudoti kartu, nes jie vienas kitą papildo.
2. Pirmas aproksimacijos metodas reikalauja labai didelio vartotojo indėlio, o reikalingi kompiuteriniai skaičiavimai atliekami greitai. Šis metodas vienu metu leidžia analizuoti/sudaryti tik vieną aproksimuojantį daugianarį. Tačiau jo pagalba, nagrinėjant įvairius duomenų pjūvius, galima vizualiai įvertinti funkcijų priklausomybes nuo pasirinktų kintamųjų bei vizualiai palyginti aproksimacijų tikslumą.
3. Antras aproksimacijos metodas reikalauja ilgų kompiuterinių skaičiavimų, o vartotojo indėlis labai mažas. Naudojant šį metodą vienu metu galima sudaryti, išanalizuoti ir palyginti keletą modelių. Taip pat šio metodo pagalba, galima žymiai suprastinti daugianario struktūrą.
4. Aproksimacijos kokybės vertinimui patartina naudoti kelis kriterijus, nes skirtingi modeliai gali būti tinkamesni skirtingų kriterijų atžvilgiu. Vertinant aproksimacijos kokybę pagal kelis kriterijus vienu metu, galima patikimiau išskirti efektyviausią modelį.
5. Modelių efektyvumo palyginimui taip pat patartina naudoti F kriterijų. Jis nustato, ar apsimoka naudoti sudėtingesnį daugianarį (atlikti daugiau skaičiavimų), norint gauti tikslesnę aproksimaciją.

LITERATŪRA

1. Katinas R., Listopadskis N. Daugiamačių duomenų aproksimavimas daugianariais. Bakalauro baigiamasis darbas. Kaunas, 2006, p. 15-23.
2. Plukas K. Skaitiniai metodai ir algoritmai. Kaunas, 2001, p. 272-421.
3. Bivariate polynomial interpolation. http://www.eso.org/projects/esomidas/doc/user/98NOV/volb/node153.html, 26/5/2006.
4. Two-dimensional polynomial interpolation from non uniform samples. http://www-video.eecs.berkeley.edu/papers/avz/Two-dimensional%20IEEE%20SP%2092.pdf, 14/5/2006.
5. Spline interpolation. http://bigwww.epfl.ch/thevenaz/interpolation, 16/5/2006.
6. Interpolation in two or more dimensions. http://www.library.cornell.edu/nr/bookcpdf/c3-6.pdf, 6/5/2006.
7. Nearest neighbour weighted interpolation. http://astronomy.swin.edu.au/~pbourke/modelling/weightinterp, 18/5/2006.
8. Nearest neighbor interpolation. http://iria.math.pku.edu.cn/~jiangm/courses/dip/html/node66.html, 12/4/2006.
9. Wikipedia, Linear interpolation. http://en.wikipedia.org/wiki/Linear_interpolation, 12/4/2006.
10. 2-dimensional linear interpolation. http://kom.aau.dk/~borre/masters/geoid/lecture4/interpol.pdf, 6/6/2006.
11. A chronology of interpolation. http://imagescience.bigr.nl/meijering/publications/download/pieee2002.pdf, 6/5/2006.
12. Polynomial approximation. http://iria.math.pku.edu.cn/~jiangm/courses/dip/html/node62.html, 7/6/2006.
13. Bilinear interpolation. http://xmm.vilspa.esa.es/scisim/release/latest/help/html/msim/node24.html, 7/6/2006.
14. Multidimensional linear interpolation. http://www.fho-emden.de/~hoffmann/multipol08062003.pdf, 7/6/2006.
15. Efunda, Least square method. http://www.efunda.com/math/leastsquares/leastsquares.cfm, 8/6/2006.
16. Interpolation methods. http://en.wikipedia.org/wiki/Interpolation, 24/4/2006.
17. Inverse distance weighting (IDW). http://en.wikipedia.org/wiki/Inverse_distance_weighting, 24/4/2006.
18. Kriging interpolation. http://en.wikipedia.org/wiki/Kriging, 18/4/2008.

19. How good the fit is. http://www.curvefit.com/goodness_of_fit.htm, 10/4/2008.

20. Comparing the fits of two models. http://www.curvefit.com/2_models__1_dataset.htm, 10/4/2008.

21. F-distribution. http://en.wikipedia.org/wiki/F-distribution, 17/4/2008.

22. Regularized incomplete Beta function. http://en.wikipedia.org/wiki/Regularized_incomplete_beta_function, 17/4/2008.

23. Free p-value calculator for the Fisher F-test. http://www.danielsoper.com/statcalc/calc07.aspx, 17/4/2008.

24. Statistics knowledge base. http://www.danielsoper.com/statkb/topic07.aspx, 17/4/2008.

25. Handbook of Mathematical Functions. http://www.nrbook.com/abramowitz_and_stegun/page_948.htm, 17/4/2008, p. 946-948.

26. Least-Squares Polynomial Approximation. http://www.chem.uoa.gr/applets/AppletPoly/Appl_Poly2.html, 17/4/2008.

27. High dimensional wavelet smoothing. http://anziamj.austms.org.au/V42/CTAC99/Niel/Niel.pdf, 20/50/2006.

28. Coeficient of determination. http://en.wikipedia.org/wiki/Coefficient_of_determination, 20/4/2008.

29. Jarque-Bera test. http://en.wikipedia.org/wiki/Jarque-Bera_test, 24/5/2008.

30. Wald-Wolfowitz runs test. http://en.wikipedia.org/wiki/Wald-Wolfowitz_runs_test, 2/6/2008.

31. Katinas, Raimondas; Listopadskis, Narimantas. Daugiamačių duomenų aproksimavimo problemos // Matematika ir matematinis modeliavimas / Kauno technologijos universitetas. - ISSN 1822-2757. - Kaunas. - T. 3 (2007), p. 177-186.

32. Katinas, Raimondas; Listopadskis, Narimantas. Daugiamačių duomenų aproksimavimo tikslumo kriterijų analizė // Taikomoji matematika = Applied mathematics : VII studentų konferencijos pranešimų medžiaga / Kauno technologijos universitetas. - ISBN 978-9955-25-488-1. - Kaunas, 2008, p. 37-38.

PRIEDAS 1. PROGRAMOS METODAI

Visas programos tekstas pateiktas kompaktiniame diske. Čia pateiksime tik keletą pačių svarbiausių programos metodų, kurie sudaro programos pagrindą.

```
//--Grazina duomenu failo varda-----------------------------------------------
String mano::grazintiDuomenuFailoVarda()
{
  if(Form1->OpenDialog1->Execute() && FileExists(Form1->OpenDialog1->FileName))
     return Form1->OpenDialog1->FileName;
  else return "";
}
//--grazina duomenu faile esanciu vardu skaiciu ir suraso vardus i comboBox1------
int mano::grazintiVarduSkaiciu(FILE * d)
{
 String c = "Buferis_reikalingas_kintamojo_vardui_nuskaityti";
 int simbolis;
 int k = 0;
 while((simbolis = fgetc(d)) != '\n')
    if(simbolis != ' ')
    {
       fseek(d,-1,1);
       fscanf(d,"%s",c);
       Form1->ComboBox1->Items->Insert(k++, c);
    }
 return k;
}
//--nuskaito duomenis, iveda juos i dinamini sarasa visasSarasas ir grazina
nuskaitymo pasekme----
bool mano::skaitytiDuomenis(FILE * d, int kiek)
{
 visasSarasas * D, *Pradzia = NULL;
 bool nuskaite = true;
 int k = 0;
 while(!feof(d))
 {
        D = new visasSarasas;
        if(D == NULL)
        {
           MessageDlg("Truksta atminties!", mtWarning, TMsgDlgButtons() << mbOK,
0);
           nuskaite = false;
```

```
        }
        else
        {
           D->S.x = new double [kiek];
              for(int i=0; i < kiek; i++)
                  fscanf(d, "%lf", D->S.x+i);
           fscanf(d, "\n");  k++;
           D->next = Pradzia;
           Pradzia = D;
        }
 }
    visoPradzia = Pradzia; tasku_sk = k;
  return nuskaite;
}
//-----sukuria mygtuka puslapyje--------------------
TButton* mano::sukurtiButtonInSheet(TTabSheet* sheet, int top, int left, int
width, String e)
{
   TButton* b = new TButton(sheet);
   b->Top = top;
   b->Left = left;
   b->Width = width;
   b->Caption = e;
   sheet->InsertControl(b);

   return b;
}
///-----------sudaromi kintamuju ir funkciju vardu masyvai--------------------
void mano::sudarytiKintamujuIrFunkcijuVarduMasyvus(int n)
{
   int j = 0;
   kintamujuVardai.Length = n;
   funkcijuVardai.Length = Form1->ComboBox1->Items->Count - n;
   for(int i = 0; i < n; i++)
       kintamujuVardai[i]=Form1->ComboBox1->Items->Strings[i];
   for(int i = n; i < Form1->ComboBox1->Items->Count; i++)
       funkcijuVardai[j++]=Form1->ComboBox1->Items->Strings[i];
   groupBoxFunkcijoms = sukurtiGroupBoxInSheet(Form1->TabSheet1, 40, 0,
24*(funkcijuVardai.Length+1), 161, "Funkcijos");
   groupBoxKintamiesiems = sukurtiGroupBoxInSheet(Form1->TabSheet1, 40, 168,
24*(kintamujuVardai.Length+1), 161, "Kintamieji");
   labelFunkcijoms.Length = funkcijuVardai.Length;
   labelKintamiesiems.Length = kintamujuVardai.Length;
```

```
   for(int i = 0; i < funkcijuVardai.Length; i++)
       labelFunkcijoms[i] = sukurtiLabelInGroup(groupBoxFunkcijoms, 16 + 24*i, 8,
funkcijuVardai[i]);
   for(int i = 0; i < kintamujuVardai.Length; i++)
       labelKintamiesiems[i] = sukurtiLabelInGroup(groupBoxKintamiesiems, 16 +
24*i, 8, kintamujuVardai[i]);
}
//-----grazina kintamojo indeksa is kintamoju vardu saraso------------------
int mano::grazintiKintamojoIndeksa(String e, DynamicArray<String> v)
{
  int indeksas = -1;
  for(int i = 0; i < v.Length; i++)
      if(v[i] == e)
         indeksas = i;

  return indeksas;
}
//---------suraso skirtingas fiksuotu kintamuju reiksmes i comboBox---------------
void mano::irasytiFixReiksmesIComboBox(int m, TComboBox* com, visasSarasas*
visoPr)
{

  bool yra = false;
  visasSarasas * D = visoPr;
  int j = 0;
  AnsiString e;

  com->Items->Clear();
  com->Items->Insert(j, FloatToStr(D->S.x[m]));
  D->next;
  j++;

  while(D)
  {
        for(int k = 0; k < com->Items->Count; k++)
        {
            e = com->Items->Strings[k];
            if (FloatToStr(D->S.x[m]) == e)
            {
                yra = true;
                break;
            }
        }
```

```
        if(yra == false)
        {
           com->Items->Insert(j, FloatToStr(D->S.x[m]));
           j++;
           yra = true;
        }

        yra = false;
        D = D->next;
  }
  if(com->Items->Count > 0)
     com->ItemIndex = 0;
}
//-------sudaromas duomenu atrinkimo filtras - fiksuotu kintamuju reiksmiu ir
vardu sarasiukas-------------------------
filtras* mano::sudarytiFiltroSarasa(DynamicArray<String> v,
DynamicArray<TComboBox*> co)
{
   filtras* f;
   f = new filtras [v.Length];
   for(int i = 0; i < v.Length; i++)
   {
      f[i].vardas = v[i];
      f[i].reiksme = StrToFloat(co[i]->Text);
   }
   return f;
}
//----------grazina atrinktu duomenu saraso pradzios rodykle --------------------
visasSarasas* mano::atrinktiPagalFixFiltra(visasSarasas* visoPr, filtras* f, int
n)
{
  visasSarasas * D, * B, *atrinktoPr;
  atrinktoPr = NULL;
  B = atrinktoPr;
  D = visoPr;
  int i = 0, indeksas = 0, j = 0;
  if(n == 0)
     atrinktoPr = visoPr;
  while(i < n)
  {
        for(int k = 0; k < visiVardai.Length; k++)
            if(f[i].vardas == visiVardai[k])
```

```
            {  indeksas = k;  k = visiVardai.Length; }

        atrinktoPr = NULL;   j = 0;
        while(D)
        {
                 if(D->S.x[indeksas] == f[i].reiksme)
                 {
                    B = new visasSarasas;   j++;
                    B->S.x = new double [visiVardai.Length];

                    for(int k = 0; k < visiVardai.Length; k++)
                        B->S.x[k] = D->S.x[k];
                    B->next = atrinktoPr;
                    atrinktoPr = B;
                 }
                 D = D->next;
        }
        D = atrinktoPr;
        i++;
  }
  return atrinktoPr;
}
//-------sukuria nauja langa-----------------
TForm* mano::sukurtiNaujaLanga(String e)
{
   TForm* form = new TForm(Form1);
   form->Top = 10;
   form->Left = 10;
   form->Height = 700;
   form->Width = 1000;
   form->Caption = "Grafikas" + e;
   form->Visible = true;
   form->Font->Style = TFontStyles()<< fsBold;
   form->Font->Size = 10;
   form->Font->Name = "Courier New";
   form->OnClose = uzdarytiForma;
   return form;
}
//------grazina aktyvios formos rodykle-------------------------
formuSarasas* mano::grazintiAktyviaForma()
{
    formuSarasas * D = formuPradzia;
    while(D)
```

```
    {
        if(D->K.forma->Active)
             break;
        D = D->sek;
    }
    return D;
//--------tiesine interpoliacija---------------------------------------
void mano::tiesineInterpoliacija(visiPasirinkimai* p, formuSarasas* f, TObject
*Sender)
{
    visiPasirinkimai * d = p;
    formuSarasas * D = grazintiAktyviaForma();
    int k = 0;
    int i = i = grazintiKintamojoIndeksa(D->X.parinktuFunkcijuVardai[0],
visiVardai);
    breztiPlokstuma(D->K.image, D->X.parinktuFunkcijuVardai,  visiVardai[D-
>X.x_ind]);
    atidetiXAsiesReiksmes(D->X.pix_x, D->R.AtrinktoPr, D->K.image, D->X.x_ind);
    atidetiFAsiesReiksmes(D->X.pix_y, D->R.AtrinktoPr, D->K.image, i);
    atidetiDuomenuPjuvioTaskus(D->X.pix_x, D->X.pix_y, i, 0, D->R.AtrinktoPr, D-
>R.GrupPr, D->K.image, D->R.L[0].checkBox, D->X.grup_ind, D->X.x_ind);
    int ind = grazintiKintamojoIndeksa(f->X.parinktuFunkcijuVardai[0],
visiVardai);
        if(d)
    {
       duomenuTinkleliai * t = d->F.t;
       while(t)
       {
             if((t->R.xi.Length > 1) && (f->R.L[0].checkBox[k]->Checked) && (f-
>R.L[0].checkBox[k]->Enabled))
             {
                double x1, y1, x2, y2, xv, yv, p1, q1, a, b;
                a = t->R.xi[0] * f->X.pix_x + 45 - min(f->X.x_ind, f-
>R.AtrinktoPr)* f->X.pix_x;
                b = f->K.image->Height - t->R.yi[0] * f->X.pix_y - 25 + min(ind,
f->R.AtrinktoPr)*f->X.pix_y;
                f->K.image->Canvas->MoveTo(a, b);
                f->K.image->Canvas->Pen->Color = t->R.spalva;
                f->K.image->Canvas->Pen->Width = 1;
                for(int i = 1; i < t->R.xi.Length; i++)
                {
                    x1 = t->R.xi[i-1]; x2 = t->R.xi[i];
                    y1 = t->R.yi[i-1]; y2 = t->R.yi[i];
```

```
                    xv = x1+(x2-x1)/2;
                    q1 = (xv-x1)/(x2-x1);  p1 = 1 - q1;
                    yv = p1*y1  + q1*y2;
                    b = f->K.image->Height - yv * f->X.pix_y - 25 + min(ind, f-
>R.AtrinktoPr)*f->X.pix_y;
                    a = xv * f->X.pix_x + 45 - min(f->X.x_ind, f->R.AtrinktoPr)*
f->X.pix_x;
                    f->K.image->Canvas->LineTo(a,b);
                    a = x2 * f->X.pix_x + 45 - min(f->X.x_ind, f->R.AtrinktoPr)*
f->X.pix_x;
                    b = f->K.image->Height - y2 * f->X.pix_y - 25 + min(ind, f-
>R.AtrinktoPr)*f->X.pix_y;
                    f->K.image->Canvas->LineTo(a,b);
                }
             }
             t = t->next;  k++;
       }
    }
    else
       MessageDlg("Pasirinkimų sąrašas tuščias!", mtInformation, TMsgDlgButtons()
<< mbOK, 0);
}
//-----------formuojamos sumos maziausiu kvadratu metodui------------
duomenuTinkleliai* mano::formuotiSumas(duomenuTinkleliai* t, int n)
{
   duomenuTinkleliai* d = t;
   double xk;
   DynamicArray< DynamicArray<double> > c;
   while(d)
   {
      sumb.Length = n+1;  sumx.Length = 2*n+1;
      for(int k = 0; k < sumx.Length; k++)
      {
         sumx[k] = 0;
         if(k < n+1)
            sumb[k] = 0;
      }
      for(int i = 0; i < d->R.xi.Length; i++)
      {
         xk = 1;
         for(int k = 0; k < sumx.Length; k++)
         {
            sumx[k] = sumx[k] + xk;
```

```
            if(k < n+1)
               sumb[k] = sumb[k] + d->R.yi[i]*xk;
            xk = xk * d->R.xi[i];
         }
      }
      /// formuojam matrica
      c.Length = n+1;
      for(int k = 0; k < n+1; k++)
      {
         c[k].Length = n+1;
         for(int j = 0; j < c[k].Length; j++)
            c[k][j] = sumx[k+j];
      }
      d->R.c = c;
      d->R.b = sumb;
      d = d->next;
   }

   return t;
}
//-------gauso metodas-------------------------
DynamicArray<double> mano::gausoMetodas(DynamicArray< DynamicArray<double> > c,
DynamicArray<double> b)
{
  double d,    //determinantas
         z,    //elementu apkeitimui
         s;    //santykis
  int l, k, i, j;       //ciklu kintamieji
  DynamicArray<double> x;

  int n = b.Length;
  x.Length = n;

    for(k = 0; k < n-1; k++)
    {          //iesko pagrindinio elemento
      l = k;
      for(i = k; i < n; i++)
         if(fabs(c[l][k]) < fabs(c[i][k]))
           l = i;
     //sukeicia eilutes vietomis
      if(l != k)
      {
        for(j = k; j < n; j++)
```

```
        {
          z = c[k][j];
          c[k][j] = c[l][j];
          c[l][j] = z;
        }
        d = -d;
      }
      z = b[k]; b[k] = b[l]; b[l] = z;     // mano
           //eiluciu perskaiciavimas
      for(i = k+1; i < n; i++)
      {
        if(c[k][k] == 0)
        {
          MessageDlg("Gauso metode dalyba iš nulio!", mtWarning, TMsgDlgButtons()
<< mbOK << mbCancel, 0);
          x.Length = 0;
          return x;
        }
        s = (double(c[i][k] / c[k][k]));
        c[i][k] = 0;
          for(j = k+1; j < n; j++)
            c[i][j]  = c[i][j] - s * c[k][j];
          b[i] = b[i] - s * b[k];       /// mano
      }
    }
    if(fabs(c[n-1][n-1]) == 0)
    {  MessageDlg("Gauso metode dalyba iš nulio!", mtWarning, TMsgDlgButtons() <<
mbOK << mbCancel, 0);
       x.Length = 0;
       return x;
    }
    else
       x[n-1] = b[n-1] / c[n-1][n-1];
    for(int i = n-2; i > -1; i--)
    {
        s = 0;
        for(int j = i+1; j < n; j++)
            s = s + c[i][j] * x[j];
        x[i] = (b[i] - s)/ c[i][i];
    }

 return x;
}
```

```
//----------------------grazina aproximuojancio polinomo reiksme---------
double mano::apskaiciuotiPolinomoReiksme(double x, int n, DynamicArray<double> b)
{
     double y, xk;
     y = 0;   xk = 1;
     for(int i = 0; i < n+1; i++)
     {
         y = y + b[i]*xk;
         xk = xk*x;
     }
     return y;
}
//--------brezia aproximacijas-------------------------
void mano::breztiAproksimuojanciasKreives(duomenuTinkleliai* t, formuSarasas* f,
int n)
{

  double xp, x, xg, y, a, b, zingsnis;
  //aproksimuojama tik parinklta pirma funkcija
  int in = grazintiKintamojoIndeksa(f->X.parinktuFunkcijuVardai[0], visiVardai);
  int k = f->R.L[0].checkBox.Length-1;
  duomenuTinkleliai* d = apsuktiRinkini(t);
  f->K.image->Canvas->Pen->Color = clBlack;
  zingsnis = (max(f->X.x_ind, f->R.AtrinktoPr)-min(f->X.x_ind, f-
>R.AtrinktoPr))/100;

  while(d)
  {
     if((d->R.a.Length != 0) && (f->R.L[0].checkBox[k]->Checked) && (f-
>R.L[0].checkBox[k]->Enabled))
     {
         xp = d->R.xi[0] * f->X.pix_x + 45 - min(f->X.x_ind, f->R.AtrinktoPr)* f-
>X.pix_x;
         y = apskaiciuotiPolinomoReiksme(d->R.xi[0], n, d->R.a);
         y = f->K.image->Height - y * f->X.pix_y - 25 + min(in, f-
>R.AtrinktoPr)*f->X.pix_y;
         f->K.image->Canvas->MoveTo(xp, y);
         f->K.image->Canvas->Pen->Color = d->R.spalva;
         f->K.image->Canvas->Pen->Width = 1;
         f->K.image->Canvas->Pen->Style = psDot;
         x = d->R.xi[0]+zingsnis;

         while(x < grazintiMaxReiksme(d->R.xi))
```

```
         {

                y = apskaiciuotiPolinomoReiksme(x, n, d->R.a);
                y = f->K.image->Height - y * f->X.pix_y - 25 + min(in, f-
>R.AtrinktoPr)*f->X.pix_y;
                xp = x * f->X.pix_x + 45 - min(f->X.x_ind, f->R.AtrinktoPr)* f-
>X.pix_x;
                f->K.image->Canvas->LineTo(xp, y);
                x = x+zingsnis;//0.1;
         }

     }
     d = d->next;  k--;
  }
}
//----grazina save dialoge pasirinkta failo varda-----------------------
String mano::grazintiSaveDialogFailoVarda(formuSarasas* D)
{
    //String failoVardas = D->K.f->Caption + ".graf";
    String failoVardas = "Grafikas" + IntToStr(D->X.nr) + ".graf";
    Form1->SaveDialog1->FilterIndex = 1;
    Form1->SaveDialog1->FileName = failoVardas;
    if(Form1->SaveDialog1->Execute())
       return Form1->SaveDialog1->FileName;
    else return "";
}
//---------tiesioginis medzio apejimas ieskant pazymetos virsunes-----------------
void mano::tiesioginisApejimas(medis* m, String e)
{
   medis* d, *mm = m;

   if(mm != NULL)
   {
      if(mm->duom.vardas + mm->duom.kodas == e)
         dabartinisMazgas = mm;

      for(int i = 0; i < mm->rod.Length; i++)
          tiesioginisApejimas(mm->rod[i],e);
   }
}
///---suranda SS---------------
double mano::grazintiPaklaiduSuma()
{
```

```
  xPTSarasas* D = xPTPradzia;
  int k = D->S.x.Length - 1, i = 0;
  double suma, skirt;
  suma = fabs(D->S.x[k-1] - D->S.x[k]);
  D = D->next;
  while(D)
  {
     skirt = (D->S.x[k-1] - D->S.x[k]) * (D->S.x[k-1] - D->S.x[k]);
     suma = suma + skirt;
     D = D->next;  i++;
  }

  return suma;
}
///-----apskaiciuojamas  determincijos koeficientas----------------
double mano::grazintiDetKoeficienta(double vid)
{
  xPTSarasas* D = xPTPradzia;
  int k = D->S.x.Length - 1;
  double sstot = 0, sserr = 0, skirt1, skirt2, determ;
  String e = "";
  while(D)
  {
    skirt1 = (D->S.x[k-1] - D->S.x[k]) * (D->S.x[k-1] - D->S.x[k]);
    sserr = sserr + skirt1;
    skirt2 = (D->S.x[k-1] - vid) * (D->S.x[k-1] - vid);
    sstot = sstot + skirt2;
    D = D->next;
  }
  if(sstot != 0)
     determ = 1 - (sserr/sstot);
  else
    ShowMessage("Dalyba iš nulio skaičiuojant determinacijos koeficientą!");
  return determ;
}

//---apskaiciuojama teorine funkcijos reiksme-----------------------------------
double mano::apskaiciuotiTeorineFunkcijosReiksme(medis* d, DynamicArray<double>
xReiksmes, int lyg, double suma)
{
   int p; /// is kokio laipsnio kintamojo padaunginsime koeficienta
   String e;
   double sandauga;//, suma;
```

```
   if(d != NULL)
   {
      if(d->duom.lygis == d->duom.maxl)
      {
         sandauga = d->duom.kofai[0];
         for(int k = lyg; k < d->duom.maxl; k++)
         {
             p = StrToInt(d->duom.kodas.SubString(k+1, 1));
             if(p > 0) /// jei kintamojo laipsnis didesnis negu 0
             {
               sandauga = sandauga * pow(xReiksmes[k], p);
             }
         }
               suma = suma + sandauga;
               sandauga = 1;
      }
         for(int i = 0; i < d->rod.Length; i++)
            suma = apskaiciuotiTeorineFunkcijosReiksme(d->rod[i], xReiksmes, lyg,
suma);
   }
   return suma;
}
//--------atsapauzdina medzio struktura ir koeficientus-----------------
void mano::spausdintiMedi(medis* m)
{
    medis* d = m, *d1 = m;
    memoEilute = "";
    int lyg = d->duom.maxl;
    for(int i = 0; i < lyg; i++)
    {
         surasytiLygioSakas(d, i);/// spausdinti lygio sakas
         memoEilute = d1->duom.x_kint + " " + memoEilute;
         Form1->Memo1->Lines->Add(memoEilute);
         memoEilute = "";
         d1 = d1->rod[0];
    }
    d = m;
    formuotiKoeficientuEilute(d);
    Form1->Memo1->Lines->Add(memoEilute);
}
////-------grazina generuojamu sandaugu kieki---------------------
int mano::grazintiSandauguKieki(DynamicArray<int> laipsniuEil)
{
```

```
   int sandauga = 1, ilgis = laipsniuEil.Length;

   for(int i = 0; i < ilgis; i++)
      sandauga = sandauga * (laipsniuEil[i] + 1);

   return sandauga;

//------is laipsniu matricos istrinamos eilutes atitinkancios nepazymetus
chekboxus--------
void mano::koreguotiLaipsniuMatrica()
{
    int ilgis = checkBoxaiSandaugoms.Length;
    sudarytiLaipsniuMatrica(grazintiMaxLaipsniuEilute());
    uncheckIndeksai = grazintiNepazymetusCheckBoxIndeksus(ilgis);
    for(int i = uncheckIndeksai.Length-1; i > -1 ; i--)
       if(laipsniai.Length-1 == uncheckIndeksai[i])
          laipsniai.Length--;
       else
          trintiLaipsniuMatricosEilute(uncheckIndeksai[i]);

//----suraso sandaugu aproksimaciojos paklaidasi memo-----------------------------
-
void mano::surasytiPaklaidasIMemo(TMemo* me)
{
  String e = "";
  for(int i = 0; i < EditaiLaipsniams.Length; i++)
      e = e + EditaiLaipsniams[i]->Text + " ";
  me->Lines->Add("Modelis: " + e);
  int ilgis = uncheckIndeksai.Length, df;
  df = laipsniai.Length-1;///EditaiLaipsniams.Length - ilgis-1;
  if(ilgis > 0)
  {
     e = "Nepažymėti: ";
     me->Lines->Add("*** *** *** *** *** ***");
     for(int i = 0; i < ilgis; i++)
        e = e + uncheckIndeksai[i] + " ";
     me->Lines->Add(e);
  }
  double vidurkis = grazintiSandauguPaklaiduVidurki();
  double dispersija = grazintiSandauguPaklaiduDispersija();
  double min = grazintiSandauguMinPaklaida();
  double max = grazintiSandauguMaxPaklaida();
  double suma = grazintiSandauguPaklaiduSuma();
```

```
  double FPvidurkis = grazintiSandauguFunkcijosPVidurki();
  double determ = grazintiDeterminacijosKoeficienta(FPvidurkis);
  me->Lines->Add("*** *** *** *** *** ***");
  me->Lines->Add(surasytiKoeficientusIEilute());
  me->Lines->Add("*** *** *** *** *** ***");
  me->Lines->Add("Paklaidų charakteristikos:");
  e = "Vidurkis = " + FloatToStrF(vidurkis, ffExponent, 5, 0);
  me->Lines->Add(e);
  e = "Dispersija = " + FloatToStrF(dispersija, ffExponent, 5, 0);
  me->Lines->Add(e);
  e = "Min = " + FloatToStrF(min, ffExponent, 5, 0);
  me->Lines->Add(e);
  e = "Max = " + FloatToStrF(max, ffExponent, 5, 0);
  me->Lines->Add(e);
  e = "SSreg = " + FloatToStrF(suma, ffExponent, 5, 0);
  me->Lines->Add(e);
  e = "Se = " + FloatToStrF(sqrt(suma/(tasku_sk-df)), ffExponent, 5, 0);
  me->Lines->Add(e);
  e = "R2 = " + FloatToStrF(determ, ffGeneral, 5, 0);
  me->Lines->Add(e);
  me->Lines->Add("*** *** *** *** *** ***");
  me->Lines->Add("");
}
///---------sudaroma laipsniu matrica---------------------
void mano::sudarytiLaipsniuMatrica(DynamicArray<int> laipsniuEil)
{
   int m = 0, kiek, laip, kiekis = grazintiSandauguKieki(laipsniuEil), ilgis =
EditaiLaipsniams.Length;
   laipsniai.Length = kiekis;// eiluciu skaicius
   for(int i = 0; i < kiekis; i++)
        laipsniai[i].Length = ilgis;   // eilutes stulpeliu skaicius
    kiek = kiekis;
    for(int i = 0; i < ilgis; i++)
    {
        laip = laipsniuEil[i] + 1;
        kiek = kiek / laip;
        while(m < kiekis)
            for(int j = 0; j < laip; j++)
                for(int p = 0; p < kiek; p++)
                    laipsniai[m++][i] = j;
        m = 0;
    }
}
```

```
///-----grazinama kintamuju sandaugos reiksme-----------------------------
double mano::grazintiSandauga(DynamicArray<double> reiksmes, int i, int j)
{
  double sandauga, sandauga1 = 1, sandauga2 = 1;

  for(int k = 0; k < reiksmes.Length; k++)
  {
     if(laipsniai[i][k] != 0)
        sandauga1 = sandauga1 * pow(reiksmes[k], laipsniai[i][k]);

     if(laipsniai[j][k] != 0)
        sandauga2 = sandauga2 * pow(reiksmes[k], laipsniai[j][k]);
  }
  sandauga = sandauga1 * sandauga2;
  return sandauga;
//--------------apskaiciuojama sandaugu polinomo reiksme----------------------
double mano::apskaiciuotiSandauguPolinomoReiksme(DynamicArray<double> reiksmes,
DynamicArray<double> b)
{
  double suma = 0, sandauga;

  for(int i = 1; i < b.Length; i++)
  {
      sandauga = grazintiSandauga(reiksmes, i, 0) * b[i];
      suma = suma + sandauga;
  }
  suma = suma + b[0];
  return suma;
}

///--------sudaroma maximaliu laipsniu eilute-----------------------
DynamicArray<int> mano::grazintiMaxLaipsniuEilute()
{
  int ilgis = EditaiLaipsniams.Length;
  DynamicArray<int> laipsniuEil;
  laipsniuEil.Length = ilgis;
  for(int i = 0; i < ilgis; i++)
     laipsniuEil[i] = StrToInt(EditaiLaipsniams[i]->Text);
  return laipsniuEil;
}
///--------surasomi kriterijai i staticText-----------------------
```

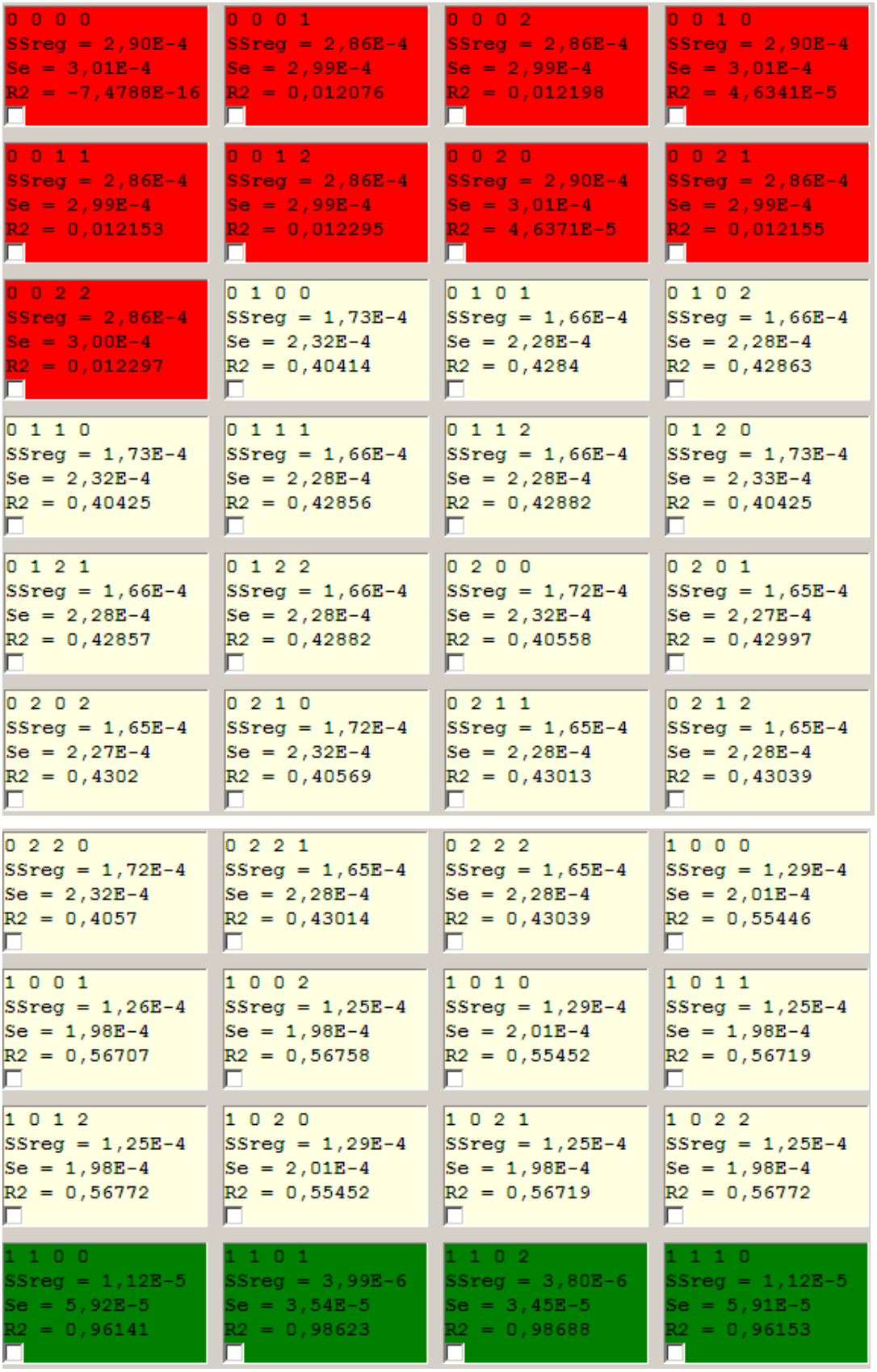
```
void mano::surasytiKriterijusIStaticText(TStaticText* st, DynamicArray<int>
laipsniuEil, int j)
{
  String e = "";
  int df = grazintiSandauguKieki(laipsniuEil) - 1;
  double ss = grazintiSandauguPaklaiduSuma(), r2 =
grazintiDeterminacijosKoeficienta(grazintiSandauguFunkcijosPVidurki());

  for(int i = 0; i < laipsniuEil.Length; i++)
     e = e + laipsniuEil[i] + " ";
  e = e + "\n";
  e = e + "SSreg = " + FloatToStrF(ss, ffExponent, 3, 0) + "\n";
  e = e + "Se = " + FloatToStrF(sqrt(ss/(tasku_sk-df)), ffExponent, 3, 0) + "\n";
  e = e + "R2 = " + FloatToStrF(r2, ffGeneral, 5, 0) + "\n";
  st->Caption = e;
  DegrOfFree[j] = df;
  SSreg[j] = ss;
  R2[j] = r2;
  if(r2 > 0.66)
     st->Color = clGreen;
  else if(r2 < 0.33)
     st->Color = clRed;
  else
     st->Color = clInfoBk;
}

//-------buttonFTest OnClick mtodas--------------------------------
void __fastcall mano::buttonFTestOnClick(TObject *Sender)
{
TCursor curs = Screen->Cursor;
Screen->Cursor = crHourGlass;
try{
   DynamicArray<int> check; /// du pazymetu check boxu indeksai;
   check.Length = 2;
   int k = 0;

   for(int i = 0; i < CheckBoxKriterijams.Length; i++)
      if(CheckBoxKriterijams[i]->Checked)
         check[k++] = i;

   int df1 = tasku_sk - DegrOfFree[check[0]], d1;
   int df2 = tasku_sk - DegrOfFree[check[1]], d2;
   String e = "";
```

```
   double ssreg1 = SSreg[check[0]], ss1;
   double ssreg2 = SSreg[check[1]], ss2;
   double F, bf, b1, fdist;

   if(df1 > df2)
   {
       if(ssreg1 <= ssreg2)
       {
          MessageDlg("Sudėtingesnio modelio nuokrypių kvadratų suma ne mažesnė.
\nRekomenduotina pasirinkti paprastesnį modelį.", mtInformation, TMsgDlgButtons()
<< mbOK, 0);
          StaticTextKriterijams[check[0]]->Font->Style = TFontStyles()<<
fsUnderline << fsBold;
       }
       else
       {
          F = ((ssreg1 - ssreg2) / (df1 - df2)) / (ssreg2 / df2);
          d1 = df1-df2;  d2 = df2;
          ///skaiciuoti p reiksme
          F = d1*F/(d1*F+d2);
          bf = grazintiBetaReiksme(F, d1/2, d2/2);
          b1 = grazintiBetaReiksme(1, d1/2, d2/2);
          fdist = bf/b1;
          if(1-fdist >= 0.05)
          {
             MessageDlg("p reikšmė didesnė už 0.05. \nRekomenduotina pasirinkti
paprastesnį modelį.", mtInformation, TMsgDlgButtons() << mbOK, 0);
             StaticTextKriterijams[check[0]]->Font->Style = TFontStyles()<<
fsUnderline << fsBold;
          }
          else
          {
             MessageDlg("p reikšmė mažesnė už 0.05. \nRekomenduotina pasirinkti
sudėtingesnį modelį.", mtInformation, TMsgDlgButtons() << mbOK, 0);
             StaticTextKriterijams[check[1]]->Font->Style = TFontStyles()<<
fsUnderline << fsBold;
          }
       }
   }
   else if(df1 < df2)
   {
       if(ssreg1 >= ssreg2)
       {
```

```
          MessageDlg("Sudėtingesnio modelio nuokrypių kvadratų suma ne mažesnė.
\nRekomenduotina pasirinkti paprastesnį modelį.", mtInformation, TMsgDlgButtons()
<< mbOK, 0);
          StaticTextKriterijams[check[1]]->Font->Style = TFontStyles()<<
fsUnderline << fsBold;
       }
       else
       {
          F = ((ssreg2 - ssreg1) / (df2 - df1)) / (ssreg1 / df1);
          d1 = df2-df1;    d2 = df1;
          F = d1*F/(d1*F+d2);
          ///skaiciuoti p reiksme
          bf = grazintiBetaReiksme(F, d1/2, d2/2);
          b1 = grazintiBetaReiksme(1, d1/2, d2/2);
          fdist = bf/b1;

          if(1-fdist >= 0.05)
          {
             MessageDlg("p reikšmė didesnė už 0.05. \nRekomenduotina pasirinkti
paprastesnį modelį.", mtInformation, TMsgDlgButtons() << mbOK, 0);
             StaticTextKriterijams[check[1]]->Font->Style = TFontStyles()<<
fsUnderline << fsBold;
          }
          else
          {
             MessageDlg("p reikšmė mažesnė už 0.05. \nRekomenduotina pasirinkti
sudėtingesnį modelį.", mtInformation, TMsgDlgButtons() << mbOK, 0);
             StaticTextKriterijams[check[0]]->Font->Style = TFontStyles()<<
fsUnderline << fsBold;
          }
       }
   }
   else  /// jei laisves laipsniu skaicius lygus modeli vertinam pagal ssreg
   {
      MessageDlg("Abiejų modelių laisvės laipsnių skaičius vienodas. \nPatartina
lyginti nuokrypių kvadratų sumas.", mtInformation, TMsgDlgButtons() << mbOK, 0);
      if(ssreg1 < ssreg2)
      {
         StaticTextKriterijams[check[0]]->Font->Style = TFontStyles()<<
fsUnderline << fsBold;
      }
      else if(ssreg1 > ssreg2)
      {
```

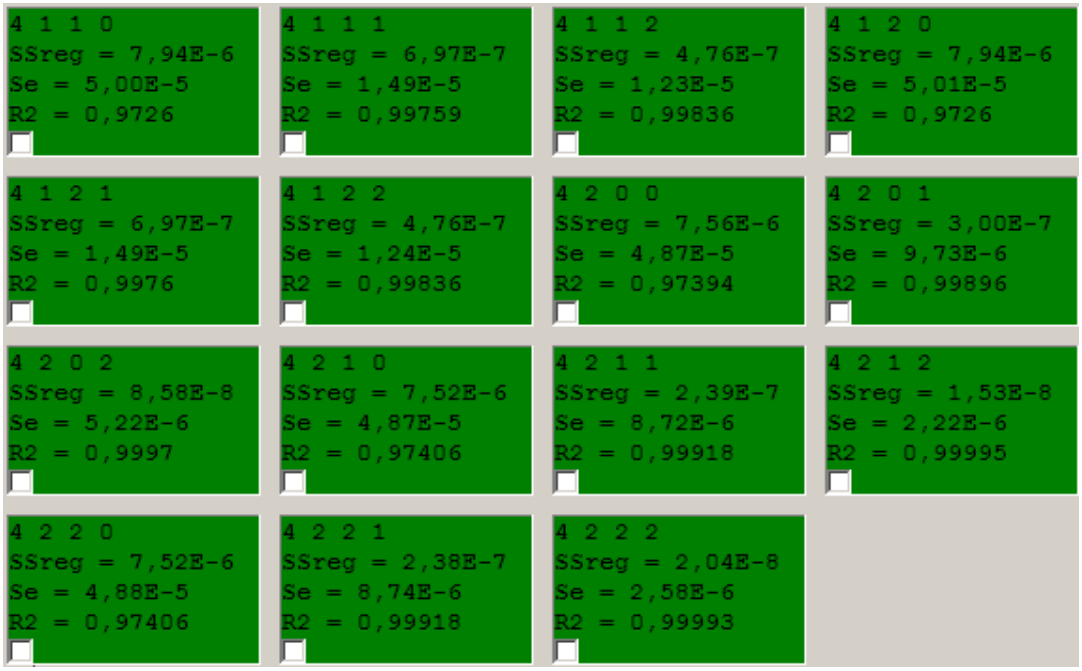
```
         StaticTextKriterijams[check[1]]->Font->Style = TFontStyles()<<
fsUnderline << fsBold;
      }
      else
      {
         MessageDlg("Abu modeliai lygiaverčiai.", mtInformation, TMsgDlgButtons()
<< mbOK, 0);
      }
   }
}
__finally
{
 Screen->Cursor = curs;
}
}
//---grtazina pointegralines Beta funkcijos reiksme-------------------------------
double mano::grazintiFunkcijosReiksme(double F, double d1, double d2)
{
   if((F == 0)||(F == 1))
     return 0;
   else
     return pow(F, d1-1)*pow(1-F, d2-1);
}
//---------------------------------------------------------------------------
double mano::grazintiBetaReiksme(double F, double d1, double d2)
{
   double zingsnis = 0.000001;
   double xk = 0, xd = zingsnis, suma = 0, fk, fd;
   fk = grazintiFunkcijosReiksme(xk, d1, d2);
   fd = grazintiFunkcijosReiksme(xd, d1, d2);
   suma = (fk+fd)*zingsnis/2;  //trapecijos plotas
   while(xd + zingsnis < F)
   {
      xk = xd; xd = xd + zingsnis; fk = fd;
      fd = grazintiFunkcijosReiksme(xd, d1, d2); suma = suma + (fk+fd)*zingsnis/2;
   }
   return suma;
}
```

PRIEDAS 2. MODELIŲ ŽEMĖLAPIS



1 1 1 1 SSreg = 3,93E-6 Se = 3,51E-5 R2 = 0,98643	1 1 1 2 SSreg = 3,73E-6 Se = 3,43E-5 R2 = 0,98712	1 1 2 0 SSreg = 1,12E-5 Se = 5,91E-5 R2 = 0,96153	1 1 2 1 SSreg = 3,93E-6 Se = 3,52E-5 R2 = 0,98644
1 1 2 2 SSreg = 3,73E-6 Se = 3,43E-5 R2 = 0,98712	1 2 0 0 SSreg = 1,08E-5 Se = 5,81E-5 R2 = 0,96286	1 2 0 1 SSreg = 3,54E-6 Se = 3,33E-5 R2 = 0,9878	1 2 0 2 SSreg = 3,34E-6 Se = 3,24E-5 R2 = 0,98846
1 2 1 0 SSreg = 1,07E-5 Se = 5,80E-5 R2 = 0,96298	1 2 1 1 SSreg = 3,48E-6 Se = 3,31E-5 R2 = 0,98801	1 2 1 2 SSreg = 3,28E-6 Se = 3,22E-5 R2 = 0,9887	1 2 2 0 SSreg = 1,07E-5 Se = 5,81E-5 R2 = 0,96299
1 2 2 1 SSreg = 3,48E-6 Se = 3,31E-5 R2 = 0,98801	1 2 2 2 SSreg = 3,32E-6 Se = 3,25E-5 R2 = 0,98855	2 0 0 0 SSreg = 1,27E-4 Se = 1,99E-4 R2 = 0,56354	2 0 0 1 SSreg = 1,23E-4 Se = 1,96E-4 R2 = 0,57621
2 0 0 2 SSreg = 1,23E-4 Se = 1,96E-4 R2 = 0,57678	2 0 1 0 SSreg = 1,27E-4 Se = 1,99E-4 R2 = 0,5636	2 0 1 1 SSreg = 1,23E-4 Se = 1,96E-4 R2 = 0,57633	2 0 1 2 SSreg = 1,23E-4 Se = 1,96E-4 R2 = 0,57693
2 0 2 0 SSreg = 1,27E-4 Se = 1,99E-4 R2 = 0,5636	2 0 2 1 SSreg = 1,23E-4 Se = 1,96E-4 R2 = 0,57633	2 0 2 2 SSreg = 1,23E-4 Se = 1,97E-4 R2 = 0,57693	2 1 0 0 SSreg = 8,55E-6 Se = 5,17E-5 R2 = 0,9705
2 1 0 1 SSreg = 1,34E-6 Se = 2,05E-5 R2 = 0,99539	2 1 0 2 SSreg = 1,13E-6 Se = 1,88E-5 R2 = 0,99612	2 1 1 0 SSreg = 8,52E-6 Se = 5,17E-5 R2 = 0,97062	2 1 1 1 SSreg = 1,27E-6 Se = 2,00E-5 R2 = 0,9956
2 1 1 2 SSreg = 1,06E-6 Se = 1,83E-5 R2 = 0,99636	2 1 2 0 SSreg = 8,52E-6 Se = 5,17E-5 R2 = 0,97062	2 1 2 1 SSreg = 1,27E-6 Se = 2,01E-5 R2 = 0,99561	2 1 2 2 SSreg = 1,84E-6 Se = 2,42E-5 R2 = 0,99364
2 2 0 0 SSreg = 8,13E-6 Se = 5,05E-5 R2 = 0,97195	2 2 0 1 SSreg = 8,79E-7 Se = 1,66E-5 R2 = 0,99697	2 2 0 2 SSreg = 6,67E-7 Se = 1,45E-5 R2 = 0,9977	2 2 1 0 SSreg = 8,10E-6 Se = 5,04E-5 R2 = 0,97208
2 2 1 1 SSreg = 8,18E-7 Se = 1,61E-5 R2 = 0,99718	2 2 1 2 SSreg = 5,96E-7 Se = 1,38E-5 R2 = 0,99794	2 2 2 0 SSreg = 8,10E-6 Se = 5,05E-5 R2 = 0,97208	2 2 2 1 SSreg = 8,17E-7 Se = 1,61E-5 R2 = 0,99718

2 2 2 2 SSreg = 6,02E-7 Se = 1,39E-5 R2 = 0,99792	3 0 0 0 SSreg = 1,26E-4 Se = 1,99E-4 R2 = 0,56537	3 0 0 1 SSreg = 1,22E-4 Se = 1,96E-4 R2 = 0,57804	3 0 0 2 SSreg = 1,22E-4 Se = 1,96E-4 R2 = 0,57862
3 0 1 0 SSreg = 1,26E-4 Se = 1,99E-4 R2 = 0,56543	3 0 1 1 SSreg = 1,22E-4 Se = 1,96E-4 R2 = 0,57817	3 0 1 2 SSreg = 1,22E-4 Se = 1,96E-4 R2 = 0,57877	3 0 2 0 SSreg = 1,26E-4 Se = 1,99E-4 R2 = 0,56543
3 0 2 1 SSreg = 1,22E-4 Se = 1,96E-4 R2 = 0,57817	3 0 2 2 SSreg = 1,22E-4 Se = 1,96E-4 R2 = 0,57877	3 1 0 0 SSreg = 8,01E-6 Se = 5,01E-5 R2 = 0,97236	3 1 0 1 SSreg = 7,97E-7 Se = 1,58E-5 R2 = 0,99725
3 1 0 2 SSreg = 5,84E-7 Se = 1,36E-5 R2 = 0,99798	3 1 1 0 SSreg = 7,98E-6 Se = 5,01E-5 R2 = 0,97248	3 1 1 1 SSreg = 7,36E-7 Se = 1,52E-5 R2 = 0,99746	3 1 1 2 SSreg = 5,15E-7 Se = 1,28E-5 R2 = 0,99823
3 1 2 0 SSreg = 7,98E-6 Se = 5,01E-5 R2 = 0,97248	3 1 2 1 SSreg = 7,35E-7 Se = 1,53E-5 R2 = 0,99747	3 1 2 2 SSreg = 5,14E-7 Se = 1,28E-5 R2 = 0,99823	3 2 0 0 SSreg = 7,59E-6 Se = 4,88E-5 R2 = 0,97381
3 2 0 1 SSreg = 3,39E-7 Se = 1,03E-5 R2 = 0,99883	3 2 0 2 SSreg = 1,25E-7 Se = 6,27E-6 R2 = 0,99957	3 2 1 0 SSreg = 7,56E-6 Se = 4,88E-5 R2 = 0,97393	3 2 1 1 SSreg = 2,77E-7 Se = 9,38E-6 R2 = 0,99904
3 2 1 2 SSreg = 5,41E-8 Se = 4,16E-6 R2 = 0,99981	3 2 2 0 SSreg = 7,56E-6 Se = 4,89E-5 R2 = 0,97393	3 2 2 1 SSreg = 2,76E-7 Se = 9,40E-6 R2 = 0,99905	3 2 2 2 SSreg = 5,82E-8 Se = 4,34E-6 R2 = 0,9998
4 0 0 0 SSreg = 1,26E-4 Se = 1,99E-4 R2 = 0,56548	4 0 0 1 SSreg = 1,22E-4 Se = 1,96E-4 R2 = 0,57816	4 0 0 2 SSreg = 1,22E-4 Se = 1,96E-4 R2 = 0,57874	4 0 1 0 SSreg = 1,26E-4 Se = 1,99E-4 R2 = 0,56554
4 0 1 1 SSreg = 1,22E-4 Se = 1,96E-4 R2 = 0,57828	4 0 1 2 SSreg = 1,22E-4 Se = 1,96E-4 R2 = 0,57889	4 0 2 0 SSreg = 1,26E-4 Se = 1,99E-4 R2 = 0,56554	4 0 2 1 SSreg = 1,22E-4 Se = 1,96E-4 R2 = 0,57828
4 0 2 2 SSreg = 1,22E-4 Se = 1,97E-4 R2 = 0,57889	4 1 0 0 SSreg = 7,98E-6 Se = 5,00E-5 R2 = 0,97248	4 1 0 1 SSreg = 7,58E-7 Se = 1,54E-5 R2 = 0,99738	4 1 0 2 SSreg = 5,46E-7 Se = 1,31E-5 R2 = 0,99812


4 1 1 0
SSreg = 7,94E-6
Se = 5,00E-5
R2 = 0,9726
4 1 1 1
SSreg = 6,97E-7
Se = 1,49E-5
R2 = 0,99759
4 1 1 2
SSreg = 4,76E-7
Se = 1,23E-5
R2 = 0,99836
4 1 2 0
SSreg = 7,94E-6
Se = 5,01E-5
R2 = 0,9726
4 1 2 1
SSreg = 6,97E-7
Se = 1,49E-5
R2 = 0,9976
4 1 2 2
SSreg = 4,76E-7
Se = 1,24E-5
R2 = 0,99836
4 2 0 0
SSreg = 7,56E-6
Se = 4,87E-5
R2 = 0,97394
4 2 0 1
SSreg = 3,00E-7
Se = 9,73E-6
R2 = 0,99896
4 2 0 2
SSreg = 8,58E-8
Se = 5,22E-6
R2 = 0,9997
4 2 1 0
SSreg = 7,52E-6
Se = 4,87E-5
R2 = 0,97406
4 2 1 1
SSreg = 2,39E-7
Se = 8,72E-6
R2 = 0,99918
4 2 1 2
SSreg = 1,53E-8
Se = 2,22E-6
R2 = 0,99995
4 2 2 0
SSreg = 7,52E-6
Se = 4,88E-5
R2 = 0,97406
4 2 2 1
SSreg = 2,38E-7
Se = 8,74E-6
R2 = 0,99918
4 2 2 2
SSreg = 2,04E-8
Se = 2,58E-6
R2 = 0,99993