Jukka Ilari Toivanen

3D-verkon generointi hybridimenetelmällä

Tietotekniikan (tieteellinen laskenta)
pro gradu -tutkielma
14. kesäkuuta 2004

Jyväskylän yliopisto

Tietotekniikan laitos

Jyväskylä

Tekijä: Jukka Ilari Toivanen
Yhteystiedot: Puh. 050 304 1579
Työn nimi: 3D-verkon generointi hybridimenetelmällä
Title in English: 3D hybrid mesh generation
Työ: Tietotekniikan (tieteellinen laskenta) pro gradu -tutkielma
Sivumäärä: 74
Tiivistelmä: Tässä pro gradu -tutkielmassa käsitellään kolmiulotteisen laskentaverkon generointia. Työssä esitellään tyypillisimmät verkon luontiin käytettävät menetelmät ja mainitaan joitakin aiheeseen liittyviä ongelmia. Työssä myös tutkitaan tapoja suorittaa olemassa olevilla verkoilla loogisia operaatioita ja esitellään kahden verkon yhdisteen muodostavan algoritmin toteutusta. Kahden rakenteellisen verkon yhdisteenä syntyy niin sanottu hybridiverkko, joka on rakenteeton ainoastaan verkkojen leikkausalueen läheisyydessä.
English abstract: This thesis provides a survey of 3D mesh generation methods. Some problems involved in this area are also mentioned. Logical operations with existing meshes are studied, and a method for creating a mesh by joining two existing meshes is presented. If the two meshes are structured, the result is a hybrid mesh that has an unstructured part only near the area where the meshes overlap.
Avainsanat: verkon generointi, hybridiverkko, verkoilla operointi
Keywords: mesh generation, hybrid mesh, logical operations with meshes

Copyright © 2004 Jukka Ilari Toivanen

All rights reserved.

Esipuhe

Tämä työ on tehty Numerola Oy:n ideoimasta aiheesta. Yritys myös rahoitti työn ja tarjosi sen tekemiseen tarvittavat tilat ja laitteet, mistä suuret kiitokset. Yritys on jo aiemmin tarjonnut minulle mielenkiintoisia ja haasteellisia kesätöitä, jotka ovat tuoneet minulle arvokasta käytännön kokemusta numeriikan alalta.

Erityiset kiitokset haluan osoittaa tutkielmanani toisena ohjaajana toimineelle FT Kai Hiltuselle paneutumisesta työhöni kaikissa sen vaiheissa. Kiitokset myös toiselle ohjaajalleni professori Raino Mäkiselle ja kaikille muille, jotka ovat antaneet työstäni palautetta.

Sisältö

Esipuhe **i**

1 Johdanto **1**

2 Peruskäsitteitä **3**

2.1 Geometrisia käsitteitä 5

2.2 Verkon laatu 6

3 Verkon generointimenetelmiä **9**

3.1 Manuaalinen generointi 9

3.2 Verkkojen muunnokset 10

3.2.1 Algebralliset muunnokset 11

3.2.2 Osittaisdifferentiaaliyhtälöpohjaiset muunnokset 12

3.3 Multiblokkimenetelmä 14

3.4 Pursotusmenetelmät 14

3.5 Delaunay-tyyppiset menetelmät 18

3.5.1 Delaunayn kolmioinnin ominaisuuksia 18

3.5.2 Delaunay-verkon luominen 19

3.5.3 Reunojen käsittely 22

3.6 Quadtree ja octree 24

3.7 Advancing front 26

4 Verkkojen muokkaus ja parantaminen **28**

4.1 Verkon silotus 28

4.1.1 Topologiset muutokset 28

4.1.2 Laplace-silotus 30

4.1.3 Winslow-silotus 31

4.1.4 Optimointiin perustuva silotus 31

4.2 Tehtävästä riippuva parantaminen 32

5 Verkoilla operointi **34**
5.1 Yhdiste . . . 34
5.1.1 Leikkauspisteiden määrääminen . . . 37
5.1.2 Leikkauskäyrän määrääminen . . . 38
5.1.3 Solmujen tuottaminen leikkauskäyrälle . . . 39
5.1.4 Leikkaavan osan purkaminen . . . 41
5.1.5 Pintaverkkojen yhdistäminen . . . 43
5.1.6 Siirtymä tahkotyyppien välillä . . . 46
5.1.7 Sisäverkon luominen . . . 47
5.2 Muut operaatiot . . . 47
5.2.1 Poisto . . . 47
5.2.2 Leikkaus . . . 48
5.3 Tietorakenteita . . . 49
5.3.1 Verkkotietorakenne . . . 49
5.3.2 R-puu . . . 50

6 Esimerkkejä **52**
6.1 Esimerkki 1: Putki ja kuutio . . . 54
6.2 Esimerkki 2: Kaksi putkea . . . 56
6.3 Esimerkki 3: Putkikulma . . . 58
6.4 Esimerkki 4: Putki ja pallo . . . 60

7 Yhteenveto **62**

8 Viitteet **64**

1 Johdanto

Osittaisdifferentiaaliyhtälöiden avulla voidaan simuloida useita fysikaalisia ilmiöitä, kuten lämmön johtumista, nesteen virtausta ja sähkömagneettista säteilyä. Ilmiön simulointia varten tarvitaan myös malli alueesta, jossa ilmiötä halutaan tarkastella. Tätä mallia kutsutaan laskenta-alueeksi (*engl. domain*).

Yhtälöt eivät useinkaan ratkea suljetussa muodossa, joten ne täytyy ratkaista numeerisesti. Usein käytettyjä ratkaisumenetelmiä ovat elementtimenetelmä (*engl. finite element method, FEM*) ja kontrollitilavuusmenetelmä (*engl. finite volume method, FVM*). Menetelmiä varten laskenta-alue on jaettava pieniin osa-alueisiin eli elementteihin. Elementtijakoa kutsutaan laskentaverkoksi ja jaon muodostamista verkon generoinniksi.

Menetelmät asettavat käytettävälle verkolle tiettyjä vaatimuksia. Menetelmän antaman ratkaisun hyvyys nimittäin riippuu käytetyn laskentaverkon elementtien tyypistä, laadusta ja koosta. Kuusitahokaselementtien käyttö esimerkiksi tuottaa joissakin sovelluksissa parempia tuloksia kuin nelitahokkaiden [6, 49]. Verkon elementtien tulisi myös olla mahdollisimman säännöllisiä, eli niissä ei esimerkiksi saa esiintyä äärimmäisen pieniä kulmia.

Mallien ja laskentamenetelmien kehittyminen on johtanut yhä realistisempiin sovelluksiin ja siten myös mallinnettavien geometrioiden monimutkaistumiseen [21]. Verkon generoinnin automatisoimiseksi on kehitetty useita enemmän ja vähemmän heuristisia menetelmiä. Mielivaltaisen alueen verkottaminen hyvälaatuisella kuusitahokasverkolla voi kuitenkin olla vaikeaa tai jopa mahdotonta. Tämän vuoksi tarvitaan menetelmä, jolla yksinkertaisempiin alueisiin luodut hyvälaatuiset verkot voidaan yhdistää toisiinsa.

Verkon generointialgoritmit tarvitsevat lähtötiedoikseen informaation, joka määrittää verkotettavan alueen reunat. Aina reunoille ei ole käytössä analyyttistä esitystä — informaatio voi olla lähtöisin esimerkiksi mittauksista. Liitoksen tekevän algoritmin tulisi siis pystyä operoimaan myös diskreetissä muodossa esitetyille verkoille [56].

Ei ole olemassa mitään yksikäsitteistä määritelmää sille, mikä tällaisen operaation lopputuloksen tulisi olla. Hyvälaatuiset verkot on kuitenkin järkevää pyrkiä säilyttämään sellaisenaan niin suurelta osin kuin mahdollista. Tämän työn tavoitteeksi asetet-

tiin kehittää verkkojen yhdistämismenetelmä, joka muokkaa verkkoja ainoastaan niiden päällekkäin menevän osan läheisyydessä. Tällainen algoritmi saatiin toteutettua, joskaan toteutus ei ole täysin yleispätevä.

Tämän tutkielman luvussa 2 käydään läpi verkon generointiin liittyviä peruskäsitteitä ja merkintöjä. Luvussa myös kerrotaan tarkemmin verkon laatuun liittyvistä tekijöistä. Luvussa 3 suoritetaan katsaus tyypillisimpiin verkon generointimenetelmiin. Luvussa 4 käsitellään verkon laadun parantamista yleisesti ja tietyn tehtävän kannalta. Luvussa 5 kerrotaan toteutetusta liitosalgoritmista ja hahmotellaan muiden loogisten operaatioiden mahdollisia toteutustapoja. Luvussa 6 algoritmin toimintaa esitellään muutamien testiesimerkkien avulla. Luvussa 7 esitetään yhteenveto tutkielman tuloksista ja ajatuksia algoritmin jatkokehittämiseksi.

2 Peruskäsitteitä

Tässä luvussa käydään läpi verkon generointiin liittyviä peruskäsitteitä ja tutkielmassa käytettäviä merkintöjä. Verkon generoinnissa lähdetään liikkeelle siitä, että on annettu jokin laskenta-alue (*engl. domain*), johon halutaan generoida verkko (*engl. mesh*). Alueesta käytetään seuraavassa symbolia Ω, ja sen dimensioksi oletetaan kaksi tai kolme.

Määritelmä 2.1 Alueen Ω konformi verkko on kokoelma $\mathcal{T}_h$ elementtejä K, joille pätee [21]

- $\overline{\Omega} = \bigcup_{K \in \mathcal{T}_h} K$
- kaikkien elementtien K sisus on epätyhjä
- kahden $\mathcal{T}_h$:n elementin leikkaus on joko tyhjä, yksi solmu, elementin särmä tai elementin tahko.

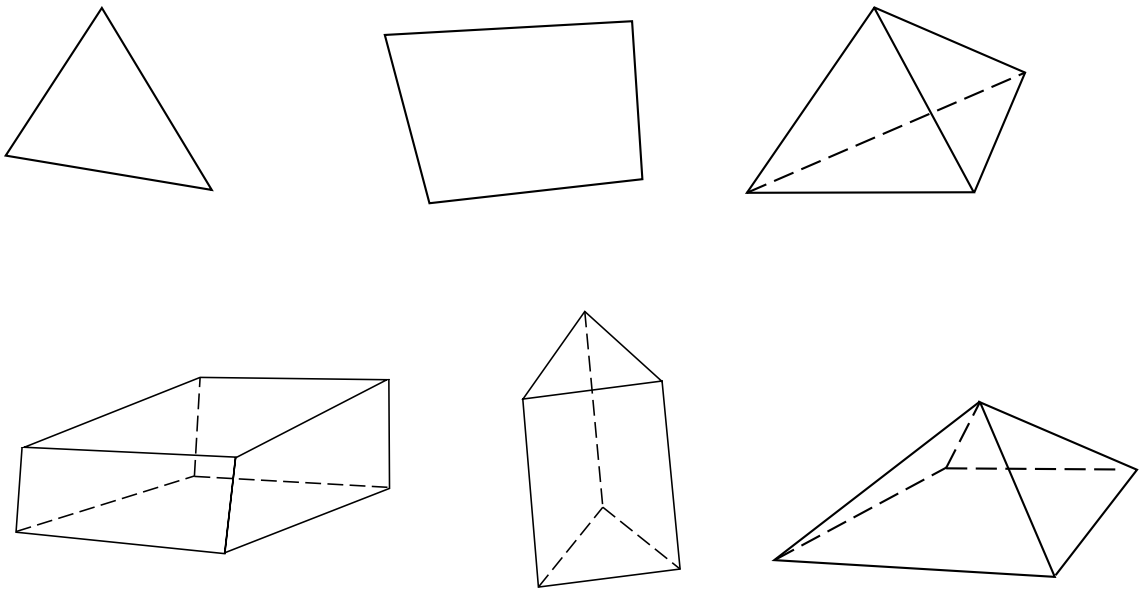
Edellä h on verkon tiheyttä kuvaava parametri. Se voidaan määritellä esimerkiksi verkon pisimmän särmän pituudeksi.

Määritelmän perusteella verkon tulisi siis täsmälleen peittää Ω. Käytännössä tämä ei kuitenkaan välttämättä ole mahdollista, koska esimerkiksi särmiltään lineaarisilla elementeillä ei voida tarkasti tuottaa kaarevia pintoja. Sovelluksissa yleensä kuitenkin riittää, että käytössä on hyvä approksimaatio laskenta-alueesta.

Yleisimmät elementit ovat kahdessa ulottuvuudessa kolmio ja nelikulmio sekä kolmessa ulottuvuudessa nelitahokas, kuusitahokas, prisma ja pyramidi. Esimerkkejä elementeistä on kuvassa 2.1. Elementtien särmät voivat olla myös korkeamman asteen käyriä [68], mutta sellaisia elementtejä ei käsitellä tässä tutkielmassa.

Kolmessa ulottuvuudessa tarvitaan joskus myös pintaverkkoa (*engl. surface mesh*), eli kaksiulotteista verkkoa, joka kuvaa laskenta-alueen pinnan. Pintaverkko muodostetaan yleensä kolmioista ja nelikulmioista. Myöhemmin käytetään merkintää $\partial\mathcal{T}$ pintaverkosta, joka muodostuu verkon $\mathcal{T}$ reunalla sijaitsevista tahkoista.

Tässä tutkielmassa solmuiksi (*engl. node*) kutsutaan elementin geometrian määrääviä pisteitä. Koska käsiteltävien elementtien särmät ovat janoja, solmut ovat käytännössä elementtien kärkipisteitä.

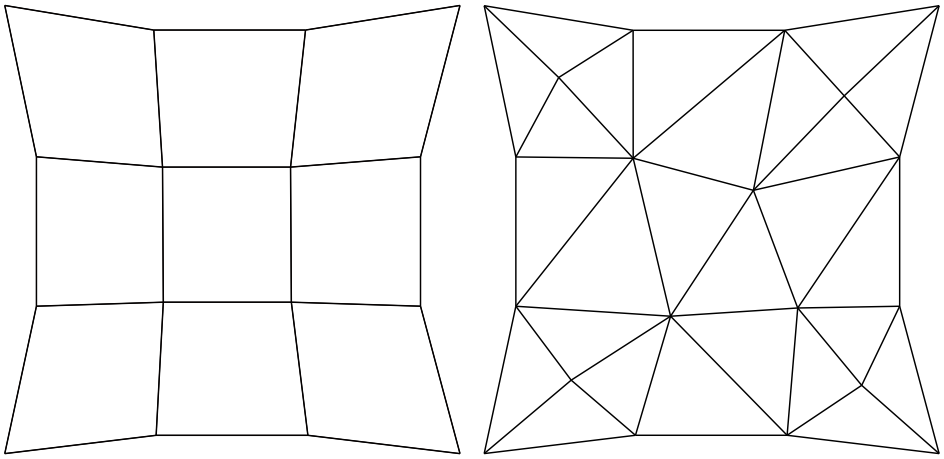


Kuva 2.1: Esimerkkejä elementeistä. Ylärivillä kolmio, nelikulmio ja nelitahokas sekä alarivillä kuusitahokas, prisma ja pyramidi.

Elementin topologia on kuvaus elementistä sen solmujen avulla. Se sisältää kuvauksen elementin särmistä (*engl. edge*) sekä kolmessa ulottuvuudessa tahkoista (*engl. face*). Termiä särmä tullaan käyttämään yhtenäisyyden vuoksi myös kaksiulotteisten monikulmioiden sivuista.

Verkot voidaan yleensä jakaa rakenteellisiin ja rakenteettomiin. Rakenteellisella verkolla on loogisessa mielessä säännöllinen, toistuva rakenne [27]. Sen solmut voidaan indeksoida siten, että kunkin solmun indeksien perusteella tiedetään myös sen naapureiden indeksit [54]. Verkon kaikkiin sisäsolmuihin liittyy sama määrä elementtejä [39]. Yleisesti rakenteellisista verkoista puhuttaessa kuitenkin tarkoitetaan nimenomaan rakenteellista nelikulmio- tai kuusitahokasverkkoa. Rakenteettomassa verkossa taas tällaista säännöllistä rakennetta ei ole, vaan solmuun voi liittyä kuinka monta elementtiä tahansa. Kuvassa 2.2 on esimerkki rakenteellisesta ja rakenteettomasta verkosta. Sekä rakenteellisia että rakenteettomia osia sisältävää verkkoa kutsutaan usein hybridiverkoksi (*engl. hybrid mesh*) ja useasta rakenteellisesta osasta muodostuvaa verkkoa blokeittain rakenteelliseksi (*engl. block-structured*).

Lisäksi on olemassa niin kutsuttuja yleistettyjä verkkoja (*engl. generalized grid*), joissa elementeillä voi olla mielivaltainen määrä särmiä tai tahkoja. Voidaan myös sallia,



Kuva 2.2: Eräs geometria verkotettuna rakenteellisella nelikulmio- ja rakenteettomalla kolmioverkolla.

että eri alueisiin luodut verkot menevät osin päällekkäin. Tällaisia verkkoja kutsutaan usein chimera-verkoiksi [58]. Tässä tutkielmassa ei käsitellä yleistettyjä- eikä chimera-verkkoja.

Verkosta käytetään kirjallisuudessa myös muita nimityksiä. Termiä hila (*engl. grid*) käytettäessä puhutaan useimmiten rakenteellisesta verkosta. Rakenteetonta kolmio- tai nelitahokasverkkoa taas saatetaan kutsua kolmioinniksi (*engl. triangulation*).

2.1 Geometrisia käsitteitä

Seuraavassa määritellään joitakin geometrisia käsitteitä siten kuin niitä tässä tutkielmassa käytetään.

Määritelmä 2.2 Kolmion (nelitahokkaan) ympäröivä pallo (*engl. circumsphere*) kulkee kaikkien sen kärkipisteiden kautta.

Määritelmä 2.3 Kolmion (nelitahokkaan) sisäpallo (*engl. insphere*) on sellainen pallo, että kaikki kolmion särmät (nelitahokkaan tahkot) ovat sen tangentteja.

Määritelmä 2.4 Geometriseen objektiin liittyvä rajoitelaatikko (*engl. bounding box*) on pienin koordinaattiakselien suuntainen suorakulmio, jonka sisään objekti kokonaisuudessaan mahtuu.

Määritelmä 2.5 Elementin keskipiste on sen solmujen geometrinen keskipiste.

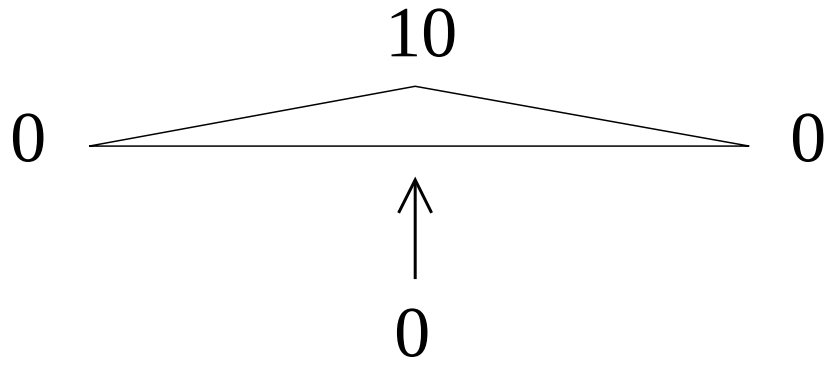
Käytännössä keskipiste siis saadaan laskemalla solmujen koordinaattien keskiarvo.

2.2 Verkon laatu

Verkon laadulle voidaan esittää useita erilaisia kriteereitä. Yleisesti voidaan sanoa, että verkko on laskennan kannalta sitä parempi, mitä lähempänä säännöllistä muotoa sen elementit ovat. Esimerkiksi nelikulmion tulisi olla mahdollisimman lähellä neliötä ja kolmion särmien tulisi olla yhtä pitkät. Joskus voidaan tehtävän luonteen vuoksi kuitenkin haluta venyttää elementtejä tiettyihin suuntiin.

Elementtien laadulla on vaikutusta esimerkiksi saadusta ratkaisusta laskettavien suureiden tarkkuuteen. Shewchuk [55] esittää seuraavantyyppisen esimerkin niin sanotusta interpolaatiovirheestä: Oletetaan, että jollekin suureelle on numeerisesti ratkaistu arvot kaksiulotteisen verkon solmupisteissä. Tarkastellaan yhtä elementtiä, jonka solmupisteissä ratkaisun arvot ovat kuten kuvassa 2.3. Kun ratkaisun arvoja interpoloidaan lineaarisesti kuvan elementissä, antaa interpolaatio kolmion alareunalla arvoksi nolla. Ratkaisu saa kuitenkin kolmion kolmannessa solmupisteessä arvokseen 10. Jos nyt kolmion suurin kulma lähestyy 180°:tta, tämä solmupiste siirtyy mielivaltaisen lähelle kolmion alareunaa. Numeerisen ratkaisun gradientti siis kasvaa elementin alueella rajatta. Näin käy täysin todellisen ratkaisun gradientista riippumatta, vaikka numeerinen ratkaisu olisi solmupisteissä tarkka. Kolmessa ulottuvuudessa vastaavan ongelman aiheuttavat muun muassa kuvan 2.5 kaltaiset elementit.

Yleisesti elementti- ja kontrollitilavuusmenetelmien antamat ratkaisut lähestyvät tarkkaa ratkaisua, kun käytetyn laskentaverkon elementtikokoa pienennetään. Babuška ja Aziz [3] kuitenkin näyttävät, että tämän konvergenssin takaavat tulokset eivät elementtimenetelmän osalta ole lainkaan voimassa, jos kolmioverkossa sallitaan mielivaltaisen lähellä oikokulmaa olevia kulmia.



Kuva 2.3: Tylppäkulmainen kolmio tuottaa ratkaisua interpoloitaessa huonoja arvioita gradienteille.

Elementtien laadulla on vaikutusta myös numeeristen menetelmien toimintaan. Esimerkiksi elementtimenetelmässä huonon muotoiset elementit lisäävät ratkaistavien lineaaristen systeemien häiriöalttiutta. Tässä mielessä ovat vaarallisia sekä pienet että suuret kulmat. Shewchuk [55] toteaa kuitenkin, että elementtien muodon ja matriisin häiriöalttiuden täsmällinen suhde riippuu myös ratkaistavasta tehtävästä.

Hän esittää kolmio- ja nelitahokaselementeille useita erilaisia laatukriteereitä, joista osa mittaa elementtien laatua interpolaatiovirheen ja osa matriisien häiriöalttiuden mielessä. Hän huomauttaa, että erilaiset laatukriteerit saattavat olla keskenään ristiriitaisia, ja että käytettävä laatukriteeri on valittava käyttötarkoituksen mukaan. Verkon elementtejä voidaan myös mitata yhtä aikaa useilla erilaisilla mittareilla ja pyrkiä näin hyvään kompromissiin.

Numeerisia mittareita elementtien laadulle tarvitaan erityisesti, jos halutaan käyttää luvussa 4.1.4 esiteltäviä optimointiin perustuvia verkon parannusmenetelmiä. Kolmiulotteisten verkkojen elementtien laatua on myös hyvin hankala tarkastella pelkästään visuaalisesti.

Canann [8] toteaa, että laatukriteerin tulisi olla

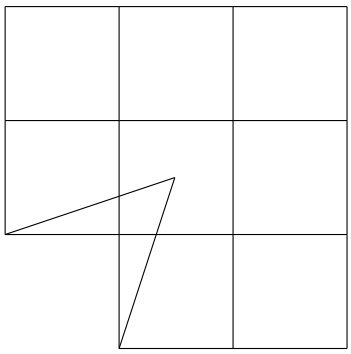
- elementin laatua mahdollisimman tarkasti kuvaava
- vertailukelpoinen eri elementtityyppien välillä
- laskennallisesti tehokas, koska sitä tarvitaan usein
- jatkuva funktio, jotta sitä voitaisiin käyttää optimoinnissa
- määritelty myös nurjille elementeille.

Nurjalla (*engl. inverted*) elementillä tarkoitetaan elementtiä, jolla on laskennallisesti negatiivinen pinta-ala. Esimerkiksi kuvassa 2.4 vasemman alakulman elementin solmupiste on siirtynyt sellaiseen paikkaan, että elementistä on tullut nurja.

Yleinen laatukriteeri on niin sanottu muotosuhde (*engl. aspect ratio*), joka mittaa elementin litistyneisyyttä. Sille on olemassa useita erilaisia määritelmiä, jotka tarkoittavat karkeasti samaa asiaa. Se voidaan esimerkiksi määritellä elementin suurimman ja pienimmän leveyden suhteena [7]. Tässä leveys tarkoittaa etäisyyttä kahden sellaisen yhdensuuntaisen hypertason välillä, joiden välissä elementti sijaitsee.

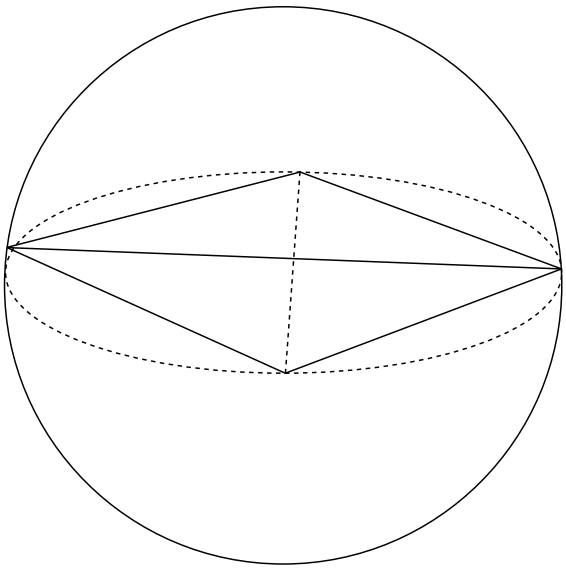
Kolmioille ja nelitahokkaille muotosuhde määritellään usein ympäröivän pallon säteen suhteena sisäpallon säteeseen (katso määritelmät 2.2 ja 2.3). Verkkoa sanotaan hyvin muodostuneeksi (*engl. well shaped*), jos sen elementtien muotosuhde on pieni.

Joissain tarkoituksissa nelitahokkaiden laatukriteerinä voidaan käyttää myös ympäröivän pallon säteen suhdetta lyhimpään särmään, mutta se on kriteerinä vähemmän



Kuva 2.4: Kaksiulotteinen verkko, jonka eräs elementti on nurja.

vaativa. Esimerkiksi lastulla (*engl. sliver*) tämä suhde voi säilyä hyvänä, vaikka sen muotosuhde huononisi rajatta. Lastu on nelitahokas, jolla on positiivimittaiset särmät ja rajoitettu ympäröivän pallon säde, mutta jonka tilavuus on silti häviävän pieni [19]. Tämä on mahdollista siten, että sen kaikki solmut ovat lähes samassa tasossa. Esimerkki lastusta on esitetty kuvassa 2.5.



Kuva 2.5: Lastu ja sen ympäröivä pallo.

Usein käytetään myös edellä mainittujen suhteiden käänteislukua. Tällöin laatu tavallisesti skaalataan välille $[0, 1]$ siten, että vastaava säännöllinen elementti saa laadukseen 1.

3 Verkon generointimenetelmiä

Tässä luvussa pyritään esittelemään tyypillisimmät verkon luontiin käytettävät menetelmät. Niistä on kuitenkin olemassa lukuisia erilaisia muunnelmia ja yhdistelmiä, eikä terminologia ole kaikkien osalta vakiintunutta.

Osa menetelmistä soveltuu vain tietynlaisiin tilanteisiin, kun taas osa pyrkii toimimaan automaattisesti kaikissa tilanteissa. Automaattiseksi voidaan kutsua menetelmää, joka vaatii käyttäjältä ainoastaan jonkinlaisen esityksen verkotettavan alueen reunoista.

Alueen reunat voidaan esittää esimerkiksi spline-käyrien tai -pintojen avulla halutulla tarkkuudella [12]. Tällainen esitys on hyödyllinen menetelmissä, jotka ovat luonteeltaan adaptiivisia, eli ne lisäävät verkotuksen kuluessa solmupisteitä myös verkotetulle alueelle elementtien laadun parantamiseksi tai koon pienentämiseksi. Kun solmuja tällöin lisätään verkon reunalle, ne voidaan sijoittaa mahdollisimman tarkasti laskenta-alueen pinnalle.

Alueen reuna on mahdollista esittää myös pelkän pintaverkon avulla. Tällaisen informaation pohjalta luodun verkon tihentäminen ei kuitenkaan paranna tarkkuutta, jolla verkko approksimoi laskenta-aluetta.

3.1 Manuaalinen generointi

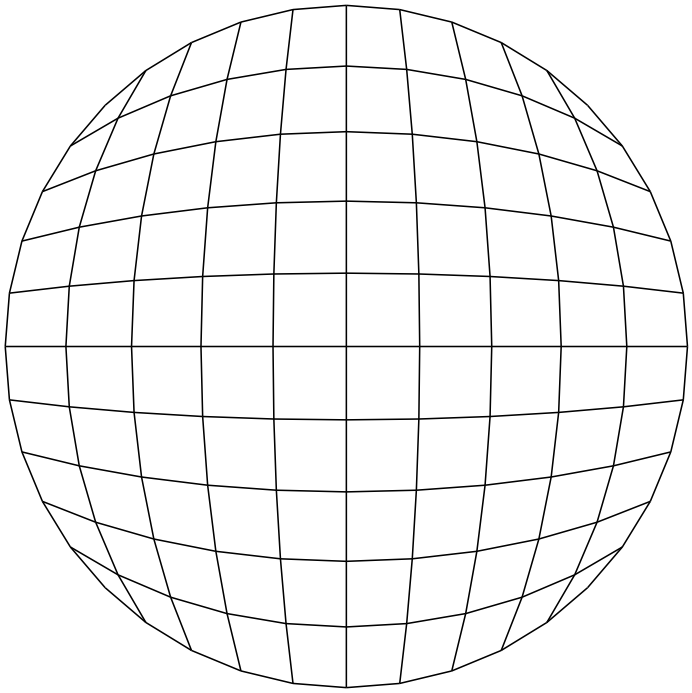
Verkon manuaalisella generoinnilla tarkoitetaan menettelyä, jossa käyttäjä itse luettelee kaikki verkkoon kuuluvat elementit [20]. Käyttäjä siis syöttää kaikkien solmupisteiden koordinaatit ja luettelee verkon elementit solmupisteiden avulla.

On selvää, että menetelmä ei sellaisenaan sovellu kovin monimutkaisille verkoille. Sitä voidaan kuitenkin käyttää pohjana sovellettaessa esimerkiksi pursotusmenetelmiä, jotka esitellään luvussa 3.4. Verkkoihin voidaan myös soveltaa erilaisia geometrisia muunnoksia, kuten kiertoa tai peilausta, ja niitä voidaan yhdistellä monimutkaisempien verkkojen luomiseksi [21].

3.2 Verkkojen muunnokset

Rakenteellista verkkoa voidaan aina ajatella yksinkertaiseen perusalueeseen luodun referenssiverkon muunnoksena. Esimerkiksi kuvan 2.2 rakenteellinen verkko on oleellisesti vain verkotettu yksikköneliö, jonka reunoja on ”puristettu” hieman sisäänpäin. Kolmioverkoille referenssialue on vastaavasti yksikkökolmio, kuusitahokasverkoille kuutio ja niin edelleen.

Verkotettavan alueen tulee tämän vuoksi loogisessa mielessä vastata referenssialuetta. Jos tällaista vastaavuutta ei ole, tuloksena syntyvä verkko ei yleensä ole järkevä. Esimerkiksi kuvassa 3.1 on ympyrä verkotettuna rakenteellisella nelikulmioverkolla. Verkossa on kuitenkin erittäin huonoja elementtejä paikoissa, jotka vastaavat ”nelikulmion” kulmia.



Kuva 3.1: Ympyrään väkisin luotu rakenteellinen nelikulmioverkko sisältää huonoja elementtejä.

Muunnos voi olla algebrallinen, esimerkiksi verkotettavan alueen reunojen interpolaatio, tai se voi perustua tiettyjen osittaisdifferentiaaliyhtälöiden ratkaisuun.

3.2.1 Algebralliset muunnokset

Niin sanottujen algebrallisten menetelmien yleinen periaate on se, että referenssialueen verkko kuvataan verkotettavaan alueeseen jonkin algebrallisen muunnoksen avulla. Menetelmät ovat laskennallisesti erittäin tehokkaita ja suhteellisen helposti ohjelmoitavissa.

Lopputuloksen laatu kuitenkin riippuu vahvasti käytettävän muunnoksen ominaisuuksista [20]. Algebralliset menetelmät voivatkin tuottaa virheellisiä verkkoja, jos verkotettavan alueen reunat ovat hyvin konkaaveja. Referenssiverkon sisäpisteet voivat tällöin kuvautua kokonaan verkotettavan alueen ulkopuolelle, jolloin syntyy nurjia elementtejä, tai pisteitä voi kasautua reunojen läheisyyteen.

Ajatellaan tilannetta, jossa verkon halutut reunapisteet on annettu. Referenssialueeseen voidaan nyt luoda reunapisteitä niin, että niiden suhteelliset etäisyydet vastaavat etäisyyksiä verkotettavan alueen reunoilla. Referenssialue voidaan tämän jälkeen verkottaa yksinkertaisesti. Esimerkiksi yksikköneliön tapauksessa riittää, että reunoilla olevat vastinpisteet liitetään suorilla toisiinsa. Lopuksi referenssiverkon solmupisteet kuvataan varsinaiseen verkotettavaan alueeseen. Lopullinen verkko muodostuu, kun kuvatut pisteet yhdistetään referenssialueessa vallitsevan topologian mukaisesti.

Kuvaukselta voidaan vaatia joitakin erityisominaisuuksia. Yleensä esimerkiksi halutaan referenssialueen reunojen kuvautuvan tarkalleen verkotettavan alueen reunoiksi. Niin sanotun transfinitiin interpolaation avulla voidaan muodostaa kuvauksia, joilla on tämä ominaisuus. Menetelmästä on kerrottu esimerkiksi lähteessä [57], johon seuraavat tiedot perustuvat.

Transfiniitti interpolaatio (*engl. transfinite interpolation*) on laajimmin käytetty algebrallinen generointimenetelmä. Menetelmässä laskennallisia koordinaattisuuntia vastaavista interpolaatioista muodostetaan Boolen summa. Tätä varten tarvitaan myös interpolaatioiden tensorituloja. Yhden muuttujan interpolaatiot muodostetaan fysikaalisen informaation (paikka- ja derivaattainformaatio) ja sekoitusfunktioiden (*engl. blending function*) avulla.

Yhden muuttujan interpolointiin voidaan käyttää mitä tahansa menetelmää, joka täyttää tietyt ehdot (katso viite [57]). Interpolointi voidaan suorittaa esimerkiksi matala-asteisilla polynomeilla. Tällöin tarvittavan syöttötiedon määrä pysyy alhaisena. Jos reunoista on käytettävissä myös normaaliderivaatta, voidaan käyttää Hermiten interpolaatiopolynomeja.

Eri suuntiin voidaan käyttää myös erilaista interpolaatiota. Tarvittaessa voidaan siis käyttää johonkin tiettyyn suuntaan tarkempaa, korkeamman asteen interpolaatiota.

Esimerkki 3.1 Olkoon verkotettavana loogisesti nelikulmiota vastaava alue kahdessa ulottuvuudessa. Halutaan siis muodostaa kuvaus $X(\xi, \eta)$, joka kuvaa yksikköneliön verkotettavaksi alueeksi. Tarkoitusta varten on käytettävissä informaatio verkotettavan alueen reunoista, eli funktion X arvot tunnetaan joukossa $[0,1]^2 \backslash]0,1[^2$. Loput funktion X arvot saadaan transfiniitin interpolaation avulla.

Lineaarisiksi sekoitusfunktioiksi α_i ja β_i kelpaavat
$\alpha_1(\xi) = 1 - \xi$, $\alpha_2(\xi) = \xi$, $\beta_1(\eta) = 1 - \eta$ ja $\beta_2(\eta) = \eta$. Tällöin

$$U(\xi, \eta) = (1 - \xi)X(0, \eta) + \xi X(1, \eta)$$

$$V(\xi, \eta) = (1 - \eta)X(\xi, 0) + \eta X(\xi, 1)$$

$$UV(\xi, \eta) = (1 - \xi)(1 - \eta)X(0, 0) + (1 - \xi)\eta X(0, 1) + \xi(1 - \eta)X(1, 0) + \xi\eta X(1, 1)$$

Tässä siis U on interpolaatio laskennalliseen koordinaattisuuntaan ξ, V suuntaan η ja UV niiden tensoritulo. Nyt Boolen summa $X(\xi, \eta) = U \oplus V = U + V - UV$ on haluttu kuvaus. On helppo tarkistaa, että interpolaatio todella säilyttää reunat sellaisenaan.

Edes alueen reunoja ei välttämättä tarvitse tuntea kaikkialla. Interpolaation avulla kuvataan vain solmupisteitä, joten riittää, että funktio X on määritelty niitä vastaavissa pisteissä (ξ_i, η_j).

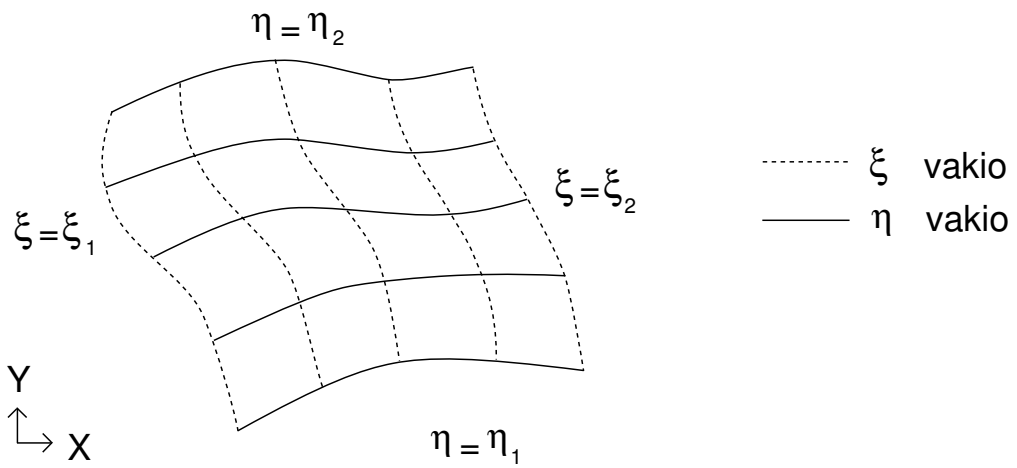
Myös muuta kuin transfiniittiä interpolaatiota voidaan käyttää. Esimerkiksi lähteessä [28] johdetaan kolme vaihtoehtoista menetelmää, jotka ovat niin sanotusti sisäisiä (*engl. intrinsic*), eli ne käyttävät hyväksi reunainformaatiota ilman sekoitusfunktioita. Menetelmät eivät kuitenkaan ole invariantteja edes tavanomaisissa muunnoksissa, kuten siirrossa ja kierrossa.

3.2.2 Osittaisdifferentiaaliyhtälöpohjaiset muunnokset

Rakenteellisia verkkoja voidaan tuottaa myös tiettyjen osittaisdifferentiaaliyhtälöiden ratkaisujen avulla. Myös tällaisissa menetelmissä täytyy verkotettavan alueen ja referenssialueen välillä olla looginen vastaavuus.

Menetelmät syntyivät ajatuksesta käyttää hyväksi Laplace-operaattorin tunnettua silottavaa vaikutusta (katso esimerkiksi [65]). Laplace-yhtälöiden avulla tuotetun verkon solmut sijoittuvat erittäin tasavälisesti verkotettavan alueen sisällä [59]. Tekniikkaa on myöhemmin jalostettu sisältämään niin sanotut kontrollifunktiot, joiden avulla verkon muodostusta voidaan ohjailla.

Esimerkki 3.2 Elliptisen menetelmän perusidea kahdessa ulottuvuudessa loogisesti nelikulmaisen alueen tapauksessa on seuraava: Etsitään verkotettavassa alueessa funktioita $\xi(x, y)$ ja $\eta(x, y)$, joille $\triangle\xi = 0$ ja $\triangle\eta = 0$. Asetetaan reunaehdot siten, että ξ ja η ovat jatkuvia koko reunalla, η saa jotkut vakioarvot η_1 ja η_2 alueen vastakkaisilla reunoilla ja ξ saa vakioarvot ξ_1 ja ξ_2 alueen toisilla vastakkaisilla reunoilla. Ratkaisujen ξ ja η tasa-arvokäyrät muodostavat alueeseen rakenteellisen nelikulmioverkon. Tilannetta havainnollistaa kuva 3.2.



Kuva 3.2: Käyräviivainen koordinaatisto. Koordinaattiviivat vastaavat funktioiden ξ ja η tasa-arvokäyriä.

Käytännössä tilanne ei kuitenkaan ole näin yksinkertainen, koska idean suoraviivainen soveltaminen vaatisi osittaisdifferentiaaliyhtälöiden ratkaisemista verkotettavassa alueessa.

Tutkitaan sen sijaan käänteisfunktioita $x(\xi, \eta)$ ja $y(\xi, \eta)$. Ne on nyt määritelty suorakulmiossa $[\xi_1, \xi_2] \times [\eta_1, \eta_2]$, johon on helppo generoida verkko. Ratkaistavana eivät kuitenkaan ole enää Laplace-yhtälöt, vaan kvasilineaariset yhtälöt

$$\begin{aligned} \alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta} &= 0 \\ \alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta} &= 0, \end{aligned} \tag{3.1}$$

missä

$$\alpha = x_\eta^2 + y_\eta^2$$
$$\beta = x_\xi x_\eta + y_\xi y_\eta$$
$$\gamma = x_\xi^2 + y_\xi^2.$$

Yhtälöt on johdettu esimerkiksi lähteessä [59].

Myös muita kuin elliptisiä yhtälöitä voidaan käyttää. Luvussa 3.4 kerrotaan hyperbolisten yhtälöiden käyttämisestä annetun geometrian ulkopuolisen alueen verkottamiseen.

3.3 Multiblokkimenetelmä

Rakenteellisten generointimenetelmien tuottamat verkot ovat usein laadultaan varsin hyviä, joten niitä pyritään käyttämään aina kun se on mahdollista. Monimutkaisemmat geometriat eivät kuitenkaan yleensä vastaa topologialtaan mitään perusgeometriaa, joten niitä ei voida verkottaa täysin rakenteellisesti. Verkosta voidaan kuitenkin pyrkiä luomaan blokeittain rakenteellinen. Blokkimaisesta rakenteesta on myös hyötyä, jos laskentaa halutaan hajauttaa [21].

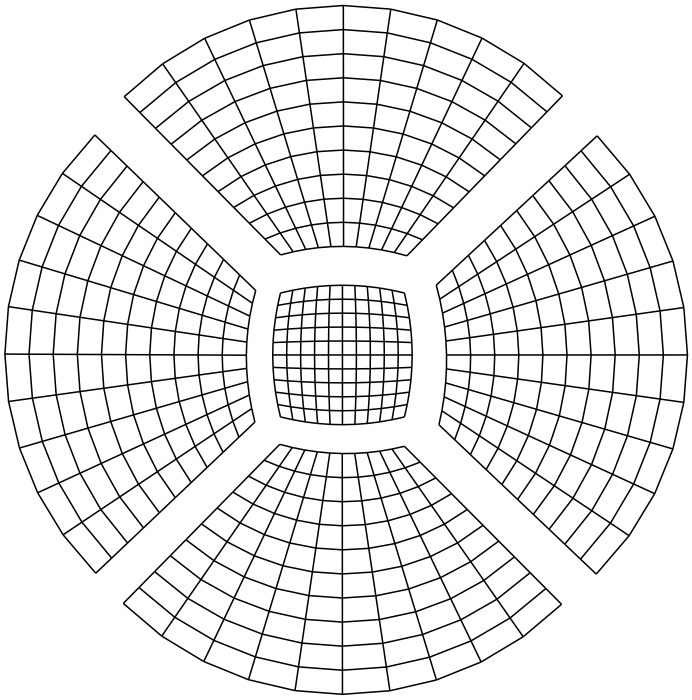
Ongelmaksi muodostuu tällöin verkotettavan alueen jako sopiviin blokkeihin. Jaon automatisointia on tutkittu jonkin verran [41], mutta sopivan jaon löytäminen jää kuitenkin usein jossakin määrin käyttäjän tehtäväksi [7].

Blokkien sisällä verkotus voidaan suorittaa jollakin edellä esitellyistä menetelmistä. Verkotus on kuitenkin suoritettava siten, että blokkeihin luodut verkot voidaan liittää toisiinsa. Blokkien pintaverkkojen tulee siis olla yhtenevät niiden välisellä rajapinnalla.

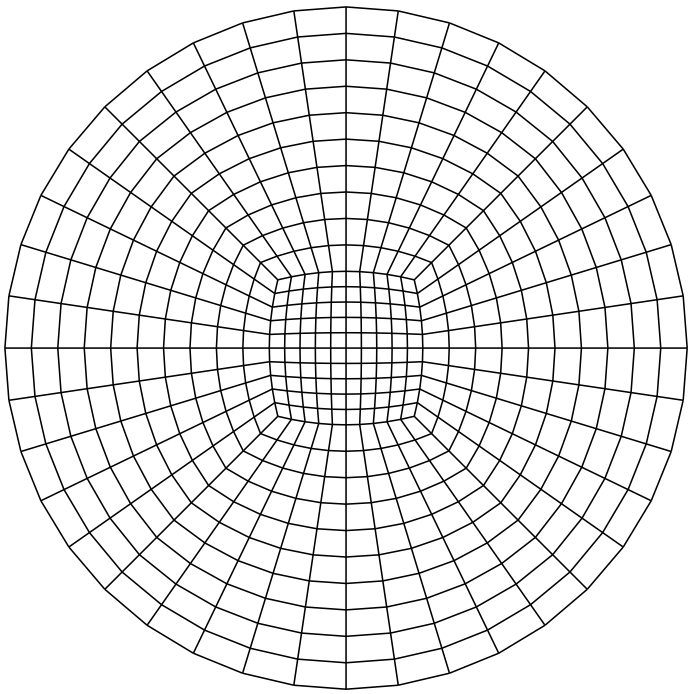
Esimerkiksi kuvassa 3.3 on esitetty yksikköympyrä jaettuna blokkeihin, jotka on verkotettu. Kuvassa 3.4 on blokeista syntyvä nelikulmioverkko, joka on rakenteellinen kunkin blokin sisällä.

3.4 Pursotusmenetelmät

Tietyllä lailla säännölliset alueet voidaan verkottaa käyttämällä hyväksi alemmassa dimensiossa määriteltyä pohjaverkkoa, jota voidaan pursottaa (*engl. extrude, sweep*) pitkin määrättyä generointilinjaa (*engl. generation line*). Kahdessa dimensiossa pohjaverkkona toimii murtoviiva.



Kuva 3.3: Yksikköympyrä jaettuna loogisesti nelikulmaisiin blokkeihin.



Kuva 3.4: Yksikköympyrään luotu blokeittain rakenteellinen nelikulmioverkko.

Kolmiulotteista aluetta sanotaan [21]

- sylinterimäiseksi, jos se voidaan kuvata kaksiulotteisen verkon ja generointilinjan avulla, ja
- kuusitahokasmaiseksi (*engl. hexahedral*), jos se voidaan kuvata murtoviivan ja kahden generointilinjan avulla.

Topologisesti sylinterimäistä kappaletta kutsutaan joskus myös kaksi ja puoli -ulotteiseksi.

Pursotusmenetelmissä luodaan generointilinjan suunnassa peräkkäisille tasoille verkkoja, jotka ovat topologisesti pohjaverkon kaltaisia. Niitä voidaankin ajatella pohjaverkon muunnoksina. Yhdistämällä nämä verkot toisiinsa muodostetaan korkeamman dimension elementtejä. Kahdessa dimensiossa muodostuu siis yleensä nelikulmioita ja kolmessa dimensiossa kuusitahokkaita ja prismaelementtejä. Menetelmällä voidaan tuottaa rakenteellisia verkkoja, jos pohjana käytettävä verkko on rakenteellinen.

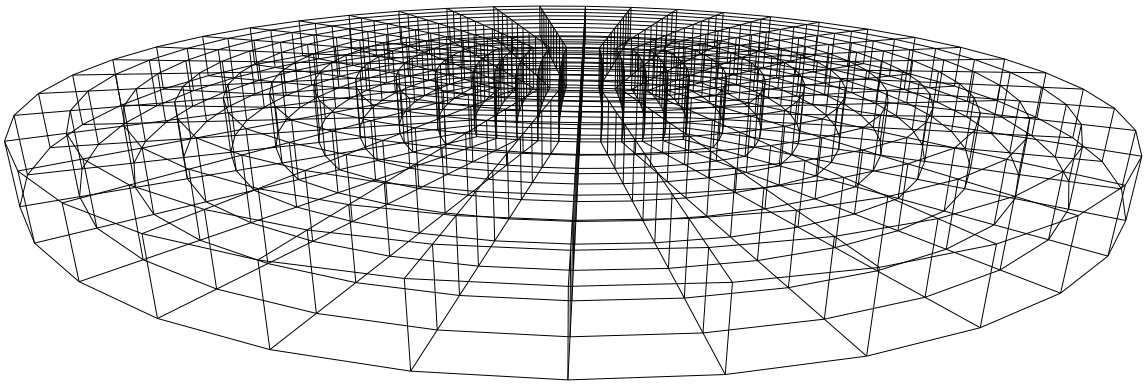
Joskus tilanne voi kuitenkin olla niin sanotusti degeneroitunut. Esimerkiksi jos viivaa pyöräytetään toisen päätepisteensä ympäri, ei lähelle keskustaa synny nelikulmioita vaan kolmioita. Tällaisessakin tapauksessa verkko voidaan yhtälöitä ratkaistaessa käsitellä rakenteellisena esimerkiksi asettamalla lähinnä pyörähdysakselia olevien elementtien pinta-alat nolliksi. Tämän on kuitenkin havaittu joissain tapauksissa hidastavan konvergenssia ja lisäävän ratkaisun virhettä akselin läheisyydessä [25].

Menetelmän lähtötiedoiksi tarvitaan siis pohjaverkko ja jonkinlainen esitys verkotettavan alueen muista reunoista. Yksinkertaisissa tilanteissa voi olla käytössä kuvaus, jonka avulla voidaan määrätä sekä reunojen että sisäpisteiden paikat. Tällaisia tilanteita ovat esimerkiksi suora pursotus jonkin vektorin suuntaan ja pyöräytys määrätyn akselin ympäri.

Yleisemmässä tapauksessa ongelmaksi muodostuu sisäpisteiden sijoittelu [29]. Eräs perinteinen pursotuksen toteutustapa on verkottaa myös pursotetun objektin generointilinjan suuntaiset reunat rakenteellisella verkolla [36]. Tällöin reunapisteiden projektio tasolta toiselle tunnetaan, ja sen avulla voidaan määrätä myös sisäpisteiden projektiot esimerkiksi keskiarvoina tunnetuista projektiovektoreista. Staten ym. [61] käyttävät reunapisteet sisältävää apuverkkoa, jonka avulla sisäpisteet interpoloidaan reunapisteistä kullakin tasolla.

Tasoille luotuja verkkoja voidaan joutua silottamaan luvussa 4.1 esiteltävillä menetelmillä elementtien laadun parantamiseksi. Muun muassa Knupp [31] esittää muunnosmenetelmän, joka perustuu Winslow-silotukseen, mutta pyrkii kuitenkin säilyttämään muunnettavan verkon mahdollisimman muuttumattomana.

Pursotusmenetelmällä voidaan varsin helposti verkottaa alueita, joiden topologia on sylinterimäinen tai kuusitahokasmainen. Esimerkiksi virtauslaskennassa tyypilliset putkimaiset geometriat voidaan verkottaa tällä menetelmällä. Kuvassa 3.4 on esimerkki yksikköympyrään luodusta nelikulmioverkosta. Siitä saadaan pursottamalla blokeittain rakenteellinen kuusitahokasverkko, jonka yksi kerros on esitetty kuvassa 3.5.



Kuva 3.5: Yksikköympyrän nelikulmioverkkoa pursottamalla saatu kuusitahokkaista muodostuva kiekko.

Toinen pursotuksen sovellus on rakenteellisen kerroksen luominen laskenta-alueen reunoille. Pintaverkkoa voidaan pursottaa jonkin matkaa alueen sisään, minkä jälkeen loput alueesta voidaan verkottaa rakenteettomalla verkolla. Reunojen lähelle saadaan näin kerros hyvälaatuisia elementtejä. Pursotus tehdään yleisesti pinnan normaalin suuntaan, mutta se vaatii varovaisuutta, jotta projisoidut särmät eivät mene ristiin. Risteäminen estetään lähteissä [26] ja [62] suorittamalla projektiovektoreille Laplace- tai Winslow-silotus.

Pursotus voidaan tehdä myös hyperbolisten osittaisdifferentiaaliyhtälöiden avulla, kuten lähteessä [10] esitetään. Yhtälöiden avulla voidaan tuottaa verkkoja lähtemällä liikkeelle käyrältä (kolmessa ulottuvuudessa pinnalta) ja laajentamalla verkkoa sen normaalin suuntaan. Menetelmässä ainoastaan alkuperäinen käyrä tai pinta on käyttäjän tarkasti määrättävissä. Se ei kuitenkaan haittaa, jos laskenta tapahtuu jonkin geometrian ulkopuolisessa alueessa. Tällöin on jopa hyödyllistä, että käyttäjän tarvitsee määritellä tarkasti ainoastaan yksi pinta. Hyperbolisella menetelmällä tuotetun verkon särmät ovat yleensä lähes kohtisuorassa toisiaan vastaan.

3.5 Delaunay-tyyppiset menetelmät

Tämän luokan menetelmät ovat ehdottamasti suosituimpia rakenteettomien kolmio- ja nelitahokasverkkojen generoimiseksi [39]. Siksi niitä käsitellään tässä hieman muita menetelmätyyppejä laajemmin. Kolmio- ja nelitahokasverkoista käytetään tässä luvussa yhteisnimitystä kolmiointi, koska se on Delaunay-tyyppisten menetelmien yhteydessä kirjallisuudessa varsin yleistä. Menetelmät perustuvat Voronoï-kaavioihin ja erityisesti niiden duaaliin, Delaunayn kolmiointiin.

Määritelmä 3.1 Olkoon solmujoukko S annettu, ja olkoot $p, q \in S$. Määritellään puolitaso $H(p,q) = \{x \mid d(p,x) < d(q,x)\}$, missä $d(\cdot,\cdot)$ on pisteiden euklidinen etäisyys. Nyt solmun p Voronoïn solu (solmujoukon S suhteen) on joukko [2]

$$V(p) = \bigcap_{q \in S, q \neq p} H(p,q)$$

Solmujoukon Delaunayn kolmiointi saadaan yhdistämällä annetussa solmujoukossa ne solmut, joiden Voronoïn solut ovat toistensa naapureita.

Olkoon d avaruuden dimensio. Tällöin tiedetään [19], että mikäli solmujoukko ei sisällä $d + 2$ samalla pallonkuorella sijaitsevaa solmua, duaali on yksikäsitteinen ja muodostuu pelkästään kolmioista (nelitahokkaista). Muussa tapauksessa duaalissa voi olla muitakin konvekseja monikulmioita, jotka täytyy pilkkoa kolmioinnin muodostamiseksi. Tämä jako ei enää ole yksikäsitteinen.

3.5.1 Delaunayn kolmioinnin ominaisuuksia

Annetun solmujoukon Delaunayn kolmioinnilla on useita erityisominaisuuksia. Kolmioinnin karakterisoi niin sanottu Delaunayn ehto.

Määritelmä 3.2 Kolmiointi on Delaunay, mikäli kuhunkin elementtiin liittyvä avoin ympäröivä pallo ei sisällä yhtään kolmioinnin solmupistettä [21].

Kuvassa 3.6 on kaksi saman solmujoukon kolmiointia. Vasemman puoleisessa kolmioinnissa nuolella merkitty solmu rikkoo Delaunayn ehtoa. Oikeanpuoleinen kolmiointi sen sijaan on Delaunay.

Kolmiointi on tietyssä mielessä optimaalinen kaikista annetun solmujoukon kolmioinneista. Piirretään esimerkiksi kaikkien kolmioinnin elementtien ympärille pienin mahdollinen pallo. Jos nyt etsitään suurin kuhunkin kolmiointiin liittyvistä tällaisista

palloista, niin Rajan [44] osoittaa, että millään muulla kolmioinnilla se ei ole pienempi kuin Delaunayn kolmioinnilla. Tulos pätee dimensiosta riippumatta.

Kaksiulotteisessa tapauksessa pätee vahvempiakin tuloksia [19, 54]: kaikista annetun solmujoukon kolmioinneista Delaunayn kolmiointi myös maksimoi pienimmän kolmioinnissa esiintyvän kulman ja minimoi suurimman ympäröivän pallon säteen.

Kolmessa ulottuvuudessa on kuitenkin ylimääräisiä ongelmia. Delaunayn ehdon täyttävässä verkossa voi nimittäin olla kuvan 2.5 kaltaisia lastuja, jotka on poistettava verkosta jollakin luvussa 4.1 esiteltävällä menetelmällä.

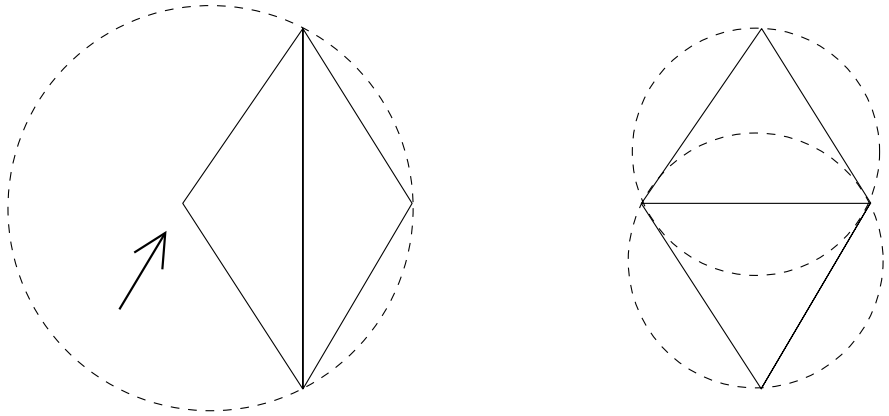
Rajan esittää myös riittävän ehdon sille, että kolmiointi on Delaunay. Näin on, mikäli kolmioinnin jokainen elementti sisältää siihen liittyvän ympäröivän pallon (katso määritelmä 2.2) keskipisteen. Kaksiulotteisessa tapauksessa ehto on yhtäpitävää sen kanssa, että kolmiossa ei ole tylppiä kulmia.

3.5.2 Delaunay-verkon luominen

On olemassa useita erilaisia algoritmeja, jotka pyrkivät luomaan Delaunayn ehdon täyttäviä verkkoja. Kuten edellä esitettiin, annetun solmujoukon Delaunayn kolmiointi on tietyin edellytyksin yksikäsitteinen. Menetelmät eroavatkin toisistaan lähinnä reunojen käsittelyssä ja siinä, kuinka verkon sisäsolmut oikeastaan luodaan.

Teoriassa verkotus voitaisiin suorittaa luomalla verkotettavaan alueeseen joukko solmuja, muodostamalla solmujoukon Voronoïn kaavio ja yhdistämällä solmut, joiden Voronoïn solut ovat naapureita. Tämä ei kuitenkaan ole osoittautunut käytännössä tehokkaaksi tavaksi [18].

Toinen tapa on lisätä solmut yksitellen niin sanotun Delaunayn ytimen (*engl. De-*



Kuva 3.6: Kaksi saman solmujoukon kolmiointia, joista oikeanpuoleinen on Delaunay.

launay kernel) avulla. Tätä kutsutaan inkrementaaliseksi menetelmäksi, tai joissain lähteissä Boyerin tai Watsonin algoritmiksi.

Delaunayn ydin on lokaali operaatio, josta seuraa $\mathcal{T}_{i+1} = \mathcal{T}_i - \mathcal{C}_P + \mathcal{B}_P$. Jos lisättävät solmut ovat aina kolmioinnin sisällä, puhutaan redusoidusta inkrementaalisesta menetelmästä ja tällöin

- P on lisättävä solmu
- $\mathcal{T}_i$ on kolmiointi vaiheessa i
- $\mathcal{C}_P$ on ne elementit, joiden ympäröivä pallo sisältää P:n
- $\mathcal{B}_P$ on ne elementit, jotka muodostuvat, kun P liitetään joukon $\mathcal{C}_P$ ulkosärmiin (tahkoihin).

Näin on mahdollista tehdä, koska $\mathcal{C}_P$:n ulkoreunat ovat aina näkyvissä P:stä [20]. Syntyvä kolmiointi on Delaunay [19].

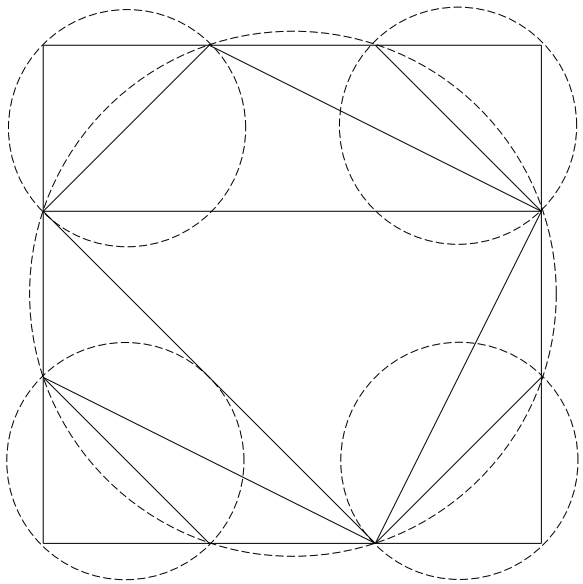
Mikäli P ei ole kolmioinnin sisällä, Delaunayn ydin määritellään hieman eri tavalla. Käytännössä verkotettava alue voidaan kuitenkin ympäröidä laatikolla, joka jaetaan sopivasti kolmioihin tai nelitahokkaisiin. Tämän jälkeen päästään käyttämään inkrementaalisen menetelmän redusoitua versiota.

Kuvissa 3.7 ja 3.8 on esitelty inkrementaalisen menetelmän käyttöä yksikköneliön verkottamiseen. Ensin luodaan niin sanottu tyhjä verkko, joka ei sisällä lainkaan sisäsolmuja. Tämän jälkeen solmuja luodaan jollakin kriteerillä huonoimmiksi valittujen elementtien ympäröivien pallojen keskipisteisiin.

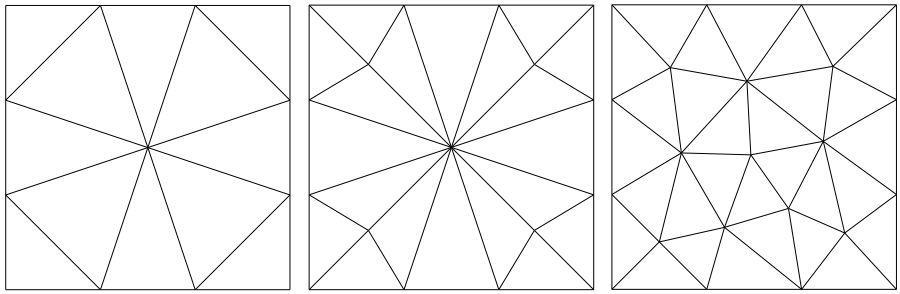
Kuvan 3.7 tyhjän verkon kaikkien muiden kuin kulmissa sijaitsevien elementtien ympäröivät pallot ovat tilanteen säännöllisyyden vuoksi samat. Jokin kyseisistä elementeistä on laadultaan huonoin, ja ensimmäinen sisäsolmu luodaan neliön keskipisteeseen. Muut kuin kulmissa sijaitsevat elementit puretaan, ja alueeseen luodaan uudet kolmiot liittämällä uusi solmu syntyneen tyhjän alueen ulkosärmiin. Lopputulos nähdään kuvassa 3.8 vasemmalla.

Seuraavassa vaiheessa huonoimpia ovat kulmissa sijaitsevat elementit. Uudet solmut lisätään niiden ympäröivien pallojen keskipisteisiin aivan kuten edellä. Tulos näkyy kuvan 3.8 keskellä.

On myös muita tapoja luoda verkkoon lisättävät solmut. George [19] ym. esittävät tavan, jossa lähdetään liikkeelle tyhjän verkon reunoilta. Kullekin reunasolmulle voidaan määrittää arvo h_{loc} keskiarvona siihen liittyvien särmien pituuksista. Tämän jälkeen verkon sisäsärmille voidaan luoda solmuja siten, että niiden etäisyydet nou-



Kuva 3.7: Yksikköneliön tyhjä verkko. Neliön kukin reuna on jaettu kolmeen yhtä pitkään särmään. Elementtien ympäröivät pallot on merkitty katkoviivoin.



Kuva 3.8: Inkrementaallisella menetelmällä luotuja verkkoja, kun neliössä on yksi, viisi ja kymmenen sisäsolmua.

dattavat mahdollisimman hyvin määrättyä jakaumaa, joka vaihtuu jossakin mielessä tasaisesti arvosta h_{loc1} arvoon h_{loc2} reunalta toiselle siirryttäessä.

Inkrementaalinen menetelmä voidaan toteuttaa myös niin sanotun Lawsonin algoritmin avulla. Menetelmä perustuu elementtien lokaaleihin muunnoksiin [54]. Esimerkiksi kaksiulotteisessa tilanteessa uuden solmun sisältävä kolmio jaetaan kolmeen uuteen kolmioon siten, että uusi solmu on kaikkien kolmioiden kärkipiste. Tämän jälkeen uusien kolmioiden ja niiden naapureiden sisällä tehdään tarpeen mukaan luvussa 4.1.1 esitettäviä diagonaalimuunnoksia niin, että kaikki alueen kolmiot täyttävät Delaunay-ehdon.

Lawsonin menetelmä ei ole herkkä pyöristysvirheille, koska verkko on menetelmän kaikissa vaiheissa kuitenkin laillinen. Aiemmin esitetyssä menetelmässä taas pyöristysvirheet voivat johtaa tilanteeseen, jossa elementtien poistaminen johtaa epäyhtenäisen alueen syntymiseen. Esimerkiksi kuvan 3.7 tapauksessa näin voisi helposti käydä, koska useiden elementtien ympäröivät pallot ovat samat. Uuden solmun lisääminen hyvin lähelle kyseisen pallon reunaa voisi nyt helposti aiheuttaa pyöristysvirheitä, jotka johtaisivat jonkun elementin poistamatta jäämiseen. Kun uusi solmu tämän jälkeen yhdistettäisiin alueen ulkosärmiin, ei syntyvä verkko enää olisi sallittu.

3.5.3 Reunojen käsittely

Joskus verkotuksessa lähdetään liikkeelle siitä, että alueen reunat on annettu reunasärmien tai pintaverkon avulla. Tällöin yleensä vaaditaan, että annetut särmät (ja tahkot) sisältyvät myös lopulliseen verkkoon.

Reunalla olevat solmut saadaan lisättyä kolmiointiin inkrementaalisen menetelmän avulla. Solmujen lisääminen ei kuitenkaan takaa, että verkko sisältäisi myös annetut särmät, eikä särmien esiintyminen verkossa vielä takaa tahkojen olemassaoloa [19].

Reunojen sisällyttämiseksi syntyvään kolmiointiin voidaan käyttää lähinnä kahta taktiikkaa:

- pilkotaan särmiä ja tahkoja pienempiin osiin, jotka on helpompi sisällyttää kolmiointiin tai
- modifioidaan verkkoa lokaalisti ja pakotetaan reunat mukaan verkkoon.

Ensimmäistä tapaa sovellettaessa alkuperäiset reunat eivät esiinny verkossa sellaisenaan vaan paloiteltuina. Tämä on ongelma esimerkiksi silloin, jos verkko on tarkoitus liittää toiseen verkkoon, koska verkkojen pintojen täytyisi tällöin olla yhtenevät.

Toinen tapa johtaa yleensä verkkoon, joka ei lokaalisti täytä Delaunayn ehtoa. Kaksiulotteisessa tilanteessa reunat kuitenkin saadaan aina näin pakotettua verkkoon, koska mistä tahansa annetun solmujoukon konveksin peitteen kolmioinnista voidaan muokata peitteen mielivaltainen toinen kolmiointi luvussa 4.1.1 esitettävää diagonaalimuunnosta käyttämällä [19].

Kaksiulotteinen verkotus voidaan siis suorittaa seuraavasti:

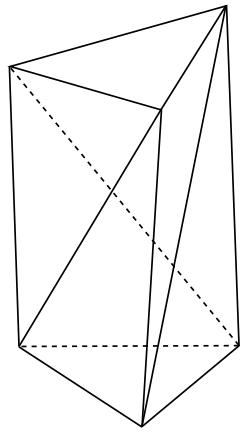
Algoritmi 3.1 Kaksiulotteinen Delaunay-verkotus inkrementaalisella menetelmällä.

- Ympäröidään verkotettava alue suorakulmiolla, joka jaetaan kolmioihin.
- Lisätään annetut reunasolmut inkrementaalisen menetelmän avulla.
- Suoritetaan muodostuneessa verkossa lokaaleja muunnoksia, joiden avulla annetut reunasärmät pakotetaan verkkoon. Reunat voidaan sisällyttää verkkoon myös lisäämällä solmuja puuttuville särmille. Näin saadaan muodostettua verkotettavan alueen tyhjä verkko.
- Poistetaan verkotettavan alueen ulkopuoliset kolmiot.
- Tihennetään verkkoa, kunnes annetut laatukriteerit täyttyvät.

Kolmiulotteiseen tapaukseen liittyy kuitenkin huomattavasti enemmän ongelmia. Kaikkia monikulmioita ei esimerkiksi voida jakaa nelitahokkaisiin lisäämättä niiden sisälle solmuja. Yksinkertainen esimerkki tällaisesta monikulmiosta on niin sanottu Schönhardtin monikulmio eli kierretty prisma, joka on esitetty kuvassa 3.9. Monikulmio on lokaalisti konkaavi prisman nelikulmiosivujen lävistäjän kohdalla. Valittiinpa prismasta mitkä tahansa neljä solmua, on kahden niistä välissä jokin tällainen särmä. Solmuista ei siis voida muodostaa nelitahokasta, joka olisi täysin monikulmion sisällä.

Yleisesti siis kiinnitetyn solmujoukon avulla ei välttämättä voida muodostaa nelitahokasverkkoa, jonka reunat muodostuisivat halutuista kolmioista. Algoritmin 3.1 tyhjän verkon muodostaminen ei siis välttämättä onnistu lisäämättä alueeseen sisäsolmuja.

Edelleen, päätösongelma ”voidaanko annettu monikulmio osittaa nelitahokkaisiin ilman sisäsolmuja” on NP-täydellinen, ja ongelma ”voidaanko annettu monikulmio osittaa nelitahokkaisiin korkeintaan k:n sisäsolmun avulla” on NP-kova [46]. Tämä ei kuitenkaan estä kehittämästä heuristisia menetelmiä, joiden avulla reunoja voidaan yrittää pakottaa mukaan verkkoon (katso esim. [18, 19]).



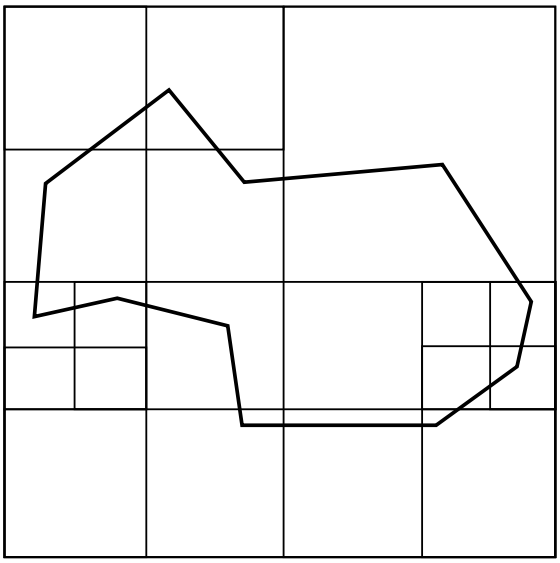
Kuva 3.9: Schönhardtin monikulmio, jota ei voida jakaa nelitahokkaisiin lisäämättä sen sisälle solmuja.

3.6 Quadtree ja octree

Quadtree- ja octree-menetelmät perustuvat verkotettavan alueen ympäröimiseen säännöllisellä muodolla, jota jaetaan rekursiivisesti samanlaisiin pienempiin osiin eli soluihin. Kaksiulotteisessa tapauksessa alue ympäröidään suorakulmiolla, joka jaetaan neljään samanlaiseen osaan. Kolmiulotteisessa tapauksessa alue vastaavasti ympäröidään suorakulmaisella särmiöllä, joka jaetaan kahdeksaan osaan. Jakoa jatketaan niin kauan, että tietty lopetusehto täyttyy.

Lopetusehto voidaan määrittää esimerkiksi kiinnittämällä rekursion tuleva syvyys etukäteen [53]. Tällöin verkotettavan alueen reunalla sijaitsevien solujen jakamista jatketaan, kunnes maksimisyvyys on saavutettu. Jos reuna on annettu diskreetissä muodossa, niin tihentäminen voidaan lopettaa, kun kukin solu sisältää enää yhden reunasolmun [21]. Tämän jälkeen jakoa voidaan vielä tasapainottaa jatkamalla jakamista niin kauan, että naapurisolujen syvyydet eroavat toisistaan korkeintaan yhdellä [20]. Esimerkiksi kuvan 3.10 tilanteessa oikean ylänurkan solu tulisi jakaa vielä kertaalleen, koska sen alapuolella on soluja, jotka ovat sitä kaksi tasoa syvemmällä.

Lopuksi verkotettavan alueen ulkopuoliset solut voidaan poistaa ja loput voidaan jakaa kolmioihin (nelitahokkaisiin). Verkon solmupisteitä ovat tällöin syntyneen jaon solujen nurkkapisteet. Kuhunkin soluun syntyvät elementit määräytyvät sen mukaan, montako solmua solun reunoilla on ja kuinka ne sijoittuvat toisiinsa nähden. Reunoilla sijaitsevat solut vaativat erityiskäsittelyä, jotta reuna tulisi mahdollisimman tarkoin otetuksi huomioon. Tämä on tehtävä huolellisesti, jotta reunojen lähistölle ei syntyisi



Kuva 3.10: Eräs kaksiulotteinen geometria ja siihen liittyvä tasapainottamaton jako quadtree-menetelmässä.

huonolaatuisia elementtejä.

Menetelmän nimi johtuu siitä, että jako voidaan helposti esittää puurakenteena. Siinä puun juuri on alkuperäinen nelikulmio tai kuusitahokas ja kunkin solmun lapset vastaavat siitä jakamalla syntyneitä pienempiä alueita. Lapsia on siis kaksiulotteisessa tapauksessa neljä ja kolmiulotteisessa tapauksessa kahdeksan. Puurakennetta voidaan käyttää tehokkaasti hyväksi verkon generointiin ja laskentaan liittyvissä operaatioissa, joissa tarvitaan tietoa tiettyyn alueeseen liittyvästä solusta tai tietyn solun naapureista [53].

Menetelmän huonona puolena voidaan pitää sitä, että se ei ole kiertoinvariantti vaan lopputulos riippuu vahvasti siitä, missä asennossa kappale käytettävässä koordinaatistossa esitetään. Menetelmän tuottama verkko on rakenteeton.

Mitchell ja Vavasis [37] esittävät octree-pohjaisen menetelmän, joka tuottaa hyvin muodostuneita nelitahokasverkkoja. Verkot eivät kuitenkaan täytä edellä esitettyä Delaunay-ehtoa.

3.7 Advancing front

Advancing front on yleisnimitys sellaisille menetelmille, joissa verkon luominen tapahtuu jonkinlaisen etenevän rintaman avulla. Rintama syntyy, kun jo verkotetun alueen reunalta valitaan niin sanottu lähtöalue (*engl. departure zone*), johon liittyen luodaan uusia elementtejä joko vanhojen solmujen avulla tai luomalla uusia [20]. Tarkoituksena on luonnollisesti muodostaa jossakin mielessä optimaalisen muotoisia elementtejä. Menetelmien tuottamat verkot ovat rakenteettomia. Muodostettavan verkon solmujoukko voi olla kiinnitetty etukäteen, esimerkiksi jos menetelmää käytetään jonkinlaiseen lokaaliin uudelleenverkotukseen [14, 34]. Tällöin uudet elementit voidaan luoda esimerkiksi siten, että ne täyttävät luvussa 3.5.1 esitetyn Delaunayn ehdon.

Toisaalta solmut voidaan luoda samalla, kun rintama etenee [42]. Tällöin solmujen sijoittelulla päästään suoraan optimoimaan jotain tiettyä laatukriteeriä. Näin toimittaessa täytyy kuitenkin erityisesti varmistaa, että luotava solmu ei ole verkotettavan alueen ulkopuolella eikä minkään olemassa olevan elementin sisällä.

Kun olemassa olevan rintaman yhteen tai useampaan särmään (kolmessa ulottuvuudessa tahkoon) liittyen luodaan elementtejä, täytyy rintamaa luonnollisesti päivittää. Tällöin särmät (tahkot), joihin ei enää ole mahdollista liittää elementtejä, poistetaan rintamasta. Mikäli elementtiä luotaessa syntyy uusia särmiä (tahkoja), ne liitetään rintamaan. Menettelyä jatketaan, kunnes rintama on tyhjä. Alue on tällöin verkotettu.

Kun menetelmää sovelletaan kolmio- tai nelitahokasverkkojen luomiseen, tarvitaan uuden elementin luomiseksi pohjasärmän (tahkon) lisäksi korkeintaan yksi uusi solmu. Uusia solmuja ei tarvita lainkaan, jos rintamasta löydetään kaksi särmää (tahkoa), jotka jo valmiiksi ovat sopivassa asennossa toisiinsa nähden.

Kolmessa ulottuvuudessa ongelmia aiheuttaa se, että rintamien väliin voi muodostua hyvin monimutkaisia tyhjiä alueita. Voi syntyä tilanteita, joissa alueeseen täytyy luoda uusia solmuja ennen kuin siihen voidaan lainkaan muodostaa nelitahokkaita. Ongelmaksi muodostuu tällöin uusien solmujen sijoittelu.

Rees [45] esittää advancing front -lähestymistapaan perustavan menetelmän, jolla voidaan luoda lähes pelkistä nelikulmioista muodostuvia rakenteettomia kaksiulotteisia verkkoja. Menetelmä etenee alueen reunoilta kohti sen keskustaa. Kolmioelementtejä luodaan vain, jos mahdollisten nelikulmioiden laatu on alle käyttäjän antaman toleranssin. Tämän vuoksi kolmiot syntyvät yleensä kauas alueen reunoista, ja reunojen lähelle syntyy hyvälaatuisia nelikulmioelementtejä.

Paving ja plastering

Eräs advancing front -ajattelutavan sovellus on paving. Se on algoritmi nelikulmioista muodostuvan verkon luomiseen. Se toteutettiin alun perin kahdessa ulottuvuudessa, mutta on myöhemmin yleistetty myös kolmiulotteisille pintaverkoille [64].

Menetelmässä verkotettavan alueen reunaa kiertää yksi tai useampia silmukoita. Esimerkiksi jos pinta sisältää reiän, voi toinen silmukka kiertää pinnan ulkoreunaa ja toinen reiän ympärystä.

Silmukoita pitkin luodaan nelikulmaisista elementeistä muodostuvia rivejä. Näin muodostuu rintamia jotka edetessään täyttävät verkotettavan pinnan. Kun rintamat kohtaavat, ne täytyy sovittaa toisiinsa. Tällöin voi syntyä uusia silmukoita tai silmukat voivat yhdistyä.

Lober ym. [35] mainitsevat joitakin menetelmän vahvuuksia. Menetelmän muodostama verkko on hyvälaatuinen erityisesti reunojen lähettyvillä, eikä lopputulos riipu pinnan asennosta avaruudessa. Menetelmä on myös osoittautunut joustavaksi tilanteissa, joissa siirrytään karkeammasta verkosta tiheämpään. Tämä tekee siitä soveliaan verkon adaptiiviseen tihentämiseen.

Owenin [39] mukaan menetelmää on yritetty yleistää myös kolmiulotteisen sisäverkon luomiseen. Tekniikkaa kutsutaan nimellä plastering. Siinä lähdetään liikkeelle alueen reunalta ja luodaan kuusitahokaselementtejä projisoimalla nelikulmioita kohti alueen keskustaa. Algoritmin ongelmaksi muodostuu kuitenkin se, että alueen keskustaan voi jäädä monimutkaisen muotoisia tyhjiä alueita, joita ei välttämättä edes voida täyttää pelkillä kuusitahokkailla. Menetelmä ei olekaan osoittautunut sellaisenaan kovin toimivaksi yleisessä tapauksessa.

4 Verkkojen muokkaus ja parantaminen

Automaattiset verkon generointialgoritmit voivat tuottaa huonolaatuisia elementtejä, jotka lisäävät ratkaisun numeerista virhettä ja voivat huonontaa konvergenssia. Tämän vuoksi verkon generointiin liittyy usein myös verkon parantamisvaihe. Elementtien laadun parantamista kutsutaan usein verkon silotukseksi (*engl. smoothing*).

Ratkaisun tarkkuutta huonontaa myös se, jos verkko on liian harva sellaisilla alueilla, missä ratkaisussa tapahtuu nopeita muutoksia. Parempaan ratkaisuun voidaankin pyrkiä arvioimalla saadun ratkaisun virhettä elementeittäin ja tihentämällä verkkoa siellä, missä virhettä esiintyy eniten. Tällaista menettelyä kutsutaan verkon adaptiiviseksi tihentämiseksi (*engl. adaptive mesh refinement*).

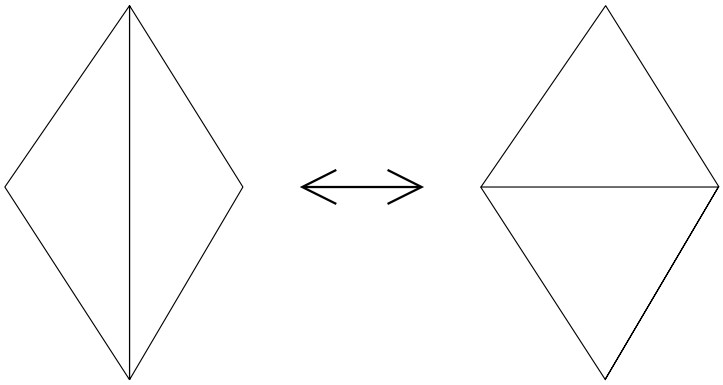
4.1 Verkon silotus

Verkon silotusmenetelmät voidaan jakaa karkeasti kahteen luokkaan: menetelmät, jotka muuttavat verkon topologiaa ja menetelmät, jotka ainoastaan muuttelevat solmupisteiden paikkoja.

4.1.1 Topologiset muutokset

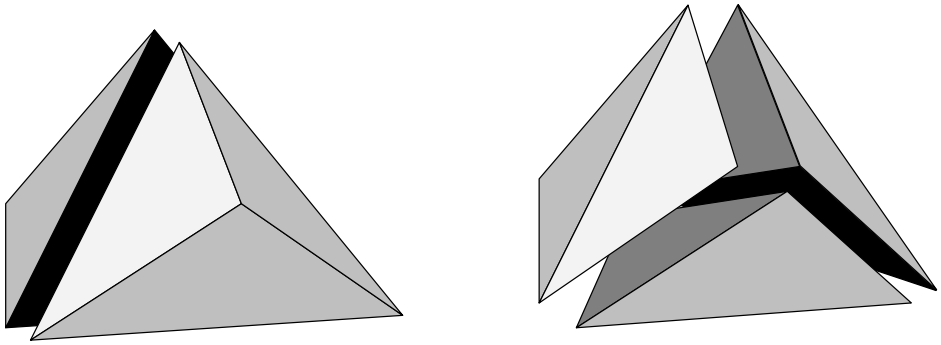
Verkosta voidaan pyrkiä hävittämään huonoja elementtejä lisäämällä solmuja sopiviin paikkoihin ja suorittamalla verkotus lokaalisti uudelleen. Esimerkiksi Li [32] käyttää pisteiden lisäystä algoritmissaan, jonka avulla Delaunay-verkosta voidaan poistaa lastut. Lisäysalgoritmi on kuitenkin suunniteltava huolellisesti, koska pisteiden lisääminen voi synnyttää uusia huonoja elementtejä.

Jo olemassa olevista solmupisteistäkin voidaan muodostaa elementtejä usealla eri tavalla. Esimerkiksi kaksiulotteisessa kolmioverkossa vierekkäiset kolmiot muodostavat nelikulmion. Mikäli nelikulmio on konveksi, kolmioiden yhteinen särmä muodostaa sille diagonaalin. Tällöin voidaan soveltaa kuvassa 4.1 esitettävää diagonaalimuunnosta (*engl. diagonal swap, edge swap*). Muunnoksessa poistetaan kolmioiden yhteinen särmä ja lisätään särmä kahden muun solmun väliin, jolloin saadaan toinen tapa jakaa kyseinen nelikulmio kolmioiksi.



Kuva 4.1: Diagonaalimuunnoksen periaate.

Kolmiulotteisen nelitahokasverkon tapauksessa on tutkittava viiden solmun muodostamia ryhmiä. Ryhmän solmut voivat sijaita toisiinsa nähden viidellä eri tavalla, joista kahdessa on mahdollista soveltaa jonkinlaista lokaalia muunnosta [23]. Niin sanottua 2-3 muunnosta on havainnollistettu kuvassa 4.2.



Kuva 4.2: Viidestä sopivasti sijaitsevasta solmusta voidaan muodostaa joko kaksi tai kolme nelitahokasta.

Joe [23] esittää algoritmin, joka muunnosten avulla parantaa verkon elementtien laatua ja poistaa lastuja. Muunnoksia voidaan tällöin kuitenkin joutua soveltamaan useita kertoja peräkkäin siten, että elementtien laatu väliaikaisesti laskee.

Elementtejä voidaan myös osittaa tai yhdistellä. Näin voidaan pyrkiä lisäämään tietyn tyyppisten elementtien määrää verkossa. Esimerkiksi Staten ja Canann [60] poistavat nelikulmioverkon tihentämisessä syntyvät kolmiot yhdistelemällä niistä nelikulmioita. Elementtejä jakamalla syntyvät elementit eivät useinkaan ole laadultaan kovin hyviä, varsinkaan jos jakamista suoritetaan useita kertoja peräkkäin.

4.1.2 Laplace-silotus

Yleiskäyttöisyytensä ja helpon toteutettavuutensa ansiosta Laplace-silotus eri muodoissaan on eräs yleisimpiä silotusmenetelmiä [30]. Menetelmän perusmuodossa kukin solmupiste siirretään sen naapurisolmujen geometriseen keskipisteeseen. Tätä toistetaan iteratiivisesti, kunnes haluttu kriteeri on saavutettu. Naapurisolmuja voidaan myös painottaa esimerkiksi niihin liittyvien elementtien pinta-alojen mukaisesti, tai käyttää naapurisolmujen sijasta viereisten elementtien keskipisteitä [7]. Operaatiot ovat lokaaleja, laskennallisesti helppoja, ja ne on määritelty myös kolmessa ulottuvuudessa.

Rakenteellisille verkoille menetelmän perusmuoto voidaan johtaa Laplace-yhtälön ratkaisusta [30]. Menetelmä voidaan nähdä myös yhtälöryhmien

$$\sum_{i \in N_j} (x_i - x_j) = 0 \qquad \sum_{i \in N_j} (y_i - y_j) = 0 \qquad (\text{ ja } \sum_{i \in N_j} (z_i - z_j) = 0\,)$$

iteratiivisena ratkaisemisena. Tässä x_j, y_j ja z_j ovat solmun j koordinaatit, N_j on solmun j naapurisolmujen indeksien joukko, ja j käy läpi kaikki verkon sisäsolmut.

Koska menetelmät eivät pisteiden sijoittelulla pyri kuitenkaan maksimoimaan mitään tiettyjä laatukriteereitä, niiden toiminta yleisessä tapauksessa on pikemminkin heuristista [17]. Usein verkon laatu tätä menetelmää soveltamalla paranee, mutta välttämättä näin ei kuitenkaan käy [7]. Menetelmän soveltaminen ilman eksplisiittisiä tarkistuksia voi jopa johtaa elementteihin, joiden pinta-ala (tilavuus) on negatiivinen. Ongelmallisia ovat erityisesti alueen konkaavit reunat, joiden kohdalla syntyy helposti nurjia elementtejä, ja joiden läheisyyteen myös pyrkii kasaantumaan pisteitä.

Silotus ei myöskään välttämättä säilytä verkkojen Delaunay-ominaisuutta. Tämä on ongelma erityisesti, jos verkkoa halutaan myöhemmin tihentää menetelmällä, joka toimii ainoastaan Delaunay-verkoilla. Ongelma voidaan kuitenkin kiertää suorittamalla solmujen siirtäminen seuraavasti [13]:

- Lasketaan solmun uusi sijainti $\mathbf{x}^*$.
- Tutkitaan säilyttääkö solmun siirto uuteen paikkaan Delaunay-ominaisuuden.
  - Jos säilyttää, niin siirretään solmu pisteeseen $\mathbf{x}^*$.
  - Jos ei säilytä, niin poistetaan solmu, kolmioidaan alue lokaalisti uudelleen ja lisätään solmu pisteeseen $\mathbf{x}^*$ Delaunayn ytimen avulla.

Menetelmää voidaan jalostaa siirtämällä solmu uuteen paikkaan vain, jos tietyt ehdot täyttyvät (katso esim. [8]). Tällöin puhutaan rajoitetusta Laplace-silotuksesta

(*engl. constrained Laplacian smoothing*). Tällöin voidaan varmistaa, että siirto ei synnytä nurjia elementtejä tai muuten tee verkosta huonompaa.

4.1.3 Winslow-silotus

Winslow [65] on esittänyt menetelmän rakenteellisen kolmioverkon generoimiseksi yhtälöiden (3.1) avulla. Knupp [30] yleistää menettelyn myös rakenteettomille verkoille, joiden solmut eivät sijoitu mihinkään esimerkin 3.2 kaltaiseen käyräviivaiseen koordinaatistoon. Menetelmässä approksimoidaan yhtälöitä lokaalisti differenssimenetelmällä ja ratkaistaan syntynyt epälineaarinen yhtälöryhmä. Hän johtaa kaavat rakenteettoman nelikulmioverkon tapauksessa ja hahmottelee vastaavaa menettelyä kolmioverkoille. Hän toteaa, että Winslow-silotus ei johda nurjiin elementteihin yhtä helposti kuin Laplace-silotus.

Menetelmä toimii sellaisenaan ainoastaan kahdessa ulottuvuudessa, koska diskretisoinnissa tarvittavaa ”kiertosuuntaa” ei voida määrittää kolmessa ulottuvuudessa. Diskretisointi voidaan kuitenkin tehdä elementtimenetelmällä samaan tapaan kuin viitteessä [1].

4.1.4 Optimointiin perustuva silotus

Käyttämällä optimointia solmupisteiden sijoittelussa voidaan taata, että verkon laatu todella paranee ja verkko säilyy sallittuna [7]. Optimointi voidaan tehdä globaalisti eli optimoimalla kerralla kaikkien liikuteltavissa olevien solmupisteiden paikkoja, mutta tällaiset menetelmät ovat usein laskennallisesti hyvin raskaita. Toisaalta elementtejä voidaan parantaa lokaalisti huonousjärjestyksessä, kunnes verkon huonoinkin elementti täyttää annetut vaatimukset.

Jos verkon laatua mitataan sen huonoimman elementin laadulla, on optimointitehtävä muotoa

$$\max \min_K Q_K$$

missä $0 \leq Q_K \leq 1$ on elementin K laatu. Funktion tarkka lauseke riippuu luonnollisesti käytetystä laatukriteeristä.

Zavattieri ym. [66] käyttävät myös differentioituvaa versiota

$$\min \sum_K (1 - Q_K)^p$$

mutta heidän tuloksensa eivät osoita siitä saatavan juurikaan hyötyä verrattuna huolellisesti implementoituun epäsileään optimointiin. Testiesimerkkien avulla he myös to-

teavat, että lokaaliin optimointiin riittää ottaa mukaan solmut, jotka ovat enintään yhden tai kahden särmän päässä huonoimmasta elementistä.

Lokaali optimointi on helppo yhdistää rajoitettuun Laplace-siloitukseen, kuten Canann [8] ja Freitag [16] tekevät. Freitag esimerkiksi käyttää tavoitteellista laatutasoa, jolle huonoimman elementin laatu pyritään nostamaan. Hän esittää menetelmästään neljä eri muunnelmaa, joissa solmuille suoritetaan Laplace-menetelmän mukainen siirto, optimointi tai molemmat peräkkäin.

Hän toteaa, että yhdistetyillä menetelmillä päästään jopa parempiin tuloksiin kuin pelkällä Laplace-siloituksella tai pelkällä optimoinnilla. Menetelmien suoritus on lisäksi selkeästi pelkkää optimointia nopeampaa.

4.2 Tehtävästä riippuva parantaminen

Ratkaistavalla tehtävällä on suuri merkitys siihen, millainen laskentaverkon tulisi olla. Tiheämpi laskentaverkko tuottaa yleisesti ottaen tarkempia ratkaisuja, mutta samalla se lisää tarvittavan laskennan määrää. Olisikin siis laskennallisesti tehokasta käyttää tiheää laskentaverkkoa sellaisilla alueilla, joissa ratkaisussa tapahtuu nopeita muutoksia. Aina ei kuitenkaan tiedetä etukäteen, missä tällaiset alueet sijaitsevat. Tämän vuoksi laskenta suoritetaan joskus adaptiivisesti siten, että verkkoa muokataan laskentojen välillä edellisen ratkaisun perusteella.

Saadusta ratkaisusta voidaan laskea niin sanottuja a posteriori -virhearvioita (*engl. a posteriori error estimates*). Niitä voidaan käyttää hyväksi verkon tihentämisessä ja silotuksessa (katso esimerkiksi [4]). Virhearvion avulla voidaan muodostaa myös iteratiivisen ratkaisemisen lopetuskriteeri, mutta tällöin arvion täytyy olla jossakin mielessä tarkka, eli sille täytyy tuntea jonkinlaiset ala- ja ylärajat.

Virhearviot voivat perustua esimerkiksi ratkaisun residuaaliin. Suoraan ratkaisusta laskettua gradienttia voidaan myös verrata jollakin lailla ”tarkennettuun” gradienttiin. Suoraan ratkaisusta ei nimittäin yleensä saada kovin hyviä arvioita ratkaisun gradienteille. Jos saatu ratkaisu on vaikkapa paloittain lineaarinen, on sen gradientti enää paloittain vakio. Perinteinen tapa on silottaa gradienttia keskiarvoistamalla tai L^2-projektion avulla. Zienkiewicz ja Zhu [67], [69] esittävät menetelmän, joka perustuu arvojen laskemiseen niin sanotuissa superkonvergenssipisteissä tai muissa optimaalisissa pisteissä. Niiden avulla gradientille voidaan saada kertaluokkaa tarkempi arvio, jota he käyttävät virhearvion muodostamiseen.

Verkon tihentäminen voidaan tehdä myös epätarkemman virheindikaattorin (*engl.*

error indicator) avulla. Virheindikaattori kertoo, minkä elementtien alueella ratkaisussa esiintyy eniten virhettä. Virheindikaattori ei kuitenkaan (välttämättä) kerro, kuinka suuri tämä virhe on. Erilaisia virheindikaattoreita on vertailtu esimerkiksi artikkelissa [38].

Verkon paikallisesta tihentämisestä

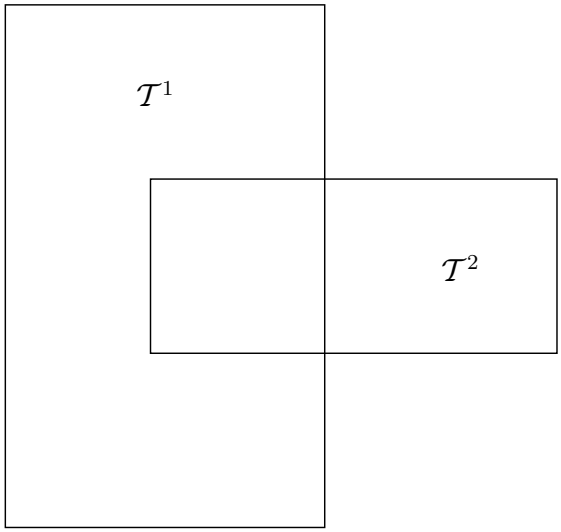
Rakenteettomat verkot sopivat rakenteellisia paremmin verkon paikalliseen tihentämiseen. Jos nimittäin jokin rakenteellisen verkon elementti jaetaan osiin, on myös sen naapurit jaettava vastaavasti, jotta verkon rakenteellisuus säilyisi. Tihennys leviää siis tiettyyn suuntaan läpi koko verkon.

Tihennys tulisi suorittaa siten, että syntyvät elementit ovat laadultaan mahdollisimman hyviä. Erityistä huolellisuutta on noudatettava, mikäli verkkoa tihennetään useita kertoja peräkkäin, kuten usein iteratiivisesti tehdään. Kolmio- ja nelitahokasverkkojen tihennysalgoritmeja ja niiden vaikutusta syntyvän verkon laatuun käsitellään artikkelissa [24]. Pulakan pro gradu -työssä [43] puolestaan käsitellään algoritmien käytännön toteutusta ja niiden tietorakenteille asettamia vaatimuksia.

5 Verkoilla operointi

Tässä luvussa kuvataan menetelmiä, joiden avulla verkoilla voidaan suorittaa loogisia operaatioita eli yhdisteitä, leikkauksia ja poistoja. Tämän tutkielman yhteydessä toteutettiin verkkojen yhdistämisen suorittava algoritmi, jota kuvataan tarkemmin luvussa 5.1. Muiden operaatioiden mahdollisia toteutustapoja hahmotellaan lyhyesti luvussa 5.2. Lopuksi luvussa 5.3 esitellään toteutuksessa käytettyjä tietorakenteita.

Operaatioiden periaatteita esitellään muun muassa poikkileikkausten avulla. Lähtötilanteen poikkileikkaus oletetaan kuvan 5.1 kaltaiseksi. Operaatioihin osallistuvat verkot siis menevät selkeästi päällekkäin.



Kuva 5.1: Poikkileikkaus operaatioiden lähtötilanteesta.

5.1 Yhdiste

Parametrisoitujen pintojen leikkauskäyrän määräämisongelmaa on tutkittu varsin paljon. Leikkauksen määrääviä algoritmeja on olemassa useita — myös tilanteisiin, joissa pinnat ovat vain C^0-jatkuvia [47]. Pintojen yhdiste taas on suhteellisen suoraviivaista muodostaa, kun niiden leikkauskäyrä tunnetaan.

Verkoille tilanne on vaikeampi, koska määritelmän 2.1 mukainen jatkuvuus täytyy säilyä myös leikkauskäyrällä. Yleisesti verkkojen liittäminen vaatii olemassa olevien verkkojen purkamista liitosalueen läheisyydessä ja jäljelle jäävien verkkojen sovittamista toisiinsa.

Esimerkiksi Lo [34] esittää algoritmin kolmioista muodostuvien pintaverkkojen yhdistämiseksi. Se ei käsittele sisäverkkoja eikä muuta pinnasta mahdollisesti annettua informaatiota. Shostko ym. [56] lähtevät liikkeelle Lon algoritmista mutta toteavat, ettei käytetty advancing front -menetelmä toimi luotettavasti kaikissa heidän esimerkeissään. Sen vuoksi he muokkaavat algoritmia siten, että uudelleenverkotus suoritetaankin kunkin leikkauskäyrällä sijaitsevan kolmion sisällä. Tämä kuitenkin johtaa verkon tihenemiseen leikkauskäyrän läheisyydessä.

Coelho, Gattass ja Figueiredo [11] taas esittävät algoritmin, joka yhdistää pintaverkkoja siten, että leikkauskäyrien määräämiseen käytetään myös pintojen parametriesitystä. Lira ym. [33] laajentavat algoritmia käsittelemään useiden pintojen yhtäaikaista leikkausta.

Hieman erilaista lähestymistapaa edustavat Cebral ym. [9]. He käyttävät taustaverkkoa, jossa he asettavat skalaarifunktion kuvaamaan pisteen etäisyyttä käsiteltävistä pinnoista. Näin pintojen leikkauksen määrääminen palautuu tasa-arvopinnan määräämiseksi taustaverkossa.

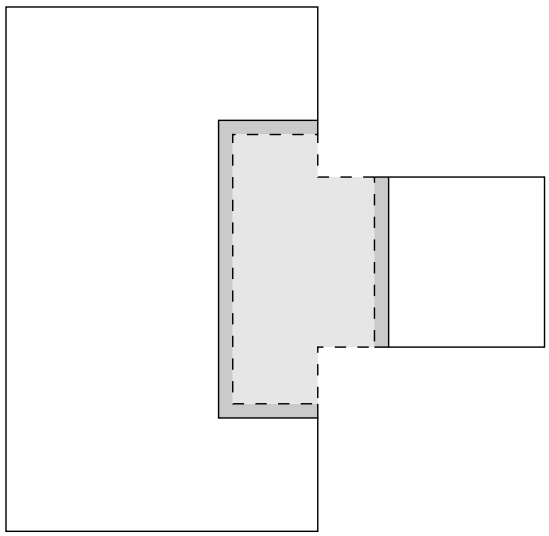
Tässä työssä mukaillaan lähinnä lähteiden [11] ja [34] algoritmeja. Työssä ei aseteta rajoituksia yhdistettävien verkkojen tai niiden pintojen elementtityypeille. Ei myöskään oleteta, että pinnoista olisi käytettävissä jonkinlainen parametriesitys. Verkkojen ei välttämättä tarvitse olla rakenteellisia, vaikka tämän tyyppisen yhdistämisen suurin hyöty onkin juuri siinä, että verkkojen olemassa oleva rakenne pystytään osin säilyttämään. Työssä ei rajoituta operoimaan pelkästään pintaverkoilla, vaan liitetään myös tilavuusverkot toisiinsa.

Verkkojen yhdistämisen periaate näkyy kuvassa 5.2. Verkkoja muokataan ainoastaan niiden pintaverkkojen leikkauksen (eli niin sanotun leikkauskäyrän) läheisyydessä. Tälle alueelle tehdään tilaa rakenteettomalle liimakerrokselle purkamalla molemmista verkoista tällä alueella sijaitsevat elementit. Pintaverkot liitetään toisiinsa tätä varten suunnitellulla advancing front -tyyppisellä pintaverkon generointimenetelmällä.

Poistetun alueen reunoilla olevia elementtejä täytyy vielä muokata siten, että alueen reunat muodostuvat ainoastaan kolmioista. Tämä tehdään käytännössä elementtejä pilkkomalla. Muokattua aluetta on kuvassa merkitty tummemmalla harmaalla.

Liitosalueelle voidaan tämän jälkeen generoida tilavuusverkko jollakin rakenteetto-

mia verkkoja luovalla generaattorilla. Kuvan katkoviivat kuvaavat generaattorin syötteenä toimivaa pintaverkkoa. Syntyvää rakenteetonta liimaverkkoa on kuvassa merkitty vaaleammalla harmaalla. Yhtenäisen viivan rajaama alue säilyy kummassakin verkossa ennallaan.



Kuva 5.2: Yhdisteen periaate poikkileikkauksena.

Yhdistämisen eteneminen on kuvattu tarkemmin algoritmissa 5.1, ja menettelyn yksityiskohdista kerrotaan seuraavissa alaluvuissa.

Algoritmi 5.1 Verkkojen yhdistäminen.

1. Lasketaan kummankin verkon reunasärmien leikkauspisteet toisen verkon tahkojen kanssa.

2. Linkitetään pintatahkoilla sijaitsevat leikkauspisteet paloittain lineaariseksi käyräksi.

3. Tuotetaan leikkauskäyrälle uusia solmuja tietyssä mielessä tasaisin välein.

4. Puretaan molemmista verkoista elementit, jotka leikkaavat toista verkkoa tai joiden läheisyydessä on leikkauspiste.

5. Muodostetaan pintaverkkojen yhdiste, eli luodaan kolmioverkko lähtien liikkeelle pintaverkkojen niistä särmistä, jotka sijaitsevat poistetun alueen reunalla.

6. Muokataan sisäverkkojen poistetun alueen reunalla sijaitsevia elementtejä siten, että niiden poistetun alueen puoleinen pinta muodostuu kolmioista.

7. Luodaan poistettuun alueeseen rakenteeton verkko yleisellä rakenteettomia nelitahokasverkkoja luovalla generaattorilla. Syntyvästä verkosta käytetään jatkossa termiä liimaverkko, koska se ikään kuin liimaa erilliset verkon osat yhteen.

8. Yhdistetään jäljelle jääneet verkkojen osat ja syntynyt rakenteeton verkko samaan tietorakenteeseen.

5.1.1 Leikkauspisteiden määrääminen

Kaikissa verkoilla operoinneissa tarvitaan särmän ja tahkon leikkauspisteen määräävää algoritmia. Sen tulee olla mahdollisimman nopea, koska se joudutaan suorittamaan lukuisia kertoja, mutta samalla sen tulee olla ehdottoman luotettava. Mukavuussyistä määritellään seuraavasti:

Määritelmä 5.1 Särmät, tahkot ja elementit ovat suljettuja joukkoja.

Leikkauspisteiden laskemisessa voidaan nyt käyttää pientä toleranssia, niin että kaikki leikkauspisteet varmasti löydetään. Tästä kuitenkin seuraa, että sama leikkauspiste voi löytyä useammaltakin tahkolta, mikä täytyy ottaa muissa algoritmeissa huomioon.

Kolmiotahkon ja suoran särmän leikkauspistettä määrättäessä työtä voidaan helpottaa tekemällä tiettyjä tarkistuksia ennen kuin leikkauspistettä lähdetään laskemaan. Kannattaa esimerkiksi varmistaa, että särmän päätepisteet ovat kolmion solmujen määräämän tason eri puolilla. Kolmion ja särmän solmuista voidaan myös muodostaa nelitahokkaita, joiden ”tilavuuden” merkistä voidaan päätellä, onko leikkauspiste kolmion sisällä [56].

Yleisessä tapauksessa tahkon solmut eivät kuitenkaan sijaitse tasossa, jolloin edellä mainittua testiä ei voida käyttää. Silloinkin kannattaa kuitenkin testata, että särmän ja tahkon rajoitelaatikot (katso määritelmä 2.4) menevät limittäin.

Varsinainen leikkauspisteiden laskeminen voidaan tehdä esimerkiksi elementtimenetelmästä tuttujen kantafunktioiden (katso esimerkiksi [68]) avulla. Tahko voidaan parametrisoida kahdella lokaalilla koordinaatilla u ja v sekä särmä yhdellä koordinaatilla t. Ratkaistavaksi tulee tällöin kolmen yhtälön ryhmä

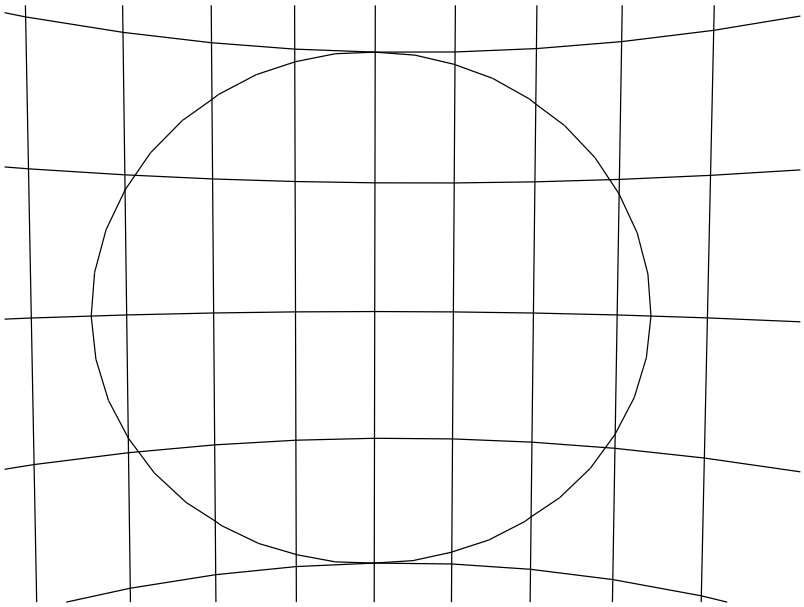
$$r_j = \sum_{i=1}^{N} \phi_i(u,v) x_{ij} - E(t)_j = 0 \qquad j = 1,2,3 \tag{5.1}$$

missä x_{ij} on solmun i j:s koordinaatti, ϕ_i on kyseiseen solmuun referenssielementissä liittyvä kantafunktio ja $E(t)_j$ on särmän parametrisoinnin j:s koordinaatti. Yhtälöryhmä on yleisessä tapauksessa epälineaarinen.

Toteutetussa algoritmissa yhtälöryhmä ratkaistiin Newtonin menetelmällä. Residuaalin r_j Jacobin matriisi on 3×3 -matriisina varsin helposti käännettävissä, joten menetelmässä ei tarvita matriisihajotelmia.

5.1.2 Leikkauskäyrän määrääminen

Leikkauskäyrällä tarkoitetaan tässä yhteydessä janoista muodostuvaa käyrää, joka syntyy, kun edellä lasketut pintaverkkojen leikkauspisteet yhdistetään järjestyksessä toisiinsa. Esimerkki leikkauskäyrästä näkyy kuvassa 5.3. Käyrä liittyy luvun 6 esimerkkiin 2.



Kuva 5.3: Kahden verkon leikkauskäyrä piirrettynä toisen verkon pinnalle.

Huomautus 5.1 Tässä työssä oletetaan käsiteltävien verkkojen menevän päällekkäin siten, että niiden pintaverkkojen leikkaus on yksi jatkuva käyrä.

Mainitun jatkuvuusoletuksen vallitessa voidaan käyttää hyväksi havaintoa, että leikkauskäyrä on jatkuva myös kunkin reunatahkon alueella. Tällöin, kun käyrä tulee tahkon alueelle leikkaamalla jotakin sen särmää, se myös poistuu tahkon alueelta leikkaamalla sen jotakin toista särmää, kuten kuvasta 5.3 nähdään.

Leikkauspisteet saadaan järjestettyä kulkemalla pintaverkoilla siten, että tahkosta siirrytään seuraavaan sen mukaan, millä tahkon särmistä jälkimmäinen leikkauspiste sijaitsee. Kunkin tahkon sisällä leikkauspisteet voidaan järjestää kulkemalla vastaavalla tavalla läpi toista pintaverkkoa.

Ongelmia muodostavat tilanteet, joissa leikkauspiste sijaitsee täsmälleen tahkon solmupisteessä. Tällöin leikkauskäyrä ei varsinaisesti kulje minkään tahkon läpi, eikä ole a priori selvää, mihin tahkoon seuraavaksi tulee siirtyä. Tällaiset tilanteet täytyykin käsitellä erikseen tutkimalla molempien verkkojen vuorossa olevien tahkojen ja niiden naapureiden keskinäisiä leikkauksia. Tällöin myös samaan leikkauspisteeseen liittyviä särmiä ja tahkoja on useita, eikä pelkkien indeksien perusteella voida tutkia, onko kyseessä uusi vai vanha leikkauspiste. Tämän vuoksi leikkauspisteen koordinaatteja täytyy vertailla aikaisempien pisteiden koordinaatteihin.

Vertailuun liittyy kuitenkin numeerisia riskejä. Jos nimittäin lasketaan lähes samassa tasossa sijaitsevien särmän ja tahkon leikkauspistettä, muodostuu yhtälöryhmästä (5.1) huonosti käyttäytyvä ja ratkaisuun voi syntyä numeerista virhettä. Tällaiset tilanteet pyrittiin toteutuksessa eliminoimaan jättämällä leikkauspiste kokonaan määräämättä, mikäli yhtälöryhmän residuaalin Jacobin determinatti oli itseisarvoltaan pieni. Näin voidaan tehdä, koska särmän päätepisteisiin liittyy muita särmiä, jotka ovat leikkauspisteen laskemisen kannalta paremmassa kulmassa tahkoon nähden. Kyseinen leikkauspiste löydetään siis näiden särmien leikkauspisteenä.

5.1.3 Solmujen tuottaminen leikkauskäyrälle

Jotta puretulle alueelle voitaisiin tuottaa uusi pintaverkko, on leikkauskäyrälle luotava uusia solmuja. Solmut olisi sijoiteltava siten, että

- niitä on tiheässä siellä, missä alkuperäiset pintaverkot ovat tiheitä, ja harvassa siellä, missä ne ovat harvoja,
- ne sijoittuvat jossakin mielessä tasaisesti, eli kaksi solmua ei sijaitse erittäin lähellä toisiaan,
- paikoissa, joissa leikkauskäyrä tekee jyrkän kulman, sijaitsee jatkossakin solmu, ja
- niitä on sopiva määrä siten, että siirtymä purettujen osien reunoilta leikkauskäyrälle ei ole liian jyrkkä.

Viimeinen vaatimus toteutuisi täydellisesti, jos solmujen välimatkat olisivat leikkauskäyrällä samat kuin alkuperäisissä pintaverkoissa. Yleisesti tämä on kuitenkin ristiriitainen vaatimus, sillä särmien pituudet voivat toisessa liitettävässä verkossa olla hyvin erilaiset kuin toisessa. Lisäksi jopa vierekkäisten särmien pituudet voivat olla hyvin erilaiset.

Toteutetussa algoritmissa vaatimukset pyritään toteuttamaan likimääräisesti käyttämällä seuraavanlaista heuristiikkaa:

Algoritmi 5.2 Solmujen tuottaminen leikkauskäyrälle.

1. Asetetaan alkujoukoksi kaikki pintaverkkojen särmien ja tahkojen leikkauspisteet. Olkoon niitä n_a kappaletta.

2. Lasketaan solmujen ideaalinen lukumäärä. Merkitään $|rs_i| :=$ niiden särmien lukumäärä, jotka jäävät poistetun alueen reunalle verkossa i, $||rs_i|| :=$ kyseisten särmien yhteenlaskettu pituus ja $||lk|| :=$ leikkauskäyrän pituus. Määritellään ideaalinen keskimääräinen välimatka

$$l = \left(\frac{||rs_1||}{|rs_1|} + \frac{||rs_2||}{|rs_2|} \right) /2$$

   ja edelleen solmujen ideaalinen lukumäärä

$$n = \left\lfloor \frac{||lk||}{l} \right\rfloor$$

3. Lukitaan sellaiset solmut, joiden kohdalla leikkauskäyrä tekee jyrkän kulman. Olkoon solmuun liittyvien janojen välinen kulma α. Kulma katsotaan jyrkäksi, jos $\cos(2\pi - \alpha) < 0.96$. Käytännössä kulma lasketaan käyrän peräkkäisten janojen normalisoituna pistetulona.

   Olkoon lukittuja solmuja n_l kappaletta. Jos $n_l \geq n$, muita solmuja ei tarvita ja algoritmi voidaan lopettaa. Lukittujen solmujen paikat säilytetään sellaisenaan. Loppuja solmuja, joita on siis $n_a - n_l$ kappaletta, kutsutaan vapaiksi.

4. Jätetään vapaista solmuista osa pois siten, että vapaiden ja lukittujen solmujen lukumääräksi tulee lopulta yhteensä n. Tämä tehdään jättämällä solmuista pois ne, joiden indeksi on $\lfloor j(n_a - n_l)/(n_a - n) \rfloor, \quad j = 1, \ldots, (n_a - n)$. Pois tulee todella jätettyä $(n_a - n)$ eri solmua, koska $(n_a - n_l)/(n_a - n) > 1$.

5. Tehdään vapaille solmuille pari silotuskierrosta.

Silotus tehdään Laplace-tyyppisesti siirtämällä kukin solmu vuorollaan sen naapureiden puoliväliin. Solmua ei kuitenkaan nyt voida siirtää naapureidensa geometriseen keskipisteeseen, koska tämä piste ei yleisesti ole enää leikkauskäyrällä. Sen sijaan siirto tehdään parametriesityksen avulla.

Määritellään leikkauskäyrän pisteen parametrinen koordinaatti sen etäisyydeksi jostakin referenssipisteestä leikkauskäyrää pitkin jaettuna koko leikkauskäyrän pituudella. Näin saadut parametriset koordinaatit vaihtelevat nollasta yhteen.

Silotuksessa asetetaan kunkin vapaan solmun parametriseksi koordinaatiksi sen naapureiden parametristen koordinaattien keskiarvo. Tässä naapureiksi lasketaan sekä vapaat että lukitut solmut, vaikka siirrot voidaankin tehdä vain vapaille solmuille.

Erityiskäsittelyä vaaditaan, kun naapurisolmut ovat alkuperäisen referenssipisteen eri puolilla, eli toisen koordinaatti on lähellä yhtä ja toisen lähellä nollaa. Tällöin pienempää koordinaattia voidaan tilapäisesti kasvattaa yhdellä. Saatu keskiarvo tulee lopuksi palauttaa välille $[0, 1]$.

Siirto tehdään kullekin vapaalle solmulle vuorollaan. Silotusta tehdään vain muutama kierros, jotta solmujen paikat jakautuisivat silotuksen jälkeenkin suurin piirtein alkuperäisten verkkojen tiheyksien mukaisesti.

5.1.4 Leikkaavan osan purkaminen

Verkkoja yhdistettäessä on ainakin jommasta kummasta verkosta purettava se osa, joka menee päällekkäin toisen verkon kanssa, jotta yhdisteverkko voisi olla sallittu. Toteutetussa algoritmissa päällekkäin menevät osat puretaan molemmista verkoista, jolloin tilaa liimana toimivalle rakenteettomalle verkolle saadaan enemmän. Tämä ei kuitenkaan ole paras mahdollinen toimintatapa, koska mitä enemmän verkkoja puretaan, sitä vähemmän niiden alkuperäistä rakennetta säilyy. Yhdisteen lopputulos myös riippuu siitä, kuinka paljon alkuperäiset verkot ”sattuvat” menemään päällekkäin.

Seuraavassa oletetaan yksinkertaisuuden vuoksi, että puretaan vain yhtä verkkoa, jota merkitään $\mathcal{T}^1$:llä. Poistettavat elementit määrätään verkon $\mathcal{T}^2$ avulla siten, että verkosta $\mathcal{T}^1$ poistetaan sellaiset elementit K, joille $K \bigcap \mathcal{T}^2 \neq \emptyset$. Jos leikkaava osa halutaan purkaa molemmista verkoista, verkkoa $\mathcal{T}^2$ purettaessa voidaan toimia täysin vastaavasti. Poistettavat elementit tulee tällöin määrätä purkamattoman verkon $\mathcal{T}^1$ avulla.

Toteutetussa algoritmissa lasketaan ensin verkon $\mathcal{T}^2$ pintatahkojen leikkauspisteet verkon $\mathcal{T}^1$ särmien kanssa. Käytännössä tämä tapahtuu käymällä läpi verkon $\mathcal{T}^2$ pintatahkot ja tekemällä tahkojen rajoitelaatikoilla hakuja R-puuhun, johon on talletettu

verkon $\mathcal{T}^1$ särmät. Tuloksena saadaan särmät, joiden rajoitelaatikot leikkaavat kyseistä tahkoa. Särmien ja tahkon mahdolliset leikkauspisteet lasketaan luvussa 5.1.1 esitetyllä tavalla. Tuloksena saaduista lokaaleista koordinaateista nähdään suoraan, onko leikkauspiste todellinen.

Näin löydetään samalla kaikki verkon $\mathcal{T}^1$ elementit, joita pintaverkko $\partial\mathcal{T}^2$ leikkaa, ja ne voidaan poistaa verkosta. Verkkoon voi kuitenkin jäädä elementtejä, jotka ovat mielivaltaisen lähellä leikkauskäyrää, kuten kuvasta 5.3 nähdään. Tällaisiin elementteihin liittyvät liimaverkon elementit olisivat laadultaan erittäin huonoja. Tämän vuoksi myös sellaiset elementit poistetaan, joille $||kp - lp|| < 1.5r$, missä kp on elementin keskipiste, lp on elementtiä lähin leikkauspiste ja r on elementin säde (joka tässä määritellään sen solmujen pisimmäksi etäisyydeksi sen keskipisteestä).

Tämän jälkeen loput poistettavat elementit määrätään seuraavasti: Käytetään apuna *poistettu*-taulukkoa, johon alustetaan ensin kaikki elementit poistetuiksi. Tämän jälkeen tehdään syvyyshaku lähtien liikkeelle jostakin elementistä, joka varmasti ei ole leikkaavassa osassa. Vieraillut elementit merkitään *poistettu*-taulukossa säilytettäviksi. Sellaiseen elementtiin ei mennä, jonka tahkolla sijaitsee jokin lasketuista leikkauspisteistä tai joka sijaitsee leikkauspisteen lähellä. Näin haussa ei voida joutua leikkaavaan osaan. Syvyyshaun pseudokoodi on esitetty algoritmina 5.3.

Algoritmi 5.3 Syvyyshaku leikkauksen ulkopuolisessa osassa.

```
pino(1) ← etsi_aloituspiste()
pino_os ← 1
while pino_os > 0 do
  nykyinen ← pino(pino_os)
  if elementillä nykyinen on käsittelemättömiä naapureita then
    seuraava ← elementin nykyinen seuraava käsittelemätön naapuri
    if not leikattu(seuraava) then
      poistettu(seuraava) ← false
      pino_os ← pino_os+1
      pino(pino_os) ← seuraava
    else
      pino_os ← pino_os-1
    end if
  end if
end while
```

Huomautus 5.2 Työssä oletetaan, että leikkauksen ulkopuolelle jäävä verkon osa on yhtenäinen. Mikäli näin ei ole, käytettävä syvyyshaku löytää ainoastaan yhden yhtenäisen komponentin, ja loput tulisivat poistettua. Epäyhtenäinen verkko tulisi käsitellä erikoistapauksena myös liitosoperaation muissa vaiheissa, joten se rajataan tämän työn ulkopuolelle.

Käytännössä pinossa säilytetään elementtien indeksien lisäksi myös viimeksi tutkitun naapurin järjestysnumeroa, jolloin naapureita ei tarvitse selata läpi, jotta tiedettäisiin mikä niistä on seuraavaksi vuorossa. Näin ollen kullekin pinon elementille suoritetaan vain sen naapureiden läpikäynti, eikä vierailtuja elementtejä laiteta pinoon enää uudestaan. Tällöin syvyyshaun aikavaativuus on $\mathcal{O}(n+m)$, missä n on leikkauksen ulkopuolelle jäävien elementtien lukumäärä ja m kyseisten elementtien naapureiden lukumäärien summa.

Alkupiste löydetään kulkemalla jostakin pintaverkon $\partial\mathcal{T}^2$ leikkauspisteestä lähtien pinnan normaalista poispäin. Nelikulmiotahkon tapauksessa normaalivektori ei yleisesti ole sama koko tahkon alueella. Kohtuullinen approksimaatio saadaan kuitenkin kaavalla

$$\vec{n} = \frac{(\vec{x_3} - \vec{x_1}) \times (\vec{x_4} - \vec{x_2})}{||(\vec{x_3} - \vec{x_1}) \times (\vec{x_4} - \vec{x_2})||} \ ,$$

missä $\vec{x_i}$:t ovat tahkon solmupisteet lueteltuna pinnan sisältä katsottuna vastapäivään.

5.1.5 Pintaverkkojen yhdistäminen

Pintaverkot yhdistetään luomalla puretulle alueelle rakenteeton kolmioverkko lähtien liikkeelle sellaisista pintasärmistä, jotka sijaitsevat puretun osan reunalla. Verkon solmuja ovat näiden särmien solmut ja leikkauskäyrälle luodut uudet solmut. Muita solmuja ei luoda.

Vaaditaan siis, että puretun alueen reunoilla olevat särmät esiintyvät verkossa sellaisenaan. Niitä ei voida esimerkiksi jakaa, koska vastaava jako tulisi tehdä myös jäljelle jäävän verkon elementeille, mikä ei ole tarkoituksenmukaista. Samoin vaaditaan, että leikkauskäyrälle luotujen vierekkäisten solmujen välillä on särmä. Näin verkotus voidaan tarvittaessa tehdä kummassakin verkossa erikseen, koska syntyvän verkon reunat on kiinnitetty.

Verkko muodostetaan Lon [34] algoritmia mukaillen advancing front -tyyppisellä menetelmällä. Rintamaan lisätään aluksi kaikki poistetun alueen reunoilla olevat pintasärmät ja leikkauskäyrän vierekkäisten solmujen väliin luotavat särmät. Verkotus ete-

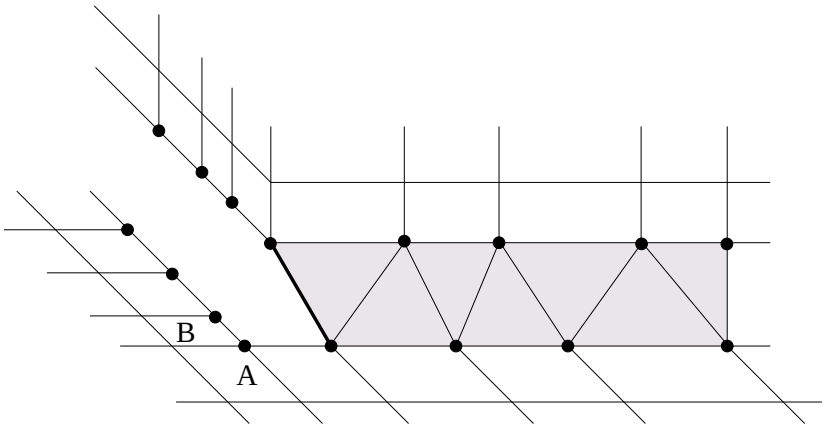
nee siten, että rintamasta valitaan jokin pohjasärmä, josta lähtien yritetään muodostaa uutta elementtiä jonkin kolmannen solmupisteen kanssa.

Jokaista pohjasärmää kohti tutkitaan kaikki käytettävissä olevat solmupisteet. Solmupiste on sallittu, jos sen avulla muodostettava elementti täyttää seuraavat ehdot:

1. sen ja jonkin sen naapurielementin pinnat ovat lähes yhdensuuntaiset,
2. muodostuvat uudet särmät eivät ”leikkaa” olemassa olevia verkon särmiä, ja
3. yhtään verkon solmupistettä ei jää elementin ”sisään”.

Ensimmäisen vaatimuksen avulla pyritään varmistamaan, että muodostuva elementti ylipäätään on järkevä. Suhteellisen sileillä pinnoilla nimittäin jokaisella elementillä on naapuri, joka sijaitsee sen kanssa lähes samassa tasossa. Solmu, jonka avulla muodostettu elementti ei totetuta tätä ehtoa, sijaitsee luultavasti kokonaan väärässä suunnassa. Pinnoissa voi kuitenkin esiintyä tasaista käyryyttä (esimerkiksi pallon tapauksessa), jolloin vierekkäisten elementtien normaalit aina poikkeavat toisistaan.

Sopivan raja-arvon löytäminen on ongelmallista. Liian väljä raja voi nimittäin aiheuttaa valintoja, jotka myöhemmin tuottavat huonoja tai jopa virheellisiä elementtejä. Esimerkiksi kuvan 5.4 tilanteessa solmut A ja B saattavat kumpikin olla muiden vaatimusten mielessä sallittuja, mutta jos algoritmi tässä vaiheessa valitsee solmun B, ei solmua A enää myöhemmin voida liittää verkkoon järkevällä tavalla. Toisaalta liian tiukat rajat voivat johtaa tilanteisiin, joissa algoritmi pysähtyy, vaikka tilanteeseen olisikin olemassa sallittu ratkaisu.



Kuva 5.4: Verkotettavan alueen jyrkkien kulmien kohdalla on vaarana, että algoritmi ”oikaisee”.

Työssä ei asetettukaan poikkeamalle mitään tiukkaa rajaa, vaan käytettiin seuraavanlaista kustannusfunktiota:

$$f(elem) = r_{elem} e^{k(1-d_{elem})} \tag{5.2}$$

Tässä r_{elem} on syntyvän elementin ympäröivän pallon säde, d_{elem} elementin ja sen naapureiden normaalivektoreiden pistetuloista suurin ja k vapaasti valittava parametri. Mitä suurempi k valitaan, sitä enemmän sakotetaan siitä, että elementti ei ole samassa tasossa naapureidensa kanssa. Ehdot 2 ja 3 täyttävistä solmuista valitaan lopulta se, jonka kustannus on mainitun funktion mielessä pienin.

Ehto 2 on selvä, jos tilannetta ajattelee kaksiulotteisesti. Vaatimus on tilanteen kolmiulotteisuuden vuoksi kuitenkin tulkittava siten, että muodostettavien särmien täytyy olla määrätyn toleranssin etäisyydellä olemassa olevista särmistä. Sopivan toleranssin löytäminen on jälleen ongelmallista. Jos pinnalla nimittäin on kohtia, jotka ovat hyvin eri tasossa kuin rintama pohjasärmän kohdalla, voi syntyvä elementti oikaista tällaisten kohtien ”ali” tai ”yli”. Toisaalta liian suuri toleranssi voi johtaa ongelmiin tilanteissa, joissa suuresta elementtikoosta joudutaan siirtymään pieneen.

Vertailussa on lisäksi otettava huomioon, että samaan solmupisteeseen liittyviä särmiä ei ole mielekästä vertailla. Tämän vuoksi tarvitaan ehto 3, joka varmistaa, että uuden elementin solmupisteistä lähtevien särmien toinen päätepiste ei sijaitse elementin sisällä. Tämäkin ehto on tulkittava kolmiulotteisessa mielessä, eli solmun projektio muodostettavan elementin määräämälle tasolle tulee olla elementin ulkopuolella tai solmun etäisyyden tästä tasosta tulee olla riittävän suuri.

Myös tähän liittyy vapaita parametreja: kuinka paljon elementin ulkopuolella projektion on oltava, tai kuinka kaukana tasosta solmun on sijaittava. Tilanne on ongelmallinen esimerkiksi silloin, kun pintaverkko tekee suoran tai jyrkemmän kulman, kuten kuvassa 5.4. Kuvassa solmun B projektio on solmun A avulla muodostettavan elementin alueella. Elementin luominen tulisi kuitenkin sallia.

Menetelmään liittyy siis useita vapaita parametreja, joiden optimaalisista arvoista ei tässä vaiheessa ole mahdollista esittää muita kuin arvioita. On myös varsin mahdollista, että joissakin tapauksissa esiintyy yhtäaikaa useita ongelmallisia kohtia siten, että oikeanlaisen lopputuloksen tuottavia parametreja ei edes ole mahdollista löytää.

Toinen menetelmän suurista ongelmista on se, että algoritmi on niin sanotusti ahne, eli se valitsee solmun ainoastaan kulloinkin muodostettavan elementin laatua ajatellen. Tällöin voi joissakin tapauksissa käydä niin, että mielivaltaisen lähelle elementin reunoja jää solmu, joka myöhemmin synnyttää laadultaan huonon elementin.

5.1.6 Siirtymä tahkotyyppien välillä

Kolmessa ulottuvuudessa nousee sisäverkkojen toisiinsa liittämisessä esiin erityyppisten elementtien ongelma. Elementtien tahkot ovat esimerkiksi kuusitahokasverkossa nelikulmioita ja nelitahokasverkossa kolmioita, eikä näitä voida suoraan liittää toisiinsa konformilla tavalla.

Liittäminen on kuitenkin mahdollista pyramidielementtien avulla, koska pyramidilla on nelikulmainen pohja, johon liittyy neljä kolmiosivua. Tämä tarjoaa mahdollisuuden siirtymälle neliötahkoista kolmiotahkoihin. Owen ym. [40] esittävät kolme tapaa, joilla kuusitahokkaan yksi tahko voidaan kolmioida:

- Lisätään solmu kuusitahokkaan keskelle. Tällöin elementti voidaan hajottaa kahdeksi nelitahokkaaksi ja viideksi pyramidiksi siten, että elementin tahkoista viisi säilyy nelikulmioina ja yksi jakautuu kahdeksi kolmioksi.
- Annetaan aluksi nelitahokkaiden käyttää nelikulmion yhtä lävistäjää särmänä. Lisätään tämän jälkeen myös toinen lävistäjä jakamalla kyseiset nelitahokkaat kahtia. Nyt voidaan muodostaa pyramidi käyttämällä pohjana nelikulmiota ja sivutahkoina syntyneitä kolmioita. Pyramidille saadaan tilavuus silotusoperaation avulla.
- Yhdistetään kaksi nelitahokasta pyramidiksi. Tämä ei kuitenkaan ole mahdollista, jos niillä ei ole yhteistä tahkoa. Tällöin nelikulmion lävistäjään liittyville nelitahokkaille täytyy suorittaa muunnoksia muodostamalla kahdesta nelitahokkaasta kolme siten, että lävistäjään liittyvien nelitahokkaiden määrä samalla vähenee. Tilanne näkyy kuvassa 4.2. Vasemmassa kombinaatiossa alimpaan särmään liittyy kaksi nelitahokasta, mutta oikeassa ainoastaan yksi.

Koska tässä työssä ei rajoituta pelkästään kuusitahokasverkkoihin, tehtiin siirtymä mahdollisimman yleisellä tavalla. Kaikki kuvan 2.1 elementit voidaan nimittäin pilkkoa seuraavalla tavalla: Lisätään uusi solmu elementin keskipisteeseen. Lisätään solmu myös sellaisten nelikulmiotahkojen keskipisteeseen, jotka täytyy pilkkoa kolmioiksi. Solmun avulla jaosta saadaan tasapainoisempi kuin pelkästään jakamalla nelikulmio diagonaaliltaan kahtia.

Elementin (vanhoista tai jakamalla syntyneistä) kolmiotahkoista muodostetaan nyt elementin keskipisteen avulla nelitahokkaita ja säilytettävistä neliötahkoista pyramideja. Nämä yhtyvät toisiinsa ongelmitta, sillä niiden vastakkain olevat tahkot ovat aina kolmioita.

5.1.7 Sisäverkon luominen

Sisäverkko luotiin LGPL-lisenssin alaisuudessa olevan Netgen-kirjaston avulla. Yhdistämisalgoritmin tuottaman pintaverkon elementit syötettiin ohjelmalle yksi kerrallaan, minkä jälkeen sisäverkko voitiin luoda yhdellä aliohjelmakutsulla.

Tuloksena syntyvä sisäverkko yhdistettiin lopuksi olemassa oleviin verkon osiin. Solmupisteiden kaksoiskappaleet poistettiin, ja indeksointeja muutettiin vastaavasti. Yhdistäminen onnistuu, koska Netgen säilyttää annetun pintaverkon sellaisenaan.

Kirjaston nykyisen version dokumentaatiossa [51] ohjelman käyttämiä algoritmeja ei ole kuvailtu kovin tarkasti. Saman tekijän aiemmassa työssä [50] algoritmeja kuvataan tarkemmin, mutta lähteessä puhutaan Netgenistä puhtaasti advancing front -tyyppisenä generaattorina. Uuden version käyttöohjeessa kuitenkin puhutaan erikseen ”nopeasta Delaunay-algoritmista” ja ”hitaammasta sääntöpohjaisesta algoritmista”, jota käytetään vain, jos nopeampi algoritmi ei pysty verkottamaan koko aluetta. Ohjelman tarkasta toimintaperiaatteesta ei näin ollen saatu täysin selvää kuvaa.

5.2 Muut operaatiot

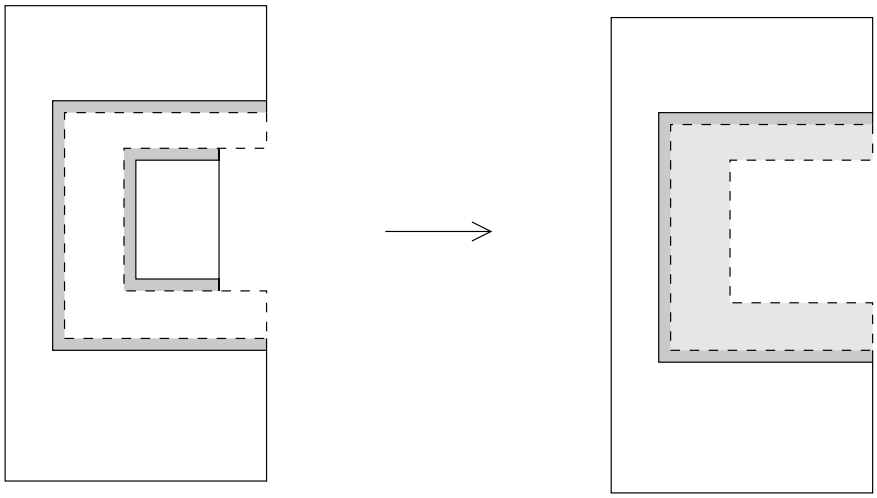
Seuraavassa hahmotellaan, kuinka edellä esitettyjen tekniikoiden avulla voitaisiin toteuttaa myös muita loogisia operaatioita. Apuna käytetään poikkileikkauskuvia, joiden merkinnät ovat samat kuin kuvan 5.2 yhteydessä. Operaatioiden lähtötilanne on edelleen sama kuin kuvassa 5.1.

5.2.1 Poisto

Eräs mahdollinen poisto-operaation toimintaperiaate on esitetty kuvassa 5.5. Verkkojen leikkauksen ulkopuolelle jäävä verkon $\mathcal{T}^2$ osa voidaan poistaa. Leikkaukseen jäävää osaa sen sijaan tarvitaan pintaverkon muodostamiseksi rakenteettomalle generaattorille.

Leikkaukseen jäävän osan pinta voidaan kolmioida, eli jakaa mahdolliset nelikulmioelementit kolmioiksi. Samoin tehdään verkon $\mathcal{T}^1$ sisäpinnalle. Leikkauskäyrän läheisyydessä pintaverkot voidaan liittää toisiinsa kuten edellä. Tämän jälkeen kolmiopintojen väliin voidaan luoda rakenteeton verkko, joka sovittaa verkon $\mathcal{T}^1$ verkon $\mathcal{T}^2$ pintaan.

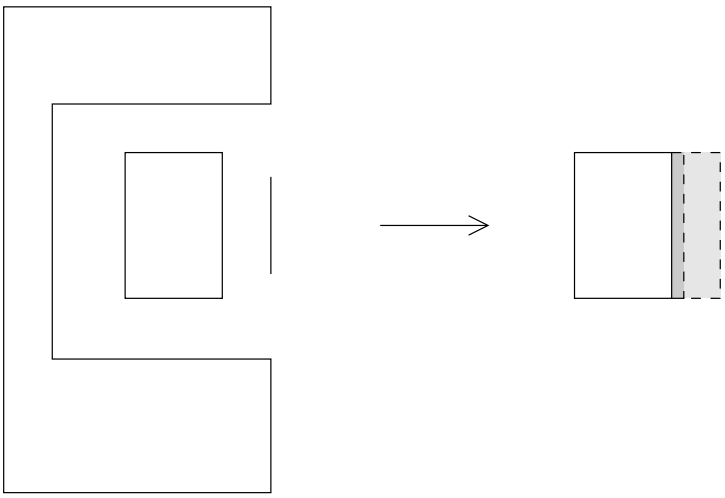
Huomattakoon kuitenkin, että mikäli verkon $\mathcal{T}^2$ pinta muodostuu nelikulmioelementeistä, joiden solmut eivät sijaitse tasossa, ei näin syntyvä pintaverkko tarkalleen noudata verkon $\mathcal{T}^2$ pintaa. Nelikulmiopinnan säilyttäminen taas vaatisi sitä, että rakenteettomassa verkossa olisi myös muita kuin nelitahokaselementtejä.



Kuva 5.5: Operaation $\mathcal{T}^1 \setminus \mathcal{T}^2$ toteutus.

5.2.2 Leikkaus

Leikkaus voitaisiin puolestaan toteuttaa esimerkiksi seuraavasti: Säilytetään verkosta $\mathcal{T}^2$ leikkaukseen jäävä osa ja verkosta $\mathcal{T}^1$ se pintaverkon pala, joka jää leikkauskäyrän rajaamalle alueelle. Kolmioidaan pintaverkon pala, ja liitetään se verkon $\mathcal{T}^2$ pintaverkon reunoihin. Näin saadaan pintaverkko rakenteettomalle generaattorille. Tilannetta havainnollistaa kuva 5.6.



Kuva 5.6: Operaation $\mathcal{T}^1 \cap \mathcal{T}^2$ toteutus.

Tämän operaation osalta huomautettakoon, että verkoilla suoritettava leikkaus ei näin toteutettuna ole vaihdannainen, eli $\mathcal{T}^1 \cap \mathcal{T}^2 \neq \mathcal{T}^2 \cap \mathcal{T}^1$. Tämä johtuu siitä, että leik-

kaus muodostetaan pääasiassa verkon $\mathcal{T}^2$ palasesta. Jos $\mathcal{T}^1$:n ja $\mathcal{T}^2$:n roolit vaihdettaisiin, myös tämä palanen otettaisiin eri verkosta ja sovitus tehtäisiin toisen verkon pintaan.

5.3 Tietorakenteita

Seuraavassa esitellään toteutetussa algoritmissa käytettyjä tietorakenteita. Merkintätapana on käytetty pseudo-fortrania. Taulukkoja merkitään syntaksilla `(ala:ylä)`, missä siis taulukon ala- ja ylärajat erotetaan kaksoispisteellä. Jos yläraja voi vaihdella, sitä ei merkitä.

5.3.1 Verkkotietorakenne

Verkot talletettiin oleellisesti seuraavan kaltaiseen rakenteeseen:

```
TYPE VERKKO
  REAL solmujen_koordinaatit(1:3,1:)
  ELEMENTTI tasot(1:3,1:)
END

TYPE ELEMENTTI
  INTEGER tyyppi
  INTEGER viittaukset(0:3,1:)
END
```

Rakenteessa verkko on jaettu tasoihin, joiden indeksit vastaavat talletettavien objektien dimensioita. Taso 1 sisältää siis verkon särmät, taso 2 tahkot ja taso 3 kolmiulotteiset elementit. Kustakin objektista talletetaan viittauksia eri tasoille. Esimerkiksi elementillä on viittaukset solmuihinsa, särmiinsä, tahkoihinsa ja naapurielementteihinsä. Tässä naapureilla tarkoitetaan objekteja, joilla on yksikin sama solmupiste. Tason 0 viittauksia ovat solmujen indeksit. Viittaukset ovat muuten kaksisuuntaisia, mutta solmut eivät kuitenkaan viittaa mihinkään.

Käytetty verkkotietorakenne sisälsi siis varsin paljon informaatiota. Rakenne oli kuitenkin erittäin hyödyllinen monissa tilanteissa. Se nimittäin soveltuu hyvin sellaisiin läpikäynteihin, joissa objektista siirrytään sen naapureihin. Tällaisia kävelyitä tarvittiin muun muassa verkkojen leikkaukseen jäävän osan poistamisessa ja pintaverkkojen leikkauskäyrien määräämisessä.

Rakenteen avulla voidaan myös nopeasti tutkia, onko särmä tai tahko verkon reunalla. Reunalla oleva tahko nimittäin kuuluu ainoastaan yhteen elementtiin. Jos taas särmä liittyy tahkoon, joka on reunalla, kulkee myös särmä kokonaisuudessaan reunalla.

5.3.2 R-puu

Esimerkiksi verkkojen leikkauspisteiden määrääminen olisi varsin raskas operaatio, jos se toteutettaisiin yksinkertaisesti vertaamalla verkon kaikkia särmiä toisen verkon kaikkiin tahkoihin. Tämän vuoksi käsiteltävien verkkojen särmät ja tahkot talletettiin Guttmanin [22] kehittämään tietorakenteeseen, jota kutsutaan R-puuksi.

Geometriset objektit talletetaan puun lehtisolmuihin. Puun sisäsolmut puolestaan tallettavat toisia solmuja. Kukin solmu voi tallettaa enintään M ja vähintään $m \leq M/2$ objektia (geometrista objektia tai solmua). Jos talletettavia objekteja on vähemmän kuin m, ne kaikki ovat juurisolmun alaisuudessa.

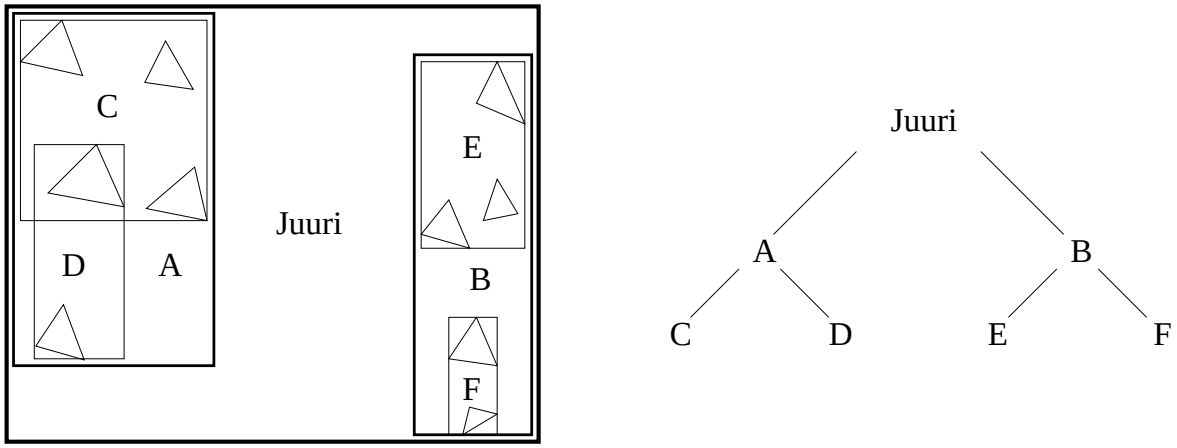
Kuhunkin geometriseen objektiin liittyy määritelmän 2.4 mukainen rajoitelaatikko. Solmujen rajoitelaatikot puolestaan määritellään siten, että ne sisältävät kaikkien niiden tallettamien objektien rajoitelaatikot. Esimerkiksi kuvassa 5.7 solmut C, D, E ja F ovat lehtisolmuja, jotka tallettavat kahdesta kolmeen kolmiota. Kolmioiden äärimmäiset koordinaatit määräävät näiden solmujen rajoitelaatikot.

Puuhun voidaan tehdä hakuja käyttämällä hakuavaimena annettua rajoitelaatikkoa. Tällöin puussa edetään juuresta lehtisolmuihin ja tutkitaan aina sellaiset solmut, joiden rajoitelaatikot leikkaavat hakuavainta. Tuloksena saadaan puun kaikki hakuavainta leikkaavat objektit. Solmujen rajoitelaatikot voivat mennä päällekkäin, joten haun aikana voidaan joutua tutkimaan useita alipuita. Tämän vuoksi hakualgoritmin tehokkuudelle ei voida taata kaikissa tilanteissa hyviä rajoja. Käytännössä puun avulla kuitenkin voidaan usein tehokkaasti rajata tutkittavien objektien joukkoa.

Kun puuhun lisätään uusi objekti, se pyritään sijoittamaan aina sellaiseen alipuuhun ja lopulta sellaiseen lehtisolmuun, missä se kasvattaa solmun rajoitelaatikkoa vähiten. Perusteena menettelylle esitetään se, että hakuja tehtäessä puussa edetään rajoitelaatikkoja vertailemalla. Näin ollen rajoitelaatikoiden pitäminen pieninä vähentää todennäköisyyttä, että alipuuta tarvitsee haun aikana tutkia. Kriteeri ei ole optimaalinen [5], mutta se on helppo toteuttaa.

Kun solmuun tulee lapsia yli sallitun määrän, se on jaettava kahtia. Jakamisen yhteydessä lapset jaetaan syntyvien solmujen kesken. Jako pyritään suorittamaan optimaalisella tavalla, mutta käytännössä optimointia ei voida tehdä tarkasti sen vaativuu-

den vuoksi. Työssä toteutettiinkin Guttmanin esittämä heuristiikka, jonka vaativuus on lineaarinen parametrin M suhteen.



Kuva 5.7: Eräs kolmioita sisältävä R-puu. Puun m=2 ja M=4.

6 Esimerkkejä

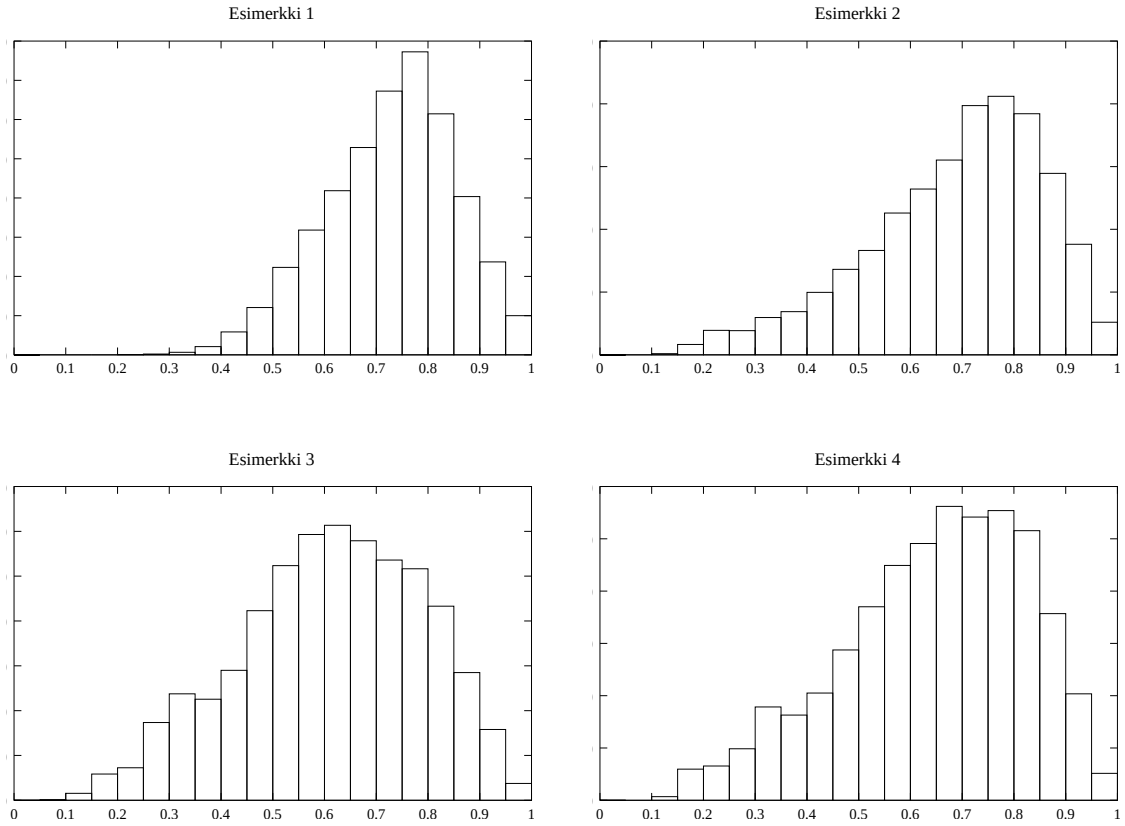
Tässä luvussa esitellään (lähinnä kuvien avulla) algoritmin toimintaa joissakin testitapauksissa. Esimerkkitapaukset esitellään tarkemmin seuraavissa alaluvuissa. Esimerkeissä tarkastellaan lähinnä syntyvää liimaverkkoa, koska sen laatu vaikuttaa suuresti lopullisen yhdisteverkon laatuun.

Testeissä käytettiin esimerkkigeometriana pääasiassa putkea, koska se on käytännön sovelluksissa varsin yleinen geometria. Toisaalta putken ja jonkin muun geometrian yhdisteenä syntyvää kappaletta ei useinkaan ole kovin yksinkertaista verkottaa esimerkiksi multiblokkimenetelmällä, koska sopivan blokkijaon löytäminen voi olla hyvin työlästä. Putki tuotettiin kaksiulotteisesta nelikulmioverkosta pursottamalla, ja sen sisäverkon rakenne onkin samanlainen kuin kuvassa 3.5.

Liimaverkon luontiin käytetty Netgen-verkkogeneraattori antaa syntyneestä rakenteettomasta verkosta myös laatutietoa. Taulukkoon 6.1 on kerätty esimerkeissä syntyneiden liimaverkkojen kolmioiden ja nelitahokkaiden pienimmät ja suurimmat kulmat. Kuvassa 6.1 on eri esimerkeissä syntyneiden elementtien jakauma tietyn, välille [0,1] skaalatun laatukriteerin mielessä. Arvoista nähdään, että syntyvä liimaverkko on laadultaan varsin hyvä. Äärimmäisen pieniä tai suuria kulmia ei esiinny, ja elementtien laatu painottuu selvästi asteikon parempaan päähän.

	Pienimmät kulmat		Suurimmat kulmat	
Esimerkki	Kolmiot	Nelitahokkaat	Kolmiot	Nelitahokkaat
1	14,5	12,1	137,5	156,2
2	7,4	6,7	159,9	166,0
3	6,9	6,2	161,4	168,5
4	9,1	6,8	161,8	170,0

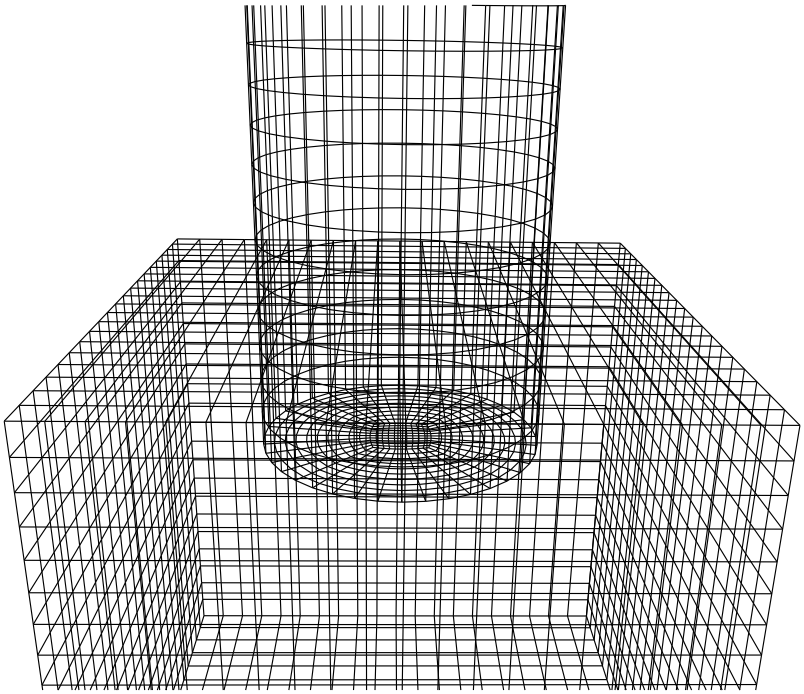
Taulukko 6.1: Kolmioiden ja nelitahokkaiden pienimmät ja suurimmat kulmat muodostuneissa liimaverkoissa.



Kuva 6.1: Elementtien jakautuminen laatuluokittain.

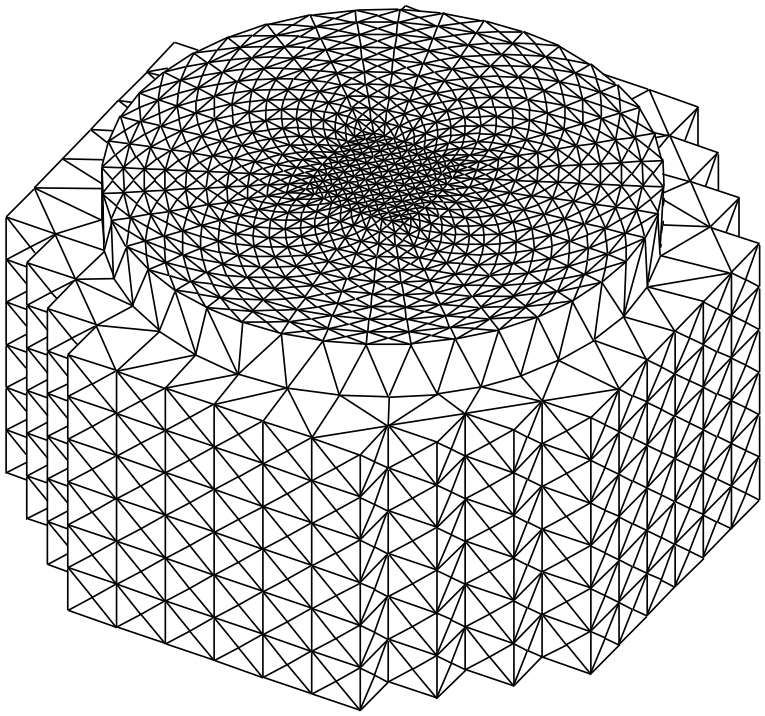
6.1 Esimerkki 1: Putki ja kuutio

Ensimmäisenä testitapauksena käytettiin putkea ja säännöllisistä kuusitahokkaista muodostuvaa kuutiota. Verkot sijaitsivat päällekkäin kuvan 6.2 osoittamalla tavalla.

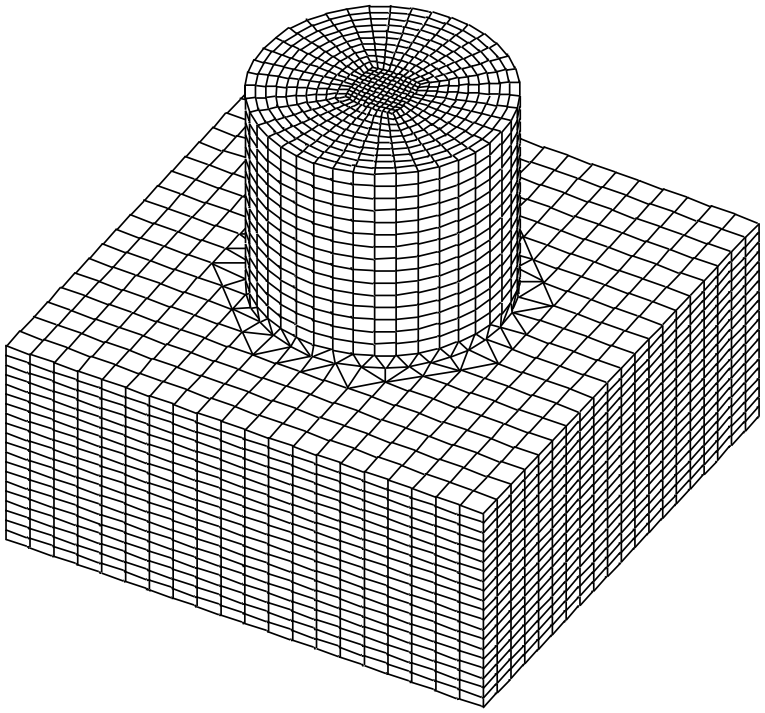


Kuva 6.2: Putki leikkaa kuutiota. Kuvassa molempien pintaverkot ennen yhdistämistä.

Kuvassa 6.3 näkyy pintaverkkojen yhdistämisen ja elementtien pilkkomisen synnyttämä pintaverkko, jota käytettiin rakenteettoman generaattorin syötteenä. Kuvasta nähdään, että pintaverkkojen yhdisteestä tuli laadultaan erittäin hyvä. Myös siirtymäkerroksiin syntyi kohtuullisia elementtejä, koska molempien verkkojen elementit olivat varsin säännöllisiä. Lopputuloksena syntyvä verkkojen yhdiste näkyy kuvassa 6.4.



Kuva 6.3: Rakenteettoman liimaverkon pinta putken ja kuution yhdisteessä.

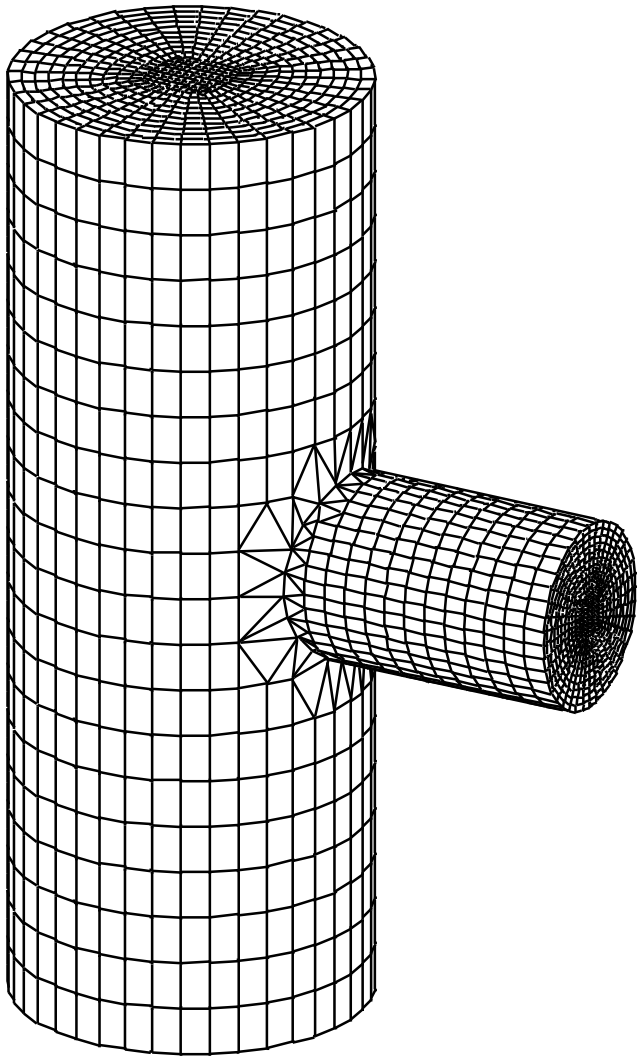


Kuva 6.4: Putken ja kuution yhdiste.

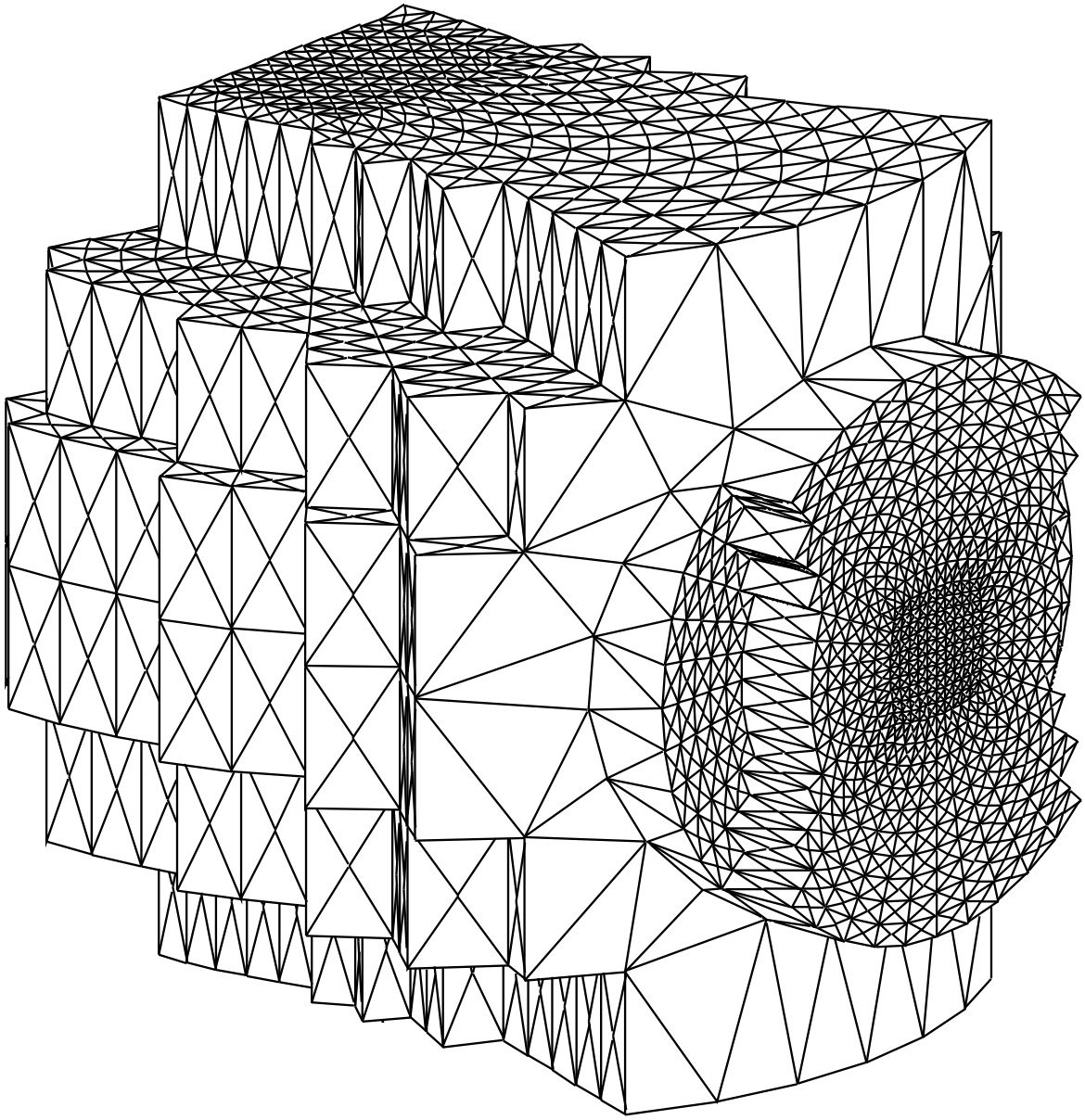
6.2 Esimerkki 2: Kaksi putkea

Toisena testitapauksena käytettiin kahta leikkaavaa putkea. Toinen putki saatiin ensimmäisestä skaalauksen ja kierron avulla. Tämän vuoksi toinen putki on halkaisijaltaan pienempi kuin toinen, mutta niissä on sama määrä elementtejä. Liitosalueella joudutaan siis siirtymään elementtikoosta toiseen.

Esimerkin lopputulos näkyy kuvassa 6.5. Rakenteettoman liimaverkon pinta on esitetty kuvassa 6.6, jossa myös liitosalue näkyy tarkemmin.



Kuva 6.5: Kahden putken yhdiste.

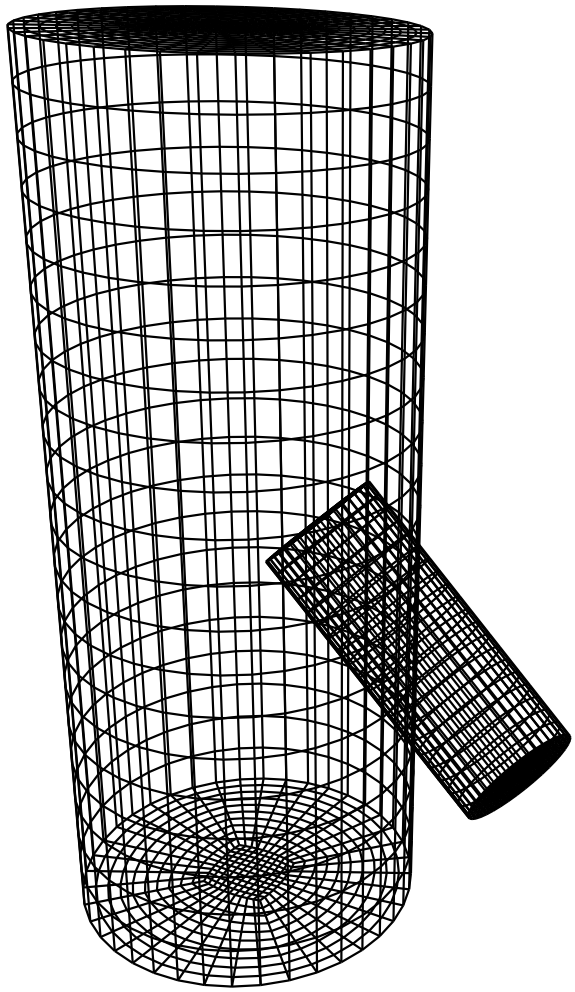


Kuva 6.6: Liimaverkon pinta putkien yhdisteessä.

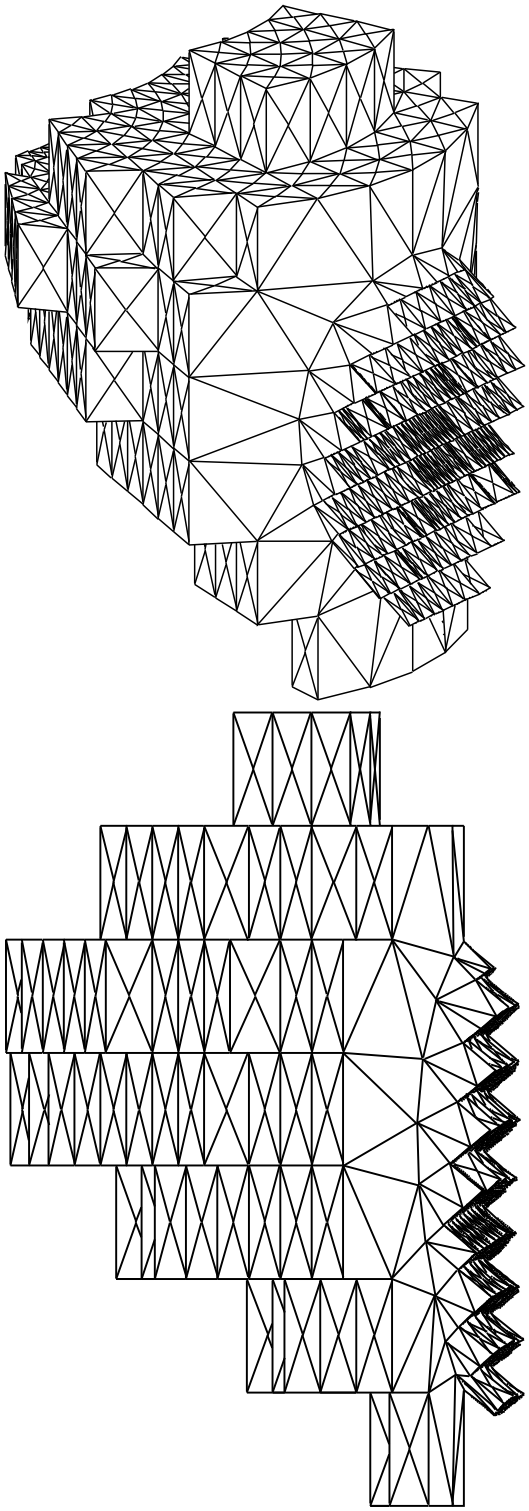
6.3 Esimerkki 3: Putkikulma

Seuraavaksi algoritmia testattiin tilanteessa, jossa putket olivat toisiinsa nähden 45 asteen kulmassa (kuva 6.7). Esimerkin vaikeutta pyrittiin lisäämään tekemällä toisesta putkesta vieläkin pienempi kuin edellisessä esimerkissä.

Syntyneen liimaverkon pinta nähdään kuvassa 6.8. Kuten odotettua, liimaverkon laatu oli hieman huonompi kuin aiemmissa esimerkeissä.



Kuva 6.7: Kaksi 45 asteen kulmassa leikkaavaa putkea.

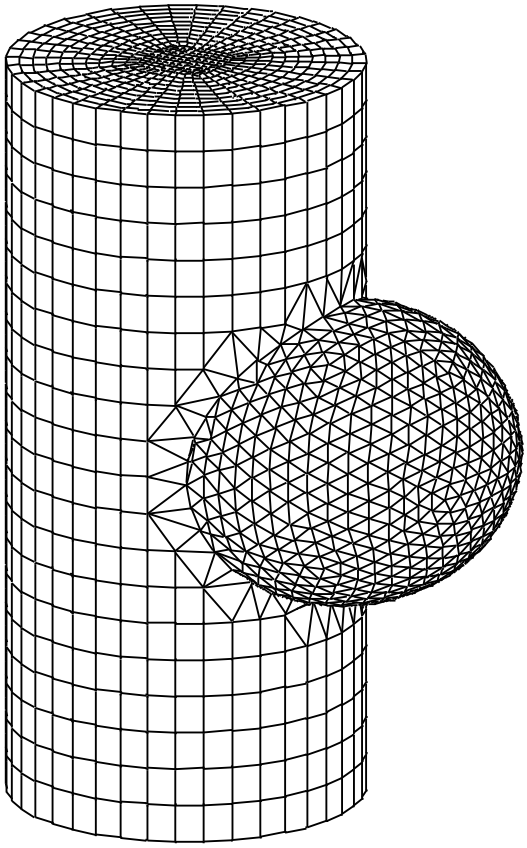


Kuva 6.8: Liimaverkon pinta 45 asteen kulmassa olevien putkien yhdisteessä.

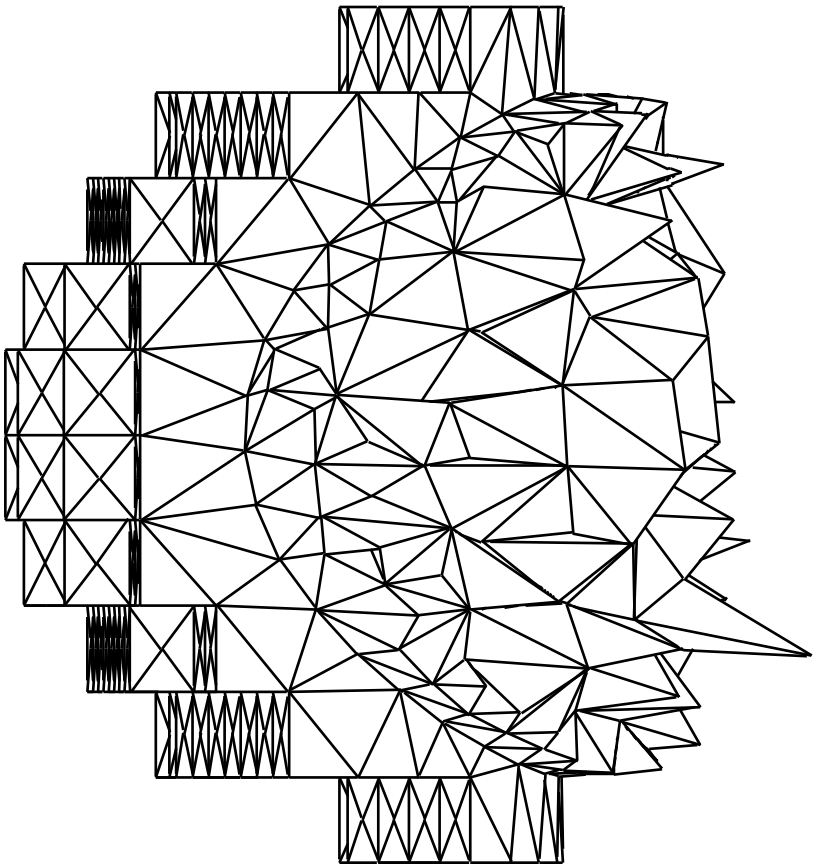
6.4 Esimerkki 4: Putki ja pallo

Viimeisenä esimerkkinä kokeiltiin tilannetta, jossa toinen verkoista oli rakenteeton. Netgen-verkkogeneraattorilla tuotettiin palloa kuvaava rakenteeton nelitahokasverkko, joka liitettiin putken kylkeen. Liitos onnistui myös tässä tapauksessa. Lopputulos näkyy kuvassa 6.9.

Liimaverkon pinta on esitetty kuvassa 6.10. Pallon rakenteettomuuden vuoksi pinta on palloon liittyvältä osalta varsin rosoinen. Tällä on kuitenkin yllättävän vähän vaikutusta syntyvän verkon laatuun, kuten kuvan 6.1 diagrammista nähdään. Pallon elementtejä ei myöskään tarvitse pilkkoa, koska niiden pinnat muodostuvat jo valmiiksi kolmioista.



Kuva 6.9: Putken ja pallon yhdiste.



Kuva 6.10: Liimaverkon pinta putken ja pallon yhdisteessä.

7 Yhteenveto

Tässä tutkielmassa suoritettiin katsaus yleisimpiin laskentaverkkojen generointimenetelmiin. Useat menetelmät ovat perusperiaatteiltaan samanlaisia kaksi- ja kolmiulotteisessa tapauksessa, joten esimerkkinä käytettiin usein kaksiulotteista tilannetta. Kolmessa ulottuvuudessa esiintyy kuitenkin enemmän erityyppisiä ongelmia, joita pyrittiin tuomaan esille kunkin menetelmätyypin yhteydessä.

Työn varsinainen tavoite oli tutkia, kuinka verkoilla voitaisiin suorittaa loogisia operaatioita eli yhdisteitä, leikkauksia ja poistoja. Työssä toteutettiin algoritmi, jonka avulla kolmiulotteisia verkkoja on mahdollista liittää toisiinsa siten, että verkkoihin tehtävät muutokset ovat kohtalaisen paikallisia. Muiden loogisten operaatioiden mahdollisia toteutustapoja hahmoteltiin lyhyesti.

Toteutetun yhdistämisalgoritmin periaate on seuraava: Ensin määrätään verkkojen pintojen leikkaus eli niin sanottu leikkauskäyrä. Tämän jälkeen verkkojen päällekkäin menevät osat ja niiden läheisyydessä sijaitsevat elementit poistetaan. Pinnat voidaan tämän jälkeen yhdistää leikkauskäyrän avulla niin, että yhdiste noudattaa varsin hyvin alkuperäisten pintojen muotoa. Jos syntyneen tyhjän alueen reunoilla sijaitsee nelikulmiotahkoja, ne kolmioidaan pilkkomalla niihin liityvät elementit sopivasti. Tyhjään alueeseen luodaan lopuksi rakenteeton kolmioverkko, joka yhdistää tilavuusverkot toisiinsa.

Algoritmia testattiin muutamien testitapausten avulla. Lopputulokset olivat varsin hyviä: verkot saatiin liitettyä toisiinsa, ja syntyvän liitosverkon laatu oli kaikissa tapauksissa vähintäänkin kohtuullinen.

Algoritmissa on kuitenkin tiettyjä puutteita. Se esimerkiksi olettaa, että pintaverkkojen leikkaus on yksi jatkuva käyrä. Algoritmia ei siis voida käyttää esimerkiksi tilanteessa, jossa toinen verkko kokonaan lävistää toisen verkon. Päällekkäin menevien osien poistaminen molemmista verkoista ei myöskään ole optimaalista. Periaatteessa riittäisi purkaa päällekkäin menevä osa vain toisesta verkosta. Toiseen verkkoon tarvitsisi tehdä tilaa ainoastaan sen verran, että pintojen yhdistäminen voidaan tehdä ja syntyvän tyhjän alueen muoto ei ole liian hankala rakenteettoman verkon generointia ajatellen.

Algoritmin osana toteutettiin pintaverkot toisiinsa liittävä advancing front -tyyppi-

nen verkon generointimenetelmä. Menetelmä operoi suoraan kolmessa ulottuvuudessa, vaikka pintaverkon generoinnissa on kyse loogisesti kaksiulotteisen verkon luonnista. Tämän vuoksi algoritmissa joudutaan tekemään useita erityyppisiä tarkasteluja, joihin liittyy suuri määrä numeerisia vakioita. Vakioille ei ole olemassa yleispäteviä optimiarvoja, minkä vuoksi algoritmi ei ole niin yleinen kuin olisi toivottavaa. Luultavasti onkin olemassa tilanteita, joissa sopivia arvoja ei edes ole olemassa eikä liitosta siis voida suorittaa. Tällaisen tilanteen voisivat muodostaa pinnat, joissa esiintyy yhtä aikaa useita hankalia piirteitä, kuten vaihtelua elementtikoossa ja teräviä kulmia.

Eräs mahdollisuus kiertää kolmessa ulottuvuudessa tehtävät tarkastelut olisi jollakin tavalla palauttaa pintaverkon luonti aidosti kaksiulotteiseen tilanteeseen. Menettelyssä voitaisiin käyttää apuna olemassa olevia pintaverkkoja. Esimerkiksi Floater [15] esittää graafiteoriaan perustuvan mentelmän, jonka avulla pintaverkko voidaan muuntaa kaksiulotteiseksi. Hänen menetelmässään vastaava kaksiulotteinen alue on valittava ennalta ja sen on oltava konveksi.

Sheffer ym. [52] puolestaan esittävät optimointiin perustuvan menetelmän. Siinä määrätään kaksiulotteisen verkon elementtien optimaaliset kulmat alkuperäisen pintaverkon avulla. Kaksiulotteinen verkko muodostetaan minimoimalla elementtien kulmien eroavuutta optimaalisista kulmista. Optimointitehtävän rajoitteet asetetaan siten, että muodostuva verkko on sallittu. Menetelmä ei aseta rajoituksia parametrisoitavan pintaverkon muodolle, eikä kaksiulotteisen verkon muotoa tarvitse kiinnittää etukäteen.

8 Viitteet

[1] Alejandro Allievi ja Sander M. Calisal. Application of Bubnov-Galerkin formulation to orthogonal grid generation. *Journal of Computational Physics*, 98:163–173, 1992.

[2] Franz Aurenhammer. Voronoi diagrams. Ks. Sack ja Urrutia [48], luku 5.

[3] I. Babuška ja A. K. Aziz. On the angle condition in the finite element method. *SIAM Journal on Numerical Analysis*, 13(2):214 – 226, 1990.

[4] Randolph E. Bank ja R. Kent Smith. Mesh smoothing using a posteriori error estimates. *SIAM Journal on Numerical Analysis*, 34:979–997, 1997.

[5] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider ja Bernhard Seeger. The R*-tree: an efficient and robust access method for points and rectangles. Kirjassa *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, ss. 322 – 331, 1990.

[6] Steven E. Benzley, Ernest Perry, Karl Merkley, Brett Clark ja Greg Sjaardema. A comparison of all hexahedral and all tetrahedral finite element meshes for elastic and elasto-plastic analysis. Kirjassa *Proceedings, 4th International Meshing Roundtable*, ss. 179–191. Sandia National Laboratories, 1995.

[7] Marshall Bern ja Paul Plassmann. Mesh generation. Ks. Sack ja Urrutia [48], luku 6.

[8] Scott A. Canann, Joseph R. Tristano ja Matthew L. Staten. An approach to combined Laplacian and optimization-based smoothing for triangular, quadrilateral, and quad-dominant meshes. Kirjassa *Proceedings, 7th International Meshing Roundtable*, ss. 479–494. Sandia National Laboratories, 1998.

[9] Juan R. Cebral, Rainald Löhner, Peter L. Choyke ja Peter J. Yim. Merging of intersecting triangulations for finite element modeling. *Journal of Biomechanics*, 34:815–819, 2001.

[10] William M. Chan. Hyperbolic methods for surface and field grid generation. Ks. Thompson ym. [63], luku 5.

[11] Luiz Cristovão G. Coelho, Marcelo Gattass ja Luiz Henrique de Figueiredo. Intersecting and trimming parametric meshes on finite element shells. *International Journal for Numerical Methods in Engineering*, 27:777 – 800, 2000.

[12] David R. Ferguson. Spline geometry: A numerical analysis view. Ks. Thompson ym. [63], luku 27.

[13] David A. Field. Laplacian smoothing and Delaunay triangulations. *Communications in Applied Numerical Methods*, 4:709–712, 1998.

[14] Peter Fleischmann ja Siegfried Selberherr. Three-dimensional Delaunay mesh generation using a modified advancing front approach. Kirjassa *Proceedings, 6th International Meshing Roundtable*, ss. 267–278. Sandia National Laboratories, 1997.

[15] Michael S. Floater. Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14:231 – 250, 1997.

[16] Lori A. Freitag. On combining Laplacian and optimization-based mesh smoothing techniques. Kirjassa *AMD-Vol. 220 Trends in Unstructured Mesh Generation*, ss. 37–43. The American Society of Mechanical Engineers, 1997.

[17] Lori A. Freitag ja Carl Ollivier-Gooch. A comparison of tetrahedral mesh improvement techniques. Kirjassa *Proceedings, 5th International Meshing Roundtable*, ss. 87–106. Sandia National Laboratories, 1996.

[18] P[aul] L[ouis] George. Improvements on Delaunay-based three-dimensional automatic mesh generator. *Finite Elements in Analysis and Design*, 25:297–317, 1997.

[19] P[aul]-L[ouis] George ja Houman Borouchaki. *Delaunay Triangulation and Meshing: Application to Finite Elements*. Editions Hermes, 1998.

[20] P.L. George. *Automatic Mesh Generation: Application to Finite Element Methods*. John Wiley & Sons, 1991.

[21] P.L. George. *Automatic Mesh Generation and Finite Element Computation*, sarjan *Handbook of Numerical Analysis* osa IV, ss. 69 – 190. Elsevier Science, 1996.

[22] Antonin Guttman. R-trees: a dynamic index structure for spatial searching. Kirjassa *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, ss. 47 – 57, 1984.

[23] Barry Joe. Construction of three-dimensional improved-quality triangulations using local transformations. *SIAM Journal on Scientific Computing*, 16:1292–1307, 1995.

[24] Mark T. Jones ja Paul E. Plassmann. Adaptive refinement of unstructured finite-element meshes. *Finite Elements in Analysis and Design*, 25:41–60, 1997.

[25] Esko Järvinen. Virtausdynamiikan teknologiaohjelma 1995–1999. Loppuraportti, Teknologian kehittämiskeskus, 2000.

[26] Yannis Kallinderis. Hybrid grids and their applications. Ks. Thompson ym. [63], luku 25.

[27] S. R. Kennon ja G. S. Dulikravich. A comparison of grid generation techniques. Kirjassa *Proceedings of 1986 Fall Joint Computer Conference*, ss. 568 – 575. IEEE Press, 1986.

[28] P. M. Knupp. Intrinsic algebraic grid generation. Kirjassa José E. Castillo, toim., *Mathematical Aspects of Numerical Grid Generation*, sarjan *Frontiers in Applied Mathematics* osa 8, luku 6. Society for Industrial and Applied Mathematics, 1991.

[29] Patrick M. Knupp. Next-generation sweep tool: A method for generating all-hex meshes on two-and-one-half dimensional geometries. Kirjassa *Proceedings, 7th International Meshing Roundtable*, ss. 505–513. Sandia National Laboratories, 1998.

[30] Patrick M. Knupp. Winslow smoothing on two-dimensional unstructured meshes. Kirjassa *Proceedings, 7th International Meshing Roundtable*, ss. 449–457. Sandia National Laboratories, 1998.

[31] Patrick M. Knupp. Applications of mesh smoothing: copy, morph, and sweep on unstructured quadrilateral meshes. *International Journal for Numerical Methods in Engineering*, 45:37–45, 1999.

[32] Xiang-Yang Li ja Shang-Hua Teng. Generating well-shaped Delaunay meshes in 3D. Kirjassa *Proceedings of the Twelfth Annual ACM-SIAM Symposium on*

Discrete Algorithms, ss. 28 – 37. Society for Industrial and Applied Mathematics, 2001.

[33] William M. Lira, Luiz C. G. Coelho ja Luiz F. Martha. Multiple intersections of finite-element surface meshes. Kirjassa *Proceedings, 11th International Meshing Roundtable*, ss. 355 – 366. Sandia National Laboratories, 2002.

[34] S. H. Lo. Automatic mesh generation over intersecting surfaces. *International Journal for Numerical Methods in Engineering*, 38:943–954, 1995.

[35] Randy R. Lober, Timothy J. Tautges ja Rich A. Cairncross. The parallelization of an advancing-front, all-quadrilateral meshing algorithm for adaptive analysis. Kirjassa *Proceedings, 4th International Meshing Roundtable*, ss. 59–70. Sandia National Laboratories, 1995.

[36] Lai Mingwu, Steven E. Benzley, Greg Sjaardema ja Tim Tautges. A multiple source and target sweeping method for generating all-hexahedral finite element meshes. Kirjassa *Proceedings, 5th International Meshing Roundtable*, ss. 217–228. Sandia National Laboratories, 1996.

[37] Scott A. Mitchell ja Stephen A. Vavasis. Quality mesh generation in three dimensions. Kirjassa *Proceedings of the Eighth Annual Symposium on Computational Geometry*, ss. 212 – 221. ACM Press, 1992.

[38] William F. Mitchell. A comparison of adaptive refinement techniques for elliptic problems. *ACM Transactions on Mathematical Software*, 15:326 – 347, 1989.

[39] Steven J. Owen. A survey of unstructured mesh generation technology. Kirjassa *Proceedings, 7th International Meshing Roundtable*, ss. 239 – 267. Sandia National Lab, 1998.

[40] Steven J. Owen, Scott A. Canann ja Sunil Saigal. Pyramid elements for maintaining tetrahedra to hexahedra conformability. Kirjassa *AMD-Vol. 220 Trends in Unstructured Mesh Generation*, ss. 123–129. The American Society of Mechanical Engineers, 1997.

[41] Sangkun Park ja Kunwoo Lee. A new approach to automated multiblock decomposition for grid generation: A Hypercube++ approach. Ks. Thompson ym. [63], luku 10.

[42] J. Peraire, J. Peiró ja K. Morgan. Advancing front grid generation. Ks. Thompson ym. [63], luku 17.

[43] Matti Pulakka. Adaptiivinen elementtimenetelmä diffuusio-konvektiotehtävälle. Pro gradu -tutkielma, Jyväskylän yliopisto, 2000.

[44] V. T. Rajan. Optimality of the Delaunay triangulation in $\mathbb{R}^d$. Kirjassa *Proceedings of the Seventh Annual Symposium on Computational Geometry*, ss. 357 – 363. ACM Press, 1991.

[45] Matthew Rees. Combining quadrilateral and triangular meshing using the advancing front approach. Kirjassa *Proceedings, 6th International Meshing Roundtable*, ss. 337–348. Sandia National Laboratories, 1997.

[46] Jim Ruppert ja Raimund Seidel. On the difficulty of triangulating three-dimensional nonconvex polyhedra. *Discrete & Computational Geometry*, 7:227 – 253, 1992.

[47] Chaman L. Sabharwal. A fast implementation of surface/surface intersection algorithm. Kirjassa *Proceedings of the 1994 ACM Symposium on Applied Computing*, ss. 333 – 337. ACM Press, 1994.

[48] J.-R. Sack ja J. Urrutia, toim. *Handbook of Computational Geometry*. Elsevier Science, 2000.

[49] Robert Schneiders. State of the art in the generation of hexahedral element meshes. Kirjassa Joachim Wendt ja Anu Piira, toim., *Automatic finite element mesh generation for industrial use*, sarjan *VTT tiedotteita* osa 1707, ss. 38 – 44. Valtion teknillinen tutkimuskeskus, 1995.

[50] Joachim Schöberl. NETGEN - an advancing front 2D/3D-mesh generator based on abstract rules. Saatavilla PS-muodossa <http://www.sfb013.uni-linz.ac.at/~joachim/papers/netgen.ps.gz>, 1997.

[51] Joachim Schöberl. *NETGEN - 4.3*, Tammikuu 2004.

[52] A Sheffer ja E. de Sturler. Parametrization of faceted surfaces for meshing using angle based flattening. *Engineering with Computers*, 17:326 –337, 2001.

[53] Mark S. Shephard ym. Automatic grid generation using spatially based trees. Ks. Thompson ym. [63], luku 15.

[54] Jonathan Richard Shewchuk. *Lecture Notes on Delaunay Mesh Generation*. Department of Electrical Engineering and Computer Science, University of California at Berkeley, Berkeley, CA 94720, 1999.

[55] Jonathan Richard Shewchuk. What is a good linear element? Interpolation, conditioning, and quality measures. Kirjassa *Proceedings, 11th International Meshing Roundtable*, ss. 115–126. Sandia National Laboratories, 2002.

[56] Alexander Shostko, Rainald Löhner ja William C. Sandberg. Surface triangulation over intersecting geometries. *International Journal for Numerical Methods in Engineering*, 44:1359 – 1376, 1999.

[57] Robert E. Smith. Transfinite interpolation (TFI) generation systems. Ks. Thompson ym. [63], luku 3.

[58] Bharat K. Soni. Grid generation: Past, present, and future. *Applied Numerical Mathematics*, 32:361–369, 2000.

[59] Stefan P. Spekreijse. Elliptic generation systems. Ks. Thompson ym. [63], luku 4.

[60] Matthew L. Staten ja Scott A. Canann. Post refinement element shape improvement for quadrilateral meshes. Kirjassa *AMD-Vol. 220 Trends in Unstructured Mesh Generation*, ss. 9–16. The American Society of Mechanical Engineers, 1997.

[61] Matthew L. Staten, Scott A. Canann ja Steve J. Owen. BMSWEEP: Locating interior nodes during sweeping. Kirjassa *Proceedings, 7th International Meshing Roundtable*, ss. 7–18. Sandia National Laboratories, 1998.

[62] David S. Thompson, Satish Chalasani ja Bharat K. Soni. Generation of volume meshes by extrusion from surface meshes of arbitrary topology. Kirjassa *Proceedings, 9th International Meshing Roundtable*, ss. 385–393. Sandia National Lab, 2000.

[63] Joe F. Thompson, Bharat K. Soni ja Nigel P. Weatherill, toim. *Handbook of Grid Generation*. CRC Press LLC, 1999.

[64] David R. White ja Paul Kinney. Redesign of the paving algorithm: Robustness enhancements through element by element meshing. Kirjassa *Proceedings, 6th International Meshing Roundtable*, ss. 323 – 335. Sandia National Lab, 1997.

[65] Alan M. Winslow. Numerical solution of the quasilinear poisson equation in a nonuniform triangle mesh. *Journal of Computational Physics*, 1:149–172, 1967.

[66] Pablo D. Zavattieri, Enzo A. Dari ja Gustavo C. Buscaglia. Optimization strategies in unstructured mesh generation. *International Journal for Numerical Methods in Engineering*, 39:2055 – 2071, 1996.

[67] J. Z. Zhu ja O. C. Zienkiewicz. A posteriori error estimation and three-dimensional automatic mesh generation. *Finite Elements in Analysis and Design*, 25:167–184, 1997.

[68] O. C. Zienkiewicz ja R. L. Taylor. *The Finite Element Method*, osa 1: The Basis. Butterworth-Heinemann, viides laitos, 2000.

[69] O. C. Zienkiewicz ja J. Z. Zhu. Superconvergence and the superconvergent patch recovery. *Finite Elements in Analysis and Design*, 19:11–23, 1995.