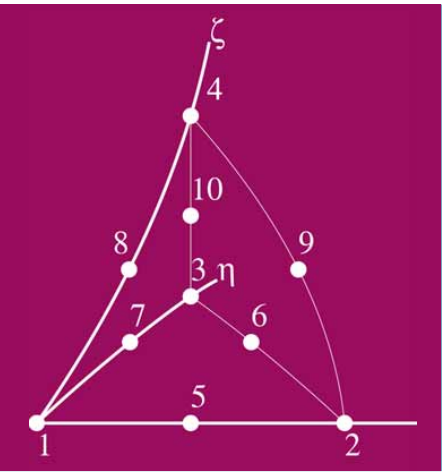
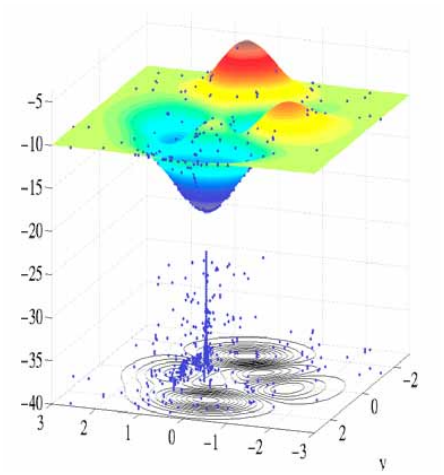
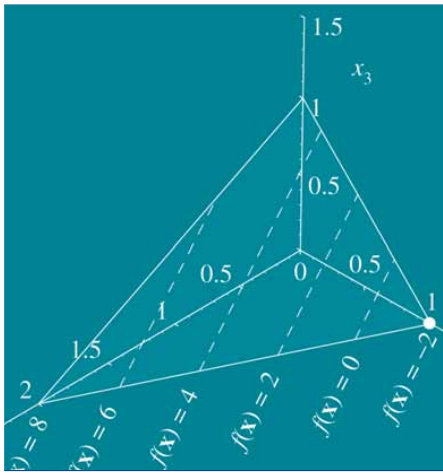
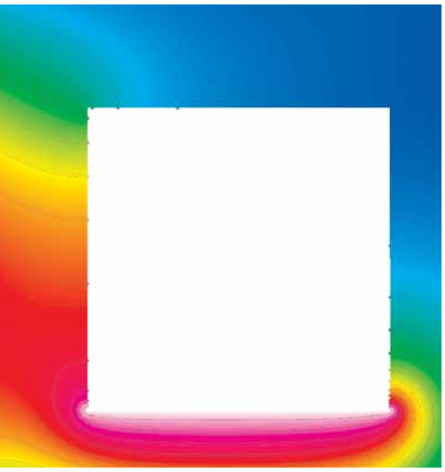
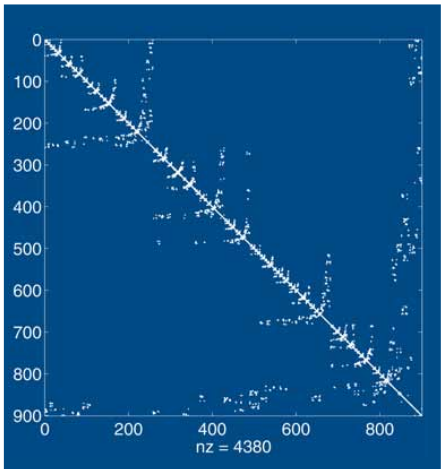
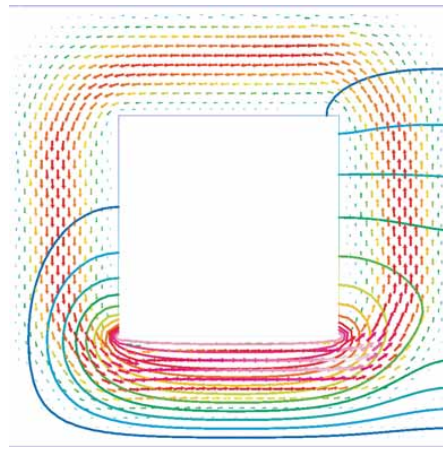












NUMEERISET MENETELMÄT
KÄYTÄNNÖSSÄ

JUHA HAATAJA, JUSSI HEIKONEN, YRJÖ LEINO,
JUSSI RAHOLA, JUHA RUOKOLAINEN JA VILLE SAVOLAINEN
CSC

Numeeriset menetelmät käytännössä

Juha Haataja

Jussi Heikonen

Yrjö Leino

Jussi Rahola

Juha Ruokolainen

Ville Savolainen

Tieteen tietotekniikan keskus CSC

ISBN 952-9821-81-6

http://www.csc.fi/oppaat/num.kayt/

Picaset Oy
Helsinki 2002

Esipuhe

Tieteessä ja tekniikassa käytetään matemaattisia malleja, kun halutaan tietää, miten luonto tai jokin laite toimii. Mallit joudutaan yleensä ratkaisemaan numeerisesti tietokoneella, koska analyyttisten ratkaisujen löytäminen on käytännön tehtävissä usein mahdotonta. Tämä teos on tarkoitettu matemaattisten mallien numeerisesta ratkaisemisesta kiinnostuneille tutkijoille. Mukana on runsaasti esimerkkejä matemaattisten tehtävien käsittelystä tietokoneessa, joten teosta voi käyttää hakuteoksena ja käsikirjana.

Pyrimme antamaan tiiviin katsauksen numeerisiin menetelmiin. Korostamme luotettavien ja tehokkaiden menetelmien merkitystä: toivomme lukijoiden käyttävän hyvin testattuja ohjelmistoja ja aliohjelmakirjastoja sen sijaan, että pääasiallisena tavoitteena olisi itse toteuttaa numeerisia menetelmiä ohjelmakoodina. Teos edellyttää lukijalta perustiedot matemaattisesta analyysistä, differentiaaliyhtälöistä ja lineaarialgebrasta.

Olemme valinneet teoksen näkökulman tieteen tietotekniikan keskus CSC:n asiakasprojekteista kertyneiden kokemusten perusteella. Keskeistä on ollut yhteistyö yliopistojen tutkijoiden kanssa sekä teollisuuden mallinnusprojekteista tulleet kokemukset. Virtauslaskentaohjelmisto ELMERin kehitystyö on tuonut käytännön kokemusta mm. elementtimenetelmästä ja yhtälöryhmien ratkaisumenetelmistä. Oma vaikutuksensa teokseen on ollut tuottamistamme kursseista ja aiemmista julkaisuista saadulla palautteella.

Olemme pyrkineet käyttämään vakiintunutta terminologiaa ja merkintätapoja. Noudatamme Kielitoimiston suosituksia ja siksi kirjoitamme esimerkiksi ”Rungen ja Kuttan menetelmät” yleisemmin käytetyn ”Runge-Kutta-menetelmät” sijaan.

Monet työtoverimme ovat oikolukeneet ja kommentoineet teosta. Kiitämme CSC:n asiantuntijoista erityisesti Juhani Käpyahoa, Juha Fagerholmia, Olli Serimaata ja Peter Råbackia. Olemme saaneet myös runsaasti palautetta yliopistomaailman asiantuntijoilta. Erityiset kiitoksemme ansaitsevat Kaisa Miettinen ja Raino A. E. Mäkinen Jyväskylän yliopistosta, Timo Eirola ja Marko Huhtanen TKK:sta sekä Robert Piché TTKK:sta.

Osa teoksen materiaalista on peräisin aiemmin ilmestyneestä kirjasta *Numeeriset menetelmät* [HKR93].

Teoksen toinen painos sisältää jonkin verran uutta materiaalia. Merkittäviä lisäyksiä on tullut approksimointia ja interpolointia käsittelevään lukuun. Lisäksi aiempien lukujen sisältöä on päivitetty ja tarkistettu.

Julkaisemme teoksen toisen painoksen kokonaisuudessaan verkkoversiona painetun kirjan lisäksi. Toivomme, että tämä antaa laajemmalle lukijakunnalle mahdollisuuden tutustua numeerisen laskennan maailmaan. Uskomme silti, että painettu kirja pitää yhä pintansa.

Teoksen toista painosta oikolukivat Janne Ignatius, Leila Puska ja Kari Vasko. Heille suuret kiitokset.

Toivomme saavamme lukijoilta palautetta. Otamme kommentteja vastaan sähköpostiosoitteessa `Juha.Haataja@csc.fi` ja lähetämme parhaille korjausehdotusten tekijöille teoksen seuraavan version.

Espoossa 2.9.2002

Tekijät

Sisältö

Esipuhe 5

Symboliluettelo 11

1 Uppoamisia ja räjähdyksiä 13

1.1 Maailman mallintaminen tietokoneessa . . . 13
1.2 Teoksen sisältö . . . 16
1.3 Lisätietoja . . . 17

2 Matemaattiset mallit ja niiden numeerinen käsittely 18

2.1 Miksi tarvitaan matemaattisia malleja . . . 18
2.2 Esimerkkejä matemaattisista malleista: populaatiomallit . . . 19
2.3 Mallien käsittely numeerisilla menetelmillä . . . 25
2.4 Algoritmit ja niiden skaalautuvuus . . . 29
2.5 Virhelähteitä tieteellisessä laskennassa . . . 31
2.6 Matemaattiset ohjelmistot . . . 33
2.7 Lisätietoja . . . 35

3 Lineaariset yhtälöryhmät 37

3.1 Mistä yhtälöryhmät ovat peräisin . . . 37
3.2 Gaussin eliminointi . . . 38
3.3 Yhtälöryhmän ratkaisemisen teoriaa . . . 42
3.4 Häiriöalttius . . . 43
3.5 Muita hajotelmia . . . 44
3.6 Ohjelmistoja . . . 46
3.7 Harvat matriisit . . . 47
3.8 Esimerkkiprobleema . . . 48
3.9 Suorat menetelmät harvoille matriiseille . . . 49
3.10 Iteratiiviset menetelmät . . . 55
3.11 Poissonin yhtälön nopea ratkaisija . . . 66
3.12 Monihilamenetelmät . . . 69
3.13 Yhteenveto . . . 71
3.14 Lisätietoja . . . 72

4 Approksimointi ja interpolointi 73

4.1 Esimerkkejä . . . 73
4.2 Approksimointi . . . 74
4.3 Teoreettisia peruskäsitteitä . . . 75
4.4 Pienimmän neliön approksimointi . . . 80
4.5 Tasainen approksimointi . . . 91
4.6 Rationaalinen approksimointi . . . 97
4.7 Interpolointi . . . 99
4.8 Lagrangen interpolointi . . . 100
4.9 Hermiten interpolointi . . . 103
4.10 Trigonometrinen approksimointi ja interpolointi . . . 106
4.11 Paloittainen interpolointi . . . 110
4.12 Aallokkeet . . . 123
4.13 Moniulotteinen interpolointi ja approksimointi . . . 136
4.14 Ohjelmistot ja aliohjelmakirjastot . . . 147
4.15 Lisätietoja . . . 148

5 Numeerinen integrointi 149

5.1 Esimerkkejä . . . 149
5.2 Perusasioita numeerisesta integroinnista . . . 151
5.3 Gaussin kvadratuurit . . . 156
5.4 Clenshaw'n ja Curtisin kaavat . . . 160
5.5 Newtonin ja Cotesin kaavat . . . 161
5.6 Tšebyševin integrointi . . . 164
5.7 Erikoistapauksia . . . 164
5.8 Ekstrapolointi ja adaptiivinen integrointi . . . 171
5.9 Moniulotteiset integraalit . . . 174
5.10 Ohjelmistokatsaus . . . 185
5.11 Lisätietoja . . . 185

6 Epälineaariset yhtälöt 187

6.1 Epälineaariset yhtälöt ja yhtälöryhmät . . . 187
6.2 Mahdollisia ongelmatilanteita nollakohdan löytämisessä . . . 189
6.3 Yhden yhtälön ratkaiseminen . . . 191
6.4 Epälineaarinen yhtälöryhmä . . . 199
6.5 Epälineaaristen yhtälöiden yhteys optimointiin . . . 205
6.6 Yhteenveto . . . 206
6.7 Ohjelmistoja epälineaaristen yhtälöiden ratkaisemiseen . . . 206
6.8 Lisätietoja . . . 207

7 Tavalliset differentiaaliyhtälöt 208

7.1 Tehtävätyyppejä ja käsitteitä . . . 209
7.2 Numeerisen ratkaisemisen teoriaa . . . 213
7.3 Yksiaskelmenetelmät alkuarvotehtäville . . . 225
7.4 Lineaariset moniaskelmenetelmät alkuarvotehtävälle . . . 229

7.5 Reuna-arvotehtävät 234
7.6 Menetelmäsuosituksia 240
7.7 Ohjelmistot 240
7.8 Lisätietoja 241

8 Elementtimenetelmä 242

8.1 Taustaa 242
8.2 Elementtimenetelmän peruskäsitteitä 246
8.3 Yhtälöiden diskretointi tietokoneratkaisua varten 248
8.4 Reunaehdot 252
8.5 Yhtälön linearisointi 253
8.6 Aikaintegrointi 255
8.7 Elementtityypit, kantafunktiot ja isoparametrinen kuvaus . 258
8.8 Lokaalin ja globaalin yhtälöryhmän kokoaminen 270
8.9 Virhearviot ja suppenemiskriteerit 272
8.10 Adaptiivinen verkon tihentäminen 275
8.11 Soveltava esimerkki 277
8.12 Esimerkkitehtävän ratkaisu 278
8.13 ODY-ryhmä: yhtälöiden kytkentä 291
8.14 Stabiloitu elementtimenetelmä 294
8.15 Yhteenveto 301
8.16 FEM-pohjaisia ohjelmistoja 302
8.17 Lisätietoja 303

9 Ominaisarvotehtävät 304

9.1 Esimerkki 304
9.2 Määritelmiä ja käsitteitä 305
9.3 Ominaisarvotehtävien numeeriset menetelmät 310
9.4 Pienet ominaisarvotehtävät 311
9.5 Suuret ominaisarvotehtävät 323
9.6 Yleistetyt tehtävät 327
9.7 Yhteenveto ja suosituksia 330
9.8 Ohjelmistokatsaus 330
9.9 Kirjallisuutta ja lisätietoja 331

10 Optimointi 332

10.1 Optimointitehtävän muodostaminen ja ratkaisu 332
10.2 Optimointitehtävien tyyppejä 333
10.3 Lokaalit ja globaalit minimit 335
10.4 Globaali ja lokaali suppeneminen 336
10.5 Konveksit optimointitehtävät 336
10.6 Ehtoja minimikohdan sijainnille 337
10.7 Yksiulotteinen optimointi 339
10.8 Moniulotteinen optimointi 344
10.9 Epälineaariset pienimmän neliösumman tehtävät 353

10.10 Rajoitteita sisältävät tehtävät 356
10.11 Globaali optimointi 366
10.12 Yhteenveto 369
10.13 Ohjelmistoja 370
10.14 Lisätietoja 372

11 Yhteenveto **374**

11.1 Numeerisen menetelmän valinta 374
11.2 Yhteyksiä menetelmien välillä 375
11.3 Lisätiedon lähteitä 376

Liitteet **377**

A Matemaattisia taustatietoja **378**

A.1 Yleisiä merkintätapoja 378
A.2 Peruskäsitteitä 379
A.3 Vektorit ja matriisit 381
A.4 Matriisien hajotelmien käyttömahdollisuuksia 385
A.5 Vektoriavaruus ja normi 387
A.6 Sisätulo 390
A.7 Suppenemisnopeus 392

B Liukulukuesitys **393**

B.1 Reaaliluvut ja liukuluvut 393
B.2 Liukulukuesityksen virheet 393
B.3 IEEE-aritmetiikka 395
B.4 Liukulukuaritmetiikan ominaisuuksia 396
B.5 Käytännön ohjeita liukulukulaskentaan 397
B.6 Virheiden kasautuminen 398
B.7 Lisätietoja 399

C Lisätietoja CSC:stä **400**

Kirjallisuutta **402**

Hakemisto **407**

Symboliluettelo

Seuraava luettelo sisältää tärkeimmät teoksessa käytetyt symbolit ja merkintätavat.

Symboli	Selitys
$\mathbb{R}$	reaalilukujen joukko
$\mathbb{R}^n$	n-ulotteinen reaaliavaruus
$\mathbb{R}^n \mapsto \mathbb{R}$	kuvaus avaruudesta $\mathbb{R}^n$ reaaliluvuille
x, y	skalaarimuuttujia
$\mathbf{x}, \mathbf{y}$	vektoreita
x_i, y_j	vektorien alkioita
A, B	matriiseja
a_{ij}	matriisin A alkio (rivi i, sarake j)
$\mathbf{x}^T, A^T$	vektorin ja matriisin transpoosi
$\boldsymbol{\tau}$	tensori
τ_{ij}	tensorin alkioita
$\mathbf{0}$	nollavektori: $\mathbf{0} = (0, 0, \ldots, 0)^T$
$\mathbf{1}$	yksikkövektori: $\mathbf{1} = (1, 1, \ldots, 1)^T$
$\mathbf{e}^i$	i:s standardikannan yksikkövektori
$\mathbf{x}^l, \mathbf{x}^u$	laatikkorajoitteet: $\mathbf{x}^l \leq \mathbf{x} \leq \mathbf{x}^u$
I, L, U	identiteettimatriisi, ala- ja yläkolmiomatriisi
$\mathrm{diag}(d_1, \ldots, d_n)$	lävistäjämatriisi jonka alkiot ovat $d_1, d_2, \ldots, d_n$
$x_k, k = 1, 2, \ldots$	skalaarimuuttujan iteraatit
$\mathbf{x}^k, k = 1, 2, \ldots$	vektorin iteraatit
$A_k, k = 1, 2, \ldots$	matriisin iteraatit
$=$	yhtäsuuruus
$\approx$	likimääräinen yhtäsuuruus
$\in, \notin$	kuuluu joukkoon, ei kuulu joukkoon
$\subset$	osajoukko
$A \setminus B$	joukko A miinus joukko B
$\forall$	kaikilla
$\Longrightarrow$	jos … niin …
$\Longleftrightarrow$	jos ja vain jos
$<, >$	pienempi kuin, suurempi kuin
$\leq, \geq$	pienempi tai yhtäsuuri, suurempi tai yhtäsuuri
$\ll$	paljon pienempi kuin
$\infty, -\infty$	ääretön, miinus ääretön

Symboli	Selitys
$\mathrm{eig}(A)$	matriisin A ominaisarvot
$\ker(A)$	matriisin A ydin
$\mathrm{R}(A)$	matriisin A kuva-avaruus
$\kappa(A)$	matriisin A häiriöalttius
$\langle \mathbf{x}, \mathbf{y} \rangle$	vektorien $\mathbf{x}$ ja $\mathbf{y}$ sisätulo: $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T\mathbf{y} = \sum_{i=1}^{n} x_i y_i$
$\lvert \cdot \rvert$	itseisarvo
$\lVert \cdot \rVert$	(euklidinen) normi
$[a, b]$	reaaliakselin suljettu väli
(a, b)	reaaliakselin avoin väli
$\{ x_1, x_2, \ldots, x_n \}$	joukon alkiot lueteltuina
$\{ \mathbf{x} \in \mathbb{R}^n \mid \lVert \mathbf{x} \rVert \leq 1 \}$	korkeintaan ykkösen suuruiset vektorit
$3.14159, (3.14159)_{10}$	desimaaliluku
$(0.11)_2$	binääriluku
$\lfloor x \rfloor$	reaaliluvun x kokonaisosa: $\lfloor 1.6 \rfloor = 1$
ϵ_{kone}	konevakio
$\sum_{i=1}^{n} s_i$	termien s_i summa: $\sum_{i=1}^{n} s_i = s_1 + s_2 + \cdots + s_n$
$\prod_{i=1}^{n} s_i$	termien s_i tulo: $\prod_{i=1}^{n} s_i = s_1 \times s_2 \times \cdots \times s_n$
$E[r]$	suureen r odotusarvo
$h \to 0$	h lähestyy arvoa 0
$\lim_{h \to 0} f(h)$	raja-arvo lausekkeelle $f(h)$
$\mathcal{O}(n)$	samaa suuruusluokkaa kuin n
$f(x)$	funktio avaruudesta $\mathbb{R}$ avaruuteen $\mathbb{R}$
$f(\mathbf{x}), \mathbf{f}(\mathbf{x})$	funktio $\mathbb{R}^n \mapsto \mathbb{R}$, funktio $\mathbb{R}^n \mapsto \mathbb{R}^m$
$f'(x)$	skalaarifunktion derivaatta
∇	osittaisderivaattaoperaattori
$\nabla f(\mathbf{x})$	funktion f gradienttivektori pisteessä $\mathbf{x}$
$H(\mathbf{x})$	Hessen matriisi pisteessä $\mathbf{x}$: $H(\mathbf{x}) = \nabla\nabla^T f(\mathbf{x})$
$J(\mathbf{x})$	Jacobin matriisi pisteessä $\mathbf{x}$
$\mathcal{F}$	optimointitehtävän käypä alue
$\mathbf{x}^*, f^*$	funktion minimipiste tai nollakohta
f^*	funktion minimiarvo
$f^* = \mathrm{minimi}_{\mathbf{x}} f(\mathbf{x})$	funktion f minimointi muuttujan $\mathbf{x}$ suhteen
$f^* = \mathrm{maksimi}_{\mathbf{x}} f(\mathbf{x})$	funktion f maksimointi muuttujan $\mathbf{x}$ suhteen
$\min_i\{x_i\}, \max_i\{x_i\}$	pienin ja suurin arvo joukosta $\{x_i\}$
$\mathbf{p}^k, \mathbf{s}^k$	iteratiivisen algoritmin hakusuunta ja hakuaskel
$\mathrm{Re}(\lambda)$	kompleksiluvun λ reaaliosa
$C[a, b]$	välillä $[a, b]$ jatkuvien funktioiden joukko
C^m	m kertaa derivoituvien funktioiden joukko
$C^m[a, b]$	välillä $[a, b]$ määriteltyjen m kertaa derivoituvien funktioiden joukko
$L_2[a, b]$	välillä $[a, b]$ neliöintegroituvien funktioiden joukko $f \in L_2[a, b] \equiv \int_a^b \lvert f \rvert \, dx < \infty.$
$\mathrm{supp} f(x)$	funktion $f(x)$ kantaja $= \overline{\{x \mid f(x) \neq 0\}}$

1 Uppoamisia ja räjähdyksiä

1.1 Maailman mallintaminen tietokoneessa

Numeerisilla menetelmillä pyritään ratkaisemaan matemaattisessa muodossa esitettyjä ongelmia tehokkaasti ja luotettavasti. Menetelmän tehokkuuteen vaikuttaa mm. ratkaisulta vaadittu tarkkuus. Tehokkuuteen ja luotettavuuteen vaikuttaa olennaisella tavalla se, miten menetelmä on ohjelmoitu tietokoneelle. Usein myös taustalla oleva matemaattinen ongelma asettaa omat rajoituksensa ratkaisemisen onnistumiselle.

Numeeriset menetelmät tarjoavat sillan reaalimaailman ongelmien ratkaisemiseen tietokoneessa matemaattisen mallintamisen pohjalta. Tämä tehtävä on erittäin haastava. On varsin tavanomaista törmätä huonosti toimiviin numeerisiin menetelmiin. Voisikin sanoa, että numeeristen menetelmien tutkijoita kiinnostaa usein se, miksi jokin menetelmä *toimii* eikä niinkään se, miksi jonkin menetelmä ei toimi.

Myös taustalla olevassa reaalimaailman matemaattisessa mallissa voi olla virhe.

Esimerkki 1.1.1 Sleipner A -öljynporauslautta tuottaa öljyä ja kaasua Pohjanlahdella 82 metrin syvyisessä vedessä. Lautta on rakennettu betoniselle alustalle. Alustasta kohoaa neljä tornia, joiden varassa on lautan laitteistokansi.

Lautan alustaa testattiin painolastin avulla 23.8.1991 ennen kannen asennusta paikalleen. Testauksessa alustaan tuli vuoto ja se upposi vuonoon Stavangerin lähellä. Uppoaminen 220 metrin syvyyteen aiheutti järistyksen, jonka suuruus oli 3.0 Richterin asteikolla. Taloudelliset tappiot olivat miljardiluokkaa.

Tutkimuksissa kävi ilmi, että yhteen alustan seinämistä tuli vakava vuoto, jota pumput eivät pystyneet kompensoimaan. Syynä oli suunnittelu- ja rakennusvirhe. Alustan lujuuslaskelmat oli tehty elementtimenetelmällä käyttäen NASTRAN-ohjelmistoa. Alustan osasten liitoskohdan analyysissä oli käytetty vääränlaista elementtimallia, jolloin osaan vaikuttavia voimia aliarvioitiin lähes 50%. Tarkemmissa laskelmissa päädyttiin tulokseen, että rakenteen kestävyys pettäisi 62 metrin syvyydessä. Todellisuudessa rakenne petti 65 metrin syvyydessä.

Matemaattisen mallin siirtäminen tietokoneen ymmärtämään muotoon vaatii suurta tarkkuutta. Ei ole tavatonta, että pieni virhe jää huomaamatta ja

aiheuttaa suurta päänvaivaa numeerisen laskentaohjelman käyttäjille ja kehittäjille.

Esimerkki 1.1.2 Vuonna 1962 Nasa laukaisi Venukseen tarkoitetun Mariner I -luotaimen. Pian lähdön jälkeen raketti alkoi käyttäytyä holtittomasti, minkä takia se jouduttiin tuhoamaan. Luotain putosi Atlantin valtamereen.

Tapaturman syynä oli laitevirhe yhdistyneenä ohjelmistovirheeseen. Laitevian takia rakettia ohjattiin maasta tutkalla. Tutkan antamissa mittaustiedoissa oli virhettä, minkä takia mittausarvoista olisi pitänyt laskea juokseva keskiarvo.

Ohjauskoodin suunnitelmista kuitenkin puuttui keskiarvostusta tarkoittava yläviiva nopeusmuuttujan päältä. Siten ohjaukseen käytetiin viimeisintä tutkan antamaa arvoa, jossa oli mukana satunnaista virhettä. Ohjausjärjestelmä kuvitteli raketin heittelehtivän ja yritti kompensoida tätä komennoilla, jotka todella saivat raketin käyttäytymään holtittomasti.

Vaikka taustalla oleva matemaattinen malli olisikin muunnettu oikein tietokoneen ymmärtämään muotoon, voi ratkaisumenetelmässä olla virhe joka saa aikaan vääriä tuloksia.

Esimerkki 1.1.3 Vuonna 1963 Nasassa kehitettiin ja testattiin aiemmilla Mercury-lennoilla käytettyä rakettisimulaattoria. Testauksessa havaittiin, että tulokset olivat kohtalaisen tarkkoja, mutta eivät kuitenkaan täysin vastanneet tunnettuja tuloksia. Usean viikon testauksen jälkeen Fortran-ohjelmakoodista löytyi rivi

```
DO 10 I=1.10
```

Tässä piti olla toistorakenne, jota suoritetaan kymmenen kertaa. Pilkun vaihtuminen pisteeksi muunsi lauseen kuitenkin sijoituslauseeksi, jossa muuttujaan `DO10I` sijoitettiin arvo 1.10. Siispä kyseinen toistorakenne suoritettiin vain kerran. Saadut tulokset olivat riittävän tarkkoja aiemmilla lennoilla, jolloin rakettien tehot olivat pienempiä. Onneksi virheestä ei aiheutunut mitään todellisia vahinkoja.

Koska numeerisia menetelmiä käytetään todellisissa tietokoneissa eikä idealisoiduissa järjestelmissä, täytyy tietokoneen todelliset ominaisuudet huomioida laskennassa. Eräs merkittävä virhelähde on tietokonearitmetiikan rajallinen tarkkuus. Pahimmassa tapauksessa jo yksi virhetilanteeseen johtava aritmetiikkavirhe voi olla katastrofaalinen.

Esimerkki 1.1.4 Euroopassa kehitetyn Ariane 5 -raketin ensimmäinen laukaisu tapahtui 4.6.1996. Rakettia oli kehitetty vuosikymmenen ajan ja kehityskulut olivat luokkaa 50 miljardia markkaa. Noin 37 sekuntia laukaisun jälkeen raketti alkoi käyttäytyä holtittomasti ja lopulta räjähti. Raketin ja sen lastin arvo oli useita miljardeja markkoja.

Turman syy selvisi kahden viikon kuluessa. Ohjelmassa käytettiin Ariane 4 -raketille kehitettyä ohjauskoodia, jossa raketin vaakasuoraa nopeuttaa kuvaava 64-bittinen liukuluku muutettiin 16-bittiseksi kokonaisluvuksi. Tässä tapauksessa lukuarvo oli kuitenkin yli 32 768, joka on suurin 16-bittisessä kokonaislukuaritmetiikassa esitettävissä oleva luku. Prosessori antoi virhetilanteesta ilmoituksen ja tulosti virheraportin.

Virhetilanteen käsittelyä ei kuitenkaan määritelty Ada-koodissa, jolloin ohjausjärjestelmä yritti tulkita tuloksen raketin ohjauskomennoiksi. Seurauksena oli holtiton käyttäytyminen ja lopulta raketin itsetuhomekanismin käynnistyminen. Kyseinen osa ohjauskoodia ei ollut tarpeen Ariane 5:ssä ja oli joka tapauksessa ohjelmoitu poistumaan käytöstä 40 sekuntia laukaisun jälkeen.

Äärellisen laskentatarkkuuden tuottama epätarkkuus voi olla myös hienovaraisempaa ja ilmetä vasta pitkähköjen laskuoperaatioiden tuloksena. Tyypillinen esimerkki tästä on liukulukuaritmetiikan pyöristys- tai katkaisuvirheiden kasautuminen.

Esimerkki 1.1.5 Vuonna 1982 Vancouverin pörssi otti käyttöön uuden pörssiindeksin, jota päivitettiin jokaisen kaupan jälkeen. Indeksin alkuarvoksi asetettiin 1000.000. Indeksin arvo putosi 20 kuukauden kuluessa 520:een.

Syynä oli laskennassa käytetty katkaiseva aritmetiikka: päivitettyä indeksin arvoa ei pyöristetty lähimpään tuhannesosaan vaan arvo katkaistiin ja loput desimaalit unohdettiin. Pyöristystä käyttäen saatiin indeksin arvoksi 1099.

Laskennassa oleva virhe voi olla sellainen, että sitä ei normaalissa toiminnassa havaita. Virhe voi tulla esiin vasta laskettaessa aiempia vaativampia malleja. Virhe voi johtua myös ohjelmiston eri osasten yhteisvaikutuksesta.

Esimerkki 1.1.6 Vuoden 1991 alussa USA ja Irak kävivät sotaa Persianlahdella. Irak ampui Scud-ohjuksia amerikkalaisiin sotilaskohteisiin ja USA käytti torjuntaan Patriot-ilmatorjuntaohjuksia. Kuitenkaan 25. helmikuuta Patriot-ohjus ei osunut kohteeseensa ja Scud-ohjus tappoi 28 amerikkalaissotilasta.

Syyksi osoittautui ohjelmistovika. Patriot-ohjuksessa on kello, joka mittaa ajan kulumista kymmenesosasekunteina käyttäen kokonaislukulaskuria. Ohjusjärjestelmä oli ollut yhtäjaksoisesti toiminnassa yli 100 tuntia. Siis laskurin arvo oli suuruusluokkaa 3.6 miljoonaa.

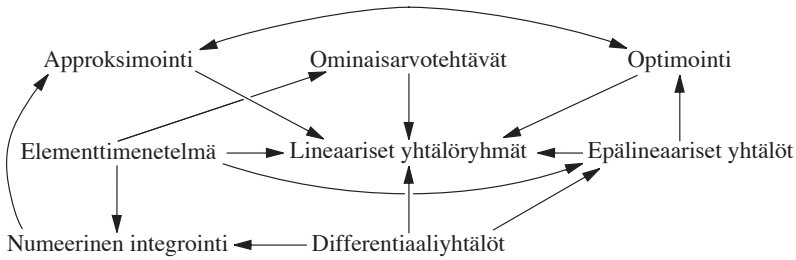
Patriot-ohjus etsii tulevaa ohjusta alueelta, jonka paikka arvioidaan edellisen mittausarvon perusteella. Kulunut aika määrätään kertomalla aikalaskurin arvo luvulla 0.1. Koska luvun 0.1 binääriesitys katkaistiin 24 bittiin, oli tämän luvun esitysmuodossa suuruusluokkaa 10^{-7} oleva virhe. Tämä luku kerrottiin luvulla $3.6 \cdot 10^6$, joten tulokseen tuli virhettä noin 0.3 sekunnin verran. Tässä ajassa Scud-ohjus ehti lentää yli 600 metriä, joten Patriot-ohjus yritti paikantaa tulevaa ohjusta väärältä suunnalta.

1.2 Teoksen sisältö

Tämä teos esittelee matemaattisten mallien numeerista ratkaisemista. Teos sisältää esimerkkejä tehtävien ratkaisemisesta yleisesti käytössä olevilla ohjelmistoilla ja välineillä. Lisäksi kuvailemme käytetyimpien ratkaisumenetelmien perusteita. Jos mahdollista, annamme yleisiä neuvoja tehtävään soveltuvan menetelmän valintaan.

Teoksen lukijalta vaaditaan perustiedot matemaattisesta analyysistä, differentiaali- ja integraalilaskennasta, differentiaaliyhtälöistä ja lineaarialgebrasta. Teos rakentuu osittain CSC:n julkaisemien kirjojen *Matemaattiset ohjelmistot* [Haa98], *Numeeriset menetelmät* [HKR93] ja *Optimointitehtävien ratkaiseminen* [Haa95] pohjalle. Kunkin luvun lopusssa luettelemme aihepiiriin liittyviä lisätietojen lähteitä.

Tämä teos ei ole oppikirja, sillä menetelmien johtamiset ja suppenemistodistukset on valtaosin sivuutettu. Kirjallisuusluettelosta kuitenkin löytyy runsaasti eri menetelmien taustaa ja numeerista problematiikkaa valaisevia teoksia. Toisaalta kirja sisältää runsaasti esimerkkejä, joista lienee hyötyä käsikirjamaisessa käytössä. Kuva 1.1 kertoo teoksen lukujen välisistä riippuvuuksista.



Kuva 1.1: *Teoksen eri osien liittyminen toisiinsa. Nuolet osoittavat aihepiirejä, joiden menetelmiä tai käsitteitä tarvitaan kyseisessä kohdassa.*

Esitämme luvussa 2 tiiviin katsauksen oppaan lukemisessa tarvittaviin käsitteisiin: mitä ovat matemaattiset mallit ja mihin tarvitaan numeerisia menetelmiä? Luku 3 käsittelee lineaaristen yhtälöryhmien ratkaisumenetelmiä. Luvussa 4 käsittelemme funktioiden approksimointia ja interpolointia ja luvussa 5 numeerista integrointia. Perustiedot epälineaaristen yhtälöiden ja yhtälöryhmien ratkaisemisesta löytyvät luvusta 6. Luku 7 käsittelee tavallisten differentiaaliyhtälöiden ratkaisua ja luku 8 osittaisdifferentiaaliyhtälöiden ratkaisua elementtimenetelmällä (FEM). Lopuksi esittelemme vielä ominaisarvotehtäviä (luku 9) ja optimointitehtävien ratkaisua (luku 10).

Symboliluettelossa (sivu 11) esittelemme käytetyn notaation. Liite A kertaa matemaattisia taustatietoja ja liite B esittelee liukulukuaritmetiikkaa. Liite C kertoo CSC:n palvelujen käytöstä.

1.3 Lisätietoja

Johdatuksia numeeriseen laskentaan ovat mm. teokset *Fundamentals of Numerical Computing* [SAP97], *Scientific Computing* [Hea02], *Numerical Mathematics and Computing* [CK94], *Numerical Analysis* [BF97] ja *Numerical Analysis: An Introduction* [EWK90]. Lineaarialgebran käsikirjaksi sopii teos *Matrix Computations* [GvL96].

Suomenkielisiä oppikirjoja ovat *Numeerinen matematiikka* [MNV82] ja *Numeeriset menetelmät* [Mäk98]. Lineaarialgebran perusteita ja matriisilaskentaa käsitellään oppikirjassa *Matriisilasku ja lineaarialgebra* [Kiv84].

Numeerisista menetelmistä on kerrottu myös CSC:n aiemmin julkaisemassa oppaassa *Numeeriset menetelmät* [HKR93], jota on käytetty apuna tämän teoksen suunnittelussa. Tämän teoksen rinnakkaismateriaalina voi käyttää CSC:n julkaisemia kirjoja *Matemaattiset ohjelmistot* [Haa98], *Datan käsittely* [Kar01], *Elementtimenetelmä virtauslaskennassa* [HJ94], *Optimointitehtävien ratkaiseminen* [Haa95], *Tieteellinen visualisointi* [RG96], *Alkuräjähdyksestä kännykkään* [Haa02] ja *Laskennallinen tuotekehitys: suunnittelun uusi ulottuvuus* [HJKR02].

2 Matemaattiset mallit ja niiden numeerinen käsittely

Matemaattisia malleja käytetään useimmilla tieteenaloilla. Perinteisiä esimerkkejä mallintamista käyttävistä tieteistä ovat fysiikka ja insinööritieteet; uudempia alueita ovat kielitiede ja biotieteet. Tässä luvussa pohdimme, mitä ovat matemaattiset mallit ja mihin tarvitaan numeerisia menetelmiä.

2.1 Miksi tarvitaan matemaattisia malleja

Tieteessä ja tekniikassa käytetään matemaattisia malleja, kun halutaan selvittää, miten luonto tai jokin laite toimii. Matemaattisessa mallissa esitetään tutkittavat ilmiöt matemaattisten lausekkeiden avulla. Mallien avulla löydämme systeemin oleelliset piirteet ja voimme ennustaa systeemin käyttäytymistä. Erityisesti insinööritieteissä pyritään myös mallitettavan systeemin optimointiin, esimerkiksi löytämään geometria, joka minimoi tai maksimoi jonkun mallitettavan suureen.

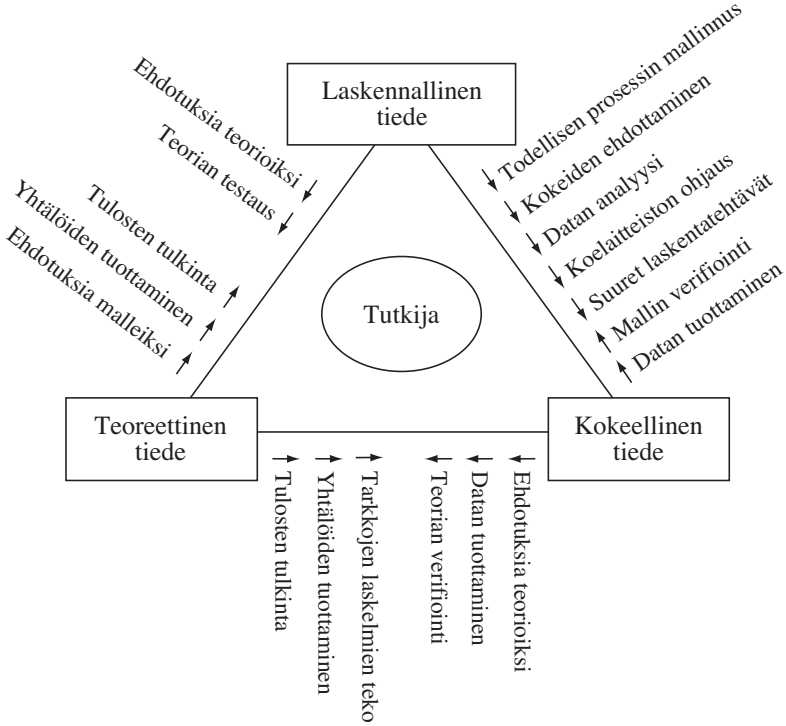
Matemaattisia malleja ei useimmiten voida ratkaista suljetussa muodossa, vaan joudumme käyttämään likiarvoisia ratkaisuja. Tähän tarkoitukseen käytetään numeerisia laskualgoritmeja.

Matemaattisia malleja muodostettaessa ja niitä numeerisesti ratkaistaessa on pyrittävä arvioimaan myös ratkaisun tarkkuutta ja luotettavuutta. Algoritmit eivät toimi kaikissa tilanteissa. Esimerkiksi huono alkuarvaus systeemin ratkaisulle voi johtaa tilanteeseen, jossa algoritmi ei suppene eli emme löydä ratkaisua tehtävälle.

Reaalimaailman ilmiöt ja insinöörien suunnittelemat laitteet ovat käytännössä aina niin monimutkaisia, ettei laskuja voi suorittaa kynällä ja paperilla, vaan on käytettävä apuna tietokoneita. Valitettavasti tietokoneet eivät ole erehtymättömiä, joten saadut tulokset voivat olla vääriä. Syy voi olla laiteviassa, ohjelmiston virheessä, laskentamenetelmässä tai matemaattisessa mallissa.

Kuva 2.1 kertoo tieteellisessä tutkimuksessä käytetyistä menetelmistä: ko-

keellisesta, teoreettisesta ja laskennallisesta tieteestä. Nämä menetelmät ovat käytännön tutkimuksessa usein monenlaisessa vuorovaikutuksessa keskenään, joten niitä on vaikea erottaa toisistaan. Esimerkiksi mittauslaitteiden ohjaukseen voidaan tarvita kehittyneitä laskennallisen tieteen menetelmiä, joita laitetta käyttävä tutkija ei välttämättä tunne.

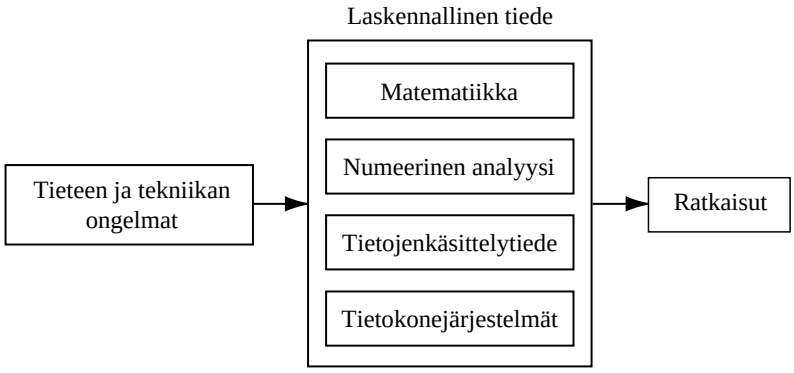


Kuva 2.1: *Tieteen kolme tutkimusmenetelmää.*

Tieteellinen tutkimus on kehittymässä kohti yhä haastavampia tehtäviä. Tyypillistä on monifysikaalisten ongelmien käsittely: luonnonilmiötä simuloidaan toisiinsa kytketyillä malleilla, jotka voivat kuvata esimerkiksi virtauksia, lämmönsiirtoa, säteilyä, sähkömagneettisia kenttiä, rakenteiden mekaniikkaa jne. Tällaisten systeemien tutkimuksessa kokeellinen ja teoreettinen tutkimus on hyvin vaikeaa ja toisinaan jopa mahdotonta. Myöskin laskennallisessa tieteessä vaatimustaso kasvaa koko ajan. Kuva 2.2 havainnollistaa välineitä, joita tarvitaan ongelman ratkaisussa laskennallisen tieteen menetelmin.

2.2 Esimerkkejä matemaattisista malleista: populaatiomallit

Esittelemme seuraavassa lyhyesti, kuinka matemaattinen malli luodaan. Tämä voi olla suurempaa kekseliäisyyttä vaativa tehtävä kuin numeeristen me-



Kuva 2.2: *Ongelmasta ratkaisuun: kaaviossa on esitelty laskennallisen tieteen käyttämiä välineitä ongelmien ratkaisemiseksi.*

netelmien valinta ja soveltaminen matemaattisen mallin ratkaisemiseksi, mitä tämä kirja varsinaisesti käsittelee. Matemaattisen mallin valitsemisessa joudutaan ja myös pyritään tutkittavan ongelman kuvaamiseen yksinkertaistaen, mutta systeemin oleelliset piirteet säilyttäen.

2.2.1 Yksinkertaiset populaatiomallit

Käytämme seuraavassa esimerkkeinä populaatiomalleja. Näitä voidaan kutsua hypoteettisiksi malleiksi, joilla ei ole esimerkiksi fysiikassa esiintyvien säilymislakien universaalia ja eksaktia luonnetta. Tässä luvussa esitettävät populaatiomallit on muodostettu järkeilemällä ja empiiriseen havaintomateriaaliin sovittamalla. Erilaisten populaatiomallien laatiminen on varhaisimpia matematiikan sovelluskohteita perinteisten insinööritieteiden ulkopuolella.

Leonardo Fibonaccin (1170–1230) mukaan nimetty lukusarja $P_{i+1} = P_i + P_{i-1}$, missä $P_0 = P_1 = 1$, syntyi yrityksistä kuvata jänispopulaation kasvua. Yksinkertaisin ja tunnetuin populaatiomalli lienee silti T. R. Malthusin 1798 esittämä käsitys, jossa populaatio kasvaa geometrisen sarjan mukaan. Siten seuraavan sukupolven yksilöiden määrä P_{i+1} on suoraan verrannollinen edellisen sukupolven kokoon P_i:

$$P_{i+1} = (1 + C)P_i.$$

Verrannollisuuskerroin $C + 1 > 1$ takaa, että populaatio kasvaa.

Edellinen *diskreetti malli* voidaan kirjoittaa myös differentiaaliyhtälönä, jos annamme sukupolvien välisen ajan lähestyä arvoa 0 ja samalla oletamme, että populaation kokoa ei tarvitse mitata kokonaisina yksilöinä. Nämä abstrahoinnit eivät tietenkään ole enää yhtäpitäviä varsinaisten luonnossa esiintyvien populaatioiden kanssa. Syntyvää *jatkuvaa mallia* pitääkin tulkita eräänlaisena keskiarvoistuksena. Se pätee hyvin, jos tarkoituksena on kuvata suur-

ta populaatiota kuten bakteeriviljelmiä, jossa yksilöiden välillä ei ole suurta vaihtelua ja ympäristö ei aseta kasvulle mitään rajoja.

Eksponentiaalista kasvua kuvaava differentiaaliyhtälö on

$$P'(t) = cP(t).$$

Kerroin $c > 0$ takaa populaation kasvun.

2.2.2 Yhden lajin mallit

Jo Malthus huomasi, että ympäristö asettaa rajat populaation kasvulle. Edellistä yksinkertaista mallia voikin kehittää tuomalla mukaan ympäristön kantokyvyn K. Voimme ajatella, että jos populaatio P on hyvin pieni, kantokyky ei aseta oleellisesti mitään rajoituksia populaation lisääntymiselle. Jos taas populaation koko on lähellä ympäristön kantokyvyn ylärajaa, populaatio ei voi enää kasvaa.

Kantokyvyn huomioon ottava kasvu on muotoa

$$P'(t) = cP(t)R(P),$$

missä rajoitefunktio $R(P)$ käyttäytyy yllä esitetyllä tavalla: kun P on pieni, $R(P) \approx 1$, ja kun $P \approx K$, niin $R(P) \approx 0$. Eräs mahdollisuus on valita $R(P) = (1 - P/K)$, jolloin lopullinen malli on siis

$$P'(t) = cP(t)(1 - P(t)/K).$$

Tämän yhtälön ratkaisukäyrät

$$P(t) = \frac{KP(0)e^{ct}}{K + (e^{ct} - 1)P(0)}$$

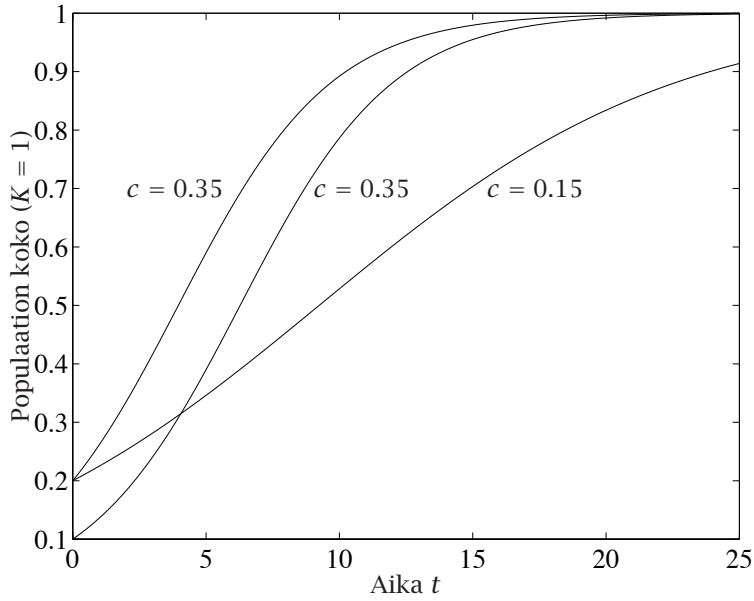
tunnetaan *logistisina käyrinä* (katso kuvaa 2.3). Ratkaisussa esiintyy populaation koko $P(0)$ hetkellä $t = 0$. Käytännössä alkuhetki on tietysti mielivaltainen eli riittää, kun populaatio on joskus laskettu, jolloin ko. hetki voidaan sopia aikaskaalan nollapisteeksi.

Seuraavaksi voimme ottaa malliin mukaan ikäjakauman. Emme ole enää kiinnostuneita pelkästään populaation koosta, vaan haluamme saada selville, kuinka monta yksilöä kuuluu johonkin ikähaarukkaan. Nyt ilmeisesti P on kahden muuttujan funktio $P(t, a)$, missä a kertoo yksilöiden iän. Ikäluokan $[a_1, a_2]$ koko hetkellä t on

$$\int_{a_1}^{a_2} P(t, a)\, da.$$

Jotta saisimme kehitettyä varsinaisen mallin, joudumme ottamaan mukaan ikäjakaumasta riippuvat suureet kuvaamaan syntyvyyttä $s(a)$ ja kuolleisuutta $k(a)$. Suure $s(a)$ kuvaa, kuinka tehokkaasti kukin ikäluokka synnyttää jälkeläisiä. Perusyhtälön voimme johtaa populaation koon muuttumisnopeuden ja kuolleisuuden välisestä yhteydestä:

$$\frac{dP(t, a)}{dt} = -k(a)P(t, a).$$



Kuva 2.3: *Erilaisia parametrin c arvoa ja alkuarvoa $P(0)$ vastaavia logistisia käyriä.*

Koska pätee

$$\frac{dP}{dt} = \frac{\partial P}{\partial t} + \frac{\partial P}{\partial a}\frac{da}{dt}$$

ja $da/dt = 1$ (jokainen yksilö ikääntyy sekunnissa sekunnin verran), päädymme lopulta osittaisdifferentiaaliyhtälöön

$$\frac{\partial P}{\partial t} + \frac{\partial P}{\partial a} = -kP.$$

Syntyvyyden otamme huomioon reunaehdossa vaatimalla, että vastasyntyneiden (siis ikä = 0) lukumäärän pitää täsmätä kaikenikäisten yksilöiden yhteensä tuottaman jälkeläismäärän kanssa:

$$P(t,0) = \int_0^\infty s(a)P(t,a)\,da.$$

Yhden lajin populaatiomalleja voi luonnollisesti tarkentaa lähes rajatta ottamalla mukaan mitä erilaisimpia ilmiöitä. Katsomme seuraavaksi kuitenkin joitakin useamman lajin vuorovaikutusta kuvaavia mallia.

2.2.3 Peto–saalis-mallit

Yksinkertaisin *peto–saalis-malli* on esitettävissä *Lotkan ja Volterran* differentiaaliyhtälöparina

$$\begin{aligned} P'(t) &= s_P P(t)Q(t) - k_P P(t), \\ Q'(t) &= s_Q Q(t) - k_Q P(t)Q(t). \end{aligned}$$

Tässä $P(t)$ on petopopulaation koko ja $Q(t)$ saalispopulaation koko. Petojen syntyvyys riippuu paitsi petojen kulloisestakin lukumäärästä myös saaliseläinten lukumäärästä. Jos saaliseläimiä ei ole, pedot eivät voi lisääntyä. Vakio s_P kertoo millä tehokkuudella pedot pystyvät hyödyntämään ravinnon uusien yksilöiden tuotannossa. Kerroin k_P karakterisoi petopopulaation luonnollista kuolleisuutta.

Saaliseläimiä puolestaan syntyy tasaista tahtia; syntyvyyskerroin on s_Q. Mallissa oletetaan, ettei saaliseläimien lukumäärä koskaan pääse kasvamaan niin suureksi, että ympäristön kantokyky muodostuisi rajoitteeksi. Saalisyksilöt kuolevat tässä mallisa pelkästään petojen uhreina. Todennäköisyys jäädä kiinni on tietysti sitä suurempi, mitä useammin saalis ja peto kohtaavat toisensa, minkä oletetaan riippuvan lajien yksilömäärien tulosta. Vakio k_Q skaalaa kuolleisuuden tämän tulon suhteen.

Alkutilanteesta ja parametrien arvoista riippuen Lotkan ja Volterran yhtälöiden ratkaisut voivat päätyä tasapainotiloihin tai jaksollisiin vaihteluihin. Malli on epäsymmetrinen lajien suhteen: jos petopopulaatio P jostain syystä häviää, sen lukumääräksi jää 0. Tällöin saalispopulaatio Q kasvaa eksponentiaalisesti. Jos sen sijaan saalispopulaatio häviää, petojen lukumäärä alkaakin vähitellen pienentyä.

Edellä esitetyt populaatiomallit ovat kiinnostavia kenties pikemminkin matemaattisen analyysin kannalta kuin todellisuutta koskevien ennustusarvojensa vuoksi. Niiden tutkiminen ei vaadi välttämättä numeriikkaa, sillä yksinkertainen päättely kynän ja paperin kanssa auttaa useimmissa tapauksissa löytämään mallin oleelliset piirteet.

2.2.4 Useamman lajin mallit

Tutustumme lopuksi yhtälöryhmään, jota on käytetty mallittamaan todellista populaatioiden välistä kilpailutilannetta. Yhtälöissä esiintyville kertoimille on saatu kokeelliset arvot laajoilla kenttätutkimuksilla.

Populaatio T koostuu toukista, jotka syövät havupuiden neulasia. Yksilöiden absoluuttisen lukumäärän sijasta T kuvaa nyt populaatiotiheyttä (yksilöitä/hehtaari). Lisäksi käytämme muuttujina puiden neulastiheyttä N ja puiden tiheyttä maastossa P.

Jos terveitä havupuita on paljon ja toukkia vähän, toukkien lukumäärä kasvaa eksponentiaalisesti:

$$T' = c_T T.$$

Kasvua rajoittavat toisaalta neulasten väheneminen ja siitä johtuva puiden kuoleminen ja toisaalta lintujen ja muiden petojen aktiivisuus. Ensin mainittu ilmiö kuvaa ympäristön kantokykyä, joten yritämme soveltaa toukkiin logistisen kasvun yhtälöä

$$T' = c_T T(1 - T/K_T). \tag{2.1}$$

Maksimaalinen toukkien tiheys K_T riippuu tietysti puuston tiheydestä P, joten $K_T = KP$, missä verrannollisuuskerroin K kuvaa, kuinka paljon toukkia yksi puu kestää. Tämä riippuu tietysti puun neulasten tilasta: jos neulasia on paljon (neulastiheys N suuri), niin K on lähellä maksimaalista kantokykyä K_T; jos taas neulasia on harvassa ($N \approx 0$), niin myös kantokyky on pieni. Kyseessä on siis jälleen logistinen käyttäytyminen, ja eräs mahdollisuus olisikin valita $K = K_T N/(N + \gamma)$. Todellisuudessa K riippuu voimakkaammin neulastiheydestä N, ja tämä otetaan huomioon kirjoittamalla yhtälö lopulta muotoon

$$K = \frac{K_T N^2}{N^2 + \gamma^2}. \tag{2.2}$$

Lintujen aiheuttama toukkien väheneminen tuottaa puhtaaseen logistiseen yhtälöön (2.1) negatiivisen lisätermin $-l(T)$. Lintujen lukumäärää ei haluta yksinkertaisuuden vuoksi malliin mukaan, joten oletamme, että saalistustermi riippuu vain toukkien määrästä. Jos toukkia on paljon, linnut syövät niitä tasaisella vauhdilla L_{max}; jos taas toukkia on vähän, niitä myös saalistetaan vähän. Jälleen kyseessä on logistinen ilmiö. Toukkien lukumäärän ollessa hyvin pieni (mutta äärellinen) linnut lakkaavat kokonaan käyttämästä niitä ravintona, sillä on edullisempaa etsiä muuta ravintoa kuin tuhlata resursseja harvojen toukkien etsimiseen. Tämän vuoksi kuvaamme riippuvuutta jälleen logistista käyrää jyrkemmällä yhtälöllä

$$l(T) = \frac{\beta T^2}{\alpha^2 P^2 + T^2}, \tag{2.3}$$

missä toukkien tiheys on skaalattu lasketuksi puuta kohden.

Sijoittamalla yhtälöt (2.3) ja (2.2) yhtälöön (2.1) saamme lopulta toukkapopulaation tiheysvaihteluita kuvaavan yhtälön

$$T' = c_T T \left[1 - \frac{T}{K_T P} \frac{(\gamma^2 + N^2)}{N^2} \right] - \frac{\beta T^2}{\alpha^2 P^2 + T^2}. \tag{2.4}$$

Puiden lukumäärä noudattaa arvatenkin niin ikään perimmiltään logistista kasvua. Rajoittavana tekijänä on paitsi metsämaaston luontainen kyky tuottaa puita myös neulasten lukumäärä N. Jos toukat hävittävät puun neulaset, puu kuolee eikä tuota jälkeläisiä. Niinpä puutiheyden muutokset noudattavat yhtälöä

$$P' = c_P P \left[1 - \frac{P}{K_P} \frac{K_N}{N} \right]. \tag{2.5}$$

Tässä K_P on maksimaalinen puutiheys ja K_N maksimaalinen neulasten tiheys puissa.

Tarvitsemme vielä kolmannen yhtälön neulastiheyden muutoksille. Lähdemme jälleen liikkeelle logistisesta riippuvuudesta

$$N' = c_N N(1 - N/K_N), \tag{2.6}$$

josta on kuitenkin vähennetttävä toukkien aiheuttama kato $\omega T/P$. Vähennys riippuu nimenomaan tekijästä T/P, siis toukkien määrästä puuta kohden. Kerroin ω kuvaa toukkien kykyä tuhota neulasia. Sen kohdalla voidaan suorittaa samanlainen päättely kuin edellä: kun neulastiheys N on pieni, kerroin ω on myös pieni, ja kun N on suuri, ω saavuttaa maksimiarvon (yksi toukka ei pysty syömään rajattomasti neulasia, vaikka niitä olisi paljonkin). Siis myös ω näyttäisi käyttäytyvän logistisesti neulastiheyden N funktiona. Kun N on pieni, toukat joutuvat kilpailemaan ravinnosta. Tällöin niitä kuolee ravinnonpuutteeseen suurempi osuus kuin jos ravintoa jaettaisiin osalle riittävästi ja loput kuolisivat heti pois kuluttamatta mitään. Niinpä kertoimen ω ja neulastiheyden N välinen riippuvuus on voimakkaampi kuin logistinen, ja kirjoitamme niiden välille yhtälön

$$\omega = \frac{\Omega N^2}{\gamma^2 + N^2}. \tag{2.7}$$

Neulastiheyden muutoksia kuvaavaksi yhtälöksi saadaan yhdistämällä yhtälöt (2.6) ja (2.7)

$$N' = c_N N \left[1 - \frac{N}{K_N}\right] - \Omega \frac{T}{P} \frac{N^2}{\gamma^2 + N^2}. \tag{2.8}$$

Onnistuimme siis kirjoittamaan kolmelle tuntemattomalle suureelle $T(t)$, $P(t)$ ja $N(t)$ kolmen differentiaaliyhtälön ryhmän (2.4), (2.5) ja (2.8). Tätä ryhmää voi tutkia analyyttisinkin keinoin, mutta epäilemättä kyseessä on sen verran monimutkainen systeemi, että numeeriset menetelmät tarjoavat nopeimman tavan saada kuva erilaisista mahdollisista tilanteista. Palaamme tämän tapaisten tavallisten differentiaaliyhtälöryhmien alkuarvotehtävien ratkaisuun luvussa 7.

2.3 Mallien käsittely numeerisilla menetelmillä

Numeeriset menetelmät ovat väline matemaattisten mallien käsittelyyn ja ratkaisuun tietokoneessa. Käsittelemme tässä kirjassa tärkeimpiin matemaattisen mallintamisen tehtävätyyppeihin liittyviä numeerisia menetelmiä. Seuraavassa esittelemme numeerisiin menetelmiin ja niiden tietokonetoteutukseen liittyviä käsitteitä.

Ratkaistavat matemaattiset mallit kuvaavat usein jatkuvia ilmiöitä. Tällaisia ovat mm. edellisessä esimerkissä esitellyt differentiaaliyhtälöt tai tässä oppaassa käsiteltävät jatkuvasti differentioituvat optimointitehtävät. Jatkuvien tehtävien muuttujat ovat reaaliarvoisia. Koska tietokoneessa ei voi laskea äärettömän tarkasti eikä äärettömän monessa pisteessä, joudumme käyttämään likiarvostuksia. Laskennassa on tärkeää luotettavuus ja tarkkuus sekä usein myös laskennan tehokkuus.

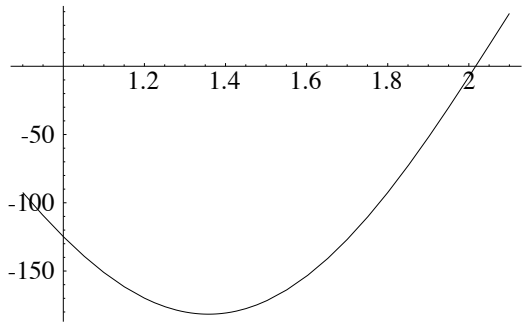
Seuraavaa epälineaarista yhtälöä ei voi ratkaista suljetussa muodossa:

$$e^x - x^2 + x = 0.$$

Myös funktion minimointi johtaa numeerisiin tehtäviin. Voimme vaikkapa etsiä minimikohdan kuvassa 2.4 esitetylle funktiolle

$$f(x) = 160\cos 2x - 64x\sin 2x$$

välillä $x \in [1, 2]$.



Kuva 2.4: *Minimointitehtävä.*

Numeerisilla menetelmillä löydetty ratkaisu on likiarvo alkuperäisen matemaattisen tehtävän ratkaisulle. Yleensä numeerisia menetelmiä käytetään tietokoneiden yhteydessä, jolloin tulee ottaa huomioon mm. liukulukujen rajallinen laskutarkkuus.

Esimerkki 2.3.1 Laskemme lausekkeen $\sin^2 x + \cos^2 x$ arvot, kun $x \in [0, 2\pi]$. Liukulukuaritmetiikan rajallisen tarkkuuden vuoksi tulos ei ole aina matemaattisen identiteetin $\sin^2 x + \cos^2 x \equiv 1$ mukainen. Olkoon esimerkiksi $x_i = ih,\ i = 0, 1, \ldots, 50,\ h = \pi/50$. Matlabilla laskettaessa useat x_i-arvot tuottavat tarkasta arvosta poikkeavan tuloksen:

```
>> x = 0:(pi/50):pi;
>> x(sin(x).^2 + cos(x).^2 ~= 1)
ans =
  Columns 1 through 7
    0.3142  0.3770  0.8796  0.9425  1.1310  1.2566  1.3195
  Columns 8 through 13
    1.6336  2.1991  2.3248  2.6389  2.7018  3.0159
```

Matlabin merkintä ~= tarkoittaa erisuuruuden vertailua. Komento siis tulosti ne pisteet, joissa lausekkeen arvo poikkesi tarkasta arvosta. Virheet ovat kuitenkin varsin pieniä:

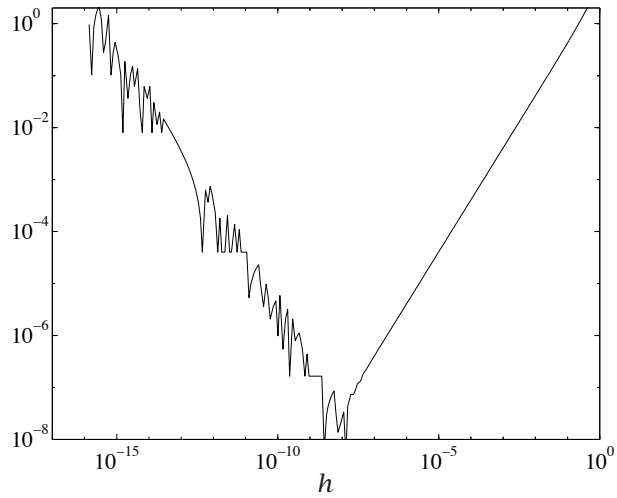
```
>> max(sin(x).^2 + cos(x).^2 - 1)
ans =
   2.2204e-16
```

Matemaattisia tehtäviä ei voi yleensä ratkaista tarkasti äärellisellä määrällä laskutoimituksia. Voimme laskea esimerkiksi sarjan summalle approksimaation käyttämällä äärellisen määrän termejä. Tällöin syntyvää virhettä kutsutaan *katkaisuvirheeksi* (truncation error). Lisäksi tietokoneen aritmetiikka tuottaa virheitä saatuun tulokseen. Esimerkiksi sarjan summauksessa tarvittavat yhteenlaskut tuottavat virheitä johtuen liukulukuaritmetiikan ominaisuuksista. Näitä virheitä kutsutaan *pyöristysvirheiksi* (rounding error).

Esimerkki 2.3.2 Voimme arvioida funktion $f(x)$ derivaattaa differenssikaavalla

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}.$$

Jos asetamme askeleen h pieneksi, voimme pienentää katkaisuvirhettä. Toisaalta tällöin pyöristysvirheet voivat kasvaa ja pudottaa arvion tarkkuutta: kun askel h on pieni, arvot $f(x+h)$ ja $f(x)$ ovat samaa suuruusluokkaa ja siten erotuksessa on vähän merkitseviä numeroita. Kuva 2.5 esittää differenssiarvion virhettä eri askelpituuksilla funktiolle $f(x) = xe^x$ pisteessä $x = 1$. Laskennassa on käytetty 64-bittistä IEEE-aritmetiikkaa, jolloin paras approksimaatio saadaan askelpituudella $h \approx \sqrt{\epsilon_{\text{kone}}} \approx 10^{-8}$.



Kuva 2.5: *Funktion derivaatan arviointi differenssillä. Kuva esittää differenssiarvion virheen askelpituuden funktiona.*

Numeerisia menetelmiä käytettäessä tulee pitää mielessä

- menetelmän matemaattinen tausta ja sen asettamat ehdot
- millaisiin tehtäviin menetelmä sopii (ja millaisiin ei)
- menetelmän skaalautuvuus tehtävän koon kasvaessa
- menetelmän käyttäytyminen laskennan kuluessa

- virheiden havaitseminen ja niistä toipuminen
- tulosten tarkistaminen alkuperäisen tehtävän kannalta (ovatko tulokset fysikaalisesti järkeviä, suuruusluokat oikein jne.).

Matemaattisten tehtävien numeerisessa ratkaisemisessa seuraavat kaksi periaatetta ovat tärkeitä:

- äärettömän prosessin korvaaminen äärellisellä tai diskreetillä
  - integraalin muuttaminen summaukseksi
  - funktion esittäminen kantafunktioiden lineaarikombinaationa
  - jatkuvan geometrian muuttaminen diskreetiksi hilaksi tai elementtiverkoksi
- epälineaarisen tehtävän korvaaminen lineaarisella
  - funktion linearisointi esimerkiksi Taylorin sarjakehitelmän avulla
  - tehtävän iteratiivinen ratkaisu: ratkaistaan jono lineaarisia tehtäviä epälineaarisen tehtävän ratkaisemiseksi
  - paloittain lineaaristen funktioiden käyttö kantafunktioina.

Suuri osa tämän kirjan käsittelemistä numeerisista menetelmistä johtaa lineaaristen yhtälöryhmien ratkaisemiseen. Tässä ja eräissä muissa tapauksissa matriiseja muunnetaan helpommin käsiteltävään muotoon

- matriisihajotelmia hyödyntämällä
- tallentamalla matriisit harvassa muodossa (ilman nolla-alkioita) muistitilan säästämiseksi.

Koska esimerkiksi sarjakehitelmän katkaisu tuottaa likiarvon sarjalle, täytyy laskennassa pystyä arvioimaan virheen suuruusluokkaa. Lisäksi tietokoneen aritmetiikan äärellinen tarkkuus saa aikaan virheiden suurenemista, varsinkin pitkissä laskutoimitusten jonoissa.

Esimerkki 2.3.3 Pisteessä $x = x_0$ analyyttinen funktio voidaan esittää Taylorin sarjakehitelmänä pisteen $x = x_0$ ympärillä:

$$f(x) = \sum_{n=0}^{\infty} \frac{1}{n!}(x - x_0)^n f^{(n)}(x_0) = f(x_0) + (x - x_0)f'(x_0) + \dots.$$

Saamme funktiolle $f(x) = \cos x$ pisteessä $x_0 = 0$ sarjakehitelmän

$$\cos x = 1 - \frac{x^2}{2} + \frac{x^4}{24} - \frac{x^6}{720} + \dots.$$

Esitämme tälle sarjakehitelmälle virhearviot kappaleessa 2.5.

Matemaattista tehtävää kutsutaan *hyvin asetetuksi* (well posed), mikäli seuraavat ehdot toteutuvat:

1. Ratkaisu on olemassa kaikilla lähtöarvoilla.
2. Ratkaisu on yksikäsitteinen.
3. Ratkaisu x riippuu jatkuvasti tehtävän lähtöarvoista y. Siis jos $y_n \to y$, niin $x_n \to x$.

Huonosti asetetuilla (ill posed) tehtävillä ei ole näitä ominaisuuksia.

Voimme määritellä *numeerisen stabiilisuuden* seuraavasti: numeerinen menetelmä on stabiili, jos ratkaisu ei riipu lähtöarvojen häiriöstä enempää kuin alkuperäinen (matemaattinen) tehtävä.

Huomautus 2.3.1 Jos matemaattinen tehtävä on huonosti asetettu, sitä ei voi ratkaista luotettavasti millään numeerisella menetelmällä.

Huonosti asetettu tehtävä voidaan mahdollisesti ratkaista jollain lähtöarvojen osavälillä, vaikka yleinen ratkaisu olisikin mahdotonta. Jos tämä lähtöarvojen — tehtävätyypistä riippuen esimerkiksi alkuarvauksen, alkuarvon, reunaehtojen tai parametrien arvojen — osaväli on mallin kannalta kiinnostava ja mielekäs, tämä toki riittää. Kuitenkin esimerkiksi differentiaaliyhtälöiden stabiilisuusalueet voivat aiheuttaa voittamattomia ongelmia tehtävän numeeriselle ratkaisemiselle.

Esimerkki 2.3.4 Korkea-asteiset polynomit ovat virhealttiita. Piirrämme polynomin

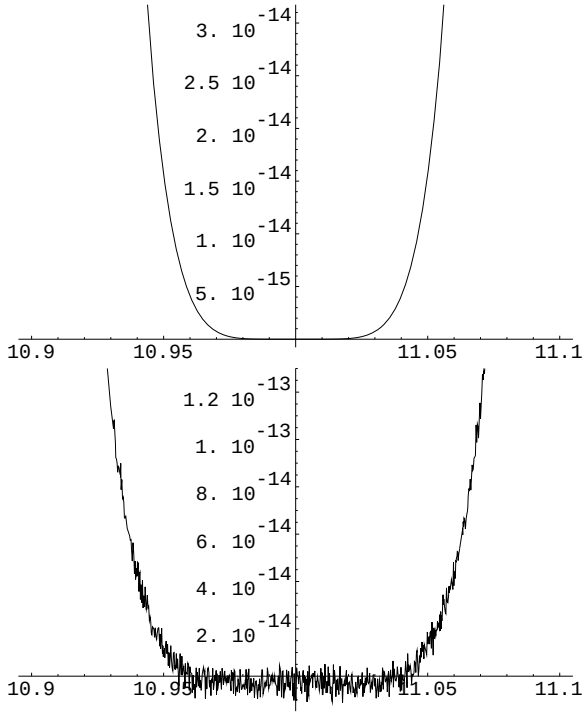
$$f(x) = (x/10 - 1.1)^6$$

kuvaajan (kuva 2.6) käyttäen yllä olevaa esitysmuotoa ja tämän jälkeen auki kirjoitettua muotoa. Johtuen liukulukuaritmetiikan epätarkkuudesta tuottaa auki kirjoitettu esitys kohinaisen kuvaajan, josta ei voi määrittää funktion nollakohtaa tai minimiä.

2.4 Algoritmit ja niiden skaalautuvuus

Algoritmi kuvaa äärellisen määrän operaatioita, jotka suoritetaan annetussa järjestyksessä. Algoritmi määrää, kuinka tehtävä ratkaistaan tai kuinka ratkaisua approksimoidaan.

Pseudokoodi on tapa kuvata algoritmeja niin, että oleelliset ideat tulevat lukijalle selviksi. Tietokoneelle täytyy algoritmi kuvata tietenkin yksityiskohtaisemmin ja eksaktimmin ohjelmointikielen avulla. Tässä teoksessa käytämme algoritmien kuvaamisessa seuraavan tyylistä pseudokoodia:



Kuva 2.6: *Funktion $f(x) = (x/10 - 1.1)^6$ kuvaajan piirtäminen kahdella eri esitystavalla. Erot johtuvat korkea-asteisen polynomin häiriöalttiudesta.*

Algoritmi 2.4.1 (Esimerkki algoritmien esittämisestä)

aseta $f_a = f(a_1)$ ja $f_b = f(b_1)$ sekä $k = 1$
repeat
  if $f_a < f_b$
    $\vdots$
  end
  $k = k + 1$
until $|a_k - b_k| < \epsilon$

Yksi keskeinen algoritmien ominaisuus on virheiden käyttäytyminen laskennan kuluessa. Voimme määritellä *algoritmin stabiilisuuden* samaan tapaan kuin numeerisen stabiilisuuden: Jos pienet muutokset syöttödatassa eivät aiheuta lopullisissa tuloksissa suurempia muutoksia kuin alkuperäisessä tehtävässä, algoritmi on stabiili. Useimmat algoritmit ovat stabiileja vain osalla mahdollisista syöttötiedoista.

Käsin tai tietokoneilla laskettavissa algoritmeissa on tärkeää tietää algoritmin vaatima työmäärä ja muistitila.

Esimerkki 2.4.1 Cramerin sääntöä voi käyttää lineaarisen yhtälöryhmän $A\mathbf{x} = \mathbf{b}$ ratkaisemiseen. Kolmen yhtälön tapauksessa saamme ratkaisukaavat

$$x_1 = \frac{\Delta_1}{\det(A)}, \quad x_2 = \frac{\Delta_2}{\det(A)}, \quad x_3 = \frac{\Delta_3}{\det(A)}.$$

Tässä $\det(A)$ on matriisin A determinantti. Merkintä Δ_i tarkoittaa determinanttia, jossa matriisin A sarake i on korvattu vektorilla $\mathbf{b}$.

Cramerin säännössä lasketaan $n \times n$ -matriisien determinantteja $n + 1$ kertaa. Kokonaistyömäärä on luokkaa $(n + 1)!(n - 1)$ laskutoimitusta, jos käytetään klassista rekursiivista determinanttikaavaa. Tätä operaatioiden lukumäärän suuruusluokkaa kutsutaan Cramerin säännön *kompleksisuudeksi*.

Lineaarisen yhtälöryhmän ratkaisu käyttäen Gaussin eliminointia vaatii puolestaan noin $n^3/3$ kertolaskuoperaatiota ja yhtä monta yhteenlaskua.

Täten kun $n = 10$, Cramerin sääntöä käyttäen tarvitaan yli 350 miljoonaa laskutoimitusta yhtälöryhmän ratkaisemiseksi. Gaussin eliminoinnissa työmäärä on luokkaa 1000 laskutoimitusta.

Cramerin sääntö on siis erittäin tehoton kun muuttujia on vähänkin enemmän. Tästä huolimatta menetelmää suositellaan silloin tällöin yhtälöryhmän ratkaisemiseen [JS97]. Nykyisillä kehittyneillä ohjelmistoilla voi sitä vastoin ratkaista yhtälöryhmiä, joissa on kymmeniä miljoonia tuntemattomia muuttujia.

Algoritmin kompleksisuutta mitattaessa useimmiten unohdetaan vakiokerroin ja käytetään $\mathcal{O}$-notaatiota. Gaussin eliminoinnin kompleksisuus on siis $\mathcal{O}(n^3)$ ja Cramerin säännön $\mathcal{O}(n!\,n)$.

Algoritmin tehokkuutta voi mitata työmäärän lisäksi muistinkäytöllä. Jos lineaarisen yhtälöryhmän kerroinmatriisi on tiheä, tarvitaan muistitilaa n^2 alkiolle. Jos matriisi on harva ja käytetään harvalle matriisille sopivaa talletustapaa, on muistitilavaatimus huomattavasti pienempi. Joissakin tapauksissa kerroinmatriisia ei edes tarvitse muodostaa, jolloin tarvitaan tilaa ehkä vain ratkaisuvektorille. Tällöin muistintarve on vain luokkaa n alkiota.

2.5 Virhelähteitä tieteellisessä laskennassa

Matemaattisten ongelmien ratkaiseminen tietokoneella on vaativa tehtävä. Virheitä syntyy matemaattisen mallin muodostamisessa, mallin diskretoinnissa tietokoneen ymmärtämään muotoon, mallin numeerisessa käsittelyssä sekä tulosten analysoinnissa.

Virheitä voi esiintyä myös käytettäessä laajasti käytettyjä ja monipuolisesti testattuja ratkaisumenetelmiä ja ohjelmistoja. Mitä monimutkaisempi tehtävä, sitä todennäköisempää on törmätä jonkinlaiseen virheeseen. Ammattimaisesti kehitetyissä ja hyvin testatuissa valmisohjelmistoissakin ilmenee silloin tällöin ongelmia — puhumattakaan itse tehdyistä ja mahdollisesti puutteellisesti testatuista ohjelmakoodeista.

Usein tietokoneiden ja ohjelmistojen vertailussa painotetaan tehokkuutta, vaikka tärkein kriteeri on luotettavuus. Väärillä tuloksilla ei ole käyttöä, vaikka ne olisi tuotettu ennätysnopeasti.

Virheet numeerisessa laskennassa voivat aiheuttaa suuria taloudellisia menetyksiä. Kun koodeja kehitetään vuosikausia ja ehkä sovelletaan tilanteissa, joihin niitä ei ole alunperin suunniteltu, on oltava äärimmäisen huolellinen. Erityisesti liukulukuaritmetiikan epätarkkuudesta johtuvia ongelmia esiintyy runsaasti, varsinkin jos simulaatiojakson pituus kasvaa. Hyvä esimerkki on Euroopassa kehitetyn Ariane 5 -raketin laukaisun epäonnistuminen, kun raketin ohjaukseen käytettiin raketin nelosversiolle kehitettyä ohjelmistoa. Tämä johti vääriin ohjauskomentoihin. Lopulta raketti täytyi räjäyttää.

Matemaattisen mallin muodostamisessa ongelmia voivat tuottaa esimerkiksi seuraavat tekijät:

- Mallissa käytetyt vakiot tai lähtödata tunnetaan epätarkasti tai on mitattu systemaattisesti virheellisesti.
- Idealisointi: tehdään liian suuria yksinkertaistuksia matemaattisessa mallissa (oletetaan homogeeninen materiajakautuma tms.).
- Ongelma on huonosti asetettu. Esimerkiksi epästabiileihin differentiaaliyhtälöihin ei sovellu mikään numeerinen menetelmä — tai numeerisia tuloksia on arvioitava tilastollisin kriteerein.

Numeerisen mallin muodostamisessa ja ratkaisussa virheellisiä tuloksia voivat tuottaa seuraavat syyt:

- Käytössä on epästabiili numeerinen menetelmä, jolloin tulos voi olla hyvin epätarkka tai täysin virheellinen.
- Pyöristysvirheet laskennan aikana: jos tietokoneessa kertoo kaksi s-numeroista liukulukua keskenään, ei voi aina odottaa vastausta s:llä tai $s-1$:llä numerolla. Vastaus on katkaistava tai pyöristettävä lukuun, joka koneessa voidaan esittää. Epästabiilissa numeerisessa menetelmässä myös pyöristysvirheet voivat kasautua. Tästä on esimerkkinä tilanne, jossa lasketaan hyvin erisuuruisia lukuja yhteen.
- Diskretointivirhe. Funktio diskretoidaan hilapisteiksi tai kantafunktioiden avulla (FEM). Derivaatta approksimoidaan esimerkiksi erotusosamäärällä (differenssillä).
- Katkaisuvirhe. Päättymätön sarja joudutaan katkaisemaan johonkin termiin. Epälineaarista funktiota approksimoidaan lineaarisella.
- Väärä menetelmä: oikeat tulokset, mutta laskenta vie vuosia.

Myös seuraavan tyyppisiin virheisiin tulee varautua:

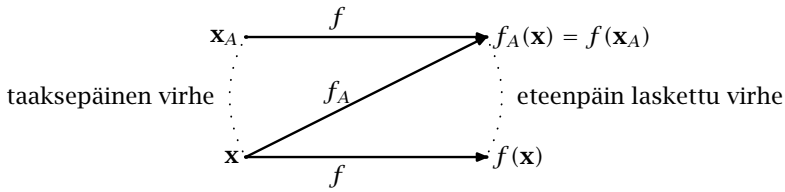
- Inhimilliset virheet. Ohjelmointivirhe, tiedostojen tuhoaminen jne.
- Tulosten tulkintavirhe.

- Laite- ja varusohjelmistovirheet: tietokone laskee väärin tai kääntäjä tuottaa virheellistä koodia.

Liukulukuaritmetiikan tuottamia virheita voidaan tarkastella kahdesta eri näkökulmasta:

- *eteenpäin laskettu virhe* (forward error): arvioimme likimääräisen tuloksen $f_A(\mathbf{x})$ etäisyyttä tarkasta ratkaisusta $f(\mathbf{x})$
- *taaksepäinen virhe* (backward error): etsimme sellaiset tehtävän parametrit $\mathbf{x}_A$, jotka tuottavat tarkalla aritmetiikalla saman arvon $f(\mathbf{x}_A)$ kuin likimääräinen tulos $f_A(\mathbf{x})$.

Lähestymistapoja on verrattu kuvassa 2.7.



Kuva 2.7: *Virheet liukulukuaritmetiikassa. Piste $\mathbf{x}_A$ on valittu siten, että se tuottaa eksaktilla aritmetiikalla saman tuloksen $f(\mathbf{x}_A)$ kuin liukulukuaritmetiikalla laskettu tulos $f_A(\mathbf{x})$.*

Esimerkki 2.5.1 Voimme arvioida funktiota $f(x) = \cos x$ Taylorin sarjakehitelmällä pisteessä $x = 0$:

$$\cos x \approx f_A(x) = 1 - \frac{x^2}{2}.$$

Eteenpäin laskettu virhe on $f(x) - f_A(x)$. Taaksepäisen virheen määrittämiseksi etsimme pistettä x_A, jolla pätee $f(x_A) = f_A(x)$ eli siis

$$\cos x_A = f_A(x) \quad \Longrightarrow \quad x_A = \arccos f_A(x).$$

Arvolla $x = 0.2$ saamme kuuden desimaalin tarkkuudella $f(0.2) = 0.980067$ ja $f_A(0.2) = 0.980000$. Siten $x_A = \arccos 0.980000 = 0.200335$. Näin saamme

$$\text{eteenpäin laskettu virhe} \;=\; f(x) - f_A(x) = 6.7 \cdot 10^{-5},$$

$$\text{taaksepäinen virhe} \;=\; x - x_A = -3.35 \cdot 10^{-4}.$$

Siispä tarkkuus on varsin hyvä kummallakin tavalla mitattuna.

2.6 Matemaattiset ohjelmistot

CSC:n opas *Matemaattiset ohjelmistot* [Haa98] kertoo yleisesti käytetyistä matemaattisista ohjelmistoista sekä verkkopalvelujen hyödyntämisestä (esimerkiksi Netlib). Teos on luettavissa verkko-osoitteessa `http://www.csc.fi/oppaat/mat.ohj/`.

Matemaattiset ohjelmistot voidaan luokitella esimerkiksi seuraavasti:

1. Tietylle sovellusalueelle kehitetyt mallinnusohjelmistot: Ansys (rakenneanalyysi), ADAMS (rakenteiden mekaniikka), FLUENT (virtauslaskenta), Gaussian98 (kvanttikemia).
2. Tiettyä menetelmää käyttävät, mutta usealle sovellusalueelle käyvät mallinnusohjelmistot: ELMER ja PDE2D (elementtimenetelmä).
3. Interaktiiviset matemaattiset yleisohjelmistot: Matlab, Mathematica, Maple.
4. Matemaattiset aliohjelmakirjastot: BLAS, LAPACK, IMSL, NAG, Harwell Subroutine Library (HSL).
5. Lähdekieliset rutiinikokoelmat: Netlib (TOMS ym.), Numerical Recipes.
6. Itse kehitetyt ohjelmistot käyttäen jotain ohjelmointikieltä, esimerkiksi Fortran 90/95-, C- tai C++-kieltä.

Ohjelmistoja Mathematica ja Maple käytetään paljon niiden monipuolisuuden vuoksi. Nämä symbolisen laskennan ohjelmistot osaavat käsitellä matemaattisia lausekkeita symbolisessa muodossa. Myös numeerinen laskenta on mahdollista, mutta usein tehotonta verrattuna käännettäviin ohjelmointikieliin.

Matlab (nimi tulee sanoista *matrix laboratory*) on matriisien käsittelyyn tehty numeerinen ohjelmisto. Ohjelmisto soveltuu myös graafisten esitysten tuottamiseen.

Matemaattiset aliohjelmakirjastot ovat algoritmikokoelmia, jotka on tarkoitettu käytettäviksi omista ohjelmista. Aliohjelmakirjastoja kannattaa käyttää aina, kun niistä löytyy sopiva rutiini käyttäjän laskentatehtävään. Useimmat näistä aliohjelmakirjastoista ovat Fortran-kielisiä, mutta niitä voi kutsua myös C-kielisistä ohjelmista.

CSC:n koneilta löytyy kaupalliset aliohjelmakirjastot IMSL ja NAG. Ohjelmistot on kuvattu selkeästi ja perusteellisesti laajoissa käsikirjoissa. Lisäksi apuna on paljon esimerkkiohjelmia. IMSL- ja NAG-aliohjelmakirjastot ovat tehokkaita ja luotettavia, joten niitä kannattaa käyttää itse kirjoitettujen koodien sijasta, jos vain mahdollista.

Numeeriseen lineaarialgebraan on kehitetty LAPACK-aliohjelmakirjasto, joka on saatavissa Fortran-lähdekoodina Netlib-verkkoarkistosta (katso alla). LAPACK-kirjastosta on myös koneenvalmistajien omia, kyseiselle laitteistolle optimoituja versioita.

Netlib on Internetissä toimiva verkkokirjasto, joka sisältää mm. lähdekielisiä aliohjelmakirjastoja. Netlib-arkistoa voi käyttää www-selainten avulla; osoite on `http://www.netlib.org`.

TOMS-algoritmit ovat kansainvälisen ACM-järjestön (Association for Computing Machinery) julkaisemia tutkijoiden kirjoittamia matemaattisten algoritmien toteutuksia. Algoritmit ja niiden toteutukset (useimmiten Fortran-kielellä) julkaistaan lehdessä *ACM Transactions on Mathematical Software* (ACM TOMS).

TOMS-algoritmeja voi käyttää esimerkiksi ohjelmissa, joiden täytyy olla siirrettäviä koneelta toiselle (eli lähdekoodi on saatavilla). Toisaalta TOMS-algoritmeissa on julkaistu sellaisia erikoisalgoritmeja, joita ei löydy kaupallisista aliohjelmakirjastoista. TOMS-algoritmit ovat saatavilla Netlib-arkiston hakemistosta `toms`.

Lähdekielisten ohjelmistojen asentaminen edellyttää ohjelmiston etsimistä www:stä, siirtämistä omalle koneelle ja asentamista. Tämä voi vaatia varsin laajaa tietämystä koneen käyttöjärjestelmästä (Windows, Mac OS, Linux, Unix jne.) ja ohjelmointiympäristöstä.

CSC:n koneympäristö perustuu Unix-käyttöjärjestelmille. Käytettävissä olevat ohjelmistot löytyvät www-osoitteesta `http://www.csc.fi/cschelp/sovellukset/ohjelmistolista.html`.

2.7 Lisätietoja

Kertaamme liitteessä A tämän oppaan lukemisessa tarvittavia matemaattisia perusasioita. Liitteessä B kerromme lyhyesti liukulukuaritmetiikan ominaisuuksia. Liitteessä C on CSC:n yhteystiedot.

Numeerisen analyysin perusteita käsitellään lukuisissa kirjoissa. *Numerical Mathematics and Computing* [CK94] ja *Numerical Analysis* [BF97] edustavat amerikkalaista oppikirjaperinnettä lukuisine esimerkkeineen. Alunperin saksalaiset teokset *Introduction to Numerical Analysis* [SB93] ja *Numerical Mathematics* [HH91] puolestaan lähestyvät aihetta paikoitellen hiukan teoreettisemmin. Käytännönläheiseksi oppikirjaksi sopii teos *Numerical Mathematics* [QSS00]. Suomenkielisen numeerisen analyysin tarjonta on niukkaa, mutta oppikirjat *Numeerinen matematiikka* [MNV82] ja *Numeeriset menetelmät* [Mäk98] esittelevät hyvin alan perusteet.

Syvällisemmin numeeristen algoritmien ominaisuuksiin voi tutustua teoksissa *Accuracy and Stability of Numerical Algorithms* [Hig96] ja *Real Computing Made Real* [Act96]. Matemaattista mallintamista käsitellään esimerkiksi teoksissa *Mathematical Modelling* [Kap88] ja *The Nature of Mathematical Modeling* [Ger99]. Teos *Mathematical Models in the Applied Sciences* [Fow97] on näitä vaativampi katsaus matemaattisiin malleihin.

Www-palvelusta *Guide to Available Mathematical Software* (`http://gams.nist.gov/`) löytyy monipuolinen hakemisto eri tyyppisiin tehtäviin soveltuvista matemaattisista ohjelmistoista. Myös www-palvelu *Math Archives* (`http://archives.math.utk.edu/`) on hyvä lähtökohta matemaattisten ohjelmistojen etsimiseen.

Teos *Introduction to Algorithms* [CLR90] soveltuu johdatukseksi algoritmien analysointiin. Ohjelmoinnista kerrotaan myös CSC:n oppikirjassa *Fortran 90/95* [HRR01] sekä oppaassa *Rinnakkaisohjelmointi MPI:llä* [HM01]. *Numerical Recipes* -teoksista [PTVF92a, PTVF92b] ja kirjasta *Numerical Algorithms with Fortran* [EMU96] löydät ratkaisualgoritmeja lähdekoodina. Tarkista kuitenkin aina koodin soveltuvuus ratkaistavaan tehtävään. Vaativiin tehtäviin

on syytä käyttää monipuolisesti testattuja rutiineja, joita löydät esimerkiksi kaupallisista ohjelmistoista tai Netlibistä.

3 Lineaariset yhtälöryhmät

Lineaaristen yhtälöryhmien ratkaiseminen on useiden laskennallisen tieteen ongelmien aikaavievin osa. Tässä luvussa esittelemme yhtälöryhmien käsittelyä ja ratkaisua. Luotettavuuden lisäksi korostamme ratkaisun tehokkuutta, joka tulee erityisen tärkeäksi ratkottaessa suuria lineaarisia yhtälöryhmiä.

3.1 Mistä yhtälöryhmät ovat peräisin

Monet ongelmat johtavat suoraan lineaarisiin yhtälöryhmiin, ja toisissa nämä ovat algoritmin osa. Lineaarisia yhtälöryhmiä syntyy mm. ratkaistaessa tavallisia differentiaaliyhtälösysteemejä implisiittisillä menetelmillä, osittaisdifferentiaaliyhtälöiden ratkaisussa sekä myös optimointialgoritmeissa ja epälineaaristen yhtälöiden ratkaisussa.

Esimerkki 3.1.1 *Juha, Jussi, Ville ja Yrjö kirjoittivat 380-sivuisen kirjan. Jussi kirjoitti neljäsosan kaikkien muiden yhteisestä sivumäärästä. Juha ja Ville kirjoittivat 32 sivua enemmän kuin Yrjö ja Jussi yhteensä. Juha kirjoitti yhtä monta sivua kuin Ville ja Yrjö yhteensä. Kuinka monta sivua kukin kirjoitti?* Ratkaisu: Merkitään Juhan, Jussin, Villen ja Yrjön kirjoittamia sivumääriä muuttujilla J_h, J_s, V ja Y. Sivumäärä voidaan ratkaista lineaarisesta yhtälöryhmästä

$$\begin{aligned} J_h + J_s + V + Y &= 380, \\ 4J_s &= J_h + Y + V, \\ J_h + V &= Y + J_s + 32, \\ J_h &= V + Y, \end{aligned}$$

eli matriisimuodossa

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & 4 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & 0 & -1 & -1 \end{pmatrix} \begin{pmatrix} J_h \\ J_s \\ V \\ Y \end{pmatrix} = \begin{pmatrix} 380 \\ 0 \\ 32 \\ 0 \end{pmatrix}.$$

Myöhemmin tässä luvussa esiteltävillä menetelmillä voidaan laskea, että Juha kirjoitti 152 sivua, Jussi 76, Ville 54 ja Yrjö 98 sivua.

Lineaariset yhtälöryhmät voivat olla joko tiheitä tai harvoja. Tiheissä yhtälöryhmissä olennaisesti kaikki matriisin alkiot ovat nollasta poikkeavia. Harvassa matriisissa on kullakin matriisin rivillä vain pieni osa, tyypillisesti luokkaa kymmenkunta, nollasta poikkeavia alkioita.

Esimerkiksi ratkaistaessa osittaisdifferentiaaliyhtälöitä elementtimenetelmällä kuluu suurin osa laskenta-ajasta ison harvan lineaarisen yhtälöryhmän ratkaisuun. Näissä tehtävissä voi olla jopa miljoonia muuttujia. Tällöin tulee käyttää tehokkainta mahdollista lineaaristen yhtälöryhmien ratkaisumenetelmää.

Lineaarisia yhtälöryhmiä voidaan ratkaista joko suorilla tai iteratiivisilla menetelmillä. Suorat menetelmät perustuvat Gaussin eliminointiin eli matriisin alkioiden nollaamiseen laskemalla eri matriisin rivejä yhteen.

Iteratiiviset menetelmät ovat alkuarvauksesta lähtevä iteraatioprosessi, jossa iteraatiota jatketaan, kunnes haluttu tarkkuus on saavutettu. Iteratiiviset menetelmät eivät yleensä muuta kerroinmatriisia, ja siten ne sopivat harvojen lineaaristen yhtälöryhmien ratkaisemiseen. Iteratiiviset menetelmät voivat olla nopeampia kuin suorat menetelmät, mutta ne ovat samalla hankalampia käyttää.

Tässä luvussa esittelemme ensin suorien menetelmien perustekniikoita sekä tarkastelemme yhtälöryhmien ratkaisemisen teoriaa. Tämän jälkeen käsittelemme harvojen yhtälöryhmien ratkaisemista suorilla menetelmillä. Sitten esittelemme tehokkaita iteratiivisia menetelmiä ja lopuksi kerromme eräistä erikoismenetelmistä.

3.2 Gaussin eliminointi

Perusmenetelmä lineaaristen yhtälöryhmien ratkaisemiseen on Gaussin eliminointi. Tässä menetelmässä matriisin lävistäjän alapuolella olevat alkiot nollataan laskemalla matriisin rivejä yhteen. Eliminoinnin jälkeen matriisista on tullut ns. yläkolmiomatriisi, eli se on muotoa

$$\begin{pmatrix} \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times \\ & & \times & \times & \times & \times \\ & & & \times & \times & \times \\ & & & & \times & \times \\ & & & & & \times \end{pmatrix}.$$

Gaussin eliminoinnin ensimmäisessä vaiheessa matriisin ensimmäinen rivi lisätään sopivalla luvulla kerrottuna muihin riveihin siten, että ensimmäisessä sarakkeessa olevat alkiot nollautuvat. Vastaavalla tavalla eliminoidaan muissa sarakkeissa lävistäjän alapuoliset alkiot. Toinen vaihe on niin sanottu takaisinsijoitusvaihe, jossa lähtien viimeisestä tuntemattomasta voidaan laskea arvot muille tuntemattomille.

Esimerkki 3.2.1 Tarkastellaan yhtälöryhmää

$$\begin{aligned} 2x_1 + 4x_2 - 2x_3 &= 2, \\ 4x_1 + 7x_2 + x_3 &= 2, \\ 1x_1 + 3x_2 - 3x_3 &= 0, \end{aligned}$$

eli matriisimuodossa

$$\begin{pmatrix} 2 & 4 & -2 \\ 4 & 7 & 1 \\ 1 & 3 & -3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \\ 0 \end{pmatrix}.$$

Gaussin eliminoinnin ensimmäisessä vaiheessa nollataan matriisin ensimmäisessä sarakkeessa olevat lävistäjän alapuoliset alkiot. Tämä saadaan aikaan lisäämällä matriisin ensimmäinen rivi toiseen riviin kerrottuna luvulla -2 ja kolmanteen riviin kerrottuna luvulla -0.5:

$$\begin{pmatrix} 2 & 4 & -2 \\ 0 & -1 & 5 \\ 0 & 1 & -2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 2 \\ -2 \\ -1 \end{pmatrix}$$

Toisessa vaiheessa matriisin toinen rivi lisätään kolmanteen riviin kerrottuna luvulla $+1$:

$$\begin{pmatrix} 2 & 4 & -2 \\ 0 & -1 & 5 \\ 0 & 0 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 2 \\ -2 \\ -3 \end{pmatrix}$$

Matriisi on nyt saatettu yläkolmiomuotoon. Muuttujien arvot saadaan nyt ns. takaisinsijoituksella: viimeisestä yhtälöstä voidaan lukea, että muuttujan x_3 arvo on -1. Tätä käyttäen saadaan toisesta yhtälöstä $x_2 = -3$. Kun x_3:n ja x_2:n arvo sijoitetaan ensimmäiseen yhtälöön, saadaan $x_1 = 6$.

Algoritmina Gaussin eliminointi ja takaisinsijoitus näyttävät seuraavilta:

Algoritmi 3.2.1 (Gaussin eliminointi ja takaisinsijoitus)

for $k = 1, 2, \ldots, n-1$ (käydään kaikki matriisin sarakkeet läpi)
  for $i = k+1, \ldots, n$ (nollataan lävistäjän alapuoliset alkiot)
    $r = a(i,k)/a(k,k)$
    $a(i, k+1:n) = a(i, k+1:n) - r * a(k, k+1:n)$
    $b(i) = b(i) - r * b(k)$ (päivitetään oikean puolen vektori)
  end
end

(Takaisinsijoitus:)
$x(n) = b(n)/a(n,n)$ (luetaan viimeisimmän muuttujan arvo)
for $k = n-1, n-2, \ldots, 1$ (muut muuttujat)
  $s = a(k, k+1:n) * x(k+1:n)$
  $x(k) = (b(k) - s)/a(k,k)$
end

Käytännön laskutehtävissä joudutaan kiinnittämään huomiota Gaussin eliminoinnin laskutarkkuuteen ja pyöristysvirheisiin. Numeerisesti edullisinta olisikin, jos algoritmissä r-kertoimia laskettaessa voitaisiin jakaa mahdollisimman suurilla luvuilla $A(k,k)$. Tätä lävistäjäalkiota kutsutaan myös tukialkioksi. *Tuenta* (pivoting) tarkoittaa matriisin rivien ja sarakkeiden vaihtamista siten, että Gaussin eliminoinnissa tukialkioksi tulee mahdollisimman suuri luku. Kullakin uloimman k-silmukan iteraatiolla etsitään koko alimatriisista $A(k:n,k:n)$ isoin luku ja vaihdetaan rivejä ja sarakkeita siten, että tämä luku tulee alkioksi $A(k,k)$.

Osittaistuenta (partial pivoting) tarkoittaa ainoastaan vaakarivien vaihtamista keskenään. Silloin siis etsitään suurin luku sarakkeesta $A(k:n,k)$ ja vaihdetaan se alkioksi $A(k,k)$. Tämä on ohjelmallisesti helpompi toteuttaa kuin täydellinen tuenta, koska osittaistuenta tarkoittaa vain yhtälöiden järjestyksen vaihtamista. Täydellinen tuenta merkitsee myös tuntemattomien järjestyksen muuttamista. Lisäksi etsintätyö on huomattavasti kevyempi osittaistuennassa. Yleensä riittää käyttää osittaistuentaa, joka onkin toteutettu useimmissa lineaaristen yhtälöryhmien ratkaisuohjelmistoissa.

Esimerkki 3.2.2 Tarkastellaan matriisia

$$\begin{pmatrix} 4 & 8 & -2 \\ 2 & 4 & 1 \\ 2 & 6 & -6 \end{pmatrix}.$$

Ensimmäisen sarakkeen eliminoinnin jälkeen matriisi muuttuu muotoon

$$\begin{pmatrix} 4 & 8 & -2 \\ 0 & 0 & 2 \\ 0 & 2 & -5 \end{pmatrix}.$$

Tällöin siis lävistäjällä oleva tukialkio $A(2,2) = 0$. Ongelmasta selvitään vaihtamalla tässä vaiheessa matriisin toinen ja kolmas rivi keskenään.

Eräille matriisiluokille voidaan taata, että Gaussin eliminointi on stabiilia ja tuentaa ei tarvita. Eräs tällainen luokka ovat sellaiset matriisit A, joiden transpoosi A^T on lävistäjävaltainen. Lävistäjävaltaisuudella tarkoitetaan sitä, että matriisin kaikilla riveillä lävistäjäalkion itseisarvo on suurempi kuin rivin muiden alkioiden itseisarvojen summa:

$$|a_{ii}| > \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}|, \quad i = 1, \ldots, n. \tag{3.1}$$

Toinen luokka, joille tuenta ei ole tarpeen, on myöhemmin esiteltävät symmetriset positiivisesti definiitit matriisit. Muille matriiseille on valittava osittaistuentaa käyttävä ratkaisumenetelmä.

3.2.1 LU-hajotelma

Monissa tehtävissä on tarpeen ratkaista useita peräkkäisiä lineaarisia yhtälöryhmiä, joilla on sama kerroinmatriisi mutta eri oikean puolen vekto-

ri. Gaussin eliminoinnista voidaan laatia versio, joka kohdistaa eliminaation vain matriisiin, mutta tallentaa myös eliminoinnissa käytetyt kertoimet r, jotka voidaan myöhemmin kohdentaa oikean puolen vektoriin. Tätä oikean puolen vektorista riippumatonta versiota kutsutaan LU-hajotelmaksi. LU-hajotelman laskeminen on täsmälleen yhtä raskasta kuin yllä esitetty Gaussin eliminointi. Sen sijaan yhtälöryhmän ratkaiseminen LU-hajotelmaa käyttäen on huomattavasti kevyempää kuin LU-hajotelman muodostaminen.

Matriisin LU-hajotelma tarkoittaa sellaisen yläkolmiomatriisin U ja alakolmiomatriisin L laskemista, että $A = LU$. Lisäksi vaaditaan, että L:n lävistäjäalkiot ovat kaikki ykkösiä. Käytännössä LU-hajotelma lasketaan Gaussin eliminoinnilla siten, että L:n alkiot ovat Gaussin eliminoinnissa laskettavat kertoimet r ja U:n alkiot ovat A:n yläkolmion alkiot eliminoinnin jälkeen. Tietokoneen muistissa LU-hajotelma voidaan laskea matriisin A päälle, eli ylimääräistä tilaa ei tarvita.

Esimerkki 3.2.3 Esimerkissä 3.2.1 esitetyn matriisin LU-hajotelma on

$$\begin{pmatrix} 2 & 4 & -2 \\ 4 & 7 & 1 \\ 1 & 3 & -3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 0.5 & -1 & 1 \end{pmatrix} \begin{pmatrix} 2 & 4 & -2 \\ 0 & -1 & 5 \\ 0 & 0 & 3 \end{pmatrix}.$$

Kun matriisin LU-hajotelma on laskettu, yhtälöryhmän ratkaisu pelkistyy ylä- ja alakolmioryhmän ratkaisuksi:

Algoritmi 3.2.2 (Yhtälöryhmän ratkaiseminen LU-hajotelmalla)

1: Ratkaise $\mathbf{z}$ yhtälöstä $L\mathbf{z} = \mathbf{b}$.

2: Ratkaise $\mathbf{x}$ yhtälöstä $U\mathbf{x} = \mathbf{z}$.

Edellistä vaihetta kutsutaan eteenpäinsijoitukseksi ja jälkimmäistä takaisinsijoitukseksi.

LU-hajotelman laskemiseen kuluu suurin piirtein $n^3/3$ kertolaskuoperaatiota ja yhtälöryhmän ratkaisemiseen hajotelmaa käyttäen noin n^2 kertolaskuoperaatiota.

3.2.2 Käänteismatriisi

Neliömatriisin A käänteismatriisia merkitään symbolilla A^{-1}. Se on neliömatriisi, jolle pätee $A^{-1}A = AA^{-1} = I$. Tässä I:llä tarkoitetaan saman kokoista yksikkömatriisia kuin A.

Käänteismatriisia käyttäen voidaan yhtälöryhmän $A\mathbf{x} = \mathbf{b}$ ratkaisu voidaan kirjoittaa muotoon $\mathbf{x} = A^{-1}\mathbf{b}$. Emme suosittele käänteismatriisin muodostamista eksplisiittisesti, koska käänteismatriisin laskeminen on työläämpää kuin vastaavan yhtälöryhmän ratkaiseminen tai kerroinmatriisin hajotelman

laskeminen. Numeerinen tarkkuus saattaa myös tulla ongelmaksi käänteismatriisia käytettäessä. Lisäksi, vaikka alkuperäinen matriisi on harva, käänteismatriisi on yleisessä tapauksessa tiheä matriisi eli harvuudesta saatava etu menetetään. Käänteismatriisin muodostaminen vie $\mathcal{O}(n^3)$ liukulukuoperaatiota.

Käänteismatriisin muodostaminen ei kannata edes silloin kun ratkaistavana on useita yhtälöryhmiä, joilla on sama kerroinmatriisi. Sen sijaan yhtälöryhmän kerroinmatriisista kannattaa laskea jokin hajotelma.

Huomautus 3.2.1 Älä muodosta käänteismatriisia eksplisiittisesti, vaan korvaa sen käyttö vastaavan lineaarisen yhtälöryhmän ratkaisemisella tai matriisin hajotelman laskemisella.

3.3 Yhtälöryhmän ratkaisemisen teoriaa

Tässä luvussa kerromme milloin lineaarisella yhtälöryhmällä on yksikäsitteinen ratkaisu. Lisäksi käsittelemme saatavan ratkaisun tarkkuutta.

Matriisi A on *säännöllinen*, jos lineaarisella yhtälöryhmällä $A\mathbf{x} = \mathbf{b}$ on yksikäsitteinen ratkaisu. Tämä toteutuu, jos A on neliömatriisi ja jokin seuraavista yhtäpitävistä ehdoista on voimassa:

- Matriisilla A on olemassa käänteismatriisi A^{-1}.
- A:n determinantti $|A| \neq 0$.
- A:n pystyvektorit (sarakkeet) ovat lineaarisesti riippumattomat.
- A:n vaakavektorit (rivit) ovat lineaarisesti riippumattomat.
- A:n *säännöllisyysaste* eli *rangi* on sama kuin matriisin dimensio (säännöllisyysaste on maksimaalinen lineaarisesti riippumattomien pysty- tai vaakavektorien lukumäärä).
- Yhtälöryhmän $A\mathbf{x} = \mathbf{0}$ ainoa ratkaisu on $\mathbf{x} = \mathbf{0}$.

Muussa tapauksessa matriisia kutsutaan singulaariseksi, jolloin yhtälöryhmällä on äärettömän monta ratkaisua tai sillä ei ole ratkaisua ollenkaan. Jos tehtäväsi kerroinmatriisi on singulaarinen, on alkuperäinen tehtävä luultavasti väärin määritelty tai kyseessä on ohjelmointivirhe.

Jos matriisissa A on enemmän rivejä kuin sarakkeita, yhtälöryhmää sanotaan *ylideterminoiduksi* ja sillä on ratkaisu pienimmän neliösumman mielessä. Vastaavasti jos matriisissa on enemmän sarakkeita kuin rivejä, yhtälöryhmää kutsutaan *alideterminoiduksi*.

Eräs tärkeä käsite yhtälöryhmien ratkaisussa on matriisin definiittisyys. Matriisi A on positiivisesti definiitti, jos pätee

$$\mathbf{x}^T A\mathbf{x} > 0 \text{ kaikilla } \mathbf{x} \neq \mathbf{0}. \tag{3.2}$$

Positiivisesti definiittien matriisien lävistäjäalkiot ovat kaikki positiivisia. Jos sekä matriisi A että sen transpoosi A^T ovat lävistäjävaltaisia ja matriisin A lävistäjäalkiot ovat positiivisia, matriisi on positiivisesti definiitti. Positiivisesti definiittien matriisien ei kuitenkaan tarvitse olla lävistäjävaltaisia.

Yhtälöryhmät, joissa kerroinmatriisi on positiivisesti definiitti, ovat stabiileja ratkaista. Kuten aikaisemmin todettiin, ratkaistaessa niitä Gaussin eliminoinnilla ei tarvita tuentaa.

3.4 Häiriöalttius

Matriisin *häiriöalttius* $\kappa(A)$ (condition number) kertoo, miten herkkä lineaarisen yhtälöryhmän ratkaisu on lähtötiedoissa esiintyville virheille. Häiriöalttius on tärkeä suure myös tarkasteltaessa iteratiivisten menetelmien suppenemista.

Näissä virhetarkasteluissa tarvitsee mitata matriisien ja vektorien ”pituuksia” tai ”kokoa”. Tähän tarkoitukseen käytetään erilaisia normeja. Yleisimmin käytetty vektorin pituuden mitta on ns. euklidinen normi, joka määritellään

$$\|x\|_2 = \sqrt{\sum_{j=1}^{m} |x_j|^2}. \tag{3.3}$$

Matriisin normi määritellään seuraavasti:

$$\|A\| = \max_{\|\mathbf{x}\|=1} \|A\mathbf{x}\|, \tag{3.4}$$

missä oikealla puolella olevat normit ovat vektorinormeja. Matriisin normi kuvaa miten pitkäksi matriisi voi kuvata ykkösen pituisen vektorin. Matriisinormin arvo riippuu käytetystä vektorinormista. Normeista kerrotaan lisää luvussa 4 sekä liitteessä A.

Yhtälöryhmien virheanalyysissa tarkastellaan yhtälöryhmän $A\mathbf{x} = \mathbf{b}$ asemesta yhtälöryhmää

$$(A + \epsilon F)\mathbf{x}(\epsilon) = \mathbf{b} + \epsilon\mathbf{f}, \tag{3.5}$$

jossa kerroinmatriisiin on lisätty häiriö ϵF ja oikean puolen vektoriin häiriö $\epsilon\mathbf{f}$. Nämä häiriöt voivat johtua esimerkiksi lähtötietojen epätarkkuuksista, epätarkasti lasketuista kerroinmatriisin alkioista tai tietokoneen pyöristysvirheistä. Virheanalyysissa ei tarvita häiriöiden tarkkaa lauseketta (siis matriisia F ja vektoria $\mathbf{f}$), vaan ainoastaan näiden häiriöiden koot eli normit.

Merkitään alkuperäisen yhtälöryhmän tarkkaa ratkaisua $\mathbf{x}(0) = \mathbf{x}$. Nyt voidaan osoittaa, että ratkaisun suhteellinen virhe on muotoa

$$\frac{\|\mathbf{x}(\epsilon) - \mathbf{x}\|}{\|\mathbf{x}\|} \leq \kappa(A)(\rho_A + \rho_{\mathbf{b}}) + \mathcal{O}(\epsilon^2). \tag{3.6}$$

Edellisessä kaavassa käytetty matriisin häiriöalttius $\kappa(A)$ määritellään

$$\kappa(A) = \|A\|\|A^{-1}\|, \tag{3.7}$$

ja häiriöitten suhteelliset koot ovat

$$\rho_A = \epsilon\frac{\|F\|}{\|A\|} \text{ ja } \rho_{\mathbf{b}} = \epsilon\frac{\|\mathbf{f}\|}{\|\mathbf{b}\|}. \tag{3.8}$$

Häiriöalttius on vahvistuskerroin, joka ilmaisee miten matriisissa ja oikean puolen vektorissa olevat pienet virheet vaikuttavat yhtälöryhmän ratkaisun tarkkuuteen. Parhaassa mahdollisessa tapauksessa häiriöalttius on yksi, jolloin lähtötietojen virheet eivät voimistu ratkaisussa. Toisaalta jos matriisin häiriöalttius on esimerkiksi 10^6, johtaa matriisin alkioiden laskemisessa tehty pieni suhteellinen virhe miljoonakertaiseen suhteellisen virheeseen yhtälöryhmän ratkaisussa. Mitä suurempi häiriöalttius on, sitä lähempänä matriisi on singulaarista matriisia. Mikäli yhtälöryhmän häiriöalttius on hyvin suuri, on vaikeaa saada luotettavaa ratkaisua yhtälöryhmälle millään algoritmilla.

Häiriöalttius riippuu käytetystä matriisinormista. Esimerkiksi 2-normin mielessä häiriöalttius on

$$\kappa(A) = \frac{\sigma_1(A)}{\sigma_n(A)}, \tag{3.9}$$

eli matriisin suurimman ja pienimmän singulaariarvon osamäärä. Matriisin A singulaariarvot ovat matriisin A^TA ominaisarvojen neliöjuuret. Liitteessä A annetaan lisätietoja singulaariarvoista. Jos matriisi on symmetrinen, häiriöalttius pelkistyy matriisin itseisarvoltaan suurimman ja pienimmän ominaisarvon osamääräksi.

Huomautus 3.4.1 Useat lineaaristen yhtälöryhmien ratkaisuohjelmistot osaavat arvioida häiriöalttiutta. Näitä arvioita kannattaakin aina käyttää hyväksi.

3.5 Muita hajotelmia

3.5.1 LDL^T-hajotelma

LU-hajotelma ei hyödynnä matriisin symmetrisyyttä. Symmetrisille matriiseille voidaan laskea LDL^T-hajotelma, missä D on lävistäjämatriisi ja L on alakolmiomatriisi, jonka lävistäjäalkiot ovat ykkösiä. LDL^T-hajotelman laskeminen on noin kaksi kertaa nopeampaa kuin LU-hajotelman laskeminen. Tätä hajotelmaa laskettaessa voi joutua käyttämään tuentaa. Symmetrisille positiivisesti definiiteille matriiseille hajotelman laskeminen on stabiilia ja tuentaa ei tarvita.

Symmetrisen matriisin definiittisyys voidaan päätellä laskemalla sille LDL^T-hajotelma. Mikäli lävistäjämatriisin D kaikki alkiot ovat positiivisia, alkuperäinen matriisi on positiivisesti definiitti.

3.5.2 Choleskyn hajotelma

Symmetriselle positiivisesti definiitille matriisille käytetään yleisesti Choleskyn hajotelmaa $A = LL^T$, missä L on alakolmiomatriisi. Tämä hajotelma on yksikäsitteinen ja stabiili laskea. Choleskyn hajotelma on noin kaksi kertaa nopeampi laskea kuin LU-hajotelma. Jos symmetriselle positiivisesti definiitille matriisille on laskettu LDL^T-hajotelma $A = MDM^T$, tämän matriisin Choleskyn tekijä on $L = M\text{diag}(\sqrt{d_1}, \sqrt{d_2}, \ldots, \sqrt{d_n})$, missä d_i ovat lävistäjämatriisin D lävistäjäalkiot.

Esimerkki 3.5.1 Seuraavassa on esimerkki Choleskyn hajotelmasta:

$$\begin{pmatrix} 9 & 3 & 9 \\ 3 & 5 & 7 \\ 9 & 7 & 49 \end{pmatrix} = \begin{pmatrix} 3 & 0 & 0 \\ 1 & 2 & 0 \\ 3 & 2 & 6 \end{pmatrix} \begin{pmatrix} 3 & 0 & 0 \\ 1 & 2 & 0 \\ 3 & 2 & 6 \end{pmatrix}^T.$$

Alla oleva algoritmi laskee Choleskyn hajotelman ja säilöö sen alkuperäisen matriisin päälle.

Algoritmi 3.5.1 (Choleskyn hajotelma)

```
for k = 1, 2, ..., n
    a(k,k) = sqrt(a(k,k))
    if k < n
        a(k+1:n, k) = a(k+1:n, k) / a(k,k)
        for j = k+1, ..., n
            a(j:n, j) = a(j:n, j) - a(j:n, k) * a(j, k)
        end
    end
end
```

3.5.3 QR-hajotelma

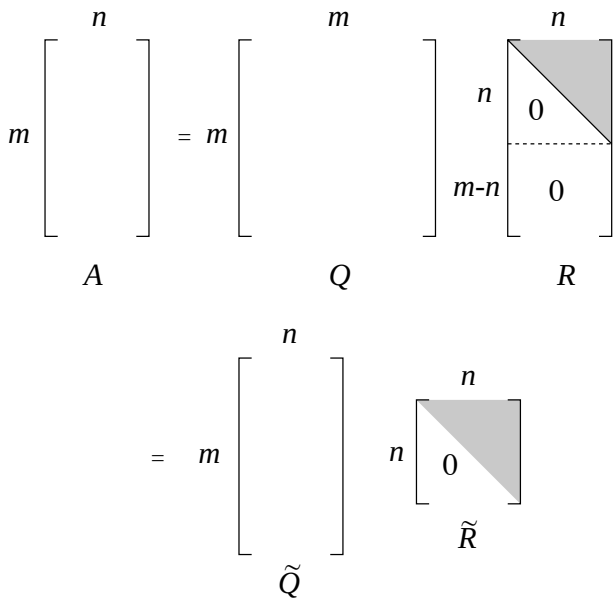
Matriisin QR-hajotelmaa tarvitaan mm. ratkaistaessa lähes singulaarisia yhtälöryhmiä. Sillä voidaan myös todeta, onko yhtälöryhmä singulaarinen. QR-hajotelmaa käytetään ylideterminoitujen yhtälöryhmien ratkaisussa pienimmän neliösumman mielessä. Lisäksi eräissä ominaisarvotehtävien ratkaisualgoritmeissa käytetään QR-hajotelmaa.

Matriisin QR-hajotelma on ortogonaalimatriisin Q ja yläkolmiomatriisin R tulo:

$$A = QR. \tag{3.10}$$

Matriisi on ortogonaalinen, jos sen käänteismatriisi on alkuperäisen matriisin transpoosi: $Q^{-1} = Q^T$. QR-hajotelmaa käyttäen yhtälöryhmän $A\mathbf{x} = \mathbf{b}$ pelkistyy yläkolmioyhtälöryhmän ratkaisuksi: $R\mathbf{x} = Q^T\mathbf{b}$. Matriisin singulaarisuus voidaan päätellä QR-hajotelmaa muodostettaessa: mikäli matriisin R lävistäjäalkioiksi tulee nollia, matriisi A on singulaarinen.

QR-hajotelmassa matriisin A ei tarvitse olla neliömatriisi. Jos matriisin A dimensio on $m \times n$ ($m \geq n$), tällöin matriisin Q dimensio on $m \times m$ ja matriisin R $m \times n$. Itse asiassa tällöin R koostuu $n \times n$ yläkolmiomatriisista ja sen alla olevasta nollalohkosta. Nollalohkosta johtuen alkuperäinen matriisi voidaan hajottaa myös muotoon $A = \tilde{Q}\tilde{R}$, jossa $m \times n$ -matriisi $\tilde{Q}$ koostuu matriisin Q n:stä ensimmäisestä sarakkeesta ja $\tilde{R}$ on $n \times n$ -kokoinen matriisissa R esiintyvä yläkolmiomatriisi. Kuva 3.1 havainnollistaa QR-hajotelmaa, kun A ei ole neliömatriisi.



Kuva 3.1: *QR-hajotelman muoto, kun A ei ole neliömatriisi.*

3.6 Ohjelmistoja

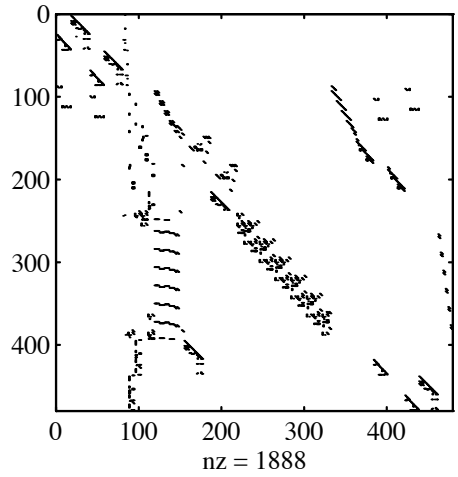
Perusohjelma tiheiden lineaaristen yhtälöryhmien ratkaisuun on LAPACK-aliohjelmakirjasto [ABB+92], jossa käytetään tehokkaita ja stabiileja algoritmejä. LAPACK sisältää rutiineja myös nauhamatriiseille, tridiagonaalimatriiseille sekä ylä- ja alakolmiomatriiseille. LAPACKin lähdekoodi on vapaasti saatavilla. LAPACK-kirjasto on siirrettävä ja tehokas, koska se on kirjoitettu käyttäen BLAS-rutiineja. BLAS-rutiinit ovat lineaarialgebran perusoperaatioita kuten vektorien sisätuloja tai matriisi-vektorikertolaskuja [Haa98]. LAPACKissa on yhtälöryhmän ratkaisijoiden lisäksi rutiineja ominaisarvotehtävien ratkaisuun ja erilaisten hajotelmien laskemiseen.

NAG- ja IMSL-kirjastoissa on vastaavat rutiinit kuin LAPACKissa [IMS, Num]. NAG-kirjaston F-alkuiset ja IMSL-kirjaston L-alkuiset rutiinit käsittelevät lineaarialgebraa. Eri tietokonevalmistajien matemaattiset aliohjelmakirjastot sisältävät lineaaristen yhtälöryhmien ratkaisijoita, joista suurin osa pohjau-

tuu LAPACK-kirjastoon. Aliohjelmakirjastojen käyttöä selvitetään myös viitteessä [Haa98].

3.7 Harvat matriisit

Monissa käytännön tehtävissä kerroinmatriisit ovat harvoja, jolloin matriisin kullakin rivillä saattaa olla esimerkiksi vain muutama nollasta poikkeava arvo. Kuvassa 3.2 on esimerkki harvasta matriisista.



Kuva 3.2: *Esimerkki harvasta matriisista. Matriisin nollasta poikkeavat alkiot on merkitty mustilla pisteillä. Matriisissa on 479 riviä ja saraketta sekä 1888 nollasta poikkeavaa alkiota eli keskimäärin alle neljä alkiota rivillä.*

Erikoistapauksia harvasta matriisista ovat tridiagonaalimatriisit, joissa vain matriisin lävistäjäalkiot ja lävistäjän viereiset alkiot ovat nollasta poikkeavia, sekä nauhamatriisit, joissa nollasta poikkeavat alkiot sijaitsevat tietyn nauhanleveyden sisällä lävistäjän ympäristössä. Seuraavissa kuvissa havainnollistetaan näitä matriisityyppejä:

- Tridiagonaalimatriisi: $a_{ij} = 0$, jos $|i - j| > 1$.

$$\begin{pmatrix} \times & \times & & & & \\ \times & \times & \times & & & \\ & \times & \times & \times & & \\ & & \times & \times & \times & \\ & & & \times & \times & \times \\ & & & & \times & \times \end{pmatrix}$$

- Nauhamatriisi.

$$\begin{pmatrix} \times & \times & \times & & & \\ \times & \times & \times & \times & & \\ \times & \times & \times & \times & \times & \\ & \times & \times & \times & \times & \times \\ & & \times & \times & \times & \times \\ & & & \times & \times & \times \end{pmatrix}$$

Gaussin eliminoinnilla voi ratkaista nopeasti ja tehokkaasti tridiagonaalimatriiseja sekä nauhamatriiseja. Mikäli tuentaa ei tarvita, ei alkuperäisen nauhan ulkopuolelle synny uusia alkioita eliminoinnin kuluessa.

Harvojen yhtälöryhmien ratkaisijat jaetaan suoriin ja iteratiivisiin menetelmiin. Suorat menetelmät ratkaisevat yhtälöryhmän Gaussin eliminoinnilla. Täten saatu vastaus on pyöristysvirheiden rajoissa tarkka ja tarvittavien laskutoimitusten määrä on etukäteen arvioitavissa. Iteratiiviset menetelmät sen sijaan lähtevät liikkeelle alkuarvauksesta ja tuottavat joukon iteraatteja, jotka toivottavasti suppenevat kohti ratkaisua. Iteraatio keskeytetään, kun virhe on hyväksyttävän pieni.

Tavallisten tiheiden matriisien ratkaisumenetelmien käyttäminen harvoille matriiseille on todellista tuhlausta, koska algoritmit laskevat suurimman osan aikaa nollilla. Nauhamatriisiratkaisijoiden käyttö on usein huomattavasti tehokkaampaa, mutta monissa tapauksissa nauhan sisälläkin valtaosa alkioista on nollia. Tehokkaat harvojen yhtälöryhmien ratkaisijat laskevat ainoastaan matriisin nollasta poikkeavilla alkioilla.

Suorat ja iteratiiviset menetelmät kilpailevat keskenään ratkaistaessa kaksiulotteisia differentiaaliyhtälöiden diskretoinneista syntyviä yhtälöryhmiä, mutta isoissa kolmiulotteisista differentiaaliyhtälöistä syntyville yhtälöryhmille iteratiiviset menetelmät ovat yleensä tehokkaampia. Kolmessa ulottuvuudessa kerroinmatriisit ovat hyvin harvoja, mutta ne voivat täyttyä merkittävästi käytettäessä suoria menetelmiä. Iteratiivisia menetelmiä voi käyttää myös täysien matriisien kanssa. Iteratiivisten menetelmien haittapuolena voi pitää niiden vaikeakäyttöisyyttä.

3.8 Esimerkkiprobleema

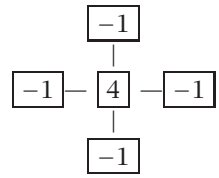
Tässä esitellään esimerkkiprobleema, jota käytetään jatkossa laskennallisena esimerkkinä. Kyseessä on kaksiulotteisen Poissonin yhtälön ratkaisu yksikköneliössä:

$$-\nabla^2 u = g(x,y), \quad x \in [0,1] \times [0,1]. \tag{3.11}$$

Diskretoidaan yhtälö viiden pisteen differenssikaavalla ja kiinnitetään reunaarvot. Differenssiapproksimaatiossa osittaisderivaatta $-(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2})$ korvataan differenssillä

$$\frac{4u_{i,j} - u_{i+1,j} - u_{i-1,j} - u_{i,j+1} - u_{i,j-1}}{h^2}, \tag{3.12}$$

mikä voidaan esittää myös kuvallisesti:



Sekä x- että y-suunnassa käytetään $N+2$ hilapistettä. Koska reuna-arvot tunnetaan, on kummassakin suunnassa N laskentapistettä. Hilavakio on $h = 1/(N+1)$. Numeroidaan tuntemattomat vaakarivi kerrallaan vasemmalta oikealle ja alhaalta ylös. Tällöin saadaan lineaarinen $N^2 \times N^2$ -yhtälöryhmä $A\mathbf{u} = \mathbf{b}$:

$$\begin{pmatrix} T & -I & & \\ -I & T & \ddots & \\ & \ddots & \ddots & -I \\ & & -I & T \end{pmatrix} \begin{pmatrix} \mathbf{u}^1 \\ \mathbf{u}^2 \\ \vdots \\ \mathbf{u}^N \end{pmatrix} = \begin{pmatrix} \mathbf{b}^1 \\ \mathbf{b}^2 \\ \vdots \\ \mathbf{b}^N \end{pmatrix}. \tag{3.13}$$

Tässä vektori $\mathbf{u}^k$ viittaa yhden hilarivin tuntemattomiin ja $N \times N$ -matriisi T on muotoa

$$\begin{pmatrix} 4 & -1 & & \\ -1 & 4 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 4 \end{pmatrix}. \tag{3.14}$$

3.9 Suorat menetelmät harvoille matriiseille

Ratkaistaessa harvaa lineaarista yhtälöryhmää suorilla menetelmillä käytetään Gaussin eliminointia mutta lasketaan ainoastaan matriisin nollasta poikkeavilla alkioilla. Gaussin eliminoinnin kuluessa syntyy uusia nollasta poikkeavia alkioita, joiden syntymistä pyritään minimoimaan tuntumattomien numerointia muuttamalla. Suorat menetelmät ovat helppokäyttöisiä ja varmoja. Suoria menetelmiä harvoille matriiseille on toteutettu monissa aliohjelmakirjastoissa. Tässä luvussa esitettyjä menetelmiä käsitellään tarkemmin viitteissä [GL81, Pis84].

Seuraavassa on esitetty matriisi ja sen LU-hajotelma eliminoinnin tuloksena. Tässä on siis asetettu matriisit L ja U päällekkäin ja matriisin L yksikkölävistäjä on jätetty ottamatta huomioon. Eliminoinnin kuluessa syntyneitä uusia alkioita on merkitty symbolilla o.

$$\begin{pmatrix} \times & & \times & & \times & \\ & \times & & \times & & \times \\ \times & & \times & & & \\ & & \times & \times & & \\ \times & \times & & & \times & \\ & & & \times & & \times \end{pmatrix} \rightarrow \begin{pmatrix} \times & & \times & & \times & \\ & \times & & \times & & \times \\ \times & & \times & & o & \\ & & \times & \times & o & \\ \times & \times & o & o & \times & o \\ & & & \times & o & \times \end{pmatrix}$$

Yhtälöiden ja tuntemattomien järjestyksellä on olennainen vaikutus uusien nollasta poikkeavien alkioiden lukumäärään. Tarkastellaan seuraavia matriisipareja. Ensimmäisestä syntyy täysi matriisi eliminoinnin jälkeen. Samasta matriisista saadaan yhtälöiden ja tuntemattomien uudelleennumeroinnilla toinen matriisi, johon ei eliminoinnin kuluessa synny lainkaan uusia alkioita:

$$\begin{pmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & & & & \\ \times & & \times & & & \\ \times & & & \times & & \\ \times & & & & \times & \\ \times & & & & & \times \end{pmatrix} \rightarrow \begin{pmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & o & o & o & o \\ \times & o & \times & o & o & o \\ \times & o & o & \times & o & o \\ \times & o & o & o & \times & o \\ \times & o & o & o & o & \times \end{pmatrix}$$

$$\begin{pmatrix} \times & & & & & \times \\ & \times & & & & \times \\ & & \times & & & \times \\ & & & \times & & \times \\ & & & & \times & \times \\ \times & \times & \times & \times & \times & \times \end{pmatrix} \rightarrow \begin{pmatrix} \times & & & & & \times \\ & \times & & & & \times \\ & & \times & & & \times \\ & & & \times & & \times \\ & & & & \times & \times \\ \times & \times & \times & \times & \times & \times \end{pmatrix}$$

Huomautus 3.9.1 Suorien menetelmien probleema harvoille matriiseille on siis lyhyesti seuraava: etsi tehokkain tuntemattomien ja yhtälöiden numerointi eli sellainen numerointi, joka minimoi uusien alkioiden syntymisen Gaussin eliminoinnin kuluessa. Samalla tulee tietenkin myös huomioida pyöristysvirheiden minimointi.

Tehokkaat algoritmit hyödyntävät graafiteoriaa ja osittain heuristiikkaa. Yleensä näissä algoritmeissa on kolme vaihetta: analyysivaihe, hajotelman laskeminen ja ratkaisuvaihe. Analyysivaiheessa ei tarvita tietoa matriisin alkioiden lukuarvoista, vaan ainoastaan alkioiden paikoista. Analyysivaiheen tietoa hyväksikäyttäen muodostetaan hajotelma, joka pyrkii minimoimaan uusien alkioiden syntymistä. Ratkaisuvaihe voidaan toistaa useille oikean puolen vektoreille.

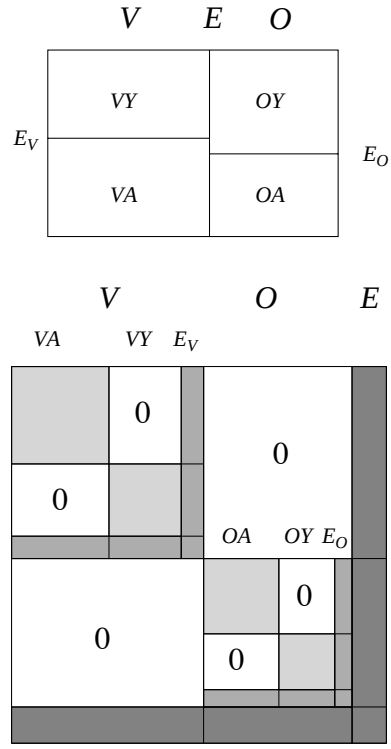
Ratkaisijat ovat erilaisia symmetrisille positiivisesti definiiteille matriiseille ja muille matriiseille. Positiivisesti definiiteille matriiseille voidaan osoittaa, että LU-hajotelman laskeminen on numeerisesti stabiilia. Riittää siis valita sellainen yhtälöiden ja tuntemattomien järjestys, joka minimoi uusien alkioiden syntymistä. Toisaalta symmetrian säilyttäminen vaatii, että yhtälöiden ja tuntemattomien, toisin sanoen rivien ja sarakkeiden järjestystä vaihdetaan samanaikaisesti.

3.9.1 Tuntemattomien numerointi

Seuraavaksi esittelemme eräitä algoritmeja tuntemattomien numeroimiseksi siten, että yhtälöryhmän ratkaiseminen olisi mahdollisimman tehokasta Gaussin eliminoinnilla. Nämä algoritmit löytyvät harvojen matriisien ratkaisijoista, joten käyttäjän ei yleensä tarvitse tuntea menetelmien toteutusta.

Eräs tehokas algoritmi tuntemattomien järjestämiselle on *minimiastemenetelmä* (minimum degree) jossa tukialkioksi valitaan eliminoimattomasta matriisista se lävistäjäalkio, jonka rivillä on vähiten nollasta poikkeavia alkioita. Graafiteoreettisesti matriisia esittävästä graafista valitaan solmu, jolla on pienin aste eli jolla on vähiten yhteyksiä graafissa.

Toinen tehokas numerointimenetelmä on *sisäkkäisjako* (nested dissection), jossa hilapisteet jaetaan erottajajoukoilla rekursiivisesti osiin, jotka eivät ole yhteyksissä keskenään. Jos erottajajoukko jakaa laskenta-alueen pystysuunnassa kahtia, ensin numeroidaan vasemmanpuoleiset hilapisteet, sitten oikeanpuoleiset hilapisteet ja viimeksi erottajajoukon tuntemattomat. Samaa algoritmia sovelletaan kumpaankin puoliskoon käyttäen vaakasuuntaisia erottajajoukkoja. Kuvassa 3.3 on esitetty sisäkkäisjaolla numeroidun kerroinmatriisin rakenne.



Kuva 3.3: *Sisäkkäisjaolla numeroidun kerroinmatriisin rakenne. Ylhäällä on esitetty laskenta-alue, joka on jaettu erottajajoukolla E osiin V (vasen) ja O (oikea). Vasen puolisko on jaettu vaakasuoralla erottajajoukolla E_V osiin VA (vasen alhaalla) ja VY (vasen ylhäällä). Alakuvassa on esitetty kerroinmatriisi ja eri alueiden tuntemattomien numerointijärjestys. Huomaa, että matriisissa on isoja nollalohkoja, koska eri puolilla erottajajoukkoa olevilla tuntemattomilla ei ole kytkentää keskenään.*

Niin sanotun käänteisen Cuthillin ja McKeen algoritmin (Reverse Cuthill-McKee algorithm, RCM) antama numerointi pyrkii minimoimaan harvan matriisin nauhanleveyden. Sen sijaan tämä numerointi ei ole optimaalinen Gaussin eliminoinnin suoritusajan kannalta. Eräs RCM-algoritmin käyttökohde on seuraava: numeroi tuntemattomat RCM-algoritmin mukaisesti ja käytä nau-

hamatriisiratkaisijaa syntyneen nauhamatriisin ratkaisemiseen. Nauhanleveyden sisällä voi olla paljon nolla-alkiota, mutta pienillä matriiseilla tämä menetelmä voi olla tehokkaampi kuin oikean harvamatriisiratkaisijan käyttö. Syy tähän on se, että nauhamatriisiratkaisija voidaan toteuttaa BLAS-rutiineja käyttäen huomattavasti tehokkaammin kuin harva ratkaisija. Toisaalta nauhamatriisiratkaisija voi vaatia huomattavasti enemmän muistia kuin harvamatriisiratkaisija.

Tarkastellaan esimerkkiprobleeman ratkaisemista suorilla menetelmillä. Kuvassa 3.4 on toteutettu tuntemattomien numerointi leksikografisesti (vasemmalta oikealle ja alhaalta ylös) ja kuvassa 3.5 tuntemattomat on numeroitu sisäkkäisjaolla.

Kuvan 3.6 vasemmassa ylänurkassa on esimerkkitehtävän (3.13) kerroinmatriisi. Tuntemattomat on numeroitu leksikografisesti. Oikeassa ylänurkassa on saman matriisin Choleskyn tekijä. Keskirivissä on permutoitu kerroinmatriisi ja sen Choleskyn tekijä, kun tuntemattomat on numeroitu minimiastealgoritmin mukaisesti. Alarivissä tuntemattomat on numeroitu käyttäen sisäkkäisjakoa. Kuten kuvista havaitaan, voitiin minimiastealgoritmin käytöllä vähentää Choleskyn tekijän nollasta poikkeavien alkioiden määrää 1009:sta 651:een. Sisäkkäisjako ei toiminut aivan yhtä hyvin, sillä Choleskyn tekijään jäi 785 alkiota.

Epäsymmetrisille matriiseille yhtälöiden ja tuntemattomien järjestäminen on monimutkaisempaa, koska uusien alkioiden syntymisen lisäksi pitää tarkkailla numeerista tarkkuutta, joten tukialkion valinta on kompromissi stabiilisuuden ja tilankäytön välillä.

3.9.2 Ohjelmistot ja matriisien tallennus

Suorien menetelmien tietokonetoteutuksissa matriisin tallentamiseen käytetyillä tietorakenteilla on keskeinen osa. Tietorakenteen pitää sallia uusien alkioiden joustava lisääminen. Matriisi voidaan ilmoittaa esimerkiksi kolmena taulukkona: `values(1:NZ)`, `cols(1:NZ)` ja `rowstart(1:N+1)`, missä `NZ` on nollasta poikkeavien alkioiden lukumäärä ja `N` on yhtälöryhmän koko. Taulukossa `values` on numeeriset arvot, taulukko `cols` kertoo, mihin sarakkeeseen kukin alkio kuuluu, ja taulukko `rowstart` kertoo, mistä kohtaa kukin rivi alkaa. Tässä siis oletetaan, että yhden rivin alkiot on säilötty peräkkäisiin alkioihin taulukoissa `values` ja `cols`. Taulukon `rowstart` viimeiseen alkion `rowstart(N+1)` arvoksi asetetaan `NZ+1`. Tässä tallennustavassa oleva matriisi voidaan tulostaa esimerkiksi seuraavalla silmukalla:

```
DO i = 1, N
  WRITE (*,*) 'Rivi ', i
  DO j = rowstart(i), rowstart(i+1)-1
    WRITE (*,*) 'Sarake ',cols(j)
    WRITE (*,*) 'Arvo ',values(j)
  END DO
END DO
```

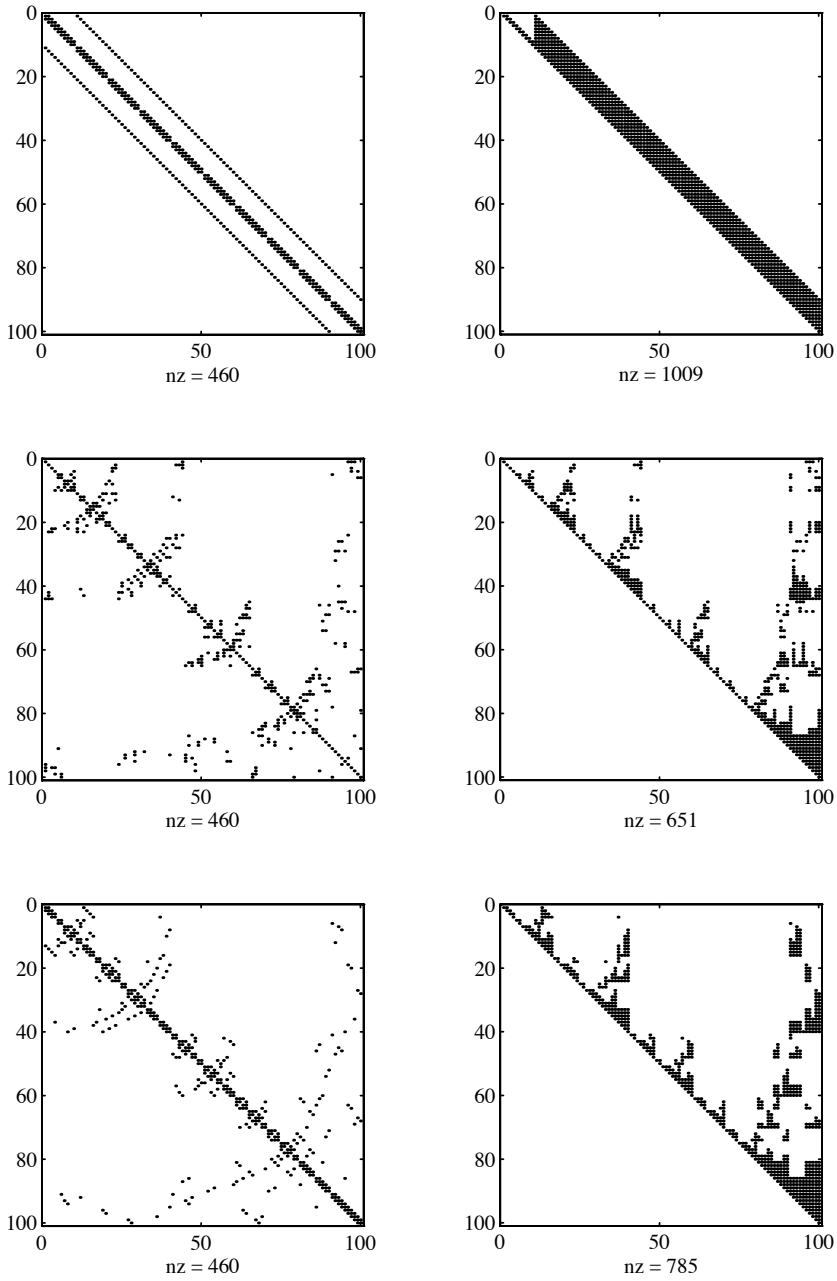
Edellä esitettyä tallennustapaa kutsutaan *pakatuksi rivitallennukseksi* (compressed row storage, CRS). Eräs toinen tallennustapa on *koordinaattitallen-*

91	92	93	94	95	96	97	98	99	100
81	82	83	84	85	86	87	88	89	90
71	72	73	74	75	76	77	78	79	80
61	62	63	64	65	66	67	68	69	70
51	52	53	54	55	56	57	58	59	60
41	42	43	44	45	46	47	48	49	50
31	32	33	34	35	36	37	38	39	40
21	22	23	24	25	26	27	28	29	30
11	12	13	14	15	16	17	18	19	20
1	2	3	4	5	6	7	8	9	10

Kuva 3.4: *Esimerkkitehtävän tuntemattomien leksikografinen numerointi riveittäin vasemmalta oikealle ja alhaalta ylös.*

26	32	30	28	100	68	66	85	78	76
25	31	29	27	99	67	65	84	77	75
33	34	35	36	98	69	70	83	79	80
18	24	22	20	97	64	62	82	74	72
17	23	21	19	96	63	61	81	73	71
37	38	39	40	95	86	87	88	89	90
4	16	10	8	94	46	44	60	54	52
3	15	9	7	93	45	43	59	53	51
2	14	11	12	92	47	48	58	55	56
1	13	5	6	91	41	42	57	49	50

Kuva 3.5: *Esimerkkitehtävän tuntemattomien numerointi sisäkkäisjaolla. Neliskulmainen laskenta-alue on ensin jaettu pystysuuntaisella erottajajoukolla kahteen osaan. Erottajajoukon alkiot on numeroitu viimeisenä (numerot 91–100), vasemmalla puolella käytetään numeroita 1–40 ja oikealla numeroita 41–90. Sekä vasen että oikea puoli on jaettu vastaavasti vaakasuuntaisilla erottajajoukolla kahteen osaan ja nämä puolestaan pienempiin osiin.*



Kuva 3.6: *Ylärivissä Poissonin yhtälön diskretoinnin kerroinmatriisi ja matriisin Choleskyn tekijä, kun tuntemattomat on numeroitu leksikografisesti. Matriisiin on keskirivissä sovellettu minimiastealgoritmin mukaista numerointia ja alarivissä sisäkkäisjakoa.*

nus, jossa kustakin matriisin alkiosta ilmoitetaan sen numeerinen arvo sekä rivi- ja sarakeindeksit.

Esimerkki 3.9.1 Tarkastellaan matriisin

$$A = \begin{pmatrix} 4 & 2 & 0 & 3 \\ 0 & 6 & 0 & 2 \\ 1 & 2 & 8 & 0 \\ 2 & 0 & 3 & 0 \end{pmatrix}$$

esitystapaa pakatussa rivitallennuksessa. Rivien ja tuntemattomien määrä on `N=4` ja nollasta poikkeavien alkioiden määrä on `NZ=10`. Taulukon alkiot pakataan taulukkoon `values` ja vastaavat sarakeindeksit taulukkoon `cols`:

$$\begin{aligned} \texttt{values} &= (4,2,3,\quad 6,2,\quad 1,2,8,\quad 2,3) \\ \texttt{cols} &= (1,2,4,\quad 2,4,\quad 1,2,3,\quad 1,3) \end{aligned}$$

Rivien aloitusindeksit pakataan taulukkoon `rows`:

$$\texttt{rows} = (1,4,6,9,11)$$

Huomaa, että `rows`-taulukon viimeisen alkion arvo on `NZ+1`.

Harwellin aliohjelmakirjasto (Harwell Subroutine Library) sisältää paljon moderneja toteutuksia suorista menetelmistä harvoille matriiseille. Hieman vanhempia ohjelmistoja ovat Sparspak ja Sparse. Lisäksi IMSL- ja NAG-aliohjelmakirjastot sekä monet tietokonevalmistajien aliohjelmakirjastot sisältävät harvojen matriisien ratkaisijoita. Lisätietoja saa esimerkiksi oppaasta [Haa98].

3.10 Iteratiiviset menetelmät

Iteratiiviset menetelmät tuottavat yhtälöryhmän ratkaisua läheneviä approksimaatioita. Käyttäjä ilmoittaa lopetusehdon, jonka toteuduttua iteraatio päättyy. Yleisesti ottaen iteratiiviset menetelmät ovat vaikeampia käyttää kuin suorat menetelmät, mutta ne voivat olla paljon tehokkaampia isoissa tehtävissä. Iteratiivisten menetelmien soveltaminen vaatii melko paljon tietämystä ja monet algoritmit ovat vasta tutkimusasteella.

Tässä kappaleessa esitellään aluksi muutamia yksinkertaisia perusmenetelmiä. Tämän jälkeen esitellään liittogradienttimenetelmä ja sen yleistykset epäsymmetrisille matriiseille. Lopuksi kerrotaan pohjustusmenetelmistä. Iteratiivisia menetelmiä käsitteleviä teoksia löytyy kirjallisuusluettelosta [FGN92, GvL96, Ort88, BBC+94, Saa96, Gre97].

3.10.1 Yksinkertaiset iteraatiomenetelmät

Seuraavassa esitellään yksinkertainen Jacobin iteraatiomenetelmä. Sitä ei sellaisenaan kannata käyttää yhtälöryhmien ratkaisemiseen, mutta se soveltuu esimerkiksi monihilamenetelmien perusiteraatioksi.

Jacobin menetelmä on yksinkertaisin ajateltavissa oleva iteraatiokaava, jossa ratkaistaan vuorollaan kukin muuttuja kaikkien muiden muuttujien suhteen. Komponenttimuodossa Jacobin menetelmän iteraatiokaava on

$$x_i^{k+1} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^k - \sum_{j=i+1}^{N} a_{ij} x_j^k \right), \quad i = 1, \ldots, N, a_{ii} \neq 0. \tag{3.15}$$

Tässä kaavassa x_j^k tarkoittaa vektorin komponenttia j iteraatiokierroksella k. Huomaa, että Jacobin menetelmässä uusi iteraatti $\mathbf{x}^{k+1}$ voidaan laskea kaikille komponenteille samanaikaisesti, mutta sen tallentamiseen tarvitaan apuvektori.

Gaussin ja Seidelin menetelmä on Jacobin menetelmän yleistys, jossa käytetään aina uusinta mahdollista informaatiota. Eli jos joitakin vektorin komponentteja on jo päivitetty, käytetään tätä informaatiota päivitettäessä muitakin komponentteja. SOR-iteraatio (successive over-relaxation) on Gaussin ja Seidelin menetelmän painotettu yleistys, johon liittyy vapaasti valittava relaksaatioparametri.

Edellä esitetyt iteraatiomenetelmät suppenevat käytännön tehtävissä paljon hitaammin kuin seuraavassa esitettävät tehokkaat iteraatiomenetelmät. SOR-menetelmä saattaa toimia tehokkaasti, jos sen relaksaatioparametri valitaan parhaalla mahdollisella tavalla.

3.10.2 Krylovin aliavaruus

Seuraavaksi esiteltävät menetelmät ovat niin sanottuja Krylovin aliavaruusmenetelmiä. Krylovin aliavaruudella tarkoitetaan sitä aliavaruutta, jonka virittävät matriisin A peräkkäisten potenssien ja annetun vektorin $\mathbf{r}$ tulot. Astetta k oleva Krylovin aliavaruus $K_k(\mathbf{r}, A)$ määritellään

$$K_k(\mathbf{r}, A) = \operatorname{span}(\mathbf{r}, A\mathbf{r}, \ldots, A^{k-1}\mathbf{r}), \tag{3.16}$$

missä span tarkoittaa annettujen vektorien kaikkien lineaarikombinaatioiden virittämää aliavaruutta.

Krylovin aliavaruusmenetelmät tuottavat iteraatteja, jotka ovat muotoa

$$\mathbf{x}^k \in \mathbf{x}^0 + K_k(\mathbf{r}^0, A), \tag{3.17}$$

missä $\mathbf{r}^0$ on alkuarvauksen virhe eli residuaali $\mathbf{r}^0 = \mathbf{b} - A\mathbf{x}^0$. Kierroksen k iteraatit ovat siten alkuarvauksen sekä k-asteisen Krylovin aliavaruuden vektoreiden lineaarikombinaatioita. Krylovin aliavaruusmenetelmille on tyypillistä, että matriisia A tarvitaan ainoastaan matriisi-vektorikertolaskujen $A\mathbf{x}$

laskemisessa. Täten nämä menetelmät eivät mitenkään muuta kerroinmatriisia. Matriisia ei itse asiassa tarvitse muodostaa eksplisiittisesti, ainoastaan sen ja annetun vektorin tulo pitää pystyä laskemaan.

Seuraavaksi esittelemme symmetrisille positiivisesti definiiteille matriiseille suunnitellun liittogradienttimenetelmän sekä eräitä epäsymmetrisille matriiseille tarkoitettuja Krylovin aliavaruusmenetelmiä.

3.10.3 Liittogradienttimenetelmä

Useat tehokkaat iteraatiomenetelmät on johdettu optimointitehtävästä: etsi minimi funktiolle

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{b}^T\mathbf{x} + \mathbf{c}, \tag{3.18}$$

missä A on symmetrinen ja positiivisesti definiitti matriisi. Yllä olevin oletuksin funktion minimikohta löytyy pisteessä, jossa gradientti on nolla, eli

$$\nabla f(\mathbf{x}) = A\mathbf{x} - \mathbf{b} = \mathbf{0}. \tag{3.19}$$

Täten minimoimalla funktiota $f(\mathbf{x})$ saadaan samalla ratkaistua yhtälöryhmä $A\mathbf{x} = \mathbf{b}$.

Yhteistä optimointiin perustuville yhtälöryhmän ratkaisijoille on se, että seuraava iteraatti on piste, joka minimoi kohdefunktion edellisestä iteraatista annettuun suuntaan:

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{p}^k, \tag{3.20}$$

missä α_k valitaan siten, että $f(\mathbf{x}^{k+1})$ minimoituu. Menetelmät poikkeavat toisistaan suunnan $\mathbf{p}^k$ valinnassa. Eräs tehokas optimointimenetelmä, jolla on saatu hyviä tuloksia lineaaristen yhtälöryhmien ratkaisussa, on *liittogradienttimenetelmä* (conjugate gradient). Siinä optimointisuunnat $\mathbf{p}^k$ valitaan siten, että ne ovat konjugoituja matriisin A suhteen eli

$$\langle \mathbf{p}^i, A\mathbf{p}^j \rangle = 0 \quad \text{kun} \quad i \neq j, \tag{3.21}$$

missä merkintä $\langle \mathbf{x}, \mathbf{y} \rangle$ tarkoittaa vektorien $\mathbf{x}$ ja $\mathbf{y}$ sisätuloa.

Seuraavassa on kuvaus liittogradienttimenetelmästä.

Algoritmi 3.10.1 (Liittogradienttimenetelmä)

$\mathbf{r}^0 = \mathbf{b} - A\mathbf{x}^0$
$\mathbf{p}^0 = \mathbf{r}^0$
$k = 0$
do
$\alpha_k = \dfrac{\langle \mathbf{r}^k, \mathbf{r}^k \rangle}{\langle \mathbf{p}^k, A\mathbf{p}^k \rangle}$
$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{p}^k$
$\mathbf{r}^{k+1} = \mathbf{r}^k - \alpha_k A\mathbf{p}^k$

$$\beta_k = \frac{\langle \mathbf{r}^{k+1}, \mathbf{r}^{k+1} \rangle}{\langle \mathbf{r}^k, \mathbf{r}^k \rangle}$$
$$\mathbf{p}^{k+1} = \mathbf{r}^{k+1} + \beta_k \mathbf{p}^k$$
$$k = k + 1$$
until $\|\mathbf{r}^{k+1}\| < \epsilon$

Tässä algoritmissa matriisiin A viitataan ainoastaan matriisi-vektorikertolaskussa $\mathbf{y} = A\mathbf{x}$. Muut operaatiot ovat kahden vektorin sisätulon laskeminen sekä vektoripäivitys $\mathbf{x} = \mathbf{x} + \alpha\mathbf{p}$. Algoritmin lopetusehdossa vaaditaan, että *residuaalin* $\mathbf{r}^{k+1} = \mathbf{b} - A\mathbf{x}^{k+1}$ normi on tarpeeksi pieni. Tässä voidaan käyttää myös muunlaisia lopetusehtoja.

Liittogradienttimenetelmän suppenemisen määrää matriisin ominaisarvojen jakauma eli spektri. Ylärajan suppenemisnopeudelle antaa matriisin häiriöalttius κ eli symmetrisille positiivisesti definiiteille matriiseille isoimman ja pienimmän ominaisarvon suhde. Liittogradienttimenetelmä suppenemistarkastetussa käytetään normia $\|\mathbf{x}\|_A = \sqrt{\langle \mathbf{x}, A\mathbf{x} \rangle}$. Olkoon $\mathbf{x}$ alkuperäisen yhtälöryhmän tarkka ratkaisu, jolloin

$$\|\mathbf{x} - \mathbf{x}^k\|_A \leq 2\|\mathbf{x} - \mathbf{x}^0\|_A \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^k . \tag{3.22}$$

Jos häiriöalttius $\kappa = 100$, on teoreettinen suppenemisvauhti 0.82^k, ja jos $\kappa = 1000000$, vauhti on 0.998^k. Iso häiriöalttius johtaa siten hyvin hitaaseen suppenemisvauhtiin, jolloin on pakko käyttää pohjustusmenetelmiä (kappale 3.10.6).

Liittogradienttimenetelmällä on seuraava ominaisuus: sen tuottama iteraatti $\mathbf{x}^k$ antaa pienimmän mahdollisimman virheen yllä määritetyllä A-normilla mitattuna, kun iteraatti on muotoa

$$\mathbf{x}^k \in \mathbf{x}^0 + K_k(\mathbf{r}^0, A). \tag{3.23}$$

Toisin sanoen liittogradienttimenetelmä on tämän normin mielessä optimaalinen Krylovin aliavaruusmenetelmä. Lisäksi liittogradienttimenetelmän iteraatit ovat helppoja laskea: sen laskemiseen tarvitaan ainoastaan edellinen iteraatti, eikä siten kaikkia aikaisempia iteraatteja tarvitse säilyttää.

Huomautus 3.10.1 Iteratiivisten menetelmien käytössä on tärkeää minimoida kerroinmatriisin häiriöalttiutta ja siten kiihdyttää iteraation suppenemista tehokkailla pohjustusmenetelmillä (katso kappaletta 3.10.6).

3.10.4 Epäsymmetriset iteraatiomenetelmät

Seuraavassa esitellään muutamia epäsymmetrisille matriiseille tarkoitettuja Krylovin aliavaruusmenetelmiä. Nämä menetelmät on joko suunniteltu siten, että ne antavat ratkaisulle pienimmän mahdollisen residuaalin kun ratkaisua haetaan tietystä Krylovin aliavaruudesta (minimointiominaisuus) tai menetelmän iteraatit voidaan laskea helposti käyttäen vain muutamaa aiempaa iteraattia hyväksi. Yleisille epäsymmetrisille matriiseille ei voida laatia

iteraatiomenetelmää, jolla olisi minimointiominaisuus ja joka olisi samalla helppo laskea.

Tämänhetkisen tietämyksen perusteella ei voida sanoa mikä on tiettyyn sovellukseen parhaiten soveltuva iteratiivinen menetelmä. Tehokkain algoritmi löytyy kokeilemalla esimerkiksi alla esitettyjä iteratiivisia menetelmiä GMRES, BiCG, CGS, Bi-CGSTAB, QMR ja TFQMR. Monissa sovelluksissa kuitenkin iteratiivisen menetelmän valintaa tärkeämpää on löytää tehtävään sopiva tehokas pohjustusmenetelmä.

Iteratiivinen menetelmä GMRES (Generalized Minimal Residual Algorithm) tuottaa iteraatteja $\mathbf{x}^k$, joilla on seuraava minimointiominaisuus:

$$\|\mathbf{b} - A\mathbf{x}^k\|_2 = \underset{\mathbf{x}\in\mathbf{x}^0+K_k(\mathbf{r}^0,A)}{\text{minimi}} \|\mathbf{b} - A\mathbf{x}\|_2 \tag{3.24}$$

GMRES-menetelmässä rakennetaan ortogonaalinen kanta Krylovin aliavaruudelle, jolloin minimointitehtävä (3.24) lasketaan kierroksella k ratkaisemalla $(k+1)\times k$ -kokoinen pienimmän neliösumman tehtävä.

GMRES tarvitsee kuitenkin seuraavan iteraatin laskemiseen kaikki aikaisemmin lasketut iteraatit, joten iteraation kuluessa laskenta hidastuu ja vie yhä enemmän muistia. GMRES voidaan myös katkaista siten, että aina m:n iteraatiokierroksen jälkeen menetelmä käynnistetään uudestaan käyttäen viimeisintä iteraattia seuraavan kierroksen alkuarvauksena. Tällöin menetelmää kutsutaan nimellä GMRES(m).

BiCG (bi-conjugate gradient method) on menetelmä, jossa ylläpidetään kahta bi-ortogonaalista vektorijoukkoa. Toinen lasketaan käyttäen normaaleja matriisi-vektorikertolaskuja, toisessa vektorijoukossa matriisi-vektorikertolaskuissa käytetään kerroinmatriisin transpoosia A^T. BiCG:n iteraatit voidaan laskea helposti, mutta niillä ei ole optimaalisuusominaisuuksia. BiCG:n iteraattien normit saattavat heilahdella, eli menetelmän suppeneminen ei ole monotonista.

Iteratiivinen menetelmä CGS (conjugate gradient squared) on sukua BiCG-menetelmälle, mutta siinä lasketaan kullakin iteraatiokierroksella kaksi matriisi-vektorikertolaskua. Myöskään CGS:n suppeneminen ei yleensä ole monotonista, vaan iteraattien normit voivat heitellä.

BiCG- ja CGS-menetelmistä on kehitetty stabiloitu versio Bi-CGSTAB (bi-conjugate gradient stabilized), jossa lasketaan kaksi matriisivektori-kertolaskua kerroinmatriisin kanssa. Tämän menetelmän suppeneminen on tasaisempaa ja yleensä yhtä nopeaa kuin BiCG- ja CGS-menetelmien.

QMR (Quasi-Minimal Residual Method) on myös menetelmä, jonka iteraatit voidaan laskea tehokkaasti käyttäen vain kahden edellisen kierroksen iteraatteja. QMR perustuu kappaleessa 9.5.3 (sivu 325) esitettävän Lanczosin menetelmän epäsymmetriseen versioon. QMR:ää voidaan pitää BiCG-menetelmän stabiloituna versiona. Myös QMR:ssä tarvitaan matriisi-vektorikertolaskut sekä kerroinmatriisin että sen transpoosin kanssa. QMR-algoritmista on olemassa versio TFQMR (transpose-free QMR), jonka kuluessa ei tarvitse laskea matriisi-vektorikertolaskuja matriisin transpoosin kanssa. Lisäksi kompleksisille symmetrisille matriiseille on olemassa oma erikoisversio QMR:stä.

3.10.5 Lopetusehto

Iteratiivisessa ratkaisijassa iteraatio lopetetaan, kun käyttäjän antama lopetusehto on täyttynyt tai iteraatiokierrosten lukumäärä on ylittänyt annetun maksimin. Hyvä lopetusehto kuvastaa jollakin tavalla ratkaisussa olevaa suhteellista virhettä. Lisäksi lopetusehdon määritelmän tulisi olla riippumaton yhtälöiden lukumäärästä kun ratkaistaan samasta fysikaalisesta tehtävästä johdettuja eri kokoisia lineaarisia yhtälöryhmiä.

Kaikki lopetusehdot perustuvat residuaalin mittaamiseen. Olkoon $\mathbf{x}^k$ iteraatti kierroksella k. Vastaava residuaali määritellään $\mathbf{r}^k = A\mathbf{x}^k - \mathbf{b}$. Pelkkää residuaalia ei suositella käytettäväksi lopetusehtona. Ensinnäkään se ei ota huomioon, mikä on oikean puolen vektorin suuruusluokka. Mikäli koko yhtälöryhmä kerrotaan jollakin luvulla, voidaan oikean puolen vektori ja kaikkien iteraattien residuaalit saada pieniksi, mutta tämä ei vielä kerro iteraattien tarkkuudesta mitään. Lisäksi kasvatettaessa muuttujien määrää residuaalin arvo usein kasvaa, vaikka suhteellinen tarkkuus pysyy samana.

Parempi lopetusehto on vaatia, että residuaalin normi jaettuna oikean puolen vektorin normilla on tarpeeksi pieni: $\|\mathbf{r}^k\|/\|\mathbf{b}\| < \epsilon$. Iteratiivisten menetelmien voi kuitenkin olla vaikea saavuttaa tätä lopetusehtoa, jos kerroinmatriisi on häiriöaltis. Monipuolinen lopetusehto on vaatia, että niin sanottu normin mukaan laskettu taaksepäinen virhe (normwise backward error) ϵ_b on tarpeeksi pieni:

$$\epsilon_b = \frac{\|\mathbf{r}^k\|}{\|A\|\|\mathbf{x}^k\| + \|\mathbf{b}\|} < \epsilon. \tag{3.25}$$

Tässä matriisin A normina voidaan käyttää esimerkiksi 1-normia: $\|A\|_1 = \max_k \sum_j |a_{jk}|$.

Taaksepäinen virhe on helppokäyttöisempi mittari kuin eteenpäin laskettu virhe, jossa mitataan iteratiivisen menetelmän sekä oikean (tuntemattoman) ratkaisun välistä erotusta. Voidaan osoittaa, että yllä olevalla kaavalla määritelty taaksepäinen virhe on pienin mahdollinen suhteellinen virhe (häiriö) kerroinmatriisissa ja oikean puolen vektorissa, jolle $\mathbf{x}^k$ on häirityn yhtälöryhmän tarkka ratkaisu. Taaksepäinen virhe mittaa kuitenkin tietyssä iteraatissa olevaa virhettä, ei kerroinmatriisissa mahdollisesti olevaa virhettä.

Kaikissa kerroinmatriiseissa on jonkin verran pyöristysvirheitä verrattuna matemaattisesti tarkkaan kerroinmatriisiin. Lisäksi kerroinmatriisia muodostettaessa saatetaan tehdä approksimaatioita, esimerkiksi käytetään epätarkkoja numeerisen integroinnin menetelmiä matriisin alkioiden muodostamiseen. Taaksepäinen virhe antaa mahdollisuuden verrata yhtälöryhmän ratkaisun virhettä kerroinmatriisin virheeseen.

Iteratiivisissa menetelmissä ei iteraatiota kannata jatkaa, mikäli iteraation taaksepäinen virhe on pienempi kuin kerroinmatriisissa oleva suhteellinen virhe, koska tällöin on saavutettu niin tarkka ratkaisu kuin kerroinmatriisia käyttäen voidaan saada. Tyypillisille tehtäville riittää vaatia, että taaksepäinen virhe on pienempi kuin 10^{-8}. Lopetusehtona ei kannata käyttää arvoa 10^{-16} pienempiä lukuja, koska tämän suuruinen suhteellinen virhe tehdään aina tietokoneella laskettaessa ja siten taaksepäinen virhe ei koskaan voi painuakaan tämän luvun alle.

Joissakin ohjelmistoissa iteraatio lopetetaan, kun alkuresiduaali on pienentynyt annetulla tekijällä. Tämä lopetusehto voi antaa hyvin epätarkkoja ratkaisuja, jos alkuarvaus tuottaa suuren alkuresiduaalin. Toisaalta jos alkuarvaus oli melko hyvä, saatetaan joutua iteroimaan liian monta iteraatiokierrosta. Tätä lopetusehtoa kannattaa käyttää asettamalla alkuarvaus nollaksi, jolloin residuaali jaetaan oikean puolen vektorin normilla.

Iteratiivisissa menetelmissä kannattaa myös tarkistaa, käytetäänkö lopetusehdossa oikeaa vai laskennan kuluessa päivitettyä residuaalia. Esimerkiksi liittogradienttimenetelmässä päivitetään joka iteraatiokierroksella residuaalia $\mathbf{r}^k$. Kun on iteroitu monta kertaa, tämä päivitetty residuaali voi tulla pienemmäksi kuin oikea residuaali $A\mathbf{x}^k - \mathbf{b}$. Tämän vuoksi iteratiivisissa menetelmissä kannattaa menetelmän supettua tarkistaa, että myös aito residuaali toteuttaa lopetusehdon.

3.10.6 Pohjustus

Krylovin aliavaruusmenetelmien suppenemista voidaan kiihdyttää *pohjustuksella* (preconditioning) eli kertomalla yhtälöryhmä sopivalla matriisilla. Pohjustimen valinta vaikuttaa ratkaisevasti iteratiivisten menetelmien suorituskykyyn, koska se voi pudottaa iteraatioiden lukumäärän murto-osaan alkuperäisestä iteraatiomäärästä. Hyvän pohjustimen valinta on vaikea tehtäväkohtainen ongelma. Erityisen vaikeaa on löytää hyvin rinnakkaistuva pohjustin, jota voidaan käyttää hajautetun muistin rinnakkaistietokoneissa.

Yhtälöryhmän $A\mathbf{x} = \mathbf{b}$ pohjustus tarkoittaa yhtälön kertomista pohjustusmatriisin M käänteismatriisilla:

$$M^{-1}A\mathbf{x} = M^{-1}\mathbf{b}. \tag{3.26}$$

Käytännössä työskennellään alkuperäisen kerroinmatriisin kanssa, mutta iteratiivisia menetelmiä muutetaan siten, että niiden kuluessa ratkaistaan yhtälöryhmiä, joissa pohjustusmatriisi M on kerroinmatriisina. Mikäli pohjustusmatriisi M muistuttaa jollakin tavalla alkuperäistä kerroinmatriisia, iteratiiviset menetelmät suppenevat nopeammin tälle pohjustetulle yhtälöryhmälle kuin alkuperäiselle yhtälölle. Joskus voidaan pohjustin jakaa kahteen osaan: kerroinmatriisi kerrotaan vasemmalta matriisilla M_1^{-1} ja oikealta matriisilla M_2^{-1}.

Pohjustamattomat Krylovin aliavaruusmenetelmät etsivät ratkaisua avaruudesta $\mathbf{x}^0 + K_k(\mathbf{r}^0, A)$. Pohjustetut menetelmät sen sijaan toimivat avaruudessa $\mathbf{x}^0 + K_k(M^{-1}\mathbf{r}^0, M^{-1}A)$. Tehokas pohjustusmatriisi on siten sellainen, joka jollakin tavalla jäljittelee matriisia A, mutta yhtälöryhmä, jossa se on kerroinmatriisina, on helpompi ratkaista kuin alkuperäinen yhtälöryhmä. Ääritapaukset ovat $M = A$, jolloin iteraatio suppenee yhdellä iteraatiolla, mutta yhtälöryhmä on sama kuin alkuperäinenkin, ja $M = I$, jolloin uusi yhtälöryhmä on helppo ratkaista, mutta pohjustuksella ei saavuteta mitään. Yleisesti vaihtoehtona on pieni määrä kalliita iteraatioita tai suuri määrä halpoja iteraatiota. Pohjustimen valinnalla pienennetään yhtälöryhmän häiriöalttiutta.

Seuraavassa esitetään pohjustettu liittogradienttimenetelmä. Huomaa, että liittogradienttialgoritmin pohjustusmatriisin M tulee olla symmetrinen.

Algoritmi 3.10.2 (Pohjustettu liittogradienttimenetelmä)

$\mathbf{r}^0 = \mathbf{b} - A\mathbf{x}^0$
$M\mathbf{z}^0 = \mathbf{r}^0$
$\mathbf{p}^0 = \mathbf{z}^0$
$k = 0$
do
$\alpha_k = \dfrac{\langle \mathbf{r}^k, \mathbf{z}^k \rangle}{\langle \mathbf{p}^k, A\mathbf{p}^k \rangle}$
$\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{p}^k$
$\mathbf{r}^{k+1} = \mathbf{r}^k - \alpha_k A\mathbf{p}^k$
$M\mathbf{z}^{k+1} = \mathbf{r}^{k+1}$ (ratkaise $\mathbf{z}^{k+1}$)
$\beta_k = \dfrac{\langle \mathbf{r}^{k+1}, \mathbf{z}^{k+1} \rangle}{\langle \mathbf{r}^k, \mathbf{z}^k \rangle}$
$\mathbf{p}^{k+1} = \mathbf{z}^{k+1} + \beta_k \mathbf{p}^k$
$k = k + 1$
until $\|\mathbf{r}^{k+1}\| < \epsilon$

Algoritmin kuluessa joudutaan joka iteraatiokierroksella ratkaisemaan yhtälöryhmä $M\mathbf{z} = \mathbf{r}$. Lisäksi joudutaan laskemaan matriisi-vektorikertolasku $A\mathbf{p}$.

3.10.7 Eri pohjustusmenetelmiä

Yksinkertainen mahdollinen pohjustus on lävistäjäpohjustus, jossa pohjustusmatriisiksi M on valittu kerroinmatriisin lävistäjäalkiot. Tällöin yhtälöryhmän $M\mathbf{z} = \mathbf{r}$ ratkaisemiseksi täytyy oikean puolen vektori ainoastaan jakaa näillä lävistäjäalkioilla.

Usein käytettyjä pohjustusmenetelmiä ovat kerroinmatriisin epätäydelliseen hajotelmaan perustuvat pohjustukset. *Epätäydellinen LU-hajotelma* (incomplete LU factorization, ILU) saadaan, kun LU-hajotelmaa laskettaessa säädellään uusien nollasta poikkeavien alkioiden syntymistä. Voidaan esimerkiksi vaatia, että LU-hajotelmaa laskettaessa ei huomioida uusia nollasta poikkeavia alkioita ollenkaan. Tämän pohjustusmenetelmän nimi on ILU(0). Pohjustusmenetelmä ILU(1) saadaan, kun hyväksytään alkuperäisen matriisin alkioista eliminoinnin kuluessa syntyneet uudet alkiot, muttei enää näistä syntyviä nollasta poikkeavia alkioita. Yleisesti määritellään menetelmä ILU(k), jossa hyväksytään vain alkuperäisistä alkioista k:lla askeleella syntyneet alkiot. Voidaan myös määritellä kynnysarvo, jota pienemmät eliminoinnissa syntyvät alkiot asetetaan nolliksi.

Epätäydellinen LU-hajotelma tuottaa pohjustusmatriisin $M = \tilde{L}\tilde{U}$, jossa $\tilde{L}$ ja $\tilde{U}$ ovat harvat ala- ja yläkolmiomatriisit. Yhtälöryhmän $M\mathbf{z} = \mathbf{r}$ ratkaisemiseksi täytyy ratkaista yksi yläkolmio- ja yksi alakolmioyhtälöryhmä. Tämä ratkaisu on yleensä paljon nopeampaa kuin epätäydellisen hajotelman muodostaminen. Mitä enemmän täyttymistä epätäydellisessä hajotelmassa sallitaan, sitä nopeammin iteratiiviset menetelmät suppenevat, mutta samalla pohjustimen muodostaminen ja soveltaminen muuttuvat raskaammiksi.

Toinen epätäydellistä LU-hajotelmaa lähellä oleva pohjustusmenetelmä on harvan approksimatiivisen käänteismatriisin laskeminen (sparse approximate inverse preconditioner). Jos alkuperäinen kerroinmatriisi on A, halutaan muodostaa harva matriisi H, jonka alkiot ovat lähellä A:n käänteismatriisin alkioita. Tässä menetelmässä voidaan etukäteen määrätä matriisin H nollasta poikkeavien alkioiden paikat tai iteratiivisesti hyväksyä vain tietyn kynnysarvon ylittävät H:n alkiot. Matriisin H alkioiden numeroarvot saadaan esimerkiksi seuraavasta minimointitehtävästä:

$$\underset{H}{\text{minimi}} \ \|I - AH\|_F^2, \tag{3.27}$$

missä $\|\cdot\|_F$ tarkoittaa matriisin Frobeniuksen normia eli alkioiden neliöiden summan neliöjuurta. Tämä minimointitehtävä voidaan hajottaa riippumattomiin pienimmän neliösumman tehtäviin, joissa ratkaistaan matriisin H yhden sarakkeen alkiot kerrallaan:

$$\underset{m_j}{\text{minimi}} \ \|e_j - Ah_j\|_2^2, \quad j = 1, \ldots, N, \tag{3.28}$$

missä h_j ja e_j ovat H:n ja yksikkömatriisi I:n j:nnet sarakkeet.

Harvan approksimatiivisen käänteismatriisin H käyttäminen pohjustimena on helppoa. Pohjustimen käyttö tarkoittaa normaalisti yhtälöryhmän $M\mathbf{z} = \mathbf{b}$ ratkaisua, mutta tässä tapauksessa pohjustus pelkistyy harvan matriisin ja vektorin kertolaskuksi: $\mathbf{z} = H\mathbf{b}$. Matriisin H muodostaminen sen sijaan voi olla raskasta, varsinkin jos H:ssa on paljon nollasta poikkeavia alkioita.

Eräs pohjustusmenetelmä, joka soveltuu hyvin rinnakkaislaskentaan on *aluehajotelmamenetelmä* (domain decomposition). Aluehajotelmamenetelmässä laskenta-alue jaetaan suurin piirtein yhtä suurin osiin. Iteraatiota pyritään kiihdyttämään ratkaisemalla kussakin osa-alueessa paikallinen osatehtävä ja yhdistämällä näiden ratkaisujen antama informaatio. Jos oletetaan osa-alueiden rajalla olevien muuttujien arvot tunnetuiksi (esimerkiksi edelliseltä iteraatiokierrokselta), voidaan osa-alueiden tehtävät ratkaista toisistaan riippumatta ja tämän jälkeen päivittää rajamuuttujien arvoa. Aluehajotelmamenetelmät poikkeavat toisistaan esimerkiksi siinä, menevätkö osa-alueet päällekkäin vaiko eivät. Lisäksi osatehtävien määritelmät ja reunaratkaisujen yhdistäminen vaihtelevat eri aluehajotelmamenetelmissä. Aluehajotelmista kerrotaan kirjassa [SBG96].

Kappaleessa 3.11 esitettävä nopea erikoisalgoritmi Poissonin yhtälölle ja kappaleessa 3.12 esiteltävä monihilamenetelmä soveltuvat myös pohjustimiksi. Esimerkiksi ratkaistaessa Poissonin yhtälöä epäsäännöllisessä alueessa voi pohjustimena käyttää Poissonin yhtälön nopeaa ratkaisijaa suorakulmaisessa alueessa.

3.10.8 Iteratiivisten menetelmien suppeneminen

Tarkastelemme ensin pohjustamattomien iteratiivisten menetelmien suppenemista. Sen jälkeen näytämme miten pohjustusmenetelmillä voidaan nopeuttaa suppenemista. Tässä esimerkissä ratkaistaan aikaisemmasta esimerkkitehtävästä muunneltu epäsymmetrinen tehtävä.

Esimerkki 3.10.1 Tarkastelemme yksikköneliössä $\bar{\Omega} = [0,1]^2$ määriteltyä differentiaaliyhtälöä

$$\begin{aligned} -\nabla^2 f(x,y) + y\mathbf{r}\cdot\nabla f(x,y) &= g(x,y), & (x,y) &\in \Omega, \\ f(x,y) &= 0, & (x,y) &\in \partial\Omega. \end{aligned}$$

Tässä on merkitty $\mathbf{r} = x\mathbf{i} + y\mathbf{j}$.

Haemme yhtälön ratkaisun tasavälisten differenssiapproksimaatioiden avulla. Laplace-operaattori $-\nabla^2 f(x,y)$ korvataan jo aiemmin esitellyllä 5 pisteen hilalla, ja 1. kertaluvun derivaattoihin sovelletaan symmetristä keskeisdifferenssiä, esimerkiksi x-suunnassa

$$\frac{\partial f(x,y)}{\partial x}|_{x_i,y_j} \approx \frac{f(x_i+h,y) - f(x_i-h,y)}{2h}.$$

Yhtälön vasemmalla puolella oleva termi $y\mathbf{r}\cdot\nabla f = y(x\frac{\partial f}{\partial x} + y\frac{\partial f}{\partial y})$ tekee kerroinmatriisista A epäsymmetrisen, eli enää ei ole yleisesti voimassa $a_{ij} = a_{ji}$ kaikilla i,j. Huomaa, että *rakenteellinen symmetria* kuitenkin säilyy edelleen: jos $a_{ij} = 0$, niin myös $a_{ji} = 0$.

Koska tarkoituksena on testata erilaisia ratkaisumenetelmiä, valitaan jokin yksinkertainen, reunaehdot toteuttava funktio $f(x,y)$, joka sijoitetaan yhtälön vasemmalle puolelle, ja katsotaan millainen lauseke funktiolle $g(x,y)$ saadaan aikaan. Eräs yksikköneliön reunoilla häviävä funktio on

$$f(x,y) = \sin(\pi x)\sin(\pi y),$$

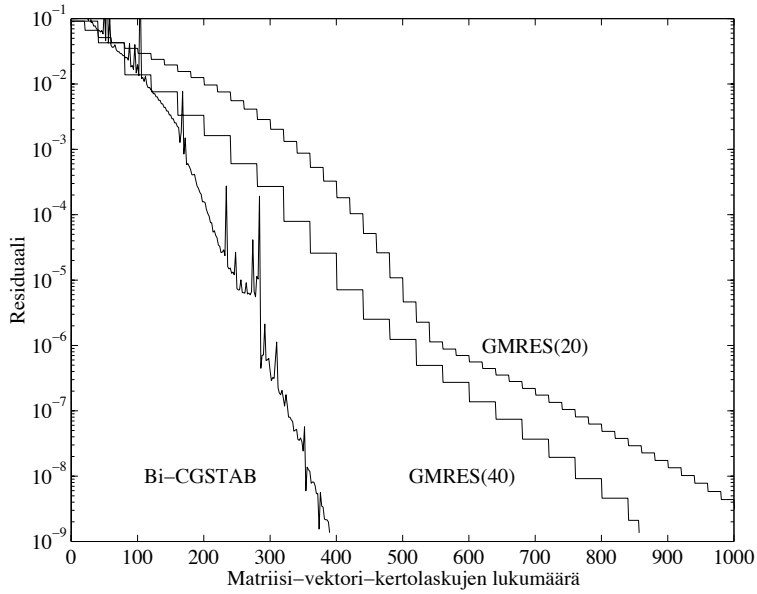
joka on ylläolevan yhtälön ratkaisu, jos

$$g(x,y) = -\pi^2\sin(\pi x)\sin(\pi y) + y\pi(x\cos(\pi x)\sin(\pi y) + y\sin(\pi x)\cos(\pi y)).$$

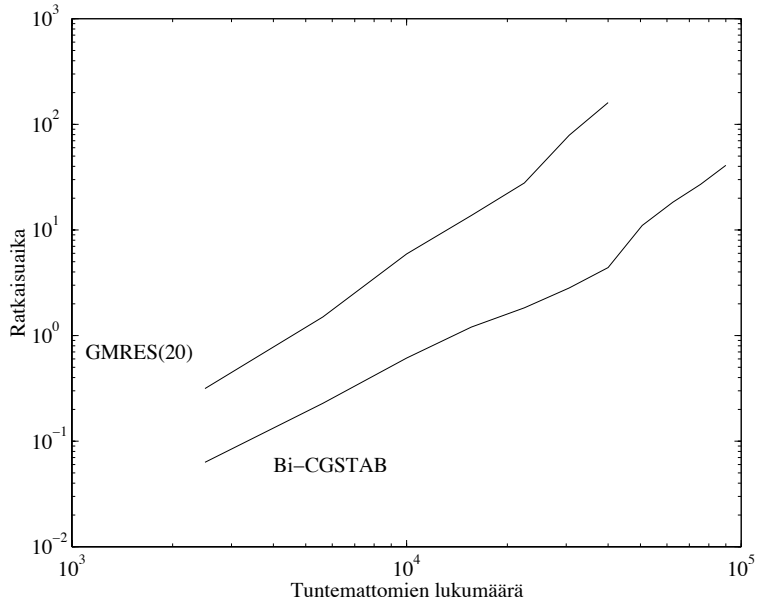
Tuntemattoman funktion $f(x,y)$ arvot hilapisteissä (x_i,y_j) ratkaistaan lineaarisesta yhtälöryhmästä, jossa kullakin rivillä on korkeintaan viisi nollasta poikkeavaa alkiota. Yhtälöryhmä on siis hyvin harva.

Esimerkiksi Harwellin aliohjelmakirjastossa on valmiiksi ohjelmoituna muutama iteratiivinen ratkaisija. Käsiteltävään tehtävään voidaan kokeilla vaikkapa rutiineja `MI04`, joka perustuu menetelmään GMRES(m) sekä `MI06`, joka perustuu Bi-CGSTAB-menetelmään. Kuvassa 3.7a on vertailtu näiden menetelmien suppenemisnopeutta (residuaalin koko) tehtyyn laskentatyöhön nähden (matriisi-vektorikertolaskujen lukumäärä). Differenssihilan kooksi on valittu 100×100, joten tuntemattomia on kaikkiaan 10000. GMRES(40) suppenee huomattavasti pienemmällä iteraatiomäärällä kuin GMRES(20), mutta koska yhteen iteraatioon kuluva aika on selvästi pidempi, kun uudelleenkäynnistys tehdään 40 iteraation välein, menetelmät ovat ajallisesti kutakuinkin yhtä tehokkaita.

Kuvassa 3.7b puolestaan nähdään ratkaisuajan riippuvuus tehtävän koosta. Yhtälöryhmä on katsottu ratkaistuksi, kun yhtälön oikean puolen normilla skaalattu residuaali alittaa arvon 10^{-8}. Bi-CGSTAB näyttää toimivan huomattavasti paremmin tämän tehtävän yhteydessä kuin GMRES(m). Huomaa, että kuvassa käytetään logaritmista asteikkoa.



a: *Eri iteraatiomenetelmien suppenemisnopeus.*



b: *Ratkaisuajan riippuvuus tehtävän koosta.*

Kuva 3.7: *Pohjustamattomien iteratiivisten ratkaisijoiden vertailua.*

Esimerkki 3.10.2 Katsomme, miten pohjustus vaikuttaa edellisessä esimerkissä käytettyihin iteratiivisiin ratkaisijoihin. Valitsemme Harwellin kirjastosta epätäydellisen LU-hajotelman (ILU) laskeva pohjustin `MI11`. Kun tuntemattomia on 10000, molemmat menetelmät suppenevat huomattavasti pienemmällä iteraatiokierrosten määrällä kuin ilman pohjustusta (kuva 3.8a). Iteraatioiden lukumäärä on alle neljäsosa pohjustamattomaan tehtävään verrattuna. Kukin iteraatiokierros on toisaalta raskaampi, koska joka kierroksella joudutaan ratkaisemaan pohjustusmatriisin määräämä yhtälöryhmä.

Kuvassa 3.8b on esitetty pohjustettujen iteraatiomenetelmien ratkaisuajat. Pohjustettu GMRES(m)-ratkaisija on noin kaksi kertaa nopeampi kuin pohjustamaton kaiken kokoisilla tehtävillä. Bi-CGSTAB-ratkaisija ei suurimpien yhtälöryhmien kohdalla näytä enää toimivan oleellisesti nopeammin kuin ilman ILU-pohjustusta. Pohjustimen laskeminen voi kestää yhtä kauan kuin kuvassa esitetty varsinaiseen ratkaisuun kuluva aika, mutta jos samaa kerroinmatriisia käytetään toistuvasti, pohjustus on ilman muuta kannattavaa.

3.10.9 Iteratiivisia ohjelmistoja

Texasin yliopistossa laadittu ITPACK-kirjasto ja sen osa NSPCG (NonSymmetric Preconditioned Conjugate Gradient) toteuttavat useita iteraatiomenetelmiä ja pohjustimia [Haa98]. Harwellin aliohjelmakirjastosta sekä IMSL- ja NAG-kirjastoista löytyy myös iteraatiomenetelmiä. Lisäksi on olemassa lukuisia aliohjelmakirjastoja, jotka tarjoavat rakennuspalikoita rinnakkaistettujen iteratiivisten menetelmien toteuttamiseen.

Iteratiivisia aliohjelmakirjastoja ei voi käyttää ”mustina laatikkoina”, vaan käyttäjän täytyy ainakin jonkin verran ymmärtää iteratiivisia menetelmiä. Parhaan pohjustimen ja iteratiivisen menetelmän valinta on vahvasti probleemakohtainen ja paras yhdistelmä löytynee parhaiten kokeilemalla.

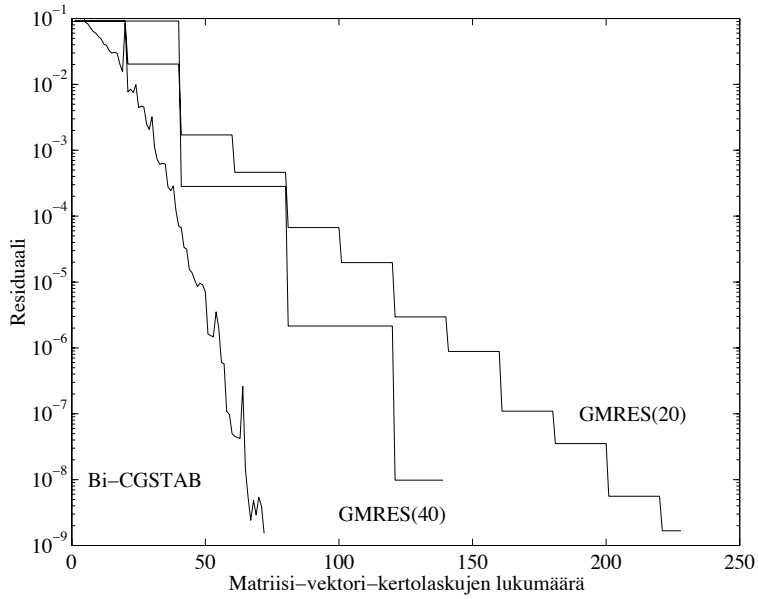
3.11 Poissonin yhtälön nopea ratkaisija

Kaksiulotteisen Poissonin yhtälön ratkaisuun on kehitetty nopeita erikoismenetelmiä, jotka käyttävät nopeaa Fourier’n muunnosta (FFT) [Swa77]. Tämä perustuu siihen, että esimerkkiprobleeman kerroinmatriisissa (3.13) esiintyvän $N \times N$ -matriisin

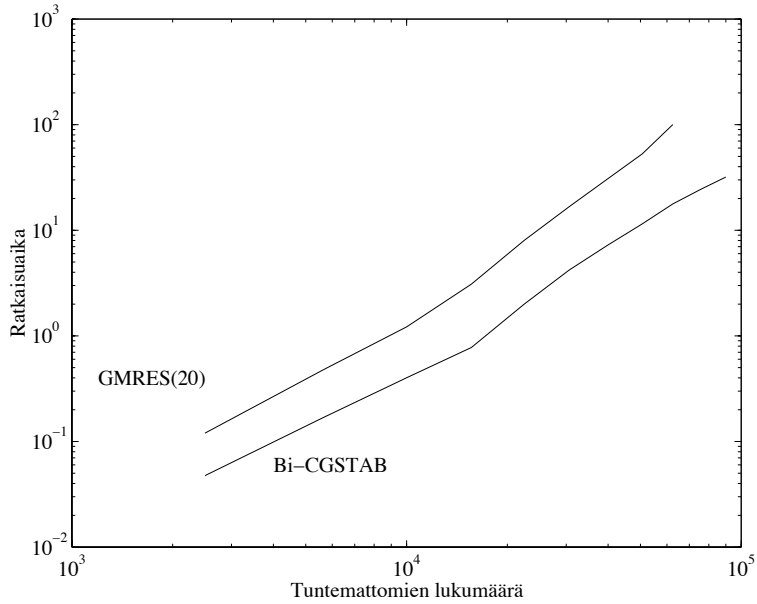
$$T = \begin{pmatrix} 4 & -1 & & \\ -1 & 4 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 4 \end{pmatrix} \tag{3.29}$$

ominaisarvot tunnetaan:

$$\lambda_i(T) = 4 - 2\cos\frac{i\pi}{N+1}, \quad i = 1, \ldots, N. \tag{3.30}$$



a: *Eri iteraatiomenetelmien suppenemisnopeus.*



b: *Ratkaisuajan riippuvuus tehtävän koosta.*

Kuva 3.8: *Pohjustettujen ratkaisijoiden vertailua.*

Vastaavat ominaisvektorit $\mathbf{q}^i$ ovat

$$q_k^i = \sqrt{\frac{2}{N+1}} \sin \frac{ki\pi}{N+1}, \quad i,k = 1,\ldots,N. \tag{3.31}$$

missä i määrää ominaisvektorin ja k sen komponentit.

Muodostetaan ominaisvektoreista ortogonaalinen matriisi

$$Q = (\mathbf{q}^1 \ \ \mathbf{q}^2 \ \ \ldots \ \ \mathbf{q}^N) \tag{3.32}$$

ja ominaisarvoista matriisi $\Lambda = \mathrm{diag}(\lambda^1 \ \ \lambda^2 \ \ \ldots \ \ \lambda^N)$, jolloin voidaan kirjoittaa $TQ = Q\Lambda$.

Alkuperäinen yhtälöryhmä on muotoa

$$\begin{pmatrix} T & -I & & \\ -I & T & \ddots & \\ & \ddots & \ddots & -I \\ & & -I & T \end{pmatrix} \begin{pmatrix} \mathbf{u}^1 \\ \mathbf{u}^2 \\ \vdots \\ \mathbf{u}^N \end{pmatrix} = \begin{pmatrix} \mathbf{b}^1 \\ \mathbf{b}^2 \\ \vdots \\ \mathbf{b}^N \end{pmatrix}. \tag{3.33}$$

Tässä vektori $\mathbf{u}^k$ viittaa yhden vaakarivin tuntemattomiin. Kerrotaan tämä yhtälöryhmä riveittäin matriisilla Q^T ja otetaan käyttöön uudet muuttujat $\mathbf{w}^k = Q^T\mathbf{u}^k$, jolloin $\mathbf{u}^k = Q\mathbf{w}^k$. Yhtälöryhmä muuttuu täten muotoon

$$\begin{pmatrix} \Lambda & -I & & \\ -I & \Lambda & \ddots & \\ & \ddots & \ddots & -I \\ & & -I & \Lambda \end{pmatrix} \begin{pmatrix} \mathbf{w}^1 \\ \mathbf{w}^2 \\ \vdots \\ \mathbf{w}^N \end{pmatrix} = \begin{pmatrix} Q^T\mathbf{b}^1 \\ Q^T\mathbf{b}^2 \\ \vdots \\ Q^T\mathbf{b}^N \end{pmatrix}. \tag{3.34}$$

Saadussa yhtälöryhmässä ei ole kytkentöjä yhden vaakarivin tuntemattomien $\mathbf{w}^k$ eri komponenttien välillä. Sen sijaan kukin vektorin $\mathbf{w}^k$ komponentti riippuu ylä- ja alapuolella olevien vektorien vastaavista komponenteista. Ominaisvektorien matriisilla Q kertominen on siten poistanut vaakasuuntaiset riippuvuudet tuntemattomien väliltä. Yhtälöryhmä (3.34) sisältää N riippumatonta N:n kokoista tridiagonaalista yhtälöryhmää, jotka voidaan ratkaista toisistaan riippumatta.

Koko algoritmi Poissonin yhtälön ratkaisemiseen on:

Algoritmi 3.11.1 (Nopea Poissonin yhtälön ratkaisija)

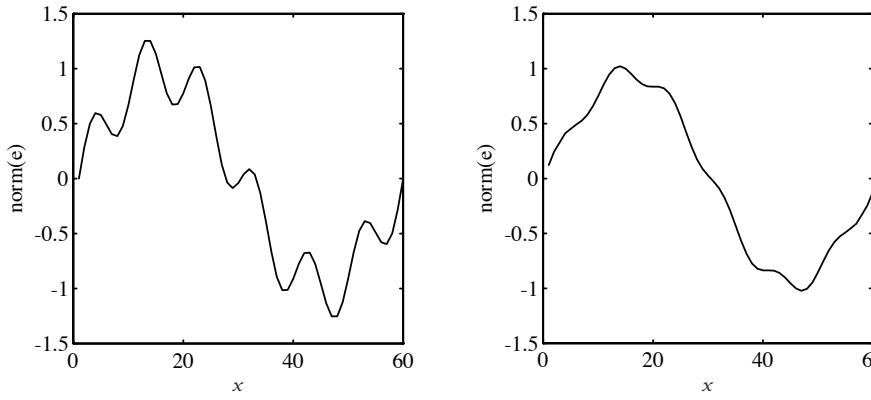
1: Laske $Q^T\mathbf{b}^k, \quad k = 1,\ldots,N.$
2: Ratkaise tridiagonaaliset yhtälöryhmät (3.34).
3: Laske $\mathbf{u}^k = Q\mathbf{w}^k, \quad k = 1,\ldots,N.$

Vaiheen 2 tridiagonaalisten yhtälöryhmien ratkaisuun menee $\mathcal{O}(N^2)$ liukulukuoperaatiota. Vaiheet 1 ja 3 ovat sinimuunnoksia, jotka voidaan laskea nopean Fourier'n muunnoksen avulla. Yhden N:n mittaisen Fourier'n muunnoksen laskemiseen kuluu $\mathcal{O}(N \log N)$ operaatiota, joten kohtien 1 ja 3 sekä samalla koko nopean Poissonin yhtälön laskenta-aika on muotoa $\mathcal{O}(N^2 \log N)$. Tämä on lähes optimaalinen, sillä yhtälöryhmässä on N^2 tuntematonta.

3.12 Monihilamenetelmät

Monihilamenetelmissä (multigrid) käytetään samanaikaisesti useaa laskentahilaa, joiden välillä siirretään informaatiota. Kussakin hilassa yhtälöryhmään tai residuaaliin sovelletaan jotakin yksinkertaista iteraatiomenetelmää. Monihilamenetelmät soveltuvat erityisesti elliptisten osittaisdifferentiaaliyhtälöiden diskretoinnista syntyvien yhtälöryhmien ratkaisemiseen. Monihilamenetelmät ovat yksinkertaisia toteuttaa differenssimenetelmille, mutta niitä voi tehokkaasti käyttää myös elementtimenetelmän yhteydessä.

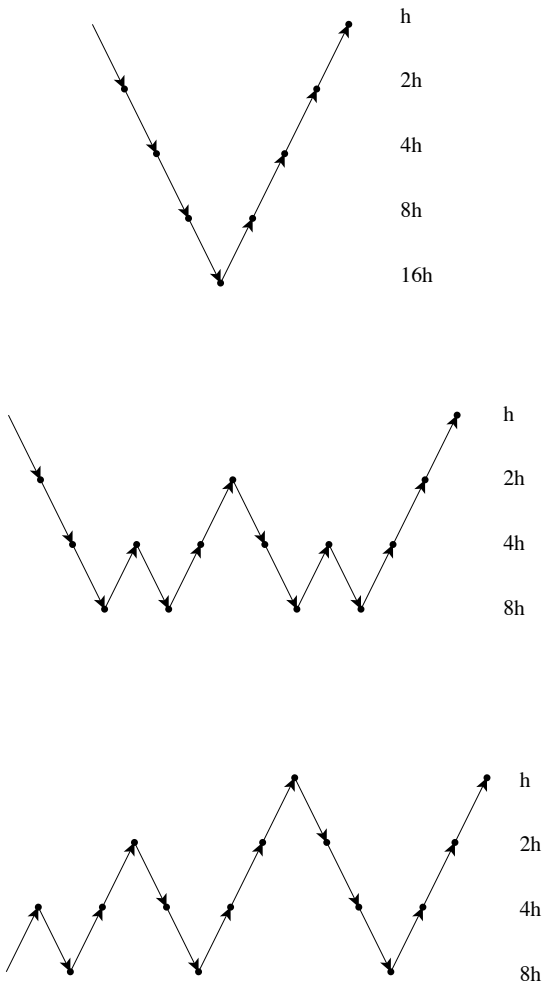
Iteraatiomenetelmän tuottaman iteraatin $\mathbf{x}^k$ ja tarkan ratkaisun $\mathbf{x}$ välinen virhe olkoon $\mathbf{e}^k = \mathbf{x}^k - \mathbf{x}$. Kun iteraatiovirhettä katsotaan alkuperäisessä laskentahilassa, virheessä on paikan suhteen nopeasti muuttuvia komponentteja ja hitaasti muuttuvia sileitä komponentteja. Monihilamenetelmät perustuvat siihen, että yksinkertaiset iteraatiomenetelmät, esimerkiksi Jacobin menetelmä, pienentävät parhaiten virheen nopeita komponentteja ja jättävät sileät komponentit lähes ennalleen. Iteraatio suppenee kaiken kaikkiaan hitaasti, mutta menetelmä on kuitenkin silottanut iteraatiovirhettä. Tämä on esitetty kuvassa 3.9 yksidimensioisessa tapauksessa.



Kuva 3.9: *Alkuperäinen iteraatiovirhe* $\mathbf{e}^0$ *ja virhe* $\mathbf{e}^6$ *kuuden Jacobin menetelmän iteraatiokierroksen jälkeen. Jacobin menetelmä silottaa nopeasti vaihtuvia virhekomponentteja, mutta hitaasti muuttuvat komponentit jäävät ennalleen.*

Monihilamenetelmissä iteraatiovirhe projisoidaan harvemmalle hilalle, jossa se näyttää nopeammin vaihtuvalta kuin aikaisemmalla hilalla. Täten uudessa hilassa voidaan Jacobin menetelmää käyttää vähentämään iteraatiovirheen nyt nopeilta näyttäviä komponentteja. Kun ratkaisu projisoidaan takaisin tiheämmälle hilalle, siellä laskettavan iteraatiovirheen sileitä komponentteja on saatu pienennettyä. Samaa menettelyä voidaan jatkaa harvemmalta hilalta rekursiivisesti vielä harvempaan hilaan, kunnes päädytään niin pieneen tehtävään, että se kannattaa ratkaista suoralla menetelmällä.

Monihilamenetelmät eroavat toisistaan käytettävän yksinkertaisen iteraatiomenetelmän eli silottajan suhteen sekä tavassa, jolla edetään eri hilojen välillä. Kuvassa 3.10 esitetään kolme tapaa siirtyä eri hilojen välillä. V-sykli



Kuva 3.10: *Eri monihilamenetelmiä. Ylinnä V-sykli, keskellä W-sykli ja alinna täysi monihilamenetelmä. Kuvissa h tarkoittaa hienointa hilaa ja karkein hila on $16h$ tai $8h$. Algoritmit etenevät vasemmalta oikealle.*

lähtee tiheimmältä hilalta, käy harvimmalla hilalla ja palaa takaisin tiheimmälle hilalle. W-syklissä siirrytään yksi hilataso harvempaan päin, tehdään siitä kaksi kertaa rekursiivisesti sama W-sykli ja palataan takaisin tiheämmälle hilalle. Tässä esitettävistä sykleistä tehokkain on täysi monihilamenetelmä, jossa lähdetään harvimmalta hilalta, siirrytään yksi askel tiheämmälle hilalle ja ajetaan V-sykli. Tätä toistetaan kunnes tullaan harvimmalle hilalle.

Huomautus 3.12.1 Yksinkertaisille probleemoille monihilamenetelmät saattavat olla optimaalisen tehokkaita.

Ratkaistaessa esimerkkiprobleemaa täysi monihilamenetelmä tekee vain luokkaa $\mathcal{O}(N^2)$ laskutoimitusta, missä N on hilapisteiden määrä yhdessä

suunnassa eli N^2 on yhtälöryhmän tuntemattomien määrä. Sovellettaessa monihilamenetelmiä epäsäännöllisiin geometrioihin, vaikeutena on muodostaa tiheästä verkosta harvemmat verkot sekä laatia interpolointioperaattorit verkkojen välille. Monihilamenetelmistä kerrotaan viitteissä [Bri87, HT82].

3.13 Yhteenveto

Lineaaristen yhtälöryhmien ratkaisumenetelmien valinta riippuu monesta tekijästä:

- Onko matriisi tiheä vai harva?
- Mikä on ratkaistavan tehtävän koko?
- Tarvitaanko mahdollisimman nopea ratkaisija vai luotettava yleiskäyttöinen ratkaisija?
- Pitääkö ratkaisijan toimia mustana laatikkona?
- Mikä on käyttäjän tietämys ratkaisumenetelmistä ja kyky ohjelmoida?
- Tarvitseeko ratkaisija rinnakkaistaa?

Lineaaristen yhtälöryhmien ratkaisumenetelmät jaetaan Gaussin eliminointiin perustuviin suoriin menetelmiin sekä matriisi-vektorikertolaskuihin perustuviin iteratiivisiin menetelmiin. Suorilla menetelmillä on seuraavat yleispiirteet:

- yleiskäyttöisiä ja luotettavia
- antavat aina tarkan ratkaisun
- valmista ratkaisijaa voi käyttää mustana laatikkona
- ei välttämättä optimaalisen tehokas isoille tehtäville
- hankala rinnakkaistaa.

Iteratiivisten menetelmien yleispiirteitä ovat:

- ratkaisumenetelmät toimivat vain tietyille tehtäväluokille
- soveltaminen vaatii paljon tietoa
- pohjustusmenetelmän valinta on tehtäväkohtaista
- ratkaisun tarkkuutta ja siten laskenta-aikaa voidaan säätää
- voivat olla optimaalisen tehokkaita
- helpompia rinnakkaistaa kun suorat menetelmät
- monet pohjustusmenetelmät ovat vasta tutkimusasteella ja ne voivat olla hankalia rinnakkaistaa.

Alla on annettu muutamia suosituksia menetelmän valinnasta.

- Kerroinmatriisi on tiheä:
  - Jos tehtävä mahtuu tietokoneen muistiin, käytä suoria menetelmiä ja LAPACK-kirjastoa.
  - Jos tehtävä on hyvin iso, sovella iteratiivisia menetelmiä ja yritä kehittää approksimatiivinen erikoismenetelmä matriisi-vektoritulojen laskemiseen.
- Kerroinmatriisi on harva:
  - Pienet tehtävät (alle kymmenentuhatta tuntematonta): numeroi tuntemattomat siten, että nauhanleveys minimoituu ja käytä nauhamatriisiratkaisijaa.
  - Keskikokoiset tehtävät: käytä harvoille matriiseille tehtyjä suoria ratkaisuohjelmistoja.
  - Isot tehtävät (satoja tuhansia tuntemattomia): käytä pohjustettuja iteratiivisia menetelmiä.

3.14 Lisätietoja

Yleisiä lineaaristen yhtälöryhmien ratkaisua ja lineaarialgebraa käsitteleviä teoksia ovat *Matrix Computations* [GvL96] ja *Matriisilasku ja lineaarialgebra* [Kiv84]. Suorista menetelmistä tiheille matriiseille ja tehokkaista lineaarialgebran toteutuksista kerrotaan teoksessa *LAPACK User's Guide* [ABB+92]. Lineaaristen yhtälöryhmien ratkaisuohjelmistoista kerrotaan myös kirjassa *Matemaattiset ohjelmistot* [Haa98].

Suoria menetelmiä harvoille matriiseille esitellään teoksissa *Sparse Matrix Technology* [Pis84] ja *Computer Solution of Large Sparse Positive Definite Systems* [GL81].

Iteratiivisia menetelmiä käsittelevät teokset *Templates for the Solution of Linear Systems* [BBC+94], *Iterative Methods for Sparse Linear Systems* [Saa96] ja *Iterative Methods for Sparse Linear Systems* [Gre97]. Lisäksi artikkeli *Iterative solution of linear systems* [FGN92] sisältää yhteenvedon iteratiivisista menetelmistä.

Aluehajotelmista kerrotaan kirjassa *Domain Decomposition.* [SBG96]. Hyvä johdatus monihilamenetelmiin on *A Multigrid Tutorial* [Bri87]. Nopeita Poissonin yhtälön ratkaisijoita käsitellään mm. artikkelissa [Swa77].

4 Approksimointi ja interpolointi

Approksimointi ja interpolointi ovat numeerisen analyysin perustekniikoita, joita sovelletaan hyvin monilla tieteen ja tekniikan aloilla. Approksimoinnissa on perimmiltään kyse informaation tiivistämisestä ja sen saattamisesta matemaattisesti helpommin käsiteltävään muotoon. Erilaisia approksimaatioita käytetään myös lähtökohtana mutkikkaampien numeeristen ongelmien ratkaisussa.

4.1 Esimerkkejä

Esimerkki 4.1.1 Haluamme laskea lausekkeen

$$F(x) = \int_x^\infty \frac{e^{-s^4}}{1+s^2} ds$$

arvon kaikilla $x \in \mathbb{R}$. Numeerisia integrointimenetelmiä esittelevässä luvussa 5 kerrotaan kuinka yksittäistä muuttujan x arvoa vastaavan funktion $F(x)$ arvon saa selville, mutta jos on odotettavissa, että funktio $F(x)$ on evaluoitava lukuisissa pisteissä hyvin tarkasti, kannattaa turvautua approksimaatioon.

Valitsemme ensin joukon solmupisteitä $\{x_i\}$, joissa selvitetään vastaavat funktion arvot $F(x_i)$. Lukuparien $\{(x_i, F(x_i)\}$ perusteella muodostamme sitten apufunktion $G(x)$, jonka arvot ovat helposti laskettavissa, ja joka on kaikkialla likimain yhtäsuuri alkuperäisen funktion $F(x)$ kanssa.

Menetelmään liittyy selvästi suuri joukko kysymyksiä. Kuinka pisteet $\{x_i\}$ kannattaa valita ja kuinka monta pistettä ylipäänsä tarvitaan? Miten apufunktio $G(x)$ muodostetaan? Miten sen hyvyyttä voidaan mitata? Toimiiko approksimaatio vain jollain äärellisellä välillä vai antaako se luotettavia tuloksia kaikilla muuttujan x arvoilla?

Esimerkki 4.1.2 Klassinen esimerkki approksimaatiosta on funktion sovittaminen mittausaineistoon. Funktion tyyppi määräytyy yleensä teoreettisin perustein. Analysointivaiheessa pitää löytää parametrit, jotka tuottavat havaintoja parhaiten kuvaavan funktion. Kuten edellisessäkin esimerkissä, jälleen on

pystyttävä määrittelemään, mitä tarkoitetaan ”parhaiten” sopivalla funktiolla ja miten siihen liittyvät parametrit saadaan helpoimmin laskettua.

Funktion sovitusta ei yleensä tehdä pelkästään kuvien piirtämistä varten, vaan funktioon liittyvät parametrit ovat usein kiinnostavia mittausaineistoa kuvaavia tunnuslukuja, joita voidaan käyttää vaikkapa turvallisia ohjearvoja määriteltäessä. Niinpä parametrien luotettavuus ja niihin mahdollisesti sisältyvät virheet tulisi tuntea ainakin karkealla tasolla.

Esimerkki 4.1.3 Teollisissa prosesseissa valmistettavan kappaleen muoto on saatava siirtymään suunnittelijan pöydältä lopputuotteeseen tarkasti mutta samalla mahdollisimman vaivattomasti. Koska nykyään käytetään tietokoneita puhtaan taiteellisen muotoilun lisäksi esimerkiksi lujuuslaskelmien tekoon (auto- ja lentokoneteollisuus) ja myös varsinaiseen materiaalien työstöön, on tärkeätä pystyä esittämään kaksi- ja kolmiulotteisten kappaleiden muodot yksinkertaisella ja yksiselitteisellä tavalla.

Eräs mahdollisuus on tietenkin valita kappaleen pinnalta riittävä määrä näytepisteitä ja siirrellä näiden koordinaatteja tuotekehitysprosessin kuluessa tietokoneesta toiseen. Pienemmällä tietomäärällä kuitenkin selvitään, jos kappaleen pinta pystytään esittämään paloittain määriteltyjen yksinkertaisten funktioiden avulla. Tällöin joudutaan tietysti miettimään kuinka suuria paikallisia virheitä voidaan hyväksyä. Suunnittelutyössä käytetyt ohjelmat ovat aiemmin pahimmillaan rajoittaneet muotoilua, koska niissä ei ole sovellettu tarpeeksi kehittyneitä algoritmeja.

4.2 Approksimointi

Approksimointi tarkoittaa jonkin suureen käyttäytymisen likimääräistä kuvaamista. Numeerisen analyysin sovellusten kannalta voidaan erottaa kaksi perustehtävää: *jatkuva* ja *diskreetti approksimointi*. Edellisessä tapauksessa approksimoitava funktio $f(x)$ on tunnettu, ja tarkoituksena on hakea jokin yksinkertaisempi funktio $\tilde{f}(x)$, joka on jossain mielessä lähellä alkuperäistä funktiota $f(x)$.

Diskreetissä approksimoinnissa funktiosta $f(x)$ tunnetään vain joukko arvoja $\{f_i\}$ joissain pisteissä $\{x_i\}$ esimerkiksi mittaustulosten perusteella. Nyt tehtävänä on muodostaa funktio $\tilde{f}(x)$, joka kulkee joko pisteiden $\{(x_i), (f_i)\}$ kautta (*interpolaatiotehtävä*) tai muuten kuvaa pistejoukon jakautumaa.

Molemmat approksimaatio-ongelmat yleistyvät tietenkin useamman muuttujan funktioille — ja muuttuvat samalla paljon vaikeammiksi (katso kappaletta 4.13).

Jos approksimoiva funktio $\tilde{f}(x)$ on saatu laskemalla yhteen jostain joukosta valittuja kantafunktioita $\{\phi_i(x)\}$, kyseessä on *lineaarinen approksimaatio:* $\tilde{f}(x) = \sum_{i=1}^{n} c_i \phi_i(x)$. Joissain tapauksissa on perusteltua käyttää myös epälineaarisia approksimaatioita, mutta näissä esiintyvien parametrien määrittäminen voi olla hankalaa.

4.3 Teoreettisia peruskäsitteitä

Approksimointitehtävän yhteydessä on aina pystyttävä vastaamaan kolmeen perusasiaan:

1. Mitä approksimoidaan ja miksi?
2. Millä approksimoidaan?
3. Miten mitataan approksimaation hyvyys?

Ensimmäiseen kysymykseen emme voi tietenkään antaa valmista vastausta. Soveltajan tulee ymmärtää mitä hän on tekemässä ja millaista on luonteeltaan se tieto, jota hän aikoo käyttää approksimoinnin pohjana. Tämä nimittäin vaikuttaa kahden jälkimmäisen kysymyksen vastaukseen. Jos on etukäteen selvää, että lähtöaineisto on luonteeltaan esimerkiksi jaksollista dataa, kannattaa approksimoinnin välineeksi harkita trigonometrisia kantafunktioita

$$\left\{1, \sin\frac{j\pi x}{l}, \cos\frac{j\pi x}{l}\right\}.$$

Jos ollaan arvioimassa jonkin rakenteen kestävyyttä, on luonnollista mitata approksimaation hyvyyttä arvioimalla suurinta mahdollista virhettä.

Jotta edellä lueteltuihin kysymyksiin olisi mahdollista kehittää systemaattisia vastauksia, pyrimme ensin muotoilemaan approksimointitehtävän matemaattisesti. Mikäli sellaiset analyysin käsitteet kuin *normi* ja *sisätulo* ovat päässeet unohtumaan, ne on syytä kerrata kirjan lopusta (liite A).

4.3.1 Approksimointi vektoriavaruudessa

Yleensä oletetaan, että funktiot ϕ_i, joiden avulla approksimointi tehdään, ovat jonkin normilla varustetun vektoriavaruuden V alkioita.

Yhteen vektoriavaruuteen on mahdollista luoda useampia normeja, jotka eivät välttämättä säilytä alkioiden suuruusjärjestystä. Siis vaikka normin $\|\cdot\|_1$ mielessä x on pienempi kuin y ($\|x\|_1 < \|y\|_1$), niin jokin toinen normi $\|\cdot\|_2$ saattaakin asettaa alkiot päinvastaiseen järjestykseen: $\|y\|_2 < \|x\|_2$.

Eri normeja vertaillaan usein käsitteillä *vahva ja heikko*. Vahva normi mittaa eroja tarkemmin kuin heikompi normi. Jos kaksi funktiota ovat vahvan normin mielessä lähellä toisiaan, ne ovat ovat lähekkäin myös heikommalla normilla mitattuna. Sen sijaan funktiot voivat olla kaukana toisistaan vahvassa normissa, ja siitä huolimatta heikko normi ei tee suurta eroa niiden välillä.

Esimerkki 4.3.1 Varustamme reaaliarvoisten, välillä $[0,1]$ määriteltyjen funktioiden muodostaman vektoriavaruuden L_2-normilla:

$$\|f\|_{L_2} = \left[\int_0^1 (f(x))^2\,dx\right]^{1/2}.$$

Jos jostain syystä haluammekin painottaa erityisesti funktion arvoa välin päätepisteessä 1, voimme valita normiksi esimerkiksi

$$\|f\|_{L_2\text{mod}} = \left[\int_0^1 (f(x))^2 \, dx + (f(1))^2 \right]^{1/2}.$$

Jos nyt vertaamme funktioita $f(x) = 1/2$ ja $g(x) = x^2$, huomaamme, että

$$\|f\|_{L_2} = 1/2 > 1/\sqrt{5} = \|g\|_{L_2},$$

mutta

$$\|f\|_{L_2\text{mod}} = 1/\sqrt{2} < \sqrt{6/5} = \|g\|_{L_2\text{mod}}.$$

Normit eivät siis säilytä suuruusjärjestystä.

Tutkimme vielä toisena esimerkkinä funktiojonoa (katso kuvaa 4.1)

$$f_n(x) = \begin{cases} 2nx, & 0 \le x \le 1/2n, \\ 2 - 2nx, & 1/2n \le 1/n, \\ 0, & \text{muulloin.} \end{cases}$$

Jos mittaamme jonon alkion $f_n(x)$ kokoa laskemalla funktion kuvaajan ja x-akselin väliin jäävän pinta-alan (L_1-normi), saamme tuloksen $\|f_n\|_{L_1} = 1/(2n)$. Siis kun n on hyvin suuri, L_1-normi ei havaitse suurtakaan eroa funktioiden f_n ja 0 välillä. Jos taas päätämme valita funktion kokoa kuvaavaksi suureeksi funktion suurimman poikkeaman arvosta 0 (L_∞-normi), kaikille jonon alkioille pätee $\|f_n\|_\infty = 1$, eivätkä jonon alkiot enää lähestykään 0-funktiota tämän normin mielessä.

Vaikka erilaisia normeja on periaatteessa loputtomasti, funktioiden approksimointitehtävissä tarvitaan yleensä vain muutamaa perusnormia. Funktioavaruuksien tapauksessa käytetään paljon L_p-normeja:

- L_1-normi (usein lyhyesti myös 1-normi):

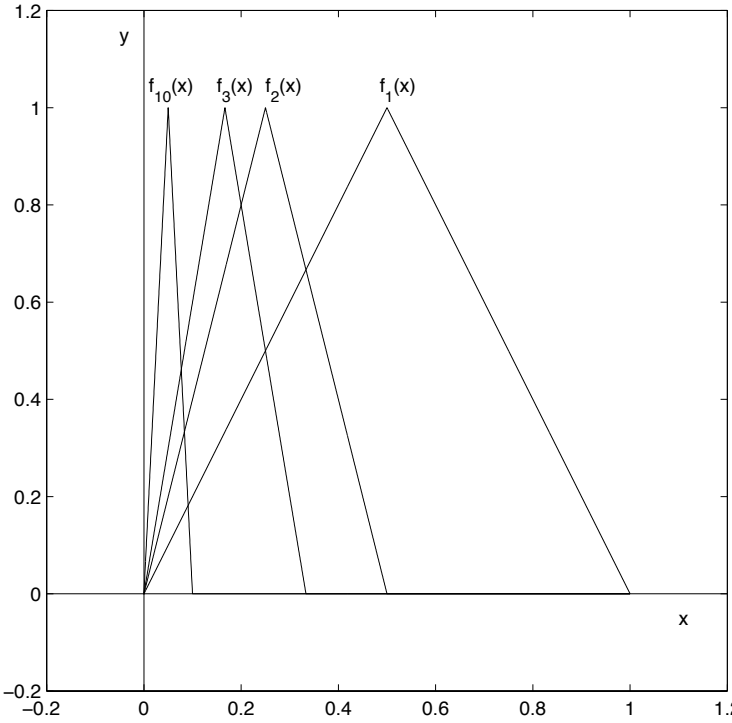
$$\|f\|_{L_1} = \|f\|_1 = \int_\Omega |f(\mathbf{x})| \, d\mathbf{x}.$$

- L_2-normi (tai vain 2-normi):

$$\|f\|_{L_2} = \|f\|_2 = \left[\int_\Omega (f(\mathbf{x}))^2 \, d\mathbf{x} \right]^{1/2}.$$

- L_∞-normi (∞-normi, Tšebyšev-normi):

$$\|f\|_{L_\infty} = \|f\|_\infty = \sup_{\mathbf{x}\in\Omega} |f(\mathbf{x})|.$$



Kuva 4.1: *Funktiojonon $f_n(x)$ alkioita.*

Näiden kanssa voidaan analogisesti määritellä l_p-normit (äärettömille) lukujonoille.

Normin valinta riippuu sovelluskohteesta. Ei ole olemassa mitään sääntöä, joka määräisi yleispätevän normin. Monimutkaisissa matemaattisissa probleemoissa ratkaisumenetelmä voi kuitenkin asettaa rajoituksia mahdollisille normivalinnoille. Tällöin on hyödyllistä tuntea eri normien välisiä epäyhtälöitä, jotta numeerisesti selvitettävissä olevien normien avulla pystytään arvioimaan myös sellaisia suureita, jotka eivät ole suoraan laskettavissa.

Mallintajalla tulee olla riittävän selvä käsitys mallissa esiintyvistä suureista ja mallinnuksen tavoitteesta, jotta hän pystyy muodostamaan kulloiseenkin tehtävään parhaiten sopivan normin. Normia ei suinkaan ole pakko valita yllä luetelluista kolmesta vaihtoehdosta, vaikka usein jokin niistä onkin riittävä. Itse kehitettyjen normien kohdalla on syytä tarkastaa, että normille asetetut vaatimukset toteutuvat. Muutenkin kannattaa pitää mielessä hyvä perussääntö ”mitä mittaa, sen havaitsee”.

Esimerkki 4.3.2 Fysikaalisista syistä voi olla tärkeätä, että approksimoitavan funktion derivaatta on koko alueessa Ω pieni. Jos jatkuvien funktioiden ava-

ruudessa $C(\Omega)$ yritetään käyttää normina lauseketta

$$\|f\| = \left[\int_\Omega (f'(x))^2\,dx\right]^{1/2},$$

pitää kyllä paikkansa, että nollafunktion normi on 0, mutta sama pätee kaikkien vakiofunktioiden suhteen. Siispä sivulla 388 annetuista ehdoista ensimmäinen toteutuu vain puoliksi. Koska muut normin ominaisuudet ovat voimassa, kyseessä on seminormi.

Edellä esitetty lauseke voidaan täydentää normiksi esimerkiksi lisäämällä tarkasteluun myös funktion $f(x)$ arvo:

$$\|f\| = \left[\int_\Omega (f(x))^2 + (f'(x))^2\,dx\right]^{1/2}.$$

Jos mitataan pelkästään derivaattaa, ei voida saada tietoa itse funktion arvoista — ellei tarkastelua ole rajoitettu aliavaruuteen, jossa funktioiden arvot on kiinnitetty esimerkiksi alueen reunalla.

Normin avulla voidaan vertailla kahden funktion samankaltaisuutta. Funktioiden f ja g välille voidaan määritellä etäisyys $d(f,g) = \|f-g\|$. Funktio f on lähellä funktiota g normin $\|\cdot\|$ mielessä, jos $d(f,g) \approx 0$. Näin on siis mahdollista päätellä myös jotain erilaisten approksimaatioiden paremmuudesta. Funktio $\tilde{f}_1$ on parempi approksimaatio funktiolle f kuin $\tilde{f}_2$, jos $d(f,\tilde{f}_1) < d(f,\tilde{f}_2)$ eli

$$\|f-\tilde{f}_1\| < \|f-\tilde{f}_2\|.$$

Normitetussa vektoriavaruudessa V voimme nyt muotoilla approksimointitehtävän yleisellä tasolla. Olkoon annettu alkio $\mathbf{x} \in V$, jota pitää approksimoida, ja (yleensä äärellisulotteinen) aliavaruus $V_1 \subset V$, jonka alkioiden joukosta parasta mahdollista approksimaatiota etsitään. Tehtävänä on siis hakea $\tilde{\mathbf{x}} \in V_1$, jolle pätee

$$\|\mathbf{x}-\tilde{\mathbf{x}}\| \le \|\mathbf{x}-\mathbf{y}\| \quad \text{kaikilla} \quad \mathbf{y} \in V_1.$$

Yleisessä tapauksessa ratkaisun olemassaolo vaatii, että joukko V_1 on avaruuden V kompakti osajoukko. Yksikäsitteisyyden edellytyksenä puolestaan on, että joukko V_1 on aidosti konveksi, eli $\|\lambda\mathbf{x}+(1-\lambda)\mathbf{y}\| < \lambda\|\mathbf{x}\|+(1-\lambda)\|\mathbf{y}\|$ kaikilla $\mathbf{x}, \mathbf{y} \in V_1$.

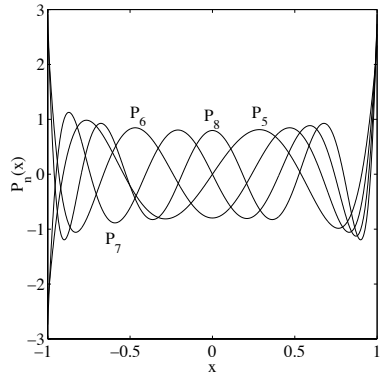
Käytännössä approksimaatio $\tilde{\mathbf{x}}$ joudutaan esittämään lineaarikombinaationa aliavaruuden V_1 kantavektoreista $\{\mathbf{v}_i\}$:

$$\tilde{\mathbf{x}} = \sum_i c_i \mathbf{v}_i.$$

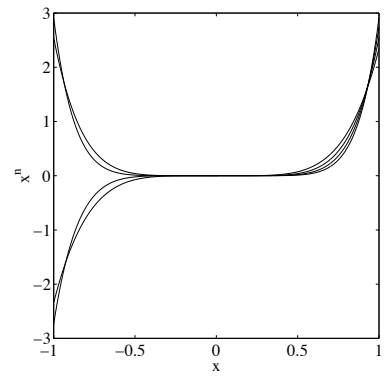
Tehtäväksi jää määrittää optimaaliset kertoimet c_i. Kyseessä on siis lineaarinen approksimointi. Käsiteellä ”lineaarinen approksimointi” tarkoitetaan joskus myös approksimointia 1. asteen polynomilla, mutta yleensä asiayhteydestä selviää kumpi tulkinta on kyseessä.

Jos rajoitutaan tarkastelemaan lineaarista approksimointia äärellisulotteisissa aliavaruuksissa, voidaan ainakin yhden ratkaisun olemassaolo aina taata. Yksikäsitteisyyteen riittää tällöin, että normi toteuttaa seuraavan ehdon: jos $\|\mathbf{x}+\mathbf{y}\| = \|\mathbf{x}\| + \|\mathbf{y}\|$, niin $\mathbf{x} = \lambda\mathbf{y}$. Siis kolmioepäyhtälössä yhtäsuuruus on voimassa vain, jos alkiot ovat lineaarisesti riippuvia. Tämä ominaisuus on mm. kaikilla sisätulon avulla määritellyillä normeilla.

Numeeristen stabiilisuusominaisuuksien kannalta ei ole yhdentekevää miten kantavektorit $\{\mathbf{v}_i\}$ valitaan. Jos approksimaatiota haetaan sisätulolla varustetussa avaruudessa, kannattaa valita ortonormaali kanta.



a: *Legendren polynomit* $\tilde{P}_5$, $\tilde{P}_6$, $\tilde{P}_7$ *ja* $\tilde{P}_8$.



b: *Polynomit* $\sqrt{n+1}x^n$, $n = 5, 6, 7, 8$.

Kuva 4.2: *Kahden eri kannan polynomeja.*

Esimerkki 4.3.3 Tarkastellaan välillä $[-1, 1]$ määriteltyjen polynomien $p(x)$ avaruutta. Kantafunktioiksi olisi helppo valita suoraviivaisesti monomit x^n. Nämä eivät kuitenkaan muodosta ortonormaalia systeemiä sisätulon (A.22) suhteen, sillä

$$\int_{-1}^{1} x^n x^m \, dx = \begin{cases} 0, & \text{jos m+n on pariton,} \\ 2/(m+n+1), & \text{jos m+n on parillinen.} \end{cases}$$

Ortonormaali kanta on muodostettavissa Gramin ja Schmidtin menetelmällä (sivu 391), jolloin päädytään normalisoituihin *Legendren polynomeihin* $\tilde{P}_0(x) = 1/\sqrt{2}$, $\tilde{P}_1(x) = \sqrt{3/2}x$, $\tilde{P}_2(x) = \sqrt{5/8}(3x^2-1)$, ...

Kuvassa 4.2a on normalisoitujen Legendren polynomien $\tilde{P}_5$, $\tilde{P}_6$, $\tilde{P}_7$ ja $\tilde{P}_8$ kuvaajat. Jos niitä verrataan alkeellisen kannan x^n vastaaviin funktioihin (kuva 4.2b), on helppo ymmärtää miksi jälkimmäinen kanta sopii huonosti äärellisellä laskentatarkkuudella suoritettaviin approksimaatioihin: yksinkertaiset monomit saavat hyvin pieniä arvoja merkittävällä osalla väliä $[-1, 1]$, ja lähellä päätyä $x = -1$ niiden käyttäytyminen riippuu oleellisesti eksponentin parillisuudesta tai parittomuudesta.

Historiallisista syistä Legendren polynomit P_n ovat yleensä vain ortogonaalisia, mutta eivät ortonormaaleja. Tässä esimerkissä molemmat kannat on normali-

soitu siten, että funktioiden normiksi tulee 1. Tästä syystä olemme käyttäneet merkintää $\tilde{P}_n$.

Käyttämällä erilaisia painofunktioita $w(x)$ ja muita laajennetun reaaliakselin välejä $[a,b]$ päädytään toisiin ortogonaalipolynomeihin. Taulukossa 4.1 on lueteltu tavallisimmat ortogonaalipolynomiperheet.

Taulukko 4.1: *Tavallisimmat ortogonaalipolynomiperheet.*

Nimi	$[a,b]$	$w(x)$	*Normituskerroin*
Legendren polynomit	$[-1,1]$	1	$\sqrt{n+1/2}$
Tšebyševin polynomit	$[-1,1]$	$1/\sqrt{1-x^2}$	$\sqrt{1/\pi}$ $(n=0)$ $\sqrt{2/\pi}$ $(n\geq 1)$
Laguerren polynomit	$[0,\infty)$	e^{-x}	1
Hermiten polynomit	$[-\infty,\infty]$	$e^{-x^2/2}$	$1/\sqrt{n!\sqrt{2\pi}}$

Koska approksimointiin ja interpolointiin usein käytetään polynomeja, esitetään vielä *Weierstrassin approksimaatiolause:* jos $f(x)$ on äärellisellä välillä $[a,b]$ jatkuva funktio ja $\epsilon > 0$ mielivaltainen reaaliluku, niin on olemassa polynomi $p(x)$, jolle pätee $|f(x)-p(x)| < \epsilon$ kaikilla $x \in [a,b]$.

Jatkuvia funktioita pystytään siis periaatteessa approksimoimaan halutulla tarkkuudella polynomeilla. Ikävä kyllä käytännössä optimaalisen polynomin määrittäminen voi olla hyvin vaikeaa. Weierstrassin lauseelle on toki olemassa konstruktiivinen todistus, joka perustuu Bernsteinin polynomien käyttöön, mutta tämä ei johda tehokkaaseen menetelmään.

4.4 Pienimmän neliön approksimointi

Pienimmän neliön approksimointi on tavallisimpia approksimointimenetelmiä. Sen edellytyksenä on, että avaruus, josta approksimaatiota haetaan, on varustettu sisätulolla. Approksimaation hyvyyttä mitataan sisätulon määräämän normin avulla.

4.4.1 Funktion arviointi pienimmän neliön mielessä

Tarkastellaan ensin jatkuvaa tapausta, jossa approksimoitavalle funktiolle f etsitään likimääräistä esitystä $\tilde{f}$ lineaarisesti riippumattomien kantafunktioiden $\{\phi_i\}$, $i = 1,\ldots,N$ avulla muodossa

$$\tilde{f} = \sum_{i=1}^{N} c_i \phi_i.$$

Kertoimet c_i pyritään valitsemaan siten, että normi

$$\|f - \tilde{f}\|_2 = \langle f - \tilde{f}, f - \tilde{f} \rangle^{1/2}$$

on mahdollisimman pieni. On osoitettavissa, että tämä on yhtäpitävää sen kanssa, että approksimaatiovirhe $f - \tilde{f}$ ei sisällä minkään kantafunktion suuntaista komponenttia eli virhe on ortogonaalinen kaikkien kantafunktioden suhteen: $\langle f - \tilde{f}, \phi_i \rangle = 0$ kaikilla i. Tämä johtaa *normaaliyhtälöihin*

$$\sum_{i=1}^{N} c_i \langle \phi_i, \phi_j \rangle = \langle f, \phi_j \rangle \quad \text{kaikilla } j = 1, ..., N. \tag{4.1}$$

Koska kantafunktiot ϕ_i oletettiin lineaarisesti riippumattomiksi, normaaliyhtälöiden ryhmällä on periaatteessa aina yksikäsitteinen ratkaisu. Tämän ratkaisun etsiminen voi olla numeerisesti pahanlaatuinen tehtävä, mikäli N on suuri ja kantafunktiot eivät ole ortonormaaleja.

Jos kantafunktiot ϕ_i ovat ortogonaalisia, tehtävä helpottuu huomattavasti, sillä edellä kirjoitetun yhtälöryhmän vasemmalle puolelle jää kullekin riville vain yksi termi ($c_i \langle \phi_i, \phi_i \rangle$), ja kertoimet c_i saadaan jakolaskulla yhtälöistä

$$c_i = \frac{\langle f, \phi_i \rangle}{\langle \phi_i, \phi_i \rangle}. \tag{4.2}$$

Vielä helpommaksi tilanne muuttuu, jos funktiojoukko ϕ_i on ortonormaali eli $\langle \phi_i, \phi_i \rangle = 1$. Tällöin tuntemattomat c_i ratkeavat suoraan approksimoitavan funktion f ja kantafunktioiden ϕ_i välisistä sisätuloista:

$$c_i = \langle f, \phi_i \rangle.$$

Tässä tapauksessa virheelle $f - \tilde{f}$ on voimassa yhtälö

$$\|f - \tilde{f}\|_2^2 = \|f\|_2^2 - \sum_{i=1}^{n} c_i^2,$$

josta seuraa edelleen *Besselin epäyhtälö*

$$\sum_{i=1}^{n} c_i^2 \leq \|f\|_2^2.$$

Mikäli ortonormaali kanta on sellainen, että kasvattamalla kantafunktioiden lukumäärää rajatta virhe saadaan mielivaltaisen pieneksi, kyseessä on *täydellinen kanta*, jolle pätee *Parsevalin yhtälö:*

$$\sum_{i=1}^{\infty} c_i^2 = \|f\|_2^2. \tag{4.3}$$

Besselin epäyhtälö ja Parsevalin yhtälö antavat mahdollisuuden arvioida approksimoitavan funktion L_2-normia kertoimien c_i avulla.

Esimerkki 4.4.1 Approksimoimme toisen asteen polynomilla pienimmän neliön mielessä funktiota $f(x) = e^x$ välillä $[-1, 1]$, kun sisätulo on määritelty yksinkertaisesti $\langle f, g\rangle = \int_{-1}^{1} f(x)g(x)\,dx$.

Valitsemme ensin kantafunktioiksi $\phi_1 = 1$, $\phi_2 = x$ ja $\phi_3 = x^2$, jotka eivät ole ortogonaalisia. Tällöin joudumme muodostamaan normaaliyhtälöt. Koska

$$\int_{-1}^{1} x^p\,dx = \begin{cases} 2/(p+1), & p \text{ parillinen,} \\ 0, & p \text{ pariton,} \end{cases}$$

yhtälöryhmän kerroinmatriisin alkio on $a_{ij} = 2/(i+j-1)$, kun $i+j$ on parillinen ja $a_{ij} = 0$ muutoin. Vastaavasti saamme laskettua oikean puolen vektorin alkiot integraalista $\int_{-1}^{1} x^p e^x\,dx$, $p = 0, 1, 2$. Lopullinen yhtälöryhmä on

$$\begin{pmatrix} 2 & 0 & \frac{2}{3} \\ 0 & \frac{2}{3} & 0 \\ \frac{2}{3} & 0 & \frac{2}{5} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} e - \frac{1}{e} \\ \frac{2}{e} \\ e - \frac{5}{e} \end{pmatrix},$$

josta ratkaisemalla saamme kertoimille likiarvot $c_1 = 0.996294$, $c_2 = 1.103638$ ja $c_3 = 0.536722$. Ratkaisu on siis $e^x \approx \tilde{f}(x) = 0.996294 + 1.103638\,x + 0.536722\,x^2$.

Jos käytämme valmiiksi ortonormeerattua kantaa (tässä tapauksessa Legendren polynomeja, sivu 79)

$$\phi_1 = 1/\sqrt{2},\ \phi_2 = \sqrt{3/2}x \quad \text{ja} \quad \phi_3 = \sqrt{5/2}(3x^2/2 - 1/2),$$

sisätulojen laskeminen on hiukan mutkikkaampaa, mutta toisaalta säästymme lineaarisen yhtälöryhmän ratkaisemiselta. Kertoimet selviävät nyt suoraan sisätuloista:

$$c_1 = \langle f, \phi_1\rangle = \int_{-1}^{1} e^x \frac{1}{\sqrt{2}}\,dx = 1.661985$$

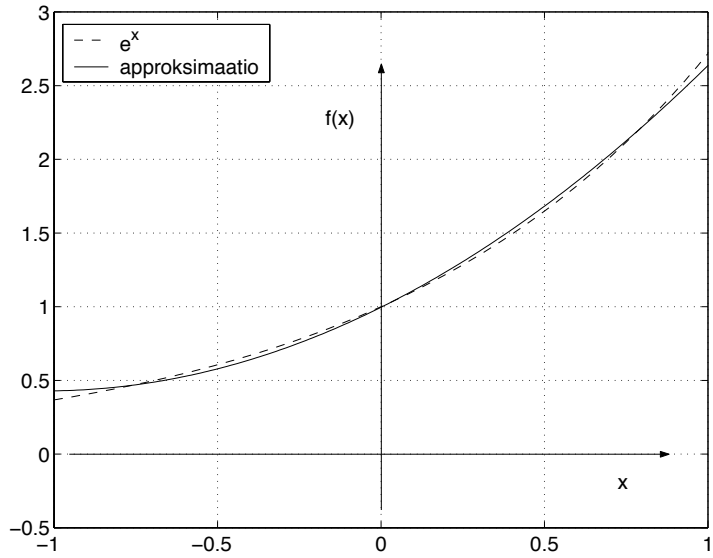
$$c_2 = \langle f, \phi_2\rangle = \int_{-1}^{1} e^x \sqrt{\frac{3}{2}}x\,dx = 0.901117$$

$$c_3 = \langle f, \phi_3\rangle = \int_{-1}^{1} e^x \sqrt{\frac{5}{2}}\left(\frac{3}{2}x^2 - \frac{1}{2}\right)dx = 0.226302$$

Lukuarvot eroavat aiemmista, koska kantafunktioita on vaihdettu, mutta lopputulos on tietysti täysin sama 2. asteen polynomi, joka on esitetty kuvassa 4.3: $e^x \approx \tilde{f}(x) = 1.661985 \cdot 1/\sqrt{2} + 0.901117 \cdot \sqrt{3/2}x + 0.226302 \cdot \sqrt{5/2}(3x^2/2 - 1/2)$.

Jälkimmäinen kanta on ortonormaali, joten voimme esittää arvion $\|f\|_2^2 \geq \sum_{i=1}^{3} c_i^2 = 3.625419$. Arvio on melko tarkka, sillä $\int_{-1}^{1} (e^x)^2\,dx = \sinh(2) = 3.626860$.

Mikäli jostain syystä haluamme käyttää approksimointiin korkea-asteisia polynomeja, ortogonaaliset kantafunktiot ovat suositeltavampi vaihtoehto laskennan stabiilisuuden kannalta.



Kuva 4.3: *Funktio e^x ja sitä approksimoiva toisen asteen polynomi.*

4.4.2 Funktion sovittaminen havaintoaineistoon

Tarkastellaan seuraavaksi diskreettiä pienimmän neliön approksimointia, josta joissain lähteissä käytetään myös nimeä *pienimmän neliösumman approksimointi*. Ideana on sovittaa yksittäisiin havaintoihin käyrä (moniulotteisissa tapauksissa pinta tai hyperpinta), joka kulkee keskimäärin havaintopisteiden läheltä. Menetelmää käytetään hyvin yleisesti, kun mittaustuloksista haetaan säännönmukaisuuksia.

Tehtävänä on etsiä kantafunktioiden $\{\phi_i(x)\}$, $i = 1, ..., n$ virittämästä aliavaruudesta V_1 funktio $\tilde{f}(x) = \sum_{i=1}^{n} c_i \phi_i(x)$, joka likimäärin vastaa havaintopisteissä x_j, $j = 1, ..., N$, tunnettuja funktion f arvoja f_j (usein mittaustuloksia) siten, että

$$\sum_{j=1}^{N} \left(f_j - \tilde{f}(x_j) \right)^2 \leq \sum_{j=1}^{N} \left(f_j - g(x_j) \right)^2 \quad \text{kaikilla } g \in V_1.$$

On siis määritettävä kertoimet c_i siten, että funktio $\tilde{f}(x)$ poikkeaa approksimoitavasta funktiosta $f(x)$ vähemmän kuin mikään muu avaruuden V_1 funktio, kun virhe mitataan havaintopisteissä f_j lasketussa L_2-normissa.

Oletus $\tilde{f}(x) = \sum_{i=1}^{n} c_i \phi_i(x)$ tarkoittaa, että kyseessä on lineaarinen pienimmän neliön sovitus. Epälineaarisessa tapauksessa approksimaation $\tilde{f}(x)$ riippuvuus tuntemattomista parametreista $\{c_i\}$ on monimutkaisempi eikä tehtävä johda seuraavaksi esitettävällä tavalla matriisimuotoon.

Otetaan käyttöön merkintä $\bar{\phi}_i$ tarkoittamaan N-komponenttista vektoria, jonka alkioina ovat kantafunktion $\phi_i(x)$ arvot pisteissä $\{x_j\}$, $j = 1, ..., N$.

Kootaan vastaavasti ratkaistavina olevat parametrit c_i n-komponenttiseen vektoriin $\mathbf{c}$ ja funktion f tunnetut arvot f_j vektoriin $\mathbf{f}$, jossa on N komponenttia. Muodostetaan vielä kerroinmatriisi A asettamalla vektorit $\bar{\phi}_i$ matriisin A pystysarakkeiksi. Tälöin päädymme N yhtälön ja n tuntemattoman lineaariseen yhtälöryhmään $A\mathbf{c} = \mathbf{f}$. Tavallisesti $N > n$, joten yhtälöryhmällä ei yleisessä tapauksessa ole tarkkaa ratkaisua.

Pienimmän neliön approksimointi merkitsee nyt parametrien c_i määrittämistä siten, että *residuaalin* $A\mathbf{c} - \mathbf{f}$ normi $\|A\mathbf{c} - \mathbf{f}\|_2$ minimoituu.

Eräs tapa löytää ratkaisu on lähteä jälleen liikkeelle normaaliyhtälöistä

$$\sum_{i=1}^{n} c_i \langle \bar{\phi}_i, \bar{\phi}_j \rangle = \langle \mathbf{f}, \bar{\phi}_j \rangle, \quad j = 1, ..., n, \tag{4.4}$$

jotka matriisin A avulla lausuttuna saavat yksinkertaisen muodon

$$A^T A\mathbf{c} = A^T \mathbf{f}. \tag{4.5}$$

Edellä esitetty perusmenetelmä sisältää oletuksen, että mittaustuloksissa f_j olevat virheet ϵ_j ovat normaalijakautuneita ja keskenään korreloimattomia. Lisäksi virheiden keskihajonnat σ_j oletetaan samansuuruisiksi.

Jos on syytä olettaa, että kaikissa mittauksissa ei ole päästy yhtä hyvään tarkkuuteen, pienimmän neliön ratkaisussa voidaan korostaa tarkempia mittauksia ja antaa epätarkoille mittauksille pienempi painoarvo. Laskujen kannalta tämä merkitsee aiemmin muodostetun yhtälöryhmän $A\mathbf{c} = \mathbf{f}$ kertomista puolittain lävistäjämatriisilla W, jonka alkio w_{jj} kertoo millä suhteellisella painoarvolla rivin j yhtälö – ja sitä kautta mittaustulos f_j – vaikuttaa lopputulokseen. Normaaliyhtälöt muodostetaan jälleen kertomalla koko yhtälöryhmä vasemman puolen kerroinmatriisin transpoosilla:

$$(WA)^T WA\mathbf{c} = (WA)^T W\mathbf{f} \quad \text{eli } A^T CA\mathbf{c} = A^T C\mathbf{f}, \tag{4.6}$$

missä otettiin käyttöön merkintä $C = W^T W$ ja lävistäjämatriisin C alkiot ovat $c_{jj} = w_{jj}^2$.

Painokertoimet c_{jj} voidaan tietysti valita monella eri tavalla. Jos mittaustulos f_j on peräisin kokeesta, jossa tulosten jakauman keskihajonta σ_j tunnetaan, niin paras valinta on

$$c_{jj} = 1/\sigma_j^2. \tag{4.7}$$

Siis mitä suurempi hajonta kokeessa on, sitä pienemmän painoarvon annamme ko. kokeesta saadulle tulokselle.

Vielä yleisemmässä tapauksessa voidaan olettaa, että eri mittaustulosten virheet ϵ_j eivät ole riipumattomia. Tällöin kerroinmatriisissa C myös muut kuin lävistäjäalkiot voivat saada nollasta poikkeavia arvoja. Matemaattisesti optimaalinen valinta on tässä tapauksessa $C = V^{-1}$, missä V on mittausvirheiden ϵ_j *kovarianssimatriisi.* Kovarianssimatriisin lävistäjällä on virhejakaumien varianssit $V_{jj} = E[\epsilon_j^2] = \sigma_j^2$ ja muut alkiot ovat virhejakaumien välisiä kovariansseja $V_{ij} = E[\epsilon_i \epsilon_j]$.

Yleensä havaintopisteitä on selvästi enemmän kuin approksimoivassa käyrässä vapaita parametreja ($N > n$). Tämä johtuu siitä, että pienimmän neliön menetelmällä halutaan löytää suhteellisen yksinkertaisia ja sileitä approksimaatioita. Sen sijaan, että yhdistettäisiin havainnot esimerkiksi murtoviivalla tai voimakkaasti heilahtelevalla korkea-asteisella polynomilla, aineistoon sovitetaan mieluummin jokin matala-asteinen polynomi. Jos on $N = n$, kyseessä on interpolointitehtävä, joka yleensä ratkeaa nopeimmin tavallisilla lineaarisia yhtälöryhmiä varten suunnitelluilla algoritmeilla.

Jos pätee $N < n$ tai jos matriisin A säännöllisyysaste eli rangi on pienempi kuin n, pienimmän neliön tehtävällä on ääretön määrä ratkaisuja. Tällöin tavallisesti valitaan pienimmän residuaalin tuottavista ratkaisuista normiltaan pienin. Esitämme myöhemmin esimerkkejä tällaisista erikoistapauksista.

Esimerkki 4.4.2 Eräällä sääasemalla on mitattu aamuyön aikana seuraavat lämpötilahavainnot:

kellonaika	03.00	04.00	05.00	06.00	07.00
lämpötila $^\circ C$	11.2	7.5	10.2	14.7	19.6

Arvioimme, milloin lämpötila T on ylittänyt $15^\circ C$. Etsimme lineaarisen polynomin $T(t) = c_1 + c_2 t$, joka kuvaa lämpötilan nousua aamun aikana. Valitsemme kantafunktioiksi stabiilisuudesta murehtimatta suoraviivaisesti $\phi_1 = 1$, $\phi_2 = t$ ja skaalaamme kellonajat symmetrisesti ajanhetken 05.00 ympärille, jolloin uudet mittauspisteet ovat -2.00, -1.00, 0.00, 1.00 ja 2.00. Oletamme, että mittausvirheet ovat korreloimattomia ja että niillä on sama keskihajonta, jolloin voimme ratkaista tehtävän ilman erillistä painokerroinmatriisia C.

Normaaliyhtälöihin tarvittavat sisätulot ovat

$$\begin{aligned}
\langle 1,1\rangle &= \textstyle\sum_{i=1}^{5} 1\cdot 1 = 5\\
\langle 1,t\rangle &= \textstyle\sum_{i=1}^{5} 1\cdot t_i = (-2.00)+(-1.00)+0.00+1.00+2.00 = 0.00\\
\langle t,t\rangle &= \textstyle\sum_{i=1}^{5} t_i^2 = 4.00+1.00+0.00+1.00+4.00 = 10.00\\
\langle 1,T\rangle &= \textstyle\sum_{i=1}^{5} 1\cdot T_i = 11.2+7.5+10.2+14.7+19.6 = 63.2\\
\langle t,T\rangle &= \textstyle\sum_{i=1}^{5} t_i\cdot T_i = (-2.00)\cdot 11.2+(-1.00)\cdot 7.5+0.00\cdot 10.2\\
&\qquad +1.00\cdot 14.7+2.00\cdot 19.6 = 24.0
\end{aligned}$$

Normaaliyhtälöt ovat matriisimuodossa (yhtälö (4.4))

$$\begin{pmatrix} 5 & 0\\ 0 & 10\end{pmatrix}\begin{pmatrix} c_1\\ c_2\end{pmatrix} = \begin{pmatrix} 63.2\\ 24.0\end{pmatrix},$$

josta on helppo ratkaista kertoimet $c_1 = 12.64$ ja $c_2 = 2.40$. Lineaarinen pienimmän neliön approksimaatio lämpötilan aikakäyttäytymiselle on siten $T(t) = 2.40(t-5) + 12.64$, josta edelleen voidaan ratkaista $t_{15} \approx 5.983$. Siis $15^\circ C$:n lämpötila on saavutettu noin kello 05.59. Tämä on siis mittaustuloksiin sovitetun mallin antama tulos. Huomaa, että klo 6.00 tehdyn mittauksen mukaan lämpötila on ollut vasta $14.7^\circ C$.

Normaaliyhtälöt on mahdollista kirjoittaa myös laskemalla ensin kaavoissa esiintyvä kerroinmatriisi A ja sitten soveltamalla yhtälöä (4.5):

$$A = \begin{pmatrix} 1 & -2.00\\ 1 & -1.00\\ 1 & 0.00\\ 1 & 1.00\\ 1 & 2.00\end{pmatrix} \Rightarrow A^T = \begin{pmatrix} 1 & 1 & 1 & 1 & 1\\ -2 & -1 & 0 & 1 & 2\end{pmatrix} \Rightarrow A^TA = \begin{pmatrix} 5 & 0\\ 0 & 10\end{pmatrix}$$

Käytännössä pienimmän neliösumman tehtävän ratkaiseminen normaaliyhtälöiden kautta ei ole aina numeerisesti järkevin tapa. Jos matriisin A häiriöalttius on $\kappa(A)$ (häiriöalttius on määritelty kappaleessa 3.4), niin normaaliyhtälöistä lasketun ratkaisun tarkkuus on verrannollinen tekijään $(\kappa(A))^2$. Stabiilimpaan menetelmään päästään käyttämällä matriisin A QR-hajotelmaa. Hintana paremmasta tarkkuudesta ratkaisussa joudutaan tekemään enemmän laskutyötä.

Jos A on $N \times n$-matriisi ($N > n$), sen QR-hajotelma on muotoa

$$A = \underset{N\times N}{Q}\ \underset{N\times n}{R} = \begin{pmatrix} \underset{N\times n}{Q_1} & \underset{N\times(N-n)}{Q_2} \end{pmatrix} \begin{pmatrix} \underset{n\times n}{R_1} \\ \underset{(N-n)\times n}{0} \end{pmatrix}.$$

Tällöin on $Q^TA = R$ ja pätee

$$Q^T\mathbf{f} = \begin{pmatrix} \mathbf{d} \\ \mathbf{e} \end{pmatrix},$$

missä $\mathbf{d}$ on n-komponenttinen ja $\mathbf{e}$ puolestaan $(N-n)$-komponenttinen pystyvektori. Koska ortogonaalimuunnos säilyttää normin, on voimassa

$$\|A\mathbf{c} - \mathbf{f}\|_2^2 = \|Q^TA\mathbf{c} - Q^T\mathbf{f}\|_2^2 = \|R_1\mathbf{c} - \mathbf{d}\|_2^2 + \|\mathbf{e}\|_2^2.$$

Näin ollen yhtälöryhmän $A\mathbf{c} = \mathbf{f}$ ratkaisu pienimmän neliön mielessä toteuttaa myös helposti ratkeavan systeemin $R_1\mathbf{c} = \mathbf{d}$, jonka kerroinmatriisi on yläkolmiomatriisi. Suurin osa työstä kuluukin matriisin A QR-hajotelman muodostamiseen. Kaikki merkittävät matemaattiset aliohjelmakirjastot sisältävät kuitenkin valmiin ohjelman QR-hajotelman muodostamiseksi.

Esimerkki 4.4.3 Laskemme edellisen esimerkin vielä QR-hajotelman avulla. Matriisin A QR-hajotelma on helppo selvittää esimerkiksi Matlab-ohjelmiston avulla:

$$Q = \begin{pmatrix} -0.4472 & -0.6325 & -0.4149 & -0.3626 & -0.3104 \\ -0.4472 & -0.3162 & 0.0672 & 0.3996 & 0.7320 \\ -0.4472 & -0.0000 & 0.8377 & -0.2013 & -0.2403 \\ -0.4472 & 0.3162 & -0.2174 & 0.6543 & -0.4739 \\ -0.4472 & 0.6325 & -0.2726 & -0.4900 & 0.2925 \end{pmatrix},$$

$$R = \begin{pmatrix} -2.2361 & 0 \\ 0 & 3.1623 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \quad \text{ja} \quad Q^T\mathbf{f} = \begin{pmatrix} -28.2639 \\ 7.5895 \\ -4.1371 \\ -3.1032 \\ -1.6692 \end{pmatrix}.$$

Ratkaistavaksi jää siis yhtälöryhmä

$$\begin{pmatrix} -2.2361 & 0 \\ 0 & 3.1623 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} -28.2639 \\ 7.5895 \end{pmatrix} \Longrightarrow \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} 12.64 \\ 2.40 \end{pmatrix},$$

joten päädyimme samaan lopputulokseen kuin normaaliyhtälöitä käyttäen.

4.4.3 Erikoistapauksia

Tähän asti olemme olettaneet, että pienimmän neliön tehtävässä on vähintään yhtä paljon lineaarisesti riippumattomia yhtälöitä kuin tuntemattomia (mittauksia vähintään yhtä paljon kuin ratkaistavia parametreja). Jos tilanne on päinvastainen, tehtävää ei saada ratkaistuksi normaaliyhtälöiden avulla, vaan joudutaan turvautumaan joko singulaariarvohajotelmaan tai QR-hajotelmaan.

Tutkimme ensin tapausta, jossa matriisin A säännöllisyysaste on pienempi kuin n. Tässä tapauksessa yhtälöitä on periaatteessa riittävästi, mutta kerroinmatriisin vaakarivit ovat lineaarisesti riippuvia. Tällöin voimme käyttää singulaariarvohajotelmaa.

Jos matriisi A on kokoa $N \times n$, sen singulaariarvohajotelma on muotoa $A = U\Sigma V^T$, missä U on $N \times N$- ja V on $n \times n$-ortogonaalimatriisi. Matriisi Σ on $N \times n$-matriisi, jonka päädiagonaalilla ovat matriisin A singulaariarvot $\sigma_1, ..., \sigma_p$, $p = \min(N, n)$. Myös singulaariarvohajotelman laskemiseen kannattaa käyttää valmiita ohjelmia.

Oletamme, että matriisin A säännöllisyysaste on r ja singulaariarvohajotelman matriisit U ja V koostuvat sarakkeista $(\mathbf{u}_1, ..., \mathbf{u}_N)$ ja $(\mathbf{v}_1, ..., \mathbf{v}_n)$. Tällöin pienimmän neliön tehtävän

$$\underset{\mathbf{c}}{\text{minimi}} \ \|A\mathbf{c} - \mathbf{f}\|_2$$

ratkaisu on

$$\mathbf{c} = \sum_{i=1}^{r} \frac{\mathbf{u}_i^T \mathbf{f}}{\sigma_i} \mathbf{v}_i. \tag{4.8}$$

Esimerkki 4.4.4 Kokeilemme singulaariarvohajotelman käyttöä yhtälöryhmään

$$\begin{cases} x + y &= 2 \\ x + y &= 3. \end{cases}$$

Vaikka yhtälöitä on yhtä paljon kuin tuntemattomia, systeemi ei ratkea normaaliyhtälöiden avulla, koska yhtälöiden vasemmat puolet ovat lineaarisesti riippuvia (itse asiassa identtisiä). Kerroinmatriisin singulaariarvohajotelma on

$$\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix} \begin{pmatrix} 2 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix}.$$

Saamme muodolliseksi "ratkaisuksi" (säännöllisyysaste $r = 1$ yhtälössä (4.8))

$$\begin{pmatrix} x \\ y \end{pmatrix} = \frac{\left(\frac{1}{\sqrt{2}} \ \frac{1}{\sqrt{2}}\right) \begin{pmatrix} 2 \\ 3 \end{pmatrix}}{2} \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} 5/4 \\ 5/4 \end{pmatrix},$$

joka on suoran $x + y = 5/2$ (alkuperäisten yhtälöiden "keskiarvo") pisteistä lähimpänä origoa ja siis L_2-normiltaan pienin.

Toinen erikoistilanne on *alimäärätty yhtälöryhmä*, jossa on vähemmän yhtälöitä kuin tuntemattomia ($N < n$). Singulaariarvohajotelma toimii tässäkin tapauksessa, mutta jos kerroinmatriisin säännöllisyysaste $r = N$, tehtävä on mahdollista ratkaista myös QR-hajotelman avulla.

Muodostamme QR-hajotelman kerroinmatriisin transpoosille A^T:

$$A^T = QR = Q \begin{pmatrix} R_1 \\ 0 \end{pmatrix},$$

missä $R_1 \in \mathbb{R}^{N \times N}$. Tällöin saamme

$$\begin{aligned} & A\mathbf{c} = \mathbf{f} \\ \Longrightarrow\ & (QR)^T\mathbf{c} = \mathbf{f} \\ \Longrightarrow\ & \begin{pmatrix} R_1^T & 0 \end{pmatrix} Q^T\mathbf{c} = \mathbf{f} \\ \Longrightarrow\ & \begin{pmatrix} R_1^T & 0 \end{pmatrix} \begin{pmatrix} \mathbf{d} \\ \mathbf{e} \end{pmatrix} = \mathbf{f}. \end{aligned}$$

Nyt vektori $\mathbf{d} \in \mathbb{R}^N$ ja vektori $\mathbf{e} \in \mathbb{R}^{n-N}$. Pienimmän normin tuottava ratkaisuvektori saavutetaan asettamalla $\mathbf{e} = \mathbf{0}$.

Jos approksimoinnin yhteydessä päädytään alimäärättyyn yhtälöryhmään, on yleensä syytä tarkistaa mallin järkevyys ja esimerkiksi vähentää kantafunktioiden (ja siten vapaiden parametrien) lukumäärää tai vaihtoehtoisesti lisätä havaintoja.

Esimerkki 4.4.5 Yhtälöryhmässä

$$\begin{cases} x + y + z &= 3 \\ x + 2y + 3z &= 6 \end{cases}$$

on vähemmän yhtälöitä kuin tuntemattomia. Koska yhtälöiden vasemmat puolet ovat lineaarisesti riippumattomia, voimme ratkaista tehtävän edellä kuvatulla tavalla (jos vasemmat puolet olisivat lineaarisesti riippuvia, joutuisimme käyttämään singulaariarvohajotelmaa). Kerroinmatriisin transpoosin hajotelma on

$$A^T = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{pmatrix} = \begin{pmatrix} 1/\sqrt{3} & -1/\sqrt{2} & -1/\sqrt{6} \\ 1/\sqrt{3} & 0 & 2/\sqrt{6} \\ 1/\sqrt{3} & 1/\sqrt{2} & -1/\sqrt{6} \end{pmatrix} \begin{pmatrix} \sqrt{3} & 2\sqrt{3} \\ 0 & \sqrt{2} \\ 0 & 0 \end{pmatrix} = QR.$$

Nyt ratkaisemme ensin vektorin $\mathbf{d}$ yhtälöryhmästä $R_1^T\mathbf{d} = \mathbf{f}$ ja lopuksi suoritamme kertolaskun $\mathbf{c} = Q'\mathbf{d}$, missä merkintä Q' viittaa siihen, että matriisista Q otetaan mukaan vain $N = 2$ ensimmäistä pystysaraketta.

$$R_1^T\mathbf{d} = \mathbf{f} \quad \Longrightarrow \quad \begin{pmatrix} \sqrt{3} & 0 \\ 2\sqrt{3} & \sqrt{2} \end{pmatrix} \begin{pmatrix} d_1 \\ d_2 \end{pmatrix} = \begin{pmatrix} 3 \\ 6 \end{pmatrix} \quad \Longrightarrow \quad \mathbf{d} = \begin{pmatrix} \sqrt{3} \\ 0 \end{pmatrix};$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = c = Q'd = \begin{pmatrix} 1/\sqrt{3} & -1/\sqrt{2} \\ 1/\sqrt{3} & 0 \\ 1/\sqrt{3} & 1/\sqrt{2} \end{pmatrix} \begin{pmatrix} \sqrt{3} \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

Alkuperäisellä yhtälöparilla on ääretön määrä ratkaisuja $\{x, 3 - 2x, x\}$, joista kolmikko $\{1, 1, 1\}$ on siis L_2-normiltaan pienin.

4.4.4 Rekursiivinen pienimmän neliön sovitus

Pohditaan tilannetta, jossa mittaustuloksista $\mathbf{f}_0$ on pienimmän neliösumman menetelmällä ratkaistu havaintoaineistoa kuvaavan funktion parametreille $\mathbf{c}$ jotkin arvot $\mathbf{c}_0$. Kun myöhemmin mittausdataa kertyy lisää, voi osoittautua tarpeelliseksi laskea parametreille $\mathbf{c}$ uudet arvot $\mathbf{c}_1$, jotka perustuvat sekä vanhoihin ($\mathbf{f}_0$) että uusiin havaintoihin ($\mathbf{f}_1$).

Periaatteessa tehtävän voi ratkaista muodostamalla normaaliyhtälöt 4.5 koko mittausdatalle:

$$\left(A_0^T A_1^T\right) \begin{pmatrix} A_0^T \\ A_1^T \end{pmatrix} \mathbf{c}_1 = \left(A_0^T A_1^T\right) \begin{pmatrix} \mathbf{f}_0 \\ \mathbf{f}_1 \end{pmatrix}. \tag{4.9}$$

Yritämme nyt sieventää normaaliyhtälöt sellaiseen muotoon, että pääsemme hyödyntämään jo aiemmin laskettuja tuloksia $\mathbf{c}_0$. Korvaamme ensin oikealla puolella termin $A_0^T\mathbf{f}_0$ tulolla $A_0^T A_0\mathbf{c}_0$, ja lisäämällä ja vähentämällä sitten oikealla puolella termin $A_1^T A_1\mathbf{c}_0$ saamme vektorin $\mathbf{c}_0$ kerroinmatriiksiksi saman matriisin, joka esiintyy yhtälöryhmän vasemmalla puolella. Kun yhtälöryhmä nyt ratkaistaan vektorin $\mathbf{c}_1$ suhteen, päädymme tulokseen

$$\begin{aligned} \mathbf{c}_1 &= \mathbf{c}_0 + \left(A_0^T A_0 + A_1^T A_1\right)^{-1} A_1^T(\mathbf{f}_1 - A_1\mathbf{c}_0) \\ &= \mathbf{c}_0 + K_1(\mathbf{f}_1 - A_1\mathbf{c}_0). \end{aligned}$$

Viimeisellä rivillä määritelty matriisi K_1 on tehtävään liittyvä *hyötymatriisi*. Uudet malliparametrit $\mathbf{c}_1$ saadaan siis vanhoista parametreistä $\mathbf{c}_0$ lisäämällä niihin korjaustermi $K_1(\mathbf{f}_1 - A_1\mathbf{c}_0)$. Tässä tekijä $\mathbf{f}_1 - A_1\mathbf{c}_0$ on alkuperäisen mallin ennustusvirhe. Jos ennustusvirhe on $\mathbf{0}$, niin mitään korjausta parametrihin $\mathbf{c}_0$ ei tarvitse tehdä.

Hyötymatriisi $K_1 = \left(A_0^T A_0 + A_1^T A_1\right)^{-1} A_1^T$ voidaan kirjoittaa lyhyesti tulona $K_1 = P_1 A_1^T$. Kun sovitaan merkinnästä $P_0^{-1} = \left(A_0^T A_0\right)$, niin $P_1^{-1} = P_0^{-1} + A_1^T A_1$, ja yleisemmin usean mittaussarjan tapauksessa $P_i^{-1} = P_{i-1}^{-1} + A_i^T A_i$. Tällöin vastaavasti $K_i = P_i A_i^T$ ja viimeisimmät mittaustulokset huomioon ottavat parametrit saadaan aina yhtälöryhmästä $\mathbf{c}_i = \mathbf{c}_{i-1} + K_i(\mathbf{f}_i - A_i\mathbf{c}_{i-1})$.

4.4.5 Täydellinen pienimmän neliön sovitus

Tavanomaisessa pienimmän neliön sovituksessa oletetaan, että mittauspisteet tunnetaan tarkasti ja mittaustuloksissa olevat virheet ovat riippumattomia ja normaalijakautuneet odotusarvon nolla ympärille. Käytännössä kuitenkin myös mittauspisteissä voi olla virhettä. Esimerkissä 4.4.2 implisiittisesti oletimme, että mittausajankohdat olisivat absoluuttisen tarkkoja ja kaikki malliin sisältyvä virhe olisi peräisin lämpötilamittauksista.

Täydellisessä pienimmän neliön sovituksessa (total least squares fit) myös lähtödata oletetaan virheelliseksi. Perustapauksessa $N \times n$ -kokoisen yhtälöryhmän $A\mathbf{c} = \mathbf{f}$ $(N \geq n+1)$ ratkaisu etenee seuraavasti: haetaan laajennetun matriisin $M = C[A\,\mathbf{f}]T$ singulaariarvohajotelma $U^T MV = \Sigma =$

diag$(\sigma_1^M, \sigma_2^M, ..., \sigma_{n+1}^M)$, jossa matriisi V on ositettavissa $N \times N$-matriisiksi, kahdeksi $N-$vektoriksi ja skalaariksi muotoon

$$V = \begin{bmatrix} V_{11} & \mathbf{v}_{12} \\ \mathbf{v}_{21} & v_{22} \end{bmatrix}.$$

Edellä C on $m \times m$-lävistäjämatriisi, jonka avulla painotetaan yksittäisten yhtälöiden merkitystä koko yhtälöryhmän ratkaisussa.

Lävistäjämatriisi T on kokoa $(n+1) \times (n+1)$. Se ottaa huomioon lähtödataan sisältyvän virheen ja mittaustuloksiin sisältyvän virheen erilaisuuden. Esimerkiksi lämpötilanmittauksessa voimme olettaa, että mittaushetket pystytään määrittämään paremmalla tarkkuudella kuin lämpötilat.

Jos alkuperäisen matriisin A pienin singulaariarvo σ_n^A toteuttaa ehdot $\sigma_n^A > 0$ ja $\sigma_n^A > \sigma_{n+1}^M$, niin täydellisen pienimmän neliön tehtävän ratkaisu on vektori

$$\mathbf{c} = -\frac{1}{v_{22} T_{n+1,n+1}} \mathbf{v}_{12}. \tag{4.10}$$

Täydellisellä pienimmän neliön tehtävällä ei suinkaan aina ole ratkaisua, tai ratkaisuja voi olla ääretön määrä. Erikoistapauksia varten kannattaa tutustua kirjallisuudessa esitettyihin teoreettisiin tuloksiin.

Esimerkki 4.4.6 Ratkaisemme esimerkin 4.4.2 uudelleen täydellisellä pienimmän neliön sovituksella. Alkuperäisessä esimerkissä oli jo laskettu tehtävään liittyvä kerroinmatriisi A. Vektori $\mathbf{f}$ sisältää siis lämpötilahavainnot:

$$A = \begin{pmatrix} 1 & -2.00 \\ 1 & -1.00 \\ 1 & 0.00 \\ 1 & 1.00 \\ 1 & 2.00 \end{pmatrix}, \quad \mathbf{f} = \begin{pmatrix} 11.2 \\ 7.5 \\ 10.2 \\ 14.7 \\ 19.6 \end{pmatrix}.$$

Oletamme, että kaksi ensimmäistä mittausta tehnyt henkilö on luonteeltaan vähemmän huolellinen kuin kolmen viimeisen mittaustuloksen kirjaaja. Tämän voimme ottaa huomioon painottamalla tuloksia esimerkiksi lävistäjämatriisilla $C = \text{diag}(1, 1, 2, 2, 2)$.

Oletamme lisäksi, että lämpötiloihin sisältyvä virhe on viisi kertaa niin merkittävä kuin mittausajanhetkiin liittyvä virhe ja valitsemme matriisiksi $T = \text{diag}(1000, 5, 1)$. Ensimmäiselle alkiolle T_{11} annamme jonkin suuren arvon (tässä tapauksessa $T_{11} = 1000$), koska se painottaa matriisin A ensimmäiseen pystyriviin liittyvää varmuutta. Ensimmäiseen pystyriviinhän on kerätty ensimmäisen kantafunktion $\phi_1 \equiv 1$ mittaushetkillä t_j saamat arvot, jotka eivät riipu mittaushetkestä lainkaan, joten voimme tulkita nämä arvot erittäin luotettaviksi.

Muodostamme nyt ensin laajennetun matriisin

$$M = C\,[A\,\mathbf{f}]\,T = \text{diag}(1, 1, 2, 2, 2) \begin{pmatrix} 1 & -2 & 11.2 \\ 1 & -1 & 7.5 \\ 1 & 0 & 10.2 \\ 1 & 1 & 14.7 \\ 1 & 2 & 19.6 \end{pmatrix} \text{diag}(1000, 1, 5).$$

Matriisin M singulaariarvohajotelmaan $U^T MV = \Sigma$ liittyvä matriisi V on (esim. Matlabia käyttäen)

$$V = \begin{pmatrix} -0.999896 & 0.010479 & -0.009894 \\ -0.003214 & -0.831373 & -0.555705 \\ -0.014049 & -0.555616 & 0.831320 \end{pmatrix}$$

$$\Rightarrow \mathbf{v}_{12} = \begin{pmatrix} -0.009894 \\ -0.555705 \end{pmatrix}, \quad v_{22} = 0.831320.$$

Siten ratkaisu on lopulta

$$\mathbf{c} = \frac{-1}{0.831320} \begin{pmatrix} -0.009894 \\ -0.555705 \end{pmatrix} = \begin{pmatrix} 11.901407 \\ 3.342306 \end{pmatrix}.$$

Saamme siis lämpötilan aikariippuvuudeksi tällä menetelmällä $T(t) \approx 3.34(t - 5) + 11.90$. Arvioitu ajanhetki, jolloin lämpötila olisi ylittänyt arvon $15°C$, on $t_{15} \approx 5.927$ tuntia puolenyön jälkeen eli noin kello 05.56.

4.5 Tasainen approksimointi

Edellisessä kappaleessa etsittiin approksimaatiota pienimmän neliön mielessä eli L_2-normin suhteen. *Tasaisessa approksimaatiossa* (joskus käytetään myös nimeä *Tšebyševin approksimaatio*) tehtävä säilyy abstraktilla tasolla muutoin ennallaan, mutta L_2-normin sijasta käytetäänkin L_∞-normia. On siis löydettävä avaruuden V alkiolle $\mathbf{f}$ likimääräinen vastine $\tilde{\mathbf{f}}$ aliavaruudesta $V_1 \subset V$, jolle maksimivirhe $\|\mathbf{f} - \tilde{\mathbf{f}}\|_\infty$ on mahdollisimman pieni, eli pätee

$$\|\mathbf{f} - \tilde{\mathbf{f}}\|_\infty \leq \|\mathbf{f} - \mathbf{g}\|_\infty \quad \text{kaikilla} \quad \mathbf{g} \in V_1.$$

Käytännön tasolla ero on merkittävä. Toisin kuin pienimmän neliön approksimoinnissa, *ei ole olemassa yksinkertaista menetelmää löytää yleisessä tapauksessa annetun funktion parasta tasaista approksimaatiota*. Itse asiassa ilman sopivia lisäoletuksia ei voida taata edes ratkaisun yksikäsitteisyyttä.

Tasaista approksimaatiota sovelletaan tilanteissa, joissa on tärkeätä, että virhe saadaan pidettyä pienenä koko approksimointivälilllä. Tilastomatemaattisin argumentein voidaan perustella tasaisen approksimaation käyttöä mittaustulosten mallinnuksessa silloin, kun aineiston virheet ovat oleellisesti seurausta tulosten pyöristämisestä. Pienimmän neliön sovitus dataaineistoon on vastaavasti paikallaan silloin, kun aineistossa olevat mittausvirheet voidaan olettaa normaalijakautuneeksi.

4.5.1 Polynomiapproksimaatiot

Seuraavassa rajoitumme tarkastelemaan polynomiapproksimaatioita. Aliavaruus V_1, josta approksimaatioita etsitään, koostuu korkeintaan astetta n

olevista polynomeista. Tälle aliavaruudelle käytetään merkintää $\mathcal{P}_n$. Jatkuvan funktion tasainen approksimointi ratkeaa yksikäsitteisesti polynomeilla laskettaessa.

Polynomiavaruuksissa parasta tasaista approksimaatiota karakterisoi *maksimivirheen alternointi*. Polynomi $\tilde{f} \in \mathcal{P}_n$ on välillä $[a,b]$ jatkuvan funktion f paras tasainen approksimaatio täsmälleen silloin, kun on olemassa pistejoukko $a \le x_1 < x_2 < \ldots < x_{n+1} < x_{n+2} \le b$, jolla on ominaisuus

$$|f(x_i) - \tilde{f}(x_i)| = \|f - \tilde{f}\|_\infty, \quad i = 1, \ldots, n+2$$

ja

$$f(x_{i+1}) - \tilde{f}(x_{i+1}) = -(f(x_i) - \tilde{f}(x_i)), \quad i = 1, \ldots, n+1.$$

Siis n-asteisen polynomiapproksimaation pitää saavuttaa maksimaalinen virhe $n+2$ pisteessä $\{x_i\}$. Virheen etumerkki vaihtuu siirryttäessä pisteestä x_i pisteeseen x_{i+1}. Intuitiivisesti asiaa voi perustella seuraavasti: niin kauan kuin suurin virhe saavutetaan vain harvoissa ääriarvopisteissä (vähemmän kuin $n+2$), virhettä voidaan lisätä muualla, jos virhettä maksimikohdissa saadaan samalla pienemmäksi. L_∞-normihan mittaa vain suurinta virhettä. Kun ääriarvopisteitä on $n+2$, n-asteisen polynomin vapausasteet on käytetty mahdollisimman tarkasti eikä yhtäkään virheen maksimia voi pienentää ilman, että jokin toinen maksimi kasvaisi. Tasaisesta approksimoinnista käytetäänkin myös nimeä *minimax-approksimointi:* on etsittävä approksimaatio, jolla on mahdollisimman pieni maksimivirhe.

Esimerkki 4.5.1 Joissain helpoissa tapauksissa tasainen approksimaatio löytyy suoraan analyyttisesti. Etsimme funktiolle $f(x) = x^2$ approksimaation 1. asteen polynomien joukosta välillä $[-1,1]$. Koska $f(x)$ on symmetrinen tarkasteluvälin suhteen, on selvää, että itse asiassa vakiofunktio $\tilde{f}(x) = c$ riittää. Vakio c on valittava siten, että erotus välin päissä on itseisarvoltaan yhtä suuri kuin välin keskipisteessä, eli $|1-c| = |0-c|$, josta saadaan tulos $c = 1/2$ (kuva 4.4a).

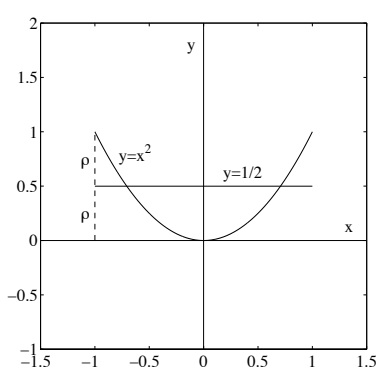
Hankaloitamme tehtävää hiukan muuttamalla väliksi $[a,b] = [0,2]$. Nyt 1. asteen termistä ei päästä eroon symmetrian nojalla, vaan approksimaatio täytyy etsiä kaikkien muotoa $c_1 x + c_2$ olevien polynomien joukosta. Erotus $d(x) = x^2 - (c_1 x + c_2)$ on 2. asteen polynomi, joten sillä voi olla suljetulla välillä kolme ääriarvokohtaa vain, jos kaksi näistä osuu välin päätepisteisiin $x_1 = 0$ ja $x_3 = 2$. Keskimmäinen ääriarvopiste voidaan valita esimerkiksi seuraavasti. Koska virheen $d(x)$ ääriarvokohtia on kolme ja etumerkki alternoi, virhe on päätepisteissä yhtäsuuri:

$$f(x_1) - c_1 x_1 - c_2 = f(x_3) - c_1 x_3 - c_2 \quad \Longrightarrow \quad c_1 = \frac{f(x_3) - f(x_1)}{x_3 - x_1}.$$

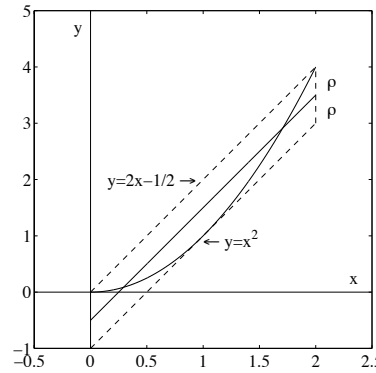
Sijoittamalla tähän jo tunnetut tiedot $f(x) = x^2$, $x_1 = 0$, $x_3 = 2$ saamme tuloksen $c_1 = 4/2 = 2$. Toisaalta virhe saavuttaa pisteessä x_2 ääriarvon, joten täytyy olla voimassa $d'(x_2) = 2x_2 - c_1 = 0 \Longrightarrow x_2 = c_1/2 = 1$. Jäljellä on vielä kaksi tuntematonta: kerroin c_2 ja erotuksen itseisarvon maksimi ρ. Nämä ratkeavat ehdoista

$$\begin{cases} 0^2 - 2\cdot 0 - c_2 = \rho \\ 1^2 - 2\cdot 1 - c_2 = -\rho \end{cases} \Longrightarrow \begin{cases} c_2 = -1/2 \\ \rho = 1/2. \end{cases}$$

Haluttu approksimaatio on siten $\tilde{f}(x) = 2x - 1/2$ (kuva 4.4b).



a: *Väli* $[-1, 1]$.



b: *Väli* $[0, 2]$.

Kuva 4.4: *Funktion* $f(x) = x^2$ *tasainen approksimointi.*

Yleisessä tapauksessa edellisen esimerkin kaltainen suoraviivainen laskenta ei onnistu vaan approksimaatio joudutaan hakemaan jonkin iteratiivisen prosessin kautta. Eräs mahdollisuus on *Remezin menetelmä*. Siinä valitaan aluksi joko satunnaisesti tai mahdollisen taustainformaation avulla päättelemällä $n+2$ pistettä $\{x_i^0\} \in [a, b]$. Seuraavaksi muodostetaan ensimmäinen iteraatti $\tilde{f}^0$ vaatimalla, että virhe $f - \tilde{f}^0$ saa pisteissä $\{x_i^0\}$ vuorotellen arvon ρ^0 ja $-\rho^0$. Tuntemattomia on $n + 2$ kappaletta (polynomin $\tilde{f}^0(x)$ kertoimet sekä ρ^0), ja syntyvällä lineaarisella yhtälöryhmällä on aina ratkaisu.

Seuraavaksi etsitään jokin piste $\xi^1 \in [a, b]$, jossa

$$|f(\xi^1) - \tilde{f}^0(\xi^1)| = \|f - \tilde{f}^0\|_\infty.$$

Jos ξ^1 on jokin pisteistä x_i^0, lopullinen approksimaatio on iteraatti $\tilde{f}^0$ maksimivirheen alternointiominaisuuden nojalla. Muussa tapauksessa jokin aiemmista pisteistä x_i^0 korvataan pisteellä ξ^1, jolloin saadaan seuraavan vaiheen pisteet $\{x_i^1\}$. Poistettava piste valitaan siten, että korvauksen jälkeen virheen etumerkki vaihtelee edelleen peräkkäisissä pisteissä (vaikka virhe ei tietenkään enää ole kaikissa pisteissä itseisarvoltaan yhtä suuri).

Parannettu iteraatti saadaan aikaiseksi etsimällä polynomi $\tilde{f}^1$, jolle virhe $f(x_i^1) - \tilde{f}^1(x_i^1) = \pm\rho^1$. Jälleen virheen etumerkin tulee vaihtua aina siirryttäessä pisteestä x_i^1 pisteeseen x_{i+1}^1. Kuten edellä, tämä johtaa $n+2$ yhtälöön $n + 2$ tuntemattomalle (uudet pisteet $\{x_i^1\}$ ja ρ^1), joista pystytään määräämään $\tilde{f}^1$ ja ρ^1.

Iteraatio jatkuu tästä eteenpäin edellä kuvatulla tavalla: paikallistetaan virheen maksimikohta ξ^2, suoritetaan korvaus ja muodostetaan uusi iteraatti uuden pisteistön avulla. Jos jollain kierroksella ξ^{k+1} on jokin pisteistä x_i^k,

approksimaatiota ei voi enää parantaa. Tätä tilannetta ei välttämättä saavuteta äärellisellä määrällä iteraatioita, joten algoritmi täytyy varatua keskeyttämään muillakin ehdoilla. Suppenemiskriteerinä voi vertailla esimerkiksi iteraatiokierroksen k virhettä ρ^k ja maksimivirhettä $\|f - \tilde{f}^k\|$.

Esimerkki 4.5.2 Etsimme Remezin menetelmällä parhaan tasaisen approksimaation 1. asteen polynomien joukosta funktiolle $f(x) = x^2$ välillä $[0, 2]$ (vertaa edelliseen esimerkkiin). Valitaan lähtöpisteiksi $\{x_i^0\} = \{0, 1/2, 2\}$, jolloin ensimmäiseen approksimaatioon $\tilde{f}^0 = c_0^0 + c_1^0 x$ liittyvät parametrit selviävät alternoivan virheen mukaisesti yhtälöryhmästä

$$\begin{array}{rcl} -c_0^0 & = & \rho^0 \\ \frac{1}{4} - c_0^0 - \frac{1}{2}c_1^0 & = & -\rho^0 \\ 4 - c_0^0 - 2c_1^0 & = & \rho^0 \end{array} \quad \Longrightarrow \quad \left\{ \begin{array}{rcl} c_0^0 & = & -\frac{3}{8} \\ c_1^0 & = & 2 \\ \rho^0 & = & \frac{3}{8} \end{array} \right.$$

Ensimmäinen iteraatio on näin ollen $\tilde{f}^0(x) = 2x - 3/8$. Virheen itseisarvo

$$|f - \tilde{f}^0| = |x^2 - 2x + 3/8|$$

saavuttaa maksimiarvonsa $5/8$ pisteessä $\xi^1 = 1$. Koska $5/8 > 3/8 = \rho^0$, jatkamme iterointia. Erotus $f - \tilde{f}^0$ on päätepisteissä $x_1^0 = 0$ ja $x_3^0 = 2$ positiivinen, keskimmäisessä pisteessä $x_2^0 = 1/2$ ja uudessa pisteessä $\xi^1 = 1$ puolestaan negatiivinen. Etumerkkien vuorottelu säilyy, kun vaihdamme pisteen x_2^0 pois ja otamme sen tilalle pisteen ξ^1. Seuraavan iteraation pohjaksi jäävät siis pisteet $\{x_i^1\} = \{0, 1, 2\}$. Uusi approksimaatio $\tilde{f}^1 = c_0^1 + c_1^1 x$ saadaan selville yhtälöistä

$$\begin{array}{rcl} -c_0^1 & = & \rho^1 \\ 1 - c_0^1 - c_1^1 & = & -\rho^1 \\ 4 - c_0^1 - 2c_1^1 & = & \rho^1 \end{array} \quad \Longrightarrow \quad \left\{ \begin{array}{rcl} c_0^1 & = & -\frac{1}{2} \\ c_1^1 & = & 2 \\ \rho^1 & = & \frac{1}{2} \end{array} \right.$$

Olemme siis päässeet tulokseen $\tilde{f}^1 = 2x - 1/2$. Tämän approksimaation maksimivirhe $\|x^2 - 2x + 1/2\|_\infty = 1/2$ saavutetaan kuitenkin kaikissa pisteissä $\{x_i^1\}$, joten Remezin algoritmi suppeni tässä yksinkertaisessa tapauksessa alkuarvauksen jälkeen yhdellä lisäiteraatiolla.

4.5.2 Tšebyševin polynomit

Tasaisen approksimaation muodostamisessa on usein hyötyä *Tšebyševin polynomeista*. Niistä on helppoa muodostaa polynomeja, jotka ovat lähellä annetun funktion L_∞-approksimaatiota. Ne ovat siis tavallaan approksimaation approksimaatioita. Lisäksi Tšebyševin polynomien avulla saadaan hyviä lähtöpisteitä edellä kuvatulle Remezin algoritmille.

Tarkastellaan aluksi seuraavaa rajoitettua ongelmaa: on löydettävä paras tasainen approksimaatio $\tilde{f}(x)$ monomille x^n välillä $[-1, 1]$ polynomien $\mathcal{P}_{n-1}$ joukosta. On siis määrättävä kertoimet $c_0, c_1, \ldots, c_{n-1}$ siten, että

$$\underset{c_0, c_1, \ldots, c_{n-1}}{\text{minimi}} \quad \max_{x \in [-1,1]} |x^n - (c_{n-1}x^{n-1} + \ldots + c_1 x + c_0)|.$$

Tämän tehtävän yleinen ratkaisu on

$$\tilde{f}(x) = x^n - \hat{T}_n(x),$$

missä polynomit $\hat{T}_n(x)$ on saatu skaalauksella

$$\hat{T}_n(x) = \frac{1}{2^{n-1}} T_n(x), \quad T_n(x) = \cos(n \arccos(x)).$$

Polynomit $T_n(x)$ ovat (1. lajin) Tšebyševin polynomeja. Trigonometrisen määrittelyn kautta niille saadaan rekursiivinen kaava

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x).$$

Alla on muutama ensimmäinen Tšebyševin polynomi:

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_2(x) &= 2x^2 - 1 \\ T_3(x) &= 4x^3 - 3x \\ T_4(x) &= 8x^4 - 8x^2 + 1. \end{aligned}$$

Tasaisen approksimoinnin kannalta Tšebyševin polynomit tekee mielenkiintoisiksi niiden *minimaalisuusominaisuus:* kaikkien muotoa (korkeimman asteen termin kerroin 1)

$$x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$$

olevien polynomien joukosta Tšebyševin polynomilla $\hat{T}_n(x)$ on välillä $[-1, 1]$ pienin L_∞-normi. Itse asiassa pätee $\|\hat{T}_n\|_\infty = 1/2^{n-1}$. Tällaisia polynomifunktioita on siis yleisesti mahdotonta saada koko välillä $[-1, 1]$ tasaisesti mielivaltaisen pieneksi.

Tšebyševin polynomit muodostavat välillä $[-1, 1]$ täydellisen ortogonaalisen systeemin painofunktionaan $w(x) = 1/\sqrt{1 - x^2}$:

$$\int_{-1}^{1} T_n(x) T_l(x) / \sqrt{1 - x^2}\, dx = 0, \quad n \neq l$$

$$\int_{-1}^{1} T_n^2(x) / \sqrt{1 - x^2}\, dx = \begin{cases} \pi & n = 0 \\ \dfrac{\pi}{2} & n \neq 0. \end{cases}$$

Mikä tahansa jatkuva funktio $f(x) \in C[-1, 1]$ on esitettävissä Tšebyševin polynomien sarjakehitelmänä. Tämän sarjan osasummat $\tilde{f}_n(x)$ approksimoivat funktiota $f(x)$ normin

$$\|f\|_w = \left(\int_{-1}^{1} \frac{f^2(x)}{\sqrt{1 - x^2}}\, dx \right)^{1/2}$$

mielessä. Voidaan kuitenkin osoittaa, että jos $f(x) \in C^2[-1, 1]$, niin sarjat lähestyvät funktiota $f(x)$ myös L_∞-normissa.

Kehitelmä on kirjoitettavissa muodossa

$$\tilde{f}_n(x) = \frac{\alpha_0}{2} + \sum_{i=1}^{n} \alpha_i T_i(x),$$

jolloin kertoimet α_i lasketaan integraaleista

$$\alpha_i = \frac{2}{\pi} \int_{-1}^{1} f(x) T_i(x) \frac{dx}{\sqrt{1-x^2}}.$$

Koska $\|T_i(x)\|_\infty = 1$, approksimaatiovirhe toteuttaa yhtälön

$$|f(x) - \tilde{f}_n(x)| = \left| \sum_{i=n+1}^{\infty} \alpha_i T_i(x) \right| \leq \sum_{i=n+1}^{\infty} |\alpha_i| .$$

Toisaalta kertoimille α_i pätee $|\alpha_i| \leq c/i^2$ (vakio c riippuu funktiosta f), ja tunnetusti $\sum_{i=1}^{\infty} 1/i^2 = \pi^2/6 < \infty$, joten funktion Tšebyševin sarjakehitelmä approksimoi funktion tasaista approksimaatiota.

Esimerkki 4.5.3 Laskemme funktiolle $f(x) = x^2$ vielä 1. asteen Tšebyševin sarjakehitelmän välille $[0, 2]$. Koska Tšebyševin polynomit on määritelty välillä $[-1, 1]$, joudumme ensin skaalaamaan muuttujaa. Jos merkitsemme välin $[-1, 1]$ muuttujaa symbolilla x ja välin $[0, 2]$ muuttuja on t, niin näille on voimassa yhteys $x = t - 1$. Yleisesti: jos t on välillä $[a, b]$, niin $x = 2(t-1)/(b-a) - 1$. Nyt siis kaksi ensimmäistä Tšebyševin polynomia ovat $T_0(t) = 1$ ja $T_1(t) = t - 1$.

Sarjakehitelmän kertoimet α_0, α_1 saadaan integraaleista

$$\alpha_0 = \frac{2}{\pi} \int_0^2 \frac{T_0(t) f(t)}{\sqrt{1-(t-1)^2}} dt = \frac{2}{\pi} \int_0^2 \frac{1 \cdot t^2}{\sqrt{2t - t^2}} dt = 3,$$

$$\alpha_1 = \frac{2}{\pi} \int_0^2 \frac{T_1(t) f(t)}{\sqrt{1-(t-1)^2}} dt = \frac{2}{\pi} \int_0^2 \frac{(t-1) t^2}{\sqrt{2t - t^2}} dt = 2.$$

Lopputulos on siis $\tilde{f}_1(t) = \frac{3}{2} + 2(t-1) = 2t - \frac{1}{2}$ eli päädyimme jälleen samaan tuloksen kuin aiemmissa esimerkeissä. Tämän ei pitäisi olla yllätys, koska $f(t) = t^2$ on polynomi, jonka polynomisarja on tietysti äärellinen.

Kertoimia α_i ei yleensä pystytä laskemaan analyyttisesti vaan integraalit joudutaan evaluoimaan numeerisesti.

Tšebyševin polynomit tuovat helpotusta myös kysymykseen kuinka Remezin algoritmissa kannattaa valita iteroinnin aloituspisteistö $\{x_i^0\}$. Tšebyševin polynomi T_n nimittäin saavuttaa ääriarvonsa $n+1$ eri pisteessä, ja lisäksi maksimit ja minimit vuorottelevat. Tšebyševin polynomin T_{n+1} ääriarvopisteet tarjoavat näin käytännössä erinomaisen lähtökohdan iteroinnin käynnistämiseen. Usein riittävän hyvä tasainen approksimaatio syntyy jo ensimmäisessä iteraatiossa.

Toinen mahdollisuus on muodostaa sopivan Tšebyševin polynomin nollakohtien kautta kulkeva interpolaatiofunktio.

Esimerkki 4.5.4 Välille $[0, 2]$ skaalattu 2. asteen Tšebyševin polynomi on $2(t-1)^2 - 1 = 2t^2 - 4t + 1$. Tämän nollakohdat ovat $1 \pm 1/\sqrt{2}$ ja ääriarvokohdat $\{0, 1, 2\}$. Esimerkissä 4.5.1 pääteltiin, että virhe $t^2 - \tilde{f}(t)$ saa ääriarvonsa pisteissä $\{0, 1, 2\}$. Toisaalta taas lineaarinen interpolaatio pisteiden $(1 - 1/\sqrt{2},\ 3/2 - \sqrt{2})$ ja $(1 + 1/\sqrt{2},\ 3/2 + \sqrt{2})$ kautta tuottaa jälleen kerran polynomin $2t - 1/2$.

4.6 Rationaalinen approksimointi

Edellisissä kappaleissa approksimointiin on käytetty polynomeja. Niillä on kuitenkin vaikea mallittaa tilannetta, jossa approksimoitava funktio kasvaa tai vähenee nopeasti esimerkiksi singulariteetin lähellä. Tällaisessa tapauksessa polynomit pyrkivät heilahtelemaan voimakkaasti. Eräs keino on turvautua paloittaisiin polynomiapproksimaatioihin, mutta jos jostain syystä on tarpeen käyttää yhtä approksimaatiota koko laskenta-alueessa, *rationaalifunktio* voi olla sopiva lähestymistapa. Rationaalifunktiolla tarkoitetaan kahden polynomin osamäärää.

Rationaalinen approksimaatio eroaa polynomista asymptoottisessa käyttäytymisessä. Polynomin $p(x)$ arvot kasvavat tai vähenevät rajatta aina, kun argumentti $x \to \pm\infty$. Sen sijaan rationaalinen approksimaatio voi lähestyä asymptoottisesti jotain äärellistä vakiota c.

Funktion $f(x)$ rationaalinen approksimaatio $\tilde{f}(x)$ on siis kahden polynomin $p_n(x) \in \mathcal{P}_n$ ja $q_m(x) \in \mathcal{P}_m$ osamäärä:

$$\tilde{f}(x) = \frac{p_n(x)}{q_n(x)} = \frac{a_n x^n + \ldots + a_0}{b_m x^m + \ldots + b_1 x + 1}.$$

Tässä polynomit on jaettu samalla luvulla siten, että nimittäjän vakiotermiksi jää 1.

Oleellinen kysymys on luonnollisesti kuinka kertoimet $\{a_i\}, \{b_j\}$ määritetään, mikä puolestaan riippuu käytetystä normista. Sekä L_2-normin että L_∞-normin valinta johtaa hankalaan minimointitehtävään, johon on vaikea kehittää yleistä teoriaa. Niinpä yleensä käytetään *Padén approksimaatiota*, joka on *Taylorin sarjan* yleistys rationaalifunktioille.

Oletetaan, että approksimoitavalla funktiolla $f(x)$ on Taylorin kehitelmä (tässä kehityskeskuksena origo)

$$f(x) = \sum_{i=0}^{N} c_i x^i + \frac{f^{(N+1)}(\xi)}{(N+1)!} x^{N+1}, \quad c_i = \frac{f^{(i)}(x)}{i!}.$$

Sarja on kirjoitettu katkaistuun muotoon, jossa astetta N olevaan polynomiin on liitetty pisteessä ξ laskettu virhetermi. Etsimme Padén approksimaation, joka täsmää Taylorin kehitelmän polynomiosan $\sum_{i=0}^{N} c_i x^i$ kanssa mahdollisimman hyvin. Tarkemmin sanottuna funktiota $f(x)$ approksimoivan osamäärän osoittaja $p_n(x)$ ja nimittäjä $q_m(x) \neq 0$ muodostetaan vaa-

timalla, että erotus $f(x) - p_n(x)/q_m(x)$ häviää identtisesti kaikilla kyseeseen tulevilla muuttujan x arvoilla mahdollisimman korkeaan kertalukuun N asti. Siten lausekkeessa

$$(c_N x^N + ...c_1 x + c_0)(b_m x^m + ... + b_1 x + 1) - (a_n x^n + ... + a_1 x + a_0)$$

asetetaan monomien x^i kertoimet nolliksi mahdollisimman monella eksponentin i arvolla alkaen arvosta $i = 0$. Tämä johtaa lineaariseen yhtälöryhmään lukujen $\{a_i\}$ ja $\{b_j\}$ välille.

Esimerkki 4.6.1 Laskemme funktiolle

$$f(x) = \arctan(x) = x - x^3/3 + x^5/5 - x^7/7 + \ldots$$

muotoa

$$\tilde{f}(x) = p_2(x)/q_2(x) = (a_2x^2 + a_1x + a_0)/(b_2x^2 + b_1x + 1)$$

olevan Padén approksimaation. Kirjoitamme auki erotuksen $E = f(x)q_2(x) - p_2(x)$:

$$\begin{aligned} E &= (x - \frac{1}{3}x^3 + \frac{1}{5}x^5 - \frac{1}{7}x^7 + ..)(b_2x^2 + b_1x + 1) - (a_2x^2 + a_1x + a_0) \\ &= (0 - a_0) + (1 - a_1)x + (b_1 - a_2)x^2 + (b_2 - \frac{1}{3})x^3 - \frac{b_1}{3}x^4 + \ldots. \end{aligned}$$

Vaadimme seuraavaksi monomien x^i kertoimien häviämistä, jolloin päädymme yhtälöihin

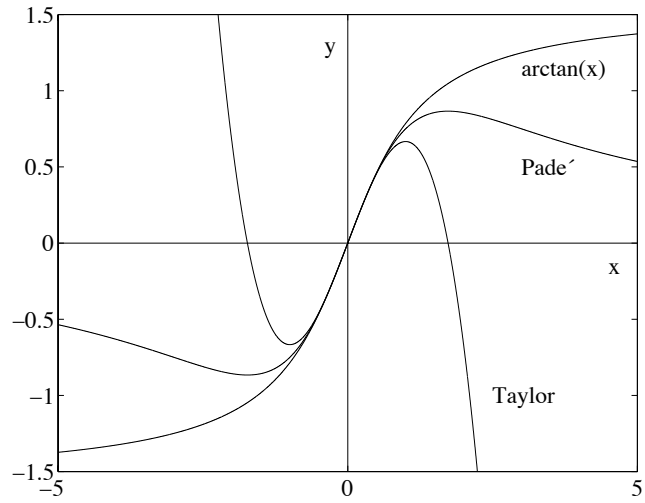
$$\begin{aligned} a_0 &= 0 \\ a_1 &= 1 \\ b_1 &= a_2 \\ b_2 &= \frac{1}{3} \\ b_1 &= 0. \end{aligned}$$

Tuntemattomien arvot ovat nyt lähes suoraan luettavissa. Approksimaatioksi tuli siis

$$\tilde{f} = \frac{x}{\frac{1}{3}x^2 + 1}.$$

Kuvassa 4.5 näemme arctan-funktion ja tässä lasketun Padén approksimaation lisäksi vastaavaa kertalukua olevan Taylorin sarjan $T_3(x) = x - x^3/3$. Molemmat approksimaatiot ovat hyvin tarkkoja origon ympäristössä, mutta heikkenevät nopeasti etäännyttäessä origosta. Suurilla argumentin arvoilla Padén approksimaatio lähestyy raja-arvoa 0, mutta Taylorin sarja hajaantuu. Oikea asymptoottinen arvo on

$$\lim_{x \to \pm\infty} \arctan x = \pm\frac{\pi}{2}.$$



Kuva 4.5: *Funktion* arctan *x Padén approksimaatio ja vastaava katkaistu Taylorin sarja.*

4.7 Interpolointi

Diskreetissä approksimoinnissa sovitetaan funktio $\tilde{f}$ annettuun dataan siten, että $\tilde{f}$ jollain normilla mitattuna on lähellä kaikkia datapisteitä. Jos funktion $\tilde{f}$ pitää kulkea täsmälleen datapisteiden kautta, kyseessä on *interpolointi*. Interpolointitehtävässä yritetään arvioida, miten tuntematon funktio f käyttäytyy tunnettujen datapisteiden välillä. Tämän vuoksi interpolointia käytetään kokeellisten tulosten analysointiin ja mahdollisesti myös mallien muodostukseen ja sitä kautta ennustuksiin.

Interpolointi on pohja myös useille muille numeriikan osa-alueille. Monet käytännössä esiintyvistä numeerisista integrointimenetelmistä (katso lukua 5) perustuvat interpolointiin. Samoin osittaisdifferentiaaliyhtälöiden ratkaisuun tarkoitettu elementtimenetelmä (luku 8) hyödyntää interpolointia niin suoranaisessa laskennassa kuin teoreettisissa tarkasteluissakin.

Seuraavassa keskitymme aluksi laajemmin yhden muuttujan funktioiden interpolointiin. Kappaleessa 4.13 esittelemme joitakin menetelmiä, jotka soveltuvat usean muuttujan funktioiden interpolointiin.

Muotoillaan yhden muuttujan funktion interpolointitehtävä. Oletetaan, että käytössä on joukko lineaarisesti riippumattomia, jatkuvia funktioita $\{\phi_0, \ldots, \phi_n\}$, jolla on seuraava ominaisuus: millä tahansa funktiolla, joka on muotoa $\sum_{i=0}^{n} c_i\phi_i$, on välillä (a,b) enintään n erillistä nollakohtaa. Tällaisen joukon muodostavat esimerkiksi monomit $\{x^i\}$ mielivaltaisella välillä $[a,b]$ tai trigonometriset funktiot $\{1, \sin(kx), \cos(kx)\}$ puoliavoimella välillä $[0, 2\pi)(k = 1, \ldots, (n-1)/2$, ja n on pariton). Lisäksi on annettu $n+1$ datapistettä $(x_0, y_0), \ldots, (x_n, y_n)$, $x_i \neq x_j$, kun $i \neq j$. On löydettävä funktio

$\tilde{f} = \sum_{i=0}^{n} c_i \phi_i$, jolle pätee $\tilde{f}(x_i) = y_i,\ i = 0, ..., n$.

Tässä esitetyssä muodossa interpolointitehtävällä on aina yksikäsitteinen ratkaisu. Periaatteessa kertoimet $\{c_i\}$ olisi mahdollista ratkaista normaaliyhtälöistä kuten approksimoinnin yhteydessä, mutta käytännössä on helpompaa ja tehokkaampaa soveltaa suoraviivaisempia menetelmiä. Eri menetelmät voivat tuottaa lausekkeita, jotka esitysmuodoista johtuen usein näyttävät erilaisilta, mutta jos vastaukset sievennetään samaan muotoon, havaitaan, että ratkaisu todella on yksikäsitteinen. Interpoloinnin luonteeseen kuuluu, että *interpolantti* $\tilde{f}(x)$ on käyttökelpoinen vain pisteiden $a = \min\{x_i\}$ ja $b = \max\{x_i\}$ välillä. Jos sitä sovelletaan välin $[a, b]$ ulkopuolisiin pisteisiin, kyseessä on *ekstrapolointi*.

Interpoloinnin yhteydessä voidaan pohtia myös optimaalisen interpolointipisteistön valinnan ongelmaa, joka sisältää edellisen tehtävän osanaan. Jos tunnetaan funktio f ja interpoloivat kantafunktiot $\{\phi_i(x)\}$, niin kuinka tulisi valita pistejoukko $\{x_i\}$, jotta paras pisteiden $\{(x_i, f(x_i))\}$ kautta asetettu interpolantti $\tilde{f}(x)$ kuvaa mahdollisimman tarkasti funktion $f(x)$ käyttäytymistä jollain välillä $[a, b]$? Jos pisteet $\{x_i\}$ sijoitetaan tasaisin välein interpolointialueelle $[a, b]$, puhutaan *tasavälisestä interpoloinnista*.

4.8 Lagrangen interpolointi

Lagrangen interpoloinnissa muodostetaan ensin kantapolynomit $\{\phi_i(x)\}$, jotka saavat arvon $\phi_i(x_i) = 1$ interpolointipisteessä $x_i \in [a, b]$ ja arvon $\phi_i(x_j) = 0$ kaikissa muissa pisteissä $x_j, j \neq i$. Kantapolynomit voidaan kirjoittaa 1. asteen monomien tulona:

$$\phi_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}. \tag{4.11}$$

Varsinainen interpolantti $\tilde{f}(x)$ on helppo ilmaista polynomien $\phi_i(x)$ avulla:

$$\tilde{f}(x) = \sum_{i=0}^{n} y_i \phi_i(x), \tag{4.12}$$

missä kertoimet $\{y_i\}$ ovat siis interpoloitavan funktion $f(x)$ arvot pisteissä $\{x_i\}$. Mikäli $f(x) \in C^{n+1}$, interpolaatiovirheelle $f(x) - \tilde{f}(x)$ pätee arvio

$$f(x) - \tilde{f}(x) = \frac{\prod_{i=0}^{n}(x - x_i)}{(n+1)!} f^{(n+1)}(\xi_x), \quad x \in [a, b],\ \xi_x \in (a, b).$$

Kaavassa esiintyy funktion $f(x)$ korkean kertaluvun derivaatta tuntemattomassa pisteessä ξ_x ja tulo $y(x) = \prod_{i=0}^{n}(x - x_i)$. Kummankin arvon laskeminen on vähintäänkin työlästä ellei mahdotonta, joten käytännössä joudutaan turvautumaan pessimistiseen yläarvioon

$$|f(x) - \tilde{f}(x)| \leq \frac{\prod_{i=0}^{n}(x - x_i)}{(n+1)!} \|f^{(n+1)}(x)\|_\infty, \tag{4.13}$$

jossa myös tulolauseke $y(x)$ usein korvataan maksimiarvollaan.

Esimerkki 4.8.1 Muodostamme funktiolle $f(x) = e^x$ pisteiden $(-1, 1/e)$, $(0,1)$ ja $(1,e)$ kautta kulkevan 2. asteen Lagrangen interpolantin. Laskemme aluksi kantafunktiot:

$$\begin{aligned}
\phi_0 &= \frac{x-x_1}{x_0-x_1}\frac{x-x_2}{x_0-x_2} = \frac{x-0}{-1-0}\frac{x-1}{-1-1} = \frac{1}{2}(x^2-x)\\
\phi_1 &= \frac{x-x_0}{x_1-x_0}\frac{x-x_2}{x_1-x_2} = \frac{x+1}{0+1}\frac{x-1}{0-1} = 1-x^2\\
\phi_2 &= \frac{x-x_0}{x_2-x_0}\frac{x-x_1}{x_2-x_1} = \frac{x+1}{1+1}\frac{x-0}{1-0} = \frac{1}{2}(x^2+x).
\end{aligned}$$

Lopullinen interpolantti on siis

$$\begin{aligned}
\tilde{f}(x) &= \frac{1}{e}\cdot\frac{1}{2}(x^2-x) + 1\cdot(1-x^2) + e\cdot\frac{1}{2}(x^2+x)\\
&= 2\sinh^2(1/2)x^2 + \sinh(1)x + 1.
\end{aligned}$$

Polynomin $\tilde{f}(x) = c_2x^2 + c_1x + c_0$ kertoimet on mahdollista määrittää tietysti myös suoraan ehdosta $\tilde{f}(x_i) = e^{x_i}$, $i = 0,1,2$.

Kuvassa 4.6 on esitetty funktion e^x ja tässä lasketun, välillä $[-1,1]$ voimassa olevan interpolaation lisäksi toinen kvadraattinen interpolantti

$$\tilde{g}(x) = \frac{1}{2}(x^2-3x+2) + e(2x-x^2) + \frac{e^2}{2}(x^2-x),$$

joka on muodostettu pisteiden $(0,1)$, $(1,e)$ ja $(2,e^2)$ avulla. Kumpikin interpolantti kulkee lähellä eksponenttifunktiota varsinaisella määrittelyalueellaan ($[-1,1]$ tai $[0,2]$), mutta erkanee nopeasti omille teilleen, kun tullaan määrittelyalueen ulkopuolelle.

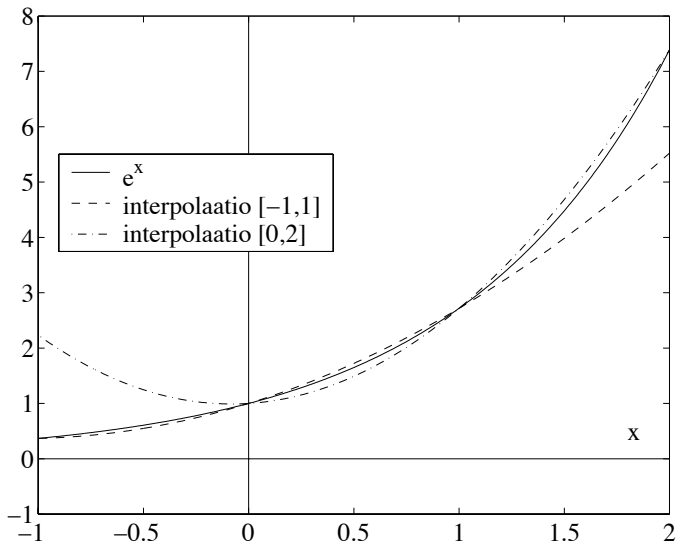
Yritämme vielä arvioida interpolointivirhettä, jos e^x korvataan edellä lasketulla funktiolla $\tilde{f}(x)$ välillä $[-1,1]$. Tulon $|\gamma(x)| = |(x+1)x(x-1)|$ maksimiarvo on $2/3\sqrt{3}$, ja $d^3(e^x)/dx^3 = e^x$, joka saa suurimman arvonsa e välin päätepisteessä $x = 1$. Virheestä voimme siis kaavan 4.13 nojalla luvata, että

$$|e^x - \tilde{f}_2(x)| \le \frac{e}{6}\frac{2}{3\sqrt{3}} \approx 0.1744.$$

Todellisuudessa virhe on suurimmillaankin vain 0.0785.

Interpolointivirheen kaavassa esiintyvä tulo $\gamma(x) = \prod_{i=0}^{n}(x-x_i)$ voidaan tietysti tulkita muuttujan x n-asteiseksi polynomiksi. Kappaleessa 4.5.2 totesimme, että Tšebyševin polynomeilla $T_n(x)$ on kaikista polynomeista pienin L_∞-normi. Näin ollen tulon $\gamma(x)$ maksimiarvo saadaan mahdollisimman pieneksi, jos interpolointipisteiksi $\{x_i\}$ valitaan Tšebyševin polynomin nollakohdat (lyhyemmin *Tšebyševin pisteet*). Välillä $[-1,1]$ nämä saadaan projisoimalla yksikköympyrän kehän ylemmälle puolikkaalle tasavälein sijoitetut pisteet x-akselille: $x_i = \cos\frac{2i+1}{n+1}\frac{\pi}{2}$. Muille väleille vastaavat pisteet selviävät skaalaamalla.

Varsinkin tasavälisessa interpoloinnissa tulo $\gamma(x) = \prod_{i=0}^{n}(x-(a+ih))$ heilahtelee hyvin voimakkaasti, kun pisteiden lukumäärä n kasvaa. Tasavälinen interpolointi tuottaa tosin paljon tarkempia tuloksia interpolointialueen keskellä, jossa Tšebyševin pisteisiin perustuva interpolantti edelleen yltää omaan maksimivirheeseensä.



Kuva 4.6: *Funktio $f(x) = e^x$ ja kaksi kvadraattista interpolanttia.*

Jos virheen mittaamiseen käytetään L_2-normia eikä L_∞-normia, optimaaliset interpolointipisteet ovat vastaavasti Legendren polynomin $L_n(x)$ nollakohdat.

Vaikka interpolointipisteiden valinta tehtäisiin huolellisestikin, niiden lukumäärän kasvattaminen johtaa lähes aina huonompiin tuloksiin jossain vaiheessa. Itse asiassa C. Runge osoitti jo 1901 tutkiessaan funktion $f(x) = 1/(1+x^2)$ interpolointia välillä $[-5, 5]$ (katso esimerkki 4.11.1), että sen tasaväliset interpolantit $\tilde{f}_n(x)$ suppenevat arvon n kasvaessa vain alueessa $|x| < c$, $c \approx 3.63$. Syynä ovat funktion kompleksiset navat pisteissä $x = \pm i$. Reaalimuuttujan reaaliarvoisen funktion f interpolantit $\tilde{f}$ suppenevatkin tasaisesti kohti funktiota f vain siinä tapauksessa, että f on analyyttinen koko kompleksitasossa myös silloin, jos f saa kompleksisia argumentteja.

Interpolointipisteiden lukumäärän kasvattamisen sijasta selvästi parempi keino tarkempien arvioiden tuottamiseen onkin interpolointivälin jakaminen useaan osaan, joista jokaisella käytetään omaa matala-asteista interpolanttia. Tällöin saadaan aikaiseksi *palapolynomi*. Välien rajapisteissä voidaan asettaa jatkuvuusehtoja sekä interpolantille että sen derivaatoille. Jos palapolynomi on sileästi jatkuva osaväliltä toiselle siirryttäessä, kyseessä on *splini*.

Kuten jo aiemmin on todettu, niin pian kuin interpolointipisteet on valittu, niiden kautta kulkeva polynomi on yksikäsitteisesti määritelty. Tästä ei kuitenkaan seuraa, että olisi yhdentekevää kuinka polynomi muodostetaan. Edellä esitelty tapa rakentaa Lagrangen interpolantti apufunktioiden $\phi_i(x)$ avulla on teoreettisesti kätevä esimerkiksi virhetarkastelujen kannalta, mutta jos interpolointipisteistöön halutaan lisätä uusi piste, koko interpolantti joudutaan muodostamaan alusta asti uudelleen. Tässä tilanteessa *Aitkenin*

ja Nevillen algoritmi on paljon kätevämpi. Se perustuu seuravaan ideaan: jos funktiosta f tunnetaan pisteiden $x_l, \ldots x_{l+m}$ kautta kulkeva interpolantti $\tilde{f}_1 \in \mathcal{P}_m$ ja toisaalta interpolantti $\tilde{f}_2 \in \mathcal{P}_m$, joka perustuu yhtä pistettä lukuunottamatta identtiseen pisteistöön $x_{l+1}, \ldots, x_{l+m+1}$, niin uusi, funktiota f kaikissa pisteissä $x_l, \ldots, x_{l+m+1}$ interpoloiva polynomi $q \in \mathcal{P}_{m+1}$ saadaan lausekkeesta

$$q(x) = \frac{(x - x_l)p_2(x) - (x - x_{l+m+1})p_1(x)}{x_{l+m+1} - x_l}. \tag{4.14}$$

Interpolantin laskeminen voidaan tämän tuloksen avulla järjestää kaavioksi, jossa ensin muodostetaan nollannen asteen polynomit eli vakiot y_i, sitten lineaariset polynomit jotka kulkevat aina kahden vierekkäisen interpolointipisteen kautta jne.

Esimerkki 4.8.2 Laskemme esimerkissä 4.8.1 selvitetyn funktion $f(x) = e^x$ kvadraattisen interpolantin (välillä $[-1, 1]$) Aitkenin ja Nevillen algoritmin avulla.

x_i	$(x - x_i)$	$f(x_i)$	$\tilde{f} \in \mathcal{P}_1$	$\tilde{f} \in \mathcal{P}_2$
-1	$(x+1)$	$1/e$		
0	x	1	$x + 1 - x/e$	
1	$(x-1)$	e	$xe - x + 1$	$\big((x+1)(xe - x + 1)$ $-(x-1)(x+1-x/e)\big)/2$

Jos nyt haluamme lisätä interpolaatioon uusia pisteitä, riittää, että laskemme taulukkoon yhden uuden rivin.

Jos emme ole kiinnostuneita interpolaatiopolynomin varsinaisesta funktionaalisesta muodosta vaan vain polynomin arvosta jossain pisteessä $x = \xi$, edellisessä kaaviossa voi muuttujan x paikalle sijoittaa kaikkialle suoraan luvun ξ.

4.9 Hermiten interpolointi

Lagrangen interpoloinnissa jokaiseen interpolointipisteeseen x_i liittyy täsmälleen yksi ehto $\tilde{f}(x_i) = y_i$, missä y_i on joko tunnetun funktion $f(x)$ arvo pisteessä x_i tai sitten muutoin hankittu lukuarvo (esimerkiksi mittaustulos). *Hermiten interpoloinnissa* yksittäisessä pisteessä x_i voidaan määritellä myös funktion derivaattoja koskevia ehtoja

$$\tilde{f}'(x_i) = y_i^{(1)}, \ldots, \tilde{f}^{(k_i)}(x_i) = y_i^{(k_i)},$$

missä luvut $y_i^{(j)}$ ovat nyt interpoloitavan funktion $f(x)$ ja sen derivaattojen $f^{(j)}(x)$ arvoja tai niitä approksimoivia lukuja. Yhteensä ehtoja on siis $N+1 = \sum_{i=0}^{n}(k_i+1)$ kappaletta, jolloin interpolaatiopolynomi $\tilde{f}(x)$ on astetta N. Hermiten interpoloinnissa ei sallita, että jonkin alemman kertaluvun

derivaatan arvo jäisi spesifioimatta; jos jossain pisteessä halutaan määrätä $\tilde{f}^{(k_i)}$, niin myös kaikki derivaatat $\tilde{f}^{(j)}, j < k_i$ on kiinnitettävä. Yleisempi tehtävä, jossa voi vapaasti valita minkä kertalukujen derivaatat otetaan interpolointiin mukaan, tunnetaan nimellä *Birkhoffin interpolointi*.

Periaatteessa Hermiten interpolantin muodostaminen voi tapahtua Lagrangen interpolaation tapaan apufunktioiden $\phi_{i,j}(x)$ kautta. Apufunktioilla on nyt kaksi indeksiä, joista toisella pidetään kirjaa interpolointipisteestä ja toisesta selviää minkä kertaluvun derivaattaan apufunktio liittyy. Apufunktioiden tulee toteuttaa ehdot ($l = 0, ..., n,\ k = 0, ..., k_l$)

$$\phi_{i,j}^{(k)}(x_l) = \begin{cases} 1, & \text{jos } i = l \text{ ja } j = k, \\ 0, & \text{muulloin.} \end{cases}$$

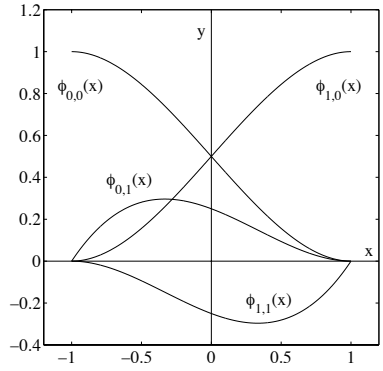
Toisin kuin Lagrangen interpoloinnissa, tässä ei ole annettu suoraa kaavaa apufunktioiden muodostamiseksi, vaan ainoastaan ehdot, joiden avulla apufunktioiden muoto on pääteltävissä.

Kun apufunktiot on tavalla tai toisella laskettu, Hermiten interpolantti saadaan kaksinkertaisena summana

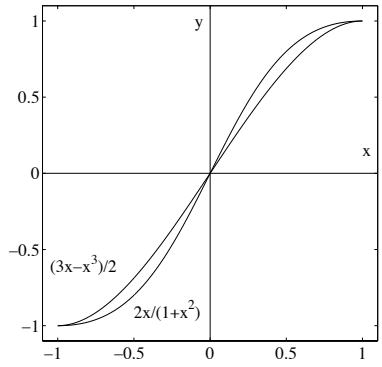
$$\tilde{f}(x) = \sum_{i=0}^{n} \sum_{j=0}^{k_i} y_i^{(j)} \phi_{i,j}(x).$$

Hermiten interpoloinnille on johdettavissa samankaltainen virhelauseke kuin Lagrangen interpolaatiollekin:

$$f(x) - \tilde{f}(x) = \frac{\prod_{i=0}^{n}(x - x_i)^{k_i}}{(N+1)!} f^{(N+1)}(\xi), \quad \xi \in [a, b].$$



a: *Apufunktiot* $\phi_{i,j}(x)$.



b: *Funktio* $f(x) = 2x/(1 + x^2)$ *ja sen interpolantti välillä* $[-1, 1]$.

Kuva 4.7: *Hermiten interpolointi.*

Esimerkki 4.9.1 Etsimme välillä $[-1,1]$ rationaalifunktion $f(x) = 2x/(1+x^2)$ Hermiten interpolantin $\tilde{f}(x)$, joka siis toteuttaa ehdot

$$\tilde{f}(x) = f(x), \quad \tilde{f}'(x) = f'(x)$$

kummassakin päätepisteessä $x_0 = -1$, $x_1 = 1$. Haemme ensin apufunktiot $\phi_{i,j}(x)$. Funktion $\phi_{0,0}(x)$ pitäisi toteuttaa seuraavat neljä ehtoa:

$$\begin{aligned} \phi_{0,0}(-1) &= 1, & \phi'_{0,0}(-1) &= 0, \\ \phi_{0,0}(1) &= 0, & \phi'_{0,0}(1) &= 0. \end{aligned}$$

Kahden jälkimmäisen ehdon nojalla piste $x_1 = 1$ on siis funktion $\phi_{0,0}(x)$ kaksinkertainen nollakohta, joten teemme yritteen $\phi_{0,0}(x) = r(x)(x-1)^2$, missä $r(x) = ax + b$ on lineaarinen polynomi, jonka kertoimet saadaan kahdesta ensimmäisestä ehdosta:

$$\phi_{0,0}(-1) = (a(-1)+b)(-1-1)^2 = 4(b-a) = 1$$

$$\phi'_{0,0}(-1) = a(-1-1)^2 + (a(-1)+b)\cdot 2(-1-1) = 4(2a-b) = 0.$$

Ehdoista seuraa $a = \frac{1}{4}$, $b = \frac{1}{2}$ ja $\phi_{0,0}(x) = \frac{1}{4}(x+2)(x-1)^2$. Vastaavasti voimme johtaa tulokset

$$\phi_{0,1}(x) = \frac{1}{4}(x+1)(x-1)^2,$$

$$\phi_{1,0}(x) = \frac{1}{4}(2-x)(x+1)^2,$$

$$\phi_{1,1}(x) = \frac{1}{4}(x-1)(x+1)^2.$$

(Katso kuvaa 4.7a.) Tarvittavat kertoimet ovat $y_0^{(0)} = f(-1) = -1$, $y_0^{(1)} = f'(-1) = 0$, $y_1^{(0)} = f(1) = 1$ ja $y_1^{(1)} = f'(1) = 0$, joten

$$\begin{aligned} \tilde{f}(x) &= (-1)\cdot\tfrac{1}{4}(x+2)(x-1)^2 + 0\cdot\tfrac{1}{4}(x+1)(x-1)^2 \\ &\quad + 1\cdot\tfrac{1}{4}(2-x)(x+1)^2 + 0\cdot\tfrac{1}{4}(x-1)(x+1)^2 \\ &= \tfrac{1}{2}(3x - x^3). \end{aligned}$$

Kuvassa 4.7b on esitetty sekä funktio $f(x) = 2x/(1+x^2)$ että sen Hermiten interpolantti $\tilde{f}(x) = \frac{1}{2}(3x - x^3)$.

Arvioimme lopuksi virhekaavan avulla kuinka suureksi virhe $f(x) - \tilde{f}(x)$ voi kasvaa. Funktion $f(x)$ 3. derivaatta on $f^{(3)}(x) = -12(x^4 - 6x + 1)/(1+x^2)^4$, jonka itseisarvo on suurimmillaan 12 (pisteessä $x = 0$). Tulolauseke $y(x) = (x+1)^2(x-1)^2$ saavuttaa maksimiarvonsa $y = 1$ niin ikään origossa, joten virhekaavan mukaan

$$|f(x) - \tilde{f}(x)| \le \frac{1}{4!}12 = 1/2,$$

mikä on selvästi liian pessimistinen arvio. Todellisuudessa virhe on suurimmillaan pisteissä $x = \pm\sqrt{2/\sqrt{3} - 1} \approx 0.3933$, jolloin funktion ja interpolaation välinen ero on ≈ 0.1217.

4.10 Trigonometrinen approksimointi ja interpolointi

Erityisesti signaalinkäsittelyssä joudutaan usein tekemisiin jaksollisten funktioiden kanssa. Tutustumme seuraavaksi lyhyesti jaksollisten yhden muuttujan funktioiden approksimointiin ja interpolointiin. Funktio $f(x)$ on *jaksollinen* , jos on olemassa jokin luku p, jolle pätee $f(x+p) = f(x)$ kaikilla x. Kaikki luvun p kokonaislukumonikerrat tietysti toteuttavat saman ehdon, joten funktion f jaksonpituudeksi sanotaan pienintä lukua p, jolle mainittu ehto on voimassa. Skaalaamalla funktion argumentti sopivasti jaksonpituudeksi voidaan aina valita 2π, joten jatkossa keskitymme tarkastelemaan välillä $[-\pi, \pi]$ määriteltyjä funktioita.

Jaksollisen funktion approksimoinnissa ja interpoloinnissa valitaan kantafunktioiksi $\{\phi_j\}$ yleensä joko trigonometriset funktiot $\{1, \cos(jx), \sin(jx)\}$, $j = 1, 2, 3, \ldots$, tai näiden kanssa ekvivalentti eksponenttifunktioiden joukko $\{e^{ijx}\}$, $j = \ldots, -2, -1, 0, 1, 2, \ldots$. Tässä kappaleessa käytämme matemaattista symbolia i merkitsemään ainoastaan imaginääriyksikköä: $i = +\sqrt{-1}$.

Kutsumme jatkossa muotoa

$$q(x) = \sum_{j=0}^{n} a_j \cos(jx) + \sum_{j=1}^{n} b_j \sin(jx)$$

olevia lausekkeita *trigonometrisiksi polynomeiksi.* Trigonometrinen polynomi $q(x)$ on astetta n jos ainakin toinen kertoimista a_n, b_n poikkeaa nollasta. Kaikkien astetta n olevien trigonometristen polynomien joukkoa merkitsemme symbolilla T_n.

Tarkastelemme aluksi L_2-approksimointia eli pienimmän neliön approksimointia. Funktion f *Fourier'n sarja* on

$$f(x) = \frac{a_0}{2} + \sum_{j=1}^{\infty} a_j \cos(jx) + b_j \sin(jx). \tag{4.15}$$

Koska funktiot $\{1, \cos(jx), \sin(jx)\}$, $j = 1, 2, 3, \ldots$ muodostavat ortogonaalisen joukon, Fourier'n sarjassa esiintyvät kertoimet a_j, b_j saadaan ratkaistua yhtälöistä (4.2). Käyttämällä hyväksi tuloksia

$$\int_{-\pi}^{\pi} \cos^2(jx)\, dx = \int_{-\pi}^{\pi} \sin^2(jx)\, dx = \pi,$$

kun $j \neq 0$, ja

$$\int_{-\pi}^{\pi} 1\, dx = 2\pi$$

voimme kirjoittaa kertoimille lausekkeet

$$a_j = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(jx)\, dx,$$
$$b_j = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(jx)\, dx. \tag{4.16}$$

Myös vakiotermin kerroin a_0 ratkeaa oikein, koska tekijä 2 on otettu huomioon jo Fourier'n sarjan muodossa (4.15).

Jaksollisen funktion esittämistä kokonaisten aaltolukujen j avulla sarjan (4.15) tapaan sanotaan myös *Fourier'n synteesiksi*. Vastaavasti kertoimien $a_j$ja b_j määrittämistä kutsutaan *Fourier'n analyysiksi.*

Joillakin muuttujan x arvoilla kertoimista a_j ja b_j saattaa tulla niin yksinkertaisia lausekkeita, että koko sarjan (4.15) summa on laskettavissa analyyttisesti. Käytännössä joudumme tietysti usein katkaisemaan sarjan ja tyytymään äärellisen määrän termejä sisältävään trigonometriseen polynomiin. Merkitsemme jatkossa symbolilla $\hat{f}_n(x)$ funktion $f(x)$ katkaistua Fourier'n sarjaa, jossa on laskettu yhteen kaikki termit sarjan alusta aina termeihin $a_n \cos(nx) + b_n \sin(nx)$ asti.

Jaksollisen funktion Fourier'n sarjalle pätee Parsevalin yhtälö (4.3), joten ottamalla sarjaan mukaan riittävästi termejä saamme approksimaatiovirheen

$$\left(\int_{-\pi}^{\pi} \left(f(x) - \hat{f}_n(x) \right)^2 dx \right)^{1/2}$$

mielivaltaisen pieneksi. Mittaamme tässä nimenomaan L_2–virhettä eli keskimääräistä virhettä. Yksittäisessä pisteessä x_0 virhe $|f(x_0) - \hat{f}_n(x_0)|$ voi pysyä suurena, vaikka sarjaa jatkettaisiin rajatta (vertaa seuraavan esimerkin b)-kohtaan).

Jokaisen termin merkitys sarjassa on arvioitavissa helposti, sillä reaalimuuttujan trigonometristen funktioiden arvot ovat aina välillä $[-1, 1]$. Kerroin a_j tai b_j siis ilmaisee kuinka merkittävästi vastaava termi vaikuttaa lopputulokseen.

Esimerkki 4.10.1 Haemme funktioiden a) $f(x) = x^2$ ja b) $g(x) = x$ Fourier'n sarjat välillä $[-\pi, \pi]$.

a) Kyseessä on parillinen funktio ($f(-x) = f(x)$), joten Fourier'n sarjaan tulee

vain muotoa $a_j \cos(jx)$ olevia termejä.

$$a_j = \frac{1}{\pi} \int_{-\pi}^{\pi} x^2 \cos(jx)\, dx = \frac{4}{j^2}(-1)^j,\ j \neq 0$$

$$a_0 = \frac{1}{\pi} \int_{-\pi}^{\pi} x^2\, dx = \frac{2\pi^2}{3}.$$

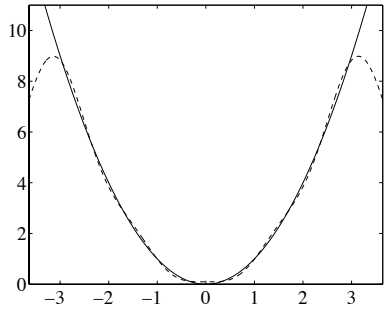
Siispä

$$x^2 = \frac{\pi^2}{3} - 4\cos(x) + \cos(2x) - \frac{4}{9}\cos(3x) + \frac{1}{4}\cos(4x) - \ldots$$

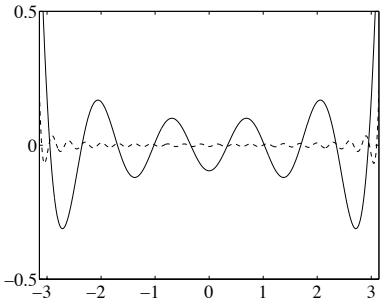
Kertoimien a_j itseisarvo pienenee kuten $1/j^2$, joten sarja suppenee melko nopeasti.

Kuvassa 4.8a on sekä funktion $f(x)$ että termin $a_4 \cos(4x)$ jälkeen katkaistun Fourier'n sarjan $\hat{f}_4(x)$ kuvaajat. Koska approksimaatio on 2π-jaksoinen funktio, se seuraa alkuperäisen funktion käyttäytymistä vain välillä $[-\pi, \pi]$.

Approksimaatioiden $\hat{f}_4(x)$ ja $\hat{f}_{20}(x)$ pisteittäin laskettujen virheiden kuvaajat on esitetty kuvassa 4.8b. Virhe on suurimmillaan välin $[-\pi, \pi]$ päätepisteissä. Kun sarjaan otetaan mukaan enemmän termejä, myös virhe pisteissä $x = \pm\pi$ pienenee kohti arvoa 0.



a: *Funktio $f(x) = x^2$ (yhtenäinen käyrä) ja sen Fourier'n approksimaatio $\hat{f}_4(x)$ (katkoviiva).*



b: *Approksimaation virhe $f(x) - \hat{f}_n(x)$, $n = 4$ (yhtenäinen käyrä) ja $n = 20$ (katkoviiva).*

Kuva 4.8: *Funktion $f(x) = x^2$ jaksollinen approksimointi välillä $[-\pi, \pi]$.*

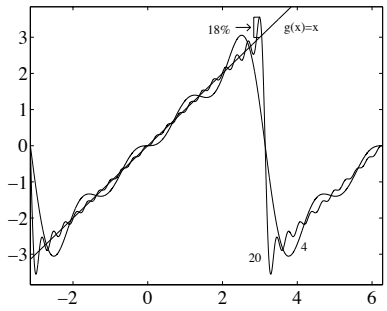
b) Koska $g(x) = x$ on pariton funktio ($f(-x) = -f(x)$), Fourier'n sarjassa esiintyy vain muotoa $b_j \sin(jx)$ olevia termejä. Kertoimet b_j lasketaan yhtälöistä (4.16):

$$b_j = \frac{1}{\pi} \int_{-\pi}^{\pi} x \sin(jx)\, dx = -\frac{2}{j}(-1)^j.$$

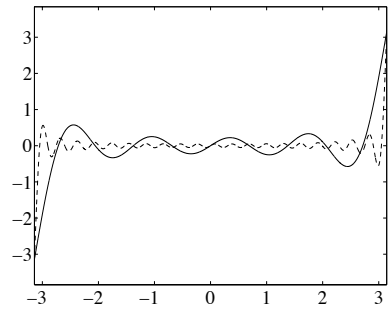
Kuvassa 4.9a on jälleen esitetty funktio ja sen katkaistu Fourier'n sarja $\hat{g}_{20}(x)$. Sarja approksimoi funktiota hyvin välin $[-\pi, \pi]$ keskellä, mutta välin päätepisteissä $x = \pm\pi$ Fourier'n sarja $\hat{g}_{20}(x)$ saa arvon 0. Olemme yrittäneet luoda

jaksollista esitystä funktiolle, joka ei saa samoja arvoja tarkasteluvälin päätepisteissä. Esimerkin a)-kohdassa tätä ongelmaa ei ollut, sillä $(-\pi)^2 = \pi^2$. Fourier'n approksimaatio $\hat{g}_n(x)$ saa pisteessä $x = \pi$ arvokseen keskiarvon luvuista $g(-\pi) = -\pi$ ja $g(\pi) = \pi$.

Pisteiden $x = \pm\pi$ lähistöllä Fourier'n sarja $\hat{g}_{20}(x)$ heilahtelee voimakkaasti. Kyseessä on *Gibbsin ilmiö*. Heilahtelu *ei* vaimene (ei myöskään toisaalta kasva rajatta), vaikka termien määrää lisättäisiin. Sen sijaan heilahtelu sijoittuu yhä tiiviimmin pisteiden $x = \pm\pi$ ympäristöön. Enimmillään Fourier'n sarja voi liioitella approksimoitavan funktion arvoja noin 18%.



a: *Funktio $g(x) = x$ ja sen Fourier'n approksimaatio $\hat{g}_{20}(x)$.*



b: *Approksimaation virhe $g(x) - \hat{g}_n(x)$, $n = 4$ (yhtenäinen käyrä) ja $n = 20$ (katkoviiva).*

Kuva 4.9: *Funktion $g(x) = x$ jaksollinen approksimointi välillä $[-\pi, \pi]$.*

Tutkimme seuraavaksi trigonometristä interpolointia. Tunnemme interpoloitavan funktion $f(x)$ arvot $2n+1$ pisteessä: $f(x_j) = f_j$, $x_j \in [-\pi, \pi)$, $j = 0, \ldots, 2n$. Tavoitteena on etsiä interpolantti $\tilde{f}(x) \in T_n$, jolle pätee $\tilde{f}(x_j) = f_j$ kaikilla j.

Yleisessä tapauksessa voimme edetä samoin kuin tavallisen Lagrangen interpoloinnin kanssa (kappale 4.8). Muodostamme ensin Lagrangen tekijät

$$\phi_j = \prod_{\substack{k=0 \\ k \neq j}}^{n} \frac{\sin\frac{x-x_k}{2}}{\sin\frac{x_j-x_k}{2}},\ j = 0,\ 1,\ ...,\ 2n, \tag{4.17}$$

joille pätee $\phi_j(x_j) = 1$, $\phi_j(x_k) = 0$, $k \neq j$. Interpoloiva trigonometrinen polynomi $\tilde{f}$ saadaan kertomalla kukin $\phi_j(x)$ tunnetulla funktion arvolla f_j ja laskemalla tulokset yhteen:

$$\tilde{f}(x) = \sum_{j=0}^{2n} f_j \phi_j(x). \tag{4.18}$$

Edellä esitetyssä menetelmässä ei oletettu mitään interpolointipisteiden $\{x_j\}$ jakautumisesta. Hyvin usein käytännössä pisteet ovat kuitenkin jakautuneet

tasaisesti koko välille $[-\pi, \pi)$, joten katsomme vielä miten interpolointi tapahtuu tässä erikoistapauksessa. Olkoon siis

$$x_j = -\frac{2n+1-2j}{2n+1}\pi, \quad j = 0, 1, ..., 2n.$$

Tällöin interpolantti $\tilde{f}(x)$ on kirjoitettavissa muotoon

$$\tilde{f}(x) = \frac{a_0}{2} + \sum_{j=1}^{n} a_j \cos(jx) + b_j \sin(jx). \tag{4.19}$$

Kertoimet a_j ja b_j saadaan nyt summalausekkeista

$$a_j = \frac{2}{2n+1} \sum_{k=0}^{2n} f_k \cos \frac{2\pi jk}{2n+1}, \quad j = 0, ...n,$$

$$b_j = \frac{2}{2n+1} \sum_{k=0}^{2n} f_k \sin \frac{2\pi jk}{2n+1}, \quad j = 1, ...n. \tag{4.20}$$

Pisteiden lukumäärän rajoittaminen parittomiin lukuihin $2n+1$ on tietysti vain keinotekoinen lisäehto, jotta voimme kirjoittaa interpolantin $\tilde{f}$ yhtälön (4.19) ilmoittamaan muotoon. Jos pisteitä on parillinen määrä, joudumme pudottamaan termin $\sin(nx)$ pois kantafunktioiden joukosta.

Kaavat (4.19) ja (4.20) ovat tulkittavissa kaavojen (4.15) ja (4.16) diskretoiduiksi versioiksi, joissa integraalit on korvattu yksinkertaisilla kvadratuureilla (katso seuraava luku). Tästä syystä trigonometristä interpolointia tasavälisessä pisteistössä kutsutaan myös nimellä *diskreetti Fourier'n muunnos*. Diskreetti Fourier'n muunnos siis kuvaa jonon $\{f_j\}_0^{2n}$ funktion f arvoja kertoimille $\{a_0, a_1, ..., a_n, b_1, ..., b_n\}$.

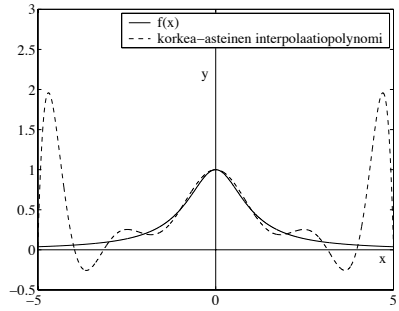
Yhtälöt (4.20) voidaan kirjoittaa myös matriisien ja vektorien avulla. Määritellään vektorit $\mathbf{a} = (a_0, ..., a_n)^T$, $\mathbf{b} = (b_1, ..., b_n)^T$, $\mathbf{f} = (f_0, ..., f_{2n})^T$ ja $(n+1) \times (2n+1)$-matriisi C, $c_{jk} = \cos \frac{2\pi(j-1)k}{2n+1}$ ja $n \times (2n+1)$-matriisi S, $s_{jk} = \sin \frac{2\pi jk}{2n+1}$. Näitä matriiseja ja vektoreita käyttäen diskreetti Fourier'n muunnos on lyhyesti

$$\begin{pmatrix} \mathbf{a} \\ \mathbf{b} \end{pmatrix} = \begin{pmatrix} C \\ S \end{pmatrix} \mathbf{f}.$$

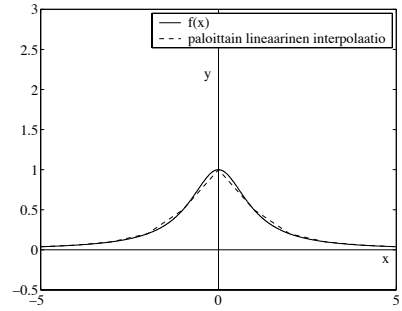
4.11 Paloittainen interpolointi

Kuten aiemmin on jo mainittu, interpoloinnin parantaminen ei yleensä onnistu nostamalla interpoloinnin kertalukua. Paljon menestyksekkäämmäksi lähtökohdaksi osoittautuu ajatus alkuperäisen interpolointivälin $[a, b]$ jakamisesta pienempiin osiin $[x_i, x_i + 1]$, $i = 0, ..., n-1$, $x_0 = a$, $x_n = b$, joista jokaiselle muodostetaan oma interpolantti.

Esimerkki 4.11.1 Vertailemme edellä esitettyjä interpolointitapoja, kun kohteena on funktio $f(x) = 1/(1+x^2)$, $x \in [-5,5]$. Valitaan interpolointia varten tasavälein 11 pistettä $x_i = -5+i$, $i = 0,\ldots 10$, ja sovitetaan pisteiden $(x_i, f(x_i))$ kautta sekä 10. asteen polynomi että paloittain lineaarinen interpolantti (eli pisteitä yhdistävä murtoviiva). Tulokset näkyvät kuvassa 4.10. Korkea-asteinen polynomi heilahtelee voimakkaasti interpolointivälin reunoilla, mutta paloittain lineaarisen interpolantin virhe ei missään kasva kovin suureksi. Jos interpolointipisteiden lukumäärää lisätään, korkea-asteisen interpolantin heilahtelu vain pahenee, kun taas pisteiden kautta kulkeva murtoviiva approksimoi entistä tarkemmin mallitettavaa funktiota.



a: *Funktio $f(x) = 1/(1+x^2)$ ja korkea-asteinen interpolantti.*



b: *Funktio $f(x) = 1/(1+x^2)$ ja paloittain lineaarinen interpolantti.*

Kuva 4.10: *Korkea-asteisen ja paloittain lineaarisen interpolantin vertailu.*

4.11.1 Splinien käyttö interpolointiin

Yleisimmin paloittaiseen interpolointiin tai approksimointiin käytetään *palapolynomeja*, eli funktioita, jotka ovat kullakin osavälillä

$$I_i = [x_{i-1}, x_i], \quad a = x_0, x_1, \ldots, x_n = b$$

ilmaistavissa polynomina. Palapolynomin $s(x)$ asteluku on p, jos kaikkien osien asteluku on p. Palapolynomin ei tietenkään tarvitse olla jatkuva osavälejä erottavissa solmupisteissä, mutta jos polynomi ja sen derivaatat kertalukuun $p-1$ ovat jatkuvia jokaisessa solmupisteessä, kyseessä on p-asteinen *splini*. Edellisessä esimerkissä käytetty interpolaatiopisteitä yhdistävä jatkuva murtoviiva on siis 1. asteen splini. Monissa fysikaalisissa sovelluksissa mallifunktiolta edellytetään enemmän jatkuvuutta, jolloin erityisesti 3. asteen splinit (*kuutiosplinit*) ovat suosittuja.

Splineillä on mahdollista ratkoa sekä approksimointi- että interpolointitehtäviä. Yksinkertaisissa tapauksissa datapisteiden $(x_i, f(x_i))$ x-koordinaatit

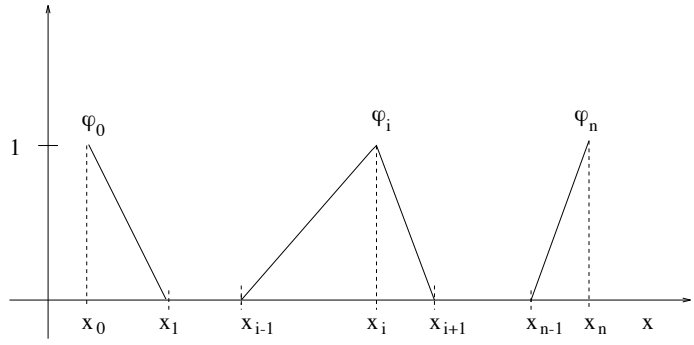
voidaan asettaa suuruusjärjestykseen, mutta jos mallitettava käyrä esimerkiksi leikkaa itsensä, splinien sovitus on tehtävä sopivan parametrisoinnin $x_i = x(t_i)$ avulla. Luonnollinen vaihtoehto on tällöin käyttää parametrina kaarenpituutta tai jotain sen laskennallista likiarvoa. Tällöin puhutaan *parametrisoiduista splineistä* (kappale 4.11.5).

Tarkastellaan ensin interpolointia splinien avulla. Ensimmäisen asteen splinin s_1 muodostaminen on tietenkin suorastaan triviaalia, koska kyseessä on datapisteitä yhdistävä murtoviiva. Sekä numeeristen laskujen että myöhempien sovellusten (integrointi, elementtimenetelmä) kannalta on kuitenkin syytä esittää seuraava apufunktioihin perustuva formulointi.

Olkoon annettuna solmupisteiden joukko $X_n = \{x_i\}$, $i = 0, 1, ..., n$ ja näitä vastaavat funktion arvot $\{f_i\} = \{f(x_i)\}$. Välin $I_i = [x_{i-1}, x_i]$ pituus $h_i = x_i - x_{i-1}$. Jokaista solmupistettä x_i kohti määritellään apufunktio $\phi_i(x)$:

$$\phi_i(x) = \begin{cases} \dfrac{x - x_{i-1}}{h_i}, & x \in I_i, \\ \dfrac{x_{i+1} - x}{h_{i+1}}, & x \in I_{i+1}, \\ 0, & \text{muulloin.} \end{cases}$$

Funktio ϕ_i saa siis arvon 1 pisteessä x_i ja vähenee lineaarisesti arvoon 0 viereisiin solmupisteisiin kuljettaessa (kuva 4.11). Välin $[x_{i-1}, x_{i+1}]$ ulkopuolella ϕ_i häviää kaikkialla. Päätypisteitä x_0 ja x_n vastaavat apufunktiot ϕ_0 ja ϕ_n poikkeavat nollasta vain yhdellä osavälillä.



Kuva 4.11: *Murtoviivaspliniin liittyvät apufunktiot.*

Funktiota $f(x)$ interpoloiva 1. asteen splini $s_1(x)$ on nyt helppo kirjoittaa apufunktioiden avulla muodossa

$$s_1(x) = \sum_{i=0}^{n} f(x_i)\phi_i(x).$$

Korkeamman asteen splinien yhteydessä on syytä miettiä, kuinka paljon ehtoja splinin määrittelyssä on varaa asettaa. Jos interpolointiväli $[a, b]$ on jaettu n osaväliin, joista jokaisella käytetään p-asteista polynomia, niin vapaita parametreja on kaikkiaan $n(p + 1)$. Toisaalta splinin ja sen derivaattojen kertalukuun $p - 1$ asti on oltava jatkuvia kaikissa solmupisteissä välin päätepisteitä lukuunottamatta, mistä saadaan yhteensä $(n - 1)p$ yhtälöä kertoimien välille. Vapaita parametreja jää näin ollen jäljelle $n(p + 1) -$

$(n-1)p = n+p$ kappaletta. Toisin sanoen n osaväliä kattavat paloittain p-asteiset splinit virittävät $(n+p)$-ulotteisen lineaariavaruuden, jota merkitään symbolilla S_{pn} tai $S_{pn}(X_n)$ (jos halutaan korostaa valittujen solmupisteiden merkitystä).

Koska nyt on kyse erityisesti interpoloinnista, asetetaan vielä lisävaatimus, että solmupisteissä ($n+1$ kappaletta) splinin on yhdyttävä annettuihin arvoihin, joten loppujen lopuksi "ylimääräisiä" vapausasteita jää $p-1$ kappaletta.

Kuutiollisen splinin s_3 kohdalla on varaa asettaa kaksi lisäehtoa jatkuvuus- ja interpolointivaatimusten lisäksi ($p = 3$). Tavallisesti nämä käytetään kiinnittämään jonkin kertaluvun derivaatan arvot interpolointivälin $[a,b]$ päätepisteissä. Lisäehdot ovat eräs syy siihen, että kuutiolliset splinit ovat suositumpia kuin 2. asteen polynomeista koostuvat neliölliset splinit; kun $p = 2$, vain yksi lisäehto on mahdollista ottaa mukaan, joten ilman erikoisjärjestelyitä derivaatan arvoja ei pystytä määräämään molemmissa päätepisteissä.

Seuraavassa on lueteltu selvyyden vuoksi vielä kaikki ehdot tapauksessa $p = 3$. Osavälillä I_i splinin $s_3(x)$ arvon ilmaisee lauseke $a_i x^3 + b_i x^2 + c_i x + d_i$.

1. Splinin jatkuvuus:
   $a_i x_i^3 + b_i x_i^2 + c_i x_i + d_i = a_{i+1} x_i^3 + b_{i+1} x_i^2 + c_{i+1} x_i + d_{i+1},\ i = 1,2,...,n-1.$
2. Derivaatan jatkuvuus:
   $3a_i x_i^2 + 2b_i x_i + c_i = 3a_{i+1} x_i^2 + 2b_{i+1} x_i + c_{i+1},\ i = 1,2,...,n-1.$
3. Toisen derivaatan jatkuvuus:
   $6a_i x_i + 2b_i = 6a_{i+1} x_i + 2b_{i+1},\ i = 1,2,...,n-1.$
4. Interpolointi: $s_3(x_i) = f(x_i),\ i = 0,1,...,n$, missä luvut $f(x_i)$ ovat interpoloitavan funktion arvot solmupisteissä.
5. Mahdolliset lisäehdot:
   - Derivaattareunaehdot (käytetään myös nimeä Hermiten reunaehdot): $s_3'(a) = f'(a)$, $s_3'(b) = f'(b)$.
   - Luonnolliset reunaehdot: $s_3''(a) = s_3''(b) = 0$.
   - Jaksolliset reunaehdot: $s_3'(a) = s_3'(b)$, $s_3''(a) = s_3''(b)$.

Näiden ehtojen perusteella on muodostettavissa lineaarinen yhtälöryhmä, josta kaikki kertoimet a_i, b_i, c_i, d_i, $i = 1,...n$, saadaan periaatteessa ratkaistua. Käytännössä yhtälöitä manipuloidaan ensin toiseen muotoon, jotta laskutoimitukset saadaan sopivin järjestelyin yksinkertaisemmiksi ja stabiilimmiksi.

Joissain tilanteissa on hyödyllistä tuntea seuraava splinien optimaalisuusominaisuus: olkoon $f(x) \in C^m[a,b]$, ja $s_{2m-1}(x)$ funktion $f(x)$ interpolaatiosplini. Jos $g(x)$ on mikä tahansa m kertaa derivoituva funktio, joka toteuttaa samat interpolaatioehdot kuin $s_{2m-1}(x)$, niin

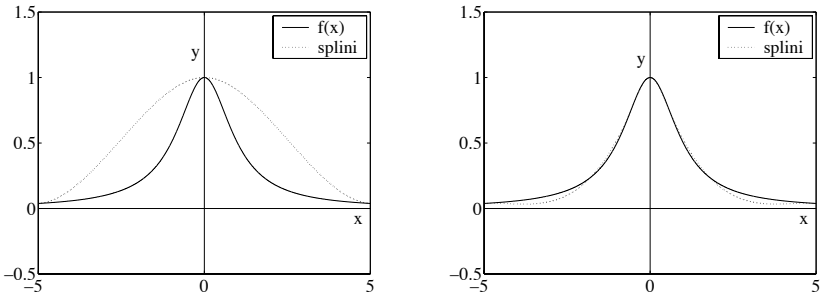
$$\|s_{2m-1}^{(m)}\|_2 \le \|g^{(m)}\|_2.$$

Kun kyseessä on kuutiollinen splini-interpolaatio ($m = 2$), ylläoleva arvio tarkoittaa, että kaikista kahdesti derivoituvista interpolaatioista splinillä on

pienin keskimääräinen kaarevuus[1]. Splini käyttäytyy siis aina sileästi eikä mukaudu nopeisiin muutoksiin. Samalle asialle on olemassa myös fysikaalinen tulkinta: splini kulkee solmupisteiden kautta kuin jännitetty metallilanka, joka asettuu sellaiseen muotoon, että taivutusenergia minimoituu.

Esimerkki 4.11.2 Sovitetaan kuutiollinen splini jo tuttuun esimerkkifunktioon $f(x) = 1/(1+x^2)$ välillä $[-5,5]$. Tehtävässä käytetään ensin vain kahta osaväliä $[-5,0]$ ja $[0,5]$, joista kummallakin on määritettävä 3. asteen polynomi, joten ratkaistavana on 8 tuntematonta kerrointa. Tarvittavat yhtälöt muodostetaan interpolointivaatimuksesta pisteissä $-5, 0, 5$, splinin ja sen 1. ja 2. derivaatan jatkuvuudesta pisteessä 0 ja reunaehdoista pisteissä $-5, 5$ (esimerkiksi splinin derivaatan yhdyttävä interpoloitavan funktion derivaattaan).

Koska funktion $f(x)$ kaarevuus vaihtelee voimakkaasti interpolointiväleillä, kahteen osäväliin perustuva kuutiollinen splini ei pysty seuraamaan funktiota (kuva 4.12a). Jos väli $[-5,5]$ jaetaan neljään osaan, ja valitaan uusiksi jakopisteiksi funktion $f(x)$ toisen derivaatan nollakohdat $x = \pm 1/\sqrt{3}$, interpolaatio paranee selvästi (kuva 4.12a).



a: *Funktio $f(x) = 1/(1+x^2)$ ja kahden osavälin splini-interpolaatio.*

b: *Funktio $f(x) = 1/(1+x^2)$ ja neljän osavälin splini-interpolaatio.*

Kuva 4.12: *Kuutiollinen splini-interpolaatio funktiolle $f(x) = 1/(1+x^2)$.*

4.11.2 B-splinit

Sekä teoreettisten tarkastelujen että käytännön laskujen kannalta on hyödyllistä tuntea jokin kanta spliniavaruudelle S_{pn}. Yleisimmin käytetään *B-splinejä* (alunperin de Boorin splinit, mutta englanninkielisessä kirjallisuu-

[1] Tasokäyrän kaarevuus pisteessä $(x, y(x))$ on

$$\kappa(x) = \frac{y''(x)}{(1+[y'(x)]^2)^{3/2}},$$

joten jos tehdään oletus $|y'(x)| \ll 1$ välillä $[a,b]$, kaarevuus on oleellisesti suoraan verrannollinen 2. derivaattaan y''.

dessa käytetään myös nimitystä Basic Spline Curves), jotka on itse asiassa helpointa esitellä valitsemalla ensin solmupisteiksi ääretön pistejoukko $X_\infty = \{x_i\}$, $i \in \mathbb{Z}$, jolle pätee $x_i < x_{i+1}$ ja $\lim_{i\to\pm\infty} x_i = \pm\infty$. Tällöin ei tarvitse huolehtia äärelliseen väliin liittyvistä reunaehdoista.

Jokaiseen pisteistön X_∞ solmupisteeseen x_i liittyvä astetta p oleva B-splini $B_{i,p}(x)$ saadaan määrättyä yksikäsitteisesti seuraavista ehdoista:

1. Äärellinen kantaja[2]: $B_{i,p}(x)$ poikkeaa nollasta vain äärellisellä välillä $[x_i, x_{i+p+1})$, jonka ulkopuolella $B_{i,p}(x) \equiv 0$. Siis mitä korkeampi asteluku p, sitä useamman osavälin $[x_i, x_{i+1}]$ ylitse kantaja ulottuu. Huomaa myös, että $B_{i,p}(x)$ ei sijaitse symmetrisesti pisteen x_i suhteen.
2. Normalisaatio:

$$\int_{-\infty}^{\infty} B_{i,p}(x)\,dx = \int_{x_i}^{x_{i+p+1}} B_{i,p}(x)\,dx = 1.$$

B-splineille esitetään kirjallisuudessa myös vaihtoehtoisia määritelmiä, mutta niiden tärkeimmät ominaisuudet säilyvät määrittelytavasta riippumatta.

- Lineaarinen riippumattomuus:

$$\sum_{i=-p}^{n-1} \alpha_i B_{i,p}(x) = 0 \quad \text{kaikilla } x \in [x_0, x_n],$$

  jos ja vain jos $\alpha_{-p} = \alpha_{-p+1} = \ldots = \alpha_{n-1} = 0$.

- Jokainen astetta p oleva splini $s(x)$ välillä $[x_0, x_n]$ on esitettävissä yksikäsitteisesti B-splinien avulla:

$$s(x) = \sum_{i=-p}^{n-1} \alpha_i B_{i,p}(x), \quad \alpha_i \in \mathbb{R}.$$

  B-splinit muodostavat siis kannan äärellisille väleille määritellyille spliniavaruuksille.

- Äärettömälle solmupisteistölle luotu B-splinisysteemi peittää reaaliakselin tasaisesti seuraavassa mielessä:

$$\sum_{i\in\mathbb{Z}} B_{i,p}(x) = 1 \quad \text{kaikilla } x \in \mathbb{R}.$$

- Palautuskaava:

$$B_{i,0}(x) = \begin{cases} 1, & x \in [x_i, x_{i+1},) \\ 0, & \text{muulloin} \end{cases}$$

$$B_{i,p}(x) = \frac{x - x_i}{x_{i+p} - x_i} B_{i,p-1}(x) + \frac{x_{i+p+1} - x}{x_{i+p+1} - x_{i+1}} B_{i+1,p-1}(x).$$

[2] Funktion $f(x)$ *kantaja* on sen joukon sulkeuma, jossa $f(x) \neq 0$.

Koska suoraan määritelmän nojalla $B_{i,0}(x)$ häviää identtisesti kaikkialla muualla paitsi välillä $[x_i, x_{i+1})$, jossa se saa vakioarvon 1, kaikki korkeamman asteiset splinit voidaan muodostaa palautuskaavan avulla. Monissa lähteissä B-splinit jopa määritellään antamalla $B_{i,0}(x)$ ja palautuskaava.

- Positiivisuus: $B_{i,p}(x) \geq 0$ kaikilla $x \in \mathbb{R}$.

Esimerkki 4.11.3 Koska pätee

$$B_{i,0}(x) = \begin{cases} 1, & x \in [x_i, x_{i+1}), \\ 0, & \text{muulloin} \end{cases}$$

ja

$$B_{i+1,0}(x) = \begin{cases} 1, & x \in [x_{i+1}, x_{i+2}), \\ 0, & \text{muulloin,} \end{cases}$$

palautuskaavan nojalla 1. asteen splini on

$$B_{i,1}(x) = \begin{cases} (x - x_i)/(x_{i+1} - x_i), & x \in [x_i, x_{i+1}), \\ (x_{i+2} - x)/(x_{i+2} - x_{i+1}), & x \in [x_{i+1}, x_{i+2}), \\ 0, & \text{muulloin.} \end{cases}$$

Vertaa tätä sivun 112 hattufunktioihin; huomaa, että B-splinejä ei indeksoida funktion maksimipisteen mukaan.

Lineaaristen splinien avulla voidaan edelleen muodostaa kvadraattiset splinit $B_{i,2}(x)$ ja näistä puolestaan kuutiolliset splinit $B_{i,3}(x)$. *Tasaväliseen* ($x_{i+1} - x_i = h$ kaikilla i) solmupisteistöön liittyvät kuutiolliset splinit saadaan paloittain lausekkeista

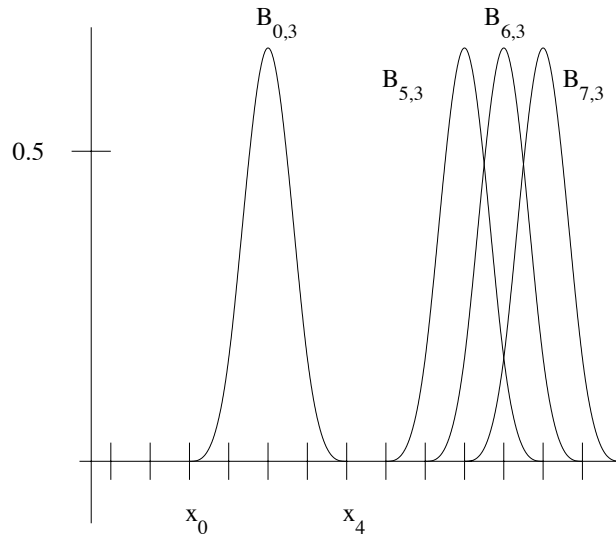
$$B_{i,3}(x) = \frac{1}{6h^2} \begin{cases} (x - x_i)^3, & x \in [x_i, x_{i+1}), \\ h^3 + 3h^2(x - x_{i+1}) + \\ \quad 3h(x - x_{i+1})^2 - 3(x - x_{i+1})^3, & x \in [x_{i+1}, x_{i+2}), \\ h^3 + 3h^2(x_{i+3} - x) + \\ \quad 3h(x_{i+3} - x)^2 - 3(x_{i+3} - x)^3, & x \in [x_{i+2}, x_{i+3}), \\ (x_{i+4} - x)^3, & x \in [x_{i+3}, x_{i+4}), \\ 0, & \text{muulloin.} \end{cases}$$

Kuvassa 4.13 on esitetty muutama kuutiollinen splini $B_{i,3}(x)$.

Funktion $f(x)$ splini-interpolaation $s_p(x)$ laskeminen äärellisessä solmupisteistössä $X_n = \{x_0, \ldots, x_n\}$ on B-splinien avulla selvästi nopeampaa kuin suoraan splinin määritelmään perustuvan yhtälöryhmän ratkaiseminen. Jotta merkinnät pysyisivät yksinkertaisina, pisteistö oletetaan jatkossakin tasaväliseksi.

Tehtävänä on siis määrittää kertoimet α_i splinin $s_p(x)$ esityksessä

$$s_p(x) = \sum_{i=-p}^{n-1} \alpha_i B_{i,p}(x).$$



Kuva 4.13: *Kuutiollisia B-splinejä tasavälisessä solmupisteistössä.*

Koska p-asteisen B-splinin kantaja on levittäytynyt p osavälin yli, ensimmäisellä osavälillä $[x_0, x_1]$ vaikuttavat kaikkiaan kantafunktiot $B_{-p,p}$, $B_{-p+1,p}$, ..., $B_{0,p}$. Tämän vuoksi summaus alkaa indeksistä $-p$, vaikka solmupisteitä $x_{-p}, ..., x_1$ ei olekaan käytössä. Pisteistön loppupäässä vastaavaa ongelmaa ei ole, koska B-splinit on määritelty epäsymmetrisesti indeksiä vastaavan solmupisteen suhteen.

Tuntemattomia kertoimia α_i on siis yhteensä $n + p$ kappaletta kuten pitääkin, kun luodaan $n + 1$ pisteen kautta kulkeva astetta p oleva splini. Interpolointiehdoista ($s_p(x_i) = f(x_i)$) saadaan $n + 1$ yhtälöä, ja loput $p - 1$ yhtälöä muodostetaan reunaehtojen avulla. B-splinien kantajien äärellisyyden vuoksi syntyvät yhtälöryhmät ovat aina nauhamaisia. Kullakin vaakarivillä on korkeintaan p nollasta poikkeavaa alkiota.

Esimerkki 4.11.4 Haetaan esimerkissä 4.11.2 välille $[-5, 5]$ muodostetun interpoloivan kuutiosplinin kertoimet α_i B-splinikannassa $\{B_{i,3}\}$, kun solmupisteiksi valitaan $X = \{-5, -4, ..., 5\}$. Tarkasteluvälillä nollasta poikkeavat B-splinit $B_{-8,3}$, $B_{-7,3}$, ..., $B_{4,3}$; yhteensä siis 13 kantafunktiota, joille tarvitaan kertoimet. Esimerkin 4.11.3 lopussa esitettiin kolmannen asteen B-splinin lausekkeet, joista voidaan laskea B-splinin arvot solmupisteissä:

	x_i	x_{i+1}	x_{i+2}	x_{i+3}	x_{i+4}
$B_{i,3}$	0	1/6	2/3	1/6	0
$B'_{i,3}$	0	$1/2h$	0	$-1/2h$	0

Koska splinin $s_3(x)$ on tarkoitus interpoloida funktiota $f(x)$ solmupisteissä

$\{x_i\}$, kertoimille α_i voidaan kirjoittaa yhtälöryhmä

$$s_3(x_i) = \sum_{j=i-3}^{i-1} \alpha_j B_{j,3}(x_i) = f(x_i), \quad -5 \le i \le 5.$$

Puuttuvat kaksi yhtälöä tulevat reunaehdoista (tässä tapauksessa Hermiten ehdot $s_3'(-5) = f'(-5)$, $s_3'(5) = f'(5)$) muodossa

$$\sum_{j=i-3}^{i-1} \alpha_j B_{j,3}'(x_i) = f'(x_i), \quad i = -5, 5.$$

Yhtälöt on nyt helppo kirjoittaa matriisimuodossa $A\alpha = \mathbf{f}$, sillä arvot $B_{j,3}(x_i)$ ja $B_{j,3}'(x_i)$ ovat luettavissa suoraan ylläolevasta taulukosta:

$$A = \frac{1}{6}\begin{pmatrix} -\frac{3}{h} & 0 & \frac{3}{h} & & & & & & \\ 1 & 4 & 1 & & & & & & \\ & 1 & 4 & 1 & & & & & \\ & & \ddots & \ddots & \ddots & & & & \\ & & & 1 & 4 & 1 & & & \\ & & & & 1 & 4 & 1 & & \\ & & & & & -\frac{3}{h} & 0 & \frac{3}{h} & \end{pmatrix}, \quad \mathbf{f} = \begin{pmatrix} f'(-5) \\ f(-5) \\ f(-4) \\ \vdots \\ f(4) \\ f(5) \\ f'(5) \end{pmatrix}.$$

Kerroinmatriisi on melkein tridiagonaalinen, joten yhtälöryhmän ratkaiseminen käy hyvin nopeasti.

4.11.3 Virhearviot

Lineaarisille splineille on helppo esittää arvioita interpolaatiovirheen suuruudesta. Mitä enemmän sileyttä interpoloitavalta funktiolta oletetaan, sitä parempia arvioita tietysti saadaan. Seuraavassa $h = \max\{x_{i+1} - x_i\}$ ja funktion jatkuvuutta mitataan suureella

$$M_f(h) = \sup_{|x-y| \le h} |f(x) - f(y)|.$$

Jos $f(x)$ on jatkuva funktio, perusvirhearvio on yksinkertaisesti $\|f - s_1\|_\infty \le M_f(h)$. Esimerkiksi Lipschitz-jatkuville funktioille[3] pätee

$$\|f - s_1\|_\infty \le Kh.$$

Siis 1. asteen interpolaatiosplinit suppenevat lineaarisella nopeudella solmupisteiden etäisyyden suhteen.

Jos funktion f lisäksi myös sen derivaatta f' on jatkuva, päästään hiukan parempaan tulokseen $\|f - s_1\|_\infty \le \frac{h}{4} M_{f'}(h)$. Jos vielä toinenkin derivaatta f'' on jatkuva, on johdettavissa arvio

$$\|f - s_1\|_\infty \le \frac{h^2}{8} \|f''\|_\infty.$$

[3] Funktio on Lipschitz-jatkuva välillä I, jos $|f(x) - f(y)| \le K|x - y|$ kaikilla $x, y \in I$.

Korkeamman asteen splineille on hankalampaa johtaa yleisiä virhearvioita, mutta tasavälisen solmupisteistön tapauksessa pätee 3. asteen splineille seuraava tulos:

$$\|f - s_3\|_\infty \leq \frac{5h^4}{384}\|f^{(4)}\|_\infty.$$

Tässä kerroin $5/384$ on paras mahdollinen.

4.11.4 Spliniapproksimaatiot

Toistaiseksi olemme käyttäneet splinejä vain interpolointiin. Koska edellä luodut B-splinit ovat eräs kanta spliniavaruudelle S_{pn}, voimme laskea myös spliniapproksimaatioita aiemmin esitellyn pienimmän neliösumman teorian mukaisesti. Jos approksimaatio s ilmaistaan summana B-splineistä ($s = \sum \alpha_i B_{i,p}$), kertoimet α_i ratkaistaan normaaliyhtälöistä

$$\sum_{i=-p}^{n-1} \alpha_i \langle B_{i,p}, B_{j,p} \rangle = \langle f, B_{j,p} \rangle, \quad j = -p, ..., n-1.$$

Splinien äärellisen kantajan ansiosta vasemman puolen kerroinmatriisi on jälleen nauhamainen.

Spliniapproksimaatioille voidaan johtaa samantapaiset virhearviot kuin interpolaatioillekin. Niinpä kahdesti jatkuvasti derivoituvalle funktiolle f pätee $\|f - s_1\|_2 \leq \frac{h^2}{\pi^2}\|f''\|_2$ ja jos $f \in C^4$, kolmannen asteen approksimaatiolle s_3 on voimassa $\|f - s_3\|_2 \leq 4\frac{h^4}{\pi^4}\|f^{(4)}\|_2$.

4.11.5 Parametrisoidut splinit

Toistaiseksi olemme käsitelleet sellaisten tasokäyrien approksimointia, jotka voidaan esittää muodossa $y = f(x)$. Käytännön sovelluksissa joudutaan usein kuitenkin approksimoimaan *parametrisoituja käyriä*, jotka on annettu muodossa

$$\begin{cases} x = x(t) \\ y = y(t) \end{cases} \quad \text{tai} \quad \begin{cases} x = x(t) \\ y = y(t) \\ z = z(t), \end{cases}$$

missä parametri t käy läpi kaikki arvot jollain reaalilukuvälillä $I_t = [t_i, t_f]$. Funktioilla $x(t)$, $y(t)$ ja $z(t)$ ei välttämättä ole yksikäsitteistä käänteisfunktiota välillä I_t, jolloin parametrimuotoinen esitys on kätevin tapa kirjoittaa käyrän yhtälö.

Periaatteessa aiemmin esitetyt approksimointimenetelmät sopivat sellaisenaan myös parametrimuotoisen käyrän approksimointiin, kunhan jokaiselle koordinaatille suoritetaan erikseen oma sovitus. Seuraavaksi tutustumme joihinkin taso- ja avaruuskäyrien approksimointiin suunniteltuihin erikoismenetelmiin.

4.11.6 Bézier'n käyrät ja Bézier'n splinit

Olkoon annettuna $n+1$ pistettä $P_j \in \mathbb{R}^d$, $j = 0, 1, ..., n$ (*Bézier'n pisteet* tai *ohjauspisteet*). Pisteen P_j paikkavektori on $\mathbf{b}_j$. Näiden pisteiden määräämä *Bézier'n käyrä* $\mathbf{b}_0^n(t)$ saadaan rekursiivisesti yhtälöstä

$$\mathbf{b}_j^k(t) = (1-t)\mathbf{b}_j^{k-1}(t) + t\mathbf{b}_{j+1}^{k-1}(t), \quad t \in [0,1],$$

kun $\mathbf{b}_j^0(t) = \mathbf{b}_j$.

Bézier'n käyrä voidaan laskea myös astetta n olevien *Bernsteinin polynomien*

$$B_j^n(t) = \binom{n}{j}(1-t)^{n-j}t^t, \quad j = 0, 1, ..., n,$$

avulla:

$$\mathbf{b}_0^n(t) = \sum_{j=0}^{n} B_j^n(t)\mathbf{b}_j, \quad t \in [0,1].$$

Esimerkki 4.11.5 Etsimme ohjauspisteiden $P_0 = (-1,0)$, $P_1 = (0,1)$ ja $P_2 = (1,2)$ avulla määritellyn Bézier'n käyrän. Ensimmäisessä vaiheessa muodostamme ohjauspisteitä yhdistävät suorat:

$$\begin{aligned}\mathbf{b}_0^1(t) &= (1-t)\mathbf{b}_0 + t\mathbf{b}_1\\ \mathbf{b}_1^1(t) &= (1-t)\mathbf{b}_1 + t\mathbf{b}_2.\end{aligned}$$

Lopuksi muodostamme näiden välitulosten avulla paraabelin yhtälön

$$\mathbf{b}_2^0(t) = (1-t)\mathbf{b}_0^1 + t\mathbf{b}_1^1.$$

Jos sijoitamme välitulokset edelliseen yhtälöön, päädymme muotoon

$$\mathbf{b}_2^0(t) = (1-t)^2\mathbf{b}_0 + 2t(1-t)\mathbf{b}_1 + t^2\mathbf{b}_2, \tag{4.21}$$

joka on tietysti lausuttavissa myös komponenteittain:

$$\begin{aligned}x(t) &= (1-t)^2\cdot(-1) + 2t(1-t)\cdot 0 + t^2\cdot 1 = 2t-1\\ y(t) &= (1-t)^2\cdot 0 + 2t(1-t)\cdot 1 + t^2\cdot 0 = 2t-2t^2.\end{aligned}$$

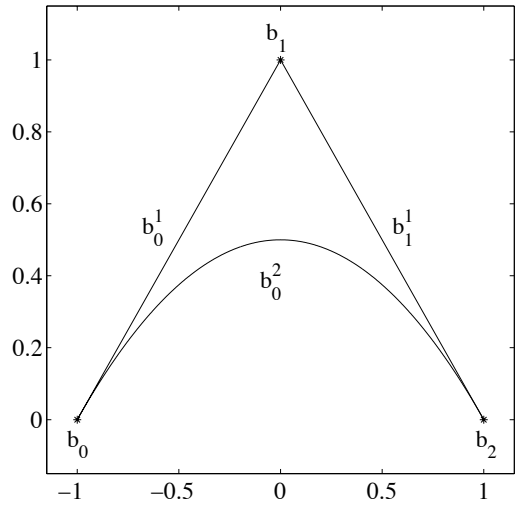
Käyrän kuvaaja on esitetty kuvassa 4.14. Vertailun vuoksi suoritamme laskun myös Bernsteinin polynomien avulla. Kvadraattiset polynomit ovat

$$\begin{aligned}B_0^2(t) &= \binom{2}{0}(1-t)^2t^0 = (1-t)^2\\ B_1^2(t) &= \binom{2}{1}(1-t)^1t^1 = 2t(1-t)\\ B_2^2(t) &= \binom{2}{2}(1-t)^0t^2 = t^2.\end{aligned}$$

Vertaamalla oikeanpuoleisia lausekkeita yhtälössä (4.21) esiintyviin kertoimiin havaitsemme, että menetelmät antavat saman lopputuloksen.

Tässä tehtävässä parametri t ratkeaa toki koordinaatin x suhteen, joten etsitty käyrä on esitettävissä myös suoraan koordinaattimuuttujien välisenä yhtälönä

$$y = \frac{1-x^2}{2}.$$



Kuva 4.14: *Ohjauspisteisiin* $(-1,0)$, $(0,1)$ *ja* $(1,0)$ *liittyvä Bézier'n käyrä.*

Bézier'n käyrä jää aina ohjauspisteidensä virittämän konveksin peitteen sisään. Erikoistilanteita lukuunottamatta käyrä kulkee ainoastaan ensimmäisen ja viimeisen ohjauspisteen kautta, joten kyseessä ei ole interpolantti. Jos $P_0 = P_n$, kyseessä on suljettu käyrä. Jos ohjauspisteet ovat samalla suoralla, niitä vastaava Bézier'n käyräkin on suora.

Monimutkaista käyrää approksimoitaessa on syytä jakaa käyrä osiin, joista jokaiselle haetaan oma Bézier'n esitys. Osiin jakamisen kautta Bézier'n käyriä on mahdollista käyttää myös interpolointiin: valitaan interpolaatiopisteet osavälien päätepisteiksi eli rajapisteiksi (joissain lähteissä nimellä solmupiste) ja lisätään kullekin välille tarpeellinen määrä ohjauspisteitä. Jos osien rajapisteissä huolehditaan riittävästä jatkuvuudesta, kyseessä on *Bézier'n splini*.

Ensimmäisen derivaatan jatkuvuusvaatimuksesta seuraa, että jokainen rajapiste P_k on naapuriohjauspisteidensä yhdysjanan keskipiste:

$$\mathbf{b}_k - \mathbf{b}_{k-1} = \mathbf{b}_{k+1} - \mathbf{b}_k.$$

Toisen derivaatan jatkuvuus puolestaan totetuu vain, jos

$$\mathbf{b}_{k-1} + (\mathbf{b}_{k-1} - \mathbf{b}_{k-2}) = \mathbf{b}_{k+1} - (\mathbf{b}_{k+2} - \mathbf{b}_{k+1}).$$

Edellisen yhtälön vasemman ja oikean puolen lausekkeiden yhteistä arvoa merkitään usein vektorilla $\mathbf{d}_k$, ja sitä kutsutaan *painoksi* tai yksinkertaisesti apupisteeksi. Koska 3. asteen splinejä käytetään paljon erilaisissa interpolointisovelluksissa, monet ohjelmat osaavat laskea painot ja ohjauspisteet valmiiksi, jos sisään syötetään interpolointipisteet. Kuten jo aiemmin on kerrottu, 3. asteen splini-interpoloinnissa on lisäksi valittava välin päädyissä jotkin reunaehdot.

Jos käyrälle etsitään paloittainen Bézier'n interpolaatio $\mathbf{b}(t)$, ei ole yhdentekevää, kuinka globaali parametri t valitaan. Jokaisella osavälillä $[P_i, P_{i+1}]$ on tietysti oma lokaali parametrinsä t_i, joka saa arvoja väliltä $[0, 1]$. Globaalia parametrisointia tarvitaan nimenomaan interpoloinnin yhteydessä. Tällöin nimittäin ainoastaan ne ohjauspisteet on annettu, joiden kautta lopullisen käyrän halutaan kulkevan, ja loput ohjauspisteet on laskettava.

Yksinkertaisin mahdollisuus on laskea paloittaiset esitykset erikseen ja liittää ne sitten yhteen, mutta tällöin ei oteta huomioon osavälien erilaisia pituuksia. Parempaan tulokseen päästäisiin, jos globaalina parametrina käytettäisiin käyrän kaarenpituutta s. Koska kaarenpituuden tarkka laskeminen on yleensä työlästä, joudutaan tavallisesti tyytymään erilaisiin likimääräisiin kaarenpituuksiin.

Interpoloivan 3. asteen splinin muodostaminen onnistuu toki sujuvammin muilla keinoin (esimerkiksi B-splinien avulla). Bézier'n käyriä käytettiin kuitenkin aiemmin niin paljon erilaisissa CAD-ohjelmissa, että interpolaatiotkin oli luonnollista tehdä Bézier'n formalismin kautta, vaikka se edellyttikin puuttuvien ohjauspisteiden laskemista. Nykyään uudemmat menetelmät ovat osittain korvanneet Bézier'n käyrät.

4.11.7 Rationaaliset B-splinit

Palaame vielä kappaleessa 4.11.2 esiteltyihin B-splineihin tarkoituksena laajentaa niiden käyttöaluetta. Kuten jo kerroimme, B-splinit on mahdollista määritellä palautuskaavan (4.21) avulla:

$$B_{i,0}(x) = \begin{cases} 1, & x \in [x_i, x_{i+1}), \\ 0, & \text{muulloin,} \end{cases}$$

$$B_{i,p}(x) = \frac{x - x_i}{x_{i+p} - x_i} B_{i,p-1}(x) + \frac{x_{i+p+1} - x}{x_{i+p+1} - x_{i+1}} B_{i+1,p-1}(x).$$

Pisteille x_i voidaan sallia samoja arvoja, jos sovitaan lisäehdosta, jonka mukaan palautuskaavassa asetetaan nollaksi jokainen termi, jonka nimittäjä häviäisi vähennyslaskun tuloksena.

Parametrisoitujen taso- ja avaruuskäyrien esittämiseen tarvitaan jälleen ohjauspisteitä P_j, $j = 0, ..., n$. Jos tarkoituksena on laskea paloittain kuutiollinen spliniapproksimaatio, käyräparametrin x arvo pitää kiinnittää $n + 5$ pisteessä x_j, $j = 0, ..., n + 4$. Ohjauspisteiden P_{j-3}, P_{j-2}, P_{j-1} ja P_j määräämä kuutiollinen splini on

$$\mathbf{b}(x) = B_{j-3,3}(x)\mathbf{b}_{j-3} + B_{j-2,3}(x)\mathbf{b}_{j-2} + B_{j-1,3}(x)\mathbf{b}_{j-1} + B_{j,3}(x)\mathbf{b}_j,$$

missä $x_j \le x \le x_{j+1}$ ja $3 \le j \le n$.

B-splinien avulla voidaan myös luoda rationaalisia approksimaatiota. Näistä käytetään yleisesti lyhennettä NURBS (Non-uniform rational B-spline). Kolmannen asteen rationaalinen B-spliniapproksimaatio on muotoa

$$\mathbf{r} = \frac{\sum_{k=j-3}^{j} w_k B_{k,3}(x)\mathbf{r}_k}{\sum_{k=j-3}^{j} w_k B_{k,3}(x)}, \quad x_j \le x \le x_{j+1},\ 3 \le j \le n.$$

Painokertoimet w_j kannattaa valita positiivisiksi, jolloin nimittäjä ei häviä millään parametrin x arvolla. Jos kaikki painokertoimet ovat yhtä suuria kyseessä on normaali kuutiollinen B-splini.

4.12 Aallokkeet

Aallokkeet (wavelet, suomeksi myös väre) ovat verrattain uusi matemaattinen työkalu. Aallokkeiden teorian tutkimus käynnistyi varsinaisesti 1980-luvun lopulla, ja hyvin nopeasti aallokkeiden käyttö on levinnyt monille sovellusalueille. Laajimmin niitä hyödynnetään signaalinkäsittelyssä ja erilaisissa datan pakkausmenetelmissä. Myös osittaisdifferentiaaliyhtälöiden ratkaisuun ja tilastotieteellisiin tehtäviin on kehitetty aallokkeisiin perustuvia menetelmiä.

Tässä kappaleessa kerromme lyhyesti aallokkeiden tärkeimpiä ominaisuuksia menemättä kuitenkaan teoreettisissa tarkasteluissa kovin syvälle. Vaikka aallokkeiden käyttö funktioiden tai datan approksimoinnissa on periaatteessa suhteellisen suoraviivaista, omien tietokoneohjelmien kirjoittaminen vaatii jonkin verran tarkkaavaista työtä. Siksi suosittelemme valmiiden ohjelmien käyttöä (esimerkiksi Matlabin Wavelet Toolbox). Asioiden oppimisen kannalta omien koodien kirjoittaminen voi tietysti olla mitä hyödyllisintä.

Aallokkeiden suhteellinen nuoruus matemaattisena työkaluna näkyy myös terminologiassa. Monilla aallokkeisiin liittyvillä käsitteillä on useita nimiä tai symboleita, koska eri sovellusaloilla nimeäminen tietysti noudattaa kunkin alan omaa käytäntöä.

4.12.1 Aallokematriisit

Aallokkeiden määrittelyn voi suorittaa useammalla tavalla. Hyvin ylimalkaisesti muotoiltuna aallokkeet ovat kokoelma funktioita, jotka toteuttavat eräät rekursiiviset yhtälöt. Ennen kuin esitämme nämä yhtälöt, käsittelemme lyhyesti yhtälöissä esiintyviä kertoimia.

Lähdemme siis liikkeelle $m \times mg$-matriisista A:

$$A = \begin{pmatrix} \dots & a_{0,-2} & a_{0,-1} & a_{0,0} & a_{0,1} & a_{0,2} & \dots \\ \dots & a_{1,-2} & a_{1,-1} & a_{1,0} & a_{1,1} & a_{1,2} & \dots \\ & \vdots & & \vdots & & \vdots & \\ \dots & a_{m-1,-2} & a_{m-1,-1} & a_{m-1,0} & a_{m-1,1} & a_{m-1,2} & \dots \end{pmatrix}$$

Rivejä m on oltava aina vähintään kaksi. Sarakkeiden lukumäärä mg taas voi olla teoriassa rajaton, käytännössä tietysti jokin äärellinen rivien lukumäärän kokonaislukumonikerta. Suure g on nimeltään matriisin A *genus*. Numeroimme matriisin A alkiot hiukan poikkeuksellisesti alkaen nollasta, ja tarvittaessa käytämme sarakkeiden numerointiin myös negatiivisia indeksejä.

Matriisi A on *aallokematriisi*, jos sen alkiot toteuttavat riveittäin seuraavat ehdot:

$$\sum_k a_{j,ml+k}\bar{a}_{j',ml'+k} = m\delta_{j,j'}\delta_{l,l'} \tag{4.22}$$

$$\sum_k a_{j,k} = m\delta_{j,0}. \tag{4.23}$$

Yhtälössä (4.22) merkintä $\bar{a}_{j',ml'+k}$ tarkoittaa alkion $a_{j',ml'+k}$ kompleksista liittolukua. Yhtälö kertoo, että matriisin A rivit ovat ortogonaaliset. Lisäksi kullakin rivillä m alkion ryhmät ovat keskenään ortogonaaliset. Jälkimmäinen yhtälö (4.23) puolestaan kertoo kunkin rivin alkioiden summan.

Ensimmäisen rivin alkiot muodostavat *skaalausvektorin*

$$a_0 = [a_{0,0}, a_{0,1}, \ldots, a_{0,mg-1}]$$

ja muiden rivien alkiot vastaavasti *aallokevektorit*

$$a_j = [a_{j,0}, a_{j,1}, \ldots, a_{j,mg-1}].$$

Esimerkki 4.12.1 Esittelemme Daubechies'n aallokematriisin D_2 ($m = 2, g = 2$):

$$D_2 = \frac{1}{4}\begin{pmatrix} 1+\sqrt{3} & 3+\sqrt{3} & 3-\sqrt{3} & 1-\sqrt{3} \\ -1+\sqrt{3} & 3-\sqrt{3} & -3-\sqrt{3} & 1+\sqrt{3} \end{pmatrix}$$

Koska $m = g = 2$, niin yhtälö (4.22) sisältää seuraavat seitsemän ehtoa, jotka on helppo todentaa:

1. $\sum_{k=0}^{3} a_{j,k}^2 = 2, \quad j = 0, 1$ (rivien normeeraus)
2. $a_{j,0}a_{j,2} + a_{j,1}a_{j,3} = 0, \quad j = 0, 1$ (kumpikin rivi itsessään kahden alkion lohkoittain ortogonaalinen)
3. $a_{0,0}a_{1,0} + a_{0,1}a_{1,1} + a_{0,2}a_{1,2} + a_{0,3}a_{1,3} = 0$ (rivit keskenään ortogonaaliset)
4. $a_{0,0}a_{1,2} + a_{0,1}a_{1,3} = 0$
5. $a_{0,2}a_{1,0} + a_{0,3}a_{1,1} = 0$ (rivien alku- ja loppupäät vielä erikseen ortogonaaliset)

Lisäksi yhtälö (4.23) antaa ehdot $\sum_{k=0}^{3} a_{0,k} = 2$ ja $\sum_{k=0}^{3} a_{1,k} = 0$, joiden voimassaolo on luettavissa suoraan matriisista.

Matriisi D_2 kuuluu Daubechies'n aallokematriisien perheeseen $\{D_n\}$.

Ennen kuin määrittelemme varsinaiset aallokefunktiot, voimme esittää jo yhden sovelluksen. Jos $f(n) : \mathbb{N} \to \mathbb{C}$ on jokin kokonaislukujen joukossa määritelty funktio, niin f on hajotettavissa muotoon

$$f(n) = \sum_{k=-\infty}^{\infty} c_{0,k}a_{0,mk+n} + \sum_{j=1}^{m-1}\sum_{k=-\infty}^{\infty} c_{j,k}a_{j,mk+n}, \tag{4.24}$$

missä kertoimet $c_{j,k}$ saadaan summalausekkeista

$$c_{j,k} = \frac{1}{m} \sum_n f(n) \bar{a}_{j,mk+n}. \tag{4.25}$$

Vaikka summausindeksi k periaatteessa saa kaikki kokonaislukuarvot, käytännössä jokaista muuttujan n arvoa kohti vain muutama eri k tuottaa sellaisen indeksin $mk + n$, että alkiot $a_{j,mk+n}$ eroavat nollasta.

Hajotelmaa voi soveltaa mihin tahansa signaaliin, jolloin funktion $f(n)$ arvoina käytetään siis signaalivektorin alkioita. Signaalin suuren mittakaavan käyttäytyminen (matalat taajuudet) jää aallokematriisin ensimmäisen rivin alkioista riippuvaan summaan ja nopeasti tapahtuvat muutokset (korkeat taajuudet) sisältyvät jälkimmäiseen kaksoissummaan, jossa käytetään matriisin muita rivejä. Jos alkuperäisessa signaalissa on N alkiota, niin laskettavia kertoimia $c_{j,k}$ on yleensä saman verran. Hajotelma tarjoaa kuitenkin mahdollisuuden informaation tiivistämiseen, sillä osa kertoimista voidaan tavallisesti jättää pois ilman, että merkittävästi informaatiota häviää.

Samaa ideaa voi edelleen soveltaa rekursiivisesti kertaalleen jo tiivistettyyn signaaliin. Näin signaalista suodattuu vähitellen pois yhä enemmän piirteitä. Riittävän monen suodatuksen jälkeen jäljelle jää vain alkuperäisen signaalifunktion keskiarvo.

Esimerkki 4.12.2 Käytämme edellisessä esimerkissä esiteltyä Daubechies'n aallokematriisia suodattamaan signaalin $f(n)$, joka koostuu sin-funktion arvosta 128 pisteessä välillä $[0, 1]$. Kuvassa 4.15a näkyy pelkästään ensimmäisen summan termien avulla rekonstruoitu signaali $\tilde{f}(n)$. Tämän kuvan tuottamiseen riittivät kertoimet $c_{0,k}$, $k = 1, 0, -1, \ldots, -64$ – siis yhteensä 66 lukua.

Kuvassa 4.15b on alkuperäisen signaalin $f(n)$ ja suodatetun signaalin $\tilde{f}(n)$ erotus, joka on siis sama kuin yhtälön (4.24) oikean puolen jälkimmäinen summatermi. Näemme, että virhe $f(n) - \tilde{f}(n)$ on pieni lukuunottamatta signaalin paria ensimmäistä ja viimeistä arvoa. Pystymme siis välittämään oleellisesti virheettömän signaalin käyttämällä vain noin puolet (66/128) siitä informaatiomäärästä, jonka alkuperäinen signaali vaati.

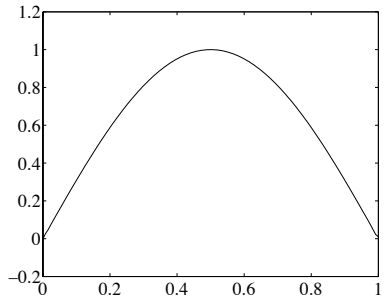
4.12.2 Aallokefunktiot

Aallokematriisin A ylimmän rivin alkioiden $a_{0,k}$ avulla voidaan kirjoittaa matriisiin A liittyvä *skaalautuvuusyhtälö*

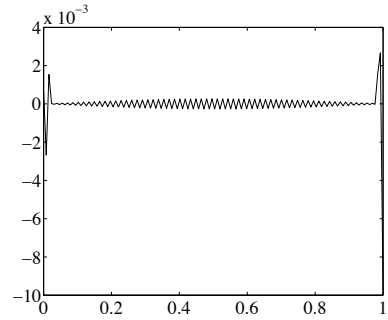
$$\phi(x) = \sum_{k=0}^{mg-1} a_{0,k} \phi(mx - k), \tag{4.26}$$

jonka ratkaisua kutsutaan matriisiin A liittyväksi *skaalausfunktioksi* (englanniksi scaling function, myös father wavelet).

Yhtälö (4.26) sitoo funktion $\phi(x)$ arvon pisteessä x saman funktion muihin arvoihin mg tasavälisessä pisteessä. Tämän yhtälön toteuttavia funktioita



a: *Suodatettu signaali* $\tilde{f}(n)$.



b: *Poistettu osuus alkueräisestä signaalista* $f(n) - \tilde{f}(n)$.

Kuva 4.15: *Signaalin* $f(n) = \sin(\frac{\pi(n-1)}{127})$, $n = 1, ..., 128$ *suodatus aallokematriisin* D_2 *avulla.*

on äärettömän monta. On kuitenkin osoitettavissa, että $\phi(x)$ saadaan yksikäsitteisesti kiinnitettyä aina kun yhtälön (4.26) lisäksi vaaditaan, että $\phi(x)$ toteuttaa ehdon

$$\int_{\mathbb{R}} \phi(x)\, dx = 1. \tag{4.27}$$

Lisäksi vaaditaan, että ratkaisu $\phi(x)$ häviää välin $[0, (g-1)\frac{m}{m-1} + 1]$ ulkopuolella. Sanomme, että funktion $\phi(x)$ *kantajalle* supp ϕ pätee supp $\phi \in [0, (g-1)\frac{m}{m-1} + 1]$.

Matriisin A muiden rivien avulla saadaan määriteltyä vastaavat funktionaaliset yhtälöt

$$\psi_j(x) = \sum_{k=0}^{mg-1} a_{j,k}\phi(mx - k), \quad j = 1, \ldots, m-1. \tag{4.28}$$

Huomaa, että yhtälön oikealla puolella olevassa summassa käytetään edelleen skaalausfunktion $\phi(x)$ arvoja. Funktiot $\psi_j(x)$ ovat matriisiin A liittyvät *aallokkeet* tai *aallokefunktiot* (englanniksi wavelet function, myös mother wavelet). Aallokefunktioille on voimassa yhtälö

$$\int_{\mathbb{R}} \psi_j(x)\, dx = 0. \tag{4.29}$$

Kaikki oleellinen tieto skaalausfunktiosta $\phi(x)$ ja aallokefunktioista $\psi_j(x)$ sisältyy matriisin A alkioihin ja alkioita sitoviin ehtoihin (4.22) ja (4.23), ja edelleen yhtälöihin (4.26) ja (4.28). *Näitä yhtälöitä ei yleensä pystytä ratkaisemaan sellaiseen muotoon, että funktioille* $\phi(x)$, $\psi_j(x)$ *saataisiin eksplisiitti-*

nen esitys. Tämän vuoksi aallokkeiden soveltaminen erilaisiin ongelmiin vaatii joskus hiukan miettimistä, ainakin jos tavoitteena on tehokkuus. Aallokkeiden ominaisuudet joudutaan päättelemään epäsuorasti yhtälöistä (4.26) ja (4.28).

Funktioiden $\{\phi(x), \psi_1(x), \ldots, \psi_{m-1}(x)\}$ kuvaajat voi selvittää iteroimalla yhtälöitä (4.26) ja (4.28). Valitsemme alkuarvaukseksi $\phi^0(x)$ esimerkiksi välin $[0,1)$ karakteristisen funktion $\chi_{[0,1)}$:

$$\phi^0(x) = \chi_{[0,1)} = \begin{cases} 1, & 0 \le x < 1 \\ 0, & x < 0 \text{ tai } x \ge 1. \end{cases}$$

Tämän jälkeen sovellamme yhtälöä (4.26) siten, että laskemme uuden iteraatin $\phi^{\nu+1}(x)$ sijoittamalla yhtälön oikealle puolelle edellisen iteraatin $\phi^\nu(x)$:

$$\phi^{\nu+1}(x) = \sum_{k=0}^{mg-1} a_{0,k}\phi^\nu(mx-k).$$

Jatkamalla iterointia saamme yhä tarkempia approksimaatioita skaalausfunktiolle $\phi(x) = \lim_{\nu\to\infty}\phi^\nu(x)$.

On myös toinen vaihtoehto yhtälöiden (4.26), (4.28) ratkaisemiseksi. Haemme vaiheittain funktion $\phi(x)$ arvon *dyadisisssa pisteissä* $q/2^p$. Sijoitamme ensin yhtälöön (4.26) muuttujan x arvoiksi kokonaisluvut $\{n_j\}$, joilla yhtälö on ei-triviaali ($\phi(x)$ poikkeaa nollasta vain muutamassa kokonaislukupisteessä). Näin päädymme ominaisarvotehtävään $\phi = L\phi$, jonka tuloksena (ominaisvektorina) saamme selville suhteelliset erot funktion ϕ arvoissa niissä kokonaislukupisteissä, joissa $\phi(n_j) \neq 0$. Tässä vaiheessa mukana on siis vielä vapaa kerroin, jonka kiinnitämme vasta lopuksi normeerausehdon (4.27) avulla.

Seuraavaksi sijoitamme yhtälöön (4.26) muuttujan x arvoiksi äsken käytettyjen kokonaislukujen puolivälissä olevat luvut $\{n_j \pm 1/2\}$. Tällöin syntyvän yhtälöryhmän vasemmalle puolelle jää joukko vielä tuntemattomia funktion ϕ arvoja, kun taas oikealla puolella esiintyy pelkästään funktion ϕ arvoja kokonaislukupisteissä. Mutta jälkimmäiset arvot ovat nyt kaikki tunnettuja, sillä nollasta poikkeavat arvot ratkaistiin (vakiota vaille) edellisessä vaiheessa. Uusi yhtälöryhmä on valmiiksi ratkaistussa muodossa, siis aiemmin tuntemattomat arvot on ilmaistu jo tunnettujen arvojen avulla. Tämän vaiheen jälkeen tunnemme funktion ϕ arvot pisteissä $q/2$.

Samalla periaatteella voimme jatkaa ottamalla joka kerta käyttöön uudet muuttujan x arvot aiemmin käytettyjen arvojen puolivälistä. Kun lopulta katsomme tuntevamme funktion ϕ arvon riittävän monessa pisteessä, voimme normeerata lopputuloksen niin, että yhtälö (4.27) on voimassa.

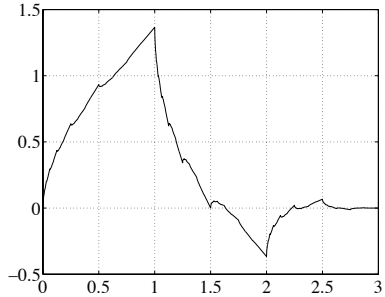
Kun skaalausfunktion $\phi(x)$ kuvaaja tunnetaan, aallokefunktioiden $\psi_j(x)$ kuvaajat voi selvittää yhtälön (4.28) avulla.

Esimerkki 4.12.3 Jatkamme Daubechies'n matriisin D_2 tutkimista. Skaalautuvuusyhtälö (4.26) saa nyt muodon

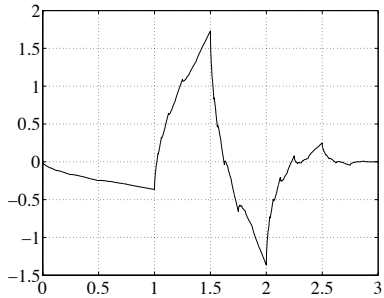
$$\begin{aligned}\phi(x) \quad &= \tfrac{1}{4}\left((1+\sqrt{3})\phi(2x) + (3+\sqrt{3})\phi(2x-1)\right. \\ &\quad \left. +(3-\sqrt{3})\phi(2x-2) + (1-\sqrt{3})\phi(2x-3)\right).\end{aligned}$$

Aallokefunktiolle $\psi(x)$ pätee samantyyppinen yhtälö, jossa kertoimiksi on valittu matriisin D_2 toisen rivin alkiot. Olemme jättäneet alaindeksin 1 pois funktiosta $\psi(x)$, koska tapauksessa $m = 2$ aallokefunktioita on vain yksi.

Edellä selostetun iterointimenetelmän tulokset 15 kierroksen jälkeen esitetty kuvassa 4.16. Karkeasti arvioiden virhe $|\phi(x) - \phi^{\nu}(x)|$ puolittuu jokaisella iteraatiokierroksella. Koska alkuarvauksessa virhe on suuruusluokkaa 1, viidentoista iteraatiokierroksen jälkeen virhe on periaatteessa $\approx 2^{-15} \approx 3 \cdot 10^{-5}$. Käytännössä kokonaislukupisteiden $x = 0, 1, 2, 3$ ympäristössä virhe voi olla jopa kymmenkertainen.



a: *Daubechies'n skaalausfunktion $\phi(x)$ approksimaatio $\phi^{15}(x)$.*



b: *Daubechies'n aallokkeen $\psi(x)$ approksimaatio.*

Kuva 4.16: *Daubechies'n matriisiin D_2 liittyvät funktiot $\phi(x)$ ja $\psi(x)$.*

Tutkimme vielä esimerkin vuoksi miten toinen tekstissä mainittu menetelmä funktion $\phi(x)$ luomiseksi toimii tässä tapauksessa. Tiedämme, että funktion ϕ kantaja on välillä $[0, 3]$. Koska ϕ on jatkuva, se saa arvon 0 kaikissa kokonaislukupisteissä lukuunottamatta pisteitä $x = 1$ ja $x = 2$. Aloitamme siis sijoittamalla yhtälöön (4.26) muuttujan x arvoiksi kokonaisluvut 1 ja 2 :

$$\begin{aligned}
\phi(1) &= \frac{1}{4}\left((1+\sqrt{3})\phi(2) + (3+\sqrt{3})\phi(1)\right) \\
\phi(2) &= \frac{1}{4}\left((3-\sqrt{3})\phi(2) + (1-\sqrt{3})\phi(1)\right) \\
\Rightarrow \phi(2) &= -\frac{(1-\sqrt{3})^2}{2}\phi(1).
\end{aligned}$$

Arvo $\phi(1)$ on siis toistaiseksi vapaa. Pyrimme tästä eteenpäin lausumaan funktion ϕ arvon kaikissa muissa pisteissä arvon $\phi(1)$ avulla.

Seuraavaksi sijoitamme muuttujalle x arvot $1/2$, $3/2$, ja $5/2$ ja saamme yhtälöt

$$\begin{aligned}
\phi(1/2) &= \frac{1}{4}(1+\sqrt{3})\phi(1) \\
\phi(3/2) &= \frac{1}{4}\left((3+\sqrt{3})\phi(2) + (3-\sqrt{3})\phi(1)\right) \\
\phi(5/2) &= \frac{1}{4}(1-\sqrt{3})\phi(2).
\end{aligned}$$

Koska ensimmäisen vaiheen nojalla osaamme lausua luvun $\phi(2)$ luvun $\phi(1)$ avulla, yhtälöt kertovat suoraan funktion ϕ arvon pisteissä $1/2$, $3/2$ ja $5/2$ luvun $\phi(1)$ monikertana.

Seuraavassa vaiheessa valitsemme muuttujan x arvoiksi luvut $1/4$, $3/4$, $5/4$, $7/4$, $9/4$ ja $11/4$, sitten luvut muotoa $(1+2n)/8$, $(1+2n)/16$, $\cdots$, $(1+2n)/2^p$. Lopulta tunnemme funktion ϕ arvon kaikissa pisteissä $x = q/2^p$, $x \in [0,3]$ luvun $\phi(1)$ avulla lausuttuna. Tämä kiinnitetään vaatimalla ehdon (4.27) toteutuminen.

4.12.3 Aallokeanalyysi

Olemme kirjan aiemmissa kappaleissa käyttäneet funktioiden L_2-approksimointiin erilaisia kantafunktiojärjestelmiä (esimerkiksi polynomit, trigonometriset funktiot). Myös aallokkeiden avulla on mahdollista etsiä funktioille L_2-approksimaatioita. Aallokkeita voi käyttää lisäksi näytteiden perusteella tapahtuvaan signaalien approksimointiin.

Määrittelemme ensin skaalausfunktion ϕ ja aallokefunktioiden ψ skaalatut ja siirretyt versiot kokonaisluvuille k ja r:

$$\phi_{rk} = m^{r/2}\phi(m^r x - k) \tag{4.30}$$

$$\psi_{j,rk} = m^{r/2}\psi_j(m^r x - k), \quad j = 1, \ldots, m-1. \tag{4.31}$$

Indeksi k yksinkertaisesti siirtää funktion kuvaajaa pitkin x-akselia. Indeksi r kertoo kuinka voimakkaasti muuttujan x muutokset vaikuttavat skaalattuun argumenttiin $m^r x - k$ ja sitä kautta funktion arvoon. Skaalausfunktion ja aallokefunktioiden kantajat ovat äärellisiä, eli funktiot saavat nollasta eroavia arvoja vain kun x on jollain äärellisellä välillä $[a,b]$. Kun r on suuri, skaalattu argumentti $m^r x - k$ käy läpi kaikki arvot väliltä $[a,b]$, samalla kun alkuperäinen argumentti x vaihtelee paljon pienemmällä välillä $[(a+k)/m^r, (b+k)/m^r]$. Funktioiden skaalatut versiot elävät siis sitä pienemmällä välillä, mitä suurempi indeksi r on. Vastaavasti jos r saa suuria negatiivisia arvoja, aallokkeet leviävät laajemmalle alueelle. Edessä oleva kerroin $m^{r/2}$ tasapainottaa normeerauksen, kun funktioiden kuvaajia tiivistetään.

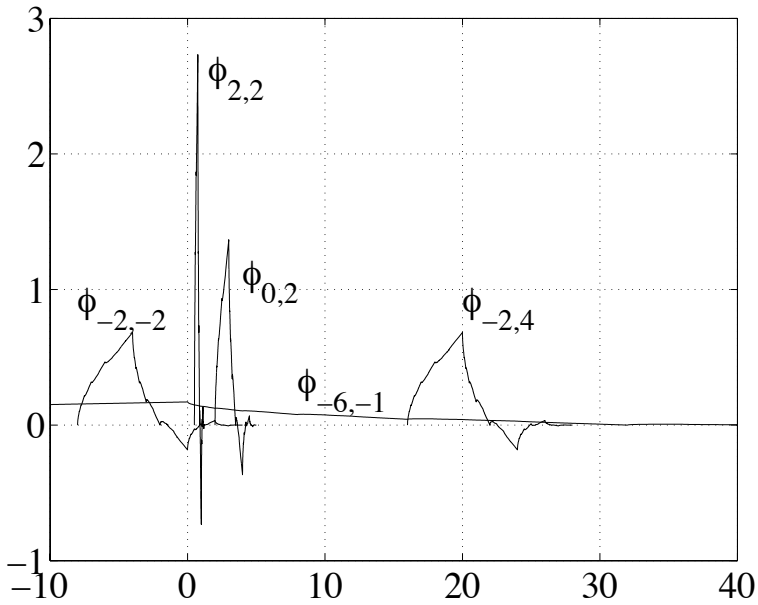
Kuvassa 4.17 on esitetty muutamia erilaisia indeksien r ja k arvoja vastaava Daubechies'n skaalausfunktion kopio ϕ_{rk}.

Edellä määritellyt funktioiden ϕ ja ψ siirretyt ja skaalatut versiot muodostavat matriisiin A liittyvän *aallokejärjestelmän* $\mathcal{W}[A]$:

$$\mathcal{W}[A] = \{\phi_{rk},\, r,\, k \in \mathbb{Z}\} \cup \left\{\psi_{j,rk},\, r,\, k \in \mathbb{Z},\, r \geq 0\, j = 1, \ldots, m-1\right\}.$$

Jokaisella kiinteällä indeksin r arvolla skaalausfunktion $\phi(x)$ kopiot $\phi_{rk}(x)$ virittävät funktioavaruuden V_r :

$$V_r = \left\{ \sum_{k \in \mathbb{Z}} c_{rk}\phi_{rk} \right\}.$$



Kuva 4.17: *Daubechies'n skaalausfunktion $\phi(x)$ kopioita $\phi_{rk}(x)$.*

Avaruus V_0 koostuu funktioiden $\phi_{0k}(x) = \phi(x-k)$ lineaarikombinaatioista. Seuraavan avaruuden V_1 alkiot rakentuvat funktioista $\phi_{1k}(x) = \sqrt{m}\phi(mx-k)$. Samaa ideaa soveltaen avaruuteen V_r kuuluvat ne funktiot $f(x)$, jotka ovat lausuttavissa muodossa

$$f(x) = m^{r/2} \sum_k c_{rk} \phi(m^r x - k).$$

Avaruudet V_r muodostavat nyt jonon toisensa sisältäviä funktioavaruuksia:

$$\cdots \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \ldots.$$

Ainoa kaikille avaruuksille V_r yhteinen alkio on nollafunktio

$$\bigcap_{r=-\infty}^{\infty} V_r = 0. \tag{4.32}$$

Valitsemalla hyvin isoja paramterin r arvoja saamme yksittäisen aallokkeen kantajan puristettua hyvin pieneen tilaan. Niinpä mitä tahansa avaruuden $L_2(\mathbb{R})$ funktiota on mahdollista approksimoida mielivaltaisen tarkasti avaruuksien yhdisteessä $\cup_{r=-\infty}^{\infty} V_r$. Lisäksi kuhunkin avaruuteen V_r kuuluvat täsmälleen ne funktiot $f(x)$, joille pätee $f(mx) \in V_{r+1}$.

Edellä luetellut ominaisuudet toteuttavaa funktioavaruusjonoa kutsutaan *moniresoluutioanalyysiksi* (multiresolution analysis). Moniresoluutioanalyysi soveltuu hyvin sellaisen ilmiön kuvaamiseen, jossa asioita tapahtuu monessa eri mittakaavassa. Valitsemalla alkuperäisen aallokematriisin A al-

kiot sopivasti voidaan vaikuttaa vastaavan skaalausfunktion $\phi(x)$ ja aallokefunktion $\psi(x)$ tuottaman moniresoluutioanalyysin approksimaatio-ominaisuuksiin, kuten funktioiden sileyteen (kuinka monta kertaa derivoituvia tuloksia halutaan).

Skaalausfunktion $\phi(x)$ kopioiden $\phi_{rk}(x)$ avulla pystymme siis luomaan avaruuden $L_2(\mathbb{R})$ erään moniresoluutioanalyysin $\{V_r\}$. Aallokefunktioiden $\psi_{j,rk}(x)$ avulla on mahdollista määritellä vastaavasti ensin avaruudet

$$W_{j,r} = \left\{ \sum_{k\in\mathbb{Z}} c_{j,rk}\psi_{j,rk} \right\}.$$

ja näiden suorina summina edelleen avaruudet $W_r = \bigoplus_{j=1}^{m-1} W_{j,r}$.

Jos siirretyt skaalausfunktiot $\phi(x-k)$ ovat keskenään ortonormaaleja, jonot V_r ja W_r täydentävät toisiaan seuraavassa mielessä. Moniresoluutioanalyysin ehtojen mukaan $V_r \subset V_{r+1}$. Nyt aliavaruuden V_r ortogonaalikomplementti avaruudessa V_{r+1} on täsmälleen W_r eli $V_{r+1} = V_r \bigoplus W_r$. Tämä tarkoittaa sitä, että jokainen avaruuden V_{r+1} alkio $f(x)$ voidaan hajottaa yksikäsitteisesti kahden muun funktion summaksi, joista toinen kuuluu avaruuteen V_r ja toinen avaruuteen W_r.

Sama päättely toimii tietysti jokaisella tasolla r, joten saamme aikaiseksi periaatteessa päättymättömän ketjun

$$V_{r+1} = V_r \bigoplus W_r = V_{r-1} \bigoplus W_{r-1} \bigoplus W_r = V_{r-2} \bigoplus W_{r-2} \bigoplus W_{r-1} \bigoplus W_r \ldots \tag{4.33}$$

Jos annamme ylläolevassa yhtälössä indeksin r kasvaa ja jaamme funktioavaruuden osiin r kertaa, päädymme lopulta tulokseen

$$\lim_{r\to\infty} V_{r+1} = V_0 \bigoplus_{r=0}^{\infty} W_r. \tag{4.34}$$

Yhtälön vasemmalla puolella olevassa avaruudessa pystytään esittämään mikä tahansa avaruuden $L_2(\mathbb{R})$ funktio mielivaltaisen tarkasti (moniresoluutioanalyysin ominaisuus). Niinpä kaikissa aallokejärjestelmissä voidaan kirjoittaa yhtälön (4.34) mukainen hajotelma funktioille $f(x) \in L_2(\mathbb{R})$:

$$f(x) = \sum_{k=-\infty}^{\infty} c_{0k}\phi_k(x) + \sum_{j=1}^{m-1}\sum_{r=0}^{\infty}\sum_{k=-\infty}^{\infty} c_{j,rk}\psi_{j,rk}(x). \tag{4.35}$$

Skaalausfunktiosta $\phi(x)$ tarvitsemme siis vain kertaluvun $r = 0$ siirretyt kopiot, sen sijaan aallokefunktioiden $\psi_{j,rk}(x)$ osalta indeksi r käy läpi kaikki arvot 0, 1, 2,...

Kuten aina L_2-approksimoinnin yhteydessä, kertoimet c_{0k}, $c_{j,rk}$ ovat lasket-

tavissa kaavoista

$$c_{0k} = \int_{-\infty}^{\infty} f(x)\phi_{0k}(x)\,dx \tag{4.36}$$

$$c_{j,rk} = \int_{-\infty}^{\infty} f(x)\psi_{j,rk}(x)\,dx \tag{4.37}$$

Suoraviivainen tapa kertoimien laskemiseksi on korvata integraalit numeerisilla kvadratuureilla (numeerisesta integroinnista kerrotaan luvussa 5). Pystymme nimittäin laskemaan funktioiden $\phi(x)$ ja $\psi(x)$ arvot sopivassa pisteistössä $\{q/2^p\}$ aiemmin kuvatulla iteraatiomenettelyllä, ja näiden arvojen avulla voimme lausua myös funktioiden $\phi_{0k}(x)$, $\psi_{j,rk}(x)$ arvot sopivasti muunnetuissa pisteistöissä.

Kaikkia hajotelman kertoimia ei kuitenkaan tarvitse laskea suoraan määritelmistä. Riittää, kun tunnemme kertoimet $c_{j,Rk}$ jollain tasolla R. Aallokkejärjestelmän $\mathcal{W}[A]$ ja matriisin A ominaisuuksien avulla on nimittäin mahdollista johtaa tehokas menetelmä (*Mallat'n algoritmi*) kertoimien c_{0k}, $c_{j,rk}$ laskemiseksi kaikilla tasoilla $r < R$.

Käymme menetelmän idean seuraavaksi läpi tärkeimmässä erikoistapauksessa $m = 2$. Koska tällöin varsinaisia aallokefunktioita on vain yksi ($j = 1$), otamme käyttöön merkinnät $\psi_{rk} = \psi_{1,rk}$ ja $d_{rk} = c_{1,rk}$. Lisäksi merkitsemme kaksirivisen aallokematriisin A ensimmäisen rivin alkioita a_{0k} lyhyemmin vain a_k ja toisen rivin alkioita a_{1k} symbolein b_k.

Vaikka hajotelmassa (4.35) indeksi r saa kaikki kokonaislukuarvot, käytännössä joudumme tietysti katkaisemaan kehitelmän johonkin äärelliseen lukuun R. Tämä asettaa ylärajan sille kuinka pieniä yksityiskohtia pystymme hajotelmassa näkemään. Funktion likimääräinen esitys summana äärellisestä määrästä aallokkeita on funktion *diskreetti aallokemuunnos*.

Moniresoluutioanalyysin ominaisuuksien ansiosta pätee

$$V_R = V_{R-1} \bigoplus W_{R-1}.$$

Niinpä funktion $f(x)$ esitys avaruudessa V_R on sama kuin sen esitys ylläolevan yhtälön oikealla puolella olevassa avaruuksien V_{R-1} ja W_{R-1} suorassa summassa:

$$\sum_{k\in\mathbb{Z}} c_{Rk}\phi_{Rk}(x) = \sum_{k\in\mathbb{Z}} c_{R-1,k}\phi_{R-1,k}(x) + \sum_{k\in\mathbb{Z}} d_{R-1,k}\psi_{R-1,k}(x). \tag{4.38}$$

Käyttämällä hyväksi ortogonaalisuusominaisuuksia yhtälön (4.38) oikealla puolella esiintyvät kertoimet $c_{R-1,k}$, $d_{R-1,k}$ ovat lausuttavissa vasemman puolen kertoimien c_{Rk} avulla ja tietysti päinvastoin. Aallokejärjestelmässä $\mathcal{W}[A]$ nämä kaavat sievenevät muotoon

$$c_{R-1,k} = \frac{1}{\sqrt{2}} \sum_l c_{Rl} a_{l-2k} \tag{4.39}$$

$$d_{R-1,k} = \frac{1}{\sqrt{2}} \sum_l c_{Rl} b_{l-2k} \tag{4.40}$$

ja käänteiseen suuntaan

$$c_{Rk} = \frac{1}{\sqrt{2}} \sum_l c_{R-1,l} a_{k-2l} + d_{R-1,l} b_{k-2l}. \tag{4.41}$$

Oleellisesti samat kaavat ovat voimassa luonnollisesti kaikkien kertalukujen r ja $r-1$ välillä. Niinpä voimme aloittaa tiheän mittakaavan hajotelmasta $\sum_k c_{Rk}\phi_{Rk}(x)$ ja laskea nopeasti kaikki alempien tasojen hajotelmiin liittyvät kertoimet kaavojen (4.39) avulla. Näin saamme lopulta funktion $f(x)$ aallokehajotelmalle kaavasta (4.35) version, jossa summaus indeksin r suhteen on katkaistu äärelliseen arvoon $R-1$:

$$f(x) \approx \sum_{k=-\infty}^{\infty} c_{0k}\phi_k(x) + \sum_{j=1}^{m-1} \sum_{r=0}^{R-1} \sum_{k=-\infty}^{\infty} c_{j,rk}\psi_{j,rk}(x). \tag{4.42}$$

Indeksin k suhteen summauksen rajat ovat edelleen käytännön kannalta ongelmalliset. Jos funktio $f(x)$ on pulssimainen eli sen kantaja on äärellinen, niin vain äärellinen määrä kertoimista c_{0k}, $c_{j,rk}$ poikkeaa nollasta, ja myös indeksi k saa vain äärellisen määrän erilaisia arvoja. Jos funktio $f(x)$ saa nollasta poikkeavia arvoja rajoittamattoman suurilla muuttujan x arvoilla, joudumme päättämään jonkin kriteerin, jonka perusteella jätämme osan kertoimista pois summasta.

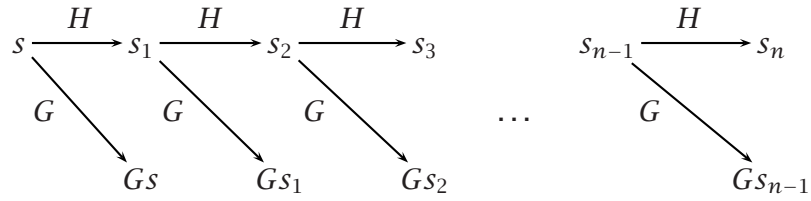
Erityisen kätevä Mallat'n algoritmi on signaalinkäsittelyssä. Ensin signaalijonoa $s = \{s_k\}$, $k = 0, 1, \ldots, N_s$ täydennetään joko sopivalla määrällä nollia (äärellinen, pulssimainen signaali) tai lisäämällä jonon loppuun alkupään alkioita s_0, s_1, s_2 (jaksollinen signaali). Alkuperäinen signaali $\{s_j\} \in V_r$ jaetaan seuraavaksi operaattoreilla H ja G kahteen osaan $Hs \in V_{r-1}$ (matalat taajuudet) ja $Gs \in W_{r-1}$ (korkeat taajuudet). Jos edelleen pidämme kiinni oletuksesta $m = 2$, niin operaattoreiden H ja G vaikutus signaaliin s on esitettävissä lausekkeina

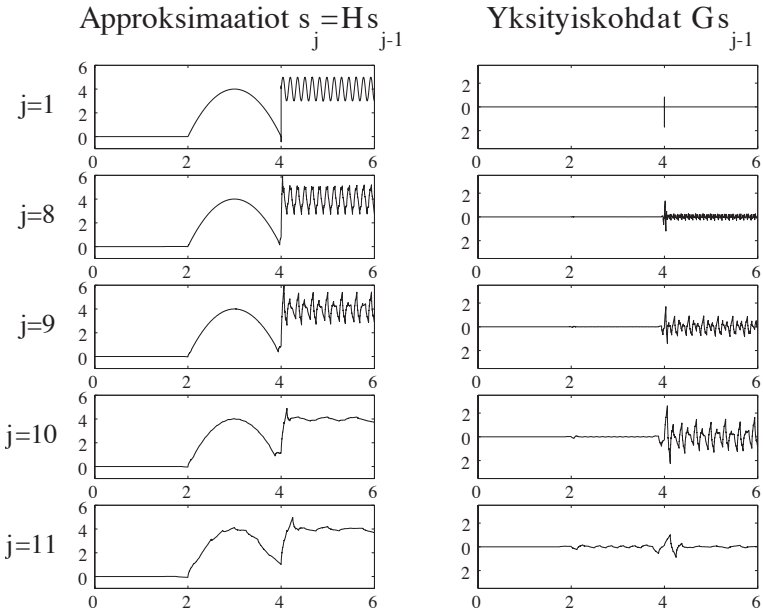
$$\{Hs\}_j = \sqrt{2} \sum_{l=2j}^{2j+2g-1} s_l a_{l-2j} \tag{4.43}$$

$$\{Gs\}_j = \sqrt{2} \sum_{l=2j}^{2j+2g-1} s_l b_{l-2j}. \tag{4.44}$$

Jaksolliselle signaalille uusien osien Hs ja Gs pituus on puolet alkuperäisen signaalin pituudesta. Pulssimaisen signaalin tapauksessa uusien osien päätyihin syntyy pari ylimääräistä alkiota.

Kertaalleen suodatetun signaalin $s_1 = Hs \in V_{r-1}$ voi jakaa jälleen kahteen osaan $s_2 = Hs_1 \in V_{r-2}$ ja $Gs_1 \in W_{r-2}$. Kierroksella p signaalista s_{p-1} häviää yksityiskohtia, jotka siirtyvät vektoriin Gs_{p-1}. Uusi, edellistä karkeampi kuvaus signaalista siirtyy vektoriin $s_p = Hs_{p-1}$.





Kuva 4.18: *Signaalin jakautuminen approksimaatioihin ja yksityiskohtiin*

Esimerkki 4.12.4 Olkoon signaali s määritelty pisteissä $t_j = j\Delta t$, $\Delta t = 2^{-13}$ seuraavasti:

$$s(t_j) = \begin{cases} 0 & \text{kun } 0 \le t_j < 2, \\ 8t_j - 4t_j^2 & \text{kun } 2 \le t_j \le 4, \\ 4 + \sin(40t_j) & \text{kun } 4 < t_j \le 6. \end{cases}$$

Kuvasarjassa 4.18 on esitetty kuinka operaattori H tuottaa alkuperäisestä signaalista yhä karkeampia approksimaatioita. Vastaavasti operaattori G poistaa signaalista kierros kierrokselta suurempia yksityiskohtia.

Ensimmäisessä vaiheessa operaattori G poimii signaalista vain epäjatkuvuuskohdan pisteessä $t_j = 4$. Koska jonon Gs alkiot ovat epäjatkuvuuskohdan ympäristöä lukuunottamatta nollia, operaattori H palauttaa alkuperäisen signaalin lähes ennallaan ($s = Hs + Gs$).

Analysointia jatkettaessa approksimaatiot s_j menettävät ensin tarkkuutta välillä $[4, 6]$, jossa signaalin muutokset ovat nopeimpia. Muutaman ensimmäisen kierroksen aikana operaattorin G näkemät yksityiskohdat ovat skaalaltaan oleellisesti pienempiä kuin signaalin s yksityiskohdat. Kahdeksannen tason hajotelmassa operaattorin G erottelukyky alkaa vaikuttaa signaalin s siniaaltoosuuteen. Approksimaatiossa s_9 erottuu vielä värähtelyä välillä $[4, 6]$, mutta seuraavassa vaiheessa melkein kaikki värähtely on siirtynyt yksityiskohdat sisältävään vektoriin Gs_9. Approksimaation $s_{10} = Hs_9$ kuvaaja välillä $[4, 6]$ sisältää nousun arvoon 4, josta eteenpäin signaali jatkuu lähes vakioarvoisena.

Toistaiseksi olemme käsitelleet L_2-approksimaatioita, jotka ovat oleellisesti *projektioita*. Approksimoitava funktio $f(x)$ projisoidaan kantafunktioiden virittämään avaruuteen V_R, ja kertoimet c_{Rk} lasketaan sisätuloista, eli

$$f(x) \approx \tilde{f}(x) = \sum_k \left(\int_{\mathbb{R}} f(x)\phi_{Rk}(x)\,dx \right) \phi_{Rk}(x). \tag{4.45}$$

Aallokkeita voi käyttää approksimointiin myös paljon suoraviivaisemmalla tavalla eräissä erikoistapauksissa. On nimittäin mahdollista valita sarjakehitelmän kertoimiksi suoraan approksimoitavan funktion arvot joissain pisteissä. Jatkamme edelleen aallokesysteemien tarkastelua tapauksessa $m = 2$, ja keskitämme matriisin A alkiot indeksin 0 ympärille.

$$A = \begin{pmatrix} \dots & a_{-1} & a_0 & a_1 & \dots \\ \dots & b_{-1} & b_0 & b_1 & \dots \end{pmatrix}, \quad b_k = (-1)^k a_{1-k}.$$

Matriisiin A liittyvä aallokesysteemi $\mathcal{W}[A]$ on *kertaluvun N Coifmanin aallokesysteemi*, jos seuraavat momenttiehdot pätevät:

$$\int \phi(x)\,dx = 1 \tag{4.46}$$

$$\int x^l \phi(x)\,dx = 0, \quad l = 1, \dots, N \tag{4.47}$$

$$\int x^l \psi(x)\,dx = 0, \quad l = 0, \dots, N. \tag{4.48}$$

Momenttiehdot ovat approksimoinnin kannalta tärkeät. Kertaluvun N momenttiehdot toteuttavan aallokesysteemin avulla on mahdollista esittää tarkasti kaikki astetta N olevat polynomit $p_N(x)$.

Tarkastellaan N kertaa derivoituvaa funktiota $f(x)$, jolla on äärellinen kantaja (funktio f siis häviää jonkin äärellisen välin ulkopuolella). Jos funktion $f(x)$ arvot tunnetaan tasavälisessä pisteistössä $x_k = k/2^R$, niin approksimaatiokertoimiksi c_{Rk} on mahdollista valita suoraan luvut $f(x_k) = f(k/2^R)$ painotettuna sopivalla vakiolla, kun kantafunktioina käytetään kertaluvun N Coifmanin aallokkeita. Tarkemmin sanottuna funktion f astetta R oleva approksimaatio $S^R(f)(x)$ on

$$S^R(f)(x) = \frac{1}{2^{R/2}} \sum_k f\left(\frac{k}{2^R}\right) \phi_{Rk}(x). \tag{4.49}$$

Approksimaation virheelle $\|f(x) - S^R(f)(x)\|$ pätee arvio

$$\|f(x) - S^R(f)(x)\|_{L_2} \leq C 2^{-RN},$$

missä vakio C riippuu vielä funktiosta f ja käytetystä aallokesysteemistä.

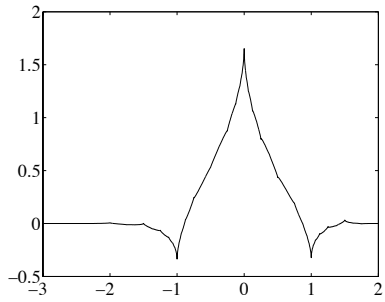
Siis mitä suuremmalla kertaluvun N arvolla momenttiehdot toteutuvat, sitä tarkempia approksimaatioita aallokkeilla on mahdollista muodostaa. Coifmanin aallokkeita tuottavat matriisit A joudutaan ratkaisemaan epälineaarisista yhtälöistä. Kun N on hyvin pieni ($N = 0, 1$), ratkaiseminen onnistuu

analyyttisesti, mutta yleensä joudutaan turvautumaan Newtonin iteraatioon. Korkeampien kertalukujen yhteydessä ($N > 10$) matriisin A alkioiden määrittäminen käy pyöristysvirheiden vuoksi numeerisestikin mahdottomaksi.

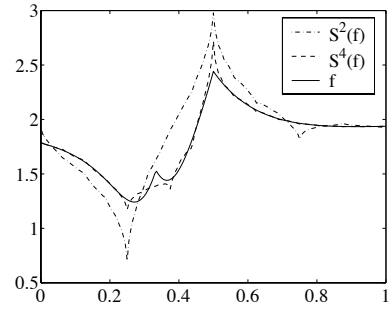
Esimerkki 4.12.5 Etsimme matalan kertaluvun Coifmanin aallokkeen skaalausfunktioon (kuva 4.19a) perustuvia approksimaatioita $S^j(f)(x)$ funktiolle

$$f(x) = e^{(1-2/(x-1/3)^2)/100} + e^{-8|x-1/3|} + e^{-8|x-1/2|}.$$

Approksimaatiota $S^j(f)$ varten funktion f arvot on tunnettava kaikkiaan 2^j pisteessä. Pisteissä $x = 1/3$ ja $x = 1/2$ funktiolla $f(x)$ on derivaatan epäjatkuvuuskohdat, joiden ympäristössä on odotettavissa vaikeuksia.



a: *Coifmanin skaalausfunktio $\phi(x)$.*



b: *Funktio $f(x)$ ja approksimaatiot $S^2(f)(x)$ ja $S^4(f)(x)$.*

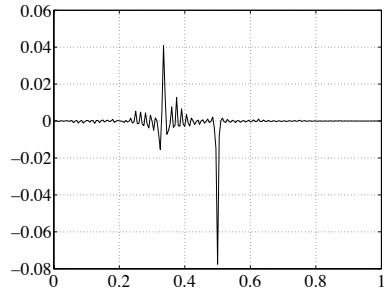
Kuva 4.19: *Coifmanin skaalausfunktio ja sen käyttö approksimointiin.*

Kuvassa (4.19b) on esitetty funktio $f(x)$ ja sen approksimaatiot $S^2(f)(x)$ ja $S^4(f)(x)$, jotka siis perustuvat funktion $f(x)$ arvoihin $2^2 = 4$ ja $2^4 = 16$ tasavälisesti valitussa pisteessä. Approksimaatiot löytävät kohdassa $x = 0.5$ sijaitsevan maksimin. Kohdassa $x = 1/3$ sijaitseva lokaali maksimi sen sijaan on niin kapea, ettei se erotu vielä näissä approksimaatioissa.

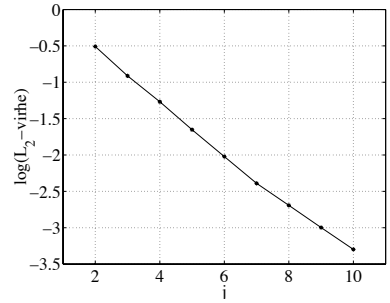
Kuvassa (4.20a) näemme approksimaatiovirheen $f(x) - S^6(f)(x)$ (64 näytepistettä). Suurimmat virheet keskittyvät pisteiden $x = 1/3$ ja $x = 1/2$ ympäristöön. Kuvaan (4.20b) on piirretty approksimaation L_2-virheen logaritmi $\log(\|f(x) - S^j(f)(x)\|_2)$ parametrin j funktiona.

4.13 Moniulotteinen interpolointi ja approksimointi

Usean muuttujan funktion $f(x^1, ..., x^d) : \mathbb{R}^d \mapsto \mathbb{R}$ interpolointi on huomattavasti mutkikkaampi tehtävä kuin yhden muuttujan funktion $f(x)$ interpolointi. Suurin syy tähän on moniulotteisissa avaruuksissa mahdollinen



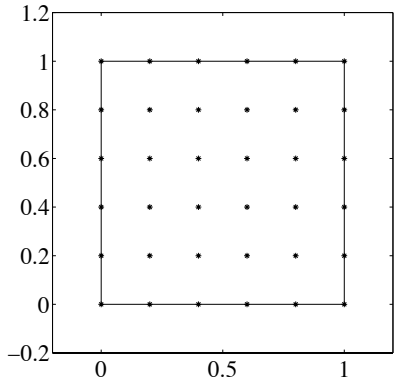
a: *Approksimaatiovirhe* $f(x) - S^6(f)(x)$.



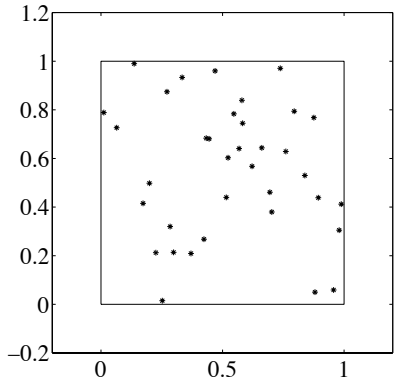
b: *Approksimaatiovirheen suppeneminen (logaritminen skaala).*

Kuva 4.20: *Approksimaation virhe.*

geometrinen vaihtelevuus. Yhdessä dimensiossa kaikki interpolointipisteet ovat samalla suoralla (korkeintaan pisteiden etäisyydet vaihtelevat) ja interpolointialue on aina väli. Jo tasossa ($d = 2$) pisteiden sijainti voi vaihdella säännöllisestä neliöhilasta täysin rakenteettomaan (kuva 4.21). Lisäksi interpolointialueen muoto saattaa aiheuttaa ongelmia.



a: *Säännöllinen neliöhila.*



b: *Satunnainen interpolointipisteistö.*

Kuva 4.21: *Interpolointihiloja.*

Perustehtävä on kuitenkin sama kuin yksiulotteisessa tapauksessa. Interpoloitavan funktion $f(\mathbf{x}) = f(x^1, x^2, ..., x^d)$ arvot f_I tunnetaan joissain pisteissä $\{\mathbf{x}_I\}$. Tällöin on määriteltävä interpolantti $\tilde{f}(\mathbf{x})$, joka toteuttaa ehdot

$$\tilde{f}(\mathbf{x}_I) = f_I. \tag{4.50}$$

Jos interpolointipisteiden sijainti moniulotteisessa avaruudessa $\mathbb{R}^d$ on hyvin

satunnaista, ei ole välttämättä olemassa yleispätevää sääntöä niiden indeksoimiseen (kuva 4.21b). Tämän korostamiseksi edellä on käytetty merkintää I kuvaamaan pisteiden indeksointia, eikä yksiulotteisesta tapauksesta tuttua indeksiä i, joka viittaa "luonnolliseen" numerojärjestykseen.

4.13.1 Tulokaavat säännöllisessä hilassa

Yritämme ensin yleistää Lagrangen interpoloinnin moniulotteisiin avaruuksiin. Tämä on verrattain suoraviivaista, jos interpolointipisteet muodostavat säännöllisen nelikulmiohilan kuten kuvan 4.21a tasoa koskevassa esimerkissä. Tällöin kutakin muuttujaa x^k, $k = 1, ...d$ kohden voidaan ensin muodostaa riippumattomat yhden muuttujan apupolynomit

$$\phi_i^k(x^k) = \prod_{\substack{j=0 \\ j\neq i}}^{n_k} \frac{x^k - x_j^k}{x_i^k - x_j^k}, \quad i = 1, ..., n_k.$$

Kuten aiemmin, apufunktiot saavat arvon 1 pisteessä x_i^k ja arvon 0 kaikilla muilla muuttujan x^k arvoilla $x_1^k, x_2^k, ..., x_{n_k}^k$. Jokaiseen hilapisteeseen $\mathbf{x}_I = (x_{i_1}^1, x_{i_2}^2, ..., x_{i_d}^d)$ liittyvä d muuttujan kantafunktio $\Phi_I(\mathbf{x})$ puolestaan saadaan kertomalla keskenään vastaavat yksiulotteiset apufunktiot $\phi_{i_k}^k(x^k)$:

$$\Phi_I(\mathbf{x}) = \phi_{i_1}^1(x^1)\phi_{i_2}^2(x^2) \cdots \phi_{i_d}^d(x^d).$$

Lopullinen interpolantti muodostetaan laskemalla tunnetuilla funktion arvoilla f_I painotettu summa kantafunktioista Φ_I:

$$\tilde{f}(\mathbf{x}) = \sum_I f_I \Phi_I(\mathbf{x}). \tag{4.51}$$

Myös interpolointivirheen arviointi hankaloituu muuttujien lukumäärän lisääntyessä. Eri muuttujien suhteen erikseen laskettujen virheiden lisäksi tarkassa analyysissä tulisi ottaa huomioon korkeamman kertaluvun virhetermit, jotka syntyvät useamman interpolaation yhteisvaikutuksesta.

Esimerkki 4.13.1 Interpoloimme funktiota $f(x, y) = e^{xy} - x^2 + \sin(\pi y)$ säännöllisessä tasohilassa, jossa muuttuja x saa arvot 0 ja 1 ja y arvot 0, 1/2 ja 1. Laskemme ensin yhden muuttujan apufunktiot:

$$\begin{aligned}
\phi_0^1(x) &= \frac{x-1}{0-1} &&= 1 - x \\
\phi_1^1(x) &= \frac{x-0}{1-0} &&= x \\
\phi_0^2(y) &= \frac{y-1/2}{0-1/2} \cdot \frac{y-1}{0-1} &&= 2y^2 - 3y + 1 \\
\phi_1^2(y) &= \frac{y-0}{1/2-0} \cdot \frac{y-1}{1/2-1} &&= -4y^2 + 4y \\
\phi_2^2(y) &= \frac{y-0}{1-0} \cdot \frac{y-1/2}{1-1/2} &&= 2y^2 - y.
\end{aligned}$$

Näistä muodostetaan kantafunktiot $\Phi_I(x,y)$ kertolaskun avulla. Indeksointi I riippuu tavasta, jolla interpolaatiopisteet päätetään numeroida. Voimme esimerkiksi sopia, että indeksin I arvo 0 vastaa pistettä $(0,0)$, jolloin

$$\Phi_0(x,y) = (1-x)(2y^2-3y+1).$$

Vastaavasti muihin interpolaatiopisteisiin liittyvät kantafunktiot ovat

$$\begin{array}{lcll}
\Phi_1(x,y) &=& x(2y^2-3y+1) & \text{piste } (1,0)\\
\Phi_2(x,y) &=& (1-x)(-4y^2+4y) & \text{piste } (0,1/2)\\
\Phi_3(x,y) &=& x(-4y^2+4y) & \text{piste } (1,1/2)\\
\Phi_4(x,y) &=& (1-x)(2y^2-y) & \text{piste } (0,1)\\
\Phi_5(x,y) &=& x(2y^2-y) & \text{piste } (1,1).
\end{array}$$

Funktio $f(x,y)$ saa interpolaatiopisteissä arvot

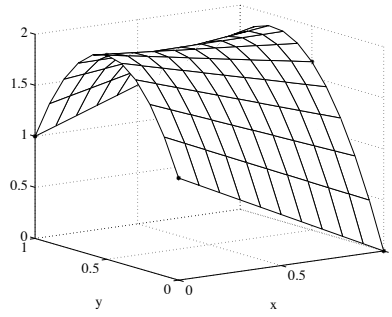
$$\begin{array}{lllllllll}
f(0,0) &=& 1, & f(0,1/2) &=& 2, & f(0,1) &=& 1,\\
f(1,0) &=& 0, & f(1,1/2) &=& \sqrt{e}, & f(1,1) &=& e-1.
\end{array}$$

Interpolaatio $\tilde{f}(x,y)$ on siis

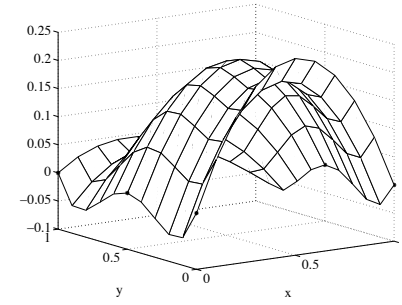
$$\begin{aligned}
\tilde{f}(x,y) &= 1\cdot(1-x)(2y^2-3y+1) + 0\cdot x(2y^2-3y+1)\\
&\quad + 2\cdot(1-x)(-4y^2+4y) + \sqrt{e}\cdot x(-4y^2+4y)\\
&\quad + 1\cdot(1-x)(2y^2-y) + (e-1)\cdot x(2y^2-y).
\end{aligned}$$

Kuvaan 4.22a on hahmoteltu interpolantin $\tilde{f}$ määrittämä pinta. Kiinteällä muuttujan y arvolla interpolantti käyttäytyy lineaarisesti muuttujan x funktiona. Jos taas x kiinnitetään, interpolantin $\tilde{f}$ poikkileikkaukset ovat y-akselin suuntaisia paraabelinkaaria.

Kuvassa 4.22b puolestaan näkyy interpolointivirhe $f-\tilde{f}$. Varsinaisissa interpolointipisteissä virhe häviää kuten pitääkin. Suurimmillaan virheen itseisarvo on pisteessä $(0.5,1)$, jossa $f-\tilde{f}=0.25$.



a: *Funktion* $f(x,y) = e^{xy} - x^2 + \sin(\pi y)$ *interpolaatio* $\tilde{f}(x,y)$.



b: *Interpoloinnin virhe* $f-\tilde{f}$.

Kuva 4.22: *Interpolointi säännöllisessä tasohilassa.*

4.13.2 Elementtikohtaiset interpolaatiot

Edellä kuvattu Lagrangen interpoloinnin yleistys vaatii datalta huomattavaa säännönmukaisuutta. Jos interpoloitavan funktion arvot selvitetään esimerkiksi mittausten kautta, voi olla mahdotonta järjestää mittauspisteet saannöllisen hilan muotoon. Lisäksi jo yhden muuttujan funktioiden kohdalla saimme huomata, että interpolantin asteluvun nosto datapisteiden lukumäärän kasvaessa johtaa yleensä katastrofaalisiin seurauksiin.

Seuraavaksi esitettävä simplekseihin perustuva elementtimenettely on paloittaisen interpoloinnin yleistys. Se tarjoaa osittaisen ratkaisun epäsäännöllisen aineiston interpolointiin. Se sopii kuitenkin edelleen paremmin tilanteisiin, joissa interpoloitavan funktion arvo $f(\mathbf{x})$ on saatavilla mielivaltaisessa pisteessä $\mathbf{x}$.

Jaamme interpolointialueen pieniin osiin eli *elementteihin*, ja muodostamme jokaista osaa varten oman matala-asteisen interpolantin. Alueen jakamisen osiin voi tietysti tehdä lukemattomin eri tavoin, mikä vaikuttaa syntyvän paloittaisen interpolantin ominaisuuksiin. Yleisessä tapauksessa käytetään tavallisesti d-ulotteisen avaruuden *simpleksejä*. Yksiulotteinen simpleksi on jana, kaksiulotteinen kolmio ja kolmiulotteinen tetraedri. Varsinkin tapauksessa $d = 2$ (taso) puhutaan alueen *kolmioinnista*. Kolmiointia varten on kehitetty monia algoritmeja. Numeriikan kannalta kolmiointi on pitkälti ohjelmointitekninen ongelma, joten tilanpuutteen vuoksi emme paneudu tässä yhteydessä tarkemmin kolmiointimenetelmiin.

Lagrangen interpolointipolynomien muodostaminen d-ulotteisessa simpleksissä on helppoa, kunhan interpolointipisteet valitaan oikein. Olkoot simpleksin kärkipisteet $\mathbf{t}_j$, $j = 1, 2, .., d+1$. Lisäksi määrittelemme vektorin $\mathbf{m}$, jonka alkioina m_j on $(d+1)$ kappaletta ei-negatiivisia kokonaislukuja. Jos interpolointipolynomien asteluku on k, vaaditaan vektorin $\mathbf{m}$ alkioilta ominaisuus $\sum_j m_j^d \leq k$. Viimeisen alkion m_{d+1} arvoksi asetetaan $k - \sum_j m_j^d$.

Kun lukujen m_j annetaan käydä läpi kaikki sallitut $(d+1)$ luvun kombinaatiot (yhteensä $\binom{k+d}{d}$ kappaletta), niin lauseke

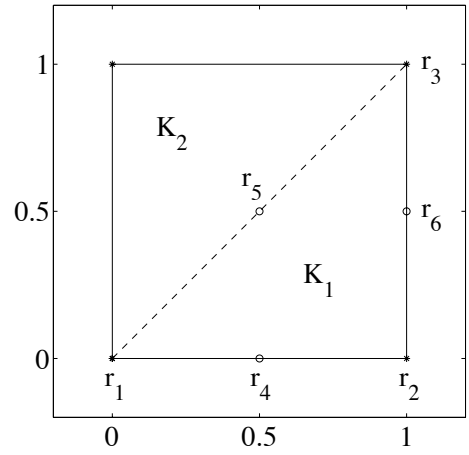
$$\mathbf{r}_I = \frac{1}{k} \sum_{j=1}^{d+1} m_j \mathbf{t}_j$$

puolestaan käy läpi joukon simpleksin pisteitä (joko sisäpisteitä tai reunapisteitä). Indeksi I riippuu järjestyksestä, jossa yhdistelmät käydään läpi.

Edellä muodostettu pistejoukko sopii Lagrangen interpoloinnin *vapausasteiksi*. Tämä tarkoittaa, että Lagrangen interpoloinnissa tarvittavat apupolynomit $l_I(\mathbf{x})$ saavat arvon 1 pisteessä $\mathbf{x} = \mathbf{r}_I$ ja arvon 0 kaikissa muissa interpolointipisteissä.

Jos on $k \geq 2$, niin edellisessä menetelmässä käytetään pisteitä, jotka eivät ole simpleksien kärjissä — siis pisteitä, jotka eivät kuulu alkuperäiseen interpolaatiodataan. Tämä luonnollisesti heikentää menetelmän käyttökelpoisuutta. Tässä tilanteessa on kaksi mahdollisuutta:

1. Interpoloitavan suureen arvot pystytään määrittämään myös uusissa



Kuva 4.23: *Yksikköneliön jakaminen kahteen kolmioelementtiin.*

pisteissä. Menetelmä toimii, mutta vaatii jonkin verran lisätyötä.

2. Interpoloitavan suureen arvoja ei ole mahdollista määrittää entisten datapisteiden lisäksi uusissa pisteissä. Tällöin on joko tyydyttävä paloittain lineaariseen interpolaatioon tai laskettava jotkin likiarvot puuttuville suureille — mahdollisesti jollain muulla interpolointimenetelmällä.

Esimerkki 4.13.2 Haemme funktiolle $f(x,y) = e^{xy} - x^2 + \sin(\pi y)$ paloittain kvadraattisen interpolantin yksikköneliössä $[0,1] \times [[0,1]$. Valitsemme lähtöpisteiksi yksikköneliön kärjet, joiden avulla muodostamme kaksi kolmiota K_1 ja K_2 puolittamalla neliön diagonaalia pitkin.

Tarkastelemme ensin alempaa kolmiota K_1, jonka kärkinä ovat pisteet $\mathbf{t}_1 = (0,0)$, $\mathbf{t}_2 = (1,0)$ ja $\mathbf{t}_3 = (1,1)$ (kuva 4.23). Tavoitteena on muodostaa kahden muuttujan polynomi, jonka termit ovat korkeintaan toista astetta, eli $d = 2$ ja $k = 2$. Tällöin mahdollisia $\mathbf{m}$-vektoreita on yhteensä 6 kappaletta:

$$\begin{array}{llll} \mathbf{m}_1 = (2,0,0), & \mathbf{m}_2 = (0,2,0), & \mathbf{m}_3 = (0,0,2), \\ \mathbf{m}_4 = (1,1,0), & \mathbf{m}_5 = (1,0,1), & \mathbf{m}_6 = (0,1,1). \end{array}$$

Näiden perusteella muodostetaan kuusi interpolointipistettä. Esimerkiksi

$$\mathbf{r}_1 = \frac{1}{2}(2 \cdot \mathbf{t}_1 + 0 \cdot \mathbf{t}_2 + 0 \cdot \mathbf{t}_3) = \mathbf{t}_1 = (0,0),$$

Vastaavasti muut pisteet ovat

$$\begin{array}{l} \mathbf{r}_2 = (2\mathbf{t}_2)/2 = (1,0) \\ \mathbf{r}_3 = (2\mathbf{t}_3)/2 = (1,1) \\ \mathbf{r}_4 = (\mathbf{t}_1 + \mathbf{t}_2)/2 = (1/2,0) \\ \mathbf{r}_5 = (\mathbf{t}_1 + \mathbf{t}_3)/2 = (1/2,1/2) \\ \mathbf{r}_6 = (\mathbf{t}_2 + \mathbf{t}_3)/2 = (1,1/2). \end{array}$$

Interpoloitavan funktion arvot on tunnettava siis näissä kuudessa pisteessä. Kolmion kärkipisteet ovat mukana joukossa, kuten pitikin, mutta lisäksi interpoloinnissa tarvitaan kolmion sivujen keskipisteitä. Koska olemme hakemassa

interpolanttia tunnetulle funktiolle $f(x,y)$, puuttuvien arvojen laskeminen sivujen keskipisteissä ei ole ongelma.

Seuraavaksi määritämme apupolynomit, jotka siis saavat aina arvon 1 yhdessä kuudesta interpolointipisteestä ja häviävät muissa. Apupolynomien muodostaminen on helppoa, kun käytämme hyväksi pisteiden kautta kulkevien suorien yhtälöitä. Laskemme malliksi pisteeseen $\mathbf{r}_1 = (0,0)$ liittyvän polynomin $l_1(x,y)$.

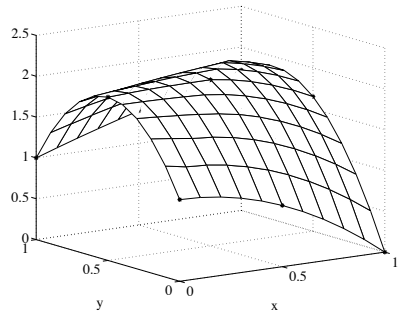
Pisteet $\mathbf{r}_2$, $\mathbf{r}_3$ ja $\mathbf{r}_6$ sijaitsevat kaikki suoralla $x = 1$, joten ottamalla polynomiin l_1 tekijäksi $(x-1)$ takaamme, että l_1 saa arvon 0 kaikissa kolmessa pisteessä. Pisteet $\mathbf{r}_4$ ja $\mathbf{r}_5$ puolestaan elimoinoituvat, kun otamme mukaan tekijän $(x - 1/2)$, joten $l_1(x,y) = \alpha(x-1)(x-1/2)$. Kerroin α on valittava siten, että $l_1(0,0) = 1$, joten $\alpha = 2$ ja $l_1(x,y) = 2(x-1/2)(x-1)$. Muut apupolynomit ovat

$$\begin{aligned} l_2(x,y) &= 2(y-x)(y-x+1/2) \\ l_3(x,y) &= 2y(y-1/2) \\ l_4(x,y) &= 4(y-x)(x-1) \\ l_5(x,y) &= 4y(1-x) \\ l_6(x,y) &= 4y(x-y). \end{aligned}$$

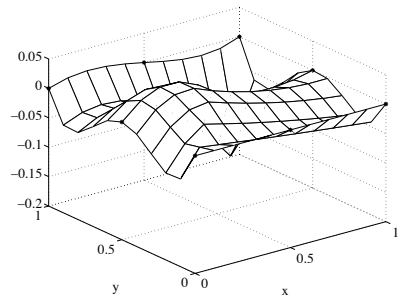
Lopulta voimme kirjoittaa funktiota $f(x,y)$ interpoloivan Lagrangen polynomin $P_1(x,y)$ alakolmiolle K_1:

$$P_1(x,y) = \sum_{j=1}^{6} f(\mathbf{r}_j) l_j(x,y).$$

Koko yksikköneliölle saamme interpolantin $\tilde{f}$ suorittamalla samat laskutoimitukset myös ylemmässä kolmiossa K_2. Diagonaalilla interpolaatiot yhtyvät. Kuvassa 4.24 on esitetty sekä interpolaation että virheen kuvaajat. Maksimivirhe on pienempi kuin esimerkissä 4.13.1, jossa tyydyimme 6 pisteen käyttöön ja lineaariseen interpolaatioon x-akselin suunnassa.



a: *Funktion* $f(x,y) = e^{xy} - x^2 + \sin(\pi y)$ *interpolaatio* $\tilde{f}(x,y)$.



b: *Interpoloinnin virhe* $f - \tilde{f}$.

Kuva 4.24: *Interpolointi säännöllisessä tasohilassa.*

Edellä kuvattu tapa valita Lagrangen interpolaatiopisteet takaa interpolantin $\tilde{f}(x,y)$ jatkuvuuden elementistä toiseen. Jos myös normaaliderivaatan

$\tilde{f}_n(x, y)$ tulee olla jatkuva (siis Hermiten interpolaation yleistys usean muuttujan funktioille), interpolaation muodostaminen ei ole lainkaan yhtä suoraviivaista. Ei nimittäin riitä, että vapausasteiksi valitaan funktion f ja sen ensimmäisen kertaluvun osittaisderivaattojen f_x, f_y arvot kolmioiden kärkipisteissä. Lisäksi on kiinnitettävä esimerkiksi kaikki toisen kertaluvun derivaatat f_{xx}, f_{yy} ja f_{xy} kärjissä sekä vielä normaaliderivaatta f_n sivujen keskipisteissä. Kaikkiaan vapaita parametrejä on tällöin 21 kappaletta, mikä riittää juuri kahden muuttujan 5. asteen polynomien esittämiseen. Käyttämällä erilaisia *makroelementtitekniikoita*, joissa useampi elementti yhdistetään, on mahdollista saavuttaa normaaliderivaatan jatkuvuus 3. asteen polynomeilla. Makroelementtien käyttö on kuitenkin hankalaa täysin satunnaisessa elementtiverkossa.

Jos riipumattomia muuttujia on enemmän kuin kaksi tai interpolantilta vaaditaan vielä suurempaa sileyttä, tarvittavien vapausasteiden lukumäärä kasvaa hyvin nopeasti.

4.13.3 Shepardin menetelmä

Edellä on esitetty Lagrangen interpolaation yleistyksiä. Säännöllisessä nelikulmiohilassa on mahdollista muodostaa koko interpolointialueen kattava polynomi. Elementtiajattelulla taas saadaan aikaiseksi paloittainen interpolaatio, jossa jokaisen elementin alueella interpolantille määritellään oma lauseke. Seuraavissa menetelmissä interpolaatiofunktiolle voidaan kirjoittaa yksi lauseke, mutta ne ovat silti luonteeltaan lokaaleja siinä mielessä, että kaukana olevien interpolaatiopisteiden vaikutusta heikennetään tai ne voidaan jättää kokonaan pois.

Shepardin menetelmä perustuu ajatukseen, että interpolantin $\tilde{f}(\mathbf{x})$ arvo pisteessä $\mathbf{x}$ riippuu sitä enemmän funktion $f(\mathbf{x})$ arvosta interpolointipisteessä $\mathbf{x}_i$ mitä pienempi etäisyys $|\mathbf{x}-\mathbf{x}_i|$ on. Menetelmän perusversiossa etäisyyden vaikutus otetaan huomioon painofunktioiden $w_i(\mathbf{x})$ avulla, jolloin interpolantti on

$$\tilde{f}(\mathbf{x}) = \frac{\sum_{i=1}^{n} w_i(\mathbf{x}) f(\mathbf{x}_i)}{\sum_{i=1}^{n} w_i(\mathbf{x})}.$$

Painofunktiot w_i ovat ei-negatiivisia, ja ne on valittava siten, että millään muuttujan $\mathbf{x}$ arvolla kaikki painofunktiot eivät saa arvoa 0. Interpolointipistettä $\mathbf{x}_i$ lähestyttäessä sitä vastaava painofunktio w_i kasvaa rajatta, jotta interpolointivaatimus toteutuisi.

Koska painofunktion w_i arvon halutaan vähenevän, kun etäisyys interpolointipisteeseen $\mathbf{x}_i$ kasvaa, eräs mahdollinen valinta painofunktioksi on

$$w_i(\mathbf{x}) = |\mathbf{x} - \mathbf{x}_i|^{-\mu}, \quad \mu > 0. \tag{4.52}$$

Edellinen painofunktio (4.52) ottaa huomioon kaikki interpolointipisteet $\mathbf{x}_i$, joten työmäärä kasvaa helposti suureksi. Menetelmä muutetaan lokaaliksi hylkäämällä ne pisteet $\mathbf{x}_i$, joiden etäisyys interpoloitavasta pisteestä $\mathbf{x}$ ylittää jonkin sopivasti asetetun katkaisuarvon R.

Suosittu ja yleensä hyvin toimiva lokaali versio Shepardin menetelmästä saadaan aikaiseksi *Franken ja Littlen painofunktiolla*

$$w_i(\mathbf{x}) = \begin{cases} 1 - \frac{|\mathbf{x}-\mathbf{x}_i|}{R}, & 0 < |\mathbf{x}-\mathbf{x}_i| < R, \\ 0, & |\mathbf{x}-\mathbf{x}_i| \geq R. \end{cases}$$

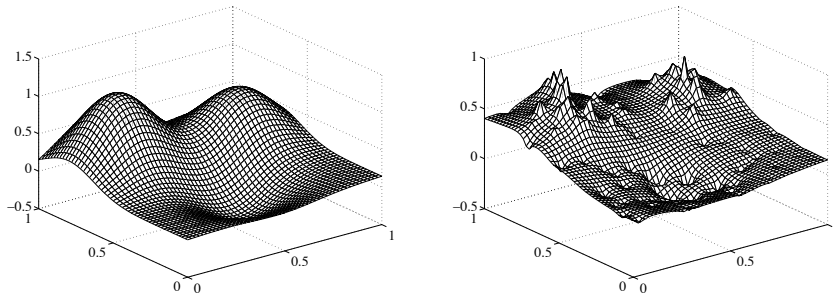
Esimerkki 4.13.3 Sovellamme sekä globaalia että lokaalia Shepardin menetelmää neliössä $[0,1] \times [0,1]$ funktion

$$f(x,y) = 0.5\sin(0.3x) + 0.7e^{-20r_1^2} + e^{-18r_2^2} - 0.4e^{-20r_3^2}$$

interpolointiin. Edellä r_1, r_2 ja r_3 tarkoittavat pisteen (x,y) etäisyyttä pisteistä $(0.8, 0.7)$, $(0.25, 0.8)$ ja $(0.5, 0.3)$. Funktion $f(x,y)$ kuvaaja on esitetty kuvassa 4.25a. Tarkastelualueessa sillä on kaksi lokaalia maksimia ja yksi minimi.

Globaalia Shepardin menetelmää varten arvomme yksikköneliöstä sata satunnaista pistettä, joiden perusteella muodostamme interpolantin $\tilde{f}(x,y)$ (kuva 4.25b). Koska globaalissa menetelmässä lasketaan painotettu keskiarvo funktion arvoista kaikissa interpolointipisteissä, alkuperäisen funktion huippujen ympäristöön osuvat interpolointipisteet nousevat piikkeinä esiin taustasta. Vastaavasti lokaalin minimin $(x,y) = (0.5, 0.3)$ lähellä interpoloinitipisteet näkyvät kuoppina. Parametrin μ arvo on 1.5.

Lokaali Shepardin interpolaatio (Franken ja Littlen painofunktio) tuottaa jo selvästi parempia tuloksia. Kuvassa 4.26 on kahden eri parametriyhdistelmän kokeilut. Lokaalin menetelmän ongelmaksi satunnaisen pisteistön yhteydessä voi muodostua katkaisusäde R: jos jokin alue on kaukana kaikista interpolointipisteistä, sitä approksimoidaan nollafunktiolla.



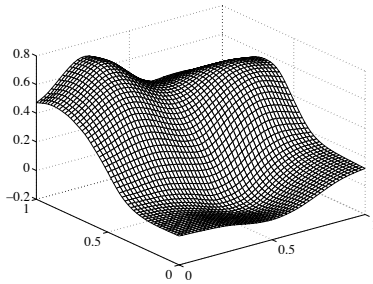
a: *Funktion* $f(x,y) = 0.5\sin(0.3x) + 0.7e^{-20r_1^2} + e^{-18r_2^2} - 0.4e^{-20r_3^2}$ *kuvaaja.*

b: *Funktion* $f(x,y)$ *globaali Shepardin interpolaatio.*

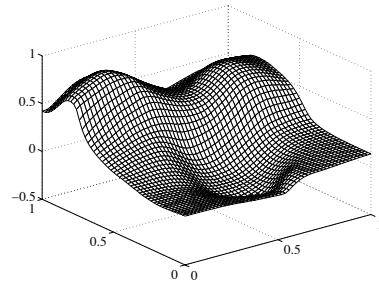
Kuva 4.25: *Testifunktio ja sen interpolaatio.*

Jos interpoloitavalle funktiolle $f(\mathbf{x})$ tunnetaan jotkin sarjakehitelmät $a_i(\mathbf{x})$ interpolointipisteiden $\{\mathbf{x}_i\}$ ympäristössä, Shepardin menetelmän voi kirjoittaa myös muodossa

$$\tilde{f}(\mathbf{x}) = \frac{\sum_{i=1}^n w_i(\mathbf{x}) a_i(\mathbf{x})}{\sum_{i=1}^n w_i(\mathbf{x})}.$$



a: *Lokaali Shepardin interpolaatio,* $\mu = 2$, $R = 0.4$.



b: *Lokaali Shepardin interpolaatio,* $\mu = 1.5$, $R = 0.25$.

Kuva 4.26: *Testifunktion lokaaleja Shepardin interpolaatioita.*

Näin on mahdollista saada aikaan sileämpi interpolantti, jonka osittaisderivaatat täsmäävät alkuperäisen funktion osittaisderivaattojen kanssa.

4.13.4 Radiaalisia kantafunktioita käyttävät menetelmät

Eräs lokaalien interpolointimenetelmien luokka perustuu radiaalisten kantafunktioiden käyttöön. Jos käytössä on interpolointipisteet $\{\mathbf{x}_i\} \in \mathbb{R}^d$, $i = 1, ..., n$, niin merkitään $r_i(\mathbf{x}) = |\mathbf{x} - \mathbf{x}_i|$. Lisäksi olkoon $\mathcal{P}_k^d$ kaikkien korkeintaan astetta k olevien d eri muuttujan polynomien muodostama avaruus. Interpolantti $\tilde{f}(\mathbf{x})$ on kirjoitettavissa muotoon

$$\tilde{f}(\mathbf{x}) = \sum_{j=1}^{n} \lambda_j \phi(r_j(\mathbf{x})) + P(\mathbf{x}),$$

missä $P(\mathbf{x}) \in \mathcal{P}_k^d$, ja polynomin $P(\mathbf{x})$ asteluku k on soveltajan valittavissa. Polynomitermi ei aina ole välttämätön, mutta joissain tapauksissa se on tarpeen, jotta tehtävällä olisi yksikäsitteinen ratkaisu.

Tuntemattomia parametreja on siis yhteensä $n + \binom{k+d}{d}$ kappaletta: painot λ_i ja polynomin $P(\mathbf{x})$ kertoimet. Interpoloinnin perusvaatimuksesta (4.50) seuraa suoraan n:n lineaarisen yhtälön ryhmä

$$\sum_{j=1}^{n} \lambda_j \phi(r_j(\mathbf{x_i})) + P(\mathbf{x_i}) = f_i, \quad i = 1, 2, ..., n.$$

Loput yhtälöt saadaan yleensä asettamalla ehdot

$$\sum_{j=1}^{n} \lambda_j p(\mathbf{x}_j) = 0 \text{ kaikilla } p(\mathbf{x}_j) \in \mathcal{P}_k^d. \tag{4.53}$$

Vaikka $p(\mathbf{x})$ on nyt mikä tahansa korkeintaan astetta k oleva d muuttujan polynomi, niin käytännössä riittää, kun lausekkeen $p(\mathbf{x})$ paikalle sijoitetaan vuorotellen polynomiavaruuden $\mathcal{P}_k^d$ kantafunktiot.

Ehdoista (4.53) seuraa, että tuntemattomille parametreille saadaan lineaarinen yhtälöryhmä, jonka kerroinmatriisi on symmetrinen. Matriisi voi kuitenkin olla singulaarinen, ellei vaadita lisäksi interpolointipisteiltä ominaisuutta

$$\{p \in \mathcal{P}_k^d \mid p(\mathbf{x}_i) = 0 \text{ kaikilla } i\} \quad \Longrightarrow \quad p(\mathbf{x}) \equiv 0.$$

Siten ainoa polynomi joka häviää kaikissa interpolointipisteissä on nollafunktio. Tämä ehto on välttämätön ratkaisun yksikäsitteisyyden kannalta.

Radiaalisen interpolaation huono puoli on kerroinmatriisin tiheys. Kun interpolaatiopisteiden lukumäärää n lisätään, yhtälöryhmän ratkaisuun kuluva aika kasvaa hyvin nopeasti. Kerroinmatriisi on myös helposti häiriöaltis, joten ratkaisualgoritmien pitää olla numeerisesti stabiileja.

Seuraavaksi on etsittävä sopiva radiaalifunktio $\phi(r)$. Väärin valittu $\phi(r)$ voi muuttaa tehtävän ratkeamattomaksi. Emme kuitenkaan esitä tässä yleisiä ehtoja, vaan tyydymme mainitsemaan muutaman hyväksi havaitun vaihtoehdon.

1. $\phi(r) = r$
2. $\phi(r) = \sqrt{r^2 + c^2}$
3. $\phi(r) = 1/\sqrt{r^2 + c^2}$
4. $\phi(r) = e^{-r^2}$
5. $\phi(r) = r^2 \log r.$

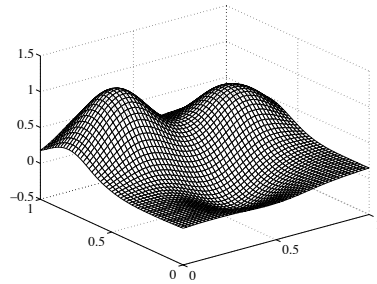
Näistä vaihtoehdot 3 ja 4 johtavat positiivisesti definiittiin yhtälöryhmään.

Yleisesti muotoa $\phi(r) = r^\gamma$ oleva funktio on kelvollinen, jos γ ei ole parillinen kokonaisluku, ja jos $\gamma < 2(k+1)$. Niinpä esimerkiksi tapauksessa $\gamma = 3$ interpolaatioon liitettävän polynomin $P(\mathbf{x})$ on oltava vähintään lineaarinen.

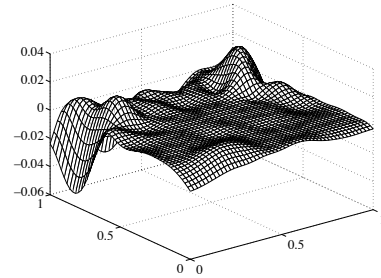
Esimerkki 4.13.4 Laskemme radiaalisen interpolaation esimerkissä 4.13.3 käytetylle funktiolle $f(x, y)$. Funktioksi $\phi(r(\mathbf{x}))$ valitsemme yllä luetelluista vaihtoehdoista toisen ($\phi(r) = \sqrt{r^2 + c^2}$), ja vakion c arvoksi kiinnitämme 0.1 Lisäksi otamme mukaan lineaarisen polynomitermin, joten interpolaatio on lopulta muotoa

$$\tilde{f}(x, y) = \sum_{j=1}^{n} \lambda_j \sqrt{(x - x_j)^2 + (y - y_j)^2 + 0.1^2} + \alpha_1 x + \alpha_2 y + \alpha_3.$$

Sadan satunnaisesti valitun interpolaatiopisteen avulle päästään erinomaisiin tuloksiin (kuva 4.27a). Virhe on suurimmillaan alueen reunalla, jossa ilmeisesti ei ole interpolointipisteitä riittävän tiheässä.



a: *Radiaalinen interpolantti* $\tilde{f}(x,y)$.



b: *Interpolaation virhe* $f(x,y) - \tilde{f}(x,y)$.

Kuva 4.27: *Radiaalinen interpolantti ja virhe.*

4.14 Ohjelmistot ja aliohjelmakirjastot

Approksimointi ja interpolointi ovat numeerisen laskennan perustehtäviä, joita varten matemaattisissa aliohjelmakirjastoissa (NAG, IMSL, Harwell Subroutine Library) ja yleisohjelmistoissa (Matlab, Mathematica) on useita eri tilanteisiin sopivia työkaluja.

4.14.1 Interpolointi

Yhden muuttujan funktioiden interpolointiin kaikki edellä mainitut ohjelmistot IMSL-kirjastoa lukuunottamatta tarjoavat koko tarkasteluvälin kattavan interpolaatiopolynomin määrittämisen lisäksi mahdollisuuden paloittaiseen interpolointiin. Oletusarvoisesti tällöin haetaan 3. asteen palapolynomeja, mutta ei välttämättä splinejä.

Mikäli halutaan spesifioida myös interpoloitavan funktion derivaatan arvot, valinnanvaraa on jo vähemmän. Ainoastaan NAG ja Mathematica osaavat laskea Hermiten interpolaation. Toisaalta kumpikaan näistä ei rajoita derivaattoja ensimmäiseen kertalukuun, vaan jokaisessa solmupisteessä voi kiinnittää vaihtelevan määrän derivaattoja.

IMSL puolestaan tukee pelkästään splini-interpolaatioita, joiden käsittelyä varten kirjastossa onkin huomattava määrä aliohjelmia. Splinit löytyvät toki myös muista ohjelmistoista. Mathematicassa splinejä ei ole valmiina, mutta käyttäjien itse koodaamia splinitoteutuksia on saatavilla verkosta ilmaiseksi.

NAG sisältää aliohjelman rationaalisen interpolaation laskemista varten. Kyseessä on todellakin interpolantti, ei esimerkiksi Padén approksimaatio (joka toki myös löytyy NAGista).

Usean muuttujan funktioiden interpolointi onnistuu kaikilla muilla pake-

teilla paitsi Harwellin kirjastolla. Yleensä vaaditaan, että data tunnetaan säännöllisessä nelikulmiohilassa, mutta IMSL, NAG ja Matlab osaavat käsitellä myös satunnaisessa hilassa tunnettua dataa. Aliohjelmakirjastot NAG ja IMSL selviävät korkeintaan kolmen muuttujan malleista, kun taas Matlab ja Mathematica eivät aseta periaatteessa rajoituksia riippumattomien muuttujien määrälle säännöllisessä hilassa.

4.14.2 Approksimointi

Tärkein menetelmä eli pienimmän neliön sovitus on mukana kaikissa tarkastelluissa ohjelmistoissa. Tosin sen toteutusten löytäminen voi vaatia hiukan kekseliäisyyttä: NAGissa pitää osata tutustua lukuihin ”E02: Curve and Surface Fitting” ja ”F04: Simultaneous Linear Equations” ja Harwellin manuaalissa lukuihin ”MA: Solution of Linear Equations” ja ”V: Optimization and nonlinear data fitting”. Matlabissa oikea komento on `polyfit` ja Mathematicassa yksinkertaisesti `Fit`.

NAGissa on valmiit aliohjelmat myös L_1-, L_∞- ja Padén approksimaatioiden laskemiseen. Lisäksi NAGin avulla on mahdollista etsiä spliniapproksimaatioita korkeintaan kahden muuttujan funktioille.

Myös IMSL ja Harwellin kirjastot selviävät rationaalisista approksimaatioista. IMSL on varustettu lisäksi kahden ja kolmen muuttujan funktioiden approksimointiin sopiville aliohjelmilla. Mathematican `Fit`-komento puolestaan ei periaatteessa rajoita riippumattomien muuttujien tai kantafunktioiden määrää lainkaan.

4.15 Lisätietoja

Approksimointi ja interpolointi ovat keskeisiä numeerisia tehtäviä, joihin liittyvät perusmenetelmät löytyvät kaikista numeerisen analyysin oppikirjoista. Pienimmän neliön tehtävien ratkaisu on käsitelty kattavasti teoksessa *Numerical Methods for Least Squares Problems* [Bjö96]. Van Huffelin ja Vandewallen laajassa artikkelissa [vHV91] on hyvä esitys täydellisen pienimmän neliön sovituksesta kaikkine erikoistapauksineen.

Bézier’n käyrien ja splinien teoriaa käydään läpi mm. kirjoissa *Curves and Surfaces for Computer Aided Geometric Design* [Far90] ja *An Introduction to Computational Geometry for Curves and Surfaces* [DS96]. Usean muuttujan funktioiden interpolointimenetelmiä epäsäännöllisessä pisteistössä on esitelty artikkelissa *Methods for Multivariable Interpolation at Scattered Data Points* [Pow97].

5 Numeerinen integrointi

Integraalien yhteydessä joudutaan usein turvautumaan numeerisiin menetelmiin. Näin on aina, kun integroitavasta funktiosta tunnetaan kokeiden tai laskujen perusteella vain pisteittäisiä arvoja tai kun integraalifunktio ei ole lausuttavissa suljetussa muodossa. Numeerisia menetelmiä käytetään myös sellaisissa tapauksissa, joissa analyyttinen ratkaisu olisi ainakin periaatteessa olemassa, mutta sen laskeminen on hyvin työlästä. Itse asiassa numeerinen lähestymistapa ei ole ainoastaan nopeampi vaan mahdollisesti jopa tarkempi, jos analyyttinen ratkaisu koostuu useista sadoista termeistä, joita summattaessa aiheutetaan merkitsevien numeroiden häviämistä.

5.1 Esimerkkejä

Esimerkki 5.1.1 Integraalin laskeminen kokeellisesti mitatusta datasta.

Diffraktiokokeessa mitataan eri sirontakulmia vastaavia intensiteettejä, jolloin havaitaan tietyissä suunnissa voimakkaita piikkejä. Kiinnostava suure on piikkien pinta-alan suhde koko intensiteettikäyrän alle jäävän alueen pinta-alaa. Pinta-alojen selvittäminen vastaa numeerista integrointia.

Esimerkki 5.1.2 Ratkaisemme tavallisen differentiaaliyhtälön

$$u'' = -\sin \pi x, \quad u(0) = u(1) = 0$$

heikossa mielessä. Valitsemme joukon reunaehdot toteuttavia *yritefunktioita*

$$\phi_i(x), \quad i = 1, \ldots, n, \quad \phi_i(0) = \phi_i(1) = 0,$$

joiden lineaarikombinaation avulla approksimoimme tarkkaa ratkaisua: $u(x) \approx u_n = \sum_{i=1}^{n} c_i \phi_i(x)$. Tuntemattomien kertoimien c_i määräämiseksi sijoitamme ensin alkuperäisen yhtälön vasemmalle puolelle funktion u paikalle approksimaation u_n. Seuraavaksi kerromme yhtälön puolittain kantafunktiolla ϕ_j ja integroimme molemmat puolet vielä tarkasteluvälin $[0, 1]$ yli, jolloin päädymme yhtälöön

$$\int_0^1 \phi_j(x) \sum_{i=1}^{n} c_i \phi_i''(x)\, dx = -\int_0^1 \phi_j(x) \sin \pi x\, dx.$$

Kun suoritamme yhtälön vasemmalla puolella osittaisintegroinnin, sijoitustermi katoaa yritefunktioiden reunaehtojen ansiosta:

$$\int_0^1 \phi_j'(x) \sum_{i=1}^n c_i \phi_i'(x)\, dx = \int_0^1 \phi_j(x) \sin \pi x\, dx.$$

Vaihtamalla vasemmalla puolella summauksen ja integroinnin järjestystä ja antamalla indeksille j kaikki arvot välillä $[1, 2, ..., n]$ saamme lopulta lineaarisen yhtälöryhmän kertoimille c_i:

$$\sum_{i=1}^n c_i \int_0^1 \phi_j'(x)\phi_i'(x)\, dx = \int_0^1 \phi_j(x) \sin \pi x\, dx \quad j = 1, 2, ..., n.$$

Voimme kirjoittaa nämä yhtälöt edelleen tutumpaan matriisimuotoon $A\mathbf{c} = \mathbf{b}$ merkitsemällä

$$a_{ij} = \int_0^1 \phi_i'(x)\phi_j'(x)\, dx$$

ja

$$b_j = \int_0^1 \phi_j(x) \sin \pi x\, dx.$$

Kertoimien $\mathbf{c}$ ratkaiseminen vaatii siis periaatteessa pahimmassa tapauksessa $\frac{1}{2}(n^2 + 3n)$ integraalin arvon laskemisen. Jos menetelmä halutaan ohjelmoida tietokoneelle, on varauduttava erilaisiin yhtälöihin ja epähomogeenisiin termeihin yhtälön oikealla puolella. Tällöin on helpointa suorittaa integroinnit numeerisesti.

Voimme tietysti pyrkiä valitsemaan yritefunktiot siten, että integraalit ovat laskettavissa analyyttisestikin, mutta tämä keino toimii yleensä vain yksiulotteisessa tapauksessa (tavallinen differentiaaliyhtälö). Edellä kuvatun menetelmän kehittäminen edelleen osittaisdifferentiaaliyhtälöitä silmälläpitäen ottamalla mukaan useampia muuttujia johtaa helposti tilanteisiin, joissa mikään yritefunktioiden valinta ei tuota suljetussa muodossa ratkeavia integraaleja. Tällöin numeerinen integrointi on todellakin ainoa tapa, jolla on mahdollista päästä eteenpäin.

Esimerkki 5.1.3 Jatkuvan satunnaismuuttujan X mahdollisten arvojen jakaumaa kuvataan *tiheysfunktiolla* $f(x)$. Tällöin todennäköisyys sille, että X saa arvon väliltä $[x_1, x_2]$, voidaan ilmaista integraalina:

$$P(x_1 \le X \le x_2) = \int_{x_1}^{x_2} f(x)\, dx.$$

Vastaavasti satunnaismuuttujan X *kertymäfunktio* $F(x)$ kertoo todennäköisyyden, että muuttujan X arvo on korkeintaan x. Kertymäfunktion arvot saadaan edellisen kaavan yleistyksenä laskettua tiheysfunktion integraalina.

$$F(x) = P(-\infty \le X \le x) = \int_{-\infty}^{x} f(s)\, ds.$$

Myös monet jatkuvia satunnaisjakaumia kuvaavat tunnusluvut määritellään integraalien avulla, kuten *odotusarvo* $E(X)$:

$$E(X) = \int_{-\infty}^{\infty} x f(x)\, dx.$$

Samaan tapaan määritellään *varianssi* $\sigma^2(X)$:

$$\sigma^2(X) = \int_{-\infty}^{\infty} (x - E(X))^2 f(x)\, dx.$$

Usein tiheysfunktio $f(x)$ on muodoltaan sellainen, ettei integraalien laskeminen ole mahdollista analyyttisesti. Tällöin joudutaan turvautumaan numeeriseen integrointiin. Äärettömien integrointivälien vuoksi on tarpeen käyttää joko erikoismenetelmiä tai skaalata väli sopivalla muuttujanvaihdoksella äärelliseksi.

Yleisimmille jakaumille kertymäfunktion arvoja on laskettu valmiiksi taulukoihin, joten niitä ei kannata ryhtyä itse laskemaan, ellei tarvitse syystä tai toisesta suurta tarkkuutta.

5.2 Perusasioita numeerisesta integroinnista

Yhden muuttujan funktion $f(x)$ määrätyn integraalin $\int_a^b f(x)\, dx$ numeerisen approksimaation eli *kvadratuurin* laskeminen on perusteellisesti tutkittu ongelma, jota varten on useamman sadan vuoden kuluessa kehitetty lukuisia ratkaisumenetelmiä. Itse asiassa vanhimmat kvadratuurit ovat peräisin ajalta, jolloin integraalin käsitettä ei vielä ollut olemassa.

5.2.1 Kvadratuurin muodostaminen

Monet numeerisista kvadratuureista on johdettu interpolaation pohjalta. Ideana on etsiä integroitavalle funktiolle $f(x)$ jokin interpolaatio $\tilde{f}(x)$ välillä $[a, b]$, ja sitten arvioida integraalia $\int_a^b f(x)$ (toivottavasti) helposti laskettavalla integraalilla $\int_a^b \tilde{f}(x)$. Lopputuloksena on yleensä painotettu keskiarvo funktion arvoista joissain tarkasteluvälin pisteissä (*kvadratuuripisteissä* eli *solmupisteissä*) x_i, $i = 1, \ldots, n$:

$$\int_a^b f(x)\, dx \approx \sum_{i=1}^{n} w_i f(x_i), \quad x_i \in [a, b].$$

Jokaiseen solmupisteeseen liittyvää *painokerrointa* on yllä merkitty symbolilla w_i. Joissain esityksissä solmupisteiden numerointi aloitetaan indeksin arvosta 0, mutta tässä käyttämämme tapa lienee luontevampi, koska n ilmoittaa suoraan solmupisteiden lukumäärän. Kvadratuurit voivat sisältää

myös integrandin derivaattoja (esimerkiksi Eulerin ja Maclaurinin summakaava tai Hermiten interpolaatioon pohjautuvat menetelmät).

Kvadratuuria voidaan lähteä muodostamaan myös suoraan painotetun keskiarvon lausekkeesta. Tällöin tavoitteena on määrätä solmupisteet ja painot jossain mielessä optimaalisesti ilman taustalla olevaa interpolaatioajatusta. Luonnollisesti esimerkiksi jokin solmupisteiden osajoukko voidaan kiinnittää etukäteen, jolloin vapaiden muuttujien määrä vähenee. Kaikki käytännössä merkittävät kvadratuurit ovat kuitenkin tulkittavissa interpolaatiokvadratuureiksi, vaikkei menetelmiä kenties alunperin olekaan tältä pohjalta johdettu.

Esimerkki 5.2.1 (Keskipistesääntö) Yksinkertaisin kvadratuuri on *keskipistesääntö*. Siinä funktion $f(x)$ integraalia välillä $[a, b]$ arvioidaan kertomalla funktion arvo välin keskipisteessä $x_1 = (a + b)/2$ välin pituudella $w_1 = b - a$:

$$\int_a^b f(x)\,dx \approx (b-a) f\left(\frac{a+b}{2}\right) = w_1 f(x_1).$$

Jos approksimaatiota halutaan parantaa, väli $[a, b]$ voidaan jakaa pienempiin osiin $[c_0, c_1]$, $[c_1, c_2]$, ..., $[c_{m-1}, c_m]$, joista jokaisella käytetään samaa keinoa integraalin arvioimiseen:

$$\int_a^b f(x), dx = \sum_{i=0}^{m-1} \int_{c_i}^{c_{i+1}} f(x)\,dx \approx \sum_{i=0}^{m-1} f\left(\frac{c_i + c_{i+1}}{2}\right)(c_{i+1} - c_i).$$

Jos integointivälin päätepisteet ovat mukana solmupisteiden joukossa, kyseessä on *suljettu* menetelmä. *Avoimessa* menetelmässä ei käytetä välin päätepisteitä. Edellä esitelty keskipistesääntö on siis esimerkki avoimesta kvadratuurista.

Esimerkki 5.2.2 (Parannettu puolisuunnikassääntö) Tavallinen *puolisuunnikas-* eli *trapetsisääntö* on yksinkertaisin suljettu kvadratuuri, joka siis hyödyntää funktion arvoja integrointivälin päätepisteissä:

$$\int_a^b f(x)\,dx \approx \frac{(b-a)}{2}\left(f(a) + f(b)\right).$$

Parannetussa puolisuunnikassäännössä oletetaan, että myös integroitavan funktion derivaatan arvot päätepisteissä ovat käytettävissä:

$$\int_a^b f(x)\,dx \approx \frac{(b-a)}{2}\left(f(a) + f(b)\right) + \frac{(b-a)^2}{12}\left(f'(a) - f'(b)\right).$$

Kvadratuurit voidaan muodostamistapansa perusteella luokitella perheisiin. Isommalla työllä eli käyttämällä enemmän solmupisteitä (itse asiassa käyttämällä korkeamman kertaluvun interpolaatiota kvadratuurin taustalla) päästään yleensä suurempaan tarkkuuteen. Jotkin menetelmätyypit ovat numeeristen stabiilisuusongelmien vuoksi käytännössä äärellisiä: solmupisteiden määrän lisääminen tietyn rajan yli ei enää tuotakaan parempaa lopputulosta vaan johtaa mahdollisesti suuriin virheisiin.

Menetelmistä esitetään kirjallisuudessa yleensä ensin versio, jossa integraalin arvo yritetään laskea kerralla koko tarkasteluvälille $[a, b]$. Käytännössä joudutaan kuitenkin useimmiten turvautumaan *yhdistettyihin menetelmiin*, joissa väli $[a, b]$ jaetaan pienempiin osaväleihin

$$[c_0, c_1],\ [c_1, c_2],\ \ldots,\ [c_{m-1}, c_m], \quad c_0 = a,\ c_m = b.$$

Perusversiota sovelletaan sitten erikseen kullakin osavälillä.

Kvadratuuria voi siis tarkentaa kahdella tavalla: siirtymällä parempiin menetelmiin saman menetelmäluokan sisällä tai soveltamalla yksinkertaista menetelmää useammalla osavälillä. Yleensä ensinmainittu tapa on tehokkaampi.

Esimerkki 5.2.3 Integroimme funktion $f(x) = e^x$ välillä $[0, 1]$ (tarkka ratkaisu on $e - 1 \approx 1.71828$). Valitsemme perusmenetelmäksi yllä esitellyn puolisuunnikassäännön, jolloin

$$\int_0^1 e^x\,dx \approx \frac{1}{2}(e^0 + e^1) \approx 1.85914...$$

Tarkempaan tulokseen voimme nyt päästä joko jakamalla välin $[0, 1]$ osaväleihin tai valitsemalla samasta menetelmäperheestä korkeamman kertaluvun menetelmän. Kokeilemme ensin kolmea yhtäsuurta osaväliä:

$$\begin{aligned}\int_0^1 e^x\,dx &\approx \frac{1}{3}\cdot\frac{1}{2}(e^0 + e^{1/3}) + \frac{1}{3}\cdot\frac{1}{2}(e^{1/3} + e^{2/3}) + \frac{1}{3}\cdot\frac{1}{2}(e^{2/3} + e^1)\\ &= 1.73416\ldots\end{aligned}$$

Vastaava korkeamman kertaluvun menetelmä on nimeltään 3/8-sääntö, ja se antaa tulokseksi

$$\int_0^1 e^x\,dx \approx \frac{1}{8}(e^0 + 3e^{1/3} + 3e^{2/3} + e^1) \approx 1.71854.$$

Edellisen suhteellinen virhe on noin 1% ja jälkimmäisen vain noin 0.15%, vaikka jouduimme tekemään saman työmäärän kumpaankin approksimaatioon. Tässä tapauksessa jopa solmupisteet ovat samat, ja eroja on vain painokertoimissa.

Huomaa, että alkuperäisen integrointivälin jakaminen yhtä pitkiin osiin on vain harvoin optimaalista. Yleensä jakoa kannattaa tihentää siellä, missä integroitava funktio heilahtelee voimakkaimmin.

Eri menetelmät poikkeavat toisistaan siis periaatteessa vain tavassa, jolla kvadratuurin vapausasteet x_i ja w_i, $i = 1, ..., n$ valitaan. Käytännön laskujen kannalta erot ovat kuitenkin huomattavia.

Jos $n + 1$–solmuisen kvadratuurin on suoriuduttava virheettömästi kaikista korkeintaan M-asteisista polynomeista välillä $[a, b]$, solmupisteet x_i ja painokertoimet w_i voidaan määrätä yhtälöryhmästä, joka syntyy, kun funktion $f(x)$ paikalle sijoitetaan vuorotellen monomit $1, x, ..., x^M$:

$$\sum_{i=1}^{n+1} w_i x_i^m = \frac{1}{m+1}(b^{m+1} - a^{m+1}), \quad m = 0, ..., M. \tag{5.1}$$

Tämä yhtälöryhmä on lineaarinen painokertoimien suhteen, mutta epälineaarinen solmupisteiden suhteen. Vapaita parametreja on kaikkiaan $2(n + 1)$ kappaletta, joten ainakin periaatteessa on mahdollisuus selvitä jopa asteen $2n + 1$ polynomeista ($M \leq 2n + 1$).

Pääsääntöisesti menetelmissä, joissa tavoitteena on mahdollisimman suuri tarkkuus, kvadratuuripisteet ja -painot joudutaan joko ratkaisemaan epälineaarisista yhtälöistä tai ne luetaan valmiista taulukoista. Tämä aiheuttaa ylimääräistä työtä algoritmin ohjelmointivaiheessa, ellei käytössä ole hyvää aliohjelmakirjastoa, josta tarvittavat rutiinit löytyvät valmiina.

Jos taas menetelmä on sellainen, että kvadratuuriparametrien määrääminen on yksinkertaista (esimerkiksi kiinnitetään pisteet x_i ennakolta ja lasketaan vain vastaavat optimaaliset painokertoimet w_i), päädytään yleensä melko karkeisiin approksimaatioihin. Tämän vuoksi integrointiväli on jaettava hyvin moneen osaan, ennen kuin saavutetaan sama tarkkuus, johon monimutkaisempi menetelmä pystyy paljon pienemmällä työmäärällä.

Jos kvadratuurien laskeminen muodostaa oleellisen osan tietokoneohjelman tekemästä kokonaistyömäärästä, kannattaa käyttää päivä tai kaksikin hyvän algoritmin valintaan. Tehokas kvadratuuri voi tuottaa jopa satakertaisia nopeutuksia laskenta-aikoihin. Yksinkertaiset menetelmät puolestaan ovat omiaan kokeiluluontoisiin testiohjelmiin, joita ei ole tarkoitettu kattaviin tutkimusprojekteihin. Vaikka peruskvadratuuri ei olisikaan kovin tarkka, parempiin tuloksiin voi päästä käyttämällä hyväksi *ekstrapolointia*, kuten kappaleessa 5.8 (sivu 171) kerrotaan.

5.2.2 Integrointivälin skaalaus

Tämän luvun alussa esitelty perustehtävä oli asetettu mielivaltaiselle välille $[a, b]$, kun taas useimmat menetelmät on johdettu jollekin kiinteälle välille $[c_0, c_1]$, esimerkiksi $[-1, 1]$. Tämän vuoksi integraalit joudutaan tavallisesti skaalaamaan ennen kvadratuurin laskemista. Sopiva muuttujanvaihdos on $x = (b - a)(t - c_0)/(c_1 - c_0) + a$, jolloin

$$\int_a^b f(x)\,dx = \int_{c_0}^{c_1} f\left(\frac{b-a}{c_1-c_0}(t - c_0) + a\right)\frac{b-a}{c_1-c_0}\,dt.$$

Toinen vaihtoehto on muuntaa taulukoissa esitetyt solmupisteet t_i ja painot q_i vastaamaan skaalaamatonta väliä $[a, b]$. Kyse on täsmälleen samasta

asiasta kuin muuttujanvaihdoksessa, nyt vain toisin päin ajateltuna. Painot skaalautuvat välien pituuksien suhteessa ja solmupisteet puolestaan jakavat integrointivälit aina samassa suhteessa. Siis taulukoidut painot q_i kerrotaan luvulla $(b-a)/(c_1-c_0)$ ja välille $[a,b]$ sopivat solmupisteet x_i saadaan välille $[c_0,c_1]$ taulukoiduista pisteistä t_i soveltamalla yllä mainittua muuttujanvaihdosta

$$x_i = (b-a)(t_i-c_0)/(c_1-c_0)+a.$$

Skaalauksesta esitetään esimerkki Gaussin kvadratuurien yhteydessä (esimerkki 5.3.1).

5.2.3 Virhearviot ja menetelmien kertaluvut

Numeerisille integrointimenetelmille on johdettavissa teoreettiset virhearviot. Niissä esiintyy tekijänä integroitavan funktion $f(x)$ jonkin kertaluvun derivaatta, joten virhekaavojen suora soveltaminen voi olla hankalaa sellaisessa tapauksessa, jossa $f(x)$ ei ole analyyttisesti tunnettu (mittausdata). Oleellisempaa onkin yleensä tietää kuinka virhe käyttäytyy, kun integrointiväli jaetaan pienempiin osaväleihin. Virhekaavat eivät myöskään kiinnitä pistettä, jossa derivaatan arvo tulisi laskea, joten tavallisesti virhettä joudutaan arvioimaan käyttämällä derivaatan maksimiarvoa integrointivälillä. Virhearvioiden teoreettinen johtaminen perustuu yleensä *Peanon ytimien* käyttöön, joita ei tässä kirjassa käsitellä.

Kvadratuurien yhteydessä mainitaan usein niiden kertaluku. Koska kvadratuurin kertaluvun käsitettä käytetään eri lähteissä hiukan eri tavoin, lienee paikallaan selvittää tilannetta.

Aloitamme polynomien integrointitarkkuuteen pohjautuvasta määritelmästä: jos kvadratuuri integroi tarkasti kaikki korkeintaan astelukua N olevat polynomit, mutta ei enää kaikkia $N+1$ –asteisia polynomeja, *kvadratuurin kertaluku* on N. Kaikki $N+1$ pisteen kvadratuurit, joiden kertaluku on vähintään N, ovat johdettavissa jonkin interpolaation pohjalta.

Kertalukuun päästään käsiksi myös virhearvion kautta. Käytämme kvadratuuria $Q(f;a,b)$ integraalin

$$I(f;a,b) = \int_a^b f(x)\,dx$$

approksimaationa välillä $[a,b]$. Oletamme, että väli $[a,b]$ on jaettu useaan pienempään osaväliin $[c_i,c_{i+1}]$, joiden koko on korkeintaan h:

$$|c_{i+1}-c_i| = h_i \le h.$$

Kvadratuurin virhe $E(f;a,b)$ on tällöin $E(f;a,b) = I(f;a,b) - Q(f;a,b)$.

Jos funktio $f \in C^{N+1}[a,b]$, niin yhdellä osavälillä $[c_i,c_{i+1}]$ kvadratuurin käyttäminen integraalin tarkan arvon sijasta aiheuttaa *lokaalin virheen*, joka on kertaluokkaa $N+2$:

$$E(f;c_i,c_{i+1}) = I(f;c_i,c_{i+1}) - Q(f;c_i,c_{i+1}) = \mathcal{O}(h_i^{N+2}).$$

Koko välillä $[a, b]$ syntyvä *globaali virhe* on kertaluokkaa $\mathcal{O}(h^{N+1})$. Joissain kirjoissa myös tätä kutsutaan kvadratuurin kertaluvuksi, mikä johtaa ristiriitatilanteeseen ylempänä esitetyn määritelmän kanssa.

Jos esimerkiksi kirjastorutiinin käyttöohjeissa sanotaan, että menetelmän kertaluku on 2, se voi siis tarkoittaa kahta eri asiaa: 1) toisen asteen polynomit integroituvat täsmälleen oikein, 2) globaali virhe pienenee välin $[a, b]$ jakoa tihennettäessä kuten $\mathcal{O}(h^2)$, jolloin vain lineaariset polynomit integroituvat tarkasti.

Esimerkki 5.2.4 Edellä esitetyn keskipistesäännön lokaali virhe välillä $[c_i, c_{i+1}]$, jonka pituus on $h_i = c_{i+1} - c_i$, on

$$h_i^3/3 \max_{x\in[c_i,c_{i+1}]} |f''(x)|.$$

Globaali virhe koko välillä $[a, b]$ on

$$\frac{h^2}{3(b-a)} \max_{x\in[a,b]} |f''(x)|,$$

joten menetelmän kertaluku on 1.

Jos tarkkuutta halutaan parantaa käyttämällä korkeamman kertaluvun menetelmää, selvitään tietysti vähemmällä työllä, jos voidaan käyttää jo entuudestaan laskettuja karkeamman menetelmän solmupisteitä. Tällöin lisätyötä aiheutuu vain täysin uusien solmupisteiden laskemisesta ja funktion arvojen selvittämisestä. Tätä kvadratuuriperheen ominaisuutta nimitetään *progressiivisuudeksi*.

5.3 Gaussin kvadratuurit

Gaussin kvadratuurit ovat joukko numeerisia integrointimenetelmiä, jotka hyödyntävät solmupisteiden ja vastaavien painojen lukumäärän optimaalisesti: mahdollisimman korkea-asteiset polynomit integroituvat eksaktisti. Kuten edellä on jo kerrottu, solmupisteet joudutaan tässä tapauksessa ratkaisemaan epälineaarisesta yhtälöryhmästä.

Käytetyimmän alaluokan muodostavat *Gaussin ja Legendren* kvadratuurit. Parametrien arvot on taulukoissa annettu yleensä välille $[-1, 1]$ skaalatuille integraaleille, jolloin solmupisteet ovat Legendren polynomien P_n nollakohtia. Taulukossa 5.1 on lueteltu muutaman alhaisimman kertaluvun menetelmän parametrien arvot. Menetelmien kertaluku on $2n - 1$.

Edellä esitelty keskipistesääntö on yksinkertaisin tämän luokan menetelmistä.

Esimerkki 5.3.1 Laskemme integraalin $\int_0^4 e^{-x}\,dx = 1 - e^{-4} \approx 0.981684$ likiarvon käyttäen kahden pisteen Gaussin kvadratuuria (kohta $n = 2$ taulukossa 5.1). Koska taulukkoarvot soveltuvat sellaisinaan vain välin $[-1, 1]$ integraaleille, joudumme joko tekemään integraalissa sopivan muuttujanvaihdoksen tai skaalaamaan taulukossa annettuja lukuja.

Kokeillaan ensin muuttujanvaihdosta $x = 2(t + 1)$ eli $t = x/2 - 1$, jolloin

$$\int_0^4 e^{-x}\,dx = \int_{-1}^1 2e^{-2(t+1)}\,dt.$$

Jälkimmäinen integraali on nyt välillä $[-1, 1]$, joten taulukon luvut ovat käytettävissä sellaisenaan, ja

$$\int_{-1}^1 2e^{-2(t+1)}\,dt \approx 1 \cdot 2e^{-2(-0.577350+1)} + 1 \cdot 2e^{-2(0.577350+1)}$$
$$\approx 0.944160.$$

Integraalin arviointi Gaussin kvadratuurilla aiheuttaa tässä tapauksessa absoluuttisen virheen $0.981684 - 0.944160 = 0.037524$.

Pisteiden ja painojen skaalaus välille $[0, 4]$ sopiviksi käy puolestaan seuraavasti: välien pituuksien suhde on $4 : 2 = 2$, joten painot joudutaan kertomaan tekijällä 2. Solmupisteet ratkeavat kaavasta

$$x_i = \frac{4 - 0}{1 - (-1)}(t_i - (-1)) + 0$$

eli $x_i = 2(t_i + 1)$, $i = 1, 2$, joten $x_1 = 2(-0.577350 + 1) = 0.845299$ ja $x_2 = 2(0.577350 + 1) = 3.154701$. Lopullinen kvadratuuri on siis

$$2 \cdot e^{-0.845299} + 2 \cdot e^{-3.154701} \approx 0.944160.$$

Käytännön laskujen kannalta ei siis ole merkitystä kummin päin hoitaa taulukoitujen arvojen soveltamisen.

Gaussin ja Legendren kvadratuurien lisäksi on olemassa muitakin optimaalisen tarkkuuden integrointimenetelmiä. Nämä sopivat tapauksiin, joissa integrandissa esiintyy erilaisia *painofunktioita* $w(x)$ varsinaisen integroitavan funktion $f(x)$ lisäksi. Välillä $[-1, 1]$ voidaan määritellä *Gaussin ja Tšebyševin* kvadratuurit integraaleille, jotka ovat muotoa

$$\int_{-1}^1 \frac{1}{\sqrt{1 - x^2}} f(x)\,dx.$$

Kun painofunktio vaimenee nopeasti, integraalit suppenevat myös äärettömillä väleillä, jos $f(x)$ kasvaa riittävän hitaasti. Tällaisia tapauksia varten on omiaan *Gaussin ja Laguerren* menetelmä, joka soveltuu muotoa

$$\int_0^\infty e^{-x} f(x)\,dx$$

Taulukko 5.1: *Gaussin ja Legendren kvadratuurien solmupisteitä ja painoja välillä* $[-1,1]$.

n	x_i	w_i
1	$x_1 = 0$	$w_1 = 2$
2	$x_2 = -x_1 = 1/\sqrt{3}$	$w_1 = w_2 = 1$
3	$x_3 = -x_1 = \sqrt{3/5}$ $x_2 = 0$	$w_1 = w_3 = 5/9$ $w_2 = 8/9$
4	$x_4 = -x_1 = 0.861136311594053$ $x_3 = -x_2 = 0.339981043584856$	$w_1 = w_4 = 0.347854845137454$ $w_2 = w_3 = 0.652145154862546$
5	$x_5 = -x_1 = 0.906179845938664$ $x_4 = -x_2 = 0.538469310105683$ $x_3 = 0$	$w_1 = w_5 = 0.236926885056189$ $w_2 = w_4 = 0.478628670499366$ $w_3 = 0.568888888888889$
6	$x_6 = -x_1 = 0.932469514203152$ $x_5 = -x_2 = 0.661209386466265$ $x_4 = -x_3 = 0.238619186083197$	$w_6 = w_1 = 0.171324492379170$ $w_2 = w_5 = 0.360761573048139$ $w_3 = w_4 = 0.467913934572691$

oleville integraaleille. *Gaussin ja Hermiten* menetelmä puolestaan sopii kahteen suuntaan äärettömälle integrointivälille:

$$\int_{-\infty}^{\infty} e^{-x^2/2} f(x)\, dx.$$

Nämä integraalit suppenevat esimerkiksi aina kun $f(x)$ on polynomi.

Gaussin kvadratuurit ovat avoimia eli integrointivälin päätepisteet eivät kuulu solmupisteiden joukkoon. Jos kvadratuuri suunnitellaan siten, että toinen tai molemmat päätepisteet otetaan mukaan, ja painot ja loput solmupisteet valitaan niin, että lopputuloksena on mahdollisimman korkean kertaluvun kvadratuuri, päädytään *Gaussin ja Radaun* tai *Gaussin ja Lobatton* menetelmiin.

5.3.1 Virhearvio

Välillä $[-1,1]$ n:n pisteen ($n = 1, 2, ...$) Gaussin ja Legendren kvadratuurin virhe $E_n(f;-1,1)$ integraalin todelliseen arvoon nähden on

$$E_n(f;-1,1) = \frac{2^{2n+1}(n!)^4}{[(2n)!]^3(2n+1)} f^{(2n)}(\xi), \quad \xi \in [-1,1].$$

Virhekaavassa siis oletetaan, että integroitava funktio $f(x)$ on vähintään $2n$ kertaa derivoituva integrointivälillä.

Kun samaa kvadratuuria sovelletaan mielivaltaiselle äärelliselle välille $[a, b]$, virhelauseke on kerrottava vielä tekijällä $((b-a)/2)^{2n+1}$.

Esimerkki 5.3.2 Laskemme virhearvion edellisessä esimerkissä käytetylle 2 pisteen kvadratuurille. Funktion e^{-x} neljäs derivaatta on e^{-x}, joka puolestaan saavuttaa välillä $[0, 4]$ suurimman arvonsa $e^0 = 1$ pisteessä $\xi = 0$. Virhekaavassa esiintyvä murtolauseke saa arvon $2^5 \cdot 2^4/(4!)^3 \cdot 5 = 2^9/24^3 \cdot 5$. Integrointivälin skaalauksesta tulee lisäksi vielä kerroin $((4-0)/2)^5 = 2^5$. Siten virhe on korkeintaan

$$\frac{2^{14}}{24^3 \cdot 5} \cdot 1 \approx 0.2370370...$$

Todellinen virhe on paljon pienempi, koska tässä derivaattatermiä on approksimoitu mahdollisimman pessimistisesti.

5.3.2 Huomautuksia Gaussin kvadratuurista

Gaussin kvadratuurien painokertoimet ovat aina positiivisia solmupisteiden lukumäärästä n riippumatta. Tämän vuoksi menetelmät ovat numeerisesti stabiileja toisin kuin esimerkiksi Newtonin ja Cotesin kvadratuurit, jotka esitellään seuraavassa kappaleessa.

Gaussin kaavojen huonona puolena on solmupisteiden epätasainen jakautuminen integrointivälille, minkä seurauksena kaavat eivät ole progressiivisia. Jos halutaan tarkempia tuloksia lisäämällä solmujen lukumäärää, vanhoista pisteistä ei ole apua, vaan korkeamman kertaluvun menetelmää varten joudutaan laskemaan täysin uudet pisteet ja funktion arvot niissä. Tämä on erityisen hankalaa, jos funktion arvot tunnetaan kokeellisten mittausten tuloksena. Myös itse solmupisteiden ratkaiseminen muuttuu työläämmäksi kertaluvun kasvaessa.

Virhearviossa esiintyvän derivaatan korkean kertaluvun vuoksi tarvittavien solmupisteiden määrän arvioiminen ennakolta voi olla vaikeaa.

Gaussin kvadratuurien pohjalta on kehitetty uusia menetelmiä, joissa yritetään välttää pahimmat hankaluudet. *Kronrod* esitti n:n pisteen Gaussin kvadratuurin täydentämistä $n+1$ lisäpisteellä, jolloin saadaan integroitua tarkasti aina astetta $3n+1$ (n parillinen) tai $3n+2$ (n pariton) olevat polynomit. Kronrodin pisteiden laskeminen vaatii kuitenkin suurta tarkkuutta.

Patterson kehitti Kronrodin ideasta lähtien edelleen yleisemmän kokoelman menetelmiä, joissa lisäpisteiden määrä p ei riipu pohjana olevan Gaussin kvadratuurin solmupisteiden lukumäärästä. Uuden menetelmän kertaluku on $n + 2p - 1$. Käytännössä useimmin lähdetään liikkeelle 3 pisteen Gaussin kaavasta, ja tarkennetussa kvadratuurissa on aina $2n+1$ pistettä, jos edellisessä oli n pistettä. Näin syntyvässä menetelmäketjussa on siis $3, 7, 15, 31, 63, ..., 2^k - 1$ pisteen kaavoja.

5.4 Clenshaw'n ja Curtisin kaavat

Kuten kappaleessa 4.5.2 kerroimme, funktion $f(x)$ polynomi-interpoloinnissa saavutetaan yleensä parempia tuloksia valitsemalla interpolointipisteiksi Tšebyševin pisteet kuin käyttämällä tasavälistä pisteistöä. Koska numeerisia integrointialgoritmeja voidaan johtaa interpolaatioista, on luontevaa tutkia millaisia kvadratuureja saadaan aikaan, jos solmupisteiksi valitaan integrointivälin Tšebyševin pisteet.

Tarkastellaan jälleen väliä $[-1,1]$. Astetta n olevan Tšebyševin polynomin nollakohdat osuvat pisteisiin

$$x_i = \cos\frac{(2i-1)\pi}{2n}, \quad i = 1,2,...,n,$$

ja ääriarvot pisteisiin

$$x_i = \cos\frac{(i-1)\pi}{n-1}, \quad i = 1,2,...,n.$$

Molemmat pistejoukot johtavat toimiviin kvadratuureihin. Nollakohtia käyttämällä päädytään *Polyan kaavoihin*. Näissä on kuitenkin sama huono puoli kuin Gaussin kvadratuureissa: kertaluvun kaksinkertaistaminen merkitsee kaikkien solmupisteiden uusimista. Tšebyševin polynomien ääriarvopisteisiin perustuvien *Clenshaw'n ja Curtisin kvadratuurien* kohdalla tätä ongelmaa ei ole, kun siirrytään n:n pisteen menetelmästä $2n-1$:n pisteen menetelmään. Siksi integraalin likiarvon tarkentaminen vaatii huomattavasti vähemmän työtä.

Painokertoimien w_i arvot saadaan summalausekkeista, jotka ovat johdettavissa Tšebyševin polynomien ortogonaalisuusominaisuuksien avulla. Lopullinen kvadratuuri näyttää seuraavalta:

$$Q_n(f;-1,1) = -\frac{4}{n}\sum_{j=0}^{n}{}'' f(x_j)\sum_{k=0}^{n}{}'' w_{j,k},$$

jossa

$$w_{j,k} = \begin{cases} (\cos\frac{\pi jk}{n})/(k^2-1), & k \text{ parillinen}, \\ 0, & k \text{ pariton}. \end{cases}$$

Summalausekkeisiin liitetty heittomerkki tarkoittaa, että summien ensimmäinen ja viimeinen termi pitää kertoa tekijällä $\frac{1}{2}$.

Clenshaw'n ja Curtisin kaavojen kertaluku on vain n, mutta käytännössä ne ovat paljon tarkempia kuin pelkän kertaluvun perusteella voisi päätellä. Lisäksi Clenshaw'n ja Curtisin kvadratuurit ovat progressiivisia, joten työmääräänsä nähden ne ovat täysin kilpailukykyisiä Gaussin kvadratuurien kanssa, kun tavoitteena on saavuttaa sama tarkkuus.

5.5 Newtonin ja Cotesin kaavat

Newtonin ja Cotesin kaavat ovat tasaväliseen interpolaatioon pohjautuvia kvadratuureja. Matalan kertaluvun Newtonin ja Cotesin kvadratuurit (puolisuunnikassääntö, Simpsonin sääntö) ovat yksinkertaisimpia ja käytetyimpiä numeerisia integrointimenetelmiä. Ne ovat myös vanhimpia: Kepler käytti Simpsonin sääntöä jo 1612 eli noin 130 vuotta ennen kuin Simpson "keksi" säännön.

Ikävä kyllä matalan kertaluvun Newtonin ja Cotesin kaavat ovat melko tehottomia uudempiin menetelmiin verrattuna. Korkean kertaluvun kaavoissa puolestaan esiintyy negatiivisia painokertoimia, joten nämä menetelmät ovat merkitsevien numeroiden kumoutumisen vuoksi epäluotettavia. Niinpä käytännössä ainoaksi keinoksi tarkkuuden parantamiseen jää pienempien osavälien käyttö.

Newtonin ja Cotesin kaavat sopivat hyvin ohjelmiin, joissa ei vaadita suurta tarkkuutta tai joita ei ole tarkoitus ajaa useita kertoja. Niitä voidaan käyttää myös yhdessä Rombergin menetelmän kanssa, jolloin tulosta parannetaan ekstrapoloimalla (sivu 171). Seuraavassa esittelemme kolme alimman kertaluvun menetelmää.

Puolisuunnikassääntö: Puolisuunnikassääntö perustuu integrandin $f(x)$ lineaariseen interpolaatioon välillä $[a,b]$. Peruskaava virhetermeineen on

$$\int_a^b f(x)\,dx = \frac{b-a}{2}\left[f(a)+f(b)\right] - \frac{(b-a)^3}{12}f''(\xi), \quad \xi \in [a,b]. \tag{5.2}$$

Puolisuunnikassäännön kertaluku on 1.

Yhdistetty puolisuunnikassääntö: jos funktion $f(x)$ arvot tunnetaan pisteissä $a = x_1 < x_2 < \ldots < x_n < x_{n+1} = b$, niin

$$\int_a^b f(x)\,dx = \sum_{j=1}^{n} \frac{x_{j+1}-x_j}{2}\left[f(x_j)+f(x_{j+1})\right].$$

Erityisesti jos jaamme välin $[a,b]$ n yhtäsuureen osaan ja merkitsemme $h = (b-a)/n$, $x_j = a + (j-1)h$, $j = 1,..,n+1$, niin

$$\int_a^b f(x)\,dx = h\left[\frac{1}{2}f(x_1) + f(x_2) + \cdots + f(x_n) + \frac{1}{2}f(x_{n+1})\right] \tag{5.3}$$

$$-\frac{b-a}{12}h^2 f''(\xi), \quad \xi \in [a,b].$$

Simpsonin sääntö: Simpsonin säännön takana on tasavälinen kvadraattinen interpolaatio, joten integrointiväliltä käytetään päätepisteiden lisäksi myös yhtä lisäpistettä, peruskaavassa välin keskipistettä. Newtonin ja Cotesin menetelmien kertaluku nousee kahdella aina kun solmupisteiden määrä kasvaa parillisesta parittomaan, joten Simpsonin säännön kertaluku on 3, mikä

osaltaan selittää sen suosiota. Jos merkitsemme ensin apumuuttujalla h välin $[a,b]$ pituuden puolikasta ($h = (b-a)/2$), Simpsonin sääntö voidaan esittää muodossa

$$\int_a^b f(x)\,dx = \frac{h}{3}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right] - \frac{h^5}{90}f^{(4)}(\xi), \quad \xi \in [a,b]. \tag{5.4}$$

Virhe on siis verrannollinen tekijään h^5.

Yhdistetty Simpsonin sääntö: Oletamme, että integroitavan funktion $f(x)$ arvo tunnetaan $2n+1$ pisteessä $a = x_1 < x_2 < \ldots < x_{2n} < x_{2n+1} = b$, jotka jakavat koko integrointivälin $[a,b]$ yhteensä n osaan

$$[x_1, x_3], [x_3, x_5], \ldots, [x_{2n-1}, x_{2n+1}].$$

Periaatteessa pisteistö $\{x_j\}$ voi sijaita integrointivälillä epätasaisestikin, mutta tavallisesti käytetään tasavälistä jakoa: $h = (b-a)/2n$, $x_j = a + (j-1)h$, $j = 1, \ldots, 2n+1$. Tällöin

$$\int_a^b f(x)\,dx = \frac{h}{3}\left[f(x_1) + 4\sum_{k=1}^{n} f(x_{2k}) + 2\sum_{l=1}^{n-1} f(x_{2l+1}) + f(x_{2n+1})\right] - \frac{b-a}{180}h^4 f^{(4)}(\xi). \tag{5.5}$$

3/8-*sääntö:* Seuraava suljettu Newtonin ja Cotesin kaava on nimeltään 3/8-sääntö. Vaikka 3/8-sääntö vaatii funktion arvon neljässä pisteessä integrointiväliltä, sen kertaluku on edelleen 3 kuten Simpsonin säännönkin, joten 3/8-säännön edellyttämä suurempi työmäärä ei yleensä tuota vastaavaa hyötyä. 3/8-sääntöä voi kuitenkin käyttää yhdessä Simpsonin säännön kanssa tilanteissa, joissa parempaan tarkkuuteen pyritään yhdistetyillä säännöillä, mutta osavälien lukumäärä ei satu olemaan parillinen.

Apumuuttuja $h = (b-a)/3$, piste $\xi \in [a,b]$ ja kokonaisuudessaan sääntö näyttää seuravalta:

$$\int_a^b f(x)\,dx = \frac{3h}{8}\left[f(a) + 3f(a+h) + 3f(b-h) + f(b)\right] - \frac{3h^5}{80}f^{(4)}(\xi).$$

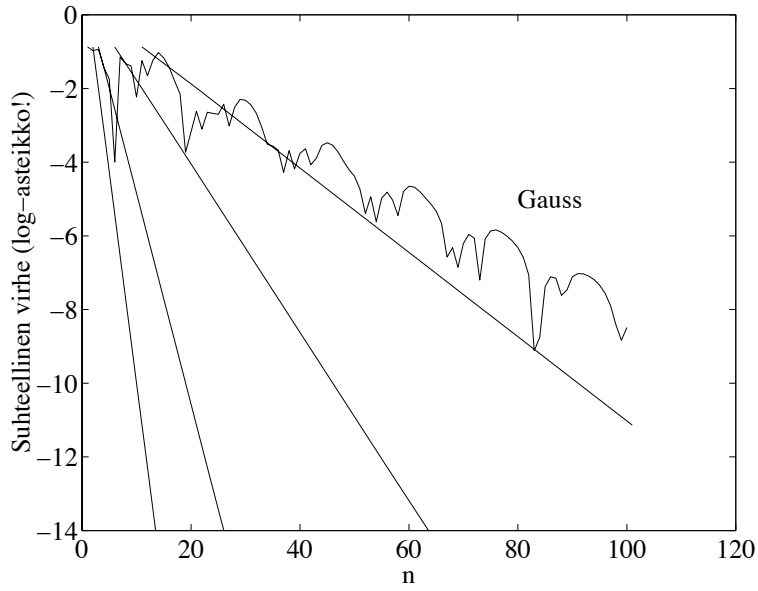
Joissain erikoistapauksissa Newtonin ja Cotesin kaavat ovat edellä mainituista huonoista puolistaan huolimatta hyvin käyttökelpoisia. Yhdistetty puolisuunnikassääntö on erittäin tehokas menetelmä, kun integroidaan jaksollista funktiota jaksonpituuden (tai sen monikerran) mittaisen välin yli. Myös sopivalla tavalla konstruoitujen aallokkeiden integroinnissa se antaa erinomaisia tuloksia. Aallokkeiden pitää tällöin toteuttaa momenttiehdot (4.46).

Esimerkki 5.5.1 Integroimme numeerisesti funktion

$$f(x) = \frac{1}{1 + \sin(10\pi x)/2}$$

välillä $[0, 1]$. Kuvassa 5.1 on esitetty suhteellisen virheen käyttäytyminen kvadratuuripisteiden lukumäärän n funktiona. Neljä suoraa vastaavat yhdistettyjä puolisuunnikassääntöjä, joissa jakovälien lukumäärä $l = n - 1$ toteuttaa seuraavat ehdot (suorat vasemmalta oikealle): [1] l pariton, $l \not\equiv 0 (\mathrm{mod} 5)$, [2] l parillinen, $l \not\equiv 0 (\mathrm{mod} 5)$, [3] l pariton, $l \equiv 0 (\mathrm{mod} 5)$, [4] l parillinen, $l \equiv 0 (\mathrm{mod} 5)$. Kvadratuuripisteet kannattaa siis valita siten, että funktiosta saadaan näytteitä mahdollisimman monipuolisesti. Jos pisteet valitaan huolimattomasti, ne osuvat aina jakson samaan kohtaan.

Vertailun vuoksi samassa kuvassa näkyy Gaussin kvadratuurin suppeneminen, joka tässä tapauksessa on selvästi hitaampaa. Työmäärää arvioitaessa on muistettava, että Gaussin menetelmän pisteet ja painot pitää erikseen syöttää sisään tai laskea iteroimalla, kun taas puolisuunnikassäännön pisteiden selvittäminen on triviaalia.



Kuva 5.1: *Puolisuunnikassäännön (ja Gaussin kvadratuurin) virhe jaksollisen funktion integroinnissa.*

Joskus on tarkoituksenmukaista kehittää erityinen kvadratuuri kokonaista integraaliluokkaa varten. Tästä on hyvä esimerkki Fourier'n integraalit $\int_a^b f(x)\cos(\omega t)\,dt$. Tällaisissa tapauksissa Newtonin ja Cotesin kaavoja on usein helpompi käsitellä kuin Gaussin kvadratuureja.

Avoimet Newtonin ja Cotesin kaavat puolestaan sopivat karkeiden arvioiden laskemiseen integraaleille, joilla on epäjatkuvuus välin päätepisteessä.

5.6 Tšebyševin integrointi

Newtonin ja Cotesin kaavoissa solmupisteet sijaitsevat tasaisin välein. Kaavoista selviävät painokertoimet, joilla funktion arvot eri interpolaatiopisteissä otetaan mukaan integraalia arvioitaessa. Kvadratuurin muodostamistehtävä voidaan asettaa myös toisin päin: kuinka solmupisteet $\{x_i\}$ tulisi valita, jotta kaikki painokertoimet $\{w_i\}$ olisivat yhtä suuria? Tällaisella kvadratuurilla on kysyntää sellaisessa tilanteessa, jossa funktion arvot joudutaan selvittämään kokeellisesti tai niiden laskeminen on hyvin monimutkaista, jolloin arvoihin sisältyy satunnaista virhettä. Samansuuruisilla painokertoimilla pyritään pitämään huoli siitä, että funktion arvoihin sisältyvät virheet kumoutuvat.

Edellä kuvattuun ideaan perustuvat numeeriset integrointikaavat tunnetaan Tšebyševin kvadratuurien nimellä. Huomaa, että tällä *ei* ole tekemistä Tšebyševin interpolaation kanssa, johon perustuvat kvadratuurit on puolestaan nimetty Clenshaw'n ja Curtisin mukaan (katso kappaletta 5.4).

Tarkastellaan väliä $[-1, 1]$. Jos muodostetaan n pisteen kvadratuuri, jokaista solmua tulee vastaamaan paino $2/n$ (vakiofunktion pitää integroitua tarkasti). Solmupisteiden $\{x_i\}$ paikat ratkeavat epälineaarisesta yhtälöryhmästä (5.1), kun painojen paikalle sijoitetaan edellä laskettu tulos. Syntyvillä systeemeillä on reaaliset ratkaisut $\{x_i\}$ vain, kun $0 \leq n \leq 7$ tai $n = 9$.

Tšebyševin kvadratuurien solmupisteet välillä $[-1, 1]$ on esitetty taulukossa 5.2.

5.7 Erikoistapauksia

Suurin osa numeerisista kvadratuureista on yleismenetelmiä, jotka toimivat periaatteessa hyvin, kun pitää integroida sileä funktio äärellisellä välillä. Monesti kvadratuureja tarvitaan kuitenkin tilanteissa, joihin perusmenetelmät eivät sovellu lainkaan tai joissa suppeneminen on parhaassakin tapauksessa hyvin hidasta. Seuraavassa käydään läpi joitain erikoistapauksia.

5.7.1 Epäoleelliset integraalit

Suomen kielessä käytetään hiukan harhaanjohtavaa nimitystä *epäoleellinen integraali* tapauksista, joissa itse integrointiväli on äärettömän pitkä (1. laji) tai integrandilla on singulariteetti integrointivälillä (2. laji). Kummankin lajin kohdalla numeeristen kvadratuurien käyttö edellyttää yleensä joko erityismenetelmiä tai tehtävän saattamista analyyttisin keinoin sellaiseen muotoon, että standardimenetelmät selviytyvät niistä.

Jos integrointiväli on ääretön, kannattaa miettiä, ovatko nimenomaan tällaista tapausta varten kehitetyt Gaussin ja Laguerren tai Gaussin ja Hermiten kvadratuurit sopivia.

Taulukko 5.2: *Tšebyševin kvadratuurien solmupisteet ja painot.*

n	x_i	w_i
1	$x_1 = 0$	$w_1 = 2$
2	$x_2 = -x_1 = 0.57735026918963$	$w_i = 1$
3	$x_3 = -x_1 = 0.70710678118655$ $x_2 = 0$	$w_i = 2/3$
4	$x_4 = -x_1 = 0.79465447229177$ $x_3 = -x_2 = 0.18759247408508$	$w_i = 2/4$
5	$x_5 = -x_1 = 0.83249748700098$ $x_4 = -x_2 = 0.37454140955358$ $x_3 = 0$	$w_i = 2/5$
6	$x_6 = -x_1 = 0.86624681810782$ $x_5 = -x_2 = 0.42251865376111$ $x_4 = -x_3 = 0.26663540151670$	$w_i = 2/6$
7	$x_7 = -x_1 = 0.88386170075805$ $x_6 = -x_2 = 0.52965677528516$ $x_5 = -x_3 = 0.32391181051991$ $x_4 = 0$	$w_i = 2/7$

Toinen vaihtoehto on katkaista integrointiväli äärelliseksi. Tällöin jäljelle jäävään integraaliin voidaan soveltaa tavanomaisia menetelmiä. Loppuosan merkitystä on mahdollista yrittää arvioida analyyttisesti esimerkiksi sarjakehitelmien avulla.

Kolmas tapa selvitä tilanteesta on muuttujanvaihdos, jolla ääretön väli supistuu äärelliseksi. Puoliäärettömälle välille $[0, \infty]$ sopivia muuttujanvaihdoksia ovat esimerkiksi $t = x/(1+x)$ ja $t = e^{-x}$. Äärettömälle välille $[-\infty, \infty]$ voi kokeilla vaikka sijoitusta $t = (e^x - 1)/(e^x + 1)$.

Esimerkki 5.7.1 Haluamme laskea arvon integraalille

$$\int_0^\infty e^{-x} \sin x \, dx.$$

Integroitava funktio on muotoa $e^{-x} g(x)$, joten Gaussin ja Laguerren kvadratuuri sopii sellaisenaan tähän tehtävään. Kuudella pisteellä saamme likiarvon 0.5000495 (tässä tehtävässä tarkka ratkaisu $1/2$ olisi helppo laskea analyyttisestikin).

Muuttujanvaihdoksella $t = e^{-x}$ päädymme tarkastelemaan integraalia

$$\int_0^1 -\sin(\ln(t))\,dt,$$

joka sisältää logaritmisen singulariteetin välin alkupisteessä. Singulariteettien käsittelyyn annamme lisäohjeita myöhemmin (esimerkki 5.7.4), jolloin kerromme myös kuinka tästä muodosta päästään eteenpäin.

Palaamme vielä alkuperäiseen probleemaan, ja toteamme, että myös integrointivälin katkaiseminen äärelliseksi esimerkiksi kohdasta 10π on toimiva vaihtoehto. Alkupään integraali $\int_0^{10\pi} e^{-x}\sin x\,dx$ on laskettavissa tavanomaisin menetelmin, vaikka se periaatteessa edustaakin oskilloivaa erikoistapausta, jolle on kehitetty omia tehokkaita kvadratuureja. Katkaisuvirhettä $\int_{10\pi}^{\infty} e^{-x}\sin x\,dx$ voi arvioida vaikkapa seuraavasti:

$$|e^{-x}\sin x| \le e^{-x} < x^{-8}, \quad \text{kun } x \ge 10\pi,$$

joten $$\int_{10\pi}^{\infty} e^{-x}\sin x\,dx < \int_{10\pi}^{\infty} x^{-8} = 1/(7\cdot(10\pi)^7) \approx 4.7\cdot 10^{-12}.$$

Singulaaristen integraalien käsittelyyn on useita menettelytapoja. Kvadratuurien ongelmattoman soveltamisen kannalta ei riitä, että integrandilla ei ole napoja integrointivälillä, vaan myös derivaattojen pitäisi olla sileitä. Kuinka sileitä, riippuu käytetyn kvadratuurin kertaluvusta. Niinpä funktion $f(x) = \sqrt{x}$ integraalien arviointi origon ympäristössä yhdistetyn Simpsonin menetelmän (kertaluku 3) avulla on epätarkempaa kuin kaukana origosta.

Muistamisen arvoinen asia on, että hajaantuva integraali ei muutu suppenevaksi, vaikka sen ”arvon” saisikin numeerisella kvadratuurilla laskettua!

Esimerkki 5.7.2 Sovellamme virheellisesti kahden pisteen Gaussin kvadratuuria *hajaantuvaan* integraaliin $\int_0^1 \frac{dx}{x}$.

$$\int_0^1 \frac{dx}{x} \approx \frac{1}{2}\left(\frac{1}{0.211325} + \frac{1}{0.788675}\right) = 6.$$

Tämänkaltainen huomattava virhe voi tapahtua, jos tietokoneohjelmassa kutsutaan integrointialiohjelmaa sokeasti varmistamatta, että integraali on järkevä.

Tarkastellaan integraalia $\int_0^1 f(x)\,dx$, jossa integrandilla $f(x)$ on singulaaripiste origossa. Muut välit ovat lineaarisella skaalauksella palautettavissa tälle välille. Jos integroitavalla funktiolla on singulariteetti keskellä integrointiväliä, tehtävä hajoaa kahdeksi integraaliksi, joilla on päätysingulariteetti.

Jos taas alkuperäisellä integraalilla on singulariteetti molemmissa päätepisteissä, sen voi kirjoittaa summaksi kahdesta sellaisesta integraalista, joilla on singulaariteetti vain pisteessä $x = 0$:

$$\int_0^1 f(x)\,dx = \frac{1}{2}\int_0^1 [f(x/2) + f(1 - x/2)]\,dx.$$

Jälkimmäisen termin kanssa on oltava huolellinen, ettei tapahdu merkitsevien numeroiden katoa, jos jokin kvadratuuripiste x_i on hyvin pieni. Tietokonearitmetiikassa $1 - x_i/2 = 1$, kun $x_i \approx 2\epsilon$, missä ϵ taas on pienin luku, jolle $1 + \epsilon > 1$.

Yksinkertaisin tapa selvitä singulariteetista on singulaaripisteen välttäminen avoimien kvadratuurien avulla tai asettamalla integraali alkamaan vasta singulaaripisteen jälkeen. Tällöin integroitavan funktion arvoa ongelmapisteessä ei tarvita. Tavalliset Gaussin kvadratuurit ovat avoimia, joten ne ovat sellaisenaan käytettävissä.

Kehittämällä tätä periaatetta pidemmälle päädytään tekniikkaan, jossa lähestytään singulariteettia askeleittain. Muodostetaan jono jakopisteitä $\{z_j\}$, $0 < z_{j+1} < z_j < 1,\ j = 0, 1, 2, \ldots$ (esimerkiksi $z_j = \theta^j$, $\theta = 0.2$), ja käytetään väleillä $I_j = [z_{j+1}, z_j]$ jotain yleistä kvadratuuria. Alkuperäiselle integraalille saadaan näin approksimaatioita sarjasta

$$S_n = \sum_{j=0}^{n} \int_{I_j} f(x)\,dx.$$

Jos tämän sarjan termeihin sovelletaan jotain suppenemista kiihdyttävää algoritmia, pystytään hyvinkin nopeasti laskemaan luotettava likiarvo summalle S_∞.

Joissain tapauksissa kvadratuurin ulkopuolelle jätetyn välin $[0, z_n]$ merkitystä voi arvioida kehittämällä integrandin sarjaksi.

Esimerkki 5.7.3 Käsittelemme osissa integraalin

$$\int_0^1 \frac{e^t}{\sqrt{t}}\,dt = \int_0^\delta \frac{e^t}{\sqrt{t}}\,dt + \int_\delta^1 \frac{e^t}{\sqrt{t}}\,dt$$

Jälkimmäinen integraali ei tuota ongelmia perusmenetelmille. Edellisessä taas voidaan osoittaja kehittää sarjaksi, jonka termit integroituvat analyyttisesti:

$$\int_0^\delta \frac{e^t}{\sqrt{t}}\,dt = \int_0^\delta \frac{1 + t + t^2/2 + t^3/6 + \ldots}{\sqrt{t}}\,dt$$

$$= 2\left(\sqrt{\delta} + \frac{\delta^{3/2}}{3} + \frac{\delta^{5/2}}{5 \cdot 2!} + \frac{\delta^{7/2}}{7 \cdot 3!} + \ldots\right).$$

Koska $\delta < 1$, sarja suppenee nopeasti. Jos δ on hyvin lähellä arvoa 0, sarja kyllä suppenee erittäin nopeasti, mutta jälkimmäisen integraalin tarkkuus alkaa kärsiä singulariteetin läheisyyden vuoksi.

Kuten 1. lajin epäoleellisten integraalien kohdalla, myös singulariteettien käsittelyssä muuttujanvaihdos on tehokas toimintatapa. Yksinkertaisin muunnos on $x = t^n$, jolla pääsee eroon monista tavanomaisista singulariteeteista, kunhan n on tarpeeksi suuri.

Esimerkki 5.7.4 Yritämme laskea likiarvon esimerkissä 5.7.1 esiintyneelle integraalille

$$\int_0^1 -\sin(\ln x)\,dx.$$

Tavanomainen Gaussin ja Legendren 6 pisteen kvadratuuri antaa tulokseksi 0.492207, ja suhteellinen virhe on 1.5%. Kokeilemme myös tasaväliseen jakoon perustuvaa yhdistettyä puolisuunnikassääntöä, jossa integroitavan funktion singulariteetti sivuutetaan asettamalla funktion arvoksi origossa 0 asiaa syvällisemmin pohtimatta. Näin menetellen saamme integraalin likiarvoksi 0.527354 (virhe 5.5%).

Suoritamme sitten muuttujanvaihdoksen $x = t^k$, jolloin päädymme integraaliin

$$\int_0^1 kt^{k-1}\sin(k\ln t)\,dt.$$

Koska integrandi nyt todella häviää origossa, kun k on vaikkapa 2, on tehtävä ratkaistavissa myös suljetuilla kvadratuureilla. Kymmenen välin yhdistetty puolisuunnikassääntö tuottaa tällä kerralla luvun 0.493456 (virhe 1.3%). Laskemme integraalin vielä vertailun vuoksi Gaussin kvadratuurilla muuttujanvaihdoksen jälkeen. Tulos on 0.502239 ja suhteellinen virhe jää alle prosentin (0.4%).

Gaussin kvadratuurin tarkentuminen selittyy muuttujanvaihdoksen aiheuttamalla integrandin siloittumisella. Mitä suurempi k, sitä sileämmin integrandi käyttäytyy origon läheisyydessä, mutta toisaalta arvon 1 lähelle alkaa syntyä voimakas piikki. Esimerkiksi k:n arvolla 5 samainen 6 pisteen Gaussin kvadratuurin virhe on vain 0.03%, mutta k:n arvolla 10 jälleen 1.2%.

Joskus singulariteetista on mahdollista päästä eroon lisäämällä integraaliin sopiva termi, jonka avulla tehtävä voidaan kirjoittaa kahden integraalin summana, joista toinen on laskettavissa analyyttisesti ja toinen numeerisin menetelmin.

Esimerkki 5.7.5 Laskemme likiarvon lausekkeelle

$$I = \int_0^1 \frac{\ln x}{1+x^2}dx.$$

Tässä tapauksessa selviämme singulariteetista yksinkertaisesti lisäämällä ja

vähentämällä termin $\ln x$:

$$\begin{aligned} I &= \int_0^1 \frac{\ln x}{1+x^2} - \ln x \, dx + \int_0^1 \ln x \, dx \\ &= -\int_0^1 \frac{x^2 \ln x}{1+x^2} \, dx + \int_0^1 \ln x \, dx. \end{aligned}$$

Ensimmäisen integraalin osoittajassa oleva x^2 vaimentaa logaritmisen singulariteetin. Jälkimmäinen integraali puolestaan ratkeaa analyyttisesti.

5.7.2 Erikoisfunktiot

Eräät matemaattiset erikoisfunktiot on määritelty integraalien avulla. Usein näissä tapauksissa esiintyvät integraalit eivät ratkea analyyttisesti, mutta funktioiden tärkeyden vuoksi integraalien laskemiseen on kehitetty hyvin pitkälle erikoistuneita menetelmiä. Erikoisfunktioiden evaluoinnissa valmiiden aliohjelmakirjastojen käyttö on yleensä selvästi paras vaihtoehto.

Esimerkki 5.7.6 Aliohjelmakirjasto Harwell Subroutine Library (HSL) sisältää useita erikoisfunktioille suunniteltuja rutiineja. Ohjelma FC01 evaluoi funktion

$$W(z) = e^{-z^2} \left(1 + \frac{2i}{\sqrt{\pi}} \int_0^z e^{t^2} \, dt \right), \quad z = x + iy, \quad x, y > 0$$

arvon jollain seuraavista approksimaatioista:

1. Jos $|z| \leq 1$, lasketaan muutama ensimmäinen termi integrandin e^{t^2} sarjakehitelmästä.
2. Jos $1 < |z| \leq 4$ ja $y < 1.4$, käytetään sarjakehitelmää jonkin sellaisen pisteen ympäristössä, jossa $W(z)$ tunnetaan.
3. Jos $1 < |z| \leq 4$ ja $y > 1.4$, sovelletaan keskipistesääntöä integraaliin

$$\frac{i}{\pi} \int_{-\infty}^{\infty} \frac{e^{-t^2}}{z-t} dt.$$

4. Jos $|z| > 4$, ohjelma suoriutuu tehtävästä Gaussin ja Hermiten menetelmän avulla.
5. Lisäksi alueen $x, y > 0$ ulkopuolella käytetään palaustuskaavoja, joilla tehtävä saadaan palautettua johonkin tapauksista 1–4.

Jos z osuu eri menetelmien sovellusalueiden rajalle, integraalin arvo lasketaan kahdella eri tavalla ja lopputulos saadaan interpoloimalla.

5.7.3 Oskilloivat funktiot

Oskilloivilla integraaleilla tarkoitetaan seuraavassa muotoa

$$\int_a^b g(x) f(x)\, dx$$

olevia lausekkeita, joissa $g(x)$ on oskilloiva osa, esimerkiksi $\sin kx$ tai $\cos kx$, $(b-a) \ll k$ ja funktio $f(x)$ on sileä.

Tämänkaltaiset integraalit esiintyvät monilla sovellusalueilla, mutta toisaalta kaikkein tavallisimmat kvadratuurit eivät yleensä selviä niistä tyydyttävästi. Hyvissä matemaattisissa aliohjelmakirjastoissa on omat rutiinit oskilloivia integraaleja varten.

Äärellisellä välillä sopivan integrointimenetelmän lähtökohdaksi kannattaa lainata Newtonin ja Cotesin tai Clenshaw'n ja Curtisin kvadratuurien ideoita. Gaussin kvadratuurin kaltaisten, erittäin tehokkaiden menetelmien suunnittelu kariutuu solmupisteiden laskentaan, joka pitäisi tehdä uudelleen paitsi eri pistemäärille myös muuttujan k eri arvoille.

Filonin menetelmässä integrointiväli jaetaan osiin, joista jokaisella integrandin sileää tekijää $f(x)$ interpoloidaan 2. asteen polynomilla. Koska monomien 1, x, x^2 ja trigonometristen funktioiden $\sin kx$, $\cos kx$ tulot ovat integroitavissa analyyttisesti, tällä keinolla saadaan aikaiseksi kvadratuuri, jota on helppo soveltaa kaikilla muuttujan k arvoilla. Filonin menetelmä on Simpsonin säännön yleistys oskilloivaan tapaukseen.

Clenshaw'n ja Curtisin kvadratuurit ovat periaattessa huomattavasti tehokkaampia kuin Newtonin ja Cotesin kaavat. Jos yritetään edetä analogisesti edellä kuvatun idean mukaan tavoitteena Clenshaw'n ja Curtisin kvadratuurien yleistys, törmätään Tšebyševin polynomien ja trigonometristen funktioiden tuloihin, jotka eivät integroidu suljetussa muodossa. Turvautumalla sarjakehitelmiin ja Tšebyševin polynomien ominaisuuksiin on kuitenkin mahdollista johtaa kvadratuuri, joka on selvästi Filonin menetelmää tarkempi [AEH76].

Jos oskilloiva integraali on määritelty puoliäärettömälle välille, se kannattaa purkaa sarjaksi $\sum_{j=0}^{N} I_j$. Tässä jokainen termi I_j on sellaisen osavälin yli laskettu integraali, joka vastaa π:n suuruista oskilloivan tekijän $g(x)$ vaihekulman muutosta. Siis jos $g(x) = \sin kx$, niin

$$I_j = \int_{x_j}^{x_{j+1}} f(x) \sin kx\, dx, \quad x_j = \frac{j\pi}{k}.$$

Sarjan suppenemista voi nopeuttaa sopivalla kiihdytinalgoritmilla.

5.8 Ekstrapolointi ja adaptiivinen integrointi

Aiemmin on mainittu kaksi tapaa tuottaa tarkempia approksimaatioita numeerisilla kvadratuureilla: menetelmien kertaluvun nostaminen ja integrointivälin jakaminen pienempiin osaväleihin. Yleensä edellinen on työmäärään nähden tehokkaampi. Kuitenkin esimerkiksi Newtonin ja Cotesin sääntöjen kohdalla kertaluvun nostaminen yli arvon 8 on käytännössä mahdotonta. *Ekstrapolointi* ja *adaptiiviset menetelmät* ovat keinoja, joilla saadaan parempia tuloksia ilman, että työmäärä kasvaa ylivoimaiseksi.

5.8.1 Ekstrapolointi

Jos yhdistettyä puolisuunnikassääntöä (5.3) sovelletaan välillä $[a, b]$ käyttämällä n osaväliä, tulos $T(h)$ voidaan kirjoittaa osavälin pituuden $h = (b-a)/n$ kehitelmänä muotoon

$$T(h) = \tau_0 + \tau_1 h^2 + \tau_2 h^4 + \dots$$

Tässä τ_0 on haettu tarkka arvo $\int_a^b f(x)\,dx$. Kehitelmän kertoimet ja jäännöstermi ovat äärellisiä, kunhan $f(x)$ on riittävän monta kertaa jatkuvasti derivoituva. Ylläoleva yhtälö on mahdollista tulkita myös siten, että kyseessä on funktion T arvon laskeminen origon lähellä sijaitsevassa pisteessä h, ja lopullinen kiinnostuksen kohde on nimenomaan $T(0)$, jota ei kuitenkaan suoraan osata laskea. Eräs keino arvon $T(0)$ selvittämiseksi on tällöin ekstrapolointi, jota varten on laskettava funktion T arvo useilla pienillä luvuilla $h_0, h_1, h_2, \dots, h_m$.

Idean ekstrapolaation käytöstä integroinnin yhteydessä esitti alunperin W. Romberg vuonna 1955, joten menetelmä on nimetty hänen mukaansa. Romberg ehdotti askelpituuksien valitsemista siten, että $h_i = (b-a)/2^i$, mutta myöhemmin on kehitetty myös muita mahdollisuuksia.

Ekstrapolaatio on kätevintä toteuttaa Nevillen interpolaatioalgoritmin (sivu 103) avulla. Merkitään ensimmäiseen sarakkeeseen arvot $T_{i,0} = T(h_i)$, $i = 0, 1, \dots, m$. Seuraavat arvot lasketaan kaavasta

$$T_{i,k} = T_{i,k-1} + \frac{T_{i,k-1} - T_{i-1,k-1}}{(h_{i-k}/h_i)^2 - 1}.$$

Lopullinen arvio integraalille $\tau_0 = \int_a^b f(x)\,dx$ on luku $T_{m,m}$.

Esimerkki 5.8.1 Laskemme integraalin

$$\int_0^1 \frac{dx}{1+x^2} = \pi/4 \approx 0.78539816$$

likiarvon ekstrapoloimalla yhdistetyllä puolisuunnikassäännöllä tuotettuja välituloksia, kun osavälien pituudet h_i ovat $h_0 = 1$, $h_1 = 0.5$ ja $h_2 = 0.25$. Kolmen

sarakkeen taulukko näyttää seuraavalta

$$\begin{array}{lll} T_{0,0} = 0.75 & & \\ T_{1,0} = 0.775 & T_{1,1} = 0.783333333 & \\ T_{2,0} = 0.78279412 & T_{2,1} = 0.78539216 & T_{2,2} = 0.78552941 \end{array}$$

Likiarvossa $T_{2,0}$ on virhettä 0.332%, kun taas ekstrapoloidussa tuloksessa $T_{2,2}$ suhteellinen virhe on enää 0.0167%. Pelkällä yhdistetyllä puolisuunnikassäännöllä tähän tarkkuuteen olisi vaadittu välin $[0,1]$ jakamista 18 osaan, jolloin olisimme joutuneet evaluoimaan integrandin 19 pisteessä. Nyt selvisimme siis kaikkiaan 5 pisteellä, mutta vastaavasti ekstrapolaatiosta aiheutui jonkin verran lisätyötä.

Tarkkaavainen lukija huomasi myös, että välitulos $T_{2,1}$ oli paljon tarkempi kuin lopullinen tulos $T_{2,2}$.

5.8.2 Adaptiivinen integrointi

Integrointitarkkuuden parantaminen kaikkialla integrointialueessa johtaa helposti turhaan työhön, jos integroitavan funktion arvo vaihtelee vain vähän suurimmassa osassa aluetta. *Adaptiivisessa integroinnissa* pyritään tihentämään jakoa pienempiin osa-alueisiin tai nostamaan kvadratuurin kertalukua vain siellä, missä siitä todella on apua.

Adaptiivisten menetelmien pohjalla on jokin peruskvadratuuri, jota on helppo tarkentaa, usein Pattersonin menetelmä (sivu 159). Myös alueen tiheämpää osittamista voidaan käyttää, esimerkiksi integrointivälien puolittamista (bisektio) yhdessä dimensiossa. Lisäksi tarvitaan riittävän luotettava keino lokaalin virheen arvioimiseksi ja päätöskriteeri, jolla valitaan ne osa-alueet, joissa laskentaa tarkennetaan.

Adaptiivinen menetelmä voi olla tarkennusstrategiasta riippuen *lokaali* tai *globaali*. Lokaalissa menetelmässä integrointialue jaetaan aluksi osiin, joista jokaiselle määritellään osan koosta riippuva hyväksyttävä virhetaso. Laskentaa tarkennetaan kullakin osa-alueella niin kauan, että virhe saadaan hyväksymisrajaa pienemmäksi.

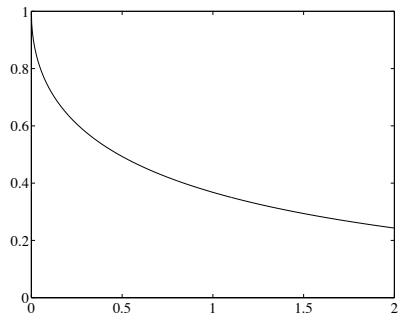
Globaalissa menetelmässä pyritään tarkentamaan kvadratuuria aina siellä, missä siitä on kulloinkin eniten hyötyä. Lokaalit menetelmät soveltuvat oivallisesti rinnakkaistietokoneille, kun taas globaalit menetelmät ovat yleensä parhaita numeerisia integrointialgoritmeja yksiprosessorisissa koneissa.

Esimerkki 5.8.2 Kokeilemme alkeellista adaptiivista algoritmia integraalin

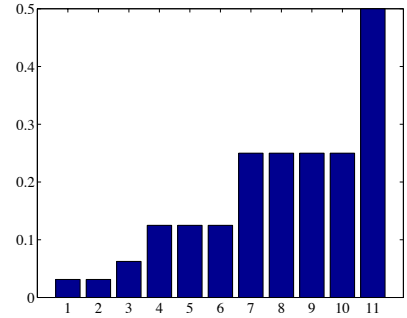
$$\int_0^2 e^{-\sqrt{x}}\,dx$$

arvon määrittämiseksi. Vaikka itse integrandi ei ole singulaarinen, sen derivaatta käyttäytyy origon lähellä kuten $1/\sqrt{(x)}$ (kuva 5.2a). On siis odotettavissa, että solmupisteitä kannattaa sijoittaa tiheämmin origon ympäristöön.

1. Valitsemme peruskvadratuuriksi puolisuunnikassäännön, ja alkutilanteeksi kahdesta osavälistä $[0,1]$ ja $[1,2]$ koostuvan integrointialueen.



a: $f(x) = e^{-\sqrt{x}}$.



b: *Osavälien pituudet.*

Kuva 5.2: *Adaptiivinen integrointivälien tihennys.*

2. Laskemme kaikilla osaväleillä tavallisen puolisuunnikassäännön antaman tuloksen Q_1.

3. Sovellamme kullakin osavälillä yhdistettyä puolisuunnikassääntöä Q_2, kun mukaan valitaan myös osavälin keskipiste.

4. Vertaamme osaväleillä laskettuja kvadratuureja ja haemme sen välin, jolla erotus $|Q_2 - Q_1|$ on suurin. Puolitamme seuraavaksi tämän välin, muilla väleillä lasketut tulokset talletetaan jatkoa varten.

5. Laskemme koko kvadratuurin arvon summaamalla osavälien tulokset. Virhearvion saamme käyttämällä integraalin tarkan arvon paikalla kahden viimeisimmän kvadratuurin avulla ekstrapoloitua tulosta. Jos virhearvio on riittävän pieni, lopetamme algoritmin suorittamisen. Päinvastaisessa tapauksessa palaamme kohtaan 2.

Esimerkissä käytämme hyvin karkeata mekanismia uuden solmupisteen paikan valinnassa, mutta periaatteessa samaan tulokseen päädyttäisiin kehittyneemmälläkin algoritmilla. Kuvassa 5.2b on esitetty solmupisteiden väliset etäisyydet järjestyksessä vasemmalta oikealle. Origoa lähimpänä olevat kuusi väliä ovat yhteispituudeltaan vain neljäsosan koko välin $[0, 2]$ pituudesta, ja loput viisi väliä kattavat sitten osuuden $[0.5, 2]$.

Adaptiivisten menetelmien heikko kohta on juuri virhearviointi. Jotta algoritmi olisi sovellettavissa mahdollisimman moniin tehtäviin, virheindikaattorin tulee selvitä singulariteeteistä tai voimakkaasti värähtelevistä funktioista ilman loputtomiin jatkuvaa osaväleihin jakamista. Toinen ongelma on tietysti päinvastainen tilanne: virhe arvioidaan huomattavasti todellista pienemmäksi, ja algoritmi päättyy, vaikka kvadratuuri on vielä kaukana integraalin oikeasta arvosta.

5.9 Moniulotteiset integraalit

Useammasta muuttujasta riippuvan funktion $f(x_1, x_2, ..., x_d)$ numeerinen integrointi on yleisessä tapauksessa selvästi vaikeampaa kuin yhden muuttujan funktion integrointi. Suurin syy tähän on integrointialueiden monimutkaistuminen. Yhdessä dimensiossa alue on aina reaalilukuakselin väli, mutta jo kahdessa dimensiossa alue voi tuoda mukanaan erilaisia pienempiä tai isompia sudenkuoppia: sen reunalla on mahdollisesti teräviä kärkiä tai reuna ei ole suoristuva, alue voi olla ei-konveksi tai sisältää reiän tai pahimmassa tapauksessa alue ei ole edes suunnistuva. Singulariteetitkaan eivät välttämättä ole enää pistemäisiä, vaan korkeampidimensioisia alimonistoja.

Toinen numeerista integrointia hankaloittava tekijä on tarvittavien solmupisteiden lukumäärän nopea kasvaminen dimensioiden lisääntyessä, kun pyritään parantamaan tarkkuutta. Jos yhden muuttujan suhteen pisteiden lukumäärä kaksinkertaistetaan, joudutaan pisteiden lukumäärä kertomaan tekijällä 2^d, kun integrointimuuttujia on d kappaletta. Tämän ilmiön kiertämiseksi alueen jakaminen pienempiin osiin on vähintään yhtä tärkeä menettelytapa kuin yksiulotteisten integraalien tapauksessa.

5.9.1 Yhden muuttujan kvadratuurien yleistyksiä

Yhden muuttujan kvadratuurin kertaluku on edellä määritelty suoraan korkeimman tarkasti integroituvan polynomin asteluvuksi. Useamman muuttujan tapauksessa luonnollinen yleistys on vaatimus kaikkien tietyn asteisten monomien tarkasta integroituvuudesta. Esimerkiksi x^3, x^2y, xy^2 ja y^3 ovat kahden muuttujan 3. asteen monomit. Interpolaatioajattelun sijasta moniulotteisia kvadratuureja lähdetäänkin usein muodostamaan asettamalla tavoitteeksi jokin kertaluku. Koska erilaisia astetta k olevia d muuttujan monomeja on $\binom{k+d-1}{d-1}$ kappaletta, tarvitaan moniulotteisissa kvadratuureissa aina useita uusia solmupisteitä, ennen kuin menetelmän kertaluku nousee. Tšakalov on osoittanut, että kaikilla (d, k)-pareilla on olemassa tehokas kertalukua k oleva kvadratuuri, jossa on korkeintaan $(k + d)!/(k!d!)$ solmupistettä.

Monille yksinkertaisille perusalueille (tapauksessa $d = 2$ neliö, kolmio, ympyrä; kun $d = 3$ vastaavasti kuutio, tetraedri ja pallo) on kehitetty lukuisia erikoiskaavoja, joilla on merkitystä esimerkiksi elementtimenetelmän kannalta. Jos integroitava alue on kuvattavissa *affiinilla muunnoksella* $A\mathbf{x} + \mathbf{b}$ jollekin perusalueelle, näitä valmiita kaavoja voi soveltaa tarkkuuden kärsimättä. Jos käytetään yleisempää muunnosta, muunnoksen Jacobin determinantti ei enää olekaan vakio eivätkä alkuperäisessä alueessa määritellyt polynomit säilytä astelukuaan. Numeerisen stabiiliuden kannalta on eduksi, jos kvadratuurin solmupisteet ovat integrointialueen sisällä ja painokertoimet positiivisia.

Esimerkki 5.9.1 Olkoon integrointialueena ensin (x, y)-tason nelikulmio P_1

(kuva 5.3a), joka saadaan (ξ, η)-tason yksikköneliöstä muunnoksella

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 3 & 1 \\ -1 & 2 \end{pmatrix} \begin{pmatrix} \xi \\ \eta \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Kuvauksen Jacobin determinantti on 7, joten voimme integroida yksikköneliössä seuraavasti:

$$\iint_{P_1} f(x, y)\, dx\, dy = \int_0^1 \int_0^1 7\hat{f}(\xi, \eta)\, d\xi\, d\eta.$$

Tässä on merkitty $\hat{f}(\xi, \eta) = f(x(\xi, \eta), y(\xi, \eta))$. Muodostamme yksinkertaisen kvadratuurin neliön kärkipisteiden avulla. Koska neliön pinta-ala on 1, painokertoimien summan täytyy myös olla 1, jotta vakiofunktiot integroituisivat tarkasti. Lisäksi symmetrian nojalla painokertoimet ovat yhtäsuuria, joten $w_i = 1/4$. Jos integroitava funktio on vaikkapa $f(x, y) = x$, tulee likiarvoksi

$$\iint_{P_1} x\, dx\, dy = \int_0^1 \int_0^1 7(3\xi + \eta + 1)\, d\xi\, d\eta \approx 7 \cdot \frac{1}{4}(1 + 4 + 5 + 2) = 21.$$

Termit $1, 4, 5$ ja 2 on saatu laskemalla lausekkeen $3\xi + \eta + 1$ arvo yksikköneliön kärkipisteissä. Vaikka käyttämämme kvadratuuri on hyvin alkeellinen, päädyimme täsmälleen oikeaan lopputulokseen, sillä menetelmä on tarkka lineaarisille polynomeille.

Sovellamme seuraavaksi samaa kvadratuuria nelikulmioon P_2 (kuva 5.3b). Nyt integrointialueen ja yksikköneliön välinen kuvaus on

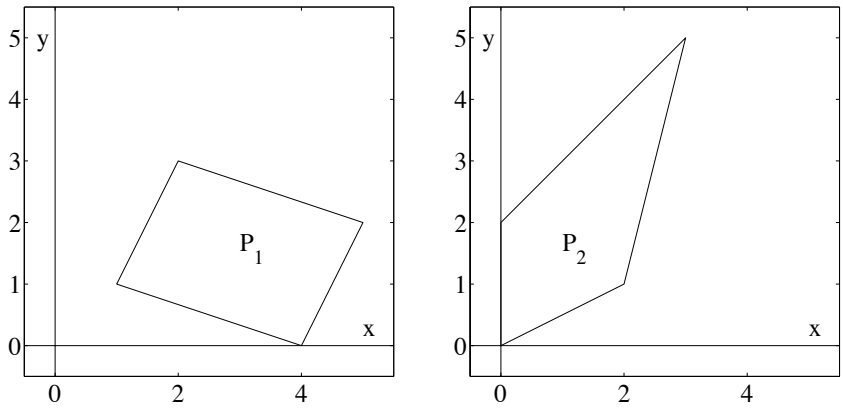
$$\begin{aligned} x &= 2\xi + \xi\eta \\ y &= \xi + 2\eta + 2\xi\eta, \end{aligned}$$

jolloin Jacobin determinantti ei enää olekaan vakio vaan $|J| = 3\xi + 2\eta + 4$ ja

$$\begin{aligned} \iint_{P_2} x\, dx\, dy &= \int_0^1 \int_0^1 (2\xi + \xi\eta)(3\xi + 2\eta + 4)\, d\xi\, d\eta \\ &\approx \frac{1}{4}(0 \cdot 4 + 2 \cdot 7 + 3 \cdot 9 + 0 \cdot 6) = 41/4 = 10.25. \end{aligned}$$

Oikea tulos on kuitenkin $53/8 = 6.625$. Virhe johtuu siitä, että muunnettu integrandi ei enää ole ensimmäisen asteen polynomi muuttujien ξ ja η suhteen, vaikka alkuperäinen integrandi $f(x, y) = x$ olikin lineaarinen.

Valmiita kvadratuureja löytyy matemaattisista taulukkokirjoista. Kaavoja on johdettu osittain heuristisesti, eikä niitä ole aina helppoa yleistää korkeampiin kertalukuihin. Uusia kvadratuureja voi yrittää muodostaa analogisesti yhden muuttujan kvadratuurien kanssa vaatimalla, että kaikki tietyn kertaluvun monomit integroituvat tarkasti — vertaa yhtälöön (5.1) — mutta tämä lähestymistapa johtaa joissain tapauksissa lineaarisesti riippuviin yhtälöihin. Yleensä kannattaa yrittää hyödyntää integrointialueen geometriasta riippuvat symmetriaominaisuudet lisäehtoina.



a: *Alue* P_1. **b:** *Alue* P_2.

Kuva 5.3: *Esimerkkiin 5.9.1 liittyvät integrointialueet.*

Esimerkki 5.9.2 Abramowitz ja Stegun luettelevat kirjassaan *Handbook of Mathematical Functions* useita geometrisiin peruskuvioihin liittyviä kvadratuureja. Esimerkiksi tasasivuiseen kolmioon T (kuva 5.4) liittyvän 4 pisteen menetelmän kvadratuuripisteet ja painot ovat

(x_i, y_i)	w_i
$(0,0)$	$9/12$
$(h,0)$	$1/12$
$(h/2, \pm\sqrt{3}h/2)$	$1/12$

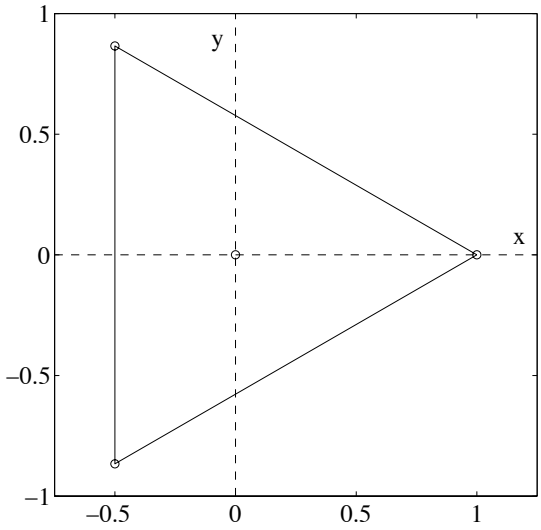
Koska koko kolmion pinta-ala on $3\sqrt{3}h^2/4$, ja kvadratuuripainojen summa on 1, niin lopullinen kvadratuurikaava on

$$\iint_T f(x,y)\,dx\,dy \approx \frac{3\sqrt{3}}{4}h^2 \sum_{i=1}^{n} w_i f(x_i, y_i).$$

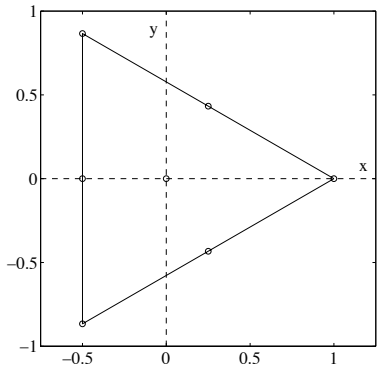
Kvadratuurin virhe on suuruusluokkaa $\mathcal{O}(h^3)$.

Jo yhden muuttujan funktioiden numeerisessa integroinnissa kvadratuuripisteiden optimaalisella sijoittelulla on dramaattinen vaikutus kaavojen tarkkuuteen. Moniulotteisissa avaruuksissa tilanne korostuu. Yksinkertaisen esimerkin tarjoavat kuvassa 5.5 esitetyt kaksi seitsemän pisteen integrointimenetelmää, joista ensimmäisen virhetermi on $\mathcal{O}(h^4)$ ja toisen $\mathcal{O}(h^6)$. Huomaa kummassakin menetelmässä kvadratuuripisteiden symmetrinen sijainti kolmion keskipisteen suhteen.

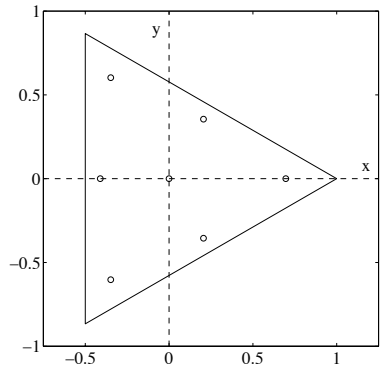
Jos moninkertainen integraali on palautettavissa *iteroiduksi integraaliksi* eli useamman toisistaan riippumattoman integraalin tuloksi, tämä kannattaa



Kuva 5.4: *Eräs tasasivuista kolmiota varten suunniteltu kvadratuuri ($h = 1$).*



a: *Virhe suurusluokkaa $\mathcal{O}(h^4)$.*



b: *Virhe suurusluokkaa $\mathcal{O}(h^6)$.*

Kuva 5.5: *Kaksi kolmiolle sopivaa saman työmäärän vaativaa kvadratuuria.*

ilman muuta tehdä ennen numeeristen kvadratuurien soveltamista. Palauttaminen voi vaatia integrointialueen jakamista osiin tai sopivia muuttujanvaihdoksia.

Esimerkki 5.9.3 Laskemme integraalin $\int_T f(x,y)\,dxdy$, kun alue T on paraabelin $y = x^2$ alle jäävä alue $T = \{(x,y) \mid 0 \le x \le 1,\ 0 \le y \le x^2\}$. Suoraan alueen määrittelyn nojalla pätee

$$\int_T f(x,y)\,dxdy = \int_0^1 \int_0^{x^2} f(x,y)\,dxdy.$$

Sama asia voidaan ilmaista myös toisin päin:

$$\int_T f(x,y)\,dxdy = \int_0^1 \int_0^{1-\sqrt{y}} f(x,y)\,dydx.$$

Huomaa, että integrointijärjestyksellä voi olla merkitystä tarkkuuteen. Ylä- tai alarajalle ilmestyvä neliöjuurilauseke vaatii enemmän työtä kuin yksinkertainen polynomi.

Käytännössä kaikki yhden muuttujan kvadratuurikaavat perustuvat jollain tavalla määritellyn interpolaatiofunktion integroimiseen (myös Gaussin kvadratuuri voidaan tulkita interpolaatiokvadratuuriksi, vaikkei sitä alunperin olekaan johdettu interpolaatiosta). Koska interpolointitehtävä itsessään muuttuu useamman muuttujan tapauksessa hyvin vaikeaksi, ei tämä lähtökohta enää tuota hyviä yleispäteviä kvadratuurisääntöjä usean muuttujan funktioille. Kappaleen lopussa esitellään toisenlaisia lähestymistapoja.

5.9.2 Tulokaavat

Jos integrointialue on yksinkertainen (tyypillisesti d-ulotteisen avaruuden särmiö), on mahdollista käyttää hyväksi yhden muuttujan kvadratuureja. Syntyvissä *tulokaavoissa* on ideana valita jokaiselle koordinaatille jonkin yksiulotteisen kvadratuurin solmupisteet. Painokertoimet saadaan kertomalla taustalla olevien kvadratuurien vastinpainot keskenään.

$$\begin{aligned}&\int_0^1 \cdots \int_0^1 f(x_1, x_2, \ldots, x_d)\,dx_1 dx_2 \ldots dx_d \\ &\quad \approx \sum_{i_1} \sum_{i_2} \cdots \sum_{i_d} w_{i_1} w_{i_2} \ldots w_{i_d} f(x_{i_1}, x_{i_2}, \ldots, x_{i_d}).\end{aligned}$$

Esimerkki 5.9.4 Valitsemme yhden muuttujan peruskvadratuuriksi puolisuunnikassäännön välillä $[0,1]$: $\int_0^1 f(x)\,dx \approx (f(1) + f(0))/2$. Siis solmupisteet ovat $x_1 = 0, x_2 = 1$ ja painot $w_1 = w_2 = 1/2$. Tästä pääsemme yksikköneliölle

soveltuvan tulokaavan solmupisteisiin muodostamalla kaikki mahdolliset x_1:n ja x_2:n arvojen kombinaatiot

$$(0,0),\ (1,0)\ (0,1)\ (1,1).$$

Painokertoimeksi tulee joka pisteen kohdalla $\frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$. Lopputulokseksi saimme siis esimerkissä 5.9.1 yksinkertaisen päättelyn ja symmetria-argumentin nojalla johdetun kvadratuurin.

Tulokaavoja voi toki soveltaa myös joihinkin mutkikkaampiin alueisiin, jos kvadratuuripisteet ja painot skaalataan sopivasti. Skaalaukseen pätee täsmälleen sama ohje kuin yhden muuttujan kvadratuureissa (sivu 155): painot kerrotaan sopivalla luvulla, jotta vakiofunktio integroituu tarkasti, ja kvadratuuripisteet jakavat välit aina samassa suhteessa.

Esimerkki 5.9.5 Tarkastelemme malliksi integraalia

$$\int_a^b \int_{g_1(x)}^{g_2(x)} f(x,y)\,dx\,dy,$$

johon sovellamme tulokaavan muodossa välille $[c_0, c_1]$ tarkoitettua N pisteen yksiulotteista menetelmää, jonka kvadratuuripisteet ovat $\{t_i\}$ ja -painot q_i. Ilman skaalausta tulokaava sopii siis neliössä $[c_0, c_1] \times [c_0, c_1]$ määriteltyjen funktioiden integrointiin (kuva 5.6a).

Lasketaan aluksi skaalattujen kvadratuuripisteiden x-koordinaatit:

$$x_i = \frac{b-a}{c_1 - c_0}(t_i - c_0) + a,\ i = 1, ..., N.$$

Kvadratuuripisteitä on tietysti N^2 kappaletta, joten kullakin koordinaatilla pitäisi olla periaatteessa kaksi alaindeksiä. Tässä esitellyssä tekniikassa kvadratuuripisteet osuvat kuitenkin y-akselin suuntaisille suorille, joten $x_{ij} = x_i,\ j = 1, ..., N$.

Seuraavaksi haetaan kvadratuuripisteiden y-koordinaatit skaalamalla jokaisen lasketun x-koordinaatin kohdalla väli $g_2(x_i) - g_1(x_i)$ vertailuvälille $[c_0, c_1]$:

$$y_{ij} = \frac{g_2(x_i) - g_1(x_i)}{c_1 - c_0}(t_j - c_0) + g_1(x_i), \quad i, j = 1, \ldots.$$

Lopuksi sovitetaan kvadratuuripainot vastaamaan uutta geometriaa:

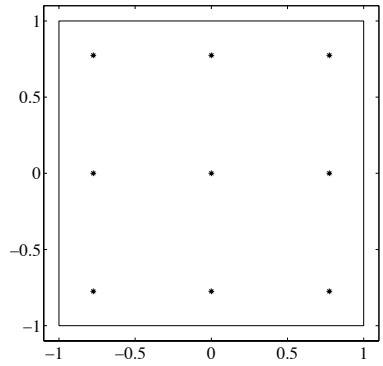
$$w_{ij} = \frac{b-a}{c_1 - c_0} \frac{g_2(x_i) - g_1(x_i)}{c_1 - c_0} q_i q_j.$$

Lopullinen kvadratuuri on

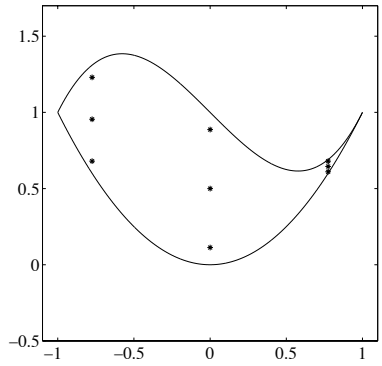
$$\int_a^b \int_{g_1(x)}^{g_2(x)} f(x,y)\,dx\,dy \approx \sum_{i,j} f\left(x_{ij}, y_{ij}\right) w_{ij}.$$

Eri muuttujien suhteen on tietysti mahdollista käyttää erilaisia kvadratuureja.

Kuvissa 5.6a ja 5.6b on esitetty tulokaavan mukainen kvadratuuri sekä perusneliössä $[-1,1]\times[-1,1]$ että kvadratuuripisteiden sijoittuminen erääseen integrointialueeseen (katso seuraavaa esimerkkiä). Tulokaavan käyttäminen näinkin yksinkertaiseen geometriaan on selvästi hyvin epätaloudellinen tapa laskea numeerisia integraaleja, koska pisteitä voi toisaalta keräytyä pienelle alueelle tiheään, kun taas muualla pisteet ovat harvemmassa. Menetelmän hyviä puolia ovat tietysti sen yleispätevyys ja helppo ohjelmoitavuus. Näissä piilee kuitenkin myös mahdolliset ongelmat: menetelmää tulee käytettyä mahdollisesti sellaisissa tilanteissa, joihin se ei sovi tai joissa pienellä harkinnalla selvitään huomattavasti pienemmällä työmäärällä.



a: *Kolmen pisteen Gaussin kvadratuurista muodostettu tulokaava.*



b: *Skaalatut kvadratuuripisteet.*

Kuva 5.6: *Tulokaavan skaalaus mielivaltaiselle alueelle.*

Esimerkki 5.9.6 Sovellamme edellistä tekniikkaa integraaliin

$$\int_{-1}^{1}\int_{x^2}^{x^3-x+1} \sin(x^2+y^2)\,dx\,dy,$$

kun yksiulotteisena perusmenetelmänä on kummankin koordinaatin suhteen 3 pisteen Gaussin kaava (taulukko sivulla 158), jonka kvadratuuripisteet ovat $t_1=-t_3=-\sqrt{0.6}$, $t_2=0$ ja painot $q_1=q_3=5/9$, $q_2=8/9$. Koska Gaussin ja Legendren kvadratuurit on määritelty alunperin juuri välille $[-1,1]$, x-akselin suunnassa ei tarvitse tehdä mitään skaalausta ja

$$x_1=x_{1j}=-\sqrt{0.6},\ x_2=x_{2j}=0,\ x_3=x_{3j}=\sqrt{0.6},\quad j=1,2,3.$$

Kohdassa $x=x_1=-\sqrt{0.6}$ integrointialueen yläraja on $(-\sqrt{0.6})^3-(-\sqrt{0.6})+1=0.4\sqrt{0.6}+1$ ja alaraja $(-\sqrt{0.6})^2=0.6$. Suoralla $x=-\sqrt{0.6}$ sijaitsevien kvadratuuripisteiden y-koordinaatit saamme siis kuvaamalla lineaarisesti välin $[-1,1]$ välille $[0.6,0.4\sqrt{0.6}+1]$ ja laskemalla edellä lueteltujen t_j-pisteiden

kuvapisteet:

$$\begin{aligned} y_{11} &= \frac{0.4\sqrt{0.6}+1-0.6}{2}(-\sqrt{0.6}+1)+0.6 = 0.68 \\ y_{12} &= \frac{0.4\sqrt{0.6}+1-0.6}{2}(0+1)+0.6 = 0.954919 \\ y_{13} &= \frac{0.4\sqrt{0.6}+1-0.6}{2}(\sqrt{0.6}+1)+0.6 = 1.229839. \end{aligned}$$

Aivan vastaavalla tavalla voimme muodostaa lineaarikuvaukset välin $[-1,1]$ kuvaamiseksi integrointialueen sisään jäävälle osille kohdissa $x = x_2$ ja $x = x_3$, joiden avulla pystymme laskemaan loputkin y-koordinaatit.

Painojen skaalaus on helppo suorittaa, kun kvadratuuripisteet on selvitetty (varsinkin kun x-akselin suhteen selvitään jälleen ilman erillistä skaalausta):

$$\begin{aligned} w_{11} &= \frac{0.4\sqrt{0.6}+1-0.6}{2}\frac{5}{9}\frac{5}{9} = 0.109543 \\ w_{12} &= \frac{0.4\sqrt{0.6}+1-0.6}{2}\frac{8}{9}\frac{5}{9} = 0.175269 \\ w_{13} &= w_{11} \\ w_{21} &= \ldots \end{aligned}$$

Kvadratuurin lopullinen arvo on

$$\sum_{i,j=1}^{3} \sin(x_{ij}^2 + y_{ij}^2) w_{ij} = 0.682889,$$

kun tarkka arvo 6 desimaalilla on 0.732840. Suhteellinen virhe on siis noin 7%. Jos tulokaavan pohjana käytetään 7 pisteen Gaussin kvadratuuria, päästään jo likiarvoon 0.732837.

Tulokaavat ovat käyttökelpoisia, kun integrointimuuttujien lukumäärä on suhteellisen pieni ($d \leq 4-5$). Jos muuttujia on paljon, solmupisteiden lukumäärän eksponentiaalinen kasvu tekee tulokaavoista hyvin tehottomia. Sen sijaan ne ovat kyllä tarkkoja ja numeerisesti stabiileja.

5.9.3 Monte Carlo -menetelmät

Kaikki edellä esitetyt numeeriset kvadratuurit perustuvat painotetun keskiarvon laskemiseen integroitavan funktion arvoista jollain kriteereillä valituissa solmupisteissä. *Monte Carlo -menetelmissä* solmupisteiden valinta tehdään satunnaisesti.

Yksinkertaisessa perustapauksessa tehtävänä on arvioida d-ulotteisen särmiön $P = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_d, b_d]$ yli laskettavaa integraalia

$$I = \int_{a_1}^{b_1}\int_{a_2}^{b_2}\cdots\int_{a_d}^{b_d} f(x_1, x_2, \ldots, x_d)\, dx_1 dx_2 \ldots dx_d.$$

Karkea likiarvo integraalille saadaan valitsemalla satunnaisesti N tasaisesti jakautunutta pistettä integrointialueesta P ja laskemalla funktion f arvojen summa näissä pisteissä sopivasti skaalattuna:

$$I \approx I_{MC} = \frac{\mathrm{Vol}(P)}{N} \sum_{i=1}^{N} f(x_i),$$

missä $\mathrm{Vol}(P)$ on alueen P tilavuus.

Monte Carlo -menetelmän antama likiarvo suppenee melko hitaasti kohti oikeaa tulosta. Koska kyseessä on satunnaisuuteen perustuva algoritmi, virheelle ei ole tarkkaa lauseketta, mutta virheen suurusluokka pystytään ennustamaan tilastollisin keinoin. Kun N on suuri, virheen keskihajonta on verrannollinen suureeseen $1/\sqrt{N}$. Tämä tulos on riippumaton integraalien lukumäärästä d.

Jos integroitavan funktion f käyttäytymisestä on jokin ennakkotieto, Monte Carlo -pisteiden valintaa voidaan painottaa esimerkiksi sellaisille alueille, joissa funktion arvojen tiedetään muuttuvan nopeasti. Tämänkaltaista *valikoivaa näytteenottoa* varten integraali I kirjoitetaan muotoon

$$I = \int_P f(x)\,dx = \int_P \frac{f(x)}{f_p(x)} f_p(x)\,dx.$$

Tässä $f_p(x)$ on todennäköisyysjakauman tiheysfunktio, jolle pätee

$$\int_P f_p(x)\,dx = 1.$$

Tiheysfunktio $f_p(x)$ määrää kuinka satunnaispisteiden x_i jakaumaa muutetaan. Integraalin I likiarvoksi saadaan

$$I \approx \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{f_p(x_i)}.$$

Kun Monte Carlo -menetelmiä sovelletaan johonkin monimutkaiseen alueeseen Q, turvaudutaan yleensä tekniikkaan, jossa Q upotetaan (mahdollisimman pieneen) ympäröivään d-ulotteiseen suoraan särmiöön P, ja funktion f sijasta integroidaankin laajennettua funktiota F:

$$F(x_1, x_2, \ldots, x_d) = \begin{cases} f(x_1, x_2, \ldots, x_d), & (x_1, x_2, \ldots, x_d) \in Q, \\ 0, & (x_1, x_2, \ldots, x_d) \in P \setminus Q. \end{cases}$$

Monte Carlo -menetelmien ongelmana on hidas suppeneminen (jopa valikoivaa näytteenottoa käytettäessä). Erityisen tehoton se on muihin menetelmiin verrattuna, kun integrandi $f(x)$ on sileä. Vaikka edellä on koko ajan puhuttu satunnaisluvuista, tietokoneissa on lähes aina turvauduttava pseudosatunnaislukuihin, joiden tuottaminen vaatii suurta huolellisuutta. Hyvilläkin satunnaislukugeneraattoreilla Monte Carlo -menetelmät vaativat suurta statistiikkaa eli laskuissa on käytettävä paljon pisteitä ja ne pitää mielellään toistaa useita kertoja. Käytännössä tämän tyyppisillä kvadratuureilla on järkevää olettaa, että tulokseen jää parhaimmillaankin prosentin suuruusluokkaa oleva virhe.

5.9.4 Hilamenetelmät

Hilamenetelmissä pyritään Monte Carlo -menetelmiä parempaan tarkkuuteen sijoittamalla kvadratuuripisteet jonkin säännön mukaan tasaisesti koko integrointialueeseen. Toisin kuin yksiulotteisista kvadratuureista rakennetuissa tulokaavoissa, pisteiden lukumäärän ei anneta kasvaa räjähdysmäisesti. Periaatteessa hilamenetelmät soveltuvat vain jaksollisten funktioiden integrointiin, mutta niistä on kehitelty joitain versioita myös yleisempiin tapauksiin. Lisäksi integrandi voidaan tarvittaessa muuttujanvaihdoksella saattaa jaksolliseksi, vaikka tästä aiheutuukin lisätyötä.

Historiallisesti hilamenetelmiä ovat ensimmäisinä kehitelleet toisaalta lukuteoreetikot ja toisaalta sirontaa tutkineet luonnontieteilijät. Yksinkertaisimmat hilamenetelmät tunnetaankin myös *lukuteoreettisten integrointimenetelmien* nimellä. Seuraavassa käydään lyhyesti läpi hilamenetelmien peruskäsitteitä.

Hilalla tarkoitetaan tässä yhteydessä avaruuden $\mathbb{R}^d$ diskreettiä osajoukkoa, joka on suljettu yhteen- ja vähennyslaskun suhteen. Hila näyttää siis samalta jokaisen hilapisteen kannalta. Koska hila on suljettu vähennyslaskun suhteen, se sisältää aina origon. Hilan virittävät kantavektorit $\{\mathbf{g}_1, \mathbf{g}_2, \cdots, \mathbf{g}_s\}$, $s \leq d$, ja jokainen hilapiste voidaan lausua kokonaislukukertoimisena lineaarikombinaationa kantavektoreista. Hilan erikoistapaus on *integrointihila* (integration lattice), joka sisältää kaikki d-ulotteisen avaruuden $\mathbb{R}^d$ kokonaislukupisteet $\mathbb{Z}^d$ alijoukkonaan.

Integrointia varten rajoitetaan tarkastelut d-ulotteiseen puoliavoimeen yksikkökuutioon $[0,1)^d$. Tällöin kuution kärjissä sijaitsevat hilapisteet eivät tule mukaan kvadratuuriin, lukuunottamatta origoa. Integrandi oletetaan yleensä kaikkien muuttujien suhteen 1-jaksolliseksi. Varsinainen kvadratuurisääntö on yhtä yksinkertainen kuin aiemmatkin: lasketaan keskiarvo funktion $f(\mathbf{x})$ arvoista hilapisteissä $\mathbf{x}_j$:

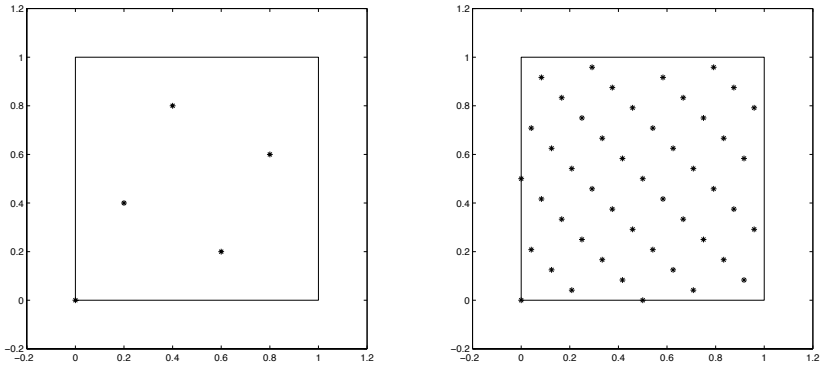
$$\int_{[0,1)^d} f(\mathbf{x})\, d\mathbf{x} \approx \frac{1}{N} \sum_{j=0}^{N-1} f(\mathbf{x}_j), \quad \mathbf{x}_j \in [0,1)^d.$$

Painokerroin on siis kaikille hilapisteille yhtä suuri eli $1/N$.

Integrointihilojen konstruointiin on kehitetty useita algoritmeja. Varhaisimmat esimerkit 1960-luvulta olivat jossain määrin heuristisia ”hyvien hilapisteiden menetelmiä” (method of good lattice points). Näissä pisteet lasketaan yksinkertaisesta kaavasta $\mathbf{x}_j = \left[\frac{j}{N}\mathbf{z}\right]$, $j = 0, \ldots, N-1$, missä $\mathbf{z}$ on sopivasti valittu vektori. Merkintä $[\cdot]$ tarkoittaa, että sulkujen sisällä olevasta lausekkeesta jätetään kokonaislukuosa pois.

Esimerkki 5.9.7 Valitsemme arvot $N = 5$ ja $\mathbf{z} = (1, 2)$. Tällöin integrointihilan pisteiksi muodostuu $(0, 0)$, $(1/5, 2/5)$, $(2/5, 4/5)$, $(3/5, 1/5)$ ja $(4/5, 3/5)$ (katso kuvaa 5.7a). Lukujen keskinäisen jaottomuuden vuoksi molemmat koordinaatit saavat kaikki mahdolliset erilaiset arvot, tässä tapauksessa viisi eri lukua. Tämä on eräs hilamenetelmien hyvä puoli. Jos integrandi sattuu olemaan

vakio jonkin muuttujan suhteen, sen integroimiseksi ei tule tehtyä ylimääräistä työtä toisin kuin tulokaavojen yhteydessä.



a: *Yksinkertainen integrointihila.* **b:** *Rangin 2 integrointihila.*

Kuva 5.7: *Esimerkkejä kaksiulotteisista integrointihiloista.*

Yleisemmin hilapisteet voidaan aina lausua muodossa

$$\left[\frac{j_1}{N_1}\mathbf{z}_1 + \cdots + \frac{j_r}{N_r}\mathbf{z}_r\right], \quad j_i = 0, \ldots, N_i - 1.$$

Kaavassa esiintyvien lineaarisesti riippumattomien vektoreiden lukumäärä r on hilan *rangi*. Edellisessä esimerkissä käsiteltiin hilaa, jonka rangi on 1. Kuvassa 5.7b on esimerkki rangin 2 integrointihilasta.

Hilamenetelmien virhearviot perustuvat Fourier'n analyysiin. Koska optimaalisen hilan valinta riippuu integroitavasta funktiosta, yleispäteviä hilamenetelmiä on vaikea konstruoida. Voidaan kuitenkin osoittaa, että jos $f(\mathbf{x})$ on riittävän sileä ja jaksollinen, on olemassa N pisteen integrointihila, jota käyttämällä virhe on suuruusluokkaa $\mathcal{O}\left((\log N)^{c(d,\alpha)}/N^{\alpha}\right)$. Tässä eksponentti c riippuu parametrista d, joka on avaruuden dimensio kuten ennenkin ja vakiosta $\alpha \geq 1$, joka mittaa funktion $f(\mathbf{x})$ jaksollisuutta. Hilamenetelmät ovat siis Monte Carlo -menetelmiä tehokkaampia — jos keksii oikean hilan!

5.9.5 Sagin ja Szekeresin menetelmä

Sag ja Szekeres ovat kehittäneet menetelmän, joka perustuu integrointialueen kuvaamiseen d-ulotteiselle yksikköpallolle siten, että sekä integroitavan funktion että sen derivaattojen arvot häviävät pallon pinnalla. Varsinaisena kvadratuurina käytetään puolisuunnikassääntöihin pohjautuvaa tulokaavaa. NAG-kirjastossa on Sagin ja Szekeresin menetelmään perustuva aliohjelma.

5.10 Ohjelmistokatsaus

Yleiskäyttöiset matemaattiset aliohjelmakirjastot (NAG, IMSL, HSL) sisältävät useita valmiita integrointirutiineja. Perehdy huolellisesti käsikirjoihin tai muuhun opasmateriaaliin ennen ohjelmien käyttöä, jotta osaat valita oikean aliohjelman ja tunnet sen mahdolliset rajoitukset. Täysin moitteettomasti kaikissa tilanteissa toimivaa integrointialiohjelmaa tuskin lienee olemassa, joten tuloksiin ei kannattaa koskaan luottaa täysin sokeasti. Pitkän kehitystyön tuloksena syntynyt NAG- tai IMSL-aliohjelma kuitenkin todennäköisesti laskee annetun integraalin nopeammin ja tarkemmin kuin itsekirjoitettu koodi. Omaan ohjelmointitaitoon on syytä turvautua sellaisissa tapauksissa, joissa ohjelman siirrettävyys tai muut syyt rajoittavat kirjastojen käyttöä.

Perusmenetelmä on yleensä Gaussin kvadratuurin avulla toteutettu globaalisti adaptiivinen integrointialgoritmi, jossa virhearvio tehdään lisäämällä osaväleille Kronrodin pisteet, ja adaptiivisuus hoidetaan jakamalla osavälit pienemmiksi. Tyypillinen yhdistelmä on 10 pisteen Gaussin kvadratuuri yhdistettynä 21 Kronrodin pisteeseen. Algoritmit yrittävät varoa mahdollisia singulariteetteja, ja käyttäjä voi joissain tapauksisssa jopa välittää aliohjelmalle etukäteen tiedon integrointialueessa sijaitsevista singulariteeteista. Lopputulosta tarkennetaan usein jollain ekstrapolaatiolla.

Gaussin ja Kronrodin kvadratuurin lisäksi suosittuja menetelmiä yksiulotteisten integraalien käsittelyyn ovat Pattersonin menetelmä ja Clenshawn ja Curtisin kvadratuuri erilaisine muunnoksineen. Erikoistapauksille (ääretön integrointiväli, värähtelevät integraalit, Hilbertin muunnos, singulaariset integraalit jne.) on omat aliohjelmansa.

Moniulotteisten integraalien laskemista varten tarjotaan Gaussin tulokaavoja, adaptiivista Monte Carlo -menetelmää sekä Korobovin ja Conroyn mukaan nimettyä hilamenetelmää. NAG-kirjastossa on myös oma rutiininsa integroinnille d-ulotteisen pallon yli.

Aliohjelmakirjastoissa on niin ikään valmiit aliohjelmat erilaisten Gaussin kvadratuurien solmupisteiden ja painojen laskemista varten.

Matemaattinen yleistyökalu Mathematica on erittäin laaja ohjelmistokokonaisuus, johon sisältyy myös numeerista integrointia suorittava komento `NIntegrate`. Tälle voi antaa parametriksi tarkkuusvaatimuksen. Jos siis jostain syystä tarvitset hyvin suurta tarkkuutta, Mathematica on eräs mahdollinen työskentely-ympäristö. Komentoon `NIntegrate` on ohjelmoitu useita kvadratuureja, joista voit valita tehtävään sopivimman.

5.11 Lisätietoja

Useimmissa numeerisen analyysin oppikirjoissa esitellään ainakin yksinkertaisimmat kvadratuurit. Kirjassa *Practical Numerical Analysis* [Eva95] on huomattavan laaja katsaus erilaisiin menetelmiin. Pelkkään numeeriseen integrointiin erikoistuneista teoksista kannattaa muistaa alan klassikko *Met-*

hods of Numerical Integration [DR84] (josta on ilmestynyt ainakin kaksi painosta) ja tuoreempi perusteos *Computational Integration* [KÜ98]. Sekä analyyttisiä että numeerisia integrointitekniikoita käydään perusteellisesti läpi kirjassa *Handbook of Integration* [Zwi92].

6 Epälineaariset yhtälöt

Kerroimme luvussa 3 lineaaristen yhtälöryhmien ratkaisemisesta. Käytännössä joudumme kuitenkin ratkaisemaan sekä epälineaarisia yhtälöitä että yhtälöryhmiä. Tämä luku esittelee epälineaaristen tehtävien ratkaisumenetelmiä.

6.1 Epälineaariset yhtälöt ja yhtälöryhmät

Yhtälö $e^x - x^2 + x = 0$ on esimerkki epälineaarisesta tehtävästä, jolle ei löydy ratkaisua suljetussa muodossa. Voimme ratkaista tehtävän numeerisilla menetelmillä, jolloin saamme likiarvon $x^* \approx -0.4441$.

Kun yhtälöitä on enemmän kuin yksi, puhumme *yhtälöryhmästä*. Seuraavassa on esimerkki kahden epälineaarisen yhtälön ryhmästä:

$$\begin{cases} f_1(\mathbf{x}) &= 100\,x_1x_2 - 1 = 0, \\ f_2(\mathbf{x}) &= e^{-x_1} + e^{-x_2} - 1.01 = 0. \end{cases} \tag{6.1}$$

Funktioiden f_1 ja f_2 käyttäytymistä on havainnollistettu kuvassa 6.1. Tämän yhtälöryhmän ratkaisu on $\mathbf{x}^* \approx (4.40061, 0.00227)^T$.

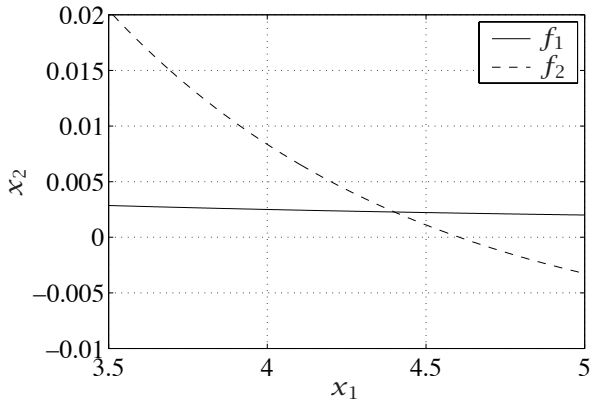
Epälineaarisella yhtälöryhmällä tarkoitetaan siis nollakohdan etsimistä vektoriarvoiselle funktiolle $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$:

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}\,. \tag{6.2}$$

Tämä on aukaistuna

$$\begin{cases} f_1(x_1, \ldots, x_n) &= 0, \\ f_2(x_1, \ldots, x_n) &= 0, \\ \qquad\qquad\vdots & \\ f_m(x_1, \ldots, x_n) &= 0. \end{cases} \tag{6.3}$$

Jatkossa oletamme, että funktiot ovat jatkuvasti derivoituvia. Kyseessä on lineaarinen yhtälöryhmä (katso lukua 3), jos $\mathbf{f}(\mathbf{x}) = A\mathbf{x} - \mathbf{b}$, missä A on $m \times n$ -matriisi ja $\mathbf{b}$ on n alkion vektori. Jos skalaarifunktio $f(x)$ on polynomi, kutsutaan tehtävää *algebralliseksi* eli *polynomiyhtälöksi*.



Kuva 6.1: *Funktioiden nollakäyrät* $f_1(\mathbf{x}) = 0$ *ja* $f_2(\mathbf{x}) = 0$.

Esimerkki 6.1.1 Valon taittumiseen kahden väliaineen rajapinnassa pätee *Snellin laki*:

$$n_2 \sin\theta_2 = n_1 \sin\theta_1.$$

Tässä n_1 ja n_2 ovat väliaineiden taitekertoimet ja kulmat θ_1 ja θ_2 ovat valonsäteen ja pinnan normaalin välisiä kulmia. Kun tiedämme valon tulokulman θ_1 saamme ratkaistua taittuneen valonsäteen suunnan θ_2:

$$\sin\theta_2 = \frac{n_1}{n_2}\sin\theta_1.$$

Tämä yhtälö on ratkaistavissa käyttämällä arcsin-funktiota tai vaihtoehtoisesti sopivaa epälineaarisen yhtälön ratkaisumenetelmää.

Esimerkki 6.1.2 Voimme esittää tavallisen differentiaaliyhtälön reuna-arvotehtävän

$$x'(t) = \lambda x(t) + c, \quad x(0) = x_0, \quad x(1) = x_1,$$

ratkaisun muodossa

$$x(t) = x_0 e^{\lambda t} + \frac{c}{\lambda}\left(e^{\lambda t} - 1\right).$$

Kun tiedämme alkutilan $x(0) = x_0$, lopputilan $x(1) = x_1$ sekä vakion c arvon, voimme ratkaista tuntemattoman kertoimen λ epälineaarisesta yhtälöstä

$$x_1 = x_0 e^{\lambda} + \frac{c}{\lambda}\left(e^{\lambda} - 1\right). \tag{6.4}$$

Ratkaisemme tämän yhtälön esimerkeissä 6.3.1 ja 6.3.3.

Esimerkki 6.1.3 Astrofysiikan eksentrinen anomalia E toteuttaa Keplerin yhtälön

$$E - e \sin E = M,$$

missä vakio e on eksentrisyys ja vakio M on keskianomalia.

Esimerkki 6.1.4 Seuraavassa tutkimme *Bratun ongelmaa*, joka esiintyy mm. yksinkertaisissa epälineaarisen diffuusion malleissa. Olkoon Ω tarkasteltava alue ja $\partial\Omega$ sen reuna. Tehtävänä on määrätä funktio $u(x, y)$, joka toteuttaa alueessa Ω yhtälön

$$-\nabla^2 u = \lambda e^u.$$

Funktio $u(x, y)$ häviää reunalla: $u(x, y) = 0$, kun $(x, y) \in \partial\Omega$.

Voimme hakea funktiolle u likimääräisratkaisua differenssikaavojen avulla. Kappaleessa 3.8 sivulla 48 on kerrottu Poissonin yhtälön

$$-\nabla^2 u = g(x, y), \quad x \in [0, 1] \times [0, 1]$$

ratkaisemisesta differenssimenetelmällä. Voimme ratkoa Bratun ongelmaa samaan tapaan. Diskretoinnin tuloksena saamme tehtävän

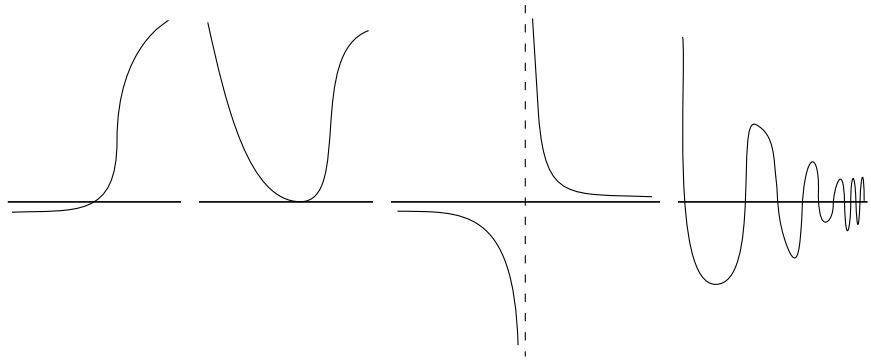
$$A\mathbf{u} = \mathbf{f}(\mathbf{u}).$$

Oikean puolen funktion $\mathbf{f}$ komponentit ovat $f_i = \lambda e^{u_i}$. Matriisin A rakenne on esitetty yhtälössä (3.13) sivulla 49.

6.2 Mahdollisia ongelmatilanteita nollakohdan löytämisessä

Tarkastelemme kuvassa 6.2 esitettyjen skalaarifunktioiden $f(x)$, $x \in \mathbb{R}$ käyttäytymistä mahdollisen nollakohdan läheisyydessä. Jos funktio on jatkuva välillä $[a, b]$ ja arvot $f(a)$ ja $f(b)$ ovat erimerkkiset, löytyy väliltä ainakin yksi nollakohta. Toisaalta nollakohta voi löytyä, vaikka funktio ei vaihtaisi merkkiä. Esimerkkinä olkoon funktio $f(x) = (x - a)^2$, joka on ei-negatiivinen kaikilla $x \in \mathbb{R}$. Tällaisessa tapauksessa nollakohta voi jäädä löytämättä jo pelkästään liukulukuesitysmuodon aiheuttamien pyöristysvirheiden ansiosta.

Eräs mahdollisuus nollakohtien löytymättä jäämiseen (tai "ylimääräisten" löytymiseen) ovat singulariteetit. Singulaarinen funktio $1/(x - a)$ vaihtaa merkkiä arvon $x = a$ ympäristössä, vaikka nollakohtaa ei ole. Silti liukulukuesitysmuodossa väliltä voi hyvinkin näyttää löytyvän juuri. Toisaalta funktiolla $\sin(1/x)$ on äärettömän monta nollakohtaa epäjatkuvuuskohdan $x = 0$ ympäristössä.



Kuva 6.2: *Erilaisia mahdollisuuksia skalaarifunktion käyttäytymiselle. Jatkuvalla ja välin päätepisteissä erimerkkisellä funktiolla on ainakin yksi nollakohta. Toisaalta funktion ei tarvitse vaihtaa merkkiä nollakohdan ympäristössä. Lisäksi singulariteetit voivat aiheuttaa ongelmia numeerisille algoritmeille.*

Myös moninkertaiset juuret voivat aiheuttaa ongelmia ratkaisumenetelmälle. Jos piste x^* on yhtälön $f(x) = 0$ juuri, voimme esittää funktion muodossa $f(x) = (x - x^*)^m g(x)$. Jos $m = 1$, on kyseessä yksinkertainen juuri, muuten moninkertainen. Periaatteessa potenssin m ei tarvitse olla kokonaisluku. Tästä on esimerkkinä funktio $f(x) = x\sqrt{x-1} = x(x-1)^{1/2}$.

Seuraavassa on muutamia esimerkkejä epälineaarisen yhtälön ratkaisujen lukumäärälle reaalilukujoukossa:

$e^x + 5 = 0$	ei ratkaisua
$e^{-x} - x = 0$	yksi ratkaisu
$x^2 - 4\sin x = 0$	kaksi ratkaisua
$x^3 + 2x^2 - 5x - 6 = 0$	kolme ratkaisua
$\cos x = 0$	äärettömän monta ratkaisua.

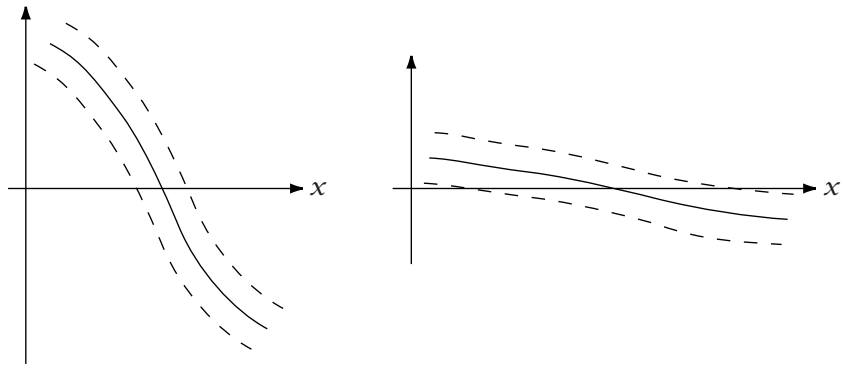
Oman ongelmakenttänsä aiheuttavat huonosti skaalatut tehtävät: ratkaisu voi löytyä esimerkiksi väliltä $x \in [1, 1 + 10^{-100}]$. Liukulukujärjestelmässä ongelma voi olla mahdoton ratkaista tällaisena. Esimerkiksi yhtälöryhmän (6.1) ratkaisussa on funktioilla f_1 ja f_2 eri skaalat.

Yhtälöryhmän $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ likimääräisen ratkaisun $\mathbf{x}_A$ hyvyyttä voi arvioida mm. seuraavilla tavoilla:

$$\|\mathbf{x}_A - \mathbf{x}^*\| \approx 0 \quad \text{tai} \quad \|\mathbf{f}(\mathbf{x}_A)\| \approx 0.$$

Tässä $\mathbf{x}^*$ on tehtävän tarkka ratkaisu. Nämä kriteerit eivät välttämättä toteudu yhtä aikaa. Kuvassa 6.3 on havainnollistettu eri mahdollisuuksia.

Tyypillinen virhe ratkaisumenetelmän käytössä on liian suuri tarkkuusvaatimus. Jos funktion arvot vaihtelevat nopeasti nollakohdan läheisyydessä, on residuaalin $\|\mathbf{f}(\mathbf{x}_A)\|$ arvo usein yllättävänkin suuri, vaikka nollakohta olisi paikallistettu melko tarkasti. Toisaalta jos funktion arvot ovat pieniä nollakohdan läheisyydessä, voi nollakohdan likiarvo olla melko kaukana todellisesta nollakohdasta, vaikka funktion arvot olisivatkin lähellä nollaa.



Kuva 6.3: *Funktion häiriöalttius etsittäessä nollakohtaa.*

6.3 Yhden yhtälön ratkaiseminen

Epälineaaristen yhtälöiden ja yhtälöryhmien ratkaisumenetelmien täytyy kyetä tekemään varsin monimutkaisia päätöksiä. Yleismenetelmää epälineaaristen yhtälöiden ratkaisemiseen ei ole olemassa. Mahdollisia algoritmeja ja ohjelmistoja on monia ja yleensä kannattaa kokeilla eri menetelmiä parhaan löytämiseksi.

6.3.1 Suosituksia menetelmän valintaan

Seuraavassa esitämme perusreseptin epälineaarisen yhtälön ratkaisemiseen:

1. Hae rajat alueelle, mistä ratkaisua etsitään.
2. Keksi hyvä alkuarvaus (tai useampia, riippuen menetelmästä) iteraatiolle. Ongelman teoreettisesta tai fysikaalisesta tuntemisesta on usein paljon apua. Mitä parempi alkuarvaus on tehtävissä, sitä luotettavammin ja tietysti nopeammin menetelmät toimivat.
3. Jos ratkaistavana on yksi epälineaarinen yhtälö, seuraavat menetelmät ovat käyttökelpoisia:
   - Jos epälineaarisen funktion $f(x)$ derivaatta ei ole saatavilla, Brentin menetelmä on monipuolinen. Algoritmi löytyy mm. IMSL-aliohjelmakirjastosta.
   - Polynomiyhtälöille (ja kun halutaan ratkaista kompleksisia juuria) käyttökelpoisia ja yksinkertaisia algoritmeja ovat Müllerin ja Laguerren menetelmät.
   - Kun derivaatat ovat helposti laskettavissa, Newtonin menetelmä yhdistettynä sopivaan globaaliin strategiaan on varsin toimiva. Hyvä alkuarvaus auttaa!

4. Kun (likimääräinen) ratkaisuehdokas on löytynyt, varmista, että se todella on ratkaisu. Likimääräistä ratkaisua voi tarkentaa käynnistämällä iteraation uudestaan sopivalla menetelmällä (esimerkiksi Newtonin menetelmällä).

Kaupallisissa ohjelmistoissa käytetyt menetelmät ovat ns. monialgoritmeja, joissa on yhdistetty varma (mutta mahdollisesti hidas) menetelmä nopeaan menetelmään, jota käytetään lähestyttäessä ratkaisua.

Seuraavassa esittelemme ensiksi yksinkertaisen puolitushakualgoritmin ja tämän jälkeen tehokkaan Newtonin menetelmän. Lisäksi kerromme yhdistelmämenetelmistä, joissa on pyritty sekä luotettavuuteen että tehokkuuteen. Lopuksi esittelemme kiintopistemenetelmiä, joista voi olla apua erikoistilanteissa.

6.3.2 Puolitushaku

Yksinkertainen ja luotettava yksiulotteisiin tapauksiin soveltuva mutta varsin tehoton ratkaisualgoritmi on puolitushaku (bisection):

1. Etsi pisteet $a_1, b_1 : a_1 < b_1$ siten, että arvot $f(a_1)$ ja $f(b_1)$ ovat erimerkkiset. Funktion f jatkuvuuden perusteella väliltä $[a_1, b_1]$ löytyy nollakohta.

2. Laske funktion arvo välin $[a_k, b_k]$ puolivälissä $m = (a_k + b_k)/2$. Jos arvot $f(m)$ ja $f(a_k)$ ovat erimerkkiset, tutkimme seuraavaksi väliä $[a_k, m]$, muuten väliä $[m, b_k]$.

3. Lopeta iterointi, kun saavutat riittävän tarkkuuden.

Puolitusmenetelmä suppenee hitaasti verrattuna kehittyneempiin menetelmiin, eikä sitä voi käyttää useampiulotteisiin tapauksiin. Puolitusmenetelmää käytetään kuitenkin usein tehokkaamman menetelmän varmistuksena luotettavuutensa vuoksi.

Esimerkki 6.3.1 Ratkaisemme esimerkin 6.1.2 yhtälön (6.4) käyttäen puolitushakua. Tätä varten muutamme yhtälön muotoon

$$f(\lambda) = x_0 e^{\lambda} + \frac{c}{\lambda}\left(e^{\lambda} - 1\right) - x_1 = 0.$$

Tehtävänä on hakea funktion $f(\lambda)$ nollakohta. Annamme vakioille seuraavat arvot: $x_0 = 0$, $x_1 = 10$ ja $c = 1$. Koska $f(3) < 0$ ja $f(4) > 0$ ja funktio f on jatkuva, löytyy väliltä $[3, 4]$ nollakohta.

Kun teemme 10 puolitushaun iteraatiota, saamme seuraavat tulokset:

k	m	$f(m)$
1	3.5000	−0.8242
2	3.7500	1.0723
3	3.6250	0.0758
4	3.5625	−0.3856
5	3.5938	−0.1579
6	3.6094	−0.0418
7	3.6172	0.0168
8	3.6133	−0.0125
9	3.6152	0.0021
10	3.6143	−0.0052

Siten löysimme nollakohdalle arvion $\lambda^* \approx 3.61$.

6.3.3 Newtonin menetelmä

Newtonin menetelmä (myös Newtonin ja Raphsonin menetelmä) on käytetyimpiä epälineaaristen yhtälöiden ratkaisumenetelmiä. Menetelmä käyttää hyväkseen funktion $f(x)$ derivaattaa pisteessä x. Newtonin menetelmä voidaan johtaa funktion $f(x)$ Taylorin kehitelmästä

$$f(x+\delta) \approx f(x) + f'(x)\delta + \frac{f''(x)}{2}\delta^2 + \dots . \tag{6.5}$$

Saamme ehdosta $f(x+\delta) = 0$ pienillä muuttujan x muutoksilla hakuaskeleen $\delta = -f(x)/f'(x)$. Tällöin olemme johtaneet seuraavan algoritmin:

Algoritmi 6.3.1 (Newtonin menetelmä)

hae nollakohdan x^* sijainnille rajat: $x^* \in [a, b]$
$x_1 = (a+b)/2$
$k = 1$
repeat
  $d = \dfrac{f(x_k)}{f'(x_k)}$
  $x_{k+1} = x_k - d$
  if $x_{k+1} \notin [a, b]$
    virhe: hypättiin ulos hakuvälistä
    exit
  end
  $k = k + 1$
until $|d| < \epsilon$

Newtonin menetelmä ei toimi, jos pätee $f'(x_k) = 0$ jossakin iteraatiopisteessä. Newtonin menetelmä suppenee parhaimmillaan neliöllisesti nollakohdan lähellä. Jos nollakohta on moninkertainen, suppeneminen on vain lineaarista. Jos pätee $f'(x^*) = 0$, menetelmä ei toimi tehokkaasti.

Esimerkki 6.3.2 Newtonin menetelmää voi käyttää neliöjuuren tai kuutiojuuren laskemiseen. Kokeilemme laskea arvon $x = \sqrt[3]{5}$ eli ratkaisemme funktion $f(x) = x^3 - 5$ nollakohdan. Saamme derivaataksi $f'(x) = 3x^2$ ja iteraatioksi

$$x_{k+1} = x_k - \frac{x_k^3 - 5}{3x_k^2}.$$

Alkuarvauksella $x_1 = 1$ saamme iteraatiopisteet

k	x_k	$f(x_k)$
1	1.0000	−4.0000
2	2.3333	7.7037
3	1.8617	1.4523
4	1.7220	0.1062
5	1.7101	0.0007
6	1.7100	0.0000

Siten saamme arvion $\sqrt[3]{5} \approx 1.7100$.

Esimerkki 6.3.3 Ratkaisemme esimerkin 6.3.1 tehtävän $f(\lambda) = 0$ käyttäen Newtonin menetelmää. Jos alkuarvauksena on $\lambda = 3.5$, saamme tulokseksi seuraavat iteraatiopisteet:

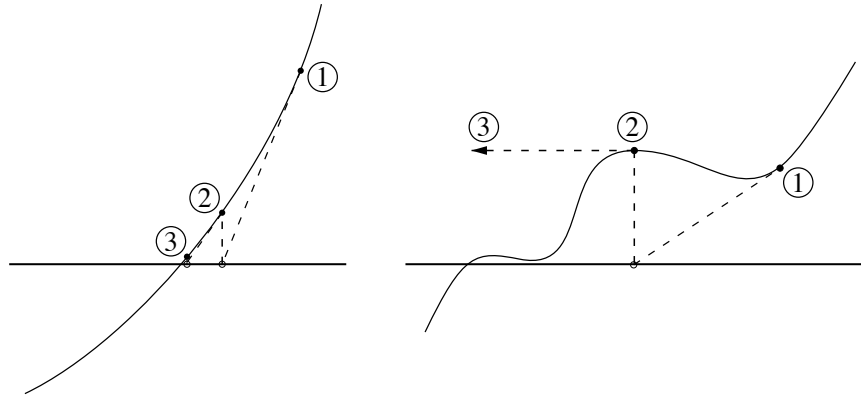
k	λ_k	$f(\lambda_k)$
1	3.5000	−0.8242
2	3.6205	0.0417
3	3.6150	0.0001
4	3.6150	0.0000

Siten Newtonin menetelmä suppeni neljässä iteraatiossa ja saimme nollakohdalle arvion $\lambda^* \approx 3.6150$.

Newtonin menetelmä soveltuu myös kompleksiluvuilla laskemiseen, mutta tällöin iteraatio pitää käynnistää alkuarvauksella, joka ei ole reaaliakselilla.

Newtonin menetelmä tarvitsee arvion funktion $f(x)$ derivaatalle pisteessä x_k. Derivaatan analyyttistä lauseketta kannattaa käyttää, jos se tunnetaan. Derivaatan likiarvot voidaan laskea funktion $f(x)$ arvoista differenssiarviolla, esimerkiksi $f'(x_k) \approx (f(x_k + h) - f(x_k))/h$. Toisaalta kvadraattinen interpolaatiomenetelmä (kappale 6.3.4) voi olla tehokkaampi kuin differenssiarviota käyttävä Newtonin menetelmä.

Kuvassa 6.4 on kuvattu Newtonin menetelmän käyttäytymistä kahdessa eri tapauksessa. Ensimmäisessä nollakohta löytyy muutamalla iteraatiolla, kun taas toisessa tapauksessa törmätään derivaatan nollakohtaan ja joudutaan kauas ratkaisusta.



Kuva 6.4: *Newtonin menetelmän toiminta kahdessa eri tapauksessa. Vasemmalla algoritmi suppenee hyvin nopeasti. Oikealla törmätään derivaatan nollakohtaan ja joudutaan kauas funktion nollakohdasta.*

6.3.4 Yhdistelmämenetelmät

Voimme arvioida funktiota $f(x)$ laskemalla funktion arvon $m+1$:ssä iteraatiopisteessä ja sovittamalla pisteisiin m-asteisen polynomin. Voimme valita tämän polynomin nollakohdan seuraavaksi iteraatiopisteeksi.

Kvadraattisen polynomin sovitukseen ($m = 2$) käytetään pisteitä x_a, x_b ja x_c sekä funktion arvoja $f(x_a)$, $f(x_b)$ ja $f(x_c)$. Ongelmana sovituksessa on se, että polynomilla ei välttämättä ole reaaliarvoisia nollakohtia. Toisaalta tämä *Müllerin menetelmä* soveltuu hyvin kompleksiarvoisten yhtälöiden ratkaisemiseen.

Toinen lähestymistapa on arvioida funktion $f(x)$ käänteisfunktiota $x = f^{-1}(y)$ nollakohdan lähellä sopivalla polynomilla h. Voimme laskea tälle polynomille arvon argumentin arvolla nolla (*käänteinen interpolointi*), jolloin saamme arvion nollakohdan sijainnille. Tämän algoritmin etuna on parempi stabiilisuus.

Käänteisessä kvadraattisessa interpoloinnissa sovitetaan arvoihin x_a, x_b ja x_c kvadraattinen polynomi arvojen $f(x_a)$, $f(x_b)$ ja $f(x_c)$ funktiona. Tälle polynomille lasketaan arvo argumentilla 0, jolloin saadaan seuraava iteraatiopiste.

Tarvittavat kaavat ovat johdettavissa Lagrangen interpoloinnin avulla (kappale 4.8 sivulla 100). Merkitsemme aluksi

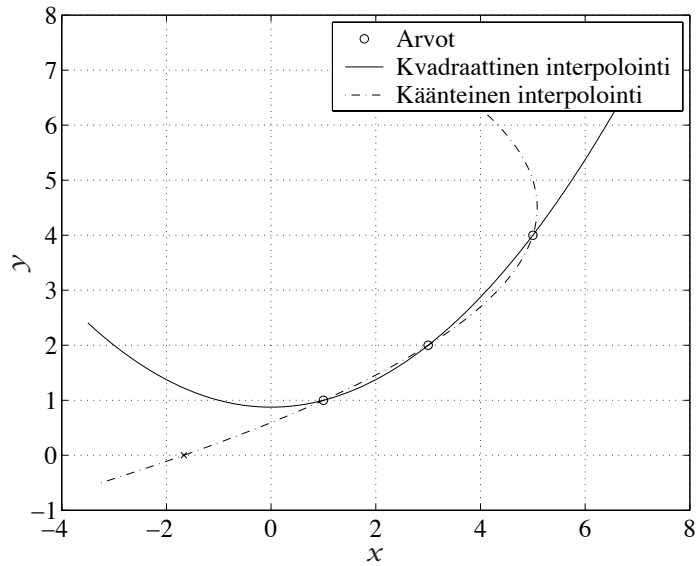
$$u = f(x_b)/f(x_c), \quad v = f(x_b)/f(x_a), \quad w = f(x_a)/f(x_c).$$

Tämän jälkeen laskemme arvot

$$p = v(w(u-w)(x_c - x_b) - (1-u)(x_b - x_a)),$$
$$q = (u-1)(v-1)(w-1).$$

Seuraava iteraatiopiste on nyt $x_+ = b + p/q$. Voimme toistaa iteraation korvaamalla pisteen b pisteellä x_+, pisteen a pisteellä b ja pisteen c pisteellä a. Kullakin iteraatiolla joudumme laskemaan funktion f arvon vain kerran.

Kuva 6.5 havainnollistaa kvadraattista ja käänteistä kvadraattista interpolointia. Tässä tapauksessa kvadraattisella interpoloinnilla saatu funktio ei leikkaa x-akselia, mutta käänteinen interpolointi tuottaa uuden iteraatiopisteen.



Kuva 6.5: *Kvadraattinen ja käänteinen kvadraattinen interpolointi.*

Käänteisen kvadraattisen interpoloinnin suppenemisnopeus lähellä nollakohtaa on $p \approx 1.839$ eli lineaarista parempi mutta kvadraattista suppenemista huonompi. Tehokkaaksi ja luotettavaksi osoittautuneessa *Brentin menetelmässä* käytetään käänteistä kvadraattista interpolaatiota yhdistettynä puolitushakuun, jolla varmistetaan suppeneminen.

6.3.5 Kiintopistemenetelmät

Olkoon funktio $g(x)$ sellainen kuvaus $\mathbb{R} \to \mathbb{R}$, jolle pätee pisteessä x

$$x = g(x).$$

Tällöin piste x on kuvauksen $g(x)$ *kiintopiste*, sillä x ei muutu kuvauksessa. Monet epälineaaristen tehtävien ratkaisualgoritmit ovat ns. kiintopisteiteraatioita, sillä algoritmit ovat muotoa

$$x_{k+1} = g(x_k),$$

missä g on sopivasti valittu funktio.

Epälineaarisen yhtälön $f(x) = 0$ ratkaisemiseen voi johtaa kiintopistemenetelmiä esimerkiksi huomaamalla, että $f(x) = 0$ on sama kuin $x + \alpha f(x) = x$. Merkitsemme siis $g(x) = x + \alpha f(x)$ ja saamme näin iteraation

$$x_{k+1} = g(x_k) = x_k + \alpha f(x_k)\,. \tag{6.6}$$

Muitakin tapoja johtaa kiintopisteiteraatioita toki löytyy. Esimerkiksi Newtonin menetelmä voidaan tulkita kiintopisteiteraatioksi.

Esimerkki 6.3.4 Yhtälön $f(x) = x^2 + x - 3 = 0$ ratkaisemiseksi voimme johtaa seuraavat kiintopisteiteraatiot:

1. $g(x) = -x^2 + 3$ (saadaan ratkaisemalla x)
2. $g(x) = \sqrt{3 - x}$ (ratkaisemalla x^2)
3. $g(x) = 3/x - 1$ (jakamalla yhtälö $x^2 = 3 - x$ muuttujalla x)
4. $g(x) = \dfrac{x^2 + 3}{2x + 1}$ (Newtonin menetelmä).

Kuvassa 6.6 on havainnollistettu iteraatioiden toimintaa alkuarvauksella $x = 1.2$. Tässä tapauksessa iteraatiot 1 ja 3 eivät suppene ja iteraatiot 2 ja 4 suppenevat.

Kuten kuvasta 6.6 näkyy, kiintopisteiteraatiot voivat käyttäytyä hyvin eri tavoin. Toiset iteraatiot voivat hajaantua ja toiset supeta hitaasti tai jopa hyvin nopeasti. Jos oletamme, että funktio g on sileä ja pätee $x^* = g(x^*)$, voimme johtaa seuraavan ehdon iteraation suppenemiselle kiintopisteen x^* läheisyydessä:

$$|g'(x^*)| < 1.$$

Voimme myös johtaa arvion kiintopisteiteraation suppenemisnopeudelle. Olkoon e_{k+1} virhe k:nnellä iteraatiolla:

$$e_{k+1} = x_{k+1} - x^* = g(x_k) - g(x^*).$$

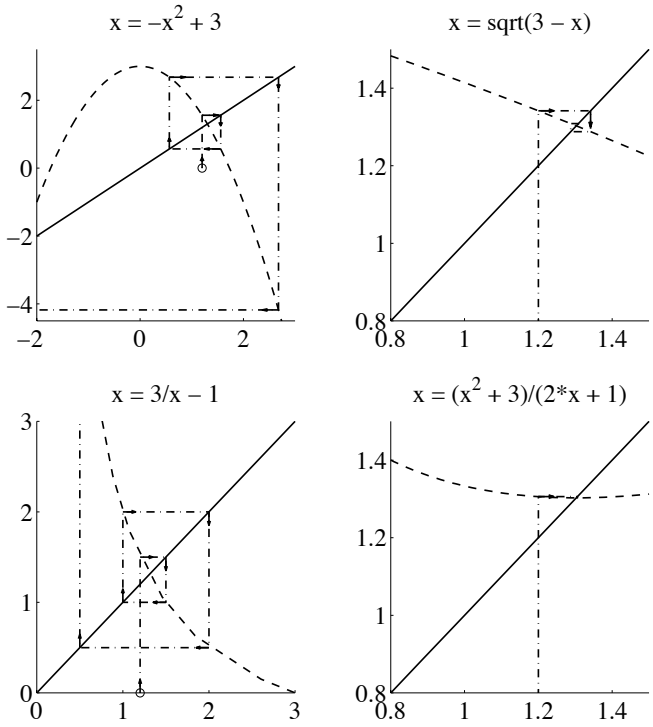
Väliarvolauseen perusteella pisteiden x_k ja x^* välillä on piste z_k, jolle pätee

$$g(x_k) - g(x^*) = g'(z_k)(x_k - x^*).$$

Saamme siis yhtälön $e_{k+1} = g'(z_k)e_k$. Jos pisteen x^* läheisyydessä pätee $|g'(x^*)| < 1$, on olemassa vakio c, jolle on voimassa epäyhtälö $|g'(x^*)| \leq c < 1$, kun $k = 1, 2, \ldots$. Näin saamme arvion

$$|e_{k+1}| \leq c|e_k| \leq c^k|e_1|.$$

Koska pätee $c^k \to 0$, iteraatio suppenee. Lisäksi suppenemisnopeus on tyypillisessä tapauksessa lineaarista. Huonosti valitulla kiintopistemenetelmällä voi kuitenkin olla ongelmia suppenemisessa ja tarkkuudessa, sillä arvon $|g'(x_k)|$ tulisi olla lähellä nollaa, jotta menetelmä olisi tehokas.



Kuva 6.6: *Yhtälön $x^2 + x - 3 = 0$ ratkaiseminen kiintopisteiteraatioilla. Alkuarvaus on $x = 1.2$. Nollakohta on $x^* \approx 1.3028$. Vasemmalla esitetyt iteraatiot eivät suppene, oikealla esitetyt suppenevat kohti nollakohtaa.*

6.3.6 Polynomiyhtälöt

Polynomeille pätee seuraava keskeinen lause:

Lause 6.3.1 (Algebran peruslause) Polynomilla, jonka aste on m, on täsmälleen m kompleksista juurta, kun muotoa $(x-a)^q$, $q \geq 1$, oleva juuritekijä lasketaan mukaan q kertaa.

Polynomiyhtälön ratkaisemiseen ei kannata käyttää yleismenetelmiä vaan varsinaisia polynomiratkaisijoita (esimerkiksi *Müllerin* ja *Laguerren menetelmät*). Suositeltavia ovat aliohjelmakirjastoista löytyvät luotettavat ja tehokkaat ratkaisijat. Esimerkiksi IMSL:n polynomiratkaisija perustuu *Jenkinsin ja Traubin menetelmään*, joka on kolmitasoinen monialgoritmi [IMS]. Tätä pidetään tehokkaimpana menetelmänä, kun halutaan löytää kaikki juuret. Polynomiratkaisijoita löytyy myös ACM:n TOMS-algoritmeista.

Jos polynomiyhtälön kertaluku on neljä tai vähemmän, voit tietenkin käyttää tunnettuja ratkaisukaavoja (löytyvät ohjelmina esimerkiksi Netlibistä).

Eräs polynomiyhtälöiden ratkaisemisessa usein käytetty menetelmä on *deflaatio:* kun on löydetty yhtälön juuri $x = r$, jaamme polynomin termillä $x-r$, jolloin saamme alempaa astetta olevan polynomin. Jos uuden polynomin korkeinta astetta olevien termien kertoimet lasketaan ensin, deflaatio tulee aloittaa pienimmistä juurista.

Polynomiyhtälön juuren löydyttyä on usein hyödyllistä tarkentaa juurta aloittamalla iteraatio uudelleen löydetystä arvosta. Yksi tai kaksi Newtonin iteraatiota on usein riittävästi. Jos juuri on kompleksinen, tulee tarkentamiseen tietenkin käyttää kompleksilukuja tukevaa algoritmia.

Esimerkki 6.3.5 Suuriasteiset polynomit voivat käyttäytyä hyvin hankalasti. Olkoon

$$f(x) = \prod_{i=1}^{6}(x-i) = x^6 - 21x^5 + 175x^4 - 735x^3 + 1624x^2 - 1764x + 720.$$

Jos muutamme korkeimman termin kerrointa ykkösestä luvuksi $1+e$, missä $e = 10^{-4}$, saamme nollakohdiksi lukujen $1, 2, \ldots, 6$ sijaan arvot

$$\{1.0000, 1.9997, 3.0062, 3.9671, 5.0659, 5.9588\}.$$

Siten pieni muutos yhden kertoimen arvossa voi muuttaa nollakohtien sijaintia huomattavasti. Täten liukulukulaskennasta syntyvillä epätarkkuuksilla voi olla suuri vaikutus saatuun ratkaisuun.

6.4 Epälineaarinen yhtälöryhmä

Esimerkki 6.4.1 Seuraavassa on yksinkertainen esimerkki epälineaarisesta yhtälöryhmästä:

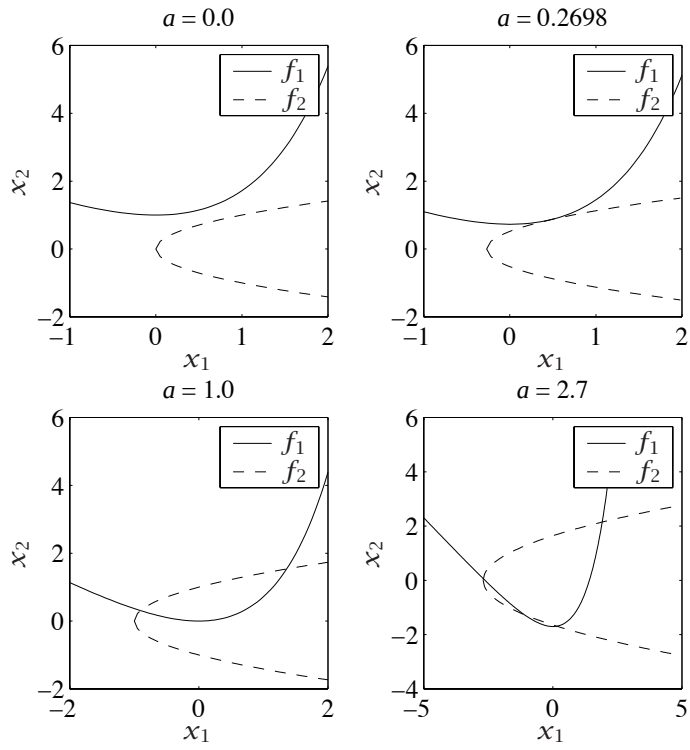
$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} e^{x_1} - x_1 - x_2 - a \\ x_1 - x_2^2 + a \end{pmatrix} = \mathbf{0}. \tag{6.7}$$

Yhtälöryhmällä on 0, 1, 2, 3 tai 4 ratkaisua riippuen vakion a arvosta. Funktioiden nollakäyrät vakion a neljällä eri arvolla on esitetty kuvassa 6.7.

Epälineaarisen yhtälöryhmän ratkaiseminen on huomattavasti vaikeampaa kuin yhden muuttujan yhtälön ratkaiseminen. Kuva 6.7 hahmottaa asiaa. Yhtälöryhmän

$$\begin{cases} f_1(x_1, x_2) = 0 \\ f_2(x_1, x_2) = 0 \end{cases}$$

ratkaisut löytyvät nollakäyrien leikkauspisteistä. Pienetkin pyöristysvirheet voivat sopivissa kohdissa saada aikaan sen, että löydämme ylimääräisen ratkaisun tai emme löydä jotain ratkaisua ollenkaan.



Kuva 6.7: *Yhtälöryhmän (6.7) nollakäyrät muutamalla vakion a eri arvolla.*

6.4.1 Suosituksia menetelmän valintaan

Epälineaaristen yhtälöiden numeeriset ratkaisumenetelmät ovat iteratiivisia: keksimme ensin alkuarvauksen $\mathbf{x}^1$ ja toistamme tämän jälkeen iteraatiota $\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{s}^k$, $k = 1, 2, 3, \ldots$. Tässä vektori $\mathbf{s}^k$ on menetelmästä riippuva hakuaskel.

Seuraavassa on lyhyt yhteenveto suositeltavista ratkaisumenetelmistä (vertaa ohjeeseen sivulla 191):

- Jos Jacobin matriisi on helposti laskettavissa, Newtonin menetelmä yhdistettynä sopivaan globaaliin strategiaan on melko luotettava.
- Käyttämällä differenssiarviota Newtonin menetelmän Jacobin matriisille päästään vain lineaariseen suppenemiseen mutta laskutoimituksia tarvitaan vähemmän iteraatiota kohden.
- Kun Jacobin matriisi ei ole helposti laskettavissa, voidaan Jacobin matriisia arvioida sekanttimenetelmillä. Tarvittavien laskutoimitusten määrä vähenee ratkaisevasti, kun päivitys tehdään iteraatiomatriisin hajotelmaan.

Toimivaan algoritmiin kuuluu myös suppenemisen havaitseminen sekä mahdollisten erikoistilanteiden hallinta. Menetelmän lopetusehto voi olla vaikkapa jonkin seuraavista:

- askelpituus $\|\mathbf{s}^k\|$ on pieni
- residuaali $\|\mathbf{f}(\mathbf{x}^k)\|$ on pieni
- menetelmän askelpituus kasvaa liian suureksi eli menetelmä epäonnistuu.

Hyvä alkuarvaus vaikuttaa suuresti tarvittavien laskutoimitusten määrään; monet menetelmät eivät edes suppene, ellei alkupiste $\mathbf{x}^1$ ole kyllin lähellä nollakohtaa $\mathbf{x}^*$. Täten ongelman ominaisuuksien tunteminen, vaikkapa hyvä arvaus ratkaisualueelle, voi ratkaisevasti auttaa laskennassa.

6.4.2 Newtonin menetelmä epälineaarisille yhtälöryhmille

Olkoon ratkaistavana epälineaarinen yhtälö $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, missä $\mathbf{f}$ on jatkuvasti derivoituva funktio $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$. Voimme arvioida funktiota pisteessä $\mathbf{x} + \mathbf{s}$ Taylorin sarjalla:

$$\mathbf{f}(\mathbf{x} + \mathbf{s}) \approx \mathbf{f}(\mathbf{x}) + J(\mathbf{x})\,\mathbf{s}.$$

Tässä vektorin $\mathbf{s}$ kertoimena on Jacobin matriisi $J_{ij} = \partial f_i / \partial x_j$. Asettamalla $\mathbf{f}(\mathbf{x} + \mathbf{s}) = \mathbf{0}$ saamme ratkaistavaksi yhtälöryhmän

$$J(\mathbf{x})\,\mathbf{s} = -\mathbf{f}(\mathbf{x}). \tag{6.8}$$

Askel $\mathbf{s}$ on siis ratkaistavissa lineaarisesta yhtälöryhmästä. Olemme siis johtaneet *Newtonin menetelmän*.

Algoritmi 6.4.1 (Newtonin menetelmä)

tee alkuarvaus $\mathbf{x}^1 \in \mathbb{R}^n$
$k = 1$
repeat
  ratkaise askel $\mathbf{s}^k$ yhtälöryhmästä $J(\mathbf{x}^k)\,\mathbf{s}^k = -\mathbf{f}(\mathbf{x}^k)$
  $\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{s}^k$
  $k = k + 1$
until $\|\mathbf{s}^k\| < \epsilon$ tai $k > k_{\max}$

Tässä $J(\mathbf{x}^k)$ on funktion f Jacobin matriisi $\partial f_i(\mathbf{x}^k)/\partial x_j$ ja vektori $\mathbf{s}^k$ on iteraatiolla k otettu Newtonin askel.

Huomautus 6.4.1 Newtonin menetelmä suppenee neliöllisesti ratkaisupisteen lähellä, mutta kaukana ratkaisusta suppeneminen on epävarmaa. Siten Newtonin menetelmä ei sovellu sellaisenaan luotettavaksi ratkaisualgoritmiksi.

Newtonin menetelmä on työläs, sillä jokaisella iteraatiokierroksella on laskettava Jacobin matriisi ja ratkaistava vektori $\mathbf{s}^k$ mahdollisesti singulaarisesta yhtälöryhmästä $J(\mathbf{x}^k)\,\mathbf{s}^k = -\mathbf{f}(\mathbf{x}^k)$. Jos yhtälö on singulaarinen, täytyy Newtonin askeleen sijaan käyttää jotakin toista menetelmää tai vaihtoehtoisesti muuttaa yhtälöryhmää vähemmän häiriöalttiiksi.

Jacobin matriisia voi yrittää arvioida differensseillä, mikäli se ei ole analyyttisesti saatavilla. Parhaassa tapauksessa saavutetaan neliöllinen suppeneminen ja huonoimmassakin useimmiten lineaarinen.

Esimerkki 6.4.2 Ratkaisemme Newtonin menetelmällä yhtälöryhmän

$$\begin{cases} e^{x_1} - x_1 - x_2 - 1 &= 0, \\ x_1 - x_2^2 + 1 &= 0. \end{cases}$$

Tehtävän Jacobin matriisi on

$$J(\mathbf{x}) = \begin{pmatrix} e^{x_1} - 1 & -1 \\ 1 & -2x_2 \end{pmatrix}.$$

Alkuarvauksella $\mathbf{x}^1 = (1,1)^T$ saamme seuraavat iteraatit:

k	x_1^k	x_2^k	$f_1(\mathbf{x}^k)$	$f_2(\mathbf{x}^k)$
1	1.0000	1.0000	−0.2817	1.0000
2	1.6417	1.8208	0.7012	−0.6738
3	1.4138	1.5732	0.1244	−0.0613
4	1.3623	1.5374	0.0054	−0.0013
5	1.3600	1.5362	$0.9800\cdot10^{-5}$	$-0.1316\cdot10^{-5}$

Siten iteraatio suppeni viidellä askeleella arvoon $\mathbf{x}^* \approx (1.360, 1.536)^T$. Tehtävällä on myös toinen ratkaisu $\mathbf{x}^* \approx (-0.904, 0.309)^T$, joka löytyy käyttämällä tämän ratkaisupisteen lähellä olevaa alkuarvausta.

6.4.3 Differenssiarvioiden laskeminen

Jos ratkaisemme epälineaarisen yhtälön Newtonin menetelmän tapaisella menetelmällä, tarvitsemme arviot funktion derivaatoille (esimerkiksi Jacobin matriisi ja gradienttivektorit).

Derivaattojen analyyttisiä lausekkeita kannattaa käyttää. Jos funktio on esitettävissä symbolisessa muodossa, derivaatat voi laskea symbolinkäsittelyjärjestelmillä (esimerkiksi Mathematica tai Maple) ja tulostaa vaikkapa Fortran-koodina tiedostoon.

Huomautus 6.4.2 Eräs tyypillisimpiä virheitä yhtälöryhmien ratkaisussa on väärin lasketut funktion derivaatat. Derivaattojen oikeellisuus kannattaakin aina varmistaa, oli sitten kyseessä differenssiarvio tai analyyttisesti lasketut derivaatat.

Symbolisen laskennan ohjelmistoilla voi laskea funktioiden derivaatat. Ohjelmistoja voi myös käyttää saatujen lausekkeiden sieventämiseen ja optimointiin. Toinen lähestymistapa on *automaattinen derivoiminen* (automatic differentiation, algorithmic differentiation). Tällä tarkoitetaan aliohjelman muodossa annetun funktion symbolista derivoimista. Automaattisen derivoinnin etuna on saadun koodin tiiviys ja tehokkuus. Lisäksi useat menetelmät tuottavat samalla virhearvion aliohjelmakoodin laskemalle funktion arvolle.

Käytettäessä derivaatoille differenssiarvioita on syytä huomioida numeerisen tarkkuuden vaikutukset arvioiden laskemiseen. Useat valmisohjelmistot (esimerkiksi aliohjelmakirjastojen IMSL ja NAG rutiinit) osaavat arvioida haluttaessa derivaattoja differensseillä, mutta tähän tietysti kuluu jonkin verran laskutoimituksia.

Funktion $\mathbf{f}(\mathbf{x})$ Jacobin matriisille $J(\mathbf{x}) = \partial f_i(\mathbf{x})/\partial x_j$ voi käyttää esimerkiksi seuraavia differenssiarvioita $J(\mathbf{x}) = J_{ij}$:

$$\begin{cases} J_{ij} = \dfrac{f_i(\mathbf{x} + h_j\mathbf{e}^j) - f_i(\mathbf{x})}{h_j}, & \text{eteenpäin laskettu differenssi,} \\ J_{ij} = \dfrac{f_i(\mathbf{x}) - f_i(\mathbf{x} - h_j\mathbf{e}^j)}{h_j}, & \text{taaksepäin laskettu differenssi,} \\ J_{ij} = \dfrac{f_i(\mathbf{x} + h_j\mathbf{e}^j) - f_i(\mathbf{x} - h_j\mathbf{e}^j)}{2h_j}, & \text{keskeisdifferenssi.} \end{cases}$$

Tässä $\mathbf{e}^j$ on j:s yksikkövektori eli $\mathbf{e}^j = (e_1, \ldots, e_n)$, $e_j = 1$ ja $e_i = 0$, $i \neq j$. Kerroin h_j on suuntaan $\mathbf{e}^j$ valittu askelpituus.

Huomautus 6.4.3 Askelpituuden valinnalla on suuri merkitys differenssiarvion tarkkuuden kannalta: liian pieni askelpituus tuottaa numeerisesti epätarkkoja tuloksia ja liian suuri antaa epätäsmällisen arvion derivaatalle (katso kuvaa 2.5 sivulla 27). Jos luvut ovat suuruusluokkaa 1, hyvä ensimmäinen arvio askelpituudelle on $\sqrt{\epsilon_{\text{kone}}}$, joka on 32-bittisessä aritmetiikassa $\approx 10^{-4}$.

6.4.4 Sekanttimenetelmät

Newtonin menetelmässä täytyy laskea kohdefunktioiden derivaatat Jacobin matriisiin. Jos derivaattoja ei voida tai haluta laskea, ovat sekanttimenetelmät hyvä vaihtoehto Newtonin menetelmälle.

Sekanttimenetelmässä päivitetään iteratiivisesti matriisia B_k, jonka voi tulkita Jacobin matriisin arvioksi. Menetelmässä ei kuitenkaan tarvitse laskea funktioiden derivaattoja.

Algoritmi 6.4.2 (Sekanttimenetelmä)

tee alkuarvaus $\mathbf{x}^1 \in \mathbb{R}^n$
$k = 1$

repeat
  laske $\mathbf{f}(\mathbf{x}^k)$
  päivitä matriisia B_k
  ratkaise hakuaskel yhtälöryhmästä $B_k\,\mathbf{s}^k = -\mathbf{f}(\mathbf{x}^k)$
  $k = k + 1$
until $\|\mathbf{s}^k\| < \epsilon$ tai $k > k_{\max}$

Sekanttimenetelmien suppeneminen on superlineaarista mutta ei neliöllistä kuten Newtonin menetelmän. *Broydenin sekanttimenetelmä* arvioi funktion $\mathbf{f}(\mathbf{x})$ Jacobin matriisia käyttäen päivitystä, joka voidaan kohdistaa edellisen arvion QR-hajotelmaan (sivu 45). Täten menetelmässä säästetään laskutyötä.

Olkoon $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$, $\mathbf{x}^1 \in \mathbb{R}^n$ ja $B_1 \in \mathbb{R}^{n\times n}$. Broydenin sekanttimenetelmässä käytetty päivitys näyttää seuraavalta:

$$B_k\mathbf{s}^k = -\mathbf{f}(\mathbf{x}^k) \tag{6.9}$$

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{s}^k$$

$$\mathbf{y}^k = \mathbf{f}(\mathbf{x}^{k+1}) - \mathbf{f}(\mathbf{x}^k)$$

$$B_{k+1} = B_k + \frac{\left(\mathbf{y}^k - B_k\mathbf{s}^k\right)\mathbf{s}^{kT}}{\langle \mathbf{s}^k, \mathbf{s}^k\rangle} = B_k + \frac{\mathbf{f}(\mathbf{x}^{k+1})\mathbf{s}^{kT}}{\langle \mathbf{s}^k, \mathbf{s}^k\rangle} \tag{6.10}$$

Broydenin sekanttimenetelmä päivitettää matriisin B_k arvoa ehdolla

$$B_{k+1}\mathbf{s}^k = f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k).$$

Tätä ehtoa kutsutaan *sekanttiyhtälöksi*. Broydenin menetelmä on työmäärältään tehokas, koska matriisia B_k muutetaan mahdollisimman vähän iteraatiosta toiseen.

Matriisiksi B_1 voi valita esimerkiksi differenssiarvion funktion $\mathbf{f}$ Jacobin matriisille pisteessä $\mathbf{x}^1$. Myös identiteettimatriisi I käy alkuarvaukseksi. Jos sekanttimenetelmän päivitys tehdään suoraan matriisin B_k hajotelmaan, algoritmin työmäärä on luokkaa n^2 kullakin iteraatiolla.

Esimerkki 6.4.3 Ratkaisemme Broydenin sekanttimenetelmällä yhtälöryhmän

$$\begin{cases} e^{x_1} - x_1 - x_2 - 1 = 0 \\ x_1 - x_2^2 + 1 = 0. \end{cases}$$

Olkoon alkuarvaus $\mathbf{x}^1 = (1,1)^T$ ja matriisi $B_1 = I$. Menetelmä käyttäytyy seuraavasti:

k	x_1^k	x_2^k	$f_1(\mathbf{x}^k)$	$f_2(\mathbf{x}^k)$
1	1.0000	1.0000	−0.2817	1.0000
2	1.2817	0.0000	1.3211	2.2817
3	2.9994	2.9666	13.1071	−4.8014
4	0.2183	1.0295	−1.0039	0.1583
5	0.4715	1.0813	−0.9504	0.3024
6	1.6569	1.7844	0.8018	−0.5272
	...			
12	1.3595	1.5362	−0.0013	−0.0005
13	1.3602	1.5363	$0.4767\cdot10^{-3}$	$0.1196\cdot10^{-3}$

Siten löysimme 13 iteraatiolla ratkaisun $\mathbf{x}^* \approx (1.360, 1.536)^T$.

6.4.5 Yhtälöryhmien ratkaiseminen luotettavasti

Edellä esitellyille yhtälöryhmien ratkaisumenetelmille ei voida taata suppenemista kaukana nollakohdasta. Yleensäkin hyvä alkuarvaus on erittäin tärkeä. Nollakohdan lähellä nopeasti suppenevaan menetelmään onkin syytä yhdistää jokin suppenemista varmistava tekniikka.

Eräs mahdollisuus varmistaa suppeneminen on korvata Newtonin menetelmän askel viivahaulla: $\mathbf{x}^{k+1} = \mathbf{x}^k + \lambda_k \mathbf{s}^k$.

Yhtälössä (6.10) esiintyvä lauseke $\mathbf{y}^k - B_k\mathbf{s}^k$ voidaan sieventää muotoon $\mathbf{f}(\mathbf{x}^{k+1})$ käyttämällä apuna yhtälöä $B_k\mathbf{s}^k = -\mathbf{f}(\mathbf{x}^k)$. Menetelmä esitetään kuitenkin edellä annetussa muodossa siksi, että käytännössä askelpituutta ei ratkaista suoraan yhtälöstä (6.9). Nimittäin jos matriisi B_k on häiriöaltis, tuloksena on liian pitkä askel. Hakuaskel ratkaistaankin usein minimointiprobleemasta

$$\underset{\mathbf{s}^k \in \mathbb{R}^n}{\text{minimi}} \ \|\mathbf{f}(\mathbf{x}^k) + B_k\mathbf{s}^k\| \text{ ehdolla } \|\mathbf{s}^k\| \leq \delta_k,$$

missä δ_k on (iteraatiosta riippuva) askeleen maksimipituus. Kyseessä on *luottamusaluemenetelmä* (trust region method), joka löytyy mm. IMSL-aliohjelmakirjastosta.

Myös MINPACK-aliohjelmakirjastossa käytetään luottamusalueeseen perustuvaa menetelmää. MINPACKissa otettava hakuaskel on kombinaatio Newton-askeleesta ja pienimmän neliösumman minimointitehtävän ratkaisuaskeleesta. MINPACK löytyy mm. Netlib-verkkoarkistosta.

Homotopiamenetelmä on eräs vaihtoehtoinen tapa ratkaista epälineaarisia yhtälöryhmiä. Netlibistä on saatavilla Fortran-aliohjelmakirjasto Hompack (TOMS-algoritmi numero 652 [WBM87]), joka sisältää joukon rutiineja epälineaaristen yhtälöryhmien $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ ja kiintopistetehtävien $\mathbf{f}(\mathbf{x}) = \mathbf{x}$ ratkaisemiseen.

6.5 Epälineaaristen yhtälöiden yhteys optimointiin

Epälineaaristen yhtälöryhmien ratkaisemiseen käytetään toisinaan optimointitehtävien algoritmeja. Esimerkiksi Newton-tyyppiset menetelmät soveltuvat pienin muutoksin molempiin tehtäviin.

Yhtälön $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ ratkaisu $\mathbf{x}^*$ minimoi summan

$$F(\mathbf{x}) = \sum_{i=1}^{n} (f_i(\mathbf{x}))^2 \ . \tag{6.11}$$

Täten kyseessä on epälineaarinen pienimmän neliösumman tehtävä (katso sivu 353). Kuitenkaan yhtälöstä 6.11 saatu optimikohta ei välttämättä toteuta yhtälöä $\mathbf{f}(\mathbf{x}) = \mathbf{0}$, sillä voi löytyä lokaali minimi, joka kuitenkaan ei ole yhtälöryhmän nollakohta. Siten epälineaarisen yhtälöryhmän ratkaisemista ei useinkaan kannata muuttaa optimointitehtäväksi.

Epälineaarisiin yhtälöihin voidaan päätyä rajoittamattomasta epälineaarisesta optimoinnista, nimittäin jos tehtävänä on löytää minimi $f(\mathbf{x})$, saamme optimikohdalle $\mathbf{x}^*$ välttämättömän ehdon $\nabla f(\mathbf{x}^*) = \mathbf{0}$.

Optimointitehtävät ovat yleensä helpompia ratkaista kuin epälineaariset yhtälöryhmät. Esimerkiksi minimoinnissa voidaan ”liukua alamäkeen” eli seuraavan iteraatin laskemiselle on luonnollinen suunta, kun taas nollakohdan etsimisessä ei sellaista ole olemassa.

6.6 Yhteenveto

Tämä luku käsitteli sekä yhden epälineaarisen yhtälön että yhtälöryhmän ratkaisua. Perusresepti on seuraava:

- Hae rajat alueelle, mistä ratkaisua etsitään.
- Keksi hyvä alkuarvaus.
- Valitse tehtävään sopiva menetelmä. Yhden yhtälön ratkaisulle löydät suosituksia kappaleesta 6.3.1 (sivu 191). Yhtälöryhmän ratkaisumenetelmiä on vertailtu kappaleessa 6.4.1 (sivu 200).
- Varmista, että löydetty ratkaisuehdokas todella on ratkaisu.

6.7 Ohjelmistoja epälineaaristen yhtälöiden ratkaisemiseen

Matlab-ohjelmiston Optimization Toolbox [CBG99] mahdollistaa epälineaaristen yhtälöiden ja yhtälöryhmien ratkaisemisen. Myös Mathematica- ja Maple-ohjelmistoja voi käyttää ratkaisemiseen. Nämä ohjelmistot soveltuvat kuitenkin pieniin ja keskisuuriin tehtäviin sekä tilanteisiin joissa ei tarvita kovin suurta laskentatehoa.

Jos tarvitset omaan ohjelmistoosi testatun aliohjelman epälineaaristen yhtälöiden ratkaisemiseen, kokeile jotakin monista vaihtoehtoisista valmiskoodeista. Näitä löytyy Netlibistä [Haa98] ja kaupallisista aliohjelmakirjastoista (esimerkiksi NAG ja IMSL).

Taulukossa 6.1 on lueteltu Netlibistä löytyviä ACM:n TOMS-algoritmeja. Näitä ohjelmakoodeja voi suositella, sillä niitä on perusteellisesti testattu ja ne on dokumentoitu sarjajulkaisussa *ACM Transactions on Mathematical Software*.

Taulukko 6.1: *Epälineaaristen yhtälöiden ratkaisemiseen soveltuvia Netlibin hakemistosta* `toms` *löytyviä algoritmeja.*

Nro	*Kuvaus*
326	Pieniasteisten polynomiyhtälöiden nollakohdat
554	Epälineaariset yhtälöt (Brentin menetelmä)
566	Ratkaisuohjelmistojen testaus
573	Epälineaariset pienimmän neliösumman tehtävät
617	Epälineaaristen yhtälöiden ratkaisu
652	Homotopia-algoritmeja (HOMPACK)
681	Epälineaarinen yhtälöryhmä ja laatikkorajoitteet
746	Automaattinen derivoiminen
755	Algoritmien automaattinen derivoiminen
768	Epälineaariset yhtälöt ja pienimmän neliösumman tehtävät
777	Homotopia-algoritmeja (HOMPACK90, kielenä Fortran 90)
795	Polynomiyhtälöiden ratkaisu homotopiamenetelmällä (PHCPACK)
801	Polynomiyhtälöiden ratkaisu homotopiamenetelmällä (POLSYS_PLP)

6.8 Lisätietoja

Teos *Numerical Methods for Unconstrained Optimization and Nonlinear Equations* [DS83] on klassinen johdatus epälineaaristen yhtälöiden ratkaisemiseen. Teos soveltuu myös käsikirjaksi. Teos *Iterative Methods for Linear and Nonlinear Equations* [Kel95] on ajan tasalla oleva johdatus yhtälöryhmien ratkaisemiseen iteratiivisilla menetelmillä. Teos *Methods for Solving Systems of Nonlinear Equations* [Rhe98] soveltuu katsaukseksi ratkaisumenetelmien ominaisuuksiin. Teos sisältää myös todistuksia menetelmien suppenemisesta.

7 Tavalliset differentiaaliyhtälöt

Tutkittaessa esimerkiksi fysikaalisia, kemiallisia ja biologisia systeemejä tarvitaan usein malleja, jotka kuvaavat joidenkin suureiden muutosta esimerkiksi ajan tai paikan suhteen. Tällaisessa mallintamisessa käytetään monesti differentiaaliyhtälöitä. Kun yhtälöissä esiintyy vain yksi riippumaton muuttuja, puhutaan tavallisista differentiaaliyhtälöistä.

Esimerkki 7.0.1 Veden ionisoituminen noudattaa reaktiota

$$H_2O \rightleftharpoons H^+ + OH^-.$$

Olkoot $[H_2O](t)$, $[H^+](t)$ ja $[OH^-](t)$ vesimolekyylien, protonien ja hydroksidi-ionien konsentraatiot (mol/l) hetkellä t. Ionisoitumisnopeus on verrannollinen veden konsentraatioon, ja veden muodostumisnopeus on verrannollinen protonien ja hydroksidi-ionien konsentraatioiden tuloon:

$$\frac{d}{dt}[H_2O](t) = -k_1[H_2O](t) + k_2[H^+](t)[OH^-](t).$$

Kertoimet k_1 ja k_2 ovat reaktionopeusvakioita. Merkitään

$$[H_2O](t) = [H_2O](0) + x(t)$$

$$[H^+](t) = [H^+](0) - x(t)$$

$$[OH^-](t) = [OH^-](0) - x(t).$$

Arvo $x(t)$ on siis poikkeama alkutilasta. Näin saadaan differentiaaliyhtälö

$$\frac{d}{dt}x(t) = -\left(k_1 + k_2([H^+](0) + [OH^-](0))\right)x(t) + k_2x^2(t) - k_1[H_2O](0) + k_2[OH^-](0)[H^+](0),$$

jonka ratkaisu alkuehdolla $x(0) = 0$ kuvaa reaktion kulkua.

Yksinkertaisimmat yhtälöt voidaan ratkaista analyyttisesti, mutta käytännön sovelluksissa joudutaan useimmiten tyytymään numeeriseen ratkaisuun. Tavallisten differentiaaliyhtälöiden numeeriseen ratkaisemiseen on kehitetty suuri joukko menetelmiä. Muun muassa ratkaistavan tehtävän ominaisuudet ja ratkaisulle asetettavat tarkkuusvaatimukset vaikuttavat sopivan menetelmän valintaan.

Aloitamme käymällä läpi muutamia peruskäsitteitä ja siirrymme sitten kuvaamaan numeerisia ratkaisumenetelmiä.

7.1 Tehtävätyyppejä ja käsitteitä

Tässä kappaleessa esittelemme lyhyesti tavallisimmat tehtäväluokat ja käsittelemme ratkaisujen olemassaoloa ja ominaisuuksia.

7.1.1 Alkuarvotehtävät

Esimerkki 7.1.1 Olkoon $j(t)$ jänisten ja $k(t)$ kettujen lukumäärä hetkellä t. Voimme kuvata jänis- ja kettupopulaatioita yksinkertaisella peto–saalis-mallilla

$$\begin{aligned} j'(t) &= (\beta_j - \mu_j)j(t) - \alpha_j j(t)k(t), \\ k'(t) &= (\beta_k - \mu_k)k(t) + \alpha_k j(t)k(t), \\ j(t_0) &= j_0, \\ k(t_0) &= k_0, \end{aligned}$$

missä j_0 ja k_0 ovat jänisten ja kettujen määrät alkuhetkellä $t = t_0$. Parametrit β_j ja β_k kuvaavat syntyvyyksiä ja parametrit μ_j ja μ_k kuolleisuuksia. Parametrit α_j ja α_k mallintavat saalistuksen aiheuttamaa jänisten ja kettujen vuorovaikutusta.

Esimerkin 7.1.1 kaltaista tehtävää kutsutaan tavallisen differentiaaliyhtälön *alkuarvotehtäväksi* (initial value problem, IVP). Niissä tunnetaan systeemin alkutila tietyllä ajanhetkellä t_0 sekä systeemin muutosta kuvaavat yhden riippumattoman muuttujan (esimerkissä aika) differentiaaliyhtälöt. Tavoitteena on määrätä systeemin tila myöhemmällä hetkellä $t > t_0$.

Useamman differentiaaliyhtälön muodostamat systeemit kirjoitetaan tavallisesti muotoon

$$\mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y}(t)), \tag{7.1}$$

$$\mathbf{y}(t_0) = \mathbf{y}_0, \tag{7.2}$$

missä tuntematon $\mathbf{y}$ on vektori ja funktio $\mathbf{f}$ on vektoriarvoinen.

7.1.2 Ratkaisun olemassaolo alkuarvotehtäville

Olkoon $\mathbf{f} : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ jatkuva joukossa $D = \{(t, \mathbf{z}) | t_0 \le t \le T, \mathbf{z} \in \mathbb{R}^n\}$, missä t_0 ja T ovat äärellisiä. Alkuarvotehtävällä (7.1)–(7.2) on olemassa yksikäsitteinen, joukossa D jatkuvasti derivoituva ratkaisu $\mathbf{y}(t)$, jos on olemassa vakio L siten, että kaikilla $t, \mathbf{z}_1, \mathbf{z}_2$, joilla $(t, \mathbf{z}_1)$ ja $(t, \mathbf{z}_2) \in D$, pätee epäyhtälö

$$\|\mathbf{f}(t, \mathbf{z}_1) - \mathbf{f}(t, \mathbf{z}_2)\| \le L\|\mathbf{z}_1 - \mathbf{z}_2\|, \tag{7.3}$$

missä $\|\cdot\|$ on jokin vektorinormi. Ehtoa (7.3) sanotaan *Lipschitzin ehdoksi*.

Jos $\mathbf{f}(t,\mathbf{z})$ on lisäksi jatkuvasti derivoituva muuttujan $\mathbf{z}$ suhteen, kun $(t,\mathbf{z}) \in D$, voidaan valita

$$L = \sup_{(t,\mathbf{z})\in D} \|J(t,\mathbf{z})\|,$$

missä $\|\cdot\|$ on epäyhtälössä (7.3) käytettyyn vektorinormiin liittyvä matriisinormi ja $J(t,\mathbf{z}) = \partial\mathbf{f}(t,\mathbf{z})/\partial\mathbf{z}$ on funktion $\mathbf{f}(t,\mathbf{z})$ Jacobin matriisi (liite A):

$$J_{ij} = \partial f_i(t,z_1,z_2,\ldots,z_n)/\partial z_j.$$

7.1.3 Korkeamman kertaluvun tehtävät

Korkeimman yhtälössä esiintyvän derivaatan kertaluku on differentiaaliyhtälön kertaluku. Esimerkiksi seuraava ongelma on kertalukua kaksi:

$$y'' = -y, \quad y(0) = 0, \quad y'(0) = 1. \tag{7.4}$$

Merkitsemme jatkossa $y'(t) = y'$ ja $y(t) = y$, kun argumentti ei ole esityksen kannalta oleellinen. Tehtävä (7.4) voidaan helposti muuntaa ensimmäisen kertaluvun differentiaaliyhtälöryhmäksi. Merkitsemme alkuperäistä muuttujaa $y_1 = y$ ja otamme käyttöön uuden muuttujan $y_2 = y_1'$. Saamme yhtälöryhmän

$$\begin{cases} y_1' = y_2, \\ y_2' = y_1'' = -y_1, \\ y_1(0) = 0, \\ y_2(0) = y_1'(0) = 1. \end{cases} \tag{7.5}$$

Ratkaisemalla tehtävä (7.4) ensimmäisen kertaluvun ryhmänä (7.5) saamme ratkaisukäyrän y approksimaation y_1 lisäksi derivaatan y' approksimaation y_2.

Kertalukua n oleva yleinen alkuarvotehtävä

$$\begin{cases} y^{(n)} &= f(t,y,y',\ldots,y^{(n-1)}), \\ y(a) &= A_1, \\ y'(a) &= A_2, \\ &\vdots \\ y^{(n-1)}(a) &= A_n \end{cases}$$

voidaan helposti muuntaa ensimmäisen kertaluvun differentiaaliyhtälöryhmäksi

$$\begin{cases} y_1' &= y_2, \\ y_2' &= y_3, \\ &\vdots \\ y_n' &= f(x,y_1,y_2,\ldots,y_n), \\ y_1(a) &= A_1, \\ y_2(a) &= A_2, \\ &\vdots \\ y_n(a) &= A_n. \end{cases}$$

7.1.4 Reuna-arvotehtävät

Esimerkki 7.1.2 Tarkastelemme palkkia, jonka toinen pää on tuettu jäykästi ja toinen pää on vapaa. Olkoon l palkin pituus ja $y(x)$ poikkeama kohdassa x mitattuna palkkia pitkin. Tietyin oletuksin kuormitetun palkin poikkeama toteuttaa yhtälön

$$EIy''''(x) = q(x) \tag{7.6}$$

ja *reunaehdot*

$$\begin{aligned} y(0) &= 0, \\ y'(0) &= 0, \\ y''(l) &= 0, \qquad (7.7) \\ y'''(l) &= 0. \qquad (7.8) \end{aligned}$$

Yhtälössä (7.6) q on kuorma, E materiaalin kimmokerroin ja I palkin poikkileikkauksen neliömomentti. Reunaehtojen (7.7) ja (7.8) mukaan palkin vapaaseen päähän ei kohdistu momenttia eikä leikkausvoimaa.

Alkuarvotehtävästä poiketen ratkaisufunktion tai sen derivaattojen täytyy toteuttaa ehtoja tarkasteluvälin *molemmissa* päissä, esimerkissä 7.1.2 pisteissä $x = 0$ ja $x = l$.

Yhtälö (7.6) reunaehtoineen voidaan palauttaa edellä esitetyllä tavalla ensimmäisen kertaluvun systeemiksi.

Yleisessä *reuna-arvotehtävässä* (boundary value problem, BVP) etsitään ratkaisua ryhmälle

$$\mathbf{y}' = \mathbf{f}(t, \mathbf{y}),$$

missä vektoreissa $\mathbf{y}$ ja $\mathbf{f}(t, \mathbf{y})$ on n komponenttia. Lisäksi ratkaisua sitoo n ehtoa tarkasteluvälin päätepisteissä

$$\mathbf{g}(\mathbf{y}(a), \mathbf{y}(b)) = \mathbf{0}.$$

Reuna-arvotehtävien ratkaiseminen on huomattavasti vaikeampaa kuin alkuarvotehtävien. Ratkaisun olemassaolokaan ei ole aina itsestäänselvyys. Esimerkiksi alkuarvotehtävällä

$$y'' = f(t, y, y'),\ y(a) = \alpha,\ y'(a) = \beta$$

on yksikäsitteinen ratkaisu lähes kaikilla riittävän sileillä funktioilla f, kun taas reuna-arvotehtävällä

$$y'' = f(t, y, y'),\ y(a) = \alpha,\ y(b) = \beta$$

ei ole välttämättä ratkaisua ollenkaan tai ratkaisuja voi olla äärettömän paljon.

Ratkaisujen olemassaoloa koskevia tuloksia on kuitenkin olemassa joillekin tehtäväluokille. Voidaan esimerkiksi osoittaa, että jos $\partial f/\partial y > 0$, tehtävällä

$$y'' = f(t, y),\ y(a) = \alpha,\ y(b) = \beta$$

on olemassa yksikäsitteinen ratkaisu välillä (a, b) kohtuullisilla lisäoletuksilla.

7.1.5 Differentiaaliyhtälöiden ratkaisujen stabiilisuus ja epästabiilisuus

Yksinkertaistaen voidaan sanoa, että alkuarvotehtävän ratkaisu on *stabiili*, jos pieni muutos alkuarvossa aiheuttaa vain pienen muutoksen ratkaisuun muilla ajanhetkillä.

Tarkastelemme testitehtävää

$$y' = \lambda y, \tag{7.9}$$

missä λ on kompleksinen vakio. Ratkaisu on

$$y(t) = y(0)e^{\lambda t},$$

kun $t \geq 0$. Kahden ratkaisun välinen ero mielivaltaisella hetkellä t on verrannollinen alkuarvojen eroon:

$$|y(t) - \hat{y}(t)| = |(y(0) - \hat{y}(0))e^{\lambda t}| = |y(0) - \hat{y}(0)|e^{\mathrm{Re}(\lambda)t}.$$

Jos $\mathrm{Re}(\lambda) \leq 0$, ratkaisujen ero pysyy rajoitettuna kaikilla hetkillä $t \geq 0$, ja ratkaisua sanotaan *stabiiliksi*. Jos $\mathrm{Re}(\lambda) < 0$, ero pienenee, ja ratkaisu on *asymptoottisesti stabiili*. Jos viimein $\mathrm{Re}(\lambda) > 0$, ratkaisujen erotus kasvaa rajatta, ja ratkaisu on *epästabiili*.

Yleisemmin sanottuna yhtälön

$$\mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y}) \tag{7.10}$$

ratkaisu $\mathbf{y}$ kaikilla $t \geq 0$ on

- *stabiili*, jos kaikilla $\epsilon > 0$ on olemassa $\delta > 0$ siten, että jokainen yhtälön (7.10) ratkaisu $\hat{\mathbf{y}}$, jolle pätee

  $$|\mathbf{y}(0) - \hat{\mathbf{y}}(0)| \leq \delta$$

  toteuttaa myös ehdon

  $$|\mathbf{y}(t) - \hat{\mathbf{y}}(t)| \leq \epsilon$$

  kaikilla $t \geq 0$.
- *asymptoottisesti stabiili*, jos se on stabiili ja

  $$|\mathbf{y}(t) - \hat{\mathbf{y}}(t)| \to 0,$$

  kun $t \to \infty$.

Voidaan osoittaa, että lineaarisen tehtävän

$$\mathbf{y}' = A\mathbf{y} + \mathbf{f}(t)$$

ratkaisu on

- stabiili, jos ja vain jos matriisin A kaikki ominaisarvot λ toteuttavat joko ehdon $\mathrm{Re}(\lambda) < 0$ tai ehdon $\mathrm{Re}(\lambda) = 0$ ja λ ei ole defektiivinen (kappale 9.2),
- asymptoottisesti stabiili, jos ja vain jos kaikille matriisin A ominaisarvoille λ pätee $\mathrm{Re}(\lambda) < 0$.

Esimerkki 7.1.3 Ongelman

$$y' = -y, \quad y(0) = a$$

ratkaisu $y(t) = ae^{-t}$ on asymptoottisesti stabiili, sillä kaikilla alkuarvoilla a ratkaisu $y(t) \to 0$, kun $t \to \infty$.

Esimerkki 7.1.4 Käsittelemme yksinkertaisen esimerkin epälineaarisesta differentiaaliyhtälöstä, jolla on epästabiili ratkaisu. Ongelman

$$y' = ty(y-2), \quad y(0) = 2$$

ratkaisu $y(t) \equiv 2$ on epästabiili. Tämän näemme ratkaisemalla tehtävän alkuehdolla $y(0) = y_0$, jolloin ratkaisu on

$$y(t) = \frac{2y_0}{y_0 + (2-y_0)e^{t^2}}.$$

Jos $y_0 < 2$, ratkaisu $y(t) \to 0$, kun $t \to \infty$. Jos taas $y_0 > 2$, ratkaisu kasvaa rajatta, ja sillä on singulariteetti, kun $y_0 + (2-y_0)e^{t^2} = 0$.

Epästabiilien ratkaisujen numeerinen laskeminen tietokoneella voi joissain tapauksissa olla lähes mahdotonta, koska laskennan aikana syntyvillä pyöristysvirheillä on sama vaikutus kuin virheellisillä alkuarvoilla. Numeerinen ratkaisu saattaa siis olla täysin epäluotettava.

Toisinaan kuitenkin joudutaan myös epästabiileja ratkaisuja approksimoimaan numeerisesti. Vaikka virhe voi kasvaa tällöin suureksi, numeerisella ratkaisulla on parhaassa tapauksessa samoja ominaisuuksia kuin oikealla ratkaisulla.

7.2 Numeerisen ratkaisemisen teoriaa

Seuraavassa tutustumme differentiaaliyhtälöiden numeeriseen ratkaisemiseen liittyviin peruskäsitteisiin yksinkertaisimpien ratkaisumenetelmien

avulla. Näihin kuuluvat Eulerin menetelmä, implisiittinen Eulerin menetelmä ja trapetsimenetelmä.

Näitä menetelmiä ei yleensä käytännön tehtävissä kannata soveltaa, vaan suositeltavampaa on valita tehokkaampia ja tarkempia vaihtoehtoja, joita esitellään myöhemmin.

7.2.1 Eulerin menetelmä

Tarkastelemme alkuarvotehtävää

$$y' = f(t, y), \quad y(a) = y_0. \tag{7.11}$$

Haluamme määrittää ratkaisun välillä $[a, b]$. Monesti numeeriset menetelmät antavat ratkaisun approksimaation kuitenkin vain äärellisessä määrässä *hilapisteitä* t_k siten, että pisteessä t_k laskettu approksimaatio y_k on jossain mielessä lähellä tarkkaa ratkaisua $y(t_k)$. Olkoon hila tasavälinen eli

$$t_k = a + kh, \quad k = 0, 1, \ldots, N, \quad h = (b - a)/N.$$

Lausumme tehtävän (7.11) ratkaisun pisteessä t_{k+1} Taylorin kehitelmänä pisteen t_k suhteen:

$$\begin{aligned} y(t_{k+1}) &= y(t_k) + hy'(t_k) + \frac{h^2}{2} y''(z_k) \\ &= y(t_k) + hf(t_k, y(t_k)) + \frac{h^2}{2} y''(z_k), \end{aligned} \tag{7.12}$$

missä $z_k \in (t_k, t_{k+1})$. Oletamme jatkossa, että ratkaisulla y on olemassa ainakin toinen derivaatta (itse asiassa niin monta kuin kulloinkin tarvitaan!). Jos y'' on rajoitettu ja h pieni, voimme unohtaa yhtälöstä (7.12) viimeisen termin ja saamme relaation $y(t_{k+1}) \approx y(t_k) + hf(t_k, y(t_k))$. Vastaavaa menetelmää

$$\begin{cases} y(a) &= y_0, \\ y_{k+1} &= y_k + hf(t_k, y_k), \quad k = 0, 1, \ldots, N \end{cases} \tag{7.13}$$

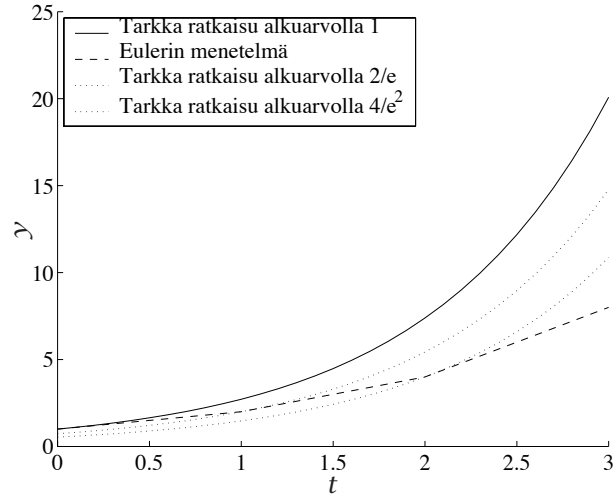
kutsutaan *Eulerin menetelmäksi*. Parametri h on askelpituus. Menetelmää voidaan luonnollisesti soveltaa, vaikka hila ei olisikaan tasavälinen. Yleistys differentiaaliyhtälösysteemeihin on suoraviivainen:

$$\mathbf{y}_{k+1} = \mathbf{y}_k + h\mathbf{f}(t_k, \mathbf{y}_k).$$

Geometrisesti Eulerin menetelmässä approksimoidaan ratkaisua pisteessä t_{k+1} etenemällä pisteessä t_k määrätyn tangentin suuntaisesti askelpituudella h. Kuvassa 7.1 on otettu tehtävää $y' = y$, $y(0) = 1$ ratkaistaessa kolme askelta ($h = 1$). Koska numeerinen ratkaisu ajautuu sivuun tarkasta ratkaisusta, etenemissuunnat ovat läheisten ratkaisukäyrien tangentteja. Askelpituus on havainnollisuuden vuoksi valittu suureksi.

Eulerin menetelmä on yksinkertainen avoin (eksplisiittinen) yksiaskelmenetelmä. Approksimaation $\mathbf{y}_{k+1}$ laskemiseen pisteessä t_{k+1} tarvitaan vain edellinen piste $(t_k, \mathbf{y}_k)$.

Käytämme jatkossa Eulerin menetelmää esimerkkinä tutkiessamme numeeristen menetelmien virhearvioita ja stabiilisuutta.



Kuva 7.1: *Eulerin menetelmässä edetään tangentin suuntaan. Menetelmän aiheuttaman virheen vuoksi derivaatta arvioidaan kuitenkin jollain läheisellä ratkaisukäyrällä oikean ratkaisun sijaan.*

7.2.2 Paikallinen virhe ja kokonaisvirhe

Eulerin menetelmän ainoa parametri, jota käyttäjä voi muuttaa, on askelpituus h. Valitsemalla askelpituus sopivan lyhyeksi toivotaan, että laskettu approksimaatio eroaa tarkasta ratkaisusta haluttua virherajaa ϵ vähemmän.

Tutkimme seuraavassa Eulerin menetelmässä syntyvää virhettä. Vähennämme tarkan ratkaisun pisteessä t_k muodostetusta Taylorin kehitelmästä (7.12) Eulerin menetelmän (7.13) kaavan, jolloin saamme

$$y(t_{k+1}) - y_{k+1} = y(t_k) - y_k + h[f(t_k, y(t_k)) - f(t_k, y_k)] + h^2 y''(z_k)/2, \quad (7.14)$$

missä $t_k < z_k < t_{k+1}$. Yhtälön vasen puoli ilmaisee *kokonaisvirheen* pisteessä t_{k+1} eli tarkan ja lasketun ratkaisun välisen eron. Juuri tätä kokonaisvirhettä yritetään numeerisessa menetelmässä pitää pienenä. Määritellään kokonaisvirhe $\mathbf{e}_k = \mathbf{y}(t_k) - \mathbf{y}_k$.

Yhtälön (7.14) oikea puoli muodostuu kahdesta virhelähteestä: edellisillä askelilla pisteeseen t_k kasautuneesta kokonaisvirheestä ja viimeisimmällä askeleella syntyneestä paikallisesta virheestä. Jos oletetaan, että edellisellä askeleella osuttiin tarkkaan ratkaisuun $y_k = y(t_k)$, jää yhtälön (7.14) oikealle puolelle vielä termi $h^2 y''(z_k)/2$. Tätä virhettä kutsutaan *paikalliseksi virheeksi* $\mathbf{l}_k$, ja se määritellään yleisessä tapauksessa

$$\mathbf{l}_k = \hat{\mathbf{y}}(t_k) - \mathbf{y}_k,$$

missä $\hat{\mathbf{y}}$ on lokaalin ongelman

$$\begin{aligned} \hat{\mathbf{y}}'(t) &= \mathbf{f}(t, \hat{\mathbf{y}}(t)), \\ \hat{\mathbf{y}}(t_{k-1}) &= \mathbf{y}_{k-1} \end{aligned}$$

tarkka ratkaisu.

Paikallisen virheen $h^2 y''(z_k)/2 = \mathcal{O}(h^2)$ perusteella sanotaan, että Eulerin menetelmä on kertalukua yksi. Menetelmä on kertalukua p, jos paikallinen virhe on $\mathcal{O}(h^{p+1})$. Voidaan myös sanoa, että menetelmän kertaluku on p, jos se löytää *tarkasti* jokaisen korkeintaan p-asteisen polynomimuotoisen ratkaisun. Yleisesti voidaan siis kirjoittaa

$$\mathbf{l}_k = \psi h^{p+1} + \mathcal{O}(h^{p+2}), \tag{7.15}$$

missä p on menetelmän kertaluku ja ψ riippuu muun muassa tarkan ratkaisun $\mathbf{y}$ derivaatoista.

Kokonaisvirheen kasautumisarviota varten linearisoimme lausekkeen (7.14) derivaattafunktion $f(t, y)$ toisen argumentin suhteen, jolloin väliarvolauseen avulla saamme arvion

$$h\left[f(t_k, y(t_k)) - f(t_k, y_k)\right] = h f_y(t_k, \xi)(y(t_k) - y_k),$$

missä derivaatta f_y on määrätty sopivassa pisteessä ξ. Sijoittamalla tämän virhelausekkeeseen (7.14) saamme virheen kasautumisarvion

$$e_{k+1} = (1 + h f_y) e_k + l_{k+1}.$$

Tekijää $(1 + h f_y)$ kutsutaan *virheenvahvistumiskertoimeksi*. Differentiaaliyhtälöryhmän tapauksessa f_y korvataan Jacobin matriisilla J (liite A).

7.2.3 Ratkaisumenetelmien stabiilisuus

Niin kauan kuin ehto $|1 + h f_y| < 1$ pätee, Eulerin menetelmän kokonaisvirhe ei kasva. Jos taas $|1 + h f_y| > 1$, kokonaisvirhe kasvaa joka askeleella ja numeerinen ratkaisu etääntyy tarkasta ratkaisusta.

Tarkastelemme jälleen testitehtävää (7.9). Eulerin menetelmää soveltamalla saamme

$$y_k = y_{k-1} + h\lambda y_{k-1} = (1 + h\lambda) y_{k-1} = \ldots = y(0)(1 + h\lambda)^k.$$

Olkoon nyt $\mathrm{Re}(\lambda) < 0$, jolloin ratkaisut ovat asymptoottisesti stabiileja ja lähestyvät nollaa. Niinpä numeerisen menetelmän antamat approksimaatiot $|y_k|$ eivät saa kasvaa. Eulerin menetelmän tapauksessa on siis vaadittava, että $|1 + h\lambda| \leq 1$.

Numeerisen menetelmän *absoluuttinen stabiilisuusalue* D on se kompleksitason osa D, johon kuuluville luvuille $z = h\lambda$ menetelmä toteuttaa testitehtävää ratkaistaessa

$$|y_{k-1}| \geq |y_k|,$$

kun $k = 1, 2, \ldots$. Absoluuttisen stabiilisuusalueen ja reaaliakselin leikkausta kutsutaan menetelmän *absoluuttiseksi stabiilisuusväliksi*.

Eulerin menetelmän absoluuttinen stabiilisuusalue on siis kompleksitason ympyrä

$$D = \{z \in \mathbb{C} : |1 + z| \leq 1\}$$

ja absoluuttinen stabiilisuusväli on $[-2, 0]$.

Edellä käsiteltiin yksittäistä differentiaaliyhtälöä (7.9). Tarkastellaan seuraavaksi yhtälöryhmää

$$\mathbf{y}' = A\mathbf{y}$$

ja oletetaan, että matriisi A on diagonalisoituva. On helppo osoittaa, että Eulerin menetelmä käyttäytyy stabiilisti, jos askelpituus h toteuttaa ehdon

$$h\lambda_1, h\lambda_2, \ldots, h\lambda_n \in D,$$

missä $\lambda_1, \lambda_2, \ldots, \lambda_n$ ovat matriisin A ominaisarvot.

Esimerkki 7.2.1 Tutkimme tehtävän

$$y' = -100y + 100, \quad y(0) = y_0 \tag{7.16}$$

ratkaisemista Eulerin menetelmällä. Ongelman tarkka ratkaisu on

$$y(t) = (y_0 - 1)e^{-100t} + 1.$$

Ratkaisu on selvästi stabiili. Sovellamme Eulerin menetelmää tehtävään, jolloin saamme

$$y_{k+1} = y_k + h(-100y_k + 100) = (1 - 100h)y_k + 100h.$$

Tämän differenssiyhtälön ratkaisu on

$$y_k = (y_0 - 1)(1 - 100h)^k + 1.$$

Olkoon lähtöarvo $y_0 = 2$, tällöin tarkka ratkaisu

$$y(t) = e^{-100t} + 1 \tag{7.17}$$

putoaa nopeasti lähtöarvosta 2 raja-arvoon 1. Approksimaation

$$y_k = (1 - 100h)^k + 1 \tag{7.18}$$

pitäisi käyttäytyä samoin kuin tarkka ratkaisu (7.17). Kuitenkin jos $h > 0.02$, jono $|y_k|$ lähestyy ääretöntä, kun k kasvaa.

Tehtävälle (7.16) $\lambda = -100$. Niinpä askelpituuden h on toteutettava $h \in (0, 0.02]$, jotta Eulerin menetelmä olisi stabiili.

Vaikka edellä esitettyä teoriaa ei voida yleistää epälineaarisiin tapauksiin

$$\mathbf{y}' = \mathbf{f}(t, \mathbf{y}), \quad t \geq t_0, \quad \mathbf{y}(t_0) = \mathbf{y}_0,$$

on tavallista asettaa numeerisen menetelmän k:nnen askeleen pituudelle h_k ehto

$$h_k\lambda_{k,1}, h_k\lambda_{k,2}, \ldots, h_k\lambda_{k,n} \in D,$$

missä kompleksiluvut $\lambda_{k,1}, \lambda_{k,2}, \ldots, \lambda_{k,n}$ ovat Jacobin matriisin $J_k(\mathbf{y}_k) = \partial\mathbf{f}(t_k, \mathbf{y}_k)/\partial\mathbf{y}$ ominaisarvot. Tämä käytäntö perustuu ajatukseen, että epälineaarisen yhtälön lokaalia käyttäytymistä voidaan yrittää approksimoida yhtälöllä $\mathbf{y}' = \mathbf{y}_k + J_k(\mathbf{y}_k)(\mathbf{y} - \mathbf{y}_k)$. Korostettakoon vielä, että askelpituuden valitseminen tällä perusteella ei tehtävän epälineaarisuuden vuoksi takaa ratkaisumenetelmän stabiilia käyttäytymistä.

Vielä 1950 luvulla tutkijat eivät kiinnittäneet mitään huomiota stabiilisuuteen. W. E. Milne analysoi 1953 siihen aikaan suositun menetelmän

$$\mathbf{y}_{k+2} = \mathbf{y}_k + 2h\mathbf{f}(t_{k+1}, \mathbf{y}_{k+1})$$

(*eksplisiittinen keskipistesääntö*, leap frog) ja havaitsi, että se on epästabiili kaikilla $h > 0$! Tästä huolimatta eksplisiittinen keskipistesääntö on hyvä valinta tietyissä tapauksissa.

Eulerin menetelmässä askelpituus h esiintyy sekä paikallisessa virheessä $h^2y''(\xi)/2$ että virheenvahvistumiskertoimessa $(1 + hf_y)$. Paikallinen virhe pitäisi pitää mahdollisimman pienenä ja vahvistumiskerroin pienempänä kuin 1 valitsemalla askelpituus h oikein. Valitettavasti derivaattoja y'' ja f_y (yhtälöryhmälle Jacobin matriisia J) ei yleensä tunneta. Useat numeeriset ratkaisuohjelmat yrittävät kuitenkin approksimoida niitä. Toista derivaattaa voidaan approksimoida erotusosamäärällä

$$y''_{\text{appr}} \approx \frac{y'_k - y'_{k-1}}{t_k - t_{k-1}}.$$

Vaatimalla että approksimatiivinen paikallinen virhe toteuttaa $|y''_{\text{appr}}|h^2/2 < \epsilon$, saadaan askelpituudelle yläraja

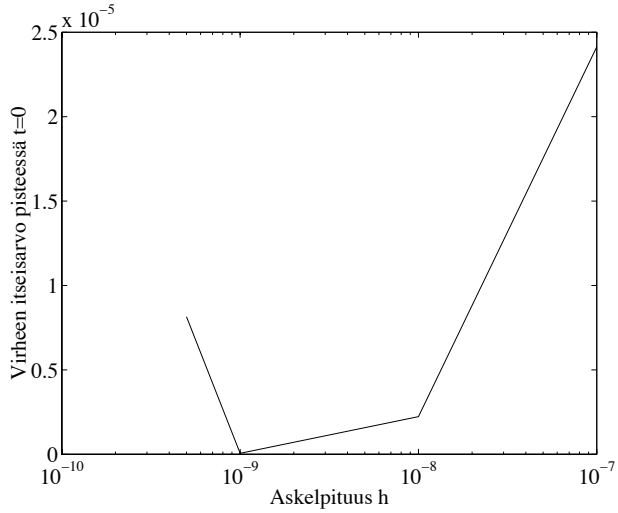
$$h < \sqrt{\frac{2\epsilon}{|y''_{\text{appr}}|}}.$$

Tekniikkaa voidaan soveltaa myös niihin korkeamman kertaluvun menetelmiin, joiden paikallinen virhe on $y^{(p+1)}h^{p+1}/p!$.

7.2.4 Pyöristysvirheet

Askelpituutta pienentämällä voidaan täyttää sekä stabiilisuuden että paikallisen virheen asettamat rajoitukset. Tietokoneella laskettaessa on vielä yksi virhelähde, *pyöristysvirhe*, joka aiheutuu äärellisestä laskentatarkkuudesta. On osoitettavissa, että Eulerin menetelmällä tai muilla numeerisilla algoritmeilla askelpituutta ei voida käytännössä pienentää rajatta, sillä kasautuvat pyöristysvirheet alkavat heikentää tulosta.

Esimerkki 7.2.2 Ratkaisemme tehtävän $y' = -200ty^2$, $y(-1) = 1/101$ Eulerin menetelmällä eri askelpituuksilla viidentoista numeron tarkkuudella. Kuvassa 7.2 on esitetty poikkeama tarkasta ratkaisusta pisteessä $t = 0$ eri askelpituuksilla. Virhe alkaa kasvaa, kun askelpituutta lyhennetään liikaa.



Kuva 7.2: *Virhe alkaa kasvaa, kun askelpituutta lyhennetään liikaa.*

7.2.5 Kankeat differentiaaliyhtälöt

Esimerkki 7.2.3 Tarkastelemme tehtävää

$$u'(t) = -2u(t) + v + 2\sin(t), \tag{7.19}$$

$$v'(t) = 998u(t) - 999v(t) + 999(\cos(t) - \sin(t)), \tag{7.20}$$

$$u(0) = 2, \tag{7.21}$$

$$v(0) = 3. \tag{7.22}$$

Ryhmän tarkka ratkaisu on

$$u(t) = 2\exp(-t) + \sin(t),$$
$$v(t) = 2\exp(-t) + \cos(t).$$

Yritämme ratkaista tehtävän (7.19)–(7.22) välillä $[0, 10]$ käyttäen Eulerin menetelmää adaptiivisella askelpituudella. Olkoon τ tarkkuusvaatimus ja ϵ poikkeama tarkasta ratkaisusta. Jos lasketussa pisteessä pätee $\epsilon > \tau$, palaamme edelliseen pisteeseen ja otamme uuden askeleen puolitetulla askelpituudella. Jos taas $\epsilon < \tau/8$, kaksinkertaistamme askelpituuden.

Asetamme $\tau = 0.01$ ja aloitamme askelpituudella $h = 0.1$ (sadasosa välin pituudesta). Laskettu ratkaisu on kuvassa 7.3. Vaikka kuvaajat ovat sileitä ja rauhallisesti käyttäytyviä, keskimääräinen askelpituus oli 0.0017 eli noin kuudestuhannesosa välin pituudesta. Lyhyt askelpituus on seuraus stabiilisuusvaatimuksesta. Tehtävän kerroinmatriisin ominaisarvot ovat -1 ja -1000, jolloin Eulerin menetelmän askelpituuden h on toteutettava $h < 0.002$.

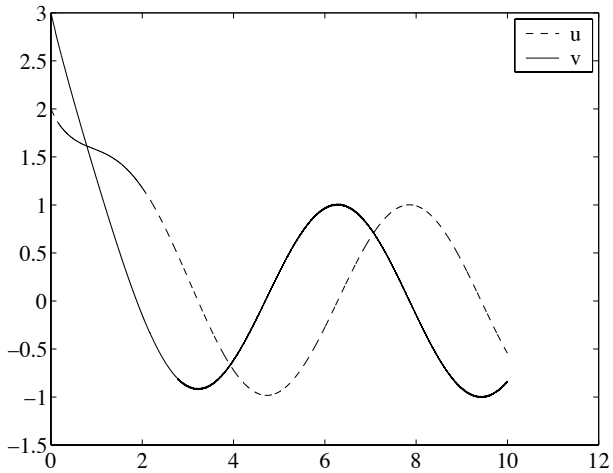
Tehtävällä

$$u'(t) = -2u(t) + v + 2\sin(t), \tag{7.23}$$

$$v'(t) = u(t) - 2v(t) + 2(\cos(t) - \sin(t)) \tag{7.24}$$

on alkuarvoilla (7.21)–(7.22) sama ratkaisu kuin systeemillä (7.19)–(7.22). Tämän tehtävän ratkaisemiseen samoilla parametreilla kuin yllä Eulerin menetelmä tarvitsee keskimääräisen askelpituuden 0.0157, mikä on lähes kymmenkertainen äskeiseen verrattuna. Tässä tapauksessa kerroinmatriisin ominaisarvot ovat -1 ja -3, joten stabiilisuus ei rajoita askelpituutta.

Analyyttiset ratkaisut ovat siis samat, joten pelkästään numeerisen ratkaisun tarkkuuden kannalta saman askelpituuden pitäisi riittää molemmissa tapauksissa. Tehtävässä (7.19)–(7.22) kuitenkin siis askelpituutta on lyhennettävä stabiilisuuden saavuttamiseksi.



Kuva 7.3: *Kankean yhtälösysteemin ratkaisu adaptiivisella Eulerin menetelmällä. Stabiilisuusvaatimus johtaa keskimääräiseen askelpituuteen 0.0017.*

Ongelman (7.19)–(7.22) kaltaisia tehtäviä kutsutaan *kankeiksi* (stiff). Kankeat tehtävät ovat hyvin yleisiä monilla sovellusaloilla, kuten kemiassa, säätöteoriassa, elektroniikassa ja matemaattisessa biologiassa.

Kankeudelle on vaikeaa antaa tarkkaa määritelmää, mutta seuraava luonnehdinta on tavallinen:

Määritelmä 7.2.1 (Kankea systeemi) Tarkastellaan differentiaaliyhtälösysteemiä mielivaltaisilla alkuehdoilla. Jos jokin numeerinen menetelmä, jolla on äärellinen absoluuttisen stabiilisuuden alue, vaatii tehtävää ratkais-

taessa jollain välillä tarkan ratkaisun sileyteen nähden hyvin lyhyen askelpituuden, systeemi on kankea kyseisellä välillä.

Oleellista on siis se, että stabiilisuuden saavuttaminen edellyttää lyhyemmän askelpituuden käyttöä kuin tarkkuusvaatimukset.

Jos ratkaisu halutaan hyvin suurella tarkkuudella, jolloin askelpituus on joka tapauksessa valittava lyhyeksi, mahdollinen kankeus aiheuttaa vähemmän ongelmia.

Kankeudelle on tyypillistä myös se, että ratkaisun eri komponenttien vaimenemisnopeudet poikkeavat voimakkaasti toisistaan.

Esimerkki 7.2.4 Tutkimme yksinkertaista tehtävää

$$\begin{cases} u' &= -2000u + 999.75v + 1000.25, & u(0) = 0, \\ v' &= u - v, & v(0) = -2. \end{cases}$$

Ongelman kerroinmatriisi on

$$A = \begin{bmatrix} -2000 & 999.75 \\ 1 & -1 \end{bmatrix}$$

Matriisin A ominaisarvot ovat $\lambda_1 = -2000.5$, $\lambda_2 = -0.5$ ja tarkka ratkaisu on

$$\begin{aligned} u(t) &= -1.499875e^{-0.5t} + 0.499875e^{-2000.5t} + 1 \\ v(t) &= \underbrace{-2.99975e^{-0.5t}}_{\text{Hitaasti häviävä}} \underbrace{-0.00025e^{-2000.5t}}_{\text{Nopeasti häviävä}} + 1 \end{aligned}$$

Tarkan ratkaisun nopeasti häviävä osa muuttuu merkityksettömän pieneksi hetken $t = 0.002$ jälkeen ja hidas osa häviää vasta hetken $t = 10$ jälkeen. Kun laskemme välillä $0 \le t \le 0.002$, on meidän luonnollisesti käytettävä lyhyttä askelta. Mutta numeerisen stabiilisuuden takia lyhyttä askelpituutta on käytettävä myös, kun $t \ge 0.002$, vaikka ratkaisussa näkyvämpi osa $1 + (\text{kerroin})\ e^{-0.5t}$ sallisi pitemmän askelpituuden h.

Niinpä lineaaristen vakiokertoimisten systeemien tapauksessa mahdollisen kankeuden voi joskus ennakoida kerroinmatriisin ominaisarvojen perusteella.

Huomautus 7.2.1 Lineaarinen vakiokertoiminen systeemi voi olla kankea, jos kaikilla ominaisarvoilla on negatiiviset reaaliosat ja jos *kankeuskerroin*

$$\frac{\max_i |\mathrm{Re}(\lambda_i)|}{\min_i |\mathrm{Re}(\lambda_i)|}$$

on suuri.

Vaikka lineaarista teoriaa ei voi suoraan yleistää epälineaariseen tapaukseen, edellinen luonnehdinta osuu usein oikeaan myös, jos kankeuskerroin lasketaan käyttämällä epälineaarisen tehtävän Jacobin matriisin ominaisarvoja.

Kankeiden tehtävien tehokkaaseen ratkaisemiseen tarvitaan menetelmiä, joilla on paremmat stabiilisuusominaisuudet kuin esimerkiksi Eulerin menetelmällä.

7.2.6 Implisiittiset menetelmät

Kankeisiin tehtäviin onkin etsittävä uuden tyyppisiä menetelmiä. Johdetaan seuraavaksi menetelmä, joka soveltuu tämän kaltaisiin tehtäviin.

Integroidaan yhtälö $\mathbf{y}' = \mathbf{f}(t, \mathbf{y})$ yhden aika-askeleen yli:

$$\mathbf{y}(t_{k+1}) = \mathbf{y}(t_k) + \int_{t_k}^{t_{k+1}} \mathbf{f}(t, \mathbf{y}(t))dt. \tag{7.25}$$

Jos integraalitermiä approksimoidaan lausekkeella $(t_{k+1} - t_k)\mathbf{f}(t_k, \mathbf{y}_k)$, saadaan Eulerin menetelmä. Approksimaationa voidaan kuitenkin käyttää myös lauseketta $(t_{k+1} - t_k)\mathbf{f}(t_{k+1}, \mathbf{y}_{k+1})$, missä integroitavan arvo on otettu välin päätepisteessä. Tulos on *implisiittinen Eulerin menetelmä* (backward Euler method)

$$\mathbf{y}_{k+1} = \mathbf{y}_k + h\mathbf{f}(t_{k+1}, \mathbf{y}_{k+1}). \tag{7.26}$$

Etenemissuuntana käytetään siis tangenttia *uudessa* pisteessä.

Menetelmän implisiittisyys tarkoittaa sitä, että laskettava approksimaatio $\mathbf{y}_{k+1}$ esiintyy myös laskennassa käytettävän derivaattafunktion $\mathbf{f}$ argumenttina, eikä siten yleisessä tapauksessa ole analyyttisesti ratkaistavissa yhtälöstä (7.26). Joka askeleella on siis ratkaistava mahdollisesti epälineaarinen yhtälö.

Teemme saman virheenkasautumistarkastelun menetelmälle (7.26) kuin Eulerin menetelmälle. Tällöin

$$y_{k+1} - y(t_{k+1}) = (1 - hf_y)^{-1}\left([y_k - y(t_k)] + \frac{h^2}{2}y''(\xi_k)\right),$$

josta edelleen saamme:

$$e_{k+1} = (1 - hf_y)^{-1}e_k + l_{k+1}.$$

Implisiittisen Eulerin menetelmän virheenvahvistumiskerroin on siis $(1 - hf_y)^{-1}$.

Implisiittinen Eulerin menetelmä on ensimmäistä kertalukua, kuten tavallinen Eulerin menetelmäkin.

Soveltamalla implisiittistä Eulerin menetelmää testitehtävään (7.9) saamme

$$y_k = (1 - \lambda h)^{-1}y_{k-1} = \ldots = y(0)(1 - \lambda h)^{-k}.$$

Menetelmän absoluuttinen stabiilisuusalue on siis

$$D = \{z \in \mathbb{C} : |1 - z|^{-1} \leq 1\}$$

eli koko kompleksitaso ympyräkiekkoa $|1 - z| < 1$ lukuun ottamatta.

Kun implisiittistä Eulerin menetelmää sovelletaan stabiiliin lineaariseen vakiokertoimiseen tehtävään, jolloin $\mathrm{Re}(\lambda) \leq 0$,

$$|1 - h\lambda|^{-1} \leq 1$$

kaikilla $h > 0$. Menetelmä on siis aina absoluuttisesti stabiili. Stabiilisuusvaatimus ei siten rajoita implisiittisen Eulerin menetelmän askelpituutta, mikä tekee menetelmän sopivaksi kankeisiin tehtäviin.

Menetelmää, jonka absoluuttisen stabiilisuuden alue sisältää vasemman puolitason, sanotaan *A-stabiiliksi*. Tällaista menetelmää käytettäessä askelpituus voidaan siis valita puhtaasti tarkkuusvaatimusten perusteella. Implisiittinen Eulerin menetelmä on A-stabiili.

Esimerkki 7.2.5 Sovellamme implisiittistä Eulerin menetelmää tehtävään (7.16), $y' = -100y + 100$, $y(0) = y_0$, jolloin saamme

$$y_{k+1} = y_k + h(-100y_{k+1} + 100)$$

ja edelleen

$$y_{k+1} = (1 + 100h)^{-1}(y_k + 100h).$$

Alkuarvolla $y_0 = 2$ ratkaisu

$$y_k = \frac{1}{(1 + 100h)^k} + 1$$

käyttäytyy kauniisti kaikilla askelpituuksilla ja y_k lähestyy arvoa 1 samoin kuin tarkka ratkaisukin $y(t) = e^{-100t} + 1$.

Jos integraalia (7.25) approksimoitaessa käytetään keskiarvoa $(\mathbf{f}(t_k, \mathbf{y}_k) + \mathbf{f}(t_{k+1}, \mathbf{y}_{k+1}))/2$, saadaan *trapetsimenetelmä* (trapezoidal method):

$$\mathbf{y}_{k+1} = \mathbf{y}_k + h/2\,[\mathbf{f}(t_{k+1}, \mathbf{y}_{k+1}) + \mathbf{f}(t_k, \mathbf{y}_k)]\,.$$

Trapetsimenetelmä on kertalukua 2 eli paikallinen virhe on $\mathcal{O}(h^3)$. Menetelmän virheenvahvistuskerroin on

$$\frac{1 + 0.5hf_y}{1 - 0.5hf_y}, \tag{7.27}$$

missä osoittajassa ja nimittäjässä esiintyvät osittaisderivaatat f_y on laskettu eri pisteissä.

Testitehtävään (7.9) soveltaminen tuottaa yhtälön

$$y_{k+1} = \frac{2 + h\lambda}{2 - h\lambda} y_k,$$

josta saadaan absoluuttisen stabiilisuuden alueeksi

$$D = \{z \in \mathbb{C} : \frac{|2 + z|}{|2 - z|} \leq 1\} = \{z \in \mathbb{C} : \mathrm{Re}(z) \leq 0\}$$

eli koko vasen puolitaso, mikä tekee trapetsimenetelmästä A-stabiilin. Trapetsimenetelmä soveltuu siis kankeiden tehtävien ratkaisemiseen.

Siitä huolimatta, että trapetsimenetelmä on kertalukua 2 ja periaatteessa tarkempi kuin implisiittinen Eulerin menetelmä, niin erittäin kankeissa tehtävissä trapetsimenetelmä voi käyttäytyä huonommin. Jos nimittäin $h\mathrm{Re}(\lambda) \to -\infty$, niin

$$\left| \frac{2 + h\lambda}{2 - h\lambda} \right| \to 1,$$

jolloin virheet eivät oleellisesti pienene. Vastaavassa tilanteessa implisiittisen Eulerin menetelmän virheenvahvistuskerroin lähestyy nollaa ja kasautuvat virheet pysyvät hyvin aisoissa.

Molemmat menetelmät vaativat approksimaation $\mathbf{y}_{k+1}$ ratkaisemista mahdollisesti epälineaarisesta yhtälöstä

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \beta h \mathbf{f}(t_{k+1}, \mathbf{y}_{k+1}) + \mathbf{g}, \tag{7.28}$$

missä vektori $\mathbf{g}$ ei riipu arvosta $\mathbf{y}_{k+1}$. Epälineaarisen tehtävän ratkaiseminen on kustannus, joka joudutaan maksamaan, kun stabiilisuuden takia on käytettävä implisiittisiä menetelmiä.

Huomautus 7.2.2 Vaikka kaikki A-stabiilit numeeriset menetelmät ovat implisiittisiä, implisiittisyys ei yksin tee menetelmästä A-stabiilia.

7.2.7 Epälineaaristen yhtälöiden ratkaiseminen implisiittisissä menetelmissä

Approksimaation $\mathbf{y}_{k+1}$ ratkaisemiseen yhtälöstä (7.28) voidaan käyttää suoraa iteraatiota

$$\mathbf{y}_{k+1}^{(n+1)} = \mathbf{y}_k + \beta h \mathbf{f}(t_{k+1}, \mathbf{y}_{k+1}^{(n)}) + \mathbf{g}, \quad n = 0, 1, \ldots, \tag{7.29}$$

missä (n) kuvaa kulloistakin iteraatiokierrosta ja alkuarvaus $\mathbf{y}_{k+1}^{(0)}$ on valittu sopivasti. Käytännössä kuitenkin iteraation (7.29) suppenemiseen tarvitaan ehdon

$$h|\beta|\|J\| < 1$$

toteutuminen, missä $J = \partial \mathbf{f}/\partial \mathbf{y}$ on funktion $\mathbf{f}$ Jacobin matriisi. Siten kankeita tehtäviä (joilla usein $\|J\| \gg 1$) ratkaistaessa on askelpituus h valittava pieneksi, jotta suora iteraatio suppenisi. Näin menetetään A-stabiilin implisiittisen menetelmän antama etu.

Tästä syystä joudutaan turvautumaan parempiin menetelmiin eli ratkaistaan $\mathbf{y}_{k+1}$ yhtälöstä

$$0 = \mathbf{F}(\mathbf{y}_{k+1}) \equiv \mathbf{y}_{k+1} - \mathbf{y}_k - \beta h \mathbf{f}(t_{k+1}, \mathbf{y}_{k+1}) - \mathbf{g}$$

esimerkiksi *Newtonin menetelmällä* (kappaleet 6.3.3 ja 6.4.2) eli

$$\mathbf{y}_{k+1}^{(n+1)} = \mathbf{y}_{k+1}^{(n)} - [\mathbf{F}'(\mathbf{y}_{k+1}^{(n)})]^{-1} \mathbf{F}(\mathbf{y}_{k+1}^{(n)}), \quad \mathbf{F}'(\mathbf{y}) = I - \beta h \frac{\partial \mathbf{f}(t_{k+1}, \mathbf{y})}{\partial \mathbf{y}}.$$

Voidaan osoittaa, että Newtonin menetelmä suppenee kaikilla askelpituuksilla h, kunhan $\mathbf{F}'$ on kääntyvä ja lähtöarvaus $\mathbf{y}_{k+1}^{(0)}$ on riittävän hyvä.

Derivaattamatriisin laskeminen on kuitenkin suhteellisen työlästä. Onneksi Newtonin menetelmä yleensä suppenee parilla iteraatiokierroksella. Toisinaan käytetään työn vähentämiseksi *modifioitua Newtonin menetelmää*, jossa derivaattamatriisia ei päivitetä jokaisella iteraatiokierroksella.

Sovellusohjelmistoissa käyttäjä voi usein antaa derivaattamatriisin aliohjelmalla. Derivaattamatriisin muodostamisessa voi tarvittaessa käyttää apuna symbolisen laskennan ohjelmistoja. Differentiaaliyhtälöitä ratkaiseva ohjelmisto osaa yleensä myös approksimoida derivaattamatriisia (kappale 6.4.3 sivulla 202).

7.3 Yksiaskelmenetelmät alkuarvotehtäville

Edellä esitetyt Eulerin menetelmä ja implisiittinen Eulerin menetelmä kuuluvat niin sanottuihin *yksiaskelmenetelmiin*. Samaan luokkaan kuuluvat myös Rungen ja Kuttan menetelmät.

Yksiaskelmenetelmät ovat yleisesti muotoa

$$\begin{aligned} \mathbf{y}_{k+1} &= \mathbf{y}_k + h_{k+1}\Phi_{\mathbf{f}}(t_k, \mathbf{y}_k, t_{k+1}, \mathbf{y}_{k+1}; h_{k+1}), \\ t_{k+1} &= t_k + h_{k+1}, \end{aligned}$$

missä funktiosta $\mathbf{f}$ ja argumenteista t_k, t_{k+1}, $\mathbf{y}_k$, $\mathbf{y}_{k+1}$ ja h_{k+1} riippuvaa funktiota $\Phi_{\mathbf{f}}$ kutsutaan lisäysfunktioksi. Uuden approksimaation $(t_{k+1}, \mathbf{y}_{k+1})$ laskemiseen tarvitaan siis vain edellinen piste $(t_k, \mathbf{y}_k)$.

7.3.1 Rungen ja Kuttan menetelmät

Integroimalla (7.1) saadaan

$$\mathbf{y}(t_{k+1}) = \mathbf{y}(t_k) + \int_{t_k}^{t_{k+1}} \mathbf{f}(\tau, \mathbf{y}(\tau))\, d\tau.$$

Rungen ja Kuttan menetelmiin päädytään korvaamalla edellisen yhtälön integraali sopivilla approksimaatioilla. Yleinen s-vaiheinen Rungen ja Kuttan menetelmä on muotoa

$$\begin{aligned} \mathbf{y}_{k+1} &= \mathbf{y}_k + h \sum_{i=1}^{s} b_i \mathbf{k}_i, \\ \mathbf{k}_i &= \mathbf{f}(t_k + c_i h, \mathbf{y}_k + h \sum_{j=1}^{s} a_{ij} \mathbf{k}_j), \end{aligned}$$

missä

$$c_i = \sum_{j=1}^{s} a_{ij}. \tag{7.30}$$

Menetelmää karakterisoivat painot b_i, solmut c_i ja matriisialkiot a_{ij} määritetään haluttujen ominaisuuksien, kuten suurimman mahdollisen tarkkuuden, perusteella.

Jos $a_{ij} = 0$, kun $j \geq i$, menetelmä on eksplisiittinen eli avoin, ja

$$\begin{aligned}
\mathbf{k}_1 &= \mathbf{f}(t_k, \mathbf{y}_k), \\
\mathbf{k}_2 &= \mathbf{f}(t_k + c_2 h, \mathbf{y}_k + c_2 h \mathbf{k}_1), \\
\mathbf{k}_3 &= \mathbf{f}(t_k + c_3 h, \mathbf{y}_k + (c_3 - a_{32}) h \mathbf{k}_1 + h a_{32} \mathbf{k}_2), \\
&\vdots \quad \vdots
\end{aligned}$$

Tässä käytettiin hyväksi yhtälöä (7.30). Muussa tapauksessa on kyse implisiittisestä menetelmästä.

Varsinkin eksplisiittiset Rungen ja Kuttan menetelmät voidaan tulkita myös seuraavasti. Otetaan pisteestä $(t_k, \mathbf{y}_k)$ yksi Eulerin askel pituudeltaan $c_2 h$. Derivaatta saadussa pisteessä on $\mathbf{k}_2$. Palataan alkuun ja otetaan uusi Eulerin askel (pituus $c_3 h$) käyttäen derivaattojen $\mathbf{k}_1$ ja $\mathbf{k}_2$ painotettua keskiarvoa, jolloin saadaan $\mathbf{k}_3$. Näin lasketaan derivaatalle joukko approksimaatioita, joiden painotettua keskiarvoa käytetään viimeisessä Eulerin askeleessa pisteestä $(t_k, \mathbf{y}_k)$ pisteeseen $(t_{k+1}, \mathbf{y}_{k+1})$. Voidaan siis sanoa, että ennen lopullisen askeleen ottamista Rungen ja Kuttan menetelmät ”tunnustelevat” derivaatan arvoa.

Esimerkki kaksivaiheisesta Rungen ja Kuttan menetelmästä on *Heunin menetelmä:*

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \frac{h}{2} \left[\mathbf{f}(t_k, \mathbf{y}_k) + \mathbf{f}(t_{k+1}, \mathbf{y}_k + h\mathbf{f}(t_k, \mathbf{y}_k)) \right].$$

Heunin menetelmä on toista kertalukua eli paikallinen virhe on $\mathcal{O}(h^3)$.

Tunnetuin Rungen ja Kuttan menetelmä on klassinen nelivaiheinen menetelmä

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \frac{h}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4),$$

jonka apuderivaatat ovat

$$\begin{cases}
\mathbf{k}_1 = \mathbf{f}(t_k, \mathbf{y}_k), & \mathbf{k}_2 = \mathbf{f}(t_k + \frac{h}{2}, \mathbf{y}_k + \frac{h}{2}\mathbf{k}_1), \\
\mathbf{k}_3 = \mathbf{f}(t_k + \frac{h}{2}, \mathbf{y}_k + \frac{h}{2}\mathbf{k}_2), & \mathbf{k}_4 = \mathbf{f}(t_{k+1}, \mathbf{y}_k + h\mathbf{k}_3).
\end{cases}$$

Menetelmän kertaluku on neljä eli paikallinen virhe on $\mathcal{O}(h^5)$.

Edellä esitettyjen kaksi- ja nelivaiheisten menetelmien (kertaluvut kaksi ja neljä) perusteella voisi olettaa, että s-vaiheisen avoimen menetelmän kertaluku olisi s. Valitettavasti kertaluku ei kuitenkaan kasva suorassa suhteessa derivaattojen määrään.

Taulukossa 7.1 on lueteltu tietyn kertaluvun saavuttamiseksi tarvittavien vaiheiden määrä ratkaistaessa *yhtä* differentiaaliyhtälöä.

Taulukko 7.1: *Halutun kertaluvun menetelmän vaiheiden määrä.*

Kertaluku	1	2	3	4	5	6	7	8	9	10
Vaiheita	1	2	3	4	6	7	9	11	$12 \leq s \leq 17$	$13 \leq s \leq 17$

Taulukosta näkee, miksi 4. kertaluvun menetelmä on suosittu: se on tarkin menetelmä, jossa ei tarvita "ylimääräisiä" derivaattoja.

Taulukko 7.1 ei kuitenkaan yleisesti päde differentiaaliyhtälösysteemeille! Jos skalaaritehtävään sovellettavan Rungen ja Kuttan menetelmän kertaluku p on suurempi kuin 4, saman menetelmän kertaluku voi olla alhaisempi systeemiä ratkaistaessa. Jos kertaluku on 4 tai alhaisempi, kertaluvut yleisille skalaaritehtäville $y' = f(t, y)$ ja systeemeille ovat samat. Lisäksi menetelmä, joka saavuttaa kertaluvun 4 tehtävässä $y' = f(y)$, ei välttämättä ole samaa kertalukua yleistä skalaaritehtävää $y' = f(t, y)$ ratkaistaessa.

Kankeisiin tehtäviin sopivat implisiittiset Rungen ja Kuttan menetelmät. Esimerkkinä annettakoon implisiittinen 2-vaihemenetelmä, jonka kertaluku on 4:

$$\begin{cases} \mathbf{y}_{k+1} &= \mathbf{y}_k + \frac{h}{2}(\mathbf{k}_1 + \mathbf{k}_2), \\ \mathbf{k}_1 &= \mathbf{f}(t_k + (\frac{1}{2} + \frac{\sqrt{3}}{6})h, \mathbf{y}_k + \frac{h}{4}\mathbf{k}_1 + (\frac{1}{4} + \frac{\sqrt{3}}{6})h\mathbf{k}_2), \\ \mathbf{k}_2 &= \mathbf{f}(t_k + (\frac{1}{2} - \frac{\sqrt{3}}{6})h, \mathbf{y}_k + (\frac{1}{4} - \frac{\sqrt{3}}{6})h\mathbf{k}_1 + \frac{h}{4}\mathbf{k}_2). \end{cases}$$

Voidaan osoittaa, että kaikilla $s \geq 0$ on olemassa s-vaiheinen implisiittinen Rungen ja Kuttan menetelmä, jonka kertaluku (tarkkuus) on $2s$. Menetelmien stabiilisuusominaisuudet ovat myös hyvät, sillä kaikki s-vaiheiset implisiittiset kertalukua $2s$ olevat menetelmät ovat A-stabiileja.

Implisiittisten menetelmien käyttäminen kuitenkin edellyttää epälineaarisen yhtälöryhmän ratkaisemista joka askeleella, mikä vaatii paljon työtä.

7.3.2 Virheen arviointi laskennan aikana

Useimmat ohjelmistot edellyttävät käyttäjän antavan jonkinlaisen ylärajan sallitulle virheelle. Ratkaisuohjelma voi vaihtaa esimerkiksi askelpituutta tai käytettävää menetelmää virheen pitämiseksi halutun rajan alapuolella. Jotta tähän päästäisiin, ratkaisuohjelman pitää kuitenkin pystyä jotenkin approksimoimaan laskennan aikana syntyviä virheitä.

Yksinkertainen tapa arvioida paikallista virhettä on laskea kahdella askelpituudella. Ottamalla yksi $2h$-askel ja kaksi h-askelta saadaan kaksi estimaattia, joiden erotuksen avulla voidaan approksimoida paikallista virhettä.

Olkoot $\mathbf{y}_k$ ja $\tilde{\mathbf{y}}_k$ ratkaisut, joihin on päästy pisteestä $\mathbf{y}_{k-2}$ kahdella h-aske-

leella ja yhdellä $2h$-askeleella. Jos $\mathbf{y}$ on tarkka ratkaisu, saadaan

$$\begin{aligned}\mathbf{y}(t_k) - \mathbf{y}_k &\approx 2\mathbf{l}_k(h) = 2h^{p+1}\psi + \mathcal{O}(h^{p+2}),\\ \mathbf{y}(t_k) - \tilde{\mathbf{y}}_k &= \mathbf{l}_k(2h) = (2h)^{p+1}\psi + \mathcal{O}(h^{p+2}).\end{aligned}$$

Edellä oletettiin, että kahden askeleen jälkeen paikallinen virhe (7.15) on kaksinkertainen yhden askeleen paikalliseen virheeseen verrattuna. Vähentämällä yhtälöt toisistaan saadaan

$$\|\tilde{\mathbf{y}}_k - \mathbf{y}_k\| \approx 2h^{p+1}(2^p - 1)\|\psi\| + \mathcal{O}(h^{p+2}). \tag{7.31}$$

Yhtälöstä (7.31) voidaan nyt ratkaista $\|\psi\|$, jolloin viimein

$$\|\mathbf{l}_k\| \approx h^{p+1}\|\psi\| \approx \frac{1}{2}\frac{\|\tilde{\mathbf{y}}_k - \mathbf{y}_k\|}{2^p - 1},$$

missä $\|\tilde{\mathbf{y}}_k - \mathbf{y}_k\|$ on siis saatujen ratkaisujen erotus.

Näin saadaan siis yleinen paikallisen virheen arvio, jonka laskeminen kuitenkin on valitettavasti melko kallista. Joillekin Rungen ja Kuttan menetelmille arvio voidaan kuitenkin tehdä huomattavasti edullisemmin.

Paikallisen virheen arvioimiseksi ja askelpituuden säätämiseksi Rungen ja Kuttan menetelmiä käytetään usein pareina: kertalukua p oleva menetelmä tuottaa ratkaisuehdotuksen, ja kertalukua $\tilde{p} \geq p + 1$ olevaa menetelmää käytetään apuna virhearvion laskemisessa. Lähdettäessä samasta pisteestä saadaan

$$\begin{aligned}\mathbf{y}_{k+1} &= \mathbf{y}(t_{k+1}) - \psi h^{p+1} + \mathcal{O}(h^{p+2}),\\ \tilde{\mathbf{y}}_{k+1} &= \mathbf{y}(t_{k+1}) + \mathcal{O}(h^{p+2}).\end{aligned}$$

Yhtälöiden vähentäminen toisistaan antaa relaation $\psi h^{p+1} \approx \tilde{\mathbf{y}}_{k+1} - \mathbf{y}_{k+1}$, joten paikalliselle virheelle saadaan arvio

$$\|\mathbf{y}_{k+1} - \tilde{\mathbf{y}}_{k+1}\|.$$

Suoraviivainen laskeminen samanaikaisesti kahdella Rungen ja Kuttan ratkaisijalla on yleensä erittäin epätaloudellista. Menetelmät voidaan kuitenkin valita niin, että matalamman kertaluvun menetelmän tarvitsemat derivaatat $\mathbf{k}_i$ sisältyvät korkeamman kertaluvun menetelmän käyttämiin derivaattoihin. Tätä menettelyä kutsutaan *upotukseksi* (embedded Runge-Kutta pair).

Tällaisia menetelmiä nimitetään joskus *Rungen, Kuttan ja Fehlbergin menetelmiksi* (RKF). Esimerkkinä mainittakoon Fehlbergin kuusivaiheinen Rungen ja Kuttan pari, jonka kertaluvut ovat neljä ja viisi. Tämä menetelmä on toteutettu monissa ohjelmistoissa. Erinomaisen kuvauksen tästä 26-kertoimisesta menetelmästä saa viitteestä [GFM77].

Saatua tulosta voidaan luonnollisesti yrittää parantaa korjaamalla sitä lasketulla virhearviolla. Rungen ja Kuttan parien tapauksessa tämä tarkoittaa yksinkertaisesti korkeamman kertaluvun ratkaisun käyttämistä lähtöarvona seuraavalle askeleelle. Menettelyä kutsutaan *lokaaliksi ekstrapolaatioksi* (local extrapolation), ja sitä käytetään monissa ohjelmistoissa. Tällaisia Rungen ja Kuttan pareja kutsutaan myös *Dormandin ja Princen pareiksi*.

7.3.3 Yksiaskelmenetelmien vertailu

Seuraavassa käytämme esittelemiämme yksiaskelmenetelmiä tehtävän

$$y' = -5ty^2 + \frac{5}{t} - \frac{1}{t^2}, \quad y(1) = 1 \tag{7.32}$$

ratkaisemiseen välillä $(1, 10)$. Tarkka ratkaisu on $y(t) = 1/t$, jonka avulla saamme laskettua menetelmien virheet.

Taulukossa 7.2 on kunkin menetelmän tuottaman ratkaisun maksimipoikkeama oikeasta vastauksesta eli arvo $\max(|1/t_i - y_i|)$. Taulukossa 7.3 on annettu virheet välin päätepisteessä eli arvot $|1/10 - y_N|$.

Taulukko 7.2: *Yksiaskelmenetelmien maksimivirheet.*

h	*Euler*	*Impl. Euler*	*Trapetsi*	*RK2*	*RK4*
0.2	3.33e-2	9.23e-3	1.39e-3	1.39e-3	5.34e-3
0.1	9.09e-3	5.21e-3	2.83e-4	2.83e-4	2.04e-4
0.05	3.43e-3	2.80e-3	7.01e-5	7.01e-5	9.65e-6
0.02	1.27e-3	1.17e-3	1.11e-5	1.11e-5	2.09e-7
0.01	6.20e-4	5.97e-4	2.77e-6	2.77e-6	1.23e-8
0.005	3.08e-4	3.01e-4	6.92e-7	6.92e-7	7.47e-10
0.002	1.22e-4	1.21e-4	1.11e-7	1.11e-7	1.88e-11

Taulukko 7.3: *Yksiaskelmenetelmien virheet välin päätepisteessä.*

h	*Euler*	*Impl. Euler*	*Trapetsi*	*RK2*	*RK4*
0.2	1.13e-2	1.98e-5	1.92e-7	1.56e-2	1.01e-5
0.1	9.90e-6	1.01e-5	5.00e-8	1.01e-5	3.35e-7
0.05	5.13e-6	5.18e-6	1.30e-8	1.71e-6	1.70e-8
0.02	2.05e-6	2.05e-6	2.07e-9	2.26e-7	3.74e-10
0.01	1.03e-6	1.03e-6	5.19e-10	5.36e-8	2.23e-11
0.005	5.15e-7	5.16e-7	1.30e-10	1.31e-8	1.36e-12
0.002	2.06e-7	2.06e-7	2.08e-11	2.07e-9	3.47e-14

7.4 Lineaariset moniaskelmenetelmät alkuarvotehtävälle

Yksiaskelmenetelmissä ei hyödynnetä aikaisemmin laskettuja ratkaisu- tai derivaatta-approksimaatioita

$$\begin{cases} \mathbf{y}_{k-1}, & \mathbf{y}_{k-2}, & \dots \\ \mathbf{f}(t_{k-1}, \mathbf{y}_{k-1}), & \mathbf{f}(t_{k-2}, \mathbf{y}_{k-2}), & \dots, \end{cases}$$

vaan käytetään vain pisteessä t_k tai t_{k+1} määrättyjä approksimaatioita. Jo laskettuja approksimaatioita voidaan kuitenkin käyttää hyväksi.

Ratkaisuenetelmiä, joissa käytetään hyväksi useampia aikaisemmin laskettuja arvoja kutsutaan *moniaskelmenetelmiksi*. Vanhojen arvojen käyttämisellä tavoitellaan tarvittavaan työmäärään nähden parempaa tarkkuutta.

Merkitään jatkossa $\mathbf{f}_j = \mathbf{f}(t_j, \mathbf{y}_j)$. Yleinen *r-askelinen lineaarinen moniaskelmenetelmä* on muotoa

$$\sum_{j=-r+1}^{1} \alpha_j \mathbf{y}_{k+j} = h \sum_{j=-r+1}^{1} \beta_j \mathbf{f}_{k+j}, \tag{7.33}$$

missä kertoimilla on ehdot $\alpha_1 = 1$, $\alpha_{-r+1} \neq 0$ ja $\beta_{-r+1} \neq 0$.

Menetelmiä voi johtaa helposti esimerkiksi approksimoimalla integraalissa

$$\mathbf{y}(t_{k+1}) = \mathbf{y}(t_k) + \int_{t_k}^{t_{k+1}} \mathbf{f}(t, \mathbf{y}(t))dt$$

integroitavaa funktiota $\mathbf{f}(t, \mathbf{y}(t))$ polynomilla

$$\mathbf{P}(t) = \mathbf{a}_0 + \mathbf{a}_1 t + \mathbf{a}_2 t^2 + \cdots + \mathbf{a}_s t^s$$

ja integroimalla tämän polynomin. Tuntemattomat kertoimet $\mathbf{a}_i$ määrätään tämän jälkeen kiinnittämällä polynomi haluttuihin arvoihin $\mathbf{y}_i$ ja $\mathbf{f}_i$, $i = k + 1, k, k - 1, \ldots$.

Moniaskelmenetelmät ovat edullisia laskennan kannalta, sillä jokaisella askeleella määrätään vain yksi uusi derivaattafunktion $\mathbf{f}(\cdot, \cdot)$ arvo. Laskennan käynnistäminen on kuitenkin hankalampaa kuin yksiaskelmenetelmiä käytettäessä; moniaskelmenetelmälle ei riitä yksi alkuarvo. Tarpeellisten lähtöarvojen $(\mathbf{y}_{k-1}, \mathbf{y}_{k-2}, \ldots)$ ja niiden derivaattojen $(\mathbf{f}_{k-1}, \mathbf{f}_{k-2}, \ldots)$ laskemiseen käytetään usein esimerkiksi sopivaa Rungen ja Kuttan menetelmää. Alkuarvojen laskennassa käytettävässä menetelmässä on kuitenkin yleensä valittava huomattavasti pienempi askelpituus kuin varsinaisessa moniaskelalgoritmissa, jotta saavutettaisiin riittävä tarkkuus.

Myös askelpituuden vaihtaminen moniaskelmenetelmissä on ongelmallista. Vanhat arvot on kyettävä muuntamaan uutta askelpituutta vastaaviksi. Moniaskelmenetelmiä käyttävissä ohjelmistoissa sovelletaan askelpituuden vaihtamiseen esimerkiksi ekstrapolointia tai ns. Nordsieckin tekniikkaa tavoitteena mahdollisimman vähäinen lisätyö.

Moniaskelmenetelmät jaetaan eksplisiittisiin eli avoimiin ($\beta_1 = 0$) ja implisiittisiin eli suljettuihin ($\beta_1 \neq 0$). Implisiittisissä on mukana derivaattalauseke $\mathbf{f}(t_{k+1}, \mathbf{y}_{k+1})$. Eksplisiittisistä moniaskelmenetelmistä tunnetuimpia ovat Adamsin ja Bashforthin menetelmät ja implisiittisistä Adamsin ja Moultonin menetelmät sekä backward-menetelmät.

7.4.1 Adamsin ja Bashforthin menetelmät

Yleinen r-askelinen *Adamsin ja Bashforthin menetelmä* on muotoa

$$\mathbf{y}_{k+1} = \mathbf{y}_k + h \sum_{j=-r+1}^{0} \beta_j \mathbf{f}_{k+j}.$$

Tuntematon uusi approksimaatio $\mathbf{y}_{k+1}$ ei esiinny yhtälön oikealla puolella, joten kyseessä on avoin menetelmä.

Seuraavassa on viisi ensimmäistä Adamsin ja Bashforthin menetelmää:

$$\begin{cases} \mathbf{y}_{k+1} = \mathbf{y}_k + h\mathbf{f}_k, \\ \mathbf{y}_{k+1} = \mathbf{y}_k + \frac{h}{2}[3\mathbf{f}_k - \mathbf{f}_{k-1}], \\ \mathbf{y}_{k+1} = \mathbf{y}_k + \frac{h}{12}[23\mathbf{f}_k - 16\mathbf{f}_{k-1} + 5\mathbf{f}_{k-2}], \\ \mathbf{y}_{k+1} = \mathbf{y}_k + \frac{h}{24}[55\mathbf{f}_k - 59\mathbf{f}_{k-1} + 37\mathbf{f}_{k-2} - 9\mathbf{f}_{k-3}], \\ \mathbf{y}_{k+1} = \mathbf{y}_k + \frac{h}{720}[1901\mathbf{f}_k - 2774\mathbf{f}_{k-1} + 2616\mathbf{f}_{k-2} - 1274\mathbf{f}_{k-3} + 251\mathbf{f}_{k-4}]. \end{cases}$$

Yksiaskelinen Adamsin ja Bashforthin menetelmä on tuttu Eulerin menetelmä.

Tarkkuudeltaan ensimmäinen menetelmä on kertalukua yksi ja viimeinen kertalukua viisi. Voidaan osoittaa, että r-askelinen Adamsin ja Bashforthin menetelmä on kertalukua r. Adamsin ja Bashforthin menetelmien absoluuttiset stabiilisuusvälit ovat erittäin pienet. Suurin väli $[-2, 0]$ on ensimmäisellä algoritmilla eli Eulerin menetelmällä. Menetelmät eivät siten sovellu kankeisiin tehtäviin.

7.4.2 Adamsin ja Moultonin menetelmät

Kankeisiin tehtäviin sopivat paremmin implisiittiset *Adamsin ja Moultonin menetelmät*. Yleinen r-askelinen Adamsin ja Moultonin menetelmä on muotoa

$$\mathbf{y}_{k+1} = \mathbf{y}_k + h \sum_{j=-r+2}^{1} \beta_j \mathbf{f}_{k+j}.$$

Koska summaus ulottuu nyt ykköseen asti, menetelmä on implisiittinen.

Viisi ensimmäistä Adamsin ja Moultonin menetelmää ovat:

$$\begin{cases} \mathbf{y}_{k+1} = \mathbf{y}_k + h\mathbf{f}_{k+1}, \\ \mathbf{y}_{k+1} = \mathbf{y}_k + \frac{h}{2}[\mathbf{f}_{k+1} + \mathbf{f}_k], \\ \mathbf{y}_{k+1} = \mathbf{y}_k + \frac{h}{12}[5\mathbf{f}_{k+1} + 8\mathbf{f}_k - \mathbf{f}_{k-1}], \\ \mathbf{y}_{k+1} = \mathbf{y}_k + \frac{h}{24}[9\mathbf{f}_{k+1} + 19\mathbf{f}_k - 5\mathbf{f}_{k-1} + \mathbf{f}_{k-2}], \\ \mathbf{y}_{k+1} = \mathbf{y}_k + \frac{h}{720}[251\mathbf{f}_{k+1} + 646\mathbf{f}_k - 264\mathbf{f}_{k-1} + 106\mathbf{f}_{k-2} - 19\mathbf{f}_{k-3}]. \end{cases}$$

Kaksi ensimmäistä menetelmää, jotka ovat yksiaskelmenetelmiä, vastaavat implisiittistä Eulerin menetelmää (kertaluku yksi) ja trapetsimenetelmää (kertaluku kaksi). Kun $r \geq 2$, r-askelinen Adamsin ja Moultonin menetelmä on kertalukua $r + 1$.

Menetelmien absoluuttiset stabiilisuusvälit nähdään seuraavasta taulukosta:

Askeleita	*Abs. stabiilisuusväli*
1	$[-\infty,\ 0]$
2	$[-6.0, 0]$
3	$[-3.0, 0]$
4	$[-1.8, 0]$

Stabiilisuusväli kuitenkin pienenee nopeasti, eivätkä menetelmät sovellu kuin lievästi kankeisiin tehtäviin.

7.4.3 Ennustaja–korjaaja-menetelmät

Adamsin ja Moultonin menetelmät tuottavat tyypillisesti tarkempia estimaatteja ratkaisulle kuin Adamsin ja Bashforthin menetelmät. Paremman tarkkuuden hinta on kuitenkin implisiittisen menetelmän edellyttämä yhtälöryhmän ratkaiseminen, mikä vaatii paljon laskentaa.

Monissa differentiaaliyhtälöiden ratkaisuohjelmistoissa käytetään Adamsin ja Bashforthin sekä Adamsin ja Moultonin menetelmiä *ennustaja–korjaaja-pareina* (predictor-corrector method). Avoin Adamsin ja Bashforthin menetelmä toimii *ennustajana* ja tuottaa alkuarvon, jota *korjataan* suljetulla Adamsin ja Moultonin menetelmällä. Vaikka korjaus voidaan tehdä useita kertoja, iteraatiota ei jatketa suppenemiseen asti. Näin laskutyön määrä pysyy kohtuullisena ja ei-kankeita tehtäviä ratkaistaessa saavutetaan hyvä tarkkuus. Vaikka korjaaja on suljettu, menetelmä on kokonaisuudessaan avoin.

Esimerkiksi kolmannen kertaluvun pari on

$$\begin{cases} \mathbf{y}^E_{k+1} &= \mathbf{y}_k + \dfrac{h}{12}[23\mathbf{f}_k - 16\mathbf{f}_{k-1} + 5\mathbf{f}_{k-2}], \\ \mathbf{y}_{k+1} = \mathbf{y}^K_{k+1} &= \mathbf{y}_k + \dfrac{h}{12}[5\mathbf{f}(t_{k+1}, \mathbf{y}^E_{k+1}) + 8\mathbf{f}_k - 1\mathbf{f}_{k-1}], \end{cases}$$

missä $\mathbf{y}^E_{k+1}$ on avoimella menetelmällä tuotettu ennuste ja $\mathbf{y}^K_{k+1}$ suljetulla menetelmällä laskettu korjaus.

Hyvät ennustaja–korjaaja-toteutukset osaavat tarvittaessa vaihtaa sekä askelpituutta että kertalukua, ja niissä voidaan myös hyödyntää kappaleessa 7.3.2 mainittua tarkkuutta parantavaa lokaalia ekstrapolaatiota.

7.4.4 Backward-moniaskelmenetelmät

Kankeisiin ongelmiin parhaita menetelmiä ovat *backward-menetelmät* (backward differentiation formulae, BDF). Toisin kuin aikaisemmin esitellyt menetelmät, jotka johdettiin differentiaaliyhtälön integroidusta muodosta lähtien, backward-menetelmät johdetaan vaatimalla, että aikaisempiin pisteisiin sovitettu interpolointipolynomi toteuttaa *differentiaaliyhtälön* ainakin yhdessä pisteessä.

Backward-menetelmät ovat suljettuja. Ne ovat yleisesti muotoa

$$\sum_{j=0}^{r} \alpha_j \mathbf{y}_{k+j} = h\beta_r \mathbf{f}_{k+r}.$$

BDF-menetelmien kertaluku on sama kuin vaiheiden lukumäärä r, mutta menetelmät ovat käyttökelpoisia vain kun $1 \leq r \leq 6$.

Seuraavassa taulukossa on viisi ensimmäistä backward-menetelmää

$$\begin{cases} \mathbf{y}_{k+1} = \mathbf{y}_k + h\mathbf{f}_{k+1}, \\ \mathbf{y}_{k+1} = 4/3\mathbf{y}_k - 1/3\mathbf{y}_{k-1} + 2h/3\mathbf{f}_{k+1}, \\ \mathbf{y}_{k+1} = 18/11\mathbf{y}_k - 9/11\mathbf{y}_{k-1} + 2/11\mathbf{y}_{k-2} + 6h/11\mathbf{f}_{k+1}, \\ \mathbf{y}_{k+1} = 48/25\mathbf{y}_k - 36/25\mathbf{y}_{k-1} + 16/25\mathbf{y}_{k-2} - 3/25\mathbf{y}_{k-3} + 12h/25\mathbf{f}_{k+1}, \\ \mathbf{y}_{k+1} = 300/137\mathbf{y}_k - 300/137\mathbf{y}_{k-1} + 200/137\mathbf{y}_{k-2} - 75/137\mathbf{y}_{k-3} \\ \qquad\qquad + 12/137\mathbf{y}_{k-4} + 60h/137\mathbf{f}_{k+1}. \end{cases}$$

Menetelmissä käytetään vanhoja $\mathbf{y}_{k-i}$ -arvoja ja derivaatta määrätään vain askeleen päätepisteessä t_{k+1}. Näin minimoidaan kankeissa tehtävissä derivaattafunktion $\mathbf{f}$ voimistava vaikutus approksimaatioiden $\mathbf{y}_{k-i}$ virheisiin.

Ensimmäisen ja toisen kertaluvun menetelmät ovat A-stabiileja eli absoluuttinen stabiilisuusalue sisältää kompleksitason koko vasemman puoliskon. Korkeamman kertaluvun menetelmien stabiilisuusalue kutistuu imaginaarisuunnassa, mutta absoluuttinen stabiilisuusväli säilyy äärettömänä ($[-\infty, 0]$).

7.4.5 Moniaskelmenetelmien vertailu

Ratkaisemme esimerkkitehtävän (7.32) nyt moniaskelmenetelmillä. Adamsin ja Bashforthin menetelmistä ovat mukana toisen (AB2) ja neljännen (AB4) kertaluvun menetelmät. Adamsin ja Moultonin menetelmistä on mukana vain neljännen kertaluvun menetelmä (AM4), koska implisiittinen Eulerin menetelmä ja trapetsimenetelmä ovat jo mukana taulukoissa 7.2 ja 7.3. Tarvittavat alkuarvot on kussakin tapauksessa laskettu saman menetelmäperheen alemman kertaluvun menetelmällä.

Taulukossa 7.4 on menetelmien virheet välin päätepisteessä, $|1/10 - y_N|$. Maksimivirheiden tarkastelu ei ole mielekästä, koska alkuarvojen tarkkuus vaikuttaa niihin ratkaisevasti. Neljännen kertaluvun BDF-menetelmän ja Adamsin ja Moultonin menetelmän tapauksissa pienillä askelpituuksilla poikkeama oikeasta ratkaisusta alkaa kasvaa pyöristysvirheiden vuoksi.

Taulukko 7.4: *Moniaskelmenetelmien virheet välin päätepisteessä.*

h	*AB2*	*AB4*	*AM4*	*BDF2*	*BDF4*
0.2	Epästabiili	Epästabiili	8.87e-10	2.07e-8	1.67e-11
0.1	6.38e-3	Epästabiili	3.33e-11	5.27e-9	1.05e-12
0.05	6.59e-8	3.94e-2	2.16e-12	1.31e-9	6.42e-14
0.02	1.04e-8	7.17e-13	5.38e-14	2.08e-10	1.62e-15
0.01	2.60e-9	4.47e-14	3.39e-15	5.21e-11	4.86e-17
0.005	6.52e-10	2.96e-15	5.55e-17	1.30e-11	6.25e-17
0.002	1.04e-10	3.89e-16	3.89e-16	2.07e-12	1.74e-14

7.5 Reuna-arvotehtävät

Tarkastelemme yhtälöä

$$\mathbf{y}' = \mathbf{f}(t, \mathbf{y}), \tag{7.34}$$

jossa vektoreissa $\mathbf{y}$ ja $\mathbf{f}(t, \mathbf{y})$ on n komponenttia. Lisäksi ratkaisua sitoo n ehtoa tarkasteluvälin päätepisteissä:

$$\mathbf{g}(\mathbf{y}(a), \mathbf{y}(b)) = \mathbf{0}. \tag{7.35}$$

Tässä $\mathbf{g}$ on annettu reunaehtokuvaus, joka on yleisesti epälineaarinen. Lineaariset reunaehdot ovat muotoa

$$B_a\mathbf{y}(a) + B_b\mathbf{y}(b) = \mathbf{b}, \tag{7.36}$$

missä B_a ja B_b ovat annettuja matriiseja ja $\mathbf{b}$ on tunnettu vektori. Jos yhtälö (7.36) voidaan kirjoittaa muotoon

$$\begin{aligned} \tilde{B}_a\mathbf{y}(a) &= \mathbf{b}_a, \\ \tilde{B}_b\mathbf{y}(b) &= \mathbf{b}_b \end{aligned}$$

sanotaan, että (lineaariset) reunaehdot *separoituvat*.

Toisinaan sekä differentiaaliyhtälössä että reunaehdoissa on mukana parametri:

$$\begin{aligned} \mathbf{y}' &= \mathbf{f}(t, \mathbf{y}, \lambda), \\ \mathbf{g}(\mathbf{y}(a), \mathbf{y}(b), \lambda) &= 0. \end{aligned}$$

Nyt halutaan määrittää ne parametrin λ arvot, joilla systeemillä on ei-triviaaleja ratkaisuja. Näitä parametrin arvoja sanotaan *ominaisarvoiksi*, ja vastaavat ratkaisut ovat *ominaisfunktioita*.

Esimerkkinä tällaisesta ominaisarvotehtävästä mainittakoon Sturmin ja Liouvillen ongelma, joka on tärkeä tutkittaessa esimerkiksi jousien värähtelyjä,

alkeishiukkasten vuorovaikutuksia ja maankuoren liikkeitä. Sturmin ja Liouvillen tehtävän yleinen muoto välillä (a, b) on

$$\begin{aligned} -(pu')' + qu &= \lambda ru, \\ c_1 u(a) + p(a)u'(a) &= 0, \\ c_2 u(b) - p(b)u'(b) &= 0. \end{aligned}$$

Reuna-arvotehtävien ja niiden ratkaisumenetelmien stabiilisuuden analysointi jätetään tässä käsittelemättä, koska se on hankalampaa kuin alkuarvotehtävien tapauksessa. Asiaa käsitellään perusteellisesti esimerkiksi teoksessa [AMR95].

Seuraavaksi esittelemme tavallisimmat reuna-arvotehtävien ratkaisemiseen käytettävät numeeriset menetelmät.

7.5.1 Tähtäysmenetelmä

Tähtäysmenetelmässä (shooting method) käytetään yksinkertaisella tavalla hyväksi alkuarvotehtäviä varten kehitettyjä menetelmiä. Idea on seuraava: yritetään valita alkuarvot (esimerkiksi tykin putken korotuskulma ja ammuksen lähtönopeus) siten, että ratkaisulla on haluttu arvo välin päätepisteessä (osutaan maaliin).

Olkoon alkuarvotehtävän

$$\begin{aligned} \mathbf{y}' &= \mathbf{f}(t, \mathbf{y}), \\ \mathbf{y}(a) &= \mathbf{s} \end{aligned}$$

ratkaisu $\mathbf{y}(t) = \mathbf{y}(t; \mathbf{s})$, missä merkinnällä halutaan korostaa ratkaisun riippuvuutta alkuarvosta $\mathbf{s}$. Tähtäysmenetelmässä alkuarvo $\mathbf{s}$ pyritään valitsemaan siten, että ratkaisu $\mathbf{y}(t; \mathbf{s})$ toteuttaa halutut reunaehdot

$$\mathbf{g}(\mathbf{y}(a; \mathbf{s}), \mathbf{y}(b; \mathbf{s})) = 0.$$

Alkuperäisen ongelman ratkaisu on siis $\mathbf{y}(t; \mathbf{s}^*)$, missä $\mathbf{s}^*$ on yhtälösysteemin

$$\mathbf{F}(\mathbf{s}) = \mathbf{g}(\mathbf{s}, \mathbf{y}(b; \mathbf{s})) = 0 \tag{7.37}$$

ratkaisu.

Arvon $\mathbf{F}(\mathbf{s})$ laskemiseksi alkuarvotehtävä on ratkaistava jollain sopivalla menetelmällä. Yhtälön (7.37) ratkaisemiseen voidaan sitten periaatteessa käyttää mitä tahansa epälineaariselle yhtälöryhmälle sopivaa menetelmää. Hyviä vaihtoehtoja ovat esimerkiksi sekanttimenetelmä ja Broydenin sekanttimenetelmä (luku 6.4.4).

Newtonin menetelmää (luku 6.3.3) käytettäessä on tunnettava derivaattamatriisi

$$\partial \mathbf{F}/\partial \mathbf{s} = \partial \mathbf{g}/\partial \mathbf{y}_1 + \partial \mathbf{g}/\partial \mathbf{y}_2 Z(b; \mathbf{s}),$$

missä $Z(t;\mathbf{s}) = \partial \mathbf{y}(t;\mathbf{s})/\partial \mathbf{s}$ on alkuarvotehtävän

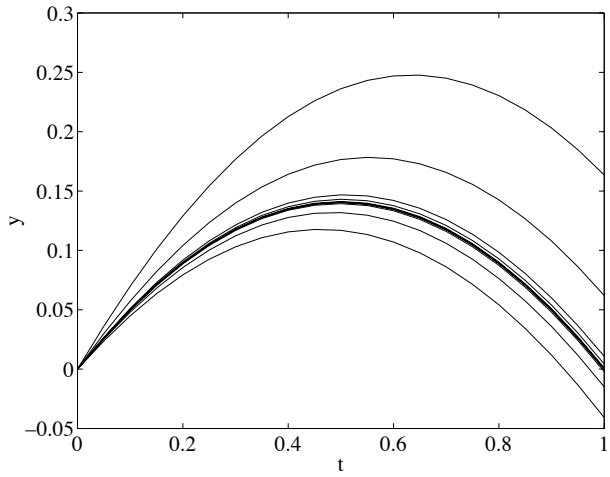
$$Z' = \partial \mathbf{f}/\partial \mathbf{y}(t, \mathbf{y}(t;\mathbf{s}))Z, \quad Z(a;\mathbf{s}) = I$$

ratkaisu. Käytännössä kuitenkin tyydytään usein approksimoimaan derivaattoja erotusosamäärillä

$$\partial \mathbf{F}/\partial s_i \approx \frac{F(s_i + \Delta s_i) - F(s_i)}{\Delta s_i},$$

joissa Δs_i valitaan riittävän pieneksi. Erotusosamäärien käyttämisestä huolimatta Newtonin menetelmä vaatii paljon laskentaa.

Esimerkki 7.5.1 Ratkaisemme tehtävän $y'' + e^y = 0$, $y(0) = 0$, $y(1) = 0$ tähtäysmenetelmällä. Yhtälön ratkaisemiseksi alkuarvotehtävänä tarvitsemme arvon $y'(0)$ eli derivaatan alkupisteessä. Tarkoituksemme on siis löytää derivaatalle arvo, jolla $y(1) = 0$. Etsimme oikean derivaatan $y'(0)$ käyttämällä yksinkertaista puolitushakua. Prosessin tuottamat ratkaisukäyrät ovat kuvassa 7.4. Oikea arvo $y'(0) \approx 0.5493$.



Kuva 7.4: *Tähtäysmenetelmä reuna-arvotehtävän ratkaisemisessa.*

7.5.2 Monimaalimenetelmä

Tähtäysmenetelmä ei ole kuitenkaan sopiva kaikkiin tehtäviin. Jos alkuarvotehtävän ratkaisu on hyvin herkkä alkuarvoille, tähtäysmenetelmä toimii huonosti. Lisäksi jopa lineaarisissa tapauksissa ratkaistavat alkuarvotehtävät voivat olla epästabiileja, vaikka alkuperäinen reuna-arvotehtävä käyttäytyy hyvin. Epälineaarissa tehtävissä taas alkuarvotehtävän ratkaisu ei välttämättä ole olemassa koko tarkasteluvälillä, jos käytetään väärää alkuarvoa.

Koska edellä mainittujen tekijöiden vaikutus ratkaisuun kasvaa integrointivälin pidentyessä, tähtäysmenetelmää voidaan parantaa jakamalla tarkasteluväli riittävän lyhyisiin osiin. Tähtäysmenetelmää sovelletaan kullakin osavälillä ja vaaditaan, että osaongelmien ratkaisut liittyvät jatkuvasti toisiinsa. Menetelmää kutsutaan *monimaalimenetelmäksi* (multiple shooting method).

Jaetaan tarkasteluväli $[a, b]$ osiin

$$a = t_1 < t_2 < \cdots < t_m = b.$$

Olkoon $\mathbf{y}(t; t_k, \mathbf{s}_k)$ alkuarvo-ongelman

$$\mathbf{y}' = \mathbf{f}(t, \mathbf{y}), \quad \mathbf{y}(t_k) = \mathbf{s}_k$$

ratkaisu välillä $[t_k, t_k + 1]$. Monimaalimenetelmässä haetaan siis vektorit $\mathbf{s}_k$, $k = 1, 2, \ldots, m-1$, jotka toteuttavat jatkuvuus- ja reunaehdot

$$\begin{aligned} \mathbf{y}(t_{k+1}; t_k, \mathbf{s}_k) &= \mathbf{s}_{k+1}, \quad k = 1, 2, \ldots, m-2 \\ g(\mathbf{s}_1, \mathbf{y}(b; t_{m-1}, \mathbf{s}_{m-1})) &= 0. \end{aligned}$$

Tuntemattomat alkuarvot

$$\mathbf{s} = \begin{bmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \\ \vdots \\ \mathbf{s}_{m-1} \end{bmatrix}$$

saadaan siis ratkaisemalla epälineaarinen systeemi

$$\mathbf{F}(\mathbf{s}) = \begin{bmatrix} \mathbf{y}(t_2; t_1, \mathbf{s}_1) - \mathbf{s}_2 \\ \mathbf{y}(t_3; t_2, \mathbf{s}_2) - \mathbf{s}_3 \\ \vdots \\ \mathbf{g}(\mathbf{s}_1, \mathbf{y}(t_m; t_{m-1}, \mathbf{s}_{m-1})) \end{bmatrix} = \mathbf{0}$$

esimerkiksi Newtonin menetelmällä (kappale 6.4.2 sivulla 201).

Monimaalimenetelmän ominaisuuksia voidaan vielä parantaa *marssitekniikalla* (marching), jossa alkuarvotehtäviä ei ratkaista yhtä aikaa, vaan lähtien tarkasteluvälin alkupisteestä. Samalla voidaan kontrolloida ratkaisua ja valita seuraava lähtöpiste sopivasti.

7.5.3 Differenssimenetelmä

Differenssimenetelmä (finite difference method) ei pohjaudu alkuarvotehtävien ratkaisemiseen kuten edellä esitetyt menetelmät, vaan reuna-arvotehtävää tarkastellaan diskreetissä pisteistössä ja derivaatat korvataan sopivilla erotusosamäärillä. Näin saadaan alkuperäistä tehtävää approksimoiva algebrallinen yhtälöryhmä.

Esimerkki 7.5.2 Esittelemme differenssimenetelmän yksinkertaisen tehtävän

$$y'' = f(t, y, y'), \quad y(a) = \alpha, \quad y(b) = \beta \tag{7.38}$$

avulla.

Jaamme välin $[a, b]$ m:ään yhtä suureen osaan siten, että $h = (b-a)/m$, $t_i = a + ih$, $i = 0, \ldots, m$. Olkoon y_i tarkan ratkaisun $y(t_i)$ approksimaatio. Korvaamme yhtälössä derivaatat approksimaatioilla

$$y'(t_n) \approx \frac{y_{n+1} - y_{n-1}}{2h}, \quad y''(t_n) \approx \frac{y_{n+1} - 2y_n + y_{n-1}}{h^2},$$

jolloin saamme differenssiyhtälöryhmän

$$y_{n+1} - 2y_n + y_{n-1} = h^2 f\left(t_n, y_n, \frac{y_{n+1} - y_{n-1}}{2h}\right), \quad n = 1, 2, \ldots, m-1,$$
$$y_0 = \alpha, \quad y_m = \beta.$$

Tämä on matriisimuodossa

$$A\mathbf{y} = h^2 \mathbf{f}(\mathbf{y}),$$

missä

$$A = \begin{pmatrix} -2 & 1 & 0 & \ldots & 0 & 0 \\ 1 & -2 & 1 & \ldots & 0 & 0 \\ 0 & 1 & -2 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & -2 & 1 \\ 0 & 0 & 0 & \ldots & 1 & -2 \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{m-1} \end{pmatrix}$$

ja

$$\mathbf{f}(\mathbf{y}) = \begin{pmatrix} f(t_1, y_1, (y_2 - y_0)/(2h)) - \alpha/h^2 \\ f(t_2, y_2, (y_3 - y_1)/(2h)) \\ f(t_3, y_3, (y_4 - y_2)/(2h)) \\ \vdots \\ f(t_{m-1}, y_{m-1}, (y_m - y_{m-2})/(2h)) - \beta/h^2 \end{pmatrix}.$$

Tämän epälineaarisen yhtälöryhmän voimme ratkaista esimerkiksi Newtonin menetelmällä (kappale 6.4.2).

Tarkastellaan nyt differenssimenetelmää hieman yleisemmin ja sovelletaan sitä tehtävään (7.34)–(7.35). Tavallisimmat diskretointimenetelmät ovat *trapetsisääntö*

$$\frac{\mathbf{y}_{i+1} - \mathbf{y}_i}{h_i} = \frac{1}{2}(\mathbf{f}(t_{i+1}, \mathbf{y}_{i+1}) + \mathbf{f}(t_i, \mathbf{y}_i))$$

ja *keskipistesääntö*

$$\frac{\mathbf{y}_{i+1} - \mathbf{y}_i}{h_i} = \mathbf{f}(t_{i+1/2}, \frac{1}{2}(\mathbf{y}_{i+1} + \mathbf{y}_i)),$$

missä $t_{i+1/2} = t_i + h_i/2$. Molemmissa tapauksissa vaaditaan luonnollisesti vielä reunaehtojen toteutuminen:

$$\mathbf{g}(\mathbf{y}_1, \mathbf{y}_{N+1}) = 0.$$

Ratkaisun arvot hilapisteissä saadaan ratkaisemalla muodostettu epälineaarinen yhtälöryhmä. Huomautettakoon, että esimerkiksi Newtonin menetelmän suppeneminen perustuu riittävän hyvään alkuarvaukseen, jonka määrittäminen ei aina ole helppoa.

Jos **f** ja **g** ovat riittävän säännöllisiä, differenssimenetelmä suppenee. Approksimaatio siis lähestyy tarkkaa ratkaisua, kun hilaa tihennetään.

Trapetsi- ja keskipistesääntö ovat yksinkertaisina helppoja soveltaa, mutta ne ovat vain toista kertalukua eli virhe on $\mathcal{O}(h^2)$. Tarkkuutta voidaan parantaa joko käyttämällä korkeamman kertaluvun diskretointimenetelmiä, kuten Rungen ja Kuttan kaavoja, tai nostamalla yksinkertaisen menetelmän kertalukua suppenemista kiihdyttämällä.

Ohjelmistoissa haluttu tarkkuus saavutetaan useimmiten jollain kiihdytysmenetelmällä, joista tavallisin on *tarkentuvien korjausten* (deferred corrections) menetelmä. Perusmenetelmällä lasketun ratkaisun tarkkuutta parannetaan iteratiivisesti laskemalla lokaalin katkaisuvirheen vähitellen tarkentuvia arvioita käyttäen interpolaatiopolynomeja. Parannetut ratkaisut voidaan laskea käyttäen alkuperäistä pisteistöä ja derivaattojen diskretointeja, mikä vähentää tarvittavaa työtä.

Suppenemista on mahdollista kiihdyttää myös käyttämällä ekstrapolointitekniikkaa, jossa tarkempi ratkaisu saadaan ottamalla lineaarikombinaatio tihentyvissä hiloissa perusmenetelmällä lasketuista ratkaisuista.

Erityyppisiin reunarvotehtäviin on kehitetty useita differenssimenetelmiä (katso viitettä [GS80]). Esimerkiksi tehtävätyyppiin $y'' = f(t, y)$ sovelletaan tavallisesti *Cowellin menetelmää* (compact differences)

$$y_{n+1} - 2y_n + y_{n-1} = \frac{h^2}{12}\left[f_{n+1} + 10f_n + f_{n-1}\right],$$

missä $f_n = f(t_n, y_n)$.

Differenssimenetelmää käytetään myös osittaisdifferentiaaliyhtälöiden numeeriseen ratkaisemiseen.

7.5.4 Äärellisulotteisiin aliavaruuksiin pohjautuvat menetelmät

Etsitään reuna-arvotehtävälle muotoa

$$x(t) = \sum_{i=1}^{N} c_i \phi_i(t) \tag{7.39}$$

olevaa ratkaisua, missä $\{\phi_1, \phi_2, \ldots, \phi_N\}$ on sopivasti valittu joukko lineaarisesti riippumattomia kantafunktioita ja kertoimet $\{c_1, c_2, \ldots, c_N\}$ ovat tuntemattomia. Valitsemalla kantafunktiot tarkoituksenmukaisesti taataan reunaehtojen toteutuminen. Eri kriteerit kertoimien määräämiseksi johtavat erilaisiin menetelmiin.

Kollokaatiomenetelmässä sijoitetaan yrite (7.39) yhtälöön ja vaaditaan, että se toteutuu tarkasteluvälin $[a,b]$ N:ssä valitussa pisteessä t_j. Esimerkkitapauksessa (7.38) saadaan yhtälöryhmä

$$\sum_{i=1}^{N} c_i \phi_i''(t_j) = f(t_j, \sum_{i=1}^{N} c_i \phi_i(t_j), \sum_{i=1}^{N} c_i \phi_i'(t_j)), \quad j = 1,2,\ldots,N,$$

josta kertoimet $\{c_1, c_2, \ldots, c_N\}$ voidaan määrätä. Kollokaatiomenetelmät ovat yhteneviä tiettyjen differenssimenetelmien kanssa.

Tavallisesti kollokaatiomenetelmässä käytetään kantafunktioina palapolynomeja, esimerkiksi B-splinejä (kappale 4.11.2) tai Hermiten splinejä. Kollokaatiomenetelmää voidaan käyttää suoraan myös korkeampiasteisten tehtävien ratkaisemiseen.

Galerkinin menetelmässä vaaditaan, että virhe $x''(t) - f(t, x(t), x'(t))$ on kohtisuorassa aliavaruutta $\{\phi_1, \phi_2, \ldots, \phi_N\}$ vastaan, mikä yhtälön (7.38) tapauksessa johtaa yhtälösysteemiin

$$\int_a^b \left(\sum_{i=1}^{N} c_i \phi_i''(\tau) - f(\tau, \sum_{i=1}^{N} c_i \phi_i(\tau), \sum_{i=1}^{N} c_i \phi_i'(\tau)) \right) \phi_j(\tau) d\tau = 0,$$

missä $j = 1,2,\ldots,N$. Kertoimet c_i voidaan ratkaista näistä N:stä ehdosta. Osittaisdifferentiaaliyhtälöiden numeerisessa ratkaisemisessa paljon käytetyn *elementtimenetelmän* perusversio on Galerkinin menetelmän erikoistapaus (luku 8).

7.6 Menetelmäsuosituksia

Alkuarvotehtävät: Ei-kankeille tehtäville, joissa derivaatan **f** määrääminen on halpaa, soveltuvat hyvin eksplisiittiset Rungen ja Kuttan ratkaisijat.

Jos derivaatan määrääminen ei-kankeassa tehtävässä on kallista, kannattaa käyttää ennustaja–korjaaja-menetelmiä tai Adamsin ja Moultonin menetelmiä.

Kankeiden tehtävien ratkaisemiseen suosituimpia ovat BDF-menetelmät, mutta myös implisiittiset Rungen ja Kuttan menetelmät ovat kilpailukykyisiä.

Reuna-arvotehtävät: Monimaalimenetelmää voidaan käyttää yksinkertaisissa tapauksissa. Kollokaatio- ja differenssimenetelmä sopivat hankalammille yhtälöille.

7.7 Ohjelmistot

Monipuolinen numeriikan ohjelmisto Matlab sisältää sekä ei-kankeille että kankeille alkuarvotehtäville sopivia helppokäyttöisiä ratkaisijoita. Symboli-

sen matematiikan ohjelmistossa Mathematicassa on numeerisia ratkaisijoita sekä alku- että reuna-arvotehtäville. Mathematica selviää myös kankeista alkuarvotehtävistä, mutta pystyy ratkaisemaan vain lineaarisia reuna-arvotehtäviä. Lisäksi Mathematica kykenee ratkaisemaan yksinkertaisia tehtäviä myös symbolisesti.

Sekä Matlab että Mathematica ovat interaktiivisuutensa vuoksi helppokäyttöisiä ja soveltuvat pienten tehtävien ratkaisemiseen sekä ratkaisumenetelmien testaamiseen. Paljon laskentaa vaativien suurien tehtävien ratkaisemiseen ne ovat kuitenkin liian hitaita; suositeltavampaa on käyttää Fortrania tai C-kieltä ja esimerkiksi valmiita aliohjelmakirjastoja.

Yleiskäyttöiset matemaattiset aliohjelmakirjastot NAG ja IMSL sisältävät monia rutiineja differentiaaliyhtälöiden ratkaisemiseen. Käyttäjä voi tehtävän mukaan valita sopivaa menetelmää käyttävän aliohjelman.

Netlib-aliohjelmakirjastossa differentiaaliyhtälöratkaisijat on sijoitettu hakemistoihin `ode` ja `odepack`. Myös `toms`-algoritmeissa on monia ratkaisijoita.

Ernst Hairerin ylläpitämältä sivulta `http://www.unige.ch/math/folks/hairer/software.html` saa sekä kankeiden että ei-kankeiden tehtävien ratkaisemiseen sopivia aliohjelmia.

7.8 Lisätietoja

Teos *A First Course in the Numerical Analysis of Differential Equations* [Ise96] on suhteellisen helppotajuinen johdatus alkuarvotehtävien numeeriseen ratkaisemiseen.

Kirjassa *Numerical Methods for Ordinary Differential Equations* [Lam91] on aiheesta pidemmälle menevä ja teoreettisempi esitys, mutta monet havainnolliset esimerkit helpottavat lukemista. Varsinkin kankeuden käsite on käyty läpi perusteellisesti ja monipuolisesti.

Monia hyviä esimerkkejä on myös kirjassa *Introduction to Numerical Analysis* [SB93], mutta kankeat tehtävät ja niiden ratkaisemiseen soveltuvat menetelmät jätetään kokonaan käsittelemättä.

Erittäin havainnollinen ja käytännönläheinen esitys alku- ja reuna-arvotehtävien numeerisesta ratkaisemisesta on kirjassa *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations* [AP98]. Mukana on myös katsaus saatavissa oleviin ohjelmistoihin.

Alkuarvotehtävistä lähes kaiken tietämisen arvoisen löytää klassisista teoksista *Solving Ordinary Differential Equations I* [HNW93] ja *Solving Ordinary Differential Equations II* [HW96].

Kirja *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations* [AMR95] käsittelee reuna-arvotehtävien numeeriset ratkaisumenetelmät perusteellisesti ja kattavasti.

8 Elementtimenetelmä

Kerromme tässä luvussa elementtimenetelmän (FEM) käytöstä osittaisdifferentiaaliyhtälöiden (ODY) ratkaisemiseksi. Käytämme esimerkkinä lämpöyhtälöä sekä ajasta riippuvassa että riippumattomassa muodossa. Esitetyt menetelmät yleistetään osittaisdifferentiaaliyhtälöryhmän ratkaisuun, mistä esimerkkinä on lämpöyhtälöstä sekä Navierin ja Stokesin yhtälöstä muodostettu kytketty yhtälöryhmä.

8.1 Taustaa

Elementtimenetelmää käytetään ratkaistaessa numeerisesti integraali-, integrodifferentiaali- ja differentiaaliyhtälöitä. Sovellusalueita ovat mm. rakenteiden mekaniikka (kuva 8.1), virtaustehtävät, lämmönsiirto, akustiikka, kvanttimekaniikka, sähkömagnetiikka sekä yhdistelmät näistä.

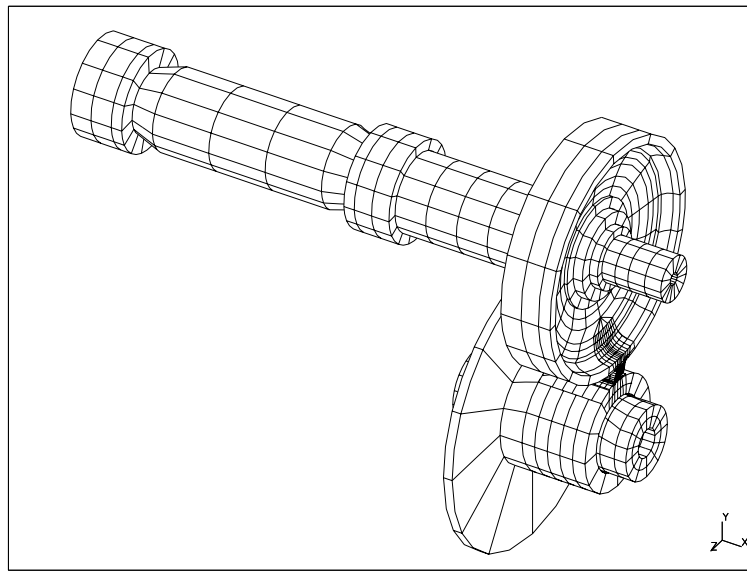
Mainittuja sovelluksia voidaan yrittää ratkoa numeerisesti elementtimenetelmän lisäksi muillakin menetelmillä kuten differenssimenetelmällä, kontrollitilavuusmenetelmällä, spektraalimenetelmillä ja sarjakehitelmillä. Elementtimenetelmän suurimpina etuina muihin menetelmiin verrattuna on sen yleispätevyys sekä se, että sen soveltaminen monimutkaisiinkin alueisiin on helppoa. Lisäksi ratkaisun laatua voi luontevasti parantaa keskittymällä laskennan kannalta vaikeisiin paikkoihin, esimerkiksi jyrkkiin kulmiin. Menetelmä on myös helppo ymmärtää ja periaatteessa suoraviivainen toteuttaa.

Seuraavassa on joitakin fysikaalisia yhtälöitä, joihin elementtimenetelmää yleisesti sovelletaan.

Esimerkki 8.1.1 Toisesta päästään lämmitetyn tangon lämpötilaa tasapainotilassa kuvaavat yhtälöt

$$\begin{cases} -kT'' = \rho h, \\ T(0) = T_0,\ [-kT']_{x=L} = g. \end{cases}$$

Tässä suure L on tangon pituus. Suure h voi tässä olla esimerkiksi lämpöhäviö pituusyksikköä kohden tangon pituussuunnassa. Tankoa lämmitetään (tai jäähdytetään) toisesta päästä vakiolämmöllä. Suureella g kuvataan tangon vapaan pään lämmönvaihtoa ympäristön kanssa.



Kuva 8.1: *Hammaspyörästön toiminnan tutkimiseen tarkoitettu elementtiverkko.*

Esimerkki 8.1.2 Lämmön siirtymistä virtaavassa aineessa kuvaa energian säilymislaista johdettu yhtälö

$$\rho c_p \left(\frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T \right) - \nabla \cdot (k \nabla T) = \rho h,$$

missä ρ on aineen tiheys, c_p ominaislämpökapasiteetti vakiopaineessa, k lämmönjohtavuus sekä h annettu lämpölähde. Suure $\mathbf{u}$ on virtausnopeus. Tuntematon T on lämpötila.

Lämpötila on muuttujana monissa käytännön tehtävissä. Esimerkiksi virtaustehtävissä virtaavan aineen lämpötila kytkeytyy mm. lämpölaajenemisen kautta virtausnopeuksiin ja paineeseen. Lämpötila pyrkii jakautumaan tasaisesti virtaavaan aineeseen eli leviämään diffuusion kautta, kunnes paikalliset lämpötilaerot ovat tasoittuneet. Tämän lisäksi lämpöä siirtyy virtaavan aineen sisällä paikasta toiseen kulkeutumalla virtaavan aineen mukana. Virtaavan aineen sisäinen kitka myös muuttaa kineettistä energiaa lämpöenergiaksi.

Esimerkki 8.1.3 Navierin ja Stokesin yhtälöt eli liikemäärän ja massan säilymislait kuvaavat nesteiden ja kaasujen virtausta

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) - \nabla \cdot \boldsymbol{\tau} = \rho \mathbf{f},$$

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0.$$

Tuntemattomat suureet ovat virtausnopeus $\mathbf{u}$, paine p ja tiheys ρ. Suure $\mathbf{f}$ on annettu ulkoinen tilavuusvoima. Tällainen voi olla esimerkiksi maan vetovoima. Jännitystensori $\boldsymbol{\tau}$ saadaan lausekkeesta

$$\tau_{ij} = 2\mu\varepsilon_{ij} - (p + \frac{2}{3}\mu\nabla \cdot \mathbf{u})\delta_{ij}, \tag{8.1}$$

missä

$$\varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right). \tag{8.2}$$

Viskositeetti μ on virtaavan aineen sisäisen kitkan suuruutta kuvaava suure. Suuretta $\boldsymbol{\varepsilon}$ kutsutaan venymänopeustensoriksi. Yhtälöissä tuntemattomia suureita on enemmän kuin yhtälöitä. Yhtälöt täytyy täydentää *tilayhtälöllä* $\rho = f(p, T)$.

Navierin ja Stokesin yhtälöt ovat termin $\mathbf{u} \cdot \nabla\mathbf{u}$ vuoksi epälineaariset, mikä johtaa yhtälöiden hyvin monimuotoiseen käyttäytymiseen ja myös vaikeuttaa yhtälöiden ratkaisua tietokoneella. Luonnossa melkein kaikki virtaukset ovatkin *turbulentteja;* toisin sanoen ison mittakaavan käyttäytymisen lisäksi virtaukset käyttäytyvät pienessä mittakaavassa sekä ajan että paikan suhteen erittäin monimutkaisella tavalla, kaoottisesti. Turbulenssia pystytään nykyisillä tietokoneilla kuvaamaan vain keskimääräisillä malleilla, joiden tarkkuus ja sopivuus erilaisiin fysikaalisiin tilanteisiin on hyvin rajallinen.

Esimerkki 8.1.4 Aaltoyhtälö, joka kuvaa mm. akustisen paineen vaihteluita on

$$\frac{1}{c^2}\frac{\partial^2\Phi}{\partial t^2} - \nabla^2\Phi = 0.$$

Oletamme, että akustinen potentiaali Φ voidaan esittää Fourier'n kehitelmänä. Koska tehtävä on lineaarinen, kutakin kehitelmän termiä voidaan tarkastella erikseen. Tällöin

$$\Phi(t, \mathbf{x}) = e^{i\omega t}\widetilde{\Phi}(\mathbf{x}).$$

Tehtävä yksinkertaistuu näin muotoon

$$\frac{\omega^2}{c^2}\widetilde{\Phi} + \nabla^2\widetilde{\Phi} = 0.$$

Tässä tuntematon suure Φ on akustinen potentiaali, suure ω värähtelytaajuus (äänenkorkeus) ja c äänen nopeus ilmassa.

Sovelluskohteena tälle yhtälölle on esimerkiksi akustiikkasuunnittelu. Mallit ovat usein erittäin suuria, koska esimerkiksi ihmiskorva kuulee äänenkorkeuksia noin taajuuteen 20 kHz. Aallonpituus on tällöin suuruusluokkaa $\lambda \approx 1$ cm. Mallin elementtien koon pitäisi olla samaa suuruusluokkaa, jotta mallilla voitaisiin kuvata näitä taajuuksia. Sivuiltaan metrin mittaiseen kuutioon pitäisi siis sijoittaa vähintään miljoona elementtiä.

Esimerkki 8.1.5 Yksiulotteinen ajasta riippumaton Schrödingerin yhtälö kuvaa epärelavistisessa approksimaatiossa yhden spinittömän alkeishiukkasen käyttäytymistä potentiaalissa V:

$$-\frac{\hbar^2}{2m}\frac{d^2\Psi}{dx^2} + V\Psi = E\Psi.$$

Tässä $h = 2\pi\hbar$ on Planckin vakio, m on hiukkasen massa, ja suure E sen kokonaisenergia. Ominaisarvoa E_i vastaava ominaisvektori Ψ_i on hiukkasen ominaistilaa vastaava aaltofunktio. Suure

$$P = \int_{x_0}^{x_1} |\Psi(x)|^2\, dx$$

antaa todennäköisyyden sille, että hiukkanen löytyy väliltä $[x_0, x_1]$.

Esimerkki 8.1.6 Sähkömagneettisia kenttiä kuvaavat Maxwellin yhtälöt

$$\begin{aligned}
\nabla \cdot \mathbf{D} &= \rho, \\
\nabla \times \mathbf{H} &= \frac{\partial \mathbf{D}}{\partial t} + \mathbf{J}, \\
\frac{\partial \mathbf{B}}{\partial t} + \nabla \times \mathbf{E} &= 0, \\
\nabla \cdot \mathbf{B} &= 0.
\end{aligned}$$

Tässä kenttä **D** on sähkövuon tiheys, **H** magneettikenttä, **B** magneettivuon tiheys, **E** sähkökenttä, **J** on virtatiheys ja ρ varaustiheys. Virtatiheys voidaan usein lausua sähkökentän avulla (Ohmin laki):

$$\mathbf{J} = \sigma\mathbf{E},$$

missä σ on materiaalin johtavuus. Lisäksi oletamme, että johtava materiaali pysyy laskentakoordinaatistoon nähden paikallaan. Konstitutiiviset yhteydet kenttien **D** ja **E** sekä kenttien **H** ja **B** välillä ovat yksinkertaisimmillaan tyhjiössä

$$\begin{aligned}
\mathbf{D} &= \epsilon_0\mathbf{E}, \\
\mathbf{H} &= \frac{1}{\mu_0}\mathbf{B},
\end{aligned}$$

missä vakiot ϵ_0 ja μ_0 ovat tyhjön permittiivisyys ja permeabiliteetti.

Kaikki edellä kuvatut yhtälöt ovat differentiaaliyhtälöitä. Tavallisia differentiaaliyhtälöitä on jo käsitelty luvussa 7. Useimmat edellä esitellyistä yhtälöistä ovat kuitenkin osittaisdifferentiaaliyhtälöitä (ODY). Osittaisdifferentiaaliyhtälöt poikkeavat tavallisista differentiaaliyhtälöistä siinä, että niissä on yhden sijasta useampi riippumaton muuttuja. Riippumattomia muuttujia voivat olla esimerkiksi aika ja paikka. Paikkakoordinaattejakin voi olla yksi tai useampia.

Lineaariset osittaisdifferentiaaliyhtälöt luokitellaan usein elliptisiin, parabolisiin ja hyperbolisiin yhtälöihin.

Elliptiset yhtälöt kuvaavat potentiaali-ilmiöitä ja tasapainotiloja. Tyyppiesimerkki elliptisestä yhtälöstä on lämmön johtumisen tasapainotilaa kuvaava yhtälö. Tällöin aikaderivaattatermi jää yhtälöstä pois ja yhtälö yksinkertaistuu muotoon

$$-k\nabla^2 T = \rho h.$$

Tässä on lisäksi oletettu, että materiaalin lämmönjohtavuus k on vakio paikan suhteen. Yhtälö on esimerkki Poissonin yhtälöstä. Elliptiset tehtävät ovat luonteensa vuoksi usein helpompia ratkaistavia kuin esimerkiksi hyperboliset tehtävät. Näiden numeerisen ratkaisemisen teoria tunnetaankin huomattavasti paremmin kuin muun tyyppisten yhtälöiden.

Ajasta riippuva lämmönjohtumisyhtälö on ensimmäisen kertaluvun aikaderivaattatermin vuoksi parabolinen osittaisdifferentiaaliyhtälö. Paraboliset yhtälöt kuvaavat siis esimerkiksi ajan myötä eteneviä ja tasoittuvia ilmiöitä.

Aaltoyhtälö on hyperbolinen. Hyperboliset yhtälöt kuvaavat ilman vaimennusta aaltoilevia ja värähteleviä tai eteneviä ilmiöitä. Fysikaalisissa tilanteissa erilaiset vaimennukset, kuten kitka, ovat kuitenkin usein niin merkittäviä, että ne on mallinnuksessa otettava huomioon. Osittaisdifferentiaaliyhtälöiden luokittelu kolmeen alalajiin ei tällöin enää toimi, vaan päädytään erilaisiin sekatehtäviin. Esimerkki tästä on lämmönsiirtoa virtaavassa aineessa kuvaava yhtälö:

$$\rho c_p \left(\frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T \right) - \nabla \cdot (k \nabla T) = \rho h.$$

Tällä yhtälöllä on sekä elliptinen, parabolinen että hyperbolinen luonne, sen mukaan mitkä termit hallitsevat yhtälön käyttäytymistä.

8.2 Elementtimenetelmän peruskäsitteitä

Elementtimenetelmässä yritetään arvioida ratkaistavaa suuretta, esimerkiksi lämpötilaa, jollain sopivan yksinkertaisella, helposti käsiteltävällä funktiolla. Yksiulotteinen suure voitaisiin välillä $[0, 1]$ esittää esimerkiksi suoralla

$$u(x) \approx \widetilde{u}(x) = u_1(1 - x) + u_2 x,$$

missä kertoimet u_i ovat tuntemattomia. Yleisemmin voimme arvioida suureen vaihteluita funktiolla, jossa on n vapaata parametria

$$u(x) \approx \widetilde{u}(x) = u_1 \Psi_1(x) + \ldots + u_n \Psi_n(x). \tag{8.3}$$

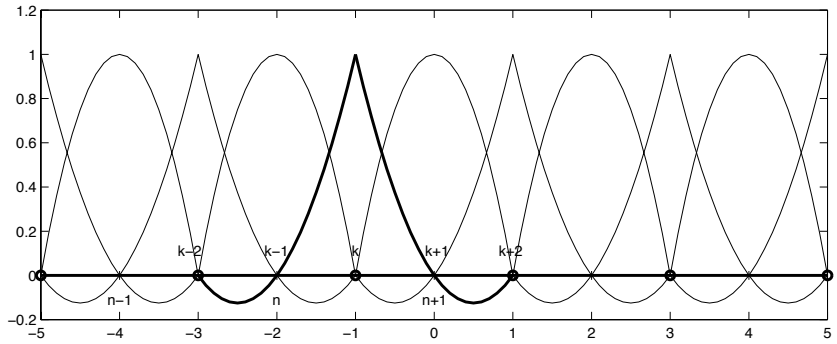
Menetelmässä laskenta-alue jaetaan pieniin osa-alueisiin eli elementteihin. Sen sijaan, että arvioisimme suuretta suoralla, käytämme murtoviivaa. Elementtien täytyy täyttää koko laskenta-alue, eivätkä ne saa olla miltään osin päällekkäin. Tasoalueiden tai pintojen jakoon käytetään kolmioita ja nelitahokkaita; vastaavasti kolmiulotteisen alueen jakoon käytetään yleisimmin

tetraedrejä ja tiiliskivielementtejä (brick element). Myös käyräviivaisia elementtejä käytetään yleisesti. Kuvissa 8.5 (sivu 263) ja 8.8 (sivu 266) on hahmoteltu yleisimpiä elementtityyppejä. Elementin geometrian määrääviä pisteitä (esimerkiksi kolmion kärkipisteitä) kutsutaan solmuiksi. Elementtien reunoilla olevat solmut ovat vierekkäisille elementeille yhteisiä.

Yhtälössä (8.3) esiintyviä funktioita Ψ_i kutsutaan kantafunktioiksi. Useimmiten kantafunktioiksi valitaan matala-asteisia *paloittain määriteltyjä* polynomeja. Kantafunktiot määritellään yleensä siten, että yksittäinen kantafunktio poikkeaa nollasta ainoastaan niiden elementtien alueella, joihin kantafunktioon liittyvä *vapausaste* u_i kuuluu. Ideaa on havainnollistettu kuvassa 8.2. Paksunnetulla viivalla näytetty globaali kantafunktio $\Psi_k(x)$ voidaan esittää seuraavasti:

$$\Psi_k(x) = \begin{cases} N_2^n(x), & \text{kun } x \text{ on elementin } n \text{ alueella,} \\ N_1^{n+1}(x), & \text{kun } x \text{ on elementin } n+1 \text{ alueella,} \\ 0, & \text{muualla.} \end{cases} \tag{8.4}$$

Tässä funktiot N_j^i ovat elementtien kantafunktioita. Elementtikohtaisia kantafunktioita käsitellään kappaleessa 8.7 (sivu 258).



Kuva 8.2: *Neliöllisten 1D-elementtien kantafuntioiden kuvaajia. Solmuun k liitetyn globaalin kantafunktion $\Psi_k(x)$ kuvaaja on näytetty paksunnettuna.*

Yhtälöiden (8.3) ja (8.4) lokaalin luonteen vuoksi voimme muodostaa ratkaisuun tarvittavat yhtälöt elementeittäin ja laskea elementtikohtaiset yhtälöt yhteen jokaista tuntematonta kohden. Tätä kutsutaan *matriisin kokoamiseksi* (assembly). Ratkaistavanamme tässä ovat siis suureen vaihteluita laskenta-alueella kuvaavan paloittain määritellyn polynomin kertoimet.

Paloittaisen approksimaation valinnasta on merkittäviä etuja. Tällöin yksittäisen kantafunktion kertoimet riippuvat suoraan vain muutamasta vierekkäisiin solmuihin liitettyjen kantafunktioiden kertoimista, toisin sanoen diskretoinnista syntyvä yhtälöryhmä on *harva*. Tämä mahdollistaa isojenkin tehtävien tehokkaan ratkaisun tietokoneella.

Seuraavassa käymme läpi yksityiskohtaisesti elementtimenetelmän soveltamiseen liittyviä asioita.

8.3 Yhtälöiden diskretointi tietokoneratkaisua varten

Yhtälöiden diskretointi elementtimenetelmässä tehdään käyttämällä yhtälön variaatiomuotoa tai vastaavasti painotettujen jäännösten menetelmää (Methods of Weighted Residuals, MWR).

Variaatiomenetelmä perustuu siihen, että tietyn integraaliyhtälön minimointi johtaa muodollisesti differentiaaliyhtälöihin. Näitä differentiaaliyhtälöitä kutsutaan variaatiointegraalin *Eulerin yhtälöiksi* tai integraalin *vahvaksi muodoksi*. Integraaliyhtälöä puolestaan kutsutaan myös differentiaaliyhtälöiden *heikoksi muodoksi. Rayleigh'n ja Ritzin menetelmässä*, jota käytetään yhtälöiden diskretointiin, haettu suure ratkaistaan suoraan variaatiomuotoa käyttäen. Siinä sijoitetaan, samaan tapaan kuin painotettujen jäännösten menetelmässä, sopiva kantafunktioapproksimaatio ratkaistavan suureen paikalle ja minimoidaan integraali kertoimien suhteen. Fysikaalinen tilanne voidaan usein myös esittää suoraan variaatiomuodossa, jolloin integraali voi esittää esimerkiksi systeemin kokonaisenergiaa.

Variaatiomenetelmä ja painotettujen jäännösten menetelmät ovat käytännössä hyvin samankaltaisia, vaikkakin niiden matemaattiset perustelut hiukan eroavat toisistaan. Variaatiomenetelmää voidaan tarkkaan ottaen käyttää vain, jos ratkaistavalle yhtälölle on löydettävissä variaatiomuoto eli jos yhtälön differentiaalioperaattori $\mathcal{L}$ on sekä lineaarinen että itseadjungoitu.[1] Painotettujen jäännösten menetelmä puolestaan on yleisempi.

Esitämme seuraavassa painotettujen jäännösten menetelmän: Hae ratkaisuvektori $\mathbf{u} = (u_1, u_2, \ldots, u_n)^T$ siten, että

$$\int_\Omega R(\mathbf{x}, \mathbf{u}) W_i(\mathbf{x})\, d\Omega = 0, \quad i = 1, \ldots, N,$$

missä R on ratkaistavana olevan yhtälön jäännös

$$R = \mathcal{L}(\widetilde{u}) - f.$$

Muuttuja $\widetilde{u}$ on tässä ratkaistavan suureen *kantafunktioapproksimaatio* tai lyhyemmin *yrite:*

$$\widetilde{u}(\mathbf{x}) = \sum_{i=1}^{N} u_i \Psi_i(\mathbf{x}).$$

[1] Operaattori $\mathcal{L} : \mathcal{X} \mapsto \mathcal{X}$ on lineaarinen, jos kaikille vektoreille (funktioille) $u, v \in \mathcal{X}$ ja luvuille $\alpha, \beta \in \mathbb{R}$ pätee

$$\mathcal{L}(\alpha u + \beta v) = \alpha \mathcal{L}(u) + \beta \mathcal{L}(v).$$

Operaattorin $\mathcal{L}$ adjungaatti on operaattori $\mathcal{L}^*$, jolle pätee

$$\langle \mathcal{L}(u), v \rangle = \langle u, \mathcal{L}^*(v) \rangle,$$

kaikille vektoreille u, v. Merkinnällä $\langle \cdot, \cdot \rangle$ tarkoitamme vektoriavaruuden $\mathcal{X}$ sisätuloa. Operaattori $\mathcal{L}$ on itseadjungoitu, jos $\mathcal{L} \equiv \mathcal{L}^*$. Ajasta riippumattoman lämmönjohtumisyhtälön differentiaalioperaattori

$$\mathcal{H}(\cdot) = -\nabla \cdot (k \nabla \cdot)$$

on lineaarinen ja itseadjungoitu, jos vektori $k \in \mathcal{X}$ ei riipu lämpötilasta.

Haluamme siis jäännöksen painotetun keskiarvon häviävän laskenta-alueella.

Jos testifunktioina W_i käytetään samoja kantafunktioita kuin ratkaistavalle suureelle, $W_i = \Psi_i$, kutsutaan painotettujen jäännösten menetelmää Galerkinin menetelmäksi. Tällä tavalla tulemme minimoineeksi jäännöksen energianormin (katso kappaletta 8.9) suhteen. Tällöin $\tilde{u}$ on tarkan ratkaisun u ortogonaaliprojektio siihen funktiojoukkoon, jossa $\tilde{u}$ on määritelty.

Muitakin vaihtoehtoja toki on. Testifunktioksi voitaisiin ottaa δ-funktio

$$\int_\Omega R(\mathbf{x},\mathbf{u})\delta(\mathbf{x},\mathbf{x}_i)\,d\Omega = R(\mathbf{x}_i,\mathbf{u}) = 0,$$

jolloin siis asetamme jäännöksen nollaksi ennalta valituissa pisteissä. Tätä kutsutaan *kollokaatiomenetelmäksi*. Asettamalla jäännöksen keskiarvo osa-alueittain nollaksi

$$\int_{\Omega_i} R(\mathbf{x},\mathbf{u})\,d\Omega_i = 0,$$

saadaan *osa-aluemenetelmä* (subdomain method). Osa-alueet voidaan valita vapaasti ja ne voivat olla myös osittain päällekkäisiä. Pienimmän neliösumman menetelmään päädytään, kun testifunktioiksi valitaan

$$W_i = \frac{\partial}{\partial u_i} R(\mathbf{x},\mathbf{u}),$$

jolloin ratkaistavat yhtälöt minimoivat neliösummafunktionaalin

$$\int_\Omega R^2\,d\Omega.$$

Galerkinin menetelmä on kuitenkin ylivoimaisesti suosituin painotettujen jäännösten menetelmä ja keskitymme seuraavassa siihen.

Historiallisesti variaatiomenetelmä on ollut vallitseva menetelmä elementtimenetelmän johtamiseksi. Galerkinin menetelmän suosio perustuu ainakin osaksi siihen, että se on identtinen variaatiomenetelmän kanssa niissä tehtävissä, joihin molemmat ovat sovellettavissa.

Seuraavassa sovellamme esimerkin vuoksi painotettujen jäännösten menetelmää Poissonin yhtälöön.

Esimerkki 8.3.1 Galerkinin menetelmä yhtälölle

$$-\nabla^2 T = f.$$

Aloitamme kertomalla yhtälön puolittain testifunktiolla W sekä integroimalla laskenta-alueen yli

$$-\int_\Omega \nabla^2 T W\,d\Omega = \int_\Omega fW\,d\Omega.$$

Termi

$$-\int_\Omega \nabla^2 T W \, d\Omega$$

on toista astetta suureen T paikkaderivaattojen suhteen, jolloin suureen esittämiseen tarvittaisiin vähintään kahteen kertaan jatkuvasti derivoituvia kantafunktioita. Osittaisintegroimalla saamme

$$-\int_\Omega \nabla^2 T W \, d\Omega = \int_\Omega \nabla T \cdot \nabla W \, d\Omega - \int_\Gamma \frac{\partial T}{\partial n} W \, d\Gamma,$$

missä olemme käyttäneet merkintää $\mathbf{n} \cdot \nabla T = \partial T / \partial n$. Suure $\mathbf{n}$ on reunan Γ ulospäin suunnattu yksikkönormaali. Reunalle syntyvää termiä kutsutaan *luonnolliseksi* (natural) reunaehdoksi. Reunaehtoja käsitellään myöhemmin kappaleessa 8.4. Kuten nähtävissä on, osittaisintegroimalla saimme pudotettua kantafunktioiden jatkuvuusvaatimuksia.

Seuraavaksi sijoitamme lämpötilalle yritteen

$$T \approx \sum_{j=1}^{N} T_j \Psi_j$$

ja testifunktion W paikalle kantafunktiot Ψ_i:

$$\sum_{j=1}^{N} \left(\int_\Omega \nabla \Psi_j \cdot \nabla \Psi_i \, d\Omega - \int_\Gamma \frac{\partial \Psi_j}{\partial n} \Psi_i \, d\Gamma \right) T_j = \int_\Omega f \Psi_i \, d\Omega, \quad i = 1, \ldots, N. \tag{8.5}$$

Integraali reunan Γ yli koskee vain reunalla olevia vapausasteita.

Matriisimuotoon kirjoitettuna yhtälö (8.5) on muotoa

$$A\mathbf{T} = \mathbf{F}. \tag{8.6}$$

Vektorin $\mathbf{F}$ ja matriisin A alkiot ovat siis

$$\begin{aligned} F_i &= \int_\Omega f \Psi_i \, d\Omega, \\ A_{ij} &= \int_\Omega \nabla \Psi_j \cdot \nabla \Psi_i \, d\Omega - \int_\Gamma \frac{\partial \Psi_j}{\partial n} \Psi_i \, d\Gamma. \end{aligned}$$

Yhtälö (8.6) on tavallinen (tässä tapauksessa lineaarinen) algebrallinen yhtälöryhmä, joka ratkeaa tavallisilla lineaarisen yhtälöryhmän ratkaisumenetelmillä. Yhtälöryhmä on useimmiten erittäin suuri mutta paloittaisille kantafunktioille myös harva, jolloin matriisiyhtälö on mahdollista ratkaista numeerisesti järkevällä työmäärällä.

Kerroinmatriisia A kutsutaan usein *jäykkyysmatriisiksi* tai *johtavuusmatriisiksi*. Sovelluksesta riippuen muitakin nimityksiä esiintyy. Nimitys ”jäykkyysmatriisi” tulee rakenteiden mekaniikasta, joka oli historiallisesti ensimmäinen nykyisenkaltaisen elementtimenetelmän sovelluskohde. Tätä nimitystä kerroinmatriisista A käytetäänkin yleisesti, ja vähän harhaanjohtavasti, myös muiden sovellusten yhteydessä.

Galerkinin menetelmän soveltaminen differentiaaliyhtälön diskretoimiseksi koostuu siis seuraavista askeleista:

- Kerrotaan yhtälön jäännös testifunktioilla ja integroidaan laskenta-alueen yli.
- Osittaisintegroidaan korkeimman asteluvun derivaatat, yhtälöstä riippuen mahdollisesti useampaan kertaan.
- Sijoitetaan ratkaistavan suureen tilalle kantafunktioapproksimaatio ja testifunktioiden tilalle kantafunktiot.
- Muodostetaan kerroinmatriisi.

Jos tehtävä on annettu variaatiointegraalin muodossa, kahta ensimmäistä vaihetta ei tarvitse suorittaa.

Yleensä on tapana osittaisintegroida yhtälön termejä kunnes muuttujan ja testifunktion derivaattojen asteluvut termeittäin poikkeavat toisistaan enintään yhdellä. Tästä on se etu, että tällöin jatkuvuusvaatimukset kantafunktioille ovat mahdollisimman lievät. Vaikka alkuperäisen differentiaaliyhtälön suurin paikkaderivaattatermien asteluku olisi kaksi, osittaisintegroinnin jälkeen tarvitsemme kantafunktioilta vain funktioiden itsensä jatkuvuutta. Kantafunktioiden paikkaderivaattojen jatkuvuutta ei tässä siis tarvitse vaatia. Suoritettujen osittaisintegrointien määrä vaikuttaa myös oleellisesti reunaehtojen luonteeseen ja käsittelyyn.

Galerkinin menetelmä ei suoraan sovellu kaiken tyyppisten osittaisdifferentiaaliyhtälöiden ratkaisuun. Ehkä tunnetuin esimerkki yhtälöstä, johon menetelmä soveltuu huonosti, on advektio-diffuusioyhtälö, kun yhtälön advektiotermillä eli kuljetustermillä on selkeästi suurempi vaikutus yhtälön käyttäytymiseen kuin diffuusiolla. Esimerkkinä advektio-diffuusioyhtälöstä on jo esitettykin virtauksen mukanaan kuljettamaa lämpökenttää kuvaava yhtälö. Myös virtausyhtälöt kuuluvat samaan tehtäväluokkaan.

Tämän tyyppiset ongelmat elementtimenetelmän soveltamisessa liittyvät differentiaaliyhtälöiden ratkaisun luonteeseen ja approksimoivan kannan valintaan. Tunnettua on, että peruselementtimenetelmä paloittaisine matalaasteisine polynomikantoineen soveltuu hyvin elliptisille tehtäville tai ajasta riippuvassa tapauksessa paikan suhteen elliptiselle osalle tehtävää. Muun tyyppisille tehtäville yritteen kantana käytetään usein samaa kantaa kuin elliptisillekin tehtäville, ”kun paremmastakaan ei ole tietoa”.

Vuosikymmenten mittaan sovellettaessa elementtimenetelmää erilaisiin tehtäviin on syntynyt suuri joukko erilaisia enemmän tai vähemmän heuristisia tapoja kiertää diskretoinnin ongelmia. Näitä ovat esimerkiksi erilaiset epäjatkuvat elementtityypit tai vastaavasti tapa integroida joitain osia ratkaistavista yhtälöistä pienemmällä tarkkuudella kuin muita.

Stabiloidut elementtimenetelmät pyrkivät laajentamaan elementtiapproksimaation kantafunktioavaruutta *paikallisesti yksittäisen elementin alueella* ja säilyttämään laajemmassa skaalassa peruselementtimenetelmän rakenteen. Tässä mielessä ne kuuluvat samaan sarjaan menetelmiä kuin mainitut epäjatkuvat elementit tai muut vastaavat menetelmät. Menetelmien hyvä puoli

on se, että ne säilyttävät kokonaisuutena peruselementtimenetelmän edut: paloittaisen approksimaation tuoman yhtälöryhmän harvuuden ja yksinkertaisen muodon. Stabiloitua elementtimenetelmää käsitellään kappaleessa 8.14 (sivu 294).

8.4 Reunaehdot

Jotta differentiaaliyhtälön reuna-arvotehtävällä olisi yksikäsitteinen ratkaisu, täytyy ratkaisun toteuttaa laskenta-alueen reunoilla reunaehdot. Jos korkein yhtälössä esiintyvä derivaatta on astetta $2k$, täytyy kaikilla reunoilla olla k reunaehtoa.

Näimme kappaleessa 8.3 kuinka osittaisintegrointi tuotti esimerkkiyhtälöömme reunalle uuden termin. Näitä osittaisintegroinnista syntyviä reunatermejä kutsutaan *luonnollisiksi* (natural) reunaehdoiksi; ne tavallaan täyttyvät formuloinnissamme luonnostaan. Luonnolliset reunaehdot ovat niitä, joissa esiintyy vain (pinnan normaalin suuntaisia) derivaattoja joiden kertaluku on vähintään k. Luonnollisia reunaehtoja kutsutaan usein myös *Neumannin reunaehdoiksi* tai voimareunaehdoiksi.

Reunaehtoja, jotka sisältävät ainoastaan derivaattoja, joiden kertaluku on pienempi kuin k, kutsutaan olennaisiksi (essential) reunaehdoiksi. Tehtävän määrittelyn täytyy sisältää ainakin joku olennainen reunaehto, jotta se olisi määritelty yksikäsitteisesti. Olennaisia reunaehtoja nimitetään usein myös *Dirichlet'n reunaehdoiksi*.

Esimerkissämme olennainen reunaehto tarkoittaa ratkaistavan suureen (lämpötilan) arvon asettamista tunnettuun arvoon kaikilla tai osalla mallin reunoista.

Olennaisen reunaehdon asettaminen voidaan tehdä usealla tavalla. Helpointa on asettaa sen vapausasteen yhtälö, jota reunaehto koskee, yhtälöksi $T = T_\Gamma$, ja ratkaista tehtävä käyttäen tätä muokattua yhtälöryhmää. Toisin sanoen asetetaan vapausastetta vastaava rivi yhtälön kerroinmatriisissa diagonaalialkiota lukuun ottamatta nollaksi. Diagonaalialkion arvoksi asetetaan arvo yksi. Vastaavasti voimavektorin vastinalkion arvoksi täytyy tällöin asettaa arvo T_Γ. Jos yhtälö alunperin on symmetrinen, niin asettamalla myös vastaava matriisin sarake nollaksi ja muokkaamalla samalla oikean puolen vektoria voidaan säilyttää tämä symmetria.

Toinen hiukan hankalampi tapa asettaa olennainen reunaehto on eliminoida reunan vapausasteet kokonaan pois globaalista yhtälöryhmästä, koska tunnemme niiden arvot etukäteen. Kappaleen 8.14 (sivu 294) esimerkissä käsittelemme reunaehtoja tähän tapaan.

Luonnollinen reunaehto määrää esimerkiksi lämpövuon reunan yli:

$$-k\frac{\partial T}{\partial n} = q.$$

Tästä seuraa yhtälön diskretoituun muotoon integraali

$$\int_{\Gamma} qW \, d\Gamma. \tag{8.7}$$

Esimerkiksi lämpöyhtälössä vuo reunan yli voi myös riippua lämpötilasta reunalla, jolloin reunaehdosta tulee termejä sekä kerroinmatriisiin että oikean puolen vektoriin. Vuo saattaa lisäksi riippua reunan lämpötilasta epälineaarisesti. Tällaisesta tilanteesta kerrotaan lisää kappaleessa 8.5.

8.5 Yhtälön linearisointi

Esimerkiksi Poissonin yhtälö on *lineaarinen* elliptinen osittaisdifferentiaaliyhtälö (ODY). Elementtimenetelmä Galerkinin formulaatiolla soveltuu erityisen hyvin tällaisten tehtävien ratkaisuun.

Määritelmä 8.5.1 (ODY:n lineaarisuus) Osittaisdifferentiaaliyhtälöä kutsutaan lineaariseksi, jos se on lineaarinen riippuvan muuttujan ja kaikkien sen derivaattojen suhteen ja kaikkien derivaattojen kertoimet ovat vain riippumattomien muuttujien funktioita.

Esimerkiksi lämpöyhtälö

$$\rho c_p \frac{\partial T}{\partial t} = \nabla \cdot (k \nabla T) + \rho h.$$

on siis lineaarinen vain, jos materiaaliparametrit (tiheys, lämmönjohtavuus ja ominaislämpökapasiteetti) ovat lämpötilariippumattomia. Jos myös reunaehdot ovat lineaarisia, käytämme nimitystä lineaarinen tehtävä. Esimerkiksi lämpöyhtälön tapauksessa lämpövuo reunalla saa riippua korkeintaan lineaarisesti lämpötilasta.

Lineaarisen tehtävän FEM-diskretointi johtaa suoraan lineaarisen yhtälöryhmän ratkaisuun. Lineaaristen tehtävien käsittelyyn olemme jo perehtyneet tässä luvussa. Lineaaristen yhtälöryhmien ratkaisumenetelmiä on esitelty luvussa 3.

Käytännössä fysikaalisesti kiinnostavat ODY:t ovat usein epälineaarisia. Osittaisdifferentiaaliyhtälön epälineaarisuus voi seurata joko itse säilymislaista tai konstitutiivisista eli materiaalirelaatioista. Esimerkiksi liikemäärän säilymislaista johdettu Navierin ja Stokesin yhtälö on aina epälineaarinen ODY. Toisaalta energian säilymislaista johdetun lämpöyhtälön tekee epälineaariseksi vasta jokin konstitutiivinen relaatio, kuten lämmönjohtavuuden lämpötilariippuvuus $k = k(T)$.

Epälineaaristen tehtävien FEM-diskretointi voidaan mieltää kahdella tavalla. Ajasta riippumattoman tehtävän voidaan ajatella johtavan epälineaarisen yhtälöryhmän ratkaisuun:

$$A(\mathbf{q})\mathbf{q} = \mathbf{F}(\mathbf{q}).$$

Yllä $\mathbf{q}$ käsittää ratkaistavat interpolointikertoimet, esimerkiksi lämpötilan solmupisteittäiset arvot. Yhtälöryhmä on kirjoitettu muodollisesti matriisiyhtälönä, jossa alkiot riippuvat kuitenkin ratkaisusta. Epälineaarisen yhtälöryhmän ratkaisuun soveltuvia menetelmiä käsitellään kappaleessa 6.4 (sivu 199), ja nämä palautuvat sarjan lineaarisia yhtälöryhmiä ratkaisuun

Toisaalta voimme linearisoida jo osittaisdifferentiaaliyhtälössä ja reunaehdoissa esiintyvät epälineaariset termit. Tämä tarkoittaa, että käytämme ratkaistavan muuttujan ensimmäistä astetta korkeammissa termeissä hyväksemme tunnettuja arvoja edellisestä iteraatista. Merkitsemme tässä luvussa linearisoituja termejä omalla kirjasinlajillaan, esimerkiksi lämpöyhtälössä symbolilla $\mathcal{T}$. Tämän arvo lasketaan käytännössä solmupistearvojen $\mathbf{T}$ edellisestä iteraatista. Täten linearisoitu tehtävä johtaa elementtimenetelmällä lineaariseen yhtälöryhmään iteraatille $\mathbf{T}^k$. Epälineaarisessa tehtävässä muodostamme tämän lineaarisen yhtälöryhmän iteroiden. Ratkaisun $\mathbf{T}^k$ suppenemiseen käytetään sopivaa kriteeriä.

Käytännössä nämä kaksi lähestymistapaa voivat johtaa samaan algoritmiin. Osittaisdifferentiaaliyhtälön linearisointi voi vastata esimerkiksi epälineaarisen yhtälöryhmän ratkaisua Newtonin menetelmällä. Vastaavaa linearisointimenetelmää, joka on hyvin käyttökelpoinen, kutsutaankin Newtonin linearisaatioksi.

Soveltamalla kohdassa 6.4.2 esitettyä Newtonin menetelmää edelliseen epälineaarisen yhtälöryhmään, saamme iteraatin $\mathbf{q}^{k+1}$ ratkaistavaksi yhtälöstä

$$J(\mathbf{q}^k)(\mathbf{q}^{k+1} - \mathbf{q}^k) = \mathbf{F}(\mathbf{q}^k) - A(\mathbf{q}^k)\mathbf{q}^k. \tag{8.8}$$

Edellisen yhtälöryhmän Jakobin matriisiksi saamme

$$J_{ij}(\mathbf{q}) = A_{ij}(\mathbf{q}) + \sum_l \frac{\partial A_{il}(\mathbf{q})}{\partial q_j} q_l - \frac{\partial F_i(\mathbf{q})}{\partial q_j}.$$

Esitämme seuraavaksi Newtonin linearisoinnin osittaisdifferentiaaliyhtälön epälineaariselle tilavuuslähde- tai -voimatermille $f(q)$ ja näytämme kuinka tämä vastaa edellä esitettyä Newtonin menetelmää. Aloitamme kirjoittamalla lausekkeen $f(q)$ pisteen $\mathcal{Q}$ ympäristössä lasketun Taylorin kehitelmän kaksi ensimmäistä termiä

$$f(q) \approx f(\mathcal{Q}) + (q - \mathcal{Q}) \left(\frac{\partial f}{\partial q} \right)_{\mathcal{Q}}.$$

Diskretoidessamme tämän Galerkinin menetelmällä saamme

$$\begin{aligned} \int_V f(q)\Psi_i \, dV &\approx \int_V f(\mathcal{Q})\Psi_i \, dV + \sum_j \int_V (q_j - \mathcal{Q}_j)\Psi_j \left(\frac{\partial f}{\partial q} \right)_{\mathcal{Q}} \Psi_i \, dV \\ &= \int_V f(\mathbf{q}^k)\Psi_i \, dV + \sum_j \int_V (q_j^{k+1} - q_j^k) \left(\frac{\partial f}{\partial q_j} \right)_{\mathbf{q}^k} \Psi_i \, dV. \end{aligned}$$

Otamme siis pisteeksi $\mathcal{Q}$ ratkaistavan muuttujan arvon edelliseltä iteraatiolta k. Viimeisen lausekkeen kaksi termiä vastaavat tilavuuslähteestä yhtälöön (8.8) oikealle puolelle ja Jacobin matriisin kautta vasemmalle puolelle

tulevia osuuksia. Newtonin linearisaatiota voidaan soveltaa vastaavaan tapaan Taylorin kehitelmästä vektoriarvoisista kenttämuuttujista koostuville ja vektoriarvoisille lausekkeille.

Linearisointimenetelmällä tarkoitetaan siis yleisesti (skalaarimuuttujan tapauksessa) sopivan approksimaation

$$f(q) \approx \hat{f}(q, \mathcal{Q})$$

valintaa, missä $\hat{f}$ on korkeintaan lineaarinen funktio muuttujasta q. Linearisointi on käytännössä suoraviivaisempi tapa ratkaista tehtävä kuin epälineaarisen yhtälöryhmän ratkaisumenetelmien käyttö. Annamme käytännön esimerkin linearisointimenetelmien käytöstä kappaleessa 8.12.5 (sivu 285).

Linearisointimenetelmiä voi keksiä itsekin. Toisaalta on myös yleisesti käytettyjä linearisointeja. Esimerkiksi Navierin ja Stokesin yhtälön konvektiotermille käytetään usein joko Picardin linearisointia

$$\mathbf{v} \cdot \nabla \mathbf{v} \approx \mathcal{V} \cdot \nabla \mathbf{v}$$

tai nopeammin suppenevaa mutta paremman alkuarvauksen vaativaa linearisointia

$$\mathbf{v} \cdot \nabla \mathbf{v} \approx \mathcal{V} \cdot \nabla \mathbf{v} + \mathbf{v} \cdot \nabla \mathcal{V} - \mathcal{V} \cdot \nabla \mathcal{V}.$$

Huomautettakoon tässä, että emme välttämättä tunne analyyttista lauseketta linearisoitavalle termille. Esimerkkinä voisi olla monimutkainen konstitutiivinen relaatio lämmönjohtavuudelle $k(T)$. Tällöin voimme joko käyttää kiintopisteiteraatiota $k(\mathcal{T})$ tai korvata Newtonin linearisoinnissa derivaatat numeerisilla differenssiarvioilla (katso kappaletta 6.4).

Epälineaarinen iteraatio voidaan lopettaa ratkaisuvektorin 2-normiin pohjautuvalla suppenemiskriteerillä

$$\frac{\|\mathbf{q}^{k+1} - \mathbf{q}^k\|_2}{\|\mathbf{q}^{k+1}\|_2} \leq \varepsilon.$$

Epälineaarista iteraatiota voi iteraatioiden välissä relaksoida parametrillä λ siten, että uusi ratkaisu $\mathbf{q}^{k+1}$ korvataan arvolla $\lambda \mathbf{q}^{k+1} + (1 - \lambda)\mathbf{q}^k$. Relaksaatioparametrin arvo on tyypillisesti välillä $0.5 \leq \lambda \leq 1$ ja sitä pienempi, mitä vaikeampi ratkaisu on pitää kasassa.

8.6 Aikaintegrointi

Haluamme ratkaista ajasta riippuvia osittaisdifferentiaaliyhtälöitä kahdesta syystä. Ensimmäinen ja tärkein syy on, että tehtävä on ajasta riippuva ja olemme kiinnostuneita systeemin dynamiikasta. Toinen syy on, että voimme käyttää ajasta riippuvaa formulaatiota ajasta riippumattoman ratkaisun hakemiseen. Tällöin systeemi ajetaan tasapainotilaan jostain kenttämuuttujien alkuarvosta. Jälkimmäisessä tapauksessa tehtävän ratkeavuus vaikeilla parametreilla parantuu usein verrattuna siihen, että tehtävää ratkaistaisiin ajasta riippumattomana samalla alkuarvauksella. (Toinen vaihtoehto on käyttää kappaleessa 8.12.5 sivulla 285 kuvattavaa kuormankasvatusta.)

8.6.1 Ajasta riippuvan tehtävän muoto

Ajan suhteen ensimmäisen kertaluvun tehtävä paikan suhteen diskretoidussa muodossa ja globaaliksi yhtälöryhmäksi koottuna on yleisesti muotoa

$$M(\mathbf{q})\frac{\partial \mathbf{q}}{\partial t} + A(\mathbf{q})\mathbf{q} = \mathbf{F}(\mathbf{q}). \tag{8.9}$$

Aikaderivaattatermiä kertovaa matriisia $M(\mathbf{q})$ kutsutaan sovellusalueesta riippuen esimerkiksi *massamatriisiksi* tai *kapasitanssimatriisiksi*. Se johdetaan kirjoittamalla ratkaistava muuttuja $q(\mathbf{x}, t)$ yritteellä

$$q(\mathbf{x}, t) \approx \sum_j q_j(t)\Psi_j(\mathbf{x}).$$

Osittaisintegrointia ei massamatriisin johdossa tarvita.

Edellisessä kappaleessa esitetyn lämpöyhtälön aikaderivaattatermi johtaa kappaleessa 8.3 kuvatussa Galerkinin menetelmässä globaaleihin matriisielementteihin

$$M_{ij} = \int_\Omega \rho c_p \Psi_j \Psi_i \, d\Omega.$$

Tämä massamatriisi voi riippua lämpötilasta tiheyden ρ tai ominaislämpökapasiteetin c_p kautta. Tällöin se linearisoidaan kuten muutkin lämpöyhtälön termit kappaleessa 8.5 esitetyin periaattein:

$$M\frac{\partial \mathbf{T}}{\partial t} + A\mathbf{T} = \mathbf{F}. \tag{8.10}$$

Tämä on ensimmäisen kertaluvun alkuarvotehtävä tavalliselle differentiaaliyhtälöryhmälle ajan suhteen. Ratkaisuun voi käyttää kaikkia tavallisia differenssimenetelmiä. Näitä ovat mm. Eulerin menetelmä, Rungen ja Kuttan menetelmä sekä moniaskelmenetelmät, joita on käsitelty luvussa 7. Aikaintegrointiin voi käyttää myös elementtimenetelmää.

Kaikista yllä mainituista aikaintegrointimenetelmistä on eksplisiittisiä ja implisiittisiä versioita. Elementtimenetelmään sovellettuna eksplisiittisillä menetelmillä saadaan ratkaisu uudella aika-askeleella eksplisiittisesti eli suoraan edellisen askeleen arvoista vain, jos massamatriisi M on lävistäjämatriisi. Implisiittisissä menetelmissä joudutaan aina ratkaisemaan eitriviaali yhtälöryhmä. Implisiittiset menetelmät ovat yleisesti ottaen stabiilimpia kankeille tehtäville ja sallivat pidemmän aika-askelen kuin eksplisiittiset menetelmät. Tehtävästä riippuen ja virhetarkastelusta huolehtien elementtimenetelmän yhteydessä voidaan käyttää molempia menetelmätyyppejä. Eksplisiittinen Eulerin menetelmä on kuitenkin monissa tehtävissä hyvin epästabiili ja vaatii laskennallisesti kohtuuttoman lyhyen aika-askelen käytön.

Käsittelemme tässä luvussa vain yleistä painotettua Eulerin menetelmää, jota kutsutaan myös mm. θ-formulaatioksi. Tavalliselle differentiaaliyhtälöryhmälle $\mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y})$ yleinen Eulerin menetelmä on

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \Delta t \left[\theta \mathbf{f}(t_n, \mathbf{y}_n) + (1-\theta)\mathbf{f}(t_{n+1}, \mathbf{y}_{n+1})\right].$$

Tästä saadaan erikoistapauksena arvolla $\theta = 1$ eksplisiittinen Eulerin menetelmä. Tapausta $\theta = 0$ kutsutaan yleensä implisiittiseksi Eulerin menetelmäksi, vaikka kaikki menetelmät $\theta \neq 1$ ovat implisiittisiä. Tapaus $\theta = 1/2$ on ainoa paikallisen virheen (katso kohta 8.6.2) suhteen toista astetta oleva Eulerin menetelmä, ja sitä kutsutaan toisinaan myös trapetsimenetelmäksi. Näitä kolmea erikoistapausta on käsitelty luvussa 7. On syytä huomata, että useat lähteet valitsevat parametrin toisinpäin, $\Theta = 1 - \theta$.

Linearisoituun yhtälöön (8.10) sovellettuna Eulerin menetelmä antaa

$$\frac{\mathbf{T}_i^k - \mathbf{T}_{i-1}}{\Delta t} = \theta M_{i-1}^{-1} \left[\mathbf{F}_{i-1} - A_{i-1}\mathbf{T}_{i-1}\right] + (1-\theta)(M_i^{k-1})^{-1} \left[\mathbf{F}_i^{k-1} - A_i^{k-1}\mathbf{T}_i^k\right].$$

Tässä alaindeksi i viittaa ratkaistavana olevaan aika-askeleeseen ja yläindeksi k epälineaariseen iteraattiin uusimmalla aika-askeleella. Ratkaisu $\mathbf{T}_{i-1}$ sekä matriisit M_{i-1} ja A_{i-1} ja voimavektori $\mathbf{F}_{i-1}$ tunnetaan edelliseltä askeleelta. Aika-askeleen i ratkaisun aikana joudumme päivittämään epälineaarista iteraattia $\mathbf{T}_i^k$ ja edellisestä iteraatista $\mathbf{T}_i^{k-1}$ riippuvia linearisoituja matriiseja M_i^{k-1} ja A_i^{k-1} sekä vektoria $\mathbf{F}_i^{k-1}$. Tämä epälineaarinen iteraatio suoritetaan samalla tavalla kuin esitimme ajasta riippumattomalle tehtävälle kappaleessa 8.5.

Oletamme seuraavassa merkinnän yksinkertaistamiseksi ja yhtenevästi esimerkkimme (kappale 8.12.6) kanssa, että massamatriisi on ajasta riippumaton. Tällöin termejä siirtelemällä saadaan

$$\left[\frac{1}{\Delta t}M + (1-\theta)A_i^{k-1}\right]\mathbf{T}_i^k = \frac{1}{\Delta t}M\mathbf{T}_{i-1} - \theta A_{i-1}\mathbf{T}_{i-1} + \theta\mathbf{F}_{i-1} + (1-\theta)\mathbf{F}_i^{k-1}.$$

Toisin sanoen ajasta riippuvan tehtävän ratkaisu muistuttaa ajasta riippumattoman tehtävän ratkaisua siten, että yllä hakasulkujen sisällä oleva lauseke on efektiivinen jäykkyysmatriisi ja oikean puolen lauseke efektiivinen voimavektori.

Massamatriisia M voidaan myös approksimoida diagonaalisena (mass lumping). Tällöin säästymme yhtälöryhmän ratkaisulta käytettäessä eksplisiittistä Eulerin menetelmää. Vastaavasti ajasta riippuvan massamatriisin tapauksessa diagonaalisella approksimaatiolla vältämme yleisessä Eulerin menetelmässä hankalan kertolaskun massamatriisin käänteismatriisilla M_{i-1}^{-1}. Suoraviivaisin massamatriisin approksimointi diagonaalimatriisilla on summata kunkin rivin elementit diagonaalille:

$$\widetilde{M}_{ii} = \sum_j M_{ij},$$

$$\widetilde{M}_{ij} = 0, \quad i \neq j.$$

Parempi tapa, erityisesti kun käytetään toisen tai korkeamman asteen elementtejä, on sijoittaa

$$\widetilde{M}_{ii} = \frac{\sum_{i,j} M_{ij}}{\sum_i M_{ii}} M_{ii},$$

$$\widetilde{M}_{ij} = 0, \quad i \neq j.$$

Elementtimenetelmällä voidaan ratkaista myös ajan suhteen toisen kertaluvun tehtäviä. Kuten luvussa 7 on esitetty, nämä voidaan aina palauttaa ensimmäisen kertaluvun tehtäviksi. Eräs vaihtoehto toisen kertaluvun aikaintegrointiin on Bossakin menetelmä, joka kuuluu Newmarkin menetelmäperheeseen [New59].

8.6.2 Virheen käyttäytyminen

Aikaintegrointimenetelmän paikallista virhettä kuvataan menetelmän asteella p, jolloin yhden aika-askeleen virhe on $\mathcal{O}(\Delta t^{p+1})$. Ratkaisun laatua kuvaa kuitenkin paremmin kokonaisvirheen kasautuminen eli numeerisen ja tarkan ratkaisun erotus annetun ajan t jälkeen. Trapetsimenetelmä ei esimerkiksi ole toisen asteen paikallisesta virheestään huolimatta optimaalinen kankeille tehtäville. Menetelmä voi johtaa oskilloivaan käyttäytymiseen ratkaisun ympärillä.

Eulerin menetelmien stabiilisuusominaisuuksia on käsitelty luvussa 7. Yleinen Eulerin menetelmä on ehdollisesti stabiili arvoilla $\theta > 1/2$. Toisin sanoen aika-askelella on tällöin maksimipituus, joka riippuu tehtävästä ja sen FEM-diskretoinnista. Arvoilla $\theta \leq 1/2$ Eulerin menetelmä on absoluuttisesti stabiili. Vaikka stabiilisuusvaatimus ei tällöin rajoita askelpituutta, esimerkiksi ratkaisun tarkkuudelle asetetut vaatimukset rajoittavat askelpituutta. Mitä implisiittisempi (pienempi θ) Eulerin menetelmä on, sitä tehokkaammin ratkaisun mahdollinen oskillointi vaimenee. Suppenemisnopeuden kannalta optimaalinen θ voi kuitenkin olla välillä $0 < \theta < 1/2$, minkä vuoksi Eulerin menetelmä kannattaa toteuttaa FEM-ratkaisijaan yleisessä painotetussa formulaatiossa.

Osa elementtimenetelmällä ratkaistavista osittaisdifferentiaaliyhtälöistä on itsessään epästabiileja, eli pienetkin erot alkuehdoissa kumuloituvat aikaintegroinnissa menetelmästä riippumatta. Tällöin ratkaisustakin kannattaa tutkia tilastollisia tunnuslukuja. Esimerkki tästä on Navierin ja Stokesin yhtälöt, kun Reynoldsin luku on suuri.

Moniaskelmenetelmät edellyttävät vakioaika-askelen käyttöä eivätkä siis suoraan sovellu adaptiiviseen aikaintegrointiin. Jos vakioaika-askelta voidaan käyttää tehtävässä, BDF-menetelmät (kappale 7.4.4 sivulla 232) ovat tehokkaita, tarkkoja ja helppoja implementoida FEM-sovellukseen.

8.7 Elementtityypit, kantafunktiot ja isoparametrinen kuvaus

Kerromme tässä kappaleessa käytännön kannalta tärkeimmistä elementtityypeistä, näiden kantafunktioista sekä isoparametrisesta kuvauksesta. Lisäksi kerromme miten suureiden integrointi ja derivointi tehdään isoparametrista kuvausta hyväksi käyttäen.

Elementtimenetelmässä arvioidaan laskettavaa suuretta, lineaarisella approksimaatiolla (katso kappaletta 4.2 sivulla 74), yksinkertaisten kantafunktioiden lineaarikombinaationa

$$u(\mathbf{x}) \approx \sum_k \Psi_k(\mathbf{x}) u_k,$$

missä vapausasteina u_k ovat usein ratkaistavan suureen arvot elementtien solmupisteissä. Vapausasteina voi esiintyä myös suureiden derivaattoja ja integraaleja. Kantafunktiot on lisäksi määritelty paloittain niin, että funktio Ψ_k määräytyy *elementtien kantafunktioista* N_i^n siten, että $\Psi_k(\mathbf{x}) = N_i^n(\mathbf{x})$ kunkin elementin n alueella. Indeksi k on vapausasteen indeksi globaalissa numeroinnissa ja i saman vapausasteen indeksi elementin sisäisessä numeroinnissa. Tässä tapauksessa voimme siis kirjoittaa suureen u kantafunktioapproksimaation myös elementtikohtaisesti:

$$u(\mathbf{x}) \approx \sum_i N_i^n(\mathbf{x}) u_i, \text{ elementin } n \text{ alueella.}$$

Jätämme seuraavassa kantafunktioiden yläindeksin yksinkertaisuuden vuoksi kirjoittamatta.

Elementit määritellään usein yksinkertaisessa peruskoordinaatistossa tai lokaalissa koordinaatistossa. Tästä on se etu, että yksittäisen elementtityypin elementit ovat kaikki identtisiä. Lokaali koordinaatisto valitaan myös siten, että integrointi elementin yli on mahdollisimman yksinkertaista. Todellisen elementin esittämiseen tarvitsemme kuvauksen $\boldsymbol{\xi} \to \mathbf{x}$, joka kuvaa lokaalin koordinaattivektorin todellisiksi koordinaateiksi. Voimme esittää tämän kuvauksen samaan tapaan interpoloivien kantafunktioiden avulla kuin laskettavan suureenkin, kunhan tunnemme solmupisteiden todelliset koordinaatit $\mathbf{x}_i$:

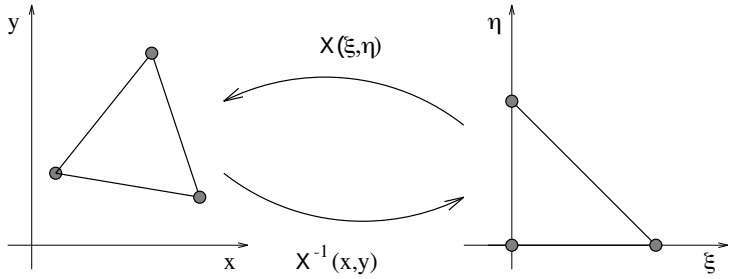
$$\mathbf{x}(\boldsymbol{\xi}) = \sum_i M_i(\boldsymbol{\xi}) \mathbf{x}_i. \tag{8.11}$$

Lausekkeessa (8.11) esiintyviä koordinaatteja $\boldsymbol{\xi}$ kutsutaan elementin lokaaleiksi koordinaateiksi. Elementti on peruskoordinaatistossaan huomattavasti yksinkertaisempi käsitellä kuin globaalissa koordinaatistossa, jossa se saattaa olla esimerkiksi kaareva. Jos koordinaattien $\mathbf{x}$ esittämiseen käytetään samoja kantafunktioita kuin ratkaistavalle suureelle $M_i = N_i$, puhutaan *isoparametrisista* elementeistä. Kuvassa 8.3 on havainnollistettu *isoparametrista kuvausta*.

Seuraavassa keskitymme kuvaamaan tilannetta, jossa yksittäisen elementin vapausasteina on vain laskettavan suureen arvoja solmupisteissä. Tilanteet, joissa vapausasteina on lisäksi esimerkiksi suureen derivaattoja solmupisteissä, eivät eroa perusperiaatteiltaan alla kuvatusta.

Elementtikohtaiset kantafunktiot saadaan määritetyksi interpolointiehdosta $u(\mathbf{x}_i) = u(\boldsymbol{\xi}_i) = u_i$. Siten

$$\begin{cases} N_i(\boldsymbol{\xi}_i) &= 1, \\ N_i(\boldsymbol{\xi}_j) &= 0, \ i \neq j, \quad i, j = 1, \ldots, n, \end{cases}$$



Kuva 8.3: *Isoparametrinen kuvaus.*

eli

$$N_i(\boldsymbol{\xi}_j) = \delta_{ij}, \quad i,j = 1,\ldots,n. \tag{8.12}$$

Tässä n on kantafunktioapproksimaation polynomitermien lukumäärä, joka on myös elementin solmupisteiden lukumäärä. Valitsemalla lokaalien koordinaattien määrittelyväliksi välin $[-1,1]$, kantafunktiot ovat siis tavanomaiset Lagrangen interpolointipolynomit.

Lagrangen interpolointipolynomit eivät ole ainoa mahdollinen valinta interpoloiviksi polynomeiksi, pikemminkin vain perinteinen. Legendren polynomeista johdetut hierarkkiset interpolointipolynomit ovat myös hyvä valinta elementtimenetelmän sovelluksiin.

Tietyntyyppisissä elementeissä saattaa lisäksi vapausasteina olla suureen derivaattoja, jolloin Lagrangen interpolointipolynomien sijaan käytetään Hermiten interpolointipolynomeja (kappale 4.8 sivulla 100), jotka määräytyvät ehtojen (8.12) sijaan ehdoista

$$\frac{d^l}{dx^l} H_{li}^k(\boldsymbol{\xi}_j) = \delta_{ij}, \quad l = 0,\ldots,k, \;\; i,j = 1,\ldots,n.$$

Tällöin myös ratkaistun suureen derivaatat astelukuun k asti ovat jatkuvia elementtien reunoilla. Lagrangen polynomit voidaan esittää Hermiten polynomien avulla:

$$L_i(\boldsymbol{\xi}) = H_{0i}^0(\boldsymbol{\xi}).$$

Yhtälö (8.12) on lineaarinen yhtälöryhmä, josta kantafunktion N_i termien kertoimet voidaan ratkaista. Esimerkiksi yksiulotteiselle elementille, jonka kantafunktiot ovat muotoa

$$N_i(\xi) = a_{i1} + a_{i2}\xi + \cdots + a_{in}\xi^{n-1},$$

yhtälö (8.12) on

$$\begin{pmatrix} 1 & \xi_1 & \cdots & \xi_1^{n-1} \\ 1 & \xi_2 & \cdots & \xi_2^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \xi_i & \cdots & \xi_i^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \xi_n & \cdots & \xi_n^{n-1} \end{pmatrix} \begin{pmatrix} a_{i1} \\ a_{i2} \\ \vdots \\ a_{ii} \\ \vdots \\ a_{in} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix}.$$

Kerroinmatriisi U on sama kaikille yksittäisen elementtityypin kantafunktioille N_i; vain oikean puolen vektori muuttuu indeksin i mukana. Yksiulotteisille elementeille kantafunktiot kannattaa kuitenkin laskea suoraan Lagrangen interpolointipolynomien lausekkeesta, joka on annettu kappaleessa 4.8 sivulla 100. Useampiulotteisille elementeille tässä kuvattu tapa on joissakin tilanteissa suoraviivaisin. Huomaa kuitenkin, että kerroinmatriisi on *Vandermonden matriisi*, joka on tyyppiesimerkki numeerisesti huonosti käyttäytyvästä matriisista, kun n on suuri.

8.7.1 Yksiulotteiset elementit

Yhdessä ulottuvuudessa yksinkertaisin elementti on lineaarinen jana. Tällä elementillä on kaksi solmua. Suureen vaihteluita siis yritetään kuvata elementin alueella polynomilla:

$$u(x) \approx a_1 + a_2 x.$$

Yhtälö voidaan kirjoittaa myös hiukan toisessa muodossa:

$$u(\xi) \approx N_1(\xi) u_1 + N_2(\xi) u_2.$$

Tässä on siirrytty elementin lokaaliin koordinaatistoon kuvauksella

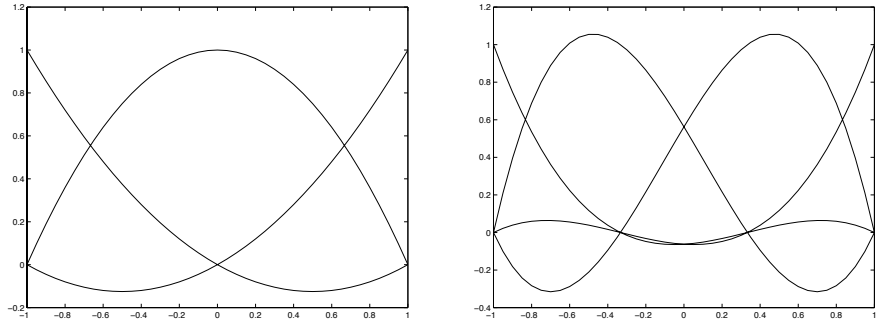
$$x(\xi) = N_1(\xi) x_1 + N_2(\xi) x_2$$

sekä valittu kantafunktiot siten, että muuttujan solmupistearvot u_1 ja u_2 ovat nyt suoraan kantafunktioiden kertoimina. Yleisin valinta elementin peruskoordinaatistolle on väli $-1 \le \xi \le 1$, jolloin kantafunktiot N_i ovat yksinkertaisesti

$$\begin{cases} N_1^{L1D}(\xi) = \frac{1}{2}(1 - \xi) \\ N_2^{L1D}(\xi) = \frac{1}{2}(1 + \xi). \end{cases} \tag{8.13}$$

Yläindeksinä on nyt käytetty elementin tyyppiä kuvaavaa lyhennettä, tässä tapauksessa ”lineaarinen yksiulotteinen elementti”. Käytämme tätä tapaa vain tässä kappaleessa erotellaksemme eri elementtityyppien kantafunktiot toisistaan. Muualla kantafunktioiden muoto selviää asiayhteydestä.

Samaan tapaan kuin suoralla suuretta voidaan arvioida elementin alueella toisen, kolmannen tai jopa korkeamman asteisilla polynomeilla (kuva 8.4). Solmut pyritään sijoittamaan elementin sisälle symmetrisesti. Tällöin esimerkiksi toisen asteen elementin vaatima kolmas solmu sijoitetaan yleensä elementin keskelle.



Kuva 8.4: *Neliöllisten ja kuutiollisten yksiulotteisten elementtien kantafunktioiden kuvaajat.*

8.7.2 Lineaarinen kolmioelementti

Useammassa ulottuvuudessa periaate on sama. Kuva 8.5 havainnollistaa yleisimpiä pintaelementtejä ja kuva 8.8 (sivu 266) yleisimpiä tilavuuselementtejä. Esimerkiksi lineaarisen kolmioelementin alueella tasossa voimme arvioida suuretta polynomilla

$$u(x, y) \approx a_1 + a_2 x + a_3 y$$

eli solmupistearvojen ja lokaalien koordinaattien avulla lausuttuna:

$$u(\xi, \eta) \approx N_1^{LK}(\xi, \eta) u_1 + N_2^{LK}(\xi, \eta) u_2 + N_3^{LK}(\xi, \eta) u_3,$$

missä $0 \leq \xi, \eta \leq 1$. Tässä on

$$x(\xi, \eta) = N_1^{LK}(\xi, \eta) x_1 + N_2^{LK}(\xi, \eta) x_2 + N_3^{LK}(\xi, \eta) x_3$$

ja y-koordinaatti on määritelty vastaavasti.

Seuraavassa käymme läpi kantafunktioiden ratkaisemisen interpolointiehdosta. Kirjoittamalla interpolointiehdon auki saamme

$$\begin{pmatrix} 1 & \xi_1 & \eta_1 \\ 1 & \xi_2 & \eta_2 \\ 1 & \xi_3 & \eta_3 \end{pmatrix} \begin{pmatrix} a_{11} \\ a_{12} \\ a_{13} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} a_{11} \\ a_{12} \\ a_{13} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix},$$

josta ratkaisemalla $a_{11} = 1$, $a_{12} = -1$ ja $a_{13} = -1$, eli

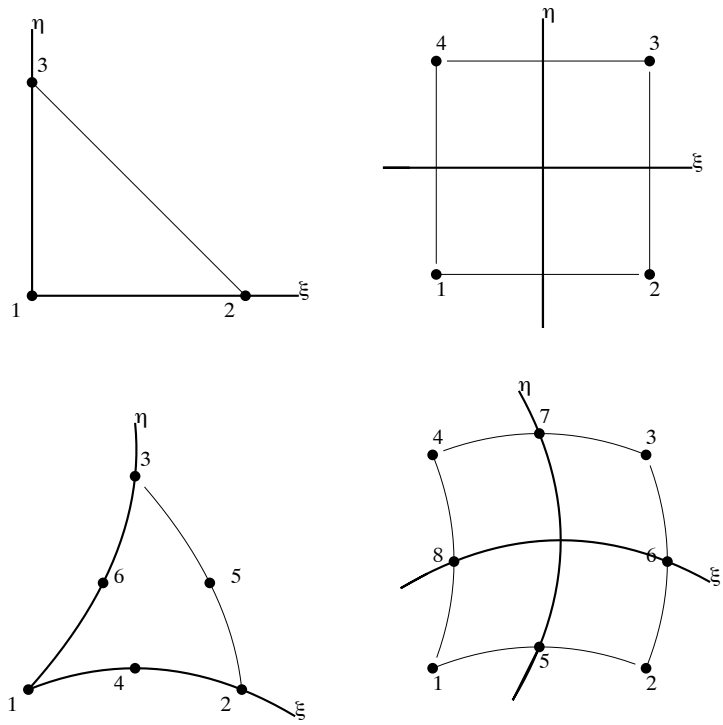
$$N_1^{LK} = 1 - \xi - \eta.$$

Perusmuotoisen kolmioelementin solmukoordinaateiksi olemme yllä valinneet pisteet

$$(\xi_i, \eta_i) = \{(0, 0), (1, 0), (0, 1)\}.$$

Tämä ei ollut sattumaa, vaan tällä valinnalla koordinaatit ξ ja η ovat kaksi kolmion kolmesta *alakoordinaatista* ϕ_i. Alakoordinaatit on määritelty kaavalla

$$\phi_i = A_i / A,$$



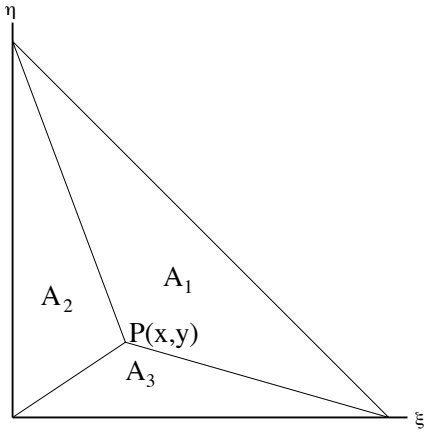
Kuva 8.5: *Yleisimpiä pintaelementtejä.*

missä kolmio on jaettu pisteestä $P = (x, y)$ kolmeen pienempään kolmioon (kuva 8.6), joiden pinta-alat ovat A_i. Koska pätee $A_1 + A_2 + A_3 = A$, kolmas alakoordinaatti ζ saadaan määrättyä ehdosta $\xi + \eta + \zeta = 1$. Alakoordinaattien käytöstä on etua esimerkiksi numeerisen integroinnin yhteydessä. Muut kaksi kantafunktiota saamme ratkaistua vastaavalla menettelyllä:

$$\begin{cases} N_1^{LK}(\xi, \eta) &= 1 - \xi - \eta \\ N_2^{LK}(\xi, \eta) &= \xi \\ N_3^{LK}(\xi, \eta) &= \eta. \end{cases}$$

Lineaarisen kolmioelementin kantafunktiot ovat siis tämän alakoordinaatit.

Eräs isoparametrisen kuvauksen hyvistä puolista on se, että vaikka pintaelementti ei olisi koordinaattitasossa, sama kantafunktioapproksimaatio suureiden vaihteluille pätee edelleen, kunhan esitämme kolmannenkin koordinaatin kantafunktioiden avulla. Lisäksi erityisesti numeerinen integrointi elementin alueen yli helpottuu oleellisesti, kun integrointirajat pysyvät aina samoina elementtien muodosta ja asennosta riippumatta.



Kuva 8.6: *Kolmion alakoordinaattien määrittely.*

8.7.3 Nelitahokas- ja tiiliskivielementit

Kolmion kantafunktiolle siis valittiin polynomitermeiksi vakiotermi sekä termit ξ ja η. Nelikulmioelementissä on neljä solmua, joten tälle elementille tarvitsemme yhden termin lisää. Luonnollinen valinta uudeksi termiksi on $\xi\eta$.

Voisimme nyt ratkaista bilineaarisen nelikulmioelementin kantafunktiot interpolointiehdoista. Helpommin näiden muodon kuitenkin näkee muodostamalla ne tensoritulona yksiulotteisen elementin kantafunktioista (8.13) seuraavasti:

$$N_k^{BL}(\xi,\eta) = N_i^{L1D}(\xi)N_j^{L1D}(\eta),\ i,j = 1,2,\ k = 1,\ldots,4.$$

Tästä saamme

$$\begin{cases} N_1^{BL}(\xi,\eta) &= \frac{1}{4}(1-\xi)(1-\eta) \\ N_2^{BL}(\xi,\eta) &= \frac{1}{4}(1+\xi)(1-\eta) \\ N_3^{BL}(\xi,\eta) &= \frac{1}{4}(1+\xi)(1+\eta) \\ N_4^{BL}(\xi,\eta) &= \frac{1}{4}(1-\xi)(1+\eta). \end{cases}$$

Nelikulmioelementin lokaalit solmukoordinaatit ovat siis

$$(\xi_i,\eta_i) = \{(-1,-1),(1,-1),(1,1),(-1,1)\}.$$

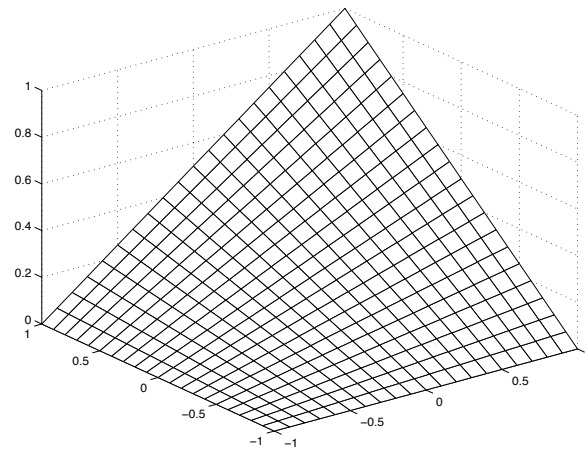
Bilineaarisen elementin solmua $(1,1)$ vastaavan kantafunktion kuvaaja on esitetty kuvassa 8.7.

Täysin vastaavasti saadaan kolmiulotteisessa tapauksessa trilineaariselle tiiliskivielementille kantafunktiot

$$N_i^{TL} = \frac{1}{8}(1+\xi_i\xi)(1+\eta_i\eta)(1+\zeta_i\zeta),\ i = 1,\ldots,8,$$

missä ξ_i, η_i ja ζ_i ovat elementin solmupisteiden lokaalit koordinaatit.

Yleisemmin kantafunktioiden termit pintaelementeille pyritään valitsemaan symmetrisesti muuttujien ξ ja η suhteen *Pascalin kolmiosta*



.

Kuva 8.7: *Bilineaarisen elementin solmua* $(1,1)$ *vastaavan kantafunktion kuvaaja.*

$$
\begin{array}{c}
1 \\
\xi \;\; \eta \\
\xi^2 \;\; \xi\eta \;\; \eta^2 \\
\xi^3 \;\; \xi^2\eta \;\; \xi\eta^2 \;\; \eta^3 \\
\xi^4 \;\; \xi^3\eta \;\; \xi^2\eta^2 \;\; \xi\eta^3 \;\; \eta^4 \\
\cdots
\end{array}
$$

Volyymielementeille termien valinta tehdään vastaavaan tapaan.

Kantafunktioiden symmetriaa elementin lokaalien koordinaattien suhteen kutsutaan joskus geometriseksi isotropiaksi. On kuitenkin tilanteita, joissa tähän ei tyydytä. Voisimme esimerkiksi haluta elementin, joka joltain sivultaan olisi kvadraattinen ja muilta sivuiltaan lineaarinen. Tällaisia elementtejä voidaan käyttää esimerkiksi, kun halutaan vaihtaa kantafunktiot korkeampiasteisiksi jollekin hankalalle osalle laskenta-aluetta.

8.7.4 Kvadraattinen nelitahokaselementti

Tensorituloa käyttämällä voimme muodostaa vain elementtejä, joissa on pintaelementeille n^2 tai volyymielementeille n^3 termiä, missä luku n on generoivan yksiulotteisen polynomin termien määrä. Jotkut solmuista tulevat tällöin elementtien sisälle. Esimerkkinä tästä on vaikkapa bikvadraattinen elementti, jonka kvadraattisen yksiulotteisen elementin kantafunktioiden

$$
\begin{cases}
N_1^{K1D} = \frac{1}{2}\xi(\xi - 1) \\
N_2^{K1D} = \frac{1}{2}\xi(\xi + 1) \\
N_3^{K1D} = (1 - \xi)(1 + \xi)
\end{cases}
$$

tensoritulona johdettuihin kantafunktioihin tulee 9 termiä:

$$
\begin{array}{c}
1 \\
\xi \;\; \eta \\
\xi^2 \;\; \xi\eta \;\; \eta^2 \\
\xi^2\eta \;\; \xi\eta^2 \\
\xi^2\eta^2
\end{array}
$$

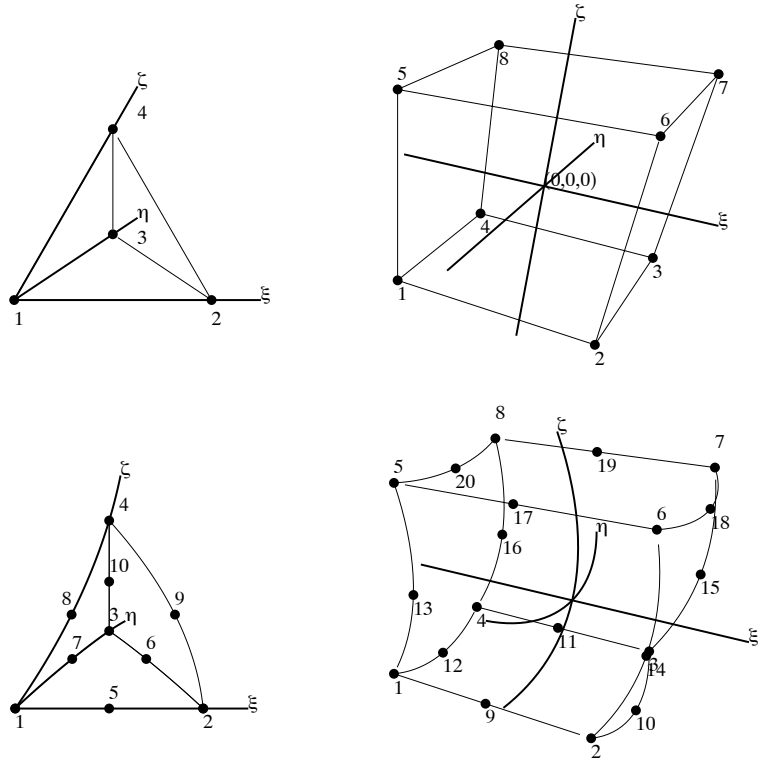
Yksi solmuista tulee elementin keskelle, mikä tietysti oli tilanne jo yksiulotteisellakin elementillä. Elementtien sisälle sijoitettuihin solmuihin liittyvät vapausasteet riippuvat pelkästään yksittäisen elementin omista vapausasteista ja voidaan haluttaessa eliminoida pois globaalista yhtälöryhmästä. Tätä vapausasteiden sisäistä eliminointia kutsutaan *staattiseksi kondensoinniksi.*

8.7.5 Kvadraattinen kolmioelementti

Kvadraattisen kolmioelementin (kuva 8.5) lokaalit solmukoordinaatit ovat

$$(\xi_i, \eta_i) = \{(0,0), (1,0), (0,1), (1/2,0), (1/2,1/2), (0,1/2)\}.$$

Tämän elementin kantafunktioiden muodon voi nähdä vaikkapa seuraavalla päättelyllä. Solmua 1 vastaavan kantafunktion tulee hävitä solmuissa, joissa



Kuva 8.8: *Yleisimpiä tilavuuselementtejä.*

$\xi_i \neq 0$ tai $\eta_i \neq 0$, toisin sanoen $\xi_i + \eta_i \neq 0$. Termi $1 - \xi - \eta$ hävittäisi kantafunktion pisteissä $\xi_i + \eta_i = 1$, ja termi $1 - 2\xi - 2\eta$ pisteissä $\xi_i + \eta_i = 1/2$. Näiden kahden termin tulo siis on hakemamme kantafunktio

$$N_1^{KK} = \gamma(1 - \xi - \eta)(1 - 2\xi - 2\eta).$$

Pisteessä $(\xi, \eta) = (0, 0)$ kantafunktion tulisi saada arvo 1, josta saamme $\gamma = 1$. Solmua 2 vastaavan kantafunktion pitäisi hävitä solmuissa, joissa $\xi_i \neq 1$, jolloin saamme kantafunktioksi

$$N_2^{KK} = \xi(2\xi - 1).$$

Vastaava päättely muiden kantafunktioiden osalta johtaa seuraaviin lausekkeisiin kvadraattisen kolmioelementin kantafunktioiksi:

$$\begin{cases} N_1^{KK} = (1 - \xi - \eta)(1 - 2\xi - 2\eta) \\ N_2^{KK} = \xi(2\xi - 1) \\ N_3^{KK} = \eta(2\eta - 1) \\ N_4^{KK} = 4\xi(1 - \xi - \eta) \\ N_5^{KK} = 4\xi\eta \\ N_6^{KK} = 4\eta(1 - \xi - \eta). \end{cases}$$

8.7.6 Derivointi globaalien koordinaattien suhteen

Elementtikohtaista yhtälöä laskettaessa tarvitaan usein suureen derivaattoja koordinaattien $\mathbf{x}$ suhteen. Isoparametrisille elementeille näitä ei tunneta suoraan, vaan ne on laskettava yhtälöstä

$$\frac{\partial u(\boldsymbol{\xi})}{\partial x^k} = \frac{\partial}{\partial x^k} \sum_i N_i(\boldsymbol{\xi}) u_i \;=\; \sum_i u_i \frac{\partial N_i(\boldsymbol{\xi})}{\partial x^k}.$$

Derivoinnin ketjusäännöstä saamme

$$\frac{\partial N_i(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}} \;=\; \sum_j \frac{\partial N_i(\boldsymbol{\xi})}{\partial x^j} \frac{\partial x^j}{\partial \boldsymbol{\xi}}$$

eli

$$\frac{\partial N_i(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}} = J \frac{\partial N_i(\boldsymbol{\xi})}{\partial \mathbf{x}},$$

tai

$$\frac{\partial N_i(\boldsymbol{\xi})}{\partial \mathbf{x}} = J^{-1} \frac{\partial N_i(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}.$$

Matriisi J on isoparametrisen kuvauksen (8.11) *Jacobin matriisi*:

$$J_{kl} = \frac{\partial x^l}{\partial \xi^k} \approx \sum_i x_i^l \frac{\partial N_i}{\partial \xi^k}.$$

Jos elementti on upotettu avaruuteen, jonka dimensio on suurempi kuin elementin lokaalin koordinaatiston, siis esimerkiksi viiva- tai pintaelementit kolmiulotteisessa avaruudessa, matriisi J^{-1} täytyy laskuissa korvata lausekkeella

$$M = J^T (J J^T)^{-1}.$$

Muutama esimerkki selventänee tilannetta.

Esimerkki 8.7.1 Viivaelementtien kantafunktioiden derivaatat globaalin koordinaatin x suhteen saadaan suoraan derivoinnin ketjusäännöstä:

$$\frac{dN_i}{dx} = \frac{dN_i}{d\xi}\frac{1}{dx/d\xi}.$$

Viivaelementille avaruudessa tilanne ei ole ihan näin yksinkertainen. Tällöin elementin Jacobin matriisi on 1×3 matriisi

$$J = \begin{pmatrix} \frac{dx}{d\xi} & \frac{dy}{d\xi} & \frac{dz}{d\xi} \end{pmatrix}.$$

Saamme nyt suureen N_i muutoksen koordinaattien $\mathbf{x}$ suhteen laskemalla

$$\begin{pmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial z} \end{pmatrix} = \frac{1}{L^2}\frac{dN_i}{d\xi}\begin{pmatrix} \frac{dx}{d\xi} \\ \frac{dy}{d\xi} \\ \frac{dz}{d\xi} \end{pmatrix},$$

missä $L^2 = (dx/d\xi)^2 + (dy/d\xi)^2 + (dz/d\xi)^2$. Lineaarisille elementeille lauseke on vakio, jolloin L on elementin pituus:

$$L = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}.$$

Esimerkki 8.7.2 Kvadraattisen kolmioelementin kantafunktioiden derivaatat lokaalien koordinaattien suhteen ovat

$$\frac{\partial N_i^{KK}}{\partial \xi} = \begin{cases} -3 + 4\xi + 4\eta \\ 4\xi - 1 \\ 0 \\ 4(1 - 2\xi - \eta) \\ 4\eta \\ -4\eta \end{cases} \qquad \frac{\partial N_i^{KK}}{\partial \eta} = \begin{cases} -3 + 4\xi + 4\eta \\ 0 \\ 4\eta - 1 \\ -4\xi \\ 4\xi \\ 4(1 - \xi - 2\eta) \end{cases}$$

Jos elementin globaalit solmukoordinaatit ovat $(x_i, y_i) = (2\eta_i, 2\xi_i)$, sen Jacobin matriisi on

$$J = \begin{pmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{pmatrix} = \begin{pmatrix} \sum_i x_i \frac{\partial N_i}{\partial \xi} & \sum_i y_i \frac{\partial N_i}{\partial \xi} \\ \sum_i x_i \frac{\partial N_i}{\partial \eta} & \sum_i y_i \frac{\partial N_i}{\partial \eta} \end{pmatrix} = \begin{pmatrix} 0 & 2 \\ 2 & 0 \end{pmatrix},$$

minkä olisi voinut aavistaakin. Yleisemmin kuitenkin Jacobin matriisin alkiot riippuvat evaluointipaikasta $J = J(\xi, \eta)$.

8.7.7 Integrointi elementin alueen yli

Kuten edellä on käynyt jo ilmi, elementtimenetelmässä integroidaan kerroinmatriisia muodostettaessa erilaisia lausekkeita yli elementtien alueiden.

Useimmiten integraalien laskeminen analyyttisesti on työlästä, jos ei jopa mahdotonta. Tällöin päädytään käyttämään numeerista integrointia. Numeerisesta integroinnista on kerrottu luvussa 5, joten kertaamme tässä vain, miten isoparametrinen kuvaus vaikuttaa integraalien laskentaan.

Suureen $u(\mathbf{x})$, jonka kantafunktioapproksimaatio on tavallista muotoa, integraali isoparametrisen elementin alueen yli muuntuu peruskoordinaatistoon kaavalla

$$\int_\Omega u(\mathbf{x})\, d\Omega \approx \sum_i u_i \int_\Omega N_i(\boldsymbol{\xi}) \det(J)\, d\boldsymbol{\xi}.$$

Avaruuden viiva- tai pintaelementille lauseke $\det(J)$ täytyy laskuissa korvata lausekkeella $\sqrt{\det(JJ^T)}$. Isoparametrisen muunnoksen Jacobin matriisin determinantti siis muuntaa integraalin käymään yli perusmuotoisen elementin todellisen elementin sijaan. Peruselementtien muoto on itse asiassa osin valittu juuri helpottamaan numeerista integrointia. Näin integraalien laskentaan voidaan helposti käyttää esimerkiksi *Gaussin kvadratuureja*.

Seuraavassa on muutama esimerkki numeerisesta integroinnista.

Esimerkki 8.7.3 Kvadraattisen yksiulotteisen elementin solmupisteet ovat

$$(x_i, y_i) = \{(-1,1), (1,1), (0,0)\}.$$

Integroimme tämän elementin yli integraalin

$$L = \int_0^L ds = \int_{-1}^1 \sqrt{\det(JJ^T)}\, d\xi \approx f\left(-\frac{1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right), \tag{8.14}$$

missä olemme merkinneet

$$f = \sqrt{\det(JJ^T)} = \sqrt{\left(\frac{dx}{d\xi}\right)^2 + \left(\frac{dy}{d\xi}\right)^2}.$$

Käytämme kahden pisteen Gaussin kvadratuuria integraalin (8.14) evaluoimiseksi. Koordinaatti s on käyränpituuskoordinaatti. Jacobin matriisi on tässä tapauksessa

$$J^T = \begin{pmatrix} \frac{dx}{d\xi} \\ \frac{dy}{d\xi} \end{pmatrix} = \begin{pmatrix} (\xi - 1/2)x_1 + (\xi + 1/2)x_2 - 2\xi x_3 \\ (\xi - 1/2)y_1 + (\xi + 1/2)y_2 - 2\xi y_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2\xi \end{pmatrix}.$$

Integraalin (8.14) arvo L saadaan laskemalla

$$L = 2f\left(\pm\frac{1}{\sqrt{3}}\right) = 2\sqrt{7/3} \approx 3.0551.$$

Tehtävän analyyttinen ratkaisu (käyrän $y = x^2$ pituus välillä $[-1,1]$) on

$$L = 2\int_0^1 \sqrt{1 + 4x^2}\, dx = \frac{1}{2}\operatorname{asinh}(2) + \sqrt{5} \approx 2.9579.$$

Tulimme siis tehneeksi integrointivirheen $e \approx 0.0972$, joka on noin 3 %. Tehtävän opetus on, että aivan näin käyräviivaisia elementtejä ei kannata käyttää. Jako kahteen osaan parantaa tilannetta oleellisesti. Tällöin integraalin arvoksi saadaan samalla kvadratuurilla kuin edellä $L \approx 2.9533$. Suhteellinen virhe on siis pienentynyt reilusti ja on nyt noin 0.15 %.

Esimerkki 8.7.4 Bilineaarisen nelitahokaselementin

$$(x_i, y_i) = \{(0.0, 0.0), (1.0, 0.0), (1.15, 1.0), (0.25, 1.0)\}$$

pinta-ala A saadaan integraalista

$$A = \int_{\Omega} d\Omega = \int_{-1}^{1}\int_{-1}^{1} det(J)\, d\xi d\eta \approx \sum f\left(\pm\frac{1}{\sqrt{3}}, \pm\frac{1}{\sqrt{3}}\right),$$

missä $f = \det(J)$. Elementin Jacobin matriisin determinantti on

$$f = \left|\frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial y}{\partial \xi}\frac{\partial x}{\partial \eta}\right|.$$

Derivaatat ovat

$$\frac{\partial x}{\partial \xi} = \frac{1}{4}\sum_{i=1}^{4} x_i \xi_i (1 + \eta_i \eta),$$

(muut vastaavasti). Laskemalla derivaattojen arvot integrointipisteissä saamme

$$A \approx 0.2447 + 0.2447 + 0.2303 + 0.2303 = 0.9500.$$

8.8 Lokaalin ja globaalin yhtälöryhmän kokoaminen

Ratkaistavan suureen paloittaisen approksimaation vuoksi voimme muodostaa yhtälön jokaiselle kantafunktioiden kertoimelle erikseen. Summaamalla eri elementeistä tulevat osuudet jokaista vapausastetta kohti saamme lopulta koottua yhtälön kokonaisuudessaan. Elementtikohtainen yhtälön diskretointi täytyy siis liittää samoihin solmuihin liitettyihin elementtien yhtälöihin. Näin saamme koottua *globaalin yhtälöryhmän*.

8.8.1 Lokaali yhtälöryhmä

Lokaalien yhtälöiden muodostaminen on tehtäväkohtaista, mutta joitain pääpiirteitä voidaan esittää yleisesti:

- Alustetaan elementtikohtaiset matriisi K ja oikean puolen vektori $\mathbf{F}$.

- Käydään läpi kaikki elementtien kantafunktiot N_i ja yritteen kantafunktiot N_j.
- Integroidaan (numeerisesti) elementtimatriisin alkiot:
  1. Käydään läpi integrointipisteet.
  2. Lasketaan funktioiden N_i ja N_j derivaatat muuttujien $\mathbf{x}$ suhteen integrointipisteessä.
  3. Lasketaan globaalien derivaattojen avulla elementin Jacobin matriisi ja tämän käänteismatriisi.
  4. Lisätään yhtälökohtaiset termit elementin matriisiin alkiolle $K_{ij} = K_{ij} + k$ ja voimavektorin alkiolle $F_i = F_i + f$.

Esimerkkimme yhtälölle lokaali matriisi on

$$K_{ij} = \int_{\Omega_e} \nabla N_j \cdot \nabla N_i \, d\Omega_e,$$

missä integraali on nyt elementin alueen yli. Indeksit i ja j ovat elementin solmujen (kantafunktioiden kertoimien) ja painofunktioiden indeksit.

Vastaavasti oikean puolen vektori saadaan laskemalla

$$F_i = \int_{\Omega_e} f N_i \, d\Omega_e.$$

Elementin lokaali yhtälö on siis tavallista muotoa

$$K\mathbf{T} = \mathbf{F}.$$

8.8.2 Globaalin yhtälöryhmän kokoaminen

Kantafunktioiden määrittelyn johdosta elementtikohtaiset matriisit voidaan laskea yhteen globaaliin yhtälöryhmään, kunhan tiedetään elementin solmujen yhteys globaaliin numerointiin. Pseudokoodina elementtimatriisin lisääminen globaaliin matriisiin voisi olla:

Algoritmi 8.8.1 (Globaalin matriisin kokoaminen)

```
for i = 1,...,n
    for j = 1,...,n
        A(Ind(i),Ind(j)) = A(Ind(i),Ind(j)) + K(i,j)
    end
end
```

Taulukko $\mathrm{Ind}(1:n)$ sisältää elementin solmujen globaalit indeksit. Vastaavasti voimavektorille saadaan:

Algoritmi 8.8.2 (Globaalin voimavektorin kokoaminen)

for $i = 1, \ldots, n$
  $b(\mathrm{Ind}(i)) = b(\mathrm{Ind}(i)) + F(i)$
end

Globaali yhtälöryhmä on siis standardimuotoa $A\mathbf{x} = \mathbf{b}$. Aivan näin yksinkertaista globaalin matriisin kokoaminen ei kuitenkaan yleensä ole, koska matriisi A on harva. Sen tallentamiseen kannattaakin käyttää harvojen matriisien tallennusmuotoja.

8.9 Virhearviot ja suppenemiskriteerit

Laajasti käsitettynä mallin virhe on laskennallisten tulosten ja mallitettavan todellisuuden välinen ero. Tämä ero voidaan määrittää eri mittareilla. Yleensä emme tietenkään edes tunne todellista ratkaisua tehtävälle, jonka numeerista ratkaisua haetaan. Virhelähteitä on käsitelty yleisesti kappaleessa 2.5 (sivu 31). Elementtimenetelmämallin virhe voi syntyä monella tavalla, esimerkiksi seuraavista lähteistä:

- yhtälöiden ja reunaehtojen valinta
- materiaaliparametrien arvot tai funktionaaliset riippuvuudet
- geometrian yksinkertaistus
- elementtiverkon tiheys ja kantafunktioiden aste
- numeerinen integrointi elementtien yli
- epälineaarisen iteraation ja lineaarisen yhtälöryhmän ratkaisun suppenemiskriteerit
- aikaintegroinnin paikallinen ja kumuloitunut virhe.

Luettelon kolme ensimmäistä tekijää muodostavat *matemaattisen mallin* ja aiheuttavat oman virheensä todellisuutta kuvattaessa. Neljä viimeistä tekijää vaikuttavat elementtiapproksimaation tarkkuuteen eli elementtimenetelmän antaman ratkaisun $q(\mathbf{x})$ eroon matemaattisen mallin tarkasta ratkaisusta $Q(\mathbf{x})$. Näiden virheiden suhteellista osuutta ja suuruutta tulisi kyetä arvioimaan.

Eräs vaihtoehto arvioida virhelähteitä on ratkaista tehtävä eri tavoilla ja verrata tuloksia. Tällä lähestymistavalla voidaan testata mallin sisäistä konsistenssia. Arvioimme soveltavassa esimerkissä kappaleessa 8.12 (sivu 278) näin hilan tiheyden ja aika-askeleen pituuden valinnan merkitystä. Huolellisemmassa tarkastelussa tulisi katsoa, suppeneeko ratkaisu kohti jotain arvoa hilaa tihennettäessä ja aika-askelta lyhennettäessä. Tämäkään ei aina takaa, että kyseinen raja-arvo hilasta ja aika-askeleesta riippuvalle elementtiapproksimaatioille $q_h(\mathbf{x})$ on matemaattisen mallin tarkka ratkaisu $Q(\mathbf{x})$.

Vastaavasti voimme kokeilla vaikkapa kantafunktioiden asteen ja integrointipisteiden lukumäärän vaikutusta. Kaikkia pois jätettyjä fysikaalisia termejä tai geometrian yksityiskohtia ei voi kuitenkaan sisällyttää malliin: mallista tuleekin jättää epäoleelliset tekijät pois. Fysikaalista intuitiota voi käyttää ratkaisun järkevyyden todentamiseen, esimerkiksi tarkastelemalla massan tai energian säilymistä.

Elementtimenetelmälle voidaan johtaa virhearvioita. Nämä voidaan jakaa *a priori* ja *a posteriori* -virhearvioihin. Edellisiä kutsutaan myös etukäteisvirhearvioiksi. Niissä yritetään FEM-ratkaisua käyttämättä arvioida virheen riippuvuutta esimerkiksi elementin koosta ja asteesta. Näissä arvioissa on tyypillisesti mukana tuntematon verrannollisuuskerroin, mikä tekee niistä epäkäytännöllisiä mittareita. Nyrkkisääntönä a priori -arviot kertovat, että tehtävälle, jonka ratkaisu on sileä ja siisti, elementtien asteen kasvattaminen (p-menetelmä) on nopeampi tapa pienentää virhettä kuin elementtien koon pienentäminen pitämällä interpolaation aste kiinnitettynä (h-menetelmä). Epäsäännölliselle ratkaisulle ei a priori -arvioilla saada eroa p- ja h-menetelmille. Verrattaessa p- ja h-menetelmiä voidaan karkeasti ottaen sanoa, että tehtävän ratkaisuaika on samaa suuruusluokkaa, kun vapausasteiden lukumäärä tehtävässä on sama.

A posteriori -virhearvioissa puolestaan käytetään laskettua FEM-ratkaisua ja arvioidaan sen eroa matemaattisen mallin tarkkaan ratkaisuun. Koska tarkkaa ratkaisua ei tiedetä, tämä tarkoittaa käytännössä FEM-ratkaisun sijoittamista takaisin differentiaaliyhtälön vahvaan muotoon ja tämän residuaalin laskemista numeerisesti. Näin saatua paikasta riippuvaa virhearviota voidaan käyttää myös laskentaverkon paikallisen tihennyksen kriteerinä. Tästä kerrotaan tarkemmin tehtävän adaptiivista ratkaisua käsittelevässä kohdassa 8.10.

Ratkaisua ja sen tarkkuuden paranemista voidaan käytännössä kuvata esimerkiksi seuraavilla suureilla (joista osa on normeja, joita käsitellään tarkemmin liitteessä A):

- Ratkaistavan suureen pistearvo $q(\mathbf{x})$ kiinnostavassa paikassa. Pistevirhe $e(\mathbf{x}) = q(\mathbf{x}) - Q(\mathbf{x})$ voi olla tulosten tulkinnan kannalta tärkein, mutta yleensä pistevirheille ei voida johtaa samanlaisia konvergenssiteoreemia kuin ratkaisua kokonaisuutena kuvaaville normeille.
- Ratkaisuvektorin $\mathbf{q}$ maksimiarvo eli normi $\|\mathbf{q}\|_\infty$.
- Ratkaisuvektorin 2-normi

  $$\|\mathbf{q}\|_2 = \sqrt{\sum_i q_i^2},$$

  joka kuitenkin riippuu oleellisesti verkosta. Jos ratkaisuvektorin 2-normi jaetaan solmupisteiden lukumäärällä, mitan riippuvuus verkosta pienenee selvästi.
- Ratkaisun $q(\mathbf{x})$ L_2-normi Sobolev-avaruudessa H_0

  $$\left[\int_\Omega [q(\mathbf{x})]^2 \, d\Omega\right]^{1/2}.$$

- Virhenormeille $\|e\| = \|q - Q\|$ voidaan johtaa etukäteisvirhearvioita.
- Fysikaalisesti mielekkään suureen integraali: esimerkiksi massavuon

$$\int_\Gamma \rho(\mathbf{x})\,\mathbf{v}(\mathbf{x}) \cdot \mathbf{n}\, d\Gamma$$

  tulisi olla mielivaltaisella reunalla Γ saman suuruinen kuin sisään- ja ulostulovirtauksessa, tai kuljetettavan aineen i kokonaismassan

$$\int_\Omega \rho(\mathbf{x}, t) c_i(\mathbf{x}, t)\, d\Omega$$

  tulisi säilyä lähteettömässä ja suljetussa tilavuudessa Ω aika-askeleesta toiseen.
- Usein puhutaan myös energianormista $\|\cdot\|_L$, joka on tehtävän bilineaarimuodon L_1-normi, siis lineaarisen lämpöyhtälön tapauksessa

$$\int_\Omega k \nabla T \cdot \nabla T\, d\Omega.$$

  Huomaa, että esimerkiksi tässä tapauksessa energianormilla ei ole energian dimensiota. Tietyin edellytyksin elementtimenetelmä antaa energianormin mielessä pienimmän virheen kantafunktioiden virittämässä avaruudessa.

Jos kiinnostava fysikaalinen suure riippuu ratkaisun derivaatoista, tämän sekä pisteittäinen että integroitu virhe voivat olla suurempia kuin itse ratkaisulle. Erityisesti on syytä olla varovainen, jos ratkaisu tai sen derivaatat sisältävät epäjatkuvuuskohtia. Korkeamman asteen kantafunktioiden käyttäminen yleensä parantaa myös johdettujen derivaattasuureiden tarkkuutta. Derivaatat voi myös tarvittaessa sisällyttää osaksi itse elementtimenetelmän ratkaisuvektoria, jos näille vaaditaan suurempaa tarkkuutta.

Ratkaisun eri iteraatiotasojen suppenemiskriteerejä voi puolestaan määritellä esimerkiksi seuraavasti:

- Lineaarisen yhtälöryhmän $A\mathbf{x} = \mathbf{b}$ iteratiivisen menetelmän suppenemisehtona voidaan käyttää normin mukaan laskettua taaksepäistä virhettä eli yhtälöä (3.25). Jos tässä käytetään matriisille Frobeniuksen normia, yhtälö (A.21), ja muille termeille 2-normia, saadaan siis esimerkiksi

$$\frac{\|A\mathbf{x} - \mathbf{b}\|_2}{\|A\|_F \|\mathbf{x}\|_2 + \|\mathbf{b}\|_2} \leq c_l.$$

- Epälineaarisen iteraation (ajasta riippumaton tehtävä tai yksi aika-askel) suppenemisehto yhden osittaisdifferentiaaliyhtälön tai useamman vahvasti kytketyn ODY:n muuttujalle $\mathbf{q}$ on

$$\frac{\|\mathbf{q}^{k+1} - \mathbf{q}^k\|_2}{\|\mathbf{q}^{k+1}\|_2} \leq c_e.$$

- Useamman heikosti kytketyn ODY:n tapauksessa sekä c_l että c_e voidaan määritellä tarvittaessa eri arvoiksi eri yhtälöryhmille. Ratkaisua ohjaavina parametreina voidaan myös määritellä, kuinka monta kertaa kutakin epälineaarista yhtälöryhmää korkeintaan iteroidaan peräkkäin.

Ratkaisun suppenemiseen voidaan vaikuttaa monin tavoin, esimerkiksi vaihtamalla linearisointimenetelmää, kun on tehty tietty määrä epälineaarisia iteraatiota tai kun annettu konvergenssikriteeri on saavutettu.

8.10 Adaptiivinen verkon tihentäminen

Virheanalyysit elementtimenetelmästä kertovat, että usein virhe on suuri paikallisesti kulmapisteiden tai muiden singulariteettien lähellä, kun taas varsinkin korkeampiasteiset elementit saattavat ratkaista sileän ratkaisun hyvin tarkasti jo pienelläkin määrällä elementtejä. Elementtimenetelmässä verkon tiheys on paikallisesti suhteellisen helposti muokattavissa. On siis järkevää pohtia, miten sijoittaa elementit laskenta-alueelle siten, että toisaalta elementtien määrä minimoituu ja toisaalta saadaan niin tarkka tulos kuin halutaan. A posteriori -virhearviot, yhdessä verkon paikallisen tihennyksen kanssa, antavat tähän mahdollisuuden.

Adaptiivisuus elementtimenetelmässä jaetaan usein kahteen luokkaan: h-adaptiivisuuteen ja p-adaptiivisuuteen. Käytännössä h-adaptiivinen ratkaisu tarkoittaa seuraavanlaista proseduuria:

1. Tehdään laskenta-alueeseen elementtiverkko.
2. Lasketaan ratkaisu.
3. Lasketaan ratkaistun kentän perusteella a posteriori -virhearvio e_Δ.
4. Jos e_Δ pienempi kuin annettu raja-arvo lopetetaan.
5. Tehdään paikallisen virhearvion η_τ perusteella päivitetty elementtiverkko ja jatketaan kohdasta kaksi.

Vaihtoehto verkon tihentämiselle on elementtien kantafunktioiden asteluvun kasvattaminen siellä missä virhearvio on korkeampi kuin annettu raja-arvo. Tätä kutsutaan p-adaptiivisuudeksi. Jos ratkaisu on sileä, p-adaptiivisuuden tiedetään suppenevan nopeammin kuin h-adaptiivisuuden. Jyrkkien kulmien yms. singulariteettien lähistöllä h-adaptiivisuus taas on useimmiten parempi vaihtoehto. Paras toteutus, joskin teknisesti hiukan hankala, olisi yhdistää nämä elementtimenetelmän adaptiiviset muodot.

Verkon muokkaamisen voi tehdä joko tihentämällä olemassa olevaa verkkoa tai tekemällä verkon kokonaan uusiksi, jos käytössä oleva verkongeneroija tukee esimerkiksi pisteittäin (tai nk. tausta-verkossa) laskenta-alueelle annettua paikallista tihennystä. Verkon tihentämisen virhearvion perusteella emme tässä käsittele enempää.

A posteriori -virhearvion tekemiseksi on useita menetelmiä, joista esitämme tässä yhden, niin sanotun eksplisiittisen estimaattorin. Periaatteessa tämä menetelmä tarkoittaa ratkaistun kenttämuuttujan sijoittamista alkuperäiseen vahvassa muodossa esitettyyn yhtälöön ja residuaalin laskemista. Voidaan osoittaa, että virheen energianormi on verrannollinen residuaalin duaalinormin kanssa:

$$\|e_\Delta\|_E = \|u_{\text{exact}} - u_\Delta\|_E \approx \|R_\Delta\|_D. \tag{8.15}$$

Tässä u_{exact} on yhtälön tarkka ratkaisu ja u_Δ on elementtimenelmän antama ratkaisu.

Residuaalin duaalinormia puolestaan voidaan arvioida diskreetisti:

$$\|R(u_\Delta, v)\|_\Delta^2 = \sum_{\tau\in\Delta} |\tau| \int_\tau R_\tau^2 d\tau + \sum_{e\in\Gamma} |e| \int_e J_e^2/2 \; de, \tag{8.16}$$

missä R_τ yhtälön vahvan muodon residuaali elementeittäin laskettuna, sekä J_e elementtien sivujen yli laskettujen luonnollisten reunaehtojen (osittaisintegroinnista tulevien reunatermien) hyppy elementtien rajojen yli. Alaindeksi Δ viittaa elementtijakoon. Termit $|\tau|$ ja $|e|$ ovat elementtien ja elementtien sivujen koot.

Käytännössä elementtimenetelmän residuaalin duaalinormia voidaan siis diskreetisti arvioida seuraavien termien avulla:

- Vahvan muodon residuaali elementeittäin laskettuna.
- Luonnollisen reunaehdon hyppy elementtien sivujen yli.
- Reunaehtoyhtälöiden residuaali. Dirichlet-ehdoille tämän termin voi ottaa nollaksi. Neumann-ehdoille residuaalin voi laskea suoraan elementeistä.

Esimerkiksi Poissonin yhtälölle elementtikohtainen virheindikaattori olisi seuraava:

$$\eta_\tau^2 = |\tau| \int_\tau R_\tau^2 d\tau + \sum_e |e| \int J_e^2/2 \; de \tag{8.17}$$

missä summa on elementin sivujen yli. Suureet $|\tau|$ ja $|e|$ ovat elementtien ja sivujen koot, R_τ on

$$R_\tau = \nabla \cdot (k\nabla u) + f \tag{8.18}$$

ja J_e on ratkaisun normaaliderivaatan hyppy elementin sivujen yli:

$$J_e = (k_+ \nabla u \cdot \vec{n})^+ - (k_- \nabla u \cdot \vec{n})^-. \tag{8.19}$$

Siis ulkoreunoille, joilla on annettu Dirichlet-ehto $u = u_\Gamma$, voidaan ottaa $J_e = 0$. Ulkoreunoille, joilla on annettu Neumann-ehto tyyppiä $-k\nabla u \cdot n = g$, voimme laskea vuon numeerisesti ja verrata sitä annettuun arvoon g.

Kokonaisvirhearvio on näin laskettuna

$$\|e_\Delta\|_E \leq C(\sum_\tau \eta_\tau^2)^{\frac{1}{2}}. \tag{8.20}$$

Elementtikohtaista virheindikaattoria η_τ voidaan nyt käyttää adaptiivisessa ratkaisijassa elementtiverkon tihentämiseen paikallisesti ja kokonaisvirhearviota $\|e_\Delta\|_E$ iteraation kontrollointiin. Virhearviossa on vielä vakiokerroin, joten tällaisenaan se ei suoraan kerro virheen ylärajan absoluuttista arvoa. Vakion voi yrittää määrittää elementtimenetelmän interpolointivirheen perusteella tai kokeellisesti tapauksista, joissa oikea ratkaisu tunnetaan.

Usein voidaan olettaa, että paikallinen virhe η käyttäytyy elementin koon h funktiona seuraavasti:

$$\eta = \alpha h^\lambda. \tag{8.21}$$

Tätä yhtälöä käyttäen voidaan muutaman iteraatiokierroksen jälkeen arvioida h_r, jolla virhe olisi alle annetun raja-arvon η_r. Näin voimme vauhdittaa menetelmän suppenemista.

Adaptiivisesta ratkaisusta on esimerkki tässä luvussa kappaleessa 8.12.7 sivulla 291.

8.11 Soveltava esimerkki

Esimerkki 8.11.1 Tarkastelemme ajasta riippuvaa lämmön johtumista kuvaavaa yhtälöä

$$\rho c_p \frac{\partial T}{\partial t} = \nabla \cdot (k \nabla T) + \rho h. \tag{8.22}$$

Haemme yhtälön ratkaisua tasoalueessa käyttäen karteesista koordinaatistoa. Kuvassa 8.9 näkyy laskenta-alueen geometria ja reunat.

Lämpöä johtavan materiaalin ominaisuuksia kuvaavat aineen tiheys ρ, ominaislämpökapasiteetti vakiopaineessa c_p, sekä lämmönjohtavuus k. Suure h on annettu lämmönlähde. Alueen reunoilla voidaan antaa lämpövuo q, joka voi myös riippua lämpötilasta reunalla. Jos lämpötila reunalla on tunnettu, voidaan lämpövuon sijaan antaa lämpötila, jolloin reunalla on vuoehdon sijaan voimassa yhtälö $T = T_\Gamma$.

Olkoot esimerkkimme reunaehdot annettu seuraavasti. Reunalla Γ_1 on voimassa vuoehto

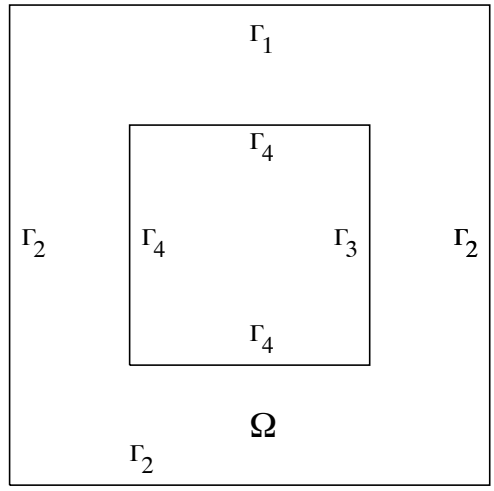
$$-k \nabla T \cdot \mathbf{n} = q,$$

missä $\mathbf{n}$ on reunalta ulospäin suunnattu yksikkönormaali. Lämpövuon q oletetaan konvektiiviselle lämmönsiirrolle olevan muotoa

$$q = \alpha(T - T_{\text{ext}})$$

sekä idealisoidulle säteilylle muotoa

$$q = \sigma \varepsilon \left(T^4 - T_{\text{ext}}^4\right).$$



Kuva 8.9: *Esimerkkitehtävän geometria.*

Huomaa että molemmat vuoehdot riippuvat lämpökentästä. Jälkimmäinen reunaehto on lisäksi epälineaarinen lämpötilan T suhteen. Parametri α on lämmönsiirtokerroin, T_{ext} ulkoinen lämpötila, σ Stefanin ja Boltzmannin vakio sekä ε reunapinnan emissiviteetti. Olkoot reunalla Γ_2 annettu Dirichlet'n ehto $T = T_2$, reunalla Γ_3 ehto $T = T_3$ sekä reunalla Γ_4 nollavuoehto (lämpöeristys). Esimerkissämme $T_3 > T_2 > T_{\text{ext}}$.

Käymme seuraavissa kappaleissa tämän esimerkin ratkaisun läpi yksityiskohtaisesti.

8.12 Esimerkkitehtävän ratkaisu

Aiemmissa kappaleissa olemme nähneet, miten yhtälöt saatiin diskretoitua painotettujen jäännösten menetelmällä. Tässä kappaleessa käymme yksityiskohtaisesti läpi esimerkkiyhtälön ratkaisun elementtimenetelmällä. Käytämme yhtälön diskretointiin lineaarisia kolmioelementtejä sekä reunoilla lineaarisia viivaelementtejä.

8.12.1 Elementtikohtaisten yhtälöiden rakentaminen

Yhtälön (8.22) lokaalin matriisin alkioiksi saamme

$$K_{ij} = \int_{\Omega_e} k \nabla N_j \cdot \nabla N_i \, d\Omega_e = \int_{\Omega_e} k \left(\frac{\partial N_j}{\partial x} \frac{\partial N_i}{\partial x} + \frac{\partial N_j}{\partial y} \frac{\partial N_i}{\partial y} \right) d\Omega_e.$$

Matriisielementtien lausekkeissa esiintyvät kantafunktioiden derivaatat globaalien koordinaattien suhteen saadaan laskettua yhtälöistä (katso kappaletta 8.7)

$$\frac{\partial N_i}{\partial x} = G_{11}\frac{\partial N_i}{\partial \xi} + G_{12}\frac{\partial N_i}{\partial \eta},$$
$$\frac{\partial N_i}{\partial y} = G_{21}\frac{\partial N_i}{\partial \xi} + G_{22}\frac{\partial N_i}{\partial \eta},$$

missä $G = J^{-1}$. Lineaaristen kolmioelementtien Jacobin matriisi on tapauksessa

$$J = \begin{pmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{pmatrix} = \begin{pmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \end{pmatrix}.$$

Tällöin Jacobin matriisin käänteismatriisi on

$$G = J^{-1} = \frac{1}{\det(J)}\begin{pmatrix} y_3 - y_1 & y_1 - y_2 \\ x_1 - x_3 & x_2 - x_1 \end{pmatrix},$$

missä $\det(J) = (x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1) = 2A$. Suure A on kolmion pinta-ala.

Suoraan laskemalla saamme kantafunktioiden globaaleille derivaatoille arvot

$$\frac{\partial N_i}{\partial x} = \begin{pmatrix} -G_{11} - G_{12} \\ G_{11} \\ G_{12} \end{pmatrix} = \frac{1}{2A}\begin{pmatrix} y_2 - y_3 \\ y_3 - y_1 \\ y_1 - y_2 \end{pmatrix} = \mathbf{N}^x,$$
$$\frac{\partial N_i}{\partial y} = \begin{pmatrix} -G_{21} - G_{22} \\ G_{21} \\ G_{22} \end{pmatrix} = \frac{1}{2A}\begin{pmatrix} x_3 - x_2 \\ x_1 - x_3 \\ x_2 - x_1 \end{pmatrix} = \mathbf{N}^y.$$

Huomaa, että osittaisderivaatat koordinaattien suhteen, ja täten myös Jacobin matriisi, ovat tälle yksinkertaiselle elementille vakioita elementin alueella. Tästä johtuen elementti-integraalien evaluointi on hyvin yksinkertaista. Käytämme keskipistesääntöä, jolloin

$$\int_{\Omega_e} f(\xi, \eta)\, d\Omega_e \approx A f(1/3, 1/3).$$

Elementtikohtaiset matriisialkiot ovat siis lopulta

$$K_{ij} = kA\left(N_j^x N_i^x + N_j^y N_i^y\right), \quad i, j = 1, 2, 3.$$

Oikean puolen vektorin alkioiksi saimme (kappale 8.8)

$$F_i = \int_{\Omega_e} \rho h N_i\, d\Omega_e,$$

joten lineaaristen kolmioelementtien tapauksessa saamme

$$F_1 = F_2 = F_3 = \frac{1}{3}\rho h A.$$

8.12.2 Vuoreunaehto

Esimerkissämme oli reunalla Γ_1 (kuva 8.9) annettu vuoehto

$$-k\frac{\partial T}{\partial n} = q.$$

Valitsemme vuoksi q konvektiivisen lämmönsiirron

$$q = \alpha(T - T_{\text{ext}}),$$

missä α on lämmönsiirtokerroin ja T_{ext} ulkoinen lämpötila. Sijoittamalla suureen q lausekkeeseen (8.7) saamme reunaintegraaliksi

$$\int_{\Gamma_1} \alpha(T - T_{\text{ext}})W \, d\Gamma.$$

Tämä yhtälö on voimassa vain reunalla Γ_1. Käytämme sen diskretointiin seuraavassa lineaarisia viivaelementtejä (katso kappaletta 8.7).

Vuoreunaehdosta tulee ratkaistavaan yhtälöön sekä matriisi- että vektorielementit

$$\begin{aligned} K_{ij} &= \int_{\Gamma_e} \alpha N_j N_i \, d\Gamma_e, \\ F_i &= \int_{\Gamma_e} \alpha T_{\text{ext}} N_i \, d\Gamma_e. \end{aligned}$$

Yksiulotteisen lineaarisen elementin Jacobin matriisi on 1×2-matriisi

$$J = \begin{pmatrix} \frac{dx}{d\xi} & \frac{dy}{d\xi} \end{pmatrix} = \frac{1}{2}\begin{pmatrix} x_2 - x_1 & y_2 - y_1 \end{pmatrix}.$$

Elementti-integraalien evaluoimiseksi tarvitsemme elementin pituuden

$$L = 2\sqrt{\det(JJ^T)} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}.$$

Integroimme reunaintegraalit kahden pisteen Gaussin kvadratuurilla

$$\int_{\Gamma_e} f(\xi) \, d\Gamma_e \approx \frac{L}{2}\left(f\left(-\frac{1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right)\right),$$

jolloin saamme kerroinmatriisin ja oikean puolen vektorin alkiot:

$$\begin{aligned} K_{11} = K_{22} &= \frac{2}{3}\alpha L, \\ K_{12} = K_{21} &= \frac{1}{3}\alpha L, \\ F_1 = F_2 &= \alpha T_{\text{ext}} L. \end{aligned}$$

8.12.3 Yhtälöryhmän kokoaminen

Globaalin yhtälöryhmän kokoaminen koostuu seuraavista askelista:

1. Alustetaan globaalit matriisit ja voimavektori nolliksi.
2. Käydään läpi laskenta-alueen elementtijako. Jokaiselle elementille lasketaan lokaali matriisi ja voimavektori, kuten edellä esitettiin. Lisätään ne globaaliin matriisiin ja voimavektoriin. Jos tehtävä on aikariippuva, myös aikaderivaatan käsittelystä syntyvät termit voi koota tässä vaiheessa. Aikaderivaattatermit voidaan koota joko omaan matriisiinsa tai lisätä ne samaan matriisiin paikkaderivaattatermien kanssa (katso kappaletta 8.6).
3. Käydään läpi reunaelementit, joille on määrätty vuoreunaehto. Lasketaan näiden lokaalit matriisit ja voimavektorit ja lisätään ne globaaliin matriisiin ja voimavektoriin.
4. Asetetaan Dirichlet'n ehdot kuten kappaleessa 8.4 esitettiin.

Näiden vaiheiden jälkeen ratkaistavanamme on yhtälöryhmä $A\mathbf{T} = \mathbf{F}$. Ratkaistavista yhtälöistä ja ratkaisualgoritmista riippuen yhtälöryhmä voi olla joko lineaarinen tai epälineaarinen.

8.12.4 Ratkaisu

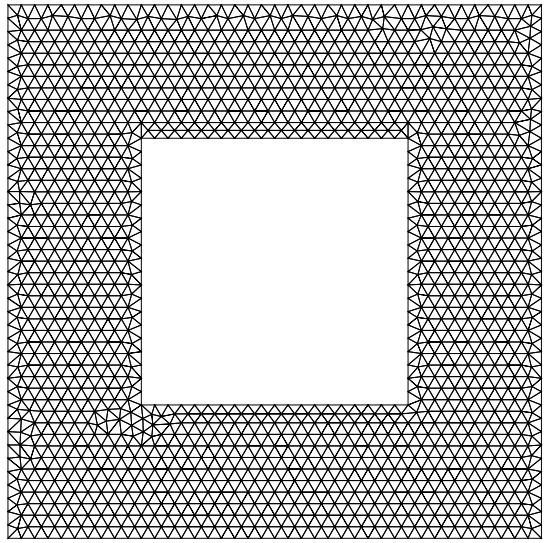
Yhtälöiden ratkaiseminen koostuu seuraavista askeleista

1. Muodostetaan laskenta-alueen elementtiverkko.
2. Alustetaan matriisirakenteet ja varataan tilaa ratkaisu- ja voimavektoreille. Tässä vaiheessa on yleensä tarpeen tehdä nauhanleveyden optimointi varsinkin, jos käytämme nauharatkaisijaa yhtälöryhmän ratkaisemiseen.
3. Jos tehtävä on aikariippuva, varataan aikaintegrointimenetelmän vaatima määrä työtilaa edellisten aika-askelten ratkaisuille ja voimavektoreille.
4. Kootaan ja ratkaistaan globaali yhtälöryhmä.
5. Jos tehtävä on epälineaarinen, hypätään takaisin kohtaan 4, kunnes asetetut suppenemiskriteerit on täytetty (kappale 8.5).
6. Jos ratkaisemme useampia riippuvia yhtälöitä heikosti kytkettynä, hypätään takaisin kohtaan 4, kunnes asetetut suppenemiskriteerit on täytetty (kappale 8.13).
7. Jos tehtävä on aikariippuva, siirrytään seuraavaan aika-askeleeseen ja jatketaan kohdasta 4 (kappale 8.6).

Ratkaisemme seuraavassa esimerkkitehtävän aikariippumattomana, toisin sanoen aikaderivaattatermi jää kokonaan pois yhtälöstä (8.22). Reunalla Γ_1 käytämme vuoreunaehtona konvektiivista lämmönsiirtoa $q = \alpha(T - T_{\text{ext}})$. Reunalla Γ_2 lämpötila pidetään arvossa 600 K ja reunalla Γ_3 arvossa 1200 K. Sisäistä lämpölähdettä ei ole, eli $h = 0$. Fysikaaliset parametrit on valittu seuraavasti.

Lämmönjohtavuus	$k = 1$ W/mK,
Ulkoinen lämpötila	$T_{\text{ext}} = 300$ K
Lämmönsiirtokerroin	$\alpha = 10$ W/m^{2}K

Näillä valinnoilla ratkaistava osittaisdifferentiaaliyhtälö on lineaarinen ja sen diskretointi johtaakin suoraan lineaarisen yhtälöryhmän ratkaisuun.



Kuva 8.10: *Laskennassa käytetty elementtiverkko.*

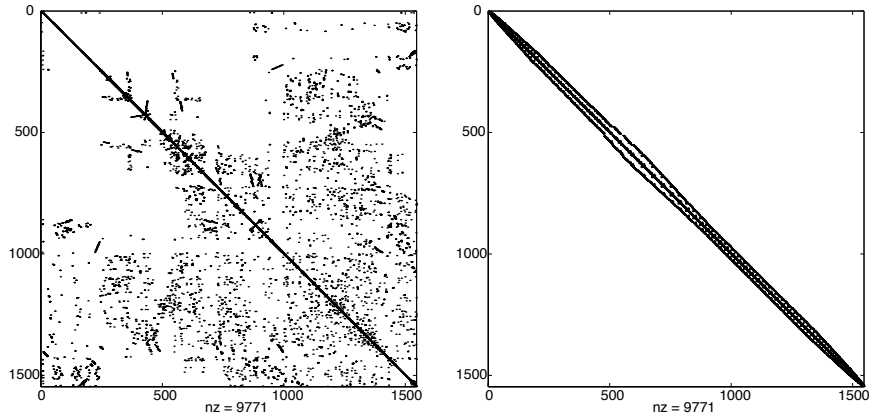
Laskennassa käytetty elementtiverkko on esitetty kuvassa 8.10. Solmupisteitä mallissa on 1548, kolmioelementtejä 2856 sekä viivaelementtejä reunoilla yhteensä 248.

Ratkaisemme syntyvän lineaarisen yhtälöryhmän sekä LU-hajotelmaan perustuvalla nauharatkaisijalla että iteratiivisella ratkaisijalla. Nauharatkaisijaa varten teemme nauhanleveyden optimoinnin käänteisellä Cuthillin ja McKeen algoritmilla. Algoritmi on pääpiirteittäin seuraava:

1. Valitaan lähtösolmuksi joku reuna- tai nurkkasolmu ja annetaan tälle uudessa numeroinnissa numero 1.

2. Numeroidaan järjestyksessä kaikki ne solmut j, joita ei vielä ole numeroitu ja joilla on yhteyksiä käsittelyssä olevaan solmuun i, eli kaikki ne solmut j joille $A(i, j) \neq 0$. Yleensä solmuilla i ja j on yhteisiä elementtejä, joihin molemmat kuuluvat.[2]

[2] Aina näin ei kuitenkaan ole. Esimerkiksi globaalin säteilyn mallinnuksessa saattaa käyte-



Kuva 8.11: *Matriisin rakenne ennen ja jälkeen nauhanleveyden optimoinnin.*

3. Jos kaikki solmut ovat tulleet numeroiduksi uudelleen, mennään kohtaan 4, muuten valitaan nykysolmuksi järjestyksessä seuraava solmu *uudessa numeroinnissa* ja jatketaan kohdasta 2.

4. Käännetään numerointi lopusta alkuun matriisin profiilin optimoimiseksi.

Algoritmi on intuitiivisestikin helppo ymmärtää. Pyrimme yksinkertaisesti vain siirtämään matriisin alkiot lähelle lävistäjää numeroimalla matriisin rivin nollasta poikkeavat alkiot mahdollisimman lähellä rivin indeksiä olevalla indeksillä. Puolinauhanleveys ennen nauhanleveyden optimointia oli tälle verkolle 1527, optimoinnin jälkeen 48 (kuva 8.11).[3] Nauhanleveyden pienentyminen vaikuttaa suoraan ratkaisuun kuluvaan aikaan. Ilman nauhanleveyden optimointia ratkaisuun kului tietokoneen keskusyksikköaikaa 3.36 s, optimoinnin jälkeen ratkaisuaika oli 0.03 s. Vapausasteiden määrän kasvaessa suhde kasvaa nopeasti vieläkin suuremmaksi.

Kokeillut iteratiiviset menetelmät olivat CGS-menetelmä (Conjugate Gradient Squared), TFQMR-menetelmä (Transpose Free Quasi Minimal Residual), stabiloitu bikonjugaattigradienttimenetelmä (BiCGStab) sekä GMRES-menetelmä (Generalized Minimal Residual). Pohjustimina käytimme lävistäjäpohjustusta sekä epätäydellistä LU-hajotelmaa (Incomplete LU Decomposition, ILU). Iteratiivisista yhtälöryhmän ratkaisumenetelmistä on kerrottu enemmän luvussa 3.

Lopetusehtona iteraatiolle käytettiin ehtoa

$$\frac{\|A\mathbf{T} - \mathbf{F}\|_2}{\|\mathbf{F}\|_2} \leq 10^{-8}.$$

tystä ratkaisualgoritmista riippuen reunasolmuilla olla matriisissa nollasta poikkeavia alkioita kaikkia mallin reunasolmuja kohden, vaikka solmut eivät kuuluisi edes samaan kappaleeseen saati sitten samaan elementtiin.

[3] Matriisin puolinauhanleveys r on suurin etäisyys lävistäjältä, jolla matriisissa on nollasta poikkeava alkio: $r = \max |j - i|$, siten että $A(i, j) \neq 0$.

Lämmönjohtumisyhtälöstä on tässä yhteydessä syytä huomauttaa, että se on helppo ratkaista myös iteratiivisilla ratkaisijoilla, toisin kuin monet muut yhtälöt kuten advektio-diffuusioyhtälö, Navierin ja Stokesin yhtälöt tai jotkut rakenteiden mekaniikan kuoritehtäviin liittyvät yhtälöt. Lisäksi yhtälö on symmetrinen ja olisi näin ratkaistavissa myös esimerkiksi tavallisella liittogradienttimenetelmällä.

Seuraavassa taulukossa on esitetty iteratiivisten menetelmien käyttäytymistä eri pohjustajilla. Taulukon alkio on n/t, missä n on iteraatioiden lukumäärä ja t suoritusaika sadasosasekunneissa. Suoritusajassa on mukana pohjustukseen kuluva aika.

Pohjustaja	*CGS*	*TFQMR*	*BiCGStab*	*GMRES*
Ei pohjustusta	85/11	83/16	50/13	6/25
Lävistäjä	106/15	105/25	40/12	6/22
ILU(0)	25/5	26/9	12/5	2/12
ILU(1)	18/5	20/8	8/5	1/8
ILU(2)	11/5	11/7	5/5	1/8

Huomattavaa on, että menetelmät tekevät eri määrän työtä iteraatiota kohti, kuten suoritusajoista on nähtävissä. Tehtävä on pienehkö ja helpohko, joten kovin suuria johtopäätöksiä tästä tehtävästä ei kannata tehdä. Bikonjugaattigradienttimenetelmä ILU(0)- tai ILU(1)-pohjustuksella näyttäisi kuitenkin toimivan varsin hyvin. Lävistäjäpohjustus ei juurikaan auta yhtälöryhmän ratkaisua, joissain tapauksissa se näyttäisi päinvastoin hidastavan suppenemista.

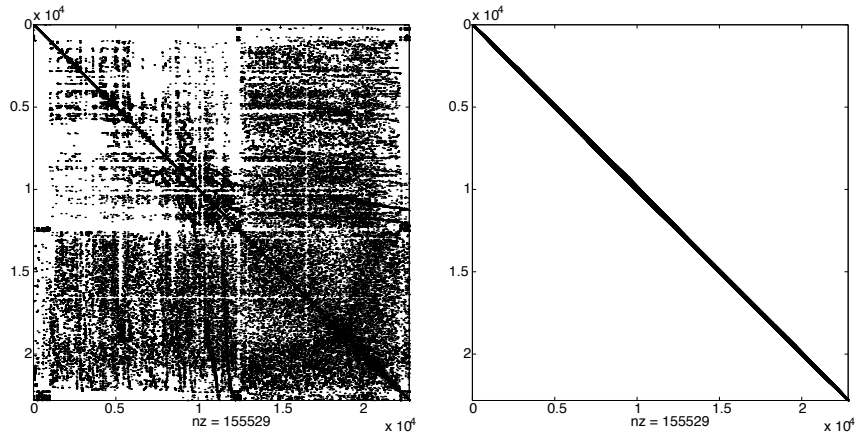
Laskimme saman tehtävän myös noin 15 kertaa isommalla verkolla (vapausasteita 22817). Alla on esitetty iteratiivisten menetelmien tulokset taulukkona. Suoritusajat ovat nyt sekunteja.

Pohjustaja	*CGS*	*TFQMR*	*BiCGStab*	*GMRES*
Ei pohjustusta	515/24	651/60	172/18	70/127
Lävistäjä	—/—	304/31	148/18	60/150
ILU(0)	122/10	116/17	47/9.7	10/23
ILU(1)	57/5.8	56/8.8	35/8.5	6/15
ILU(2)	38/4.5	40/8.3	21/5.9	4/11

Merkintä —/— tarkoittaa, että lasku ei ollut supennut tuhannella iteraatiolla. Tulokset ovat hyvin samansuuntaisia kuin pienemmälläkin verkolla.

Nauhanleveyden optimointi tuotti puolinauhanleveydeksi tällä verkolla arvon $r = 171$, kun alunperin $r = 22725$. Kuvassa 8.12 nauhanleveyden muutos näkyy selkeästi. Suoran ratkaisijan suoritusaika nauhanleveyden optimoinnin kanssa oli 4.2 s. Ilman muuttujien uudelleen numerointia kerroinmatriisi ei mahtunut käytettävissä olevaan keskusmuistiin (1 GB!) ja ratkaisuaikakin olisi ollut arviolta vähintään useita tunteja.

Kuva 8.13 esittää ratkaistua lämpökenttää. Ratkaisu näyttäisi olevan fysikaalisesti järkevän näköinen. Kappale jäähtyy ylälaidalta, jossa lämpöä siirtyy



Kuva 8.12: *Isomman yhtälöryhmän nollasta poikkeavien alkioiden sijoittelu matriisiin ennen ja jälkeen nauhanleveyden optimoinnin.*

kappaleesta ulos, koska ulkolämpötila on pienempi kuin kappaleen. Reunalla Γ_4, jossa oli määrätty nollavuoreunaehto, lämpötilan tasa-arvokäyrät ovat kohtisuorassa reunaa vastaan, toisin sanoen $\partial T/\partial n = 0$ tällä reunalla, kuten pitääkin. Kvantitatiivisia tarkistuksia kuten energian tai massan säilymisen varmistaminen, on myös suhteellisen helppo suorittaa ja näin olisikin suositeltavaa tehdä. Numeerisen ratkaisun järkevyys on kuitenkin aina tarkistettava, paitsi sisäisten konsistenssitarkistusten kautta, myös vertaamalla tuloksia vastaavaan fysikaaliseen tilanteeseen ja toisiin simulointeihin, jos vain suinkin mahdollista. Tästä aiheesta puhutaan enemmän seuraavissa kappaleissa. Ensin kuitenkin käsittelemme epälineaaristen ja aikariippuvien tehtävien ratkaisua.

8.12.5 Epälineaarisen tehtävän ratkaisu

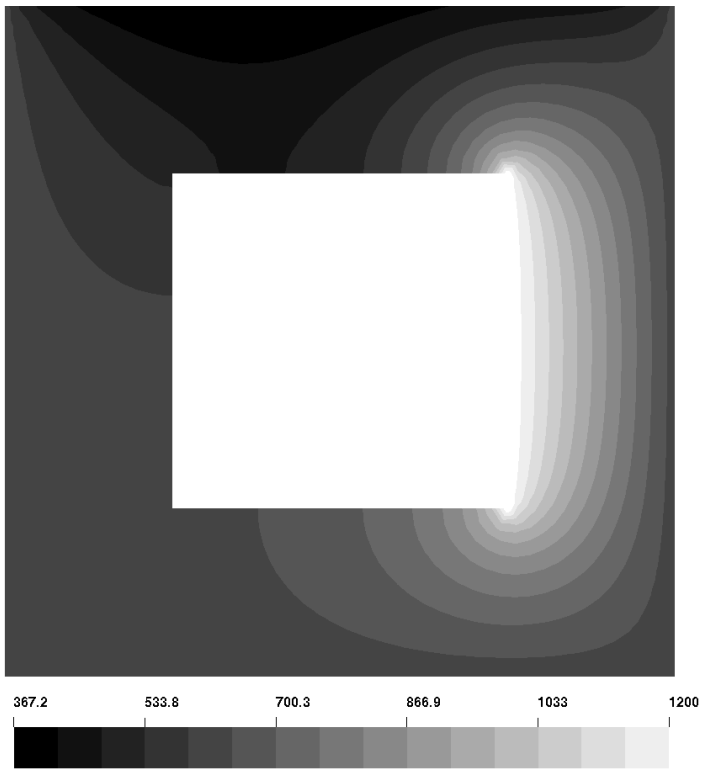
Esitämme tässä kappaleessa, kuinka epälineaarinen tehtävä ratkaistaan. Muutamme tätä varten esimerkin 8.22 reunaehdon reunalla Γ_1 (katso kuvaa 8.9) idealisoiduksi säteilyreunaehdoksi. Muilta osin tehtävän määrittely on sama kuin aiemmin. Esimerkki on edelleen ajasta riippumaton.

Säteilyreunaehdosta seuraava vuoreunaehdon epälineaarisuus johtaa epälineaarisiin reunaintegraaleihin. Nämä linearisoidaan samalla tavalla kuin tilavuusintegraalien (tai kahdessa dimensiossa pintaintegraalien) epälineaariset termit. Esimerkkimme antaa siis yleisen reseptin epälineaaristen tehtävien käsittelyyn.

Ainoa linearisoitava termi on siis idealisoitu säteilyreunaehto

$$-k\frac{\partial T}{\partial n} = \sigma\varepsilon\left(T^4 - T_{\text{ext}}^4\right).$$

Linearisoimme tämän termin ratkaistavan kenttämuuttujan T suhteen.



Kuva 8.13: *Esimerkkiyhtälön ratkaisu.*

Yksinkertaisinta olisi korvata koko T^4 edellisen iteraation arvolla $\mathcal{T}^4$. Tätä kutsutaan joskus kiintopisteiteraatioksi tai Picardin linearisoinniksi. Tästä seuraa FEM-formulaatiossa kontribuutio ainoastaan yhtälön oikean puolen vektoriin $\mathbf{F}$. Kiintopisteiteraatio suppenee varsin hitaasti.

Ensimmäistä kelvollista linearisointia varten kirjoitetaan säteilyvuo muodossa

$$-k\frac{\partial T}{\partial n} = \sigma\varepsilon\left(T^3 + T^2T_{\text{ext}} + TT_{\text{ext}}^2 + T_{\text{ext}}^3\right)(T - T_{\text{ext}})\,. \tag{8.23}$$

Otamme tässä $T = \mathcal{T}$ ensimmäisessä sulkulausekkeessa. Tämä linearisaatio muistuttaa konvektiivista lämmönsiirtoa reunalta sillä erotuksella, että lämmönsiirtokerroin on korvattu edellisestä iteraatista $\mathcal{T}$ riippuvalla lausekkeella. Kutsummekin tätä "lämmönsiirtokerroin"-linearisaatiota seuraavassa α-linearisaatioksi. Tämä on tehokkaampi menetelmä kuin kiintopisteiteraatio, ja sillä on useimmiten hyvä *suppenemissäde*, eli ratkaisun löytyminen ei ole kovin herkkä alkuarvaukselle.

Variaatiomuodon reunaintegraali vuoehdon reunalla Γ_1 on α-linearisaatiossa

$$\int_{\Gamma_1} q_n\Psi\,d\Gamma_1 = \int_{\Gamma_1} \sigma\varepsilon\left(\mathcal{T}^3 + \mathcal{T}^2T_{\text{ext}} + \mathcal{T}T_{\text{ext}}^2 + T_{\text{ext}}^3\right)(T - T_{\text{ext}})\,\Psi\,d\Gamma_1,$$

mistä saamme elementtikohtaiset kontribuutiot jäykkyysmatriisiin ja voimavektoriin:

$$K_{ij} = \int_{\Gamma_e} \sigma\varepsilon \left(\mathcal{T}^3 + \mathcal{T}^2 T_{\text{ext}} + \mathcal{T} T_{\text{ext}}^2 + T_{\text{ext}}^3\right) N_i N_j \, d\Gamma_e$$

$$F_i = \int_{\Gamma_e} \sigma\varepsilon \left(\mathcal{T}^3 + \mathcal{T}^2 T_{\text{ext}} + \mathcal{T} T_{\text{ext}}^2 + T_{\text{ext}}^3\right) T_{\text{ext}} N_i \, d\Gamma_e.$$

Newtonin linearisaatioksi kutsuttu vaihtoehto on kirjoittaa kaksi ensimmäistä termiä säteilyreunaehdon Taylorin kehitelmästä edellisen iteraation arvon $\mathcal{T}$ ympärillä:

$$-k\frac{\partial T}{\partial n} \approx \sigma\varepsilon \left(4\mathcal{T}^3 T - 3\mathcal{T}^4 - T_{\text{ext}}^4\right). \tag{8.24}$$

Tämä johtaa samaan iteraatioon kuin Newtonin menetelmä. Menetelmä suppenee nopeasti (neliöllisesti), mutta voi olla herkkä alkuarvaukselle.

Newtonin linearisaation tapauksessa variaatiomuodon reunaintegraali on

$$\int_{\Gamma_1} q_n \Psi \, d\Gamma_1 = \int_{\Gamma_1} \sigma\varepsilon \left(4\mathcal{T}^3 T - 3\mathcal{T}^4 - T_{\text{ext}}^4\right) \Psi \, d\Gamma_1,$$

mistä saamme elementtikohtaiset matriisi- ja vektorialkiot:

$$K_{ij} = \int_{\Gamma_e} 4\sigma\varepsilon\mathcal{T}^3 N_i N_j \, d\Gamma_e$$

$$F_i = \int_{\Gamma_e} \sigma\varepsilon \left(3\mathcal{T}^4 + T_{\text{ext}}^4\right) N_i \, d\Gamma_e.$$

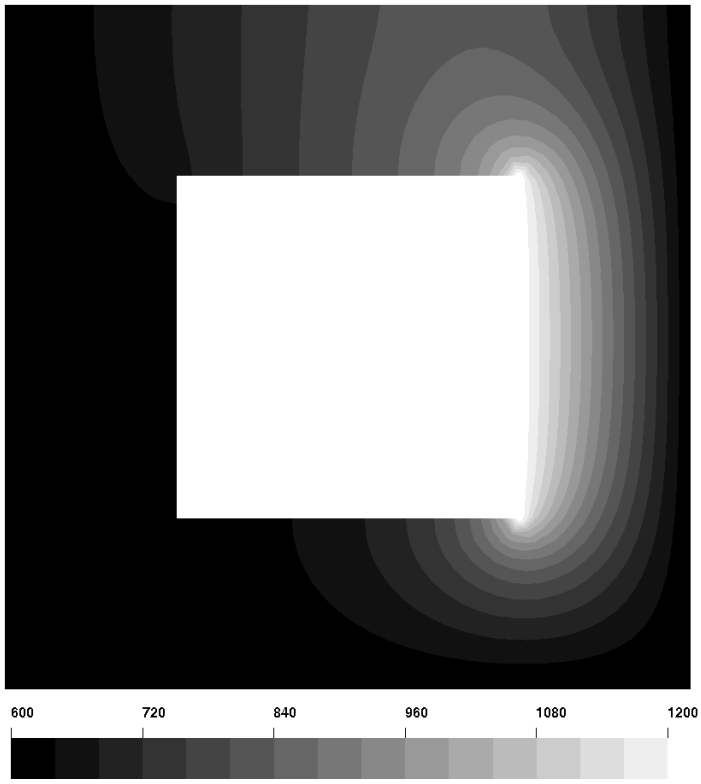
Usein käytännössä paras lähestymistapa on laskea alussa muutama askel yhtälön (8.23) linearisoinnilla ja käyttää sen jälkeen yhtälöä (8.24).

Ratkaisemme esimerkkimme nyt käyttäen kuvassa 8.10 esitettyä lineaarista kolmioelementtiverkkoa ja Galerkinin formulaatiota. Lopetamme epälineaarisen iteraation suppenemiskriteerillä

$$\frac{\|\mathbf{T}^{k+1} - \mathbf{T}^k\|_2}{\|\mathbf{T}^{k+1}\|_2} \leq 10^{-8}.$$

Esitämme kuvassa 8.14 lämpötilajakauman, kun lämmönjohtavuuden arvo on $k = 100$ W/mK. Ratkaisu eroaa lineaarisen tehtävän ratkaisusta lähinnä vuoreunan läheisyydessä kuvan yläosassa.

Seuraavassa taulukossa esitetään ratkaisuun tarvittavien iteraatioiden lukumäärä eri linearisointeja ja relaksaatioparametrin (katso kappaleen 8.5 loppua) arvoja käyttäen. Lämmönjohtavuudelle käytämme kahta arvoa: $k = 1$ W/mK ja 100 W/mK. Emissiviteetin arvo on $\varepsilon = 1$.



Kuva 8.14: *Lämpötilakenttä säteilyreunaehdolla*

	$k = 1$ W/mK		$k = 100$ W/mK	
	α	Newton	α	Newton
$\lambda = 1$	—	7	8	5
$\lambda = 0.8$	35	12	14	14
$\lambda = 0.6$	21	21	23	23
$\lambda = 0.5$	27	27	27	28

Merkintä ”—” tarkoittaa, että ratkaisu ei suppene ja jää oskilloimaan oikean arvon ympärillä. Tehtävä on helppo: kummallakin lämmönjohtavuuden arvolla nopeinta on aloittaa suoraan Newtonin linearisaatiolla ilman relaksaatiota. Lämmönsiirron lineaarisen osuuden kasvaessa α-linearisaatio tulee kuitenkin kilpailukykyiseksi.

Esimerkin ratkaisu on supennut annettua elementtiverkkoa ja diskretointia käyttäen kuvatun kriteerin mielessä. On kuitenkin syytä arvioida, riittääkö hilan tiheys kuvaamaan ratkaisua. Tämä on helpointa tehdä laskemalla ongelma tiheämpää hilaa käyttäen, esimerkiksi kaksinkertaisella määrällä elementtejä, ja vertaamalla tuloksia. Tavoitteenamme on, että käytetty verkko on riittävän mutta ei liian tiheä. Erityisesti jos suoritamme sarjan simulointeja, käytetty laskenta-aika voi kasvaa laskentaresurssien ylärajoille. Ongelman ratkaisu yhden kerran liian tiheää verkkoa käyttäen ei kuitenkaan lisää

laskentavaatimuksia liikaa.

Ratkaisemme tehtävän seuraavaksi 6466 elementillä. Visuaalisesti tarkasteltuna lämpötilajakauma näyttää samanlaiselta kuin harvemman verkon ratkaisu. Kenttäratkaisujen eroja kuvaava skalaarisuure, jonka ei tulisi riippua elementtiverkosta, on esimerkiksi lämpötilan L_0-normi Sobolev-avaruudessa H_0, eli integraali

$$I = \int_\Omega T(\mathbf{x})\, d\Omega.$$

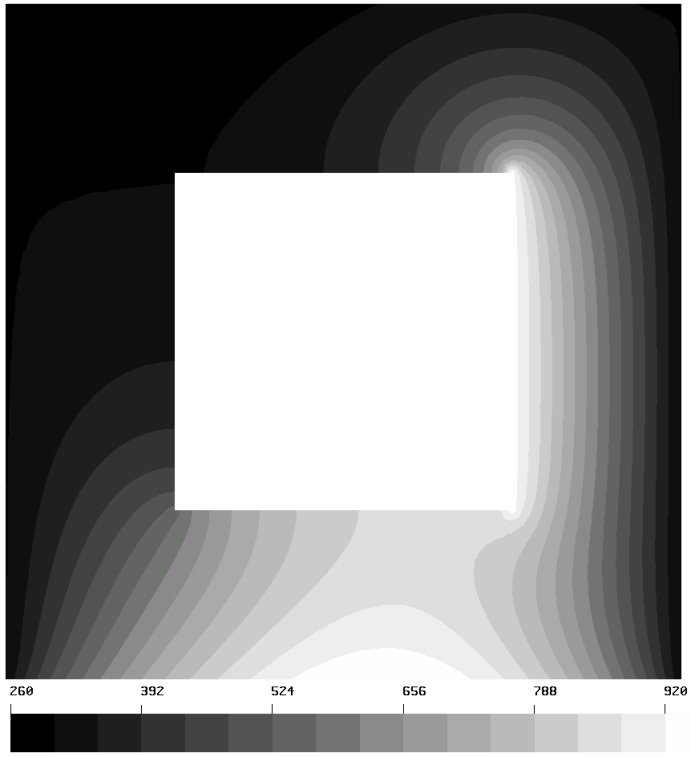
Laskemme tämän integraalin numeerisesti kolmen pisteen Gaussin kvadratuurilla. Harvemmalla hilalla saamme arvon $I = 309.282$ ja tiheämmällä $I = 307.900$, joten ratkaisut eroavat tämän mitan mielessä noin 0.45%. Hilan tihennys voi siis olla paikallaan. Huomattavasti taloudellisempaa on kuitenkin tihentää verkkoa adaptiivisesti vain siellä, missä se on ratkaisun kannalta tarpeen. Tästä kerrotaan kohdassa 8.10. Perusteellisemmassa tarkastelussa voisi myös käyttää sarjaa hiloja ja katsoa suppeneeko ratkaisu kohti jotain kenttää hilaa tihennettäessä. Kohdassa 8.9 käsiteltiin enemmän suppenemiskriteerejä ja virhearvioita.

Kun haemme ajasta riippumatonta tilaa epälineaariselle tehtävälle, voimme käyttää hyväksi myös kuorman asteittaista kasvatusta: usein jonkun parametrin (esim. virtauslaskennassa Reynoldsin luvun) tai reunaehdon muutos vaikeuttaa tehtävän ratkaisua, ja suppenemisongelmia voidaan välttää kasvattamalla tai pienentämällä parametria asteittain kohti oikeaa arvoa. Tällaista ratkaisustrategiaa kutsutaan *kuormankasvatukseksi*. Kuorman taso vaikuttaa myös linearisointimenetelmän ja relaksointiparametrin valintaan.

8.12.6 Ajasta riippuvan tehtävän ratkaisu

Muutamme nyt edellistä esimerkkiämme siten, että lämpövuo geometrian pohjalla on ajasta riippuva. Aloitamme edellä lasketusta ajasta riippumattomasta ratkaisusta. Kasvatamme välillä $t = 0 \ldots 10$ s lämpövuon nollasta lineaarisesti arvoon 1000 W/kg, josta pienennämme sen takaisin nollaan välillä $t = 10 \ldots 20$ s. Lämmönjohtavuudelle käytämme arvoa $k = 1$ W/mK, ominaislämpökapasiteetille $c_p = 1$ J/kgK ja tiheydelle $\rho = 1$ kg/m^3. Elementtiverkkona käytämme edellisen kohdan harvempaa kolmioverkkoa. Kuva 8.15 esittää lämpötilakenttää hetkellä $t = 10$ s.

Laskemme seuraavaksi 50 aika-askelta, joiden pituus on $\Delta t = 1$ s. Ratkaisemme tehtävän näin sekä implisiittistä Eulerin menetelmää että trapetsimenetelmää käyttäen. Molemmilla menetelmillä epälineaarinen yhtälöryhmä suppenee aika-askeleen sisällä muutamalla iteraatiolla ensimmäisen 20 s aikana, jolloin lämpövuo on ajasta riippuva, ja yhdellä iteraatiolla laskennan lopussa, kun tasapainotila on jälleen saavutettu. Vertaamme menetelmien antamia tuloksia ratkaisuvektorin normilla $\|\mathbf{T}_i\|_2$ — identtisillä elementtiverkoilla saman hetken t_i normit ovat yhtä suuret eri menetelmillä, jos ratkaisut ovat samat. Havaitsemme, että implisiittisen Eulerin ja trapetsimenetelmän ratkaisujen normit ovat 3–5 merkitsevällä numerolla samoja ensimmäisen 20 s aikana ja lopussa 11 merkitsevällä numerolla samat.



Kuva 8.15: *Ajasta riippuvan tehtävän lämpötilakenttä hetkellä $t = 10$ s.*

Eksplisiittisessä Eulerin menetelmässä virhe puolestaan kumuloituu, vaikka käytämme paljon lyhyempää askelta. Edes askelpituudella $\Delta t = 0.001$ s ratkaisu ei pysy kasassa vaan ylittää jo kymmenennellä askeleella liukulukuesityksen ylärajan. Askelpituudella $\Delta t = 0.0001$ s ratkaisu ei enää räjähdä, mutta lämpötilan normi pienenee vähitellen, vaikka sen pitäisi kasvaa. Tehtävä on selkeästi liian kankea eksplisiittiselle Eulerin menetelmälle. Voidaankin osoittaa, että kaksiulotteiselle elementtimenetelmällä diskretoidulle lämpöyhtälölle jouduttaisiin käyttämään stabiilin ratkaisun löytämiseksi aika-askelta, joka on luokkaa [Bur87]

$$\Delta t \leq \frac{\rho c_p}{k} \frac{\delta^2}{\pi^2},$$

missä δ on pienin solmupisteiden välinen etäisyys. Harvemmalla hilalla tämä asettaa luokkaa $\Delta t \leq 6 \cdot 10^{-5}$ olevan vaatimuksen aika-askeleelle. Erityisen ikävää on, että kun hilaa joudutaan tihentämään paikan suhteen, aika-askelta joudutaan lyhentämään suhteessa hilan neliöön.

Myös aikaintegroinnin suhteen on syytä tarkistaa diskretoinnin tarkkuus: onko $\Delta t = 1$ s tarpeeksi lyhyt aika-askel implisiittiselle Eulerin menetelmälle ja trapetsimenetelmälle? Yksinkertaisin lähestymistapa on laskea sama tehtävä lyhyemmällä aika-askeleella. Puolitetulla aika-askeleella $\Delta t = 1/2$ s tu-

lokset pysyvät implisiittistä Eulerin menetelmää käyttäen noin viidellä merkitsevällä numerolla samoina välillä $t = 0 \ldots 20$ s ja 13 merkitsevällä numerolla aika-integroinnin lopussa. Trapetsimenetelmän tulokset muuttuvat hiukan enemmän: puolitetulla aika-askeleella normi on ensimmäisen 20 sekunnin aikana noin kolmella merkitsevällä numerolla sama kuin pidemmällä aika-askeleella ja noin neljällä merkitsevällä numerolla sama kuin implisiittinen Eulerin menetelmän. Implisiittisen Eulerin menetelmän tuloksiin voinee siis luottaa 4–5 merkitsevän numeron tarkkuudella. Kokeilu eri arvoilla θ osoittaa, että arvoilla $\theta > 1/2$ Eulerin menetelmä on epästabiili, kun aika-askel $\Delta t = 1/2$ s.

Teoksessa [Bur87] suositellaan käytettäväksi arvoja $1 - \theta = 2/3,\ 3/4,\ 0.878$. Samaa aika-askelta käytettäessä näiden määrittämien menetelmien sekä implisiittisen ja trapetsimenetelmän vertailu osoittaa, että esimerkissämme trapetsimenetelmä vaatii noin kaksinkertaisen määrän iteraatioita kullakin aika-askeleella verrattuna muihin menetelmiin, jotka ovat keskenään suunnilleen yhtä hyviä.

8.12.7 Adaptiivinen ratkaisu

Laskimme adaptiivisella ratkaisijalla tässä luvussa aiemmin kuvatun esimerkin. Kuvissa 8.16 ja 8.17 on esitetty adaptiivsesta ratkaisusta saatu lopullinen verkko sekä vertailutuloksen saamiseksi käytetty tasavälisesti tihennetty verkko. Adaptiivisesti tihennettyyn verkkoon tuli 694 solmupistettä ja 1396 elementtiä. Tiheässä verkossa oli 24676 solmupistettä ja 49360 elementtiä.

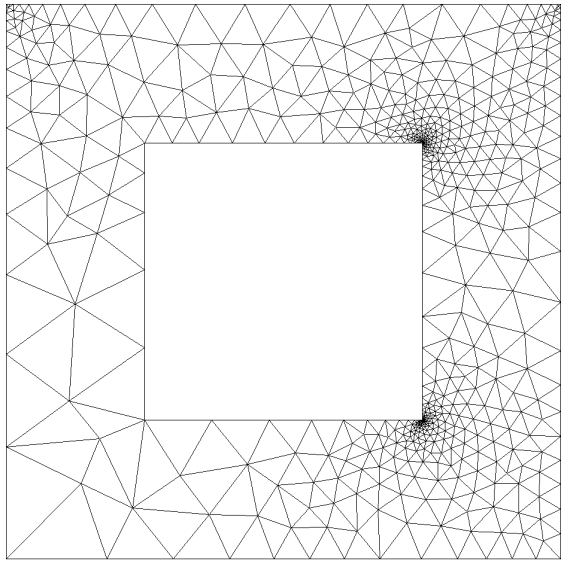
Luvussa 8.10 esitetyllä a posteriori -virhearviolla laskettu (suhteellinen) virhearvio oli adaptiiviselle verkolle 0.095, kun tasavälisesti tihennetylle verkolle virhearvio oli 0.188.

Adaptiivisesti tihennetty verkko antoi siis arvion mukaan tarkemman tuloksen noin 35 kertaa pienemmällä määrällä vapausasteita. Verkon tihennysten paikallisesta jakautumisesta huomataan, kuinka kulmapisteiden lähistöllä virheet ovat suuria verrattuna reunoista kauempana oleviin alueisiin.

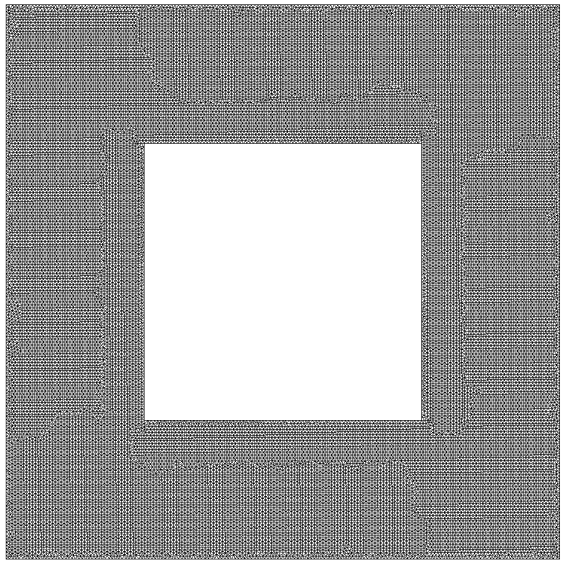
8.13 ODY-ryhmä: yhtälöiden kytkentä

Useissa sovelluksissa joudumme ratkomaan kytkettyjä osittaisdifferentiaaliyhtälöitä. Tyypillinen esimerkki on kytketty lämpötila- ja virtauskenttä. Virtaavaa ainetta kuvaavassa lämpöyhtälössä on konvektiotermin $\mathbf{v} \cdot \nabla T$ kautta kytkentä virtauskenttään. Lämmönsiirtoa voidaan kuvata kokoonpuristumattomassa nesteessä yhtälöllä

$$\rho c_p \left(\frac{\partial T}{\partial t} + \mathbf{v} \cdot \nabla T \right) = \nabla \cdot (k \nabla T) + \rho h.$$



Kuva 8.16: *Adaptiivisesti tihennetty elementtiverkko.*



Kuva 8.17: *Tiheä elementtiverkko adaptiivisen tuloksen vertailuun.*

Kun virtaus on kokoonpuristumaton ja viskositeetti μ vakio, Navierin ja Stokesin yhtälöt ovat

$$\nabla \cdot \mathbf{v} = 0,$$
$$\rho\left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v}\right) = -\nabla p + \mu \nabla^2 \mathbf{v} + \mathbf{f}.$$

Navierin ja Stokesin yhtälöt muodostavat (kahdessa dimensiossa) kolmen ODY:n kytketyn ryhmän kolmelle kenttämuuttujalle $(\mathbf{v}, p)$. Emme puutu tässä yksityiskohtaisesti Navierin ja Stokesin yhtälöiden linearisointiin emmekä FEM-diskretointiin. Yhtälöiden ratkaisuun soveltuvat stabiloidut elementtimenetelmät, joita on käsitelty kappaleessa 8.14. Oletamme seuraavassa, että Navierin ja Stokesin yhtälöt ratkaistaan vahvasti kytkettynä: niille kootaan yksi linearisoitu yhtälöryhmä. Tuntemattomina ovat nopeuskomponenttien ja paineen solmupistearvot.

Navierin ja Stokesin yhtälöissä virtauskenttä kytkeytyy lämpökenttään esimerkiksi tiheyden lämpötilariippuvuuden $\rho(T)$ kautta. Ratkaisemme edelliset yhtälöt kytkettynä *luonnollisen konvektion* kautta eli siten, että tiheyden vaihtelu gravitaatiovoimassa $\mathbf{f} = \rho\mathbf{g}$ kytkee yhtälöt. Tälle voimalle käytämme *Boussinesqin approksimaatiota*

$$\mathbf{f} = -\rho_0 \beta (T - T_0)\mathbf{g},$$

jossa lämpölaajenemista kuvaa lineaarinen kerroin β ja referenssitiheyden ρ_0 aiheuttama vakiopotentiaali on sisällytetty paineratkaisuun p. Kiinteillä seinillä nopeudet häviävät: käytämme siis kaikilla seinillä reunaehtoja $\mathbf{v} = \mathbf{0}$ (”no-slip”-reunaehto).

Esimerkissämme lämpöyhtälön sekä Navierin ja Stokesin yhtälöiden kytkentä on kaksisuuntainen eli molempien ratkaisut riippuvat toisistaan ja ne joudutaan ratkaisemaan joko iteroiden eli *heikosti kytkettynä* tai samanaikaisesti eli *vahvasti kytkettynä*. Käytämme ensimmäistä lähestymistapaa ja käsittelemme systeemien välistä iterointistrategiaa. Olkoon meillä siis linearisoidut yhtälöryhmät

$$M\frac{\partial \mathbf{T}}{\partial t} + A\mathbf{T} = \mathbf{F}, \quad \text{(lämpöyhtälö)}$$
$$N\frac{\partial \mathbf{q}}{\partial t} + D\mathbf{q} = \mathbf{G}, \quad \text{(Navierin ja Stokesin yhtälöt)}$$

missä $\mathbf{q}$ käsittää nopeuskomponenttien ja paineen solmupistearvot. Lämpöyhtälön matriisi A riippuu siis nyt myös nopeudesta termien

$$A_{ij} = \int_\Omega \rho c_p \mathbf{v} \cdot \nabla \Psi_j \Psi_i \, d\Omega$$

kautta. Vastaavasti Navierin ja Stokesin yhtälöiden vektori $\mathbf{G}$ riippuu lämpötilasta.

Esitämme seuraavassa kytkettyjen yhtälöiden iteraatioalgoritmin aika-askeleen $\Delta t = t_i - t_{i-1}$ (oletetaan tässä samaksi molemmille diskretoiduille yhtälöryhmille) tai ajasta riippumattoman ratkaisijan (eli muodollisesti

$\Delta t \to \infty$) sisällä. Aika-askeltaminen suoritetaan normaalisti tämän ulkopuolella. Otamme aikaintegrointimenetelmäksi seuraavassa yksinkertaisuuden vuoksi implisiittisen Eulerin menetelmän:

Algoritmi 8.13.1 (Kytkettyjen yhtälöiden ratkaiseminen)

1: Lämpöyhtälön epälineaarisen lämpötilariippuvuuden iterointi (indeksin k suhteen):

$$\frac{1}{\Delta t} M \mathbf{T}_i^k + A \mathbf{T}_i^k = \frac{1}{\Delta t} M \mathbf{T}_{i-1} + \mathbf{F}_i^{k-1}.$$

Tästä ratkaistaan $\mathbf{T}_i^k$ iteroiden n_T kertaa, tai kunnes iteraatio on supennut. Matriisi A riippuu viimeisimmästä nopeuden iteraatista (ajan hetkeltä t_i, jos saatavissa). Käytetään mahdollisesti alussa α-linearisointia ja vaihdetaan muutaman iteraation jälkeen tai suppenemisen alettua Newtonin menetelmään. Iteraatioiden välissä voi lämpötilan päivitystä relaksoida kappaleessa 8.5 esitetyllä tavalla.

2: Navierin ja Stokesin yhtälön epälineaarisen nopeusriippuvuuden iterointi (indeksi l):

$$\frac{1}{\Delta t} N \mathbf{q}_i^l + D \mathbf{q}_i^l = \frac{1}{\Delta t} N \mathbf{q}_{i-1} + \mathbf{G}_i^{l-1}.$$

Tätä iteroidaan n_q kertaa tai kunnes iteraatio on supennut. Vektori $\mathbf{G}$ riippuu viimeisimmästä lämpötilan iteraatista hetkeltä t_i. Käytetään mahdollisesti eri linearisointia ensimmäisillä iteraatioilla kuin lopussa. Iteraatioiden välissä voidaan relaksoida nopeuskomponenttien ja paineen päivitystä.

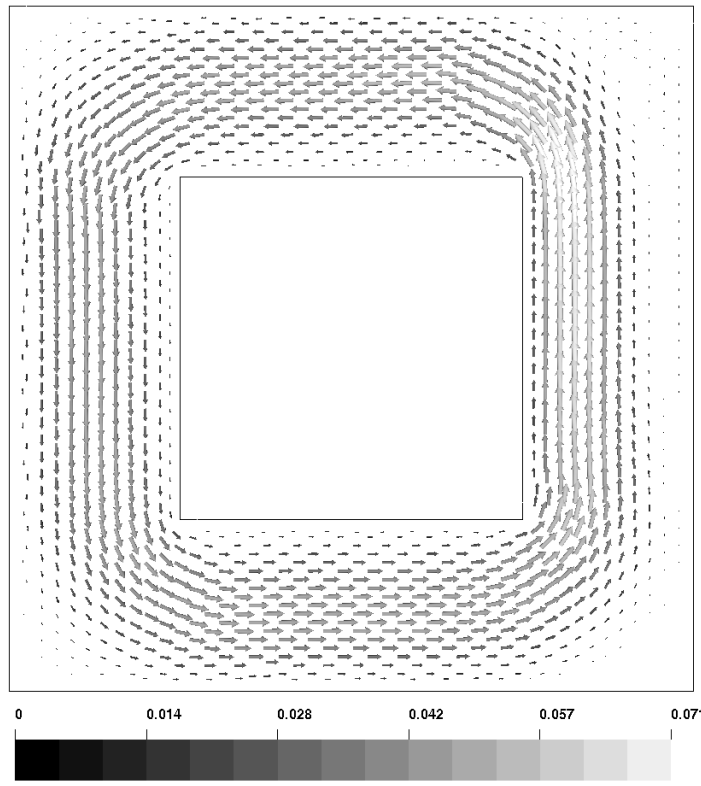
3: Jos molemmat suppenemiskriteerit on täytetty, siirrytään seuraavaan aika-askeleeseen. Muuten palataan kohtaan 1.

Esimerkkimme on ajasta riippumaton ja ratkaisemme sen seuraavalla strategialla: luetaan lämpötilan alkuarvoksi ajasta riippumattoman lämpöyhtälötehtävän ratkaisu, joka kuvaa kohtalaisen hyvin myös kytketyn tehtävän lämpötilakenttää. Iteroidaan osatehtäviä kerrallaan korkeintaan $n_T = 1$ ja $n_q = 3$ kertaa. Kumpikin osatehtävä voidaan linearisoida suoraan Newtonin menetelmällä eikä ratkaisua tarvitse relaksoida. Navierin ja Stokesin yhtälössä esitellyille materiaaliparametreille (viskositeetti, lämpölaajenemiskerroin ja sen referenssilämpötila) käytämme arvoja $\mu = 10^{-3}$ kg/m^3, $\beta = 10^{-5}$ 1/K ja $T_0 = 0$ K. Näillä tehtävä ratkeaa kuudella algoritmin 8.13.1 iteraatiolla. Kuva 8.18 esittää nopeuskenttää.

8.14 Stabiloitu elementtimenetelmä

Galerkinin menetelmä käyttäytyy tunnetusti huonosti ratkaistaessa esimerkiksi advektio-diffuusioyhtälöä

$$\frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T - \nabla \cdot (\alpha \nabla T) = g, \tag{8.25}$$



Kuva 8.18: *Nopeuskenttä*

kun suhde $\|u\|L/\alpha$, missä L on tehtävän pituusskaala, on riittävän suuri. Intuitiivisesti näemme, että tässä tilanteessa kuljetustermin dominoidessa informaatio tulevasta käyttäytymisestä kulkee oleellisesti kokonaan nopeusvektorin suunnassa, eli tieto tulee ylävirralta. Tätä havaintoa ei kuitenkaan millään tavalla käytetä hyväksi ratkaistaessa yhtälöä Galerkinin menetelmällä. Yksinkertainen ja yleisesti käytetty stabilointimenetelmä yhtälölle (8.25) elementtimenetelmän yhteydessä onkin *Streamline Upwinding Petrov-Galerkin -menetelmä* (SUPG). Englanninkielisen nimensä mukaisesti tämä on virtaviivojen suuntaisesti ylävirtaa painottava menetelmä. Menetelmässä testifunktio Ψ korvataan uudella testifunktiolla

$$\Psi_{\text{SUPG}} = \Psi + \tau \mathbf{u} \cdot \nabla \Psi,$$

missä parametri τ on niin kutsuttu stabilointiparametri, jonka arvon valintaan palaamme myöhemmin. Menetelmää kutsutaan Petrovin ja Galerkinin menetelmäksi erotukseksi tavanomaisesta Galerkinin menetelmästä, koska käytämme testifunktiona eri funktioita kuin yritteelle. Uusi testifunktio on epäjatkuva ja lisää yhtälöön ylävirtaa eli informaation tulosuuntaa painottavan termin. Tällä tavallaan ennakoidaan tulevaa kehitystä ja stabiloidaan ratkaistavaa yhtälöä.

Seuraava esimerkki valottaa advektio-diffuusioyhtälön diskretoinnin ongel-

mia ja näiden ratkaisua. Kuvaamme tässä esimerkissä myös tämän yhtälön ratkaisun elementtimenetelmällä yksityiskohtaisesti.

Esimerkki 8.14.1 Tarkastelemme yksiulotteista ajasta riippumatonta advektio-diffuusioyhtälöä välillä $[0,1]$. Olkoot yhtälön parametrit asetettu siten, että $\alpha = 1, u = 50, g = 0$ sekä reunaehdot $T(0) = 0$ ja $T(1) = 1$. Tällöin yhtälö (8.25) yksinkertaistuu muotoon

$$u\frac{dT}{dx} - \alpha\frac{d^2T}{dx^2} = 0.$$

Kertomalla yhtälö testifunktiolla, integroimalla välin $[0,1]$ yli ja osittaisintegroimalla saamme

$$\int_0^1 \alpha\frac{dT}{dx}\frac{d\Psi}{dx}\,dx - \left[\alpha\frac{dT}{dx}\Psi\right]_0^1 + \int_0^1 u\frac{dT}{dx}\Psi\,dx = 0.$$

Sijoittamalla yrite

$$T \approx \sum_j T_j\Psi_j$$

edelliseen yhtälöön saamme edelleen

$$\sum_{j=1}^{N}\int_0^1 \left(\alpha\frac{d\Psi_j}{dx}\frac{d\Psi_i}{dx} + u\frac{d\Psi_j}{dx}\Psi_i\right)dx\; T_j = F_i,\ i = 1,\ldots,N. \tag{8.26}$$

Yrite on valittu siten, että annetut reunaehdot täyttyvät, jolloin oikean puolen vektorin alkiot saavat nollasta poikkeavia arvoja. Yhtälö on jälleen muotoa

$$A\mathbf{T} = \mathbf{F},$$

missä matriisin A alkiot ovat

$$A_{ij} = \int_0^1 \left(\alpha\frac{d\Psi_j}{dx}\frac{d\Psi_i}{dx} + u\frac{d\Psi_j}{dx}\Psi_i\right)dx.$$

Analysoidaksemme tilannetta jaamme laskenta-alueen aluksi kolmeen samankokoiseen elementtiin ja käytämme yritteenä murtoviivaa:

$$T \approx (1-\xi_k)T_k + \xi_k T_{k+1},\ \xi_k = \frac{1}{h}(x - x_k),\ x_k \le x \le x_{k+1},\ k = 1,2,3,$$

missä $h = (x_{k+1} - x_k) =$ vakio. Koska jako oli tasavälinen, koordinaatti ξ_k on oleellisesti sama kaikille k. Lieneekin selkeämpää jättää alaindeksi kirjoittamatta ja merkitä $\xi_k = \xi$. Solmuun k liittyvä kantafunktio on siten murtoviiva eli "hattufunktio"

$$\Psi_k(x) = \begin{cases} \xi, & x_{k-1} \le x \le x_k \\ 1-\xi, & x_k \le x \le x_{k+1}. \end{cases}$$

Kantafunktioiden derivaatat ovat

$$\frac{d\Psi_k}{dx} = \frac{d\Psi_k}{d\xi}\frac{d\xi}{dx} = \begin{cases} 1/h \\ -1/h \end{cases}.$$

Lisäksi differentiaali on $dx = h\,d\xi$, joten voimme kirjoittaa integraalit seuraavaan tapaan:

$$\int_{x_{k-1}}^{x_{k+1}} \Psi_k(x)\,dx = \int_0^1 \xi h\,d\xi + \int_0^1 (1-\xi)h\,d\xi.$$

Käyttämällä näitä tietoja yhtälössä (8.26) (integraalit voi laskea myös *elementeittäin* ja summata vastinalkioiden elementtikohtaiset yhtälöt yhteen) saamme yhtälöryhmäksi:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ -\alpha - \frac{1}{2}uh & 2\alpha & -\alpha + \frac{1}{2}uh & 0 \\ 0 & -\alpha - \frac{1}{2}uh & 2\alpha & -\alpha + \frac{1}{2}uh \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

Huomaa yhteys keskeisdifferenssikaavoihin. Reunaehdot $T(0) = T_1 = 0$ ja $T(1) = T_4 = 1$ on otettu huomioon asettamalla nollaksi kerroinmatriisista vastaavat rivit lävistäjäalkiota lukuunottamatta ja sijoittamalla oikean puolen vektoriin vastaavat arvot. Tämän toimenpiteen voi myös ajatella muokkaavan kantafunktioita siten, että reunaehdot täyttyvät. Voimme nyt eliminoida muuttujat T_1 ja T_4 yhtälöstä, joten yhtälöt yksinkertaistuvat muotoon

$$\begin{pmatrix} 2\alpha & -\alpha + \frac{1}{2}uh \\ -\alpha - \frac{1}{2}uh & 2\alpha \end{pmatrix} \begin{pmatrix} T_2 \\ T_3 \end{pmatrix} = \begin{pmatrix} 0 \\ \alpha - \frac{1}{2}uh \end{pmatrix}.$$

Tästä saamme esimerkkimme parametreilla ratkaisuksi $T_2 \approx 0.74$ ja $T_3 \approx -0.2$. Ratkaisuvektori kokonaisuudessaan on

$$\mathbf{T} \approx \begin{pmatrix} 0 & 0.74 & -0.2 & 1 \end{pmatrix}^T.$$

Lopputulos oskilloi voimakkaasti ja on selvästikin väärin. Kerroinmatriisin A ominaisarvot ovat

$$\lambda_{1,2} = \frac{1}{2}\left(4\alpha \pm \sqrt{4\alpha^2 - u^2h^2}\right).$$

Jotta saisimme oskilloimattoman ratkaisun, täytyisi olla $h < 2\alpha/|u|$. Esimerkissämme määritellyillä parametreilla tarvitsisimme tähän tavoitteeseen päästäksemme siis vähintään 25 elementtiä.

Jos käytämme testifunktioina kantafunktioiden Ψ_i sijaan funktioita

$$\Psi_{SUPG,i} = \Psi_i + \tau u \frac{d\Psi_i}{dx} = \begin{cases} \xi + \tau u/h \\ 1 - \xi - \tau u/h \end{cases},$$

saamme yhtälöryhmäksi samaan tapaan kuin edellä:

$$\begin{pmatrix} 2\alpha + 2\tau u^2 & -\alpha - \tau u^2 + \frac{1}{2}uh \\ -\alpha - \tau u^2 - \frac{1}{2}uh & 2\alpha + 2\tau u^2 \end{pmatrix} \begin{pmatrix} T_2 \\ T_3 \end{pmatrix} = \begin{pmatrix} 0 \\ \alpha + \tau u^2 - \frac{1}{2}uh \end{pmatrix}.$$

Tämän tehtävän ratkaisu on sama kuin, jos olisimme käyttäneet alkuperäisessä yhtälössä diffuusiokertoimelle α arvoa $\alpha + \tau u^2$. Näin SUPG-stabilointi vastaa keinotekoisen diffuusion lisäämistä alkuperäiseen yhtälöön. Moniulotteisessa tehtävässä diffuusiota lisätään vain vektorin $\mathbf{u}$ suunnassa.

Jotta yhtälö olisi siinä mielessä konsistentti, että hilakoon pienentyessä (eli kun $h \to 0$) alkuperäinen yhtälö palautuisi, täytyy parametrin τ riippua hilan koosta. Vastaavasti sen täytyy riippua myös kantafunktioiden asteluvusta m.

Lisäksi se selvästi riippuu suureista u ja α, joten $\tau = \tau(h, m, u, \alpha)$. Yksiulotteiselle tehtävälle ja lineaarisille elementeille voidaankin osoittaa, että parametrin τ optimaalinen arvo τ_{opt} on

$$\tau_{\text{opt}} = \frac{h}{2|u|}\left(\coth(\text{Pe}(x)) - \frac{1}{\text{Pe}(x)}\right), \ \text{Pe}(x) = \frac{|u(x)|h}{2\alpha}.$$

Tällä parametrin τ valinnalla ratkaisu saa tarkan arvon elementtien solmupisteissä. Useampiulotteisille tehtäville parametrin optimaalista arvoa ei osata esittää analyyttisessä muodossa. Stabilointiparametrin arvon määräämistä käsitellään edempänä hiukan lisää.

Tarkastellaan vielä raja-arvoa $\alpha \to 0$, jolloin tehtävästämme tulee pelkkä advektioyhtälö

$$u\frac{dT}{dx} = g.$$

Suoraan Galerkinin menetelmällä diskretoituna saamme kerroinmatriisiksi (vakiokerroin on jätetty kirjoittamatta)

$$A = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix},$$

jonka ominaisarvot ovat $\lambda_{1,2} = \pm i$. *Menetelmä onkin aina epästabiili hilakoosta riippumatta.* Stabiloituna yhtälön kerroinmatriisi on

$$A = \begin{pmatrix} 2\tau u^2 & -\tau u^2 + \frac{1}{2}uh \\ -\tau u^2 - \frac{1}{2}uh & 2\tau u^2 \end{pmatrix},$$

joka on myös keskeisdifferenssiapproksimaatiota käyttävä diskretointi tehtävälle

$$-\tau u^2 \frac{d^2T}{dx^2} + u\frac{dT}{dx} = g.$$

Stabiloidun advektioyhtälön kerroinmatriisi on aina stabiili riippumatta hilakoosta h, jos valitsemme $\tau = h/2|u|$. Tämän saa myös raja-arvona $\alpha \to 0$ yllä olevasta optimaalisen parametrin τ_{opt} lausekkeesta. Tällöin kerroinmatriisiksi tulee (kun $u > 0$)

$$A = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix},$$

mikä vastaa advektioyhtälön diskretointia toispuoleisilla differensseillä

$$\frac{dT}{dx} \approx \frac{u}{h}\left(T_i - T_{i-1}\right).$$

Differenssi lasketaan SUPG-menetelmässä paikallisesti ylävirtaan, eli virtauksen tulosuuntaan. Toisin sanoen, jos tässä olisi $u(x) < 0$, differenssi tulisi laskea toisin päin

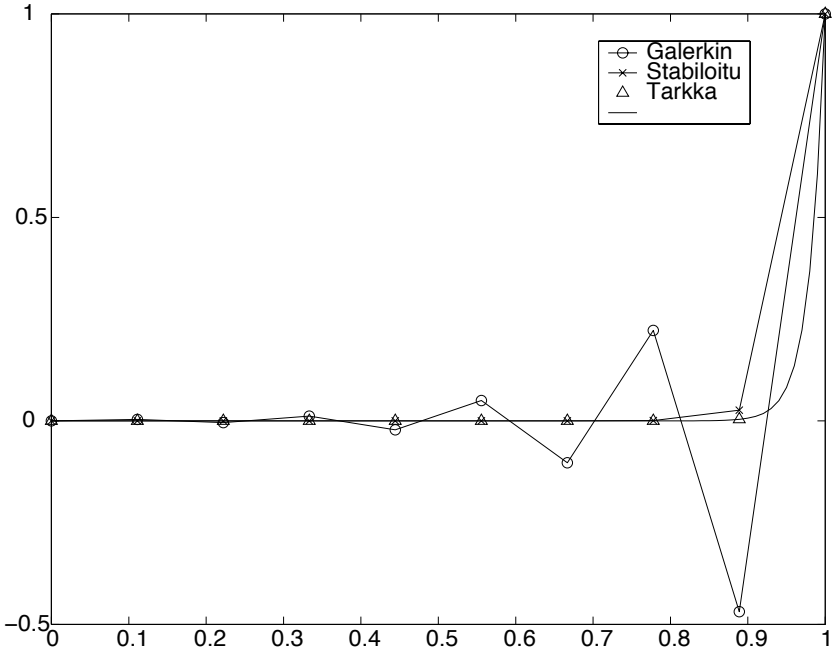
$$\frac{dT}{dx} \approx \frac{u}{h}\left(T_{i+1} - T_i\right).$$

Yhdistettynä nämä lausekkeet ovat differenssimenetelmässä hyvin tunnettu *upwinding*-differenssikaava. Huomaa kuitenkin, että näiden differenssiapproksimaatioiden tarkkuus on vain luokkaa $\mathcal{O}(h)$.

Kuvassa 8.19 on esitetty yksiulotteisen advektio-diffuusioyhtälön ratkaisu välillä $[0, 1]$ tässä esimerkissä käytetyillä parametreilla. Ratkaisimme tehtävän sekä Galerkinin menetelmällä että stabiloidulla elementtimenetelmällä. Kuvassa näkyvään ratkaisuun on käytetty kymmentä elementtiä. Tehtävän analyyttinen ratkaisu on

$$T(x) = \frac{e^{ux/\alpha} - 1}{e^{u/\alpha} - 1}.$$

Kuvassa 8.19 esitettyyn stabiloituun ratkaisuun ei ole käytetty optimaalista parametrin τ arvoa, joten ratkaisu ei myöskään ole solmupisteittäin tarkka (vaikkakin suhteellisen lähellä). Jos parametri tulee valituksi liian pieneksi, saamme oskilloivan ratkaisun. Jos parametri taas on liian suuri, saamme liian diffuusin ratkaisun. Parametria τ kutsutaankin usein syystäkin stabiloidun elementtimenetelmän viritysparametriksi.



Kuva 8.19: *Advektio-diffuusioyhtälön ratkaisu Galerkinin menetelmällä ja stabiloidulla elementtimenetelmällä.*

Advektio-diffuusioyhtälön ratkaisun ongelma peruselementtimenetelmällä on liian pieni approksimaatioavaruus. Tämä saattaa aiheuttaa ratkaisun suurtakin numeerista oskillointia (kuva 8.19). Kuten edellä olevasta esimerkistä käy ilmi, elementtiverkon tihentäminen lokaalisti ongelmakohdissa saattaa parantaa tilannetta lisäämällä ratkaisun vapausasteita. Usein tihennys on kuitenkin hankalaa toteuttaa tai se kasvattaa ratkaistavaa yhtälöryhmää niin, ettei näin voida toimia käytännössä. Advektioyhtälön tilannehan lisäksi (katso ylläolevaa esimerkkiä) oli se, että verkon tihentäminen ei parantanut tilannetta millään tavalla. Jos advektio dominoi yhtälön käyttäytymistä voimakkaasti, verkkoa jouduttaisiin tihentämään hyvin radikaalisti. Stabiloi-

dun menetelmän idea on laajentaa kantafunktioavaruutta siten, että yhtälöt säilyvät konsistentteina, ilman että vapausasteiden määrä kasvaa.

Kirjoitamme yksinkertaisuuden vuoksi Galerkinin menetelmän muodossa

$$B(u,v) = \langle \mathcal{L}(u) - f, v\rangle = \langle R(u), v\rangle = 0,$$

missä $\mathcal{L}$ on ratkaistavan yhtälön differentiaalioperaattori ja $\langle\cdot,\cdot\rangle$ määrittelyavaruuden sisätulo. Stabiloidussa elementtimenetelmässä kantafunktioavaruutta laajennetaan lisäämällä bilineaarimuotoon B *elementtikohtainen jäännöksestä riippuva termi*

$$B(u,v) = \langle R(u), v\rangle + \sum_K \langle R(u), \tau W(v)\rangle,$$

missä W on painofunktio ja τ siis elementin koosta, kantafunktioapproksimaatiosta sekä yhtälön parametreista riippuva tekijä. Summa käy yli laskenta-alueen elementtijaon. Painofunktio voidaan valita monella tavalla. Yllä esiteltyyn SUPG-menetelmään päästään valitsemalla

$$W = \mathbf{u}\cdot\nabla v.$$

Ehkä suosituin valinta kuitenkin on $W = \mathcal{L}(v)$ eli advektio-diffuusioyhtälön tapauksessa,

$$W = \mathbf{u}\cdot\nabla v - \nabla\cdot(\alpha\nabla v),$$

mistä saadaan *Galerkinin pienimmän neliösumman menetelmä* (Galerkin Least-Squares Method, GLS). Vielä eräs vaihtoehto on valita $W = -\mathcal{L}^*(v)$, eli

$$W = \mathbf{u}\cdot\nabla v + \nabla\cdot(\alpha\nabla v),$$

jonka ainoa ero GLS-menetelmän painofunktioon on se, että ”diffuusiotermin” etumerkki on vaihdettu. Tästä valinnasta kehitettyä menetelmää kutsutaan Douglasin ja Wangin menetelmäksi. Lineaarisille kantafunktioille ja paikasta riippumattomalle diffuusiokertoimelle α elementtikohtaisesti laskettuna termi $\nabla\cdot(\alpha\nabla v)$ häviää ja menetelmät redusoituvat SUPG-menetelmään.

Parametrin τ valinta on vaikeampi asia, ja sitä tutkitaankin aktiivisesti. Eräs mahdollinen valinta parametrille τ advektio-diffuusioyhtälön yhteydessä on seuraava:

$$\begin{aligned}
\tau &= \frac{h_K}{2\|\mathbf{u}\|}\min(1, \mathrm{Pe}_K(x)),\\
\mathrm{Pe}_K(x) &= \frac{m_K\|\mathbf{u}\|h_K}{2\alpha},\\
m_K &= \min(1/3, 2C_K),\\
C_K &\sum_K h_K^2\|\nabla^2 v\|_K^2 \le \|\nabla v\|^2.
\end{aligned}$$

missä h_K on elementin karakteristinen koko (elementin poikkileikkauksen pituus vektorin $\mathbf{u}$ suunnassa) ja m_K elementin tyypistä riippuva vakio. Lineaarisille kantafunktioille sekä bilineaarisille nelikulmioille parametrille m_K on

ehdotettu arvoa $1/3$, parabolisille kolmioille arvoa $1/24$, parabolisille nelikulmioille arvoa $1/12$ ja trilineaariselle laatikkoelementille arvoa $1/6$.

Stabiloitu elementtimenetelmäperhe, jossa ei tarvita ylimääräisiä parametreja, (tällaiset parametrit ovat yhtälökohtaisia ja vähän epämääräisiä), juontuu kuplafunktioista (bubble functions). Kuplat ovat *elementeittäin määriteltyjä* kantafunktioiden laajennuksia. Menetelmä johtaa samantyyppisiin lisätermeihin kuin yllä esitetyt menetelmätkin. Lupaavin kehityssuunta lienee *jäännöksettömät kuplat* (Residual Free Bubbles, RFB). Menetelmä on seuraava:

$$\langle \mathcal{L}(u + u_b) - f, v \rangle = 0,$$

missä u_b täyttää elementtikohtaisesti yhtälön

$$\mathcal{L}(u_b) = -\mathcal{L}(u) + f,$$

kun reunaehtona on $u_b = 0$. Etenemme ratkaisemalla funktiot Φ_i ja Φ_f yhtälöistä

$$\begin{cases} \mathcal{L}(\Phi_i) &= -\mathcal{L}(N_i) \\ \mathcal{L}(\Phi_f) &= f \end{cases}$$

reunaehdoilla $\Phi_i = 0$ ja $\Phi_f = 0$. Funktiot N_i ovat alkuperäiset elementin kantafunktiot, jotka määritellään kappaleessa 8.7. Kuplafunktio u_b voidaan nyt lausua muodossa

$$u_b = \sum_i u_i \Phi_i + \Phi_f.$$

Kun funktioiden Φ_i ja Φ_f muoto tunnetaan, voidaan suure u_b siis eliminoida alkuperäisestä yhtälöstä. Ongelma näiden menetelmien soveltamisessa ainakin vielä on se, että ne ovat laskennallisesti suhteellisen raskaita. Kuten yllä käy ilmi, jäännösvapaan kuplafunktion muoto saadaan ratkaisemalla useita alkuperäisen tapaisia differentiaaliyhtälöitä jokaista mallin elementtiä kohti. Yleisesti siis joutuisimme käyttämään jotain muuta menetelmää ratkaistaksemme kuplien muodon, esimerkiksi jotain aikaisemmin esitetyistä menetelmistä. Tämän jälkeen voimme ratkaista alkuperäisen tehtävän kuplafunktioiden avulla.

8.15 Yhteenveto

Elementtimenetelmä (FEM) on osittaisdifferentiaali-, integrodifferentiaali- ja integraaliyhtälöiden yleiskäyttöinen ratkaisumenetelmä. Tässä oppaassa rajoituimme osittaisdifferentiaaliyhtälöiden ratkaisuun.

Elementtimenetelmässä ratkaistavaa suuretta arvioidaan yksinkertaisella funktiolla. Useimmiten tämä tarkoittaa interpolointia ratkaisun solmupistearvosta matala-asteisilla paloittain määritellyillä polynomeilla. Lisäksi ratkaisualue jaetaan pieniin osa-alueisiin eli elementteihin. Elementit määritellään yleensä yksinkertaisessa lokaalissa koordinaatistossa, josta ne kuvataan todellisiksi käytetyiksi elementiksi. FEM soveltuu muihin menetelmiin

verrattuna erityisen hyvin monimutkaisiin geometrioihin, joissa voidaan tarvita laskentahilan tihennyksiä.

Käytännössä tärkeimpiä elementtimenetelmän lajeja ovat tässä luvussa käsitellyt menetelmät:

- Galerkinin menetelmä, joka soveltuu hyvin esimerkiksi lämpö- tai diffuusioyhtälön ratkaisemiseen.
- Stabiloidut elementtimenetelmät, jotka soveltuvat muun muassa advektioyhtälön sekä Navierin ja Stokesin yhtälöiden ratkaisemiseen.

Tehtävän määrittelevät yhtälöt, reunaehdot, materiaalirelaatiot ja geometria johtavat harvan yhtälöryhmän ratkaisemiseen. Jos tehtävä on epälineaarinen, se joudutaan linearisoimaan ja ratkaisemaan iteroiden. Ajasta riippuvassa tehtävässä epälineaarinen iteraatio suoritetaan aika-askeleen sisällä.

Yhtälöryhmää vastaava matriisi kannattaa koota elementtikohtaisesti ja summata elementtikohtaiset osuudet globaaliin matriisiin. Matriisielementit muodostavat integraalit lasketaan yleensä sopivan asteisesta Gaussin kvadratuuria käyttäen. Lineaarisen yhtälöryhmän ratkaisuun voi käyttää suoria tai iteratiivisia menetelmiä. Yhtälöryhmän talletukseen ja ratkaisuun kannattaa käyttää harvoille matriiseille tai nauhamatriiseille sopivia tehokkaita menetelmiä sekä iteratiivisten menetelmien tapauksessa pohjustusta.

Ajasta riippuvan tehtävän aikaintegrointiin olemme käyttäneet tavallisten differentiaaliyhtälöiden alkuarvotehtävien käsittelystä tuttuja menetelmiä keskittyen yleiseen Eulerin menetelmään.

Kytketyt osittaisdifferentiaaliyhtälöt voi ratkaista joko kokoamalla yhden linearisoidun yhtälöryhmän, jonka tuntemattomina ovat kaikkien muuttujien solmupistearvot, tai iteroimalla eri ODY:istä muodostettuja pienempiä yhtälöryhmiä.

8.16 FEM-pohjaisia ohjelmistoja

Olemme ratkaisseet tämän luvun esimerkit ELMER-ohjelmistolla. Tätä yleiskäyttöistä FEM-ohjelmistoa ovat kehittäneet suomalaiset yliopistot ja yritykset yhdessä CSC:n kanssa. ELMER-paketti sisältää esikäsittelijän, ratkaisijan ja jälkikäsittelijän. ELMERin ratkaisijassa on valmiina muun muassa kokoonpuristumaton ja -puristuva virtaus, lämpöyhtälö käsittäen myös säteilytehtävät, advektio-diffuusio-yhtälö, rakenneanalyysin tehtävät ja magnetohydrodynamiikka. Näitä on mahdollista kytkeä toisiinsa. Käyttäjä voi myös itse lisätä ratkaisijaan elementtimenetelmällä diskretoituja osittaisdifferentiaaliyhtälöitä. ELMER toimii useissa ympäristöissä (Unix, Linux, Windows). Lisätietoja ELMER-ohjelmistosta löytyy www-sivuilta `http://www.csc.fi/elmer/`.

CSC tarjoaa myös sovellusalakohtaisia ja yleisiä FEM-pohjaisia ohjelmistoja. Nämä tarjoavat tehtävään soveltuessaan nopean vaihtoehdon mallintamiseen. Tyypillisesti malli luodaan esikäsittelijän graafisella käyttöliittymällä,

ratkaisija ajetaan erätyönä. Jälkikäsittely suoritetaan jälleen interaktiivisesti.

Myös valmisohjelmistoja käytettäessä on syytä tuntea elementtimenetelmän ja näihin liittyvän numeriikan perusteet sekä virheiden välttämiseksi että tehokkaan ratkaisumenetelmän valitsemiseksi. Esimerkiksi seuraavat FEM-pohjaiset ohjelmistot soveltuvat ajasta riippuvien ja riippumattomien tehtävien ratkaisuun, osa myös ominaisarvotehtäviin (katso lukua 9):

- ABAQUS on monipuolinen lujuus- ja lämpölaskentaohjelma, joka soveltuu myös virtaustehtävien ratkaisuun huokoisesssa materiaalissa. ABAQUS-ohjelmistolla voi ratkaista myös lujuusopin ominaisvärähtelytehtäviä.
- ANSYS soveltuu rakenneanalyysin, lämmönsiirron ja sähkömagnetiikan tehtäviin sekä yksinkertaisiin virtauslaskentatehtäviin. Näitä tehtäviä voidaan ratkaista myös kytkettyinä monifysikaalisina ongelmina. Myös ominaisarvotehtäviä voidaan ratkoa.
- FIDAP on moderni ohjelmisto virtausmekaniikkaan ja lämmönsiirtotehtäviin. Myös kontrollitilavuusmenetelmään perustuvilla virtauslaskentaohjelmistoilla (CFX ja Fluent), voidaan ratkoa samanlaisia tehtäviä kuin FIDAPilla. Fluentin esikäsittelijä GAMBIT tuottaa myös FIDAPille soveltuvia verkkoja.
- FEMLAB on Matlabiin perustuva graafisella käyttöliittymällä varustettu ODY-ratkaisija. FEMLAB sisältää mm. lujuuslaskennan, sähkömagnetiikan ja lämmönsiirron tehtävien ratkaisijat ja soveltuu hyvin myös monifysikaalisten tehtävien käsittelyyn. Käyttäjän on mahdollista määritellä yleisiä ODY-ryhmiä ratkaistavaksi. Matlabin ohjelmointikieli ja funktiot ovat käytettävissä FEMLABista.
- PDE2D on yleinen kaksi- ja kolmiulotteisten tehtävien ODY-ratkaisija, joka soveltuu myös joihinkin ominaisarvotehtäviin. Kolmessa ulottuvuudessa PDE2D voi ratkoa kuitenkin vain suhteellisen yksinkertaisen muotoisissa alueissa määriteltyjä tehtäviä. Käyttäjä määrittelee tekstipohjaisessa käyttöliittymässä funktiomuotoiset kertoimet yhtälön termeille sekä reunaehdot ja geometrian. Tämän jälkeen PDE2D muodostaa Fortran-ohjelman, jolla tehtävä ratkaistaan.

8.17 Lisätietoja

Erinomaisia johdatuksia elementtimenetelmään ovat teokset *An Analysis of the Finite Element Method* [SF73] ja *Finite Element Analysis* [Bur87]. Muita hyviä FEM-kirjoja ovat mm. *Finite Elements for Solids, Fluids, and Optimization* [Moh92], *Finite Element Handbook* [Kar87] ja *Finite Element Method Basics* [SW98]. Suomenkielistä kirjallisuutta edustaa CSC:n opas *Elementtimenetelmä virtauslaskennassa* [HJ94].

9 Ominaisarvotehtävät

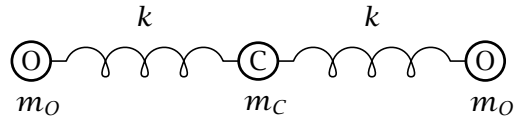
Monilla tieteenaloilla on tärkeitä ongelmia, jotka voidaan muotoilla ominaisarvotehtäviksi. Seuraavassa listassa on muutamia esimerkkejä:

- Monimutkaisen rakenteen, esimerkiksi laivan tai lentokoneen, ominaisvärähtelyjen ja niiden taajuuksien määrittäminen.
- Radiovastaanottimien ja -lähettimien halutaan toimivan resonanssitaajuuksilla; sähköverkoissa resonansseja yritetään välttää.
- Kvanttifysiikassa ja kvanttikemiassa systeemejä kuvataan Schrödingerin yhtälöllä, jonka ominaisarvot ovat eri tilojen energiat.
- Kemiallisten reaktioiden simuloinnissa halutaan määrittää olosuhteet, joissa reaktiot tapahtuvat jaksollisesti.
- Makrotalousmalleissa niin sanottujen tasapainotettujen hinnoittelustrategioiden ja tuotantorakenteiden määrittäminen johtaa ominaisarvotehtäviin.
- Markovin ketjuilla kuvattavien systeemien, kuten jonotus- ja tietokonesysteemien, aikakäyttäytymisen tutkiminen.

Tässä luvussa esittelemme tärkeimmät ominaisarvotehtäviin liittyvät käsitteet ja kuvailemme tavallisimpia ominaisarvotehtävien ratkaisemiseen tarkoitettuja numeerisia menetelmiä.

9.1 Esimerkki

Esimerkki 9.1.1 Tarkastelemme yksinkertaista hiilidioksidimolekyylin (CO_2) mallia. Kuvaamme molekyyliä kolmella massalla, jotka on kytketty kahdella jousella toisiinsa. Hiiliatomi (massa m_C) on kahden happiatomin (massa m_O) välissä kuvan 9.1 mukaisesti. Olkoon k atomien välistä vuorovaikutusta vastaava jousivakio. Newtonin toisen lain perusteella saamme systeemin liikeyh-



Kuva 9.1: *Hiilidioksidimolekyylin yksinkertainen malli.*

tälöiksi

$$x_1'' = -\frac{k}{m_O}(x_1 - x_2), \tag{9.1}$$

$$x_2'' = -\frac{k}{m_C}(x_2 - x_1) - \frac{k}{m_C}(x_2 - x_3), \tag{9.2}$$

$$x_3'' = -\frac{k}{m_O}(x_3 - x_2), \tag{9.3}$$

missä x_1, x_2 ja x_3 ovat atomien poikkeamat tasapainoasemistaan.

Nyt haluamme löytää liikeyhtälöille (9.1)–(9.3) ratkaisun, jossa kaikki atomit värähtelevät *samalla taajuudella*. Tehdään yrite

$$x_j = x_{j0}\cos\omega t, \quad j = 1, 2, 3, \tag{9.4}$$

missä ω on yhteinen taajuus ja atomien alkupaikat ovat x_{j0} (kaikkien atomien alkunopeus on 0). Sijoittamalla yritteen (9.4) yhtälöihin (9.1)–(9.3) ja supistamalla yhteisen tekijän saamme yhtälöryhmän

$$\begin{pmatrix} \frac{k}{m_O} & -\frac{k}{m_O} & 0 \\ -\frac{k}{m_C} & \frac{2k}{m_C} & -\frac{k}{m_C} \\ 0 & -\frac{k}{m_O} & \frac{k}{m_O} \end{pmatrix} \begin{pmatrix} x_{10} \\ x_{20} \\ x_{30} \end{pmatrix} = \omega^2 \begin{pmatrix} x_{10} \\ x_{20} \\ x_{30} \end{pmatrix}. \tag{9.5}$$

Haluamme siis löytää ne taajuudet ω, joilla yhtälön (9.5) ratkaisu $(x_{10}\ x_{20}\ x_{30})^T$ poikkeaa nollasta. Tämä on *ominaisarvotehtävä*. Etsimme yhtälön (9.5) vasemmalla puolella esiintyvän matriisin *ominaisarvoja* ω^2 ja *ominaisvektoreita* $(x_{10}\ x_{20}\ x_{30})^T$.

9.2 Määritelmiä ja käsitteitä

Kompleksiluku λ on $n \times n$-neliömatriisin A *ominaisarvo*, jos on olemassa vektori $\mathbf{x} \neq 0$, jolle pätee

$$A\mathbf{x} = \lambda\mathbf{x}. \tag{9.6}$$

Ehdon toteuttava vektori $\mathbf{x}$ on matriisin A *ominaisvektori*. Matriisin ominaisarvojen muodostamaa joukkoa kutsutaan matriisin *spektriksi*.

Ominaisarvot ovat ne parametrin λ arvot, joilla yhtälöllä $(A - \lambda I)\mathbf{x} = \mathbf{0}$ on nollasta poikkeava ratkaisu $\mathbf{x}$. Niinpä ominaisarvot toteuttavat yhtälön

$\det(A - \lambda I) = 0$. Determinantti voidaan kirjoittaa auki, jolloin saadaan matriisin *karakteristiseksi polynomiksi* kutsuttu n-asteinen polynomi $p_n(\lambda)$. Ominaisarvot ovat siis karakteristisen polynomin nollakohdat. Kompleksiset ominaisarvot ovat pareittain toistensa kompleksikonjugaatteja.

Ominaisarvon *algebrallinen kertaluku* kertoo, kuinka moninkertainen karakteristisen polynomin juuri ominaisarvo on. *Geometrinen kertaluku* on ominaisarvoon liittyvien lineaarisesti riippumattomien ominaisvektoreiden lukumäärä. Jos algebrallinen kertaluku ylittää geometrisen kertaluvun, ominaisarvoa sanotaan *defektiiviseksi*.

Itseisarvoltaan suurinta ominaisarvoa kutsutaan *dominoivaksi ominaisarvoksi* ja vastaavaa ominaisvektoria *dominoivaksi ominaisvektoriksi*. Käytämme näitä termejä myös monikossa puhuessamme itseisarvoltaan suurimmista ominaisarvoista ja vastaavista ominaisvektoreista.

Esimerkki 9.2.1 Jatkamme esimerkin 9.1.1 tarkastelua ja haemme ominaisarvot ω^2 karakteristisen polynomin avulla. Determinanttiyhtälöstä

$$\begin{vmatrix} \frac{k}{m_O} - \omega^2 & -\frac{k}{m_O} & 0 \\ -\frac{k}{m_C} & \frac{2k}{m_C} - \omega^2 & -\frac{k}{m_C} \\ 0 & -\frac{k}{m_O} & \frac{k}{m_O} - \omega^2 \end{vmatrix} = 0$$

saamme polynomiyhtälön

$$\omega^2 \left(\frac{k}{m_O} - \omega^2 \right) \left(\omega^2 - \frac{2k}{m_C} - \frac{k}{m_O} \right) = 0.$$

Ominaisarvot ovat siis $\omega_1^2 = 0$, $\omega_2^2 = \frac{2k}{m_C}$ ja $\omega_3^2 = \frac{2k}{m_C} + \frac{k}{m_O}$.

Vastaavat ominaisvektorit löydämme ratkaisemalla yhtälön (9.5) kullekin ominaisarvolle erikseen. Kun $\omega^2 = 0$, ominaisvektori on $\alpha(1 \ 1 \ 1)^T$. Tämä vastaa puhdasta siirtymää ilman värähtelyä. Ominaisarvoa $\omega^2 = 2k/m_C$ vastaava ominaisvektori on $\beta(1 \ 0 \ -1)^T$. Nyt happiatomit värähtelevät vastakkaisiin suuntiin ja keskellä oleva hiiliatomi pysyy paikallaan. Ominaisarvoa $\omega^2 = 2k/m_C + k/m_O$ vastaava ominaisvektori on $\gamma(1 \ -2m_0/m_C \ 1)^T$, eli happiatomit värähtelevät samaan suuntaan ja hiiliatomi liikkuu siten, että systeemin massakeskipiste pysyy paikallaan.

Vaikka ominaisarvot voidaan periaatteessa hakea karakteristisen polynomin nollakohtina, näin ei kannata tehdä, koska polynomin juuret saattavat olla erittäin herkkiä kertoimien suhteen. Suositeltavampaa on käyttää jotain erityisesti ominaisarvotehtäviä varten kehitettyä menetelmää.

Taulukossa 9.1 luetellaan muutamia ominaisarvotehtävien ominaisuuksia. Yleisesti voidaan sanoa, että hermiittiset tehtävät ovat helpompia ratkaista, ja niiden ominaisuudet tunnetaan paremmin.

Perustehtävän (9.6) lisäksi on olemassa myös monimutkaisempia ominaisarvotehtäviä. Esimerkiksi rakenteiden mekaniikassa joudutaan usein tarkastelemaan *yleistettyä ominaisarvotehtävää*, joka on muotoa

$$A\mathbf{z} = \lambda B\mathbf{z},$$

Taulukko 9.1: *Ominaisarvojen λ_i ominaisuuksia eräille matriisityypeille.*

Matriisi A	*Ominaisarvot λ_i*
ylä- tai alakolmiomatriisi	ominaisarvot ovat A:n lävistäjäalkiot
hermiittinen	ominaisarvot ovat reaaliset
positiivisesti definiitti	ominaisarvot ovat positiivisia
positiivisesti semidefiniitti	ominaisarvot ovat ei-negatiivisia

missä A ja B ovat molemmat neliömatriiseja.

Jatkossa keskitymme perustehtävään (9.6), mutta käsittelemme kuitenkin lyhyesti myös yleistettyä ominaisarvotehtävää.

9.2.1 Singulaariarvot

Ominaisarvoon läheisesti liittyvä käsite on singulaariarvo. Singulaariarvot on määritelty myös muille kuin neliömatriiseille. Kokoa $m \times n$ olevan matriisin A *singulaariarvohajotelma* on muotoa

$$A = U\Sigma V^H,$$

missä U on unitaarinen $m \times m$-matriisi, V on unitaarinen $n \times n$-matriisi ja Σ on $m \times n$-matriisi:

$$\Sigma_{ij} = \begin{cases} 0, & \text{kun } i \neq j, \\ \sigma_i \geq 0, & \text{kun } i = j. \end{cases}$$

Alkiot σ_i ovat matriisin A *singulaariarvot* ja matriisien U ja V sarakkeet ovat vastaavat *singulaarivektorit*.

Singulaariarvot ovat matriisin $A^H A$ ominaisarvojen ei-negatiiviset neliöjuuret. Singulaariarvot voidaan tulkita myös geometrisesti. Kun yksikköpallo kuvataan reaalisella matriisilla A, tuloksena saatavan ellipsoidin puoliakselien pituudet ovat matriisin singulaariarvot.

Esimerkki 9.2.2 Matriisin

$$A = \begin{pmatrix} 1 & 2 & 3 \\ -4 & 1 & 2 \end{pmatrix}$$

singulaariarvohajotelma on

$$A \approx \begin{pmatrix} -0.413 & -0.911 \\ -0.911 & 0.413 \end{pmatrix} \begin{pmatrix} 4.777 & 0 & 0 \\ 0 & 3.491 & 0 \end{pmatrix} \begin{pmatrix} 0.676 & -0.734 & 0.060 \\ -0.364 & -0.403 & -0.840 \\ -0.641 & -0.546 & 0.540 \end{pmatrix}^H.$$

Singulaariarvohajotelma on tärkeä työkalu erilaisissa sovellutuksissa. Hajotelmaa voidaan käyttää esimerkiksi laskettaessa matriisin 2-normia ja häiriöalttiutta sekä pienimmän neliösumman ratkaisua lineaarisille yhtälöryhmälle (kappale 4.4.2 sivulla 83).

Muutamia jatkossa esiteltävistä numeerisista menetelmistä voidaan käyttää myös singulaariarvojen laskemiseen. Tulemme kuitenkin käsittelemään vain ominaisarvojen hakemista.

9.2.2 Hajotelmia ja muunnoksia

Monissa ominaisarvojen laskentamenetelmissä käytetään hyväksi sitä, että matriisin ominaisarvot pysyvät muuttumattomina *similariteettimuunnoksessa*

$$A \to Z^{-1}AZ,$$

missä Z on jokin säännöllinen matriisi. Matriiseilla A ja $Z^{-1}AZ$ on siis täsmälleen samat ominaisarvot. Lisäksi vektori $Z^{-1}\mathbf{x}$ on matriisin $Z^{-1}AZ$ ominaisvektori, jos ja vain jos $\mathbf{x}$ on alkuperäisen matriisin A ominaisvektori.

Alkuperäinen matriisi pyritään peräkkäisillä similariteettimuunnoksilla vähitellen saattamaan muotoon, josta ominaisarvot ja -vektorit on helppo määrittää. Esittelemme nyt lyhyesti muutamia jatkossa tarvittavia matriisihajotelmia ja -muunnoksia.

Jos matriisilla ei ole defektiivisiä ominaisarvoja, se voidaan *diagonalisoida:* on olemassa säännöllinen matriisi X siten, että

$$X^{-1}AX = D,$$

missä

$$D = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$$

on diagonaalimatriisi. Luvut $\lambda_1, \ldots, \lambda_n$ ovat matriisin A ominaisarvot, ja matriisin X sarakkeet ovat ominaisvektoreita. Jos A on hermiittinen (symmetrinen), matriisi X voidaan valita niin, että se on unitaarinen (ortogonaalinen). Hajotelmaa XDX^{-1} kutsutaan matriisin *ominaisarvohajotelmaksi*.

Kaikkia matriiseja ei voida diagonalisoida. Jokaiselle matriisille voidaan kuitenkin muodostaa *Schurin hajotelma:* on olemassa unitaarinen matriisi Q siten, että

$$Q^H AQ = T,$$

missä T on yläkolmiomatriisi, jonka lävistäjä muodostuu matriisin A ominaisarvoista. Matriisin Q sarakkeita kutsutaan *Schurin vektoreiksi*. Kokoelma Schurin vektoreita muodostaa vastaavien ominaisvektorien virittämän aliavaruuden ortonormaalin kannan.

Laskentatyön vähentämiseksi on monesti suositeltavaa similaarimuuntaa matriisi yksinkertaisempaan muotoon ennen varsinaista laskentaa. Hermiittiset matriisit voidaan muuntaa tridiagonaalimuotoon (kappale 3.7 sivulla 47) ja ei-hermiittiset Hessenbergin muotoon. *Hessenbergin matriisin* alkiot toteuttavat $a_{ij} = 0$, kun $i - j > 1$.

Tähän päästään sekä Householderin muunnoksilla että Givensin rotaatioilla, joita kutsutaan toisinaan myös Jacobin rotaatioiksi. Nämä ovat numeerisesti stabiileja muunnoksia.

Householderin muunnos tarkoittaa matriisin kertomista Householderin matriisilla

$$P = I - 2\frac{\mathbf{u}\mathbf{u}^H}{\alpha}, \quad \alpha = \|\mathbf{u}\|^2.$$

Kun valitaan $\mathbf{u} = \mathbf{x} - \|\mathbf{x}\|\mathbf{e}_1$, Householderin muunnos $P\mathbf{x}$ nollaa vektorin $\mathbf{x}$ kaikki muut paitsi ensimmäisen komponentin. Sopivalla tavalla valituista Householderin matriiseista saaduilla similariteettimuunnoksilla matriisi voidaan saattaa edellä mainittuun yksinkertaisempaan muotoon:

$$P_{n-2} \cdots P_2 P_1 A P_1 P_2 \cdots P_{n-2} = B.$$

Toinen hyvä tapa on käyttää *Givensin rotaatioita*, joilla muutetaan samanaikaisesti kahta matriisin riviä ja saraketta siten, että jotkin matriisin alkiot saadaan nollattua. Givensin rotaatio on matriisi

$$P_{pq}(\theta) = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & c & \cdots & s & & \\ & & \vdots & 1 & \vdots & & \\ & & -s & \cdots & c & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix},$$

missä c ja s ovat *kiertokulman* θ kosini ja sini. Alkiot c ja s sijaitsevat riveillä ja sarakkeilla p ja q. Käytännössä rotaatiomatriisia ei muodosteta eksplisiittisesti, vaan kertolaskussa $P^H AP$ tarvitaan ainoastaan matriisin A rivejä ja sarakkeita p ja q sekä kertoimia c ja s. Kiertokulmaa θ ei tarvitse laskea eksplisiittisesti. Jos halutaan nollata alkio a_{qr} kierrolla $P_{pq}(\theta)$, kertoimien arvoksi saadaan

$$c = \frac{a_{pr}}{\sqrt{a_{pr}^2 + a_{qr}^2}}, \tag{9.7}$$

$$s = -\frac{a_{qr}}{\sqrt{a_{pr}^2 + a_{qr}^2}}. \tag{9.8}$$

Givensin menetelmässä hermiittisen matriisin tapauksessa nollataan rotaatiolla P_{23} ensin alkiot a_{31} ja a_{13}:

$$A_1 = P_{23}^H A P_{23}.$$

Tämän jälkeen nollataan alkioita yksi kerrallaan. Matriisi A saadaan tridiagonaaliseksi, kun käytetään rotaatiojärjestystä

$$P_{23}, P_{24}, \ldots, P_{2n}; P_{34}, \ldots, P_{3n}; \ldots; P_{n-1,n},$$

missä P_{jk} nollaa alkiot $a_{k,j-1}$ ja $a_{j-1,k}$. Tämä järjestys säilyttää aikaisemmilla kierroksilla syntyneet nolla-alkiot.

9.2.3 Ominaisarvotehtävien herkkyydestä

Jos pienet muutokset matriisin alkioissa aiheuttavat suuria muutoksia johonkin ominaisarvoon, sanotaan että ominaisarvo on herkkä tai häiriöaltis. Myös ominaisvektorit ja niiden virittämät aliavaruudet voivat olla häiriöalttiita. Herkkyys aiheuttaa hankaluuksia ratkaistaessa tehtävää numeerisesti, sillä pyöristysvirheet voivat äärimmäisessä tapauksessa tehdä lopputuloksesta täysin merkityksettömän.

Jos matriisi on normaali eli $AA^H = A^H A$, ominaisarvot eivät ole kovin herkkiä matriisin alkioiden suhteen. Niinpä voi odottaa, että äärellisellä tarkkuudella laskemisesta huolimatta tulokset ovat melko luotettavia. Toisaalta, vaikka matriisi ei olisikaan normaali, kaikki ominaisarvot eivät välttämättä ole häiriöalttiita.

Yksinkertaisen ominaisarvon λ herkkyyttä voidaan arvioida seuraavasti. Olkoot x ja y vastaavat normeeratut oikean ja vasemman puoleiset ominaisvektorit: $Ax = \lambda x$ ja $y^H A = \lambda y^H$. Luvun

$$s(\lambda) = |y^H x|$$

käänteislukua kutsutaan ominaisarvon λ häiriöalttiudeksi. Jos matriisiin A tehdään muutos, jonka suuruus on ϵ, muutos ominaisarvossa λ saattaa olla $\epsilon / s(\lambda)$.

Moninkertaiset tai defektiiviset ominaisarvot saattavat herkkyytensä vuoksi olla erityisen hankalia laskettavia.

Ominaisvektorin herkkyys puolestaan riippuu sekä vastaavan ominaisarvon herkkyydestä että sen etäisyydestä muihin ominainaisarvoihin. Lähellä toisiaan olevia ominaisarvoja vastaavia ominaisvektoreita voi olla vaikeaa laskea tarkasti.

9.3 Ominaisarvotehtävien numeeriset menetelmät

Kaikki ominaisarvojen hakumenetelmät ovat iteratiivisia prosesseja, mikä on luonnollista, koska tehtävä vastaa polynomin nollakohtien määrittämistä. Jotkut menetelmät tuottavat ominaisarvoja ja -vektoreita suoraan, toiset taas määrittävät esimerkiksi tiettyjen ominaisvektoreiden virittämän aliavaruuden kannan, jolloin ominaisarvot ja -vektorit on laskettava vielä erikseen.

Aloitamme numeeristen menetelmien esittelyn pienten tehtävien ratkaisemiseen soveltuvilla yksinkertaisilla algoritmeilla. Näitä edelleen kehittämällä saadaan sekä tehokkaampia menetelmiä pienten tehtävien ratkaisemiseen että menetelmiä, joilla voi ratkaista suuria tehtäviä. Lopuksi käsittelemme lyhyesti yleistettyjä ominaisarvotehtäviä.

Yksinkertaisuuden vuoksi esitämme algoritmeista vain niiden perusversiot. Ohjelmistoissa käytetään muunnelmia, joissa on kiinnitetty huomiota tehokkaaseen laskentaan, mahdollisimman taloudelliseen muistin käyttöön ja

numeeristen virheiden vaikutuksen minimointiin. Tarkoituksemme on, että lukija ymmärtää eri menetelmien peruspiirteet ja toiminnan yksinkertaisissa tapauksissa. Esimerkiksi moninkertaiset ominaisarvot saattavat vaatia muutoksia algoritmeihin.

9.4 Pienet ominaisarvotehtävät

Seuraavaksi esiteltävät menetelmät soveltuvat parhaiten pienille tehtäville. Pienenä voidaan pitää tehtävää, jonka matriisi nolla-alkiot mukaan lukien mahtuu tietokoneen keskusmuistiin. Monesti halutaan määrittää kaikki ominaisarvot ja -vektorit.

Käsittelemme ensin yksinkertaisen potenssimenetelmän, joka on useiden tehokkaampien menetelmien perusta. Sitä parantamalla ja edelleen kehittämällä saadaan aliavaruusmenetelmä ja QR-menetelmä. Lisäksi käymme läpi puolitushaku- ja divide-and-conquer -menetelmät.

Suurten tehtävien ratkaisemiseen tarvitaan kuitenkin muita menetelmiä (kappale 9.5). Potenssimenetelmä ja aliavaruusmenetelmä ovat tehottomia, ja QR-menetelmä vaatii liikaa muistia.

9.4.1 Potenssimenetelmä

Potenssimenetelmä on yksinkertainen iteraatiomenetelmä itseisarvoltaan suurimman ominaisarvon ja sitä vastaavan ominaisvektorin määrittämiseksi. Yksinkertaistaen voidaan sanoa, että mielivaltainen lähtövektori kääntyy matriisilla toistuvasti kerrottaessa itseisarvoltaan suurinta ominaisarvoa vastaavan ominaisvektorin suuntaiseksi.

Potenssimenetelmä on hitaan suppenemisensa vuoksi harvoin käyttökelpoinen. Se on kuitenkin tärkeä, koska monet tehokkaammat menetelmät perustuvat siihen.

Oletetaan, että matriisilla A on ominaisarvot $|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|$ ja lineaarisesti riippumattomat normeeratut ominaisvektorit $\mathbf{x}_1, \ldots, \mathbf{x}_n$. Potenssimenetelmän perusalgoritmi on

Algoritmi 9.4.1 (Potenssimenetelmä)

Valitse aloitusvektori $\mathbf{q}_0$
for $k = 1, 2, 3, \ldots$
$\quad \mathbf{z}_k = A\mathbf{q}_{k-1}$
$\quad \mathbf{q}_k = \frac{\mathbf{z}_k}{\|\mathbf{z}_k\|}$
$\quad \lambda_1^{(k)} = \mathbf{q}_k^H A\mathbf{q}_k = \langle \mathbf{q}_k, A\mathbf{q}_k \rangle$
end

Edellä merkintä $\langle \cdot, \cdot \rangle$ tarkoittaa sisätuloa. Ominaisarvon estimaatti voidaan luonnollisesti laskea myös iteraation suppenemisen jälkeen.

On helppo osoittaa, että $\lambda_1^{(k)} \to \lambda_1$ ja $\mathbf{q}_k \to \mathbf{x}_1$, kun $k \to \infty$. Ominaisvektorien lineaarisen riippumattomuuden takia aloitusvektori voidaan lausua lineaarikombinaationa

$$\mathbf{q}_0 = a_1\mathbf{x}_1 + \ldots + a_n\mathbf{x}_n.$$

Jos $a_1 \neq 0$,

$$A^k\mathbf{q}_0 = a_1\lambda_1^k\left(\mathbf{x}_1 + \sum_{i=2}^{n}\frac{a_i}{a_1}\left(\frac{\lambda_i}{\lambda_1}\right)^k\mathbf{x}_i\right).$$

Koska

$$\left(\frac{\lambda_i}{\lambda_1}\right)^k \to 0$$

kaikilla $i = 2, \ldots, n$, kun $k \to \infty$, vektori $\mathbf{q}_k$ suppenee kohti ominaisarvoa λ_1 vastaavaa ominaisvektoria.

Edellä olevan perusteella havaitaan, että potenssimenetelmän suppenemisnopeus riippuu suhteesta $|\lambda_2|/|\lambda_1|$. Niinpä potenssimenetelmä on käyttökelpoinen vain, jos suhde on pieni. Potenssimenetelmän yleistä käyttäytymistä käsitellään viitteessä [Wil65].

Esimerkki 9.4.1 3×3-matriisin

$$A = \begin{pmatrix} 1 & 0.1 & 0 \\ 0.01 & 1.1 & 0.1 \\ 0 & 0.1 & 1.15 \end{pmatrix}$$

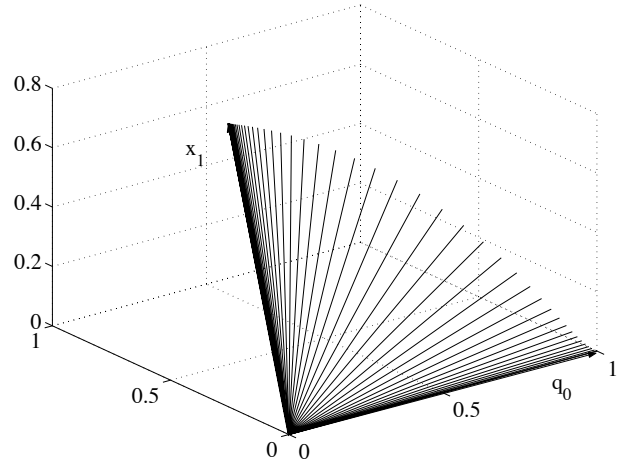
ominaisarvot ovat 1.2297, 1.0376 ja 0.9826. Suurinta ominaisarvoa vastaa ominaisvektori $(0.2619,\ 0.6017,\ 0.7545)^T$.

Haemme suurimman ominaisarvon ja siihen liittyvän ominaisvektorin potenssimenetelmällä. Kuvassa 9.2 näkyy kuinka aloitusvektori $(1, 0, 0)^T$ kääntyy iteraation kuluessa ominaisvektorin suuntaiseksi. Ominaisarvon suppeneminen on esitetty kuvassa 9.3. Tässä tapauksessa $|\lambda_2/\lambda_1| \approx 0.8438$, joten suppeneminen on melko hidasta.

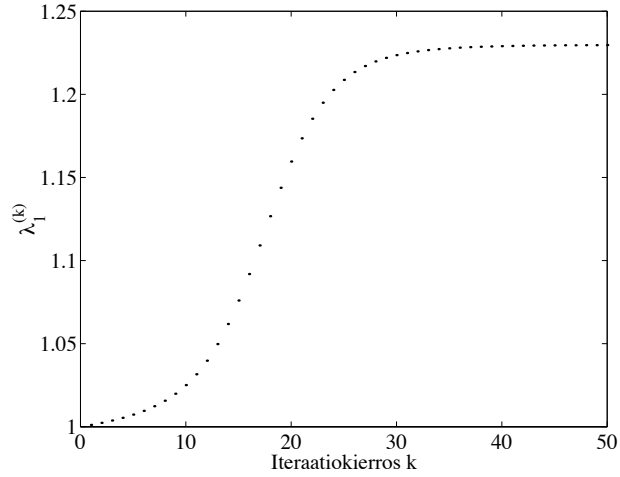
9.4.2 Käänteinen potenssimenetelmä ja siirretty käänteinen potenssimenetelmä

Käänteismatriisin ominaisarvot ovat alkuperäisen matriisin ominaisarvojen käänteislukuja. Oletetaan, että matriisin A ominaisarvot toteuttavat ehdon $0 < |\lambda_n| < |\lambda_{n-1}| \leq \ldots \leq |\lambda_1|$. Jos matriisi A on säännöllinen, saadaan potenssimenetelmää matriisiin A^{-1} soveltamalla laskettua matriisin A itseisarvoltaan pienin ominaisarvo λ_n. Tätä menetelmää kutsutaan *käänteiseksi potenssimenetelmäksi* (inverse iteration).

Käänteismatriisin muodostaminen iteraatiota varten kannattaa vain harvoin. Niinpä käänteisessä potenssimenetelmässä yleensä ratkaistaan lineaarinen



Kuva 9.2: *Potenssimenetelmä kääntää aloitusvektorin* $\mathbf{q}_0$ *ominaisvektorin* $\mathbf{x}_1$ *suuntaiseksi.*



Kuva 9.3: *Potenssimenetelmän tuottamat approksimaatiot suurimmalle ominaisarvolle.*

yhtälöryhmä jokaisella iteraatiokierroksella. Käytännössä on suositeltavaa laskea ensin LU-hajotelma matriisille A, jolloin yhtälöryhmän ratkaiseminen iteraation aikana käy nopeasti.

Käänteisen potenssimenetelmän perusalgoritmi on esitetty seuraavassa.

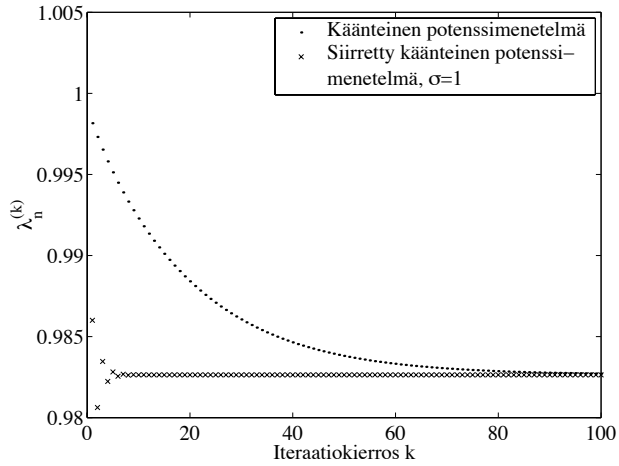
Algoritmi 9.4.2 (Käänteinen potenssimenetelmä)

Valitse aloitusvektori $\mathbf{x}_0$
for $k = 1, 2, 3 \ldots$
  $A\mathbf{y}_k = \mathbf{x}_{k-1}$ (yhtälöryhmän ratkaisu)
  $\mathbf{x}_k = \mathbf{y}_k / \|\mathbf{y}_k\|$
end

Itseisarvoltaan pienimmälle ominaisarvolle saadaan iteraatiokierroksen m jälkeen approksimaatio kaavasta

$$1/\lambda_n \approx \langle \mathbf{x}_{m-1}, A^{-1}\mathbf{x}_{m-1} \rangle = \langle \mathbf{x}_{m-1}, \mathbf{y}_m \rangle.$$

Esimerkki 9.4.2 Etsimme käänteisellä potenssimenetelmällä esimerkin 9.4.1 matriisin itseisarvoltaan pienimmän ominaisarvon 0.9826. Kuten kuvasta 9.4 näkyy, suppeneminen on erittäin hidasta. Sadan iteraatiokierroksen jälkeen saamme approksimaation 0.9827.



Kuva 9.4: *Käänteisen potenssimenetelmän ja siirretyn käänteisen potenssimenetelmän tuottamat approksimaatiot itseisarvoltaan pienimmälle ominaisarvolle.*

Käänteiseen potenssimenetelmään voidaan yhdistää *origon siirto* (shift). Tällöin haetaan matriisin $(A - \sigma I)^{-1}$ itseisarvoltaan suurinta ominaisarvoa $1/(\lambda' - \sigma)$, missä λ' on matriisin A se ominaisarvo, jonka etäisyys siirtoparametrista σ on pienin. Tätä muunnosta kutsutaan nimellä *shift-invert*. Näin saatu menetelmä on *siirretty käänteinen potenssimenetelmä*.

Alkuperäiset ominaisarvot siis kuvautuvat matriisin $(A - \sigma I)^{-1}$ ominaisarvoiksi kaavan $\lambda \mapsto 1/(\lambda - \sigma)$ mukaisesti, missä σ on valittu siirtoparametri. Suppenemisnopeuden määrää suhde

$$\frac{|\lambda_1' - \sigma|}{|\lambda_2' - \sigma|},$$

missä λ_1' ja λ_2' ovat ne matriisin A ominaisarvot, jotka ovat lähinnä siirtoparametria σ. Suhteen voi periaatteessa saada mielivaltaisen pieneksi sopivan siirtoparametrin avulla.

Niinpä käänteinen potenssimenetelmä suppenee kohti matriisin A mielivaltaista ominaisarvoa, kunhan siirtoparametri on lähempänä sitä kuin muita ominaisarvoja.

Eräissä ohjelmistoissa siirrettyä käänteistä potenssimenetelmää käytetään ominaisvektorien määrittämiseen, kun ominaisarvot ovat tiedossa.

Esimerkki 9.4.3 Sovellamme siirrettyä käänteistä potenssimenetelmää esimerkin 9.4.2 ongelmaan. Oletetaan, että tiedämme itseisarvoltaan pienimmän ominaisarvon olevan lähempänä ykköstä kuin muut ominaisarvot. Valitsemalla siirtoparametrin arvoksi $\sigma = 1$ saamme ominaisarvon neljän desimaalin tarkkuudella jo kahdeksan iteraatiokierroksen jälkeen.

Suppenemiskäyttäyminen on esitetty kuvassa 9.4. Ero ilman siirtoa tehtyyn laskuun on huomattava, mutta edellytyksenä oli, että pystyimme edes karkeasti arvioimaan halutun ominaisarvon sijainnin.

Siirtoparametrin arvoa voidaan muuttaa myös iteraation aikana. Käänteistä potenssimenetelmää käytettäessä joudutaan laskemaan uusi LU-hajotelma joka kerta, kun siirtoparametria muutetaan. *Rayleigh'n osamääräiteroinnissa* (Rayleigh quotient iteration) käytetään siirtoparametrina *Rayleigh'n osamäärää*

$$\rho(\mathbf{x}_k) = \frac{(\mathbf{x}_k, A\mathbf{x}_k)}{(\mathbf{x}_k, \mathbf{x}_k)}.$$

Jos $\mathbf{x}_k$ on lähellä jotain ominaisvektoria, Rayleigh'n osamäärä on hyvä approksimaatio vastaavalle ominaisarvolle. Seuraavassa on esitetty iteraation toiminta.

Algoritmi 9.4.3 (Rayleigh'n osamääräiteraatio)

Valitse aloitusvektori $\mathbf{x}_0$
for $k = 1, 2, 3, \ldots$
  $\rho_k = \rho(\mathbf{x}_{k-1})$
  $(A - \rho_k I)\mathbf{y}_k = \mathbf{x}_{k-1}$ (yhtälöryhmän ratkaisu)
  $\mathbf{x}_k = \mathbf{y}_k / \|\mathbf{y}_k\|$
end

Iteraatio suppenee nopeasti, mutta löytyvä ominaisvektori ja ominaisarvo riippuvat valitusta alkuvektorista.

9.4.3 Deflaatio

Kun esimerkiksi potenssimenetelmällä tai käänteisellä potenssimenetelmällä on laskettu yksi ominaisarvo, voidaan siirtyä seuraavaan ominaisarvoon, kun ensimmäisen ominaisarvon vaikutus matriisiin on tietyssä mielessä poistettu. Tätä menettelyä kutsutaan *deflaatioksi*.

Olkoon λ_1 ominaisarvo ja $\mathbf{z}_1$ vastaava normeerattu ominaisvektori. Wielandtin deflaatiossa siirrytään käsittelemään matriisia

$$\tilde{A} = A - \sigma \mathbf{z}_1 \mathbf{x}^H,$$

missä σ on parametri ja vektori $\mathbf{x}$ toteuttaa ehdon $\mathbf{x}^H \mathbf{z}_1 = 1$. Voidaan osoittaa, että matriisin $\tilde{A}$ spektri on $\{\lambda_1 - \sigma, \lambda_2, \ldots, \lambda_p\}$.

Niinpä esimerkiksi matriisin

$$\tilde{A} = A - \lambda_1 \mathbf{z}_1 \mathbf{z}_1^H.$$

ominaisarvot ovat muuten samat kuin alkuperäisen matriisin, mutta ominaisarvo λ_1 on muutettu nollaksi.

Nyt esimerkiksi potenssimenetelmän soveltaminen matriisiin $\tilde{A}$ tuottaa matriisin A itseisarvoltaan toiseksi suurimman ominaisarvon. Menetelmää voidaan toistaa, kunnes haluttu määrä ominaisarvoja on laskettu.

Toisaalta potenssimenetelmää tai käänteistä potenssimenetelmää voidaan soveltaa siten, että valitaan lähtövektori, joka on kohtisuorassa jo löydettyä ominaisvektoria vastaan. Laskennan kuluessa täytyy kuitenkin huolehtia siitä, etteivät pyöristysvirheet tuo laskettaviin vektoreihin aikaisemmin löydetyn ominaisvektorin suuntaisia komponentteja.

9.4.4 Aliavaruusmenetelmä

Aliavaruusmenetelmässä (orthogonal iteration) toimitaan kuten potenssimenetelmässä, mutta lasketaan $m < n$ vektorilla samanaikaisesti. Menetelmä ei tuota suoraan ominaisarvoja eikä -vektoreita, vaan dominoivien ominaisvektorien virittämän aliavaruuden kannan.

Aliavaruusmenetelmää käytetään useimmiten shift-invert-muunnoksen kanssa hakemaan siirtoa σ lähimpänä olevat m ominaisarvoa ja vastaavat ominaisvektorit. Vaikka aliavaruusmenetelmä ei ole kovin tehokas, sitä sovelletaan toisinaan myös suuriin tehtäviin.

Laskennan aikana on huolehdittava vektoreiden ortonormaalisuudesta. Ilman ortogonalisointia kaikki vektorit kääntyisivät dominoivan ominaisvektorin suuntaiseksi. Potenssimenetelmän ominaisuuksiin kuuluu, että dominoivien ominaisvektorien suuntaiset komponentit vahvistuvat. Iteraatio tuottaa approksimaation dominoivien ominaisvektoreiden $\{x_i, \ldots, x_m\}$ virittämän invariantin aliavaruuden ortonormaalille kannalle.

Algoritmi 9.4.4 (Aliavaruusmenetelmä)

Valitse aloitusvektorit sisältävä matriisi S_0
for $k = 1, 2, 3, \ldots$
$C_k = AS_{k-1}$
$C_k = Q_k R_k$ (QR-hajotelma)
$S_k = Q_k$
end

Ortonormeeraus tehdään QR-hajotelman avulla ja matriisit S_k sisältävät kantavektoreiden approksimaatiot.

Matriisit S_k, C_k ja Q_k ovat kokoa $n \times m$ ja matriisi R_k kokoa $m \times m$. Yleensä $n \times m$-matriisin QR-hajotelmassa Q on kokoa $n \times n$ ja R kokoa $n \times m$, mutta tässä matriisin R lävistäjän alapuoliset nolla-alkiot ja matriisin Q vastaavat alkiot jätetään ottamatta huomioon.

Ortonormeeraus, eli QR-hajotelman laskeminen, on kallis operaatio, eikä sitä tarvitse välttämättä tehdä joka askeleella.

Halutut approksimaatiot ominaisarvoille saadaan laskemalla projisoidun matriisin $S_k^H A S_k$ ominaisarvot. Ominaisvektorit saadaan projisoidun matriisin ominaisvektoreista kertomalla ne vasemmalta matriisilla S_k.

Aliavaruusmenetelmän suppenemisnopeus riippuu $m+1$:n dominoivan ominaisarvon välisistä suhteista, kuten potenssimenetelmän perusteella on helppo uskoa. Voidaan osoittaa, että aliavaruusmenetelmälle pätee

$$|\lambda_i^{(k)} - \lambda_i| \approx \left| \frac{\lambda_{i+1}}{\lambda_i} \right|^k,$$

missä $i = 1, 2, \ldots, m$.

Aliavaruusmenetelmän suppenemista voidaan kiihdyttää sopivalla projektiomenetelmällä. *Rayleigh'n ja Ritzin menettely*, jossa lasketaan projisoidun matriisin $H_k = Q_k^H A Q_k$ Schurin vektorit ja muunnetaan ne takaisin lähtöavaruuteen, tuottaa tietyssä mielessä optimaaliset approksimaatiot kantavektoreille. Projektiota hyödyntävä versio aliavaruusiteraatiosta on seuraava:

Algoritmi 9.4.5 (Kiihdytetty aliavaruusmenelmä)

Valitse aloitusvektorit sisältävä matriisi S_0
for $k = 1, 2, 3, \ldots$
$C_k = AS_{k-1}$
$C_k = Q_k R_k$ (QR-hajotelma)
$H_k = Q_k^H A Q_k$
$H_k = U_k T_k U_k^H$ (Schurin hajotelma)
$S_k = Q_k U_k$
end

Hermiittisessä tapauksessa Schurin hajotelman yläkolmiomatriisi T_k pelkistyy matriisin H_k ominaisarvojen muodostamaksi diagonaalimatriisiksi ja matriisin U_k pystyvektorit ovat vastaavia ominaisvektoreita.

Kiihdytetyn version suppeneminen puolestaan riippuu m:n dominoivan ominaisarvon suhteista itseisarvoltaan seuraavaksi suurimpaan ominaisarvoon. Menetelmä suppenee hyvin, jos matriisin m dominoivaa ominaisarvoa ovat itseisarvoiltaan selvästi suurempia kuin muut ominaisarvot.

9.4.5 QR-menetelmä

QR-menetelmä on tehokas tapa matriisin kaikkien ominaisarvojen hakemiseen ja perustuu nimensä mukaisesti QR-hajotelman käyttöön.

Oletamme, että matriisin A ominaisarvot ovat itseisarvoiltaan eri suuria ja yritämme laskea ne kaikki aliavaruusmenetelmällä. Tietyin edellytyksin jono

$$T_k = Q_k^H A Q_k,$$

missä matriisit Q_k ovat aliavaruusiteraation tuottamat approksimatiiviset kantavektorimatriisit, suppenee kohti yläkolmiomatriisia, jonka lävistäjäalkiot ovat matriisin A ominaisarvot. Voidaan siis sanoa, että aliavaruusmenetelmä tuottaa matriisin A Schurin hajotelman.

QR-algoritmiin päädytään havaitsemalla, että T_k voidaan laskea kertomalla edellisen iteraatin T_{k-1} QR-hajotelman tekijät päinvastaisessa järjestyksessä:

$$\begin{aligned} \text{QR-hajotelma } T_{k-1} &= Q_k R_k \\ \text{Aseta } T_k &= R_k Q_k. \end{aligned}$$

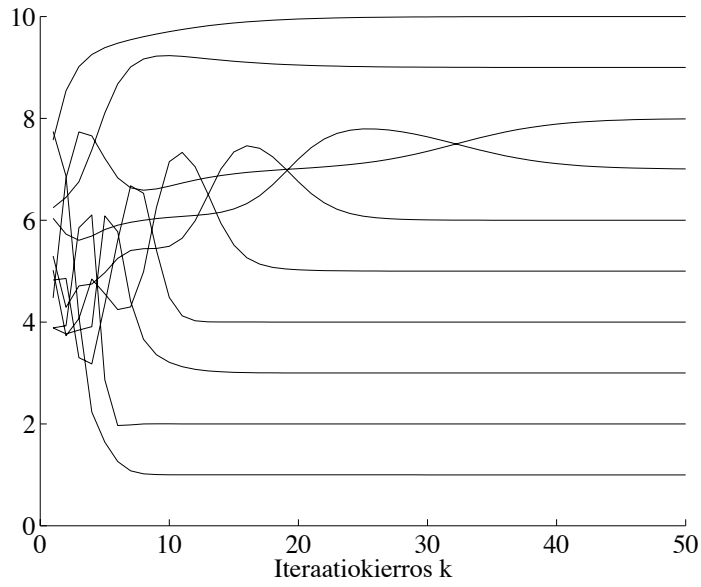
Tällaisenaan menetelmä kuitenkin vaatii liikaa laskentatyötä ollakseen tehokas. Käyttökelpoisempi versio saadaan similaarimuuntamalla matriisi A ensin hermiittisessä tapauksessa tridiagonaalimuotoon ja ei-hermiittisessä tapauksessa Hessenbergin muotoon:

$$T_0 = U_0^H A U_0.$$

Muunnosmatriisi U_0 voidaan laskea Householderin muunnoksilla tai Givensin rotaatioilla. Iteraation aikana matriisit T_k pysyvät tridiagonaalisessa tai Hessenbergin muodossa, mikä vähentää laskentatyötä huomattavasti. QR-algoritmin perusversio on esitetty seuraavassa.

Algoritmi 9.4.6 (QR-menetelmä)

$T_0 = U_0^H A U_0$
for $k = 1, 2, 3, \ldots$
  $T_{k-1} = Q_k R_k$ (QR-hajotelma)
  $T_k = R_k Q_k$
end



Kuva 9.5: *QR-menetelmän tuottamien matriisien T_k lävistäjäalkioiden suppeneminen kohti ominaisarvoja.*

Esimerkki 9.4.4 Laskemme 10×10 -kokoisen symmetrisen matriisin ominaisarvot QR-menetelmällä. Iteraation tuottamien matriisien T_k lävistäjäalkioiden suppeneminen kohti ominaisarvoja näkyy kuvassa 9.5.

Kuva 9.6 esittää alilävistäjän elementtien suppenemisen kohti nollaa. Viidenkymmenen iteraatiokierroksen jälkeen saamme arvot 9.9999, 9.0001, 7.9914, 7.0086, 6.0000, 5.0000, 4.0000, 3.0000, 2.0000, 1.0000. Oikeat ominaisarvot ovat $1, 2, 3, \ldots, 10$.

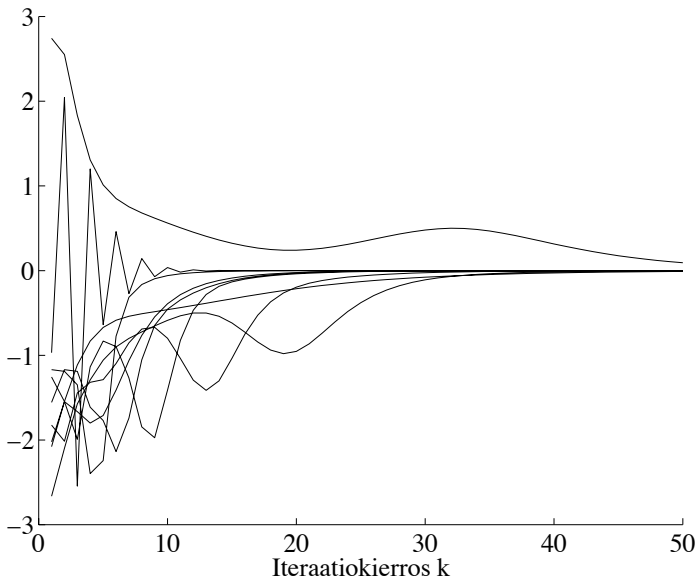
QR-algoritmin suorituskykyä parannetaan käytännössä aina käyttämällä iteraatiossa suppenemista kiihdyttävää siirtomenettelyä. Voidaan osoittaa, että matriisin T_k alidiagonaalin i:s alkio suppenee kohti nollaa nopeudella

$$\left| \frac{\lambda_{i+1}}{\lambda_i} \right|^k .$$

Niinpä sovellettaessa QR-menetelmää siirrettyyn matriisiin $A - \mu I$ suppenemisnopeus on

$$\left| \frac{\lambda_{i+1} - \mu}{\lambda_i - \mu} \right|^k ,$$

mikä saadaan mielivaltaisen suureksi olettaen, että $\lambda_{i+1} \neq \lambda_i$. Nopea suppeneminen saavutetaan valitsemalla sopiva siirto joka iteraatiokierroksella.



Kuva 9.6: *QR-menetelmän tuottamien matriisien T_k alilävistäjien alkioiden suppeneminen kohti nollaa.*

Yksinkertainen heuristinen siirtostrategia on valita $\mu_k = T_k(n,n)$. Algoritmi on esitetty seuraavassa.

Algoritmi 9.4.7 (Kiihdytetty QR-menetelmä)

$T_0 = U_0^H A U_0$
for $k = 1, 2, 3, \ldots$
  $\mu_{k-1} = T_{k-1}(n,n)$
  $T_{k-1} - \mu_{k-1} I = Q_k R_k$ (QR-hajotelma)
  $T_k = R_k Q_k + \mu_{k-1} I$
end

Kun siirtoparametri on lähellä ominaisarvoa, menetelmä suppenee neliöllisesti ja antaa yhden ominaisarvon approksimaation matriisin T_k viimeisessä lävistäjäalkiossa. Tämän jälkeen iteraatiota voidaan jatkaa pienemmällä matriisilla $T_k(1:n-1, 1:n-1)$.

Jos matriisilla on kompleksisia ominaisarvoja, siirtoparametri μ_k on huono approksimaatio ominaisarvolle eikä toivottua suppenemisen kiihtymistä tapahdu. Ratkaisu ongelmaan on käyttää kahta peräkkäistä kompleksista siirtoa a_1 ja a_2, jotka ovat matriisin $T_k(n-1:n, n-1:n)$ ominaisarvot. Tridiagonaalisessa tapauksessa kannattaa käyttää sitä matriisin $T_k(n-1:n, n-1:n)$ ominaisarvoa, joka on lähempänä alkiota $T_k(n,n)$. Tätä kutsutaan *Wilkinsonin siirroksi*.

Wilkinsonin siirtoa käytettäessä hermiittinen QR-algoritmi suppenee kuutiollisesti. Yhtä ominaisarvoa kohti tarvitaan keskimäärin vain kaksi QR-iteraatiokierrosta, joiden jälkeen iteraatiota jatketaan pienemmällä matriisilla.

9.4.6 Divide-and-conquer

Divide-and-conquer on nopea menetelmä hermiittisen tridiagonaalimatriisin kaikkien ominaisarvojen ja -vektorien hakemiseen. Menetelmää voi soveltaa myös yleiseen hermiittiseen matriisiin, kunhan se ennen laskentaa similaarimuunnetaan tridiagonaalimuotoon esimerkiksi Householderin muunnosten avulla.

Divide-and-conquer on rekursiivinen algoritmi, joka perustuu tridiagonaalimatriisin osittamiseen. Rajoitumme seuraavassa reaaliseen tapaukseen. Olkoon T symmetrinen $n \times n$-tridiagonaalimatriisi, ja oletetaan yksinkertaisuuden vuoksi, että $n = 2m$. Nyt voidaan kirjoittaa

$$T = \begin{pmatrix} T_1 & 0 \\ 0 & T_2 \end{pmatrix} + t_{m,m+1}\mathbf{v}\mathbf{v}^T,$$

missä n-vektori $\mathbf{v}$ on

$$v_i = \begin{cases} 1, & \text{kun } i = m,\ \ m+1, \\ 0, & \text{muulloin.} \end{cases}$$

Tridiagonaalimatriisien T_1 ja T_2 alkiot saadaan suoraan vastaavilta paikoilta alkuperäisestä matriisista T sillä poikkeuksella, että $(t_1)_{m,m} = t_{m,m} - t_{m,m+1}$ ja $(t_2)_{1,1} = t_{m+1,m+1} - t_{m,m+1}$.

Oletetaan nyt, että matriisit T_1 ja T_2 on jo diagonalisoitu (rekursio), jolloin on olemassa ortogonaaliset matriisit Q_1 ja Q_2 siten, että matriisit $Q_1^T T_1 Q_1 = \Lambda_1$ ja $Q_2^T T_2 Q_2 = \Lambda_2$ ovat diagonaalisia. Tehdään matriisilla

$$U = \begin{pmatrix} Q_1 & 0 \\ 0 & Q_2 \end{pmatrix}$$

similaarimuunnos

$$U^T T U = \Lambda + t_{m,m+1}\mathbf{z}\mathbf{z}^T,$$

missä

$$\Lambda = \begin{pmatrix} \Lambda_1 & 0 \\ 0 & \Lambda_2 \end{pmatrix}$$

ja

$$z = U^T\mathbf{v}.$$

Matriiseilla T ja $\Lambda + t_{m,m+1}\mathbf{z}\mathbf{z}^T$ on siis samat ominaisarvot.

Menetelmän nopeus seuraa siitä, että yksinkertaisen rakenteen vuoksi matriisin $\Lambda + t_{m,m+1}\mathbf{z}\mathbf{z}^T$ ominaisarvot ja -vektorit ovat helposti laskettavissa.

Ominaisarvot haetaan karakteristisen polynomin juurina käyttäen Newtonin menetelmää (kappale 6.3.3).

Rekursiivinen algoritmi on kokonaisuudessaan seuraava:

Algoritmi 9.4.8 (Divide-and-Conquer)

procedure DaC(T,Q,Λ)
  input: T
  output: Q,Λ
  if matriisin T koko on 1×1
    $Q = 1, \Lambda = T$
  else
    $T = \text{diag}(T_1, T_2) + t_{m,m+1}\mathbf{v}\mathbf{v}^T$
    call DaC(T_1,Q_1,Λ_1)
    call DaC(T_2,Q_2,Λ_2)
    Muodosta $D = \Lambda + t_{m,m+1}\mathbf{z}\mathbf{z}^T$ matriisien Λ_1,Λ_2,Q_1 ja Q_2 avulla
    Laske matriisin D ominaisarvot Λ ja ominaisvektorit Q'
    $Q = \text{diag}(Q_1, Q_2)Q'$
  end if

9.4.7 Puolitushakumenetelmät

Puolitushakumenetelmät soveltuvat vain hermiittisille matriiseille. Puolitusmenetelmillä voidaan hakea suoraan k:nneksi suurin ominaisarvo tai etsiä kaikki ominaisarvot halutulta väliltä. Menetelmät perustuvat funktioihin, joiden arvo kertoo argumenttia pienempien ominaisarvojen lukumäärän. Tällöin yksinkertaisella puolitushaulla voidaan haarukoida haluttu ominaisarvo.

Ensimmäinen askel on matriisin redusoiminen tridiagonaalimuotoon esimerkiksi Householderin muunnoksilla.

Olkoon $T_r = T(1:r, 1:r)$, missä T on hermiittinen tridiagonaalimatriisi, ja määritellään polynomit $p_r(x) = \det(T_r - xI)$, $r = 1, 2, \ldots, n$. Kertokoon luku $a(\mu)$, kuinka monta kertaa etumerkki vaihtuu jonossa

$$1, p_1(\mu), p_2(\mu), \ldots, p_n(\mu),$$

kun etumerkin katsotaan vaihtuvan myös nollan kohdalla. Tällöin $a(\mu)$ on matriisin T lukua μ pienempien ominaisarvojen lukumäärä.

Hermiittisen matriisin A tiettyä lukua pienempien ominaisarvojen lukumäärä saadaan myös LDL^H-hajotelman avulla. Olkoon

$$LDL^H = A - \mu I,$$

missä $D = \text{diag}(d_1, \ldots, d_n)$. Matriisilla D on negatiivisia alkioita yhtä paljon kuin matriisilla A on lukua μ pienempiä ominaisarvoja.

Esimerkki 9.4.5 Tarkastelemme esimerkkinä symmetristä matriisia

$$A = \begin{pmatrix} 5 & -1 & -1 \\ -1 & 4 & -1 \\ -1 & -1 & 4 \end{pmatrix}.$$

Matriisin A LDL^T-hajotelma on

$$A = \begin{pmatrix} 1 & 0 & 0 \\ -0.2 & 1 & 0 \\ -0.2 & -0.3158 & 1 \end{pmatrix} \begin{pmatrix} 5 & 0 & 0 \\ 0 & 3.8 & 0 \\ 0 & 0 & 3.4211 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ -0.2 & 1 & 0 \\ -0.2 & -0.3158 & 1 \end{pmatrix}^T,$$

ja matriisille $A - 3I$ saamme vastaavasti hajotelman

$$A - 3I = \begin{pmatrix} 1 & 0 & 0 \\ -0.5 & 1 & 0 \\ -0.5 & -3 & 1 \end{pmatrix} \begin{pmatrix} 2 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & -4 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ -0.5 & 1 & 0 \\ -0.5 & -3 & 1 \end{pmatrix}^T.$$

Voimme siis päätellä, että matriisilla A on yksi ominaisarvo välillä $(0, 3)$. Puolittamalla välin ja tekemällä LDL^T-hajotelman matriisille $A - 1.5I$ havaitsemme, että ominaisarvo on välillä $(1.5, 3)$. Näin jatkamalla voimme löytää ominaisarvon 2.2679 halutulla tarkkuudella.

Puolitushaku on melko hidas, jos ominaisarvo halutaan laskea tarkasti. Menetelmää voidaan kuitenkin käyttää muilla menetelmillä laskettujen tulosten tarkistamiseen ja toisaalta alkuarvauksien hakemiseen.

9.5 Suuret ominaisarvotehtävät

Vaikka matriisi olisi hyvin suuri, ominaisarvo-ongelman voi yrittää ratkaista tietokoneella, jos matriisi on riittävän harva tai muodostettavissa siten, ettei muistia tarvita liikaa. On oleellista, että nollia ei talleteta muistiin eikä niillä lasketa. Käytettävä menetelmä ei saa aiheuttaa matriisin täyttymistä eli nollasta poikkeavien alkioiden lisääntymistä.

Potenssimenetelmä ja aliavaruusmenetelmä soveltuvat vähäisen muistintarpeensa vuoksi myös suurten tehtävien ratkaisemiseen. Krylovin menetelmät ovat kuitenkin tehokkaampia.

9.5.1 Krylovin menetelmät

Krylovin menetelmät pohjautuvat tavallaan potenssimenetelmään. Kuten aliavaruusmenetelmässä Krylovin menetelmissä alkuperäinen matriisi projisoidaan aliavaruuteen. Yksinkertaisimmillaan käytettävä aliavaruus muodostetaan matriisi-vektoritulojen avulla. Näin saatua aliavaruutta

$$K_m = \text{span}\{\mathbf{v}, A\mathbf{v}, A^2\mathbf{v}, \ldots, A^{m-1}\mathbf{v}\}$$

kutsutaan alkuvektorin $\mathbf{v}$ virittämäksi matriisin A m-dimensioiseksi *Krylovin aliavaruudeksi*.

Arnoldin menetelmässä lasketaan ominaisarvot Krylovin aliavaruuteen K_m projisoidulle, alkuperäistä pienemmälle matriisille $G_m = V_m^H A V_m$, missä matriisin V_m pystyvektorit muodostavat Krylovin aliavaruuden ortonormaalin kannan. Usein jo pienillä m:n arvoilla matriisin G_m äärimmäiset ominaisarvot ovat hyviä approksimaatioita alkuperäisen matriisin A äärimmäisille ominaisarvoille. Äärimmäisillä ominaisarvoilla tarkoitetaan karkeasti sanottuna spektrin reunaa lähinnä olevia ominaisarvoja.

Approksimatiivinen ominaisarvotehtävä on siis

$$G_m \mathbf{s} = \theta \mathbf{s}.$$

Ominaisarvoa θ kutsutaan *Ritzin arvoksi* ja takaisin muunnettua ominaisvektoria $\mathbf{x} = V_m \mathbf{s}$ *Ritzin vektoriksi*.

Arnoldin menetelmä tuottaa projisoidun matriisin G_m, jonka ominaisarvot on laskettava vielä erikseen. Matriisi G_m on muodoltaan Hessenbergin matriisi.

Kun Arnoldin menetelmää sovelletaan hermiittiseen matriisiin, matriisi G_m on tridiagonaalinen ja laskenta yksinkertaistuu huomattavasti. Menetelmää kutsutaan tällöin *Lanczosin menetelmäksi*.

Arnoldin tai Lanczosin menetelmä on hyvä valinta silloin, kun ominaisarvoista halutaan laskea vain pieni joukko. Menetelmät soveltuvat suuriin tehtäviin, koska approksimatiivinen ongelma on pieni ja laskennan aikana tarvitaan vain matriisi-vektorituloja. Suurissa tehtävissä matriisi on yleensä harva ja tulon laskeminen on usein mahdollista optimoida ottamalla matriisin rakenne huomioon. Lisäksi menetelmien muistinkäyttö on mahdollista pitää pienenä niin sanotulla uudelleenkäynnistyksellä.

Seuraavassa esitellään menetelmien tarkan aritmetiikan perusversiot. Käytännössä algoritmeja joudutaan muokkaamaan huomattavasti, jotta äärellisellä tarkkuudella laskeminen onnistuisi. Erityisesti alivaruuden kannan ortogonaalisuuden säilyttäminen on ongelmallista. Myös muistinkäytön optimoiminen edellyttää muutoksia algoritmeihin.

9.5.2 Arnoldin menetelmä

Arnoldin menetelmän tarkan aritmetiikan perusversiossa Krylovin aliavaruudelle muodostetaan ortonormaali kanta $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m\}$ Gramin ja Schmidtin ortonormeerausprosessilla:

Algoritmi 9.5.1 (Arnoldin menetelmä)

Valitse ykkösen pituinen aloitusvektori $\mathbf{v}_1$.
for $j = 1, \ldots, m-1$
  $h_{ij} = \langle A\mathbf{v}_i, \mathbf{v}_j \rangle, \quad i = 1, 2, \ldots, j$
  $\mathbf{w}_j = A\mathbf{v}_j - \sum_{i=1}^{j} h_{ij}\mathbf{v}_i$

$h_{j+1,j} = \|\mathbf{w}_j\|_2$
$\mathbf{v}_{j+1} = \mathbf{w}_j / h_{j+1,j}$
end

Projisoitu Hessenbergin matriisi $G_m = V_m^H A V_m$ voidaan muodostaa samalla kuin aliavaruuden kantakin, jolloin algoritmista saadaan matriisinotaatiolla seuraava versio:

Algoritmi 9.5.2 (Arnoldin menetelmä matriisinotaatiolla)

Valitse ykkösen pituinen aloitusvektori $\mathbf{v}_1$.
$\mathbf{w} = A\mathbf{v}_1$
$\alpha_1 = \langle \mathbf{v}_1, \mathbf{w} \rangle$
$\mathbf{f}_1 = \mathbf{w} - \alpha_1 \mathbf{v}_1$
$V_1 = (\mathbf{v}_1)$ (sarakevektori)
$G_1 = (\alpha_1)$ (yksi alkio)
for $j = 1, \ldots, m-1$
$\beta_j = \|\mathbf{f}_j\|$
$\mathbf{v}_{j+1} = \mathbf{f}_j / \beta_j$
$V_{j+1} = (V_j, \mathbf{v}_{j+1})$ (lisätään sarake)
$\hat{G}_j = \begin{pmatrix} G_j \\ \beta_j \mathbf{e}^{jT} \end{pmatrix}$, missä $e_i^j = \delta_{ij}$
$\mathbf{w} = A\mathbf{v}_{j+1}$
$\mathbf{g} = V_{j+1}^H \mathbf{w}$
$\mathbf{f}_{j+1} = \mathbf{w} - V_{j+1}\mathbf{g}$
$G_{j+1} = (\hat{G}_j, \mathbf{g})$
end

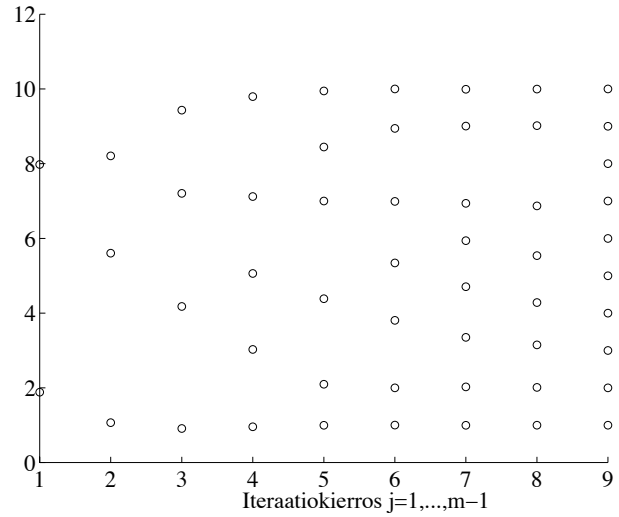
Esimerkki 9.5.1 Tarkastelemme 10×10 -matriisia, jonka ominaisarvot ovat $1, 2, 3, \ldots, 10$. Viiden iteraatiokierroksen jälkeen saamme kokoa 5×5 olevan projisoidun matriisin, jonka ominaisarvot ovat

$$0.9976, 2.0954, 4.3854, 7.0015, 8.4482 \text{ ja } 9.9477.$$

Äärimmäiset ominaisarvot 0.9976 ja 9.9477 alkavat olla jo melko lähellä oikeita arvoja. Projisoitujen matriisien ominaisarvot on esitetty iteraatiokierroksittain kuvassa 9.7. Kuvasta näkyy selvästi, että suurin ja pienin ominaisarvo suppenevat nopeimmin.

9.5.3 Lanczosin menetelmä

Kun Arnoldin menetelmää sovelletaan hermiittiseen matriisiin, voidaan algoritmia yksinkertaistaa ja tuloksena on *Lanczosin menetelmä*. Arnoldin menetelmän tuottamat Hessenbergin matriisit G_m ovat tässä tapauksessa reaalisia ja symmetrisiä, eli ne ovat tridiagonaalimatriiseja.



Kuva 9.7: *Arnoldin menetelmän tuottamien matriisien ominaisarvot.*

Lähtökohtana on unitaarinen muunnosmatriisi V, jolla matriisi A saatetaan tridiagonaalimuotoon

$$T = V^H A V, \tag{9.9}$$

missä reaalinen ja symmetrinen T on muotoa

$$\begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{n-1} & \alpha_{n-1} & \beta_n \\ & & & \beta_n & \alpha_n \end{pmatrix}.$$

Unitaarisen matriisin V pystyvektoreiden $\mathbf{v}_i$ ja tridiagonaalimatriisin T alkioiden välille saadaan edellisen perusteella palautuskaava

$$A\mathbf{v}_i = \beta_i \mathbf{v}_{i-1} + \alpha_i \mathbf{v}_i + \beta_{i+1} \mathbf{v}_{i+1},$$

joka voidaan muokata Lanczosin menetelmäksi:

Algoritmi 9.5.3 (Lanczosin menetelmä)

Valitse ykkösen pituinen aloitusvektori $\mathbf{v}_1$.
$\beta_1 = 0$
$\mathbf{v}_0 = 0$
for $j = 1, \ldots, m-1$
  $\mathbf{w}_j = A\mathbf{v}_j - \beta_j \mathbf{v}_{j-1}$
  $\alpha_j = \langle \mathbf{w}_j, \mathbf{v}_j \rangle$
  $\mathbf{w}_j = \mathbf{w}_j - \alpha_j \mathbf{v}_j$

$\beta_{j+1} = \|\mathbf{w}_j\|_2$
$\mathbf{v}_{j+1} = \mathbf{w}_j / \beta_{j+1}$
end

Vektorit $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m\}$ muodostavat ortogonaalisen kannan Krylovin aliavaruudelle K_m. Kuten Arnoldin menetelmässäkin, projisoidun matriisin $G_m = V_m^H A V_m$ äärimmäiset ominaisarvot ovat usein hyviä approksimaatioita matriisin A äärimmäisille ominaisarvoille jo suhteellisen pienillä indeksin m arvoilla.

Lanczosin algoritmista voidaan johtaa menetelmiä myös lineaaristen yhtälöryhmien ratkaisemiseen. Hermiittisille matriiseille Lanczosin algoritmin vektorit $\mathbf{v}_j$ ovat kerrointa vailla samat kuin liittogradienttialgoritmin tuottamat residuaalit $\mathbf{r}_i$, jos Lanczosin algoritmissa alkuvektoriksi otetaan ratkaistavan lineaarisen yhtälöryhmän oikean puolen vektori $\mathbf{b}$. Myös iteratiivinen menetelmä QMR on johdettu Lanczosin algoritmista.

9.5.4 Suppenemisen kiihdyttäminen

Shift-invert-menettely sopii myös Krylovin menetelmien suppenemisen kiihdyttämiseen. Tällöin lasketaan matriisilla $(A - \sigma I)^{-1}$, ja lähinnä siirtoa σ olevat matriisin A ominaisarvot kuvautuvat matriisin $(A - \sigma I)^{-1}$ spektrin reunoille. Näin varsinkin spektrin sisällä olevien ominaisarvojen hakeminen nopeutuu huomattavasti. Siirtoparametrin sopiva valinta vaatii kuitenkin tietoa ominaisarvojen likimääräisestä sijainnista.

Shift-invert-kiihdytys edellyttää lineaarisen yhtälöryhmän ratkaisemista joka kierroksella esimerkiksi LU-hajotelmaa käyttämällä.

9.6 Yleistetyt tehtävät

Esimerkki 9.6.1 Rakenneanalyysissä halutaan usein tietää suurin kuorma, jonka tietty rakenne kestää. Tässä esimerkissä tarkastelemme T-palkkia.

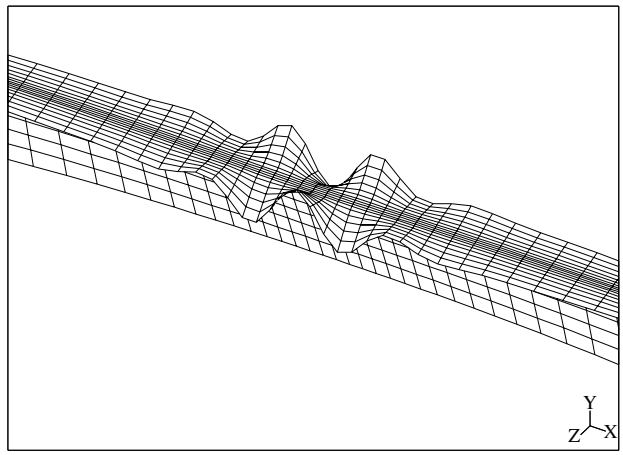
Palkkiyhtälöiden diskretointi elementtimenetelmällä (luku 8) kuorielementtejä käyttäen johtaa yleistettyyn ominaisarvotehtävään

$$K\mathbf{x} = \lambda G\mathbf{x},$$

missä K on alkutilan jäykkyysmatriisi ja G on geometrinen jäykkyysmatriisi (tai alkujännitysmatriisi). Molemmat matriisit ovat symmetrisiä, ja K on positiivisesti definiitti ja G on indefiniitti. Ominaisarvot kuvaavat eri lommahdustiloihin johtavia kuormaintensiteettejä. Kuvassa 9.8 näkyy palkin paikallinen lommahdus.

Olkoot A ja B $n \times n$-neliömatriiseja. *Yleistetyssä ominaisarvotehtävässä* etsitään ominaisarvoa λ ja vektoria $\mathbf{x} \neq 0$, jotka toteuttavat yhtälön

$$A\mathbf{x} = \lambda B\mathbf{x}.$$



Kuva 9.8: *T-palkin paikallinen lommahdus. Kuva: Reijo Kouhia, TKK.*

Tietyissä tapauksissa on mahdollista palauttaa yleistetty tehtävä tavalliseksi ominaisarvotehtäväksi, mutta toisinaan ongelman ratkaiseminen suoraan on ainoa mahdollisuus.

9.6.1 Palauttaminen tavalliseksi tehtäväksi

Jos matriisi B on hermiittinen ja positiivisesti definiitti, sille voidaan laskea Choleskyn hajotelma $B = L_B L_B^H$, missä L_B on alakolmiomatriisi. Alkuperäinen tehtävä voidaan tällöin saattaa muotoon

$$C\mathbf{y} = \lambda\mathbf{y}, \tag{9.10}$$

missä

$$C = L_B^{-1} A L_B^{-H} \quad \text{ja} \quad \mathbf{y} = L_B^H \mathbf{x}.$$

Tällä reduktiolla on kuitenkin haittapuolensa. Vaikka matriisit A ja B olisivat harvoja, matriisi C ei välttämättä ole harva. Menetelmää ei myöskään voi soveltaa, jos B on singulaarinen.

Toinen tapa palauttaa yleistetty ominaisarvotehtävä tavalliseksi ominaisarvotehtäväksi on tarkastella tehtävää

$$B^{-1} A\mathbf{x} = \lambda\mathbf{x}, \tag{9.11}$$

joka on yleensä ei-hermiittinen tehtävä. Myös tämä tapa toimii vain, jos matriisi B on säännöllinen. Samoin menetetään matriisien B ja A mahdollisesta harvuudesta saatava hyöty.

Harvuus voidaan toisaalta säilyttää soveltamalla esimerkiksi Lanczosin algoritmia muunnettuun ominaisarvotehtävään (9.10). Lanczosin menetelmässä kerroinmatriisia C tarvitaan ainoastaan matriisi-vektorikertolaskussa, jolloin kertolasku $\mathbf{d} = C\mathbf{x}$ voidaan palauttaa yläkolmioyhtälöryhmän ratkaisuksi ($L_B^H\mathbf{z} = \mathbf{x}$), matriisi-vektorituloksi ($\mathbf{y} = A\mathbf{z}$) ja alakolmioyhtälöryhmän ratkaisuksi ($L_B\mathbf{d} = \mathbf{y}$). Muotoon (9.11) muunnetussa tehtävässä kertolasku $\mathbf{y} = B^{-1}A\mathbf{x}$ korvataan kertolaskulla $\mathbf{u} = A\mathbf{x}$ ja yhtälöryhmän ratkaisulla $B\mathbf{y} = \mathbf{u}$.

9.6.2 Menetelmiä yleistetyn tehtävän ratkaisemiseksi

Eräät ominaisarvojen ratkaisualgoritmit voidaan yleistää muotoon, joka toimii myös yleistetylle ominaisarvotehtävälle. Siirretty potenssimenetelmä saa muodon

$$B\mathbf{y}^k = (A - \sigma_k B)\mathbf{x}^{k-1},$$
$$\mathbf{x}^k = \frac{\mathbf{y}^k}{\|\mathbf{y}^k\|},$$

ja siirretty käänteinen potenssimenetelmä on yleisen tehtävän tapauksessa

$$(A - \sigma_k B)\mathbf{y}^k = B\mathbf{x}^{k-1},$$
$$\mathbf{x}^k = \frac{\mathbf{y}^k}{\|\mathbf{y}^k\|}.$$

Kummassakin algoritmissa joudutaan joka iteraatiokierroksella ratkaisemaan lineaarinen yhtälöryhmä. Vastaavasti voidaan muotoilla aliavaruusmenetelmä yleistetylle ominaisarvotehtävälle.

Täysien matriisien yleistetyn ominaisarvotehtävän kaikki ominaisarvot saadaan *QZ-algoritmilla* [GvL96]. Se laskee ortogonaaliset matriisit Q ja Z siten, että $Q^HAZ = T$ ja $Q^HBZ = S$ ovat yläkolmiomatriiseja. Yleistetyt ominaisarvot voidaan lukea näiden matriisien lävistäjäalkioista $\lambda_i = t_{ii}/s_{ii}$.

Esimerkki 9.6.2 Olkoot matriisit A ja B esimerkiksi seuraavat:

$$A = \begin{pmatrix} 229 & 163 \\ 163 & 116 \end{pmatrix}, \qquad B = \begin{pmatrix} 81 & 59 \\ 59 & 43 \end{pmatrix}.$$

QZ-algoritmi palauttaa ortogonaalimatriisit

$$Q = \begin{pmatrix} 0.8137 & -0.5812 \\ 0.5812 & 0.8137 \end{pmatrix}, \qquad Z = \begin{pmatrix} 0.6 & 0.8 \\ -0.8 & 0.6 \end{pmatrix}.$$

Muodostamme yläkolmiomatriisit

$$Q^TAZ = \begin{pmatrix} 8.6023 & 344.9067 \\ 0 & -0.5812 \end{pmatrix}, \qquad Q^TBZ = \begin{pmatrix} 1.7205 & 123.9665 \\ 0 & 1.1625 \end{pmatrix},$$

mistä näemme, että ominaisarvot ovat $8.6023/1.7205 \approx 5$ ja $-0.5812/1.1625 \approx -0.5$.

9.7 Yhteenveto ja suosituksia

Matriisin kaikkien tai useimpien ominaisarvojen laskemiseen paras valinta on divide-and-conquer tai QR-menetelmä sopivalla siirtomenettelyllä.

Muutaman ominaisarvon laskemiseen tai spektrin sijainnin arvioimiseen kannattaa käyttää Arnoldin tai Lanczosin menetelmää, varsinkin jos matriisi on suuri ja harva. Myös aliavaruusmenetelmää voi soveltaa, jos algoritmin tehokkuudelle ei aseteta suuria vaatimuksia. Jos ominaisarvojen likiarvot ovat tiedossa, shift-invert-menettelyllä päästään huomattavasti nopeampaan suppenemiseen. Pienissä tapauksissa myös puolitushakumenetelmä voi olla käyttökelpoinen.

9.8 Ohjelmistokatsaus

Matlabissa ja Mathematicassa on helppokäyttöisiä rutiineja ominaisarvojen ja -vektoreiden hakemiseen. Näitä tulisi kuitenkin käyttää vain pienissä tehtävissä.

Aliohjelmakirjastot NAG, IMSL ja LAPACK sisältävät QR- ja puolitushakumenetelmää käyttäviä rutiineja sekä täysille matriiseille että nauhamatriiseille. NAGissa ja IMSL:ssä on toteutettu myös QZ-algoritmi yleistetyille tehtäville.

Divide-and-conquer on toteutettu LAPACKissa rutiiniperheissä `ssyevd` ja `sstevd`.

ARPACK on helppokäyttöinen Arnoldin menetelmää käyttävä kirjasto, joka sopii erityisesti suurien harvojen tehtävien ratkaisemiseen. ARPACKilla voi ratkaista myös yleistettyjä tehtäviä. Kirjastosta on olemassa myös rinnakkaistettu versio P_ARPACK. ARPACKin ja P_ARPACKin käyttöohjeineen voi kopioida itselleen osoitteesta `http://www.caam.rice.edu/software/ARPACK`.

Harwell-kirjastossa on suuri valikoima menetelmiä: käänteinen siirretty potenssimenetelmä, QR-menetelmä, aliavaruusmenetelmä, Jacobin menetelmä sekä Lanczosin ja Arnoldin menetelmät. Kirjaston käyttöä hankaloittaa käyttöoppaan vaikeaselkoisuus.

Kaikki edellä mainitut ohjelmistot tai kirjastot sisältävät rutiineja myös singulaariarvohajotelmien laskemiseen.

Netlibissä on saatavilla aliohjelmakokoelmat Lanz, Laso ja Lanczos, jotka kaikki pohjautuvat Lanczosin menetelmään. Myös muissa Netlibin kokoelmissa on ominaisarvotehtäviä ratkaisevia rutiineja.

9.9 Kirjallisuutta ja lisätietoja

Kirjassa *Applied Numerical Linear Algebra* [Dem97] on selkeästi kuvattu tärkeimmät ominaisarvotehtävien ratkaisumenetelmät. Mukana on havainnollisia vertailuja, viittauksia ohjelmistoihin ja suosituksia sopivan menetelmän valintaan.

Kirja *Matrix Computations* [GvL96] esittelee ja johtaa aliavaruusmenetelmän ja QR-menetelmän erittäin havainnollisesti lähtien potenssimenetelmästä. Lanczosin menetelmä johdetaan siten, että sen suppenemisominaisuudet on helppo ymmärtää. Kirjassa käsitellään myös paljon käytännön laskennan kannalta tärkeitä kysymyksiä, kuten tarvittavien laskutoimitusten lukumäärää, laskennan stabiilisuutta, pyöristysvirheiden vaikutusta ja niin edelleen.

Symmetristen tehtävien perusteos *The Symmetric Eigenvalue Problem* [Par80] on hyvin yksityiskohtainen ja kattava.

Suuriin tehtäviin soveltuvia menetelmiä käsitellään kirjassa *Numerical Methods for Large Eigenvalue Problems* [Saa92], joka painottuu Krylovin menetelmiin.

10 Optimointi

Optimointitehtäviä löytyy kaikilta tieteen ja tekniikan aloilta. Optimoinnissa pyrimme rajallisten resurssien tehokkaaseen käyttöön. Esimerkiksi fysiikan minimienergiaperiaate tuottaa monenlaisia optimointitehtäviä. Toisaalta insinöörit kehittävät tehokkaampia paperikoneita tai taloudellisempia moottoreita. Toisen tyyppisen esimerkin tarjoaa kokeellinen tutkimus, jossa malli voidaan sovittaa mittauksista saatuun dataan käyttämällä optimointimenetelmiä.

10.1 Optimointitehtävän muodostaminen ja ratkaisu

Optimointitehtävä

$$\operatorname*{minimi}_{x\in\mathbb{R}} f(x) = \arg\min_{x\in\mathbb{R}} f(x) = x^2 + 300/(\pi x), \quad \text{kun } x \geq 0,$$

on tyyppiesimerkki yhden muuttujan optimointitehtävästä. Tehtävässä tulee löytää funktion f minimiarvo, kun kohdemuuttuja on $x \geq 0$. Minimikohta löytyy pisteestä $x^* \approx 3.6$ ja minimiarvo on $f(x^*) \approx 39.5$.

Optimoinnissa voimme etsiä funktion minimiä tai maksimia. Yleensä optimikohtaa etsitään tietyssä alueessa, jota kutsutaan *käyväksi alueeksi*. Maksimointi- ja minimointitehtävät ratkeavat samoilla menetelmillä, koska voimme muuttaa maksimointitehtävän minimointitehtäväksi:

$$\text{maksimi } f(\mathbf{x}) = -\left(\text{minimi } -f(\mathbf{x})\right).$$

Käsittelemme siten vain funktion minimoimista.

Minimoitava *kohdefunktio* $f : \mathcal{F} \mapsto \mathbb{R}$ on kuvaus käyvältä alueelta $\mathcal{F} \subset \mathbb{R}^n$ reaaliluvuille $\mathbb{R}$. Minimoinnissa etsimme pistettä $\mathbf{x}^* \in \mathcal{F}$, jolle pätee $f(\mathbf{x}^*) \leq f(\mathbf{y})$ kaikilla $\mathbf{y} \in \mathcal{F}$.

Esimerkki 10.1.1 Valmistettavanamme on lieriön muotoinen metallitölkki, johon tulisi kulua mahdollisimman vähän materiaalia. Jos tölkin säde on r ja korkeus on h, on sen tilavuus $\pi r^2 h$ ja pinta-ala $A = 2\pi r^2 + 2\pi r h$.

Merkitsemme $x_1 = r$ ja $x_2 = h$. Näin saamme kahden reaaliarvoisen muuttujan x_1 ja x_2 optimointitehtävän

$$\begin{aligned} &\underset{\mathbf{x}}{\text{minimi}}\ 2\pi(x_1^2 + x_1x_2), \\ &\qquad \text{kun } \pi x_1^2 x_2 \geq V \text{ ja } \mathbf{x} \geq \mathbf{0}. \end{aligned} \tag{10.1}$$

Tölkiltä vaadittu tilavuus V on tehtäväkohtainen vakio. Tehtävän kohdefunktio on $f(\mathbf{x}) = 2\pi(x_1^2 + x_1x_2)$ ja käypä alue $\mathcal{F} = \{\mathbf{x} \in \mathbb{R}^2 \mid \pi x_1^2 x_2 \geq V \text{ ja } \mathbf{x} \geq \mathbf{0}\}$.

10.2 Optimointitehtävien tyyppejä

Ei ole olemassa ratkaisualgoritmia, joka soveltuisi minkä tahansa optimointitehtävän ratkaisemiseen. Tässä luvussa rajoitumme tavanomaisimpiin optimointitehtäviin.

Optimointitehtävät voidaan esittää muodossa

$$\underset{\mathbf{x}}{\text{minimi}}\ f(\mathbf{x}),\ \text{kun } \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \text{ ja } \mathbf{h}(\mathbf{x}) = \mathbf{0}. \tag{10.2}$$

Valitettavasti esitystapa vaihtelee ratkaisuohjelmistosta riippuen. *Epäyhtälörajoite* $\mathbf{g}$ on funktio $\mathbb{R}^n \mapsto \mathbb{R}^p$. *Yhtälörajoite* $\mathbf{h}$ on puolestaan funktio $\mathbb{R}^n \mapsto \mathbb{R}^q$. Usein optimointitehtävään liittyy lisäksi positiivisuusehto $\mathbf{x} \geq \mathbf{0}$. Rajoitteiden leikkausjoukko on käypä alue $\mathcal{F}$. Monissa ratkaisumenetelmissä oletetaan, että funktiot f, $\mathbf{g}$ ja $\mathbf{h}$ ovat jatkuvia ja jatkuvasti derivoituvia eli *sileitä*.

Jos funktiot f, $\mathbf{g}$ ja $\mathbf{h}$ ovat lineaarisia, on kyseessä *lineaarinen optimointitehtävä*. Jos jokin näistä funktioista on epälineaarinen, on kyseessä *epälineaarinen optimointitehtävä*. Jos tehtävälle ei ole annettu rajoitteita eli käypä alue on $\mathcal{F} = \mathbb{R}^n$, kyseessä on *rajoitteeton optimointitehtävä*. Muussa tapauksessa on kyse *rajoitteellisesta optimointitehtävästä*.

Keskitymme tässä luvussa pääasiassa epälineaarisiin optimointitehtäviin, joissa on yksi tai useampi optimoitava muuttuja. Emme käsittele *diskreettejä optimointitehtäviä*, joissa muuttujat saavat kokonaislukuarvoja. Tällaiset tehtävät ovat usein hyvin vaikeita ja niiden ratkaisuun tarvitaan tehtäväkohtaisia erikoismenetelmiä. Emme käsittele myöskään muita erikoistyyppisiä tehtäviä kuten *monitavoiteoptimointia* (kohdefunktioita enemmän kuin yksi) tai *epäsileää optimointia* (funktiot eivät ole kaikkialla derivoituvia). Jatkossa oletamme kohdefunktion ja rajoitefunktioiden olevan jatkuvasti derivoituvia eli sileitä.

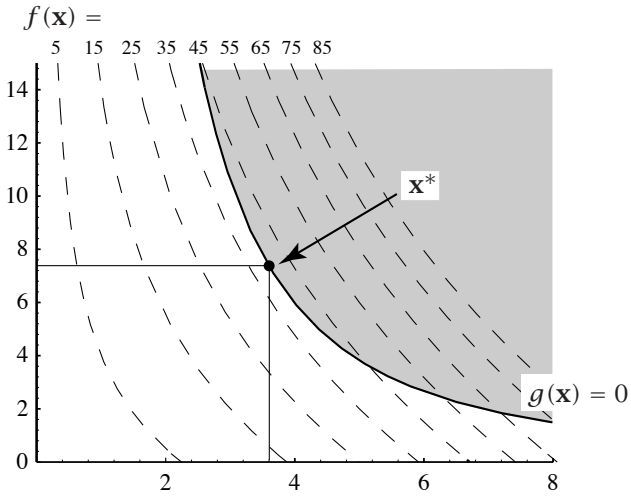
Esimerkki 10.2.1 Voimme esittää esimerkin 10.1.1 optimointitehtävän muodossa

$$\begin{aligned} &\underset{\mathbf{x}}{\text{minimi}}\ f(\mathbf{x}) = x_1^2 + x_1x_2, \\ &\qquad \text{kun } g(\mathbf{x}) = -x_1^2 x_2 + V/\pi \leq 0 \text{ ja } \mathbf{x} \geq \mathbf{0}. \end{aligned} \tag{10.3}$$

Tässä kohdefunktio f on kuvaus $\mathbb{R}^2 \mapsto \mathbb{R}$ ja rajoitefunktio g on kuvaus $\mathbb{R} \mapsto \mathbb{R}$. Kohdefunktiosta f on jätetty kerroin 2π pois, sillä se ei vaikuta optimikohdan sijaintiin.

Tehtävän kohdefunktio f on epälineaarinen, joten ratkaisuun on käytettävä epälineaarisille tehtäville sopivaa algoritmia. Myös rajoitefunktio g on epälineaarinen.

Tehtävän geometriaa on havainnollistettu kuvassa 10.1. Tässä on asetettu vakion V arvoksi 300 (yksikkö on cm^3). Kuvaan on merkitty minimipisteen $\mathbf{x}^* \approx (3.62, 7.26)^T$ sijainti. Funktion minimiarvo on $f(\mathbf{x}^*) \approx 39.5$. Saamme tölkin pinta-alaksi $2\pi f(\mathbf{x}^*) \approx 248\,\text{cm}^2$.



Kuva 10.1: *Esimerkissä 10.1.1 käsitellyn epälineaarisen minimointitehtävän geometrinen havainnollistus. Kohdefunktio on* $f(\mathbf{x}) = x_1^2 + x_1 x_2$ *ja rajoite* $g(\mathbf{x}) = -x_1^2 x_2 + 300/\pi \leq 0$. *Muuttujat* x_1 *ja* x_2 *ovat positiivisia.*

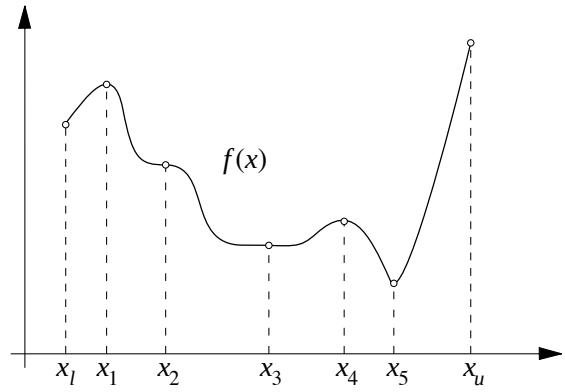
Epälineaarisella optimointitehtävällä voi olla mm. seuraavia ratkaisua hankaloittavia ominaisuuksia:

- Tehtävällä ei ole yksikäsitteistä optimiratkaisua $\mathbf{x}^*$, vaan useampi piste, jotka ovat globaaleja minimejä.
- Funktiolla f on useita lokaaleja eli paikallisia minimejä.
- Eri muuttujilla on eri skaalat. Esimerkiksi muuttuja x_1 kuuluu välille $[0, 10^{-10}]$ ja muuttuja x_2 kuuluu välille $[1, 10^6]$.
- Funktio f on raskas laskea: funktion arvo voi olla peräisin esimerkiksi prosessisimulaatiosta tai differentiaaliyhtälösysteemistä.
- Funktion f derivaatat eivät ole saatavilla: niitä ei esimerkiksi voida johtaa analyyttisesti tai f on ”musta laatikko”, jonka rakennetta ei tunneta. Tällöin ei voida käyttää gradienttipohjaisia ratkaisumenetelmiä.
- Hyvän alkuarvauksen löytäminen tehtävän ratkaisemiseksi on vaikeaa.

10.3 Lokaalit ja globaalit minimit

Funktiolla f on *lokaali minimi* käyvässä pisteessä $\mathbf{x}^*$, jos kaikille käypään alueeseen $\mathcal{F}$ kuuluville pisteille $\mathbf{x}$ pisteen $\mathbf{x}^*$ ympäristössä pätee $f(\mathbf{x}^*) \leq f(\mathbf{x})$. *Globaali minimi* $\mathbf{x}^*$ on puolestaan pienin lokaaleista minimeistä eli $f(\mathbf{x}^*) \leq f(\mathbf{x})$ kaikilla $\mathbf{x} \in \mathcal{F} \subset \mathbb{R}^n$.

Kuvassa 10.2 on esimerkki yhden muuttujan funktion käyttäytymisestä. Funktio on määritelty välillä $[x_l, x_u]$, ja funktiolla on useita minimejä ja maksimeja. Kohdassa x_5 saavutetaan kaikkein pienin arvo eli globaali minimi välillä $[x_l, x_u]$. Globaali maksimi saavutetaan reunapisteessä x_u. Reunapiste x_l on funktion lokaali minimi.



Kuva 10.2: *Erään funktion käyttäytyminen välillä* $[x_l, x_u]$.

Epälineaaristen optimointitehtävien ratkaisualgoritmit löytävät lokaalin minimin, eikä globaalisuutta voi osoittaa kuin erikoistapauksissa. Yleensä ratkaisualgoritmit löytävät lähellä alkuarvausta $\mathbf{x}^1$ olevan minimipisteen käyttäen iteratiivista menetelmää

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \lambda_k \mathbf{p}^k, \ \lambda_k > 0, \ k = 1, 2, \ldots. \tag{10.4}$$

Kerroin λ_k määrää suuntaan $\mathbf{p}^k$ otetun askeleen pituuden. Kertoimen λ_k arvo määritetään esimerkiksi *viivahaulla*. Iteraatiota toistetaan kunnes esimerkiksi $\|\mathbf{x}^{k+1} - \mathbf{x}^k\| \to 0$.

Usein optimoinnissa riittää paikallisen minimin löytäminen. Globaalia optimikohtaa voi etsiä käyttämällä useita toisistaan poikeavia alkuarvauksia. Tosin tämä lähestymistapa voi olla tehoton, jos käytössä ei ole menetelmää hyvien alkuarvausten generointiin. On olemassa myös globaaliin optimointiin kehitettyjä algoritmeja. Nämä voivat olla tehokkaita kullekin menetelmälle sopivissa tehtävissä.

Globaali optimointi on huomattavasti vaikeampaa kuin yksittäisen lokaalin minimipisteen hakeminen. Aluksi käsittelemme vain lokaalien minimipisteiden hakemista. Kappale 10.11 (sivu 366) käsittelee globaalia optimointia.

10.4 Globaali ja lokaali suppeneminen

Useimmat optimointitehtävien ratkaisumenetelmät etsivät lokaaleja minimejä. Keskeinen optimointialgoritmin ominaisuus on se, kuinka hyvä alkuarvaus tarvitaan lokaalin minimin löytämiseen.

Määritelmä 10.4.1 (Globaali ja lokaali suppeneminen) Minimointialgoritmin *globaalilla suppenemisella* tarkoitetaan sitä, että lähdettäessä liikkeelle mistä tahansa pisteestä löydetään lokaali minimi. *Lokaalilla suppenemisella* tarkoitetaan sitä, että lähdettäessä liikkeelle kyllin läheltä minimikohtaa algoritmi suppenee kohti kyseistä minimiä.

Useimmat optimointialgoritmit suppenevat lokaalisti. Jos tehtävällä on sopivia erityispiirteitä (esimerkiksi seuraavassa käsitelty konveksisuus), voi suppeneminen olla myös globaalia.

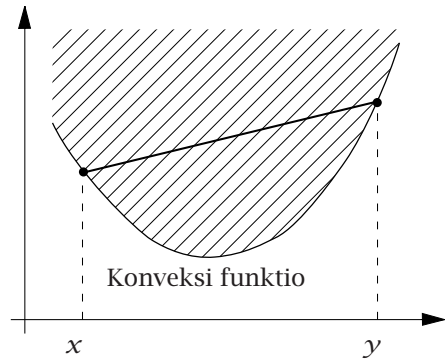
10.5 Konveksit optimointitehtävät

Konvekseja optimointitehtäviä löytyy monilta erilaisilta alueilta. Lineaarinen optimointitehtävä (kappale 10.10.1 sivulla 356) on tavanomaisin esimerkki konveksista tehtävästä. Aluksi määrittelemme konveksin funktion ja konveksin joukon, jotta voimme määritellä konveksin optimointitehtävän.

Määritelmä 10.5.1 (Konveksi funktio) Funktion $f\mathbb{R}^n \mapsto \mathbb{R}$ sanotaan olevan *konveksi funktio*, jos kaikilla $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{y} \in \mathbb{R}^n$ ja $\mathbf{x} \neq \mathbf{y}$ pätee

$$f(a\,\mathbf{x} + (1-a)\,\mathbf{y}) \leq a f(\mathbf{x}) + (1-a) f(\mathbf{y}), \tag{10.5}$$

missä $0 < a < 1$. Funktio on *aidosti konveksi*, jos epäyhtälössä (10.5) pätee aito erisuuruus $<$.

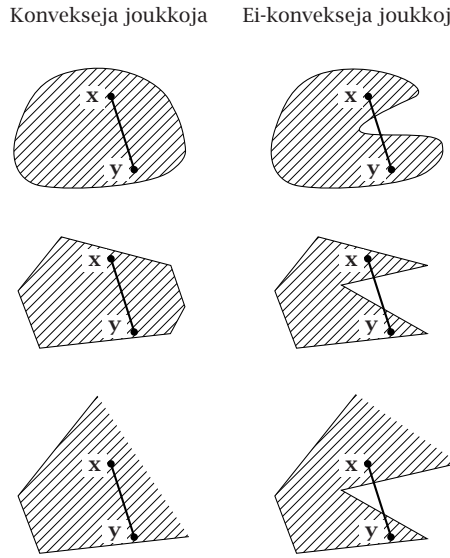


Kuva 10.3: *Aidosti konveksi funktio.*

Määritelmä 10.5.2 (Konveksi joukko) Joukko $C \subset \mathbb{R}^n$ on *konveksi* (convex), jos kaikilla $\mathbf{x}, \mathbf{y} \in C$ pätee

$$a\,\mathbf{x} + (1-a)\,\mathbf{y} \in C, \quad \text{kun } 0 < a < 1. \tag{10.6}$$

Konveksisuus merkitsee, että joukon kaikkien pisteiden väliset janat sisältyvät joukkooon. Esimerkiksi lineaaristen rajoitteiden määräämä alue $A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \in \mathbb{R}^n$ on konveksi. Konveksisuutta eli "kuperuutta" on tutkittu kuvassa 10.4 pisteitä $\mathbf{x}$ ja $\mathbf{y}$ yhdistävän janan $\{\mathbf{x} + a(\mathbf{y} - \mathbf{x}),\ 0 < a < 1\}$ avulla.



Kuva 10.4: *Konvekseja ja ei-konvekseja tason joukkoja.*

Konveksissa optimointitehtävässä kohdefunktio on konveksi ja käypä alue on konveksi. Seuraava lause on keskeinen konveksien optimointitehtävien ratkaisussa.

Lause 10.5.1 Jos piste $\mathbf{x}^*$ on konveksin optimointitehtävän käypään alueeseen $\mathcal{F}$ sisältyvä lokaali minimi, on se myös globaali käypä minimi.

Erityisen tärkeitä konveksit tehtävät ovat siksi, että niille voidaan kehittää algoritmeja, jotka joko löytävät ratkaisun tai sitten osoittavat, ettei ratkaisua ole olemassa.

10.6 Ehtoja minimikohdan sijainnille

Useimmat epälineaaristen optimointitehtävien ratkaisualgoritmit kykenevät käsittelemään vain sileitä eli jatkuvasti derivoituvia kohdefunktioita ja rajoitteita. Tärkeimmät menetelmät perustuvat Taylorin sarjakehitelmän käyttöön. Reaalimuuttujan funktion tapauksessa Taylorin kehitelmä on muotoa

$$f(x + d) = f(x) + f'(x)d + \frac{f''(x)}{2}d^2 + \mathcal{O}(d^3). \tag{10.7}$$

Jos tehtävällä ei ole rajoitteita, saamme minimikohdalle ehdon $f'(x^*) = 0$. Täten optimoinnilla on yhteys epälineaaristen yhtälöiden ratkaisemiseen.

Myös menetelmien nimet ovat osittain samoja, vaikkakin algoritmit ovat erilaisia: epälineaaristen yhtälöiden ratkaisussa etsitään funktion nollakohtaa, kun taas optimoinnissa etsitään derivaatan nollakohtaa.

Derivoituvan skalaarifunktion optimaalisuus voidaan siis (periaatteessa) selvittää hakemalla funktion derivaatan nollakohdat ja toisen derivaatan merkki. Useammassa ulottuvuudessa tämä ei ole enää yhtä yksinkertaista, sillä funktion arvot voivat kasvaa joihinkin suuntiin mentäessä ja toisissa suunnissa arvot voivat puolestaan pienentyä tai pysyä samoina. Täten tarvitaan monipuolisempia välineitä optimaalisuuden havaitsemiseksi.

Usean muuttujan derivoituvaa funktiota $f : \mathbb{R}^n \mapsto \mathbb{R}$ voidaan arvioida Taylorin kehitelmällä:

$$\begin{aligned} f(\mathbf{x}^* + \mathbf{d}) &= f(\mathbf{x}^*) + \langle \mathbf{d}, \nabla f(\mathbf{x}^*) \rangle + \tfrac{1}{2} \langle \mathbf{d}, H(\mathbf{z})\, \mathbf{d} \rangle \\ &\approx f(\mathbf{x}^*) + \langle \mathbf{d}, \nabla f(\mathbf{x}^*) \rangle + \tfrac{1}{2} \langle \mathbf{d}, H(\mathbf{x}^*)\, \mathbf{d} \rangle. \end{aligned} \tag{10.8}$$

Tässä pisteet $\mathbf{x}^*$ ja $\mathbf{d}$ kuuluvat avaruuteen $\mathbb{R}^n$ ja piste $\mathbf{z}$ on pisteiden $\mathbf{x}^*$ ja $\mathbf{x}^* + \mathbf{d}$ välissä. Sarjakehitelmässä tarvitaan funktion gradienttivektori

$$\nabla f(\mathbf{x}) = \begin{pmatrix} \dfrac{\partial f(\mathbf{x})}{\partial x_1} \\ \vdots \\ \dfrac{\partial f(\mathbf{x})}{\partial x_n} \end{pmatrix} \tag{10.9}$$

ja Hessen matriisi

$$H(\mathbf{x}) = \nabla \nabla^T f(\mathbf{x}) = \begin{pmatrix} \dfrac{\partial^2 f(\mathbf{x})}{\partial^2 x_1} & \cdots & \dfrac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_n} & \cdots & \dfrac{\partial^2 f(\mathbf{x})}{\partial^2 x_n} \end{pmatrix}. \tag{10.10}$$

Esimerkki 10.6.1 Laskemme Taylorin sarjakehitelmän pisteessä $\mathbf{x}^* = (-1, 1)^T$ Rosenbrockin funktiolle $f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$. Funktion gradienttivektori on

$$\nabla f(\mathbf{x}) = \begin{pmatrix} -400x_1(x_2 - x_1^2) + 2(x_1 - 1) \\ 200(x_2 - x_1^2) \end{pmatrix}$$

ja funktion Hessen matriisi on

$$H(\mathbf{x}) = \begin{pmatrix} 1200x_1^2 - 400x_2 + 2 & -400x_1 \\ -400x_1 & 200 \end{pmatrix}.$$

Saamme siten arvion

$$\begin{aligned} f(\mathbf{x}^* + \mathbf{d}) &\approx f(\mathbf{x}^*) + \langle \mathbf{d}, \nabla f(\mathbf{x}^*) \rangle + \tfrac{1}{2} \langle \mathbf{d}, H(\mathbf{x}^*)\, \mathbf{d} \rangle \\ &= 4 + \mathbf{d}^T \begin{pmatrix} -4 \\ 0 \end{pmatrix} + \tfrac{1}{2}\, \mathbf{d}^T \begin{pmatrix} 802 & 400 \\ 400 & 200 \end{pmatrix} \mathbf{d} \\ &= 802 d_1^2 + 800 d_1 d_2 + 200 d_2^2 - 4 d_1 + 4. \end{aligned}$$

Funktion f *kriittisiksi pisteiksi* $\mathbf{x}^c$ sanotaan yhtälön $\nabla f(\mathbf{x}) = \mathbf{0}$ ratkaisuja. Usean muuttujan funktion kriittinen piste vastaa yhden muuttujan funktion derivaatan nollakohtaa $f'(x) = 0$. Yhden muuttujan funktion tapauksessa tutkimme toisen derivaatan merkkiä minimi- tai maksimikohdan löytämiseksi. Kriittisen pisteen optimaalisuutta voi tutkia funktion toisista derivaatoista muodostetun Hessen matriisin $H(\mathbf{x})$ avulla. Tosin matriisin positiivisuutta tai negatiivisuutta ei voi tutkia kuten yhden muuttujan tapauksessa derivaatan merkkiä. Sen sijaan tutkimme matriisin $H(\mathbf{x}^c)$ definiittisyyttä, jonka voi selvittää mm. laskemalla Choleskyn hajotelman tai LDL^T-hajotelman.

Jos funktion f Hessen matriisi $H(\mathbf{x}^c)$ on positiivisesti definiitti, pätee

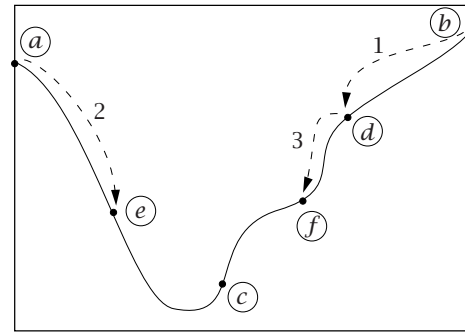
$$\langle \mathbf{d}, H(\mathbf{x}^c)\,\mathbf{d} \rangle > 0 \quad \text{kaikilla } \mathbf{d} \in \mathbb{R}^n, \mathbf{d} \neq \mathbf{0}. \tag{10.11}$$

Tällöin kriittinen piste $\mathbf{x}^c$ on funktion f lokaali minimi. Jos $H(\mathbf{x})$ on positiivisesti semidefiniitti kaikilla $\mathbf{x} \in \mathbb{R}^n$, piste $\mathbf{x}^c$ on funktion $f(\mathbf{x})$ globaali minimi. Jos matriisi on negatiivisesti definiitti, on kyseessä maksimikohta. Muussa tapauksessa sanotaan, että piste $\mathbf{x}^c$ on funktion f *satulapiste*.

10.7 Yksiulotteinen optimointi

Olkoon minimoitavana jatkuva funktio $f : \mathbb{R} \mapsto \mathbb{R}$. Paikallisen minimin sisältävä väli $[a, b]$ on tiedossa, kun on löydetty kolme pistettä $a < c < b$ siten, että $f(c) < f(a)$ ja $f(c) < f(b)$. Kuva 10.5 esittää minimoinnin periaatteen.

Kuva 10.5: *Yhden muuttujan funktion minimointi.*



Aluksi minimikohta on välillä $[a, b]$. Tämän jälkeen laskemme funktion arvon pisteessä d, jolla voimme korvata pisteen b jne. Kullakin askeleella valitsemme jatkoon keskipisteen, jossa funktion arvo on pienempi kuin sitä reunustavissa pisteissä. Iteraation 3 jälkeen minimi on pisteiden e ja f välissä.

Suositeltavat minimointimenetelmät ovat hybridimenetelmiä, joissa käytetään jotakin varmaa mutta hidasta menetelmää varmistamaan nopean mutta mahdollisesti epäluotettavan menetelmän suppeneminen. Tutustumme

siksi aluksi kultaisen leikkauksen menetelmään. Tämän jälkeen esittelemme interpolointiin perustuvan menetelmän sekä nopean mutta epävarman Newtonin menetelmän. Käytännön ratkaisumenetelmät ovat yleensä yhdistelmiä tässä esitellyistä menetelmistä.

10.7.1 Kultaisen leikkauksen menetelmä

Oletamme seuraavassa, että minimoitava funktio on jatkuvasti derivoituva. *Kultaisen leikkauksen menetelmässä* eli *jatkuvasuhdehaussa* funktion oletetaan olevan lisäksi *unimodaalinen*.

Määritelmä 10.7.1 (Unimodaalinen funktio) Funktio f on unimodaalinen välillä $[a,b]$, jos jollekin pisteelle $x^* \in [a,b]$ pätee, että funktio f on aidosti vähenevä välillä $[a,x^*]$ ja aidosti kasvava välillä $[x^*,a]$.

Kultaisen leikkauksen menetelmässä tarvitsee kullakin iteraatiolla laskea funktion arvo vain yhdessä uudessa pisteessä. Väli $[a,b]$ jaetaan osaväleihin $[a,x]$ ja $[x,b]$ joiden suhteelliset koot ovat $r = (\sqrt{5}-1)/2 \approx 0.61803$ ja $(1-r) = (3-\sqrt{5})/2 \approx 0.38197$. (Menetelmän nimi tulee tästä lukuarvosta r.) Hakuvälin jakaminen kultaisen leikkauksen suhteessa mahdollistaa laskettujen funktion arvojen hyödyntämisen seuraavilla iteraatiokierroksilla.

Unimodaalisuuden takia funktiolla f on vain yksi minimikohta x^* välillä $[a,b]$. Jokaisella iteraatiokierroksella voidaan toinen alkuperäisistä reunapisteistä korvata uudella pisteellä x.

Toimivaan kultaisen leikkauksen menetelmään kuuluu sopivan välin $[a,b]$ löytäminen iteraation aloittamiseksi. Periaatteessa riittää lähteminen jostain pisteestä funktiota pienentävään suuntaan, kunnes funktion arvot taas kasvavat. Otettujen askelten pituudet saavat olla suuriakin.

Olkoon $\gamma = (3-\sqrt{5})/2 \approx 0.38197$. Oletamme, että minimikohta löytyy väliltä $[a_1,b_1]$. Kultaisen leikkauksen menetelmä toimii seuraavasti:

Algoritmi 10.7.1 (Kultaisen leikkauksen menetelmä)

$k = 0$
$c_1 = a_1 + \gamma(b_1 - a_1), \quad f_c = f(c_1)$
$d_1 = b_1 - \gamma(b_1 - a_1), \quad f_d = f(d_1)$
repeat
  $k = k + 1$
  if $f_c < f_d$
    $[a_{k+1}, d_{k+1}, b_{k+1}] = [a_k, c_k, d_k]$
    $c_{k+1} = a_{k+1} + \gamma(b_{k+1} - a_{k+1})$
    $f_d = f_c$
    $f_c = f(c_{k+1})$
  else
    $[b_{k+1}, a_{k+1}, c_{k+1}] = [b_k, c_k, d_k]$
    $d_{k+1} = b_{k+1} - \gamma(b_{k+1} - a_{k+1})$
    $f_c = f_d$
    $f_d = f(d_{k+1})$

end
until $b_k - a_k < \epsilon$

Esimerkki 10.7.1 Minimoimme kuvassa 10.6 esitetyn funktion $f(x) = x^4 + (x-1)^2$ kultaisen leikkauksen menetelmällä. Seuraavassa taulukossa on esitetty pisteet c_k ja d_k sekä funktion arvot näissä pisteissä:

k	c_k	$f(c_k)$	d_k	$f(d_k)$
1	−0.4721	2.2169	0.4721	0.3283
2	0.4721	0.3283	1.0557	1.2454
3	0.1115	0.7897	0.4721	0.3283
4	0.4721	0.3283	0.6950	0.3264
5	0.6950	0.3264	0.8328	0.5090
6	0.6099	0.2905	0.6950	0.3264
7	0.5573	0.2924	0.6099	0.2905
8	0.6099	0.2905	0.6424	0.2982
9	0.5898	0.2893	0.6099	0.2905
10	0.5774	0.2897	0.5898	0.2893
11	0.5898	0.2893	0.5975	0.2895
12	0.5851	0.2893	0.5898	0.2893
13	0.5898	0.2893	0.5927	0.2893
14	0.5880	0.2893	0.5898	0.2893
15	0.5898	0.2893	0.5909	0.2893
16	0.5891	0.2893	0.5898	0.2893

Jatkamalla iteraatioita löydämme minimin kohdasta $x^* \approx 0.5898$. Minimiarvo on $f(x^*) \approx 0.2893$.

Kultaisen leikkauksen menetelmä suppenee lineaarisesti. Olkoon x_k likimääräinen ratkaisu iteraatiolla k (esim. hakuvälin keskipiste). Siten

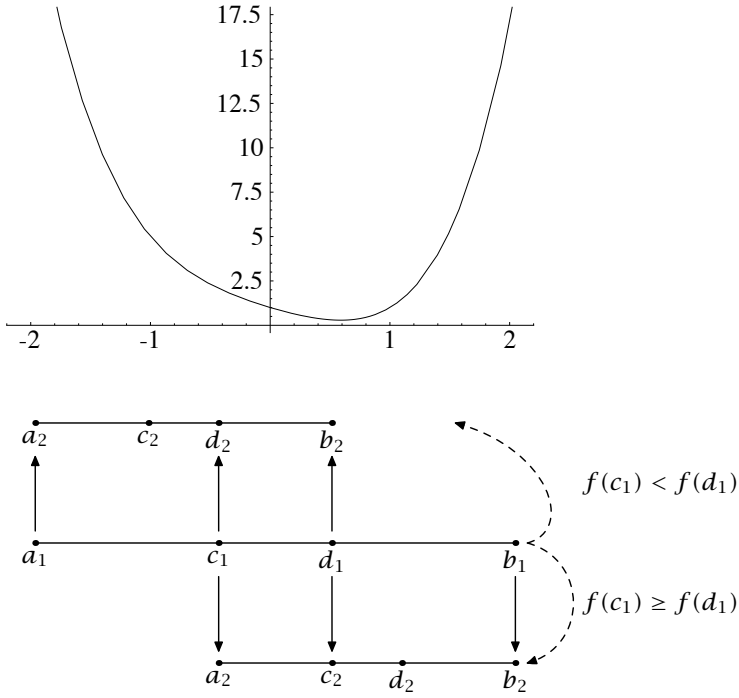
$$|x_{k+1} - x^*| \leq c\,|x_k - x^*|\,, \tag{10.12}$$

kun $k \geq N$, missä $N \geq 1$ on jokin annettu kokonaisluku. Vakion c arvo on $r \approx 0.618$. Jos haluamme lyhentää hakuvälin $[a, b]$ pituuden $b - a$ pienemmäksi kuin $\epsilon > 0$, tarvitaan k iteraatiota, missä $r^k(b-a) < \epsilon$.

10.7.2 Toistettu kvadraattinen interpolointi

Kultaisen leikkauksen menetelmässä käytämme funktion arvoja vain keskinäiseen vertailuun. *Toistetussa kvadraattisessa interpoloinnissa* laskemme funktion arvon kolmessa pisteessä ja sovitamme tulokseen toisen asteen polynomin. Jos tälle funktiolle löytyy hyväksyttävä minimikohta, sovitamme uuden kvadraattisen funktion niiden kolmen pisteen kautta, joissa on saavutettu funktion pienin tähänastinen arvo.

Kvadraattinen interpolointi on epäluotettavampi menetelmä kuin kultaisen leikkauksen menetelmä, sillä minimikohta ei välttämättä löydy annetun hakuvälin sisäpuolelta. Tämän vuoksi menetelmään yhdistetäänkin usein kultaisen leikkauksen menetelmä suppenemisen varmistamiseksi.



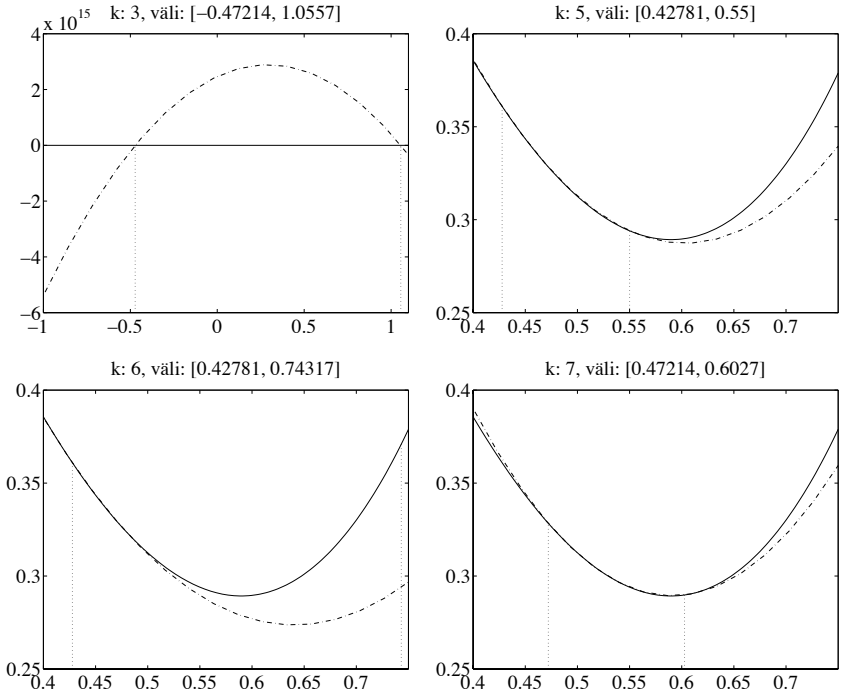
Kuva 10.6: *Kultaisen leikkauksen menetelmän käyttö funktion $f(x) = x^4 + (x-1)^2$ minimointiin. Alkuperäinen hakuväli on $[a_1, b_1]$. Menetelmä valitsee seuraavaksi hakuväliksi alemman välin, sillä $f(c_1) \geq f(d_1)$.*

Esimerkki 10.7.2 Minimoimme funktion $f(x) = x^4 + (x-1)^2$ käyttäen Matlabin rutiinin `fmin` toteutusta kvadraattisesta interpoloinnista. Kun alkuarvauksena on väli $[-2, 2]$, saamme iteraatiopisteet

k	x_k	$f(x_k)$
*1	−0.4721	2.2169
*2	0.4721	0.3283
*3	1.0557	1.2454
4	0.4278	0.3609
*5	0.5500	0.2940
6	0.7432	0.3710
7	0.6027	0.2898
8	0.5908	0.2893
9	0.5895	0.2893
10	0.5898	0.2893
11	0.5897	0.2893
12	0.5898	0.2893

Merkintä * tarkoittaa, että iteraatioaskeleella käytetään kultaisen leikkauksen menetelmää. Kuvassa 10.7 on esitetty kvadraattisen interpoloinnin tulokset iteraatioilla 3, 5, 6 ja 7. Interpolointi epäonnistuu iteraatiolla 3 (funktiolla ei

ole minimiä) ja iteraatiolla 5 (minimikohta on hakuvälin ulkopuolella). Saamme minimille likiarvon $x^* \approx 0.5898$.



Kuva 10.7: *Toistettu kvadraattinen interpolointi, iteraatiot 3, 5, 6 ja 7. Yhtenäinen viiva kuvaa minimoitavaa funktiota ja katkoviiva interpoloivaa polynomia.*

10.7.3 Newtonin menetelmä

Jos funktion ensimmäinen ja toinen derivaatta ovat tiedossa, voimme johtaa minimointialgoritmin Taylorin kehitelmän avulla:

$$f(x+d) \approx f(x) + f'(x)d + \frac{f''(x)}{2}d^2.$$

Haluamme löytää askeleen d, joka tuottaisi funktion minimikohdan. Derivoimalla funktion $f(x+d)$ askelpituuden d suhteen saamme ehdon $d = -f'(x)/f''(x)$. Nyt olemmekin johtaneet *Newtonin menetelmän*:

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}. \tag{10.13}$$

Newtonin menetelmä suppenee neliöllisesti lähellä minimiä, mutta jos alkuarvaus on kaukana minimistä, Newtonin menetelmä ei välttämättä toimi. Menetelmä voi myös supeta kohti funktion maksimia tai käännepistettä minimin sijaan. Tämän vuoksi suppeneminen kannattaa varmistaa käyttämällä jotain luotettavampaa menetelmää.

Esimerkki 10.7.3 Minimoimme funktion $f(x) = x^4 + (x-1)^2$ Newtonin menetelmällä. Derivaatat ovat

$$\begin{aligned} f'(x) &= 4x^3 + 2x - 2, \\ f''(x) &= 12x^2 + 2. \end{aligned}$$

Siten saamme Newtonin iteraatioksi

$$x_{k+1} = x_k - \frac{4x_k^3 + 2x_k - 2}{12x_k^2 + 2}.$$

Jos alkuarvauksena on $x_1 = 0.0$, saamme tulokseksi

k	x_k	$f(x_k)$
1	0.0000	1.0000
2	1.0000	1.0000
3	0.7143	0.3419
4	0.6052	0.2900
5	0.5900	0.2893
6	0.5898	0.2893

Tässä tapauksessa Newtonin menetelmä suppeni kuudella iteraatiolla.

Huomautus 10.7.1 Newtonin menetelmä funktion minimoimiseksi muistuttaa epälineaarisen yhtälön $f(x) = 0$ ratkaisumenetelmää

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}.$$

Tässä kuitenkin lasketaan funktion arvo ja derivaatta, kun minimoinnissa puolestaan lasketaan ensimmäinen ja toinen derivaatta. Minimoinnissa onkin kyse funktion $f'(x)$ nollakohdan hakemisesta.

10.8 Moniulotteinen optimointi

Käsittelemme aluksi rajoitteetonta moniulotteista optimointitehtävää

$$\operatorname*{minimi}_{\mathbf{x}} f(\mathbf{x}), \text{ kun } \mathbf{x} \in \mathbb{R}^n. \tag{10.14}$$

Tehtävälle löytyy tehokkaita ratkaisumenetelmiä, kun minimoitava kohdefunktio f on jatkuvasti derivoituva. Eräiden algoritmien voidaan jopa osoittaa suppenevan globaalisti, jos kohdefunktio f on konveksi. Toisaalta luotettavin menetelmä ei välttämättä ole tehokkain.

Suositeltavia moniulotteisten rajoitteettomien tehtävien ratkaisualgoritmeja ovat esimerkiksi seuraavat.

- *Newtonin menetelmä* on tehokas, jos derivaatat on helppo laskea. Menetelmä vaatii varmistuksia toimiakseen luotettavasti.

- *Sekanttimenetelmä BFGS-päivityksellä* on luotettava ja stabiili menetelmä, mutta isoissa tehtävissä liittogradienttimenetelmät tarvitsevat vähemmän muistia.
- *Liittogradienttimenetelmistä* tunnetuimmat ovat Fletcherin ja Reevesin menetelmä sekä Polakin ja Ribièren menetelmä. Liittogradienttimenetelmät ovat tehokkaita, kun muuttujia on luokkaa 100 tai enemmän eikä kohdefunktion Hessen matriisi ole harva (vähän nollasta poikkeavia alkioita).
- *Nelderin ja Meadin polytooppihaku* soveltuu myös epäsileille funktioille. Menetelmä ei ole kuitenkaan tehokas, varsinkin jos muuttujia on paljon.

Näitä menetelmiä esitellään myöhemmin tässä luvussa. Useimmat näistä menetelmistä sisältyvät mm. IMSL- ja NAG-aliohjelmakirjastoihin. Ohjelmistoja on esitelty sivulla 370.

10.8.1 Ratkaisumenetelmien yleispiirteitä

Rajoitteettomaan optimointiin kehitetyistä algoritmeista osa käyttää vain funktion arvoja (esimerkiksi polytooppihaku), osa taas vaatii gradienttivektorin (liittogradienttimenetelmä) ja mahdollisesti myös Hessen matriisin laskemisen tai arvioimisen (Newtonin menetelmän tyyppiset algoritmit).

Newtonin tyyppiset menetelmät soveltuvat sileiden rajoitteettomien optimointitehtävien ratkaisemiseen, jos kohdefunktion f gradienttivektori ∇f ja Hessen matriisi H voidaan laskea tai arvioida tehokkaasti. Seuraavassa on esitetty algoritmin perusrakenne.

Algoritmi 10.8.1 (Newtonin tyyppinen ratkaisumenetelmä)

valitse alkuarvaus $\mathbf{x}^1$ ja aseta $k = 1$
repeat
  ratkaise hakusuunta $\mathbf{p}^k$ yhtälöstä $H_k\mathbf{p}^k = -\nabla f(\mathbf{x}^k)$
  hae uusi piste ratkaisuavaruudesta: $\mathbf{x}^{k+1} = \mathbf{x}^k + \lambda_k\mathbf{p}^k$
  $k = k + 1$
until $\|\mathbf{x}^k - \mathbf{x}^{k-1}\| < \epsilon$

Liittogradienttimenetelmissä hakusuunta $\mathbf{p}^k$ saadaan kaavasta

$$\mathbf{p}^k = -\nabla f(\mathbf{x}^k) + \lambda_k\mathbf{p}^{k-1} \tag{10.15}$$

ja sekanttimenetelmissä kaavasta

$$B_k\,\mathbf{p}^k = -\nabla f(\mathbf{x}^k)\,, \tag{10.16}$$

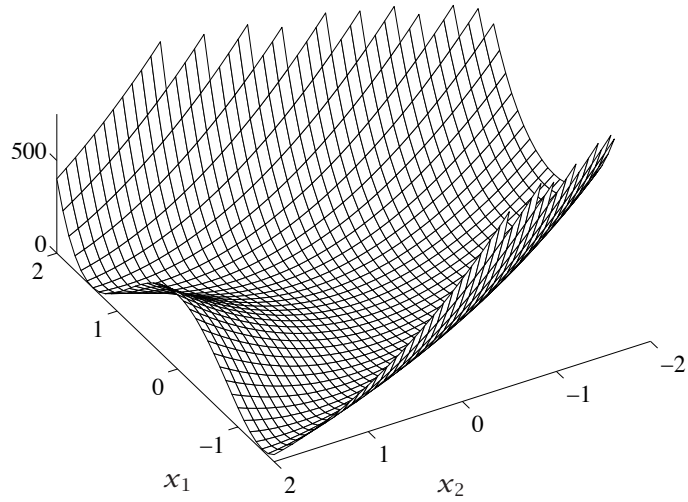
missä B_k on Hessen matriisin arvio. Sekanttimenetelmissä matriisia B_k päivitetään kullakin iteraatioaskeleella. Jos matriisiksi B_k asetetaan identiteettimatriisi, saadaan jyrkimmän laskun menetelmä (steepest descent), ja jos

matriisiksi B_k asetetaan Hessen matriisi $H(\mathbf{x}^k)$, saadaan Newtonin menetelmä.

Hakuaskeleen pituus λ_k riippuu käytetystä menetelmästä. Askel voidaan määrittää tähän tarkoitukseen kehitetyillä yksiulotteisilla optimointimenetelmillä (niin sanottu *viivahaku*, line search).

Merkintä $\mathbf{x}^{k+1} = \mathbf{x}^k + \lambda_k \mathbf{p}^k$ tarkoittaa, että minimoimme funktiota f suuntaan $\mathbf{p}^k$. Siis haemme pistettä $\mathbf{x}^{k+1} = \mathbf{x}^k + \lambda_k \mathbf{p}^k$, jossa funktio $f(\mathbf{x}^k + \lambda_k \mathbf{p}^k)$, $\lambda_k > 0$, saa minimin. Tähän yksiulotteiseen minimointitehtävään käytetään usein *takautuvia viivahakualgoritmeja* (backtracking line search): arvataan ensin (suuri) askelpituus, jota pienennetään kunnes löydetään riittävän hyvä likiarvo minimipisteelle.

Suurten tehtävien ratkaisuun vaikuttaa erittäin paljon tehtävän tyyppi, esimerkiksi Hessen matriisin harvuus. Newtonin tyyppiset menetelmät ovat luotettavia ja tehokkaita myös suurissa tehtävissä, varsinkin *katkaistu Newtonin menetelmä*. Muistinkäytön vuoksi voimme kuitenkin joutua valitsemaan rajoitetun muistin sekanttimenetelmän tai liittogradienttimenetelmän.



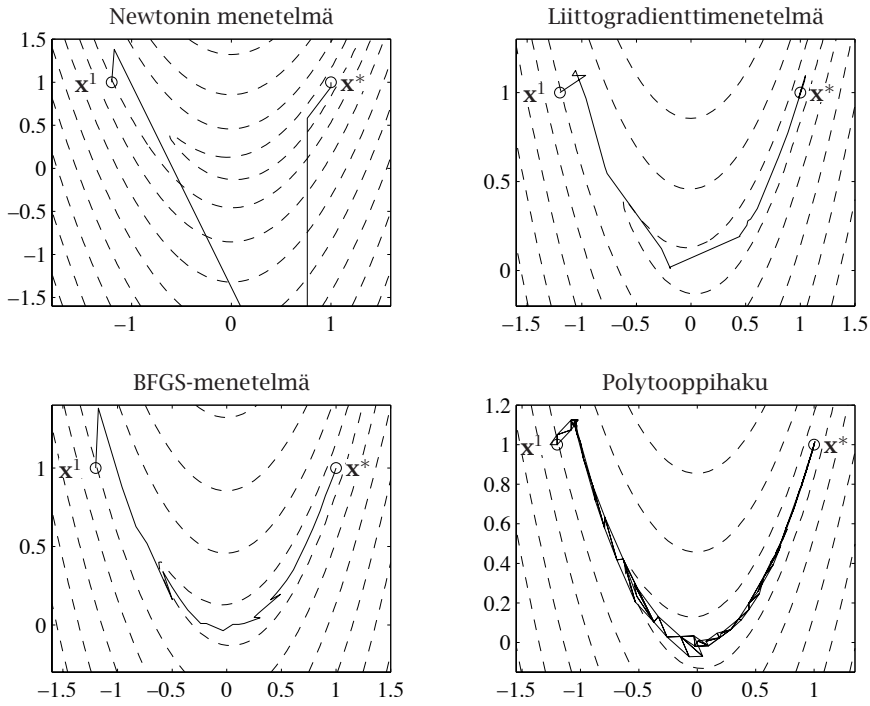
Kuva 10.8: *Rosenbrockin funktio* $f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$.

Käytämme kaksiulotteista Rosenbrockin funktiota $f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$ algoritmien havainnollistamiseen (kuva 10.8). Tehtävän minimipiste on $(1, 1)^T$. Kaareutuva kapea laakso aiheuttaa ongelmia useimmille ratkaisumenetelmille. Kuva 10.9 havainnollistaa eri ratkaisualgoritmien käyttäytymistä.

10.8.2 Newtonin menetelmä

Aluksi johdamme Newtonin menetelmän kvadraattiselle funktiolle

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} + \mathbf{b}^T\mathbf{x} + \mathbf{c}.$$



Kuva 10.9: *Rosenbrockin kaksiulotteisen funktion minimointi eri menetelmillä. Piste* $\mathbf{x}^1$ *on alkuarvaus ja piste* $\mathbf{x}^*$ *tehtävän minimipiste.*

Saamme funktion gradientiksi $\nabla f(\mathbf{x}) = A\mathbf{x} + \mathbf{b}$ ja Hessen matriisiksi $H(\mathbf{x}) = A$. Gradientti pisteessä $\mathbf{x} + \mathbf{p}$ on

$$\begin{aligned} \nabla f(\mathbf{x}+\mathbf{p}) &= A(\mathbf{x}+\mathbf{p}) + \mathbf{b} = (A\mathbf{x}+\mathbf{b}) + (A\mathbf{p}) \\ &= \nabla f(\mathbf{x}) + H(\mathbf{x})\,\mathbf{p}. \end{aligned}$$

Saamme yhtälöstä $\nabla f(\mathbf{x}+\mathbf{p}) = \mathbf{0}$ ehdon kvadraattisen funktion nollakohtaan vievälle askeleelle $\mathbf{p}$:

$$H(\mathbf{x})\,\mathbf{p} = -\nabla f(\mathbf{x}).$$

Newtonin iteraatio toimii myös ei-kvadraattisille funktioille, sillä voimme arvioida funktiota $f : \mathbb{R}^n \to \mathbb{R}$ kvadraattisella mallilla (luku 4)

$$f(\mathbf{x}+\mathbf{d}) \approx f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{d}\rangle + \tfrac{1}{2}\langle \mathbf{d}, H(\mathbf{x})\,\mathbf{d}\rangle,$$

missä $H(\mathbf{x})$ on funktion f Hessen matriisi. Newtonin menetelmässä siis ratkaistaan jokaisella iteraatioaskeleella lineaarinen yhtälöryhmä

$$H(\mathbf{x}^k)\,\mathbf{p}^k = -\nabla f(\mathbf{x}^k). \tag{10.17}$$

Koska Hessen matriisi $H(\mathbf{x}^k)$ on symmetrinen ja monissa käytännön tehtävissä myös harva, voidaan ratkaisussa käyttää erikoismenetelmiä. Newtonin iteraatio voidaan lopettaa esimerkiksi silloin, kun suuntavektorin normi $\|\mathbf{p}^k\|$ on riittävän pieni.

Newtonin menetelmä suppenee, jos iteraatiopiste $\mathbf{x}^k$ on tarpeeksi lähellä paikallista minimiä ja Hessen matriisi on positiivisesti definiitti. Suppeneminen on lisäksi neliöllistä:

$$\|\mathbf{x}^{k+1} - \mathbf{x}^*\| \leq \beta \|\mathbf{x}^k - \mathbf{x}^*\|^2, \quad \beta \geq 0\,. \tag{10.18}$$

Täten Newtonin menetelmän suppenemisnopeus on erittäin hyvä lähellä minimipistettä $\mathbf{x}^*$.

Esimerkki 10.8.1 Tarkastelemme kuvassa 10.8 esitetyn Rosenbrockin funktion

$$f(\mathbf{x}) = 100\left(x_2 - x_1^2\right)^2 + (1 - x_1)^2$$

minimointia. Funktion gradienttivektori on

$$\nabla f(\mathbf{x}) = \begin{pmatrix} -400x_1(x_2 - x_1^2) + 2(x_1 - 1) \\ 200(x_2 - x_1^2) \end{pmatrix}$$

ja Hessen matriisi on

$$H(\mathbf{x}) = \begin{pmatrix} 1200x_1^2 - 400x_2 + 2 & -400x_1 \\ -400x_1 & 200 \end{pmatrix}.$$

Alkuarvaus olkoon $(-1.2, 1.0)^T$. Lopetusehtona on $\|\mathbf{p}^k\| < 10^{-8}$. Saamme iteraatiopisteiksi $\mathbf{x}^k$ ja funktion arvoiksi $f(\mathbf{x}^k)$:

k	x_1^k	x_2^k	$f(\mathbf{x}^k)$
1	−1.200	1.000	24.200
2	−1.175	1.381	4.732
3	0.763	−3.175	1412.000
4	0.763	0.583	0.056
5	1.000	0.944	0.313
6	1.000	1.000	$1.853 \cdot 10^{-11}$
7	1.000	1.000	$3.433 \cdot 10^{-20}$

Iteraatiolla $k = 3$ joudumme melko kauas minimipisteestä. Tarvitsimme minimiarvon löytämiseen 7 Newtonin askelta.

Jotta Newtonin menetelmä olisi käyttökelpoinen, tarvitaan keino selviytyä Hessen matriisin mahdollisesta singulaarisuudesta. *Modifioidussa Newtonin menetelmässä* pyritään lisäksi pitämään Hessen matriisi positiivisesti definiittisenä. Suppenemista voi myös yrittää varmistaa viivahaun avulla (sivu 10.8.1).

Yhtälöryhmän (10.17) tarkka ratkaiseminen vaatii paljon resursseja (keskusmuistia ja laskenta-aikaa), kun muuttujia on paljon. Usein laskennan alkuvaiheissa kannattaakin ratkaista yhtälöryhmä likimääräisesti iteratiivisten ratkaisumenetelmien avulla. Tarkkaan ratkaisemiseen kannattaa siirtyä vasta lähestyttäessä optimipistettä. *Katkaistussa Newtonin menetelmässä* sallitaan nollasta poikkeava jäännösvektori $\mathbf{r}^k = H_k\mathbf{p}^k + \nabla f(\mathbf{x}^k)$. Residuaalin $\mathbf{r}^k$ sallittu suuruusluokka määritellään esimerkiksi kriteerillä (skalaari η_k on menetelmän parametri):

$$\tilde{r}_k = \frac{\|\mathbf{r}^k\|}{\|\nabla f(\mathbf{x}^k)\|} \leq \eta_k < 1. \tag{10.19}$$

10.8.3 Sekanttimenetelmät

Sekanttimenetelmissä arvioidaan Newtonin menetelmässä tarvittavaa Hessen matriisia iteratiivisten päivitysten avulla. Koska matriisin päivitysoperaatio vie vähän työtä verrattuna Hessen matriisin laskemiseen, menetelmä on tehokas huolimatta Newtonin menetelmää hitaammasta suppenemisesta. Iteraatiomatriisi B_k pyritään pitämään positiivisesti definiittinä. Päivitykset tehdään menetelmästä riippuvalla matriisilla Q_k:

$$B_{k+1} = B_k + Q_k\,. \tag{10.20}$$

Kullakin iteraatioaskeleella ratkaistaan hakusuunta $\mathbf{p}^k$ yhtälöryhmästä

$$B_k\,\mathbf{p}^k = -\nabla f(\mathbf{x}^k), \tag{10.21}$$

jota voi verrata Newtonin menetelmän yhtälöryhmään (10.17). Paljon käytetty BFGS-sekanttimenetelmä (Broyden, Fletcher, Goldfarb ja Shanno) on esitetty seuraavassa.

Algoritmi 10.8.2 (BFGS-sekanttimenetelmä)

valitse $\mathbf{x}^1$ ja B_1
valitse vakio $\epsilon > 0$
for $k = 1, 2, \ldots$
  ratkaise suunta $\mathbf{p}^k$ yhtälöryhmästä $B_k\mathbf{p}^k = -\nabla f(\mathbf{x}^k)$
  tee viivahaku: $\mathbf{x}^{k+1} = \mathbf{x}^k + \lambda_k\mathbf{p}^k$
  $\mathbf{s}^k = \mathbf{x}^{k+1} - \mathbf{x}^k$
  $\mathbf{y}^k = \nabla f(\mathbf{x}^{k+1}) - \nabla f(\mathbf{x}^k)$
  $B_{k+1} = B_k + \dfrac{\mathbf{y}^k\mathbf{y}^{k^T}}{\langle \mathbf{y}^k, \mathbf{s}^k\rangle} - \dfrac{B_k\mathbf{s}^k\mathbf{s}^{k^T}B_k}{\langle \mathbf{s}^k, B_k\mathbf{s}^k\rangle}$
until $\|\nabla f(\mathbf{x}^{k+1})\| < \epsilon$

Matriisi B_k on Hessen matriisin approksimaatio. BFGS-sekanttimenetelmä pitää matriisin B_k positiivisesti definiittinä, joten sille voi tehdä Choleskyn hajotelman LL^T. Matriisipäivitykset voi tehdä suoraan hajotelmaan, mikä säästää työtä. Menetelmän työmäärä on siis verrannollinen lausekkeeseen n^2.

Alkuarvaus B_1 voi olla esimerkiksi identiteettimatriisi tai Hessen matriisin differenssiarvio. BFGS-iteraatiot voidaan lopettaa, kun $\|\mathbf{s}^k\|$ on kyllin pieni ja gradienttivektori $\nabla f(\mathbf{x}^k)$ lähestyy nollaa. BFSG-menetelmän suppeneminen on yleensä lineaarista ja lähellä optimikohtaa superlineaarista.

Esimerkki 10.8.2 Haemme kuvassa 10.8 esitetyn Rosenbrockin funktion minimin BFGS-menetelmällä. Matriisiksi B_1 asetamme Hessen matriisin lähtöpisteessä $H(\mathbf{x}^1)$. Lopetusehtomme on $\|\nabla f(\mathbf{x}^k)\| < 10^{-6}$. Käytämme viivahakua suppenemisen varmistamiseksi.

Saamme iteraatiopisteiksi $\mathbf{x}^k$ ja funktion arvoiksi $f(\mathbf{x}^k)$:

k	x_1^k	x_2^k	$f(\mathbf{x}^k)$
1	−1.200	1.000	24.200
2	−1.175	1.381	4.732
3	−1.151	1.324	4.626
	...		
38	1.000	1.000	$3.860 \cdot 10^{-11}$
39	1.000	1.000	$2.715 \cdot 10^{-14}$
40	1.000	1.000	$2.245 \cdot 10^{-18}$

Tarvitsimme minimiarvon löytämiseen 40 BFGS-askelta.

10.8.4 Liittogradienttimenetelmät

BFGS-sekanttimenetelmä on luotettava ja tehokas, mutta isoille tehtäville ovat *liittogradienttimenetelmät* (conjugate gradient) usein tehokkaampia pienemmän muistintarpeensa ansiosta.

Algoritmi 10.8.3 (Liittogradienttimenetelmä)

hae alkuarvaus $\mathbf{x}^1 \in \mathbb{R}^n$
repeat
  $k = 1$
  repeat
    $\mathbf{p}^k = \nabla f(\mathbf{x}^k)$
    if $k = 1$
      $\beta_1 = 0, \quad \mathbf{s}^0 = \mathbf{0}$
    else
      laske β_k alla esitetyllä kaavalla (10.22) tai (10.23)
    end
    $\mathbf{s}^k = -\mathbf{p}^k + \beta_k \mathbf{s}^{k-1}$
    tee viivahaku: $\mathbf{x}^{k+1} = \mathbf{x}^k + \lambda_k \mathbf{s}^k$
    $k = k + 1$
  until $\|\mathbf{p}^k\| < \epsilon$ tai $k > \eta$
until $\|\mathbf{p}^k\| < \epsilon$

Tämä on *uudelleenkäynnistetty* liittogradienttimenetelmä, jossa käynnistetään iteraatio uudestaan aina η askeleen välein. Skalaarikerroin β_k lasketaan Fletcherin ja Reevesin menetelmässä kaavalla

$$\beta_k = \frac{\langle \mathbf{p}^k, \mathbf{p}^k \rangle}{\langle \mathbf{p}^{k-1}, \mathbf{p}^{k-1} \rangle}, \quad k = 2, 3, \ldots, \tag{10.22}$$

sekä Polakin ja Ribièren menetelmässä kaavalla

$$\beta_k = \frac{\langle \mathbf{p}^k - \mathbf{p}^{k-1}, \mathbf{p}^k \rangle}{\langle \mathbf{p}^{k-1}, \mathbf{p}^{k-1} \rangle}, \quad k = 2, 3, \ldots. \tag{10.23}$$

Esimerkki 10.8.3 Haemme Rosenbrockin funktion minimin Polakin ja Ribièren liittogradienttimenetelmällä. Seuraavassa on iteraatiopisteet ja funktion arvot:

k	x_1^k	x_2^k	$f(\mathbf{x}^k)$
1	−1.200	1.000	24.200
2	−0.965	1.096	6.612
3	−1.088	1.095	5.142
4	−1.057	1.124	4.236
5	−0.962	0.967	4.021
6	−0.788	0.582	3.347
	...		
30	1.000	1.000	$2.061 \cdot 10^{-9}$
31	1.000	1.000	$5.372 \cdot 10^{-11}$
32	1.000	1.000	$2.771 \cdot 10^{-16}$

Tarvitsimme minimiarvon löytämiseen 32 liittogradienttimenetelmän iteraatiota.

Jyrkimmän laskun menetelmässä (steepest descent) käytetään hyväksi gradienttivektoria vain nykyisessä iteraatiopisteessä:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \lambda_k \nabla f(\mathbf{x}^k).$$

Vaikka menetelmä tuntuu houkuttelevan yksinkertaiselta, ei sitä tulisi käyttää. Menetelmä suppenee vain lineaarisesti. Jos Hessen matriisi $H(\mathbf{x})$ on häiriöaltis optimikohdan $\mathbf{x}^*$ läheisyydessä, suppeneminen on hyvin hidasta. Lisäksi menetelmän vaatima työmäärä ja muistitila eivät juuri poikkea liittogradienttimenetelmästä. Menetelmällä on kuitenkin yksi etu: se suppenee globaalisti kohti lokaalia minimiä.

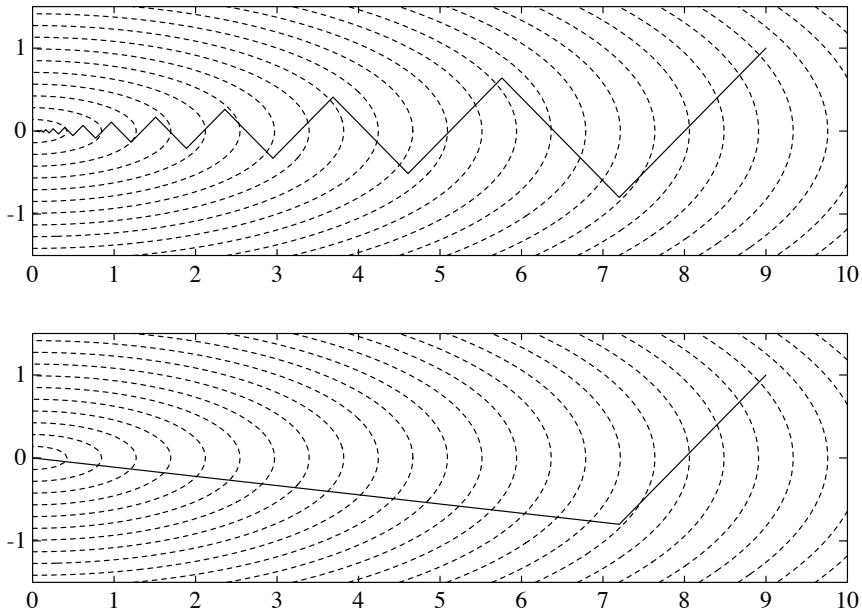
Esimerkki 10.8.4 Funktion $f(\mathbf{x}) = \frac{1}{2}x_1^2 + \frac{9}{2}x_2^2$ minimointi on esimerkki hiukan huonosti skaalatusta tehtävästä. Funktion Hessen matriisi on

$$H(\mathbf{x}) = \begin{pmatrix} 1 & 0 \\ 0 & 9 \end{pmatrix} \text{ kaikilla } \mathbf{x} \in \mathbb{R}^2.$$

Jyrkimmän laskun menetelmän tuottamat iteraatiopisteet tälle tehtävälle alkuarvauksella $\mathbf{x}^1 = (9, 1)^T$ ovat

$$\mathbf{x}^k = \begin{pmatrix} 9 \\ (-1)^{k-1} \end{pmatrix} (0.8)^{k-1}, \; k = 2, 3, \ldots$$

Menetelmän käyttäytymistä on havainnollistettu kuvassa 10.10. Jyrkimmän laskun menetelmä tuottaa hakupolun, jossa uusi hakusuunta on kohtisuorassa edellistä hakusuuntaa vastaan. Liittogradienttimenetelmä puolestaan ratkaisee tehtävän kahdella iteraatiolla.



Kuva 10.10: *Jyrkimmän laskun menetelmän (yläkuva) ja liittogradienttimenetelmän (alakuva) suppeneminen kvadraattisen funktion* $f(\mathbf{x}) = x_1^2/2 + 9x_2^2/2$ *minimoinnissa. Kuvaan on piirretty katkoviivoilla funktion tasa-arvokäyrät. Menetelmien tuottamat hakupisteet on yhdistetty murtoviivalla; alkupiste on* $\mathbf{x}^1 = (9,1)^T$.

10.8.5 Suorat menetelmät

Suorissa menetelmissä ei tarvitse laskea funktion derivaattoja eli gradienttivektoria tai Hessen matriisia. Suosittu epälineaarisen optimoinnin suora menetelmä on *Nelderin ja Meadin polytooppihaku*. Menetelmässä muodostetaan n-ulotteinen monitahokas eli polytooppi (jota kutsutaan myös *simpleksiksi*), jota sopivasti skaalaamalla ja sivujen suhteen peilaamalla pyritään löytämään minimin sijainti.

Polytooppi sisältää $n+1$ kärkipistettä $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n+1})$. Merkitsemme pienintä ja suurinta funktion arvoa polytoopin kärkipisteissä $f_l = \min_i f(\mathbf{x}_i)$ ja $f_h = \max_i f(\mathbf{x}_i)$. Olkoon lisäksi $f_i = f(\mathbf{x}_i)$.

Laskemme polytoopin n parhaan kärjen painopisteen $\mathbf{x}_c = \frac{1}{n}\sum_{i \neq h} \mathbf{x}_i$. Peilauskerroin on $\alpha = 1 > 0$, laajennuskerroin $\gamma = 2 > 1$ ja pienennyskerroin $\beta = 0.5 \in [0,1]$. Kuva 10.11 havainnollistaa polytooppihaussa käytettyjä skaalaus- ja peilausoperaatioita.

Algoritmi 10.8.4 (Polytooppihaku)

1: Peilaa huonoin kärki $\mathbf{x}_h$ vastapäätä olevan sivun suhteen:

$$\mathbf{x}_r = (1+\alpha)\,\mathbf{x}_c - \alpha \mathbf{x}_h\,.$$

Jos $f_r > f_i$ kaikilla $i \neq h$, mene vaiheeseen 3. Jos taas $f_r \in [f_l, f_h]$, aseta $\mathbf{x}_h = \mathbf{x}_r$ ja toista vaihe 1 saadulle uudelle polytoopille. Muuten jatka vaiheesta 2.

2: Jos $f_r < f_l$, suurenna polytooppia löydettyyn suuntaan:

$$\mathbf{x}_e = \gamma \mathbf{x}_r + (1 - \gamma)\, \mathbf{x}_c \,.$$

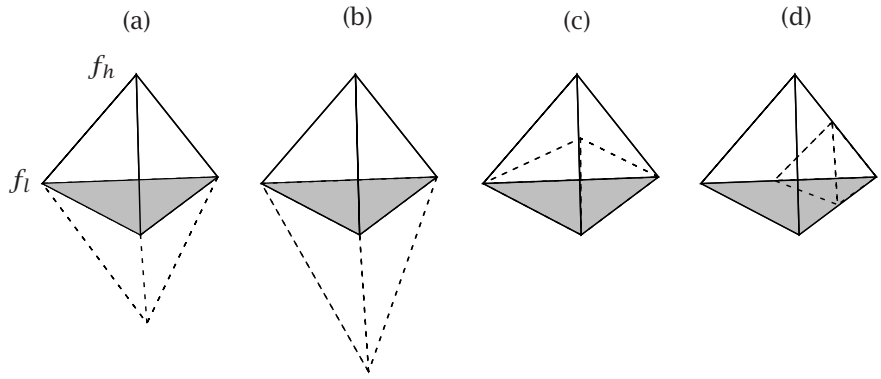
Jos $f_e < f_l$, aseta $\mathbf{x}_h = \mathbf{x}_e$, muuten $\mathbf{x}_h = \mathbf{x}_r$ (polytoopin suurentaminen ei kannata). Jatka vaiheesta 1.

3: Jos $f_r < f_h$, aseta $\mathbf{x}_h = \mathbf{x}_r$. Pienennä polytooppia:

$$\mathbf{x}_s = \beta \mathbf{x}_h + (1 - \beta)\, \mathbf{x}_c \,.$$

Jos $f_s < \min\{f_h, f_r\}$, aseta $\mathbf{x}_h = \mathbf{x}_s$. Muussa tapauksessa aseta $\mathbf{x}_i = (\mathbf{x}_i + \mathbf{x}_l)/2$ eli siis puolita polytoopin sivut. Mene vaiheeseen 1.

Polytooppihaku on kohtalaisen luotettava, mutta sen suppenemisnopeus on huono verrattuna derivaattoja käyttäviin menetelmiin. Lisäksi menetelmä voi (huonosti toteutettuna) juuttua ikuiseen silmukkaan. Tällaisesta tilanteesta voi selvitä käynnistämällä algoritmin uudestaan. Toisaalta polytooppihaku soveltuu tapauksiin, jossa optimoitava funktio ei ole sileä. Polytooppihakuun perustuva rutiini löytyy esimerkiksi IMSL-aliohjelmakirjastosta.



Kuva 10.11: *Esimerkkejä polytooppihaussa käytetyistä monitahokkaan skaalausja peilausoperaatioista. Uusi polytooppi on piirretty katkoviivaa käyttäen. Kohdassa (a) on esitetty peilausoperaatio, kohdassa (b) peilaus ja laajennus, kohdassa (c) pienennys yhden kärkipisteen osalta ja kohdassa (d) pienennys monen kärkipisteen osalta.*

10.9 Epälineaariset pienimmän neliösumman tehtävät

Sovitettaessa parametreista $\mathbf{a} \in \mathbb{R}^n$ riippuva epälineaarinen funktio $y = f(\mathbf{t}, \mathbf{a})$ mittausdataan $(\mathbf{t}_i, y_i)$, $i = 1, \ldots, m$, joudutaan usein epälineaarisiin

pienimmän neliösumman tehtäviin. (Katso myös approksimoinnista kertovaa kappaletta 4.2 sivulla 74.) Parametreille $\mathbf{a}$ haetaan sellaiset arvot, jotka minimoivat neliösumman

$$F(\mathbf{a}) = \sum_{i=1}^{m} w_i^2 \left(y_i - f(\mathbf{t}_i, \mathbf{a})\right)^2. \tag{10.24}$$

Arvot w_i ovat sovituksessa käytettyjä painokertoimia.

Yleinen muoto epälineaariselle pienimmän neliösumman tehtävälle on

$$\underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimi}}\, F(\mathbf{x}) = \sum_{i=1}^{m} \left(f_i(\mathbf{x})\right)^2 = \langle \mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x}) \rangle, \tag{10.25}$$

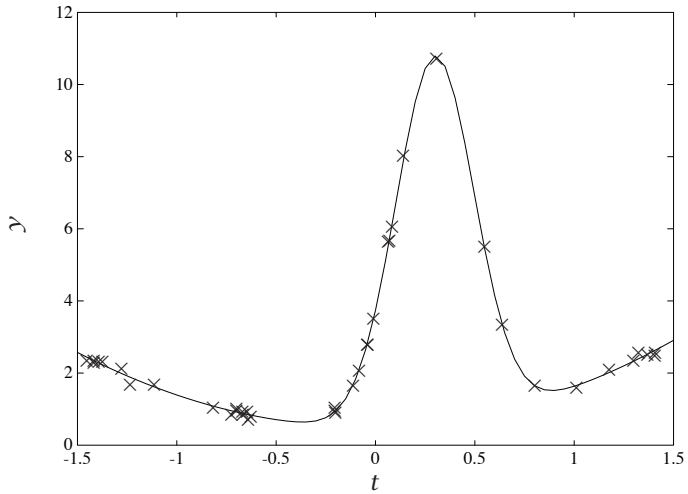
missä piste $\mathbf{x}$ kuuluu joukkoon $\mathbb{R}^n$ ja funktio $\mathbf{f}(\mathbf{x})$ on kuvaus $\mathbb{R}^n \mapsto \mathbb{R}^m$. Saamme tehtävän (10.24) tähän muotoon asettamalla $\mathbf{x} = \mathbf{a}$ ja $f_i(\mathbf{x}) = w_i(y_i - f(\mathbf{t}_i, \mathbf{x}))$.

Esimerkki 10.9.1 Käytettävissä on mittausdata (t_i, y_i), $i = 1, \ldots, m$. Sovitamme dataan seuraavan mallin:

$$y = f(t, \mathbf{a}) + \epsilon = a_1 + a_2 t + a_3 t^2 + a_4 e^{-(t-a_5)^2/(2a_6^2)} + \epsilon, \tag{10.26}$$

missä ϵ on virhetermi. Saamme minimointitehtävään yhtälön (10.24) mukaisen kohdefunktion $F(\mathbf{a})$, $\mathbf{a} \in \mathbb{R}^6$. Olkoot kaikkien mittauspisteiden painot samat: $w_i = 1, i = 1, \ldots, m$.

Mittausdatan pisteet ja saatu malli on esitetty kuvassa 10.12. Sovituksessa on käytetty Levenbergin ja Marquardtin menetelmää. Myös Gaussin ja Newtonin menetelmää voi käyttää. Menetelmiä on kuvattu kappaleissa 10.9.2 ja 10.9.3.



Kuva 10.12: *Pienimmän neliösumman sovituksen tuottama malli (yhtenäinen viiva) ja käytetty mittausdata (merkitty symbolilla ×).*

10.9.1 Newtonin tyyppiset ratkaisumenetelmät

Tarkastelemme yhtälössä (10.25) esitetyn funktion $F(\mathbf{x})$ minimoimista. Vektoriarvoisen funktion $\mathbf{f}(\mathbf{x}) : \mathbb{R}^n \mapsto \mathbb{R}^m$ Jacobin matriisi on

$$J(\mathbf{x}) = \begin{pmatrix} \nabla^T f_1(\mathbf{x}) \\ \vdots \\ \nabla^T f_m(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_m(\mathbf{x})}{\partial x_n} \end{pmatrix}.$$

Saamme funktion $F(\mathbf{x}) = \langle \mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x}) \rangle$ gradientiksi $\nabla F(\mathbf{x}) = 2J(\mathbf{x})^T \mathbf{f}(\mathbf{x})$ ja Hessen matriisiksi $H(\mathbf{x}) = 2J(\mathbf{x})^T J(\mathbf{x}) + 2S(\mathbf{x})$, missä matriisi S määritellään

$$S(\mathbf{x}) = \sum_{i=1}^{m} f_i(\mathbf{x}) \nabla \nabla^T f_i(\mathbf{x}).$$

Saamme näin johdettua Newtonin iteraation epälineaarisille pienimmän neliösumman tehtäville:

$$\left(J(\mathbf{x}^k)^T J(\mathbf{x}^k) + S(\mathbf{x}^k) \right) \mathbf{p}^k = -J(\mathbf{x}^k)^T \mathbf{f}(\mathbf{x}^k) \tag{10.27}$$

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{p}^k. \tag{10.28}$$

Iteraatio on raskas, sillä matriisin $S(\mathbf{x}^k)$ laskemiseen tarvitaan funktioiden f_i Hessen matriisit. Jos matriisin $S(\mathbf{x}^k)$ laskeminen jätetään pois iteraatiosta, saamme *pienen residuaalin algoritmin*, joka soveltuu tapauksiin, joissa $|F(\mathbf{x}^*)| \approx 0$. Muuten saamme *suuren residuaalin algoritmin*. Seuraavassa esittelemme kaksi menetelmätyyppiä: Gaussin ja Newtonin menetelmän sekä Levenbergin ja Marquardtin menetelmän. Kumpikin menetelmä soveltuu hyvin toteutettuna kohtalaisen vaikeisiinkin tehtäviin.

10.9.2 Gaussin ja Newtonin menetelmä

Koska funktioiden f_i Hessen matriisit ovat raskaita laskea, Gaussin ja Newtonin menetelmässä ratkaistaan yhtälöryhmä

$$J(\mathbf{x}^k)^T J(\mathbf{x}^k)\, \mathbf{p}^k = -J(\mathbf{x}^k)^T \mathbf{f}(\mathbf{x}^k) \tag{10.29}$$

vektorin $\mathbf{p}^k$ suhteen. Tässä matriisi $J(\mathbf{x}^k)^T J(\mathbf{x}^k)$ on aina vähintään positiivisesti semidefiniitti.

Gaussin ja Newtonin menetelmä kuuluu pienen residuaalin menetelmiin. Menetelmä tarvitsee hyvän alkuarvauksen, sillä algoritmi voi supeta hitaasti tai törmätä singulariteetteihin. Usein askel $\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{p}^k$ korvataan viivahaulla suuntaan $\mathbf{p}^k$, jolloin suppeneminen tulee varmemmaksi.

Hakuaskelta $\mathbf{p}^k$ ei kuitenkaan kannata ratkaista suoraan yhtälöryhmästä (10.29), sillä tämä yhtälöryhmä voidaan tulkita normaaliyhtälöksi lineaariselle pienimmän neliösumman tehtävälle

$$J(\mathbf{x}^k)\, \mathbf{p}^k = -\mathbf{f}(\mathbf{x}^k). \tag{10.30}$$

Tämä tehtävä on ratkaistavissa luotettavasti käyttämällä sopivaa ortogonaalista hajotelmaa Jacobin matriisille $J(\mathbf{x}^k)$. Käytettäessä esimerkiksi singulaariarvohajotelmaa ratkaisu löytyy, vaikka matriisi olisi singulaarinen. Toisaalta hajotelman laskeminen vaatii melko paljon työtä.

10.9.3 Levenbergin ja Marquardtin menetelmä

Levenbergin ja Marquardtin menetelmässä ratkaistaan askel $\mathbf{p}^k$ yhtälöryhmästä

$$\left(J(\mathbf{x}^k)^T J(\mathbf{x}^k) + \mu_k I\right)\mathbf{p}^k = -J(\mathbf{x}^k)^T \mathbf{f}(\mathbf{x}^k), \tag{10.31}$$

missä I on yksikkömatriisi. Tällöin kullakin askeleella ratkaistaan seuraava lineaarinen pienimmän neliösumman tehtävä:

$$\begin{pmatrix} J(\mathbf{x}^k) \\ \sqrt{\mu_k}\, I \end{pmatrix}\mathbf{p}^k = \begin{pmatrix} -\mathbf{f}(\mathbf{x}^k) \\ \mathbf{0} \end{pmatrix}. \tag{10.32}$$

Jos parametri μ on suuri, vektorin $\mathbf{p}^k$ suunta lähestyy kohdefunktion gradienttivektorin suuntaa. Jos μ on pieni, menetelmä on lähellä Gaussin ja Newtonin menetelmää. Kerroin μ_k on käytännössä parasta valita adaptiivisella strategialla, jossa parametrin arvoa kasvatetaan tarvittaessa.

10.10 Rajoitteita sisältävät tehtävät

Rajoitteellisten optimointitehtävien yleinen muoto on

$$\underset{\mathbf{x}}{\text{minimi}}\ f(\mathbf{x}),\ \text{kun}\ \mathbf{g}(\mathbf{x}) \le \mathbf{0}\ \text{ja}\ \mathbf{h}(\mathbf{x}) = \mathbf{0}. \tag{10.33}$$

Oletamme, että kohdefunktio f ja rajoitefunktiot $\mathbf{g}$ ja $\mathbf{h}$ ovat sileitä. Jos funktiot ovat lineaarisia, on kyseessä lineaarinen optimointitehtävä. Muussa tapauksessa kyseessä on epälineaarinen rajoitteellinen optimointitehtävä, jonka ratkaiseminen voi vaatia eri menetelmien kokeilua ja ehkä räätälöintiä probleemaan sopivaksi.

Tässä teoksessa emme pysty käymään läpi kaikkia vaihtoehtoisia ratkaisumenetelmiä. Kappaleessa 10.10.1 esittelemme lineaarista optimointia ja kappaleessa 10.10.2 kvadraattista optimointia. Kappaleessa 10.10.3 esitelty SQP-menetelmä soveltuu tilanteisiin, joissa sekä kohdefunktio että rajoitteet ovat epälineaarisia. Kappaleessa 10.10.4 esitellään sakko- ja estefunktioiden käyttöä rajoitteellisen tehtävän muuntamiseksi rajoitteettomaksi.

10.10.1 Lineaarinen optimointi

Lineaarinen optimointitehtävä eli LP-tehtävä (linear programming) määritellään useimmiten käyttäen epäyhtälömuotoisia rajoitteita. Tällöin on kysees-

sä minimointitehtävä

$$\operatorname*{minimi}_{\mathbf{x}} \langle \mathbf{c}, \mathbf{x} \rangle, \text{ kun } A\mathbf{x} \leq \mathbf{b} \text{ ja } \mathbf{x} \geq \mathbf{0}, \tag{10.34}$$

missä vektorit $\mathbf{x}$ ja $\mathbf{c}$ kuuluvat joukkoon $\mathbb{R}^n$ ja vektori $\mathbf{b}$ kuuluu joukkoon $\mathbb{R}^m$. Matriisin A vaakarivien lukumäärä olkoon m ja sarakkeiden määrä n.

Esimerkki 10.10.1 Kuvassa 10.13 on esitetty seuraavan LP-tehtävän käypä alue:

$$\operatorname*{minimi}_{x_1,x_2} -3x_1 - x_2, \text{ kun } \begin{cases} -6x_1 + 5x_2 \leq 30, \\ -7x_1 + 12x_2 \leq 84, \\ 19x_1 + 14x_2 \leq 266, \\ 4x_1 - 7x_2 \leq 28, \\ 0 \leq x_1 \leq 10, \\ 0 \leq x_2 \leq 9. \end{cases} \tag{10.35}$$

Kuvassa 10.13 on kohdefunktiolle $f(\mathbf{x}) = -3x_1 - x_2$ piirretty katkoviivoilla suorat $f(\mathbf{x}) =$ vakio. Käypä alue on merkitty paksulla yhtenäisellä viivalla. Tehtävä on matriisimuodossa

$$\operatorname*{minimi}_{\mathbf{x}} \langle \mathbf{c}, \mathbf{x} \rangle, \text{ kun } A\mathbf{x} \leq \mathbf{b} \text{ ja } \mathbf{x} \geq \mathbf{0}, \tag{10.36}$$

missä vektorit ja matriisit määritellään seuraavasti:

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \quad \mathbf{c} = \begin{pmatrix} -3 \\ -1 \end{pmatrix}, \quad A = \begin{pmatrix} -6 & 5 \\ -7 & 12 \\ 19 & 14 \\ 4 & -7 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 30 \\ 84 \\ 266 \\ 28 \\ 10 \\ 9 \end{pmatrix}.$$

Standardimuotoisella lineaarisella optimointitehtävällä tarkoitetaan tehtävää

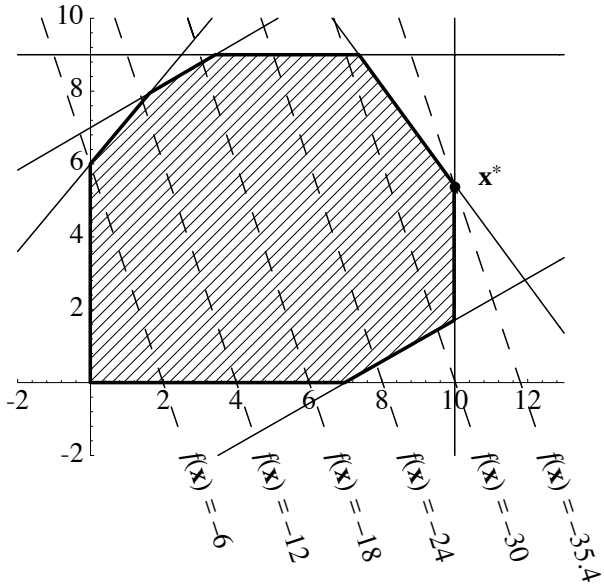
$$\operatorname*{minimi}_{\mathbf{x}} \langle \mathbf{c}, \mathbf{x} \rangle, \text{ kun } A\mathbf{x} = \mathbf{b} \text{ ja } \mathbf{x} \geq \mathbf{0}, \tag{10.37}$$

missä vektorit $\mathbf{c}$ ja $\mathbf{x}$ kuuluvat avaruuteen $\mathbb{R}^n$, vektori $\mathbf{b}$ kuuluu avaruuteen $\mathbb{R}^m$ ja matriisi A on $m \times n$ -matriisi, $m \leq n$. Tehtävän rajoitteet $A\mathbf{x} = \mathbf{b}$ ovat siis yhtälömuotoisia ja lisäksi käytetään positiivisuusehtoa $\mathbf{x} \geq \mathbf{0}$.

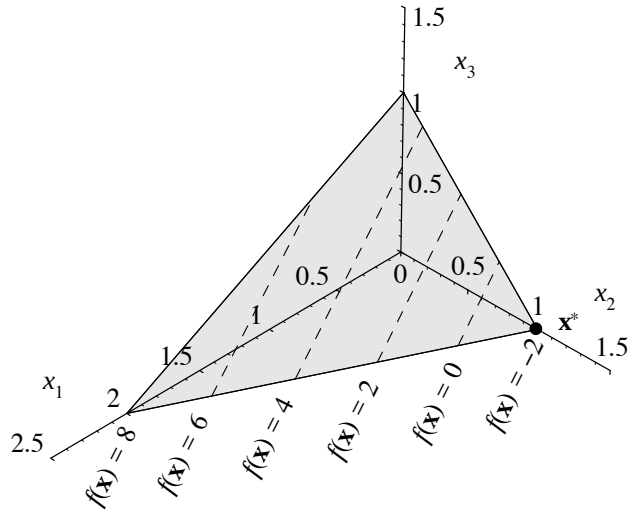
Esimerkki 10.10.2 Kuvassa 10.14 on esitetty standardimuotoinen lineaarinen optimointitehtävä

$$\operatorname*{minimi}_{\mathbf{x}} f(\mathbf{x}) = 4x_1 - 2x_2 + 5x_3, \text{ kun } x_1 + 2x_2 + 2x_3 = 2 \text{ ja } \mathbf{x} \geq \mathbf{0}.$$

Kolmen muuttujan lineaariselle kohdefunktiolle f on piirretty tasojen $f(\mathbf{x}) =$ vakio sijainti. Yhtälörajoitteen $x_1 + 2x_2 + 2x_3 = 2$ ja positiivisuusehtojen $\mathbf{x} \geq \mathbf{0}$ määrittelemä käypä alue on merkitty harmaalla pohjavärillä. Minimiarvo $f(\mathbf{x}^*) = -2$ saavutetaan kuvaan merkityssä käyvän alueen kulmapisteessä $\mathbf{x}^*$.



Kuva 10.13: *Kahden muuttujan lineaarinen optimointitehtävä.*



Kuva 10.14: *Kolmen muuttujan standardimuotoinen optimointitehtävä.*

Yhä eniten käytetty LP-tehtävien ratkaisualgoritmi on klassinen *simplex-menetelmä*. Menetelmässä siirrytään käyvän alueen nurkkapisteestä johonkin naapurina olevaan nurkkapisteeseen, kunnes minimi on löytynyt.

Tehokkaissa simplex-toteutuksissa käytetään harvoille matriiseille kehitettyjä hajotelmia ja esimerkiksi muuttujien ylä- ja alarajat otetaan huomioon erikseen. Pienissä ja keskisuurissa tehtävissä ($n + m < 1\,000 \ldots 100\,000$) simplex on tehokkaasti koodattuna käytännössä paras ja luotettavin menetelmä.

Suurissa tehtävissä (satoja tuhansia tai miljoonia muuttujia) ovat *sisäpistemenetelmät* usein tehokkaampia. Näissä menetelmissä sisällytetään lineaarisen optimointitehtävän epäyhtälörajoitteet kohdefunktioon käyttämällä *estefunktioita* (kappale 10.10.4). Ratkaistavaksi saadaan siten jono epälineaarisia optimointitehtäviä:

$$\operatorname*{minimi}_{\mathbf{x}} P(\mathbf{x}, \mu_k) = f(\mathbf{x}) - \mu_k \sum_{i=1}^{p} \ln x_i, \text{ kun } A\mathbf{x} = \mathbf{b} \text{ ja } \mu_k > 0. \tag{10.38}$$

Tässä on sisällytetty LP-tehtävän rajoitteet $\mathbf{x} \geq \mathbf{0}$ estefunktioon, joka pakottaa iteraatit käyvän alueen sisäpuolelle. Lähestyttäessä optimia sakkofunktiota jyrkennetään eli kasvatetaan kertoimen μ_k arvoa.

LP-tehtävän ratkaisemiseen sisäpistemenetelmillä kuluu työmäärä, joka on polynomisesti verrannollinen rajoitteiden lukumäärään m. Simplex-algoritmi on pahimmassa tapauksessa eksponentiaalinen eli ratkaisuaika voi olla verrannollinen arvoon 2^m. Käytännössä simplex-algoritmin työmäärä on usein kuitenkin lineaarinen verrattuna tehtävän kokoon.

10.10.2 Kvadraattiset tehtävät

Tärkeä epälineaarisen optimointitehtävän erikoistyyppi on *kvadraattinen optimointitehtävä*:

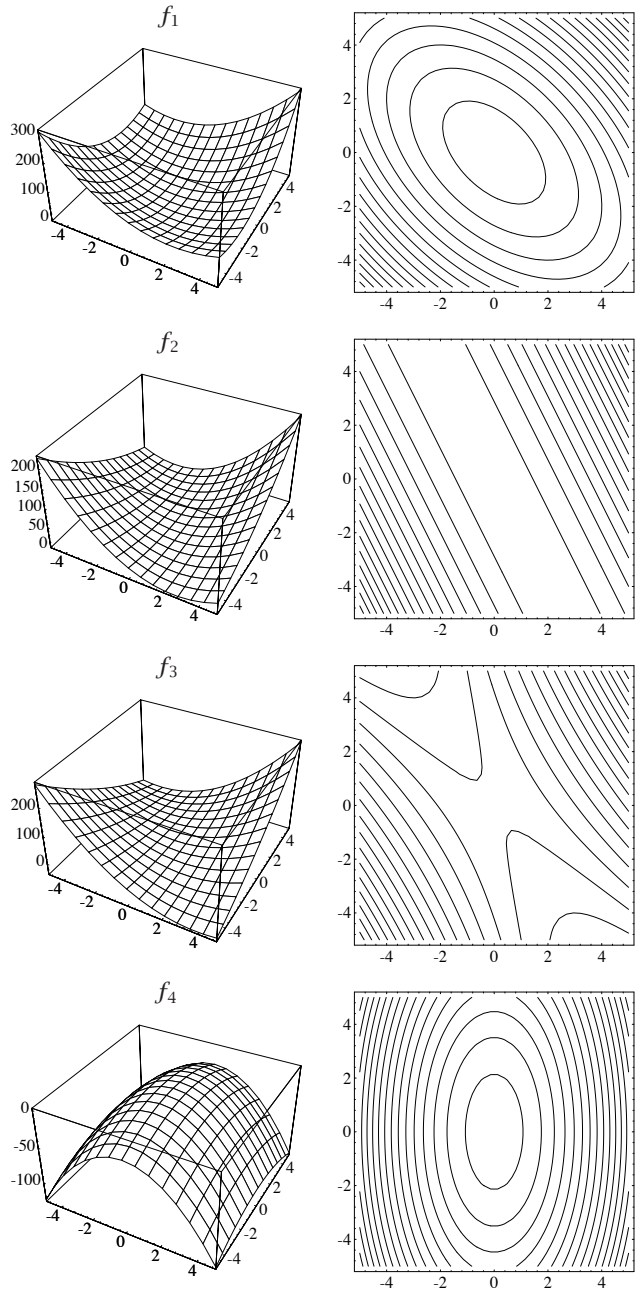
$$\operatorname*{minimi}_{\mathbf{x}} \langle \mathbf{c}, \mathbf{x} \rangle + \frac{1}{2} \langle \mathbf{x}, Q\mathbf{x} \rangle, \text{ kun } A\mathbf{x} = \mathbf{b} \text{ ja } \mathbf{x} \geq \mathbf{0}, \tag{10.39}$$

missä vektori $\mathbf{c}$ kuuluu joukkoon $\mathbb{R}^n$, matriisi Q on symmetrinen $n \times n$ -matriisi, vektori $\mathbf{b}$ kuuluu joukkoon $\mathbb{R}^m$ ja A on $m \times n$ -matriisi. Minimoitava kohdefunktio on siis aukaistuna

$$\langle \mathbf{c}, \mathbf{x} \rangle + \frac{1}{2} \langle \mathbf{x}, Q\mathbf{x} \rangle = \sum_{i=1}^{n} c_i x_i + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij} x_i x_j.$$

Esimerkki 10.10.3 Tutkimme eri tyyppisten kvadraattisten funktioiden käyttäytymistä. Esitämme funktiot muodossa $f_i(\mathbf{x}) = \langle \mathbf{x}, E_i\mathbf{x} \rangle$, missä E_i on jokin seuraavista matriiseista:

$$E_1 = \begin{pmatrix} 4 & 2 \\ 2 & 2 \end{pmatrix}, \quad E_2 = \begin{pmatrix} 4 & 2 \\ 2 & 1 \end{pmatrix},$$
$$E_3 = \begin{pmatrix} 4 & 3 \\ 3 & 1 \end{pmatrix}, \quad E_4 = \begin{pmatrix} -4 & 0 \\ 0 & -1 \end{pmatrix}.$$



Kuva 10.15: *Pintakaavioina ja tasa-arvokäyrinä esitettyjä kvadraattisia funktioita* $f_i(\mathbf{x}) = \langle \mathbf{x}, E_i\mathbf{x} \rangle$, $i = 1, \ldots, 4$, *missä matriisi* E_i *on positiivisesti definiitti, positiivisesti semidefiniitti, indefiniitti tai negatiivisesti definiitti matriisi (järjestyksessä ylhäältä alas). Ylimpänä esitetty funktio* f_1 *on aidosti konveksi ja funktio* f_2 *on konveksi. Alimpana esitetty funktio* f_4 *on aidosti konkaavi. Funktio* f_3 *ei ole konveksi eikä konkaavi.*

Seuraavassa on laskettu matriisien ominaisarvot niiden definiittisyyden määrittämistä varten:

$$E_1 = \begin{pmatrix} 4 & 2 \\ 2 & 2 \end{pmatrix}, \quad \text{eig}(E_1) = \{3 - \sqrt{5}, 3 + \sqrt{5}\} \approx \{0.764, 5.24\},$$
$$E_2 = \begin{pmatrix} 4 & 2 \\ 2 & 1 \end{pmatrix}, \quad \text{eig}(E_2) = \{0, 5\},$$
$$E_3 = \begin{pmatrix} 4 & 3 \\ 3 & 1 \end{pmatrix}, \quad \text{eig}(E_3) = \left\{\frac{5 - 3\sqrt{5}}{2}, \frac{5 + 3\sqrt{5}}{2}\right\} \approx \{-0.854, 5.85\},$$
$$E_4 = \begin{pmatrix} -4 & 0 \\ 0 & -1 \end{pmatrix}, \quad \text{eig}(E_4) = \{-4, -1\}.$$

Matriisi E_1 on positiivisesti definiitti ja matriisi E_2 on positiivisesti semidefiniitti. Matriisi E_3 on puolestaan indefiniitti ja matriisi E_4 on negatiivisesti definiitti.

Kuvaan 10.15 on piirretty näiden funktioiden kuvaajat pintakaavioina ja tasa-arvokäyrien avulla. Funktiolla f_1 on yksikäsitteinen minimi ja funktiolla f_4 yksikäsitteinen maksimi.

Suuria kvadraattisia tehtäviä varten on viime aikoina kehitetty sisäpistemenetelmiä aivan kuten LP-tehtävillekin. Yleisiä epälineaarisia optimointitehtäviä voidaan puolestaan ratkoa käyttämällä apuna kvadraattisten optimointitehtävien ratkaisualgoritmeja.

10.10.3 Toistettu kvadraattinen optimointi (SQP)

Yhtälö- ja epäyhtälörajoitteita sisältävä epälineaarinen optimointitehtävä on yhtälön (10.33) muotoa. Tämän optimointitehtävän *Lagrangen funktio* on

$$\begin{aligned} L(\mathbf{x}, \mathbf{u}, \mathbf{v}) &= f(\mathbf{x}) + \langle \mathbf{u}, \mathbf{g}(\mathbf{x}) \rangle + \langle \mathbf{v}, \mathbf{h}(\mathbf{x}) \rangle \\ &= f(\mathbf{x}) + \left(\sum_{j=1}^{p} u_j g_j(\mathbf{x})\right) + \left(\sum_{j=1}^{q} v_j h_j(\mathbf{x})\right). \end{aligned} \tag{10.40}$$

Kerroinvektoreita $\mathbf{u}$ ja $\mathbf{v}$ kutsutaan optimointitehtävän *Lagrangen kertoimiksi*. Vektorin $\mathbf{u}$ alkioille pätee ehto $u_i \geq 0,\ i = 1, \ldots, p$.

Toistettu kvadraattinen optimointi (Sequential Quadratic Programming eli SQP) on menetelmä, jossa epälineaarisen tehtävän Lagrangen funktiota arvioidaan kvadraattisella mallilla. Ideana on muodostaa jono kvadraattisia tehtäviä, joiden ratkaisut lähenevät minimipistettä $\mathbf{x}^*$ ja joiden Lagrangen kertoimet lähestyvät Lagrangen kertoimia optimipisteessä koko tehtävälle. Menetelmää kutsutaan myös nimellä *Lagrangen ja Newtonin menetelmä*.

Sovellamme Lagrangen funktioon (10.40) Taylorin kehitelmää, jolloin saamme

$$\nabla L(\mathbf{x}^k + \mathbf{p}^k, \mathbf{u}^k, \mathbf{v}^k) \approx \nabla L(\mathbf{x}^k, \mathbf{u}^k, \mathbf{v}^k) + \left(\nabla \nabla^T L(\mathbf{x}^k, \mathbf{u}^k, \mathbf{v}^k)\right) \mathbf{p}^k, \tag{10.41}$$

missä ∇ tarkoittaa derivointia vektorin $\mathbf{x}$ suhteen. Jos jätämme Taylorin kehitelmästä huomiotta korkeamman kertaluvun termit ja asetamme $\nabla L(\mathbf{x}^k + \mathbf{p}^k, \mathbf{u}^k, \mathbf{v}^k) = 0$, saamme ratkaistua hakuaskeleen $\mathbf{p}^k$. Merkitsemme nyt $W_k = \nabla \nabla^T L(\mathbf{x}^k, \mathbf{u}^k, \mathbf{v}^k)$. Saamme näin yhtälön

$$W_k \mathbf{p}^k = -\nabla L(\mathbf{x}^k, \mathbf{u}^k, \mathbf{v}^k). \tag{10.42}$$

Toistetussa kvadraattisessa optimoinnissa ratkaisemme kussakin vaiheessa tehtävän

$$\operatorname*{minimi}_{\mathbf{p}} f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{p}^k \rangle + \frac{1}{2}\langle \mathbf{p}^k, W_k \mathbf{p}^k \rangle, \text{ kun } \mathbf{p}^k \in \mathbb{R}^n, \tag{10.43}$$

missä matriisi W_k on arvio Lagrangen funktion Hessen matriisille pisteessä $\mathbf{x}^k$. Kvadraattisen tehtävän rajoitteet saamme linearisoimalla rajoitefunktiot $\mathbf{g}$ ja $\mathbf{h}$ nykyisessä pisteessä $\mathbf{x}^k$:

$$\begin{cases} g_i(\mathbf{x}^k) + \langle \nabla g_i(\mathbf{x}^k), \mathbf{p}^k \rangle \le 0, & i = 1, \ldots, p, \\ h_j(\mathbf{x}^k) + \langle \nabla h_j(\mathbf{x}^k), \mathbf{p}^k \rangle = 0, & j = 1, \ldots, q. \end{cases} \tag{10.44}$$

Toistettu kvadraattinen optimointi käyttää hyväkseen Lagrangen funktion Hessen matriisia ja muistuttaa rajoitteettomien optimointitehtävien Newtonin menetelmää. Oltaessa lähellä minimipistettä voi suppenemisnopeus olla neliöllistä eli menetelmä on varsin tehokas.

Esimerkki 10.10.4 Olkoon ratkaistavana optimointitehtävä

$$\operatorname*{minimi}_{x_1, x_2} f(\mathbf{x}) = (x_1 - 2)^2 + (x_2 - 1)^2$$

rajoitteilla

$$\begin{cases} g_1(\mathbf{x}) = x_1^2/4 + x_2^2 - 1 \le 0, \\ g_2(\mathbf{x}) = x_1 - 2x_2 + 1 \le 0. \end{cases}$$

Tehtävää on havainnollistettu kuvassa 10.16. Kohdefunktion tasa-arvokäyrät on piirretty katkoviivoilla ja rajoitteita vastaava ellipsi ja suora yhtenäisellä viivalla. Kumpikin rajoite on siis epäyhtälömuotoa. Saamme tehtävän Lagrangen funktioksi

$$L(\mathbf{x}, \mathbf{u}) = (x_1 - 2)^2 + (x_2 - 1)^2 + u_1(x_1^2/4 + x_2^2 - 1) + u_2(x_1 - 2x_2 + 1).$$

Lagrangen funktion L gradienttivektori vektorin $\mathbf{x}$ suhteen on

$$\nabla L(\mathbf{x}, \mathbf{u}) = \begin{pmatrix} 2(x_1 - 2) + u_1 x_1/2 + u_2 \\ -2u_2 + 2(x_2 - 1) + 2u_1 x_2 \end{pmatrix}$$

ja Hessen matriisi on

$$\nabla \nabla^T L(\mathbf{x}, \mathbf{u}) = \begin{pmatrix} u_1/2 + 2 & 0 \\ 0 & 2u_1 + 2 \end{pmatrix}.$$

Asetamme kvadraattisen optimointitehtävän rajoitteiksi pisteessä $\mathbf{x}^k$ linearisoidut rajoitteet:

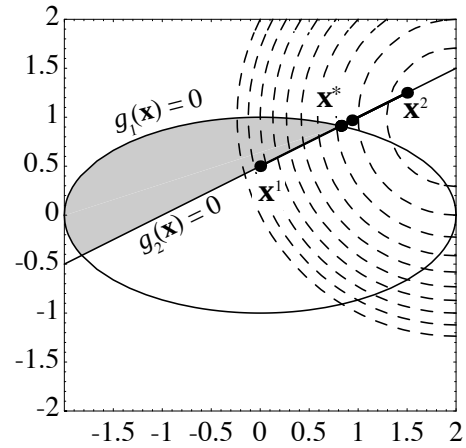
$$\begin{cases} (x_1^k)^2/4 + (x_2^k)^2 - 1 + \begin{pmatrix} x_1^k/2 \\ 2x_2^k \end{pmatrix}^T \mathbf{p}^k \le 0, \\ x_1^k - 2x_2^k + 1 + \begin{pmatrix} 1 \\ -2 \end{pmatrix}^T \mathbf{p}^k \le 0. \end{cases}$$

Valitsemme vektorin $\mathbf{x}$ alkuarvaukseksi $\mathbf{x}^1 = (0,\ 0.5)^T$ ja Lagrangen kertoimien alkuarvaukseksi $\mathbf{u}^1 = (0,\ 0)^T$. Tällöin saamme iteraatiopisteet

k	$\mathbf{x}^k$		$\mathbf{u}^k$		$\mathbf{g}(\mathbf{x}^k)$	
1	0.000	0.500	0.000	0.000	−0.750	0.000
2	1.500	1.250	1.500	1.000	1.125	0.000
3	0.938	0.969	1.500	1.422	0.158	0.000
4	0.827	0.914	1.806	1.581	0.006	0.000
5	0.823	0.911	1.846	1.594	0.000	0.000

Tässä ei pakotettu iteraatteja $\mathbf{x}^k$ pysymään käyvän alueen sisällä ja viivahakua suuntaan $\mathbf{p}^k$ ei tehty. Viidennellä iteraatiolla olemme jo hyvin lähellä tehtävän optimikohtaa. Saamme minimipisteen arvioksi $\mathbf{x}^5 \approx (0.8229, 0.9114)^T$ ja Lagrangen kertoimien arvioksi $\mathbf{u}^5 \approx (1.846, 1.594)^T$.

Kuva 10.16: *Esimerkin 10.10.4 ratkaisu SQP-menetelmällä.*



10.10.4 Sakko- ja estefunktioiden käyttö

Sakko- ja estefunktiomenetelmissä (penalty and barrier function) muutetaan rajoitteita sisältävä optimointitehtävä rajoitteettomaksi optimointitehtäväksi, jossa kohdefunktioon on lisätty sopivalla sakkotermillä painotettuna rajoitteen rikkomisesta tuleva kustannus. Siis minimoitavana on funktio

$$P(\mathbf{x}, s) = f(\mathbf{x}) + s \sum_j G(g_j(\mathbf{x})), \tag{10.45}$$

missä funktion $G : \mathbb{R} \mapsto \mathbb{R}$ avulla painotetaan rajoitefunktion arvoja. Skalaari $s > 0$ on funktion $P(\mathbf{x}, s)$ *sakkoparametri*.

Sakko- ja estefunktioiden käyttämiä muunnosfunktioita $G(g_j(\mathbf{x}))$ on seuraavanlaisia.

- *Sakkofunktiot* (eli *ulkopuoliset sakkofunktiot*):

$$\begin{cases} G(g_j(\mathbf{x})) > 0, & \text{kun } g_j(\mathbf{x}) > 0, \\ G(g_j(\mathbf{x})) = 0, & \text{kun } g_j(\mathbf{x}) \le 0. \end{cases}$$

Esimerkkinä olkoon neliöllinen sakkofunktio: $\left(\max\{0, g_j(\mathbf{x})\}\right)^2$.

- *Eksaktilla sakkofunktiolla* tarkoitetaan sellaista sakkofunktiota, jonka minimi on sama kuin rajoitteellisen tehtävän minimi kaikilla kyllin suurilla sakkoparametreilla eli kun $s > \tilde{p}$.

  Esimerkki eksaktista sakkofunktiosta on $G(g_j(\mathbf{x})) = \max\{0, g_j(\mathbf{x})\}$. Tämän eksaktin sakkofunktion ongelmana on epäsileys pisteissä, joissa rajoitteiden arvo muuttuu negatiivisesta positiiviseksi.

- *Estefunktiot* (eli *sisäpuoliset sakkofunktiot*):

$$\begin{cases} G(g_j(\mathbf{x})) & > & 0, & \text{kun } g_j(\mathbf{x}) \leq 0, \\ G(g_j(\mathbf{x})) & \to & \infty, & \text{kun } g_j(\mathbf{x}) \to 0. \end{cases}$$

  Esimerkkejä estefunktioista ovat $G(g_j(\mathbf{x})) = 1/g_j(\mathbf{x})$ sekä $G(g_j(\mathbf{x})) = -\ln(-g_j(\mathbf{x}))$ (logaritminen estefunktio).

  Estefunktiota ei voi käyttää sellaisenaan yhtälömuotoisten rajoitteiden $\mathbf{h}(\mathbf{x}) = \mathbf{0}$ kanssa, sillä estefunktio kasvaa rajatta lähestyttäessä rajoitefunktion nollakohtaa. Lisäksi menetelmä vaatii käyvän aloituspisteen löytämisen.

Esimerkki 10.10.5 Olkoon optimointitehtävänä

$$\underset{\mathbf{x}}{\text{minimi}}\ f(\mathbf{x}) = 10 - 20\left((x_1 + 1.5)^2 + 2x_2^2\right),$$

kun rajoitteena on $g(\mathbf{x}) = 2x_1^2 + x_2^2 - 1 \leq 0$.

Kuvassa 10.17 on esitetty logaritmisen estefunktion ja neliöllisen sakkofunktion käyttäytyminen kahdella eri sakkoparametrin arvolla. Estefunktion parametrin s lähestyessä nollaa jyrkkenee estefunktion reuna. Tällöin lähestytään asymptoottisesti tehtävän optimikohtaa käyvän alueen sisältä käsin. Sakkofunktion parametrin s kasvaessa lähestytään tehtävän optimikohtaa käyvän alueen ulkopuolelta käsin.

Esimerkki 10.10.6 Olkoon ratkaistavana minimointitehtävä

$$\underset{\mathbf{x}}{\text{minimi}}\ f(\mathbf{x}) = x_1^2 + x_2^2, \text{ kun } g(\mathbf{x}) = 1 - x_1 - x_2 \leq 0. \tag{10.46}$$

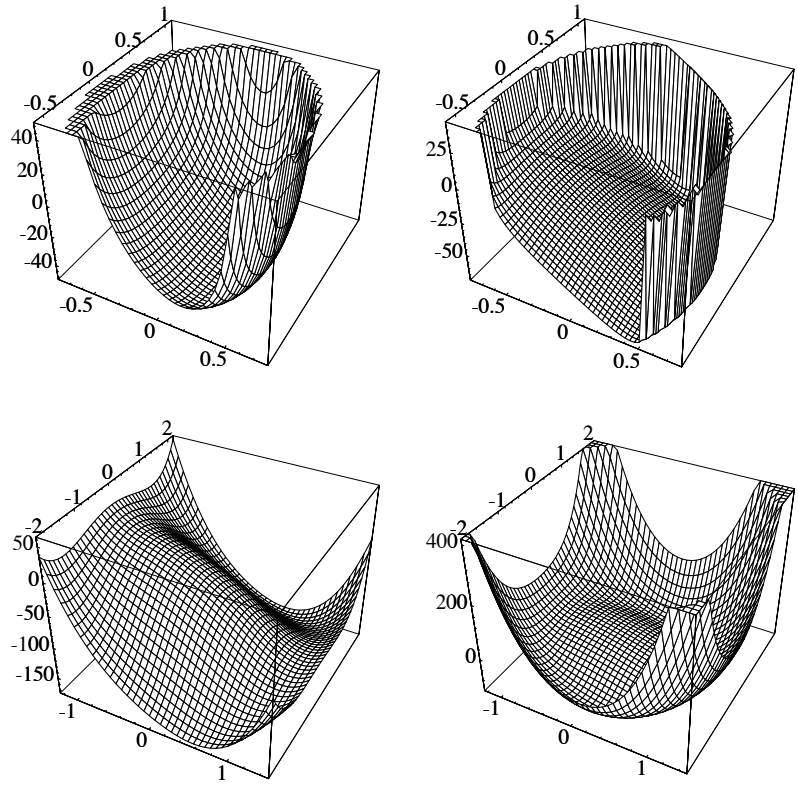
Tehtävää on havainnollistettu kuvassa 10.18. Valitsemme ratkaisumenetelmäksi neliöllisen sakkofunktion:

$$P(\mathbf{x}, s) = f(\mathbf{x}) + s\left(\max\{0, g(\mathbf{x})\}\right)^2 = x_1^2 + x_2^2 + s\left(\max\{0, 1 - x_1 - x_2\}\right)^2.$$

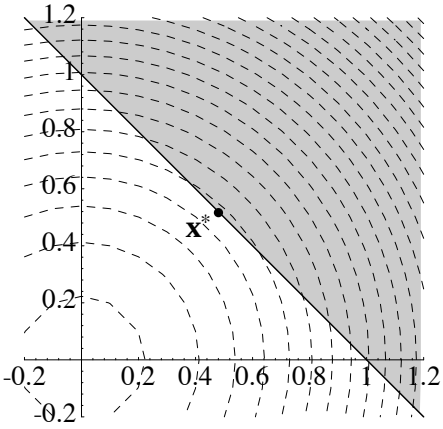
Saamme funktion $P(\mathbf{x}, s)$ kriittisen pisteen $\mathbf{x}^c$ käyvän alueen sisällä yhtälöstä

$$\nabla P(\mathbf{x}, s) = \begin{pmatrix} \dfrac{\partial P(\mathbf{x}, s)}{\partial x_1} \\ \dfrac{\partial P(\mathbf{x}, s)}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 2x_1 - 2s(1 - x_1 - x_2) \\ 2x_2 - 2s(1 - x_1 - x_2) \end{pmatrix} = \mathbf{0}.$$

Siten $\mathbf{x}^c = (s/(1 + 2s),\ s/(1 + 2s))^T$. Asettamalla $s \to +\infty$, saamme optimikohdalle arvion $\mathbf{x}^* \approx (0.5, 0.5)^T$. Kyseessä on ulkopuolinen sakkofunktio eli optimikohtaa lähestytään käyvän alueen ulkopuolelta käsin.



Kuva 10.17: *Ylempänä on käytetty logaritmista estefunktiota ja alempana neliöllistä sakkofunktiota. Estefunktion sisältävän funktion* $P(\mathbf{x}, s) = f(\mathbf{x}) - s\,\ln(-g(\mathbf{x}))$ *parametrin* s *arvot ovat* $s = 50$ *(vasemmalla) ja* $s = 10$ *(oikealla). Neliöllisen sakkofunktion sisältävän funktion* $P(\mathbf{x}, s) = f(\mathbf{x}) + s\,(\min(0, g(\mathbf{x})))^2$ *sakkoparametrin* s *arvot ovat* $s = 5$ *(vasemmalla) ja* $s = 20$ *(oikealla).*



Kuva 10.18: *Esimerkissä 10.10.6 käsiteltävä optimointitehtävä. Tehtävän käypä alue on merkitty harmaalla värillä. Kohdefunktion tasa-arvokäyrät on piirretty katkoviivoilla.*

10.11 Globaali optimointi

Epälineaaristen optimointitehtävien ratkaisualgoritmit löytävät lokaalin minimin, eikä globaalisuutta yleensä voida osoittaa, elleivät kohdefunktio ja käypä alue ole konvekseja. Yleensä ratkaisualgoritmit löytävät lähellä alkuarvausta $\mathbf{x}^1$ olevan minimipisteen käyttäen iteratiivista menetelmää

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \lambda_k \mathbf{p}^k,\ \lambda_k > 0,\ k = 1, 2, \ldots, \tag{10.47}$$

kunnes iteraatio suppenee eli esimerkiksi $\|\mathbf{p}^k\| \to 0$.

10.11.1 Globaali optimointi on vaikeaa

Globaalissa optimoinnissa etsitään optimointitehtävän pienintä paikallista minimipistettä. Tehtävä on huomattavasti vaikeampi kuin yksittäisen lokaalin minimipisteen hakeminen. Voimme tietenkin kokeilla useita eri lähtöpisteitä (esimerkiksi satunnaisesti valittuja), jolloin saamme tilastollista aineistoa globaalin optimin löytämisen varmistamiseksi.

Huomautus 10.11.1 Mikään optimointimenetelmä ei voi taata globaalin minimin löytymistä mille tahansa funktiolle. Voi olla, ettei probleemalle edes löydy luotettavaa ratkaisumenetelmää.

Globaalin optimoinnin menetelmät voidaan jakaa kahteen luokkaan: deterministisiin ja tilastollisiin menetelmiin. Monet menetelmät käyttävät deterministisiä menetelmiä tilastollisen tai heuristisen menetelmän osana. Esimerkki tästä ovat toistuvasti käynnistetyt menetelmät (multistart), joissa haetaan lokaaleja minimejä eri puolilta ratkaisuavaruutta. Ryväsmenetelmät (cluster methods) pyrkivät jakamaan hakuavaruuden pienempiin osiin, joista haetaan lokaaleja minimejä. Myös väliaritmetiikkaa (interval arithmetic) käyttävät menetelmät tarjoavat joitakin ratkaisumahdollisuuksia. Seuraavissa kappaleissa esittelemme kaksi heuristista menetelmää: geneettiset algoritmit ja jäähdytysmenetelmän.

Joskus on tärkeää löytää lokaalit minimit globaalin minimin lisäksi tai sijasta. Esimerkiksi kemiallisten yhdisteiden rakenteiden tutkimisessa halutaan löytää lokaaleja minimipisteitä eikä vain globaalia minimiä. Tällöin voi paras menetelmä olla toistuvasti käynnistetty paikallinen optimointimenetelmä, joissa etsitään minimipisteitä eri puolilta ratkaisuavaruutta.

10.11.2 Geneettiset algoritmit

Geneettiset algoritmit (GA) ovat luonnon evoluutiomekanismeja matkivia optimointimenetelmiä. GA-menetelmät eivät vaadi kohdefunktion tai rajoitefunktioiden jatkuvuutta, joten niitä voidaan käyttää perinteisillä menetelmillä vaikeasti ratkaistaviin ongelmiin. Lisäksi geneettiset algoritmit soveltuvat luonnostaan rinnakkaislaskentaan.

Geneettiset algoritmit sopivat tilanteisiin, joissa tarvoitteena on ”riittävän hyvän” optimikohdan löytäminen. Geneettisten algoritmien tehokkuuteen vaikuttaa suuresti mm. ongelman esitystapa.

Geneettisen algoritmin toimintaperiaate on seuraava:

Algoritmi 10.11.1 (Geneettinen algoritmi)

alusta populaatio $\{\mathbf{x}^i\}$, $i = 1, \ldots, l$
laske kohdefunktion arvot: $f(\mathbf{x}^i)$, $i = 1, \ldots, l$
repeat
  for $i = 1, 2, \ldots, l$
    risteytä kaksi valittua vektoria $\mathbf{x}^a$ and $\mathbf{x}^b$ vektoriksi $\mathbf{x}'$
    tuota mutaatioita: $\mathbf{x}' \rightarrow \mathbf{x}''$
    laske kohdefunktion arvo: $f_i = f(\mathbf{x}'')$
    tallenna vektori $\mathbf{x}''$ uuteen populaatioon
  end
until löytyy tarpeeksi hyvä ratkaisu

Valitsemme alkupopulaatioksi siis joukon vektoreita $\{\mathbf{x}^i\}$, $i = 1, \ldots, l$. Populaation koko säilyy samana sukupolvesta toiseen. Kaksi sopivalla mekanismilla valittua yksilöä risteytetään keskenään vektoriksi $\mathbf{x}'$. Tätä vektoria puolestaan muutetaan satunnaisen mutaation avulla. Ratkaisuehdokkaiden paremmuuden vertaamiseen käytetään kohdefunktiota $f(\mathbf{x})$ tai siitä johdettua hyvyysfunktiota, joka kuvaa alkioiden keskinäistä paremmuutta.

Geneettisiä algoritmeja ei kannata käyttää (ainakaan ainoana menetelmänä) jatkuvasti derivoituvien optimointitehtävien ratkaisemiseen: jos saatavilla on hyvä alkuarvaus, perinteiset menetelmät ovat yleensä paljon tehokkaampia. Toisaalta jos optimointitehtävässä on epäjatkuvuuksia ja paljon paikallisia minimejä, geneettiset algoritmit voivat olla käyttökelpoisia.

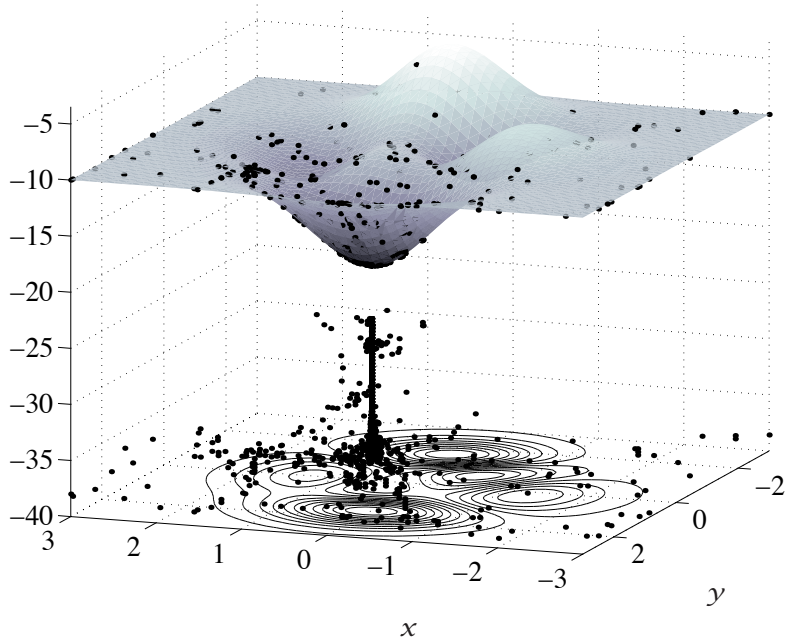
Kuva 10.19 havainnollistaa menetelmän toimintaa. Tämän tehtävän minimikohta löytyy pisteestä $\mathbf{x}^* \approx (-0.0093, 1.5814)^T$ ja minimiarvo on $f(\mathbf{x}^*) \approx 18.1$.

Rajoitteita sisältävien optimointitehtävien ratkaisu geneettisillä algoritmilla on hankalaa. Apuna voi käyttää vaikkapa sakkofunktioihin perustuvia menetelmiä, joilla rajoitteellinen tehtävä voidaan muuntaa rajoitteettomaksi. Geneettisen algoritmin toimintaa voi tehostaa käyttämällä löydettyjä ratkaisuehdokkaita lokaalien optimointimenetelmien alkuarvauksena.

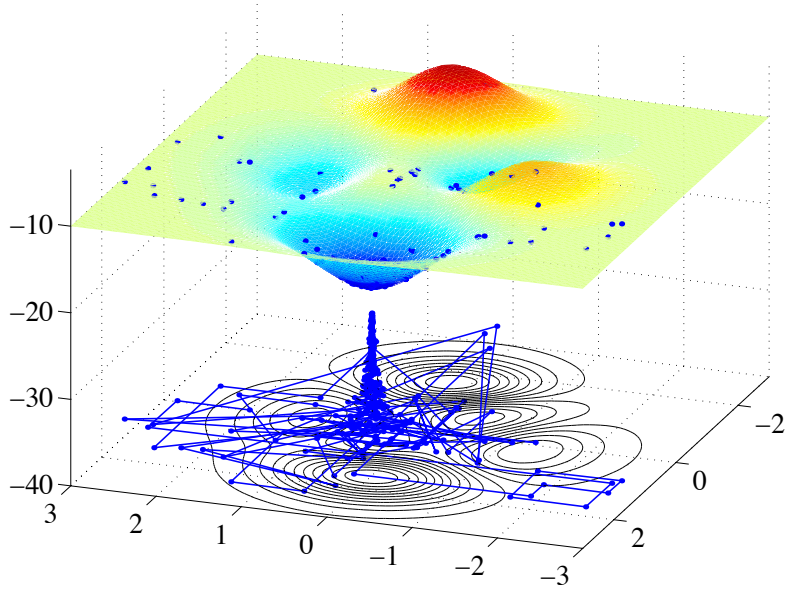
Geneettisiä algoritmeja on käytetty muilla menetelmillä vaikeasti ratkaistavissa optimointitehtävissä [Haa95]. Vahvoja kilpailijoita ovat mm. jäähdytysmenetelmät, jotka perustuvat tilastollisessa fysiikassa tutkittuihin jäähdytysilmiöihin.

10.11.3 Jäähdytysmenetelmät

Jäähdytysmenetelmät (simulated annealing eli SA, suora suomennos ”simuloitu myöstö”) ovat tilastollisesta fysiikasta (jossa käytetään *Metropoliksen*



Kuva 10.19: *Geneettisen algoritmin toiminta. Kohdefunktion käyttäytymistä kuvataan sekä pintakaaviolla että tasa-arvokäyrillä. Kuvan alareunassa on esitetty populaatiot päällekkäin. Viimeisin iteraatiokierros on ylimpänä.*



Kuva 10.20: *Jäähdytysmenetelmän toiminta. Kuvan alareunassa on esitetty menetelmän iteraatiopisteet päällekkäin murtoviivalla yhdistettyinä. Viimeisin piste on ylimpänä.*

algoritmia) vaikutteita saaneita vaikeiden optimointitehtävien ratkaisumenetelmiä.

Jäähdytysmenetelmässä generoidaan kullakin iteraatiolla uusi ratkaisuehdokas $\mathbf{x}^j$, jota verrataan edelliseen ratkaisuvektoriin $\mathbf{x}^i$. Jos uusi ehdokas antaa pienemmän arvo eli $f(\mathbf{x}^j) \leq f(\mathbf{x}^i)$, hyväksytään se uudeksi ehdokkaaksi. Jos taas uusi ehdokas on huonompi eli $f(\mathbf{x}^j) > f(\mathbf{x}^i)$, valitaan se uudeksi ratkaisuvektoriksi esimerkiksi todennäköisyydellä

$$\frac{1}{1 + \exp(\Delta f_{ij}/T)} \approx \exp(-\Delta f_{ij}/T), \tag{10.48}$$

missä $\Delta f_{ij} = f(\mathbf{x}^j) - f(\mathbf{x}^i)$ on minimoitavan kohdefunktion arvojen erotus. Skalaari T on parametri, jota kutsutaan jäähdytysmenetelmän (keinotekoiseksi) lämpötilaksi. Tässä on kyseessä *Boltzmannin jäähdytys*, joka liittyy läheisesti tilastollisen fysiikan menetelmiin.

Jäähdytysmenetelmä vaatii, että tehtävälle voidaan muodostaa alhaalta rajoitettu energiafunktio, jonka minimoiva tilavektori on tehtävän ratkaisu. Alussa annetaan systeemin käyttäytyä melko vapaasti eli ratkaistava systeemi muuttaa usein tilavektoriaan ja valitsee myös energianormin mielessä korkeammalla tasolla olevia tiloja. Tämän jälkeen systeemiä ryhdytään jäähdyttämään, jolloin konfiguraatio voi yhä harvemmin vaihtaa tilaansa ”ylämäkeen”.

Sopivalla jäähdytysstrategialla systeemi löytää alussa globaalisti hyvän ratkaisun ja jäähdytyksen edetessä tilavektorin eri komponentit saavat lopulliset optimaaliset arvonsa. Kuva 10.20 havainnollistaa jäähdytysmenetelmän toimintaa.

10.12 Yhteenveto

Optimointitehtäville ei ole yleispätevää ratkaisumenetelmää, joten menetelmä on valittava tehtävän tyypin mukaan. Tämä luku käsittelee seuraavien optimointitehtävien ratkaisua:

- yksiulotteinen optimointi
  - kultaisen leikkauksen menetelmä
  - toistettu kvadraattinen interpolointi
  - Newtonin menetelmä
- rajoitteeton moniulotteinen optimointi
  - Newtonin menetelmä
  - sekanttimenetelmät
  - liittogradienttimenetelmät
  - suorat menetelmät (polytooppihaku)
- epälineaariset pienimmän neliösumman tehtävät

  - Gaussin ja Newtonin menetelmä
  - Levenbergin ja Marquardtin menetelmä
- lineaarinen optimointi
  - tehtävän muodostus, simplex-algoritmi ja sisäpistemenetelmät
- kvadraattiset optimointitehtävät
  - tehtävän muodostus
- rajoitteelliset epälineaariset tehtävät
  - toistettu kvadraattinen optimointi (SQP)
  - este- ja sakkofunktioiden käyttö
- globaali optimointi
  - geneettiset algoritmit
  - jäähdytysmenetelmät.

10.13 Ohjelmistoja

Matlabin *Optimization Toolbox* [CBG99] sisältää joukon optimointirutiineja, joita voi käyttää mm. eri menetelmien testaamiseen ja erilaisiin kokeiluihin.

Mallinnuskieli GAMS on kehitetty erityisesti taloustieteen optimointitehtävien ratkaisemiseen. GAMSin mallikirjastosta löytyy esimerkkejä erityyppisten tehtävien ratkaisemisesta. GAMS-ohjelmistosta (General Algebraic Modeling System) kerrotaan suomenkielisessa pikaoppaassa [Haa01] ja ohjelmiston käsikirjassa [BKM88].

GAMS ei ole ainoa optimointiin soveltuva mallinnuskieli. Vaihtoehtoja ovat mm. AMPL ja LINGO. Lisäksi tilastollisista ohjelmista SAS ja Splus löytyy rutiineja mm. pienimmän neliösumman tehtävien ratkaisemiseen.

Kaupalliset aliohjelmakirjastot IMSL ja NAG sisältävät testattuja ja tehokkaita Fortran-rutiineja, joiden käyttö on melko vaivatonta käsikirjojen ja esimerkkiohjelmien avulla.

Eräillä ohjelmistoilla on rajoituksia ongelman koon suhteen. Jos tehtävässä on epälineaarinen kohdefunktio ja rajoitteet, tehtävän ratkaiseminen tehokkaasti voi vaatia paljon työtä. Oikean menetelmän valintaan kannattaa kiinnittää huomiota. Epäjatkuvuuksista selviävät lähinnä vain tilastollisiin menetelmiin perustuvat ohjelmistot, jotka vievät runsaasti CPU-aikaa.

Uusia jäähdytysmenetelmiin tai geneettisiin algoritmeihin perustuvia ohjelmia on jonkin verran jo saatavilla. Joissakin tehtävissä näillä menetelmillä on saavutettu hyviä tuloksia. Esimerkiksi Harwell-aliohjelmakirjastosta löytyy geneettisten algoritmien toteutus.

Taulukko 10.1 luettelee erityyppisistä optimointitehtävistä käytettyjä lyhenteitä. Taulukko 10.2 kuvaa eri optimointiohjelmistojen ominaisuuksia. Ohjelmistot on jaoteltu käyttötavan mukaan aliohjelmakirjastoiksi (useimmat ovat Fortran-pohjaisia) ja erillisiksi sovellusohjelmiksi. Osa ohjelmista tarjoaa lisäksi interaktiivisen käyttöliittymän. Fortran-aliohjelmakirjastoja voi käyttää myös C-kielisestä ohjelmasta [Haa98]. Taulukossa on lueteltu lähinnä ohjelmistoja, jotka ovat saatavissa Unix-ympäristöön.

Taulukko 10.1: *Optimointitehtävien tyyppien lyhenteitä.*

Tyyppi	*Selitys*
LP	Lineaarinen optimointi
QP	Kvadraattinen optimointi
NET	Verkko- ja kuljetustehtävät
MIP	Sekalukuoptimointi, osa muuttujista saa kokonaislukuarvoja
NLP	Optimointitehtävän kohdefunktio tai rajoitteet epälineaarisia
LSQ	Lineaariset pienimmän neliösumman tehtävät
NLSQ	Epälineaariset pienimmän neliösumman tehtävät

Taulukko 10.2: *Eräitä optimointiin soveltuvia ohjelmistoja.*

Ohjelmisto	*Tehtävät*	*Käyttöliittymä*
GAMS	LP, MIP, NLP, NLSQ	Komentotiedosto
Minos	LP, NLP	Komentotied. ja aliohjelmakirjasto
Lapack	LSQ	Aliohjelmakirjasto
IMSL	LP, NLP, LSQ, NLSQ	Aliohjelmakirjasto
NAG	LP, MIP, NLP, LSQ, NLSQ	Aliohjelmakirjasto
Harwell	LP, MIP, NLP, NLSQ	Aliohjelmakirjasto
Minpack	NLSQ	Aliohjelmakirjasto
FSQP	NLP	Aliohjelmakirjasto
Matlab	LP, NLP, LSQ, NLSQ	Interaktiivinen
Mathematica	LP, NLP, LSQ, NLSQ	Interaktiivinen
Maple	LP, NLP, LSQ	Interaktiivinen
SAS/OR	LP, NET, NLP	Interaktiivinen ja komentotiedosto
SAS/STAT	LSQ, NLSQ	Interaktiivinen ja komentotiedosto
Splus	LSQ	Interaktiivinen
IDL	NLP, LSQ	Interaktiivinen

Internetistä löytyy useita optimointia käsitteleviä www-palveluita. Tasoikkaimpia ovat

- *NEOS Guide to Optimization Software* osoitteessa `http://www.mcs.anl.gov/home/otc/Guide/guide.html`
- *GAMS (Guide to Available Mathematical Software)* osoitteessa `http://gams.nist.gov`

- *Decision Tree for Optimization Software* osoitteessa `http://plato.la.asu.edu/guide.html`

NEOS-palvelusta löytyy mm. optimointiohjelmistojen katsauksen [MW93] ajantasainen versio.

Netlib-ohjelmistoarkistosta löytyy optimointitehtävien ratkaisemiseen käyttökelpoisia ohjelmistoja hakemistoista `minpack`, `matlab`, `opt/praxis`, `opt/subplex`, `opt/ve08` ja `slatec`. Netlib löytyy www-osoitteesta `http://www.netlib.org`.

Taulukossa 10.3 on lueteltu Netlibin hakemistosta `toms` (ACM Transactions on Mathematical Software) löytyviä TOMS-algoritmeja.

10.14 Lisätietoja

CSC:n oppaassa *Optimointitehtävien ratkaiseminen* [Haa95] kerrotaan tässä esitettyä laajemmin optimoinnin teoriasta ja käytännöstä. CSC:n opas *Matemaattiset ohjelmistot* [Haa98] kuvaa eräitä yleisesti käytettyjä matemaattisia ohjelmistoja.

Numeerisia optimointimenetelmiä käsitteleviä hyvätasoisia oppikirjoja ovat *Practical Methods of Optimization* [Fle87], *Practical Optimization* [GMW81], *Nonlinear Programming* [BSS93], *Optimointi* [Mie98],q *Nonlinear Programming* [Ber96], *Linear and Nonlinear Programming* [NS96] sekä *Numerical Optimization* [NW99].

Epälineaaristen yhtälöryhmien ja pienimmän neliösumman tehtävien ratkaisumenetelmiä esitellään teoksessa *Numerical Methods for Unconstrained Optimization and Nonlinear Equations* [DS83]. Teos *Iterative Methods for Optimization* [Kel99] analysoi tehokkaiden optimointialgoritmien ominaisuuksia. Matlabin *Optimization Toolbox* on monipuolinen kokoelma optimointialgoritmeja pieniin ja keskisuuriin tehtäviin [CBG99].

Geneettisistä algoritmeista kerrotaan mm. Mitchellin tiiviissä katsaustyyppisessä teoksessa [Mit98]. Hyödyllisiä ovat myös Hollandin, Goldbergin, Davisin, Michalewiczin ja Schwefelin teokset [Hol92, Gol89, Dav91, Mic96, MF99, Sch95]. CSC:n julkaisemasta Fortran 90/95 -oppikirjasta löytyy differentiaalievoluution toteutus [HRR01].

Kirjat *Simulated Annealing and Boltzmann Machines* [AK89] ja *Simulated Annealing: Theory and Applications* [vLA88] ovat hyviä johdatuksia jäähdytysmenetelmien ominaisuuksiin ja taustaan. Kombinatoristen optimointitehtävien ratkaisua jäähdytysmenetelmillä on analysoitu teoksessa *The Annealing Algorithm* [OvG89]. Teoksessa *Genetic algorithms and simulated annealing* [Dav87] on esitelty jäähdytysmenetelmiä ja geneettisiä algoritmeja.

Taulukko 10.3: *Optimointitehtävien ratkaisemiseen soveltuvia Netlibin hakemistosta* `toms` *löytyviä algoritmeja.*

Nro	*Kuvaus*
500	Sekanttimenetelmä (yleistetty liittogradienttimenetelmä)
520	Verkkotehtävät
548	Sijoittelutehtävät (assignment problems)
552	Rajoitteellinen L_1-arviointi, *simplex*-menetelmä
557	Monitavoiteoptimointi
558	Sijoittelutehtävät (facility location)
559	Kvadraattinen optimointi
562	Lyhimmän reitin tehtävät
573	Epälineaariset pienimmän neliösumman tehtävät
587	Kvadraattinen optimointi
608	Kvadraattiset sijoittelutehtävät
611	Rajoitteeton minimointi, luottamusaluemenetelmä
630	Rajoitteeton minimointi, liittogradientti- ja sekanttimenetelmä
632	Selkärepputehtävät (knapsack)
659	Globaali optimointi (satunnaisotanta)
667	Globaali optimointi (SIGMA)
702	Katkaistu Newtonin menetelmä (TNPACK)
711	Rinnakkaistettu katkaistu Newtonin menetelmä
733	Optimisäätötehtävät
745	Fortran 90 -versio algoritmista 630
746	Automaattinen derivointi
750	Suuret kauppamatkustajan ongelmat
754	Kvadraattiset sijoittelutehtävät
755	Algoritmien automaattinen derivointi
765	Suuret rajoitteettomat optimointitehtävät
768	Epälineaariset yhtälöt ja pienimmän neliösumman tehtävät
769	Kvadraattiset sijoittelutehtävät
774	Laatikkorajoitteisten optimointitehtävien generointi
778	Suuret laatikkorajoitteiset optimointitehtävät
811	Epäsileiden optimointitehtävien ratkaisu (NDA)
813	Rajoitteiset optimointitehtävät (SPG)

11 Yhteenveto

Esittelemme tässä kirjassa matemaattisiin tehtäviin sopivia numeerisia menetelmiä. Useimmat menetelmät sopivat sopivat sellaisenaan jonkin matemaattisen ongelman ratkaisemiseen, mutta usein menetelmiä pitää yhdistellä tai kehittää paremmin ongelmaa vastaavaksi. Yritämme tuoda myös esille yhteyksiä eri menetelmien välillä.

11.1 Numeerisen menetelmän valinta

Numeerinen menetelmä pitää valita tehtävän koon ja vaikeuden mukaisesti. Pienille ja helpoille tehtäville kannattaa käyttää mahdollisimman helppoa, yksinkertaista ja luotettavaa numeerista menetelmää. Jos ison laskentatehtävän ratkaisuajan pitää olla mahdollisimman lyhyt, joudumme valitsemaan monimutkaisempia ja vaikeammin hallittavia numeerisia menetelmiä. Näiden toteuttaminen tietokoneella ei ole aina suoraviivaista ja helppoa.

Käytännön mallinnustehtävissä joudumme usein yhdistämään ideoita erilaisista numeerisista menetelmistä. Tämän kirjan menetelmiä voi pitää numeerisen mallin rakennuspalikoina. Vaativissa sovelluksissa joudumme tekemään tutkimustyötä numeeristen menetelmien valinnassa, soveltamisessa ja menetelmän parametrien virittämisessä.

Kun toteutamme numeerisia menetelmiä tietokoneella, joudumme tekemään kompromisseja ohjelmiston tehokkuuden, luotettavuuden ja ohjelmiston toteutukseen käytettävissä olevan ajan välillä. Menetelmät tulisi valita ja ohjelmisto testata siten, että voimme olla varmoja ohjelman luotettavasta toiminnasta koko ohjelmiston käyttöalueella.

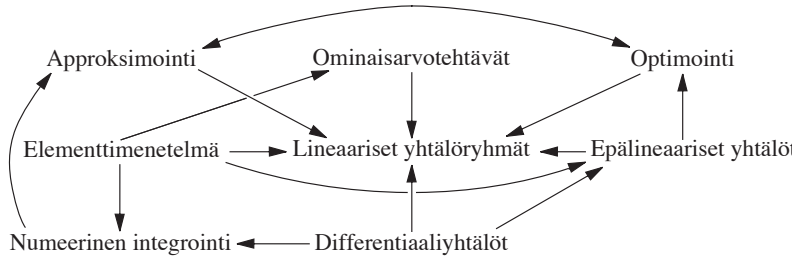
Suosittelemme, että numeerisia menetelmiä toteuttava ohjelma rakennetaan prototyyppityöskentelyn avulla. Tällöin rakennetaan ensiksi matemaattisia aliohjelmakirjastoja sekä interaktiivisia ratkaisuympäristöjä hyväksi käyttäen ohjelman prototyyppi, jonka ei tarvitse olla erityisen nopea. Vasta tämän jälkeen voi ohjelman suoritusaikaa ryhtyä optimoimaan. Suosittelemme matemaattisten aliohjelmakirjastojen hyvin testattujen rutiinien käyttöä. Jos lopputuotteen tulee olla siirrettävä ja riippumaton aliohjelmakirjastoista, voi osan näistä menetelmistä toteuttaa itse.

11.2 Yhteyksiä menetelmien välillä

Tässä kirjassa esiteltyjen numeeristen menetelmien välillä on monenlaisia yhteyksiä. Esimerkiksi melkein kaikki kirjassa esitetyt numeeriset menetelmät pelkistyvät tavalla tai toisella lineaaristen yhtälöryhmien ratkaisuksi:

- Funktioiden approksimointi pienimmän neliösumman menetelmällä: sovituksen kertoimet saadaan lineaarisesta yhtälöryhmästä.
- Epälineaarisen yhtälöryhmän tai monen muuttujan optimointitehtävän ratkaiseminen Newtonin menetelmällä tai sekanttimenetelmällä: lineaarisia yhtälöryhmiä ratkaistaan toistuvasti.
- Kun ratkaisemme tavallisten differentiaaliyhtälöiden alkuarvotehtäviä implisiittisillä menetelmillä, joudumme ratkaisemaan lineaarisen yhtälöryhmän.
- Käytettäessä differenssi- tai elementtimenetelmää reuna-arvotehtävien tai osittaisdifferentiaaliyhtälöiden ratkaisemiseen, laskennallinen tehtävä on jälleen lineaarinen yhtälöryhmä.
- Monet ominaisarvotehtävien ratkaisumenetelmät edellyttävät lineaaristen yhtälöryhmien ratkaisua tai matriisin hajotelmien laskemista.

Kertauksen vuoksi esitämme oheisessa kuvassa 11.1 kaavion menetelmien välisistä yhteyksistä.



Kuva 11.1: *Yhteyksiä menetelmien välillä.*

Toinen tässä kirjassa esitettyjä menetelmiä yhdistävä tekijä on elementtimenetelmä, jossa tarvitaan melkein kaikkia tässä teoksessa esitettyjä tekniikoita:

- Elementtiverkon generoinnissa saatetaan käyttää optimointialgoritmeja verkon säännöllisyyden lisäämiseksi.
- Elementtiverkon jakamisessa osiin tehtävän rinnakkaistusta varten ratkaistaan eräissä menetelmissä ominaisarvotehtäviä.
- Elementtimenetelmässä osittaisdifferentiaaliyhtälön ratkaisua approksimoidaan esimerkiksi polynomimuotoisen kantafunktiojoukon lineaarikombinaationa. Ratkaisun jälkikäsittelyssä ja visualisoinnissa tarvitaan kantafunktioiden arvojen interpolointia, koska ratkaisu on määritelty kantafunktoiden summana.

- Ajasta riippuvan osittaisdifferentiaaliyhtälön ratkaisuun käytetään tavallisten differentiaaliyhtälöiden yhteydessä esitettyjä aikaintegrointimenetelmiä.
- Epälineaarisen osittaisdifferentiaaliyhtälön ratkaisu palautetaan lineaariseksi tehtäväksi joko linearisoimalla tehtävä oletetun ratkaisun lähellä tai käyttämällä jotakin epälineaarisen yhtälöryhmän ratkaisumenetelmää.
- Yhtälöryhmän kerroinmatriisin kokoamisessa käytetään numeerista integrointia sekä interpolointia.
- Monen menetelmän laskennallinen pullonkaula on ison harvan lineaarisen yhtälöryhmän ratkaisu joko suorilla menetelmillä tai pohjustetuilla iteratiivisillä menetelmillä.
- Alkuperäinen osittaisdifferentiaaliyhtälö voi olla annettu ominaisarvoyhtälön muodossa. Diskretoinnin jälkeen tämä palautuu algebralliseksi ominaisarvotehtäväksi.
- Osittaisdifferentiaaliyhtälö voi olla osa optimointialgoritmia, jossa optimoitavan kohdefunktion laskeminen edellyttää joka kerta uuden osittaisdifferentiaaliyhtälön ratkaisua.

11.3 Lisätiedon lähteitä

Teos *Mathematical Models in the Applied Sciences* [Fow97] on melko vaativa katsaus matemaattisiin malleihin. Teos *Computational Differential Equations* [EEHJ96] on johdatus mallintamiseen elementtimenetelmän avulla. Lineaarialgebran käsikirjaksi sopii teos *Matrix Computations* [GvL96]. Teos *Introduction to Algorithms* [CLR90] on hyvä johdatus tietokoneohjelmissa käytettyjen algoritmien analysointiin.

Myös CSC:n julkaisemista oppikirjoista ja oppaista löytyy lisätietoa numeerisista menetelmistä ja ohjelmoinnista:

- *Matemaattiset ohjelmistot* [Haa98]
- *Datan käsittely* [Kar01]
- *Elementtimenetelmä virtauslaskennassa* [HJ94]
- *Optimointitehtävien ratkaiseminen* [Haa95]
- *Fortran 90/95* [HRR01]
- *Rinnakkaisohjelmointi MPI:llä* [HM01].

Liitteet

A Matemaattisia taustatietoja

A.1 Yleisiä merkintätapoja

Pyrimme käyttämään vakiintuneita matemaattisia merkintätapoja. Luettelemme tärkeimmät käytetyt symbolit sivulla 11. Merkitsemme skalaariarvoisia muuttujia (yleensä reaalilukuja) kirjaimilla x, y sekä vektoriarvoisia suureita lihavoiduilla symboleilla $\mathbf{x}, \mathbf{y}$. Matriiseille olemme varanneet isot kirjaimet A. Symboli $\mathbb{R}$ tarkoittaa reaalilukujen joukkoa ja symboli $\mathbb{R}^n$ n-ulotteista reaaliavaruutta.

Iteratiivisissa algoritmeissa merkitsemme iteraatteja symboleilla x_i (skalaari $\in \mathbb{R}$), $\mathbf{x}^i$ (vektori $\in \mathbb{R}^n$) ja A_i (matriisi). Viittaamme vektorin alkioihin myös merkinnällä x_i, mutta tämä selviää aina asiayhteydestä. Elementtimenetelmää käsittelevässä luvussa 8 joudumme käyttämään vektorin iteraatille $\mathbf{x}_i^k$ kahta indeksiä kahden sisäkkäisen iteraatiotason vuoksi.

Nollavektori on $\mathbf{0} = (0, 0, \ldots, 0)^T$. Vektori $\mathbf{e}^i = (0, 0, ..., 0, 1, 0, ..., 0)$ on standardikannan i:s yksikkövektori:

$$e^i_j = \begin{cases} 1, & \text{kun } j = i, \\ 0, & \text{kun } j \neq i. \end{cases}$$

Desimaali- ja binääriluvuissa käytämme erottimena pistettä: $3.14159\ldots$ ja $(0.1011)_2$. Tämä mahdollistaa numerolistojen selkeän käytön. Esimerkiksi vektorien alkiot esitetään muodossa $(1.2, -1.2)^T$.

Joukkoja merkitään tavanomaisella notaatiolla: $\mathcal{F} = \{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{b}\}$. Joukkoon $\mathcal{F}$ siis kuuluvat ne avaruuden $\mathbb{R}^n$ pisteet, jotka toteuttavat yhtälön $A\mathbf{x} = \mathbf{b}$.

Vektorille $\mathbf{x}$ esitetyt epäyhtälöt (niin sanotut *laatikkorajoitteet*)

$$\mathbf{x}^l \leq \mathbf{x} \leq \mathbf{x}^u$$

tarkoittavat, että vektorien $\mathbf{x}^l$, $\mathbf{x}$ ja $\mathbf{x}^u$ alkioille pätee $x_i^l \leq x_i \leq x_i^u$, $i = 1, \ldots, n$.

Eri yhteyksissä käytämme merkintää "y on suuruusluokkaa $\mathcal{O}(g(n))$". Tällä tarkoitetaan tässä oppaassa, että $y \in \mathcal{O}(g(n))$, joka on määritelty seuraavasti:

Määritelmä A.1.1 (Suuruusluokka $\mathcal{O}(g(n))$) Olkoon funktio $g(n)$ annettu. Merkintä $\mathcal{O}(g(n))$ tarkoittaa funktioiden $f(n)$ joukkoa

$$\begin{aligned}\mathcal{O}(g(n)) \;=\; \{f(n) \mid &\text{on olemassa positiiviset vakiot } c \text{ ja } N \text{ s.e.}\\ &0 \le f(n) \le cg(n) \text{ kaikilla } n \ge N\}\end{aligned}$$

A.2 Peruskäsitteitä

Raja-arvon ja jatkuvuuden käsitteet ovat keskeisiä matemaattisessa analyysissä, joten esitämme aluksi muutaman määritelmän.

Määritelmä A.2.1 (Raja-arvo) Olkoon f reaalilukujoukossa X määritelty funktio. Funktiolla f on raja-arvo L pisteessä x_0 eli

$$\lim_{x \to x_0} f(x) = L,$$

jos jokaista reaalilukua $\epsilon > 0$ kohden on olemassa reaaliluku $\delta > 0$ siten että $|f(x) - L| < \epsilon$ aina kun $x \in X$ ja $0 < |x - x_0| < \delta$.

Määritelmä A.2.2 (Jatkuvuus) Olkoon funktio f määritelty reaalilukujoukossa X ja piste $x_0 \in X$. Funktio f on jatkuva pisteessä x_0, jos $\lim_{x \to x_0} f(x) = f(x_0)$. Funktio f on jatkuva joukossa X, jos se on jatkuva kaikissa joukon pisteissä.

Määritelmä A.2.3 (Derivoituvuus) Olkoon funktio f määritelty avoimella, pisteen x_0 sisältävällä välillä. Funktio f on derivoituva pisteessä x_0, jos raja-arvo

$$f'(x_0) = \lim_{x \to x_0} \frac{f(x) - f(x_0)}{x - x_0}$$

on olemassa. Lukua $f'(x_0)$ kutsutaan funktion f derivaataksi pisteessä x_0. Jos funktiolla on derivaatta kaikissa joukon X pisteissä, on funktio derivoituva joukossa X.

Lause A.2.1 (Väliarvolause) Olkoon funktio $f(x)$ jatkuva äärellisellä suljetulla välillä $[a, b]$ ja derivoituva avoimella välillä (a, b). Tällöin on olemassa piste $c \in (a, b)$, jolle pätee

$$f(a) - f(b) = f'(c)(a - b).$$

Usean muuttujan skalaariarvoiselle funktiolle $f : \mathbb{R}^n \mapsto \mathbb{R}$ voidaan muodostaa erotusosamäärän raja-arvo

$$\lim_{h \to 0} \frac{f(\mathbf{x} + h\mathbf{e}^i) - f(\mathbf{x})}{h}, \tag{A.1}$$

missä $\mathbf{e}^i$ on i:s yksikkövektori. Jos raja-arvo on olemassa, sitä kutsutaan funktion f *osittaisderivaataksi* pisteessä $\mathbf{x}$ koordinaatin i suhteen ja merkitään

$$\frac{\partial f(\mathbf{x})}{\partial x_i}. \tag{A.2}$$

Jos funktiolla $f(\mathbf{x})$ on pisteessä $\mathbf{x}$ osittaisderivaatat kaikkien koordinaattien suhteen, f on *derivoituva* pisteessä $\mathbf{x}$.

Esimerkki A.2.1 Tarkastelemme kahden muuttujan funktiota $f(x_1, x_2) = x_1^2(1 - x_1x_2)$. Laskemme funktion osittaisderivaatan muuttujan x_1 suhteen:

$$\frac{\partial f}{\partial x_1} = \frac{\partial}{\partial x_1}\left(x_1^2(1 - x_1x_2)\right) = 2x_1(1 - x_1x_2) + x_1^2(-x_2) = 2x_1 - 3x_1^2x_2.$$

Derivoimisoperaattori ∇ eli *nabla* on aukikirjoitettuna

$$\nabla = \begin{pmatrix} \dfrac{\partial}{\partial x_1} \\ \vdots \\ \dfrac{\partial}{\partial x_n} \end{pmatrix}. \tag{A.3}$$

Funktion f osittaisderivaatoista muodostettua vektoria eli *gradienttia* pisteessä $\mathbf{x}$ merkitään notaatiolla

$$\nabla f(\mathbf{x}) = \begin{pmatrix} \dfrac{\partial f(\mathbf{x})}{\partial x_1} \\ \vdots \\ \dfrac{\partial f(\mathbf{x})}{\partial x_n} \end{pmatrix}. \tag{A.4}$$

Jos $\nabla f(\mathbf{x})$ on jatkuva ja olemassa kaikissa pisteissä $\mathbf{x} \in \mathbb{R}^n$, sanotaan funktion f olevan *jatkuvasti derivoituva* eli *sileä* (smooth). Muussa tapauksessa f on *epäsileä* (non-smooth).

Esimerkki A.2.2 Tarkastelemme esimerkin A.2.1 funktiota $f(\mathbf{x}) = x_1^2(1 - x_1x_2)$. Laskemme gradienttivektorin $\nabla f(\mathbf{x})$:

$$\nabla f(\mathbf{x}) = \begin{pmatrix} 2x_1 - 3x_1^2x_2 \\ -x_1^3 \end{pmatrix}.$$

Skalaariarvoisen funktion $f : \mathbb{R}^n \mapsto \mathbb{R}$ toisen kertaluvun derivaatoista muodostetusta *Hessen matriisista* (engl. Hessian; Ludwig Otto Hesse, 1811–74) käytetään merkintöjä

$$H(\mathbf{x}) = \nabla\nabla^T f(\mathbf{x}) = \begin{pmatrix} \dfrac{\partial^2 f(\mathbf{x})}{\partial^2 x_1} & \cdots & \dfrac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_n} & \cdots & \dfrac{\partial^2 f(\mathbf{x})}{\partial^2 x_n} \end{pmatrix}.$$

Näemme määritelmästä, että Hessen matriisi on symmetrinen.

Vektoriarvoiselle funktiolle $\mathbf{f} : \mathbb{R}^n \mapsto \mathbb{R}^m$ käytetään osittaisderivaattojen matriisista eli *Jacobin matriisista* merkintää

$$J(\mathbf{x}) = \begin{pmatrix} \nabla^T f_1(\mathbf{x}) \\ \vdots \\ \nabla^T f_m(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_m(\mathbf{x})}{\partial x_n} \end{pmatrix}.$$

Funktioiden derivaattoja voi yrittää laskea symbolinkäsittelyjärjestelmillä kuten Mathematica tai Maple. Tulosten järkevyys on tietenkin syytä tarkistaa.

A.3 Vektorit ja matriisit

Vektoria $\mathbf{x} \in \mathbb{R}^n$ merkitään alkioidensa $x_i, i = 1, \ldots, n$ avulla seuraavasti:

$$\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix},$$

missä kukin alkio x_i kuuluu reaalilukujen joukkoon $\mathbb{R}$. Esimerkiksi vektoriarvoinen funktio $\mathbf{f} : \mathbb{R}^n \mapsto \mathbb{R}^m$ on auki kirjoitettuna

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} f_1(x_1, \ldots, x_n) \\ \vdots \\ f_m(x_1, \ldots, x_n) \end{pmatrix},$$

missä kukin komponenttifunktio $f_i, i = 1, \ldots, m$ on kuvaus $\mathbb{R}^n \mapsto \mathbb{R}$.

Matriisin A alkioita merkitään pikkukirjainnotaatiolla a_{ij} eli

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}.$$

Kahden samanmuotoisen matriisin summalla $C = A + B$ tarkoitetaan matriisia, jossa summattavien matriisien alkiot on laskettu yhteen: $c_{ij} = a_{ij} + b_{ij}$. Sama pätee vektoreille:

$$\mathbf{x} + \mathbf{y} = \begin{pmatrix} x_1 + y_1 \\ \vdots \\ x_n + y_n \end{pmatrix}.$$

Matriisien A (dimensio $l \times m$) ja B (dimensio $m \times n$) tuloa merkitään AB ja se on määritelty

$$AB = \begin{pmatrix} \sum_{i=1}^{m} a_{1i}b_{i1} & \cdots & \sum_{i=1}^{m} a_{1i}b_{in} \\ \vdots & \ddots & \vdots \\ \sum_{i=1}^{m} a_{li}b_{i1} & \cdots & \sum_{i=1}^{m} a_{li}b_{in} \end{pmatrix}. \tag{A.5}$$

Matriisin A ja vektorin $\mathbf{x}$ tulo $A\mathbf{x}$ määritellään vastaavasti.

Skalaarilla kertomisessa kerrotaan matriisin tai vektorin alkiot yksitellen. Esimerkiksi kun $\mathbf{x} \in \mathbb{R}^n$ ja $c \in \mathbb{R}$ saadaan

$$c\mathbf{x} = \begin{pmatrix} cx_1 \\ cx_2 \\ \vdots \\ cx_n \end{pmatrix}. \tag{A.6}$$

Vektorin tai matriisin transpoosissa vaihdetaan rivi- ja sarakealkioiden paikat. Transpoosia merkitään yläindeksillä T: A^T, $\mathbf{x}^T$ jne.

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix}, \qquad A^T = \begin{pmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \\ a_{13} & a_{23} \end{pmatrix}.$$

Vektoreille pätee vastaavasti esimerkiksi

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \quad \mathbf{x}^T = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \end{pmatrix}.$$

Reaaliarvoisten matriisien transpoosille on voimassa $(AB)^T = B^T A^T$.

Vektoreille ja matriiseille on määritelty myös hermitointi, jota merkitään yläindeksillä H. Hermitointi on kompleksikonjugoinnin ja transponoinnin yhdistelmä: $\mathbf{x}^H = \overline{\mathbf{x}}^T$ ja $A^H = \overline{A}^T$.

Matriiseille pätevät osittelulait

$$\begin{aligned} (A+B)C &= AC + BC, \\ A(B+C) &= AB + AC \end{aligned} \tag{A.7}$$

sekä liitäntälaki

$$A(BC) = (AB)C, \tag{A.8}$$

mutta vaihdantalaki ei ole voimassa eli yleensä on $AB \neq BA$.

Matriisien tai vektorien yhtäsuuruudella tarkoitetaan, että nämä ovat alkioittain yhtä suuria. Yhtälömerkintä $A\mathbf{x} = \mathbf{b}$, missä A on $m \times n$ -matriisi ja $\mathbf{b}$ dimensioltaan m, tulkitaan siis:

$$\sum_{i=1}^{n} a_{ji}x_i = a_{j1}x_1 + a_{j2}x_2 + \cdots + a_{jn}x_n = b_j, \text{ missä } j = 1, \ldots, m.$$

Numeerisilla matriiseilla laskemiseen on kehitetty monia ohjelmistoja, esimerkiksi helppokäyttöinen Matlab-ohjelmisto ja Fortran-kielinen Lapack-aliohjelmakirjasto. Fortran 90/95 -ohjelmointikieli sisältää myös monipuolisen taulukkojen käsittelyn ja standardifunktiot perusmatriisioperaatiohin.

Määritelmä A.3.1 (Aliavaruus) Joukko $S \subset \mathbb{R}^n$ on avaruuden $\mathbb{R}^n$ *aliavaruus*, jos pätee $c_1\,\mathbf{x} + c_2\,\mathbf{y} \in S$ kaikilla $\mathbf{x}, \mathbf{y} \in S$ ja kaikilla $c_1, c_2 \in \mathbb{R}$. Äärellinen joukko vektoreita $\mathcal{X} = \{\mathbf{x}^1, \ldots, \mathbf{x}^m\}$, $\mathbf{x}^i \in \mathbb{R}^n$, *virittää aliavaruuden*

$$S(\mathcal{X}) = \{\ \mathbf{y} \in \mathbb{R}^n \ \mid\ \mathbf{y} = \textstyle\sum_{i=1}^{m} c_i\,\mathbf{x}^i,\ c_i \in \mathbb{R},\ \mathbf{x}^i \in \mathbb{R}^n\ \}. \tag{A.9}$$

Siis aliavaruus $S(\mathcal{X})$ muodostuu niistä avaruuden $\mathbb{R}^n$ pisteistä, jotka voidaan esittää joukon $\mathcal{X}$ vektorien lineaarikombinaatioina.

Määritelmä A.3.2 (Vektorien lineaarinen riippumattomuus) Joukkoon $\{\mathbf{x}^i,\ i = 1, \ldots, n\}$ kuuluvien vektorien sanotaan olevan *lineaarisesti riippumattomia* eli *vapaita*, mikäli

$$\sum_{i=1}^{n} c_i\mathbf{x}^i = \mathbf{0} \qquad \Longrightarrow \qquad c_i = 0, \quad i = 1, \ldots, n. \tag{A.10}$$

Täten vektorijoukko $\{\mathbf{x}^i\}$ on *lineaarisesti riippuva* eli *sidottu*, mikäli nollavektori $\mathbf{0}$ voidaan esittää vektorien $\mathbf{x}^i$ lineaarikombinaationa siten, että ainakin yksi skalaarikerroin $c_i \neq 0$.

Esimerkki A.3.1 Tarkastelemme matriisia A, joka muodostuu kolmesta sarakevektorista $\mathbf{x}^i$:

$$A = \begin{pmatrix} \mathbf{x}^1 & \mathbf{x}^2 & \mathbf{x}^3 \end{pmatrix} = \begin{pmatrix} 1 & 4 & 2 \\ -1 & 3 & 5 \\ 2 & 2 & -2 \end{pmatrix}.$$

Nyt esimerkiksi $-2\mathbf{x}^1 + \mathbf{x}^2 - \mathbf{x}^3 = \mathbf{0}$ eli vektorijoukko $\{\mathbf{x}^i\}$ on lineaarisesti riippuva.

Määritelmä A.3.3 (Ominaisarvot ja ominaisvektorit) Neliömatriisin A *ominaisarvoksi* (eigenvalue) kutsutaan sellaista lukua λ, jolle pätee

$$A\mathbf{z} = \lambda\mathbf{z} \tag{A.11}$$

jollekin vektorille $\mathbf{z} \neq \mathbf{0}$. Tällaista vektoria $\mathbf{z}$ kutsutaan *ominaisvektoriksi* (eigenvector). Jatkossa käytämme ominaisarvojen λ_i joukosta merkintää

$$\mathrm{eig}(A) = \{\ \lambda_1, \ldots, \lambda_m\ \}.$$

Esimerkki A.3.2 Esimerkin A.3.1 matriisin A ominaisarvoiksi λ_i ja ominaisvektoreiksi $\mathbf{z}^i$ saadaan

$$\begin{aligned} \mathrm{eig}(A) &= \{\ -3,\ \ 0,\ \ 5\ \}, \\ \begin{pmatrix} \mathbf{z}^1 & \mathbf{z}^2 & \mathbf{z}^3 \end{pmatrix} &= \begin{pmatrix} 4 & 2 & 4 \\ -11 & -1 & 3 \\ 14 & 1 & 2 \end{pmatrix}. \end{aligned}$$

Ominaisarvoa nolla vastaavat vektorit $\{\mathbf{z} \in \mathbb{R}^3 \mid A\mathbf{z} = \mathbf{0}\}$. Näitä kutsutaan myös matriisin A ytimen virittämiksi vektoreiksi.

Määritelmä A.3.4 (Matriisin säännöllisyys ja singulaarisuus) Neliömatriisin A sanotaan olevan *säännöllinen*, jos lineaarisella yhtälöryhmällä $A\mathbf{x} = \mathbf{b}$ on yksikäsitteinen ratkaisu. Jos näin ei ole, matriisi A on *singulaarinen*.

Säännöllisyys on yhtäpitävää mm. seuraavien asioiden kanssa:

- Matriisilla A on käänteismatriisi A^{-1}.
- Yhtälöryhmän $A\mathbf{x} = \mathbf{0}$ ainoa ratkaisu on $\mathbf{x} = \mathbf{0}$.
- Matriisin A vaaka- ja pystyrivit ovat lineaarisesti riippumattomia.
- Matriisin A determinantti ei ole 0.

Määritelmä A.3.5 (Matriisin definiittisyys) Matriisi A on *positiivisesti semidefiniitti*, jos ja vain jos

$$\langle \mathbf{y}, A\mathbf{y} \rangle \geq 0 \text{ kaikilla } \mathbf{y} \in \mathbb{R}^n. \tag{A.12}$$

Symmetristen matriisien tapauksessa yhtäpitävä ehto on

$$\mathrm{eig}(A) \geq 0, \tag{A.13}$$

eli symmetrisen positiivisesti semidefiniitin matriisin ominaisarvojen tulee olla positiivisia tai nollia. Vastaavasti *positiivisesti definiitille* matriisille pätee

$$\langle \mathbf{y}, A\mathbf{y} \rangle > 0 \text{ kaikilla } \mathbf{y} \in \mathbb{R}^n,\ \mathbf{y} \neq \mathbf{0}\,. \tag{A.14}$$

Negatiivisesti definiitille matriisille pätee puolestaan

$$\langle \mathbf{y}, A\mathbf{y} \rangle < 0 \text{ kaikilla } \mathbf{y} \in \mathbb{R}^n,\ \mathbf{y} \neq \mathbf{0}. \tag{A.15}$$

Indefiniitti matriisi ei ole positiivisesti tai negatiivisesti (semi)definiitti. Indefiniitillä symmetrisellä matriisilla on vähintään yksi positiivinen ja vähintään yksi negatiivinen ominaisarvo.

A.4 Matriisien hajotelmien käyttömahdollisuuksia

Yleinen matriisi A on *tiheä*, jos suurin osa sen alkioista on nollasta poikkeavia. Vastaavasti *harva matriisi* koostuu pääosin nollista, ja nollasta poikkeavia alkioita on ehkä vain muutama prosentti. Matriisin harvuutta voidaan hyödyntää sen tallentamisessa tietokoneen muistiin sekä esimerkiksi lineaarista yhtälöryhmää ratkaistaessa.

Lävistäjämatriisissa (diagonaalimatriisissa) on nollasta poikkeavia alkioita vain lävistäjällä eli matriisi D on muotoa

$$D = \operatorname{diag}(d_1, d_2, \ldots, d_n) = \begin{pmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & d_n \end{pmatrix}.$$

Yksikkömatriisin I (identiteettimatriisin) lävistäjällä on ykkösiä ja kaikkialla muualla nollia:

$$I = \begin{pmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{pmatrix}.$$

Tällöin pätee $AI = IA = A$, missä A ja I ovat $n \times n$ -matriiseja. *Permutaatiomatriisi* saadaan yksikkömatriisista vaihtamalla rivejä (tai sarakkeita) keskenään.

Yläkolmiomatriisi on muotoa

$$\begin{pmatrix} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \\ & & & & \times \end{pmatrix}.$$

Alakolmiomatriisi määritellään vastaavasti. *Symmetriselle* reaalialkioiselle matriisille A pätee $a_{ij} = a_{ji}$ eli $A = A^T$. *Ortogonaalimatriisille* Q pätee

$$Q^T Q = Q Q^T = I$$

eli käänteismatriisina on $Q^{-1} = Q^T$.

Matriisi A on *hermiittinen* jos

$$A^H = A$$

ja *unitaarinen* jos

$$A^H A = I,$$

jolloin $A^{-1} = A^H$. Jos

$$A^H A = A A^H,$$

matriisia A sanotaan *normaaliksi*.

Symmetrinen positiivisesti definiitti matriisi B voidaan kirjoittaa muotoon $B = LDL^T$, missä L on alakolmiomatriisi (lävistäjäelementit ykkösiä) ja D on lävistäjämatriisi, jonka alkiot ovat positiivisia. Matriisille B voidaan tehdä esimerkiksi Matlabilla *Choleskyn hajotelma* $B = LL^T$, missä L on alakolmiomatriisi.

Esimerkki A.4.1 Olkoon matriisi B määritelty seuraavasti:

$$B = \begin{pmatrix} 2 & 3 & 1 \\ 3 & 6 & 1 \\ 1 & 1 & 6 \end{pmatrix}.$$

Matriisille B saadaan Choleskyn hajotelma $B = LL^T$, missä

$$L \approx \begin{pmatrix} 1.414 & 0 & 0 \\ 2.121 & 1.225 & 0 \\ 0.707 & -0.408 & 2.309 \end{pmatrix}.$$

Choleskyn hajotelman olemassaolosta voidaan päätellä, että matriisi B on positiivisesti definiitti.

Yleiselle matriisille A käyttökelpoinen hajotelma on QR-hajotelma

$$QA = \begin{pmatrix} R \\ 0 \end{pmatrix}, \tag{A.16}$$

missä Q on ortogonaalinen matriisi ja R on yläkolmiomatriisi. LU-hajotelma määritellään kaavalla

$$A = LU, \tag{A.17}$$

missä L on alakolmiomatriisi (lävistäjäelementit ykkösiä) ja U yläkolmiomatriisi. Singulaariarvohajotelma puolestaan määritellään kaavalla

$$A = U\Sigma V^T, \tag{A.18}$$

missä U ja V ovat ortogonaalisia matriiseja ja lävistäjämatriisi Σ sisältää matriisin A *singulaariarvot*.

Esimerkki A.4.2 Olkoon

$$A = \begin{pmatrix} 1 & 4 & 2 \\ -1 & 3 & 5 \\ 2 & 2 & -2 \end{pmatrix}.$$

Seuraavassa esitetään Matlabin avulla lasketut eri tyyppiset hajotelmat matriisille A:

$$A = PLU = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1.000 & 0 & 0 \\ -0.500 & 1.000 & 0 \\ 0.500 & 0.750 & 1.000 \end{pmatrix} \begin{pmatrix} 2 & 2 & -2 \\ 0 & 4 & 4 \\ 0 & 0 & 0 \end{pmatrix}$$

$$A = QR \approx \begin{pmatrix} -0.408 & -0.636 & -0.655 \\ 0.408 & -0.769 & 0.492 \\ -0.817 & -0.067 & 0.574 \end{pmatrix} \begin{pmatrix} -2.450 & -2.041 & 2.858 \\ 0 & -4.983 & -4.983 \\ 0 & 0 & 0.000 \end{pmatrix}$$

$$A = U\Sigma V^T \approx \begin{pmatrix} -0.5786 & -0.4855 & -0.6554 \\ -0.8148 & 0.3075 & 0.4915 \\ 0.0371 & -0.8184 & 0.5735 \end{pmatrix} \times \begin{pmatrix} 7.0842 & 0 & 0 \\ 0 & 4.2206 & 0 \\ 0 & 0 & 0.0000 \end{pmatrix} \begin{pmatrix} 0.0438 & -0.5757 & 0.8165 \\ -0.6613 & -0.6293 & -0.4082 \\ -0.7489 & 0.5220 & 0.4082 \end{pmatrix}^T$$

Edellä lasketussa LU-hajotelmassa on P permutaatiomatriisi, jonka avulla matriisien rivit tulevat oikeaan järjestykseen. Lisäksi QR-hajotelma on laskettu yhtälön (A.16) määritelmästä poikkeavalla tavalla, mutta koska matriisi Q on ortogonaalinen eli $Q^{-1} = Q^T$, on esitysmuotoon (A.16) helppo siirtyä.

Näemme singulaarihajotelmasta $U\Sigma V^T$, että kolmas singulaariarvo on nolla (katso matriisia Σ), joten matriisi A on singulaarinen. Singulaarisuus näkyy myös QR-hajotelman R-matriisista sekä LU-hajotelman matriisista U.

Matriisien hajotelmien laskemista käsitellään tarkemmin ja ohjelmistosuosituksia annetaan luvussa 3.

A.5 Vektoriavaruus ja normi

Joukko olioita muodostaa *vektoriavaruuden* V, jos niiden välille on määritelty laskutoimitus (voidaan merkitään vaikkapa $+$-merkillä) ja seuraavat ehdot toteutuvat kaikilla alkioilla $\mathbf{x}, \mathbf{y}, \mathbf{z} \in V$ (luvut α, β ovat kerroinkunnan alkioita, esimerkiksi reaalilukuja):

1. Joukko V on suljettu laskutoimituksen $+$ suhteen: $\mathbf{x} + \mathbf{y} \in V$, ja summaalkio on yksikäsitteinen.
2. Laskutoimitus $+$ on vaihdannainen ja liitännäinen: $\mathbf{x} + \mathbf{y} = \mathbf{y} + \mathbf{x}$, $(\mathbf{x} + \mathbf{y}) + \mathbf{z} = \mathbf{x} + (\mathbf{y} + \mathbf{z})$
3. Joukossa V on *nolla-alkio* $\mathbf{0}$, joka on neutraali laskutoimituksen suhteen: $\mathbf{x} + \mathbf{0} = \mathbf{x}$.
4. Jokaisella alkiolla $\mathbf{x}$ on *vasta-alkio* $-\mathbf{x}$: $\mathbf{x} + (-\mathbf{x}) = \mathbf{0}$.
5. Jokaisen alkion monikerrat $a\mathbf{x}$ kuuluvat joukkoon V.

6. $\alpha(\mathbf{x}+\mathbf{y}) = \alpha\mathbf{x}+\alpha\mathbf{y}$, $(\alpha+\beta)\mathbf{x} = \alpha\mathbf{x}+\beta\mathbf{x}$ ja $(\alpha\beta)\mathbf{x} = \alpha(\beta\mathbf{x})$. Huomaa, että keskimmäisessä yhtälössä $+$-merkki esiintyy kahdessa eri merkityksessä.

7. $1\mathbf{x} = \mathbf{x}$.

Esimerkki A.5.1 Tutuin esimerkki vektoriavaruuksista lienee n-komponenttisten reaalisten vektorien joukko $\mathbb{R}^n$, kun laskutoimitukset määritellään komponenteittain ja nolla-alkioksi sovitaan nollavektori $\mathbf{0} = (0, 0, ...0)$. Vasta-alkio saadaan vaihtamalla vektorin komponentit vastaluvuikseen.

Välillä $[a, b]$ määritellyistä reaaliarvoisista funktioista $f : [a, b] \mapsto \mathbb{R}$ saadaan aikaiseksi ääretönulotteinen vektoriavaruus, kun laskutoimitus $+$ ja reaaliluvulla kertominen tulkitaan pisteittäin tapahtuvaksi. Nolla-alkiona toimii *nollafunktio* $0(t) \equiv 0$ kaikilla $t \in [a, b]$. Jokaiselle alkioille f löytyy vasta-alkio yksinkertaisesti kertomalla f luvulla -1.

Vektoriavaruus voi siis periaatteessa koostua mistä hyvänsä olioista, kunhan niiden välillä pätevät yllä luetellut ehdot. Vektoriavaruuden alkioiden vertailua varten on määriteltävä erikseen *normi*, joka liittää jokaiseen avaruuden alkioon yksikäsitteisen positiivisen reaaliluvun, joka kuvastaa alkioiden suuruutta. Normi on siis kuvaus $V \mapsto [0, \infty)$, ja sen merkkinä käytetään kaksinkertaisia pystyviivoja: $\|\mathbf{x}\|$. Jos halutaan korostaa itse normia eikä niinkään alkiota, jonka normi lasketaan, voidaan normin sisään merkitä vain piste: $\|\cdot\|$.

Jotta $\|\cdot\|$ olisi normi, sen täytyy toteuttaa seuraavat kolme ehtoa:

1. $\|\mathbf{x}\| = 0 \iff \mathbf{x} = \mathbf{0}$
2. $\|\alpha\mathbf{x}\| = |\alpha|\|\mathbf{x}\|$
3. $\|\mathbf{x}+\mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$

Mikäli ehdot ovat muuten voimassa, mutta ensimmäisessä kohdassa pätee vain implikaatio vasemmalle (nolla-alkion normi on nolla), kyseessä on *seminormi*. Tällöin siis avaruudessa on nolla-alkion $\mathbf{0}$ lisäksi muitakin alkiota, joiden koko on normilla $\|\cdot\|$ mitattuna 0. Käytännön kannalta tämä tarkoittaa, että tehtäviin ei välttämättä saada yksikäsitteisiä ratkaisuja, koska normi ei pysty erottelemaan avaruuden alkiota riittävän tarkasti.

Olkoon V alueessa Ω määritellyistä mitallisista funktioista koostuva avaruus. Tällöin voidaan ottaa käyttöön seuraavat normit:

- L_1-normi (usein lyhyesti myös 1-normi):

$$\|f\|_{L_1} = \|f\|_1 = \int_\Omega |f(\mathbf{x})|\, d\mathbf{x}.$$

- L_2-normi (tai vain 2-normi):

$$\|f\|_{L_2} = \|f\|_2 = \left[\int_\Omega (f(\mathbf{x}))^2 \, d\mathbf{x}\right]^{1/2}.$$

- L_∞-normi (∞-normi, Tšebyšev-normi) :

$$\|f\|_{L_\infty} = \|f\|_\infty = \sup_{\mathbf{x}\in\Omega} |f(\mathbf{x})|.$$

Normin avulla voidaan vertailla kahden funktion samankaltaisuutta. Funktioiden f ja g välille voidaan määritellä etäisyys $d(f,g) = \|f-g\|$. Funktio f on lähellä funktiota g normin $\|\cdot\|$ mielessä, jos $d(f,g) \approx 0$.

Äärettömien lukujonojen f_i, missä $i = -\infty, \ldots, \infty$, normina käytetään l_p-normeja, jotka määritellään samaan tapaan L_p-normien kanssa:

$$\|f\|_p = \left(\sum_{l=-\infty}^{\infty} |f_i|^p\right)^{1/p}.$$

Äärellisulotteisissa avaruuksissa $\mathbb{R}^n$ käytetään p-normeja, jotka ovat l_p-normien erikoistapaus:

$$\|\mathbf{x}\|_p = \left(\sum_{l=1}^{n} |x_i|^p\right)^{1/p}.$$

Myös p-normeja käytettäessä valitaan yleensä $p = 1, 2$ tai ∞, joille pätee $\|x\|_1 = \sum |x_i|$, $\|x\|_2 = \sqrt{\sum x_i^2}$ ja $\|x\|_\infty = \max_i |x_i|$. Tapausta $p = 2$ kutsutaan myös *euklidiseksi normiksi.*

Esimerkki A.5.2 Olkoon vektori $\mathbf{x} = (1, 2, -1)^T$. Tällöin vektorin $\mathbf{x}$ euklidinen normi on $\|\mathbf{x}\|_2 = \sqrt{6}$. Vektorin voi myös normeerata yksikkövektoriksi:

$$\hat{\mathbf{x}} = \frac{\mathbf{x}}{\|\mathbf{x}\|} = \frac{1}{\sqrt{6}}(1, 2, -1)^T = (\frac{1}{\sqrt{6}}, \frac{2}{\sqrt{6}}, \frac{1}{\sqrt{6}})^T.$$

Tällöin pätee $\|\hat{\mathbf{x}}\|_2 = 1$.

Vastaavasti vektorin $\mathbf{x}$ 1-normi on $\|\mathbf{x}\|_1 = 4$ ja ∞-normi $\|\mathbf{x}\|_\infty = 2$.

Matriisin normi voidaan määritellä vektorinormin avulla seuraavasti:

$$\|A\| = \max_{\|x\|=1} \|Ax\|, \tag{A.19}$$

jolloin sanotaan, että kyseinen matriisinormi on oikealla puolella käytetyn vektorinormin *indusoima.* Myös muunlaisia matriisinormeja on olemassa. Matriisin normi kertoo miten pitkäksi matriisi voi kuvata ykkösen pituisen vektorin. Matriisinormin arvo riippuu käytetystä vektorinormista. Jos

vektorinormina käytetään p-normia, merkitään vastaavaa matriisinormia $\|A\|_p$:llä.

Yleisimpien matriisinormien lausekkeet ovat

$$\begin{aligned} \|A\|_1 &= \max_j \sum_i |a_{ij}|, \\ \|A\|_2 &= \sigma_1 = \sqrt{\lambda(A^T A)}, \\ \|A\|_\infty &= \max_i \sum_j |a_{ij}|, \end{aligned} \tag{A.20}$$

missä σ_1 on matriisin A suurin *singulaariarvo* eli matriisin $A^T A$ isoimman ominaisarvon neliöjuuri (katso myös sivua 307). Symmetriselle matriisille 2-normi on sama kuin itseisarvoiltaan suurin ominaisarvo.

Joskus käytetään myös Frobeniuksen normia, jossa lasketaan neliöjuuri kaikkien matriisin alkioiden neliöiden summasta:

$$\|A\|_F = \sqrt{\sum_j \sum_i |a_{ij}|^2}. \tag{A.21}$$

Huomaa, että Frobeniuksen normia ei voida johtaa mistään vektorinormista.

A.6 Sisätulo

Vektoriavaruudessa V voidaan määritellä *sisätulo*, joka liittää kaikkiin alkiopareihin reaaliluvun. Sisätulo voi olla myös kompleksiarvoinen, mutta tämän kirjan tarpeisiin riittää tuntea reaalinen sisätulo. Mikä tahansa kuvaus $V \times V \mapsto \mathbb{R}$ ei kelpaa sisätuloksi, vaan sen on toteutettava seuraavat ehdot:

1. Symmetria: $\langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{y}, \mathbf{x} \rangle$.
2. Ei-negatiivisuus: $\langle \mathbf{x}, \mathbf{x} \rangle \geq 0 \quad \forall \mathbf{x} \in V$.
3. Definiittisyys: $\langle \mathbf{x}, \mathbf{x} \rangle = 0 \iff \mathbf{x} = 0$.
4. Lineaarisuus: $\langle \alpha\mathbf{x} + \beta\mathbf{y}, \mathbf{z} \rangle = \alpha\langle \mathbf{x}, \mathbf{z} \rangle + \beta\langle \mathbf{y}, \mathbf{z} \rangle$.

Jokainen sisätulo toteuttaa lisäksi *Cauchyn ja Schwarzin epäyhtälön*

$$\langle \mathbf{x}, \mathbf{y} \rangle^2 \leq \langle \mathbf{x}, \mathbf{x} \rangle \langle \mathbf{y}, \mathbf{y} \rangle.$$

Sisätuloon on aina mahdollista liittää myös normi: $\|\mathbf{x}\| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$. Käänteinen ei päde yleisesti: kaikkia normeja ei ole mahdollista johtaa jonkin sisätulon avulla. Esimerkiksi normit L_1 ja L_∞ ovat tällaisia.

Kaksi alkiota ovat keskenään *ortogonaaliset*, jos niiden välisen sisätulon arvo on 0. Joukko alkioita $\{\mathbf{x}_i\}$ muodostaa *ortogonaalisen systeemin*, jos alkiot

ovat pareittain ortogonaalisia. Ortogonaalinen systeemi on *ortonormaali*, jos

$$\langle \mathbf{x}_i, \mathbf{x}_j \rangle = \delta_i^j = \begin{cases} 0, & i \neq j, \\ 1, & i = j. \end{cases}$$

Mielivaltaisen vektoriavaruuden kanta on aina mahdollista ortonormalisoida. Yksinkertaisin tapa on *Gramin ja Schmidtin menettely*, joka ei kuitenkaan ole numeerisesti paras mahdollinen. Jos tunnetaan kanta $\{\mathbf{x}_i\}$, niin valitaan uuden kannan $\{\mathbf{y}_i\}$ ensimmäiseksi alkioksi $\mathbf{y}_1 = \mathbf{x}_1 / \|\mathbf{x}_1\|$ (normi on nyt nimenomaan ortogonaalisuuden määrittelevään sisätuloon perustuva normi). Uuden kannan loput alkiot lasketaan yksi kerrallaan muodostamalla ensin aina apuvektori

$$\tilde{\mathbf{y}}_i = \mathbf{x}_i - \sum_{j=1}^{i-1} \langle \mathbf{x}_i, \mathbf{y}_j \rangle \mathbf{y}_j,$$

joka sitten normalisoidaan: $\mathbf{y}_i = \tilde{\mathbf{y}}_i / \|\tilde{\mathbf{y}}_i\|$.

Avaruuden $\mathbb{R}^n$ vektoreiden välille määritellään tavallisesti sisätulo kertomalla vastinkomponentit keskenään ja laskemalla tulojen summa:

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{n} x_i y_i$$

Tämän sisätulon indusoima normi on edellä esitelty 2-normi eli euklidinen normi.

Samalla periaatteella funktioiden muodostamassa avaruudessa voidaan kirjoittaa sisätulo

$$\langle f, g \rangle = \int_\Omega f(\mathbf{x}) g(\mathbf{x}) \, d\mathbf{x}, \tag{A.22}$$

ja vastaava normi on tietysti L_2-normi:

$$\|f\|_{L_2} = \sqrt{\int_\Omega f^2(\mathbf{x}) \, d\mathbf{x}}$$

Hiukan yleisemmässä muodossa sisätuloon lisätään vielä positiivinen painofunktio $w(\mathbf{x}) \geq 0$, jolloin $\langle f, g \rangle = \int w(\mathbf{x}) f(\mathbf{x}) g(\mathbf{x}) \, d\mathbf{x}$.

Esimerkki A.6.1 Yksinkertaisimman esimerkin ylempänä mainitun sisätulon suhteen ortonormaalista systeemistä tarjoavat n-ulotteisen avaruuden kantavektorit

$$e_1 = [1\,0\,0 \ldots 0]^T,\ e_2 = [0\,1\,0 \ldots 0]^T,\ \ldots,\ e_n = [0\,0\,0 \ldots 1]^T.$$

Myös funktiot voivat muodostaa ortonormaaleja systeemeitä. Reaalilukuvälillä $[-1, 1]$ sopivasti skaalatut Legendren polynomit $\sqrt{n + 1/2} P_n(x)$ toteuttavat ehdon

$$\begin{aligned} \langle \sqrt{n+1/2} P_n, \sqrt{m+1/2} P_m \rangle &= \int_{-1}^{1} \sqrt{n+1/2} \sqrt{m+1/2} P_n(x) P_m(x) \, dx \\ &= \delta_n^m. \end{aligned}$$

A.7 Suppenemisnopeus

Iteratiivisten menetelmien ominaisuuksiin kuuluu *suppenemisen kertaluku* (order of convergence). Jos on olemassa vakiot $p \leq 1$, $c \geq 0$ ja $N \geq 1$ siten, että jono x_k suppenee kohti pistettä x^* ja kaikilla $k \geq N$ pätee

$$|x_{k+1} - x^*| \leq c\,|x_k - x^*|^p, \tag{A.23}$$

sanotaan jonon x_k suppenemisen olevan kertalukua p. Jos $p = 1$ ja $c < 1$, on suppeneminen *lineaarista*. Jos $p > 1$, on suppeneminen *superlineaarista*. Suppeneminen on *neliöllistä* , kun $p = 2$.

Esimerkki A.7.1 Funktion nollakohdan etsimisessä käytettävän puolitushaun (katso sivua 192) suppenemisnopeus on helposti selvitettävissä. Etsintävälin päätepisteet ovat a_k ja b_k. Uusi päätepiste x_k saadaan kaavasta $x_k = (a_k + b_k)/2$. Olkoon virhearvio $e_k = b_k - a_k \approx x^* - x_k$, missä piste x^* on tarkka ratkaisu. Saamme

$$e_{k+1} = \frac{1}{2}e_k = \frac{1}{2}(b-a)2^{-(k-1)}.$$

Puolitushaku suppenee lineaarisesti, sillä jokaisella iteraatiolla hakuväli $[a_k, b_k]$ puolittuu.

Esimerkki A.7.2 Newtonin menetelmä soveltuu skalaarifunktion minimikohdan etsintään (katso sivua 193). Menetelmän iteraatiokaava on

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}.$$

Oletamme, että funktio $f(x)$ on kahdesti jatkuvasti derivoituva ratkaisupisteen x^* läheisyydessä. Lisäksi oletamme, että pisteessä x^* pätee $f'(x^*) \neq 0$. Tällöin pätee

$$e_{k+1} = \frac{-\frac{1}{2}f''(\xi_1)}{f'(\xi_2)}\,e_k^2.$$

Tässä ξ_1 ja ξ_2 ovat kaksi pistettä valittuina sopivasti nollakohdan x^* läheltä. Siten Newtonin menetelmän suppeneminen on neliöllistä juuren lähellä: merkitsevien numeroiden lukumäärä kaksinkertaistuu jokaisella iteraatioaskeeleella.

B Liukulukuesitys

B.1 Reaaliluvut ja liukuluvut

Tietokoneessa ei voi esittää reaalilukuja tarkasti, vaan joudumme käyttämään *liukulukuaritmetiikkaa*. Liukulukuaritmetiikka perustuu yleensä lukujen binääriseen esitysmuotoon. Esimerkiksi desimaaliesityksessä annetun luvun 13 binääriesitys on

$$13 = (13)_{10} = 1 \cdot 10^1 + 3 \cdot 10^0 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = (1101)_2.$$

Käytämme alaindeksiä 2 erottamaan desimaali- ja binääriesitykset toisistaan. Binääriesitys koostuu jonosta numeroita 0 ja 1; yhtä binääriesityksen alkiota kutsutaan bitiksi. Bittien arvoja vastaa tietokoneen laitteistotasolla esimerkiksi kaksi eri tasoista jännitettä.

Myös murtoluvut voidaan esittää binäärilukuina, esimerkiksi

$$0.5 = (0.5)_{10} = 5 \cdot 10^{-1} = 2^{-1} = (0.1)_2.$$

Tässä erotamme pisteellä kantaluvun negatiivisia potensseja vastaavat kertoimet. Kaikkia lukuja ei voi esittää äärellisen mittaisina binäärilukuina. Esimerkiksi

$$\begin{aligned} 0.1 = 10^{-1} &= 2^{-4} + 2^{-5} + 2^{-8} + 2^{-9} + 2^{-12} + 2^{-13} + \cdots \\ &= (0.0001100110011\ldots)_2. \end{aligned}$$

Tällöin joudumme pyöristämään tai katkaisemaan liukulukuesityksen. Jos käytämme kymmenen termin binääriesitystä, saamme katkaisemalla eli jättämällä pois ylimääräiset termit

$$(0.0001100110011\ldots)_2 \approx (0.000110011)_2 = 0.099609375.$$

B.2 Liukulukuesityksen virheet

Määrittelemme virheen alkuperäisen reaaliluvun x ja tietokoneen esitysmuodon (yleensä liukulukuesityksen) x_f avulla seuraavasti:

$$|x_f - x| \quad \textit{absoluuttinen virhe}, \tag{B.1}$$

$$\frac{|x_f - x|}{|x|} \quad \textit{suhteellinen virhe}. \tag{B.2}$$

Esimerkki B.2.1 Reaaliluvun 0.1 tapauksessa binääriesityksen absoluuttinen virhe on

$$|(0.000110011)_2 - 0.1| = 0.000390625$$

ja suhteellinen virhe

$$\frac{|(0.000110011)_2 - 0.1|}{|0.1|} = 0.00390625.$$

Voimme käyttää kumpaakin määritelmää virheen testaamiseen. Esimerkiksi jos $|a - a_f| < 0.5 \times 10^{-s}|a|$, liukuluku a_f approksimoi a:ta s:llä desimaalilla. Lisäksi pätee

$$x_f = x(1 + \text{suhteellinen virhe}).$$

Voimme lisäksi määritellä funktion *häiriöalttiuden* C seuraavasti:

$$C = \frac{\left| \dfrac{f(x_f) - f(x)}{f(x)} \right|}{\left| \dfrac{x_f - x}{x} \right|}.$$

Häiriöalttius kuvaa, millä tekijällä suhteelliset virheet funktion argumentissa kasvavat funktion arvoa laskettaessa. Häiriöalttius riippuu pisteestä x.

Voimme esittää kaavat virheiden kasaantumisesta eri peruslaskutoimituksille $(+, -, \times, /)$. Olkoon ϵ_a luvun a liukulukuesityksen absoluuttisen virheen yläraja ($|a - a_f| \leq \epsilon_a$).

- Yhteenlaskussa absoluuttinen virhe on

$$\epsilon_{a+b} \leq \epsilon_a + \epsilon_b.$$

- Vähennyslaskussa pätee

$$\epsilon_{a-b} \leq \epsilon_a + \epsilon_b.$$

- Kertolaskussa pätee

$$\epsilon_{ab} \leq \epsilon_a |a_f| + \epsilon_b |b_f| + \epsilon_a \epsilon_b.$$

- Jakolaskussa pätee

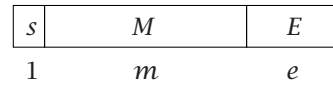
$$\epsilon_{a/b} \leq \frac{\epsilon_b |a_f| + \epsilon_a |b_f|}{b|b_f|} \approx \frac{\epsilon_b |a_f| + \epsilon_a |b_f|}{|b_f|^2}.$$

B.3 IEEE-aritmetiikka

Nollasta poikkeava luku x esitetään IEEE-aritmetiikan mukaisessa liukulukuesityksessä muodossa

$$x = (-1)^s M \cdot 2^E. \tag{B.3}$$

Kuva B.1 havainnollistaa liukuluvun binääristä talletusmuotoa tietokoneessa.



Kuva B.1: *Liukuluvun talletusmuoto tietokoneessa. M on liukuluvun mantissa, E eksponentti ja s kertoo luvun merkin. Mantissan esittämiseen käytetään m bittiä ja eksponenttiin e bittiä; merkin ilmaisemiseen riittää yksi bitti.*

Liukuluku koostuu etumerkin osoittavasta bitistä s sekä mantissa- ja eksponenttiosista M ja E. Mantissan binääriesitys on muotoa

$$M = b_0.b_1b_2b_3\ldots b_m,$$

missä pätee $b_i \in \{0, 1\}$, $i = 1, 2, \ldots, m$. Käytettäessä *normalisoitua liukulukuesitystä* on lisäksi voimassa epäyhtälö $1 \leq M < 2$. Esimerkiksi $+(1.10110)_2 \cdot 2^2 = 6.25$ on normalisoitu binääriesitys. Vastaavasti $+(110.110)_2 \cdot 2^0$ on saman luvun eräs normalisoimaton esitys.

Useimmissa tietokoneissa käytetään nykyisin IEEE:n (Institute of Electrical and Electronics Engineers) standardin mukaista liukulukujen esitystä ja aritmetiikkaa. IEEE-standardissa määritellään 32-bittisten liukulukujen mantissan M pituudeksi 23 bittiä ja eksponenttiosan E pituudeksi 8 bittiä. Vastaavasti määritellään *kaksoistarkkuuden liukuluvut* (64 bittiä), joiden lukualue on laajempi.

Eksponentille E pätee 32-bittisessä IEEE-aritmetiikassa $-126 \leq E \leq 127$, mikä saadaan aikaan esimerkiksi vähentämällä eksponentin 8-bittisen esityksen arvosta luku 126. Tällöin eksponentin bittiesitystä $(10010010)_2 = (146)_{10}$ vastaa eksponentti $146 - 126 = 20$ ja eksponentin bittiesitystä $(01101010)_2 = (106)_{10}$ vastaa eksponentti $106 - 126 = -20$.

Pienin 32-bittisessä IEEE-aritmetiikassa esitettävissä oleva positiivinen liukuluku on $+(1.00\ldots00)_2 \cdot 2^{-126} \approx 1.17549 \cdot 10^{-38}$ ja suurin positiivinen liukuluku on $+(1.11\ldots11)_2 \cdot 2^{127} \approx 3.40282 \cdot 10^{38}$. 64-bittisessä IEEE-aritmetiikassa on lukualue likimain $[2.22507 \cdot 10^{-308}, 1.79769 \cdot 10^{308}]$.

IEEE-aritmetiikka tuntee myös aritmeettisten erikoistapausten symbolit `Inf` (ääretön) ja `NaN` (not-a-number). Symboli `Inf` on tuloksena mm. operaatiosta $1.0/0.0$. Symboli `NaN` on tuloksena mm. operaatioista $0.0/0.0$, $0.0 \times$ `Inf` ja $\arcsin 2.0$.

B.4 Liukulukuaritmetiikan ominaisuuksia

Koska liukulukuesitys on kiinteän mittainen, voimme esittää sen avulla vain äärellisen määrän liukulukuja. Seuraavassa käytämme esimerkkinä 32-bittistä IEEE-aritmetiikkaa, jolloin voimme esittää noin 2^{31} erilaista normalisoitua liukulukua.

Kuten edellä totesimme, on olemassa suurin liukuluku eli ylivuototaso (OFL, overflow level), joka on luokkaa 10^{38}. Samoin on olemassa pienin positiivinen liukuluku eli alivuototaso (UFL, underflow level), joka on luokkaa 10^{-38}.

Esimerkki B.4.1 Ehtoa $f(x)f(y) < 0$ käytetään useissa algoritmissa testaamaan, ovatko funktion arvot erimerkkiset pisteissä x ja y. Jos arvot $f(x)$ ja $f(y)$ ovat pieniä eli alivuototason luokkaa, kertolaskun tuloksena on alivuoto. Useimmissa tietokoneissa alivuodon tuloksena on luku 0, joten ehto on epätosi, vaikka arvot olisivat erimerkkiset.

Liukuluvut eivät ole tasaisesti jakautuneet lukusuoralla, vaan ne ovat keskittyneet lähelle nollaa: väleillä $(2^{-126}, 2^{-125})$ ja $(2^{127}, 2^{128})$ on yhtä monta liukulukua.

Jokaisessa liukuluvuilla laskevassa tietokoneessa on lisäksi *konevakio* ϵ_{kone} eli pienin liukuluku, joka toteuttaa epäyhtälön $1.0 + \epsilon_{\text{kone}} > 1.0$. Konevakio kertoo tietokoneen aritmetiikan suhteellisen tarkkuuden. 32 bitin IEEE-aritmetiikassa konevakio on $2^{-23} \approx 10^{-7}$ ja 64 bitin IEEE-aritmetiikassa $2^{-52} \approx 2 \cdot 10^{-16}$.

Bittien määrä eksponentissa määrää ylivuoto- ja alivuototason. Konevakio ϵ_{kone} määräytyy bittien määrästä mantissaosassa. Tällöin pätee

$$0 < \text{alivuototaso} < \epsilon_{\text{kone}} \ll \text{ylivuototaso}.$$

Esimerkki B.4.2 Fortran 90/95:n standardifuntioiden `TINY`, `EPSILON` ja `HUGE` avulla voi selvittää käytettyjen liukulukujen alivuototason, konevakion ja ylivuototason [HRR01].

Liukulukuesitykseen liittyy pyöristys- tai katkaisuvirheitä, joita syntyy käytetyissä aritmeettisissa operaatioissa. Esimerkiksi kahden liukuluvun yhteenlaskun tulos täytyy katkaista tai pyöristää normaalin kokoiseksi liukuluvuksi (esimerkiksi 32 bittiä). Katkaisussa unohdetaan ne bitit, jotka eivät mahdu mantissaan; pyöristyksessä valitaan mantissaan se esitys, joka on lähempänä summauksesta saatua tulosta.

Esimerkki B.4.3 Sarjan

$$\sum_{i=1}^{\infty} \frac{1}{i}$$

summa on ääretön. Liukulukuaritmetiikalla laskettaessa tuloksena on kuitenkin äärellinen luku, sillä termien pienentyessä tarpeeksi ne eivät enää vaikuta osasumman arvoon. Jos laskemme summaa 32-bittisellä aritmetiikalla aloittaen suurimmista termeistä, tulos ei muutu noin kahden miljoonan termin jälkeen ja tulos on ehkä yllättävänkin pieni luku.

Jos laskemme jonon aritmeettisia operaatiota, voivat peräkkäiset liukulukuaritmetiikan katkaisu- tai pyöristysvirheet saada aikaan sen, että laskennan tarkkuus vähenee huomattavasti. Tätä kutsutaan *virheiden kasautumiseksi.*

Esimerkki B.4.4 Laskuvirheet voivat kasautua esimerkiksi laskettaessa sarjan summaa. Olkoon laskettavana

$$\sum_{i=1}^{n} s_i = \sum_{i=1}^{n} \frac{1}{i^2} = 1 + \frac{1}{4} + \frac{1}{9} + \frac{1}{16} + \cdots + \frac{1}{n^2}.$$

Kyseessä on sarja, jonka termit pienenevät, kun i kasvaa. Esimerkiksi $s_{1000} = 10^{-6}$. Sarjan summa, kun $n \to \infty$, on puolestaan $\pi^2/6 \approx 1.645$. Riippuen summausjärjestyksestä (pienimmät termit viimeiseksi tai ensimmäiseksi) saamme liukulukuaritmetiikalla eri tulokset. Sarja kannattaa summata aloittaen pienimmistä alkioista, koska siten välttyy eri suuruusluokka olevien lukujen yhteenlaskusta, mikä heikentää tuloksen tarkkuutta.

B.5 Käytännön ohjeita liukulukulaskentaan

Numeerisen algoritmin *stabiilisuudella* tarkoitetaan virheiden vaikutusta algoritmin käyttäytymiseen. Ohjelmoinnissa on hyvä noudattaa seuraavia periaatteita:

- Käytä riittävää laskentatarkkuutta. Yksinkertainen testi: laske ensin yksinkertaisella ja sitten kaksinkertaisella tarkkuudella (esimerkiksi 32 ja 64 bittiä) ja vertaa tuloksia. Esimerkiksi Fortran 90/95:llä on varsin helppo muuttaa käytettyä laskentatarkkuutta.
- Mikäli kerto- tai jakolaskun välitulokset eivät ole itseisarvoltaan hyvin suuria tai pieniä, ei tapahdu yli- ja alivuotoja.
- Minimoimalla aritmeettisten operaatioiden määrä (esimerkiksi analyyttisiä kaavoja sieventämällä) vähenevät pyöristysvirheet.
- Peruslaskutoimituksissa $(+, -, \times, /)$ tapahtuva virheiden kasvu tulisi minimoida. Esimerkiksi yhteenlaskussa tulisi summata keskenään samansuuruisia lukuja. Toisaalta vähennyslaskussa samansuuruisten lukujen vähentäminen toisistaan voi tuottaa hyvin epätarkan tuloksen.

Seuraavassa on esimerkki peruslaskutoimituksissa tapahtuvasta virheiden kasvusta, kun lasketaan yhteen erisuuruisia lukuja.

Esimerkki B.5.1 Tarkastelemme liukulukujen summausta, kun mantissan koko on 4 bittiä. Olkoon laskettavana summa $(1.000)_2 \cdot 2^0 + (1.000)_2 \cdot 2^{-4}$. Tulokseksi saamme normalisoidun liukuluvun $(1.0001)_2 \cdot 2^0$, josta pyöristämällä neljän bitin mantissaan saamme arvon $(1.001)_2 \cdot 2^0$. Katkaisemalla mantissan saamme tulokseksi arvon $(1.000)_2 \cdot 2^0$.

Huolimattomasti koodatuissa numeerisissa algoritmeissa saattaa usein tapahtua yli- tai alivuotoja välitulosten osalta, vaikka lopputulos olisikin esitettävissä liukulukuna. Seuraavassa on tästä esimerkki.

Esimerkki B.5.2 Laskemme kompleksiluvun $z = x + yi$ itseisarvon

$$|z| = \sqrt{x^2 + y^2}.$$

Suoraviivaisesti laskisimme reaalilukujen x ja y neliön, jolloin luvut jotka ovat suurempia kuin $\sqrt{\text{OFL}}$ aiheuttavat ylivuodon. Täten laskukaavaa ei voi käyttää esimerkiksi 32-bittisessä aritmetiikassa suuruusluokkaa 10^{19} suuremmille luvuillelle (OFL $\approx 10^{38}$).

Parempi tapa laskea itseisarvo on

$$|x + yi| = \begin{cases} |x|\sqrt{1 + (y/x)^2}, & \text{kun } |x| \geq |y|, \\ |y|\sqrt{1 + (x/y)^2}, & \text{kun } |x| < |y|. \end{cases} \tag{B.4}$$

Ennen laskuja on tarkistettava, että kompleksiluvun x- ja y-komponentit ovat $\neq 0$.

B.6 Virheiden kasautuminen

Pitkissä peräkkäisissä laskutoimituksissa voivat virheet kasautua jopa niin, että tarkkuus menetetään täysin.

Esimerkki B.6.1 Tutkimme eksponenttifunktion laskemista sarjakehitelmällä

$$e^x = 1 + x + \frac{x^2}{2!} + \cdots = \sum_{i=0}^{\infty} \frac{x^i}{i!} \tag{B.5}$$

Jos muuttuja x on negatiivinen, saamme vuorottelevan summan, jossa peräkkäiset summan alkiot ovat vastakkaismerkkiset. Olkoon esimerkiksi $x = -15$, jolloin saamme summaksi

$$\begin{aligned} e^{-15} \approx\ & 1 - 15 + 112.5 - 562.5 + 2109 - 6328 + \cdots - 14.09 \\ & + 5.872 - 2.380 + 0.9396 \ldots. \end{aligned}$$

Tuloksen pitäisi olla $e^{-15} \approx 3.059 \times 10^{-7}$ eli summa on hyvin pieni yhteenlaskettaviin verrattuna. Siten suhteellinen virhe kasvaa hyvin suureksi.

Tässä tapauksessa ($x < 0$) voimme käyttää identiteettiä $e^x = 1/e^{-x}$, jolloin sarja suppenee nopeammin ja saamme tarkemman tuloksen:

$$e^{-15} = \frac{1}{e^{15}} \approx \frac{1}{1 + 15 + 112.5 + 562.5 + 2109 + 6328 + \ldots}.$$

Tosin tämäkään ei ole paras mahdollinen tapa laskea funktion arvo.

B.7 Lisätietoja

Artikkeli *What every computer scientist should know about floating-point arithmetic* [Gol91] on hyvä katsaus liukulukuaritmetiikan ominaisuuksiin. CSC:n Fortran 90/95 -oppikirjassa [HRR01] kerrotaan Fortran-ohjelmointikielen tarjoamista mahdollisuuksista laskutoimitusten tarkkuuden määräämiseen.

C Lisätietoja CSC:stä

Tieteen tietotekniikan keskus CSC on opetusministeriön omistama laskennallisen tieteen keskus. CSC:ssä on töissä noin 100 eri tieteenalojen ja tietotekniikan asiantuntijaa.

CSC tarjoaa laajoja palvelukokonaisuuksia tieteellistekniseen laskentaan. CSC:n asiantuntijat neuvovat esimerkiksi matemaattisten mallien ja numeeristen menetelmien käyttöön liittyvissä asioissa. CSC myös järjestää eri sovellusalojen koulutusta, esimerkiksi mallintamisesta, numeerisista menetelmistä, visualisoinnista ja ohjelmankehityksestä.

Lisätietoja CSC:stä saa www-osoitteesta `http://www.csc.fi/`.

CSC:n julkaisemia kirjoja

Edistääkseen matemaattisen mallintamisen, numeeristen menetelmien ja ohjelmankehityksen osaamista CSC on julkaissut useita numeerisia menetelmiä ja ohjelmistoja käsitteleviä oppikirjoja ja käsikirjoja. Näitä teoksia on käytetty korkeakoulutason kurssimateriaalina eri puolilla Suomea. Osa teoksista on saatavissa PDF-muodossa www:stä.

Seuraavat kaksi teosta johdattavat lukijan numeerisen laskennan ja laskennallisen tieteen pariin. Esitietoja aiheesta ei tarvita.

- *Alkuräjähdyksestä kännykkään — näkökulmia laskennalliseen tieteeseen* (Haataja, toim.; CSC, 2002) `http://www.csc.fi/oppaat/lask.tiede/`
- *Laskennallinen tuotekehitys: suunnittelun uusi ulottuvuus* (Haataja, Järvinen, Koponen ja Råback, toim.; CSC, 2002) `http://www.csc.fi/oppaat/tuotekehitys/`

Lisäksi CSC on julkaissut seuraavat tieteellisen laskennan oppikirjoina ja käsikirjoina käytettyä teosta:

- *Datan käsittely* (Karttunen; CSC, 2001)
- *Elementtimenetelmä virtauslaskennassa* (Hämäläinen ja Järvinen; CSC, 1994)
- *Numeeriset menetelmät* (Haataja, Käpyaho ja Rahola; CSC, 1993)

- *Optimointitehtävien ratkaiseminen* (Haataja; CSC, 1995)
- *Tieteellinen visualisointi* (Ruokolainen ja Gröhn; CSC, 1996)

Numeeriseen laskentaan liittyviä ohjelmistoja on kuvattu seuraavissa teoksissa, jotka ovat saatavissa PDF-muodossa www:stä:

- *GAMS-ohjelmiston pikaopas* (Haataja; CSC, 2001)
  `http://www.csc.fi/oppaat/gams/`
- *Matemaattiset ohjelmistot* (Haataja, toim.; CSC, 1998)
  `http://www.csc.fi/oppaat/mat.ohj`
- *Ohjeita Mathematican käyttöön* (Haataja, toim.; CSC, 2000)
  `http://www.csc.fi/oppaat/mathematica/`
- *Ohjeita Matlabin käyttöön* (Haataja, toim.; CSC, 2000)
  `http://www.csc.fi/oppaat/matlab/`

CSC:llä on myös ohjelmointia ja ohjelmankehitystä käsitteleviä oppaita:

- *CSC User's Guide* (Kupila-Rantala, toim.; CSC, 2000)
  `http://www.csc.fi/oppaat/cscuser/`
- *Fortran 90/95* (Haataja, Rahola ja Ruokolainen; CSC, 2001)
- *Käytännön ohjeita ohjelmankehitykseen* (Haataja, toim.; CSC, 2000)
  `http://www.csc.fi/oppaat/ohjelmointi/`
- *Metakoneen käyttöopas* (Kupila-Rantala, toim.; CSC, 2001)
  `http://www.csc.fi/oppaat/metakone/`
- *Rinnakkaisohjelmointi MPI:llä* (Haataja ja Mustikkamäki; CSC, 2001)

Löydät lisätietoja CSC:n julkaisemista teoksista www-osoitteesta

`http://www.csc.fi/lehdet/`

CSC:n oppaita voit tilata postiosoitteesta

Opastilaus / Paula Mäki-Välkkilä
CSC – Tieteellinen laskenta Oy
PL 405
02101 ESPOO

Voit lähettää opastilauksen myös sähköpostiosoitteeseen

`tilaus@csc.fi`

Tilauksen voi tehdä puhelimella numeroon (09) 457 2718 tai telekopiolla numeroon (09) 457 2302.

Voit lähettää tätä teosta koskevia kommentteja ja kysymyksiä sähköpostitse osoitteeseen

`Juha.Haataja@csc.fi`

Kirjallisuutta

[ABB+92] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov ja D. Sorensen. *LAPACK User's Guide*. SIAM Publications, Philadelphia, 1992.

[Act96] Forman S. Acton. *Real Computing Made Real: Preventing Errors in Scientific and Engineering Calculations*. Princeton University Press, 1996.

[AEH76] A. Alaylioglu, G. A. Evans ja J. Hyslop, The use of Chebyshev series for the evaluation of oscillatory integrals, *Computer Journal*, 3/1976, Volume 19, sivut 258–267.

[AK89] Emile Aarts ja Jan Korst. *Simulated Annealing and Boltzmann Machines*. Wiley, 1989.

[AMR95] Uri M. Ascher, Robert M. M. Mattheij ja Robert D. Russell. *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*. SIAM, 1995.

[AP98] Uri M. Ascher ja Linda R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. SIAM, 1998.

[BBC+94] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine ja H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. SIAM, Philadelphia, PA, 1994.

[Ber96] Dimitri P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1996.

[BF97] Richard L. Burden ja J. Douglas Faires. *Numerical Analysis*. Brooks/Cole, 1997.

[Bjö96] Å. Björck. *Numerical Methods for Least Squares Problems*. SIAM, 1996.

[BKM88] Anthony Brooke, David Kendrick ja Alexander Meeraus. *GAMS—A User's Guide*. The Scientific Press, 1988.

[Bri87] William L. Briggs. *A Multigrid Tutorial*. SIAM, 1987.

[BSS93] Mokhtar S. Bazaraa, Hanif D. Sherali ja C. M. Shetty. *Nonlinear Programming: Theory and Algorithms*. Wiley, 1993.

[Bur87] David S. Burnett. *Finite Element Analysis from Concepts to Applications*. Addison-Wesley, 1987.

[CBG99] Thomas Coleman, Mary Ann Branch ja Andy Grace. *MATLAB Optimization Toolbox — User's Guide*. The MathWorks, Inc, 1999.

[CK94] Ward Cheney ja David Kincaid. *Numerical Mathematics and Computing*. Brooks/Cole, 1994.

[CLR90] T. Cormen, C. Leiserson ja R. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[Dav87] Lawrence Davis. *Genetic algorithms and simulated annealing*. Pitman, 1987.

[Dav91] Lawrence Davis, toim. *Handbook of Genetic Algorithms*. Van Hostrand Reinhold, 1991.

[Dem97] James W. Demmel. *Applied Numerical Linear Algebra*. SIAM, 1997.

[DR84] P. J. Davis ja P. Rabinowitz. *Methods of Numerical Integration*. Academic Press, 1984.

[DS83] J. E. Dennis, Jr ja Robert B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, 1983.

[DS96] A. Davies ja P. Samuels. *An Introduction to Computational Geometry for Curves and Surfaces*. Oxford University Press, 1996.

[EEHJ96] K. Eriksson, D. Estep, P. Hansbo ja C. Johnson. *Computational Differential Equations*. Studentlitteratur, 1996.

[EMU96] G. Engeln-Müllges ja F. Uhlig. *Numerical Algorithms with Fortran*. Springer, 1996.

[Eva95] Gwynne Evans. *Practical Numerical Analysis*. John Wiley & Sons, Ltd, 1995.

[EWK90] Lars Eldén ja Linde Wittmeyer-Koch. *Numerical Analysis: An Introduction*. Academic Press, 1990.

[Far90] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, 1990.

[FGN92] Roland W. Freund, Gene H. Golub ja Noël M. Nachtigal. Iterative solution of linear systems. Teoksessa: *Acta Numerica 1992*, sivut 57–100. Cambridge University Press, 1992.

[Fle87] R. Fletcher. *Practical Methods of Optimization*. Wiley, 1987.

[Fow97] A. C. Fowler. *Mathematical Models in the Applied Sciences*. Cambridge University Press, 1997.

[Ger99] Neil Gershenfeld. *The Nature of Mathematical Modeling*. Cambridge University Press, 1999.

[GFM77] M. Malcolm G. Forsythe ja C.B. Moler. *Computer Methods for Mathematical Computations*. Prentice-Hall, 1977.

[GL81] Alan George ja Joseph W. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall, 1981.

[GMW81] Philip E. Gill, Walter Murray ja Margaret H. Wright. *Practical Optimization*. Academic Press, 1981.

[Gol89] David E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, 1989.

[Gol91] David Goldberg, What every computer scientist should know about floating-point arithmetic, *ACM Computing Surveys*, 1/1991, Volume 23, sivut 5–28. Www-osoite `http://www.acm.org/pubs/contents/journals/surveys/1991-23/`.

[Gre97] Anne Greenbaum. *Iterative Methods for Solving Linear Systems*. SIAM, 1997.

[GS80] I. Gladwell ja D. K. Sayers, toim. *Computational Techniques for Ordinary Differential Equations*. Academic Press, 1980.

[GvL96] Gene H. Golub ja Charles F. van Loan. *Matrix Computations*. The Johns Hopkins University Press, 1996.

[Haa95] Juha Haataja. *Optimointitehtävien ratkaiseminen.* CSC – Tieteellinen laskenta Oy, 1995.

[Haa98] Juha Haataja, toim. *Matemaattiset ohjelmistot.* CSC – Tieteellinen laskenta Oy, 1998. Www-osoite `http://www.csc.fi/oppaat/mat.ohj`.

[Haa01] Juha Haataja. *GAMS-ohjelmiston pikaopas.* CSC – Tieteellinen laskenta Oy, 2001. Www-osoite `http://www.csc.fi/oppaat/gams/`.

[Haa02] Juha Haataja, toim. *Alkuräjähdyksestä kännykkään — näkökulmia laskennalliseen tieteeseen.* CSC – Tieteellinen laskenta Oy, 2002. Www-osoite `http://www.csc.fi/oppaat/lask.tiede/`.

[Hea02] Michael T. Heath. *Scientific Computing — An Introductory Survey.* McGraw-Hill, 2002.

[HH91] G. Hämmerlin ja K.-H. Hoffmann. *Numerical Mathematics.* Springer, 1991.

[Hig96] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms.* SIAM, 1996.

[HJ94] Jari Hämäläinen ja Jari Järvinen. *Elementtimenetelmä virtauslaskennassa.* CSC – Tieteellinen laskenta Oy, 1994.

[HJKR02] Juha Haataja, Jari Järvinen, Jari Koponen ja Peter Råback, toim. *Laskennallinen tuotekehitys: suunnittelun uusi ulottuvuus.* CSC – Tieteellinen laskenta Oy, 2002. Www-osoite `http://www.csc.fi/oppaat/tuotekehitys/`.

[HKR93] Juha Haataja, Juhani Käpyaho ja Jussi Rahola. *Numeeriset menetelmät.* CSC – Tieteellinen laskenta Oy, 1993.

[HM01] Juha Haataja ja Kaj Mustikkamäki. *Rinnakkaisohjelmointi MPI:llä.* CSC – Tieteellinen laskenta Oy, 2001.

[HNW93] E. Hairer, S. P. Nørsett ja G. Wanner. *Solving Ordinary Differential Equations I.* Springer, 1993.

[Hol92] John H. Holland. *Adaptation in Natural and Artificial Systems.* Bradford Books, 1992.

[HRR01] Juha Haataja, Jussi Rahola ja Juha Ruokolainen. *Fortran 90/95.* CSC – Tieteellinen laskenta Oy, 2001.

[HT82] W. Hackbusch ja U. Trottenberg. *Multigrid Methods.* Springer-Verlag, 1982.

[HW96] E. Hairer ja G. Wanner. *Solving Ordinary Differential Equations II.* Springer, 1996.

[IMS] IMSL Inc. *MATH/LIBRARY.*

[Ise96] Arieh Iserles. *A First Course in the Numerical Analysis of Differential Equations.* Cambridge University Press, 1996.

[JS97] D. W. Jordan ja P. Smith. *Mathematical Techniques: An Introduction for the Engineering, Physical and Mathematical Sciences.* Oxford University Press, 1997.

[Kap88] J. N. Kapur. *Mathematical Modelling.* Wiley, 1988.

[Kar87] H. Kardestuncer. *Finite Element Handbook.* McGraw-Hill, 1987.

[Kar01] Hannu Karttunen. *Datan käsittely.* CSC – Tieteellinen laskenta Oy, 2001.

[Kel95] C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations.* SIAM, 1995.

[Kel99] C. T. Kelley. *Iterative Methods for Optimization.* SIAM, 1999.

[Kiv84] Simo K. Kivelä. *Matriisilasku ja lineaarialgebra*. Otakustantamo, 1984.

[KÜ98] A. R. Krommer ja C. W. Überhuber. *Computational Integration*. SIAM, 1998.

[Lam91] John D. Lambert. *Numerical Methods for Ordinary Differential Systems*. Wiley, 1991.

[MF99] Zbigniew Michalewicz ja David B. Fogel. *How to Solve It: Modern Heuristics*. Springer Verlag, 1999.

[Mic96] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, 1996.

[Mie98] Kaisa Miettinen. *Optimointi*. Jyväskylän yliopisto, Matematiikan laitos, 1998. Luentomoniste 33.

[Mit98] Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, 1998.

[MNV82] Matti Mäkelä, Olavi Nevanlinna ja Juhani Virkkunen. *Numeerinen matematiikka*. Gaudeamus, 1982.

[Moh92] G. A. Mohr. *Finite Elements for Solids, Fluids, and Optimization*. Oxford, 1992.

[MW93] Jorge J. Moré ja Stephen J. Wright. *Optimization Software Guide*. SIAM, 1993. Frontiers in applied mathematics 14.

[Mäk98] Raino A. E. Mäkinen. *Numeeriset menetelmät*. Jyväskylän yliopisto, Matematiikan laitos, 1998. Luentomoniste 12.

[New59] N. M. Newmark, A method of computation for structural dynamics, *ASCE J. of the Engineering Mechanics Division*, EM3/1959, Volume 85, sivut 67–94.

[NS96] Stephen G. Nash ja Ariela Sofer. *Linear and Nonlinear Programming*. McGraw-Hill, 1996.

[Num] Numerical Algorithms Group, Ltd. *NAG Fortran Library Manual, Mark 18*.

[NW99] Jorge Nocedal ja Stephen J. Wright. *Numerical Optimization*. Springer Verlag, 1999.

[Ort88] James M. Ortega. *Introduction to Parallel and Vector Solution of Linear Systems*. Plenum Press, 1988.

[OvG89] R. H. J. M. Otten ja L. P. P. P. van Ginneken. *The Annealing Algorithm*. Kluwer Academic Publishers, 1989.

[Par80] Beresford N. Parlett. *The Symmetric Eigenvalue Problem*. Prentice-Hall, 1980.

[Pis84] Sergio Pissanetzky. *Sparse Matrix Technology*. Academic Press, 1984.

[Pow97] M. J. D. Powell. Methods for Multivariable Interpolation at Scattered Data Points. Teoksessa: I. S. Duff ja G. A. Watson, toim., *The State of the Art in Numerical Analysis*, sivut 283–309. Institute of Mathematics and its Applications, Oxford University Press, 1997.

[PTVF92a] William H. Press, Saul A. Teukolsky, William T. Vetterling ja Brian P. Flannery. *Numerical Recipes in C — The Art of Scientific Computing*. Cambridge University Press, 1992.

[PTVF92b] William H. Press, Saul A. Teukolsky, William T. Vetterling ja Brian P. Flannery. *Numerical Recipes in Fortran — The Art of Scientific Computing*. Cambridge University Press, 1992.

[QSS00] A. Quarteroni, R. Sacco ja F. Saleri. *Numerical Mathematics*. Springer, 2000.

[RG96] Juha Ruokolainen ja Matti Gröhn. *Tieteellinen visualisointi*. CSC – Tieteellinen laskenta Oy, 1996.

[Rhe98] Werner C. Rheinboldt. *Methods for Solving Systems of Nonlinear Equations*. SIAM, 1998.

[Saa92] Yousef Saad. *Numerical Methods for Large Eigenvalue Problems*. Manchester University Press, 1992.

[Saa96] Yousef Saad. *Iterative methods for sparse linear systems*. PWS Publishers, 1996.

[SAP97] L. F. Shampine, Jr Allen, R. C. ja S. Pruess. *Fundamentals of Numerical Computing*. Wiley, 1997.

[SB93] J. Stoer ja R. Bulirsch. *Introduction to Numerical Analysis*. Springer, 1993.

[SBG96] Barry Smith, Petter Bjørstad ja William Gropp. *Domain Decomposition*. Cambridge University Press, 1996.

[Sch95] Hans-Paul Schwefel. *Evolution and Optimum Seeking*. Wiley, 1995.

[SF73] G. Strang ja G. J. Fix. *An Analysis of the Finite Element Method*. Prentice-Hall, Inc., 1973.

[SW98] Alf Samuelsson ja Nils-Erik Wiberg. *Finite Element Method Basics*. Studenlitteratur, 1998.

[Swa77] P. N. Swartztrauber, The methods of cyclic reduction, Fourier analysis and the FACR algorihm for the discrete solution of Poisson's equation on a rectangle, *SIAM Review*, 1977, Volume 19, sivut 490–501.

[vHV91] S van Huffel ja J. Vandewalle. *The Total Least Squares Problem: Computational Aspects and Analysis*. SIAM, 1991.

[vLA88] P. J. M. van Laarhoven ja E. H. L. Aarts. *Simulated Annealing: Theory and Applications*. D. Reidel Publishing Company, 1988.

[WBM87] Layne T. Watson, Stephen C. Billups ja Alexander P. Morgan, Algorithm 652—Hompack: A Suite of Codes for Globally Convergent Homotopy Algorithms, *ACM Transactions on Mathematical Software*, 1987, Volume 13, sivu 281.

[Wil65] J. H. Wilkinson. *The Algebraic Eigenvalue Problem*. Claredon Press, 1965.

[Zwi92] D. Zwillinger. *Handbook of Integration*. Jones & Bartlett, 1992.

Hakemisto

Symbolit

3/8-sääntö, 162

A

A-stabiilisuus, 223
aalloke, 123
absoluuttinen stabiilisuusalue, 216
absoluuttinen stabiilisuusväli, 217
absoluuttinen virhe, 393
ACM-järjestö, 34
Adamsin ja Bashforthin menetelmät, 230
Adamsin ja Moultonin menetelmät, 231
adaptiivinen elementtimenetelmä, 275
adaptiivinen integrointi, 171, 172
affiini kuvaus, 174
aidosti konveksi, 78
aidosti konveksi, 336
Aitkenin ja Nevillen algoritmi, 103
alakoordinaatit, 262
algebralinen yhtälö, 187
algebrallinen kertaluku, 306
Algebran peruslause, 198
algoritmi, 29
  kompleksisuus, 31
  pseudokoodina, 29
aliavaruus, 383
aliavaruusmenetelmä, 316
  kiihdytetty, 317
alideterminoitu yhtälöryhmä, 42
aliohjelmakirjastot, 34
alivuototaso, 396
alkuarvotehtävät, 256
  eksplisiittiset menetelmät, 230
  implisiittiset menetelmät, 222, 230
  ratkaisumenetelmien stabiilisuus, 216
  stabiilisuus, 212
aluehajotelmamenetelmä, 63
approksimointi, 74
Ariane 5 -raketti, 14
aritmetiikkavirhe, 14
Arnoldin menetelmä, 324
asymptoottinen stabiilisuus, 212
automaattinen derivoiminen, 203
avoin kvadratuuri, 152

B

B-splinit, 114, 122
backward differentiation formulae, 232
backtracking line search, 346
backward error, 33
backward Euler method, 222
backward-menetelmät, 232, 258
barrier function, 363
BDF-menetelmät, 232
Bernsteinin polynomit, 120
Besselin epäyhtälö, 81
Bézier'n käyrät, 120
Bézier'n pisteet, 120
Bézier'n splini, 120, 121
BFGS-päivitys, 345
BFGS-sekanttimenetelmä, 349
Bi-CGSTAB, 59
BiCG, 59
Birkhoffin interpolointi, 104
bisection, 192
bitti, 393
BLAS, 34
Boltzmannin jäähdytys, 369
boundary value problem, 211
Boussinesqin approksimaatio, 293
Bratun ongelma, 189
Brentin menetelmä, 196
BVP, 211

C

Cauchyn ja Schwarzin epäyhtälö, 390
CGS, 59
Choleskyn hajotelma, 45, 386
Clenshaw'n ja Curtisin kvadratuuri, 160
cluster methods, 366
condition number, 43
conjugate gradient, 57, 350
convex, 336
Cowellin menetelmä, 239

Cramerin sääntö, 31
CSC – Tieteellinen laskenta Oy, 400
Cuthillin ja McKeen algoritmi, 51

D

defektiivinen ominaisarvo, 306
deferred corrections, 239
definiittisyys, 384
deflaatio, 199, 316
diagonalisointi, 308
differenssiarvio, 27, 202
differenssikaava, 27
differenssimenetelmä
  keskipistesääntö, 238
  reuna-arvotehtävät, 237
  tarkentuvat korjaukset, 239
  trapetsisääntö, 238
differentiaaliyhtälö
  alkuarvotehtävä, 209
  kankea, 219
  kertaluku, 210
  reuna-arvotehtävä, 211, 234
  stabiilit tehtävät, 212
  tavallinen, 208
Dirichlet'n reunaehdot, 252
diskreetti approksimointi, 74
diskreetti Fourier'n muunnos, 110
diskreetti malli, 20
diskreetti optimointitehtävä, 333
divide-and-conquer, 321
domain decomposition, 63
dominoiva ominaisarvo, 306
dominoiva ominaisvektori, 306
Dormandin ja Princen pari, 228

E

eigenvalue, 383
eigenvector, 383
eksakti sakkofunktio, 364
eksentrinen anomalia, 189
eksplisiittinen keskipistesääntö, 218
eksplisiittiset menetelmät
  alkuarvotehtävät, 230
eksponentiaalinen kasvu, 21
ekstrapolointi, 100, 171
elementtimalli, 13
elementtimenetelmä
  adaptiivisuus, 275
  aikaintegrointi, 255, 289
  diskretointi, 248
  globaali matriisi, 270
  isoparametrinen kuvaus, 258
  Jacobin matriisi, 267
  johtavuusmatriisi, 250
  jäykkyysmatriisi, 250, 257
  kantafunktio, 258
  kuormankasvatus, 255, 289
  linearisointi, 253, 256, 285
    Newtonin linearisointi, 254, 287
    Picardin linearisointi, 255, 286
  lokaali matriisi, 270
  lokaalit koordinaatit, 259
  massamatriisi, 256
    mass lumping, 257
  normit, 273
  osittaisdifferentiaaliyhtälöryhmä, 291
  relaksointi, 255
  reunaehdot, 252
  stabilointi, 294
  suppenemiskriteerit, 272
  variaatiomuoto, 248
  virhearviot, 272
  voimavektori, 257
embedded Runge-Kutta pair, 228
ennustaja–korjaaja-menetelmät, 232
epälineaarinen optimointitehtävä, 333
epälineaarinen yhtälö, 187, 191
  kiintopistemenetelmät, 196
  Müllerin menetelmä, 198
  Newtonin menetelmä, 193
  ohjelmistot, 206
  pienimmän neliösumman tehtävä, 205
  polynomiyhtälö, 198
  puolitushaku, 192
  suosituksia menetelmän valintaan, 191
  yhdistelmämenetelmät, 195
epälineaarinen yhtälöryhmä, 199
  luotettavat menetelmät, 205
  Newtonin menetelmä, 201, 254
  sekanttimenetelmä, 203
  suosituksia menetelmän valintaan, 200
epäsileä optimointi, 333
epäsileä funktio, 380
epästabiilisuus
  alkuarvotehtävien ratkaisumenetelmät, 216, 258
epäsymmetriset iteraatiomenetelmät, 58
epätäydellinen LU-hajotelma, 62
erikoisfunktioiden integrointi, 169
esimerkkiprobleema, 48
estefunktio, 359, 363, 364
eteenpäin laskettu virhe, 33
euklidinen normi, 43, 389
Eulerin menetelmä, 214, 216, 256, 289

F

Fehlbergin menetelmä, 228
Filonin kvadratuuri, 170
Fletcherin ja Reevesin menetelmä, 345, 350
Fortran, 14
forward error, 33
Fourier'n analyysi, 107
Fourier'n approksimaatio, 109
Fourier'n muunnos, 110
Fourier'n sarja, 106, 107
Franken ja Littlen painofunktio, 144
Frobeniuksen normi, 390

G

GA, 366, *katso* geneettinen algoritmi
Galerkinin menetelmä, 249
GAMS, 370, 371
Gaussin eliminointi, 31, 38
Gaussin ja Hermiten kvadratuuri, 158
Gaussin ja Kronrodin kvadratuuri, 159
Gaussin ja Laguerren kvadratuuri, 157
Gaussin ja Legendren kvadratuuri, 156
Gaussin ja Lobatton kvadratuuri, 158
Gaussin ja Newtonin menetelmä, 355
Gaussin ja Radaun kvadratuuri, 158
Gaussin ja Seidelin menetelmä, 56
Gaussin ja Tšebyševin kvadratuuri, 157
Gaussin kvadratuuri, 156
geneettinen algoritmi, 366, 367
General Algebraic Modeling System, 370
geometrinen kertaluku, 306
Givensin rotaatio, 309
globaali minimi, 335
globaali optimointi, 335, 366
globaali suppeneminen, 336
globaali yhtälöryhmä, 270
GMRES, 59
gradienttivektori, 338, 380
Gramin ja Schmidtin menettely, 391
Gramin ja Schmidtin ortonormeerausprosessi, 324
Guide to Available Mathematical Software, 371

H

hajotelmat, 385
harvat matriisit, 47, 385
  iteratiiviset menetelmät, 55
  ohjelmistot, 55, 66
  suorat menetelmät, 49
Harwell Subroutine Library, 34, 55, 64
Hermiten interpolointi, 103
hermitointi, 382
Hessen matriisi, 338, 380
Hessenbergin matriisi, 308
Hessian, 380
Heunin menetelmä, 226
hilakvadratuurit, 183
homotopiamenetelmä, 205
Householderin matriisi, 309
Householderin muunnos, 309
HSL, 34
huonosti asetettu tehtävä, 29
hyvin asetettu tehtävä, 29
häiriöalttius, 43, 394

I

identiteettimatriisi, 385
IEEE-aritmetiikka, 395
ill posed, 29
ILU, 62
implisiittiset menetelmät
  alkuarvotehtävät, 222, 230
  Eulerin menetelmä, 222
  Rungen ja Kuttan menetelmä, 227
IMSL, 34
incomplete LU factorization, 62
indefiniitti matriisi, 384
initial value problem, 209
Institute of Electrical and Electronics Engineers, 395
integrointihila, 183
integrointivälin skaalaus, 154
interpolaatio, 151
interpolantti, 100
interpolointi, 99
interval arithmetic, 366
inverse iteration, 312
isoparametrinen elementti, 259
isoparametrinen kuvaus, 259
iteratiiviset menetelmät, 55
  epäsymmetriset iteraatiomenetelmät, 58
  Krylovin aliavaruusmenetelmät, 56
  lopetusehto, 60
  pohjustus, 61
  residuaali, 58
iteroitu integraali, 176
IVP, 209

J

Jacobin determinantti, 174
Jacobin matriisi, 210, 381
Jacobin menetelmä, 56
Jacobin rotaatio, 309
jaksollinen funktio, 106
jatkuva approksimointi, 74

jatkuva malli, 20
jatkuvuus, 379
Jenkinsin ja Traubin menetelmä, 198
jyrkimmän laskun menetelmä, 345, 351
jäähdytysmenetelmä, 367

K

kaarevuus, 114
kaksoistarkkuuden liukuluvut, 395
kankeat differentiaaliyhtälöt, 219, 258
kankeus, 221
kankeuskerroin, 221
kantafunktioapproksimaatio, 248
kantaja, 115
karakteristinen polynomi, 306
katkaistu Newtonin menetelmä, 346, 348
katkaisuvirhe, 27
Keplerin yhtälö, 189
kertaluku, 216
keskipistesääntö, 152, 238
kiintopistemenetelmä, 196
kohdefunktio, 332
kokeellinen tiede, 19
kokonaisvirhe, 215, 258
kollokaatiomenetelmä, 240
kolmiointi, 140
kompleksisuus, 31
konevakio, 396
konstitutiiviset relaatiot, 253
konveksi
  funktio, 336
  joukko, 78, 336
  optimointitehtävä, 336
kovarianssimatriisi, 84
kriittinen piste, 339
Krylovin aliavaruus, 56, 324
Krylovin aliavaruusmenetelmät, 56
Krylovin menetelmät, 323
kultaisen leikkauksen menetelmä, 340
kuutiosplini, 111
kvadraattinen interpolointi, 341
kvadraattinen optimointi, 359
kvadratuuri, 151
kvadratuurin globaali virhe, 156
kvadratuurin kertaluku, 155, 174
kvadratuurin lokaali virhe, 155
kvadratuurin progressiivisuus, 156
kvadratuuripaino, 151
kvadratuuripiste, 151
käypä alue, 332, 357
käänteinen interpolointi, 195
käänteinen potenssimenetelmä, 312
käänteismatriisi, 41

L

L_1-normi, 76, 388
L_2-normi, 76, 388, 389
laatikkorajoitteet, 378
Lagrangen interpolointi, 100
Lagrangen ja Newtonin menetelmä, 361
Lagrangen kertoimet, 361
Laguerren menetelmä, 198
Lanczosin menetelmä, 324–326
LAPACK, 34
laskennallinen tiede, 19, 20
LDL^T-hajotelma, 44
leap frog, 218
Legendren polynomit, 79
Levenbergin ja Marquardtin menetelmä, 354, 356
liittogradienttimenetelmä, 57, 345, 350
  pohjustettu, 61
liitäntälaki, 382
line search, 346
lineaarinen approksimointi, 78
lineaarinen approksimaatio, 74
lineaarinen optimointi, 333, 356
  simplex-menetelmä, 359
  sisäpistemenetelmä, 359
  standardimuotoinen tehtävä, 357
lineaarinen riippumattomuus, 383
lineaarinen suppeneminen, 341
lineaariset moniaskelmenetelmät, 229
lineaariset yhtälöryhmät, 37
  alideterminoitu, 42
  esimerkkiprobleema, 48
  harvat matriisit, 47
  iteratiiviset menetelmät, 55
  ohjelmistot, 46, 55, 66
  suorat menetelmät, 49
  tuntemattomien numerointi, 50
  ylideterminoitu, 42
linear programming, 356
Lipschitzin ehto, 209
liukuluku, 26, 393
  kaksoistarkkuus, 395
  normalisoitu, 395
liukulukuaritmetiikka, 393, 396
local extrapolation, 228
logaritminen estefunktio, 364
logistinen käyrä, 21, 22
lokaali ekstrapolaatio, 228
lokaali minimi, 335
lokaali suppeneminen, 336
lopetusehto, 60, 201
Lotkan ja Volterran yhtälöt, 23
LP-tehtävä, 356
l_p-normi, 389
LU-hajotelma, 40, 386

lukuteoreettiset kvadratuurit, 183
luonnollinen reunaehto, 250, 252
luotain, 14
luottamusaluemenetelmä, 205
lämpöyhtälö, 291
L_∞-normi, 76, 388, 389

M

maksimointitehtävä, 332
mallintaminen, 13, 18
Maple, 34
Mariner I -luotain, 14
mass lumping, 257
matemaattiset aliohjelmakirjastot, 34
matemaattiset mallit, 13, 18
matemaattiset ohjelmistot, 33
matematiikka, 20
materiaalirelaatiot, 253
Mathematica, 34
Matlab, 34
matriisi
  alakolmiomatriisi, 385
  Choleskyn hajotelma, 45
  definiittisyys, 42, 384
  hajotelmat, 385
  harva, 385
  hermiittinen, 385
  käänteismatriisi, 41
  LDL^T-hajotelma, 44
  LU-hajotelma, 40
  lävistäjämatriisi, 385
  nauhamatriisi, 48
  normaali, 386
  normi, 43, 389
  ortogonaalinen, 45, 385
  permutaatiomatriisi, 385
  QR-hajotelma, 45
  rangi, 42
  singulaarisuus, 384
  symmetrinen, 385
  säännöllinen, 42, 384
  säännöllisyysaste, 42
  tallennus, 52
  tiheä, 385
  tridiagonaalimatriisi, 47
  unitaarinen, 385
  yksikkömatriisi, 385
  yläkolmiomatriisi, 385
matriisinormi, 43, 389
matriisinotaatio, 381
Methods of Weighted Residuals, MWR, 248
Metropoliksen algoritmi, 369
minimax-approksimointi, 92
minimi, 332
  ehdot sijainnille, 337
  globaali, 335
  lokaali, 335
minimiastemenetelmä, 51
minimointitehtävä, 332
minimum degree, 51
modifioitu Newtonin menetelmä, 348
moniaskelmenetelmät, 256, 258
  Adamsin ja Bashforthin menetelmät, 230
  Adamsin ja Moultonin menetelmät, 231
  backward-menetelmät, 232
  ennustaja–korjaaja-menetelmät, 232
  yleinen r-askelinen menetelmä, 230
monihilamenetelmät, 69
monimaalimenetelmä, 236
moniresoluutioanalyysi, 130
monitavoiteoptimointi, 333
moniulotteinen optimointi, 344
moniulotteiset kvadratuurit, 174
Monte Carlo -kvadratuurit, 181
multigrid, 69
multiple shooting method, 237
multistart, 366
Müllerin menetelmä, 195, 198

N

nabla (∇), 380
NAG, 34
NASTRAN, 13
nauhamatriisi, 48
Navierin ja Stokesin yhtälöt, 293
Nelderin ja Meadin polytooppihaku, 345, 352
neliöllinen sakkofunktio, 364
neliöllinen suppeneminen, 392
NEOS, 371
nested dissection, 51
Netlib, 34, 372
Neumannin reunaehdot, 252
Nevillen algoritmi, 103
Newtonin ja Cotesin kvadratuurit, 161
Newtonin ja Raphsonin menetelmä, 193
Newtonin menetelmä, 193, 201, 343, 344, 346
  epälineaarinen yhtälö, 193
  epälineaarinen yhtälöryhmä, 201
  optimointi, 343, 345, 346
Newtonin tyyppiset ratkaisumenetelmät, 355
nollakohdat, 190
non-smooth, 380
normaaliyhtälöt, 84

normalisoitu liukulukuesitys, 395
normwise backward error, 60
numeerinen analyysi, 20
numeerinen stabiilisuus, 29
numeeriset menetelmät, 13, 25
NURBS, 122

O

$\mathcal{O}$-notaatio, 31
OFL, 396, *katso* overflow level
ohjelmistot, 33, 371
    epälineaariset yhtälöt, 206
    iteratiiviset menetelmät, 66
    lineaariset yhtälöryhmät, 46, 55, 66
    optimointi, 370
    suorat menetelmät, 55
ohjelmistovirhe, 14
olennaiset reunaehdot, 252
ominaisarvo, 305, 383
    algebrallinen kertaluku, 306
    defektiivinen, 306
    dominoiva, 306
    geometrinen kertaluku, 306
    äärimmäinen, 324
ominaisarvohajotelma, 308
ominaisarvotehtävät, 304
    aliavaruusmenetelmä, 316
    Arnoldin menetelmä, 324
    deflaatio, 316
    divide-and-conquer, 321
    herkkyys, 310
    Krylovin menetelmät, 323
    Lanczosin menetelmä, 324, 325
    pienet, 311
    potenssimenetelmä, 311
    puolitushaku, 322
    Rayleigh'n osamääräiterointi, 315
    suuret, 323
    yleistetyt, 327
ominaisvektori, 305, 383
    dominoiva, 306
Optimization Toolbox, 206
optimointi, 332
    diskreetti, 333
    epälineaarinen, 333
    epäsileä, 333
    lineaarinen, 333
    moniulotteinen, 344
    rajoitteellinen, 356
    rajoitteeton, 344
    tehtävien tyypit, 333, 371
    yksiulotteinen, 339
order of convergence, 392
orthogonal iteration, 316
ortogonaalinen matriisi, 45, 385
ortogonaalisuus, 390
ortonormaali, 79, 391
osittaisderivaatta, 379
osittaisdifferentiaaliyhtälö
    epälineaarinen, 253
    lineaarinen, 253
    Poissonin yhtälö, 48
osittaistuenta, 40
osittelulait, 382
oskilloivien funktioiden integrointi, 170
overflow level, 396

P

Padén approksimaatio, 97
paikallinen virhe, 215, 258
painofunktio, 157, 391
painotettujen jäännösten menetelmä, 248
palapolynomi, 111
paloittainen interpolointi, 110
parametrisoidut splinit, 119
Parsevalin yhtälö:, 81
Parsevalin yhtzlz, 81
partial pivoting, 40
Patriot-ohjus, 15
Pattersonin menetelmä, 159
Peanon ydin, 155
penalty function, 363
peto–saalis-malli, 23
pienen residuaalin algoritmi, 355
pienimmän neliön approksimointi, 80
pienimmän neliösumman menetelmä, 249
pienimmän neliösumman approksimointi, 83
pienimmän neliösumman tehtävä, 205
    epälineaarinen, 353
    lineaarinen, 83
pivoting, 40
p-normi, 389
pohjustettu liittogradienttimenetelmä, 61
pohjustus, 61
Poissonin yhtälö, 48
    nopea ratkaisija, 66
Polakin ja Ribièren menetelmä, 345, 350
Polyan kvadratuuri, 160
polynomiyhtälö, 187, 198
polytooppihaku, 345, 352
populaatiomallit, 19
positiivisesti definiitti, 384
positiivisesti semidefiniitti, 384
potenssimenetelmä, 311

käänteinen, 312
siirretty käänteinen, 312, 314
preconditioning, 61
predictor-corrector method, 232
pseudokoodi, 29
puolisuunnikassääntö, 152, 161
puolitushaku, 192, 322
pyöristysvirhe, 27, 218
pörssi-indeksi, 15

Q

QMR, 59
QP-tehtävät, 359
QR-hajotelma, 45, 386
QR-menetelmä, 318
kiihdytetty, 319
QZ-algoritmi, 329

R

raja-arvo, 379
rajoitteellinen optimointitehtävä, 333, 356
rajoitteet, 356
rajoitteeton optimointitehtävä, 333
raketti, 14
rakettisimulaattori, 14
rangi, 42
rationaalinen approksimointi, 97
rationaaliset B-splinit, 122
Rayleigh quotient iteration, 315
Rayleigh'n osamääräiterointi, 315
Rayleigh'n ja Ritzin menettely, 317
reaaliluvut, 393
Remezin menetelmä, 93
residuaali, 58, 84
reuna-arvotehtävät, 234
differenssimenetelmä, 237
Galerkinin menetelmä, 240
kollokaatiomenetelmä, 240
monimaalimenetelmä, 236
tähtäysmenetelmä, 235
äärellisulotteisiin aliavaruuksiin pohjautuvat menetelmät, 239
reunaehdot, 211
Ritzin arvo, 324
Ritzin vektori, 324
RKF, 228
Rombergin menetelmä, 171
Rosenbrockin funktio, 346
rounding error, 27
Rungen ja Kuttan menetelmät, 225, 256
4. kertaluvun menetelmä, 226
Dormandin ja Princen parit, 228
Fehlbergin menetelmä, 228
Heunin menetelmä, 226
implisiittiset menetelmät, 227
lokaali ekstrapolaatio, 228
virhearvio, 227
Rungen, Kuttan ja Fehlbergin menetelmä, 228
ryväsmenetelmät, 366

S

SA, 367, *katso* jäähdytysmenetelmä
Sagin ja Szekeresin menetelmä, 184
sakkofunktio, 363
sakkoparametri, 363
Schurin hajotelma, 308, 317
Schurin vektori, 308
Scud-ohjus, 15
sekanttimenetelmä, 203, 345, 349
seminormi, 388
Sequential Quadratic Programming, 361
Shepardin menetelmä, 143
shift-invert, 314, 327
shooting method, 235
SIGMA, 373
siirretty käänteinen potenssimenetelmä, 312, 314
sileä funktio, 380
similariteettimuunnos, 308
simpleksi, 140, 352
simplex-menetelmä, 359
Simpsonin sääntö, 161
simulated annealing, 367
singulaariarvo, 307, 386
singulaariarvohajotelma, 307, 386
singulaarinen, 384
singulaariset integraalit, 166
singulaarivektori, 307
singulariteetit, 189
sisäkkäisjako, 51
sisäpistemenetelmä, 359
sisäpuolinen sakkofunktio, 364
sisätulo, 80, 390
skaalausfunktio, 125
skaalautuvuus, 29
Sleipner A -öljynporauslautta, 13
smooth, 380
Snellin laki, 188
solmupiste, 151
SOR, 56
spektri, 305
splini, 111
spliniapproksimaatiot, 119
SQP-menetelmä, 361
stabiilisuus, 29, 30, 212
A-, 223

alkuarvotehtävien ratkaisumenetelmät, 216
asymptoottinen, 212
differentiaaliyhtälöt, 212
numeeriset algoritmit, 397
stabiloitu elementtimenetelmä, 251, 294
steepest descent, 345, 351
stiff, 220
suhteellinen virhe, 393
suljettu kvadratuuri, 152
suorat menetelmät, 49, 352
ohjelmistot, 55
tuntemattomien numerointi, 50
SUPG, 295
suppeneminen, 336
globaali, 336
kertaluku, 392
lokaali, 336
suppenemisnopeus, 392
suuren residuaalin algoritmi, 355
symboliluettelo, 11
säännöllinen matriisi, 42, 384
säännöllisyysaste, 42

T

taaksepäinen virhe, 33, 60
takaisinsijoitus, 39
takautuva viivahaku, 346
tarkentuvat korjaukset, 239
tasainen approksimaatio, 91
tasavälinen interpolointi, 100
teoreettinen tiede, 19
tiede
laskennallinen, 20
tutkimusmenetelmät, 19
tieteen tietotekniikan keskus CSC, 400
tiheä, 385
TNPACK, 373
toistettu kvadraattinen interpolointi, 341
toistettu kvadraattinen optimointi, 361
TOMS, 35
transpoosi, 382
trapetsimenetelmä, 223
trapetsisääntö, 238
tridiagonaalimatriisi, 47
trigonometrinen interpolointi ja approksimointi, 106
trigonometriset polynomit, 106
truncation error, 27
trust region method, 205
Tšebyševin pisteet, 101
Tšebyševin approksimaatio, 91
Tšebyševin integrointi, 164
Tšebyševin interpolaatio, 160
Tšebyševin polynomit, 94
tuenta, 40
tulokaavat, 138, 178
tuntemattomien numerointi, 50
tähtäysmenetelmä, 235
täydellinen kanta, 81
täydellinen pienimmän neliön sovitus, 89

U

UFL, 396, *katso* underflow level
ulkopuolinen sakkofunktio, 363
underflow level, 396
unimodaalinen funktio, 340
usean muuttujan funktioiden interpolointi, 136

V

vaihdantalaki, 382
vapausaste, 140
wavelet, 123
Weierstrassin approksimaatiolause, 80
vektoriavaruus, 75, 387
vektorit, 381
well posed, 29
Wielandtin deflaatio, 316
viivahaku, 346
Wilkinsonin siirto, 320
virheenvahvistumiskerroin, 216
virheiden kasautuminen, 397, 398
virhelähteet, 31
väliaritmetiikka, 366

Y

yhden yhtälön ratkaiseminen, 191
yhdistetty kvadratuuri, 153
yhteystiedot, 400
yhtälö
algebrallinen, 187
epälineaarinen, 187
yksiaskelmenetelmä, 225
eksplisiittinen keskipistesääntö, 218
Eulerin menetelmä, 214
implisiittinen Eulerin menetelmä, 222
Rungen ja Kuttan menetelmät, 225
trapetsimenetelmä, 223
virhearvio, 227
yksiulotteinen optimointi, 339
yleistetty ominaisarvotehtävä, 306, 327
QZ-algoritmi, 329
ylideterminoitu yhtälöryhmä, 42
ylivuototaso, 396
yrite, 248

Ä

äärellisulotteisiin aliavaruuksiin pohjautuvat menetelmät, 239
ääretön integrointiväli, 164
äärimmäinen ominaisarvo, 324