KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ KATEDRA

Ramūnas Lukošius

Laisvaisiais trianguliuotais paviršiais apribotų erdvinių kūnų tinklelių generavimo algoritmų sukūrimas

Magistro darbas

Darbo vadovas
prof. habil. dr. Rimantas Barauskas

Kaunas, 2011

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ KATEDRA

Ramūnas Lukošius

Laisvaisiais trianguliuotais paviršiais apribotų erdvinių kūnų tinklelių generavimo algoritmų sukūrimas

Magistro darbas

Recenzentas
doc. dr.
2011-05-

Vadovas
prof. habil. dr. R.Barauskas
2011-05-

Atliko
IFM-9/1 gr. stud.
Ramūnas Lukošius
2011-05-

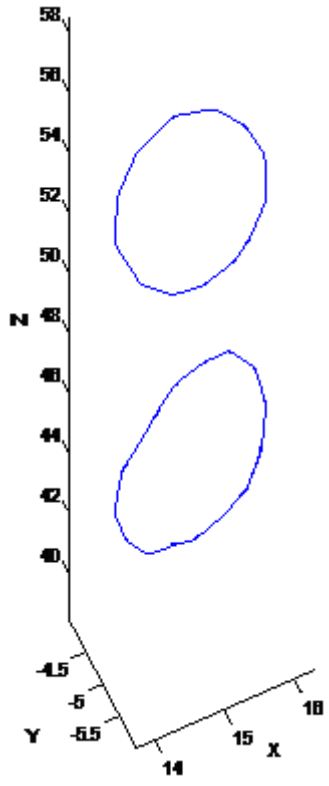
Kaunas, 2011

TURINYS

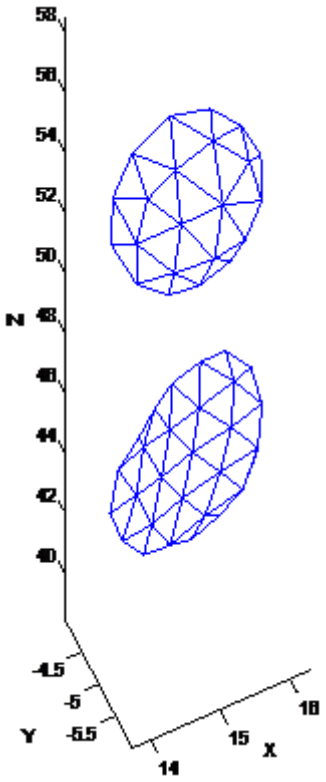
1 Įvadas

Šio magistrinio darbo bendroji tema yra tinklelių generavimas. Apibendrinant, sugeneruoti tinkleliai naudojami tyrimams atlikti. Tyrimai gali būti įvairūs, pavyzdžiui: šilumos sklidimo, magnetinių jėgų kūne moduliacija, jėgos dydžiui į tam tikrą tašką nustatymui.

Magistrinio darbo pasirinkta sritis yra labai problemiška, tai laisvaisiais trianguliuotais paviršiais apribotų erdvinių kūnų tinklelių generavimo algoritmų sukūrimas. Problemiška dalis yra ta, kad erdvinis kūnas yra iš anksto apribotas laisvaisiais trianguliuotais tinkleliais. Tai labai pasunkina užduotį, nes sutrianguliuoti paprastą cilindro formos kūną(pav. 1.1), nuo pradžių, pačiam pasirenkant taškus yra gana paprasta. Magistrinio tyrimo objektas yra jau esamų dvi-dimensinių taškų erdvinė trianguliacija(pav. 1.2).



Pav. 1.1 Tuščias cilindras

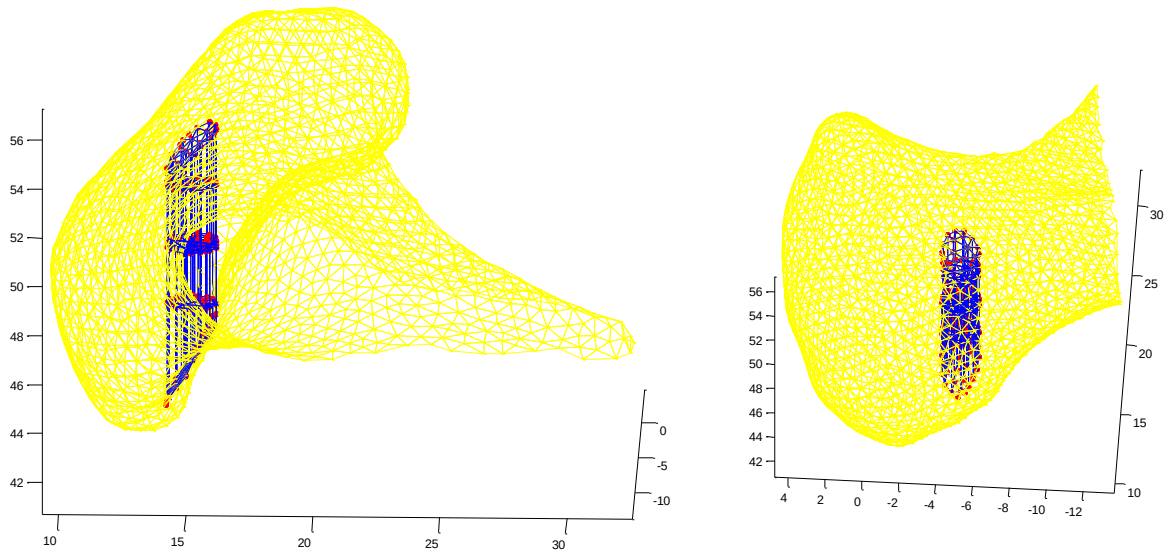


Pav. 1.2 Apribotas tinkleliu cilindras

Pavaizduoti kūnai(pav. 1.1 ir pav. 1.2) tėra didesnio laisvaisiais trianguliuotais paviršiais apriboto erdvinio kūno išpjova, kurią tiriame.

Literatūroje toks uždavinys nėra aprašomas. Yra labai daug literatūros apie įvairias trianguliacijos formas ir metodus. Matematiniai metodai aprašyti literatūroje, tokie kaip reguliarūs, projekciniai, plintančio fronto ir kiti aptariame tolimesniame skyriuje. Naudojantis šių metodų duomenų struktūra, bei kai kuriomis idėjomis, plėtojamas magistrinis darbas.

Tokio tipo uždavinio sprendimas, būtų labai naudingas tiriant laisvaisiais trianguliuotais paviršiais apribotų erdvinių kūnų savybes. Išsprendus sujungimo problemą, iš laisvojo kūno galima būtų tyrinėti tam tikrą jo dalį. Paveiksle 1.3 pavaizduota sąnario galvutė, ir „iškirptas“ cilindras. Šis cilindras ir jo trianguliacija ir yra pagrindinė tyrinėjamos problemos dalis.



Pav. 1.3. Sąnario ir tyrinėjamo objekto vaizdas

Darbo tikslas:

Tobulinti skaitinei fizikinių sistemų analizei taikomus automatinio tinklelių generavimo algoritmus, pritaikant juos erdviniams kūnams, aprašytiems jų ribinių paviršių tinkleliais.

Darbo uždaviniai:

- Atlikti žinomų automatinių tinklelių generavimo matematinių metodų, algoritmų ir juos realizuojančių programinės įrangos sistemų apžvalgą.
- Sukurti algoritmą plokštuminio tinklelio generavimui, kai dvimatė sritis duota apribojant ją dalinai ar pilnai diskretizuotomis kreivėmis.
- Sukurti algoritmus erdvinių tinklelių generavimui, kai diskretizuojamas tūris duotas apribojant jį dalinai ar pilnai diskretizuotais paviršiais
- Matematiškai suformuluoti ir programiškai pritaikyti normas sugeneruotų tinklelių kokybės įvertinimui.

Reikalavimai sugeneruotų tinklelių kokybei ir programinei realizacijai:

- Tinkleliai turi kiek galima geriau priglusti prie ribojančių paviršių. Jų paviršius nebūtinai turi atkartoti ribojančių paviršių trianguliaciją. Mat, toliau jie naudojami fizikinio proceso modeliavimui, todėl „praslydimas“ paviršiaus atžvilgiu neišvengiamas. Po praslydimo besiglaudžiančių paviršių tinkleliai, kaip taisyklė, nesutampa.
- Tinklelių elementai gali būti laisvos tetraedrų , prizmių arba piramidžių (trikampių arba keturkampių) formos. Svarbiausia, kad nebūtų „išsigimusių“ elementų, tai optimizavimo kriterijus. Išsigimęs elementas turi kampų, artimų 0 arba pi. Geriausi yra tinkleliai, kurių visi elementai yra neišsigimę heksaedrai.
- Pageidautina, kad algoritmai būtų realizuoti MATLAB-e ar bent jau būtų galimybė panaudoti juos MATLAB terpėje.

2 Baigtinių elementų tinklelių generavimo metodų ir algoritmų apžvalga

Daugelis inžinerijos programų naudoja trianguliacijas (tinklelius), kaip erdvinio vaizdo pagrindą.

Atsižvelgiant į duotus taškus R^d (d>2), šių taškų trianguliacija užpildo atitinkamą iškilųjį korpusą tam tikru kiekiu elementų, kurie bendrai yra natūraliai paprasti (trikampis d=2, keturšonis d=3). Taip pat šių figūrų (d=2,d=3) tinklelis padengia sritį paprastomis geometrinėmis figūromis, kaip ketursienis, penkiasienis ir t.t.

Algoritmams, kurie naudojami sudaryti trianguliacijas, yra paskirtas gausus kompiuterinės geometrijos kiekis dokumentų ir knygų. Labai daug dėmesio yra skiriama Delone trianguliacijoms. Delone algoritmai yra svarbi dalis tų algoritmų, kurie naudojami konstruojant tinklelius. Taigi trianguliacijos algoritmai yra labai svarūs apibrėžiant tinklelių sudarymo algoritmus.

Baigtinių elementų metodas yra labai plačiai taikomas kompiuterių inžinerijoje sprendžiant įvairių tipų diferencialinių lygčių problemas, tokias kaip standžioji mechanika, skysčių mechanika, šilumos modeliavimas ir t.t. Šiam metodui reikalingas srities tinklelis, apie kurį yra formuojamos lygtys. Taigi tinklelių algoritmai yra labai svarbūs kiekviename modeliavime pagrįstame baigtinių elementų metodu.

Baigtinių elementų metodas kilo iš reikėjimo spręsti sudėtingas lankstumo ir struktūrinės analizės problemas civilinėje(tiltai, užtvankos, pastatai..) ir aeronautikos inžinerijoje. BEM vystymosi ištakos skaičiuojamos nuo Alexander Hrennikoff ir Richard Courant darbų 1941-1942 metais. Jų sprendimo metodai labai smarkiai skiriasi nuo dabartinių, tačiau išliko pagrindinė charakteristika- ploto tinklelių diskretizacija į grupę mažesnių plotų, plotai dažniausiai vadinami elementais.

XX amžiaus šeštajame dešimtmetyje naudota standumo matrica ir elementų surinkimas naudojamas iki šiol baigtinių elementų metodų darbuose. 1973 metais NASA užsakymu buvo sukurta baigtinių elementų metodo programinio paketo vystymosi kryptis su griežtu matematiniu pagrindu. Nuo to laiko baigtinių elementų metodas buvo apibendrinamas, kaip taikomosios matematikos šaka skaitiniam fizinių sistemų modeliavimui[1,6].

2.1 Baigtiniai elementai

Baigtinių elementų metodas yra esminis įrankis sudėtingų figūrų supratimui ir vaizdavimui. Tipinė baigtinio elemento sesija prasideda su supaprastintu modeliu, tam kad sugeneruoti grubų tinklelį, su kuriuo įgaunamas pradinis supratimas apie figūrą. Šis supratimas nuves iki tinklelio tankumo ir formos perdirbimo[4].

Formaliai baigtinius elementus galima aprašyti naudojanti trigubą elementą[7]:

$$[K , P_K, \sum_K] \quad (1)$$

Šiame trigubame elemente „K“ yra geometrinis elementas (tinklelio elementas). „P_K“ reiškia baigtinio elemento užimamos erdvės funkcijas. Pagrindinėmis funkcijomis yra laikomos baigtinių elementų formos funkcijos. „$\sum_K$“ yra rinkinys laisvės laipsnių susijusių su „K“

Geometriniu požiūriu ir pasak erdvinės dimensijos, „K“ gali būti trikampis, keturkampis, penkiakampis, šešiakampis ir t.t.

Erdvė „P_K“ paprastai sudaryta iš polinomų. Ši erdvė su baigtine dimensija N sudaryta iš N pagrindinių funkcijų. Taigi, jei p_i yra bazinė funkcija, tai bet kuri p funkcija esanti „P_K“ gali būti užrašyta kaip:

$$p = \sum_{i=1}^{N} \alpha_i p_i \quad (2)$$

Kur α_i yra laisvės laipsniai.

Laisvės laipsnių rinkinys „$\sum_K$“ gali būti vieni iš funkcijos p reikšmių, kur p yra funkcija, esanti „P_K“ viename iš mazgų „K“ elemente. Toks mazgas apibūdinamas α. Baigtinis elementas turintis tokio tipo mazgą vadinamas Lagrandžo tipo elementu.

Kai $\varphi_i(p)$ apima nors vieną p išvestinės reikšmę, pavyzdžiui akivaizdžiuose žymėjimuose

$$Dp, \frac{\partial p}{\partial x}, \frac{\partial p}{\partial y}, \frac{\partial p}{\partial n}, D^2 p, \frac{\partial^2 p}{\partial x^2} \text{ ir t.t} \quad (3)$$

Tai tada sakome, kad tai Ermito tipo baigtinis elementas.

Sekant šiuos apibrėžimus, galima pastebėti, kad „P_K“ matomos kaip erdvinės funkcijos (p) arba tiesiogiai kaip bazinių funkcijų sąrašas (p_i), kas yra du skirtingi būdai užrašant tą patį. Panašiai „$\sum_K$“ gali būti apibūdintas kaip laisvės taškų rinkinys ($\varphi_i(p)$). Taip mes pasiekiame globalų taškų apibrėžimą arba mazgų rinkinį (α) su kuriuo susiję vienas arba keli laisvės laipsniai. Priklausant nuo to koks yra kontekstas, viena iš šių sąvoku bus naudojama.

Svarbus faktas yra tas, kad trigubas elementas [K , P_K, $\sum_K$] deramai aprašo baigtinį elementą, jei rinkinys $\sum_K$ yra P_K vienintelis sprendinys. Tai reiškia, kad egzistuoja tokios N funkcijos p_i erdvėje P_k, kurios yra tiesiškai nepriklausomos. Konkrečiai mes turime:

$$\varphi_j(p_i) = \delta_{ij} \tag{4}$$

Su δ_{ij} Kronekerio delta ($\delta_{ij} = 1\ jei\ i = j\ ir\ \delta_{ij} = 0\ jei\ i <> j$)

Labiau bendras šios sąvokos apibrėžimas yra panaudoti nuorodos elementą K kartu su koordinačių žymėjimo funkcija F_K, tam kad apibrėžti prieš tai naudotą trigubą nelygybę šia charakteristika:

$$[\widehat{K}\ , \widehat{P}, \widehat{\sum}] \tag{5}$$

Kitaip tariant unikali charakteristika $[\widehat{K}\ , \widehat{P}, \widehat{\sum}]$ gali būti panaudota charakterizuoti visus baigtinius elementus tinklelyje, kurie turi tą patį geometrinį tipą, su sąlyga, kad duotos atitinkamos funkcijos F_K. Taigi pradedant nuo vieno $[\widehat{K}\ , \widehat{P}, \widehat{\sum}]$ ir vienos funkcijos F_K yra įmanoma rasti [K , P_K, $\sum_K$] kiekvienam elementui K. Jei $\hat{p}$ nusako funkcijas $\widehat{P}$, mes turime:

$$K = F_K\left(\widehat{K}\right) \tag{6}$$

$$P_K = \{p = \hat{p}F_K^{-1}, \hat{p} \in \widehat{P}\} \tag{7}$$

$$\sum\nolimits_K = \{p(a), a = F_K(\hat{a}), \hat{a} \in \widehat{K}\} \tag{8}$$

Kur, dėl paprastumo mes priimame tik vieną laisvės laipsnį formos p ir, kur $\hat{a}$ reiškia $\widehat{K}$ taškus. P_K dar gali būti aprašyta kaip:

$$P_K = \{p = \sum \alpha_i p_i\ su\ p_i = \hat{p}_i F_K^{-1}, \hat{p}_i \in \widehat{P}, i = 1, N\} \tag{9}$$

Dėl patogumo $\sum_K$ gali būti išreikštas dviem būdais

$$\sum\nolimits_K = \{\varphi_i(p), i = 1, N\} \tag{10}$$

$$\sum\nolimits_K = \{\alpha_j, j = 1, M; \varphi_i\left(p(a_j^i)\right), i = 1, N\} \tag{11}$$

Kitais žodžiais sakant arba naudotis visu laisvės laipsnių sąrašu, arba naudotis mazgų sąrašu ir kiekvienam atitinkančio laisvės laipsnio sąrašu.

Ankščiau paminėtas apibrėžimas reiškia, kad F_K yra inversiškas, iš to išplaukia, kad abu trigubi dariniai yra ekvivalentiški. Tokiu būdu mes apibrėžėme baigtinių elementų šeimos narius, kurie gali būti atstovaujami naudojantis unikaliu atstovavimo elementu. Būtina pastebėti, kad naudotis atstovavimo elementu nėra griežtai reikalaujama supaprastinto elemento atveju, bet tai yra paprastumo šaltinis dėl lengvo įgyvendinimo[7].

2.1.1 Elementarios figūros

Šiame skyriuje aptarsiu pagrindinius baigtiniu elementų elementus, tokius kaip trikampis ir ketursienis.

2.1.1.1 Trikampis

2.1.1.1.1 Elementarus apibrėžimas

Trikampis yra gerai žinomas geometrinis kūnas. Grubiai tariant trikampis yra trijų kampų daugiakampis. Jis apibrėžiamas trimis vertikalėmis, žymimomis, kaip P_i ,kurios duotos prieš laikrodžio rodyklę[6].

$$K = (P_1, P_2, P_3) \tag{12}$$

Trikampį galima išreikšti 6 būdais su šiomis trimis vertikalėmis. Jei nustatyta orientacija, tada galima išreikšti trimis būdais. Taigi jei trikampis yra plokštumoje, jo orientacija netiesiogiai išreikšta plokštumos normalės ir 3 galimi apibūdinimai laidžia manyti kad jo paviršius, pažymėtas kaip S_K, išreikštas. Todėl trikampiai apie kuriuos kalbėsiu bus tiksliai išreikšti. Taigi plotą S_K galima aprašyti kaip:

$$S_K = \frac{1}{2}\begin{vmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{vmatrix} \tag{13}$$

arba

$$S_K = \frac{1}{2}\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} \tag{14}$$

kur x_i, y_i yra viršūnių P_i koordinatės, o |.| reškia determinantą.

Šis apibrėžimas leidžia aiškiai apibrėžti kraštines(ar kampus) duoto trikampio. Taip pat su trikampiu galima susieti papildomą informaciją naudojant trianguliaciją ar tinklelius.

2.1.1.1.2 Apskritimas apie trikampį

Kaip matysime vėliau trikampio apskritimas bus labai svarbus trianguliacijos algoritmuose. Šis apskritimas apibrėžiamas savo centru ir spinduliu. Šiuos du subjektus galima apskaičiuoti dviem būdais[6].

Spręsti tiesinę sistemą yra pirmas būdas, o naudoti apskritimo lygtį yra alternatyvus būdas apibrėžti apskritimą. Apskritimo lygtis yra:

$$\Delta_K(x, y) = 0 \tag{15}$$

$$\Delta_K(x, y) = \begin{vmatrix} l_1^2 - l^2 & l_2^2 - l_1^2 & l_3^2 - l_1^2 \\ x_1 - x & x_2 - x_1 & x_3 - x_1 \\ y_1 - y & y_2 - y_1 & y_3 - y_1 \end{vmatrix} \tag{16}$$

Antras būdas yra rasti susikirtimą tarp statinių(susijusių su trikampiu K) ir apskritimo spindulys tada randamas kaip atstumas tarp šio susikirtimo taškų iki kraštinės. Tai galima išreikšti lygtimi:

$$\begin{pmatrix} x_3 - x_1 & y_3 - y_1 \\ x_3 - x_2 & y_3 - y_2 \end{pmatrix} \begin{pmatrix} x_C \\ y_C \end{pmatrix} = \frac{1}{2} \begin{pmatrix} (x_3^2 + y_3^2) & (x_1^2 + y_1^2) \\ (x_3^2 + y_3^2) & (x_2^2 + y_2^2) \end{pmatrix} \tag{17}$$

Formulė su kuria randamas spindulys yra:

$$r_K = \frac{L_1 L_2 L_3}{4 S_K} \tag{18}$$

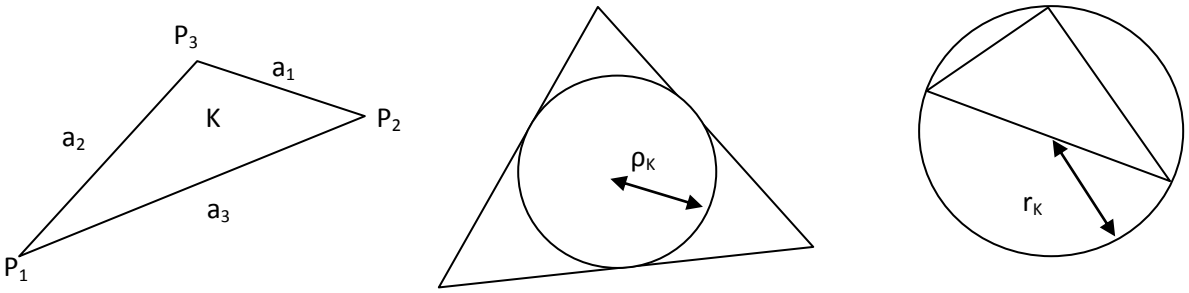
kur L_i yra kraštinės ilgis, o S_K – trikampio plotas.

2.1.1.1.3 Apskritimas trikampyje

Trikampį susiedami su apskritimu įvedame naują dydį ρ_K:

$$\rho_K = \frac{S_K}{p_K} \tag{19}$$

kur p_K yra pusperimetris.



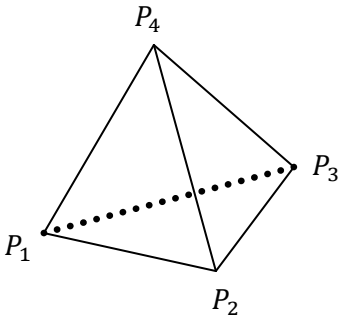
2.1 pav. Trikampio pusperimetris

2.1.1.2 Ketursienis

2.1.1.2.1 Elementarus apibrėžimas

Ketursienis yra daugiakampis sudarytas iš keturių trikampių, turintis tetraedro savybių[6]. Jį galima apibūdinti keturgubu surikiuotu sąrašu sudarytų iš jo viršūnių P_i:

$$K = (P_1, P_2, P_3, P_4) \tag{20}$$



2.2 pav. Ketursienis

Ketursienį galima aprašyti dvylika būdų, kad jis būtų orientuotas. Kadangi manome kad paviršius orientuotas, reiškia normalės orientuotos taip pat. Tarkime tūris V_K aprašytas iš K elementų, tada V_K aprašoma:

$$V_K = \frac{1}{6}\begin{vmatrix} x_2 - x_1 & x_3 - x_1 & x_4 - x_1 \\ y_2 - y_1 & y_3 - y_1 & y_4 - y_1 \\ z_2 - z_1 & z_3 - z_1 & z_4 - z_1 \end{vmatrix} \tag{21}$$

arba

$$V_K = \frac{1}{6}\begin{vmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ x_4 & y_4 & z_4 & 1 \end{vmatrix} \tag{22}$$

kur x_i, y_i, z_i yra viršūnės P_i koordinatės.

Šis formatas leidžia mums aprašyti keturias tetraedro sienas:

- Paviršius 1: P_4 P_3 P_2
- Paviršius 2: P_1 P_3 P_4
- Paviršius 3: P_4 P_2 P_1
- Paviršius 4: P_1 P_2 P_3
-

Atitinkamai galima aprašyti ir tetraedro briaunas:

- Briauna 1: P_1 P_2
- Briauna 2: P_1 P_3

- Briauna 3: P_1 P_4
- Briauna 4: P_2 P_3
- Briauna 5: P_2 P_4
- Briauna 6: P_3 P_4

2.1.1.2.2 Apskritimas apie ketursienį

Kiekvienas ketursienis turi juo susijusi apskritimą. Jo centras ir spindulys randamas tokiu pačiu būdu, kaip ir dviejų dimensijų. Arba sprendžiant lygtį[6]:

$$\Delta_K(x,y,z) = 0 \tag{23}$$

su

$$\Delta_K(x,y,z) = \begin{vmatrix} l_1^2 - l^2 & l_2^2 - l_1^2 & l_3^2 - l_1^2 & l_4^2 - l_1^2 \\ x_1 - x & x_2 - x_1 & x_3 - x_1 & x_4 - x_1 \\ y_1 - y & y_2 - y_1 & y_3 - y_1 & y_4 - y_1 \\ z_1 - z & z_2 - z_1 & z_3 - z_1 & z_4 - z_1 \end{vmatrix} \tag{24}$$

kur $l^2 = x^2 + y^2 + z^2$ ir $l_i^2 = x_i^2 + y_i^2 + z_i^2$ (i=1,4)

arba sprendžiant tiesinę sistemą. Ši sistema parodo, kad šis taškas yra ties pusiaukampinių susikirtimu. Radus centrą, spindulys randamas apskaičiuojant atstumą nuo centro iki bet kurios iš viršūnių. Verta paminėti, kad šis būdas vartoja mažiau kompiuterio resursų, nei pirmasis.

Be centrinio taško spindulį dar galima rasti šia formule:

$$r_k = \frac{\sqrt{(a+b+c)(a+b-c)(b+c-a)(a-b+c)}}{24V_k} \tag{25}$$

kur a,b,c yra ilgis tarp priešingų viršūnių.

2.1.1.2.3 Apskritimas ketursienyje

Šio apskritimo ketursienyje spindulys aprašomas formule:

$$\rho_K = \frac{3V_K}{S_1 + S_2 + S_3 + S_4} \tag{26}$$

kur S_i – i elemento plotas.

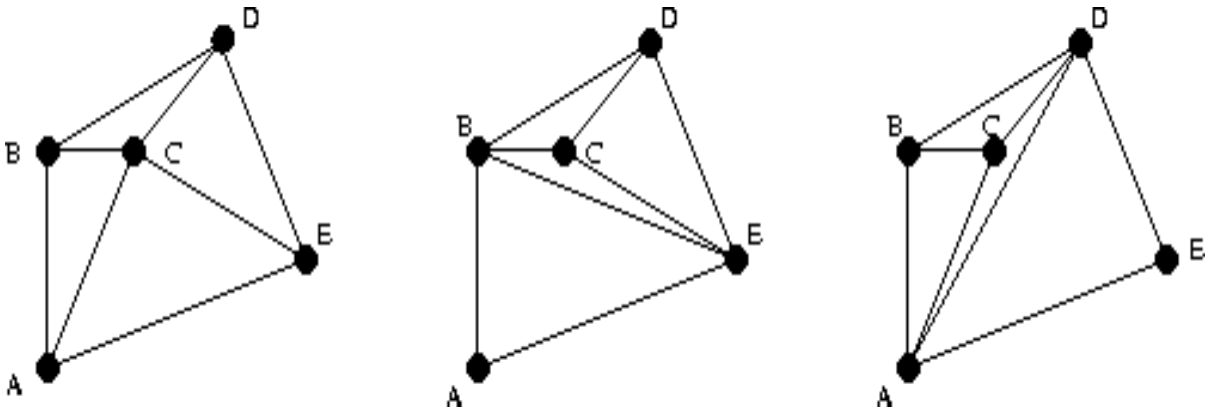
2.2 Matematiniai tinklelių generavimo metodai

2.2.1 Delone trianguliacija

Daugelis tinklelių generavimo algoritmų remiasi Delone trianguliacija ir Delone kriterijumi. Nors pats Delone metodas priskiriamas plintančio fronto algoritmui (šį algoritmą plačiau apžvelgsiu kitame skyriuje), tačiau Delone kriterijus ir jo idėjos taikomos beveik visuose kompiuteriu generuojamose sistemose[2].

Delone algoritmas yra vienas iš pagrindinių, dėlto jį apžvelgsiu plačiau modeliuojant dviejų dimensijų tinklelių generavimą:

Kelis erdvėje esančius kūnus galima sutrianguliuoti keliais būdais. Šiuos penkis taškus galima sutrianguliuoti trimis būdais[3]:



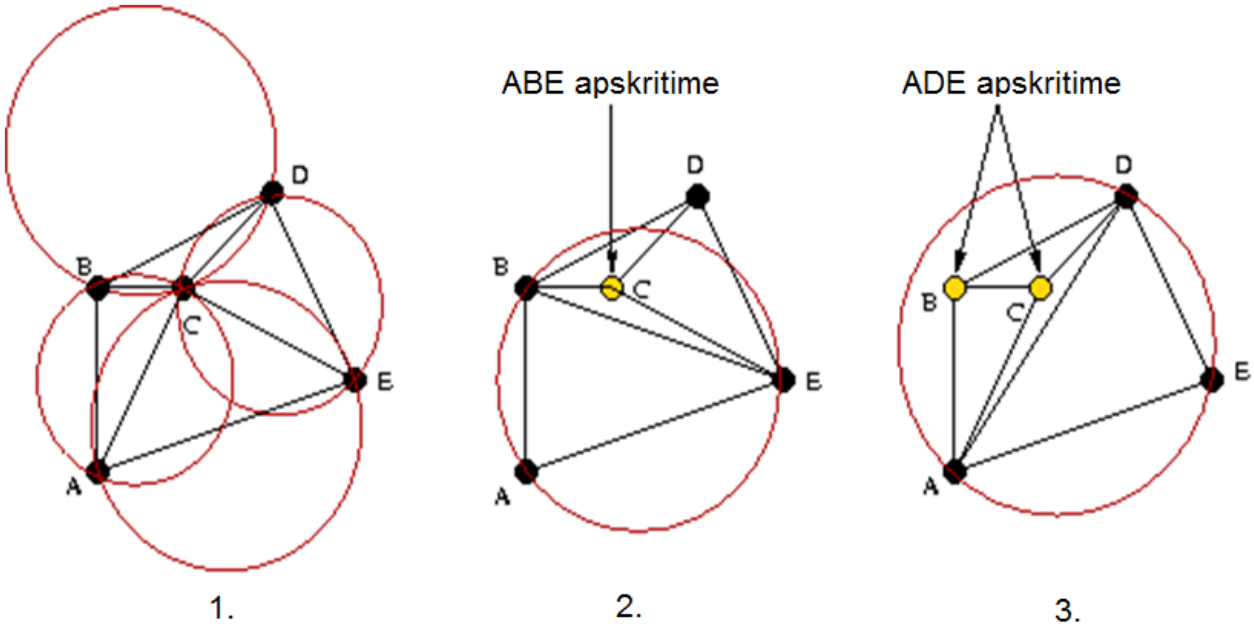
2.1 pav. Trianguliacijos būdai

Kartais, mums reikia sutrianguliuoti taškus taip, kad jais būtu patogu naudotis. Vienas iš geriausių ir daugiausiai naudojamų trianguliacijų yra Delone trianguliacija.

Delone trianguliacija duotiems taškams aprašoma taip:

Atkarpa AB yra kraštinė Delone trianguliacijos, jei yra apskritimas jungiantis A ir B taip, kad visos kitos taškų grupės C, kur C nelygu A ar B yra apskritimo išorėje.

Tai reiškia, kad visi Delone trikampiai duotiems taškams turės apibrėžtus apskritimus ir nė vienas taškas nebus apskritimo viduje.

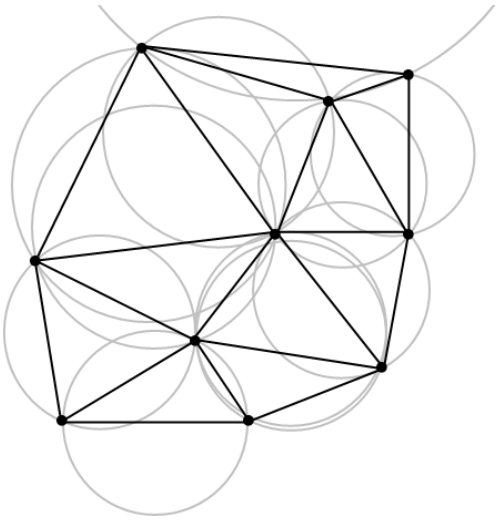


2.2 pav. Trianguliacijos būdai su apskritimais

Iš pateikto 2.2 paveikslėlio iš karto galima pasakyti, kad Delone trianguliacija yra 1.

Dviejuose dimensijose(2d), beveik visoms duotoms taškų grupėms egzistuoja Delone trianguliacija. Ši trianguliacija visada yra unikali, jei tik į apskritimą neįeina 4 taškai. Delone trianguliacija minimizuoja mažus kampus, taip sumažindama išsigimusių(smailių) trikampių buvimą. Delone trianguliacijos neįmanoma įvykdyti, jei duoti taškai rikiuojasi vienoje tiesėje. Trijų dimensijų(3d) trianguliacijoje vietoj apskritimo naudojama sfera, kurios viduje, taip pat negali būti taškų.

Dešimties taškų rinkinio Delone trianguliacijos pavyzdys[2]:



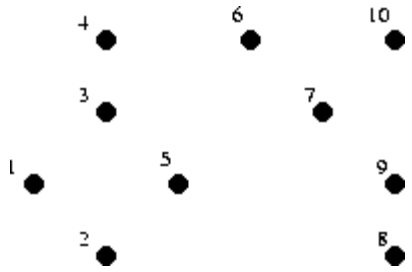
2.3 pav. Delone 10 taškų trianguliacijos pavyzdys

2.2.1.1 Delone trianguliacijos pavyzdys

Mes norime sugeneruoti baigtį trikampių tinklą, kuris tenkintų Delone trianguliaciją. Tai yra mes norime galėti apskaičiuoti modifikuotą Delone trianguliaciją, kurioje vartotojas pats nurodo kraštines(sujungimus). Kadangi Delone trianguliacija yra unikali bet kokioms taškų grupėms (su išimtimi, jei į apskritimą įeina daugiau nei 3 taškai), sugeneruota Delone trianguliacija greičiausiai turės keletą kraštinių, kurios neatitiks Delone kriterijų.

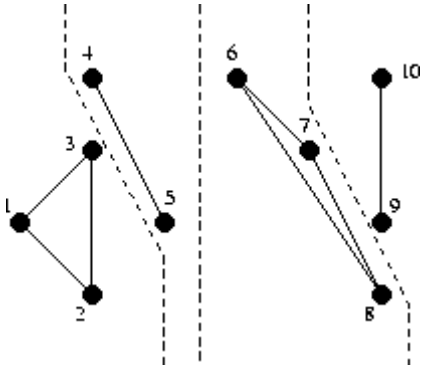
Šiame pavyzdyje naudosime skaldyk ir valdyk algoritmą (divide and conquer). Skaldyk ir valdyk algoritmas skaičiuoja tik išgaubto korpuso trikampius duotam taškų rinkiniui[3].

Pirmasis žingsnis yra surikiuoti visus taškus didėjančiai x koordinatės atžvilgiu. Jei taškai x koordinatėse sutampa, tada atsižvelgiama į y koordinatę. Surikiuoti taškai 2.4 paveiksle.



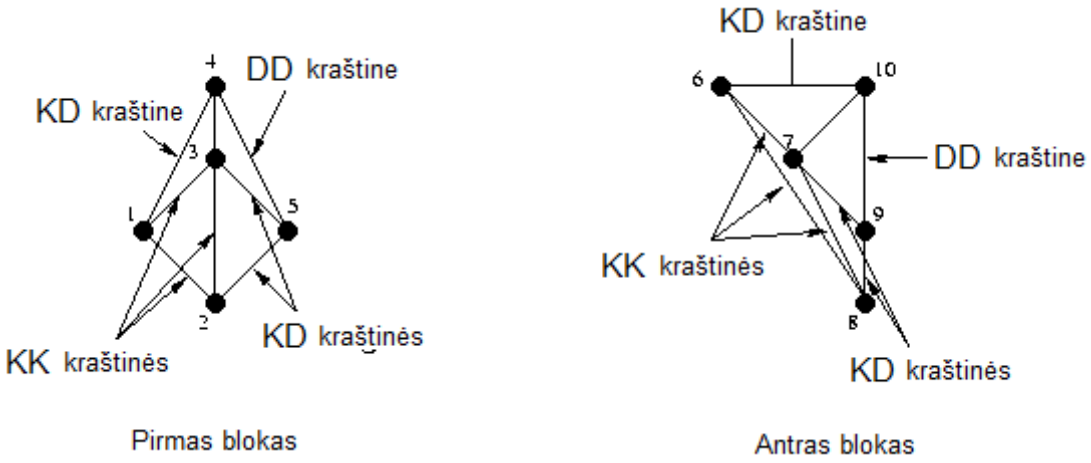
2.4 pav. Surikiuoti taškai

Kai taškai surikiuoti, jie iš eilės padalinami perpus į „blokus“, tol kol viename bloke nėra daugiau nei 3 taškų(2.5pav). Šie taškai blokuose gali būti sutrianguliuoti iškarto. Atkarpa- jei bloke du taškai, bei trikampiu, jei bloke trys taškai.



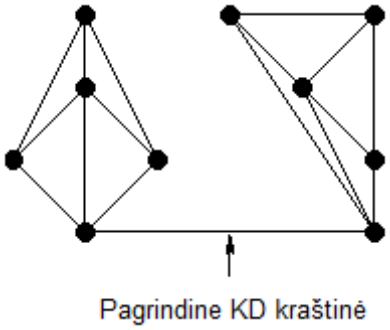
2.5 pav. Padalinti į blokus taškai

Tolimesniame žingsnyje sutrianguliuoti blokai sujungiami. Sujungimas vyks kairiame (KK) bloke ir dešiniame bloke(DD). Šiuo atveju šių bloku yra po 2. Tam kad išliktų Delone reikalavimai sujungtuose blokuose, gali tekti ištrinti KK ar DD kraštines, tačiau niekada nesukuriam naujų KK ir DD kraštinių(2.6pav).



2.6 pav. Sujungti blokai į KK ir DD

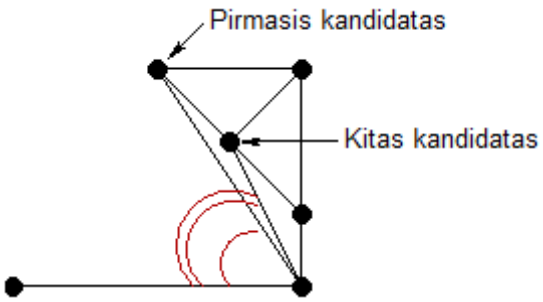
Jungiat likusius 2 blokus pirmasis žingsnis yra įterpti KD kraštinę. KD kraštinė yra apatinė kraštinė, kuri nesikerta su jokia KK ar DD kraštine.



2.7 pav. Pagrindinė KD kraštinė

Dirbant toliau į viršų, mus reikia rasti kitą KD kraštinę, esančią virš pagrindinės KD kraštinės. Aišku, kad šios naujos kraštinės vienas iš taškų priklausys kairiai ar dešiniai pagrindinės kraštinės pusei. O kitas galas bus arba kairio arba dešinio bloko taškas. Taigi siaurindami paiešką mes pasirenkame du taškus: vieną iš kairio bloko ir vieną iš dešinio bloko.

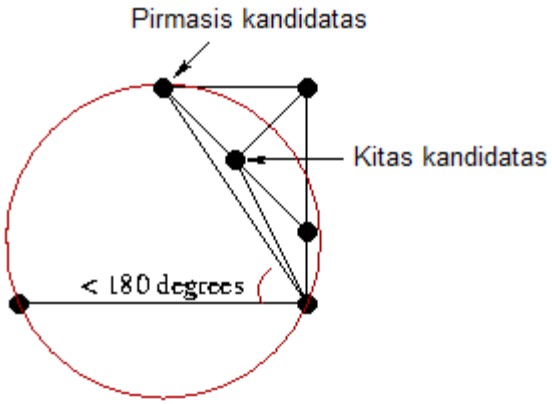
Pradėkime nuo dešinė pusės. Pirmasis kandidatas yra taškas sujungtas su pagrindine KD kraštine per KK kraštinę, kuris nusako mažiausią kampą einant pagal laikrodžio rodyklę nuo pagrindinės KD kraštinės. Einant toliau, kitas potencialus kandidatas nusako kita mažiausią kampą.



2.8 pav. Pagrindinė KD kraštinė

Toliau šie kandidatai tikrinami pagal:

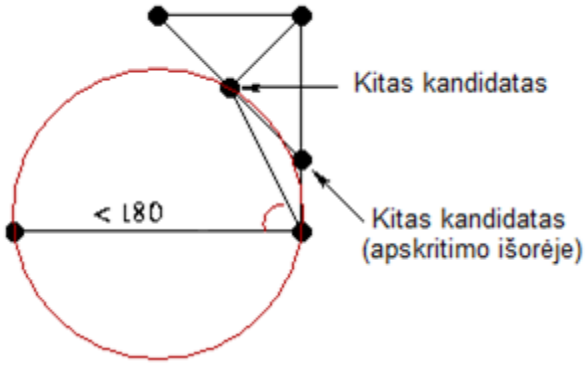
- Kampas, pagal laikrodžio rodykle nuo pagrindines KD kraštinės iki kandidato turi būti mažesnis už 180 laipsnių;
- Apskritimas einantis per pagrindinės kraštinės taškus bei kandidatą, neturi talpinti kito kandidato viduje.



2.9 pav. Pirmasis kandidatas

Kaip matome iš 2.9 paveikslo pirmasis kandidatas tenkina pirmą sąlygą, tačiau netenkina antrosios.

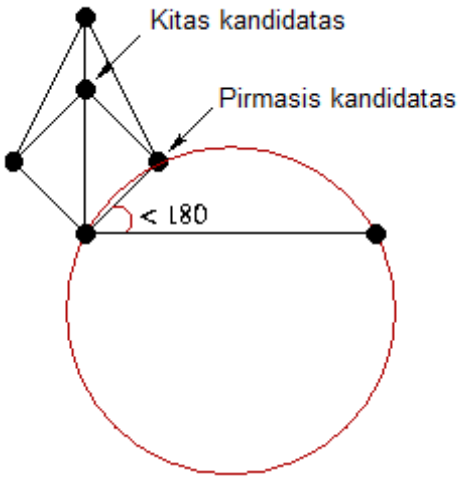
Jeigu kandidatas tenkina abi sąlygas, šis kandidatas tampa mūsų galutinis kandidatus iš dešiniosios bloko pusės. Jei pirma sąlyga nėra tenkinama iš dešiniosios pusės, tada galutinis kandidatas nėra pasirenkamas visai. Jei pirmas kriterijus tenkinamas, bet netenkinamas antrasis, tada KK kraštinė nuo potencialaus kandidato iki dešinės pagrindinės KD kraštinės yra ištrinama. Šis procesas yra kartojamas su kitu potencialiu kandidatu, kol taškas yra išsirenkamas galutiniu arba nusprendžiama, kad galutinis taškas nebus pasirinktas.



2.10 pav. Galutinis dešinės pusės kandidatas

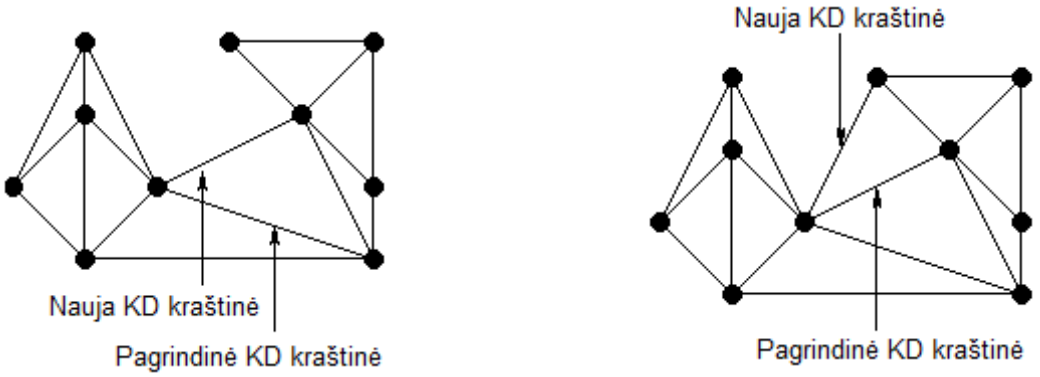
Dešiniojoje pusėje ištrynus KK kraštinę antrasis taškas tenkina abu kriterijus, ir jis yra pasirenkamas kaip galutinis (2.10 pav).

Kairiojoje pusėje algoritmą vykdome taip pat(veidrodiniu principu nei dešiniojoje). Rastas galutinis kandidatas pavaizduotas 2.11 paveiksle.

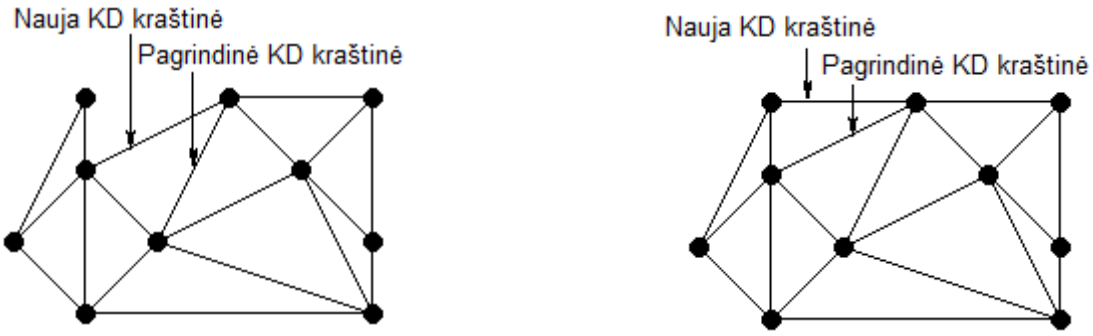


2.11 pav. Galutinis kairės pusės kandidatas

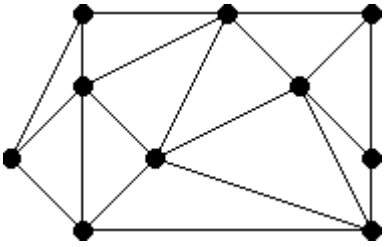
Jei nerandamas nei dešinys, nei kairys kandidatas, tai reškia kad sujungimas baigtas. Jei randamas tik vienas kandidatas, jis automatiškai prijungiamas į trianguliacijos tinklelį. Jei randami abu kandidatai, tada atliekamas paprastas testas: brėžiami apskritimai, per pagrindine KD kraštinę, ir žiūrima, kuris apskritimas netalpina savo viduje kito kandidato. Tas kuris netalpina, tas ir yra pasirenkamas. Kadangi Delone trianguliacija yra unikali, bent vienas kandidatas tenkins šią sąlyga(išskyrus atvejus, kai 4 taškai yra ant apskritimo kraštinės).



2.12 pav. Atrinkimo 2-3 žingsniai



2.13 pav. Atrinkimo 4-5 žingsniai



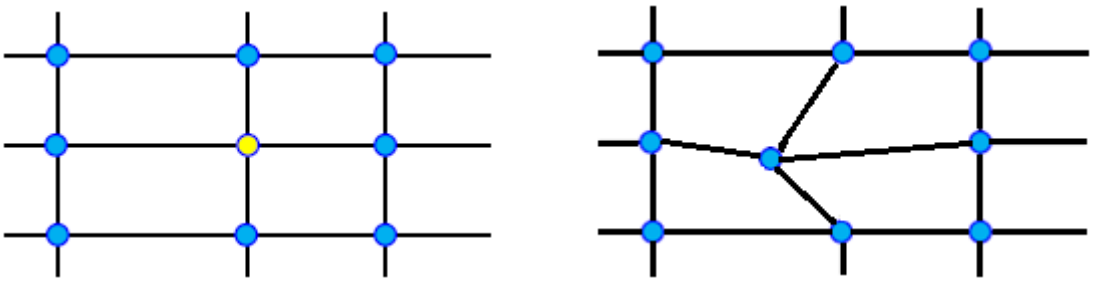
2.14 pav. Galutinė Delone trianguliacija

Paveikslėliuose 2.12 ir 2.13 pateikiami žingsniai, kaip toliau vyksta trianguliacija, bei galutinis variantas (2.14pav).

2.2.2 Reguliarus tinklelis

Tinkleliai vadinami reguliariais jei visi elementai turi toki patį kampų skaičių ir visos viršūnės turi tokį pati kampų skaičių[7].

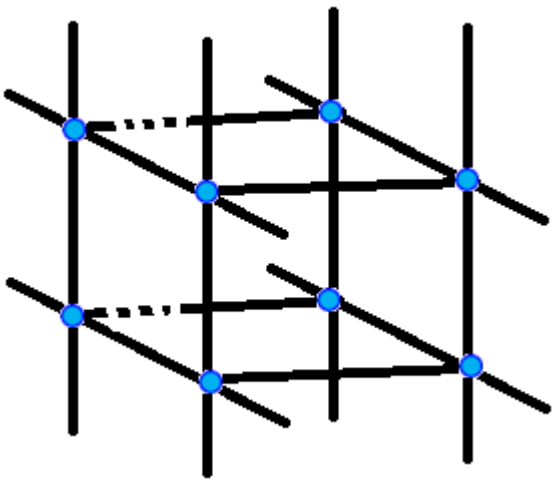
Pavyzdžiui kiekvienas elementas iš reguliaraus keturkampio tinklelio yra keturšonis ir visos viršūnės turi keturis kampus(2.15 pav. a). Tinklelio elementai nebūtinai turi būti taisyklingi (2.15 pav. b)



a b

2.15 pav. Keturkampis reguliarus tinklelis

Panašiai atrodo ir trijų dimensijų tinklelis



2.16 pav. 3d reguliarus tinklelis.

Apdorojant reguliarius tinklelius, skaičiavimai yra greiti, o sudarymo laikas yra tiesiškas.

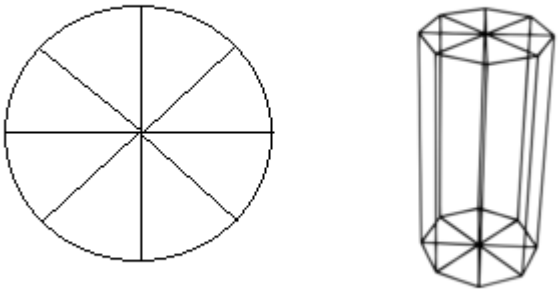
2.2.3 Projekcinis tinklelis

Projekciniai tinkleliai gali būti generuojami trijų dimensijų vaizdą projektuojant į dviejų dimensijų. Pavyzdžiui cilindras gali būti apibrėžtas, kaip apskritimas su nuorodom į tris dimensijas. Tiksliau sakant, taškas veda į eilę sektorių, o sektorius į sugeneruoto tinklelio grupę. Metodo efektyvumas priklauso nuo vartotojo gabumų pritaikant rekomenduotą tinklelį.

Dirbant su trimis dimensijomis, naudoti dviejų dimensijų rekomendacinį tinklelį tikslas yra suteikti modelį(raštą), kuriuo naudojantis galima sugeneruoti baigtinį 3d tinklelį. Sluoksnių skaičius, ir jų pozicijos(pvz. mazgų koordinatės išilgai rekomendacinės linijos) gali būti perduodamos netiesiogiai (kai duotas linijos diskretizavimas) arba tiesiogiai(naudojantis besitempiančia funkcija). Tada rekomenduojamas tinklelis sugeneruojamas palei duotą kryptį ir sukuria norimą skaičių trijų dimensijų elementų sluoksnius tarp apatinės ir viršutinės ribos. Remiantis dviejų dimensijų elementais(trikampis, keturkampis) sugeneruojami trijų dimensijų elementai(tetraedrai, prizmės)[7,8].

Pagrindinis šio metodo trūkumas yra tas, kad galimi išsigimę elementai, pavyzdžiui atvejuose, kai srities riba sutampa su ašies riba.

Šiame pavyzdyje parodytas projekcinio tinklelio generavimo pavyzdys. Dėl paprastumo vaizduojamas tik vienas generavimo sluoksnis



2.17 pav. Dimensijų projekcija

2.2.4 Plintančio fronto metodas

Plintančio fronto trianguliacijos algoritmas yra vienas iš galingiausių tarp paviršių „auginimo“ metodų. Jis paremtas paviršiaus auginimu judinant jo ribines kraštinės kreives iki kol viso objekto geometrija ir topologija yra apdirbama.

Klasikiniai plintančio fronto požiūriai pradeda nuo trianguliuojamos srities ribų diskretizacijos į kampų rinkinį dviejuose dimensijose arba trikampių elementų rinkinį trijuose dimensijose. Kaip pasufleruoja metodo vardas, šio metodo strategija susideda iš tinklelio kūrimo paeiliui- elementą po elemento. Sukūrus vieną elementą, sukuriami nauji taškai ir sujungiami su prieš tai sukurtais elementais, taip žygiuojant per dar nesutrianguliuotą erdvę „šluojant“ tuščios erdvės frontą. Procesas sustoja, kai frontas yra tuščias – kai erdvė yra visiškai sutrianguliuota. Frontas atskiria jau sutrianguliuotas ir dar tuščias, trianguliuojamos figūros erdves. Frontas gali turėti kelis regionus kuriuose elementai yra sujungiami[7,9].

Viena iš kritinių plintančio fronto metodų ypatybė yra vidinių taškų kriterijus. Šis metodas turėtų duoti rezultatą tokį, kad gauti elementai tenkintų dydžio ir formos kriterijus. Toks dydžio kriterijus ir atitinkamai elementų dydžio paskirstymo funkcija iš anksto apibūdina norimą elementų dydį ir formą.

Pagrindinis plintančio fronto metodų privalumas yra jų euristinė tinklelių generavimo strategija, kuri linkusi sugeneruoti aukštos kokybės elementus ir gražiai surūšiuotus tinklelius. Be to, ne taip kaip kiti automatiniai metodai, šis metodas išsaugo kraštinių vientisumą, nes figūros kraštinių diskretizacija yra pradinis frontas[7] .

Plintančio fronto metodą galima suskirstyti į kelis žingsnius. Kiekviename žingsnyje yra pasirenkama fronto detalė ir sukuriama nauja optimali ašis, kuri įterpiama į trianguliaciją, jei ji suformuoja geros formos tinklelį. Plintančio fronto žingsnius galima apibendrinti šiais[7]:

1. Žingsnis: Parengtinis apibrėžimas
    - Trianguliuojamos figūros ribos duomenų nuskaitymas (pvz. ribos tinklelis *T*)
    - Elementų dydžio paskirstymo funkcijos apibūdinimas(reglamentuotas tinklelio generavimas) arba geriausias tokios funkcijos įmanomas sprendimas.

2. Žingsnis: Fronto *F* inicializacija su *T*

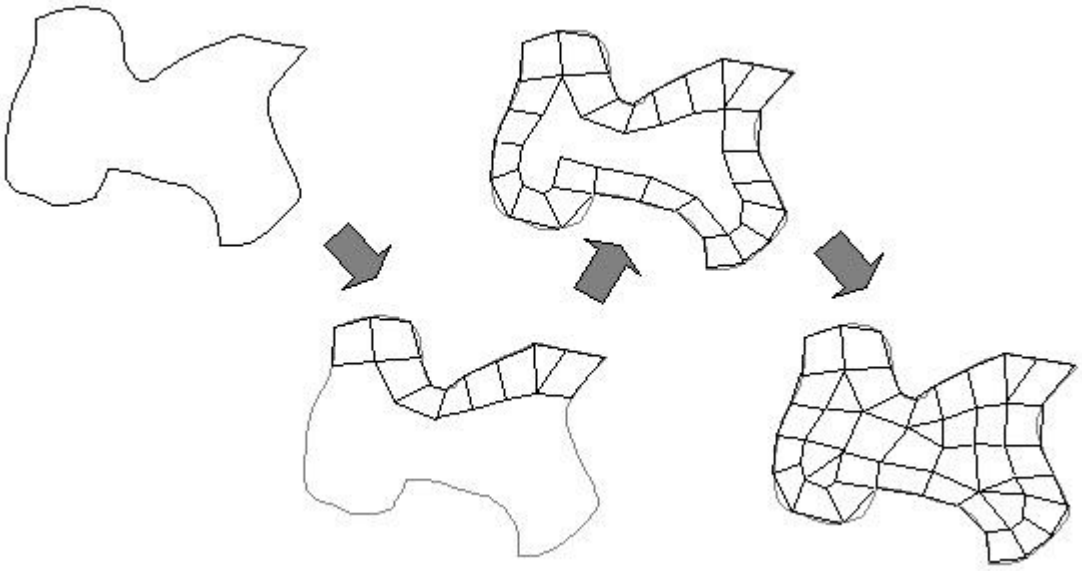
3. Žingsnis: Fronto *F* analizė (tol kol *F* nėra tuščias):
    a. Pasirinkti fronto elementą *f* (pagrįstą apibūdinimo kriterijumi)
    b. Rasti geriausią taškų poziciją P_{opt} šiam elementui
    c. Rasti, ar taškas *P* egzistuoja dabartiniame tinklelyje, kuris turėtų būti panaudotas pirmiau P_{opt}. Jei toks taškas egzistuoja, *P* laikome P_{opt}
    d. Suformuojame elementą *K* su *f* ir P_{opt}
    e. Patikriname ar elementas *K* kertasi su kuriuo nors tinklelio elementu. Jei kertasi, pasirenkam nauja tašką *P* ir grįžtam į 3 d punktą

4. Žingsnis: Atnaujiname frontą ir esamą tinklelį:
    - Pašaliname elementą *f* iš fronto *F*. O *F* elementai naudojami iš *K*
    - Pridedame elementus iš naujai sukurto fronto *K*
    - Atnaujiname esamą tinklelį *T*

5. Žingsnis: Jei frontas F nėra tuščias grįžtame į 3 žingsnį

6. Žingsnis: Tinklelio optimizavimas

Šiame paveiksle galima pamatyti kaip formuojamas tinklelis remiantis žingsniais:



2.18 pav. Plintančio fronto tinklelio generavimas

Plintančio fronto Delone požiūris

Delone paremti metodai laikomi kaip greitesni, nei plintančio fronto stiliaus metodai. Daugiausiai dėlto, kad tinklelis konstruojamas mazgas po mazgo, ir tada kiekvienas taškas leidžia sukurti keletą elementų. Lokaliniu požiūriu, elementų skaičius sukonstruotų iš mazgo yra lokalių situacijų funkcija. Ši funkcija veikia nuo keleto elementų iki kelių dešimčių elementų. Tačiau, kadangi elementai yra sukonstruojami ir pašalinami algoritmo metu, reikalinga labiau tiksli analizė, tam kad įsitikinti, kad bendras efektyvumas susijęs su šiuo mazgu[7,10].

Idėja yra ta, kad plintančio fronto algoritmas automatiškai randa naujus mazgus ir konstruoja kaip elementus tuos, kurie atitinka Delone kriterijų. Taip pasiekiamas didelis šių metodų efektyvumas. Algoritmas daugiausiai sudarytas iš šių trijų konfigūracijų identifikavimo:

- Kažkokia erdvė šalia P_{opt} yra tuščia(šioje erdvėje nėra mazgų), arba ne
- Ši zona yra tuščia, tačiau yra apskritimų (arba sferų), kurios apibrėžia jau egzistuojančio tinklelio elementus, kuris aptveria pasirinktą mazgą
- Ši zona yra tuščia ir joje nėra jokių apskritimų(ar sferų)

Surinkus šiuos duomenis, kiekvienai šiai struktūrai informacija leidžia numatyti lokalaus konteksto pobūdį.

Sujungto metodo efektyvumas susijęs su faktu, kad yra prieinama papildoma informacija, perduodama Delone kriterijaus. Konkrečiai, tai reiškia kad mes automatiškai turime tam tikras žinias apie svarstomos erdvės kaimynus. Kas klasikiniame plintančio fronto metode gali būti surasta tik su milžiniškomis skaičiavimo pastangomis ir per reliatyviai sudėtingas duomenų struktūras. Šio metodo tvirtumas taip pat pagerinamas, nes Delone kriterijus leidžia numatyti ir išvengti potencialių susidūrimų tarp dviejų frontų. Taip pat leidžia naudoti kriterijų sprendžiant konfliktinių elementų trynimą ir galiausiai sukurti keletą elementų vienu metu. Be to, Delone kriterijus, jei mazgai yra gerose pozicijose, garantuoja geros kokybės ir formos elementų tinklelį[7,9,10].

2.3 Tinklelio tankumo ir kokybės valdymas.

Tinklelio tankumas parodo elementų dydi tyrinėjamo kūno dydžio atžvilgiu. Tinklelio tankumas nebūtinai turi būti vienodas visame sutrianguliuotame kūne[12]. Kai kuriuose kūno vietose gali būti atliktas tinklelio patobulinimas- jis gali būti arba tankesnis, arba retesnis.

Tinklelių generatoriai gamina tokias trianguliacijas, kurios bando patenkinti tris tikslus[11,15]:

- Elementų sąjunga yra trianguliacijos sritis, o trianguliacija paiso segmentų- kiekvienas segmentas yra trianguliacijos kampų sąjunga.

- Elementai (trikampiai, keturkampiai ir t.t) santykinai turėtų būti apvalūs savo forma. Todėl, kad pavyzdžiui trikampiai su dideliais arba mažais kampais gali mažinti skaitinę baigtinio elemento kokybę. Atliekant interpoliaciją, trikampiai su dideliais kampais gali sukelti dideles klaidas interpoliuojamo paviršiaus gradientuose. Baigtinių elementų metode dideli kampai gali sukelti diskretizacijos klaidų, sprendimas gali būti mažiau tikslus, nei metodas sugeneruotų paprastai. Maži kampai gali lemti „susietų sistemų“ algebrines lygtis, kurias baigtinio elemento metodas laiko kaip prastos sąlygos

  Trianguliacijos mažiausio kampo žemiausia riba netiesiogiai apriboja didžiausią kampą. Jei nėra tokio kampo, kuris būtų mažesnis nei φ, tai nėra tokio kampo, kuri būtų didesnis nei 180°-2φ. Todėl daug tinklelio generavimo algoritmų, tai pat ir Delone perdirbimo(valdymo) algoritmų remiasi šiuo santykiu ir bando apriboti mažiausią kampą.

- Trečiasis tikslas yra kaip įmanoma labiau reguliuoti elementų (trikampių, keturkampiu ir t.t) dydžius tinklelyje. Kai kurie trianguliavimo algoritmai sugeneruoja tik vienodus tinklelius, kuriuose visi elementai yra maždaug tokio pačio dydžio. Kiti algoritmai siūlo spartaus rūšiavimo galimybę- leidžia elementams kisti nuo didelio iki mažo per sąlyginai trumpą atstumą. Maži, tankiai supakuoti elementai yra geresnio tikslumo, nei dideli retai išdėstyti. Tačiau generavimo(skaičiavimo) laikas per kurį sprendžiamas uždavinys yra proporcingas elementų skaičiui. Taigi renkantis elementų dydį reiškia iškeisti generavimo greitį. Baigtinių elementų metode elementų dydis, kuris yra reikalaujamas priklauso nuo fizikinio kūno elgesio, ir gali kisti per visą kūno sritį.

Turint grubų tinkleli, su sąlyginai mažai trikampių, jį nėra sudėtinga perdirbti į kita tinklelį, kuris turėtų didesnį kiekį mažesniu trikampių. Atvirkštinis procesas nėra toks lengvas. Taigi tinklelių generavimo algoritmai dažnai nusistato toki tikslą, kad iš principo sugeneruotų tokius tinklelius, kad būtų kuo mažiau trikampių. Paprastai, tokie algoritmai siūlo galimybę perdirbti tam tikrą sritį į tankesnę (tokios kokios reikia), jei tos srities elementai nėra pakankamai maži. Kartais figūros forma ir reikalaujamas tankumas gerai kokybei išgauti gali pareikalauti smulkesnių elementų, nei numatyta tam tikrose tinklelio srityse. Trys paminėti tikslai kartais gali prieštarauti vienas kitam[13].

2.3.1 Delone kokybės valdymas

Delone kokybes valdymo(perdirbimo) technika generuojant trikampius tinklelius yra tinkama interpoliavimo naudojimui ir baigtinių elementų metodui. Didžiausia problema yra surasti tokį tinklelį, kuris padengia nurodytą sritį ir yra sudarytas tik iš trikampių, kurių formos ir dydžiai patenkina šiuos suvaržymus: kampai neturi būti nei per dideli, nei per maži ir trikampiai neturėtų būti per daug maži nei būtina ir ne didesni, nei norima. Delone perdirbimo algoritmai siūlo matematines garantijas, kad šie suvaržymai gali būti patenkinti. Jie taip pat puikiai dirba ir praktiškai[11,12].

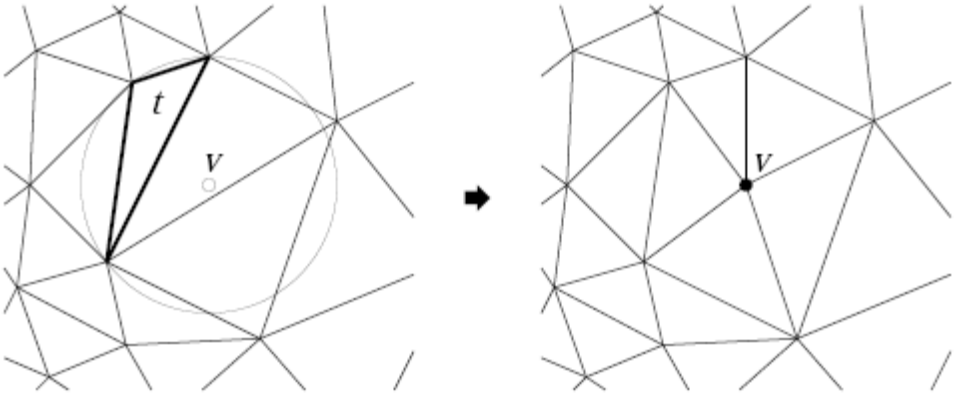
Delone perdirbimo algoritmai veikia išlaikydami Delone trianguliaciją, kuri perdirbama įterpiant atsargiai padėtas viršūnes, iki tol kol tinklelis patenkina numatytus trikampio kokybes ir dydžio parametrus. Pati svarbiausia Delone trianguliacijos savybė yra „tuščia apibrėžto apskritimo savybė“. Apskritimo apibrėžtas trikampis yra unikalus apskritimas, kuris eina per tris trikampio viršūnes. Taškų rinkinio Delone trianguliacija yra tokia trianguliacija (dažniausiai, bet ne visada, unikali), kurioje kiekvienas trikampis turi tuščią apibrėžtą apskritimą, tai reiškia, kad apskritimo viduje nėra jokios kitos viršūnės(mazgo). Labiausiai natūralus ir geriausiai tinkantis matas analizuojant Delone perdirbimo algoritmus yra apibrėžto apskritimo spindulys iki arčiausios trikampio(arba ketursienio) briaunos santykis. Trikampio apibrėžto kampo spindulio ir jo ilgio iki artimiausios briaunos dalmuo yra toks santykis, kuris natūraliai pagerėja su Delone perdirbimo algoritmais. Kuo šis santykis yra mažesnis, tuo geresnė trianguliacija[12,13,14].

Optimizuojant šį santykį tai atsiliepia praktiškiems uždaviniams. Trikampio apibrėžto apskritimo spindulys iki artimiausios briaunos santykis r/l yra susijęs su savo mažiausiu kampu φ_{min} formule:

$$\frac{r}{l} = \frac{1}{2\sin\varphi_{min}} \tag{27}$$

Kuo mažesnis trikampio santykis, tuo didesnis jo mažiausias kampas. Jei *B* yra viršutinė šio santykio visų tinklelio trikampių riba, tada nėra mažesnio kampo nei $\arcsin\frac{1}{2B}$ (ir atvirkščiai). Trianguliuojant figūra, *B* yra protinga daryti kuo mažesniu[15].

Tinklelyje (pav. 2.19) pavaizduotas tinklelio kokybės gerinimo rezultatas. Į bet kurį trikampį, kurio santykis didesnis, nei nustatyta riba *B* yra suskaidomas įterpiant mazgą jo apibrėžto apskritimo centre. Delone savybės yra išlaikomos, o blogos kokybės trikampis yra panaikinamas. Kiekviena nauja briauna tenkina pasirinktą *B* kriterijų.



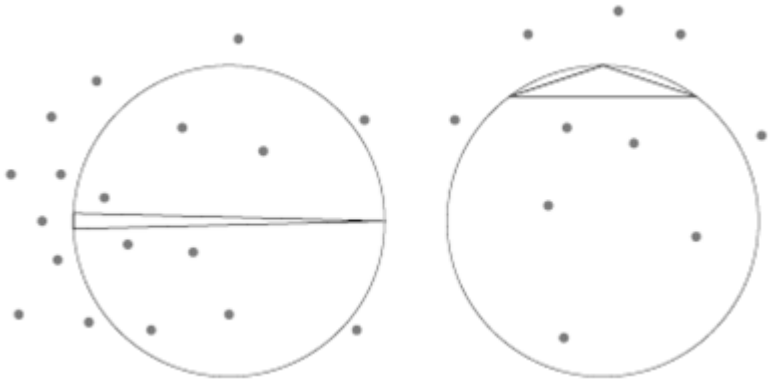
2.19 pav. Tinklelio perdribimas

Pagrindinis Delone perdirbimo algoritmo supratimas yra tas, kad perdirbimo ciklas garantuoja blogos kokybės trikampių ištrynimą, jei jie yra „blogi“ tik dėl šio kriterijaus, kuris yra nustatyta riba *B*. Vienintelės naujos briaunos sukurtos Delone viršūnės *V* įterpimo yra briaunos susijungusios su *V* (2.19 pav.). Todėl, kad *V* yra tam tikro Delone trikampio *t* apibrėžto trikampio apskritimo centras ir šio apskritimo viduje nebuvo jokių kitų viršūnių prieš įterpiant *V*, tai nė viena nauja briauna negali būti trumpesnė už apibrėžto apskritimo *t* spindulį. Todėl, kad *t* turi apibrėžto apskritimo iki artimiausios briaunos santykį didesnį nei *B*, tai kiekviena nauja briauna turi bent jau *B* kartų ilgesnė nei trumpiausia *t* briauna[15].

2.3.2 Ruperto ir Chew požiūriai

Ruperto algoritme riba $B = \sqrt{2}$, o Chew antrajame Delone perdirbimo algoritme $B = 1$. Chew pirmajame Delone perdirbimo algoritme perskiria bet kuri trikampį, kurio apibrėžto apskritimo spindulys yra didesnis nei trumpiausia briauna visame tinklelyje, taip pasiekdamas $B = 1$ ribą, bet priversdavo visus trikampius turėti vienodą dydį[15]. Su tokiom ribom, kiekviena naujai sukurta briauna yra bent jau tokio pačio ilgio, kokia jau yra sugeneruotame tinklelyje. Taigi , jokia viršūnė nėra įterpiama arčiau kitos viršūnės, nei trumpiausios tinklelio briaunos ilgis. Delone perdirbimo algoritmas galiausiai turi pabaigti darbą, nes vis turtiname tinklelyje nebelieka vietos, kur įterpti naujas viršūnes. Kai algoritmas pasibaigia visi kampai yra tarp $20.7° \; ir \; 138.6°$ Ruperto algoritme ir tarp $20.7° \; ir \; 138.6°$ Chew algoritme[15].

Taigi trikampis, kurio apibrėžto apskritimo spindulio iki artimiausios briaunos santykis yra didesnis nei B vadinamas liesu. Paveikslas 2.20 parodo, kodėl visi liesi trikampiai yra galiausiai panaikinami Delone perdirbimo algoritmo. Naujos viršūnės(pilki taškai), kurios yra įterpiamos į trianguliaciją yra išdėstytos grubiai pagal ilgi iki trumpiausios artimiausios briaunos. Todėl kad liesi trikampiai turi palyginti didelį spindulį, jų apibrėžti apskritimai neišvengiamai iššokę. Kai įterpiama pakankamai viršūnių, taip kad jų išsidėstymas maždaug vienodas, dideli tušti apibrėžti apskritimai negali susijungti su mažais kampais. Todėl Delone trianguliacijoje negali likti liesų trikampių[15].



2.20 pav. Liesi trikampiai

2.20 pav. Pavaizduoti liesi trikampiai turi didesnius apibrėžtus trikampius, nei jų trumpiausia kraštinė. Kiekvienas liesas trikampis gali būti klasifikuotas, kaip adata, kurios ilgiausia briauna yra žymiai ilgesnė negu jo trumpiausia briauna.

Šios Delone perdirbimo idėjos beveik nepakitusios taikomos ir aukštesnėse dimensijose (pvz. 3 dimensijų). Įsivaizduokime trianguliaciją, kuri neturi ribų, kuri, galbūt, tęsiasi iki begalybės arba, galbūt, ji yra pasikartojančioje erdvėja, kuri apsivynioja palink ties ribomis. Nepaisant dimensijų, Delone perdirbimo algoritmas gali panaikinti visus liesus elementus, kurių B didesnis nei kriterijus, nesukurdamas jokių briaunų trumpesnių nei tokių kokios jau yra tinklelyje[14,15].

2.4 Tinklelių generavimo programinės įrangos sistemos

2.4.1 ANSYS sistemos

ANSYS yra bendros paskirties baigtinių elementų analizės programinis paketas. Baigtinių elementų analizė yra skaitmeninis metodas, kuris išskaido sudėtingą sistemą į labai mažas daleles- elementus. Ši programinė įranga įgyvendina lygtis, kurios valdo šių elementų elgesį. Sukuria visapusišką paaiškinimą, kaip sistema veikia kaip visuma. Šie rezultatai pateikiami įvairiomis lentelinėmis arba grafiko formomis. Šio tipo analizė paprastai naudojama suprojektuoti ir optimizuoti sistemą, kuri yra per daug sudėtinga atlikti ranka[16].

Remiantis ANSYS terminologija, „modelių generavimas“ reiškia siauresnę sritį- mazgų ir elementų generavimą, kurie atspindi erdvinį tūrį ir sistemos ryšį. Taigi modelių generavimas reiškia modelio mazgų ir elementų geometrinės konfigūracijos procesą[16].

Generuojant modelį galima naudotis dviem skirtingais metodais: vientisu(solid) ir tiesioginiu(direct) [16].

Naudojant vientisą metodą nustatomi parametrai yra:

- modelio geometrinės ribos
- elementų dydis ir forma (ketursienis, šešiasienis)

Turint šiuos parametrus ANSYS sugeneruoja mazgus ir elementus automatiškai.

Naudojantis tiesioginiu metodu parametrai yra:

- kiekvieno mazgo koordinatės
- kiekvieno elemento dydis, forma
- kiekvieno elemento ryšys tarpusavyje

Nors ir galimas nedidelis generavimo automatizavimas, tiesioginis generavimo metodas iš esmės yra „rankinis“ metodas, kuris reikalauja pačiam sekti visus mazgų numerius kuriant baigtini elementų tinklelį. Šis detalus informacijos laikymas gali tapti varginantis dideliems modeliams, bei galima pridaryti daug klaidų. Vientisas modeliavimas paprastai yra galingesnis ir universalesnis, nei tiesioginis ir modeliuojant yra naudojamas dažniausiai. Nepaisant daugelio vientiso modelio privalumų, kartais galima sutikti tokių situacijų, kai prireikia tiesioginio modeliavimo. ASNYS siūlo tokią sistema, kad galima keisti šias modeliavimo sistemas pagal pasirinkimą, tam tikrose pasirinkto kūno srityse[16].

2.4.1.1 Tinklelių generavimas

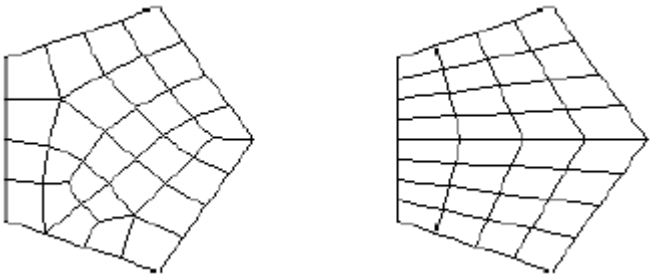
Mazgų ir elementų tinklelio generavimas susideda iš trijų bendrų žingsnių:

- Nustatyti elementų parametrus
- Nustatyti tinklelio parametrus(nebūtinas)
- Generuoti tinklelį

Nustatyti tinklelio parametrus nėra visada būtina, nes ANSYS įrankiai jau turi numatytus, kurie tinkami daugeliui modelių.

Šiuos punktus panagrinėsiu plačiau.

Pirmiausia reikia nuspręsti ar norime laisvojo, ar suvaržyto tinklelio savo figūros analizei. Laisvasis tinklelis neturi elementų formos suvaržymų, taigi neturi ir jokio šablono(rašto). Suvaržyto tinklelio visi elementai turi vienodą formą, bei turi šabloną. Suvaržytas paviršinis tinklelis susideda arba tik iš keturkampių arba tik iš trikampių elementų, o suvaržytas tūrinis tinklelis susideda tik iš šešiasienių elementų. Suvaržytas tinklelis turi aiškų raštą išsidėsčiusį eilėmis(2.21 pav.) [16].



a b

2.21 pav. Laisvasis(a) ir suvaržytas(b) tinkleliai

Nusprendus kokio tipo tinklelį darysim, reik nustatyti elementų parametrus, tokius kaip:

- Elemento tipas(trikampis, keturkampis ir t.t.)
- Realių konstantų rinkinys(nusako elemento storuma, skerspjūvis ir t.t.)
- Materialūs parametrai(terminis laidumas ir pan.)
- Elementų koordinačių sistema

Įrašant šiuos parametrus sukuriamos specialios lenteles ANSYS sistemoje ir generuojamas tinklelis.

2.4.2 COMSOL Multiphysics

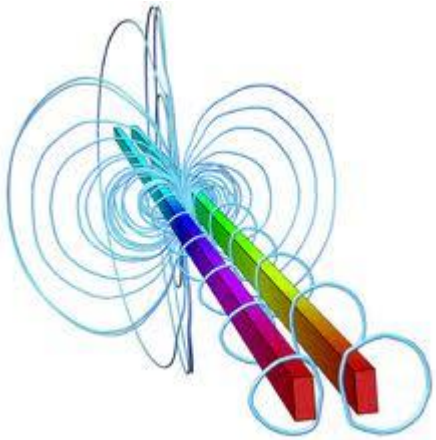
COMSOL Multiphysics sistema leidžia generuoti įvarius tinklelius ir modeliuoti jų elgesį. Patyrę naudotojai gali įvairiai išbandyti modeliuojamo kūno savybes, kurios atspindi realaus pasaulio elgseną[17,18].

Ši programa turi įvairų paketų, kurie suderinami vienas su kitu, taip leidžiant atspindėti realias situacijas. Pavyzdžiui modeliuojant elektros elgseną gali įskaičiuoti ir šilumos poveikį.

COMSOL turi įvairius įrankius modeliavimui, tokius kaip[18]:

- Tinklelių generavimas- geometrinio kūno suskirstymas į mazgus ir elementus (trikampius, dvimatėje erdvėje, ketursienius, trimatėje)
- Tinklelių perdirbimas- metodas leidžia patobulinti tinklelių sujungimą, tam kad pagerinti fizinį figūros elgesį
- LU faktorizavimas- linijinių sistemų lygtims (Gauso eliminavimo versija), kuri pateikia koeficientų matricos A=LU faktorizaciją. L ir U atitinkamai yra viršutinė ir apatinė trikampė matrica. Paketas leidžia lengvai ir greitai išspręsti sistemas su tokių pačių koeficientų matricomis
- Deformuota geometrija- geometrija, kurioje forma keičiasi su judančio tinklelio algoritmu
- Fizinės sąsajos- nustato skirtingų tipų fizines savybes. Šis paketas turi iš anksto numatytas lygtis ir ribos sąlygas, tam kad būtų galima modeliuoti kūnus su tam tikra fizika.
- Kartotinis skaičiuotuvas- sprendžia tiesines lygtis, kurios naudoja pakartotinį metodą. Skaičiuoja vis tikslesnį lygties sprendinį.

Paveiksle parodyta COMSOL Multiphysics magnetinių jėgų moduliacija:



2.22 pav. COMSOL Multiphysics moduliacija

2.4.3 TrueGrid

TrueGrid programa yra komercinės programa, kuri yra daugelio tinklelių generavimo sistemų(programų) pradininkė. Ji naudojama generuoti aukštos kokybės, daug elementų, tiesišką ar kvadratinį(šešiasienės), apvilktą, ar permatomą tinklelį. Taip pat generuoja prizmes, tetraedrus ar trikampius labai tankiai, kaip to reikalauja gera trianguliacija[19].

Generuojant tinklelį naudojamos skirtingos komandos skirtingiems modeliavimas. Šie skirtingi modeliavimai siūlo skirtingas fizines galimybes.

Šios skirtingos savybės TrueGrid skirstomi taip:

1. Materialūs modeliai, būsenos skaičiavimai, kvadratūra ir elementų tipai- kiekvienas modeliavimo būdas turi skirtinus medžiagos tipus. Visi jie turi izotropinės plastinės medžiagos. Kiekviena būsena pralaidumą apibūdina skirtingai. Elemento tipas ir jo būsenos lygtis yra pasirenkami[19].
2. Analizavimo parametrai, tai - laiko žingsnis, iteracijų skaičius, duomenų parametrai, sprendimo metodai ir kiti bendri parametrai
3. Elemento skerspjūvio parametrai. Šie parametrai apibūdinami medžiagos modelio aprašyme. Įvairūs apvalkalo storiai nėra palaikomi. Briaunos storiai yra palaikomi ir gali kisti įvairiai. Šie briaunų parametrai susiejami su medžiagos apibūdinimu arba, komandomis, kurios apibūdina briaunas.
4. Susiduriantys paviršiai. Metodai naudojami moduliacijai. Su tam skirta komanda susiduriančių paviršių parametrai gali būti pasirenkami. Vienas iš bendrų tokių paviršių parametrų yra sujungti neparandant daug tikslumo. Su tam tikra komanda dvi tos pačios trianguliacijos dalys gali būti „suklijuojamos“
5. Spyruoklės, slopintuvai- materialūs parametrai gali kisti tarp savybių. Šias savybes galima pasirinkti parametruose, nurodant jų laisves kampus ir kitas savybes.

2.5 Analizės išvados

Baigtinis elementas yra kertinė tinklelių generavimo dalis. Baigtinį elementą sudaro elementarios geometrinės figūros jų forma ir mazgų koordinatės erdvėje.

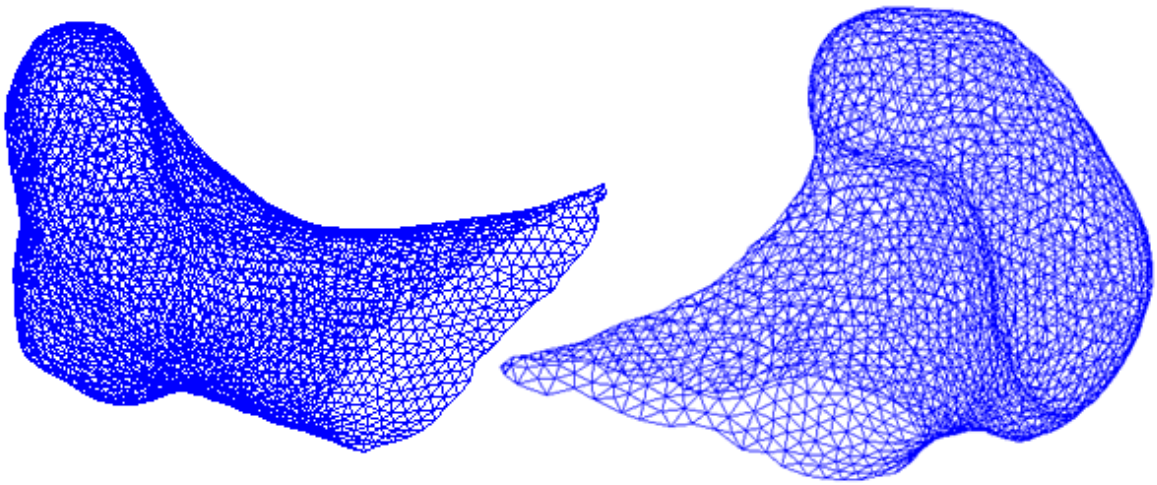
Teoriškai ir praktiškai yra sukurta labai daug įvairių baigtinių elementų tinklelių generavimo algoritmų. Daugelis jų remiasi Delone trianguliacijos kriterijais, kurie leidžia sugeneruoti aukštos kokybės tinklelius. Svarbiausias visų tinklelių(dviejų ar trijų dimensijų) parametras yra mažiausias elemento kampas.

Aptartose sistemose yra įvairių būdų generuoti tinklelius, tačiau konkrečiai tokio būdo, kaip siekiamoje užduotyje nėra- laisvaisiais trianguliuotais paviršiais apribotu erdviniu kūnų tinklelių generavimo. Generuojant trijų dimensijų tinklelius yra labai svarbu išlaikyti elementų forma, kad jie neturėtų labai mažų kampų.

3 Baigtinių elementų tinklelio generavimas

3.1 Geometrinio kūno apribotų plokščiųjų tinklelių generavimo algoritmas

Pradinėje eksperimento stadijoje duota įvairių kūnų paviršinė trianguliacija. Kūnai yra tokie, kaip kairio ir dešinio sąnario galvutės(3.1 pav.)



a b

3.1 pav. Kairio (a) ir dešinio(b) sąnario galvutės

Šioje stadijoje algoritmas iškerpa pasirinktą geometrinį kūną(mano atveju cilindrą) duotuose pradiniuose kūnuose. Programuojama yra MATLAB aplinkoje.

3.1.1 Duomenų nuskaitymas

Programa nuskaito duotus duomenis ir sudaro du masyvus. Viename laikoma visų mazgų tikslios koordinatės, kitame išsaugoma informacija kurie mazgai sudaro elementą(3.2 pav.)

```
mazgai =                          elementai =
    57     54     50                 14.0022   -4.9341   42.6901
    58     57     52                 14.0205   -4.7985   52.3403
    50     54     45                 14.0295   -5.0831   43.1058
    57     50     48                 14.0479   -5.3058   42.9564
    58     52     53                 14.0848   -5.4030   52.2720
```

3.2 pav. Duomenų masyvo pavyzdys

3.1.2 Norimos dalies išrinkimas

Programoje nusistatome figūros centro x, y koordinates. Turėdami šias koordinates, randami mazgai už spindulio ribos naudojantis koordinačių atstumo formule:

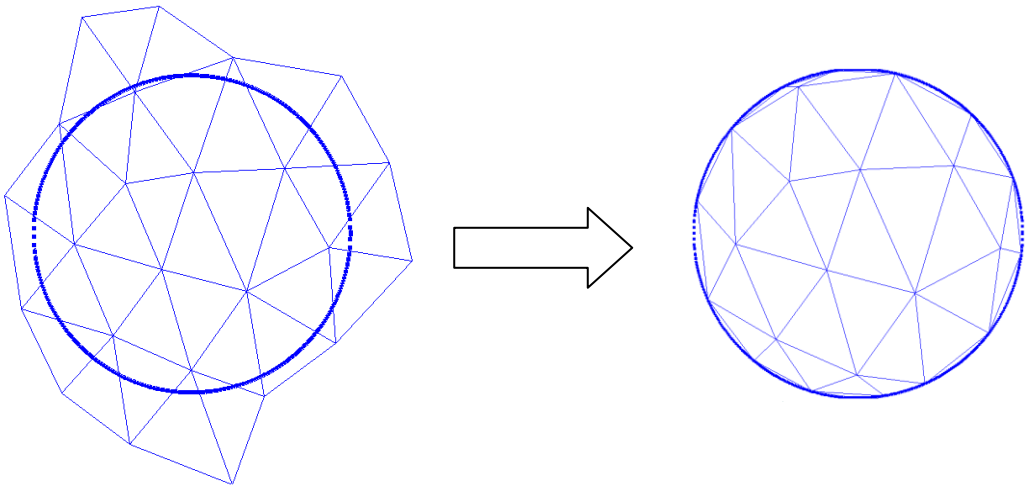
$$l = \sqrt{x^2 + y^2} \tag{28}$$

Mazgai atrenkami šia nelygybe:

$$R < \sqrt{(x_{mazgo} - x_{centro})^2 + (y_{mazgo} - y_{centro})^2} \tag{29}$$

Suradus šiuos mazgai naudojamas i MATLAB funkcija *isamember* ir atmetami mums netinkami(esantys už spindulio) mazgai.

Jei mazgas yra už ribos, bet jo elementas turi kitų taškų apskritime, jis paliekamas. Tokie mazgai, jie yra „pritraukiami“ į cilindrą, kad nesusigadintų bendra cilindro trianguliacija. 3.3 paveiksle galime matyti tokius mazgus, kurie yra pritraukiami.



a b

3.3 pav. Pritraukiami mazgai

Paveiksle 3.3 pavaizduota tinklelio struktūra po ir prieš mazgų pritraukimo. Dalyje a ant duotojo tinklelio pavaizduota ta dalis, kuri bus iškerpama, o ant apskritimo (storesnį linija) bus pritraukiami taškai. Dalis b, tai tinklelis po mazgų pritraukimo.

Mazgai perkeliami ant apskritimo ribos naudojantis skaitinių argumentų trigonometrinėmis funkcijomis. Iš pradžių nustatoma, kuriame ketvirtyje yra mazgas. Tada nustatomi statinių ilgiai ir randama įžambinė. Turėdami visas kraštines, pagal kampą nustatome pritraukto mazgo koordinates.

Pavyzdžiui, mes ieškome taško P_α koordinačių x_α ir y_α (3.4 pav.). Kadangi mazgas yra kažkur pirmajame ketvirtyje, už apskritimo spindulio, ieškome α kampo.

Pirma randame kraštines:

$$OA = x_{mazgo} - x_{centro} \tag{30}$$

$$AB = y_{mazgo} - y_{centro} \tag{31}$$

Pagal Pitagorą:

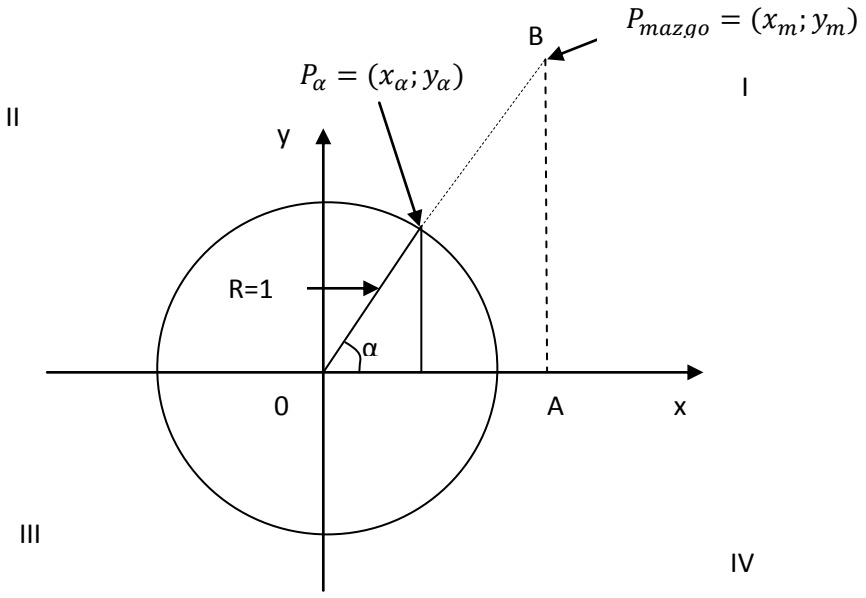
$$OB = \sqrt{AB^2 + OA^2} \tag{32}$$

Turėdami visas tris kraštines galime rasti taško kampą į apskritimo centrą.

$$\alpha = \arccos\frac{OA}{OB} = \arcsin\frac{AB}{OB} \tag{33}$$

Nustatę šį kampą randame vienetinio spindulio ($R = 1$) mazgo P_α koordinates.

$$x_\alpha = \cos\alpha \tag{34}$$
$$y_\alpha = \sin\alpha$$



3.4 pav. Mazgo pritraukimas

Jei mazgas būtų II- IV ketvirčiuose, atitinkamai reikėtų prie kiekvieno pridėti kampą $\frac{\pi}{2}$:

$$x_\alpha = \cos(\alpha + \frac{\pi}{2} * ketvirtis)$$
$$y_\alpha = \sin(\alpha + \frac{\pi}{2} * ketvirtis) \quad (35)$$

Radę P_α mazgo koordinates, sudauginame jį su duotuoju figūros spinduliu (R) gauname pritraukto ant apskritimo ribos mazgo koordinates.

$$X = R * x_\alpha \quad (36)$$
$$Y = R * y_\alpha$$

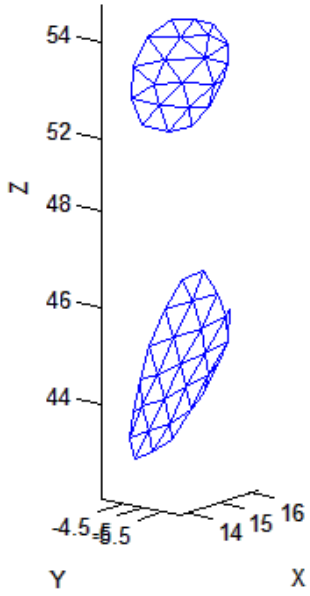
3.1.3 Aštrių kampų atsisakymas

Perkeliant mazgus gali atsirasti netinkamų trikampių, tokie trikampiai pašalinami. Trikampiai atrenkamai pagal jų mažiausio kampo dydį. Jei kampas netenkina minimalio užduoto kampo, jis yra pašalinamas. Trikampio kampai randami naudojantis kosinusų teorema:

$$\alpha = arccos\frac{a^2 + b^2 - c^2}{2ab} \quad (37)$$

Čia a ir b yra kraštinės sudarančios kampą, o c kraštinė priešais kampą

„Atsispjovus“ reikimą figūrą į ekraną išvedamas vaizdas pasitikrinimui(3.5 pav.)

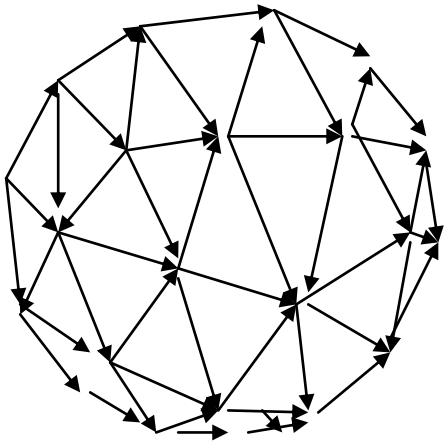


3.5 pav. Išpjauta figūra

3.1.4 Geometrinio kūno apribotų erdvinių kūnų tinklelių generavimo algoritmas

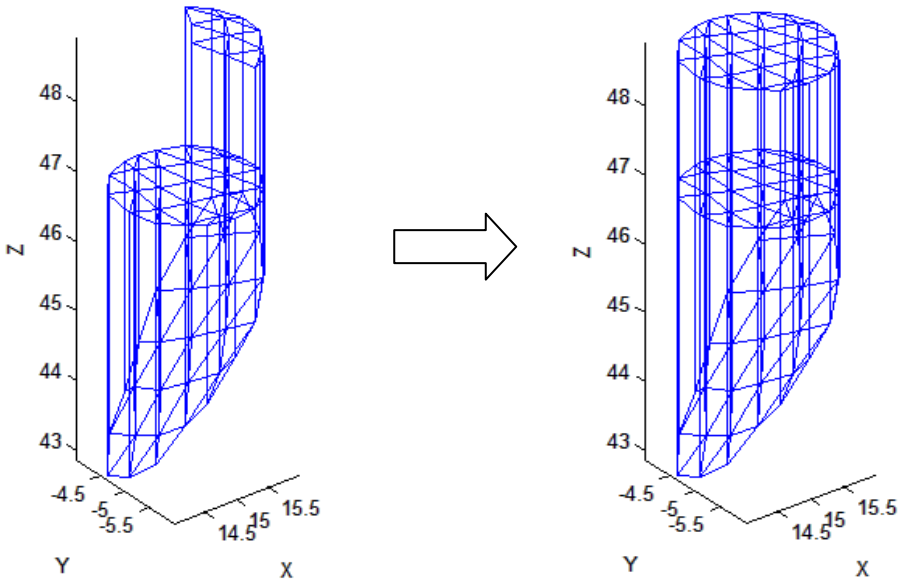
Turėdami paviršinę elementų trianguliaciją, generuojame erdvinę (trijų dimensijų) trianguliaciją. Tai darau naudodamas penkiasienius (prizmes). Kadangi elementų masyve neaišku elementai sąlyginai yra „viršutinėje“ ar „apatinėje“ dviejų dimensijų trianguliacijos dalyje, juos atskiriu.

Atskirimo algoritmas iš pradžių paima bet kokį elementą ir susieja jo mazgus su kitais elementais, taip vis augdamas(3.6 pav.). Suradus vieną iš tinklelių, elementus atskiriame ir galiausiai turime dvi skirtingus elementų masyvus. Turėdami masyvus, palyginame jų aukščio koordinates ir nusistatome jų padėtį erdvėje vienas kito atžvilgiu.



3.6 pav. Tinklelio atskyrimas

Atlikę paruošiamuosius skaičiavimus, „iškeliame“ figūras (3.7 pav.). Taip trianguliuodami galime padaryti kiek norima sluoksnių.



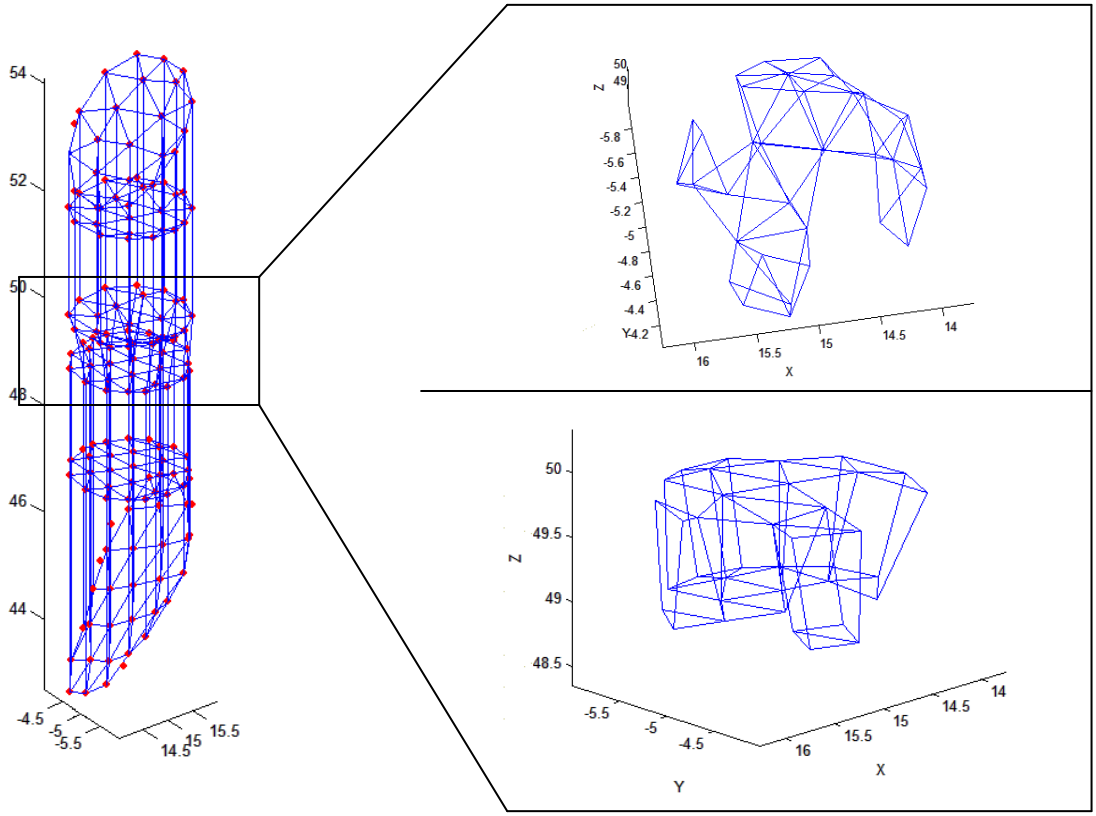
3.7 pav. Erdvins tinklelis

3.2 Geometrinio kūno apribotų erdvinių kūnų tinklelių sujungimo generavimo algoritmas

3.2.1 Elementų sujungimas santykiu 1:1

Šioje stadijoje mes jau turime du trijų dimensijų tinklelius, kuriuos reikia sujungti vienas su kitu. Kadangi elementų skaičius ir „raštas“ nesutampa, neišvengiamai atsiras elementų – trikampių, kurie turės jungtis santykiu 2:1.

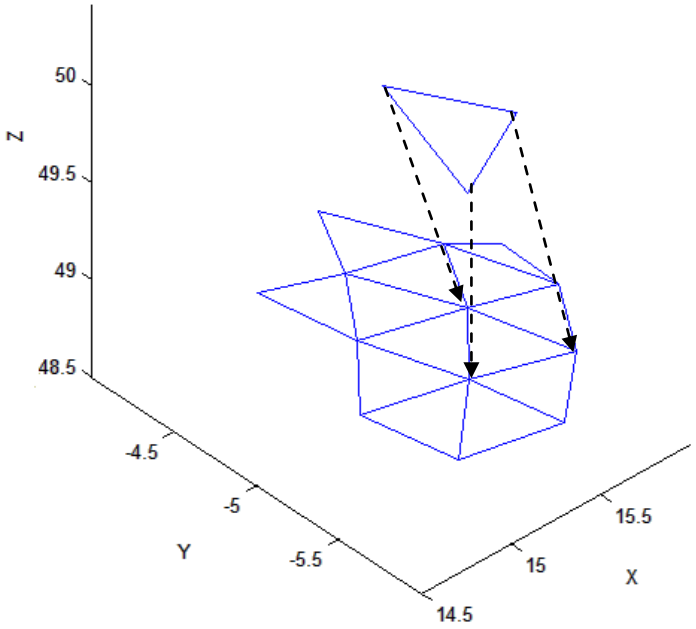
Iš pradžių algoritmas patikrina ar yra gerai sutampančių trikampių vienas su kitu, ir juos sujungia:



3.8 pav. Penkiašonių sujungimas

Siekiant surasti tokius, sąlyginai gerus elementus metodas dirba artimiausių mazgų principu. Tai yra iš pradžių elementui iš masyvo A ieško artimiausio mazgo masyve B pagal koordinates(x, y). Suradę tokį artimiausią mazgą, nustatome kuriems elementams šis mazgas priklauso iš tinklelio B.

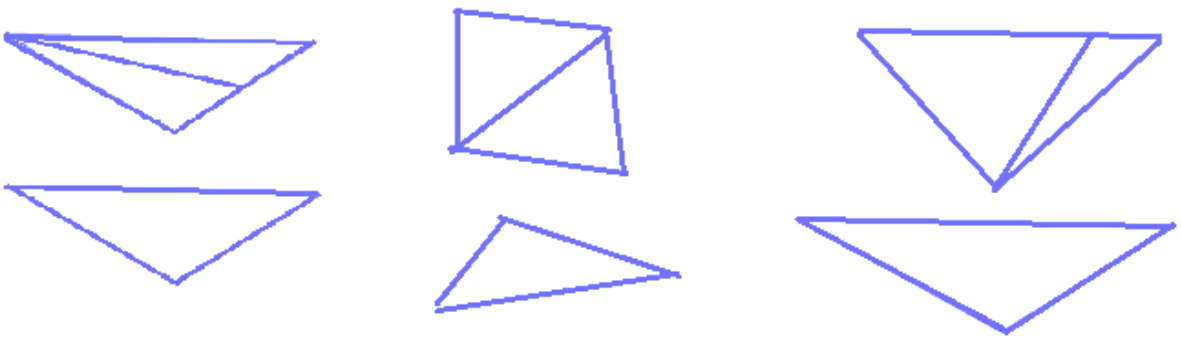
Turėdami artimiausius elementus, tikriname kitus du kampus ir žiūrime(3.9 pav.), kurie iš jų yra arčiausiai, taip surasdami tinkamą elementų porą.



3.9 pav. Penkiasienių sujungimas

3.2.2 Elementų sujungimas santykiu 1:2

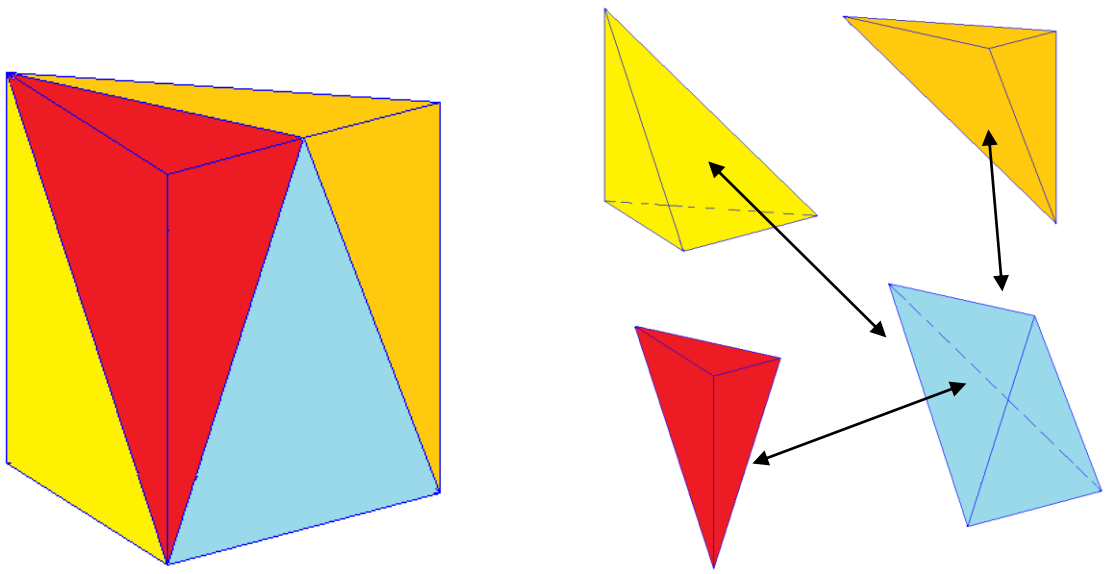
Sujungę gerai sutampančius elementus ieškome kurie du galėtų jungtis su vienu. Rade juos sujungiame ketursieniais elementais. Toks sujungimas yra ganėtinai sudėtingas, nes tinkleliai susiduria labai netolygiai. Šis sujungimo metodas atrenka, kuris iš turimų sujungimų ketursieniais yra geriausias (turi mažiausią kampą). Geriausi atrinkti elementai gali būti labai įvairūs:



3.10 pav. Įvairūs sujungiami elementai

Tokie elementai, kurie geriausiai tinkami sujungimui, randami dar karta nuskanuojant masyvus, be jau gerai tinkančių elementų(sutampančių santykiu 1:1). Suformuojamas kitas masyvas, kuriame kaupiama informacija apie gerai sutampančius masyvus santykiu 2:1, kuris siunčiamas tolimesniam metodui apdoroti ir sujungti ketursienius elementus.

Žemiau pateiktas(3.11 pav.) vienas iš elementų sujungimo išskaidymas. Kairėje (a) pusėje matomi sujunti elementai, kai viršuje yra du, o apačioje vienas. Šiuo atveju elementai yra taisyklingi ir jiems užtenka 4 ketursienių sujungimui. Paprastai reikia 5 ir daugiau ketursienių, nes tikroje trianguliacijoje tokių tobulų formų dažniausiai nebūna. Dešinėje pusėje (b) matomi išrinkti 4 ketursieniai elementai.

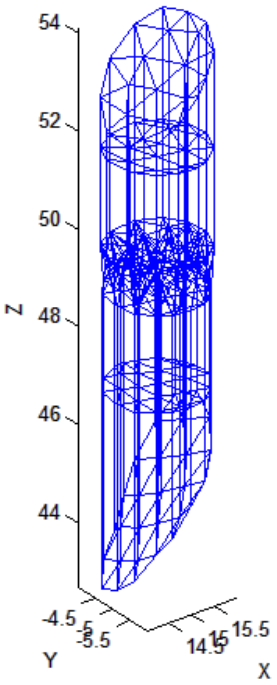


a b

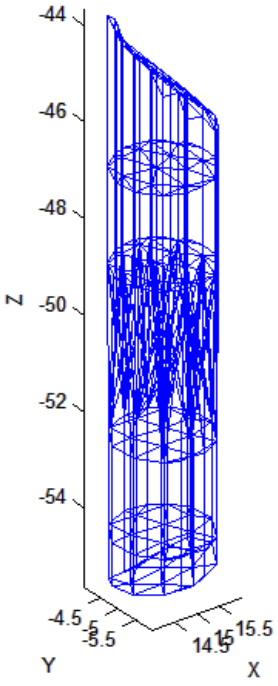
3.11 pav. Elementų sujungimas ketursieniais

4 Sugeneruotų tinklelių kokybės tyrimas

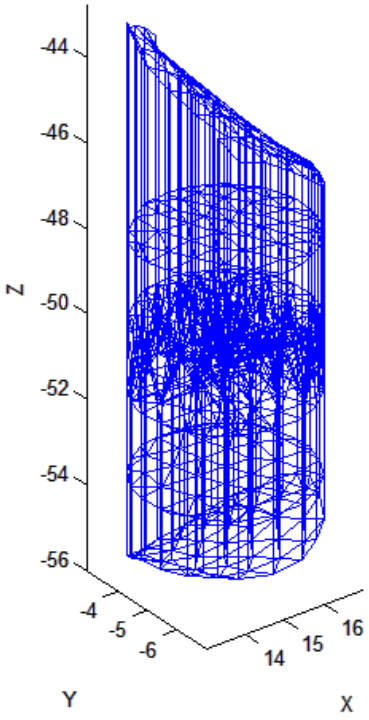
Sukurta programa sugeneruojama keletas skirtinų tinklelių:



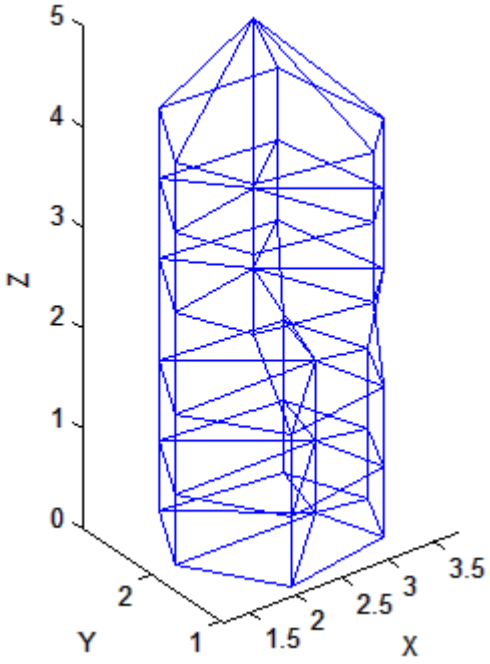
a



b



c



d

4.1 pav. Tinklelių pavyzdžiai

Tinkleliai skiriasi savo padėtimi nagrinėjamojo figūroje, bei sutampančiu raštu skirtiniuose paviršiuose.

a. Figūra (4.1 pav.) suformuota iš dešinio sąnario (3.2 pav. b) duotos paviršinės trianguliacijos. Spindulys R=1.
b. Figūra (4.1 pav.) suformuota iš kairio sąnario (3.2 pav. a) duotos paviršinės trianguliacijos . Spindulys R=1.
c. Figūra (4.1 pav.) suformuota iš kairio sąnario (3.2 pav. a) duotos paviršinės trianguliacijos . Spindulys R=3.
d. Figūra (4.1 pav.) suformuota savarankiškai, norint atkurti sąlygas artimas idealioms.

Sugeneravus apskaičiuojami visų elementų mažiausias kampas. Taip sprendžiama apie tinklelių kokybę, nes vienas mažiausias elementas nulemia viso tinklo kokybę.

Kampai apskaičiuojami naudojantis kosinusų teorema:

$$\alpha = arccos\frac{a^2 + b^2 - c^2}{2ab} \tag{38}$$

Čia a ir b yra kraštinės sudarančios kampą, o c kraštinė priešais kampą.

Kraštinių ilgiai randami naudojantis Pitagoro teorema pagrysta formule:

$$l = \sqrt{x^2 + y^2 + z^2} \tag{39}$$

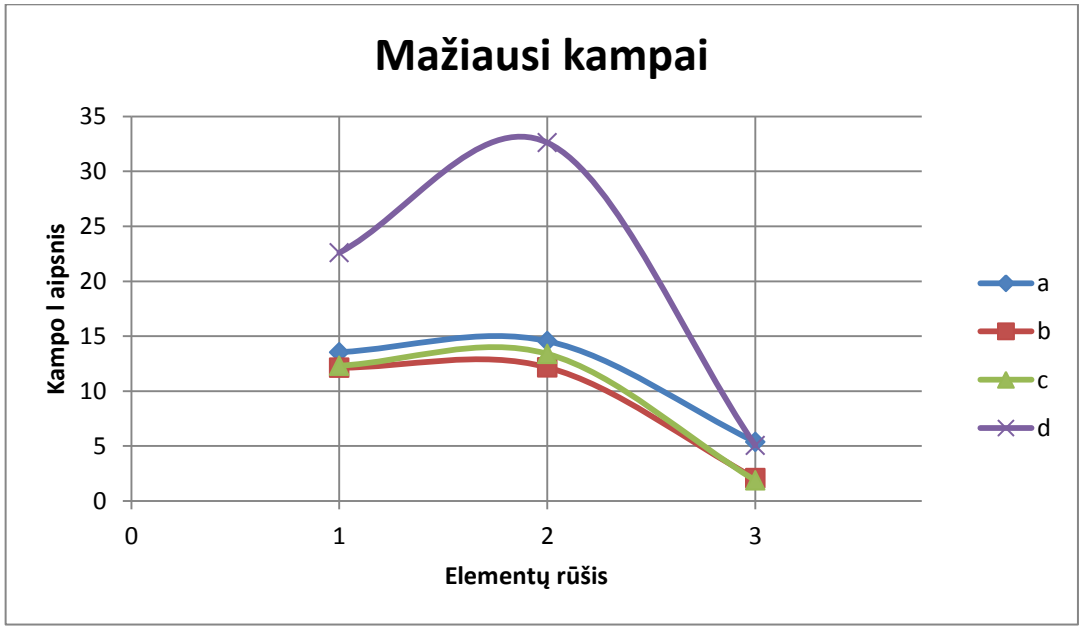
$$l = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \tag{40}$$

Čia l-kraštinės ilgis, o x, y, z atitinkamos koordinatės.

1 lentelė. Kampai

Tinklelis	Mazgų skaičius	Elementų skaičius	Mažiausias kampas iškeltų penkiasienių	Mažiausias kampas sujungtų penkiasienių	Mažiausias kampas ketursienių
a	161	226	13,54	14,55	5,37
b	136	180	12,13	12,16	2,07
c	351	518	12,31	13,39	1,87
d	42	34	22,59	32,61	5,05

Pamatavus visus kampus matoma tendencija, kad ketursienių kampai visada mažiausi, tai yra todėl, kad jie yra jungiami su blogiausiai sutampančiais elementais, taip bandant sujungti visus mazgus, ir nepalikti tuščių tarpų viduje trianguliuojamos kūno. Tokie maži kampai nėra gerai, reikėtų tobulinti algoritmą, kuris jungia santykiu 1:2, arba svarstyti mazgų perkėlimus erdvėje, pasidedant juos patogiau. Mazgų ir elementų skaičius didelės įtakos tinklelio kokybei nepadarė. Grafike 4.2 galime matyti visų keturių figūrų kampų išsidėstymą.



4.2 pav. Mažiausių kampų grafikas

Elementų rūšys:

1. Mažiausias kampas iškeltų penkiasienių
2. Mažiausias kampas sujungtų penkiasienių
3. Mažiausias kampas ketursienių

5 Išvados

- Sukurta programa, kuri iš duotojo kūno, pagal pasirinktas koordinates išpjauna pasirinkto spindulio figūrą. Neiškraipant elementų sudėties, mazgai pritraukiami prie norimos formos. Naudojant penkiasienius užpildoma dalis trianguliuojamos figūros tūrio. Dvi atskiros dalys su skirtingais elementų mazgų kiekiais, bei skirtingu mazgų išsidėstymų erdvėje yra sujungiamos naudojantis penkiasienius ir ketursienius.
- Atliktas tyrimas parodė, kad nors penkiasieniai elementai yra geros formos- jų kampai nėra „adatos“ formos(daugiau nei 10 laipsnių), silpniausia programos dalis yra ketursienių elementų sujungimas, vertinant profesionaliai tokie maži kampai yra netinkami ir netgi kenksmingi, norit atlikti tolimesnius bandymus(šilumos sklidimo ar jėgų veikimo) su tokiu tinkleliu.
- Programą reikėtų tobulint sulyginant penkiasienius elementus iki kol jų visos kraštines butų apylygės (idealus elementas, jeigu įbrėžta sfera liečia visus šonus). Taip pat reikia tobulinti ketursienių sujungimų mechanizmą. Prie programos galima būtų pridėti taškų kokybės gerinimo metodus.

6 Literatūra

1. Farrashkhalvat, M. ; ir Miles, J.P. „Basic Structured Grid Generation“. Great Britain , 2003. ISBN 0 7506 5058 3.
2. Delaunay_triangulation. [žiūrėta 2010-06-10]. Prieiga per Internetą: <http://en.wikipedia.org/wiki/Delaunay_triangulation>
3. Computing constrained Delaunay triangulations, 1998. [žiūrėta 2010-05-20]. Prieiga per Internetą: < http://www.geom.uiuc.edu/~samuelp/del_project.html>
4. Finite element method. [žiūrėta 2010-05-20]. Prieiga per Internetą: http://en.wikipedia.org/wiki/Finite_element_method
5. Simple mesh algorithm. [žiūrėta 2010-11-20]. Prieiga per Internetą: <http://stef2cnrs.wordpress.com/2008/07/27/matlab-m-distmesh-a-simple-mesh-generator-in-matlab/>
6. George, P.-L.; ir Borouchaki, H. „Delaunay Triangulation and Meshing“. HERMES, Paris, 1998. ISBN 2 86601 692 0.
7. Frey, P. J.; ir George, P.-L. „Mesh Generation Application To Finite Elements“. HERMES Science Europe Ltd, 2000
8. Yemez, Y.; ir Sahillioglu, Y. „Pattern Recognition Letters“. Turkey, 2009.
9. Medeiros, E.; Velho, L.; ir Lopes, H. „A Topological Framework for Advancing Front Triangulation“. Brasil, 2004.
10. Pons, J.-P.; ir Boissonnat J.-D. „Delaunay Deformable Models: Topology-Adaptive Meshes Based on the Restricted Delaunay Triangulation“. France, 2007.
11. Bern, M.; ir Eppstein, D. „Meh generation and optimal triangulation“. JAV, 1995.
12. Jurzcyk, T.; ir Glut, B. „Metric 3D Surface Mesh Generation Using Delaunay Criteria“. Lenkija, 2006.
13. Miller, G.L.; Pav, S.E.; ir Walkington, N.J.; „Fully Incremental 3D Delaunay Refinement Mesh Generation“.2003.
14. Ruppert, J. „ A delaunay Refirement algorythm for quality 2-dimensional mesh generation “. JAV, 1994.
15. Shewchuk, J.R.; „Delaunay Refinement Algorithms for Triangular Mesh Generation“. JAV 2001.
16. „Modeling and meshing guide“. JAV, 2009.
17. Pozo, A.; ir Álvarez, J. „Introduction to COMSOL Multiphysics systems“.
18. „Introduction to COMSOL Multiphysics“. JAV, 2010.
19. „TrueGrid® Manual“. JAV, 2001.

Mesh generation algorithms for 3D solid bodies defined by arbitrary triangulated boundaries

Summary

This master's work focuses on the finite elements 3D mesh generation algorithm. First mathematical methods are analyzed found in literature, such as regular, topology based, advancing front and refinement methods. Most of the attention is given for the Delaunay methods, which are best way for computing triangulations. Based on this methods analysis master's work is developed.

This work is quite challenging, because given solid bodies are defined by arbitrary triangulated boundaries. Because bodies are defined arbitrary it is much harder to triangulate good quality mesh. My task is triangulate 3d volume mesh from 3d surface mesh, bound by circle.

Based on the developed system's results we can say that it is triangulating meshes, but in other hand it is poor quality, because some of the smallest angles are 5^o and less.