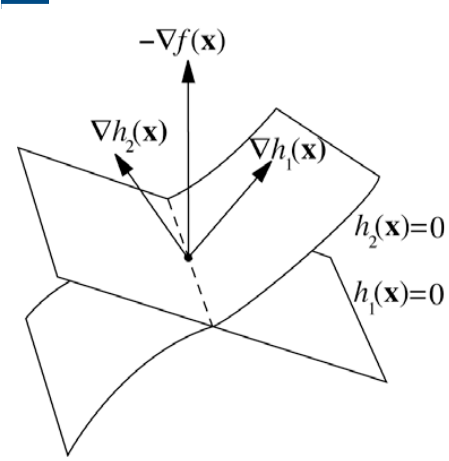
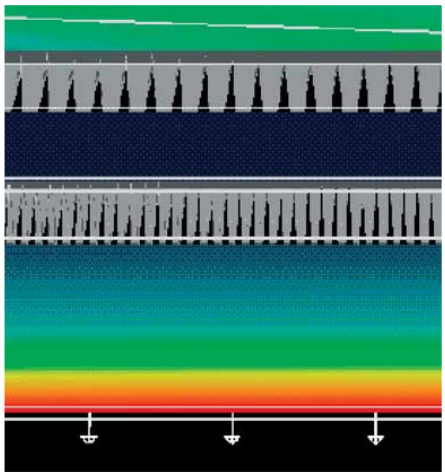
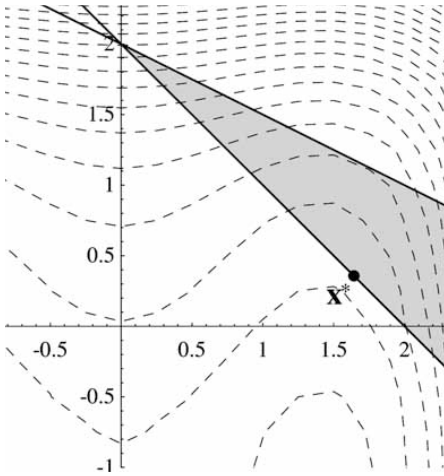
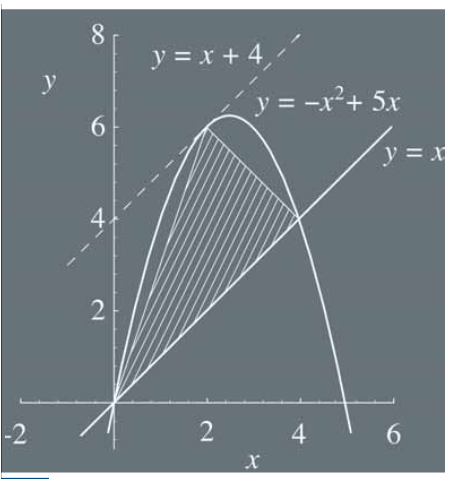
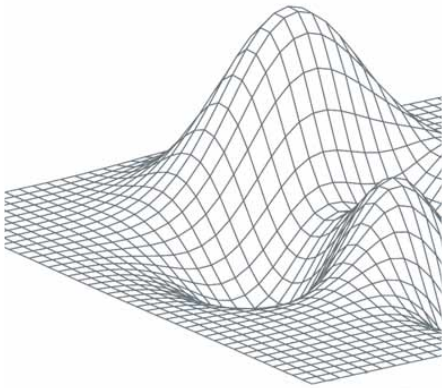
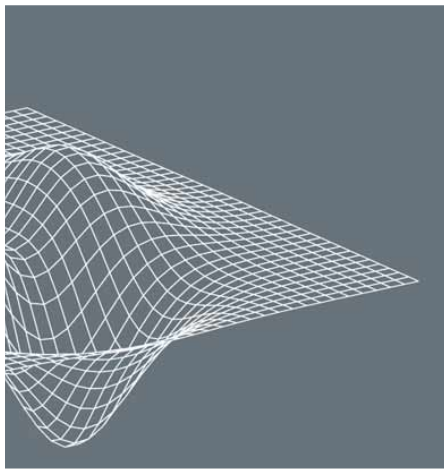












OPTIMOINTITEHTÄVIEN RATKAISEMINEN

JUHA HAATAJA
CSC

Optimointitehtävien ratkaiseminen

Optimointitehtävien ratkaiseminen

Juha Haataja

Tieteen tietotekniikan keskus CSC

3. uudistettu painos

ISBN 952-9821-95-6

http://www.csc.fi/oppaat/optimointi/

Picaset Oy
Helsinki 2004

Esipuhe

Optimointitehtävien ratkaiseminen on keskeinen osa monia laskennallisen tieteen ja tekniikan tehtäviä. Mallien monimutkaisuuden vuoksi ratkaisussa käytetään numeerisia menetelmiä, jolloin optimointimalli on kuvattava tietokoneen ymmärtämässä muodossa. Oikean menetelmän valinta voi ratkaisevasti nopeuttaa laskentaa ja parantaa tulosten luotettavuutta.

Teos on kirjoitettu ajatellen optimointitehtävien ratkaisemista työssään tarvitsevaa tutkijaa tai insinööriä. Kirja sisältää esimerkkejä optimointitehtävien matemaattisista malleista ja niiden numeerisesta ratkaisemisesta tietokoneella. Täten kirjaa voi käyttää hakuteoksena ja käsikirjana sekä itseopiskeluun.

Opas esittelee ratkaisuohjelmistoja katsauksenomaisesti. Lukija voi hankkia lisätietoja kirjallisuusluettelossa mainitusta teoksista. Liitteessä A on lisäksi eräiden englanninkielisten oppikirjojen lyhyet esittelyt.

Lukijalta edellytetään lukion pitkän matematiikan tasoiset tiedot sekä lisäksi matriisilaskennan ja optimoinnin perusteiden tuntemus.

Teoksen ensimmäisen painoksen ilmestymisestä tulee kolmannen painoksen myötä kuluneeksi kymmenen vuotta. Optimoinnin käyttö ja sovelluskohteet ovat tänä aikana suuresti lisääntyneet, mutta perusteet ovat pysyneet samoina. Teos sisältää muutaman uuden esimerkin optimoinnin käytöstä, mutta muuten tekstin sisältöön on tehty vain pienehköjä lisäyksiä ja korjauksia. Eräiden matemaattisten käsitteiden määrittelyt ja liukulukuaritmetiikan perusteet on siirretty liitteisiin. Lisäksi kolmas painos saattaa kirjallisuusviittauksia ajan tasalle.

Suuret kiitokset kaikille, jotka vaikuttivat oppaan syntymiseen ja esittivät aihepiiriin liittyviä rakentavia huomautuksia ja hyödyllisiä neuvoja. Erityiset kiitokset ansaitsevat CSC:ssä työskenneet asiantuntijat, jotka antoivat palautetta teoksen eri versioita kirjoittaessani: Olli Serimaa, Jussi Rahola, Juhani Käpyaho, Jari Järvinen, Matti Nykänen ja Peter Råback. Näiden työtoverien lisäksi Jukka Korpela, Markku Lindroos ja Raija Kukkonen kommentoivat toisen painoksen käsikirjoitusta. Harri Ehtamo ja Marko M. Mäkelä antoivat osuvaa palautetta oppaan alkutaipaleella. Oppaan kolmannen painoksen liite C on peräisin Yrjö Leinolta [HHL+02]. Lisäksi kiitän Suomen Akatemiaa tutkijankoulutuksen rahoituksesta vuosina 1993–1994.

Optimointitehtävien ratkaisemista sekä muita keskeisiä menetelmiä esitellään myös teoksessa *Numeeriset menetelmät käytännössä* [HHL+02]. Teok-

sessa *Matemaattiset ohjelmistot* [HHL+03] esitellään optimointiin soveltuvia ohjelmistoja.

Toivon saavani lukijoilta palautetta. Otan kommentteja vastaan sähköpostiosoitteessa `Juha.Haataja@csc.fi`.

Espoon Otaniemessä 21.1.2004

Juha Haataja

Sisältö

Symboliluettelo

Seuraavaan luetteloon on koottu tärkeimmät oppaassa käytetyt symbolit ja merkintätavat. Lisätietoja löytyy luvusta 1.

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
$\mathbf{0}$	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, ylä- ja alakolmiomatriisi
$\mathrm{diag}(d_1, \ldots, d_n)$	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
$\in$	kuuluu joukkoon
$\subset$	osajoukko
$\cup, \cap$	unioni, leikkaus
$\Longrightarrow$	jos … niin …
$\Longleftrightarrow$	jos ja vain jos
$\square$	todistus loppuu
$<, >$	pienempi kuin, suurempi kuin
$\leq, \geq$	pienempi tai yhtäsuuri, suurempi tai yhtäsuuri

Symboli	Selitys
$\infty, -\infty$	ääretön, miinus ääretön
$\mathrm{eig}(A)$	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: $\lfloor 1.6 \rfloor = 1$
x_f	reaaliluvun x liukulukuesitys
ϵ_M	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: $\prod_{i=1}^{n} s_i = s_1 \times s_2 \times \cdots \times s_n$
$h \to 0$	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 ja minimiarvo
$f^* = \mathrm{minimi}_{\mathbf{x}} f(\mathbf{x})$	funktion f minimiarvo muuttujan $\mathbf{x}$ suhteen
$f^* = \mathrm{maksimi}_{\mathbf{x}} f(\mathbf{x})$	funktion f maksimiarvo muuttujan $\mathbf{x}$ suhteen
$\min_i\{x_i\}, \max_i\{x_i\}$	pienin ja suurin arvo joukosta $\{x_i\}$
$\inf_{\mathbf{x}} f(\mathbf{x})$	Pienin alaraja funktiolle f
$\mathbf{x}^c$	funktion f kriittinen piste: $\nabla f(\mathbf{x}) = \mathbf{0}$
$\mathbf{g}(\mathbf{x}), \mathbf{h}(\mathbf{x})$	optimointitehtävän rajoitefunktiot
$\mathbf{p}^k, \mathbf{s}^k$	iteratiivisen algoritmin hakusuunta ja hakuaskel
$\mathbf{u}, \mathbf{v}$	Lagrangen kertoimet
$L(\mathbf{x}, \mathbf{u}, \mathbf{v})$	Lagrangen funktio
$\mathrm{epi}(f)$	funktion f epigraafi
$\mathrm{P}(\mathbf{x}_k \in \mathcal{P}^*)$	todennäköisyys että vektori $\mathbf{x}_k$ on joukossa $\mathcal{P}^*$
$\partial_c f$	funktion f aligradientti

1 Hyvä lukija!

Tämä luku kertoo oppaan tarkoituksesta ja sisällöstä sekä antaa ohjeita kirjan lukemiseen. Luvussa esitellään käytetyt merkintätavat sekä tuodaan esille kirjassa tarvittavia lineaarialgebran ja analyysin keskeisiä käsitteitä.

1.1 Optimointitehtävät

Monenlaisia ilmiöitä voidaan kuvailla energia- tai kustannusfunktioon perustuvien optimointimallien avulla. Täten syntyvien optimointitehtävien ratkaiseminen on useiden tieteen ja tekniikan alojen keskeisiä välineitä. Sovelluksia löytyy esimerkiksi fysiikasta, mekaniikasta, taloustieteestä, kemiasta ja tähtitieteestä.

Tyypillinen optimointitehtävä on matemaattisen mallin tuntemattomien parametrien arvojen hakeminen eli mallin sovittaminen dataan. Optimointimallien käyttöalue on kuitenkin hyvin laaja, kuten seuraavasta esimerkistä käy ilmi.

Esimerkki 1.1.1 Kun potilas menee silmälääkäriin hakemaan silmälasireseptiä, joutuu lääkäri ratkaisemaan usean muuttujan optimointitehtävän.

Yleisin silmälasin valinnassa optimoitava muuttuja on kauko- tai likitaittoisuus. Likitaittoisen silmämuna on liian pitkä. Likitaittoisuus korjataan sopivalla koveralla linssillä (miinus-vahvuudet), joka on keskikohdaltaan ohuempi kuin laidoiltaan. Kaukotaittoisuutta taas voidaan korjata kuperalla linssillä (plus-vahvuudet).

Silmälaseilla voidaan korjata myös hajataitteisuutta (astigmatismi), jossa silmä taittaa valoa niin, että erisuuntaisten säteiden leikkauspisteet silmän sisällä sattuvat eri kohtiin. Hajataitteisuutta korjataan ns. sylinterilinssillä.

Oletetaan, että tarvitaan korjausta sekä likitaittoisuuteen että hajataittoisuuteen. Tällöin silmälääkäri joutuu ratkomaan kolmen muuttujan optimointitehtävän, jossa koveran linssin vahvuuden lisäksi täytyy määrittää sylinterilinssin akselin suunta ($0 \ldots 180°$) ja vahvuus.

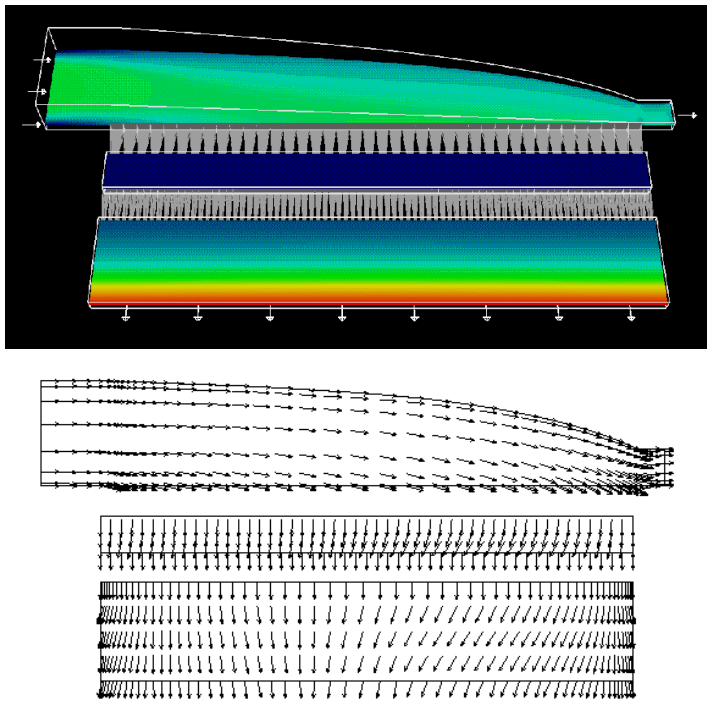
Silmälääkäri ei voi käyttää optimointitehtävän ratkaisuun normaaleja optimointimenetelmiä, esimerkiksi Newtonin menetelmää, sillä potilas ei yleensä pysty

antamaan kvalitatiivista arviota nähdyn kuvan suttuisuudesta. Sen sijaan potilas pannaan vertaamaan kahta vaihtoehtoa. Tätä vertailua toistamalla silmälääkäri pystyy löytämään optimiratkaisun kolmiulotteisesta hakuavaruudesta muutamalla iteraatiolla.

Esimerkki 1.1.2 Paperikonetta suunniteltaessa halutaan, että tuotettu paperi olisi tasalaatuista [HT02]. Perälaatikosta tulevan virtauksen jakautuminen koko koneen leveydeltä näkyy esityisesti valmiin paperin neliöpainossa (grammaa pinta-alayksikköä kohden). Paperin käyttäytymiseen esimerkiksi kopiokoneessa vaikuttaa myös kuitujen suunnan (kuituorientaatio) jakautuma paperiarkilla.

Paperikoneen perälaatikossa olevan jakotukin tehtävä on jakaa massa tasaisesti koko paperikoneen leveydeltä (kuva 1.1). Jakotukin muoto vaikuttaa merkittävästi neliöpainoprofiiliin ja siten paperikoneen kuivasta päästä tulevan paperin laatuun.

Paperimassan virtaus pyritään saamaan tasaiseksi jakotukin muotoa optimoimalla. Massan jakautumisen hyvyys voidaan kuvata keskihajontana. Näin saa-



Kuva 1.1: *Paperikoneen sisällä oleva virtausnopeus (ylempi kuva) ja virtaussuunnat (alempi kuva). Tehtävänä on löytää optimaalinen muoto jakotukin suppenemiselle tuloputkelta. Tuloputki on vasemmassa yläreunassa. (Lähde: Jari P. Hämäläinen, Metso Paper Oy. Visualisointi: Matti Gröhn, CSC.)*

daan määriteltyä optimoitava kohdefunktio. Jos paperimassan jakautuminen on optimaalista, niin keskihajonta on nolla. Keskihajonta on sitä suurempi, mitä epätasaisempi massan jakautuminen jakotukin pituudelta on.

Jakotukin muotoparametrien ja kohdefunktion välinen riippuvuus saadaan virtauslaskennan avulla. Kun jakotukin muotoa vastaava virtauskenttä lasketaan virtauslaskennan avulla, voidaan virtauskentästä määritellä kustannusfunktion arvo. Optimointitehtävä on etsiä optimaaliset muotoparametrit siten, että kustannusfunktio minimoituu mahdollisimman lähelle nollaa.

Esimerkki 1.1.3 Sivuntaitto-ohjelmiston toimintaa voi pitää esimerkkinä optimoinnista: teksti täytyy pilkkoa riveiksi, rivit yhdistää kappaleiksi ja kappaleista muodostaa peräkkäisiä sivuja. Teksti ei saa olla liian tiivistä (kirjaimet, sanat tai rivit liian lähellä toisiaan) mutta toisaalta ei liian harvaakaan (isoja välejä kirjainten tai sanojen välissä). Lisäksi voidaan vaatia, ettei rivien lopuissa saa olla liian monta tavuviivaa päällekkäin tai että otsikkorivi ei saa jäädä yksinään sivun alalaitaan.

Tässä tehtävässä kohdefunktio muodostuu eri tyyppisistä sivujen luettavuuteen ja selkeyteen vaikuttavista tekijöistä, ja lisäksi on asetettu rajoitteita, joiden puitteissa täytyy toimia. Tämän oppaan taittoon käytetty LaTeX-ohjelmisto perustuu sakkotermien käyttöön sivujen ulkoasuun vaikuttavien tekijöiden yhdistelyssä.

Joskus on mahdollista löytää optimointitehtävälle tarkat analyyttiset ratkaisut, mutta useimmiten on turvauduttava numeerisiin menetelmiin. Näin on varsinkin silloin, kun ratkaistavana on useasta eri muuttujasta riippuva tehtävä.

Viime vuosina ovat optimointitehtävien ratkaisemiseen käytetyt numeeriset menetelmät ja ohjelmistot kehittyneet huomattavasti. Osaltaan tähän ovat vaikuttaneet uusien tietokonearkkitehtuurien tarjoamat mahdollisuudet (esimerkkinä rinnakkaistietokoneet). Toisaalta on myös löydetty uusia näkökulmia ja lähestymistapoja ratkaisualgoritmien kehittelyyn.

1.1.1 Oppaan sisältö

Tämä opas esittelee eräitä tyypillisiä optimointitehtäviä sekä antaa ohjeita sopivien ratkaisumenetelmien valintaan. Opas sisältää esimerkkejä tehtävien ratkaisemisesta yleisesti käytössä olevilla ohjelmistoilla ja välineillä. Lisäksi kuvaillaan käytetyimpien ratkaisumenetelmien perusteita.

Opas rakentuu CSC:n julkaisemien kirjojen *Numeeriset menetelmät käytännössä* [HHL+02] ja *Matemaattiset ohjelmistot* [HHL+03] pohjalle. Näitä teoksia kannattaa hyödyntää tätä opasta lukiessa. Suurin osa kirjallisuusviitteistä on tekstin luettavuuden parantamiseksi siirretty kunkin luvun loppuun *Lisätietoja*-kappaleeseen. Samassa yhteydessä on lueteltu kunkin luvun aihepiiriin liittyviä lisätietojen lähteitä.

Teoksessa käsitellään pääasiassa jatkuvasti differentioituvien (sileiden) epälineaaristen optimointitehtävien ratkaisemista, mutta muunkin tyyppisiä tehtäviä esitellään. Aihepiirejä ovat

- ratkaisumenetelmän valinta
- optimointitehtävien tausta
- lineaarinen ja kvadraattinen optimointi
- kokonaisluku- ja sekalukuoptimointi
- rajoitteettomat ja rajoitteelliset epälineaariset optimointitehtävät
- epälineaariset pienimmän neliösumman tehtävät
- erikoismenetelmät ja -ohjelmistot epäsileiden tai epäjatkuvien tehtävien ratkaisemiseen sekä globaaliin optimointiin
- derivaattojen laskeminen tai arvioiminen optimointitehtävissä
- viivahaun tai luottamusalueen käyttö
- optimointitehtävien ratkaisuohjelmistot.

Tämä teos on esimerkkejä sisältävä reseptikirja. Optimointitehtävien ratkaisumenetelmien tausta ja yleisperiaatteet on esitetty tiiviissä muodossa. Teoriaosuus sisältyy pääosin lukuun 3 sekä esimerkkeinä käytettyjen optimointitehtävien ratkaisemisen yhteyteen.

Oppaassa ei käsitellä dynaamista optimointia, optimisäätöä eikä stokastista tai monitavoiteoptimointia, vaikkakin kirjan menetelmiä voidaan soveltaa näihin tehtäviin. Globaalia optimointia käsitellään suppeasti. Myös epäsileiden ja epäjatkuvien tehtävien ratkaisemisessa rajoitutaan yleisesittelyyn. Ratkaisumenetelmien rinnakkaistaminen ja numeerinen tausta on jätetty pääosin kirjallisuusviitteiden varaan.

Oppaan lukijalta vaaditaan matemaattisen analyysin ja lineaarialgebran perusteiden hallinta sekä yleistiedot optimointitehtävien ratkaisemisesta. Pitkälle menevää asiantuntemusta ei vaadita, vaikkakin numeeristen menetelmien yleistuntemuksesta on etua oppaan hyödyntämisessä.

Teos ei pyri olemaan oppikirja, sillä menetelmien johtamiset ja suppenemistodistukset on useimmiten sivuutettu. Kirjallisuusluettelosta kuitenkin löytyy runsaasti eri menetelmien taustaa ja numeerista problematiikkaa valaisevia teoksia. Toisaalta kirja sisältää runsaasti esimerkkejä, joista lienee hyötyä käsikirjamaisessa käytössä.

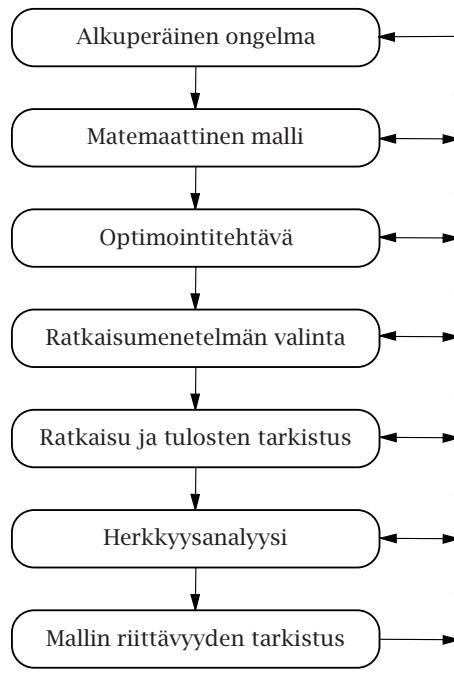
Tästä luvusta löytyy käytetyn notaation esittely (kappale 1.3). Luvun loppupuolella esitetään tiivis katsaus oppaan lukemisessa tarvittaviin lineaarialgebran, analyysin ja numeeristen menetelmien käsitteisiin. Kappaleessa 1.4 käydään läpi lineaarialgebran perusteita ja esitellään numeerisissa algoritmeissa käytettyjä matriisien hajotelmia. Kappaleessa D on havainnollistettu numeerisen laskennan perusteita.

Oppaan luvussa 2 on kerrottu ratkaisumenetelmän valintaan vaikuttavista tekijöistä. Luvussa 3 on esitelty optimoinnin keskeisiä käsitteitä. Tämän jälkeen käsitellään eri tyyppisten optimointitehtävien ratkaisemista. Oppaan lopussa liitteessä E on CSC:n yhteystietojen ja www-pohjaisten verkkopalvelujen esittely. Liitteessä B on katsaus ohjelmistoihin, joita voi käyttää optimointitehtävien ratkaisemiseen

1.2 Optimointitehtävän muodostaminen ja ratkaiseminen

Kuvassa 1.2 on esitetty kaaviona optimointitehtävän muodostamisen ja ratkaisemisen eri vaiheita. Kaaviossa lähdetään tieteellisestä tai teknillisestä ongelmasta, jonka pohjalta muodostetaan matemaattinen malli. Tämä voi olla sellaisenaan optimointitehtävä tai siinä voi olla osana optimointitehtävän ratkaiseminen. Lisäksi joissain tapauksissa toisen tyyppinen matemaattinen malli voidaan muuttaa optimointitehtävän muotoon.

Kun optimointitehtävä on muodostettu, joudutaan etsimään sille sopiva ratkaisumenetelmä. Aluksi kannattaa pyrkiä ratkaisemaan mahdollisimman yksinkertainen malli, jolloin voidaan paitsi varmistua ratkaisumenetelmän ja -ohjelmiston toimivuudesta myös mallin mielekkyydestä. Mallinnusproses-



Kuva 1.2: *Optimointitehtävän muodostaminen ja ratkaiseminen.*

sissa tarkennetaan ja laajennetaan matemaattisia malleja ja pyritään lopulta ratkaisemaan tehtäviä, jotka kuvaavat riittävän tarkasti ja luotettavasti tutkittavaa ilmiötä.

Kun ratkaisu on saatu, täytyy sen järkevyys ja tarkkuus tarkistaa, minkä jälkeen voidaan tutkia saadun ratkaisun herkkyyttä annetun mallin syöttödatan suhteen (herkkyysanalyysi). Lopuksi on syytä tutkia ratkaisun osuvuutta alkuperäisen tieteellisen tehtävän kannalta. Usein mallia joudutaan hiomaan ja tarkentamaan, jolloin kaavion esittämää prosessia käydään toistuvasti läpi.

Tässä oppaassa käsitellään lähinnä optimointitehtävän ratkaisemiseen liittyviä asioita eli siis ratkaisumenetelmän ja ohjelmiston valintaa sekä ratkaisuohjelmistojen käyttöä. Matemaattista mallintamista käsitellään eri tieteenalojen omassa kirjallisuudessa. Mallien riittävyyttä tutkittavan ilmiön kuvailemiseen käsitellään kullakin tieteenalalla eri tavalla. Joskus riittää kvalitatiivinenkin tieto, mutta toisinaan voidaan vaatia tarkkoja numeroarvoja. Tämä vaikuttaa myös ratkaisumenetelmän valintaan, mitä on kuvattu luvussa 2 (sivu 32).

Monen muuttujan optimointitehtävien ratkaisemiseen käytetään yleensä numeerisia menetelmiä. Täten esiin tulevat tyypilliset numeeristen menetelmien ominaisuudet ja ongelmat eli esimerkiksi liukulukulaskennan äärellinen laskutarkkuus sekä suppenemisnopeuden merkitys.

1.3 Kirjassa käytetyt merkinnät

Tässä teoksessa pyritään käyttämään vakiintuneita matemaattisia merkintätapoja. Tärkeimmät symbolit on esitelty symboliluettelossa sivulla 11.

Skalaariarvoisia muuttujia (yleensä reaalilukuja) merkitään kirjaimilla x, y. Vektoriarvoisia suureita merkitään lihavoiduilla symboleilla $\mathbf{x}, \mathbf{y}$. Matriiseille on varattu isot kirjaimet tyyliin A. Symboli $\mathbb{R}$ tarkoittaa reaalilukujen joukkoa ja symboli $\mathbb{R}^n$ n-ulotteista reaaliavaruutta.

Iteratiivisissa algoritmeissa merkitään iteraatteja symboleilla x_i (skalaari $\in \mathbb{R}$), $\mathbf{x}^i$ (vektori $\in \mathbb{R}^n$) ja A_i (matriisi). Vektorin alkioihin viitataan myös merkinnällä x_i, mutta tämä selviää aina asiayhteydestä.

Nollavektoria merkitään notaatiolla $\mathbf{0} = (0, 0, \ldots, 0)^T$. Vektori $\mathbf{e}^i$ on i:s standardikannan yksikkövektori:

$$e^i_j = \begin{cases} 1, & \text{kun } j = i, \\ 0, & \text{kun } j \neq i. \end{cases}$$

1.3.1 Vektorit ja matriisit

Vektori $\mathbf{x} \in \mathbb{R}^n$ muodostuu alkioistaan $x_i, i = 1, \ldots, n$ seuraavasti:

$$\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix},$$

missä kukin 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 seuraavasti:

$$AB = \begin{pmatrix} \sum_{i=1}^{m} a_{1i}b_{i1} & \cdots & \sum_{i=1}^{m} a_{1i}b_{in} \\ \vdots & \ddots & \vdots \\ \sum_{i=1}^{m} a_{li}b_{i1} & \cdots & \sum_{i=1}^{m} a_{li}b_{in} \end{pmatrix}. \tag{1.1}$$

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

$$c\mathbf{x} = \begin{pmatrix} cx_1 \\ cx_2 \\ \vdots \\ cx_n \end{pmatrix}. \tag{1.2}$$

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 $(AB)^T = B^T A^T$.

Matriiseille pätevät osittelulait

$$\begin{aligned} (A+B)C &= AC + BC, \\ A(B+C) &= AB + AC \end{aligned} \tag{1.3}$$

sekä liitäntälaki

$$A(BC) = (AB)C, \tag{1.4}$$

mutta vaihdantalaki ei ole voimassa eli yleensä on $AB \neq BA$.

Numeerisilla matriiseilla laskemiseen on kehitetty monia ohjelmistoja kuten esimerkiksi helppokäyttöinen Matlab-ohjelmisto ja Fortran-kielinen Lapack-aliohjelmakirjasto (liite B sivulla 206).

1.3.2 Derivaatat

Raja-arvon ja jatkuvuuden käsitteet ovat keskeisiä matemaattisessa analyysissä, joten esitämme aluksi muutaman määritelmän.

Määritelmä 1.3.1 (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 1.3.2 (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 1.3.3 (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 1.3.1 (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

$$\lim_{h\to 0} \frac{f(\mathbf{x} + h\mathbf{e}^i) - f(\mathbf{x})}{h}, \tag{1.5}$$

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

$$\frac{\partial f(\mathbf{x})}{\partial x_i}. \tag{1.6}$$

Jos funktiolla $f(\mathbf{x})$ on pisteessä $\mathbf{x}$ osittaisderivaatat kaikkien koordinaattien suhteen, f on *derivoituva* pisteessä $\mathbf{x}$.

Esimerkki 1.3.1 Tarkastellaan kahden muuttujan funktiota $f(x_1,x_2) = x_1^2(1-x_1x_2)$. Funktio f on jatkuvasti differentioituva (selitä miksi!). Lasketaan funktion osittaisderivaatta 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

$$\nabla = \begin{pmatrix} \frac{\partial}{\partial x_1} \\ \vdots \\ \frac{\partial}{\partial x_n} \end{pmatrix}. \tag{1.7}$$

Funktion f osittaisderivaatoista muodostettua vektoria eli *gradienttia* pisteessä $\mathbf{x}$ merkitään notaatiolla

$$\nabla f(\mathbf{x}) = \begin{pmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} \\ \vdots \\ \frac{\partial f(\mathbf{x})}{\partial x_n} \end{pmatrix}. \tag{1.8}$$

Jos $\nabla f(\mathbf{x})$ on jatkuva ja olemassa kaikissa pisteissä $\mathbf{x} \in \mathbb{R}^n$, sanotaan funktion f olevan *jatkuvasti differentioituva*. Muussa tapauksessa f on ei-differentioituva eli *epäsileä* (non-smooth).

Esimerkki 1.3.2 Tarkastellaan esimerkin 1.3.1 funktiota $f(\mathbf{x}) = x_1^2(1 - x_1x_2)$. Lasketaan gradienttivektori $\nabla f(\mathbf{x})$:

$$\nabla f(\mathbf{x}) = (2x_1 - 3x_1^2x_2,\ -x_1^3)^T.$$

Skalaariarvoisen funktion $f : \mathbb{R}^n \mapsto \mathbb{R}$ toisen kertaluvun derivaatoista muodostetusta *Hessen matriisista* (engl. Hessian; Ludwig Otto Hesse, 1811–74) käytetään merkintöjä

$$H(\mathbf{x}) = \nabla\nabla^T f(\mathbf{x}) = \begin{pmatrix} \frac{\partial^2 f(\mathbf{x})}{\partial^2 x_1} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_n} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial^2 x_n} \end{pmatrix}.$$

Kuten määritelmästä nähdään, on Hessen matriisi 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. Asiaa on käsitelty laajemmin luvussa 12 (sivu 184).

1.3.3 Muita merkintöjä

Tässä oppaassa käytetään desimaali- ja binääriluvuissa erottimena pistettä, esimerkiksi $3.14159\ldots$ ja $(0.1011)_2$. Tämä mahdollistaa numerolistojen selkeän käytön. Esimerkiksi vektorien alkiot esitetään muodossa $(1.2, -1.2)^T$.

Merkinnällä "y on suuruusluokkaa $\mathcal{O}(g(n))$" eli $y \in \mathcal{O}(g(n))$ tarkoitetaan seuraavaa:

Määritelmä 1.3.4 (Suuruusluokka $\mathcal{O}(g(n))$) Olkoon funktio $g(n)$ annettu. Merkintä $\mathcal{O}(g(n))$ tarkoittaa funktioiden $f(n)$ joukkoa

$$\begin{aligned} \mathcal{O}(g(n)) \ = \ & \{f(n) \mid \text{on olemassa positiiviset vakiot } c \text{ ja } N \text{ s.e.} \\ & 0 \le f(n) \le cg(n) \text{ kaikilla } n \ge N\} \end{aligned}$$

Vektorille $\mathbf{x}$ esitetyt epäyhtälöt (niin sanotut *laatikkorajoitteet*)

$$\mathbf{x}^l \le \mathbf{x} \le \mathbf{x}^u$$

tarkoittavat, että vektorien $\mathbf{x}^l$, $\mathbf{x}$ ja $\mathbf{x}^u$ alkioille pätee $x_i^l \leq x_i \leq x_i^u$, $i = 1, \ldots, n$. Vastaavasti voidaan käyttää yhtälömerkintää, esimerkiksi $A\mathbf{x} = \mathbf{b}$, missä A on $m \times n$ -matriisi. Yhtälö tulkitaan siis alkioittain:

$$\sum_{i=1}^{n} a_{ji} x_i = a_{j1}x_1 + a_{j2}x_2 + \cdots + a_{jn}x_n = b_j, \text{ missä } j = 1, \ldots, m.$$

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 $A\mathbf{x} = \mathbf{b}$.

Optimointitehtävät esitetään useimmiten muodossa

$$f^* = \operatorname*{minimi}_{\mathbf{x}} f(\mathbf{x}).$$

Funktion $f(\mathbf{x})$ optimipisteestä (yleensä minimistä) käytetään merkintää $\mathbf{x}^*$. Jos halutaan selville minimipiste $\mathbf{x}^*$ minimiarvon sijaan, käytetään merkintää

$$\mathbf{x}^* = \arg \operatorname*{minimi}_{\mathbf{x}} f(\mathbf{x}).$$

Siis on voimassa

$$f^* = f(\mathbf{x}^*) = \operatorname*{minimi}_{\mathbf{x}} f(\mathbf{x}) = f(\arg \operatorname*{minimi}_{\mathbf{x}} f(\mathbf{x})).$$

Algoritmien kuvaamisessa käytetään mm. Fortran 90 -kielisten esimerkkien lisäksi vapaasti mukailtua Matlab-tyyppistä pseudokoodia:

Algoritmi 1.3.1 (Esimerkki algoritmien esittämisestä)

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

1.4 Lineaariavaruudet ja matriisit

Tärkeimmät kirjassa käytetyt merkintätavat on esitelty edellä sivulta 18 alkaen. Täten skalaarit x ja vektorit $\mathbf{x}$ erotetaan lihavoinnilla ja matriiseja merkitään isoilla kirjaimilla A.

Liitteessä C (sivu 217) esitellään vektoriavaruuden ja normin käsitteet. Liitteessä D (sivu 222) esitellään liukulukuaritmetiikan ominaisuuksia.

Määritelmä 1.4.1 (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*

$$S(\mathcal{X}) = \{\ \mathbf{y} \in \mathbb{R}^n \ \mid\ \mathbf{y} = \textstyle\sum_{i=1}^m c_i\,\mathbf{x}^i,\ c_i \in \mathbb{R},\ \mathbf{x}^i \in \mathbb{R}^n\ \}. \tag{1.9}$$

Siis aliavaruus $S(\mathcal{X})$ muodostuu niistä avaruuden $\mathbb{R}^n$ pisteistä, jotka voidaan esittää joukon $\mathcal{X}$ vektorien lineaarikombinaatioina.

Määritelmä 1.4.2 (Vektorien lineaarinen riippumattomuus) Joukkoon $\{\mathbf{x}^i,\ i = 1, \ldots, n\}$ kuuluvien vektorien sanotaan olevan *lineaarisesti riippumattomia*, mikäli

$$\sum_{i=1}^{n} c_i \mathbf{x}^i = \mathbf{0} \qquad \Longrightarrow \qquad c_i = 0, \quad i = 1, \ldots, n. \tag{1.10}$$

Täten vektorijoukko $\{\mathbf{x}^i\}$ on *lineaarisesti riippuva*, mikäli nollavektori $\mathbf{0}$ voidaan esittää vektorien $\mathbf{x}^i$ lineaarikombinaationa siten, että ainakin yksi skalaarikerroin $c_i \neq 0$.

Esimerkki 1.4.1 Tarkastellaan 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 1.4.3 (Kuva-avaruus, maaliavaruus ja ydin) Lineaarikuvauksen $A : \mathbb{R}^n \mapsto \mathbb{R}^m$ *kuva-avaruus* eli *kuva* (range) $R(A)$ on *maaliavaruuden* $\mathbb{R}^m$ osajoukko:

$$\mathrm{R}(A) = \{\ \mathbf{y} \in \mathbb{R}^m \ \mid\ \text{on olemassa } \mathbf{x} \in \mathbb{R}^n \text{ siten että } \mathbf{y} = A\mathbf{x}\ \}.$$

Ytimeksi (kernel, null space) kutsutaan joukkoa

$$\ker(A) = \{\ \mathbf{x} \in \mathbb{R}^n \ \mid\ A\mathbf{x} = \mathbf{0}\ \}.$$

Esimerkki 1.4.2 Esimerkissä 1.4.1 esitetyn matriisin A ydin koostuu vektoreista, jotka ovat muotoa

$$\ker(A) = \{\ c\,(2, -1, 1)^T,\ \ c \in \mathbb{R}\ \}.$$

Kuva-avaruus voidaan puolestaan muodostaa matriisin A kahden pystyvektorin avulla, esimerkiksi seuraavasti:

$$\mathrm{R}(A) = \{\ c_1(1, -1, 2)^T + c_2(4, 3, 2)^T,\ \ c_1, c_2 \in \mathbb{R}\ \}.$$

Aliavaruuden käsitettä ja kuva-avaruutta sekä ydintä käytetään esimerkiksi aktiivijoukkomenetelmiin perustuvia optimointialgoritmeja kehitettäessä ja analysoitaessa.

Määritelmä 1.4.4 (Ominaisarvot ja ominaisvektorit) Neliömatriisin A *ominaisarvoksi* (eigenvalue) kutsutaan sellaista lukua λ, jolle pätee

$$A\mathbf{z} = \lambda\mathbf{z} \tag{1.11}$$

jollekin vektorille $\mathbf{z} \neq \mathbf{0}$. Tällaista vektoria $\mathbf{z}$ kutsutaan *ominaisvektoriksi* (eigenvector). Jatkossa käytetään ominaisarvojen λ_i joukosta merkintää

$$\mathrm{eig}(A) = \{\ \lambda_1, \ldots, \lambda_m\ \}.$$

Esimerkki 1.4.3 Esimerkin 1.4.1 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 matriisin A ytimen virittämät vektorit, sillä tällöin pätee $A\mathbf{z} = \mathbf{0}$.

Määritelmä 1.4.5 (Matriisin säännöllisyys ja singulaarisuus) Matriisin 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 1.4.6 (Matriisin definiittisyys) Symmetrinen matriisi A on *positiivisesti semidefiniitti*, jos ja vain jos

$$\langle \mathbf{y}, A\mathbf{y} \rangle \geq 0 \text{ kaikilla } \mathbf{y} \in \mathbb{R}^n. \tag{1.12}$$

Yhtäpitävä ehto on

$$\mathrm{eig}(A) \geq 0, \tag{1.13}$$

eli symmetrisen positiivisesti semidefiniitin matriisin ominaisarvojen tulee olla positiivisia tai nollia. Vastaavasti *positiivisesti definiitille* matriisille pätee

$$\langle \mathbf{y}, A\mathbf{y} \rangle > 0 \text{ kaikilla } \mathbf{y} \in \mathbb{R}^n,\ \mathbf{y} \neq \mathbf{0}\,. \tag{1.14}$$

Negatiivisesti definiitille matriisille pätee puolestaan

$$\langle \mathbf{y}, A\mathbf{y} \rangle < 0 \text{ kaikilla } \mathbf{y} \in \mathbb{R}^n, \mathbf{y} \neq \mathbf{0}. \tag{1.15}$$

Indefiniitti matriisi ei ole positiivisesti tai negatiivisesti (semi)definiitti. Indefiniitillä symmetrisellä matriisilla on vähintään yksi positiivinen ja vähintään yksi negatiivinen ominaisarvo.

Matriisin definiittisyyden käsitettä tarvitaan kvadraattisten funktioiden käsittelyssä. Asiaa on käsitelty esimerkissä 3.5.3 sivulla 71.

Esimerkki 1.4.4 Tutkitaan eräiden matriisien definiittisyyttä. Seuraavassa on annettu matriisit ja niiden ominaisarvot.

$$\begin{aligned}
Q_1 &= \begin{pmatrix} 4 & 2 \\ 2 & 2 \end{pmatrix}, \quad \operatorname{eig}(Q_1) = \{\, 3-\sqrt{5},\ 3+\sqrt{5} \,\} \approx \{\, 0.764,\ 5.24 \,\}, \\
Q_2 &= \begin{pmatrix} 4 & 2 \\ 2 & 1 \end{pmatrix}, \quad \operatorname{eig}(Q_2) = \{\, 0,\ 5 \,\}, \\
Q_3 &= \begin{pmatrix} 4 & 3 \\ 3 & 1 \end{pmatrix}, \quad \operatorname{eig}(Q_3) = \{\, \tfrac{5-3\sqrt{5}}{2},\ \tfrac{5+3\sqrt{5}}{2} \,\} \approx \{\, -0.854,\ 5.85 \,\}, \\
Q_4 &= \begin{pmatrix} -4 & 0 \\ 0 & -1 \end{pmatrix}, \quad \operatorname{eig}(Q_4) = \{\, -4,\ -1 \,\}.
\end{aligned}$$

Edellisen määritelmän nojalla on siis matriisi Q_1 positiivisesti definiitti ja matriisi Q_2 on positiivisesti semidefiniitti. Matriisi Q_3 on puolestaan indefiniitti ja matriisi Q_4 on negatiivisesti definiitti.

1.5 Lineaaristen yhtälöryhmien ratkaiseminen

Seuraavassa annetaan yksinkertainen esimerkki lineaarisesta yhtälöryhmästä.

Esimerkki 1.5.1 Tarkastellaan lineaarista yhtälöryhmää

$$\begin{cases} x_1 + 4x_2 + 2x_3 &= 2, \\ -x_1 + 3x_2 + 5x_3 &= -1, \\ 2x_1 + 2x_2 - 2x_3 &= 4. \end{cases}$$

Yhtälöryhmä on matriisimuodossa

$$A\mathbf{x} = \mathbf{b}, \tag{1.16}$$

missä

$$A = \begin{pmatrix} 1 & 4 & 2 \\ -1 & 3 & 5 \\ 2 & 2 & -2 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 2 \\ -1 \\ 4 \end{pmatrix}.$$

Yhtälöryhmää voi yrittää ratkaista Gaussin eliminoinnilla. Tässä tapauksessa yhtälöryhmä on kuitenkin singulaarinen, sillä matriisin A säännöllisyysaste (riippumattomien rivien tai sarakkeiden lukumäärä) on 2 (vertaa esimerkkiin 1.4.1).

Singulaarisella yhtälöryhmällä on joko ääretön määrä ratkaisuja tai ei yhtään ratkaisua. Tässä tapauksessa yhtälöryhmällä ei ole ratkaisua.

Vaikka algoritmeissa joskus kirjoitetaan yhtälöryhmän ratkaisuksi $\mathbf{x} = A^{-1}\mathbf{b}$, ei käänteismatriisia A^{-1} suinkaan kannata laskea, vaan sen sijaan yhtälöryhmä $A\mathbf{x} = \mathbf{b}$ ratkaistaan käyttäen hyväksi matriisin A hajotelmia tai sopivaa iteratiivista menetelmää.

Menetelmät voidaan jakaa *suoriin menetelmiin* ja *iteratiivisiin menetelmiin*. Esimerkiksi Gaussin eliminoinnissa (suora menetelmä) lopputuloksena on matriisin A hajotelma. Jos matriisi A on harva (suurin osa alkioista nollia), ei käänteismatriisi A^{-1} ole yleensä harva. Harvojen matriisien suorien menetelmien perusprobleema onkin mahdollisimman hyvän eli vähän nollasta poikkeavia alkioita sisältävän hajotelman löytäminen.

Iteratiivisia menetelmiä käytetään erityisesti suurten ja harvojen lineaaristen yhtälöryhmien ratkaisemiseen. Erityisesti jos yhtälölle $A\mathbf{x} = \mathbf{b}$ tarvitaan vain likimääräinen ratkaisuvektori $\tilde{\mathbf{x}} \approx \mathbf{x}$, ovat iteratiiviset yhtälöryhmän ratkaisijat paras vaihtoehto. Tunnetuin iteratiivinen menetelmä lienee *liittogradienttimenetelmä*, joka soveltuu symmetrisille positiivisesti definiiteille matriiseille. Esimerkiksi ns. katkaistussa Newtonin menetelmässä ratkaistaan lineaarinen yhtälöryhmä likimääräisesti käyttäen liittogradienttimenetelmää (kappale 6.5).

1.6 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 keskusmuistiin sekä esimerkiksi lineaarista yhtälöryhmää ratkaistaessa.

Lävistäjämatriisissa (diagonaalimatriisissa) on nollasta poikkeavia alkioita vain lävistäjällä eli matriisi D on muotoa

$$D = \operatorname{diag}(d_1, d_2, \ldots, d_n) = \begin{pmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_n \end{pmatrix}.$$

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* reaalikomponenttiselle matriisille A pätee $a_{ij} = a_{ji}$ eli $A = A^T$. *Ortogonaalimatriisille* Q pätee

$$Q^T Q = QQ^T = I$$

eli käänteismatriisina on $Q^{-1} = Q^T$.

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 1.6.1 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

$$QA = \begin{pmatrix} R \\ 0 \end{pmatrix}, \tag{1.17}$$

missä Q on ortogonaalinen matriisi ja R on yläkolmiomatriisi. LU-hajotelma määritellään kaavalla

$$A = LU, \tag{1.18}$$

missä L on alakolmiomatriisi (lävistäjäelementit ykkösiä) ja U yläkolmiomatriisi. Singulaariarvohajotelma puolestaan määritellään kaavalla

$$A = U\Sigma V^T, \tag{1.19}$$

missä U ja V ovat ortogonaalisia matriiseja ja lävistäjämatriisi Σ sisältää matriisin A *singulaariarvot*.

Esimerkki 1.6.2 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 LU-matriisin rivit tulevat oikeaan järjestykseen. Lisäksi QR-hajotelma on laskettu yhtälön (1.17) määritelmästä poikkeavalla tavalla, mutta koska matriisi Q on ortogonaalinen eli $Q^{-1} = Q^T$, on esitysmuotoon (1.17) helppo siirtyä.

Singulaarihajotelmasta $U\Sigma V^T$ nähdään, että kolmas singulaariarvo on nolla (matriisi Σ), joten matriisi A on singulaarinen. Singulaarisuus näkyy myös QR-hajotelman R-matriisista sekä LU-hajotelman matriisista U.

Matriisien hajotelmien laskemiseen voi käyttää esimerkiksi Matlabia, Mathematica-ohjelmistoa tai Lapack-aliohjelmakirjastoa (liite B). Käsiteltäessä isoja tiheitä matriiseja on aliohjelmakirjasto Lapack yleensä tehokkain vaihtoehto. Harvoille matriiseille on kehitetty erikoisohjelmistoja kuten Sparse ja Y12M.

1.7 Lineaarisen yhtälöryhmän häiriöalttius

Lineaarisen yhtälöryhmän ratkaiseminen on keskeinen osa monia optimointitehtävien ratkaisualgoritmeja. Seuraavassa käsitellään yhtälöryhmän *häiriöalttiutta*, joka vaikuttaa keskeisesti ratkaisemisen tarkkuuteen.

Esimerkki 1.7.1 Tarkastellaan yhtälöryhmän $A\mathbf{x} = \mathbf{b}$ ratkaisemista, kun

$$A = \begin{pmatrix} 1.0 & 0.8 \\ 0.6 + \epsilon & 0.48 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 0.5 \\ 0.6 \end{pmatrix}.$$

Tarkaksi ratkaisuksi saadaan

$$\mathbf{x} = \begin{pmatrix} \dfrac{0.3}{\epsilon}, & 0.625 - \dfrac{0.375}{\epsilon} \end{pmatrix}^T.$$

Jos luku ϵ on hyvin pieni, aiheuttaa pienikin muutos liukulukujen arvoissa (esimerkiksi katkaisu- tai pyöristysvirhe) suuren muutoksen tulokseen. Olkoon $\epsilon = 10^{-4}$. Tarkastellaan matriisin A singulaariarvohajotelmaa:

$$A = U\Sigma V^T \approx \begin{pmatrix} 0.85752 & 0.51446 \\ 0.51446 & -0.85752 \end{pmatrix} \times \begin{pmatrix} 1.49341 & 0 \\ 0 & 0.00005 \end{pmatrix} \begin{pmatrix} 0.78086 & -0.62471 \\ 0.62471 & 0.78086 \end{pmatrix}^T.$$

Matriisin A *häiriöalttius* $\kappa(A)$ kertoo, miten paljon muutokset yhtälöryhmän $A\mathbf{x} = \mathbf{b}$ kerroinmatriisissa A ja vektorissa $\mathbf{b}$ vaikuttavat tulokseen. Häiriöalttius $\kappa(A)$ määritellään suurimman ja pienimmän singulaariarvon σ_i osamääränä.

Koska esimerkissä on $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2)$, saadaan

$$\kappa(A) = \frac{\sigma_1}{\sigma_2} \approx 1.49349/0.00005 \approx 3 \cdot 10^4,$$

joten matriisin A häiriöalttius on hyvin suuri.

Tarkastellaan kahta ratkaisuvaihtoehtoa yhtälöryhmälle $A\mathbf{x} = \mathbf{b}$:

$$\mathbf{x}^1 = \begin{pmatrix} 3000 \\ -3749 \end{pmatrix}, \quad \mathbf{x}^2 = \begin{pmatrix} 2375 \\ -2968 \end{pmatrix}.$$

Lasketaan residuaalit $\mathbf{r}^k = A\mathbf{x}^k - \mathbf{b}$:

$$\mathbf{r}^1 = A\mathbf{x}^1 - \mathbf{b} \approx \begin{pmatrix} 0.3000 \\ 0.1800 \end{pmatrix}, \quad \mathbf{r}^2 = A\mathbf{x}^2 - \mathbf{b} \approx \begin{pmatrix} 0.1000 \\ -0.0025 \end{pmatrix}.$$

Yhtälöryhmän tarkka ratkaisu on neljällä desimaalilla $\mathbf{x}^1$. Kuitenkin vektorin $\mathbf{x}^1$ antama residuaali on suurempi kuin ratkaisuehdokkaan $\mathbf{x}^2$.

Vektori $\mathbf{x}^2$ onkin laskettu lisäämällä tarkkaan ratkaisuun $\mathbf{x}$ matriisin V pienintä singulaariarvoa vastaava sarake (toinen sarake) kerrottuna isolla kertoimella, jolloin tulo $A\mathbf{x}$ muuttuu suhteessa vain vähän.

1.8 Lisätietoja

Hyviä johdatuksia numeeriseen laskentaan ovat teokset *Numerical Linear Algebra and Optimization* [GMW91] ja *Numerical Analysis: An Introduction* [EWK90]. Suomenkielisiä oppikirjoja ovat *Numeerinen matematiikka* [MNV82] ja *Numeeriset menetelmät* [MS89]. Matemaattista mallintamista käsitellään esimerkiksi teoksessa *Mathematical Modelling* [Kap88]. Oppaassa *Matemaattiset ohjelmistot* [HHL+03] on puolestaan kerrottu yleisesti käytetyistä matemaattisista ohjelmistoista. Optimoinnin sovelluksista kerrotaan CSC:n julkaisemissa yleistajuisissa kirjoissa [Haa02, HJKR02].

Lineaarialgebran perusteita ja matriisilaskentaa käsitellään oppikirjassa *Matriisilasku ja lineaarialgebra* [Kiv84] ja käsikirjassa *Matrix Computations* [GvL89] sekä CSC:n oppaassa *Numeeriset menetelmät käytännössä* [HHL+02].

Lineaaristen yhtälöryhmien iteratiivisten ratkaisumenetelmien yleiskuvaus löytyy teoksesta [HHL+02]. Menetelmiä on kuvattu myös *Acta Numerica 1992* -sarjajulkaisun artikkelissa [FGN92] sekä teoksessa [Vos93]. Matriisin häiriöalttiutta on käsitelty lineaarialgebran oppikirjoissa. Matriisien hajotelmia on kuvattu teoksissa [GvL89, GMW91, GMW81]. Harvoja matriiseja käsitellään teoksessa *Sparse Matrix Technology* [Pis84].

Numeerisia optimointimenetelmiä käsitteleviä oppikirjoja ovat *Practical Methods of Optimization* [Fle87], *Practical Optimization* [GMW81], *Nonlinear Programming* [BSS93], *Epälineaarinen optimointi* [Mie98], *Nonlinear Multiobjective Optimization* [Mie99] sekä *The Mathematics of Nonlinear Programming* [PSU88].

Optimointitehtävien ratkaisumenetelmiä käsitteleviä artikkeleita löytyy esimerkiksi sarjajulkaisuista *SIAM Review*, *Operations Research*, *Mathematical Programming* ja *ACM Transactions on Mathematical Software* (ACM TOMS). TOMS-algoritmeja on lisäksi saatavissa Netlibistä.

Numerical Recipes -teoksista [PTVF92a, PTVF92b] löytyy optimointitehtävien ratkaisualgoritmeja lähdekoodina. Kannattaa kuitenkin aina tarkistaa koodin soveltuvuus ratkaistavaan tehtävään. Vaativiin tehtäviin on syytä käyttää monipuolisesti testattuja rutiineja, joita löytyy esimerkiksi Netlibistä tai kaupallisista ohjelmistoista (liite B).

Rinnakkaislaskentaa on esitelty teoksessa *Parallel and Distributed Computation* [BT88] ja *Parallel Computing* -julkaisun artikkeleissa (esimerkiksi [LR88]). CSC on julkaissut oppaan *Rinnakkaisohjelmointi MPI:llä* [HM01].

Tämän oppaan liitteessä E on kerrottu CSC:n palveluihin ja www-palvelujen käyttöön liittyvistä asioista. Liitteessä B on esitelty optimointitehtävien ratkaisuohjelmistoja. Www-osoitteesta `http://gams.nist.gov/` löytyy kätevä ohjelmisto-opas *Guide to Available Mathematical Software*.

2 Ratkaisumenetelmän ja -ohjelmiston valinta

Optimointitehtävien ratkaisemiseen ei ole olemassa yleismenetelmää. Eri tyyppisille tehtäville on syytä käyttää eri algoritmeja ja ohjelmistoja. Seuraavassa kerrotaan ratkaisualgoritmien ominaisuuksista, optimointitehtävän muodostamisesta sekä ratkaisumenetelmän ja -ohjelmiston valinnasta. Lisäksi kuvaillaan suurten tehtävien erityispiirteitä, esimerkiksi ratkaisumenetelmien muistinkäyttöä.

2.1 Mitä tarkoitetaan optimoinnilla?

Optimointitehtäviä ratkaistaan kaikilla tieteen ja tekniikan alueilla: Mikä on materiaalikustannuksiltaan edullisin rakenne, joka kestää annetun kuorman? Missä pisteessä systeemi saavuttaa minimienergiansa? Miten teollisuusprosessia ohjataan parhaalla tavalla? Miten metsien uudistushakkuut hoidetaan tuottavimmin? Miten maksimoidaan osakesalkun tuotto?

Jokapäiväinen elämäkin on optimointia: Mikä on nopein reitti lomakohteeseen? Miten minimoidaan polttoaineen kulutus työmatkalla? Kuinka kylmälaukun saa pakattua niin, että kylmyys säilyy mahdollisimman pitkään?

Useat käytännön optimointitehtävistä ovat varsin väljästi esitettyjä, ja ratkaiseminen vaatii runsaasti yksinkertaistuksia. Joskus kohtuullisen hyvä ratkaisu riittää, mutta toisaalta voidaan olla kiinnostuneita nimenomaan tarkasta globaalista optimista, esimerkiksi tutkittavan systeemin minimienergiatilasta.

Tässä luvussa tarkastellaan ratkaisumenetelmän valintaan liittyviä tekijöitä erityisesti suurten tehtävien ratkaisemisen kannalta. Esimerkiksi tietokoneen keskusmuistin suuruus tai ratkaisuun tarvittava laskenta-aika vaikuttavat ratkaisemiseen käytetyn menetelmän valintaan. Ratkaisun tehokkuutta voidaan lisäksi parantaa käyttämällä hyväksi kunkin tehtävän erikoisominaisuuksia.

Optimointitehtävän muodostaminen on oma tehtävänsä ja liittyy yleiseen matemaattiseen mallintamiseen. Seuraavassa on yksinkertainen esimerkki optimointimallin muodostamisesta.

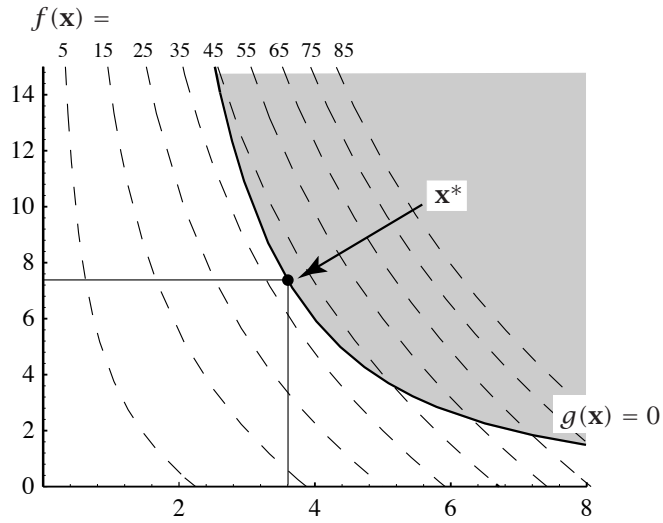
Esimerkki 2.1.1 Olkoon valmistettavana 3 desilitran ($300\,\mathrm{cm}^3$) metallitölkki, johon tulisi kulua mahdollisimman vähän materiaalia. Jos tölkin säde on r ja korkeus on h, on sen tilavuus $\pi r^2 h$ ja pinta-ala $2\pi r^2 + 2\pi r h$.

Merkitään $x_1 = r$ ja $x_2 = h$ ja jätetään vakiokerroin 2π pois kohdefunktiosta. Näin saadaan muodostettua kahden reaaliarvoisen muuttujan x_1 ja x_2 optimointitehtävä

$$\begin{aligned} &\underset{\mathbf{x}}{\text{minimi}}\ f(\mathbf{x}) = x_1^2 + x_1 x_2, \\ &\qquad \text{kun } g(\mathbf{x}) = -x_1^2 x_2 + 300/\pi \le 0 \text{ ja } \mathbf{x} \ge 0. \end{aligned} \tag{2.1}$$

Optimointitehtävän geometriaa on havainnollistettu kuvassa 2.1. Kuvasta nähdään, että rajoite g on vahvempi kuin positiivisuusehto $\mathbf{x} \ge \mathbf{0}$, joten positiivisuusehtoa ei välttämättä tarvita. Toisaalta monet ratkaisumenetelmät hyödyntävät muuttujien positiivisuutta ratkaisuavaruuden käsittelyssä. Kuvaan on myös merkitty minimipisteen $\mathbf{x}^* \approx (3.6, 7.3)^T$ sijainti. Funktion minimiarvo on $f(\mathbf{x}^*) \approx 39$. Tölkin pinta-alaksi saadaan $2\pi f(\mathbf{x}^*) \approx 250\,\mathrm{cm}^2$.

Tehtävän numeerinen ratkaisu esitetään kappaleessa 8.1.2 sivulla 135.



Kuva 2.1: *Esimerkissä 2.1.1 käsitellyn epälineaarisen minimointitehtävän geometrinen havainnollistus. Kohdefunktio on* $f(\mathbf{x}) = x_1^2 + x_1 x_2$ *ja rajoite* $g(\mathbf{x}) = -x_1^2 x_2 + 300/\pi \le 0$. *Muuttujat* x_1 *ja* x_2 *ovat positiivisia.*

Tarkasteltavan systeemin optimointia varten pyritään muodostamaan valituista muuttujista x_i riippuva *kohdefunktio* $f(\mathbf{x})$ sekä *epäyhtälö- ja yhtälörajoitteet* $\mathbf{g}(\mathbf{x}) \le \mathbf{0}$ ja $\mathbf{h}(\mathbf{x}) = \mathbf{0}$. Lisäksi voidaan tarvita tehtävän muokkausta, jotta päästään normaalimenetelmillä ratkeaviin optimointitehtäviin. Näin muodostettu avaruuden $\mathbb{R}^n$ optimointitehtävä pyritään sitten ratkaisemaan mahdollisimman luotettavasti ja tehokkaasti.

2.2 Optimointitehtävien eri tyyppejä

Tässä oppaassa käsitellään optimointitehtävien numeerista ratkaisemista. Matemaattisilla tarkasteluilla on tietenkin tärkeä osa ratkaisemisessa. Toisinaan tehtävän taustan perusteella voi päätellä jotain optimikohdan sijainnista ja ominaisuuksista.

Optimoinnilla tarkoitetaan seuraavassa vektoriavaruuden $\mathbb{R}^n$ reaaliarvoisen kohdefunktion $f : \mathbb{R}^n \mapsto \mathbb{R}$ minimointia (tai maksimointia) eli tehtävänä on hakea minimiarvo

$$\underset{\mathbf{x}}{\text{minimi}}\ f(\mathbf{x}),\ \text{kun}\ \mathbf{x} \in \mathcal{F} \subset \mathbb{R}^n. \tag{2.2}$$

Yleensä tehtävälle haetaan *lokaalia* eli paikallista minimipistettä $\mathbf{x}^* \in \mathcal{F} \subset \mathbb{R}^n$, jossa pätee

$$f(\mathbf{x}^*) \leq f(\mathbf{x}) \text{ kaikilla } \mathbf{x} \in \mathcal{F} \subset \mathbb{R}^n \text{ siten että } \|\mathbf{x}^* - \mathbf{x}\| < \epsilon,\ \epsilon > 0. \tag{2.3}$$

Jos halutaan löytää optimointitehtävän *globaali minimi* (pienin lokaaleista minimeistä), täytyy optimointitehtävän joko olla konveksi (jolloin lokaali minimi on globaali) tai joudutaan käyttämään globaaliin optimointiin kehitettyjä menetelmiä. Globaalien optimointitehtävien ratkaiseminen on yleensä erittäin vaikeaa (luku 9).

Huomautus 2.2.1 Optimointitehtävän ratkaisumenetelmä tulisi valita paitsi tehtävän tyypin myöskin ratkaisulta vaadittavien ominaisuuksien perusteella. Lisäksi numeerinen ratkaisu on parhaimmillaankin vain likiarvo, eikä globaalin optimin saavuttamisesta useinkaan ole takeita.

Optimointitehtävällä voi olla myös rajoitteita. Tässä teoksessa tarkastellaan mm. lineaarisia rajoitteita $A\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0}$. Toisaalta epälineaarisia rajoitteita sisältävien optimointitehtävien ratkaisussa muodostetaan usein lineaarisia rajoitteita sisältäviä osatehtäviä.

Optimointitehtävän käyvän alueen $\mathcal{F}$ määräävät rajoitteet (määritelmä 3.3.1 sivulla 57) jaetaan epäyhtälö- ja yhtälörajoitteiksi, joita vastaavat funktiot $\mathbf{g} : \mathbb{R}^n \mapsto \mathbb{R}^p$ ja $\mathbf{h} : \mathbb{R}^n \mapsto \mathbb{R}^q$. Yhtälö- ja epäyhtälörajoitteita sisältävä yleinen epälineaarinen optimointitehtävä on siis muotoa

$$\underset{\mathbf{x}}{\text{minimi}}\ f(\mathbf{x}),\ \text{kun}\ \mathbf{g}(\mathbf{x}) \leq \mathbf{0}\ \text{ja}\ \mathbf{h}(\mathbf{x}) = \mathbf{0}. \tag{2.4}$$

Monissa rajoitteita sisältävien tehtävien ratkaisumenetelmissä luodaan jono rajoitteettomia tehtäviä, joiden ratkaisu lähestyy rajoitteellisen tehtävän ratkaisua. Rajoitteettomien tehtävien ratkaisua on käsitelty luvussa 6 (sivu 99) ja rajoitteita sisältäviä tehtäviä luvussa 8 (sivu 134).

Tärkeä optimointitehtävien erikoistyyppi on *lineaarinen optimointi* (linear programming, LP):

$$\underset{\mathbf{x}}{\text{minimi}}\ \langle \mathbf{c}, \mathbf{x} \rangle,\ \text{kun}\ A\mathbf{x} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0}, \tag{2.5}$$

missä vektorit $\mathbf{c}$ ja $\mathbf{x}$ kuuluvat avaruuteen $\mathbb{R}^n$ ja vektori $\mathbf{b}$ kuuluu avaruuteen $\mathbb{R}^m$. Matriisi A on kooltaan $m \times n, m \leq n$. *Kvadraattinen optimointitehtävä* (quadratic programming, QP) on puolestaan muotoa

$$\underset{\mathbf{x}}{\text{minimi}} \ \frac{1}{2}\langle \mathbf{x}, Q\mathbf{x}\rangle + \langle \mathbf{c}, \mathbf{x}\rangle, \ \text{kun } A\mathbf{x} = \mathbf{b}, \ \mathbf{x} \geq \mathbf{0}. \tag{2.6}$$

Tässä Q on symmetrinen $n \times n$ -matriisi. Lineaarista ja kvadraattista optimointia tarkastellaan luvussa 4.

Kombinatorisissa optimointitehtävissä ratkaisuavaruus on diskreetti ja ratkaisuvaihtoehtoja on äärellinen määrä. Tyypillisesti kuitenkin ratkaisujen määrä kasvaa kuten 2^n tai $n!$ muuttujien lukumäärän n suhteen. Esimerkiksi sivulla 94 kuvatun sijoittelutehtävän ratkaisujen määrä kasvaa kuten $n!$. Vähänkään isommissa tehtävissä ei kaikkia mahdollisia ratkaisuja voi käydä läpi. Käytännön tehtävissä sovelletaankin usein heuristisia tai likimääräisiä menetelmiä, joilla pyritään löytämään jokin riittävän hyvä ratkaisuehdokas.

Tässä oppaassa kuvataan optimointitehtäviä taulukossa 2.1 annetun lukujaottelun mukaisesti. Esimerkkejä optimointitehtävistä ja niiden ratkaisemisesta löytyy eri tehtävätyyppejä käsittelevistä luvuista. Optimointitehtävien matemaattiseen ja numeeriseen taustaan voi tutustua kunkin luvun lopussa annettujen kirjallisuusviitteiden avulla.

Taulukko 2.1: *Optimointitehtäviä ja niiden ratkaisumenetelmiä käsittelevät oppaan luvut.*

Luku	*Aihepiiri*
4	Lineaarinen ja kvadraattinen optimointi
5	Kokonaisluku- ja sekalukutehtävät
6	Rajoitteeton epälineaarinen optimointi
7	Pienimmän neliösumman tehtävät
8	Rajoitteellinen epälineaarinen optimointi
9	Erikoismenetelmiä: globaali ja epäsileä optimointi
10	Geneettiset algoritmit
11	Jäähdytysmenetelmät
12	Derivaattojen laskeminen
13	Yksiulotteinen optimointi, viivahaku ja luottamusalue

2.3 Optimointitehtävien ratkaisumenetelmät

Ei ole olemassa algoritmia, joka soveltuisi kaikkien optimointitehtävien ratkaisemiseen. Seuraavassa kuvaillaan muutamia ratkaisumenetelmän valintaan vaikuttavia tekijöitä.

Ratkaisualgoritmi ja -ohjelmisto täytyy valita optimointitehtävän tyypin perustella. Parhaimmassa tapauksessa voi käyttää valmiiksi kehitettyä teho-

kasta ja luotettavaa ohjelmistoa. Usein tehtävän voi muuntaa muotoon, joka ratkeaa olemassaolevalla ratkaisumenetelmällä. Huonoimmassa tapauksessa on yritettävä ratkaisemista jollakin mahdollisesti tehottomalla yleismenetelmällä tai koetettava löytää tai kehittää kyseisen optimointitehtävän ratkaiseva erikoisalgoritmi.

2.3.1 Analyyttiset ratkaisumenetelmät

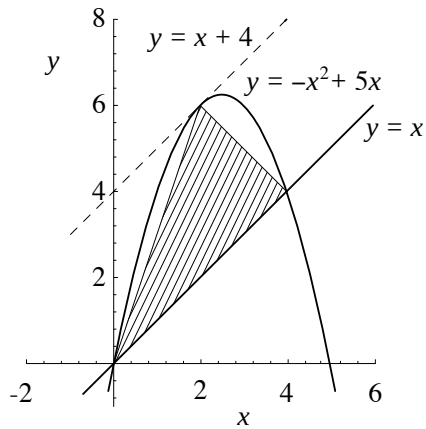
Yksinkertaiset optimointitehtävät on joskus mahdollista ratkaista analyyttisesti tutkimalla derivaattojen nollakohtia. Esimerkiksi sileiden (jatkuvasti differentioituvien) skalaarifunktioiden optimoinnissa käytetään hyväksi seuraavaa matemaattisen analyysin perusteisiin kuuluvaa lausetta.

Lause 2.3.1 Olkoon $f : \mathbb{R} \mapsto \mathbb{R}$ jatkuvasti differentioituva funktio välillä $\mathcal{F} = [x_l, x_u]$. Jos x^* on funktion f paikallinen minimi tai maksimi, on joko $x^* = x_l$ tai $x^* = x_u$ tai $f'(x^*) = 0$.

Esimerkki 2.3.1 Olkoon tehtävänä löytää suurin kolmio, joka mahtuu paraabelin $y = -ax^2 + bx$ ja suoran $y = kx$ väliin (kuva 2.2). Oletetaan, että $a > 0$ ja $b \geq 0$ sekä $k \leq b$.

Kolmion kulmapisteet x_i valitaan siten, että kaksi pistettä ovat suoran ja paraabelin leikkauspisteet ja kolmas piste on paraabelilla suoran yläpuolella (selitä miksi!). Olkoon x paraabelilta valitun pisteen x-koordinaatti. Tällöin y-koordinaatiksi saadaan $-ax^2 + bx$. Lisäksi pisteen x täytyy olla välillä $\mathcal{F} = [0, (b-k)/a]$. Alueen pinta-alaksi $f(x)$ saadaan tällöin

$$\begin{aligned} f(x) &= \frac{1}{2}x(-ax^2 + bx - kx) + \frac{1}{2}(-ax^2 + bx - kx)(\frac{b-k}{a} - x) \\ &= \frac{b-k}{2a}(-ax^2 + (b-k)x). \end{aligned}$$



Kuva 2.2: *Paraabelin ja suoran väliin jäävän kolmion pinta-alan maksimointi.*

Pinta-alafunktion f ääriarvot (minimit ja maksimit) löydetään lauseen 2.3.1 perusteella pisteestä, jossa pätee $f'(x) = 0$, tai vaihtoehtoisesti välin $\mathcal{F}$ päätepisteistä. Lasketaan siis derivaatta $f'(x)$:

$$f'(x) = \frac{b-k}{2a}(-2ax + (b-k)).$$

Asettamalla $f'(x) = 0$ saadaan $x = (b-k)/2a$. Tämän pisteen erikoisominaisuus on se, että paraabelin tangentti on saman suuntainen annetun suoran kanssa. Tarkastelemalla kuvassa 2.2 esitettyä tilannetta nähdään, että pinta-ala on maksimissaan tässä pisteessä.

Esimerkki 2.3.1 on varsin yksinkertainen, ja analyysia helpottaa se, että rajoitefunktiot voidaan sisällyttää kohdefunktioon, jolloin tehtävään jää vain yksi tuntematon muuttuja. Monimutkaisemmissa tapauksissa ei ratkaiseminen tietenkään ole näin suoraviivaista, ja tällöin tarvitaan tässä oppaassa esiteltäviä numeerisia menetelmiä.

Moniulotteisissa tehtävissä (muuttujia enemmän kuin yksi) voidaan käyttää esimerkiksi Lagrangen kerrointen menetelmää (kappale 3.3.2). Kuitenkin käytännössä vastaan tulevat optimointitehtävät ovat yleensä suuria ja lisäksi mahdollisesti epälineaarisia, jolloin analyyttisten ratkaisujen hakeminen on vaikeaa tai mahdotonta.

2.3.2 Ratkaisumenetelmien tyyppejä

Jos optimointitehtävä on jotain erikoistyyppiä, kannattaa aina käyttää kyseiseen tehtävään kehitettyjä algoritmeja ja ohjelmistoja. Tällaisia tehtävien tyyppejä ovat muun muassa lineaarinen tai sekalukuoptimointi, kvadraattinen optimointi, pienimmän neliösumman tehtävä ja verkko- tai kuljetustehtävä.

Optimointitehtävien ratkaisualgoritmit voidaan jakaa *iteratiivisiin algoritmeihin* ja *luettelointimenetelmiin*. Iteratiiviset menetelmät hakevat likimääräistä optimikohtaa, kunnes sopiva suppenemiskriteeri toteutuu eli ollaan kyllin lähellä paikallista optimikohtaa. Luettelointimenetelmissä puolestaan käydään ainakin periaatteessa läpi kaikki mahdolliset ratkaisuehdokkaat ja saatu optimi (jos sellainen löydetään) on myös tehtävän globaali ratkaisu.

Tunnetuin esimerkki luettelointimenetelmästä on lineaaristen kokonaislukutehtävien ratkaisemiseen käytetty *branch and bound* -algoritmi (algoritmia 5.2.1 sivulla 95). Monet iteratiiviset optimointitehtävien ratkaisumenetelmät perustuvat puolestaan Newtonin menetelmään (kappale 6.5 sivulla 106).

Numeeristen algoritmien kehittelyyn vaikuttaa tietenkin myös tietokonearkkitehtuurien kehitys. Viime aikoina erityisesti rinnakkaiskoneiden vaikutus algoritmien ja ohjelmistojen kehittämiseen on ollut merkittävä.

2.3.3 Algoritmien ominaisuuksia

Kukin optimointitehtävän ratkaisualgoritmi on tarkoitettu tietyn tyyppisiin optimointitehtäviin, esimerkiksi rajoitteettomiin tai rajoitteita sisältäviin tehtäviin ja yhtälö- tai epäyhtälörajoitteille. Osa algoritmeista vaatii, että optimointitehtävän kohdefunktio ja rajoitteet ovat jatkuvasti differentioituvia, osalle riittää funktioiden jatkuvuus. Eräät algoritmit soveltuvat vain lineaaristen rajoitteiden tapauksiin ($A\mathbf{x} = \mathbf{b}$ tai $A\mathbf{x} \leq \mathbf{b}$), toiset menetelmät kykenevät käsittelemään myös epälineaarisia rajoitteita.

Käytetyn algoritmin tehokkuuteen ja luotettavuuteen kannattaa kiinnittää huomiota. Mille tahansa optimointialgoritmille voidaan kehittää esimerkkitehtävä, jota se ei osaa ratkaista tehokkaasti. Toisaalta tehtävä voi olla sen tyyppinen, ettei ratkaisumenetelmä esimerkiksi numeerisista vaikeuksista johtuen löydä minimipistettä. Esimerkiksi rajoitteiden epälineaarisuus voi olla ratkaiseva tekijä. Usein on tärkeämpää saada likimääräinen ratkaisu, joka toteuttaa rajoitteet eli on käyvällä alueella, kuin tarkka ratkaisu, joka on käypä vasta iteraatioiden lopussa.

Esimerkki 2.3.2 Olkoon minimoitavana funktio

$$f_1(\mathbf{x}) = \sqrt{x_1^3 + x_2^3}.$$

Funktio f_1 on määritelty reaalilukualueessa vain, kun neliöjuurilausekkeen argumentti on positiivinen tai nolla. Täten rajoitteettomien optimointitehtävien ratkaisualgoritmit eivät sovellu sellaisenaan tehtävään. Kohdefunktion voisi määritellä erikseen arvoille $x_1^3 + x_2^3 < 0$, mutta tällöin täytyisi pitää huolta muodostuvan funktion jatkuvuudesta ja sileydestä. Toinen mahdollisuus on muuntaa tehtävä rajoitteelliseksi optimointitehtäväksi:

$$\underset{\mathbf{x}\in\mathbb{R}^3}{\text{minimi}}\ f_2(\mathbf{x}) = \sqrt{x_3},\ \text{kun}\ x_3 = x_1^3 + x_2^3\ \text{ja}\ x_3 \geq 0.$$

Ratkaisumenetelmältä vaaditaan, että se generoi rajoitteet toteuttavia pisteitä, jotta neliöjuurilauseke on määritelty.

Esimerkiksi simplex-algoritmi lineaarisille optimointitehtäville soveltuu vaiheittaiseen ongelman ratkaisemiseen, jossa ratkaistaan useita tehtäviä lähtökohtana edellinen ratkaisu. Toiset menetelmät joutuvat aloittamaan joka kerta alusta (mm. useat lineaaristen optimointitehtävien sisäpistemenetelmät). Usein sopivalla alkuarvojen valinnalla tai iteraatiomatriisien pohjustuksella voi olla suuri merkitys ratkaisunopeudelle.

Gradienttimenetelmissä tarvitaan kohdefunktion derivaatat (gradienttivektori ja mahdollisesti Hessen matriisi), *suorat menetelmät* puolestaan käyttävät pelkästään funktion arvoja. Yleensä derivaattoja käyttävät menetelmät suppenevat nopeammin, mutta esimerkiksi Hessen matriisin laskemiseen kuluu runsaasti aikaa ja muistitilaa. Derivaattoja kannattaa yleensä käyttää, jos ne voidaan laskea analyyttisesti. Luvussa 12 (sivu 184) on käsitelty derivaattojen laskemista.

Jos ratkaistava optimointitehtävä on epäsileä tai jopa epäjatkuva, valmisohjelmistojen menetelmät eivät välttämättä toimi luotettavasti. Näissä tapauksissa tehokkaan menetelmän valintaan ei voi antaa yleisohjetta. Tämän teoksen luvuissa 9, 10 ja 11 on kuvattu muutamia perinteisillä menetelmillä vaikeasti ratkaistavien optimointitehtävien ratkaisumahdollisuuksia.

Erikoisalgoritmeja löytyy lähdekoodina Netlib-verkkoarkistosta (liite B, taulukko B.6 sivulla 214). TOMS-algoritmien kuvauksia löytyy sarjajulkaisusta *ACM Transactions on Mathematical Software*.

2.3.4 Iteratiivisten algoritmien suppenemisnopeus

Monien algoritmien suppenemisnopeuteen voidaan vaikuttaa muuttamalla menetelmän parametreja, esimerkkinä viivahaun tarkkuus ja sakkofunktion kerroin. Hyvä menetelmä kykenee ratkaisemaan ison joukon tehtäviä kiinteillä parametrien arvoilla, toiset taas vaativat runsaasti kokeiluja toimiakseen.

Määritelmä 2.3.1 (Suppeneminen) Jono $x_k \in \mathbb{R}$, $k = 1, 2, \ldots$ *suppenee* kohti arvoa $x^* \in \mathbb{R}$, jos

$$\lim_{k \to \infty} |x_k - x^*| = 0. \tag{2.7}$$

Iteratiivisen menetelmän ominaisuuksiin kuuluu suppenemisen kertaluku (order of convergence).

Määritelmä 2.3.2 (Lineaarinen ja neliöllinen suppeneminen) Jos on olemassa vakio $0 \leq c < 1$ ja kokonaisluku $N \geq 1$ siten, että kaikilla $k \geq N$ pätee

$$|x_{k+1} - x^*| \leq c\,|x_k - x^*|, \tag{2.8}$$

sanotaan jonon x_k *suppenevan lineaarisesti*. Siten lineaarinen suppeneminen tarkoittaa, että virhe $|x_k - x^*|$ vähenee likimain vakiotekijällä jokaisella iteraatiolla. Jos puolestaan jollakin nollaan suppenevalla jonolla c_k pätee

$$|x_{k+1} - x^*| \leq c_k\,|x_k - x^*|, \tag{2.9}$$

sanotaan suppenemisen olevan *superlineaarista*. Lisäksi jos on olemassa vakiot $p > 1$, $c \geq 0$ ja $N \geq 1$ siten, että jono x_k suppenee kohti pistettä x^* ja kaikilla $k \geq N$ pätee

$$|x_{k+1} - x^*| \leq c\,|x_k - x^*|^p, \tag{2.10}$$

sanotaan jonon x_k suppenemisen olevan kertalukua p, esimerkiksi *neliöllistä suppenemista*, kun $p = 2$.

Esimerkki 2.3.3 Jatkuvan skalaarifunktion $f : \mathbb{R} \mapsto \mathbb{R}$ nollakohtaa x^* voidaan hakea *puolitushaun* avulla. Oletetaan, että funktio f on erimerkkinen pisteissä a_1 ja b_1. Hakuväli $[a_k, b_k] \subset \mathbb{R}$ puolitetaan jokaisella iteraatioaskeleella. Jatkoon valitaan se väli, jonka päätepisteissä funktio on erimerkkinen. Ratkaisun likiarvoksi voidaan asettaa hakuvälin keskipiste $x_k = (a_k + b_k)/2$.

Ratkaisun virhearvioksi saadaan $|x_k - x^*| \leq |a_k - b_k|/2$. Koska hakuväli puolitetaan jokaisella iteraatioaskeleella, pätee jonolle x_k:

$$|x_{k+1} - x^*| \leq \frac{1}{2}|x_k - x^*| \leq \frac{1}{4}|a_k - b_k|.$$

Yhtälössä (2.8) annetun määritelmän perusteella puolitushaku suppenee lineaarisesti.

2.4 Optimointitehtävän muodostaminen

Optimointitehtävän muodostaminen on tärkeä vaihe tehtävän ratkaisemisessa. Huonosti asetetun optimointitehtävän ratkaiseminen voi epäonnistua esimerkiksi häiriöalttiuden tai rajoitteiden ristiriitaisuuden vuoksi. Stabiilillakin ratkaisumenetelmällä voi olla ongelmia huonosti asetettujen tehtävien käsittelyssä. Toisaalta hyvissä valmisohjelmistoissa on monipuolisia tarkistuksia, jotka pyrkivät eliminoimaan virhetilanteita.

Seuraavassa muutamia käyttökelpoisia ohjeita optimointitehtävien muodostamiseen ja ratkaisemiseen:

1. Kehitä ensin yksinkertainen malli ja ratkaise se. Lisää malliin ominaisuuksia vähitellen, ja ratkaise malli kussakin vaiheessa.
2. Pyri käyttämään sileitä kohdefunktioita ja rajoitteita (muussa tapauksessa ratkaisemiseen on käytettävä erikoismenetelmiä, katso kappaletta 9.3).
3. Selvitä tarkkaan rajoitteiden merkitys ja tyyppi. Määrittele erikseen lineaariset ja epälineaariset rajoitteet sekä laatikkorajoitteet. Monet menetelmät käsittelevät erikseen myös yhtälö- ja epäyhtälörajoitteita.
4. Pyri vähentämään optimointitehtävän häiriöalttiutta. Vältä hyvin lähellä toisiaan olevia yhtälörajoitteita (pienet virheet rajoitteiden parametreissa voivat vaikuttaa minimin löytymiseen ratkaisevasti). Skaalaa muuttujat ja rajoitteet järkevästi, mieluiten lähelle ykköstä.
5. Tarkista, että kohdefunktion ja rajoitteiden arvot ja derivaatat lasketaan oikein (luku 12).

Rajoitteiden linearisointi voi huomattavasti tehostaa epälineaarisen optimointitehtävän ratkaisemista. Esimerkiksi muotoa

$$\frac{a_1}{x_1^\alpha} + \frac{a_2}{x_2^\beta} + \cdots \leq b$$

olevan rajoitteen (kohdemuuttujat x_i) voi linearisoida määrittelemällä uudet muuttujat

$$x_1' = \frac{1}{x_1^\alpha}, \quad x_2' = \frac{1}{x_2^\beta}, \quad \ldots$$

Vaikka kohdefunktio tulisikin epälineaarisemmaksi, on lineaarisia rajoitteita sisältävän tehtävän ratkaiseminen yleensä helpompaa.

Varsinkin jos käsiteltävänä on epälineaarinen optimointitehtävä, on syytä skaalata muuttujien ja vakioiden arvot lähelle ykköstä. Rajoitteiden skaalaus on usein tärkeämpää kuin kohdefunktion. Jotkin ratkaisuohjelmistot (esimerkiksi Minos) kykenevät skaalaamaan tehtävän automaattisesti. Skaalaus varmistaa ja usein myös nopeuttaa algoritmin suppenemista.

Esimerkki 2.4.1 Optimointitehtävä

$$\underset{\mathbf{x}}{\text{minimi}}\ f(\mathbf{x}) = x_1^2 + x_2^2 + 1000$$

tarjoaa esimerkin skaalauksen tärkeydestä, sillä vakion 1000 poistaminen kohdefunktiosta lisää saadun tuloksen numeerista tarkkuutta huomattavasti.

Tarkka minimipiste on $\mathbf{x}^* = \mathbf{0}$ ja minimiarvo $f(\mathbf{0}) = 1000$, mutta jos lasketaan käyttäen 32 bitin liukulukuaritmetiikkaa, saadaan vektorilla $\mathbf{x} = (0.003, 0.003)^T$ funktion arvoksi

$$f(\mathbf{x}) = 0.003^2 + 0.003^2 + 1000 = 1.8 \cdot 10^{-5} + 1000 = 1000.000018,$$

jolloin pyöristys- tai katkaisuvirheiden ansiosta tulokseksi saadaan luku 1000. Tämä johtuu siitä, että konevakio ϵ_M on luokkaa 10^{-7} (sivu 225).

Jos tämän tyyppiselle tehtävälle yritetään löytää tarkempi optimiarvo, joudutaan joko käyttämään kaksoistarkkuutta tai parantamaan skaalausta, mikä tietenkin on paras vaihtoehto.

Hyvä *alkuarvaus* tehostaa tehtävän ratkaisemista merkittävästi. Jos alkuarvaus on muuten hankalaa tuottaa, voi koettaa aluksi yksinkertaistetun tehtävän ratkaisemista ja tämän jälkeen varsinaisen tehtävän ratkaisua saadulla datalla. Alkuarvauksen olisi hyvä olla käyvällä alueella, varsinkin jos tehtävän kohdefunktiossa on singulariteetteja tai se ei ole määritelty käyvän alueen ulkopuolella.

Mikäli on mahdollista määritellä muuttujien arvoille x_i ylä- ja alarajat $x_i^l \leq x_i \leq x_i^u$ (*laatikkorajoitteet*), kannattaa se yleensä tehdä. Näillä rajoitteilla voi paitsi välttää singulariteetteja kuten nollalla jakamisen myös pienentää hakualueen kokoa.

Jos ratkaistavana on kombinatorinen optimointitehtävä (esimerkiksi kokonaislukutehtävä, luku 5), on syytä saattaa tehtävän koko mahdollisimman pieneksi. Tiukat rajat muuttujien arvoille auttavat, samoin kaikki tehtävää pienentävät yksinkertaistukset.

Optimointitehtävissä on usein paljon enemmän rajoitteita ja monimutkaisempi kohdefunktio kuin vaikkapa sivun 33 esimerkissä 2.1.1. Yksinkertaiset tehtävät voidaan ratkaista hakemalla kohdefunktion derivaattojen nollakohdat ja tutkimalla niiden optimaalisuutta. Kuitenkin dimension kasvaessa on esimerkiksi yhtälöryhmän

$$\nabla f(\mathbf{x}) = \mathbf{0}$$

ratkaiseminen yleensä huomattavasti hankalampaa kuin varsinaisen optimointitehtävän ratkaiseminen, erityisesti jos kohdefunktion derivaatat ovat epälineaarisia.

2.5 Suurten tehtävien ratkaiseminen

Suuren optimointitehtävän määritelmä tietysti muuttuu ajan myötä. Kaksikymmentä vuotta sitten voitiin pitää suurina tehtäviä, joita tänään ratkaistaan rutiininomaisesti työasemilla. Määrittely riippuu tietysti myös tehtävätyypistä: lineaarisessa optimoinnissa ratkaistaan minuuteissa tehtäviä, joissa on tuhansia rivejä ja kymmeniätuhansia nollasta poikkeavia alkioita. Toisaalta epälineaarisessa optimoinnissa tai kokonaislukutehtävissä voi muutama kymmenen muuttujaa ja rajoitetta asettaa suuria haasteita ratkaisualgoritmille.

Eräs tapa määritellä tehtävän suuruus on pitää laskenta-aikaa kriteerinä: esimerkiksi kymmenen tuntia Cray-supertietokoneen laskenta-aikaa vaativa tehtävä on ”suuri”. Toisaalta tarvittava tietokoneen keskusmuisti voi olla usein kriittisempi resurssi: laskuissa voidaan tarvita esimerkiksi $10\,000 \times 10\,000$ -kokoisia matriiseja. Kuitenkin esimerkiksi harvoille matriiseille kehitetyt talletustavat ja hajotelmat voivat merkittävästi vähentää muistin kulutusta, ja sopivasti valitut algoritmit (esimerkiksi pohjustettu liittogradienttimenetelmä lineaaristen yhtälöryhmien ratkaisemisessa) voivat pienentää laskentatyön määrää.

Numeerinen stabiilisuus käy yhä tärkeämmäksi tehtävän koon kasvaessa. Tämä ei johdu pelkästään kasvavasta laskentatyön määrästä, vaan myös esimerkiksi matriisien häiriöalttius ja optimointialgoritmin stabiilisuus voivat mennä huonompaan suuntaan tehtävän koon kasvaessa.

Esimerkki 2.5.1 Yksinkertainen havainnollistus laskentatyön määrän vaikutukselle syntyvien virheiden määrään on seuraava: Oletetaan että tehdään peräkkäin N laskutoimitusta, joissa kussakin syntyy virhettä vakion ϵ verran. Jos kussakin laskutoimituksessa virhe (pyöristys- tai katkaisuvirhe) voi syntyä yhtä suurella todennäköisyydellä ylös- tai alaspäin, on N laskutoimituksen jälkeen keskimääräinen virhe suuruusluokkaa $\sqrt{N}\epsilon$. Jos virhe syntyy joka kerralla samaan suuntaan, on lopullinen virhe luokkaa $N\epsilon$.

Tämä päättely on äärimmäisen yksinkertaistettu, eivätkä virheet yleensä summaudu edellä esitellyllä tavalla. Hyvässä algoritmissa laskutoimitusten virheet eivät kasaudu, vaan algoritmi pystyy korjaamaan tai ainakin havaitsemaan äärellisen laskutarkkuuden aiheuttamia virheitä.

Mielenkiintoinen havainto on, että eräät suuret tehtävät ovat melko helppoja ratkaista verrattuna eräisiin pienidimensioisiin tehtäviin: esimerkiksi kaksidimensioisen Rosenbrockin funktion minimin löytämiseen tarvitaan usein

parikymmentä iteraatiota (luku 6). Toisaalta monet jopa tuhansia muuttujia ja rajoitteita sisältävät tehtävät ratkeavat samalla määrällä Newton-tyyppisen menetelmän iteraatioita.

Isojen tehtävien ratkaisemisessa ongelmana ei usein olekaan toimivan ratkaisumenetelmän löytäminen vaan ratkaisemisen ja muistinkäytön tehokkuus. Lineaarialgebran tarjoamat mahdollisuudet — erilaisten hajotelmien käyttö, harvojen matriisien talletustavat sekä iteratiiviset ratkaisumenetelmät — ovat keskeisiä ratkaisumenetelmien komponentteja. Näitä on havainnollistettu kappaleessa 1.6 sivulla 27.

2.5.1 Ratkaisumenetelmien muistinkäyttö

Osa ratkaisumenetelmistä vaatii tietokoneen keskusmuistia luokkaa n^2 liukulukua (n on muuttujien lukumäärä), kun taas toiset menetelmät selviävät noin n liukuluvun muistitilalla. Esimerkiksi sekanttimenetelmät ylläpitävät $n \times n$ -kokoista Hessen matriisin arviota, mutta liittogradienttimenetelmät selviävät muutamalla n-vektorilla. Jos Hessen matriisin arvioon ei voi soveltaa harvojen matriisien tekniikkoja, ovat liittogradienttimenetelmät usein ylivoimaisia, kun muuttujien lukumäärä n on suuri.

Isoissa tehtävissä muistinkäyttö on tärkeä kriteeri menetelmän valinnalle. Jos tehtävän Hessen matriisi on harva, voi olla mahdollista käyttää harvoille matriiseille kehitettyjä algoritmeja (liite B) ja vaikkapa lineaaristen yhtälöryhmien iteratiivisia ratkaisumenetelmiä ja -ohjelmistoja. Harvaa arviota voi kokeilla myös tiheälle Hessen matriisille: asetetaan lähellä nollaa olevat matriisin alkiot nolliksi. Ongelmaksi voi kuitenkin tulla näin muodostetun matriisin singulaarisuus.

Separoituviksi kutsutaan tehtäviä, joissa kohdefunktio on muotoa

$$f(\mathbf{x}) = \sum_{i=1}^{m} f_i(\mathbf{x}). \tag{2.11}$$

Tässä f_i riippuu r_i:stä vektorin $\mathbf{x}$ alkiosta, $r_i \ll n$. Separoituville tehtäville on kehitetty tehokkaita algoritmeja, esimerkiksi ositettu sekanttimenetelmä (sivu 121).

Separoituvuutta yleisempi mutta hankalampi ominaisuus on monien suurten tehtävien *rakenteellisuus*. Esimerkiksi Hessen matriisi voi koostua joukosta pienempiä tiheitä matriiseja, jotka voivat muistuttaa läheisesti toisiaan. Mikäli tätä ominaisuutta voidaan hyödyntää, saavutetaan huomattava tehokkuuden lisäys.

Harvoja Hessen matriiseja syntyy mm. diskretoitaessa jatkuvia (ääretöndimensioisia) tehtäviä palafunktioiden avulla (esimerkkinä elementtimenetelmä, FEM eli finite element method). Tällöin jonkin tietyn muuttujan optimiarvoon vaikuttaa yleensä vain pieni osa muuttujista. Tällainen paikallinen vuorovaikutus tuottaa usein harvoja tai separoituvia Hessen ja Jacobin matriiseja. Lisäksi funktioiden derivaatat voi olla mahdollista laskea symbolisessa muodossa (luku 12), riippuen valitusta kantafunktiojoukosta. Ongelmana voi puolestaan olla palafunktioiden epäsileys esimerkiksi reunapisteissä.

Lähes kaikki matemaattisen fysiikan tehtävät voidaan esittää optimointitehtävän muodossa eli variaatiolaskennan avulla. Seuraavassa esitetään yksinkertainen fysiikan optimointitehtävä.

Esimerkki 2.5.2 Kuvassa 2.3 on esimerkki *äärellisestä katenoidista*, joka koostuu k:sta kitkattomasti toisiinsa liitetystä jäykästä tangosta. Olkoot tankojen pituudet $d_1, \ldots, d_k > 0$ ja massat $m_1, \ldots, m_k > 0$. Tankojen kiinnityspisteet olkoot $P_0, P_1, \ldots, P_k$, joita vastaa kolmiulotteinen koordinaattiesitys $P_i = (\alpha_i, \beta_i, \zeta_i)$. Kuvassa 2.3 tankoja on neljä kappaletta eli $k = 4$. Reunojen kiinnityspisteille asetetaan ehdot

$$P_0 = (\alpha_0, \beta_0, \zeta_0) = (0, 0, 0),\ P_k = (\alpha_k, \beta_k, \zeta_k) = (0, D, 0),$$

missä D on katenoidin reunapisteiden etäisyys toisistaan. Katenoidin potentiaalienergialle E saadaan lauseke

$$E(P_1, \ldots, P_k) = \sum_{i=1}^{k} g m_i \frac{\alpha_i + \alpha_{i-1}}{2}, \tag{2.12}$$

missä $g > 0$ on maan vetovoiman kiihtyvyys. Kiinnityspisteitä koskeviksi rajoitteiksi saadaan

$$\begin{aligned} h_i &= (\alpha_i - \alpha_{i-1})^2 + (\beta_i - \beta_{i-1})^2 + (\zeta_i - \zeta_{i-1})^2 - d_i^2 = 0, \\ &\quad i = 1, \ldots, k. \end{aligned} \tag{2.13}$$

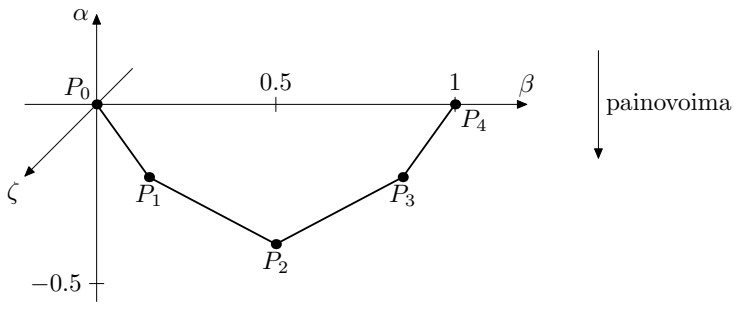
Stabiilissa tasapainotilassa potentiaalienergia E minimoituu.

Koska P_0 ja P_k on kiinnitetty, tuntemattomia kohdemuuttujia ovat koordinaatit $\alpha_i, \beta_i, \zeta_i$, $i = 1, \ldots, k-1$, joita on siis $n = 3(k-1)$ kappaletta. Kootaan tuntemattomat muuttujat vektoriksi

$$\mathbf{x} = (x_1, x_2, \ldots, x_{3(k-1)})^T = (\alpha_1, \ldots, \alpha_{k-1}, \beta_1, \ldots, \beta_{k-1}, \zeta_1, \ldots, \zeta_{k-1})^T.$$

Potentiaalienergian E minimointitehtävän kohdefunktioksi saadaan

$$f(\mathbf{x}) = \frac{g}{2}\left(m_1 x_1 + \sum_{i=2}^{k-1} m_i(x_i + x_{i-1}) + m_k x_{k-1}\right) = \frac{g}{2}\sum_{i=1}^{k-1}(m_i + m_{i+1})x_i,$$



Kuva 2.3: *Neljästä tangosta muodostuvan katenoidin kuvaaja. Tankojen liitoskohdat P_i on merkitty kuvaan, samoin koordinaattiakselit α, β ja ζ.*

ja vakio $g/2$ voidaan jättää pois kohdefunktiosta. Yhtälörajoitteiksi $\mathbf{h}(\mathbf{x})$ saadaan

$$\begin{cases} h_1(\mathbf{x}) = x_1^2 + x_k^2 + x_{2k-1}^2 - d_1^2 = 0, \\ h_i(\mathbf{x}) = (x_i - x_{i-1})^2 + (x_{i+k-1} - x_{i+k-2})^2 + \\ \qquad (x_{i+2k-2} - x_{i+2k-3})^2 - d_i^2 = 0, \quad i = 2, \ldots, k-1, \\ h_k(\mathbf{x}) = x_{k-1}^2 + (D - x_{2(k-1)})^2 + x_{3(k-1)}^2 - d_k^2 = 0. \end{cases}$$

Tämän optimointitehtävän kohdefunktio ja rajoitteet riippuvat muutamasta vektorin $\mathbf{x}$ komponentista. Lisäksi tehtävä on rakenteellinen eli esimerkiksi kohdefunktioiden lausekkeet ovat esitettävissä tiiviissä yleispätevässä muodossa riippumatta tehtävän dimensiosta.

Yllä kuvattu optimointitehtävä ratkaistaan GAMS-ohjelmiston avulla sivulla 46. Toisaalta jos tankojen lukumäärä k on suuri, tehtävä ei ratkea muuten kuin erikoismenetelmillä. Hyvä alkuarvaus tietysti auttaa aina. Yksinkertaisissa tapauksissa (esimerkiksi kun tangot ovat identtisiä) tehtävälle voidaan löytää tarkka analyyttinen ratkaisu [Ves95].

Modernit tietokonearkkitehtuurit perustuvat usein hierarkkiseen muistiin, jolloin muistinkäyttö voi olla pullonkaula, vaikka koneen keskusmuisti riittäisikin. Niin sanotut *lohkoalgoritmit* pyrkivät optimoimaan lineaarialgebran operaatioiden muistinkäyttöä. Esimerkiksi Lapack-ohjelmisto tarjoaa tehokkaita rutiineja useimpiin lineaarialgebran operaatioihin, mutta vain tiheille ja nauhamatriiseille.

Mikäli ratkaistavana on tehtävä, joka ei kerta kaikkiaan mahdu koneen keskusmuistiin, voi tehtävän matriiseille yrittää löytää hajotelmia (tai arvioita), jotka mahtuvat pienempään tilaan. Harvoille matriiseille kannattaa käyttää mahdollisimman tehokasta talletustapaa (kappale 1.4). Rajoitetun muistin sekanttimenetelmissä talletetaan iteraatiomatriisien sijaan pieni joukko vektoreita, joiden avulla voidaan muodostaa uusi hakusuunta. Toisaalta hakusuunnan antava lineaarinen yhtälöryhmä voidaan ratkaista epätarkasti esimerkiksi käyttämällä pohjustettua liittogradienttimenetelmää, jolloin säästetään laskentatyötä.

2.5.2 Rajoitteelliset optimointitehtävät

Varsinkin epälineaarisia rajoitteita sisältävien suurten optimointitehtävien käsittely on hankalaa: vaikka kohdefunktio olisikin harva tai separoituva, voivat rajoitteet tehdä hankalaksi harvoille matriiseille kehitettyjen talletusmuotojen käytön. Esimerkiksi Minos-ohjelmisto käsittelee tyypillisesti tiheitä matriiseja, joiden dimensio on luokkaa muuttujien lukumäärä miinus aktiivisten rajoitteiden lukumäärä. Siten Minosta voi käyttää varsin suurtenkin lineaarisia rajoitteita sisältävien tehtävien ratkaisemiseen, jos suuri osa rajoitteista on aktiivisia.

Lineaarisessa ja kvadraattisessa optimoinnissa ovat *sisäpistemenetelmät* kehittyneet nopeasti viime aikoina. Nämä menetelmät soveltuvat erityisesti suurten tehtävien ratkaisemiseen ja mahdollistavat kenties tulevaisuudessa myös yleisten epälineaaristen tehtävien entistä tehokkaamman ratkaisemisen.

2.6 Ratkaisuohjelmiston valinta

Optimointitehtävien ratkaisemiseen on tarjolla suuri joukko ohjelmistoja. Helpointa on käyttää matemaattisia mallinnuskieliä (esimerkiksi GAMS). Toisaalta erikoistarkoituksiin kehitetyt vaikkapa Fortran-aliohjelmakirjastona käytettävät rutiinit tarjoavat tehokkuutta ja ratkaisemisen kaikkien vaiheiden hallintaa.

Seuraavassa on henkilökohtaisiin kokemuksiin perustuva, suuntaa antava resepti optimointitehtävän ratkaisuohjelmiston valintaan. Luonnollisesti kannattaa aina pyrkiä mahdollisimman luotettavan menetelmän (ja sen toteutuksen!) käyttöön.

Suuriin tehtäviin kehitettyjä koodeja löytyy jonkin verran esimerkiksi TOMS-algoritmeista (liite B). Eräitä näistä koodeista on käytetty esimerkkinä tässä oppaassa. Monet suuret tehtävät sisältävät erikoispiirteitä (esimerkiksi tehtävän rakenteellisuus), joita hyödyntämällä päästään parhaaseen tulokseen. Täten mikään valmisohjelmisto ei välttämättä sovellu sellaisenaan tietyn optimointitehtävän ratkaisemiseen. Paras tulos saavutetaankin yleensä käyttämällä hyväksi tehtävän rakennetta, esimerkiksi kohdefunktion separoituvuutta.

Suositeltavia ohjelmistoja on esitelty liitteessä B. Katso esimerkiksi taulukkoa B.2 sivulla 207.

Matlabin Optimization Toolbox (taulukko B.3 sivulla 209) sekä Mathematica- ja Maple-ohjelmistot tarjoavat interaktiivisen ympäristön algoritmien kehittelyyn, mutta nämä ohjelmistot eivät välttämättä ole tehokkaita isojen tehtävien ratkaisemisessa.

Jos kyseessä on ennen kaikkea mallinnustehtävä, GAMSin tai muun vastaavan mallinnuskielen käyttö on vaivattomin tapa päästä liikkeelle. GAMSia voi myös käyttää erikoisalgoritmeista saatujen tulosten varmistamiseen. GAMSia on käytetty hyvinkin suurten esimerkiksi kansantaloustieteeseen liittyvien optimointitehtävien ratkaisemiseen.

Seuraavassa on esimerkki GAMSin käytöstä optimointitehtävän ratkaisemiseen.

Esimerkki 2.6.1 Listauksen 2.1 GAMS-mallissa minimoidaan esimerkissä 2.5.2 esitellyn äärellisen katenoidin potentiaalienergiaa. Jos oletetaan, että kaikki tangot ovat samanlaista materiaalia, saadaan kunkin tangon massaksi $m_k = \rho A d_k$, missä ρ on materiaalin tiheys, A poikkileikkauksen pinta-ala ja d_k tangon pituus.

Sivulla 44 esitetyssä kuvassa 2.3 tankoja on neljä ja pätee $d_1 = d_4 = 0.25$ metriä ja $d_2 = d_3 = 0.4$ metriä. Optimointitehtäväksi saadaan tällöin:

$$\operatorname*{minimi}_{\mathbf{x}\in\mathbb{R}^9} \rho A \left(\sum_{k=1}^{3} (d_k + d_{k+1}) x_k \right), \tag{2.14}$$

jolloin tehtävän minimipiste ei enää riipu materiaalin tiheydestä ρ tai poikki-

leikkauksen pinta-alasta A. Rajoitefunktioita $h_i(\mathbf{x})$ on neljä:

$$\begin{cases} h_1(\mathbf{x}) = x_1^2 + x_4^2 + x_7^2 - d_1^2 = 0, \\ h_2(\mathbf{x}) = (x_2 - x_1)^2 + (x_5 - x_4)^2 + (x_8 - x_7)^2 - d_2^2 = 0, \\ h_3(\mathbf{x}) = (x_3 - x_2)^2 + (x_6 - x_5)^2 + (x_9 - x_8)^2 - d_3^2 = 0, \\ h_4(\mathbf{x}) = x_3^2 + (D - x_6)^2 + x_9^2 - d_4^2 = 0. \end{cases} \tag{2.15}$$

Tehtävää kuvaava GAMS-malli on hyvä esimerkki mallinnuskielen eduista: minimointitehtävä voidaan esittää muodossa, joka muistuttaa tehtävän matemaattista mallia. Toisaalta tehtävän dimension kasvaessa mallinnuskielen käyttö voi johtaa tehottomuuteen tehtävän ratkaisemisessa. Tällöin saattaa tehtävään sopivan ratkaisurutiinin käyttö olla suositeltavampaa.

Minimointitehtävän ratkaisu on tässä esitetyssä tapauksessa (koordinaatit on ilmoitettu metreissä):

$$\begin{cases} P_1 = (-0.203, 0.146, 0), \quad P_2 = (-0.390, 0.5, 0), \\ P_3 = (-0.203, 0.854, 0). \end{cases}$$

Kuvassa 2.3 on katenoidi esitetty energian minimoivassa tilassa. Koska kaikki kiinnityspisteet ovat samassa tasossa, voitaisiin z-koordinaatit eliminoida optimointitehtävästä, jolloin ratkaiseminen tapahtuisi tehokkaammin.

Jos kyseessä on standardimuotoon saatettu optimointitehtävä, voi useimmiten käyttää valmiita ratkaisuohjelmistoja. Oppaassa on käytetty Lamps-

Listaus 2.1: *Äärellisen katenoidin potentiaalienergiaa minimoiva GAMS-malli. Katso esimerkkejä 2.5.2 ja 2.6.1 sekä kuvaa 2.3. GAMSin notaatio / 1*3 / tarkoittaa indeksijoukkoa* $\{1, 2, 3\}$*. Yhtäsuuruudesta käytetään merkintää =E=. Merkinnällä* `x.fx` *asetetaan muuttujajoukon* `x` *arvoja vakioiksi.*

```
SETS i Koordinaatit / 1*3 /, k Rajoitteet / 1*4 /;

FREE VARIABLES x(i,k)  Koordinaattipisteet
               Energia Minimoitava potentiaalienergia;

SCALAR Leveys  Katenoidin kokonaisleveys / 1 /;
PARAMETER d(k) Pituudet / 1 0.25, 2 0.4, 3 0.4, 4 0.25 /;

EQUATIONS Obj Potentiaalienergian lauseke, h(k) Rajoitteet;

Obj.. Energia =E= SUM(k, (d(k) + d(k+1))*x("1",k));
h(k).. (x("1",k)-x("1",k-1))**2 + (x("2",k)-x("2",k-1))**2 +
                  (x("3",k)-x("3",k-1))**2 - d(k)**2 =E= 0;
x.fx("1","4") = 0.0; x.fx("2","4") = Leveys;
x.fx("3","4") = 0.0;

MODEL Kateno / ALL /;
SOLVE Kateno MINIMIZING Energia USING NLP;
```

ohjelmistoa lineaaristen optimointitehtävien ja sekalukutehtävien ratkaisemiseen. Minos-ohjelmistoa käytetään lineaaristen optimointitehtävien ja rajoitteita sisältävien epälineaaristen optimointitehtävien ratkaisemiseen.

Jos tarvitaan oman koodin yhteyteen optimointitehtävän ratkaisurutiineja, voi käyttää esimerkiksi aliohjelmakirjastoja IMSL tai NAG (taulukot B.4 ja B.5). Nämä ohjelmistot sisältävät korkeatasoiset käsikirjat ja avustusjärjestelmät, joiden avulla on melko helppo löytää oikea menetelmä ja rutiini. Toisaalta useimmat näiden aliohjelmakirjastojen rutiinit on tarkoitettu pieniin ja keskisuuriin tehtäviin. Erittäin suurissa optimointitehtävissä tehokkuus voi olla melko huono.

Jos valmisohjelmistojen menetelmät eivät sovellu ratkaistavalle tehtävälle tai ei haluta käyttää tai ostaa kaupallista ohjelmistoa, voi kokeilla esimerkiksi verkkoarkistoista löytyviä algoritmien lähdekoodeja. Erityisesti Netlibistä löytyviä TOMS-algoritmeja voi suositella (taulukko B.6 sivulla 214). Netlibistä löytyy lisäksi Minpack- ja Lapack-ohjelmistot. Verkkoarkistojen käyttöä helpottavat erilaiset hakumekanismit, joita on esitelty CSC:n oppaassa *Matemaattiset ohjelmistot* [HHL+03] ja tämän oppaan liitteessä E.

Jos sopivaa algoritmia ei vieläkään löydy, kannattaa etsiä sarjajulkaisuista tai kokoelmateoksista sopivaa katsaustyyppistä artikkelia ja kirjallisuusviitteiden avulla jäljittää algoritmien kuvauksia. Joskus voi jopa löytää teoksen, jossa on listattuna algoritmien lähdekoodeja (esimerkkinä *Numerical Recipes* -teokset [PTVF92a, PTVF92b]). Tarkista huolella koodien toimivuus ja luotettavuus!

Eri ohjelmistojen helppo- tai vaikeakäyttöisyys vaihtelee suuresti. Kaupallisten yritysten tarjoamat ohjelmistot ovat yleensä melko helppokäyttöisiä käsikirjojen, avustusjärjestelmien ja malliesimerkkien ansiosta. Monet vapaasti saatavat koodit puolestaan saattavat olla kokonaan ilman dokumentaatiota, lukuunottamatta ohjelmakoodin yhteyteen kirjoitettua joskus hankalasti ymmärrettävää kommentointia. Valitettavasti ohjelmakoodien kirjoittajat usein pitävät käytettyä optimointitehtävän esitysmuotoa (esimerkiksi Hessen matriisin talletustapaa) ja muita vastaavia asioita itsestään selvinä, joten alkuun pääseminen voi olla hankalaa. Netlibin sisältämät TOMS-algoritmit ovat yleensä varsin hyvin dokumentoituja ja niiden käytöstä löytyy melko hyvätasoisia esimerkkejä.

2.7 Lisätietoja

Optimoinnin matemaattinen teoria on varsin vakiintunutta, joten asiasta kiinnostunut voi perehtyä aihepiiriin lukuisien oppikirjojen ja katsausartikkelien avulla (katso lukujen 1 ja 3 kirjallisuusluetteloita sivuilla 31 ja 73).

Numeerisia ratkaisumenetelmiä esitellään teoksissa *Practical Methods of Optimization* [Fle87], *Numerical Linear Algebra and Optimization* [GMW91], *Practical Optimization* [GMW81] ja *Epälineaarinen optimointi* [Mie98]. Numeeristen menetelmien puolella tapahtuu jatkuvaa kehitystä, ja aikaisemmin epärealistisina hylätyt menetelmät saattavat myöhemmin osoittautua

käyttökelpoisiksi. Hyvä esimerkki on Karmarkarin lineaaristen optimointitehtävien ratkaisuun kehittämän menetelmän [Kar84] vaikutus sisäpistemenetelmien kehitykseen.

Optimointitehtävien ratkaisumenetelmän valintaa on käsitelty esimerkiksi teoksissa *Nonlinear Programming: Theory and Algorithms* [BSS93] ja *Introduction to Non-linear Optimization* [Sca85]. Laskentatyön määrää on analysoitu teoksissa [Col84, GMW91, SF92].

Ohjelmistoihin ja algoritmeihin liittyvissä ongelmissa kannattaa kysellä tietoja kyseisen alueen asiantuntijoilta (esimerkiksi CSC:stä, liite E). Ohjelmistojen kuvauksia löytyy eri tietokonevalmistajien ohjelmaluetteloista ja CSC:n oppaasta *Matemaattiset ohjelmistot* [HHL+03]. Numeeristen menetelmien käyttöön johdattaa CSC:n opas *Numeeriset menetelmät käytännössä* [HHL+02].

3 Optimoinnin perusteita

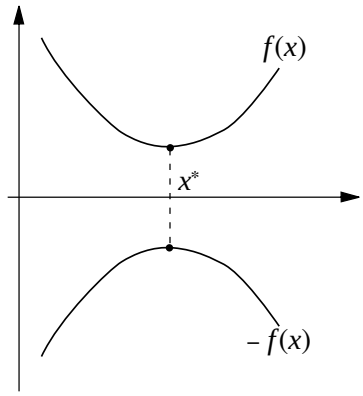
Tässä luvussa esitetään lyhyesti optimointitehtävien matemaattista taustaa ja havainnollistetaan optimoinnin keskeisiä käsitteitä. Useimmista lauseista on joko annettu todistukset tai hahmoteltu niiden taustaa. Lisätietoja saa tämän luvun lopussa mainituista teoksista.

3.1 Optimointitehtävät

Avaruuden $\mathbb{R}^n$ optimointitehtävissä haetaan minimi- tai maksimiarvoa (jatkuvalle) funktiolle f, joka riippuu yhdestä tai useammasta muuttujasta x_i, $i = 1, \ldots, n$. Minimointi- ja maksimointitehtävien minimi- ja maksimiarvoille pätee tietysti

$$\operatorname*{minimi}_{\mathbf{x}} f(\mathbf{x}) = -\operatorname*{maksimi}_{\mathbf{x}} \left(-f(\mathbf{x})\right). \tag{3.1}$$

Minimoinnin ja maksimoinnin yhteyttä on havainnollistettu kuvassa 3.1.



Kuva 3.1: *Funktion $f(x)$ minimi ja funktion $-f(x)$ maksimi löytyvät samasta pisteestä x^*.*

Optimointavalla kohdevektorille $\mathbf{x}$ voidaan asettaa myös rajoitteita. Oman alueensa muodostavat luvussa 5 esiteltävät kokonaislukutehtävät.

Tämän kirjan esimerkeissä on yleensä ratkaistavana minimointitehtävä. Ratkaisu pitää luonnollisesti löytää mahdollisimman tehokkaasti. Lisäksi varsinkin isoissa tehtävissä tietokoneen keskusmuistin koko vaikuttaa ratkaisualgoritmin valintaan.

Seuraavassa esitellään lyhyesti optimoinnissa käytettävää terminologiaa ja optimoinnin matemaattista taustaa. Useimpien lauseiden todistukset on esitetty tiiviissä muodossa, ja ne voi halutessaan sivuuttaa ensimmäisellä lukukerralla. Luvun lopussa on annettu kirjallisuusviitteitä, jotka helpottavat optimointioppiin perehtymistä.

3.2 Optimaalisuustarkasteluja

Aluksi käsitellään optimoinnin yleisiä käsitteitä ja erityisesti yhden muuttujan funktioiden optimointia.

Määritelmä 3.2.1 (Lokaali ja globaali minimi) Funktiolla f on *lokaali minimi* pisteessä $\mathbf{x}^* \in \mathbb{R}^n$, jos on olemassa ϵ-säteinen ympäristö, jossa funktio saavuttaa pienimmän arvonsa keskipisteessä $\mathbf{x}^*$. Määritellään ϵ-pallo U_ϵ:

$$U_\epsilon(\mathbf{x}^*) = \{\, \mathbf{x} \in \mathbb{R}^n \mid \|\mathbf{x} - \mathbf{x}^*\| < \epsilon \,\},\ \epsilon > 0.$$

Lokaalissa minimissä $\mathbf{x}^*$ on siis olemassa skalaari $\epsilon > 0$ siten, että

$$f(\mathbf{x}^*) \leq f(\mathbf{x}) \text{ kaikilla } \mathbf{x} \in U_\epsilon(\mathbf{x}^*). \tag{3.2}$$

Globaali minimi $\mathbf{x}^*$ on pienin lokaaleista minimeistä eli $f(\mathbf{x}^*) \leq f(\mathbf{x})$ kaikilla $\mathbf{x} \in \mathbb{R}^n$.

Yhden muuttujan funktiolle $f(x), x \in \mathbb{R}$ pätee seuraava lause:

Lause 3.2.1 Olkoon $f(x)$ jatkuvasti differentioituva alhaalta rajoitettu funktio välillä $[x_l, x_u]$. Jos x^* on funktion f lokaali minimi tai maksimi, on joko $x^* = x_l$ tai $x^* = x_u$ tai $f'(x^*) = 0$.

Todistus: Olkoon $x^* \in (x_l, x_u)$ lokaali minimipiste. Tällöin on

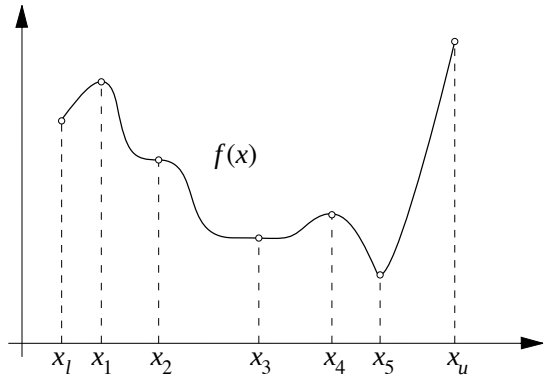
$$f'(x^*) = \lim_{x \to x^*} \frac{f(x) - f(x^*)}{x - x^*}.$$

Koska x^* on lokaali minimi, on $f(x) - f(x^*) \geq 0$, kun x on kyllin lähellä pistettä x^*. Tällöin saadaan

$$\frac{f(x) - f(x^*)}{x - x^*} \geq 0, \text{ kun } x^* < x, \quad \frac{f(x) - f(x^*)}{x - x^*} \leq 0, \text{ kun } x^* > x. \tag{3.3}$$

Kun $x \to x^*$, saadaan $f'(x^*) \geq 0$ ja $f'(x^*) \leq 0$ eli $f'(x^*) = 0$. □

Kuvassa 3.2 on esimerkki yhden muuttujan funktion $f(x)$ käyttäytymisestä. Funktio on määritelty välillä $[x_l, x_u]$, ja funktiolla on useita ääriarvokohtia. Kohdassa x_5 saavutetaan kaikkein pienin arvo eli globaali minimi välillä $[x_l, x_u]$. Globaali maksimi saavutetaan reunapisteessä x_u. Reunapiste



Kuva 3.2: *Erään skalaarifunktion $f(x)$ käyttäytyminen välillä $[x_l, x_u]$.*

x_l on funktion lokaali minimi. Pistettä x_2 kutsutaan *käännepisteeksi* (point of inflection), sillä funktion toinen derivaatta vaihtaa merkkiään kyseisessä pisteessä. Kohdassa x_2 on myös funktion derivaatan nollakohta, muttei minimiä tai maksimia.

Määritelmä 3.2.2 (Vahva ja heikko minimi) Funktion $f(\mathbf{x})$ minimi $\mathbf{x}^*$ on *vahva*, jos funktion arvot kasvavat lähdettäessä minimipisteestä mitä tahansa käyrää $\mathbf{x}(t)$ pitkin. *Heikossa minimissä* funktion arvot pysyvät samoina ainakin yhdellä minimipisteen kautta kulkevalla käyrällä.

Kuvassa 3.2 on piste x_3 heikko lokaali minimi. Kohdasta x_5 löytyy funktion $f(x)$ vahva minimi.

Lause 3.2.2 (Taylorin lause) Kahdesti jatkuvasti differentioituvalle funktiolle $f : \mathbb{R} \mapsto \mathbb{R}$ on olemassa kaikilla x_1 ja $x_2 \in \mathbb{R}$ välin (x_1, x_2) piste x siten, että

$$f(x_2) = f(x_1) + f'(x_1)(x_2 - x_1) + \frac{f''(x)}{2}(x_2 - x_1)^2. \tag{3.4}$$

Soveltamalla Taylorin lausetta differentioituvaan yhden muuttujan funktioon $f : \mathbb{R} \mapsto \mathbb{R}$ saadaan optimipisteessä x^*

$$f(x) = f(x^*) + f'(x^*)(x - x^*) + \frac{f''(z)}{2}(x - x^*)^2, \tag{3.5}$$

missä piste z on pisteiden x ja x^* välissä. Merkitsemällä $d = x - x^*$ saadaan

$$f(x^* + d) \approx f(x^*) + f'(x^*)d + \frac{f''(x^*)}{2}d^2. \tag{3.6}$$

Skalaarifunktion optimaalisuustarkastelussa voidaan siis käyttää myös toista derivaattaa $f''(x)$.

Lause 3.2.3 Olkoon yhden muuttujan funktio $f(x)$ kahdesti jatkuvasti differentioituva. Jos ensimmäisen derivaatan nollakohdassa $f'(x^*) = 0$ pätee epäyhtälö $f''(x^*) > 0$, on kyseessä funktion vahva lokaali minimi. Toisaalta jos

$$f''(x) \geq 0 \text{ kaikilla } x \in [x_l, x_u] \subset \mathbb{R},$$

on piste x^* funktion f globaali minimi.

Todistus: Jos $x \in [x_l, x_u]$ ja $x \neq x^*$, saadaan Taylorin kaavasta (3.5) oletuksella $f'(x^*) = 0$ yhtälö

$$f(x) = f(x^*) + \frac{f''(z)}{2}(x - x^*)^2,$$

missä z on pisteiden x ja x^* välissä. Jos $f''(x) \geq 0$ kaikilla $x \in [x_l, x_u]$, on $f(x) \geq f(x^*)$ ja siten x^* on globaali minimi. Jos taas $f''(x^*) > 0$, saadaan jatkuvuuden perusteella

$$f(x) > f(x^*) \text{ kaikilla } x \in (x^* - \epsilon, x^* + \epsilon),\ x \neq x^*,\ \epsilon > 0.$$

Täten x^* on vahva lokaali minimi. □

Funktion maksimikohtaa etsittäessä käännetään lauseen 3.2.3 epäyhtälöiden suunta.

Differentioituvan yhden muuttujan funktion 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.

Määritelmä 3.2.3 (Kriittinen piste) Funktion $f(\mathbf{x})$ *kriittisiksi pisteiksi* $\mathbf{x}^c$ sanotaan yhtälön

$$\nabla f(\mathbf{x}) = \begin{pmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} \\ \vdots \\ \frac{\partial f(\mathbf{x})}{\partial x_n} \end{pmatrix} = \mathbf{0} \tag{3.7}$$

ratkaisuja.

Usean muuttujan funktion kriittinen piste vastaa yhden muuttujan funktion derivaatan nollakohtaa $f'(x) = 0$. Kriittisen pisteen optimaalisuutta voidaan tutkia funktion toisista derivaatoista muodostetun Hessen matriisin $H(\mathbf{x})$ avulla:

$$H(\mathbf{x}) = \nabla\nabla^T f(\mathbf{x}) = \nabla\nabla f(\mathbf{x}) = \begin{pmatrix} \frac{\partial^2 f(\mathbf{x})}{\partial^2 x_1} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_n} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial^2 x_n} \end{pmatrix}. \tag{3.8}$$

Yhden muuttujan funktion tapauksessa tutkittiin toisen derivaatan merkkiä minimi- tai maksimikohdan löytämiseksi. Funktion $f(\mathbf{x})$ toisista derivaatoista muodostetun matriisin $H(\mathbf{x})$ positiivisuutta tai negatiivisuutta ei

voi tutkia samalla tavalla, koska kyseessä on matriisi. Sen sijaan käytetään matriisin *definiittisyyttä*, joka on määritelty kappaleessa 1.4 (sivu 23).

Usean muuttujan differentioituvaa funktiota $f(\mathbf{x}) : \mathbb{R}^n \mapsto \mathbb{R}$ voidaan arvioida Taylorin sarjalla (vertaa lauseseen 3.2.2)

$$\begin{aligned} f(\mathbf{x}^* + \mathbf{d}) &= f(\mathbf{x}^*) + \langle \mathbf{d}, \nabla f(\mathbf{x}^*) \rangle + \tfrac{1}{2} \langle \mathbf{d}, H(\mathbf{z})\, \mathbf{d} \rangle \\ &\approx f(\mathbf{x}^*) + \langle \mathbf{d}, \nabla f(\mathbf{x}^*) \rangle + \tfrac{1}{2} \langle \mathbf{d}, H(\mathbf{x}^*)\, \mathbf{d} \rangle, \end{aligned} \tag{3.9}$$

missä 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.

Lause 3.2.4 Olkoon funktio $f(\mathbf{x})$ jatkuvasti differentioituva. Tarkastellaan minimointitehtävää

$$\operatorname*{minimi}_{\mathbf{x}} f(\mathbf{x}),$$

missä vektori $\mathbf{x}$ kuuluu avaruuteen $\mathbb{R}^n$. Tällaista tehtävää kutsutaan *rajoitteettomaksi minimointitehtäväksi*. Jos piste $\mathbf{x}^*$ on minimointitehtävän ratkaisu, on voimassa yhtälö

$$\nabla f(\mathbf{x}^*) = \mathbf{0}. \tag{3.10}$$

Siis vain kriittinen piste voi olla funktion ääriarvokohta.

Todistus: Osoitetaan, että $\partial f(\mathbf{x}^*)/\partial x_i = 0$, $i = 1, 2, \ldots, n$. Valitaan $i = 1$. Olkoon

$$g(x) = f(x, {x_2}^*, \ldots, {x_n}^*).$$

Funktio $g(x)$ on differentioituva ja lisäksi jatkuvuuden perusteella

$$g({x_1}^*) \leq g(x) \text{ kaikilla } x \in ({x_1}^* - \epsilon, {x_1}^* + \epsilon),\ \epsilon > 0.$$

Siis ${x_1}^*$ on funktion $g(x)$ lokaali minimi ja on oltava $g'({x_1}^*) = 0$. Täten

$$g'({x_1}^*) = \left. \frac{\partial f({x_1}, {x_2}^*, \ldots, {x_n}^*)}{\partial x_1} \right|_{x_1 = {x_1}^*} = \left. \frac{\partial f(\mathbf{x})}{\partial x_1} \right|_{\mathbf{x} = \mathbf{x}^*} = 0.$$

Sama päättely voidaan tehdä kaikille $i = 1, \ldots, n$. □

Lause 3.2.5 Olkoon piste $\mathbf{x}^c$ kaksi kertaa jatkuvasti differentioituvan funktion $f(\mathbf{x})$ kriittinen piste. Mikäli funktion f Hessen matriisi $H(\mathbf{x}^c)$ on positiivisesti definiitti eli

$$\langle \mathbf{d}, H(\mathbf{x}^c)\, \mathbf{d} \rangle > 0 \text{ kaikilla } \mathbf{d} \in \mathbb{R}^n, \mathbf{d} \neq \mathbf{0}, \tag{3.11}$$

on piste $\mathbf{x}^c$ funktion f vahva lokaali minimi. Mikäli $H(\mathbf{x})$ on positiivisesti semidefiniitti kaikilla $\mathbf{x} \in \mathbb{R}^n$, on kriittinen piste $\mathbf{x}^c$ funktion $f(\mathbf{x})$ globaali minimi.

Todistus: Lause voidaan johtaa yhtälön (3.9) perusteella. Koska kriittisessä pisteessä pätee $\nabla f(\mathbf{x}^c) = \mathbf{0}$, saadaan

$$f(\mathbf{x}^c + \mathbf{d}) = f(\mathbf{x}^c) + \frac{1}{2} \langle \mathbf{d}, H(\mathbf{z})\, \mathbf{d} \rangle,$$

missä piste $\mathbf{z}$ määritellään $\mathbf{z} = \mathbf{x}^c + \lambda\mathbf{d},\ \lambda \in (0,1)$. Täten esimerkiksi

$$f(\mathbf{x}^c + \mathbf{d}) - f(\mathbf{x}^c) = \frac{1}{2}\langle \mathbf{d}, H(\mathbf{z})\,\mathbf{d}\rangle > 0$$

ja siten epäyhtälö $f(\mathbf{x}^c + \mathbf{d}) > f(\mathbf{x}^c)$ pätee kaikilla $\mathbf{d} \neq \mathbf{0}$. □

Funktion $f(\mathbf{x})$ Hessen matriisin $H(\mathbf{x}) = \nabla\nabla^T f(\mathbf{x})$ täytyy siis olla positiivisesti definiitti (riittävä ehto) tai positiivisesti semidefiniitti (välttämätön ehto), jotta kriittinen piste $\mathbf{x}^c$ olisi lokaali minimi. Funktion maksimille pätee vastaava ehto kuin lause 3.2.5, mutta Hessen matriisin tulee olla negatiivisesti definiitti.

Esimerkki 3.2.1 Funktion $f(\mathbf{x}) = x_1^4 + x_2^2$ kriittinen piste on $\mathbf{x}^c = \mathbf{0}$, jolloin saadaan Hessen matriisiksi

$$H(\mathbf{x}^c) = \begin{pmatrix} 0 & 0 \\ 0 & 2 \end{pmatrix}.$$

Hessen matriisi on positiivisesti semidefiniitti, ja tässä tapauksessa piste $\mathbf{x}^c = \mathbf{0}$ on myös globaali minimi, sillä $f(\mathbf{x}) \geq 0 = f(\mathbf{0})$.

Esimerkki 3.2.2 Kvadraattiselle funktiolle $f(\mathbf{x}) = \frac{1}{2}\langle \mathbf{x}, A\mathbf{x}\rangle + \langle \mathbf{b}, \mathbf{x}\rangle + c$, missä $\mathbf{x} \in \mathbb{R}^n$, A on symmetrinen $n \times n$ -matriisi ja c on skalaarivakio, saadaan gradienttivektoriksi $\nabla f(\mathbf{x})$ ja Hessen matriisiksi $H(\mathbf{x})$:

$$\begin{aligned} \nabla f(\mathbf{x}) &= A\mathbf{x} + \mathbf{b}, \\ H(\mathbf{x}) &= A. \end{aligned}$$

Täten vaikkapa vain paikallisesti kvadraattisesti käyttäytyvien funktioiden käsittely on melko vaivatonta, sillä tarvitaan vain lineaarialgebraa. Epälineaaristen funktioiden kvadraattinen approksimointi onkin tärkeä väline kehiteltäessä optimointialgoritmeja.

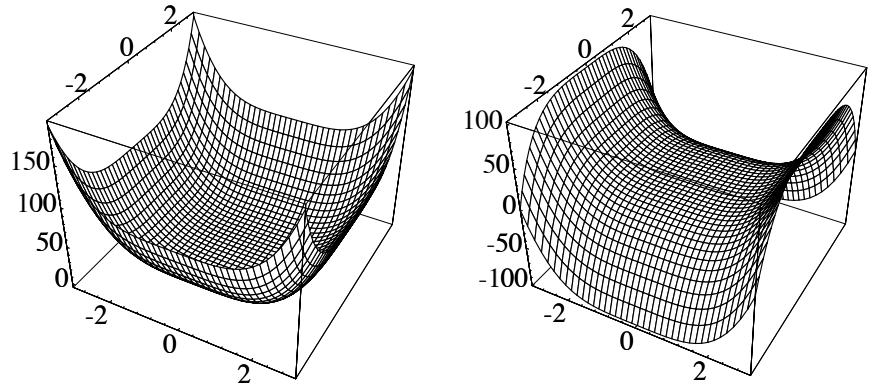
Määritelmä 3.2.4 (Satulapiste) Funktion *satulapisteestä* löytyy funktion lokaali minimi kuljettaessa tiettyyn suuntaan, mutta johonkin toiseen suuntaan kuljettaessa kyseessä onkin lokaali maksimi.

Kriittinen piste ei ole välttämättä minimi- tai maksimikohta. Esimerkiksi satulapiste täyttää ehdon (kuva 3.3).

Lause 3.2.6 Olkoon funktio f kahdesti jatkuvasti differentioituva kuvaus $\mathbb{R}^n \to \mathbb{R}$. Jos piste $\mathbf{x}$ on funktion $f(\mathbf{x})$ kriittinen piste ja lisäksi Hessen matriisi $H(\mathbf{x}) = \nabla\nabla^T f(\mathbf{x})$ on indefiniitti, on piste $\mathbf{x}$ satulapiste funktiolle f.

Todistus: Hessin matriisin indefiniittisyys pisteessä $\mathbf{x}$ tarkoittaa, että on olemassa vektorit $\mathbf{d}^1$ ja $\mathbf{d}^2 \in \mathbb{R}^n$, joille pätee

$$\langle \mathbf{d}^1, H(\mathbf{x})\,\mathbf{d}^1\rangle < 0 \quad \text{ja} \quad \langle \mathbf{d}^2, H(\mathbf{x})\,\mathbf{d}^2\rangle > 0.$$



Kuva 3.3: *Vasemmalla esitetyllä funktiolla $x_1^4 + x_2^4$ on vahva (globaali) minimi pisteessä $(0,0)^T$. Funktiolla $x_1^4 - x_2^4$ on puolestaan satulapiste kohdassa $(0,0)^T$.*

Koska $f(\mathbf{x})$ on kahdesti differentioituva, on olemassa luku $\epsilon > 0$ siten, että

$$\langle \mathbf{d}^1, H(\mathbf{x} + t\,\mathbf{d}^1)\,\mathbf{d}^1 \rangle < 0 \quad \text{ja} \quad \langle \mathbf{d}^2, H(\mathbf{x} + t\,\mathbf{d}^2)\,\mathbf{d}^2 \rangle > 0,$$

kun parametri t on kyllin pieni eli $|t| < \epsilon$. Määritellään funktiot

$$d_1(t) = f(\mathbf{x} + t\,\mathbf{d}^1) \quad \text{ja} \quad d_2(t) = f(\mathbf{x} + t\,\mathbf{d}^2).$$

Nyt $d_1'(0) = d_2'(0) = 0$ ja $d_1''(0) = \langle \mathbf{d}^1, H(\mathbf{x})\,\mathbf{d}^1 \rangle < 0$ sekä $d_2''(0) > 0$. Siis kohta $t = 0$ on funktion d_1 lokaali maksimi ja funktion d_2 lokaali minimi. Siten piste $\mathbf{x}$ on funktion f satulapiste. □

Hessen matriisin indefiniittisyys kriittisessä pisteessä on riittävä ehto satulapisteelle. Toisaalta satulapiste voi löytyä myös pisteestä, jossa Hessen matriisi on semidefiniitti.

Esimerkki 3.2.3 Kuvassa 3.3 on esitetty funktion $x_1^4 - x_2^4$ käyttäytyminen. Kyseisen funktion kriittinen piste $\mathbf{x}^c$ saadaan yhtälöstä

$$\nabla f(\mathbf{x}^c) = \begin{pmatrix} 4x_1^3 \\ 4x_2^3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

eli $\mathbf{x}^c = \mathbf{0}$. Kyseisessä pisteessä saadaan Hessen matriisiksi

$$H(\mathbf{x}^c) = \nabla\nabla f(\mathbf{x}^c) = \begin{pmatrix} 12x_1^2 & 0 \\ 0 & 12x_2^2 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

eli $H(\mathbf{x}^c)$ on positiivisesti semidefiniitti. Kyseessä on funktion $x_1^4 - x_2^4$ satulapiste. Toisaalta esimerkiksi funktiolle $x_1^4 + x_2^4$ piste $\mathbf{x}^c = \mathbf{0}$ on globaali minimi ja tässä pisteessä on

$$H(\mathbf{x}^c) = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

Hessen matriisin semidefiniittisyys annetussa pisteessä ei siis ole riittävä ehto minimille tai satulapisteelle.

Yksinkertaiset optimointitehtävien ratkaisualgoritmit eivät välttämättä kykene hakeutumaan pois satulapisteestä. Toisaalta satulapisteessä $\mathbf{x}$ voi esiintyä johonkin suuntaan $\mathbf{p}$ *negatiivista kaarevuutta* eli on olemassa $\mathbf{p} \in \mathbb{R}^n$, jolle pätee

$$\langle \mathbf{p}, H(\mathbf{x})\,\mathbf{p} \rangle < 0 \text{ ja } f(\mathbf{x}+\mathbf{p}) < f(\mathbf{x}), \tag{3.12}$$

missä $H(\mathbf{x}) = \nabla\nabla f(\mathbf{x})$ on kohdefunktion Hessen matriisi pisteessä $\mathbf{x}$.

Määritelmä 3.2.5 (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.

Optimointitehtävien yhteydessä ratkaisumenetelmän *stabiilisuudella* tarkoitetaan algoritmin globaalia suppenemista. Joillekin algoritmeille voidaan todistaa globaali suppeneminen, kun funktioiden epälineaarisuus ei ole kovin suurta.

Huomautus 3.2.1 Optimointitehtävän globaalin minimin löytymistä ei voida taata kuin korkeintaan tilastollisessa mielessä (haetaan suuri määrä ratkaisuehdokkaita eri alkuarvauksilla) tai riittävän vahvoilla oletuksilla funktion ja rajoitteiden käyttäytymisestä (kappale 9.1 sivulla 154).

Siis vaikka algoritmi olisikin stabiili (suppenisi kaikilla alkuarvauksilla), voi löydetty optimikohta olla vain lokaali minimi, ellei tehtävälle voida osoittaa esimerkiksi konveksisuutta, jota käsitellään kappaleessa 3.4 (sivu 65).

3.3 Rajoitteelliset optimointitehtävät

Usein optimointitehtävä sisältää rajoitteita parametrivektorille $\mathbf{x}$. Kyseessä voi olla esimerkiksi maksimikustannus tai optimoitavaa systeemiä koskeva fysikaalinen rajoitus.

Rajoitteellisen (constrained) jatkuvasti differentioituvan optimointitehtävän perustyyppi on

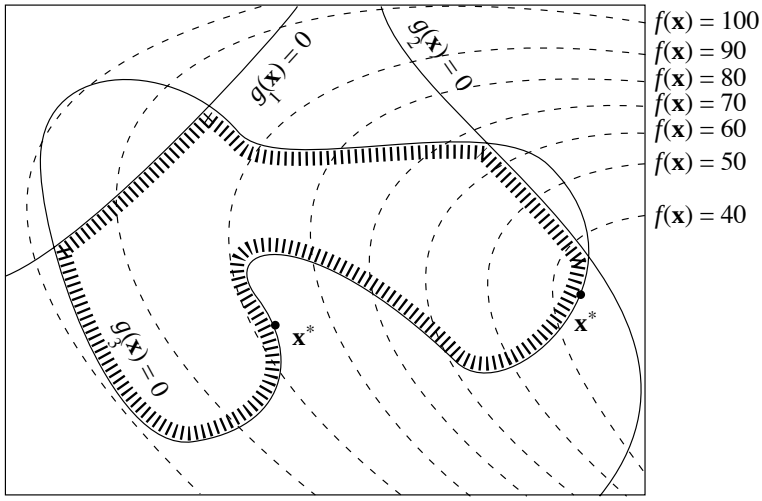
$$\underset{\mathbf{x}}{\text{minimi}}\ f(\mathbf{x}),\ \text{ kun } \mathbf{g}(\mathbf{x}) \le \mathbf{0} \text{ ja } \mathbf{h}(\mathbf{x}) = \mathbf{0}, \tag{3.13}$$

missä $f : \mathbb{R}^n \mapsto \mathbb{R}$ on minimoitava *kohdefunktio*. Vektoriarvoinen funktio $\mathbf{g} : \mathbb{R}^n \mapsto \mathbb{R}^p$ muodostaa *epäyhtälörajoitteet* $\mathbf{g}(\mathbf{x}) \le \mathbf{0}$ ja funktio $\mathbf{h} : \mathbb{R}^n \mapsto \mathbb{R}^q$ *yhtälörajoitteet* $\mathbf{h}(\mathbf{x}) = \mathbf{0}$.

Määritelmä 3.3.1 (Käyvät pisteet ja käypä alue) Rajoitteet $\mathbf{g}(\mathbf{x}) \le 0$ ja $\mathbf{h}(\mathbf{x}) = 0$ toteuttavia pisteitä $\mathbf{x} \in \mathbb{R}^n$ kutsutaan *käyviksi pisteiksi* ja niiden muodostamaa joukkoa $\mathcal{F}$ kutsutaan *käyväksi alueeksi* (feasible region).

Osa tehtävän rajoitteista voi olla erikoistyyppiä, esimerkkinä *laatikkorajoitteet*

$$\mathbf{x}^l \le \mathbf{x} \le \mathbf{x}^u, \tag{3.14}$$



Kuva 3.4: *Optimointitehtävän epälineaariset rajoitteet. Merkinnällä* $\mathbf{x}^*$ *on osoitettu tehtävän paikalliset minimikohdat. Funktion* $f(\mathbf{x})$ *tasa-arvokäyriä on merkitty katkoviivoilla. Epäyhtälömuotoisten rajoitteiden* $\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$ *käypiä suuntia on merkitty paksulla katkoviivoituksella.*

missä vektorit $\mathbf{x}^l$, $\mathbf{x}$ ja $\mathbf{x}^u$ kuuluvat joukkoon $\mathbb{R}^n$. Osa rajoitevektoreiden $\mathbf{x}^l$ ja $\mathbf{x}^u$ komponenteista voi olla $\pm\infty$. Laatikkorajoitteet ovat erikoistapaus *lineaarisista rajoitteista*, jotka ovat muotoa

$$a_{j1}x_1 + a_{j2}x_2 + \cdots + a_{jn}x_n \leq b_j, \quad j = 1, \ldots, m, \tag{3.15}$$

eli matriisimuodossa

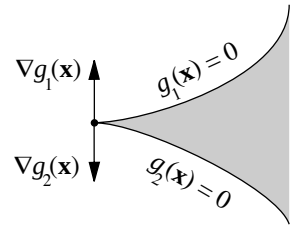
$$A\mathbf{x} \leq \mathbf{b}. \tag{3.16}$$

Jos epäyhtälörajoite on annettu muodossa $\mathbf{g}(\mathbf{x}) \geq \mathbf{0}$, voidaan se muuntaa kertomalla luvulla -1 muotoon $-\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$. Epäyhtälörajoitteesta $\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$ voidaan tehdä yhtälörajoite positiivisten apumuuttujien eli ns. *slack*-muuttujien $\mathbf{z}$ avulla:

$$\mathbf{g}(\mathbf{x}) + \mathbf{z} = \mathbf{0}, \quad \mathbf{z} \geq \mathbf{0}. \tag{3.17}$$

Rajoitteellisen tehtävän lokaali ja globaali minimi määritellään kuten rajoitteettomille tehtäville, mutta lisäksi optimipisteen tulee olla käypä. Tällöin globaalille optimipisteelle $\mathbf{x}^* \in \mathcal{F}$ pätee $f(\mathbf{x}^*) \leq f(\mathbf{x})$ kaikilla $\mathbf{x} \in \mathcal{F}$.

Esimerkki 3.3.1 Kuvassa 3.4 on esitetty kahden muuttujan funktio $f(\mathbf{x})$ ja optimointitehtävän epälineaariset epäyhtälörajoitteet $g_i(\mathbf{x}) \leq 0$, $i = 1, 2, 3$. Kuvan optimointitehtävällä on kaksi paikallista minimiä $\mathbf{x}^*$, jotka kumpikin saavutetaan käyvän alueen reunalla eli kun $g_3(\mathbf{x}) = 0$.



Kuva 3.5: *Säännöllisyysehto ei ole voimassa kuvassa harmaalla värillä merkityn alueen kulmapisteessä.*

Määritelmä 3.3.2 (Käyvät suunnat) Pisteestä $\mathbf{x}$ lähtevillä *käyvillä suunnilla* $\mathcal{D}(\mathbf{x})$ tarkoitetaan joukkoa

$$\mathcal{D}(\mathbf{x}) = \{\ \mathbf{d} \in \mathbb{R}^n \ \mid \text{ on olemassa } \sigma > 0 \text{ siten, että } \sigma \geq \tau \geq 0 \Rightarrow \mathbf{x} + \tau\mathbf{d} \in \mathcal{F}\ \}. \tag{3.18}$$

Siis suunta $\mathbf{d}$ kuuluu käypien suuntien joukkoon $\mathcal{D}(\mathbf{x})$, jos pisteestä $\mathbf{x}$ voidaan kulkea vähän matkaa kyseiseen suuntaan ja pysytään yhä käyvällä alueella (kuva 3.4).

Määritelmä 3.3.3 (Aktiivinen epäyhtälörajoite) Optimointitehtävän epäyhtälörajoitteen $g_i(\mathbf{x}) \leq 0$ sanotaan olevan *aktiivinen* pisteessä $\mathbf{x}$, jos pätee $g_i(\mathbf{x}) = 0$. *Aktiivijoukkomenetelmissä* käytetään hyväksi aktiivisia epäyhtälörajoitteita ratkaistaessa optimointitehtäviä.

Määritelmä 3.3.4 (Säännöllinen piste ja säännöllisyysehto) Käyvän pisteen $\mathbf{x} \in \mathcal{F}$ sanotaan olevan *säännöllinen piste* (regular point), jos aktiivisten rajoitteiden gradienttivektoreista koostuva joukko

$$\{\ \nabla g_i(\mathbf{x}) \ \mid \ i \in \{1, \ldots, p\} \text{ siten että } g_i(\mathbf{x}) = 0\ \} \ \cup \ \{\ \nabla h_j(\mathbf{x}) \ \mid \ j = 1, \ldots, q\ \} \tag{3.19}$$

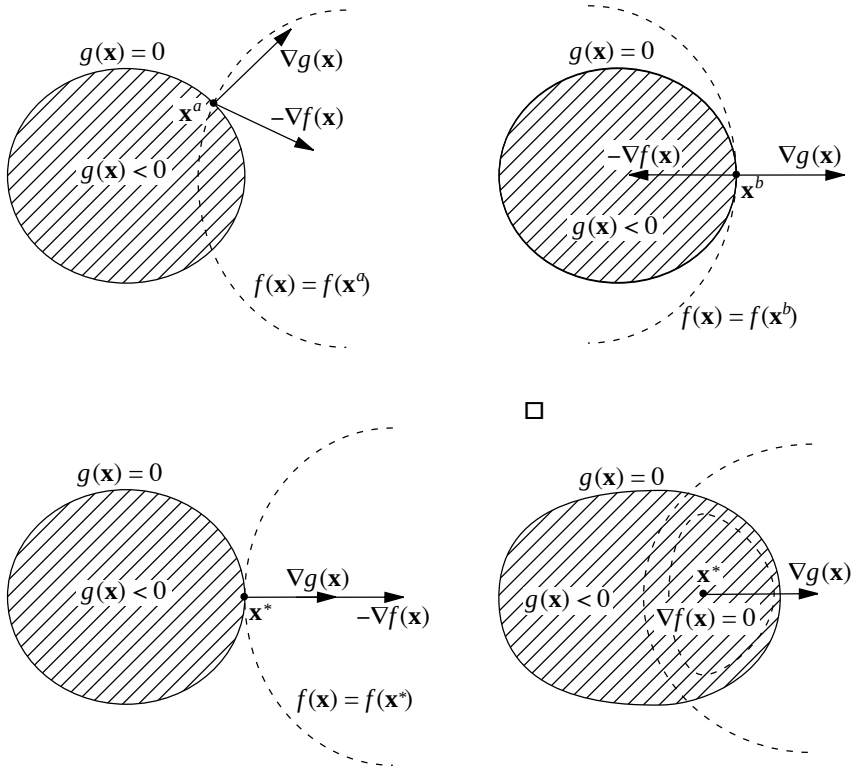
on lineaarisesti riippumaton. Tätä ehtoa kutsutaan usein *säännöllisyysehdoksi*.

Säännöllisyysehto on tärkeä käsite tutkittaessa optimointitehtävän ratkaisupisteen optimaalisuutta. Monet menetelmät perustuvat oletukseen, että iteraatiopisteet toteuttavat säännöllisyysehdon. Kuvassa 3.5 on esitetty tapaus, jossa säännöllisyysehto ei kuitenkaan ole voimassa.

3.3.1 Rajoitteellisten tehtävien minimipisteet

Tutkitaan aluksi epäyhtälömuotoisia rajoitteita $\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$ sisältäviä minimointitehtäviä. Oletetaan, että rajoitteiden gradienttivektorit ovat lineaarisesti riippumattomia (määritelmän 3.3.4 säännöllisyysehto).

Oletetaan, että epäyhtälörajoitteet g_i, $i \in \mathcal{A} \subset \{1, \ldots, p\}$ ovat aktiivisia pisteessä $\mathbf{x}^*$: $g_i(\mathbf{x}^*) = 0$, $i \in \mathcal{A}$. Joukko $\mathcal{A}$ sisältää siis aktiivisten rajoitteiden indeksit.



Kuva 3.6: *Rajoitteellisen minimointitehtävän optimipisteen geometria.*

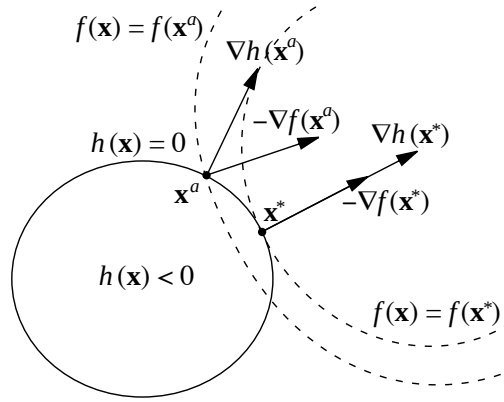
Mikäli pätee $\nabla f(\mathbf{x}^*) \neq \mathbf{0}$, täytyy kohdefunktion gradienttivektori optimipisteessä pystyä esittämään aktiivisten rajoitteiden negatiivisena lineaarikombinaationa: $-\nabla f(\mathbf{x}^*) = \sum_{i \in \mathcal{A}} u_i^* \nabla g_i(\mathbf{x}^*)$, $u_i^* \geq 0$.

Lisäksi pisteen $\mathbf{x}^*$ täytyy olla käypä. Tällöin joko ollaan käyvän alueen reunalla eli $g_i(\mathbf{x}^*) = 0$ tai skalaarikerroin $u_i^* = 0$. Tästä saadaan *komplementaarisuusehto* $u_i^* g_i(\mathbf{x}^*) = 0$.

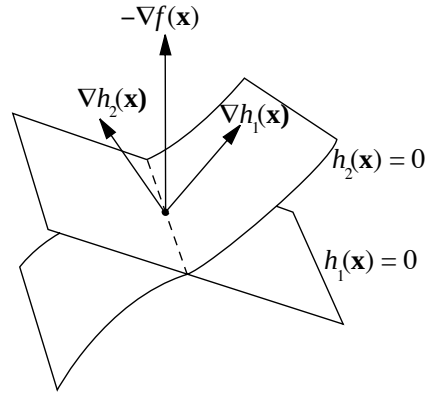
Minimipisteen $\mathbf{x}^*$ optimaalisuusehdot ovat siten

$$\begin{cases} g_i(\mathbf{x}^*) \leq 0, \\ u_i^* g_i(\mathbf{x}^*) = 0, \\ -\nabla f(\mathbf{x}^*) = \sum_{i \in \mathcal{A}} u_i^* \nabla g_i(\mathbf{x}^*), \quad u_i^* \geq 0. \end{cases} \tag{3.20}$$

Kuvassa 3.6 on havainnollistettu rajoitteellisen optimointitehtävän geometriaa. Optimointitehtävällä on epäyhtälörajoite $g(\mathbf{x}) \leq 0$. Kuvaan on piirretty kohdefunktiolle $f(\mathbf{x})$ ja rajoitteelle $g(\mathbf{x})$ tasa-arvokäyrät. Pisteet $\mathbf{x}^a$ ja $\mathbf{x}^b$ eivät ole minimikohtia (kaksi ylintä kuvaa). Piste $\mathbf{x}^*$ on tehtävän lokaali minimi. Jos minimi löytyy alueen sisäpisteestä, on $\nabla f(\mathbf{x}^*) = \mathbf{0}$, muussa tapauksessa on oltava $-\nabla f(\mathbf{x}^*) = \sum_{i \in \mathcal{A}} u_i^* \nabla g_i(\mathbf{x}^*)$, $u_i^* \geq 0$ eli gradientti voidaan esittää aktiivisten rajoitteiden (tässä tapauksessa vain yksi rajoite $g(\mathbf{x}) \leq 0$) negatiivisena lineaarikombinaationa.



Kuva 3.7: *Yhtälörajoitteen $h(\mathbf{x}) = 0$ sisältävää minimointitehtävää havainnollistava kuva.*



Kuva 3.8: *Kuvassa on esitetty yhtälörajoitteet $h_1(\mathbf{x}) = 0$ ja $h_2(\mathbf{x}) = 0$ sisältävä avaruuden $\mathbb{R}^3$ optimointitehtävä.*

Jos optimointitehtävällä on yhtälömuotoisia rajoitteita, optimipisteelle saadaan välttämättömiksi ehdoiksi

$$\begin{cases} h_i(\mathbf{x}^*) = 0, \quad i = 1, \ldots, q, \\ -\nabla f(\mathbf{x}^*) = \sum_{i=1}^{q} v_i{}^* \nabla h_i(\mathbf{x}^*). \end{cases} \tag{3.21}$$

Tässä skalaarikertoimet $v_i{}^*$ voivat olla negatiivisia, positiivisia tai nolla.

Kuvan 3.7 tehtävällä on yhtälörajoite $h(\mathbf{x}) = 0$. Kuvaan on piirretty kohdefunktiolle $f(\mathbf{x})$ kaksi tasa-arvokäyrää. Piste $\mathbf{x}^a$ ei ole minimikohta. Piste $\mathbf{x}^*$ on tehtävän lokaali minimi.

Kuvassa 3.8 on piirretty kohdefunktiolle $f(\mathbf{x})$ pinnat, joilla rajoitefunktioiden h_1 ja h_2 arvot ovat nollia, sekä pisteestä $\mathbf{x}$ lähtevät kohdefunktion

ja rajoitefunktioiden gradienttivektorit. Paikallisessa optimissa gradienttivektori $\nabla f(\mathbf{x})$ on rajoitefunktioiden gradienttien $\nabla h_1(\mathbf{x})$ ja $\nabla h_2(\mathbf{x})$ lineaarikombinaatio.

Seuraavassa kappaleessa laajennetaan tässä esitettyä rajoitteita sisältävien optimointitehtävien analyysiä ottamalla käyttöön *Lagrangen kertoimet*.

3.3.2 Lagrangen funktio rajoitteellisille tehtäville

Rajoitteelliselle optimointitehtävälle (3.13) määritellään *Lagrangen funktio* (Lagrangian) $L(\mathbf{x}, \mathbf{u}, \mathbf{v})$ seuraavasti:

$$\begin{aligned} L(\mathbf{x}, \mathbf{u}, \mathbf{v}) &= f(\mathbf{x}) + \langle \mathbf{u}, \mathbf{g}(\mathbf{x}) \rangle + \langle \mathbf{v}, \mathbf{h}(\mathbf{x}) \rangle \\ &= f(\mathbf{x}) + \left(\textstyle\sum_{j=1}^{p} u_j g_j(\mathbf{x}) \right) + \left(\textstyle\sum_{j=1}^{q} v_j h_j(\mathbf{x}) \right). \end{aligned} \tag{3.22}$$

Kerroinvektoreita $\mathbf{u}$ ja $\mathbf{v}$ kutsutaan tehtävän *Lagrangen kertoimiksi*.

Edellisessä kappaleessa esitettyä päättelyä soveltamalla (kuvat 3.6, 3.7 ja 3.8) voidaan esittää seuraavat ehdot pisteen $\mathbf{x}^*$ optimaalisuudelle.

Lause 3.3.1 Olkoot minimoitava kohdefunktio $f(\mathbf{x})$ ja rajoitefunktiot $\mathbf{g}(\mathbf{x})$ ja $\mathbf{h}(\mathbf{x})$ kahdesti differentioituvia. Jos säännöllinen piste $\mathbf{x}^*$ on rajoitteellisen optimointitehtävän lokaali minimi, on olemassa Lagrangen kertoimet $\mathbf{u}^* \in \mathbb{R}^p$ ja $\mathbf{v}^* \in \mathbb{R}^q$ siten, että seuraavat ensimmäisen kertaluvun välttämättömät ehdot (*Karush-Kuhn-Tucker -ehdot*) toteutuvat:

- Piste $\mathbf{x}^* \in \mathbb{R}^n$ on käypä: $\mathbf{g}(\mathbf{x}^*) \leq \mathbf{0}$ ja $\mathbf{h}(\mathbf{x}^*) = \mathbf{0}$.
- Kertoimille $\mathbf{u}^*$ pätee $\mathbf{u}^* \geq \mathbf{0}$.
- Komplementaarisuusehto on voimassa: $u_i{}^* g_i(\mathbf{x}^*) = 0,\ i = 1, \ldots, p$.
- Pätee

$$\nabla L(\mathbf{x}^*, \mathbf{u}^*, \mathbf{v}^*) = \nabla f(\mathbf{x}^*) + \langle \nabla \mathbf{g}(\mathbf{x}^*), \mathbf{u} \rangle + \langle \nabla \mathbf{h}(\mathbf{x}^*), \mathbf{v} \rangle = \mathbf{0}. \tag{3.23}$$

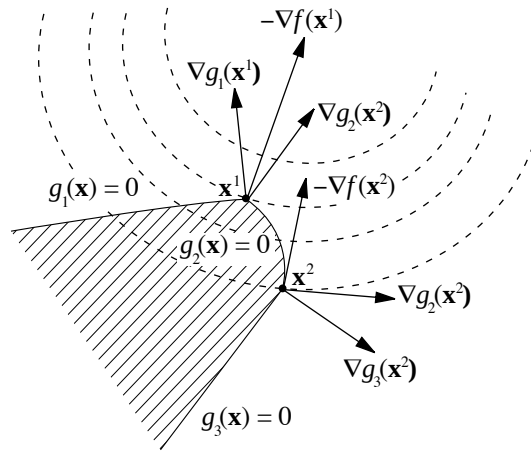
  Tässä derivaattaoperaattori ∇ tarkoittaa derivoimista vektorin $\mathbf{x}$ komponenttien suhteen.

Todistus: Lauseen johtamisen taustaa on esitelty edellisessä kappaleessa. Lisätietoja löytyy teoksista [FM68, PSU88]. □

Yhtälö (3.23) on komponenttimuodossa

$$\frac{\partial f(\mathbf{x}^*)}{\partial x_i} + \sum_{j=1}^{p} u_j{}^* \frac{\partial g_j(\mathbf{x}^*)}{\partial x_i} + \sum_{j=1}^{q} v_j{}^* \frac{\partial h_j(\mathbf{x}^*)}{\partial x_i} = 0, \quad i = 1, \ldots, n. \tag{3.24}$$

Karush-Kuhn-Tucker -ehtoja on havainnollistettu kuvassa 3.9. Kuvaan on piirretty katkoviivoilla kohdefunktion $f(\mathbf{x})$ tasa-arvokäyriä. Piste $\mathbf{x}^1$ on tehtävän lokaali minimikohta. Piste $\mathbf{x}^2$ ei ole minimikohta.



Kuva 3.9: *Epäyhtälörajoitteita sisältävän minimointitehtävän Karush-Kuhn-Tucker -ehtoja havainnollistava kuva.*

Lause 3.3.2 Olkoon voimassa lauseessa 3.3.1 esitetyt Karush-Kuhn-Tucker -ehdot, eli voidaan löytää Lagrangen kertoimet $\mathbf{u}^*$ ja $\mathbf{v}^*$. Piste $\mathbf{x}^*$ on funktion $f(\mathbf{x})$ minimipiste, jos kaikilla suunnilla $\mathbf{d} \in \mathbb{R}^n$, $\mathbf{d} \neq \mathbf{0}$, joille pätee

$$\begin{cases} \langle \mathbf{d}, \nabla g_i(\mathbf{x}^*) \rangle = 0, & \text{kaikilla } i = 1, \ldots, p \text{ joilla } {u_i}^* > 0, \\ \langle \mathbf{d}, \nabla g_i(\mathbf{x}^*) \rangle \geq 0, & \text{kaikilla } i = 1, \ldots, p \text{ joilla } {u_i}^* = 0 \text{ ja } g_i(\mathbf{x}^*) = 0, \\ \langle \mathbf{d}, \nabla h_j(\mathbf{x}^*) \rangle = 0, & \text{kaikilla } j = 1, \ldots, q, \end{cases} \tag{3.25}$$

on voimassa toisen kertaluvun ehto

$$\langle \mathbf{d}, (\nabla \nabla^T L(\mathbf{x}^*, \mathbf{u}^*, \mathbf{v}^*))\mathbf{d} \rangle > 0, \tag{3.26}$$

missä $\nabla \nabla^T L(\mathbf{x}^*, \mathbf{u}^*, \mathbf{v}^*)$ on Lagrangen funktion L Hessen matriisi vektorin $\mathbf{x}$ komponenttien suhteen.

Todistus: Lauseen johtamisen taustaa on esitelty kappaleessa 3.3.1. Lisätietoja löytyy teoksista [FM68, PSU88]. □

Lauseita 3.3.1 ja 3.3.2 on yleensä vaikea käyttää käytännön tehtävissä. Toisaalta ne muodostavat perustan monille optimointitehtävien ratkaisualgoritmeille.

Esimerkki 3.3.2 Olkoon ratkaistavana optimointitehtävä

$$\underset{\mathbf{x}}{\text{minimi}}\ f(\mathbf{x}) = x_1^2 + x_2^2 - x_3^2.$$

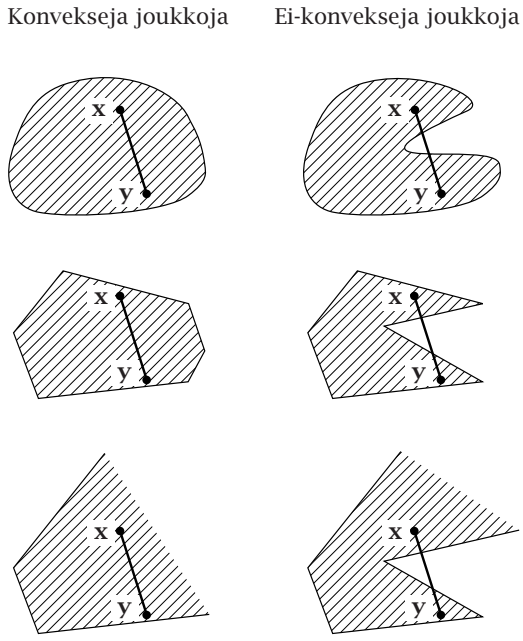
Lineaariset rajoitteet ovat

$$\begin{cases} g_1(\mathbf{x}) = 2x_1 + x_2 - 5 \le 0, \\ g_2(\mathbf{x}) = x_1 + x_3 - 2 \le 0, \\ g_3(\mathbf{x}) = -x_1 + 1 \le 0, \\ g_4(\mathbf{x}) = -x_2 + 2 \le 0, \\ g_5(\mathbf{x}) = -x_3 \le 0. \end{cases}$$

Karush-Kuhn-Tucker -ehdoiksi saadaan (vektorissa $\mathbf{u}$ on Lagrangen kertoimet u_i, $i = 1, \ldots, 5$):

$$\begin{aligned} \mathbf{u} &\ge \mathbf{0}, \\ \begin{pmatrix} 2x_1 \\ 2x_2 \\ -2x_3 \end{pmatrix} + \begin{pmatrix} 2 & 1 & -1 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & -1 \end{pmatrix} \mathbf{u} &= \mathbf{0}, \\ u_i g_i(\mathbf{x}) &= 0, \quad i = 1, \ldots, 5, \\ \mathbf{g}(\mathbf{x}) &\le \mathbf{0}. \end{aligned}$$

Minimipiste on $\mathbf{x}^* = (1, 2, 1)^T$. Piste $\mathbf{x}^*$ on käypä eli se toteuttaa rajoitteet: $\mathbf{g}(\mathbf{x}^*) = (-1, 0, 0, 0, -1)^T \le \mathbf{0}$. Ratkaisemalla Karush-Kuhn-Tucker -ehdot saadaan $\mathbf{u}^* = (0, 2, 4, 4, 0)^T \ge \mathbf{0}$. Koska $g_1(\mathbf{x}^*) = -1 \ne 0$, on oltava $u_1{}^* = 0$. Samoin $u_5{}^* = 0$. Loput Lagrangen kertoimet saadaan lineaarisesta yhtälöryhmästä sijoittamalla $\mathbf{x}^*$ sekä $u_1{}^*$ ja $u_5{}^*$.



Kuva 3.10: *Konvekseja ja ei-konvekseja tason joukkoja.*

3.4 Konveksisuus

Määritelmä 3.4.1 (Konveksi joukko) Joukko $C \subset \mathbb{R}^n$ on *konveksi* (convex), jos kaikilla $\mathbf{x}, \mathbf{y} \in C$ pätee

$$a\,\mathbf{x} + (1-a)\,\mathbf{y} \in C, \quad \text{kun } 0 < a < 1. \tag{3.27}$$

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 3.10 pisteitä $\mathbf{x}$ ja $\mathbf{y}$ yhdistävän janan $\{\mathbf{x} + a(\mathbf{y} - \mathbf{x}),\ 0 < a < 1\}$ avulla.

Määritelmä 3.4.2 (Konveksi ja konkaavi funktio) Funktion $f(\mathbf{x})$ sanotaan olevan *konveksi funktio*, jos kahdelle toisistaan eroavalle pisteelle $\mathbf{x}^1$ ja $\mathbf{x}^2 \in \mathbb{R}^n$ pätee

$$f(a\,\mathbf{x}^1 + (1-a)\,\mathbf{x}^2) \leq a f(\mathbf{x}^1) + (1-a) f(\mathbf{x}^2), \tag{3.28}$$

missä $0 < a < 1$. Jos yhtäsuuruus on toisin päin ($\geq$), on kyseessä *konkaavi funktio*. Funktio on *aidosti konveksi*, jos epäyhtälössä (3.28) pätee aito erisuuruus $<$ (aidosti konkaavissa tapauksessa $>$).

Seuraavia lauseita käytetään usein hyödyksi konveksisuustarkasteluissa.

Lause 3.4.1 Olkoon funktio $f : \mathbb{R}^n \mapsto \mathbb{R}$ jatkuvasti differentioituva konveksissa joukossa $\mathcal{F}$.

1. Funktio f on konveksi täsmälleen silloin, kun

$$f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle \leq f(\mathbf{y}) \text{ kaikilla } \mathbf{x}, \mathbf{y} \in \mathcal{F}. \tag{3.29}$$

2. Funktio f on aidosti konveksi täsmälleen silloin, kun

$$f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle < f(\mathbf{y}) \text{ kaikilla } \mathbf{x}, \mathbf{y} \in \mathcal{F},\ \mathbf{x} \neq \mathbf{y}. \tag{3.30}$$

Todistus: Valitaan pisteet $\mathbf{x}$, $\mathbf{y} \in \mathcal{F}$. Konveksisuuden nojalla pätee yhtälö $\mathbf{z} = \mathbf{y} + a(\mathbf{x} - \mathbf{y}) \in \mathcal{F}$, $0 < a < 1$. Tarkastellaan suuntaderivaattaa

$$\lim_{a \to 0} \frac{f(\mathbf{x} + a(\mathbf{y} - \mathbf{x})) - f(\mathbf{x})}{a} = \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle.$$

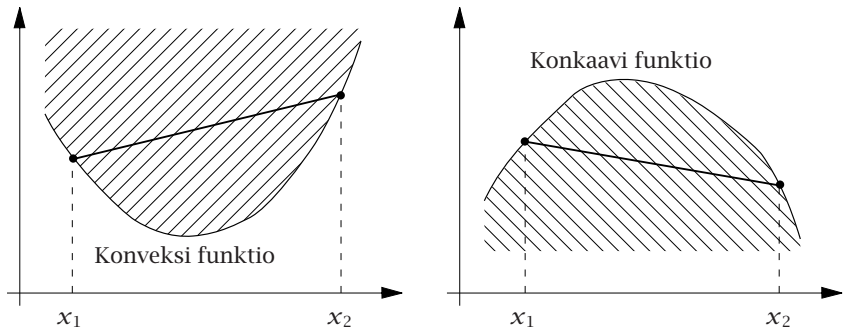
Koska f on konveksi, pätee epäyhtälö

$$f(\mathbf{x} + a(\mathbf{y} - \mathbf{x})) \leq a f(\mathbf{y}) + (1-a) f(\mathbf{x}), \quad 0 < a < 1.$$

Tästä saadaan yhtälö (3.29).

Pyritään seuraavaksi todistamaan konveksisuus. Yhtälöstä (3.29) seuraa

$$f(\mathbf{x}) \geq f(\mathbf{z}) + \langle \nabla f(\mathbf{x}), \mathbf{x} - \mathbf{z} \rangle \quad \text{ja} \quad f(\mathbf{y}) \geq f(\mathbf{z}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{z} \rangle.$$



Kuva 3.11: *Aidosti konveksi ja konkaavi funktio.*

Kerrotaan ensimmäinen epäyhtälö vakiolla a ja toinen vakiolla $1-a$ ja summataan:

$$\begin{aligned} af(\mathbf{x}) + (1-a)f(\mathbf{y}) &\geq f(\mathbf{z}) + \langle \nabla f(\mathbf{z}), a\,\mathbf{x} + (1-a)\,\mathbf{y} - \mathbf{z} \rangle \\ &= f(\mathbf{z}) = f(a\,\mathbf{x} + (1-a)\,\mathbf{y}). \end{aligned}$$

Siten f on konveksi.

Yhtälö (3.30) todistetaan samaan tapaan. □

Lause 3.4.2 Olkoon funktio $f(\mathbf{x})$ kahdesti differentioituva avoimessa konveksissa joukossa $\mathcal{F} \subset \mathbb{R}^n$. Funktio f on konveksi, jos Hessen matriisi $H(\mathbf{x})$ on positiivisesti semidefiniitti kaikilla $\mathbf{x} \in \mathcal{F}$, ja aidosti konveksi, jos $H(\mathbf{x})$ on positiivisesti definiitti kaikilla $\mathbf{x} \in \mathcal{F}$.

Todistus: Valitaan pisteet $\mathbf{x}$, $\mathbf{y} \in \mathcal{F}$. Olkoon $\mathbf{z} = (1-a)\,\mathbf{x} + a\,\mathbf{y}$, missä $0 < a < 1$. Koska $\mathcal{F}$ on konveksi, kuuluu piste $\mathbf{z}$ joukkoon $\mathcal{F}$. Käytetään Taylorin sarjakehitelmää:

$$f(\mathbf{y}) = f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{1}{2} \langle \mathbf{y} - \mathbf{x}, H(\mathbf{z})(\mathbf{y} - \mathbf{x}) \rangle.$$

Jos Hessen matriisi $H(\mathbf{z})$ on positiivisesti semidefiniitti, pätee

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle.$$

Jos $H(\mathbf{z})$ on positiivisesti definiitti, saadaan aito epäyhtälö kun $\mathbf{x} \neq \mathbf{y}$. Tulos seuraa lauseesta 3.4.1. □

Kuvassa 3.11 on esimerkit aidosti konvekseista ja konkaaveista skalaarifunktioista. Konveksin funktion yläpuolelle jäävien pisteiden joukko on konveksi. Aidosti konveksille funktiolle f pätee $f(ax_1 + (1-a)x_2) < af(x_1) + (1-a)f(x_2)$, $a \in (0,1)$. Aidosti konkaaville funktiolle f pätee puolestaan $f(ax_1 + (1-a)x_2) > af(x_1) + (1-a)f(x_2)$, $a \in (0,1)$. Jos funktio $f(\mathbf{x})$ on konveksi, funktio $-f(\mathbf{x})$ on tietenkin konkaavi.

Seuraavat lauseet yhdistävät konveksit funktiot ja joukot.

Lause 3.4.3 Konveksien joukkojen $\mathcal{F}_i$ leikkaus $\mathcal{F} = \cap\mathcal{F}_i$ on konveksi joukko.

Todistus: Valitaan kaksi pistettä $\mathbf{x}^1$ ja $\mathbf{x}^2 \in \mathcal{F} = \cap\mathcal{F}_i$. Määritellään piste $\mathbf{x}^a = (1-a)\,\mathbf{x}^1 + a\,\mathbf{x}^2$, $0 < a < 1$. Koska joukot $\mathcal{F}_i$ ovat konvekseja, pätee $\mathbf{x}^a \in \mathcal{F}_i$ kaikilla i eli $\mathbf{x}^a \in \mathcal{F}$. Siis joukko $\mathcal{F} = \cap\mathcal{F}_i$ on konveksi. □

Lause 3.4.4 Olkoon määriteltynä joukko konvekseja funktioita $\{g_i(\mathbf{x})\}$, missä $i = 1, \ldots, p$. Joukko $\mathcal{F} = \{\mathbf{x} \in \mathbb{R}^n \mid g_i(\mathbf{x}) \le b_i,\ i = 1, \ldots, p\}$ on konveksi.

Todistus: Olkoon $\mathcal{F}_i = \{\mathbf{x} \in \mathbb{R}^n \mid g_i(\mathbf{x}) \le b_i,\}$. Valitaan $\mathbf{x}^1$ ja $\mathbf{x}^2 \in \mathcal{F}_i$ ja asetetaan $\mathbf{x}^a = (1-a)\,\mathbf{x}^1 + a\,\mathbf{x}^2$, $0 < a < 1$. Funktioiden g_i konveksisuuden perusteella pätee

$$g_i(\mathbf{x}^a) = g_i((1-a)\,\mathbf{x}^1 + a\,\mathbf{x}^2) \le (1-a)\,g_i(\mathbf{x}^1) + a g_i(\mathbf{x}^2).$$

Koska $\mathbf{x}^1$ ja $\mathbf{x}^2 \in \mathcal{F}_i$, pätee $g_i(\mathbf{x}^1) \le b_i$ ja $g_i(\mathbf{x}^2) \le b_i$. Siis

$$(1-a)\,g_i(\mathbf{x}^1) + a\,g_i(\mathbf{x}^2) \le (1-a)\,b_i + a b_i = b_i.$$

Siis $g_i(\mathbf{x}^a) \le b_i$, joten $\mathbf{x}^a \in \mathcal{F}_i$. Täten joukko $\mathcal{F}_i$ on konveksi.

Lauseen 3.4.3 perusteella joukko $\mathcal{F} = \cap\mathcal{F}_i$ on konveksi. □

Määritelmä 3.4.3 (Konveksi optimointitehtävä) Olkoon ratkaistavana minimointitehtävä

$$\min_{\mathbf{x}} f(\mathbf{x}),\ \text{kun } \mathbf{g}(\mathbf{x}) \le \mathbf{0} \text{ ja } \mathbf{h}(\mathbf{x}) = \mathbf{0}. \tag{3.31}$$

Jos kohdefunktio $f(\mathbf{x})$ on konveksi, epäyhtälörajoitteet $g_i(\mathbf{x}) \le 0$ ovat konvekseja ja yhtälörajoitteet $h_i(\mathbf{x}) = 0$ ovat lineaarisia, on kyseessä *konveksi optimointitehtävä*.

Esimerkki konveksista optimointitehtävästä (convex programming) on lineaarinen optimointitehtävä (luku 4).

Konveksisuus on vahva ominaisuus optimointitehtävän ratkaisemisessa. Konveksin optimointitehtävän globaalille minimille pätee seuraava lause.

Lause 3.4.5 Jos piste $\mathbf{x}^*$ on konveksin optimointitehtävän käypään alueeseen $\mathcal{F}$ sisältyvä lokaali minimi, on se myös globaali käypä minimi. Lisäksi konveksin optimointitehtävän minimien joukko $\mathcal{F}^*$ on konveksi.

Todistus: Oletetaan, että $\mathbf{x}^* \in \mathcal{F}$ on lokaali mutta ei globaali minimipiste. Siis on olemassa piste $\mathbf{x}^1$ siten että $f(\mathbf{x}^1) < f(\mathbf{x}^*)$. Määritellään piste $\mathbf{x}^a = (1-a)\,\mathbf{x}^* + a\,\mathbf{x}^1$, $0 < a < 1$. Koska joukko $\mathcal{F}$ on konveksi, on $\mathbf{x}^a \in \mathcal{F}$. Koska funktio f on konveksi, on voimassa

$$f(\mathbf{x}^a) \le (1-a) f(\mathbf{x}^*) + a f(\mathbf{x}^1) = f(\mathbf{x}^*) + a\,\Big(f(\mathbf{x}^1) - f(\mathbf{x}^*)\Big) < f(\mathbf{x}^*).$$

Jos parametri a on kyllin pieni, saadaan $f(\mathbf{x}^*) < f(\mathbf{x}^*)$ eli ristiriita. Siis piste $\mathbf{x}^*$ on globaali minimikohta.

Valitaan kaksi minimipistettä $\mathbf{x}^1$ ja $\mathbf{x}^2 \in \mathcal{F}^*$. Olkoon $\mathbf{x}^a = (1-a)\,\mathbf{x}^1 + a\,\mathbf{x}^2$, missä $0 < a < 1$. Koska $\mathbf{x}^1$ ja $\mathbf{x}^2$ ovat globaaleja minimipisteitä, pätee $f(\mathbf{x}^a) \geq f(\mathbf{x}^1) = f(\mathbf{x}^2)$. Funktion f konveksisuudesta johtuen saadaan

$$f(\mathbf{x}^a) \leq (1-a)f(\mathbf{x}^1) + af(\mathbf{x}^2) = f(\mathbf{x}^1) = f(\mathbf{x}^2).$$

Siis $f(\mathbf{x}^a) \leq f(\mathbf{x}^1) = f(\mathbf{x}^2)$. Siten $\mathbf{x}^a \in \mathcal{F}^*$ ja joukko $\mathcal{F}^*$ on konveksi. □

Seuraaviin kahteen lauseeseen on koottu joitakin konveksisuuden löytämiseen soveltuvia tuloksia.

Lause 3.4.6 Konvekseille joukoille pätee:

1. Konveksien joukkojen leikkaus $\cap \mathcal{F}_i$ on konveksi.
2. Joukko $\{\mathbf{x} \in \mathbb{R}^n \mid g(\mathbf{x}) \leq b\}$ on konveksi, jos funktio $g : \mathbb{R}^n \mapsto \mathbb{R}$ on konveksi.
3. Joukko $\{\mathbf{x} \in \mathbb{R}^n \mid g(\mathbf{x}) \geq b\}$ on konveksi, jos funktio $g : \mathbb{R}^n \mapsto \mathbb{R}$ on konkaavi.
4. Joukko $\{\mathbf{x} \in \mathbb{R}^n \mid g(\mathbf{x}) = b\}$ on konveksi, jos funktio $g : \mathbb{R}^n \mapsto \mathbb{R}$ on lineaarinen.

Todistus: Lauseet 3.4.3 ja 3.4.4 sekä konveksisuuden määritelmä. □

Lause 3.4.7 Konveksin funktion voi tunnistaa seuraavista ominaisuuksista:

1. Jos funktio $-f(\mathbf{x})$ on konkaavi, on funktio $f(\mathbf{x})$ konveksi, ja kääntäen.
2. Lineaarinen funktio $f(\mathbf{x}) = A\mathbf{x},\ \mathbf{x} \in \mathbb{R}^n$ on sekä konveksi että konkaavi.
3. Jos funktiot $f_i(\mathbf{x})$ ovat konvekseja, on funktio $f(\mathbf{x}) = \max_i\{f_i(\mathbf{x})\}$ konveksi.
4. Vektorinormi $\|\cdot\|$ on konveksi.
5. Jos funktiot $f_i(\mathbf{x})$ ovat konvekseja, on funktio $\sum_i a_i f_i(\mathbf{x}),\ a_i \geq 0$ konveksi.
6. Jos funktio $f : \mathbb{R}^n \mapsto \mathbb{R}$ on konveksi ja $A : \mathbb{R}^m \mapsto \mathbb{R}^n$ on lineaarikuvaus, on yhdistetty funktio $f(A\mathbf{x})$ konveksi.
7. Jos funktio $f_1 : \mathbb{R} \mapsto \mathbb{R}$ on konveksi ja kasvava sekä funktio $f_2 : \mathbb{R}^n \mapsto \mathbb{R}$ on konveksi, on yhdistetty kuvaus $f_1(f_2(\mathbf{x}))$ konveksi.
8. Jos funktio $f_1 : \mathbb{R} \mapsto \mathbb{R}$ on konveksi ja vähenevä sekä funktio $f_2 : \mathbb{R}^n \mapsto \mathbb{R}$ on konkaavi, on yhdistetty kuvaus $f_1(f_2(\mathbf{x}))$ konveksi.
9. Kahdesti differentioituva funktio $f : \mathbb{R} \mapsto \mathbb{R}$ on konveksi täsmälleen silloin, kun pätee

$$\frac{d^2 f(x)}{dx^2} \geq 0 \text{ kaikilla } x \in \mathbb{R}.$$

10. Kahdesti differentioituva funktio $f : \mathbb{R}^n \mapsto \mathbb{R}$ on konveksi täsmälleen silloin, kun Hessen matriisi $\nabla\nabla f(\mathbf{x})$ on positiivisesti semidefiniitti kaikilla $\mathbf{x}$.

Todistus: Tulokset saadaan määritelmää 3.4.2 käyttämällä. Katso myös lausetta 3.4.2.

Esimerkiksi funktion $\max_i\{f_i(\mathbf{x})\}$ konveksisuus voidaan todistaa seuraavasti. Määritellään funktion f *epigraafi*:

$$\text{epi}(f) = \{\ (\mathbf{x}, b) \in \mathbb{R}^{n+1}\ \mid\ \mathbf{x} \in \mathbb{R}^n,\, b \in \mathbb{R},\, b \geq f(\mathbf{x})\ \}. \tag{3.32}$$

Konveksin funktion epigraafi on konveksi eli joukot $\mathcal{F}_i = \text{epi}(f_i)$ ovat konvekseja. Maksimifunktion $f = \max_i\{f_i(\mathbf{x})\}$ epigraafi on $\mathcal{F} = \text{epi}(f) = \cap\mathcal{F}_i$. Koska konveksien joukkojen leikkaus on konveksi, on $\mathcal{F}$ konveksi eli funktio f on konveksi.

Vektorinormin konveksisuus johdetaan seuraavasti (tässä $0 < a < 1$):

$$\|a\,\mathbf{x} + (1-a)\,\mathbf{y}\| \leq \|a\,\mathbf{x}\| + \|(1-a)\,\mathbf{y}\| = a\|\mathbf{x}\| + (1-a)\|\mathbf{y}\|.$$

Tässä käytettiin normin ominaisuuksia eli epäyhtälöä $\|\mathbf{x}+\mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$ ja yhtälöä $\|a\,\mathbf{x}\| = |a|\|\mathbf{x}\|$ (sivu 218). □

Esimerkki 3.4.1 Olkoon ratkaistavana optimointitehtävä

$$\underset{\mathbf{x}}{\text{minimi}}\ f(\mathbf{x}) = 4x_1^4 - 4x_1^2x_2 + x_2^2 + x_3^3,$$

kun rajoitteina on

$$\begin{cases} g_1(\mathbf{x}) = x_3 \geq 1, & g_2(\mathbf{x}) = x_2 - x_3 \geq 2, \\ g_3(\mathbf{x}) = x_1 \geq 2, & g_4(\mathbf{x}) = x_2 \leq 7, \\ g_5(\mathbf{x}) = |x_1 + x_2 + 2x_3| \leq 10, \\ g_6(\mathbf{x}) = 2x_1 - x_1^2 + 2x_1x_3 - 3x_3^2 \geq 2. \end{cases}$$

Seuraavassa käytetään lauseita 3.4.6 ja 3.4.7 tehtävän konveksisuuden osoittamiseen.

Lausutaan kohdefunktio muodossa

$$f(\mathbf{x}) = (2x_1^2 - x_2)^2 + x_3^3.$$

Funktio $2x_1^2 - x_2$ on konveksi, koska $2x_1^2$ on konveksi ja $-x_2$ on konveksi ja koska funktio $\sum_i a_i f_i(\mathbf{x})$, $a_i \geq 0$, on konveksi, kun funktiot f_i ovat konvekseja. Funktio t^2 on konveksi ja kasvava kun $t > 0$. Nyt $x_1 \geq 2$ ja $x_2 \leq 7$, joten $2x_1^2 - x_2 > 0$ ja siten $(2x_1^2 - x_2)^2$ on konveksi.

Lasketaan

$$\frac{d^2(x_3^3)}{dx_3^2} = 6x_3 > 0,$$

kun $x_3 > 0$. Rajoitteista nähdään, että $x_3 \geq 1 > 0$, joten x_3^3 on konveksi.

Koska kohdefunktio on kahden konveksin funktion positiivinen summa, on kohdefunktio konveksi.

Lineaariset rajoitefunktiot $g_i, i = 1,\ldots,4$ ovat luonnollisesti konvekseja, jolloin riittää tarkastella rajoitteita g_5 ja g_6. Koska pätee $|f(\mathbf{x})| = \max\{f(\mathbf{x}), -f(\mathbf{x})\}$, ja koska lineaarinen funktio on sekä konveksi että konkaavi, on rajoitefunktio $g_5(\mathbf{x})$ konveksi.

Muutetaan rajoitefunktio g_6 muotoon $g_7(\mathbf{x}) = (x_1 - 1)^2 - 2x_1x_3 + 3x_3^2 \leq 3$ kertomalla luvulla -1 ja sieventämällä. Funktio $(x_1 - 1)^2$ on konveksi, koska $x_1 \geq 2$ eli $x_1 - 1 > 0$ ja koska t^2 on konveksi ja kasvava, kun $t > 0$.

Funktion $-2x_1x_3 + 3x_3^2$ Hessen matriisiksi saadaan

$$H = \begin{pmatrix} 0 & -2 \\ -2 & 6 \end{pmatrix}.$$

Matriisin H ominaisarvot $\text{eig}(H) = \{\lambda_i\}$ saadaan yhtälöstä

$$(0 - \lambda)(6 - \lambda) - 4 = 0$$

eli $\text{eig}(H) = \{3 \pm \sqrt{5}\} \approx \{5.2, 0.8\} > 0$. Siis Hessen matriisi on positiivisesti definiitti. Koska funktio g_7 on kahden konveksin funktion positiivinen summa, on rajoitefunktio konveksi. Täten myös rajoitefunktio g_6 määrittelee konveksin joukon.

Koska kaikki rajoitteet olivat konvekseja, on myös käypä alue konveksi, sillä konveksien joukkojen leikkaus on konveksi. Täten kyseessä on konveksi optimointitehtävä ja tehtävän käypä minimipiste on globaali minimi.

3.5 Kvadraattiset funktiot

Funktio

$$\langle \mathbf{x}, Q\mathbf{x} \rangle = \sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij} x_i x_j,$$

missä matriisi Q on symmetrinen $n \times n$ -matriisi ja vektori $\mathbf{x}$ kuuluu joukkoon $\mathbb{R}^n$, määrittelee kvadraattisen funktion $\mathbb{R}^n \mapsto \mathbb{R}$. Yleinen kvadraattinen funktio $\mathbb{R}^n \mapsto \mathbb{R}$ voidaan kirjoittaa muotoon (niin sanottu *neliömuoto*)

$$f(\mathbf{x}) = \frac{1}{2}\langle \mathbf{x}, Q\mathbf{x} \rangle + \langle \mathbf{c}, \mathbf{x} \rangle + d. \tag{3.33}$$

Esimerkki 3.5.1 Haetaan funktion

$$\begin{aligned} f(\mathbf{x}) &= (x_1 - x_2)^2 + (x_1 + 2x_2 + 1)^2 - 8x_1x_2 \\ &= 2x_1^2 - 6x_1x_2 + 5x_2^2 + 2x_1 + 4x_2 + 1 \end{aligned} \tag{3.34}$$

esitys muodossa (3.33). Nyt

$$\frac{1}{2}\begin{pmatrix} x_1 & x_2 \end{pmatrix}\begin{pmatrix} \alpha & \beta \\ \beta & \gamma \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \frac{1}{2}\alpha x_1^2 + \beta x_1x_2 + \frac{1}{2}\gamma x_2^2,$$

joten vertaamalla yhtälöön (3.34) saadaan

$$Q = \begin{pmatrix} 4 & -6 \\ -6 & 10 \end{pmatrix}, \quad \mathbf{c} = \begin{pmatrix} 2 & 4 \end{pmatrix}, \quad d = 1.$$

Kvadraattiset funktiot ovat tärkeitä optimoinnissa mm. siksi, että differentioituva epälineaarinen funktio voidaan esittää Taylorin lauseen mukaisesti Taylorin sarjana, jonka ensimmäisistä termeistä muodostuu kvadraattinen funktio. Siten yleisempiä funktioita voidaan arvioida paikallisesti kvadraattisella mallilla.

Seuraava lause kytkee yhteen matriisin Q definiittisyyden ja funktion $\langle \mathbf{x}, Q\,\mathbf{x}\rangle$ konveksisuuden.

Lause 3.5.1 Funktio $f(\mathbf{x}) = \langle \mathbf{x}, Q\,\mathbf{x}\rangle$ on aidosti konveksi ainoastaan silloin, kun matriisi Q on positiivisesti definiitti. Vastaavasti funktio $f(\mathbf{x}) = \langle \mathbf{x}, Q\,\mathbf{x}\rangle$ on konveksi ainoastaan silloin, kun matriisi Q on positiivisesti semidefiniitti.

Todistus: Olkoot $\mathbf{x}^1$ ja $\mathbf{x}^2$ pisteitä avaruudessa $\mathbb{R}^n$. Koska matriisi Q on symmetrinen, funktion $\langle \mathbf{x}, Q\,\mathbf{x}\rangle$ gradienttivektori on $2Q\mathbf{x}$ ja Hessen matriisi on $2Q$. Taylorin sarjakehitelmän perusteella pätee

$$f(\mathbf{x}^2) = f(\mathbf{x}^1) + \langle 2Q\mathbf{x}^1, \mathbf{x}^2 - \mathbf{x}^1\rangle + \frac{1}{2}\langle \mathbf{x}^2 - \mathbf{x}^1, 2Q(\mathbf{x}^2 - \mathbf{x}^1)\rangle.$$

Jos matriisi Q on positiivisesti semidefiniitti, saadaan epäyhtälö

$$f(\mathbf{x}^2) \geq f(\mathbf{x}^1) + \langle 2Q\mathbf{x}^1, \mathbf{x}^2 - \mathbf{x}^1\rangle.$$

Kyseessä on aito epäyhtälö, mikäli Q on positiivisesti definiitti. Lauseen 3.4.1 perusteella saadaan haluttu tulos. □

Esimerkki 3.5.2 Tutkitaan esimerkissä 3.5.1 käsitellyn kvadraattisen funktion $f : \mathbb{R}^2 \mapsto \mathbb{R}$ konveksisuutta. Funktio voidaan esittää muodossa

$$f(\mathbf{x}) = \frac{1}{2}\langle \mathbf{x}, Q\mathbf{x}\rangle + \langle \mathbf{c}, \mathbf{x}\rangle + d.$$

Lineaarinen termi $\langle \mathbf{c}, \mathbf{x}\rangle + d$ on luonnollisesti konveksi. Matriisin Q ominaisarvoiksi saadaan

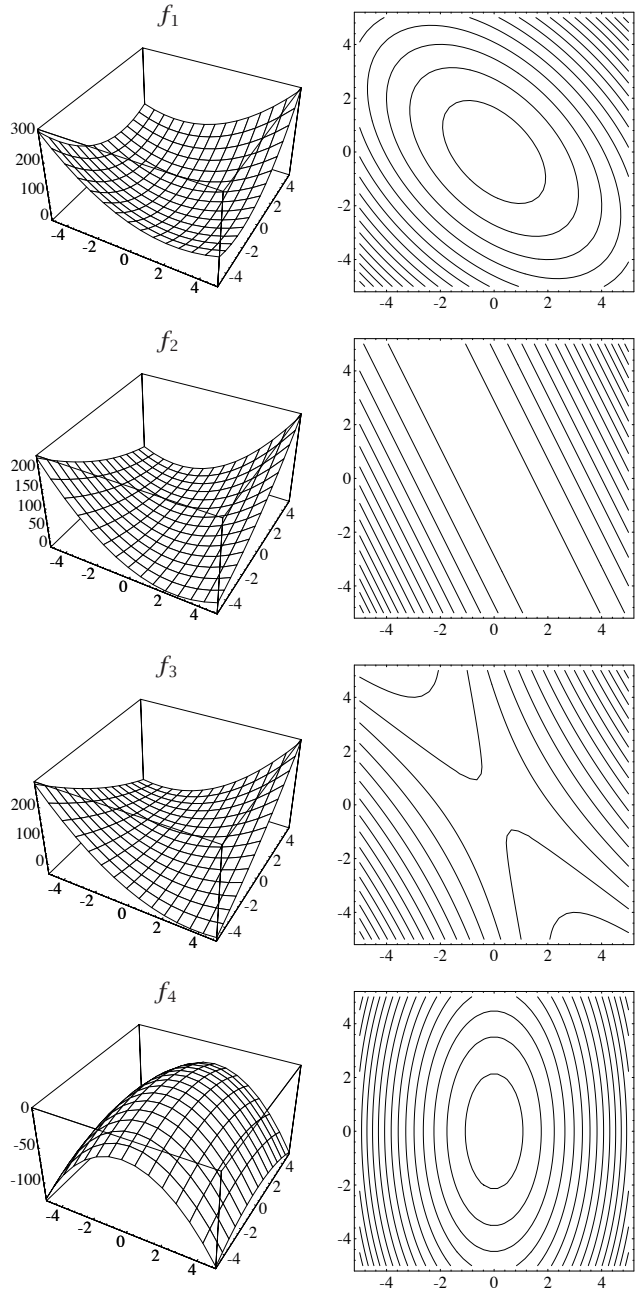
$$\operatorname{eig}\begin{pmatrix} 4 & -6 \\ -6 & 10 \end{pmatrix} = \{\ \frac{14 - 6\sqrt{5}}{2},\ \frac{14 + 6\sqrt{5}}{2}\ \} \approx \{\ 0.292,\ 13.7\ \} > 0.$$

Täten termi $\frac{1}{2}\langle \mathbf{x}, Q\mathbf{x}\rangle$ on konveksi. Koska kahden konveksin funktion positiivinen summa on konveksi, on esimerkin 3.5.1 funktio f konveksi.

Esimerkki 3.5.3 Havainnollistetaan eri tyyppisten kvadraattisten funktioiden käyttäytymistä. Esitetään funktiot muodossa $f_i(\mathbf{x}) = \langle \mathbf{x}, Q_i\mathbf{x}\rangle$, missä Q_i on jokin seuraavista matriiseista:

$$Q_1 = \begin{pmatrix} 4 & 2 \\ 2 & 2 \end{pmatrix},\quad Q_2 = \begin{pmatrix} 4 & 2 \\ 2 & 1 \end{pmatrix},$$
$$Q_3 = \begin{pmatrix} 4 & 3 \\ 3 & 1 \end{pmatrix},\quad Q_4 = \begin{pmatrix} -4 & 0 \\ 0 & -1 \end{pmatrix}.$$

Kuvaan 3.12 on piirretty näiden funktioiden kuvaajat pintakaavioina ja tasa-arvokäyrien avulla. Funktiolla f_1 on yksikäsitteinen minimi ja funktiolla f_4 yksikäsitteinen maksimi. Matriisien Q_i definiittisyyttä on käsitelty esimerkissä 1.4.4 sivulla 26.



Kuva 3.12: *Pintakaavioina ja tasa-arvokäyrinä esitettyjä kvadraattisia funktioita* $f_i(\mathbf{x}) = \langle \mathbf{x}, Q_i\mathbf{x} \rangle$, $i = 1, \ldots, 4$, *missä matriisi* Q_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.*

3.6 Lisätietoja

Optimoinnin perusteita on käsitelty teoksissa *The Mathematics of Nonlinear Programming* [PSU88], *Nonlinear Programming: Theory and Algorithms* [BSS93] sekä *Optimization: Theory and Applications* [Rao79]. Optimointitehtävien numeerista ratkaisemista on käsitelty teoksissa *Practical Optimization* [GMW81], *Practical Methods of Optimization* [Fle87] ja *Introduction to Non-linear Optimization* [Sca85].

Katsausartikkeli *Lagrange Multipliers and Optimality* [Roc93] on perusteellinen johdatus optimoinnin teoriaan. Käytännönläheisempiä esityksiä löytyy *Acta Numerica* -julkaisusta [Wri92a, Noc92]. Optimointitehtävien teoriaa ja erikoismenetelmiä käsitellään esimerkiksi teoksissa *Optimization by Vector Space Methods* [Lue69], *Optimal Control: An Introduction to the Theory with Applications* [Hoc91], *Stochastic Models in Operations Research, Volume II: Stochastic Optimization* [HS84] ja *Multiple Criteria Optimization* [Ste86]. Monitavoiteoptimointiin johdattaa teos *Nonlinear Multiobjective Optimization* [Mie99].

4 Lineaarinen ja kvadraattinen optimointi

Tässä luvussa esitellään suppeasti lineaarista ja kvadraattista optimointia. Luvussa kerrotaan lähinnä mallien muodostamisesta ja eräiden ratkaisuohjelmistojen käyttämisestä ja esitellään tyypillisiä LP-tehtäviä. Lineaaristen optimointimallien kuvaamiseen käytetään MPS-formaattia sekä GAMS-ohjelmistoa.

Ratkaisualgoritmeista käsitellään mm. lineaaristen optimointitehtävien *simplex-menetelmää* sekä sisäpistealgoritmien taustaa. Tarkempia tietoja löytyy luvun kirjallisuusluettelossa mainituista teoksista.

4.1 Lineaariset optimointitehtävät

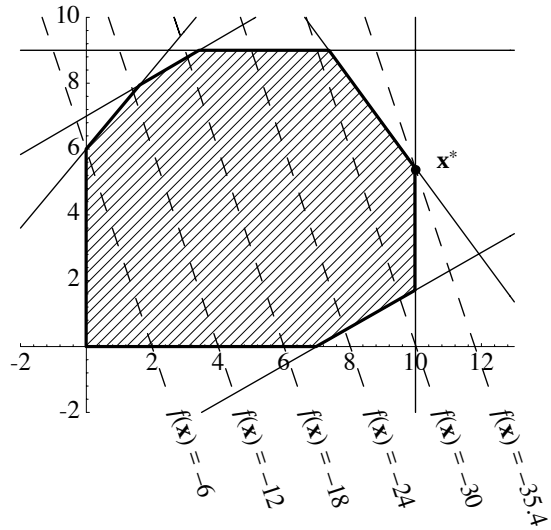
Lineaarista optimointia käsittelevässä kirjallisuudessa käytetään toisinaan esimerkiksi epälineaarisen optimoinnin terminologiasta eroavia termejä (ohjelmointi eikä optimointi jne.). Kuitenkin uudemmissa oppikirjoissa alueet ovat lähentyneet toisiaan.

Tässä oppaassa pyritään käyttämääään yhtenäistä terminologiaa, joten *lineaarisen ohjelmoinnin* (linear programming eli LP) sijaan puhutaan *lineaarisesta optimoinnista*.

Lineaariset optimointitehtävät eli LP-tehtävät määritellään käytännössä useimmiten käyttäen epäyhtälömuotoisia rajoitteita. Tällöin on kyseessä minimointitehtävä

$$\min_{\mathbf{x}} \langle \mathbf{c}, \mathbf{x} \rangle, \text{ kun } A\mathbf{x} \leq \mathbf{b} \text{ ja } \mathbf{x} \geq \mathbf{0}, \tag{4.1}$$

missä vektorit $\mathbf{x}$ ja $\mathbf{c}$ kuuluvat joukkoon $\mathbb{R}^n$ ja vektori $\mathbf{b}$ kuuluu joukkoon $\mathbb{R}^m$. Matriisin A vaakarivien lukumäärää merkitään m:llä ja sarakkeiden määrää n:llä.



Kuva 4.1: *Kahden muuttujan lineaarinen optimointitehtävä.*

Esimerkki 4.1.1 Kuvassa 4.1 on esitetty seuraavan LP-tehtävän käypä alue:

$$\underset{x_1,x_2}{\text{minimi}}\ -3x_1 - x_2,\ \text{kun}\ \begin{cases} -6x_1 + 5x_2 \le 30, \\ -7x_1 + 12x_2 \le 84, \\ 19x_1 + 14x_2 \le 266, \\ 4x_1 - 7x_2 \le 28, \\ 0 \le x_1 \le 10, \\ 0 \le x_2 \le 9. \end{cases} \tag{4.2}$$

Kuvassa 4.1 on kahden muuttujan lineaariselle kohdefunktiolle $f(\mathbf{x}) = -3x_1 - x_2$ piirretty katkoviivoilla suorat $f(\mathbf{x}) =$ vakio. Epäyhtälöehtojen rajaama käypä alue $A\mathbf{x} \le \mathbf{b}$ on merkitty paksulla yhtenäisellä viivalla. Tehtävä on matriisimuodossa

$$\underset{\mathbf{x}}{\text{minimi}}\ \langle \mathbf{c}, \mathbf{x} \rangle,\ \text{kun}\ A\mathbf{x} \le \mathbf{b}\ \text{ja}\ \mathbf{x} \ge \mathbf{0}, \tag{4.3}$$

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

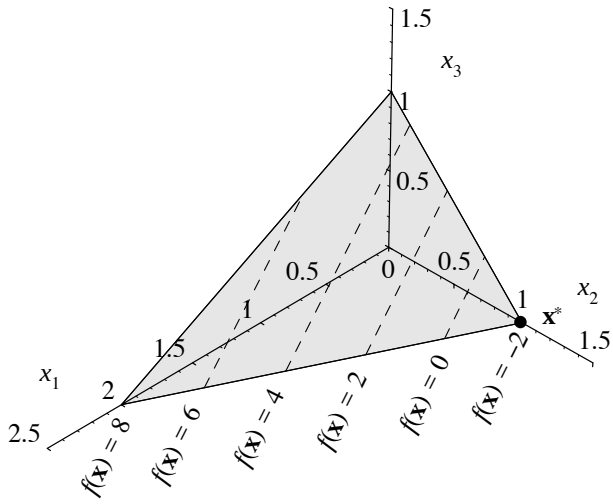
$$\underset{\mathbf{x}}{\text{minimi}}\ \langle \mathbf{c}, \mathbf{x} \rangle,\ \text{kun}\ A\mathbf{x} = \mathbf{b}\ \text{ja}\ \mathbf{x} \ge \mathbf{0}, \tag{4.4}$$

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, $m \le n$.

Esimerkki 4.1.2 Kuvassa 4.2 on esitetty standardimuotoinen lineaarinen optimointitehtävä

$$\underset{\mathbf{x}}{\text{minimi}}\ f(\mathbf{x}) = 4x_1 - 2x_2 + 5x_3, \text{ kun } x_1 + 2x_2 + 2x_3 = 2 \text{ ja } \mathbf{x} \geq \mathbf{0}.$$

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 4.2: *Kolmen muuttujan standardimuotoinen optimointitehtävä.*

LP-tehtävässä voi olla erikseen määritelty muuttujien ylä- ja alarajat (*laatikkorajoitteet*):

$$\mathbf{x}^l \leq \mathbf{x} \leq \mathbf{x}^u, \quad \mathbf{x}^l, \mathbf{x}^u \in \mathbb{R}^n. \tag{4.5}$$

Eräät algoritmit käsittelevät erikseen tämän tyyppisiä rajoitteita, toiset taas samalla tavalla kuin muitakin rajoitteita. Epäyhtälöehto

$$a_{j1}x_1 + a_{j2}x_2 + \cdots + a_{jn}x_n \leq b_j,$$

voidaan muuntaa yhtälössä (4.4) esitettyyn standardimuotoon positiivisen *slack*-muuttujan x_{n+i} lisäyksellä:

$$a_{j1}x_1 + a_{j2}x_2 + \cdots + a_{jn}x_n + x_{n+i} = b_j.$$

Epäyhtälömuotoisesta tehtävästä saadaan seuraava standardimuotoinen tehtävä:

$$\begin{pmatrix} & 1 & 0 & \cdots & 0 \\ & 0 & 1 & & \vdots \\ A & \vdots & & \ddots & 0 \\ & 0 & 0 & \cdots & 1 \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ x_{n+1} \\ \vdots \\ x_{n+k} \end{pmatrix} = A'\mathbf{x}' = \mathbf{b}. \tag{4.6}$$

Matriisin A' dimensio on $m \times (n + k)$, missä k on epäyhtälörajoitteiden lukumäärä. Jos slack-muuttujat x_{n+i} sijoitetaan vektoriin $\mathbf{z}$, saadaan tehtävä standardimuotoon:

$$\underset{\mathbf{x}}{\text{minimi}}\ \langle \mathbf{c}, \mathbf{x} \rangle, \text{ kun } A\mathbf{x} + \mathbf{z} = \mathbf{b},\ \mathbf{z} \geq \mathbf{0} \text{ ja } \mathbf{x} \geq \mathbf{0}. \tag{4.7}$$

Yhtälömuotoinen tehtävä voitaisiin vastaavasti muuntaa epäyhtälömuotoon, mutta tähän suuntaan tehtävää muunnosta tarvitaan harvemmin.

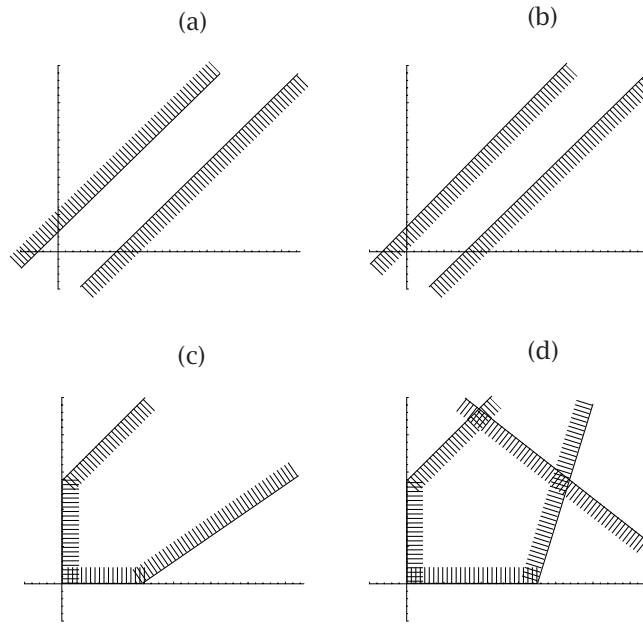
Kuvassa 4.3 on esitetty eräitä mahdollisia lineaaristen rajoitteiden kombinaatioita. Tapauksessa (a) ovat rajoitteet ristiriitaiset eli käypä alue on tyhjä, ja tapauksessa (b) on toinen rajoitteista tarpeeton. Tapauksessa (c) on käypä alue rajoittamaton ja tapauksessa (d) rajoitettu.

Vapaat muuttujat eli myös negatiivisia arvoja saavat muuttujat voi esittää kahden positiivisen muuttujan erotuksena $x_k = x_{k'} - x_{k''}$. Tämä voi kuitenkin tuottaa numeerisia stabiilisuusongelmia joillekin ratkaisualgoritmeille.

Standardimuotoiselle LP-tehtävälle voidaan muodostaa *duaalitehtävä*

$$\underset{\mathbf{y}}{\text{maksimi}}\ \langle \mathbf{b}, \mathbf{y} \rangle, \text{ kun } A^T\mathbf{y} \leq \mathbf{c}. \tag{4.8}$$

LP-tehtävän duaali on siten myös lineaarinen optimointitehtävä. Alkuperäistä LP-tehtävää (4.4) kutsutaan *primaaliksi*. Primaalilla ja duaalilla on seuraava yhteys keskenään:



Kuva 4.3: *Esimerkkejä lineaarisista rajoitteista. Rajoitteiden käyvän alueen puoli on merkitty katkoviivoituksella.*

Lause 4.1.1 Tarkastellaan standardimuotoista lineaarista optimointitehtävää (4.4) eli primaalitehtävää sekä duaalitehtävää (4.8). Yksi seuraavista vaihtoehdoista toteutuu:

1. Primaalin käypä alue ei ole tyhjä, ja ratkaisupisteessä $\mathbf{x}^*$ on kohdefunktion arvo rajoittamaton. Tällöin duaalitehtävän käypä alue on tyhjä.
2. Duaalin käypä alue ei ole tyhjä, ja ratkaisupisteessä $\mathbf{y}^*$ on kohdefunktion arvo rajoittamaton. Tällöin primaalitehtävän käypä alue on tyhjä.
3. Kummallakin tehtävällä on käyvät ratkaisut $\mathbf{x}^*$ ja $\mathbf{y}^*$, jolloin pätee
$$\langle \mathbf{c}, \mathbf{x}^* \rangle = \langle \mathbf{b}, \mathbf{y}^* \rangle \quad \text{ja} \quad \langle \mathbf{c} - A^T\mathbf{y}^*, \mathbf{x}^* \rangle = 0. \tag{4.9}$$
4. Kummallakaan tehtävällä ei ole käypää ratkaisua.

Todistus: Lauseen todistus löytyy lineaarista optimointia käsittelevistä oppikirjoista (katso kirjallisuusviitteitä sivulla 90). □

4.2 Yleisohjeita LP-tehtävien ratkaisemiseen

Lineaarinen optimointitehtävä ratkaistaan useimmiten kahdessa vaiheessa: ensiksi etsitään käypä piste ja tämän jälkeen käypä optimiratkaisu. Jos tehtävälle ei löydy käypää pistettä, voi apua löytää rajoitteiden muokkaamisesta siten, että ylä- tai alarajan rikkomisesta seuraa suurella kertoimella painotettu sakko, sen sijaan että käytettäisiin alkuperäisiä tiukkoja ylärajoja. Tällöin on helpompi löytää käypä piste, ja tarvittaessa voi tiukentaa rajoitusehtoja käyttämällä suurempia sakkokertoimia.

Jos järkevää ratkaisua ei tahdo löytyä, voi olla hyötyä LP-tehtävän skaalaamisesta sopivasti, eli yleensä kannattaa pyrkiä lähellä ykköstä oleviin muuttujien ja kertoimien arvoihin. Toisinaan LP-mallia muodostettaessa käytetään edellä mainittuja sakkokertoimia, joilla taataan käyvällä alueella pysyminen. Näitä vakioita ei tulisi valita liian suuriksi, jolloin ne voivat aiheuttaa numeerisia ongelmia.

Mikäli LP-tehtävä on jotain erikoistyyppiä, tulisi aina käyttää tehtävään kehitettyä erikoisalgoritmia. Esimerkiksi *verkkotehtävien* (network flow) ratkaiseminen tavallisina LP-tehtävinä voi olla hyvin tehotonta. Muita tällaisia tehtäviä ovat kuljetusten optimointi, tuotantolaitosten sijoittelu ja lyhimmän polun ongelma. Näille tehtäville on kehitetty tavalliseen simplex-menetelmään verrattuna huomattavasti tehokkaampia algoritmeja.

4.3 Simplex-algoritmi LP-tehtäville

Yhä eniten käytetty LP-tehtävien ratkaisualgoritmi on klassinen 40 vuotta vanha *simplex-menetelmä*. Seuraavassa algoritmin yleiskuvaus:

Algoritmi 4.3.1 (Simplex-menetelmän yleiskuvaus)

1: Hae käypä kulmapiste $\mathbf{x}^k \in \{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$.

2: Tarkista nykyisen pisteen $\mathbf{x}^k$ optimaalisuus. Mikäli optimi on löytynyt, voi iteroinnin lopettaa.

3: Hae suunta, johon kohdefunktion arvoa voidaan pienentää, ja kulje kyseiseen suuntaan kunnes tullaan käyvän alueen reunalle. Jatka vaiheesta 2.

Simplex-menetelmän tuottama ratkaisupiste on lineaarisia rajoitteita sisältävän tehtävän nurkkapiste.

Seuraavassa esitetään Lagrangen kertoimiin (kappale 3.3.2) perustuva esitys simplex-algoritmia läheisesti muistuttavasta *aktiivijoukkomenetelmästä*. Ratkaistavana on tehtävä

$$\operatorname*{minimi}_{\mathbf{x}} \langle \mathbf{c}, \mathbf{x} \rangle, \text{ kun } A\mathbf{x} \leq \mathbf{b} \text{ ja } \mathbf{x} \geq \mathbf{0}. \tag{4.10}$$

Esitystä kannattaa verrata epälineaarisen optimoinnin menetelmiin (luvut 6 ja 13).

Algoritmi 4.3.2 (LP-tehtävien aktiivijoukkomenetelmä)

1: Aseta $k = 1$. Hae käypä kulmapiste $\mathbf{x}^k$ ja aktiiviset rajoitteet sisältävä matriisi $\widetilde{A}_k$.

2: Laske Lagrangen kertoimet $\mathbf{u}^k$ yhtälöstä $\widetilde{A}_k^T \mathbf{u}^k = \mathbf{c}$. Jos $\mathbf{u}^k \leq \mathbf{0}$, piste $\mathbf{x}^k$ on haettu optimi.

3: Valitse jokin Lagrangen kertoimien $\mathbf{u}^k$ komponentti $u_s^k > 0$. Laske suunta $\mathbf{p}^k : \widetilde{A}_k \mathbf{p}^k = \mathbf{e}^s$, missä $\mathbf{e}^s$ on s:s koordinaattivektori.

4: Hae maksimiaskel α_k suuntaan $\mathbf{p}^k$ (kunnes törmätään rajoitteisiin). Siis $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{p}^k$.

5: Jos maksimiaskel α_k vie äärettömyyteen, kohdefunktio on rajoittamaton ja iteroinnin voi lopettaa.

6: Aseta $\mathbf{x}^{k+1} = \mathbf{x}^k + \alpha_k \mathbf{p}^k$ ja päivitä matriisiin $\widetilde{A}_{k+1}$ uudet aktiiviset rajoitteet (poista rivi s ja lisää uusi rajoite t).

7: Palaa vaiheeseen 2.

Tehokkaissa simplex-toteutuksissa käytetään harvoille matriiseille kehitettyjä hajotelmia ja esimerkiksi muuttujien ylä- ja alarajat otetaan huomioon erikseen. Edellä voisi aktiiviset rajoitteet sisältävän matriisin $\widetilde{A}_k$ tallettaa esimerkiksi LU-hajotelmana. Pienissä ja keskisuurissa tehtävissä ($n + m < 1\,000 \ldots 10\,000$) simplex on tehokkaasti koodattuna käytännössä paras ja luotettavin menetelmä. Suurissa tehtävissä (tuhansia muuttujia) ovat *sisäpistemenetelmät* usein tehokkaampia.

Simplex-menetelmän ongelmana on erittäin huono tehokkuus eräissä tehtävissä. On esimerkiksi mahdollista luoda ongelma, jota ratkaistessa simplex

käy läpi kaikki n:n muuttujan tehtävän 2^n kulmapistettä [HH91, Wri92a]. Toisaalta reaalimaailman tehtävissä simplex-menetelmän vaatima aika kasvaa useimmiten polynomisesti verrannollisena muuttujien lukumäärään n ja rajoitusehtojen lukumäärään m. Itse asiassa monissa tehtävissä kasvu on lineaarista!

4.4 Sisäpistemenetelmät

Jo vuonna 1968 esiteltiin sisäpistemenetelmien periaatteet ja keskeisiä teoreettisia tarkasteluja. Kuitenkin sisäpistemenetelmät olivat suhteellisen vähän käytettyjä ja tutkittuja menetelmiä, kunnes Karmarkar julkisti vuonna 1984 sisäpistealgoritminsa (interior-point method) LP-tehtäville. Karmarkarin menetelmä on simplex-menetelmää tehokkaampi erityisesti suurissa tehtävissä. Algoritmi generoi käyvän alueen sisäpisteitä ratkaisuehdokkaiksi ja on täten lähellä epälineaarisen optimoinnin algoritmeja (luku 6).

Karmarkarin algoritmi on *polynominen* eli LP-tehtävän ratkaisemiseen kuluu työmäärä, joka on polynomisesti verrannollinen muuttujien lukumäärään n. Simplex-algoritmihan on pahimmassa tapauksessa eksponentiaalinen eli ratkaisuaika voi olla verrannollinen potenssiin 2^n.

Karmarkarin menetelmän tehokkuus perustuu käyvän alueen reunan välttämiseen eli sisäpisteiden generointiin. Sisäpistemenetelmät käyttävät epälineaarisesta optimoinnista tuttuja este- ja sakkofunktioita (kappale 8.6) rajoitteiden toteuttamiseen. Näiden menetelmien kehittely jatkuu edelleen aktiivisena.

Useimmat sisäpistemenetelmät perustuvat *estefunktioiden* käyttämiseen. Estefunktion etuna on se, että se pakottaa etsintäalgoritmin iteraatit käyvän alueen sisälle. Usein käytetään logaritmista estefunktiota (sivu 148), jolloin ratkaistavaksi saadaan jono optimointitehtäviä:

$$\operatorname*{minimi}_{\mathbf{x}} P(\mathbf{x}, \mu_k) = f(\mathbf{x}) - \mu_k \sum_{i=1}^{p} \ln x_i, \text{ kun } A\mathbf{x} = \mathbf{b} \text{ ja } \mu_k > 0. \qquad (4.11)$$

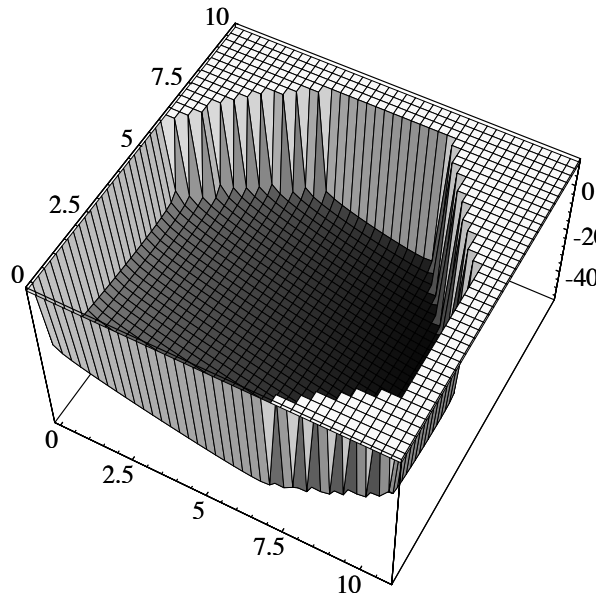
Tässä on sisällytetty LP-tehtävän (4.4) rajoitteet $\mathbf{x} \geq \mathbf{0}$ estefunktioon. Logaritminen estefunktio ei sovi sellaisenaan yhtälömuotoisille rajoitteille, sillä sen arvot kasvavat äärettömyyteen lähestyttäessä käyvän alueen reunaa. Estefunktion toimintaa LP-tehtävän yhteydessä on havainnollistettu kuvassa 4.4.

Seuraavassa esitetään *Newtonin menetelmään perustuva primaali estefunktioalgoritmi*, joka on pohjana monille sisäpistemenetelmille:

Algoritmi 4.4.1 (Primaali estefunktiomenetelmä)

aseta $k = 1$ ja valitse estefunktion parametri $\mu_k > 0$
hae $\mathbf{x}^k > \mathbf{0}$ siten, että $A\mathbf{x}^k = \mathbf{b}$
while piste $\mathbf{x}^k$ ei ole tarpeeksi lähellä LP-tehtävän optimia

$i = 1$
$\mathbf{x}_i^k = \mathbf{x}^k$
while piste $\mathbf{x}_i^k$ ei ole tarpeeksi lähellä tehtävän (4.11) optimia
hae estefunktion $P(\mathbf{x}, \mu_k)$ Newton-suunta $\mathbf{p}_i$ pisteessä $\mathbf{x}_i^k$
tee viivahaku: hae $\mathbf{x}_{i+1}^k = \mathbf{x}_i^k + \lambda_i \mathbf{p}_i,\ \lambda > 0$ siten, että pysytään käyvällä alueella ja $P(\mathbf{x}_{i+1}^k, \mu_k) < P(\mathbf{x}_i^k, \mu_k)$
$i = i + 1$
end
$\mathbf{x}^{k+1} = \mathbf{x}_i^k$
valitse $\mu_{k+1} < \mu_k$
$k = k + 1$
end



Kuva 4.4: *Kuvassa esitetään LP-tehtävän kohdefunktio muunnettuna logaritmisen estefunktion avulla. Uuden kohdefunktion arvot menevät äärettömän suuriksi käyvän alueen reunalla (kuvassa on esitetty vain arvot jotka ovat* ≤ 10*).*

Raskain osa algoritmia on estefunktion $P(\mathbf{x}, \mu_k)$ Newtonin hakusuunnan $\mathbf{p}_i$ laskeminen (kappale 6.5 sivulla 106). Viivahakuun voidaan käyttää logaritmiselle sakkofunktiolle kehitettyjä algoritmeja.

Primaalia estefunktioalgoritmia tehokkaampia ovat menetelmät, joissa käsitellään yhtä aikaa primaali- ja duaalitehtäviä. Standardimuotoisen LP-tehtävän duaali on

$$\underset{\mathbf{y}}{\text{maksimi}}\ \langle \mathbf{b}, \mathbf{y} \rangle,\ \text{kun}\ \mathbf{s} = \mathbf{c} - A^T\mathbf{y} \geq \mathbf{0}. \tag{4.12}$$

Duaalista saadaan johdettua LP-tehtävän optimikohdalle ehdot

$$A\mathbf{x} = \mathbf{b},\ A^T\mathbf{y} + \mathbf{s} = \mathbf{c},\ S\mathbf{x} = \mathbf{0},\ \mathbf{x} \geq \mathbf{0},\ \mathbf{s} \geq \mathbf{0}, \tag{4.13}$$

missä merkitään

$$S = \operatorname{diag}(\mathbf{s}) = \begin{pmatrix} s_1 & & \\ & \ddots & \\ & & s_n \end{pmatrix},$$

jolloin pätee $S\mathbf{x} = (s_1x_1, s_2x_2, \ldots, s_nx_n)^T$. Sisäpistemenetelmän iteraateille $(\mathbf{x}^k, \mathbf{y}^k, \mathbf{s}^k)$ pätee

$$\|A\mathbf{x}^k - \mathbf{b}^k\| \to 0,\ \|A^T\mathbf{y}^k + \mathbf{s}^k - \mathbf{c}\| \to 0,\ \langle \mathbf{x}^k, \mathbf{s}^k \rangle \to 0, \tag{4.14}$$

kun $k \to \infty$. Uusi iteraatiopiste saadaan asettamalla

$$\begin{pmatrix} \mathbf{x}^{k+1} \\ \mathbf{y}^{k+1} \\ \mathbf{s}^{k+1} \end{pmatrix} = \begin{pmatrix} \mathbf{x}^k \\ \mathbf{y}^k \\ \mathbf{s}^k \end{pmatrix} + \begin{pmatrix} \lambda_P \Delta\mathbf{x} \\ \lambda_D \Delta\mathbf{y} \\ \lambda_D \Delta\mathbf{s} \end{pmatrix}, \tag{4.15}$$

missä hakusuunnat $\Delta\mathbf{x}$, $\Delta\mathbf{y}$ ja $\Delta\mathbf{s}$ saadaan yhtälöryhmästä

$$\begin{cases} S_k\Delta\mathbf{x} + X_k\Delta\mathbf{s} = \sigma_k\mu_k\mathbf{1} - S_k\mathbf{x}^k, \\ A\Delta\mathbf{x} = \mathbf{b} - A\mathbf{x}^k, \\ A^T\Delta\mathbf{y} + \Delta\mathbf{s} = \mathbf{c} - A^T\mathbf{y}^k - \mathbf{s}^k. \end{cases} \tag{4.16}$$

Tässä käytettiin merkintää $S_k = \operatorname{diag}(\mathbf{s}^k)$ ja $X_k = \operatorname{diag}(\mathbf{x}^k)$. Kerroin $\sigma_k \in [0,1]$ on menetelmän vaimennuskerroin ja kerroin μ_k saadaan asettamalla $\mu_k = \langle \mathbf{x}^k, \mathbf{s}^k \rangle / n$.

Riippuen askelpituuksien λ_P ja λ_D valinnasta saadaan erilaisia sisäpistemenetelmiä. Suppenemiseen tarvittavien iteraattien lukumäärä on tyypillisesti luokkaa $\mathcal{O}(n)$ tai luokkaa $\mathcal{O}(n^{3/2})$.

Sisäpistealgoritmeilla voi olla joissakin tapauksissa numeerisia ongelmia. Esimerkiksi jos rajoitematriisissa A on ns. tiheitä sarakkeita (suurin osa alkioista nollasta eroavia), voi tehokkuus kärsiä. Stabiilit sisäpistemenetelmät käsittelevät esimerkiksi iteraatiomatriisien LDL^T-hajotelmia pyrkien mahdollisimman tehokkaaseen muistinkäyttöön ja tehokkuuteen.

4.5 LP-tehtävien ratkaisuohjelmistot

Tehokkaat kaupalliset LP-ratkaisijat raporttigeneraattoreineen voivat olla hyvinkin kalliita. Toisaalta saatavilla on muutamia vapaasti levitettäviä LP-ohjelmistoja. Minos-ohjelmiston (liite B) saa akateemiseen käyttöön nimellistä korvausta vastaan.

Useat LP-tehtävien ratkaisuohjelmistot osaavat lukea MPS-formaatissa (kappale 4.6) annettuja LP-tehtävien määrittelyjä. Kappaleessa 4.5.1 esitetään dieettiongelma ja sen ratkaisu GAMS-mallinnuskielellä. Kappaleessa 4.6 esitetään saman tehtävän ratkaisu MPS-formaattia käyttävillä ongelmistoilla.

4.5.1 GAMS-ohjelmiston käyttö

Seuraavassa esitetään yksinkertaisen LP-mallin muodostaminen sekä mallin esittäminen ja ratkaiseminen GAMS-ohjelmistolla.

Esimerkki 4.5.1 *Dieettiongelmassa* täytyy valita mahdollisimman halpa ateria, jotta päivittäinen ravinnontarve tulee tyydytettyä. Olkoon muuttujissa $x_i \geq 0$, $i = 1, \ldots, n$ ruokalajien i päivittäiset määrät. Valitun ruokavalion on sisällettävä ravintoainetta $j = 1, \ldots, m$ määrä ρ_j (tässä ravintoaineella tarkoitetaan varsinaisten ravintoaineiden lisäksi myös energiasisältöä). Olkoon r_{ij} kunkin ruokalajin i ravintoainepitoisuus. Rajoitteiksi saadaan siis

$$\sum_{i=1}^{n} r_{ij} x_i \geq \rho_j, \quad j = 1, \ldots, m.$$

Lisäksi voidaan asettaa esimerkiksi ylärajoja tietyn ruokalajin päivittäisille määrille.

Listauksessa 4.1 on esitetty erästä dieettiongelmaa kuvaava GAMS-malli. Indeksijoukkoja määritellään `SETS`-lauseella ja parametrien arvot annetaan lauseissa `PARAMETER` ja `TABLE`. Yhtälöiden määrittelyissä käytetään vertailuoperaatioita `=G=` eli ”suurempi kuin” ja `=E=` eli ”yhtä suuri kuin”. Käskyllä `x.up(r) = a(r,'MAXANNOS');` asetetaan muuttujalle `x` ylärajat indeksijoukon `r` arvoilla. Merkillä `*` alkava rivi on kommentti. Tämä LP-malli on matriisimuodossa

$$\operatorname*{minimi}_{\mathbf{x}} \langle \mathbf{c}, \mathbf{x} \rangle, \text{ kun } A\mathbf{x} \geq \mathbf{b} \text{ ja } \mathbf{0} \leq \mathbf{x} \leq \mathbf{x}^u. \tag{4.17}$$

Tässä kohdefunktion painokertoimet $\mathbf{c}$ vastaavat ruokalajien hintoja. Listauksessa 4.1 on annettu seuraavat numeroarvot:

$$A = \begin{pmatrix} 110 & 205 & 160 & 160 & 420 & 260 \\ 4 & 32 & 13 & 8 & 4 & 14 \\ 2 & 12 & 54 & 285 & 22 & 80 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 2000 \\ 55 \\ 800 \end{pmatrix}.$$

Painokertoimiksi $\mathbf{c}$ ja muuttujien ylärajoiksi $\mathbf{x}^u$ asetetaan

$$\mathbf{c} = \begin{pmatrix} 1 & 8 & 4 & 4 & 7 & 6 \end{pmatrix}^T, \quad \mathbf{x}^u = \begin{pmatrix} 4 & 3 & 2 & 8 & 2 & 5 \end{pmatrix}^T.$$

Mallin minimoivassa ruokavaliossa kaurapuuroa ja piirakkaa otetaan maksimimäärä eli 4 ja 2 annosta, kun taas kanaa, kananmunia ja hernekeittoa ei oteta yhtään (muuttujat alarajoillaan). Maitoa otetaan 4.5 annosta. Energiantarve saadaan tyydytettyä vähimmäismäärällä, ja proteiineja ja kalsiumia saadaan yli vähimmäistarpeen. Kokonaishinnaksi tulee 31.50 mk mallissa annetuilla hinnoilla.

4.6 MPS-formaatti

Listauksessa 4.2 on esimerkin 4.5.1 dieettitehtävän esitys MPS-formaatissa. MPS-formaatti määrittelee tarkan sarakejaon. Esimerkiksi matriisin A alkiot luetaan alkaen sarakkeista 1, 5, 15, 25, 40 ja 50.

Rivien tunnuksina voi käyttää mm. kirjaimia L (tarkoittaa ≤), G (≥), E (=) ja N (rajoittamaton, esimerkiksi kohdefunktion kerroinrivi). Muuttujien tyyppiä kuvaamaan voi käyttää seuraavassa esitettyjä tunnuskirjaimia.

Taulukko 4.1: *MPS-formaatin BOUNDS-osassa käytettäviä tunnuksia ja niiden merkityksiä.*

Tunnus	*Merkitys*
UP	Yläraja
LO	Alaraja
FR	Vapaa muuttuja
FX	Kiinteä arvo
MI	Alaraja miinus ääretön
PL	Yläraja plus ääretön
BV	Binäärimuuttuja
UI	Kokonaislukumuuttujan yläraja

Listaus 4.1: *Dieettiongelman GAMS-malli.*

```
SETS nh Ravintoaineet ja hinta / ENERGIA kCal, PROTEIINIT g,
                  KALSIUM mg, HINTA mk, MAXANNOS kpl /
     n(nh) Ravintoaineet / ENERGIA, PROTEIINIT, KALSIUM /
     r Ruokalajit / KAURAPUURO, KANA, KANANMUNAT, MAITO,
                  PIIRAKKA, HERNEKEITT /;

PARAMETER b(n) Alarajat / ENERGIA 2000, PROTEIINIT 55,
                        KALSIUM 800 /;
TABLE a(r,nh) Ruokalajien sisaltamat ravintoaineet ja hinta
              ENERGIA  PROTEIINIT  KALSIUM   HINTA  MAXANNOS
*              kcal        g         mg       mk      kpl
KAURAPUURO     110.0      4.0        2.0      1.0     4.0
KANA           205.0     32.0       12.0      8.0     3.0
KANANMUNAT     160.0     13.0       54.0      4.0     2.0
MAITO          160.0      8.0      285.0      3.0     8.0
PIIRAKKA       420.0      4.0       22.0      7.0     2.0
HERNEKEITT     260.0     14.0       80.0      6.0     2.0

POSITIVE VARIABLE x(r) Paivittaisen ruoan maara annoksina;
FREE VARIABLE Kustannus Ruoan kokonaishinta;

EQUATIONS nb(n), cb;

nb(n).. SUM(r, a(r,n)*x(r)) =G= b(n);
cb..    Kustannus =E= SUM(r, a(r,'HINTA')*x(r));
x.up(r) = a(r,'MAXANNOS');

MODEL Dieetti / nb, cb /;
SOLVE Dieetti MINIMIZING Kustannus USING LP;
```

Listaus 4.2: *Dieettiongelman MPS-mallitiedosto. Ylhäällä on esitetty sarakkeiden numerointi. MPS-formaatti vaatii syöttötietojen sijaitsevan tietyissä sarakkeissa ja lisäksi avainsanat tulee kirjoittaa isoin kirjaimin. Itse mallissa saa käyttää sekaisin pieniä ja isoja kirjaimia. Tunnuksella* `N` *merkitty rivi* `kustannus` *sisältää kohdefunktion kertoimet.* `RHS`*-osuudessa voi tarvittaessa määritellä useita rajoitevektoreita antamalla niille eri nimen kunkin rivin ensimmäisessä kentässä.*

```
         1         2         3         4         5         6
123456789012345678901234567890123456789012345678901234567890
```

```
NAME          DIEETTI
ROWS
 G  energia
 G  proteiin
 G  kalsium
 N  kustann
COLUMNS
    kaurapu   energia     110.0        proteiin      4.0
    kaurapu   kalsium       2.0        kustann       1.0
    kana      energia     205.0        proteiin     32.0
    kana      kalsium      12.0        kustann       8.0
    kmunat    energia     160.0        proteiin     13.0
    kmunat    kalsium      54.0        kustann       4.0
    maito     energia     160.0        proteiin      8.0
    maito     kalsium     285.0        kustann       3.0
    piirakka  energia     420.0        proteiin      4.0
    piirakka  kalsium      22.0        kustann       7.0
    hernekei  energia     260.0        proteiin     14.0
    hernekei  kalsium      80.0        kustann       6.0
RHS
    minimit   energia    2000.0        proteiin     55.0
    minimit   kalsium     800.0
BOUNDS
 UP annokset  kaurapu       4.0
 UP annokset  kana          3.0
 UP annokset  kmunat        2.0
 UP annokset  maito         8.0
 UP annokset  piirakka      2.0
 UP annokset  hernekei      2.0
ENDATA
```

MPS-tiedostossa ei kerrota, onko kyse minimointi- vai maksimointitehtävästä. Kukin ratkaisuohjelmisto toimii omien asetustensa perusteella. MPS-mallista on erilaisia laajennuksia kokonaislukutehtäville ja kvadraattisille tehtäville.

4.7 Kvadraattiset tehtävät

Epälineaarisen optimointitehtävän (luku 6) erikoistyyppi on *kvadraattinen optimointi* (quadratic programming eli QP):

$$\underset{\mathbf{x}}{\text{minimi}}\ \langle \mathbf{c}, \mathbf{x}\rangle + \frac{1}{2}\langle \mathbf{x}, Q\mathbf{x}\rangle, \text{ kun } A\mathbf{x} = \mathbf{b} \text{ ja } \mathbf{x} \geq \mathbf{0}, \tag{4.18}$$

missä vektori $\mathbf{q}$ 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 auki kirjoitettuna

$$\langle \mathbf{c}, \mathbf{x}\rangle + \frac{1}{2}\langle \mathbf{x}, Q\mathbf{x}\rangle = \sum_{i=1}^{n} c_i x_i + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij} x_i x_j.$$

Esimerkki 4.7.1 Perinteinen esimerkki kvadraattisesta tehtävästä on tilastollinen osakesalkkumalli:

$$\underset{\mathbf{x}}{\text{minimi}}\ f(\mathbf{x}) = \langle \mathbf{x}, S\mathbf{x}\rangle = \sum_{i=1}^{n}\sum_{j=1}^{n} s_{ij} x_i x_j,$$

kun kohdemuuttujille x_i asetetaan positiivisuusehto $x_i \geq 0$ sekä lisäksi rajoitteet

$$\sum_{i=1}^{n} x_i = \alpha \quad \text{ja} \quad \sum_{i=1}^{n} r_i x_i \geq \beta.$$

Tässä parametri r_i on osakkeille i odotettu vuosittainen tuotto ja s_{ii} on tuoton varianssi, joka kuvastaa sijoituksen riskiä. Vastaavasti s_{ij} on osakkeiden i ja j tuottojen kovarianssi. Tavoitteena on saavuttaa voittoa ainakin määrä β, mutta minimoida samalla osakesalkussa olevien osakkeiden tuoton varianssi. Kokonaisinvestoinnin määrä on α.

Seuraavassa esimerkissä ratkaistaan edellä esitetty kvadraattinen optimointitehtävä GAMS-ohjelmiston avulla.

Esimerkki 4.7.2 Esimerkissä 4.7.1 esitetyssä osakesalkkumallissa pyritään minimoimaan riskiä. Listauksessa 4.3 on esitetty GAMS-malli, joka määrittelee erään kvadraattisen osakesalkkutehtävän. Minimoitava kvadraattinen kohdefunktio on tässä tapauksessa

$$f(\mathbf{x}) = 4x_1^2 + 6x_1x_2 + 6x_2^2 - 2x_1x_3 + 2x_2x_3 + 10x_3^2,$$

missä x_1 on sijoitus rautaan, x_2 sijoitus softaan jne.

Varianssit ja kovarianssit sijoitetaan taulukkoon `v(i,j)`, missä `i` ja `j` ovat sijoituskohteiden indeksijoukot. Mallissa pyritään minimoimaan sijoitusten varianssia. Lauseke `SUM(i,`*lauseke*`)` vastaa matemaattista notaatiota $\sum_i$ *lauseke*.

Kun minimointitehtävän ratkaisemiseen käytetään GAMSin Minos-moduulia, saadaan sijoitusten prosenttiosuuksiksi: 30.3% rauta, 8.7% softa, 50.5% viihde ja 10.6% kiinteäkorkoinen talletus. Minimivarianssiksi saadaan 2.9.

4.7.1 Kvadraattisen tehtävän optimaalisuusehdot

Muodostetaan kvadraattiselle optimointitehtävälle Lagrangen funktio:

$$L(\mathbf{x},\mathbf{u},\mathbf{v}) = \langle \mathbf{c},\mathbf{x}\rangle + \frac{1}{2}\langle \mathbf{x}, Q\mathbf{x}\rangle - \langle \mathbf{u},\mathbf{x}\rangle + \langle \mathbf{v}, A\mathbf{x}-\mathbf{b}\rangle. \tag{4.19}$$

Minimikohdassa $\mathbf{x}^*, \mathbf{u}^*, \mathbf{v}^*$ saadaan Karush-Kuhn-Tucker -ehdoiksi (sivu 62)

$$\begin{cases} A\mathbf{x}^* - \mathbf{b} = \mathbf{0},\ \mathbf{x}^* \geq \mathbf{0},\ \mathbf{u}^* \geq \mathbf{0}, \\ \langle \mathbf{u}^*, \mathbf{x}^*\rangle = \mathbf{0}, \\ \nabla L(\mathbf{x}^*, \mathbf{u}^*, \mathbf{v}^*) = \mathbf{c} + Q\mathbf{x}^* - \mathbf{u}^* + A^T\mathbf{v}^* = \mathbf{0}. \end{cases} \tag{4.20}$$

Listaus 4.3: *Kvadraattisen osakesalkkutehtävän GAMS-malli.*

```
SET i Osakkeet / RAUTA, SOFTA, VIIHDE, KIINTKORKO /;
ALIAS (i,j);

SCALAR Tavoite Keskim. vuosittainen tuotto (%) / 10 /;
PARAMETER Keskim(i) Keskim. vuosittainen tuotto (%)
     / RAUTA 8, SOFTA 9, VIIHDE 12, KIINTKORKO 7 /;

TABLE v(i,j) Varianssit ja kovarianssit
                RAUTA   SOFTA   VIIHDE   KIINTKORKO
  RAUTA           4       3       -1         0
  SOFTA           3       6        1         0
  VIIHDE         -1       1       10         0
  KIINTKORKO      0       0        0         0

POSITIVE VARIABLE x(i) Kohteeseen i sijoitettu osuus;
VARIABLE Varianssi Osakesalkun varianssi;

EQUATIONS Sijsumma, Keskituot, Vareq;
Sijsumma.. SUM(i, x(i)) =E= 1.0;
Keskituot.. SUM(i, Keskim(i)*x(i)) =G= Tavoite;
Vareq.. SUM(i, x(i)*SUM(j, v(i,j)*x(j))) =E= Varianssi;

MODEL Osakkeet / ALL /;
SOLVE Osakkeet USING NLP MINIMIZING Varianssi;
```

Tehtävän (4.18) *duaali* määritellään seuraavasti:

$$\underset{\mathbf{u},\mathbf{v}}{\text{maksimi}}\ d(\mathbf{u},\mathbf{v}),\ \text{kun}\ \mathbf{u} \geq \mathbf{0}, \tag{4.21}$$

missä $d(\mathbf{u},\mathbf{v}) = \inf_{\mathbf{x}} L(\mathbf{x},\mathbf{u},\mathbf{v})$. Yhtälön (4.19) Lagrangen funktio on konveksi funktio, kun matriisi Q on positiivisesti definiitti. Tällöin Lagrangen funktion minimipiste $\mathbf{x}^*$ löytyy Lagrangen funktion gradientin nollakohdasta eli saadaan ehto

$$\nabla L(\mathbf{x},\mathbf{u},\mathbf{v}) = \mathbf{c} + Q\mathbf{x} - \mathbf{u} + A^T\mathbf{v} = \mathbf{0}. \tag{4.22}$$

Minimikohdassa $\mathbf{x}^*$ pätee $\langle \mathbf{x}^*, \nabla L(\mathbf{x}^*,\mathbf{u},\mathbf{v})\rangle = 0$. Siten termi $\inf_{\mathbf{x}} L(\mathbf{x},\mathbf{u},\mathbf{v})$ saadaan yksinkertaisempaan muotoon sijoituksella

$$\langle \mathbf{c},\mathbf{x}\rangle + \langle \mathbf{v}, A\mathbf{x}\rangle = -\langle \mathbf{x}, Q\mathbf{x}\rangle + \langle \mathbf{u},\mathbf{x}\rangle. \tag{4.23}$$

Eliminoimalla vektori $\mathbf{u} \geq \mathbf{0}$ yhtälöstä (4.22) saadaan kvadraattisen optimointitehtävän duaalitehtäväksi

$$\underset{\mathbf{x},\mathbf{v}}{\text{maksimi}}\ d(\mathbf{x},\mathbf{v}) = -\langle \mathbf{b},\mathbf{v}\rangle - \frac{1}{2}\langle \mathbf{x}, Q\mathbf{x}\rangle,\ \text{kun}\ \mathbf{c} + Q\mathbf{x} + A^T\mathbf{v} \geq \mathbf{0}. \tag{4.24}$$

Yhtälöitä (4.19) – (4.24) hyväksi käyttäen voidaan johtaa kvadraattisten tehtävien ratkaisualgoritmeja, joita käytetään apuna esimerkiksi toistetussa kvadraattisessa optimoinnissa (kappale 8.5 sivulla 143).

4.7.2 Aktiivijoukkomenetelmät

Konveksit kvadraattiset optimointitehtävät ovat vain hiukan LP-tehtäviä vaikeampia ratkaista. Ratkaisumenetelmiä on johdettu käyttämällä runkona tavallisten LP-tehtävien algoritmeja tai erilaisia aktiivijoukkostrategioita. Toisinaan kvadraattisten tehtävien ratkaisemiseen käytetään myös yleisten epälineaaristen optimointitehtävien algoritmeja, vaikkakaan niiden tehokkuus ei ole erikoisalgoritmien luokkaa.

Aktiivijoukkomenetelmiä kutsutaan joskus myös *projektiomenetelmiksi*, sillä niissä haetaan hakuaskel $\mathbf{s}^k$ projisoiduista Newtonin yhtälöistä

$$Z_k^T Q Z_k \mathbf{p}^k = -Z_k^T (Q\mathbf{x}_k + \mathbf{c}), \tag{4.25}$$

$$\mathbf{s}^k = Z_k \mathbf{p}^k, \tag{4.26}$$

missä matriisi Z_k on kanta aktiivisista rajoitteista muodostetun matriisin A_k' ytimelle (siis $A_k' Z_k = \mathbf{0}$). Matriisia A_k' päivitetään kullakin iteraatiolla epäyhtälömuodossa annettujen rajoitteiden suhteen. Uusi iteraatiopiste saadaan yhtälöstä

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \lambda_k \mathbf{s}^k. \tag{4.27}$$

Hakuaskeleen pituudelle λ_k on tehtävän kvadraattisuudesta johtuen kaksi vaihtoehtoa. Askelpituudella $\lambda_k = 1$ päästään funktion minimiin matriisin A_k' ytimen virittämässä aliavaruudessa. Jos tämä askel tuottaa käyvän pisteen, on kyseessä kvadraattisen optimointitehtävän minimipiste. Muuten lähimpään rajoitteeseen päästään ykköstä lyhyemmällä askeleella, ja seuraavalla iteraatiokierroksella lisätään aktiivijoukkoon uusi rajoite.

4.7.3 Sisäpistemenetelmät

Suuria kvadraattisia tehtäviä varten on viime aikoina kehitetty sisäpistemenetelmiä aivan kuten LP-tehtävillekin. Kvadraattisten tehtävien ratkaisualgoritmeja käytetään apuna esimerkiksi toistetussa kvadraattisessa optimoinnissa, joten tehostamalla QP-tehtävien ratkaisua voidaan ratkaista tehokkaammin yleisempiäkin optimointitehtäviä.

Kappaleessa 4.7.1 johdettiin kvadraattisen optimointitehtävän duaali:

$$\underset{\mathbf{x},\mathbf{v}}{\text{maksimi}}\ d(\mathbf{x},\mathbf{v}) = -\langle \mathbf{b},\mathbf{v}\rangle - \frac{1}{2}\langle \mathbf{x}, Q\mathbf{x}\rangle,\ \text{kun}\ \mathbf{c} + Q\mathbf{x} + A^T\mathbf{v} \geq \mathbf{0}. \tag{4.28}$$

Kvadraattisten optimointitehtävien sisäpistemenetelmissä hakusuunnat $\Delta\mathbf{x}$, $\Delta\mathbf{v}$ ja $\Delta\mathbf{s}$ saadaan ratkaisemalla seuraava lineaarinen yhtälöryhmä, joka on johdettu käyttäen hyväksi duaalia:

$$\begin{cases} D_x\Delta\mathbf{s} + D_s\Delta\mathbf{x} = \mu\mathbf{1} - X\,\mathbf{s}, \\ A\Delta\mathbf{x} = \mathbf{0}, \\ \Delta\mathbf{s} = Q\Delta\mathbf{x} + A^T\Delta\mathbf{v}. \end{cases} \tag{4.29}$$

Edellä käytetään merkintää $\mathbf{s} = \mathbf{c} + Q\mathbf{x} + A^T\mathbf{v}$ sekä notaatiota

$$X = \operatorname{diag}(\mathbf{x}) = \begin{pmatrix} x_1 & & 0 \\ & \ddots & \\ 0 & & x_n \end{pmatrix}.$$

Riippuen matriisien D_x ja D_s sekä skalaarin μ valinnasta saadaan erilaisia sisäpistemenetelmiä. Tehokkaimpia menetelmiä on *primaali-duaali potentiaalin vähennysalgoritmi* (potential reduction algorithm), jossa asetetaan

$$D_x = \operatorname{diag}(\mathbf{x}), \quad D_s = \operatorname{diag}(\mathbf{s}), \quad \mu = \frac{\langle \mathbf{x},\mathbf{s}\rangle}{\rho}, \tag{4.30}$$

missä parametri $\rho \geq n + \sqrt{n}$. Yhtälöiden (4.29) ja (4.30) määrittelemä algoritmi suppenee polynomisesti konvekseissa kvadraattisissa optimointitehtävissä. Menetelmä suppenee luokkaa $\mathcal{O}(nL)$ iteraatiolla, kun alkuarvaus on kyllin hyvä. Tässä L on kvadraattisen tehtävän syöttödatan kokoon (matriisien alkioiden lukumäärä) verrannollinen termi.

Jos kvadraattinen optimointitehtävä ei ole konveksi, on tehtävän ratkaiseminen erittäin vaikeaa. Yksikin matriisin Q negatiivinen ominaisarvo riittää tekemään tehtävästä kombinatorisen optimointitehtävän ja vieläpä ns. NP-kovan tehtävän [PV90, Roh95]. Ratkaisuohjelmiston nimittäin täytyisi periaatteessa käydä läpi kaikki rajoitteiden muodostamat kulmapisteet.

4.8 Ohjelmistoja

Edellä kerrottiin GAMS- ja Minos-ohjelmistojen käytöstä lineaaristen ja kvadraattisten optimointitehtävien ratkaisemiseen. Myös IMSL- ja NAG-aliohjelmakirjastot ja Netlib tarjoavat ratkaisemiseen soveltuvia rutiineja. Varsinaisia LP-ratkaisijoita on tarjolla monia (liite B).

4.8.1 Aliohjelmakirjastot IMSL ja NAG

IMSL-aliohjelmakirjasto sisältää rutiinin DLPRS tiheiden LP-tehtävien ratkaisemiseen ja rutiinin QPROG kvadraattisten tehtävien ratkaisemiseen. NAG-aliohjelmakirjasto puolestaan tarjoaa rutiinin E04MFF tiheiden LP-tehtävien ratkaisemiseen ja rutiinit E04NAF ja E04NCF tiheiden lineaaristen ja kvadraattisten optimointitehtävien ratkaisemiseen.

IMSL- ja NAG-aliohjelmakirjastojen käsikirjat [IMS, Num] tarjoavat perusteelliset kuvaukset ja esimerkit rutiinien käyttämisestä (viite [HHL+03] ja liite B).

4.8.2 Muita ohjelmistoja

Mathematica- ja Maple-ohjelmistot osaavat ratkaista pienehköjä LP-tehtäviä. Matlabin Optimization Toolbox sisältää myös LP-ratkaisijan. SAS-ohjelmistoon kuuluva SAS/OR osaa ratkaista LP-tehtäviä ja sisältää lisäksi monipuoliset raportointimahdollisuudet. SAS-ohjelmiston LP-proseduurilla voi myös suorittaa herkkyysanalyysin eli tutkia LP-tehtävän parametrien vaikutusta ratkaisuun. Lisätietoja löytyy liitteestä B.

Netlib-verkkoarkiston hakemistosta `lp` löytyy testiprobleemoja LP-ratkaisijoiden vertaamiseen. Netlibin hakemistoista `opt` ja `toms` löytyy ohjelmakoodeja lineaaristen ja kvadraattisten optimointitehtävien ratkaisemiseen (taulukko B.6 sivulla 214). Netlibistä löytyy lisäksi HOPDM-ohjelmisto, joka käyttää sisäpistemenetelmää LP-tehtävien ratkaisemiseen.

Ohjelmistoja on esitelty tämän oppaan liitteessä B. CSC:n opas *Matemaattiset ohjelmistot* [HHL+03] kertoo käytettävissä olevista ohjelmistoista.

4.9 Lisätietoja

Teoksessa *Numerical Linear Algebra and Optimization* [GMW91] on erittäin hyvä lineaarialgebraan pohjautuva esitys simplex-menetelmän perusteista. Teos *Large Sparse Numerical Optimization* [Col84] esittelee algoritmeja harvoille tehtäville. Teoksessa *The Mathematics of Nonlinear Programming* [PSU88] on esitelty LP-tehtävien ratkaisemista matemaattisesta näkökulmasta. Uusinta tietoa optimointimenetelmistä saa sarjajulkaisuista *Operations Research*, *SIAM Review* ja *ACM Transactions on Mathematical Software*.

Sisäpistemenetelmien periaatteet esiteltiin vuonna 1968 teoksessa *Nonlinear Programming: Sequential Unconstrained Minimization Techniques* [FM68]. Lineaaristen optimointitehtävien sisäpistemenetelmien nopea kehitys sai alkunsa Karmarkarin artikkelista [Kar84]. Artikkeli [Fra87] käsittelee Karmarkarin sisäpistemenetelmän suppenemista. Artikkelissa [GMS+86] osoitetaan Karmarkarin menetelmän olevan ekvivalentti sisäpistemenetel-

mien kanssa. Artikkelissa [BGL+92] kerrotaan erittäin suurten LP-tehtävien ratkaisemisesta yhdistämällä simplex- ja sisäpistemenetelmät.

Artikkelit [Gon92, Wri92a] ovat perusteellisia katsaustyyppisiä johdatuksia sisäpistemenetelmiin. Teos [Meg89] sisältää katsauksia sisäpistemenetelmien kehitykseen. Sisäpistemenetelmien katsaustyyppisiä esityksiä löytyy viitteistä [Wri92b, Öst92, Tod95]. Teoksessa [HH91] on esitelty LP-tehtävien ratkaisemiseen liittyviä numeerisia seikkoja. Sisäpistemenetelmien sovelluksia on esitelty artikkeleissa [HPY91, Wrib]. Raportissa [MW] on käsitelty sisäpistemenetelmissä tarvittavan viivahaun toteuttamista.

Verkkotehtävien (network flow) ratkaisua on käsitelty viitteissä [AMO91, Mey84, KH80].

Kvadraattisten optimointitehtävien aktiivijoukkomenetelmiä on esitelty teoksessa [GMW81]. Kvadraattisten optimointitehtävien ratkaisumenetelmien johtamista on käsitelty viitteissä [Ye, PY91, LMS92a, LMS92b, Wria]. Epäkonveksien kvadraattisten tehtävien tapausta on tarkasteltu viitteessä [PV90].

Yleisten lineaarisia rajoitteita sisältävien optimointitehtävien ratkaisumenetelmiä on kuvattu teoksessa [BSS93]. Esimerkiksi *Wolfen menetelmä* [Wol59, PSU88] on hyvin lähellä LP-tehtävien simplex-algoritmia.

5 Kokonaisluku- ja sekalukuoptimointi

Tässä luvussa esitellään lyhyesti kokonaislukutehtävien taustaa sekä eräiden ratkaisuohjelmistojen käyttöä. Ratkaisualgoritmien toimintaperiaatteita ei esitellä tarkemmin, mutta joitakin perusohjeita tehtävien ratkaisemiseen annetaan.

5.1 Kokonaislukutehtävät

Kokonaislukuoptimoinnissa (integer programming) optimointitehtävän muuttujat saavat kokonaislukuarvoja. Osa muuttujista voi olla jatkuvia, jolloin puhutaan *sekalukuoptimoinnista* (*mixed-integer programming* eli MIP). Ääritapauksessa muuttujat saavat arvoja $\{0, 1\}$, jolloin tehtävää kutsutaan $0/1$-optimoinniksi.

Seuraavassa käsitellään *sekalukutehtäviä* eli lineaarisia optimointitehtäviä, jossa osa muuttujista saa kokonaislukuarvoja. Näitä tehtäviä varten löytyy kohtuullisen tehokkaita ratkaisuohjelmistoja. Epälineaaristen rajoitusehtojen tai kohdefunktion tapauksille ei löydy tehokkaita yleisalgoritmeja, jos muuttujat saavat kokonaislukuarvoja. Kuitenkin esimerkiksi polynomifunktioita sisältävät $0/1$-tehtävät voidaan saattaa tavallisiksi $0/1$-optimointitehtäviksi.

Tässä luvussa käsitellään siis sekalukutehtäviä, jotka ovat muotoa

$$\min_{\mathbf{x},\mathbf{y}} \langle \mathbf{c}, \mathbf{x} \rangle + \langle \mathbf{d}, \mathbf{y} \rangle, \text{ kun } A\mathbf{x} + D\mathbf{y} \leq \mathbf{b}, \tag{5.1}$$

ja lisäksi $\mathbf{x} \geq \mathbf{0}$ ja $\mathbf{0} \leq \mathbf{y} \leq \mathbf{y}^u$. Muuttujat y_i, $i = 1, \ldots, r$ ovat kokonaislukuarvoisia.

Seuraavassa on klassinen esimerkki kokonaislukutehtävänä ratkeavasta ongelmasta.

Esimerkki 5.1.1 Tuotetaan n:ää eri tuotetta. Tuotantokustannus koostuu tuotteen i kertakustannuksesta s_i sekä määrästä x_i riippuvasta kustannuksesta $c_i x_i$. Tuotteet myydään yksikköhinnalla p_i. Haetaan optimaalinen tuotantosuunnitelma, kun kutakin tuotetta voidaan valmistaa maksimissaan määrä α_i ja kokonaistuotannolla on raja β.

Tuotteelle i saadaan kustannusfunktioksi $C_i(x_i)$:

$$C_i(x_i) = \begin{cases} s_i + c_i x_i, & \text{jos } x_i > 0, \\ 0, & \text{jos } x_i = 0. \end{cases}$$

Tämä kustannusfunktio on kuitenkin epäjatkuva, kun $x_i = 0$, joten syntyvän tehtävän ratkaiseminen on vaikeaa. Lisätään tehtävään binäärimuuttuja y_i:

$$y_i = \begin{cases} 1, & \text{jos } x_i > 0, \\ 0, & \text{jos } x_i = 0. \end{cases}$$

ja muuttujia x_i ja y_i koskeva ehto

$$x_i \le \alpha_i y_i.$$

Tässä α_i on muuttujaa x_i koskeva yläraja eli ehtona on $x_i \le \alpha_i$. Maksimoidaan tuottoa, joten optimointitehtäväksi saadaan

$$\underset{\mathbf{x}}{\text{maksimi}}\ f(\mathbf{x}),$$

missä kohdefunktio f on

$$f(\mathbf{x}) = \sum_{i=1}^{n} p_i x_i - \sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{n} s_i y_i = \langle \mathbf{p} - \mathbf{c}, \mathbf{x} \rangle - \langle \mathbf{s}, \mathbf{y} \rangle , \tag{5.2}$$

missä $x_i \ge 0$ ja $y_i \in \{0, 1\},\ i = 1, \ldots, n$. Rajoitteiksi saadaan

$$\begin{cases} 0 \le x_i \le \alpha_i y_i, & \text{kaikilla } i = 1, \ldots, n, \\ y_i = 0 \text{ tai } y_i = 1, & \text{kaikilla } i = 1, \ldots, n, \\ \sum_i x_i \le \beta. \end{cases} \tag{5.3}$$

Esimerkin 5.1.1 tehtävä ratkaistaan esimerkissä 5.4.1. Seuraavassa käytetään kokonaislukumuuttujia loogisten määrittelyjen esittämiseen.

Esimerkki 5.1.2 Olkoon käsiteltävinä joukko toimintoja P_i. Tehtävänä on mallittaa binäärimuuttujilla toimintoja koskevia loogisia lauseita.

1. Määritellään lauseke "toiminnosta P_1 seuraa P_2 ja P_3" eli $P_1 \Rightarrow P_2$ ja P_3. Lauseke voidaan esittää kolmen binäärimuuttujan y_i avulla:

$$y_1 \le \frac{1}{2}(y_2 + y_3).$$

Toinen mahdollisuus on huomata, että lauseke voidaan esittää muodossa: $(P_1 \Rightarrow P_2)$ ja $(P_1 \Rightarrow P_3)$. Tämä voidaan mallittaa epäyhtälöillä

$$y_1 \le y_2, \quad y_1 \le y_3.$$

Jälkimmäinen tapa on käytännön laskuissa yleensä tehokkaampi.

2. Vain yksi vaihtoehdoista $P_1, P_2, \ldots, P_n$ voidaan toteuttaa. Määritellään binäärimuuttujat $y_1, \ldots, y_n$ ja asetetaan ehto

$$\sum_{i=1}^{n} y_i \leq 1.$$

Siis vain yksi binäärimuuttujista y_i voi olla nollasta poikeava.

Edellisessä esimerkissä mainittuja mallinnusideoita voi käyttää seuraavassa esitellyllä tavalla.

Esimerkki 5.1.3 *Sijoittelutehtävässä* (assignment problem) työntekijöille $i = 1, \ldots, n$ on tarjolla tehtävät $j = 1, \ldots, n$. Kunkin työntekijän kustannukset $c_{ij} > 0$ tiedetään. Tehtävänä on minimoida kokonaiskustannukset sijoittelemalla työntekijät tehtäviinsä mahdollisimman hyvin.

Otetaan käytöön binäärimuuttujista koostuva matriisi R, jonka alkiot ovat $r_{ij} \in \{0, 1\}$, $i, j = 1, \ldots, n$. Jos työntekijä i tekee tehtävän j asetetaan $r_{ij} = 1$ ja muussa tapauksessa $r_{ij} = 0$. Kukin työntekijä tekee vain yhden tehtävän ja kaikki tehtävät on hoidettava. Näin saadaan muodostettua matemaattinen malli sijoittelutehtävälle:

$$\underset{r_{ij}}{\text{minimi}} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} r_{ij}, \quad \text{kun} \begin{cases} \sum_{i=1}^{n} r_{ij} = 1 \text{ kaikilla } j, \\ \sum_{j=1}^{n} r_{ij} = 1 \text{ kaikilla } i. \end{cases} \tag{5.4}$$

Tämä tehtävä on hyvin vaikea ratkaista, kun n on suuri: tehtävällä on kaikkiaan $n! = n(n-1)(n-2) \cdots 2 \cdot 1$ eri ratkaisuvaihtoehtoa.

5.2 Ratkaisualgoritmit

Kokonaislukutehtävät ovat kombinatorisen luonteensa ja epäjatkuvuutensa takia paljon vaikeampia ratkaista kuin LP-tehtävät.

Esimerkki 5.2.1 Tarkastellaan 35 binäärimuuttujaa sisältävää tehtävää. Erilaisia vaihtoehtoja on $2^{35} \approx 34 \times 10^9$. Jos sekunnissa voisi käsitellä 1000 vaihtoehtoa, kestäisi tehtävän läpikäyminen noin 400 vuorokautta. Käytännön algoritmit ovat tietysti suoraviivaista läpikäyntiä tehokkaampia, mutta silti suurten kokonaislukutehtävien ratkaiseminen voi vaatia runsaasti laskenta-aikaa.

Useat kokonaislukutehtävien ratkaisualgoritmit perustuvat tehtävän relaksoimiseen (kokonaislukumuuttujat saavat jatkuvia arvoja) ja ratkaisemiseen tavallisena LP-tehtävänä sekä kokonaislukuratkaisujen generointiin lähtökohtana täten saatu jatkuva-arvoinen ratkaisu.

Kokonaislukutehtävien ratkaisumenetelmät voidaan jakaa leikkaustasomenetelmiin (cutting plane methods) ja luettelointimenetelmiin (enumerative methods). Viime aikoina on tutkittu *branch and bound* -menetelmien (eräs luettelointimenetelmä) ja leikkaustasomenetelmien yhdistämistä melko lupaavin tuloksin. Lisäksi LP-tehtävien sisäpistemenetelmiä on sovellettu branch and bound -menetelmän osatehtävien ratkaisemiseen.

5.2.1 Leikkaustasomenetelmät

Leikkaustasomenetelmissä johdetaan kokonaislukuvaatimusten ja rajoitteiden perusteella uusia epäyhtälöitä, jotka rajaavat alkuperäistä käypää aluetta pienemmäksi säilyttäen kuitenkin kokonaislukuratkaisut. Loppujen lopuksi tuloksena on LP-tehtävä, jonka optimaalisen ratkaisuvektorin muuttujat toteuttavat alkuperäisen tehtävän kokonaislukuehdot.

Leikkaustasomenetelmiä löytyy sekä puhtaille kokonaislukutehtäville että sekalukutehtäville. Tunnetuimpia leikkaustasomenetelmiä ovat Gomoryn menetelmä ja Balasin algoritmi 0/1-tehtäville.

5.2.2 Luettelointimenetelmät

Luettelointimenetelmissä generoidaan kaikki mahdolliset kokonaisluku- tai sekalukutehtävän ratkaisuehdokkaat. Algoritmit koostuvat kirjanpitorutiinista sekä eliminointialgoritmista, joka pyrkii osoittamaan, että tietyt kokonaislukuarvot eivät voi tuottaa nykyistä parempia ratkaisuja.

Branch and bound -algoritmi on tunnetuin luettelointimenetelmä ja myös käytetyin kokonais- ja sekalukutehtävien ratkaisualgoritmi. Yleensä se myös on tehokkain ratkaisumenetelmä.

Branch and bound -algoritmi (Lang ja Doig -versio [Lok77, Sal75]) sekalukutehtäville on esitetty seuraavassa. Tässä merkintä $\lfloor x \rfloor$ merkitsee luvun kokonaisosaa, esimerkiksi $\lfloor 1.6 \rfloor = 1$.

Algoritmi 5.2.1 (Branch and bound)

1: Olkoon z^* yläraja MIP-tehtävän minimiarvolle. Ratkaistaan aluksi tavallisena LP-tehtävänä *relaksoitu tehtävä* eli sallitaan kokonaislukumuuttujille jatkuvat arvot. Mikäli käypää ratkaisua ei löydy, ei ratkaisua löydy MIP-tehtävällekään, eli voidaan lopettaa. Jos taas saatu LP-ratkaisu toteuttaa kokonaislukuehdot, on tehtävä ratkaistu.

2: Olkoon esimerkiksi x_r muuttuja, jonka LP-ratkaisun arvo ${x_r}^*$ ei ole kokonaisluku. Tällöin välissä $\lfloor {x_r}^* \rfloor < x_r < \lfloor {x_r}^* \rfloor + 1$ ei voi olla käypää kokonaislukuratkaisua. Siis kokonaislukuratkaisun täytyy olla *joko* $\leq \lfloor {x_r}^* \rfloor$ *tai* $\geq \lfloor {x_r}^* \rfloor + 1$. Koska kohdefunktio on optimikohdan ympäristössä konveksi, saadaan tarkasteltavaksi kaksi haaraa: $x_r \leq \lfloor {x_r}^* \rfloor$ ja $x_r \geq \lfloor {x_r}^* \rfloor + 1$.

3: Valitaan jatkoon ne osatehtävät, joiden LP-ratkaisu on $< z^*$ eli paras tähänastinen MIP-ratkaisu.

4: Tarkistetaan, löytyykö osatehtävistä uutta ylärajaa MIP-ratkaisuille ja päivitetään tarvittaessa z^*. Kaikki tätä huonommat osaprobleemat voidaan unohtaa.

5: Jos osaratkaisujen lista on tyhjä, on z^* joko löydetty tai MIP-tehtävälle ei ole ratkaisua. Muussa tapauksessa haetaan osaratkaisu, jolla on paras minimiarvon yläraja. Valitaan uusi muuttuja x_r, jonka suhteen haaraudutaan. Jatketaan vaiheesta 2.

Branch and bound -menetelmän työläin osa on jokaisessa haarautumisessa ratkaistava LP-tehtävä. Lisäksi menetelmä on tehokas vain jos osaratkaisujen ylärajat voidaan arvioida mahdollisimman pieniksi algoritmin alkuvaiheissa, jolloin vaihtoehtojen eliminointi on tehokasta.

Branch and bound -algoritmin käyttökelpoisuus ei rajoitu suinkaan lineaarisiin kokonaislukutehtäviin, vaan sitä voi yrittää käyttää myös esimerkiksi globaalissa optimoinnissa. Menetelmä on varsin tehokas, jos kohdefunktion arvoille voidaan laskea hyviä ylä- ja alarajoja.

5.3 Yleisohjeita

Kokonaislukumalleissa kannattaa aina käyttää mahdollisimman vähän kokonaislukumuuttujia ja antaa tiukat rajat muuttujien arvoille. Mahdollisimman hyvä alkuarvaus on tietenkin tärkeää. Kannattaa aina aluksi ratkaista vähintäänkin relaksoitu LP-tehtävä ja käynnistää sekalukuoptimointi saadusta jatkuva-arvoisesta ratkaisusta.

Mikäli kyseessä on jokin erikoistyyppinen kokonaislukutehtävä, kannattaa käyttää kyseiseen tehtävään kehitettyjä erikoisalgoritmeja mikäli mahdollista. Esimerkiksi melko yksinkertaisenkin sijoittelutehtävän ratkaiseminen tavallisena kokonaislukutehtävänä voi olla hyvin raskasta.

Huomautus 5.3.1 Kokonaislukuoptimoinnin tärkeimpänä sääntönä voi pitää: älä yritä ratkaista isoa kokonaislukutehtävää, ennen kuin olet varma, että tehtävä on muodostettu järkevästi.

Kannattaa siis aina tarkistaa, voiko tehtävää yksinkertaistaa esimerkiksi eliminoimalla turhia rajoitteita tai muuttujia. Muutamankin rajoitteen tai muuttujan poistaminen voi tuottaa useita kertaluokkia helpommin ratkeavan tehtävän. Toisinaan voi isokin MIP-tehtävä (kymmeniä kokonaislukumuuttujia) ratketa varsin helposti, joskus taas yksinkertaiselta näyttävä tehtävä voi vaatia paljon laskentaresursseja.

5.4 Ohjelmistoja

Kokonaislukutehtäviä voi esittää GAMS-mallinnuskielellä, joka puolestaan käyttää sopivaa ratkaisuohjelmistoa mallien ratkaisemiseen.

Esimerkki 5.4.1 Esimerkin 5.1.1 tuotantomallia vastaava GAMS-malli löytyy listauksesta 5.1. GAMS-mallin ratkaisemiseen tarvittaan muutamia ratkaisuohjelmiston branch and bound -iteraatioita. Relaksoidun MIP-mallin maksimiarvo oli 205.5 ja MIP-mallin maksimiarvo 202.

GAMSin ja Lampsin lisäksi kokonaislukumalleja voi yrittää ratkaista mm. NAG-aliohjelmakirjaston rutiinilla `h02bbf` (taulukko B.5 sivulla 211). SAS-ohjelmiston SAS/OR-osuuden LP-proseduuri osaa ratkaista myös kokonaislukutehtäviä.

Jos käytettävissä on ratkaistavaan tehtävään soveltuva erikoisohjelmisto, kannattaa sitä käyttää. TOMS-algoritmeista löytyy menetelmiä kokonaislu-

Listaus 5.1: *GAMS-esimerkki tuotantomallista, jossa on kiinteät tuotannon aloituskustannukset sekä tuotantomäärään verrannolliset kustannukset. Kokonaistuotannolle ja eri tuotteiden tuotannolle on asetettu maksimimäärät. Binäärimuuttuja `Bintuot(i)` määrää, tuotetaanko tuotetta `i` vai ei. Asettamalla `OPTCR = 0.0` päästään sekalukutehtävän tarkkaan optimiarvoon.*

```
SET i Tuotteet / T1 * T4 /;

SCALAR Maxtuot Maksimi kokonaistuotannolle / 450 /;
PARAMETERS Ylarajat(i) / T1 250, T2 170, T3 150, T4 300 /
           Hinnat(i)   / T1 4.5, T2 2.6, T3 4.0, T4 1.5 /
           Kust(i)     / T1 4.0, T2 2.0, T3 4.0, T4 1.0 /
           Aloitus(i)  / T1 10.0, T2 25.0, T3 5.0, T4 15.0 /;

POSITIVE VARIABLE Tuotanto(i) Tuotantomaara;
BINARY VARIABLE Bintuot(i)    Binaarimuuttuja tuotannolle;
FREE VARIABLE Tuotto          Voiton kokonaissumma;

EQUATIONS Eq1(i), Eq2, Obj;

Eq1(i).. Tuotanto(i) =L= Ylarajat(i)*Bintuot(i);
Eq2.. SUM(i, Tuotanto(i)) =L= Maxtuot;
Obj.. Tuotto =E= SUM(i, (Hinnat(i) - Kust(i))*Tuotanto(i)) -
                 SUM(i, Aloitus(i)*Bintuot(i));

MODEL Tmalli / ALL /;
OPTION OPTCR = 0.0;
SOLVE Tmalli USING MIP MAXIMIZING Tuotto;
```

kutehtäviin (taulukko B.6 sivulla 214). Esimerkiksi *sijoittelutehtävien* (job assignment problem) ratkaisuohjelmistoja löytyy TOMS-algoritmeista.

5.5 Lisätietoja

Lisätietoja kokonaislukutehtävien ratkaisemisesta löytyy mm. teoksista *Operations Research* [Tah87], *Combinatorial Optimization* [PS82], *Integer Programming* [Sal75] ja *Integer programming: theory, applications, and computations* [Tah75] sekä esimerkiksi sarjajulkaisuista *Operations Research* ja *Mathematical Programming*. Eri menetelmien yhdistämistä on kuvailtu artikkeleissa [PR91, Bor92]. Lisätietoja ohjelmistoista löytyy liitteestä B (sivu 206).

6 Rajoitteettomat epälineaariset optimointitehtävät

Tässä luvussa esitetään yleiskatsaus jatkuvasti differentioituvien epälineaaristen optimointitehtävien ratkaisumenetelmiin ja -ohjelmistoihin. Lisäksi esitellään tarkemmin joitakin tehtävätyyppejä ja niiden ratkaisemista.

6.1 Epälineaariset tehtävät

Luvuissa 6, 7 ja 8 käsitellään epälineaarisia optimointitehtäviä

$$\operatorname*{minimi}_{\mathbf{x}} f(\mathbf{x}), \text{ kun } \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \text{ ja } \mathbf{h}(\mathbf{x}) = \mathbf{0}. \tag{6.1}$$

Tehtävän kohdefunktio $f(\mathbf{x})$ tai rajoitefunktiot $\mathbf{g}(\mathbf{x}) : \mathbb{R}^n \mapsto \mathbb{R}^p$ ja $\mathbf{h}(\mathbf{x}) : \mathbb{R}^n \mapsto \mathbb{R}^q$ voivat olla epälineaarisia.

Epälineaariselle funktiolle ei päde $f(\mathbf{x}+\mathbf{y}) = f(\mathbf{x}) + f(\mathbf{y})$ kaikilla $\mathbf{x}, \mathbf{y}$ tai $f(\alpha\mathbf{x}) = \alpha f(\mathbf{x})$ kaikilla $\mathbf{x}$. Esimerkiksi $2x_1^2 + x_2^2 - 1 \leq 0$ on epälineaarinen epäyhtälörajoite.

Pienimmän neliösumman minimointitehtäviä käsitellään luvussa 7 ja rajoitteellisia optimointitehtäviä luvussa 8. Yksiulotteista optimointia käsitellään luvussa 13, jossa esitellään mm. viivahakumenetelmiä moniulotteiseen optimointiin. Lisätietoja löytyy tämän luvun lopusta löytyvän kirjallisuusluettelon teoksista.

Epälineaarisuus tekee optimointitehtävän ratkaisemisen vaikeaksi varsinkin, kun ratkaisuavaruuden dimensio kasvaa. Epälineaarisella optimointitehtävällä voi olla seuraavia ominaisuuksia:

- Tehtävällä ei ole yksikäsitteistä optimiratkaisua $\mathbf{x}^*$, vaan joukko paikallisia ääriarvokohtia.
- Eri muuttujilla on eri skaalat. Esimerkiksi muuttuja x_1 kuuluu välille $[0, 10^{-10}]$ ja muuttuja x_2 kuuluu välille $[1, 10^6]$.

- Funktio $f(\mathbf{x})$ on raskas laskea: funktion arvo voi olla peräisin esimerkiksi prosessisimulaatiosta tai differentiaaliyhtälösysteemistä.
- Funktion $f(\mathbf{x})$ derivaatat eivät ole saatavilla: niitä ei esimerkiksi voida johtaa analyyttisesti tai $f(\mathbf{x})$ on ”musta laatikko”, jonka rakennetta ei tunneta.
- Hyvän alkuarvauksen löytäminen tehtävän ratkaisemiseksi on vaikeaa.

Esimerkki 6.1.1 Minimointitehtävä

$$\underset{\mathbf{x}}{\text{minimi}} \sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{n} x_i \ln \frac{x_i}{\sum_{k=1}^{n} x_k}, \text{ kun } \mathbf{x} \geq \mathbf{0} \text{ ja } A\mathbf{x} = \mathbf{b}$$

on tyyppiesimerkki vaikeasta epälineaarisesta optimointitehtävästä. Tälläisiä tehtäviä esiintyy esimerkiksi kemian reaktioyhtälöiden yhteydessä. Logaritmi- tai eksponenttifunktio aiheuttaa usein skaalausongelmia: pieni ero muuttujien arvoissa voi aiheuttaa suuria muutoksia kohdefunktion käyttäytymisessä. Tehtävän ratkaisemista vaikeuttavat myös (tässä tapauksessa lineaariset) rajoitteet.

Yleismenetelmää epälineaaristen optimointitehtävien ratkaisemiseen ei ole olemassa. Mahdollisia algoritmeja ja ohjelmistoja on monia ja kannattaa kokeilla eri menetelmiä parhaan löytämiseksi. Ratkaistavan ongelman erikoisominaisuuksia tuleekin pyrkiä hyödyntämään.

6.2 Rajoitteettomien tehtävien ratkaisu

Rajoitteettomille optimointitehtäville (unconstrained optimization)

$$\underset{\mathbf{x}}{\text{minimi}}\ f(\mathbf{x}),\ \text{kun } \mathbf{x} \in \mathbb{R}^n, \tag{6.2}$$

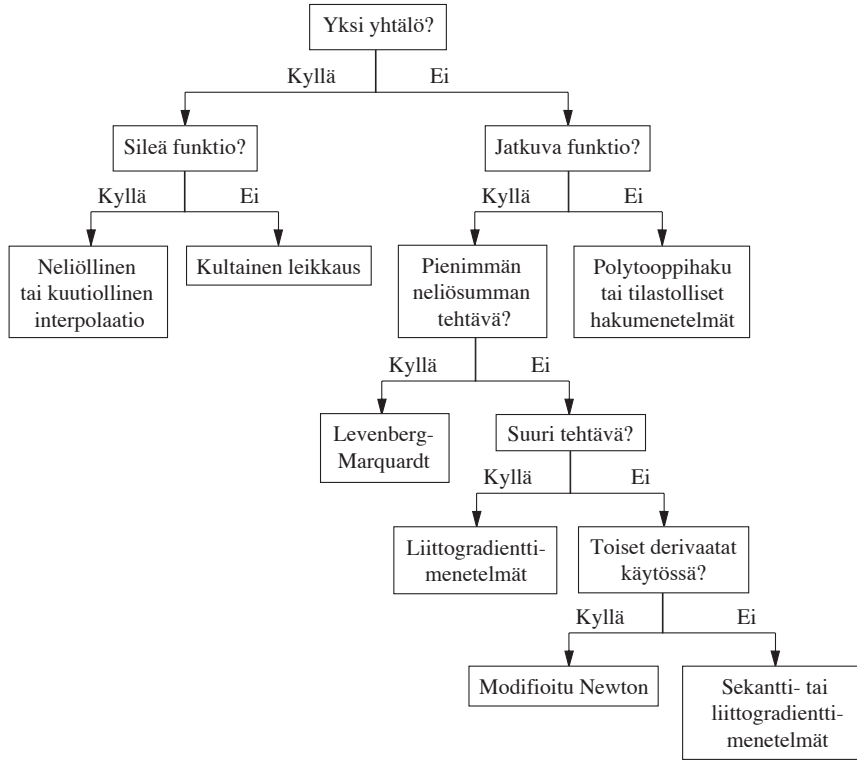
löytyy tehokkaita ratkaisumenetelmiä, kun minimoitava kohdefunktio f on jatkuvasti differentioituva. Eräille algoritmeille voidaan jopa osoittaa globaali suppeneminen, kun kohdefunktio f toteuttaa sopivat ehdot. Toisaalta luotettavin menetelmä ei välttämättä ole tehokkain.

Suositeltavia moniulotteisten rajoitteettomien tehtävien ratkaisualgoritmeja ovat (kuva 6.1):

- *Newtonin menetelmä* sisältyy mm. IMSL- ja NAG-aliohjelmakirjastoihin. Menetelmä on tehokas, mikäli derivaatat on helppo laskea. Toisaalta menetelmä vaatii esimerkiksi luottamusalueen tai viivahaun käytön ollakseen stabiili.
- *Sekanttimenetelmä BFGS-päivityksellä* löytyy mm. IMSL- ja NAG-kirjastoista. BFGS on luotettava ja stabiili menetelmä, mutta isoissa tehtävissä liittogradienttimenetelmät tarvitsevat vähemmän muistia.

- *Liittogradienttialgoritmeista* tunnetuimmat ovat Fletcherin ja Reevesin menetelmä sekä Polakin ja Ribièren menetelmä. Ohjelmistot CONMIN [SP80] ja BBVSCG [BL85] ovat stabiileja liittogradienttimenetelmien yleistyksiä. Liittogradienttimenetelmät ovat tehokkaita, kun muuttujia on yli 250 eikä kohdefunktion Hessen matriisi $H(\mathbf{x}) = \nabla\nabla^T f(\mathbf{x})$ ole harva, jolloin harvuutta hyödyntävät menetelmät voivat olla tehokkaita.
- Jokin suorahakumenetelmä, esimerkiksi *Nelderin ja Meadin polytooppihakumenetelmä*, joka soveltuu myös ei-differentioituville funktioille. Menetelmä löytyy mm. IMSL- ja NAG-aliohjelmakirjastoista sekä Matlabin Optimization Toolboxista.

Näitä menetelmiä esitellään myöhemmin tässä luvussa. Ohjelmistoja on esitelty luvun lopussa sivulla 124 sekä liitteessä B.



Kuva 6.1: *Rajoitteettomien optimointitehtävien ratkaisumenetelmien soveltuvuus erityyppisiin tehtäviin esitettynä yksinkertaisena kaaviona.*

6.3 Ratkaisumenetelmien yleispiirteitä

Rajoitteettomaan optimointiin kehitetyistä algoritmeista osa käyttää vain funktion arvoja (esimerkiksi polytooppihaku), osa taas vaatii gradienttivek-

torin (liittogradienttimenetelmä) ja mahdollisesti myös Hessen matriisin laskemisen tai arvioimisen (Newtonin menetelmän tyyppiset algoritmit).

Newton-tyyppiset menetelmät soveltuvat sileiden rajoitteettomien optimointitehtävien ratkaisemiseen, jos kohdefunktion $f(\mathbf{x})$ gradienttivektori $\nabla f(\mathbf{x})$ ja Hessen matriisi $H(\mathbf{x}) = \nabla\nabla^T f(\mathbf{x})$ voidaan laskea tai arvioida tehokkaasti. Seuraavassa on esitetty algoritmin perusrakenne:

Algoritmi 6.3.1 (Newton-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 $\|\mathbf{x}^k - \mathbf{x}^{k-1}\| < \epsilon$

Hakusuunta $\mathbf{p}^k$ saadaan liittogradienttimenetelmissä kaavasta

$$\mathbf{p}^k = -\nabla f(\mathbf{x}^k) + \beta_k\mathbf{p}^{k-1} \tag{6.3}$$

ja sekanttimenetelmissä kaavasta

$$B_k\,\mathbf{p}^k = -\nabla f(\mathbf{x}^k)\,, \tag{6.4}$$

missä B_k on Hessen matriisin arvio. Sekanttimenetelmissä matriisia B_k päivitetään kullakin iteraatioaskeleella. Jos matriisi B_k asetetaan identiteettimatriisi, saadaan jyrkimmän laskun menetelmä (steepest descent), ja jos matriisiksi B_k asetetaan $\nabla\nabla^T f(\mathbf{x}^k)$ eli Hessen matriisi, saadaan Newtonin menetelmä.

Newtonin menetelmässä hakuaskeleen pituus λ_k valitaan usein käyttäen *viivahakua* (line search). Vaihtoehtoisesti voidaan käyttää *luottamusalueeseen* perustuvia menetelmiä (trust-region methods). Viivahakua ja luottamusaluetta kuvataan luvussa 13 sivulta 193 alkaen.

Viivahaun tai luottamusalueen käyttö on tärkeää mm. indefiniittien tai lähes singulaaristen Hessen matriisien vuoksi. Pisteessä, jossa Hessen matriisi on lähes singulaarinen, on Newtonin menetelmän yhtälöryhmästä saatu hakuaskel usein huono valinta. Myös liittogradienttimenetelmä ja sekanttimenetelmät perustuvat viivahaun käyttöön.

Epälineaaristen optimointitehtävien viivahakuun perustuvissa ratkaisualgoritmeissa valitaan aluksi käypä hakusuunta $\mathbf{p}^k \in \mathbb{R}^n$ ja tämän jälkeen kuljetaan kyseiseen suuntaan kunnes löydetään minimikohta. Askelpituudeksi siis valitaan

$$\lambda_k = \arg\underset{\lambda>0}{\text{minimi}}\, f(\mathbf{x}^k + \lambda\mathbf{p}^k) \tag{6.5}$$

siten, että $\mathbf{x}^k + \lambda_k\mathbf{p}^k$ on käyvällä alueella $\mathcal{F}$. Rajoitteettomassa optimoinnissa käypä alue on $\mathcal{F} = \mathbb{R}^n$. Kyseessä on siten yksiulotteinen parametrin λ minimointitehtävä avaruuden $\mathbb{R}^n$ puolisuoralla $\mathbf{x}^k + \lambda\mathbf{p}^k, \lambda > 0$.

Luvussa 13 on esitetty *takautuvan viivahaun* periaate ja eräs menetelmän toteutus sekä sen sovelluksia.

Luottamusalueeseen perustuvissa menetelmissä rajoitetaan hakuaskeleen pituutta ja tarvittaessa kasvatetaan tai pienennetään luottamusalueen kokoa. Kyseessä on viivahaun käytölle vaihtoehtoinen tapa selvitä esimerkiksi Hessen matriisin indefiniittisyydestä ja varmistaa algoritmin suppeneminen.

Luottamusalueen ideana on havainto, että Newtonin menetelmässä käytetty funktion kvadraattinen malli on käyttökelpoinen vain nykyisen hakupisteen $\mathbf{x}^k$ lähellä. Jos Hessen matriisi on indefiniitti (myös negatiivisia ominaisarvoja), ei kvadraattisella mallilla ole alarajaa. Täten kvadraattinen approksimaatio on huono, kun askel $\mathbf{s}^k = \lambda_k \mathbf{p}^k$ on suuri.

6.3.1 Suppenemiskriteerejä

Optimointitehtävien ratkaisualgoritmeissa tarvitaan ehto iteroinnin lopettamiselle. Edellä algoritmissa 6.3.1 ehdoksi asetettiin $\|\mathbf{x}^{k+1} - \mathbf{x}^k\| \leq \epsilon$, missä parametri ϵ on positiivinen skalaari. Tämä ehto ei sellaisenaan ole aina riittävä eikä sovellu kaikkiin tilanteisiin. Esimerkkinä tästä on tilanne, jossa $\|\mathbf{x}^*\|$ on hyvin suuri.

Jos käytetään kaksoistarkkuuden aritmetiikkaa, ovat seuraavassa esitetyt suppenemisehdot varsin käyttökelpoisia:

$$\begin{cases} \|\nabla f(\mathbf{x}^{k+1})\| \leq 10^{-6}, \\ \dfrac{\|\mathbf{x}^{k+1} - \mathbf{x}^k\|}{\max\{1,\ \|\mathbf{x}^{k+1}\|\}} \leq 10^{-16}, \\ \dfrac{\|f(\mathbf{x}^{k+1}) - f(\mathbf{x}^k)\|}{\max\{1,\ \|f(\mathbf{x}^{k+1})\|\}} \leq 10^{-16}. \end{cases} \tag{6.6}$$

Suppenemiskriteerien tarkkuus täytyy tietenkin valita ratkaisulta vaadittavan tarkkuuden perusteella. Edellä annetut numeroarvot ovat vain suuntaa antavia.

6.3.2 Menetelmien luotettavuus

Tarkastellaan aluksi pienten ja keskisuurten tehtävien (muuttujien lukumäärä n on alle 250) ratkaisemista. Luotettavuudella tarkoitetaan tässä paikallisen optimin löytymisen todennäköisyyttä, mihin vaikuttaa suppenemisnopeus ja -varmuus. Rajoitteettomien tehtävien ratkaisualgoritmit voidaan järjestää luotettavuuden perusteella esimerkiksi seuraavasti:

- toisia derivaattoja käyttävä modifioitu Newtonin menetelmä
- modifioitu Newtonin menetelmä ja differenssiarvio Hessen matriisille
- gradienttivektoria käyttävä sekanttimenetelmä
- sekanttimenetelmä ja gradienttivektorin differenssiarvio
- gradienttivektoria käyttävä liittogradienttimenetelmä

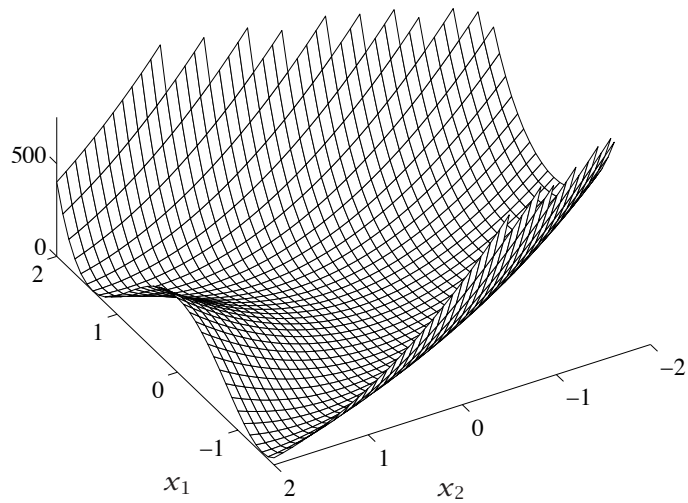
- liittogradienttimenetelmä ja differenssiarvio gradienttivektorille
- suorahakumenetelmät, esimerkiksi polytooppihaku.

Menetelmä tulisi valita aina mahdollisimman läheltä listan alkupäätä. Jos derivaatat eivät ole saatavilla, joudutaan tietenkin valitsemaan epäluotettavampi menetelmä. Menetelmän toteutus vaikuttaa suuresti ratkaisemisen luotettavuuteen. Tässä on oletettu, että käytetään valmisohjelmistoista löytyviä toteutuksia, jotka on perusteellisesti testattu.

Suurten tehtävien ratkaisuun vaikuttaa erittäin paljon tehtävän tyyppi, esimerkiksi Hessen matriisin harvuus. Newton-tyyppiset menetelmät ovat luotettavia ja tehokkaita myös suurissa tehtävissä, esimerkkinä katkaistu Newtonin menetelmä. Muistinkäytön vuoksi voidaan joutua valitsemaan rajoitetun muistin sekanttimenetelmä tai liittogradienttimenetelmä.

6.4 Ratkaisuohjelmistot ja esimerkkitehtävät

Tässä oppaassa käytetty mallinnuskieli GAMS käyttää mm. Minos-ohjelmistoa mallien ratkaisemiseen. Eräistä keskeisistä algoritmeista annetaan suppeat Mathematica- tai Fortran-toteutukset sanallisten kuvausten lisäksi. Mathematicaa käytetään kuvauskielenä tiiviytensä ja monipuolisuutensa vuoksi. Todellisissa tehtävissä kannattaa toki käyttää käännettäviä kieliä ja mieluiten tietenkin valmisohjelmistojen testattuja algoritmeja tai esimerkiksi Netlibistä löytyviä paljon käytettyjä koodeja.



Kuva 6.2: *Kaksiulotteinen Rosenbrockin funktio* $f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$. *Globaali minimipiste on* $(1,1)^T$. *Kaareutuva kapea laakso aiheuttaa ongelmia useimmille ratkaisumenetelmille.*

Epälineaaristen optimointitehtävien ratkaisumenetelmien ja -ohjelmistojen käyttöä havainnollistetaan kuvassa 6.2 esitetyn Rosenbrockin funktion avulla. Rosenbrockin funktion minimi haetaan tässä luvussa aluksi GAMSin avulla. Lisäksi ratkaisemista havainnollistetaan perusalgoritmien Mathematica-toteutuksilla. Suurten ja vaikeiden optimointitehtävien ratkaisemiseen on syytä käyttää luotettavia valmisohjelmistoja, kuten yleiskäyttöisiä aliohjelmakirjastoja IMSL ja NAG sekä lineaarialgebran aliohjelmakirjastoa Lapack. Luvussa myös vertaillaan suositeltavia suurten optimointitehtävien ratkaisuohjelmistoja.

6.4.1 GAMS-ohjelmiston käyttö

GAMS-mallinnuskieltä voi käyttää epälineaaristen optimointitehtävien määrittelemiseen. Ratkaisemiseen soveltuu vaikkapa Minos-ohjelmistomoduuli. GAMS laskee tarvittaessa kohdefunktion ja rajoitefunktioiden derivaatat. GAMS on tehokas standardityyppiä olevissa tehtävissä, joissa kohdefunktio ja rajoitteet ovat sileitä ja suurin osa rajoitteista on lineaarisia.

Esimerkki 6.4.1 Tarkastellaan kuvassa 6.2 esitetyn Rosenbrockin funktion

$$f(\mathbf{x}) = 100\left(x_2 - x_1^2\right)^2 + (1 - x_1)^2 \tag{6.7}$$

minimointia. Funktion Hessen matriisi on

$$H(\mathbf{x}) = \begin{pmatrix} 1200x_1^2 - 400x_2 + 2 & -400x_1 \\ -400x_1 & 200 \end{pmatrix} \tag{6.8}$$

ja gradienttivektori

$$\nabla f(\mathbf{x}) = \begin{pmatrix} -400x_1(x_2 - x_1^2) + 2(x_1 - 1) \\ 200(x_2 - x_1^2) \end{pmatrix}. \tag{6.9}$$

Alkuarvaus olkoon $(-1.2, 1.0)^T$.

Minimointitehtävä on esitetty GAMS-mallina listauksessa 6.1. Ratkaisemalla malli GAMSilla saadaan tulokseksi piste $\mathbf{x}^* = (1, 1)^T$, joka on minimikohta: funktion gradientiksi saadaan $\nabla f(\mathbf{x}^*) = (0, 0)^T$ eli kyseessä on kriittinen piste. Hessen matriisiksi tässä pisteessä saadaan

$$H(\mathbf{x}^*) = \begin{pmatrix} 802 & -400 \\ -400 & 200 \end{pmatrix}.$$

Hessen matriisi on positiivisesti definiitti, koska ominaisarvoiksi saadaan

$$\text{eig}(H(\mathbf{x}^*)) \approx \{0.3994, 1002\} > 0.$$

Täten kyseessä on vahva lokaali minimi. Toisaalta koska $f(\mathbf{x}) \geq 0 = f(\mathbf{x}^*)$ kaikilla $\mathbf{x}$, on kyseessä myös globaali minimi.

Listaus 6.1: *Rosenbrockin funktion minimointi GAMS-mallinnuskielellä. Funktio SQR korottaa argumenttinsa neliöön. Notaatiolla `x1.l = ...` voidaan asettaa muuttujalle `x1` alkuarvo. Optimoitavan kohdemuuttujan (tässä tapauksessa `f`x) täytyy olla vapaa muuttuja. Yhtäsuuruutta merkitään `=E=`.*

```
FREE VARIABLES f, x1, x2;
EQUATION Func;
Func.. f =E= 100*SQR(x2 - SQR(x1)) + SQR(1 - x1);
x1.l = -1.2; x2.l = 1.0;
MODEL Rosenbr / ALL /;
SOLVE Rosenbr MINIMIZING f USING NLP;
```

6.5 Newtonin menetelmä

Newtonin iteraatio epälineaarisen minimointitehtävän ratkaisemiseksi toimii seuraavasti:

$$H(\mathbf{x}^k)\,\mathbf{p}^k = -\nabla f(\mathbf{x}^k), \tag{6.10}$$

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{p}^k. \tag{6.11}$$

Newtonin menetelmässä täytyy ratkaista lineaarinen yhtälöryhmä (6.10) jokaisella iteraatioaskeleella. Koska Hessen matriisi $H(\mathbf{x}^k) = \nabla\nabla^T f(\mathbf{x}^k)$ on symmetrinen ja monissa käytännön tehtävissä myös harva, voidaan ratkaisussa käyttää erikoismenetelmiä (katso kappaletta 1.4 sivulla 23). Iteraatio voidaan lopettaa esimerkiksi silloin, kun askelpituus $\|\mathbf{p}^k\|$ 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* (kvadraattista) eli

$$\|\mathbf{x}^{k+1} - \mathbf{x}^*\| \le \beta \|\mathbf{x}^k - \mathbf{x}^*\|^2, \quad \beta \ge 0\,. \tag{6.12}$$

Täten Newtonin menetelmän suppenemisnopeus on erittäin hyvä oltaessa lähellä optimointitehtävän minimipistettä $\mathbf{x}^*$.

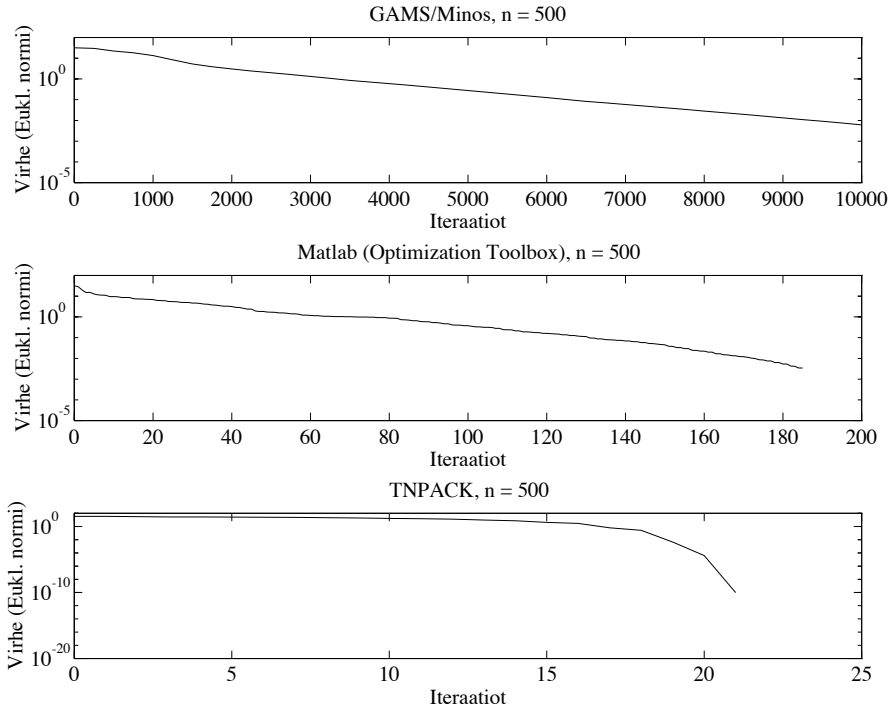
Esimerkki 6.5.1 Eräiden menetelmien suppenemista n-ulotteisen Rosenbrockin funktion minimoinnissa on havainnollistettu kuvassa 6.3. Olkoon $\eta = n/2$, jolloin kohdefunktio määritellään seuraavasti:

$$f(\mathbf{x}) = \sum_{i=1}^{\eta} 100\left(x_{2i} - x_{2i-1}^2\right)^2 + (1 - x_{2i-1})^2\,.$$

Muuttujien lukumäärä n on 500. Kuvaajiin on piirretty ratkaisun virhe iteraatiokertojen funktiona. Virheellä tarkoitetaan ratkaisupisteen etäisyyttä tarkasta ratkaisusta euklidisen normin mielessä.

Huomaa, että asteikot ovat erilaisia eri kuvaajissa. Koska liittogradienttimenetelmän iteraatiossa tehdään paljon vähemmän työtä kuin esimerkiksi Newtonin menetelmän iteraatiossa, on todellisen työmäärän vertaaminen vaikeaa.

Ylimmässä kuvaajassa esitetään GAMS/Minos-ohjelmiston suppeneminen. Kuvaajasta havaitaan, että suppeneminen on lineaarista ja erittäin hidasta. Keskimmäisessä kuvaajassa on esitetty Matlabin Optimization Toolboxista löytyvän BFGS-menetelmää käyttävän rutiinin `fminu` superlineaarinen suppeneminen. Alimmassa kuvaajassa on esitetty TNPACK-ohjelmiston käyttäytyminen samassa tehtävässä. Katkaistuun Newtonin menetelmään perustuvan ohjelmiston suppeneminen on likimain kvadraattista lähestyttäessä minimiä.



Kuva 6.3: *Eräiden menetelmien ja ohjelmistojen käyttäytyminen minimoitaessa Rosenbrockin funktiota, kun muuttujien lukumäärä on* $n = 500$.

Huomautus 6.5.1 Newtonin menetelmä suppenee neliöllisesti minimikohdan lähellä, mutta menetelmä on herkkä Hessen matriisin singulariteeteille, joten suppeneminen tulee varmistaa. Tällöin käytetään joko viivahakua tai luottamusalueeseen perustuvia menetelmiä.

Esimerkki 6.5.2 Tarkastellaan esimerkistä 6.4.1 tuttua Rosenbrockin funktion minimointia. Olkoon alkuarvaus $\mathbf{x}^1 = (-1.2, 1.0)^T$. Listauksessa 6.2 on esitetty Newton-iteraation toteutus Mathematica-ohjelmistolla. Lopetusehtona on $\|\mathbf{p}^k\| < 10^{-8}$. Koodi ei tarkista Hessen matriisin mahdollista singulaarisuutta. Rutiini `LinearSolve` ei hyödynnä Hessen matriisin symmetrisyyttä lineaarisen yhtälöryhmän ratkaisemisessa.

Iteraatiopisteiksi $\mathbf{x}^k$ ja funktion arvoiksi $f(\mathbf{x}^k)$ saadaan

k	$\mathbf{x}^k$	$f(\mathbf{x}^k)$
1	$(-1.2, 1)$	24.2
2	$(-1.175, 1.381)$	4.732
3	$(0.7631, -3.175)$	1412.
4	$(0.7634, 0.5828)$	0.05597
5	$(1, 0.944)$	0.3132
6	$(1, 1)$	$1.853 \cdot 10^{-11}$
7	$(1, 1)$	$3.433 \cdot 10^{-20}$

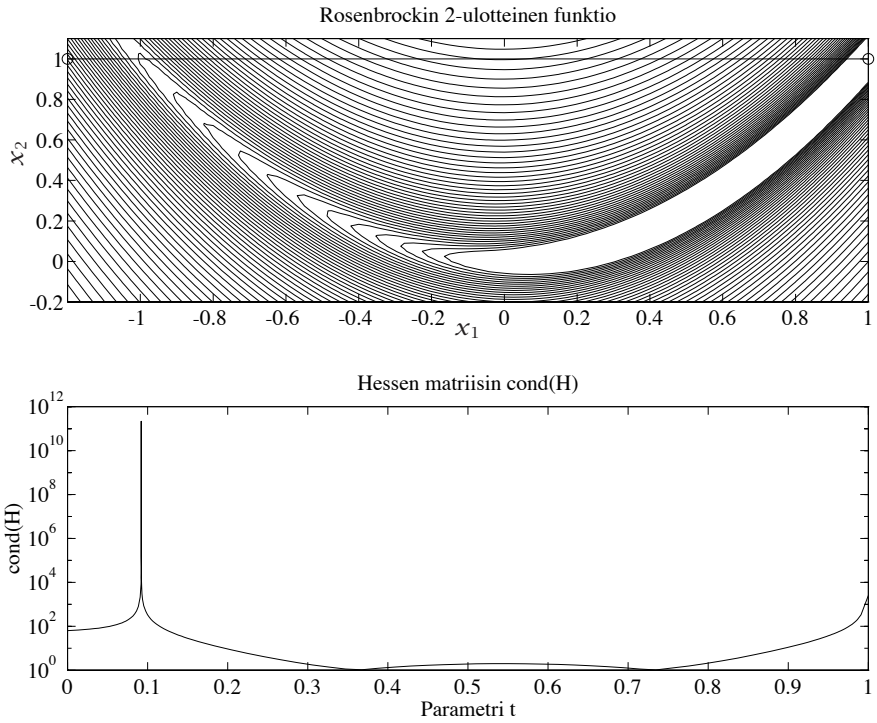
Kohdassa $k = 3$ jouduttiin melko kauas minimipisteestä. Minimiarvon löytämiseen tarkkuudella 10^{-16} tarvittiin 7 Newton-askelta. Kuten tästä esimerkistä nähdään, listauksessa 6.2 esitetty Newtonin menetelmä törmää helposti Hessen matriisin singulariteetteihin.

Listaus 6.2: *Rosenbrockin funktion minimin hakeminen Newtonin menetelmällä. Esimerkissä käytetään luvussa 12 esiteltäviä Mathematica-rutiineja* `Grad` *ja* `Hesse` *derivaattojen laskemiseen. Rutiini* `NewtonStep` *tekee yhden Newtonin menetelmän iteraation.*

```
<<Derivaatat.m

(* Funktion arvo, gradientti ja Hessen matriisi *)
f[x1_,x2_] = 100 (x2 - x1^2)^2 + (1 - x1)^2;
gf[x1_,x2_] = Simplify[Grad[f[x1,x2],{x1,x2}]];
Hf[x1_,x2_] = Simplify[Hesse[f[x1,x2],{x1,x2}]];

(* Newton-askeleen laskeva rutiini *)
NewtonStep[f_, gf_, Hf_, x0_List] := Block[{p, newx},
  p = - LinearSolve[Hf @@ x0, gf @@ x0]; newx = x0 + p]

X0 = N[{-12/10,1},16]; (* Alkuarvaus *)

(* Iteroidaan, kunnes askel on kyllin pieni *)
NewtonIterates = FixedPointList[NewtonStep[f, gf, Hf, #]&, X0,
  SameTest -> (Sqrt[Apply[Plus,(#1 - #2)^2]] < 10.^-8 &)]

(* Lasketaan funktion arvot iteraatiopisteissa *)
functionValues = Map[Apply[f,#]&, NewtonIterates]
```

Esimerkki 6.5.3 Kuvassa 6.4 on havainnollistettu Hessen matriisin käyttäytymistä Rosenbrockin funktion tapauksessa. Ylempään kuvaan on piirretty kohdefunktion tasa-arvokäyriä. Alemmassa kuvassa on esitetty Hessen matriisin häiriöalttius (condition number) liikuttaessa janalla $\mathbf{x}(t) = (1-t)\,(-1.2, 1)^T + t\,(1, 1)^T, t \in [0, 1]$.

Jotta Newtonin menetelmä olisi käyttökelpoinen, tarvitaan keino selviytyä Hessen matriisin mahdollisesta singulaarisuudesta. *Modifioidussa Newtonin*



Kuva 6.4: *Kaksiulotteisen Rosenbrockin funktion Hessen matriisin häiriöalttius janalla* $\mathbf{x}(t) = (1-t)\,(-1.2, 1)^T + t\,(1,1)^T, t \in [0,1]$.

menetelmässä pyritään pitämään Hessen matriisi positiivisesti definiittisenä. Suppenemista voidaan myös yrittää varmistaa luottamusalueen tai viivahaun avulla (luku 13).

Positiivisdefiniittisyyden voi tarkistaa esimerkiksi laskemalla Hessen matriisille hajotelman $L_k D_k L_k^T = H(\mathbf{x}^k)$, missä D_k on lävistäjämatriisi. Yhtälöryhmän $H(\mathbf{x}^k)\mathbf{p}^k = -\nabla f(\mathbf{x}^k)$ ratkaiseminen tehdään vaiheittain:

$$L_k \mathbf{t}^k = -\nabla f(\mathbf{x}^k), \tag{6.13}$$

$$D_k L_k^T \mathbf{p}^k = \mathbf{t}^k. \tag{6.14}$$

Jos Hessen matriisi on singulaarinen, on jokin lävistäjäalkio $d_{ii} = 0$, jolloin voidaan kokeilla lävistäjämatriisin R_k, $r_{ii} \geq 0$, lisäystä LDL^T-hajotelmaan ennen yhtälöryhmän ratkaisua:

$$\widetilde{L}_k \widetilde{D}_k \widetilde{L}_k^T = H(\mathbf{x}^k) + R_k. \tag{6.15}$$

Tässä siis lisättiin tarpeen mukaan positiivisia lukuja Hessen matriisin lävistäjälle.

Esimerkki 6.5.4 Omien Fortran-koodien rakennuspalikkoina voi käyttää esimerkiksi Lapack-aliohjelmakirjastoa. Listauksessa 6.3 on käytetty Lapackia symmetrisen yhtälöryhmän ratkaisemiseen.

Newtonin menetelmää käyttävä ohjelma on kirjoitettu Fortran 90 -ohjelmointikielellä. Hessen matriisista lasketaan vain ylempi puolisko, sillä Lapackin aliohjelma `ssysv` hyödyntää matriisin symmetrisyyttä yhtälöryhmän ratkaisemisessa. Aliohjelma `ssysv` laskee Hessen matriisille LDL^T-hajotelman ja tarkistaa mahdollisen singulaarisuuden.

Lapackin käyttöön tarvittavat käännöskomennot ja -valitsimet vaihtelevat koneittain. Lapack sisältyy CSC:n Cray-superkoneen LibSci-aliohjelmakirjastoon. Crayllä ajettu ohjelma tuottaa neljällä desimaalilla saman tuloksen kuin listauksen 6.2 Mathematica-koodi.

Newtonin menetelmän yhtälöryhmän tarkka ratkaiseminen vaatii paljon resursseja (keskusmuistia ja laskenta-aikaa). Usein laskennan alkuvaiheissa kannattaakin ratkaista yhtälö likimääräisesti ja siirtyä tarkkaan ratkaisemiseen vasta lähestyttäessä optimipistettä. *Katkaistussa Newtonin menetelmässä* (truncated Newton) 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):

$$\tilde{r}_k = \frac{\|\mathbf{r}^k\|}{\|\nabla f(\mathbf{x}^k)\|} \leq \eta_k < 1. \tag{6.16}$$

6.6 Sekanttimenetelmät

Sekanttimenetelmissä (secant methods, myös quasi-Newton methods eli kvasi-Newton -menetelmät) arvioidaan Newtonin menetelmässä tarvittavaa Hessen matriisia (tai sen käänteismatriisia) iteratiivisten päivitysten avulla ja pyritään pitämään iteraatiomatriisi B_k positiivisesti definiittinä. Päivitykset tehdään menetelmästä riippuvalla matriisilla Q_k:

$$B_{k+1} = B_k + Q_k\,. \tag{6.17}$$

Kullakin iteraatioaskeleella ratkaistaan hakusuunta $\mathbf{p}^k$ yhtälöryhmästä

$$B_k\,\mathbf{p}^k = -\nabla f(\mathbf{x}^k), \tag{6.18}$$

jota voi verrata Newtonin menetelmän yhtälöryhmään $H(\mathbf{x}^k)\mathbf{p}^k = -\nabla f(\mathbf{x}^k)$. Sekanttimenetelmien perusongelmana on: miten löydetään matriisi B_{k+1}, kun tiedetään B_k?

Oletetaan, että optimoitava funktio f on kahdesti jatkuvasti differentioituva. Integraalilaskennan väliarvolauseen perusteella saadaan

$$\nabla f(\mathbf{x}^{k+1}) = \nabla f(\mathbf{x}^k) + \left(\int_0^1 \nabla\nabla^T f(\mathbf{x}^k + t(\mathbf{x}^{k+1} - \mathbf{x}^k))\,dt\right)(\mathbf{x}^{k+1} - \mathbf{x}^k).$$

Merkitään $\mathbf{s}^k = \mathbf{x}^{k+1} - \mathbf{x}^k$ ja $\mathbf{y}^k = \nabla f(\mathbf{x}^{k+1}) - \nabla f(\mathbf{x}^k)$ sekä lisäksi $\mathbf{z}(t) = \mathbf{x}^k + t\mathbf{s}^k$, jolloin saadaan

$$\left(\int_0^1 \nabla\nabla^T f(\mathbf{z}(t))\,dt\right)\mathbf{s}^k = \mathbf{y}^k. \tag{6.19}$$

Listaus 6.3: *Rosenbrockin funktion Newtonin menetelmällä minimoiva Fortran 90 -koodi. Yhtälöryhmän ratkaisu tehdään Lapack-aliohjelmakirjaston aliohjelmalla* `ssysv`.

```
PROGRAM Lapack_test
  IMPLICIT NONE
  INTEGER :: n, nrhs, lwork, j, info
  INTEGER, DIMENSION(:), ALLOCATABLE :: ipiv
  CHARACTER, PARAMETER :: uplo = 'u'
  REAL, DIMENSION(:), ALLOCATABLE :: x, gx, work
  REAL, DIMENSION(:,:), ALLOCATABLE :: hx
  REAL :: fx, xdiff, xtol
  ! Muuttujien alustus
  n = 2; nrhs = 1; lwork = n*n
  ALLOCATE(x(n), gx(n), hx(n,n), ipiv(n), work(lwork))
  x = (/ -1.2d0, 1.0d0 /) ! Alkuarvaus
  xdiff = 1.d0; xtol = 1.d-7
  j = 0
  ! Tehdään Newton-iteraatioita
  DO WHILE (xdiff > xtol)
    CALL derivfn(x, fx, gx, hx); WRITE(*,*) x, fx
    ! Ratkaistaan Newton-yhtälö Lapackilla
    CALL ssysv(uplo, n, nrhs, hx, n, ipiv, gx, n, &
        work, lwork, info)
    IF (info /= 0) STOP 'Hessen matriisi singulaarinen'
    x(:) = x(:) - gx(:)
    xdiff = SQRT(SUM(gx(:)**2))
    j = j + 1
  END DO
  WRITE(*,*) 'x = ', x, '; fx = ', fx, '; iter = ', j
CONTAINS
  SUBROUTINE derivfn(x, fx, gx, hx)
    IMPLICIT NONE
    REAL, DIMENSION(:), INTENT(IN) :: x
    REAL, INTENT(OUT) :: fx
    REAL, DIMENSION(SIZE(x)), INTENT(OUT) :: gx
    REAL, DIMENSION(SIZE(x),SIZE(x)), INTENT(OUT) :: hx
    ! Funktion arvo
    fx = 100.d0*(x(2) - x(1)**2)**2 + (1.d0 - x(1))**2
    ! Gradienttivektori
    gx(1) = -400.d0*x(1)*(x(2) - x(1)**2) + 2.d0*(x(1) - 1)
    gx(2) = 200.d0*(x(2) - x(1)**2)
    ! Hessen matriisin yläkolmio
    hx(1,1) = 1200.d0*x(1)**2 - 400.d0*x(2) + 2.d0
    hx(1,2) = -400.d0*x(1)
    hx(2,2) = 200.d0
  END SUBROUTINE derivfn
END PROGRAM Lapack_test
```

Integraalissa laskettavaa matriisia voi pitää ikään kuin keskimääräisenä Hessen matriisina janalla $\mathbf{z}(t)$. Kun tällä matriisilla kerrotaan otettu askel $\mathbf{s}^k$, saadaan gradientin muutos $\mathbf{y}^k$ (vrt. Newtonin menetelmän johtaminen yhtälöryhmän ratkaisemiseksi teoksessa [HHL+02]).

Edellisen tarkastelun perusteella sekanttimenetelmien matriisi B_{k+1} valitaan niin, että se toteuttaa *sekanttiyhtälön*

$$B_{k+1}(\mathbf{x}^{k+1} - \mathbf{x}^k) = \nabla f(\mathbf{x}^{k+1}) - \nabla f(\mathbf{x}^k). \tag{6.20}$$

Matriisin B_{k+1} tulisi olla myös symmetrinen ja positiivisesti definiitti. Esimerkiksi alla esiteltävän BFGS-sekanttimenetelmän tapauksessa positiivisdefiniittisyyden säilymisen takaa ehto $\langle \mathbf{y}^k, \mathbf{s}^k \rangle > 0$. Seuraavassa osoitetaan, että iteraatiomatriisien definiittisyys voidaan säilyttää eli ehto $\langle \mathbf{y}^k, \mathbf{s}^k \rangle > 0$ pysyy voimassa.

Olkoon iteraatiolla k matriisi B_k symmetrinen ja positiivisesti definiitti. Alussa voidaan valita esimerkiksi $B_1 = I$ (identiteettimatriisi). Jos hakusuunta $\mathbf{p}^k$ saadaan yhtälöstä $B_k \mathbf{p}^k = -\nabla f(\mathbf{x}^k)$, on se funktion vähenemissuunta matriisin B_k positiivisdefiniittisyyden perusteella. Hakuaskel $\mathbf{s}^k = \mathbf{x}^{k+1} - \mathbf{x}^k$ löydetään viivahaulla eli $\mathbf{x}^{k+1} = \mathbf{x}^k + \lambda_k \mathbf{p}^k,\ \lambda_k > 0$. Tällöin pätee

$$\langle \mathbf{y}^k, \mathbf{s}^k \rangle = \lambda_k \left(\langle \nabla f(\mathbf{x}^{k+1}), \mathbf{p}^k \rangle - \langle \nabla f(\mathbf{x}^k), \mathbf{p}^k \rangle \right). \tag{6.21}$$

Koska vektori $\mathbf{p}^k$ on funktion vähenemissuunta, on termi $-\langle \nabla f(\mathbf{x}^k), \mathbf{p}^k \rangle$ positiivinen. Lisäksi termi $\langle \nabla f(\mathbf{x}^{k+1}), \mathbf{p}^k \rangle$ saadaan mielivaltaisen pieneksi tarkalla viivahaulla (mikäli funktio on alhaalta rajoitettu), joten pätee $\langle \mathbf{y}^k, \mathbf{s}^k \rangle > 0$. Täten matriisit B_k pysyvät positiivisesti definiitteinä, kunhan käytetään kyllin tarkkaa viivahakua.

Edellä esitettyyn analyysiin perustuvat *sekanttimenetelmät* (joita kutsutaan joskus *muuttuvan metriikan menetelmiksi*) ovat käytetyimpiä optimointimenetelmiä. Usein käytetty BFGS-sekanttimenetelmä on seuraava:

Algoritmi 6.6.1 (BFGS-sekanttimenetelmä)

valitse $\mathbf{x}^1$ ja B_1 sekä 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 siis Hessen matriisin approksimaatio, jolle voidaan tehdä esimerkiksi Choleskyn hajotelma LL^T (tai LDL^T-hajotelma) ja päivittää sitä. Etuna esimerkiksi Choleskyn hajotelman käytössä on, että matriisi B_k voidaan pitää numeerisesti stabiilisti positiivisesti definiittinä.

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 6.6.1 Listauksessa 6.4 on esitetty yksinkertainen BFGS-menetelmän toteutus Mathematicalla. Matriisiksi B_1 asetetaan $H(\mathbf{x}^1)$. Matriisin B_1 voisi asettaa myös identiteettimatriisiksi. Lopetusehtona käytetään testiä $\|\nabla f(\mathbf{x}^k)\| < 10^{-6}$. Viivahaun tarkkuus voi olla pienempi kuin liittogradienttimenetelmässä (tässä tarkkuus on 10^{-4}).

Iteraatiopisteiksi $\mathbf{x}^k$ ja funktion arvoiksi $f(\mathbf{x}^k)$ saadaan

k	$\mathbf{x}^k$	$f(\mathbf{x}^k)$
1	$(-1.2, 1)$	24.2
2	$(-1.175, 1.381)$	4.732
3	$(-1.151, 1.324)$	4.626
⋮	⋮	⋮
38	$(1, 1)$	$3.86 \cdot 10^{-11}$
39	$(1, 1)$	$2.715 \cdot 10^{-14}$
40	$(1, 1)$	$2.245 \cdot 10^{-18}$

Tässä asetettiin $B_1 = H(\mathbf{x}^1)$ eli lähdettiin tarkasta Hessen matriisista pisteessä $\mathbf{x}^1$. Minimiarvon löytämiseen tarkkuudella 10^{-16} tarvitaan 40 BFGS-askelta. Jos asetetaan $B_1 = I$ (identiteettimatriisi), tarvitaan tässä esimerkissä kolme iteraatiota vähemmän. Siis Newton-askeleen suunta ei ole pisteessä $\mathbf{x}^1$ kovin hyvä alkuarvaus.

Huomautus 6.6.1 BFGS-sekanttimenetelmä on luotettava ja tehokas, mutta isoille tehtäville ovat liittogradienttimenetelmät pienemmän muistintarpeensa ansiosta yleensä tehokkaampia.

BFGS-menetelmästä on olemassa myös rajoitetun muistin versioita (limited memory BFGS) suurille tehtäville. Ideana on tallettaa Hessen matriisin (tai käänteismatriisin) sijaan joukko vektoreita $\mathbf{s}^k$ ja $\mathbf{y}^k$, joista päivitys muodostetaan.

Esimerkki 6.6.2 Listauksessa 6.5 on esimerkki NAG-aliohjelmakirjaston rutiinin E04DGE käytöstä Rosenbrockin funktion minimointiin, kun muuttujien lukumäärä n on 1000. Olkoon $\eta = n/2$, jolloin kohdefunktio määritellään seuraavasti:

$$f(\mathbf{x}) = \sum_{i=1}^{\eta} 100 \left(x_{2i} - x_{2i-1}^2\right)^2 + (1 - x_{2i-1})^2 ,$$

Minimointiin tarvittava pääohjelma ja aliohjelma on kirjoitettu Fortran 90 -kielellä. Tässä käytetyn NAG-rutiinin sisältämä rajoitetun muistin BFGS-menetelmä tekee Hessen matriisin käänteismatriisin arvion $A_k \approx H(\mathbf{x}^k)^{-1}$ päivityksen

kaavoilla

$$\mathbf{y}^k = \nabla f(\mathbf{x}^{k-1}) - \nabla f(\mathbf{x}^k) \tag{6.22}$$

$$\mathbf{s}^k = \mathbf{x}^{k+1} - \mathbf{x}^k \tag{6.23}$$

$$A_{k+1} = A_k - \frac{1}{\langle \mathbf{y}^k, \mathbf{s}^k \rangle} \left(A_k \mathbf{y}^k \mathbf{s}^{kT} + \mathbf{s}^k \mathbf{y}^{kT} A_k \right) + \frac{1}{\langle \mathbf{y}^k, \mathbf{s}^k \rangle} \left(1 + \frac{\langle \mathbf{y}^k, A_k \mathbf{y}^k \rangle}{\langle \mathbf{y}^k, \mathbf{s}^k \rangle} \right) \mathbf{s}^k \mathbf{s}^{kT} \tag{6.24}$$

Menetelmässä riittää tallettaa vain joukko vektoreita $\mathbf{y}^k$ ja $\mathbf{s}^k$, jotka määrittelevät BFGS-päivityksen — koko matriisia A_k ei tarvita.

6.7 Liittogradienttimenetelmät

Liittogradienttimenetelmien (conjugate gradient methods) voi katsoa olevan muistittomia sekanttimenetelmiä. Seuraava liittogradienttimenetelmä soveltuu epälineaariseen rajoitteettomaan optimointiin.

Algoritmi 6.7.1 (Liittogradienttimenetelmä)

hae alkuarvaus $\mathbf{x}^1 \in \mathbb{R}^n$
repeat
  $k = 1$
  repeat
    $\mathbf{g}^k = \nabla f(\mathbf{x}^k)$
    if $k = 1$
      $\beta_1 = 0, \quad \mathbf{s}^0 = \mathbf{0}$
    else
      laske β_k jollakin alla esitetyistä menetelmistä
    end
    $\mathbf{s}^k = -\mathbf{g}^k + \beta_k \mathbf{s}^{k-1}$
    viivahaku: $\mathbf{x}^{k+1} = \mathbf{x}^k + \lambda_k \mathbf{s}^k$
    $k = k + 1$
  until $\|\mathbf{g}^k\| < \epsilon$ tai $k > \eta$
until $\|\mathbf{g}^k\| < \epsilon$

Tässä esitettiin *uudelleenkäynnistetty* liittogradienttimenetelmä, jossa käynnistetään iteraatio uudestaan aina η askeleen välein. Viivahaussa minimoidaan funktio $f(\mathbf{x})$ suuntaan $\mathbf{s}^k$ eli haetaan piste $\mathbf{x}^{k+1} = \mathbf{x}^k + \lambda_k \mathbf{s}^k$ siten, että funktio $f(\mathbf{x}^k + \lambda_k \mathbf{s}^k)$, $\lambda_k > 0$ saa minimin.

Skalaarikerroin β_k lasketaan Fletcherin ja Reevesin menetelmässä kaavalla

$$\beta_k = \frac{\langle \mathbf{g}^k, \mathbf{g}^k \rangle}{\langle \mathbf{g}^{k-1}, \mathbf{g}^{k-1} \rangle}, \quad k = 2, 3, \ldots, \tag{6.25}$$

Listaus 6.4: *Rosenbrockin funktion minimin hakeminen BFGS-sekanttimenetelmällä. Rutiini* `BFGSStep` *laskee yhden BFGS-menetelmän iteraatioaskeleen.*

```
<<Derivaatat.m
<<LinSearch.m

DOT[x_, y_] := Part[Transpose[x] . y, 1, 1];
InDOT[x_, y_] := x . Transpose[y];

(* Funktion arvo, gradientti ja Hessen matriisi *)
f[x1_,x2_] = 100 (x2 - x1^2)^2 + (1 - x1)^2;
gf[x1_,x2_] = Simplify[Grad[f[x1,x2],{x1,x2}]];
Hf[x1_,x2_] = Simplify[Hesse[f[x1,x2],{x1,x2}]];

(* BFGS-päivityksen suorittava rutiini *)
BFGSStep[f_, gf_, {x0_List, fx0_, gfx0_List, B0_List}] :=
Block[{p, newx, fxn, RCODE, maxstep = 100, tol = 10^(-4),
       s, gfnew, y, Bs, B},
  p = - LinearSolve[B0, gfx0];
  {newx,fxn,RCODE} = LinSearch[f,x0,p,maxstep,tol,fx0,gfx0];
  s = Transpose[{newx - x0}];
  gfnew = gf @@ newx;
  y = Transpose[{gfnew - gfx0}];
  Bs = B0 . s;
  B = B0 + InDOT[y,y]/DOT[y,s] -
      (Bs.Transpose[s].B0)/DOT[s,Bs];
  {newx, fxn, gfnew, B}]

(* Asetetaan alkuarvot minimin hakemiseksi *)
x0 = N[{-12/10,1}, 16];
fx0 = Apply[f, x0];
gf0 = Apply[gf, x0];
B0 = Apply[Hf, x0];
InitVal = {x0, fx0, gf0, B0};

(* Iteroidaan, kunnes gradientin normi on kyllin pieni *)
BFGSResults = FixedPointList[
  Apply[BFGSStep[f,gf,#]&,{#}]&, InitVal,
  SameTest -> ((Sqrt[Apply[Plus,#2[[3]]^2]] < 10.^-6) &)];

(* Haetaan iteraatiopisteet ja funktion arvot *)
BFGSIterates = Map[#[[1]]&, BFGSResults]
functionValues = Map[Apply[f,#]&, BFGSIterates]
```

Listaus 6.5: *Moniulotteisen (1000 muuttujaa) Rosenbrockin funktion minimin hakeminen rajoitetun muistin BFGS-menetelmällä. Fortran 90 -ohjelma käyttää optimointitehtävän ratkaisemiseen NAG-aliohjelmakirjaston rutiinia E04DGE.*

```
PROGRAM NAG_test
  IMPLICIT NONE
  INTEGER :: n, ifail, iter, i, iuser(1)
  REAL :: objf, user(1)
  REAL, DIMENSION(:), ALLOCATABLE :: x, objgrd, work
  INTEGER, DIMENSION(:), ALLOCATABLE :: iwork
  EXTERNAL e04dge, e04dke, fun
  n = 1000
  ALLOCATE(x(n), objgrd(n), work(13*n), iwork(n+1))
  ! Alkuarvaus ratkaisulle
  x(1:n-1:2) = (/ (-1.2d0 - 0.1d0*COS(REAL(i)), i=1,n-1,2) /)
  x(2:n:2) = 1.d0 + 0.1d0*COS(x(1:n-1:2))
  ! Parametrit ratkaisurutiinille
  CALL e04dke('   Maximum step length = 100')
  CALL e04dke('   Optimality tolerance = 1.0d-14')
  CALL e04dke('   Iteration limit = 1000')
  CALL e04dke('   Print level = 1')
  ifail = -1
  CALL e04dge(n, fun, iter, objf, objgrd, x, iwork, &
      work, iuser, user, ifail)
  WRITE(*,*) 'Minimiarvo: ', objf
END PROGRAM NAG_test

! Minimoitava funktio ja gradienttivektori
SUBROUTINE fun(mode, n, x, objf, objgrd, nstate, iuser, user)
  IMPLICIT NONE
  INTEGER, INTENT(IN) :: mode, n, nstate
  INTEGER, DIMENSION(:), INTENT(IN) :: iuser
  REAL, DIMENSION(n), INTENT(IN) :: x
  REAL, DIMENSION(:), INTENT(IN) :: user
  REAL, INTENT(OUT) :: objf
  REAL, DIMENSION(SIZE(x)), INTENT(OUT) :: objgrd
  INTEGER :: i
  REAL ::  t1, t2
  objf = 0.d0
  DO i = 1, n-1, 2
    t1 = 1.d0 - x(i); t2 = 10.d0*(x(i+1) - x(i)**2)
    ! Kohdefunktion arvo
    objf = objf + t1**2 + t2**2
    ! Gradienttivektori
    objgrd(i+1) = 20.d0*t2
    objgrd(i) = -2.d0*(x(i)*objgrd(i+1) + t1)
  END DO
END SUBROUTINE fun
```

sekä Polakin ja Ribièren menetelmässä kaavalla

$$\beta_k = \frac{\langle \mathbf{g}^k - \mathbf{g}^{k-1}, \mathbf{g}^k \rangle}{\langle \mathbf{g}^{k-1}, \mathbf{g}^{k-1} \rangle}, \quad k = 2, 3, \ldots \tag{6.26}$$

Huomautus 6.7.1 Liittogradienttimenetelmä ei tarvitse toisia derivaattoja, ja esimerkiksi Fletcherin ja Reevesin menetelmä tarvitsee vain kolme n-vektoria. Täten liittogradienttimenetelmät ovat käyttökelpoisia, kun tehtävän muuttujien lukumäärä n on suurempi kuin 250 ja kohdefunktion $f(\mathbf{x})$ Hessen matriisi $H(\mathbf{x})$ ei ole harva.

Esimerkki 6.7.1 Haetaan Rosenbrockin funktion minimi Polakin ja Ribièren liittogradienttimenetelmällä (vertaa esimerkkeihin 6.5.2 ja 6.6.1). Listauksessa 6.6 on esitetty menetelmän toteutus Mathematica-ohjelmistolla. Koodissa käytetään luvussa 12 esitettyä Mathematica-rutiinia `Grad` gradientin laskemiseen ja viivahakuun listauksen 13.1 rutiinia `LinSearch`. Lopetusehtona on $\|\nabla f(\mathbf{x}^k)\| < 10^{-6}$. Mathematica-koodiin ei ole ohjelmoitu uudelleenkäynnistystä.

Seuraavassa on esitetty iteraatiopisteet ja funktion arvot:

k	$\mathbf{x}^k$	$f(\mathbf{x}^k)$
1	$(-1.2, 1)$	24.2
2	$(-0.9645, 1.096)$	6.612
3	$(-1.088, 1.095)$	5.142
4	$(-1.057, 1.124)$	4.236
5	$(-0.9619, 0.9668)$	4.021
6	$(-0.788, 0.5823)$	3.347
⋮	⋮	⋮
30	$(1, 0.9999)$	$2.061 \cdot 10^{-9}$
31	$(1, 1)$	$5.372 \cdot 10^{-11}$
32	$(1, 1)$	$2.771 \cdot 10^{-16}$

Minimiarvon löytämiseen tarvittiin 32 Polakin ja Ribièren liittogradienttimenetelmän iteraatiota. Samaan tehtävään vaadittaisiin 72 Fletcherin ja Reevesin menetelmän iteraatiota tässä käytetyllä Mathematica-koodilla.

6.8 Suorahakumenetelmät: polytooppihaku

Suorahakumenetelmissä (direct search) käytetään vain kohdefunktion arvoja eikä lasketa derivaattoja. Suorahakumenetelmien suosio kasvoi merkittävästi 1990-luvun aikana, ja eräille menetelmille on saatu myös teoreettisia suppenemistuloksia [Wri95, KLT03].

Merkittävä syy suorahakumenetelmien suosioon on ratkaistavien mallien diskretoinnin tai numeerisen laskennan tuottama häiriö. Esimerkiksi elementtimenetelmällä tehty diskretointi voi tehdä lasketuista funktion arvois-

Listaus 6.6: *Rosenbrockin funktion minimin hakeminen Polakin ja Ribièren liittogradienttimenetelmällä. Rutiini* `CGStep` *laskee yhden iteraatioaskeleen.*

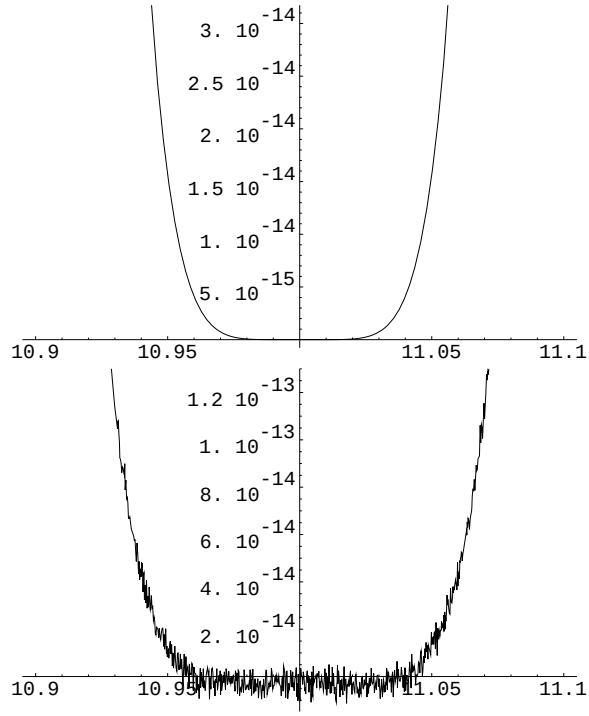
```
<<Derivaatat.m
<<LinSearch.m

(* Funktion arvo ja gradientti *)
f[x1_,x2_] = 100 (x2 - x1^2)^2 + (1 - x1)^2;
gf[x1_,x2_] = Simplify[Grad[f[x1,x2],{x1,x2}]];

(* Liittogradienttiaskeleen laskeva rutiini *)
CGStep[f_, gf_, {x0_List, fx0_, gfx0_List, gf0_List,
       step0_List}] :=
Block[{fxn, gfn, prbeta, step, newx, maxstep = 100,
       steptol = 10^(-8)},
  prbeta = Apply[Plus,(gfx0 - gf0)*gfx0]/Apply[Plus,gf0^2];
  step = -gfx0 + prbeta*step0;
  If[Apply[Plus, step*gfx0] > 0, step = - step];
  {newx, fxn, RETCODE} =
    LinSearch[f,x0,step,maxstep,steptol,fx0,gfx0];
  gfn = gf @@ newx;
  {newx, fxn, gfn, gfx0, step}]

(* Asetetaan alkuarvot *)
gf0 = {1,1};
step0 = {0,0};
x0 = N[{-12/10,1}, 16];
fx0 = f @@ x0;
gfx0 = gf @@ x0;
InitVal = {x0, fx0, gfx0, gf0, step0};

(* Iteroidaan, kunnes gradientin normi on < 10.^-6 *)
CGResults = FixedPointList[
  Apply[CGStep[f,gf,#]&,{#}]&, InitVal,
  SameTest -> ((Sqrt[Apply[Plus,#2[[3]]^2]] < 10.^-6) &)];

(* Haetaan iteraatiopisteet ja funktion arvot *)
CGIterates = Map[#[[1]]&, CGResults]
functionValues = Map[Apply[f,#]&, CGIterates]
```

ta epäjatkuvia. Numeerisen laskennan häiriöitä havainnollistaa seuraava esimerkki. Tällaisessa tapauksessa suorahakumenetelmä voi olla paras (ja ainoa) toimiva ratkaisualgoritmi.

Esimerkki 6.8.1 Korkea-asteiset polynomit ovat virhealttiita. Piirrämme polynomin

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

kuvaajan (kuva 6.5) käyttäen yllä olevaa esitysmuotoa ja tämän jälkeen auki kirjoitettua muotoa. Johtuen liukulukuaritmetiikan epätarkkuudesta tuottaa auki kirjoitettu esitys kohinaisen kuvaajan, josta ei voi määrittää luotettavasti funktion minimiä.



Kuva 6.5: *Funktion $f(x) = (x/10 - 1.1)^6$ kuvaajan piirtäminen kahdella eri esitystavalla. Erot johtuvat korkea-asteisen polynomin häiriöalttiudesta.*

Nelderin ja Meadin polytooppihaussa muodostetaan n-ulotteinen monitahokas (jota kutsutaan usein *simpleksiksi*), jota sopivasti skaalaamalla ja sivujen suhteen peilaamalla pyritään löytämään nollakohdan sijainti.

Polytooppi sisältää $n + 1$ kärkipistettä $\mathbf{x}_i$. Merkitään 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 $f_i = f(\mathbf{x}_i)$.

Olkoon polytoopin n parhaan kärjen painopiste $\mathbf{x}_c = \frac{1}{n}\sum_{i\neq h}\mathbf{x}_i$. Valitaan peilauskerroin $\alpha = 1 > 0$, laajennuskerroin $\gamma = 2 > 1$ ja pienennyskerroin $\beta = 0.5 \in [0,1]$.

Polytooppihaussa käytetyt skaalaus- ja peilausoperaatiot toimivat seuraavasti (kuva 6.6):

Algoritmi 6.8.1 (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:

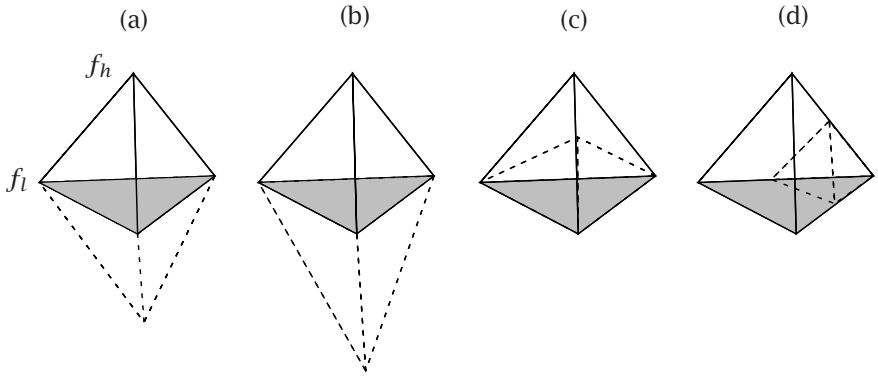
$$\mathbf{x}_e = \gamma\mathbf{x}_r + (1-\gamma)\,\mathbf{x}_c\,.$$

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.



Kuva 6.6: *Esimerkkejä polytooppihaussa käytetyistä monitahokkaan skaalaus- ja 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.*

Polytooppihaun lopetusehtona on esimerkiksi

$$f_h - f_l \leq \epsilon\,(1 + |f_l|) \quad \text{tai} \quad \sum_{i=1}^{n+1} \left(f(\mathbf{x}_i) - \frac{\sum_{j=1}^{n+1} f(\mathbf{x}_j)}{n+1} \right)^2 \leq \epsilon\,. \tag{6.27}$$

Polytooppihaku on varsin luotettava, mutta sen suppenemisnopeus saattaa olla huono verrattuna sekantti- tai liittogradienttimenetelmiin. Toisaalta polytooppihaku soveltuu tapauksiin, jossa optimoitava funktio ei ole sileä. Polytooppihakuun perustuva rutiini löytyy esimerkiksi IMSL-aliohjelmakirjastosta (katso taulukkoa B.4 sivulla B.4). IMSL:n polytooppihakurutiinia on käytetty kuvassa 6.7 esitetyssä menetelmien vertailussa.

6.9 Menetelmiä separoituville tehtäville

Separoituviksi kutsutaan tehtäviä, joissa kohdefunktio on muotoa

$$f(\mathbf{x}) = \sum_{i=1}^{m} f_i(\mathbf{x}).$$

Tässä f_i riippuu r_i:stä vektorin $\mathbf{x}$ alkiosta, $r_i \ll n$. Separoituville tehtäville on kehitetty mm. ositettuja sekanttimenetelmiä. Menetelmissä jaetaan Hessen matriisin arvio osamatriiseihin, ja suoritetaan laskutoimitukset näillä. Ongelmana on tietysti ositetun matriisin mahdollinen singulaarisuus.

Myös TOMS-algoritmin 702 eli TNPACKin käyttämä katkaistu Newtonin menetelmä sopii separoituville tehtäville, sillä tällöin voidaan tehokkaasti hyödyntää Hessen matriisin harvaa rakennetta.

Esimerkki 6.9.1 Rosenbrockin n-ulotteinen funktio (tässä $\eta = n/2$)

$$f(\mathbf{x}) = \sum_{i=1}^{\eta} 100 \left(x_{2i} - x_{2i-1}^2 \right)^2 + (1 - x_{2i-1})^2\,,$$

on separoituva, sillä se muodostuu kahden muuttujan osafunktioiden summasta. Hessen matriisi $H(\mathbf{x})$ koostuu lävistäjällä olevista 2×2 -lohkoista

$$\begin{cases} h_{2i-1,2i-1} = 1200 x_{2i-1}^2 - 400 x_{2i} + 2, \\ h_{2i,2i-1} = h_{2i-1,2i} = -400 x_{2i-1}, \\ h_{2i,2i} = 200, \end{cases}$$

missä $i = 1, 2, \ldots, \eta$. Kappaleessa 6.10 käytetään tätä tehtävää ratkaisuohjelmistojen testaamiseen. Tapauksessa $n = 4$ on Hessen matriisi muotoa

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

missä merkillä $\times$ tarkoitetaan nollasta poikkeavia alkioita.

6.10 Ratkaisumenetelmien vertailu

Aikaisemmin tässä luvussa on esitetty Mathematicalla ohjelmoitu Newtonin menetelmä, BFGS-sekanttimenetelmä ja liittogradienttimenetelmä.

Kuvassa 6.7 on esitetty näiden menetelmien generoimat hakupisteet minimoitaessa Rosenbrockin funktiota. Polytooppihaussa käytettiin IMSL-aliohjelmakirjaston rutiinia UMPOL.

Kaikissa tapauksissa oli alkuarvauksena $\mathbf{x}^1 = (-1.2, 1)^T$. Kaikki menetelmät löysivät likimääräisesti minimipisteen $\mathbf{x}^* = (1, 1)^T$. Newtonin menetelmä harhautui eräässä vaiheessa kauas optimista, liittogradienttimenetelmä "ampui yli" minimipisteestä, ja polytooppihaku eteni hitaasti kohti minimiä. Tasaisimmin käyttäytyi esimerkissä 6.6.1 käytetty BFGS-sekanttimenetelmä.

Newtonin menetelmä tarvitsi 7 iteraatiota, liittogradienttimenetelmä 32 ja BFGS-menetelmä 40 iteraatiota. IMSL:n polytooppihakurutiini laski kohdefunktion arvon 186 kertaa.

Seuraavassa tutkitaan suurten, epälineaaristen ja rajoitteettomien optimointitehtävien ratkaisuohjelmistoja. Tehtävänä on esimerkissä 6.9.1 esitetyn n-ulotteisen Rosenbrockin funktion minimoiminen. Kyseessä on separoituva tehtävä, jonka Hessen matriisi on harva.

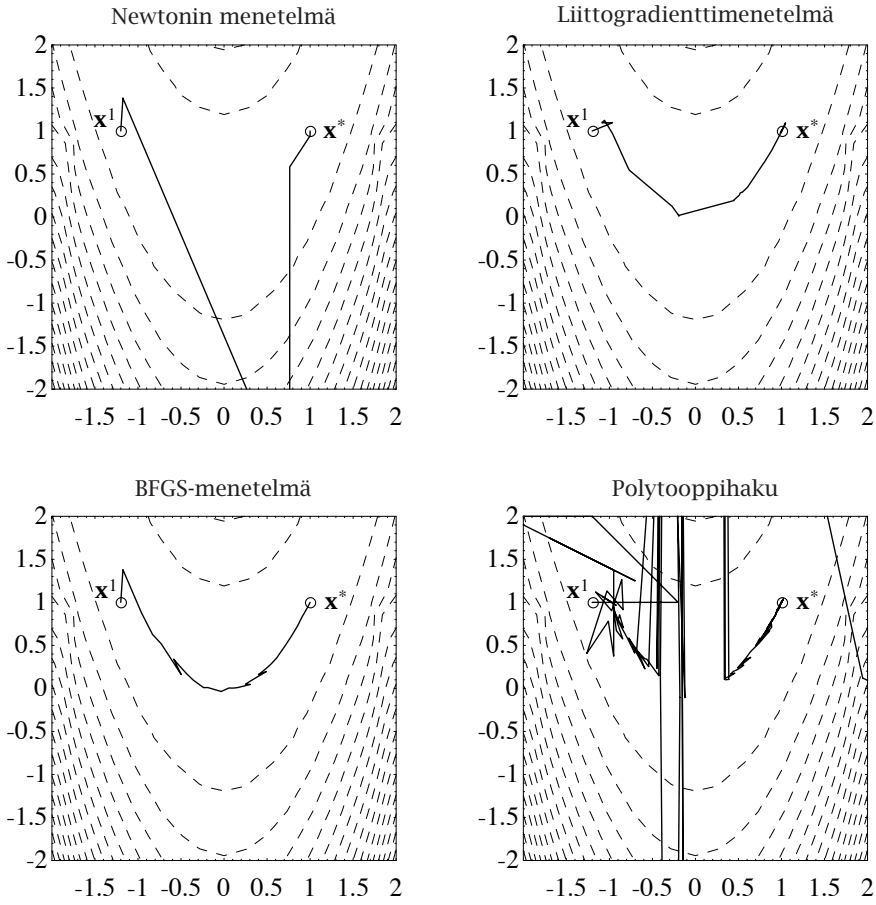
Taulukossa 6.1 on lueteltu testatut rutiinit ja niiden kuvaukset.

Taulukko 6.1: *Testatut suurten rajoitteettomien optimointitehtävien ratkaisurutiinit. CG on liittogradienttimenetelmä ja BFGS on BFGS-sekanttimenetelmä. Sarake "muistinkäyttö" pyrkii osoittamaan algoritmin vähintään vaatiman tietokoneen keskusmuistin koon liukulukuina (n on muuttujien lukumäärä).*

Rutiini ja lähde	*Menetelmä*	*Muistinkäyttö*
BBVSCG (TOMS 630)	CG ja sekanttimenetelmä	$3n \ldots n(n+7)/2$
TNPACK (TOMS 702)	Katkaistu Newton	$\approx 11n$
VE08 (Netlib)	Ositettu sekanttimenetelmä	$> 6n$
E04DGF (NAG)	Rajoitetun muistin BFGS	$13n$
DUMCGG (IMSL)	Uudelleenkäynnistetty CG	$6n$

Taulukossa 6.2 on testien tuloksia. Laajempi tässä esiteltyjen menetelmien ja ohjelmistojen testausraportti löytyy teoksesta [Haa94]. Kaikki testit on suoritettu CSC:n Convex C3840-minisuperilla. Kaikista ohjelmistoista oli käytössä kaksoistarkkuuden liukulukuja (64 bittiä) käyttävä versio.

Taulukossa 6.2 on kerrottu kunkin ohjelmiston käyttämien iteraatioiden lukumäärä ja käytetty CPU-aika sekunteina. Piste $\mathbf{x}^f$ on ohjelmiston antama ratkaisu ja $\|\mathbf{x}^* - \mathbf{x}^f\|$ on ratkaisun virhe euklidisen normin mielessä. BBVSCG:n yhteydessä tutkittiin myös muistitilan vaikutusta (luokkaa $5n$ tai $20n$ liukulukua). IMSL-aliohjelmakirjastosta käytettiin versiota 2.0 ja NAG-aliohjelmakirjastosta versiota Mark 14.



Kuva 6.7: *Rosenbrockin kaksiulotteisen funktion minimointi eri menetelmillä. Piste* $\mathbf{x}^1$ *on alkuarvaus ja piste* $\mathbf{x}^*$ *tehtävän minimipiste.*

Muuttujien lukumäärä on $n = 10\,000$. Alkuarvauksena käytettiin kolmea pistettä:

$$\begin{cases} \mathbf{x}_1^1 = (-1.2, 1.0, -1.2, 1.0, \cdots)^T, \\ \mathbf{x}_2^1 = (-0.5, -1.5, -1.5, 0.5, -0.5, -1.5, \cdots)^T, \\ \mathbf{x}_3^1 = \mathbf{x}_1^1 + \mathbf{r}, \end{cases}$$

missä $\mathbf{r}$ on pieni muutosvektori ($|r_i| < 0.5$). Kunkin ohjelmiston asetukset olivat samat kaikilla alkuarvauksilla.

IMSL- ja NAG-aliohjelmakirjastoja käytettäessä on pyritty valitsemaan mahdollisimman tehokkaat asetukset tehtävän ratkaisemiselle. Ohjelmistot TNPACK ja BBVSCG on asennettu Convexille käyttäen Fortran-kääntäjän vektorointivalitsinta, mutta teho ei aivan vastaa koneen huipputehoa johtuen runsaasta harvojen matriisien indeksijoukkojen käsittelystä. TNPACKissa käytetään muista testatuista menetelmistä poiketen myös Rosenbrockin funktion

Hessen matriisia, joka on talletettu hyödyntämällä sen rakenteellisuutta.

Taulukko 6.2: *Rajoitteettomien optimointitehtävien ratkaisuohjelmistojen vertaamisesta saatuja tuloksia. Optimoitavana on Rosenbrockin testifunktio. Muuttujien lukumäärä n on* 10 000.

Rutiini	$f(\mathbf{x}^f)$	$\|\mathbf{x}^* - \mathbf{x}^f\|$	*Iter.*	*CPUs*
Alkuarvaus $\mathbf{x}_1^1$				
BBVSCG ($5n$)	2.5×10^{-28}	3.5×10^{-14}	28	1.7
BBVSCG ($20n$)	1.3×10^{-25}	7.2×10^{-13}	33	2.0
TNPACK	8.1×10^{-19}	4.3×10^{-10}	21	3.8
VE08	4.3×10^{-20}	4.6×10^{-10}	21	12.9
E04DGF	1.2×10^{-15}	7.7×10^{-8}	23	0.8
DUMCGG	9.1×10^{-14}	6.1×10^{-7}	70	0.5
Alkuarvaus $\mathbf{x}_2^1$				
BBVSCG ($5n$)	1.0×10^{-11}	7.1×10^{-6}	62	2.2
BBVSCG ($20n$)	2.2×10^{-17}	1.1×10^{-8}	84	3.4
TNPACK	2.3×10^{-20}	2.4×10^{-10}	27	5.6
VE08	3.2×10^{-21}	1.3×10^{-10}	27	17.2
E04DGF	2.8×10^{-9}	1.2×10^{-4}	75	2.5
DUMCGG	2.2×10^{-18}	6.7×10^{-10}	133	0.9
Alkuarvaus $\mathbf{x}_3^1$				
BBVSCG ($5n$)	8.1×10^{-16}	6.4×10^{-8}	56	2.1
BBVSCG ($20n$)	6.7×10^{-16}	3.9×10^{-8}	66	2.9
TNPACK	1.6×10^{-16}	5.2×10^{-8}	23	6.3
VE08	1.8×10^{-19}	3.9×10^{-10}	305	205
E04DGF	6.2×10^{-7}	1.8×10^{-3}	539	18.8
DUMCGG	2.5×10^{-14}	3.4×10^{-8}	108	0.7

Tässä testitehtävässä IMSL:n liittogradienttimenetelmään perustuva rutiini DUMCGG oli tehokkaampi kuin NAG tai TNPACK. NAGin rajoitetun muistin BFGS-menetelmää käyttävä E04DGF ei toiminut kovin hyvin alkuarvauksen $\mathbf{x}_3^1$ tapauksessa. Merkillepantavaa on, että TNPACK ratkaisi tehtävän parillakymmenellä iteraatiolla, vaikka muuttujia oli 10 000.

Huomautus 6.10.1 Valittaessa ratkaisumenetelmää tulisi menetelmiä ja ohjelmistoja vertailla käyttäen realistista mallia ja tuotantoajoja vastaavaa dataa. Tässä esitetty vertailu perustuu vain yhden testitehtävän käyttöön eikä siten ole yleispätevä.

6.11 Ohjelmistoja

Matlabin Optimization Toolbox tarjoaa mahdollisuuden pienten ja keskisuurten optimointitehtävien ratkaisemiseen (taulukko B.3 sivulla 209). Myös Netlibin TOMS-algoritmeista löytyy käyttökelpoisia rutiineja (taulukko B.6 sivulla 214)

IMSL:n rutiinit UMIAH ja UMIDH ratkaisevat rajoitteettoman minimointitehtävän modifioidulla Newtonin menetelmällä käyttäen lisäksi luottamusaluetta suppenemisen varmistamiseen [DS83, IMS]. NAG-aliohjelmakirjastossa puolestaan on viivahakua käyttävät toteutukset Newtonin menetelmästä (rutiinit E04KDF ja E04LBF, katso [Num]). Kummastakin aliohjelmakirjastosta löytyy tehokkaita BFGS-menetelmien toteutuksia.

Separoituville tehtäville on kehitetty mm. ositettuja sekanttimenetelmiä. Netlibistä löytyvä rutiini VE08 käyttää myös hyväkseen kohdefunktion separoitumista. Isoille tehtäville on kehitetty rajoitetun muistin algoritmeja BFGS-menetelmästä. Newtonin menetelmästä on olemassa katkaistuja versioita, jotka vaativat vähemmän keskusmuistia. Esimerkiksi TOMS-algoritmi 702 (TNPACK) sopii isojen tehtävien ratkaisemiseen [SF92].

Mathematica-ohjelmisto sisältää rutiinin `FindMinimum`, joka näyttäisi perustuvan liittogradienttimenetelmään, mikäli derivaatat ovat käytettävissä, ja ns. Powellin menetelmään [Fle87], mikäli derivaattoja ei ole käytettävissä. Tätä rutiinia ei tässä luvussa käytetä, koska yksityiskohtaista kuvausta rutiinin toiminnasta ei ole saatavilla [Rus93a].

Tilastomatematiikan ohjelmisto SAS sisältää SAS/OR-osuudessa proseduurin epälineaariseen optimointiin [Rus93b].

CSC:n oppaassa *Numeeriset menetelmät* [HKR93] on esimerkki Matlabin polytooppihakuun perustuvan rutiinin `fmins` käyttämisestä Rosenbrockin funktion minimointiin. Lisäksi aliohjelmakirjastot IMSL ja NAG sisältävät polytooppihakurutiinin [IMS, Num].

Epälineaarisen optimoinnin ohjelmistoja on esitelty teoksissa [Sca85, DS83] sekä käsikirjoissa [IMS, Num]. Erityisen hyödyllinen on *Optimization Software Guide* [MW93].

Tiheiden lineaaristen yhtälöryhmien käsittelyyn on tehokkain vaihtoehto Lapack-aliohjelmakirjasto [ABB+92]. Lisätietoja löytyy CSC:n oppaasta *Matemaattiset ohjelmistot* [HHL+03].

6.12 Lisätietoja

Teokset *Practical Optimization* [GMW81] ja *Practical Methods of Optimization* [Fle87] tarjoavat käytännöllisen ja perusteellisen johdatuksen epälineaariseen optimointiin. Myös teoksia [Sca85, DS83, Mie98] voi käyttää oppikirjoina. Lyhyt katsaus aiheeseen löytyy CSC:n oppaasta *Numeeriset menetelmät* [HKR93].

Epälineaarisen optimoinnin perusteita on esitelty teoksessa *Nonlinear Programming: Theory and Algorithms* [BSS93]. Ratkaisumenetelmien tehokkuutta on käsitelty teoksissa [DS83, Noc92, Sca85].

Eri tyyppisiä sekanttimenetelmiä on vertailtu raportissa [Luk92]. LDL^T-hajotelman käyttöä on esitelty teoksissa [HKR93, Sca85, ES91]. Rajoitetun muistin BFGS-menetelmää on käsitelty viitteissä [Col84, Noc92]. Harvoille matriiseille kehitettyjä menetelmiä esitellään teoksissa [HKR93, Col84, Pis84].

Liittogradienttimenetelmiä on esitelty teoksissa [Hes80, HKR93, Sca85]. Viivahaun tarkkuuden merkitystä on käsitelty teoksissa [Sca85, GMW81]. Luottamusaluemenetelmiä on kuvattu kappaleessa 13.4 sivulla 199.

7 Pienimmän neliösumman tehtävät

Tässä luvussa esitellään perusmenetelmät epälineaaristen pienimmän neliösumman tehtävien ratkaisemiseen. Yksinkertaisen esimerkkitehtävän ratkaisuun käytetään Matlabia.

7.1 PNS-tehtävät

Epälineaarisessa pienimmän neliösumman tehtävässä (PNS, nonlinear least squares eli NLSQ) minimoidaan funktiota

$$F(\mathbf{x}) = \sum_{i=1}^{m} (f_i(\mathbf{x}))^2 = \langle \mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x}) \rangle, \tag{7.1}$$

missä piste $\mathbf{x}$ kuuluu joukkoon $\mathbb{R}^n$ ja funktio $\mathbf{f}(\mathbf{x})$ on kuvaus $\mathbb{R}^n \mapsto \mathbb{R}^m$. Minimointitehtävä voi yleisessä tapauksessa sisältää myös rajoitteita.

Pienimmän neliösumman tehtävä liittyy läheisesti myös epälineaaristen yhtälöryhmien ratkaisemiseen, sillä yhtälöryhmän

$$\mathbf{f}(\mathbf{x}) = \mathbf{0} \tag{7.2}$$

ratkaisu $\mathbf{x}^*$ on pienimmän neliösumman tehtävän globaali minimi. Yhtälöryhmän ratkaisua ei kuitenkaan useimmiten kannata muuntaa pienimmän neliösumman tehtäväksi, sillä tällöin ratkaisuksi voidaan saada lokaali minimipiste, jolla ei päde $\mathbf{f}(\mathbf{x}) = \mathbf{0}$.

Esimerkki 7.1.1 Esimerkissä 6.9.1 minimoitiin Rosenbrockin n-ulotteista funktiota (tässä $\eta = n/2$)

$$f(\mathbf{x}) = \sum_{i=1}^{\eta} 100 \left(x_{2i} - x_{2i-1}^2 \right)^2 + (1 - x_{2i-1})^2 .$$

Tehtävä voidaan esittää myös epälineaarisena pienimmän neliösumman tehtävänä, jossa minimoitavana on funktio

$$F(\mathbf{x}) = \sum_{i=1}^{n} (f_i(\mathbf{x}))^2 , \quad \begin{cases} f_i = 1 - x_i, & i \text{ pariton}, \\ f_i = 10 \left(x_i - x_{i-1}^2 \right), & i \text{ parillinen}. \end{cases} \tag{7.3}$$

Harjoitustehtäväksi jätetään tehtävän esittäminen yhtälöryhmänä.

7.1.1 Datan sovitus

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 myös epälineaarisiin pienimmän neliösumman tehtäviin. Parametreille $\mathbf{a}$ haetaan sellaiset arvot, jotka minimoivat neliösumman

$$F(\mathbf{a}) = \sum_{i=1}^{m} w_i^2 \left(y_i - f(\mathbf{t}_i, \mathbf{a})\right)^2. \tag{7.4}$$

Arvot w_i ovat sovituksessa käytettyjä painokertoimia. Tehtävä saadaan muotoon (7.1) asettamalla $\mathbf{x} = \mathbf{a}$ ja $f_i(\mathbf{x}) = w_i(y_i - f(\mathbf{t}_i, \mathbf{x}))$.

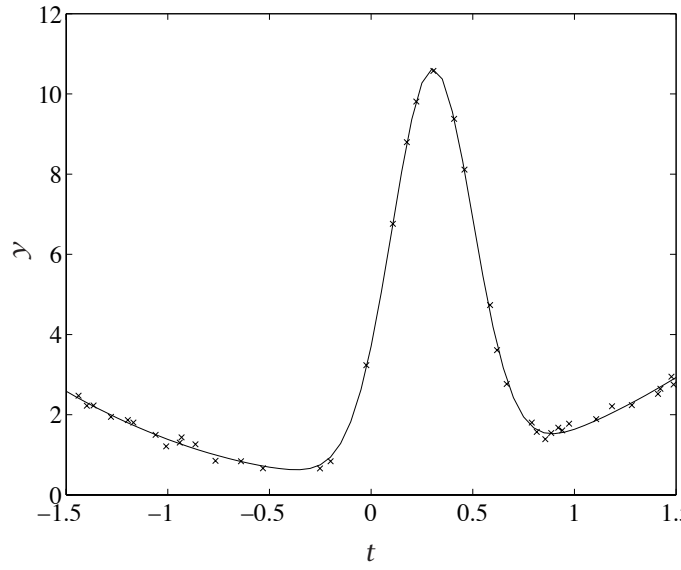
Esimerkki 7.1.2 Käytettävissä on mittausdata (t_i, y_i), $i = 1, \ldots, m$. Sovitetaan dataan seuraava malli:

$$y = f(t, \mathbf{a}) + \epsilon = a_1 + a_2 t + a_3 t^2 + a_4 e^{-(t-a_5)^2/(2a_6^2)} + \epsilon, \tag{7.5}$$

missä ϵ on virhetermi. Minimointitehtävään saadaan yhtälön (7.4) mukainen kohdefunktio $F(\mathbf{a})$, $\mathbf{a} \in \mathbb{R}^6$. Olkoot kaikkien mittauspisteiden painot samat: $w_i = 1, i = 1, \ldots, m$.

Listauksessa 7.1 on esitetty, miten minimointitehtävä ratkaistaan käyttäen Matlabin Optimization Toolboxia. Saatu malli ja mittausdatan pisteet on esitetty kuvassa 7.1. Rutiini `nlfd` laskee mallin ja datan erotuksen $y_i - f(t_i, \mathbf{a})$ ja rutiini `nlfun` laskee sovitettavan funktion arvot annetuilla parametrien $\mathbf{a}$ arvoilla pisteessä t.

Sovituksessa on käytetty Levenbergin ja Marquardtin menetelmää, joka on rutiinin `lsqnonlin` oletusarvoinen menetelmä. Myös Gaussin ja Newtonin menetelmää voi käyttää. Menetelmiä on kuvattu kappaleissa 7.3 ja 7.4.



Kuva 7.1: *Pienimmän neliösumman sovituksen tuottama malli (yhtenäinen viiva) ja käytetty mittausdata (merkitty symbolilla ×).*

Listaus 7.1: *Mallin sovittaminen dataan käyttäen Matlabia.*

```
% Sovitetaan m-tiedoston 'nlfd' malli dataan (t,y).
% Alkuarvaus:
a0 = 2*ones(1,6);

% Haetaan optimi rutiinilla 'lsqnonlin':
opt = optimset('lsqnonlin');
[res, optvalue] = lsqnonlin(@nlfd,a0,[],[],opt,t,y)

% Lasketaan mallin ja datan erotuksen neliosumma:
diff = sum(nlfd(res,t,y).^2);

% Piirretaan kuva:
tr = -1.5:0.05:1.5;
plot(t,y,'x',tr,nlfun(res,tr),'-')
xlabel('t'); ylabel('y')
```

```
function d = nlfd(a,t,y)
% Lasketaan mallin ja datan (t,y) erotus
d = y - nlfun(a,t);
```

```
function y = nlfun(a,t)
% Lasketaan mallin tuottamat arvot datalle t
y = a(1) + a(2).*t + a(3).*t.^2 + a(4).* ...
    exp(-(t-a(5)).^2/(2.*a(6).^2));
```

7.2 Ratkaisumenetelmät

Tarkastellaan yhtälössä (7.1) esitetyn funktion $F(\mathbf{x})$ minimoimista. Vektoriarvoisen funktion $\mathbf{f}(\mathbf{x}) : \mathbb{R}^n \mapsto \mathbb{R}^m$ Jacobin matriisi on

$$J(\mathbf{x}) = \begin{pmatrix} \nabla^T f_1(\mathbf{x}) \\ \vdots \\ \nabla^T f_m(\mathbf{x}) \end{pmatrix} = \begin{pmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial f_m(\mathbf{x})}{\partial x_n} \end{pmatrix}.$$

Funktion $F(\mathbf{x}) = \langle \mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x}) \rangle$ gradientiksi saadaan $\nabla F(\mathbf{x}) = 2J(\mathbf{x})^T \mathbf{f}(\mathbf{x})$ ja Hessen matriisiksi $H(\mathbf{x}) = 2J(\mathbf{x})^T J(\mathbf{x}) + 2S(\mathbf{x})$, missä matriisi S määritellään

$$S(\mathbf{x}) = \sum_{i=1}^{m} f_i(\mathbf{x}) \nabla \nabla f_i(\mathbf{x}).$$

Newtonin iteraatioksi epälineaarisen pienimmän neliösumman tehtävän ratkaisemiseksi saadaan

$$\left(J(\mathbf{x}^k)^T J(\mathbf{x}^k) + S(\mathbf{x}^k) \right) \mathbf{s}^k = -J(\mathbf{x}^k)^T \mathbf{f}(\mathbf{x}^k) \tag{7.6}$$

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{s}^k \tag{7.7}$$

Iteraatio on raskas, sillä matriisin $S(\mathbf{x}^k)$ laskemiseen tarvitaan $\frac{1}{2}mn(n+1)$ kappaletta toisia derivaattoja. Jos matriisin $S(\mathbf{x}^k)$ laskeminen jätetään pois iteraatiosta, saadaan *pienen residuaalin algoritmi*, joka soveltuu tapauksiin $|F(\mathbf{x}^*)| \approx 0$. Muuten saadaan *suuren residuaalin algoritmi*.

Termiä $S(\mathbf{x}^k)$ voi myös koettaa arvioida iteratiivisesti laskennan kuluessa sekanttimenetelmien tapaan.

Myös globaalin optimoinnin menetelmiä on käytetty vaikeiden PNS-tehtävien ratkaisemiseen. Esimerkiksi jäähdytysmenetelmiä (sivu 174), geneettisiä algoritmeja (sivu 162) ja neuroverkkoja [CU93] voi yrittää käyttää mallien sovittamiseen mittausdataan.

7.3 Gaussin ja Newtonin menetelmä

Pienen residuaalin menetelmiin kuuluvan Gaussin ja Newtonin menetelmän iteraatiossa ratkaistaan yhtälöryhmä

$$J(\mathbf{x}^k)^T J(\mathbf{x}^k)\, \mathbf{s}^k = -J(\mathbf{x}^k)^T \mathbf{f}(\mathbf{x}^k) \tag{7.8}$$

vektorin $\mathbf{s}^k$ suhteen. Tässä matriisi $J(\mathbf{x}^k)^T J(\mathbf{x}^k)$ on aina vähintään positiivisesti semidefiniitti. Menetelmä tarvitsee hyvän alkuarvauksen ja jos $\|S(\mathbf{x}^k)\|$ on suuri, algoritmi suppenee hitaasti ja voi törmätä singulariteetteihin. Usein korvataan askel $\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{s}^k$ viivahaulla suuntaan $\mathbf{s}^k$, jolloin suppeneminen tulee varmemmaksi.

Gaussin ja Newtonin menetelmä toimii luotettavimmin, mikäli $\|\mathbf{f}(\mathbf{x}^k)\| \to 0$, kun $\mathbf{x}^k \to \mathbf{x}^*$. Täten menetelmää voidaan käyttää mm. epälineaaristen yhtälöryhmien ratkaisemiseen.

Hakuaskeleen $\mathbf{s}^k$ ratkaisemiseen ei kannata käyttää suoraan kerroinmatriisia $J(\mathbf{x}^k)^T J(\mathbf{x}^k)$, sillä jos Jacobin matriisi $J(\mathbf{x}^k)$ on singulaarinen tai häiriöaltis, on tulo vielä häiriöalttiimpi. Eräs stabiili tapa ratkaista yhtälöryhmä on tehdä singulaariarvohajotelma

$$J(\mathbf{x}^k) = U_k \Sigma_k V_k^T, \tag{7.9}$$

missä matriisit U_k ja V_k ovat ortogonaalimatriiseja ja lävistäjämatriisi $\Sigma_k = \operatorname{diag}(\sigma_1, \ldots, \sigma_n)$ sisältää Jacobin matriisin singulaariarvot. Ortogonaalimatriisille U_k pätee $U_k^{-1} = U_k^T$ ja vastaavasti matriisille V_k. Määritellään lävistäjämatriisin $D_k = \operatorname{diag}(d_1, \ldots, d_n)$ alkiot seuraavasti:

$$\begin{cases} d_i = 1/\sigma_i, & \text{jos } \sigma_i \neq 0, \\ d_i = 0, & \text{jos } \sigma_i = 0. \end{cases}$$

Olkoon $J_k^+ = V_k D_k U_k^T$. Nyt pätee $J_k^+ J(\mathbf{x}^k) = I$ eli matriisi J_k^+ on Jacobin matriisin *pseudoinverssi*. Askel $\mathbf{s}^k$ saadaan ratkaistua asettamalla

$$\mathbf{s}^k = -J_k^+ \mathbf{f}(\mathbf{x}^k). \tag{7.10}$$

Etuna singulaariarvohajotelman käytössä on se, että ratkaisu löytyy vaikka Jacobin matriisi olisi singulaarinen. Toisaalta hajotelman laskeminen vaatii melko paljon työtä.

Gaussin ja Newtonin menetelmä löytyy mm. Matlabin Optimization Toolboxista ja NAG-aliohjelmakirjastosta.

7.4 Levenbergin ja Marquardtin menetelmä

Myös luottamusaluetta voi käyttää Gaussin ja Newtonin menetelmän suppenemisen varmistamiseksi. Tähän ideaan perustuva Levenbergin ja Marquardtin menetelmä on käytetyimpiä menetelmiä epälineaaristen pienimmän neliösumman tehtävien ratkaisemisessa. Menetelmässä ratkaistaan askel $\mathbf{s}^k$ yhtälöryhmästä

$$\left(J(\mathbf{x}^k)^T J(\mathbf{x}^k) + \mu_k D_k\right) \mathbf{s}^k = -J(\mathbf{x}^k)^T \mathbf{f}(\mathbf{x}^k) \tag{7.11}$$

missä D_k on lävistäjämatriisi, jonka alkiot ovat positiivisia, esimerkiksi identiteettimatriisi I.

Jos parametri μ on suuri, lähestyy vektorin $\mathbf{s}^k$ suunta kohdefunktion gradienttivektorin suuntaa, jolloin kyseessä on jyrkimmän laskun menetelmä. Jos μ on pieni, on menetelmä lähellä Gaussin ja Newtonin menetelmää. Eräs tapa valita μ_k on minimoida $F(\mathbf{x}^k + \mathbf{s}^k)$, vaikkakin käytännössä tehokkaimmaksi on osoittautunut adaptiivinen menettely, jossa tarvittaessa kasvatetaan parametrin μ_k arvoa iteraatio iteraatiolta.

Algoritmi 7.4.1 (Levenbergin ja Marquardtin menetelmä)

valitse alkuarvaus $\mathbf{x}^1$ ja suppenemiskriteeri $\epsilon > 0$
aseta $\mu_1 = 0.01$ ja $\nu = 10$ sekä $k = 1$
repeat
  aseta $\mu_k = \mu_k / \nu$
  repeat
    ratkaise askel $\mathbf{s}^k$ yhtälöstä (7.11)
    aseta $\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{s}^k$
    if $F(\mathbf{x}^{k+1}) > F(\mathbf{x}^k)$
      aseta $\mu_k = \nu \mu_k$
    end
  until $F(\mathbf{x}^{k+1}) < F(\mathbf{x}^k)$
  aseta $\mu_{k+1} = \mu_k$ ja $k = k + 1$
until $\|J(\mathbf{x}^{k+1})^T \mathbf{f}(\mathbf{x}^{k+1})\| \leq \epsilon$

Menetelmän saa tehokkaammaksi ja varmemmaksi parantamalla parametrin μ_k muuttamiseen liittyvää logiikkaa. Algoritmi kuuluu Netlibistä löytyviin Minpack-rutiineihin ja sisältyy myös esimerkiksi IMSL-aliohjelmakirjastoon. Se löytyy myös *Numerical Recipes* -teoksista [PTVF92a, PTVF92b], mutta niissä esitetty versio ei ole kaikissa tilanteissa stabiili.

7.5 Ohjelmistoja

Epälineaaristen pienimmän neliösumman tehtävien perusohjelmistoksi voi katsoa Netlibistä löytyvän Minpackin, jonka rutiineja käytetään monissa muissakin ohjelmistoissa. Myös IMSL- ja NAG-aliohjelmakirjastot sisältävät rutiineja epälineaaristen pienimmän neliösumman tehtävien ratkaisemiseen.

Matlabin Optimization Toolbox tarjoaa mahdollisuuden pienten ja keskisuurten optimointitehtävien ratkaisemiseen (taulukko B.3 sivulla 209). Epälineaaristen pienimmän neliösumman tehtävien ratkaisurutiinia `lsqnonlin` on käytetty esimerkissä 7.1.2.

IMSL:n ja NAGin käyttöä pienimmän neliösumman probleemojen ratkaisemiseen on esitelty käsikirjoissa [IMS, Num]. Myös Matlabin Optimization Toolboxin käyttöön löytyy käsikirja [Gra93].

Mathematica-ohjelmiston paketti `NonlinearFit.m` käyttää Levenbergin ja Marquardtin menetelmää epälineaariseen datan sovitukseen. Tietysti voi yrittää käyttää myös tavallisia optimointiohjelmistoja, esimerkiksi Minosta. Lisäksi tehtäviä voi ratkaista GAMS-mallinnuskielellä. Myös Netlibin TOMS-algoritmeista löytyy käyttökelpoisia rutiineja (taulukko B.6 sivulla 214).

Tämän oppaan luvussa 11 sivulla 180 on ratkaistu epälineaarinen PNS-tehtävä käyttäen jäähdytysmenetelmään perustuvaa ohjelmistoa.

7.6 Lisätietoja

Teos *Practical Optimization* [GMW81] tarjoaa käytännöllisen ja perusteellisen johdatuksen epälineaariseen optimointiin. Teoksia *Numerical Methods for Unconstrained Optimization and Nonlinear Equations* [DS83] ja *Introduction to Non-linear Optimization* [Sca85] voi käyttää lähdeteoksina PNS-tehtävien ratkaisemisessa. CSC:n oppaissa *Numeeriset menetelmät* [HKR93] ja *Datan käsittely* [Kar95] on kuvattu lineaaristen pienimmän neliösumman tehtävien ratkaisemista.

Suurten PNS-tehtävien ratkaisemiseen pätevät pitkälle samantyyppiset huomautukset kuin isoille optimointitehtäville yleensä (kappale 2.5 sivulla 42). Rajoiteellisten PNS-tehtävien ratkaisua on käsitelty artikkelissa [MAB89]. Katso myös lukuja 6 ja 8.

8 Rajoitteelliset epälineaariset optimointitehtävät

Tässä luvussa käsitellään jatkuvasti differentioituvien rajoitteellisten optimointitehtävien ratkaisumenetelmiä ja ohjelmistoja. Lisäksi esitellään joitakin tehtävätyyppejä ja niiden ratkaisemista.

8.1 Rajoitteita sisältävät tehtävät

Seuraavassa käsitellään rajoitteita sisältävien jatkuvasti differentioituvien optimointitehtävien ratkaisumenetelmiä. Tehtävien yleinen muoto on

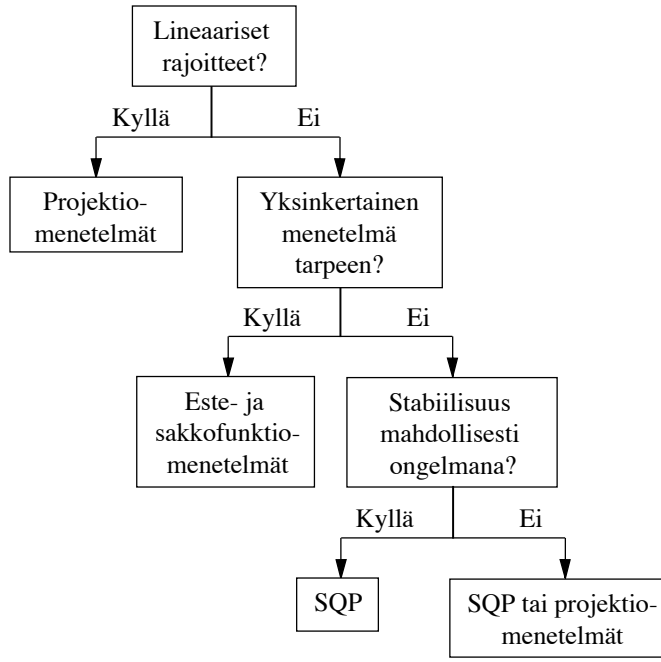
$$\underset{\mathbf{x}}{\text{minimi}}\ f(\mathbf{x}),\ \text{kun}\ \mathbf{g}(\mathbf{x}) \leq \mathbf{0}\ \text{ja}\ \mathbf{h}(\mathbf{x}) = \mathbf{0}, \tag{8.1}$$

missä vektoriarvoisten rajoitefunktioiden $\mathbf{g}$ ja $\mathbf{h}$ määritelmät ovat $\mathbf{g}(\mathbf{x}) = (g_1(\mathbf{x}), \ldots, g_p(\mathbf{x}))^T$ ja $\mathbf{h}(\mathbf{x}) = (h_1(\mathbf{x}), \ldots, h_q(\mathbf{x}))^T$. Rajoitteiden määrittelemää aluetta $\mathcal{F} = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{g}(\mathbf{x}) \leq \mathbf{0},\ \mathbf{h}(\mathbf{x}) = \mathbf{0}\}$ kutsutaan *käyväksi alueeksi* (feasible region). Tässä luvussa oletetaan, että kohdefunktio f ja rajoitefunktiot $\mathbf{g}$ ja $\mathbf{h}$ ovat jatkuvasti differentioituvia eli sileitä.

Rajoitteellisten optimointitehtävien (constrained optimization) taustaa on kuvattu sivulla 57. Rajoitteettomien tehtävien ratkaisua on käsitelty luvussa 6 sivulta 99 alkaen. Epälineaarisia pienimmän neliösumman tehtäviä käsitellään luvussa 7 (sivu 127).

Rajoitteellisessa optimoinnissa on merkittävä ero sillä, ovatko rajoitteet lineaarisia vai eivät. Lineaaristen rajoitteiden tapauksessa voidaan käyttää melko tehokkaita *projektiomenetelmiä* tai *aktiivijoukkomenetelmiä*. Epälineaaristen rajoitteiden tapauksessa muutetaan optimointitehtävä jonoksi rajoitteettomia tai lineaarisia rajoitteita sisältäviä tehtäviä. Käyttökelpoisia menetelmiä ovat mm. *täydennetyn Lagrangen funktion menetelmä*, *sakko- ja estefunktiomenetelmät* sekä yleisesti käytetty *toistettu kvadraattinen optimointi*, jossa optimointitehtävä palautetaan jonoksi kvadraattisia optimointitehtäviä. Osa menetelmistä pysyy käyvän alueen sisällä, osa voi muodostaa myös käyvän alueen ulkopuolella olevia iteraatiopisteitä.

Kuvassa 8.1 on yksinkertainen kaavio rajoitteelliseen optimointitehtävään sopivan ratkaisumenetelmän valitsemiseksi. Esimerkiksi IMSL- ja NAG-aliohjelmakirjastoista löytyy testattuja koodeja pienistä keskisuuriin optimointitehtäviin. Näiden ohjelmistojen käsikirjat sisältävät kaavioita, joista voi ongelmatyypin perusteella lähteä etsimään oikeaa menetelmää.



Kuva 8.1: *Yksinkertainen kaavioesitys rajoitteita sisältävien optimointitehtävien ratkaisumenetelmien soveltuvuudesta erityyppisiin tehtäviin.*

8.1.1 Esimerkkitehtävät

Tämä luku sisältää yksinkertaisia esimerkkejä rajoitteita sisältävistä optimointitehtävistä. Ratkaisuun käytetään GAMS-mallinnuskieltä sekä valmisohjelmistoja IMSL, NAG, Lapack ja Minos. Lineaaristen ja kvadraattisten optimointitehtävien ratkaisemista on esitelty erikseen luvussa 4 (sivu 74).

Epälineaarisia rajoitteita sisältäviä optimointitehtäviä havainnollistetaan käyttäen GAMSia, Minosta ja NAGia. Menetelminä ovat mm. toistettu kvadraattinen optimointi sekä sakko- ja estefunktioalgoritmit. Lisäksi vertaillaan eräiden ohjelmistojen tehokkuutta tehtävien ratkaisemisessa.

8.1.2 GAMS-ohjelmiston käyttö

Seuraavassa esimerkissä käytetään GAMS-ohjelmistoa epälineaarisia rajoitteita sisältävän tehtävän kuvaamiseen ja ratkaisemiseen. Ratkaisumoduulina GAMS käyttää Minosta.

Esimerkki 8.1.1 Tölkin materiaalin minimointitehtävä on esitetty esimerkissä 2.1.1 sivulla 33:

$$\underset{\mathbf{x}}{\text{minimi}}\ f(\mathbf{x}) = x_1^2 + x_1 x_2,\ \text{kun}\ -x_1^2 x_2 + 300/\pi \le 0 \text{ ja } \mathbf{x} \ge 0.$$

Tehtävää kuvaava GAMS-malli löytyy listauksesta 8.1. Muuttujat x_1 ja x_2 on määritelty positiivisiksi, alkuarvauksena on $(1,1)^T$. Ratkaisemalla malli GAMSilla saadaan tölkin optimaaliseksi säteeksi $x_1 \approx 3.6\,\text{cm}$ ja korkeudeksi $x_2 \approx 7.3\,\text{cm}$. Tulosta kannattaa verrata tavallisen virvoitusjuomatölkin mittoihin!

Listaus 8.1: *Esimerkin 8.1.1 tehtävän ratkaiseva GAMS-malli.*

```
FREE VARIABLE fx kohdefunktio;
POSITIVE VARIABLES x1, x2;
SCALAR Pi; Pi = 4.0*ARCTAN(1.0);

EQUATIONS Func, EQ;
FUNC.. fx =E= SQR(x1) + x1*x2;
EQ.. -SQR(x1)*x2 + 300/Pi =L= 0;
x1.l = 1; x2.l = 1;

MODEL Tolkki / ALL /;
SOLVE Tolkki MINIMIZING fx USING NLP;
```

8.2 Lineaariset rajoitteet

Olkoon ratkaistavana jompikumpi lineaarisia rajoitteita sisältävistä tehtävistä

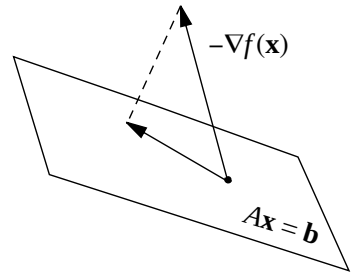
$$\underset{\mathbf{x}}{\text{minimi}}\ f(\mathbf{x}), \quad A\mathbf{x} = \mathbf{b}, \tag{8.2}$$

$$\underset{\mathbf{x}}{\text{minimi}}\ f(\mathbf{x}), \quad A\mathbf{x} \le \mathbf{b}, \tag{8.3}$$

missä A on $m \times n$ -matriisi.

Lineaaristen rajoitteiden tapauksessa joudutaan hakusuunnat rajaamaan rajoitteiden määrittelemään käypään alueeseen (tai aliavaruuteen), mutta muuten monet menetelmät ovat lähellä rajoitteettomien optimointitehtävien menetelmiä:

- Lineaarisille yhtälörajoitteille käytetään usein *projisoidun gradientin menetelmää* eli muodostetaan matriisi Z, jonka avulla voidaan projisoida hakusuunnat yhtälörajoitteiden määrittelemään avaruuteen.
- Lineaarisille epäyhtälörajoitteille käytetään mm. *aktiivijoukkomenetelmiä*, joissa haetaan yhtälömuodossa toteutuvat rajoitteet ja tämän jälkeen käytetään esimerkiksi projektiomenetelmiä.



Kuva 8.2: *Projektiomenetelmien käyttö lineaaristen yhtälörajoitteiden $A\mathbf{x} = \mathbf{b}$ tapauksessa.*

8.2.1 Lineaariset yhtälörajoitteet

Rajoitteettomien tehtävien algoritmeista voidaan johtaa menetelmiä lineaaristen rajoitteiden tapauksiin. Rajoitteen $A\mathbf{x} = \mathbf{b}$ toteuttavat pisteet muodostavat avaruuden $\mathbb{R}^n$ aliavaruuden (oletetaan että $m \times n$ -matriisin A rivit ovat toisistaan riippumattomat). Olkoon Z kanta matriisin A ytimelle (null space) eli $AZ = 0$. Tällöin pätee $\mathbf{s}^k = Z\mathbf{y}^k$ jollekin vektorille $\mathbf{y}^k \in \mathbb{R}^{n-m}$. Oletetaan että nykyinen iteraatiopiste $\mathbf{x}^k$ toteuttaa ehdon $A\mathbf{x}^k = \mathbf{b}$. Nyt saadaan hakusuunnalle $\mathbf{s}^k$ lauseke

$$A(\mathbf{x}^k + \mathbf{s}^k) = A(\mathbf{x}^k + Z\mathbf{y}^k) = A\mathbf{x}^k + AZ\mathbf{y}^k = A\mathbf{x}^k = \mathbf{b}.$$

Siis valitsemalla suuntia matriisin A ytimestä pysytään yhtälörajoitteiden määrittelemässä aliavaruudessa. Kohdefunktiolle f voidaan muodostaa kvadraattinen malli käyttäen projektiomatriisia Z:

$$f(\mathbf{x}^k + Z\mathbf{y}^k) \approx f(\mathbf{x}^k) + \langle \mathbf{y}^k, Z^T \nabla f(\mathbf{x}^k) \rangle + \frac{1}{2} \langle \mathbf{y}^k, Z^T H(\mathbf{x}^k) Z\mathbf{y}^k \rangle. \tag{8.4}$$

Projisoidun gradientin menetelmissä käytetään varsinaisena optimointialgoritmina esimerkiksi Newtonin menetelmää. Rajoitteettoman optimoinnin tapauksessa pätee yhtälö $H(\mathbf{x}^k)\mathbf{s}^k = -\nabla f(\mathbf{x}^k)$. Yhtälöstä (8.4) saadaan johdettua *projisoidut Newtonin yhtälöt*, joissa hakusuunnat $\mathbf{s}^k$ muodostetaan projektiomatriisin Z avulla:

$$Z^T H(\mathbf{x}^k) Z\, \mathbf{y}^k = -Z^T \nabla f(\mathbf{x}^k), \tag{8.5}$$

$$\mathbf{s}^k = Z\mathbf{y}^k. \tag{8.6}$$

Kyseessä on *projisoitu Newtonin menetelmä*. Tässä $H(\mathbf{x}^k)$ on kohdefunktion f Hessen matriisi $\nabla\nabla f(\mathbf{x}^k)$. Vektoria $Z^T \nabla f(\mathbf{x}^k)$ kutsutaan lineaarisesia rajoitteita sisältävän tehtävän *projisoiduksi gradientiksi* ja matriisia $Z^T H(\mathbf{x}^k) Z$ *projisoiduksi Hessen matriisiksi*.

Projisoidun Newtonin menetelmän toimintaa on havainnollistettu kuvassa 8.2. Gradienttivektori projisoidaan käyvälle alueelle yhtälöiden (8.5) ja (8.6) avulla.

Matriisin A ytimen kantavektoreista koostuva matriisi Z voidaan muodostaa QR-hajotelman avulla seuraavasti:

Algoritmi 8.2.1 (Projektiomatriisin muodostaminen)

1: Tehdään $m \times n$ -matriisille A hajotelma

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

missä Q on $n \times n$ -ortogonaalimatriisi ja R on $m \times m$ -yläkolmiomatriisi.

2: Olkoon $Q = (Q_1 \quad Q_2)$, missä $Q_1 \in \mathbb{R}^{n \times m}$ ja $Q_2 \in \mathbb{R}^{n \times (n-m)}$.

3: Matriisiksi Z voidaan valita ortogonaalimatriisi Q_2.

QR-hajotelman sijaan voi käyttää TQ-hajotelmaa

$$AQ = \begin{pmatrix} 0 & T \end{pmatrix}, \tag{8.7}$$

missä matriisi A on kooltaan $m \times n$. Matriisin Q koko on $n \times n$ ja matriisi T on $m \times m$ -kolmiomatriisi tyyppiä

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

Matriisiksi Z voidaan valita Q-matriisin $n - m$ ensimmäistä saraketta.

Esimerkki 8.2.1 Olkoon ratkaistavana minimointitehtävä

$$\underset{\mathbf{x} \in \mathbb{R}^6}{\text{minimi}}\ f(\mathbf{x}),\ \text{kun}\ \begin{cases} x_1 + x_3 + x_5 = 1, \\ x_2 + x_4 + x_6 = 2. \end{cases}$$

Siis rajoitteet ovat muotoa $A\mathbf{x} = \mathbf{b}$ ja

$$A = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

Listauksen 8.2 Fortran-ohjelma laskee QR-hajotelman matriisille A Lapack-aliohjelmakirjaston rutiineilla `sgeqrf` ja `sorgqr`. Hajotelma lasketaan matriisin A päälle. Matriisista $Q = (Q_1,\ Q_2)$ saadaan projektiomatriisiksi

$$Z = Q_2 = \begin{pmatrix} -0.577350 & 0 & -0.577350 & 0 \\ 0 & -0.577350 & 0 & -0.577350 \\ 0.788675 & 0 & -0.211325 & 0 \\ 0 & 0.788675 & 0 & -0.211325 \\ -0.211325 & 0 & 0.788675 & 0 \\ 0 & -0.211325 & 0 & 0.788675 \end{pmatrix}.$$

Pätee $Z^T Z = I$ eli Z on ortogonaalimatriisi kuten pitikin. Projektiomatriisia Z voidaan käyttää ratkaisemaan optimointitehtävä esimerkiksi projisoidulla Newtonin menetelmällä.

Listaus 8.2: *QR-hajotelman laskeminen Lapack-aliohjelmakirjastolla.*

```
PROGRAM QR_test
  IMPLICIT NONE
  INTEGER :: m, n, lwork, i, j, info
  REAL, DIMENSION(:,:), ALLOCATABLE :: a
  REAL, DIMENSION(:), ALLOCATABLE :: tau, work
  CHARACTER*(*), PARAMETER :: form = '(6(2x,f8.6))'

  m = 6; n = 2; lwork = 2*m
  ALLOCATE(a(m,m), tau(MIN(m,n)), work(lwork))
  a = 0.d0
  a(:,1) = (/ 1.d0, 0.d0, 1.d0, 0.d0, 1.d0, 0.d0 /)
  a(:,2) = (/ 0.d0, 1.d0, 0.d0, 1.d0, 0.d0, 1.d0 /)
  CALL sgeqrf(m, n, a, m, tau, work, lwork, info)
  CALL sorgqr(m, m, MIN(m,n), a, m, tau, work, lwork, info)
  WRITE (*,form) a
END PROGRAM QR_test
```

8.2.2 Lineaariset epäyhtälörajoitteet

Epäyhtälörajoitteita sisältävä optimointitehtävä

$$\underset{\mathbf{x}}{\text{minimi}}\ f(\mathbf{x}),\ \text{kun}\ A\mathbf{x} \leq \mathbf{b}, \tag{8.8}$$

voidaan muuntaa yhtälömuotoon lisäämällä positiiviset muuttujat $\mathbf{z} \geq \mathbf{0}$, jolloin saadaan $A\mathbf{x}+\mathbf{z} = \mathbf{b}$. Tehtävä voidaan ratkaista aktiivijoukkomenetelmillä (active set methods). Näissä menetelmissä hyödynnetään rajoitteita, jotka ovat voimassa yhtäsuuruusmuodossa. Tällöin voidaan muodostaa matriisi $\widetilde{A}$ niistä matriisin A riveistä, joille pätee yhtäsuuruus nykyisessä pisteessä $\mathbf{x}^k$. Tällöin pätee $\widetilde{A}\mathbf{x}^k = \widetilde{\mathbf{b}}$, missä $\widetilde{\mathbf{b}}$ sisältää vektorin $\mathbf{b}$ aktiiviset rivit.

Tämän jälkeen voidaan etsiä hakusuuntia matriisin $\widetilde{A}$ ytimen muodostamassa aliavaruudessa käyttämällä projektiomenetelmiä (ks. edellinen kappale). Kun hakusuunta on löytynyt, täytyy laskea, kuinka pitkälle kyseiseen suuntaan voidaan mennä siten, että pysytään käyvällä alueella.

Menetelmät siis generoivat peräkkäisiä aktiivisten rajoitteiden määräämiä optimointitehtäviä. Jos törmätään johonkin aktiivijoukon ulkopuolella olevaan rajoitteeseen, otetaan tämä mukaan matriisiin $\widetilde{A}$ ja tarvittaessa poistetaan vanhoja rajoitteita.

Tunnetuin aktiivijoukkomenetelmä on tietenkin lineaaristen optimointitehtävien ratkaisemisessa käytetty simplex-algoritmi (luku 4).

Aktiivijoukkomenetelmät voivat olla tehottomia isoissa tehtävissä, sillä jos rajoitteita on paljon, on aktiivijoukkoa päivitettävä usein. Lisäksi aktiivisten rajoitteiden löytämisessä joudutaan käyttämään numeerisia arvioita, mikä voi heikentää menetelmien stabiilisuutta.

Esimerkki 8.2.2 Olkoon ratkaistavana tehtävä

$$\underset{\mathbf{x}}{\text{minimi}}\ f(\mathbf{x}) = x_1^4 + x_2^4 - 4x_1^2 + 4x_2, \text{ kun } \begin{cases} g_1(\mathbf{x}) = x_1 + 2x_2 - 4 \le 0, \\ g_2(\mathbf{x}) = -x_1 - x_2 + 2 \le 0. \end{cases}$$

Optimointitehtävää on havainnollistettu kuvassa 8.3. Kohdefuktion gradienttivektori ja Hessen matriisi ovat

$$\nabla f(\mathbf{x}) = \begin{pmatrix} 4x_1^3 - 8x_1 \\ 4x_2^3 + 4 \end{pmatrix}, \quad H(\mathbf{x}) = \begin{pmatrix} 12x_1^2 - 8 & 0 \\ 0 & 12x_2 \end{pmatrix}.$$

Kirjoitetaan rajoitteet muotoon $A\mathbf{x} \le \mathbf{b}$, jolloin saadaan

$$A = \begin{pmatrix} 1 & 2 \\ -1 & -1 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 4 \\ -2 \end{pmatrix}.$$

Olkoon alkuarvaus vektori $\mathbf{x}^1 = (1, 1)^T$. Tässä pisteessä on rajoitteiden rivi 2 voimassa yhtälömuodossa. Siis pätee $\widetilde{A} = (-1,\ -1)$. QR-hajotelmaa käyttämällä saadaan laskettua projektiomatriisi Z:

$$Z \approx \begin{pmatrix} -0.7071 \\ 0.7071 \end{pmatrix}.$$

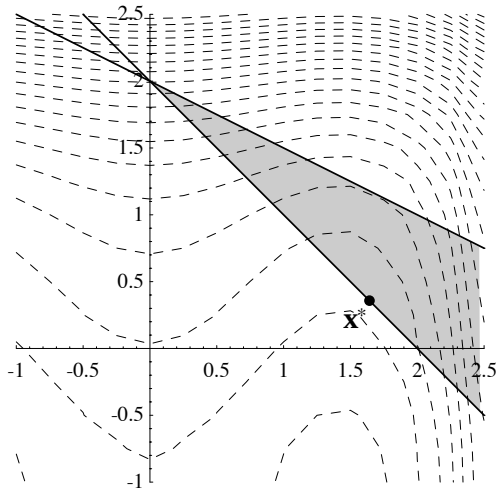
Lasketaan kohdefuntion f gradienttivektori ja Hessen matriisi:

$$\nabla f(\mathbf{x}^1) = \begin{pmatrix} -4 \\ 8 \end{pmatrix}, \quad H(\mathbf{x}^1) = \begin{pmatrix} 4 & 0 \\ 0 & 12 \end{pmatrix}.$$

Projisoiduiksi Newton-yhtälöiksi saadaan

$$\begin{pmatrix} -0.7071 \\ 0.7071 \end{pmatrix}^T \begin{pmatrix} 4 & 0 \\ 0 & 12 \end{pmatrix} \begin{pmatrix} -0.7071 \\ 0.7071 \end{pmatrix} \mathbf{y}^1 = -\begin{pmatrix} -0.7071 \\ 0.7071 \end{pmatrix}^T \begin{pmatrix} -4 \\ 8 \end{pmatrix},$$

$$\mathbf{s}^1 = \begin{pmatrix} -0.7071 \\ 0.7071 \end{pmatrix} \mathbf{y}^1,$$



Kuva 8.3: *Esimerkissä 8.2.2 käsiteltävä optimointitehtävä.*

mistä saadaan hakusuunnaksi $\mathbf{s}^1 = (0.7500, -0.7500)^T$ ja uudeksi iteraatiopisteeksi $\mathbf{x}^2 = \mathbf{x}^1 + \mathbf{s}^1 = (1.7500, 0.2500)^T$.

Pisteessä $\mathbf{x}^2$ on rajoite g_2 edelleen aktiivinen eli $g_2(\mathbf{x}^2) = 0$ ja lisäksi $g_1(\mathbf{x}^2) = -1.7500$ eli ei törmätty rajoitteeseen g_1. Jatkamalla edelleen saadaan askeleeksi $\mathbf{s}^2 = (-0.1144, 0.1144)^T$ ja iteraatiopisteeksi $\mathbf{x}^3 = (1.6356, 0.3644)^T$ sekä edelleen hakusuunnaksi $\mathbf{s}^3 = (-0.0087, 0.0087)^T$ ja iteraatiopisteeksi $\mathbf{x}^4 = (1.6269, 0.3731)^T$.

Optimikohdaksi saadaan $\mathbf{x}^* = (1.6268, 0.3732)^T$ ja minimiarvoksi $f(\mathbf{x}^*) = -2.07$. Tässä esimerkissä pysytään rajoitteen g_2 määräämässä aliavaruudessa.

8.3 Epälineaariset rajoitteet

Epälineaaristen rajoitteiden vallitessa vaihtoehtoisia menetelmiä on useita. *Toistettu kvadraattinen optimointi* (Sequential Quadratic Programming eli SQP) on varsin tehokas menetelmä, mutta vaatii runsaasti tarkistuksia suppenemisen varmistamiseksi. Toisaalta menetelmä soveltuu melko hankalienkin tehtävien ratkaisemiseen ja se lienee käytännössä merkittävin epälineaarisia rajoitteita sisältävien optimointitehtävien ratkaisumenetelmä. Menetelmästä on monia variaatioita, joista kehittyneimmät käyttävät esimerkiksi täydennettyä Lagrangen funktiota varmistamassa suppenemista.

Jos halutaan tehdä ratkaisurutiinin koodista yksinkertainen, *sakko- ja estefunktiomenetelmät* ovat paras vaihtoehto. Ongelmana on tehokkaan sakkoparametrien valintamekanismin löytäminen. Lisäksi estefunktiomenetelmillä on ongelmana Hessen matriisin häiriöalttius lähestyttäessä käyvän alueen reunalla olevaa minimikohtaa.

Täydennetyn Lagrangen funktion menetelmä on monissa tapauksissa sakkofunktiomenetelmiä luotettavampi. Täydennetyn Lagrangen funktion menetelmä on käyttökelpoinen, kun suuri osa rajoitefunktioista on lineaarisia ja voidaan löytää hyvä alkuarvaus.

Epälineaarisia rajoitteita sisältäviä tehtäviä voi ratkaista SQP-menetelmiä käyttävillä IMSL- ja NAG-aliohjelmakirjastojen rutiineilla (kappale 8.5). Minos-ohjelmiston käyttämä täydennetyn Lagrangen funktion menetelmä on luotettava ja menetelmästä on kehitetty suurten tehtävien ratkaisemiseen soveltuvia versioita.

Joissakin tapauksissa voi sakko- tai estefunktiomenetelmän sekä luotettavan rajoitteettomien optimointitehtävien ratkaisurutiinin käyttö osoittautua tehokkaaksi, mutta tällöin joudutaan etsimään oikeita sakkoparametreja suppenemisen varmistamiseksi. Kuitenkin suurille LP- ja kvadraattisille optimointitehtäville on kehitetty tehokkaita estefunktiota käyttäviä menetelmiä (luku 4). Näiden kokemusten perusteella on myös kehitteillä yleisten epälineaaristen tehtävien ratkaisumenetelmiä.

8.4 Täydennetyn Lagrangen funktion menetelmä

Monien optimointikoodien mallina ollut Minos-ohjelmisto perustuu *projektiomenetelmän* (ns. projected augmented Lagrangian) käyttöön [MS87]. Projektiomenetelmissä yritetään kulkea rajoitteiden määräämillä pinnoilla siten, että kohdefunktion arvo pienenee. Menetelmissä linearisoidaan rajoitteet nykyisessä hakupisteessä. Epälineaarisuus saa luonnollisesti aikaan sen, että lineaarista approksimaatiota joudutaan korjaamaan varsin usein, jolloin suppeneminen voi olla hidasta.

Minos-ohjelmistoa voi käyttää epälineaarisen kohdefunktion ja epälineaarisia rajoitefunktioita sisältävien tehtävien ratkaisemiseen. Ohjelmisto käsittelee epälineaarisia rajoitteita täydennetyn Lagrangen funktion ja projektiomenetelmien avulla. Minos on tehokkaimmillaan tehtävissä, joiden rajoitteista suurin osa on lineaarisia tai lähes lineaarisia. Esimerkiksi GAMS voi käyttää epälineaaristen tehtävien ratkaisemiseen Minos-ohjelmistoa (esimerkki 8.1.1).

Tarkastellaan epälineaarisesia rajoitteita sisältävää optimointitehtävää, jossa on pelkästään yhtälömuotoisia rajoitteita $\mathbf{h}(\mathbf{x}) = \mathbf{0}$. Vaihtoehtoisesti tiedetään nykyisessä pisteessä aktiivisten epäyhtälörajoitteiden joukko.

Määritellään *täydennetty Lagrangen funktio* (augmented Lagrangian):

$$F(\mathbf{x}) = f(\mathbf{x}) - \langle \mathbf{w}, \mathbf{h}(\mathbf{x}) \rangle + \frac{1}{2} \langle \mathbf{h}(\mathbf{x}), S(\mathbf{x})\mathbf{h}(\mathbf{x}) \rangle, \tag{8.9}$$

missä $S(\mathbf{x})$ on sopiva positiivisesti definiitti matriisi, esimerkiksi lävistäjämatriisi ρI. Vektori $\mathbf{w}$ on tehtävän Lagrangen kertoimien $(v_1, ..., v_n)^T$ arvio. Eräs tyypillinen täydennetyn Lagrangen funktion muoto on

$$F(\mathbf{x}) = f(\mathbf{x}) - \langle \mathbf{w}, \mathbf{h}(\mathbf{x}) \rangle + \frac{1}{2\mu} \langle \mathbf{h}(\mathbf{x}), \mathbf{h}(\mathbf{x}) \rangle. \tag{8.10}$$

Täydennettyä Lagrangen funktiota käyttämällä saadaan aikaan seuraava iteratiivinen algoritmi:

Algoritmi 8.4.1 (Täydennetyn Lagrangen funktion menetelmä)

valitse alkuarvaus $\mathbf{x}^1$ ja laske alkuarvio $\mathbf{w}^1$ Lagrangen kertoimille
valitse sopiva painomatriisi $S(\mathbf{x}^1)$
aseta $k = 1$
repeat
  hae täydennetyn Lagrangen funktion $F(\mathbf{x}^k)$ likimääräinen
    minimi $\mathbf{x}^{k*}$
  laske Lagrangen kertoimille arvio $\mathbf{w}^{k+1}$
  päivitä tarvittaessa painomatriisia $S(\mathbf{x}^{k+1})$
  aseta $k = k + 1$
until ollaan kyllin lähellä optimia

Algoritmissa lasketaan funktiot $L(\mathbf{x}^k)$ arvioimalla vektorilla $\mathbf{w}^k$ Lagrangen kertoimia $\mathbf{u}$ nykyisessä pisteessä. Mikäli $\mathbf{w}^k \to \mathbf{u}$, kun $k \to \infty$ ja Lagrangen funktiolla $L(\mathbf{x}^k)$ on äärellinen minimi, pätee $\mathbf{x}^{k*} \to \mathbf{x}^*$.

Eräs mahdollisuus vektorin $\mathbf{w}^k$ arvioksi on

$$\mathbf{w}^{k+1} = \mathbf{w}^k - S(\mathbf{x}^{k*})\,\mathbf{h}(\mathbf{x}^{k*}), \tag{8.11}$$

missä $S(\mathbf{x}^{k*})$ on sopiva painomatriisi, esimerkiksi ρI.

Minos-ohjelmistossa käytetään muotoa

$$F(\mathbf{x}) = f(\mathbf{x}) - \langle \mathbf{w}, \mathbf{h}(\mathbf{x}) - \widetilde{\mathbf{h}}(\mathbf{x})\rangle + \frac{1}{2}\rho\langle \mathbf{h}(\mathbf{x}) - \widetilde{\mathbf{h}}(\mathbf{x}), \mathbf{h}(\mathbf{x}) - \widetilde{\mathbf{h}}(\mathbf{x})\rangle \tag{8.12}$$

olevaa täydennettyä Lagrangen funktiota, missä ρ on painokerroin ja funktio $\widetilde{\mathbf{h}}$ sisältää nykyisessä iteraatiopisteessä $\mathbf{x}^k$ linearisoidut rajoitteet:

$$\widetilde{h}_i(\mathbf{x}) = h_i(\mathbf{x}^k) + \langle \nabla h_i(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k\rangle,\ i = 1, \ldots, q.$$

Täydennettyä Lagrangen funktiota F minimoitaessa Minos asettaa rajoitteiksi $\widetilde{\mathbf{h}}(\mathbf{x}) = \mathbf{0}$. Erikseen voidaan ottaa mukaan mahdolliset lineaariset rajoitteet sekä laatikkorajoitteet $\mathbf{x}^l \le \mathbf{x} \le \mathbf{x}^u$.

8.5 Toistettu kvadraattinen optimointi

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*.

Yhtälö- ja epäyhtälörajoitteita sisältävän epälineaarisen optimointitehtävän

$$\operatorname*{minimi}_{\mathbf{x}} f(\mathbf{x}),\ \text{kun } \mathbf{g}(\mathbf{x}) \le \mathbf{0} \text{ ja } \mathbf{h}(\mathbf{x}) = \mathbf{0} \tag{8.13}$$

Lagrangen funktio on

$$L(\mathbf{x}, \mathbf{v}) = f(\mathbf{x}) + \langle \mathbf{u}, \mathbf{g}(\mathbf{x})\rangle + \langle \mathbf{v}, \mathbf{h}(\mathbf{x})\rangle, \tag{8.14}$$

missä vektorit $\mathbf{u} \ge \mathbf{0}$ ja $\mathbf{v}$ sisältävät Lagrangen kertoimet. Lagrangen funktioon voidaan soveltaa Taylorin lausetta, jolloin saadaan

$$L(\mathbf{x}^k + \mathbf{p}^k, \mathbf{u}^k, \mathbf{v}^k) = \nabla L(\mathbf{x}^k, \mathbf{u}^k, \mathbf{v}^k) + \left(\nabla\nabla L(\mathbf{x}^k, \mathbf{u}^k, \mathbf{v}^k)\right)\mathbf{p}^k + \cdots, \tag{8.15}$$

missä ∇ tarkoittaa derivointia vektorin $\mathbf{x}$ suhteen. Jos Taylorin sarjakehitelmästä jätetään huomiotta korkeamman kertaluvun termit ja asetetaan $L(\mathbf{x}^k + \mathbf{p}^k, \mathbf{u}^k, \mathbf{v}^k) = 0$, saadaan ratkaistua hakuaskel $\mathbf{p}^k$. Merkitään nyt matriisilla W_k Hessen matriisia $\nabla\nabla L(\mathbf{x}^k, \mathbf{u}^k, \mathbf{v}^k)$. Saadaan yhtälö

$$W_k\,\mathbf{p}^k = -\nabla L(\mathbf{x}^k, \mathbf{u}^k, \mathbf{v}^k). \tag{8.16}$$

Toistetussa kvadraattisessa optimoinnissa ratkaistaan siten kussakin vaiheessa tehtävä

$$\operatorname*{minimi}_{\mathbf{p}} \frac{1}{2}\langle \mathbf{p}^k, W_k\mathbf{p}^k\rangle + \langle \nabla f(\mathbf{x}^k), \mathbf{p}^k\rangle,\ \text{kun } \mathbf{p}^k \in \mathbb{R}^n, \tag{8.17}$$

missä matriisi W_k on arvio Lagrangen funktion Hessen matriisille pisteessä $\mathbf{x}^k$. Kvadraattisen tehtävän rajoitteet saadaan linearisoimalla rajoitefunktiot $\mathbf{g}$ ja $\mathbf{h}$ nykyisessä pisteessä $\mathbf{x}^k$:

$$\begin{cases} g_i(\mathbf{x}^k) + \langle \nabla g_i(\mathbf{x}^k), \mathbf{p}^k \rangle \leq 0, \quad i = 1, \ldots, p, \\ h_j(\mathbf{x}^k) + \langle \nabla h_j(\mathbf{x}^k), \mathbf{p}^k \rangle = 0, \quad j = 1, \ldots, q. \end{cases} \tag{8.18}$$

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.

Seuraavan iteraatiopisteen löytämiseen voidaan käyttää viivahakua

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \lambda \mathbf{p}^k, \quad \lambda \in (0, 1], \tag{8.19}$$

esimerkiksi siten, että *ansiofunktio* (merit function) saa pienemmän arvon uudessa pisteessä. Ansiofunktiona voidaan käyttää esimerkiksi täydennettyä Lagrangen funktiota (kappale 8.4). Ansiofunktion käyttö varmistaa menetelmän suppenemista oltaessa kaukana minimipisteestä. Jos uusi piste ei ole minimi, voidaan Lagrangen funktion Hessen matriisin arviota W_k päivittää esimerkiksi BFGS-menetelmällä (algoritmi 6.6.1 sivulla 112).

Esimerkki 8.5.1 Ratkaistaan epälineaarinen optimointitehtävä

$$\underset{x_1, x_2}{\text{minimi}}\ (x_1 - 2)^2 + (x_2 - 1)^2, \text{ kun } \begin{cases} g(\mathbf{x}) = \frac{x_1^2}{4} + x_2^2 - 1 \leq 0, \\ h(\mathbf{x}) = x_1 - 2x_2 + 1 = 0, \end{cases}$$

missä kohdefunktio ja epäyhtälörajoite ovat epälineaarisia ja yhtälörajoite on lineaarinen. Olkoon alkuarvaus $\mathbf{x}^1 = (2, 2)^T$.

Ratkaistaan tehtävä NAG-aliohjelmakirjaston SQP-rutiinilla E04VCE. Listauksessa 8.3 on Fortran 90 -kielinen pääohjelma `sqpmain.f90` ja listauksessa 8.4 on rajoitteet ja niiden derivaatat laskeva ohjelmatiedosto `sqpfun.f90`. Ratkaisurutiini perustuu toistettuun kvadraattiseen optimointiin.

Kuten listauksesta 8.3 nähdään, tarvitsee NAGin SQP-toteutus melkoisen määrän aliohjelmakutsussa annettavia parametreja. Onneksi ohjelmiston avustusjärjestelmästä löytyy rutiinien käyttöä kuvaavia esimerkkiohjelmia.

NAG-rutiinin löytämä minimipiste on $\mathbf{x}^* \approx (0.8229, 0.9114)^T$ ja funktion arvo $f(\mathbf{x}^*) \approx 1.3935$. Epäyhtälörajoite $g(\mathbf{x}) \leq 0$ on aktiivinen optimipisteessä.

Seuraavassa esimerkissä havainnollistetaan SQP-menetelmän toimintaa yksityiskohtaisemmin.

Esimerkki 8.5.2 Olkoon ratkaistavana edellisestä esimerkistä hiukan muutettu optimointitehtävä

$$\underset{x_1, x_2}{\text{minimi}}\ f(\mathbf{x}) = (x_1 - 2)^2 + (x_2 - 1)^2, \begin{cases} g_1(\mathbf{x}) = x_1^2/4 + x_2^2 - 1 \leq 0, \\ g_2(\mathbf{x}) = x_1 - 2x_2 + 1 \leq 0. \end{cases}$$

Listaus 8.3: *Esimerkin 8.5.1 tehtävän NAG-aliohjelmakirjaston rutiinilla E04VCE ratkaiseva Fortran 90 -kielinen pääohjelma.*

```
PROGRAM SQP_test
  IMPLICIT NONE
  INTEGER :: n, nclin, ncnln, nctotl, nrowa, nrowj, nrowr, &
      liwork, lwork, i, ifail, iter, itmax, j, mode, msglvl, &
      nstate
  INTEGER, DIMENSION(:), ALLOCATABLE :: istate, iwork
  LOGICAL :: cold, fealin, ortho
  REAL :: bigbnd, epsaf, epsmch, eta, ftol, objf, rteps
  REAL, DIMENSION(:,:), ALLOCATABLE :: a, cjac, r
  REAL, DIMENSION(:), ALLOCATABLE :: x, bl, bu, c, clamda, &
      featol, objgrd, work, x02aje
  EXTERNAL x02aje, confun, e04vce, e04zce, objfun
  ! Asetetaan tehtävän dimensiot
  n = 2; nclin = 1; ncnln = 1; nctotl = n+nclin+ncnln
  nrowa = nclin; nrowj = ncnln; nrowr = n
  liwork = 3*n+nclin+2*ncnln
  lwork = 2*n*n+n*nclin+2*n*ncnln+20*n+11*nclin+21*ncnln
  ALLOCATE(a(nrowa,n), bl(nctotl), bu(nctotl), c(nrowj), &
      cjac(nrowj,n), clamda(nctotl), featol(nctotl), &
      objgrd(n), r(nrowr,n), work(lwork), x(n), &
      istate(nctotl), iwork(liwork))
  ! Alkuasetukset
  nstate = 1; mode = 1; ifail = 1; msglvl = 1
  bigbnd = 1.d10; itmax = 100; eta = 0.9d0
  x = (/ 2.d0,2.d0 /); a = RESHAPE((/1.d0,-2.d0/), (/1,2/))
  bl = (/ -10.d0,-5.d0,-1.d0,-1.d0 /)
  bu = (/ 10.d0,5.d0,-1.d0,1.d10 /)
  cold = .TRUE.; fealin = .TRUE.; ortho = .TRUE.
  ifail = 1  ! Tarkistetaan gradientit
  CALL e04zce(n, nclin, nrowj, confun, objfun, c, cjac, &
      objf, objgrd, x, work, lwork, ifail)
  IF (ifail == 0) THEN  ! Gradienttivektori on ok...
    epsmch = x02aje() ! Selvitetään laskentatarkkuus
    ftol = 10.0d0*epsmch; rteps = SQRT(epsmch)
    featol(1:nctotl) = rteps
    CALL objfun(mode, n, x, objf, objgrd, nstate)
    epsaf = epsmch*ABS(objf)
    ifail = 1  ! Ratkaistaan tehtävä
    CALL e04vce(itmax, msglvl, n, nclin, ncnln, nctotl, &
        nrowa, nrowj, nrowr, bigbnd, epsaf, eta, ftol, a, &
        bl, bu, featol, confun, objfun, cold, fealin, ortho, &
        x, istate, r, iter, c, cjac, objf, objgrd, clamda, &
        iwork, liwork, work, lwork, ifail)
    IF (ifail /= 0) WRITE(*,*) 'e04vce:n ifail =', ifail
  ELSE
    WRITE(*,*) 'Virhe gradienttivektorissa. ifail =', ifail
  END IF
END PROGRAM SQP_test
```

Tehtävää on havainnollistettu kuvassa 8.4. Kohdefunktion tasa-arvokäyrät on piirretty katkoviivoilla ja rajoitteita vastaava ellipsi ja suora yhtenäisellä viivalla. Rajoitteet ovat siis kummatkin epäyhtälömuotoa. Tehtävän Lagrangen funktioksi saadaan

$$L(\mathbf{x},\mathbf{u}) = (x_1-2)^2 + (x_2-1)^2 + u_1(x_1^2/4 + x_2^2 - 1) + u_2(x_1 - 2x_2 + 1).$$

Lagrangen funktion L gradienttivektori vektorin $\mathbf{x}$ suhteen on

$$\nabla L(\mathbf{x},\mathbf{u}) = \begin{pmatrix} 2(x_1-2) + u_1x_1/2 + u_2 \\ -2u_2 + 2(x_2-1) + 2u_1x_2 \end{pmatrix}$$

ja Hessen matriisi on

$$\nabla\nabla L(\mathbf{x},\mathbf{u}) = \begin{pmatrix} u_1/2 + 2 & 0 \\ 0 & 2u_1 + 2 \end{pmatrix}.$$

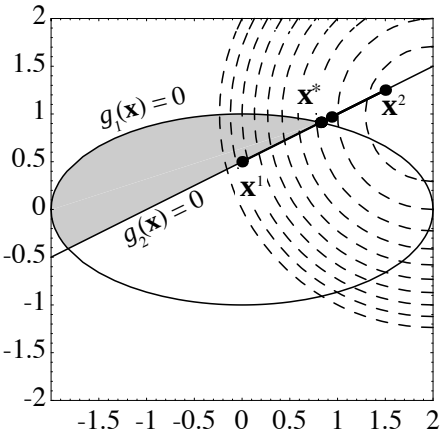
Ratkaisemisessa käytetään pisteessä $\mathbf{x}^k$ linearisoituja rajoitteita:

$$\begin{cases} (x_1^k)^2/4 + (x_2^k)^2 - 1 + \begin{pmatrix} x_1^k/2 \\ 2x_2^k \end{pmatrix}^T \mathbf{p}^k \le 0, \\ x_1^k - 2x_2^k + 1 + \begin{pmatrix} 1 \\ -2 \end{pmatrix}^T \mathbf{p}^k \le 0. \end{cases}$$

Annetaan vektorille $\mathbf{x}$ alkuarvaus $\mathbf{x}^1 = (0,\ 0.5)^T$ ja Lagrangen kertoimille alkuarvaus $\mathbf{u}^1 = (0,\ 0)^T$. Tällöin saadaan iteraatiopisteet

k	$\mathbf{x}^k$	$\mathbf{u}^k$	$\mathbf{g}(\mathbf{x}^k)$
1	$(0, 0.5)^T$	$(0, 0)^T$	$(-0.75, 0.)^T$
2	$(1.5, 1.25)^T$	$(1.5, 1.)^T$	$(1.125, 0.)^T$
3	$(0.9375, 0.9688)^T$	$(1.5, 1.422)^T$	$(0.1582, 0.)^T$
4	$(0.8274, 0.9137)^T$	$(1.806, 1.581)^T$	$(0.0061, 0.)^T$
5	$(0.8229, 0.9114)^T$	$(1.846, 1.594)^T$	$(0.0000, 0.)^T$

Tässä ei pakotettu iteraatteja $\mathbf{x}^k$ pysymään käyvän alueen sisällä ja viivahaku suuntaan $\mathbf{p}^k$ jätettiin tekemättä. Viidennellä iteraatiolla ollaan jo hyvin lähellä tehtävän optimikohtaa ja tällöin saadaan 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 8.4: *Esimerkissä 8.5.2 käsiteltävä optimointitehtävä.*

Listaus 8.4: *Fortran 90 -kieliset aliohjelmat* `objfun` *ja* `confun` *laskevat kohdefunktion arvon ja gradienttivektorit sekä epälineaaristen rajoitteiden arvot ja Jacobin matriisin rivit.*

```
SUBROUTINE objfun(mode,n,x,objf,objgrd,nstate)
  ! Kohdefunktio ja gradienttivektori
  IMPLICIT NONE
  REAL, INTENT(OUT) :: objf
  INTEGER, INTENT(IN) :: mode, n, nstate
  REAL, DIMENSION(n), INTENT(IN) :: x
  REAL, DIMENSION(n), INTENT(OUT) :: objgrd
  objf = (x(1)-2.0d0)**2 + (x(2)-1.0d0)**2
  objgrd(1) = 2.0d0*(x(1)-2.0d0)
  objgrd(2) = 2.0d0*(x(2)-1.0d0)
END SUBROUTINE objfun

SUBROUTINE confun(mode,ncnln,n,nrowj,x,c,cjac,nstate)
  ! Rajoitteet ja Jacobin matriisi
  IMPLICIT NONE
  INTEGER, INTENT(IN) :: mode, ncnln, n, nrowj, nstate
  REAL, DIMENSION(n), INTENT(IN) :: x
  REAL, DIMENSION(nrowj), INTENT(OUT) :: c
  REAL, DIMENSION(nrowj,n), INTENT(INOUT) :: cjac
  ! Alustetaan Jacobin matriisi
  IF (nstate == 1) cjac(1:ncnln,1:n) = 0.0d0
  ! Epälineaarinen rajoite 1
  c(1) = -(x(1)**2)/4.0d0 - x(2)**2
  ! Jacobin matriisin rivi 1
  cjac(1,1) = -0.5d0*x(1)
  cjac(1,2) = -2.0d0*x(2)
END SUBROUTINE confun
```

8.6 Sakko- ja estefunktiomenetelmät

Sakko- ja estefunktiomenetelmissä (penalty and barrier function methods) muutetaan rajoitteita sisältävä optimointitehtävä rajoitteettomaksi optimointitehtäväksi, jossa kohdefunktioon on lisätty sopivalla sakkotermillä painotettuna rajoitteen rikkomisesta tuleva kustannus.

Sakko- ja estefunktiomenetelmät, varsinkin logaritminen estefunktio, ovat osoittautuneet erittäin lupaaviksi menetelmiksi sisäpistemenetelmien rakennuspalikkoina. Estefunktiomenetelmät pysyttelevät käyvän alueen sisällä ja lähestyvät alueen reunalla olevaa minimiä asymptoottisesti.

Sakko- ja estefunktiomenetelmissä on minimoitavana funktio

$$P(\mathbf{x}, s) = f(\mathbf{x}) + s \sum_j G(g_j(\mathbf{x})), \tag{8.20}$$

missä funktion $G : \mathbb{R} \mapsto \mathbb{R}$ avulla painotetaan rajoitefunktion arvoja. Skalaari $s > 0$ on funktion $P(\mathbf{x}, s)$ *sakkoparametri*.

Menetelmässä voidaan yhdistellä erityyppisiä sakkotermejä erityyppisille rajoitteille. Edellä olivat mukana vain epäyhtälörajoitteet $\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$.

Sakko- ja estefunktioiden käyttämiä muunnosfunktioita $G(g_j(\mathbf{x}))$ on seuraavanlaisia:

- *Sakkofunktiot* (eli *ulkopuoliset sakkofunktiot*):

$$\begin{cases} G(g_j(\mathbf{x})) > 0, & \text{kun } g_j(\mathbf{x}) > 0, \\ G(g_j(\mathbf{x})) = 0, & \text{kun } g_j(\mathbf{x}) \leq 0. \end{cases}$$

  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 $s > \tilde{s}$.

  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 8.6.1 Olkoon optimointitehtävänä

$$\underset{\mathbf{x}}{\text{minimi}}\ f(\mathbf{x}) = 10 - 20\left((x_1 + 1.5)^2 + 2x_2^2\right),$$

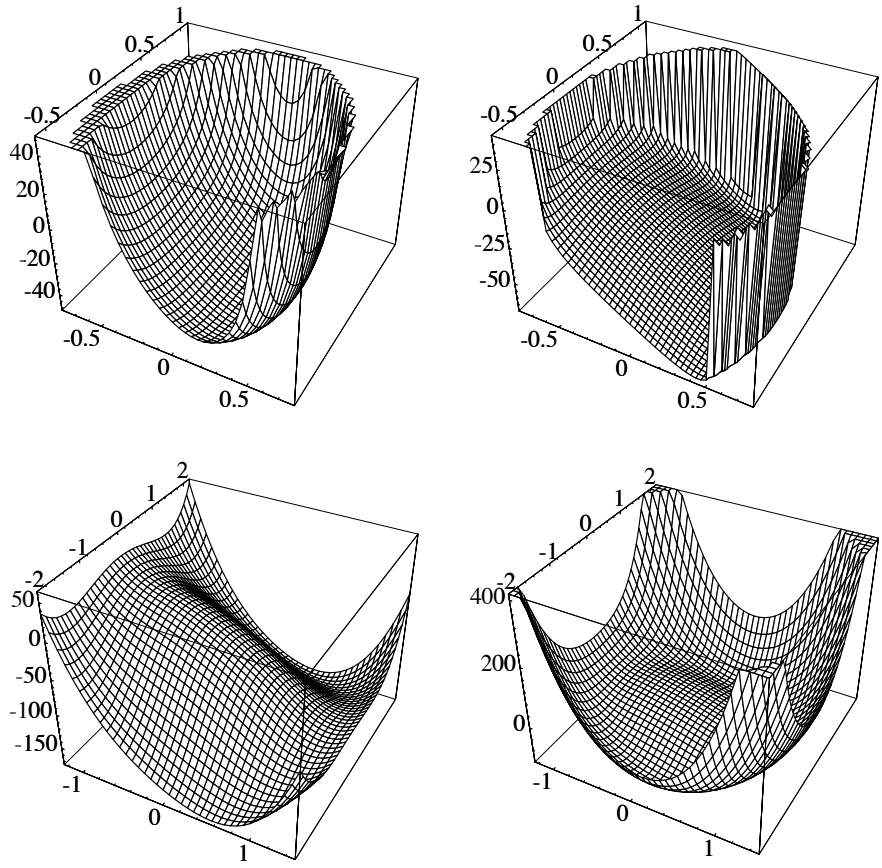
kun rajoitteena on $g(\mathbf{x}) = 2x_1^2 + x_2^2 - 1 \leq 0$.

Kuvassa 8.5 on esitetty logaritmisen estefunktion ja neliöllisen sakkofunktion käyttäytyminen kahdella eri sakkoparametrin arvolla. Estefunktion parametrin s lähestyessä nollaa jyrkkenee estefunktion reuna ja lähestytään asymptoottisesti rajoitteellisen tehtävän optimikohtaa käyvän alueen sisältä käsin. Sakkofunktion parametrin s kasvaessa lähestytään käyvän alueen ulkopuolelta käsin rajoitteellisen tehtävän optimikohtaa.

Seuraavassa esitetään logaritmista estefunktiota

$$P(\mathbf{x}, s) = f(\mathbf{x}) - s\sum_j \ln(-g_j(\mathbf{x})) \tag{8.21}$$

käyttävä optimointialgoritmi yleisessä muodossa. Ratkaistavana on epäyhtälörajoitteita $\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$ sisältävä optimointitehtävä.



Kuva 8.5: *Ylempänä on kuvattu logaritmisen estefunktion käyttäytymistä kahdella eri sakkoparametrin arvolla. Alempana on käytetty 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). Funktion arvot lähestyvät ääretöntä käyvän alueen reunalla. 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).*

Algoritmi 8.6.1 (Estefunktiomenetelmä)

1: Hae käypä alkuarvaus $\mathbf{x}^1$, jossa pätee $\mathbf{g}(\mathbf{x}^1) < \mathbf{0}$. Valitse $s_1 > 0$ ja aseta $k = 1$.

2: Jos piste $\mathbf{x}^k$ on rajoitteellisen optimointitehtävän lokaali minimi, lopeta iterointi.

3: Hae minimipiste $\mathbf{x}^*$ estefunktiolle $P(\mathbf{x}, s_k)$ siten, että $\mathbf{x}^*$ on käyvän alueen $\mathcal{F}$ sisäpiste.

4: Aseta uudeksi iteraatiopisteeksi $\mathbf{x}^{k+1} = \mathbf{x}^*$. Valitse $s_{k+1} \in (0, s_k)$. Aseta $k = k + 1$ ja jatka vaiheesta 2.

Sakko- ja estefunktiomenetelmiä ei voi pitää yleismenetelminä, sillä sakkoparametrin tehokkaaseen valintaan ei ole löydetty yleispätevää ratkaisua. Lisäksi estefunktiomenetelmässä kasvaa tehtävän häiriöalttius, kun estefunktion reuna jyrkkenee. Toisaalta konvekseille tehtäville on saatu hyviä tuloksia estefunktioalgoritmien suppenemisesta.

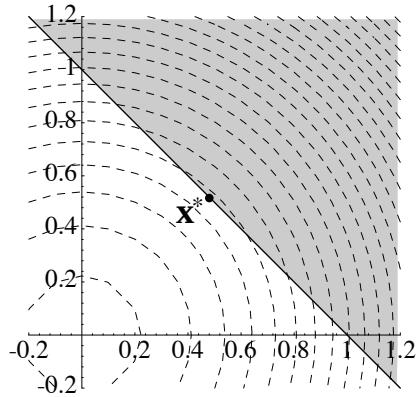
Estefunktiomenetelmiä on viime aikoina tutkittu runsaasti lähinnä lineaaristen optimointitehtävien sisäpistealgoritmien ansiosta. Nämä menetelmät perustuvat käyvän alueen reunan välttämiseen, jolloin suppeneminen on nopeampaa hyvin isoissa tehtävissä. LP-tehtävien ratkaisua on käsitelty luvussa 4 sivulta 74 alkaen.

Esimerkki 8.6.2 Olkoon ratkaistavana minimointitehtävä

$$\underset{\mathbf{x}}{\text{minimi}}\ f(\mathbf{x}) = x_1^2 + x_2^2, \text{ kun } g(\mathbf{x}) = 1 - x_1 - x_2 \leq 0. \tag{8.22}$$

Tehtävää on havainnollistettu kuvassa 8.6. Valitaan ratkaisumenetelmäksi neliöllinen sakkofunktio:

$$P(\mathbf{x}, s) = f(\mathbf{x}) + s\left(\max\{0, g(\mathbf{x})\}\right)^2 = x_1^2 + x_2^2 + s\left(\max\{0, 1 - x_1 - x_2\}\right)^2.$$



Kuva 8.6: *Esimerkissä 8.6.2 käsiteltävä optimointitehtävä. Tehtävän käypä alue on merkitty harmaalla värillä. Kohdefunktion tasa-arvokäyrät on piirretty katkoviivoilla.*

Funktion $P(\mathbf{x}, s)$ kriittinen piste $\mathbf{x}^c$ käyvän alueen sisällä saadaan yhtälöstä

$$\nabla P(\mathbf{x}, s) = \begin{pmatrix} \dfrac{\partial P(\mathbf{x}, s)}{\partial x_1} \\ \dfrac{\partial P(\mathbf{x}, s)}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 2x_1 - 2s(1 - x_1 - x_2) \\ 2x_2 - 2s(1 - x_1 - x_2) \end{pmatrix} = \mathbf{0}$$

eli $\mathbf{x}^c = (s/(1+2s),\ s/(1+2s))^T$. Asettamalla $s \to +\infty$, saadaan optimikohdalle arvio $\mathbf{x}^* \approx (0.5, 0.5)^T$. Kyseessä on ulkopuolinen sakkofunktio eli optimikohtaa lähestytään käyvän alueen ulkopuolelta käsin.

8.7 Menetelmien tehokkuusvertailu

Seuraavassa vertaillaan eräitä rajoitteellisten optimointitehtävien ratkaisumenetelmiä ja -ohjelmistoja. Ratkaistavana on 200 muuttujaa sisältävä epälineaarinen optimointitehtävä.

Esimerkki 8.7.1 Minimoidaan Rosenbrockin moniulotteista funktiota

$$f(\mathbf{x}) = \sum_{i=1}^{\eta} 100 \left(x_{2i} - x_{2i-1}^2\right)^2 + (1 - x_{2i-1})^2,$$

missä $\eta = n/2 = 100$. Rajoitteet ovat

$$\begin{cases} \|\mathbf{x}\|^2 = x_1^2 + x_2^2 + \cdots + x_{200}^2 \le 190, \\ x_{20} + x_{179} \le 1.5, \\ x_{63} + x_{100} \le 1.5. \end{cases}$$

Lisäksi asetetaan laatikkorajoitteet $-2 \le x_i \le 2$, $i = 1, \ldots, 200$. Alkuarvauksia on kaksi: $\mathbf{x}_1^1 = (1, 1, \ldots)^T$ ja $\mathbf{x}_2^1 = (0, 0, \ldots)^T$. Piste $\mathbf{x}_1^1$ on rajoitteettomasta tehtävästä saatu globaali minimipiste, joka ei ole käyvällä alueella.

Taulukossa 8.1 on esitetty käytetyt ohjelmistot ja taulukossa 8.2 on testeistä saatuja tuloksia. Kaikki testit on suoritettu CSC:n Convex C3840 -tietokoneella. IMSL-aliohjelmakirjastosta käytettiin versiota 2.0 ja NAG-aliohjelmakirjastosta versiota Mark 14.

Taulukko 8.1: *Testattavat rajoitteellisten optimointitehtävien ratkaisurutiinit. Muistinkäyttö pyrkii osoittamaan algoritmin vaatiman minimityötilan liukulukuina.*

Rutiini	*Menetelmä*	*Muistinkäyttö*
GAMS/Minos	Projektiomenetelmä	—
DNCONG (IMSL)	SQP-menetelmä	$> n(3n + p + q)$
E04UCF (NAG)	SQP-menetelmä	$> n(2n + p + q)$

Tässä testitehtävässä IMSL:n SQP-algoritmin toteutus vaikutti tehokkaimmalta ratkaisurutiinilta. GAMSilla oli suppenemisvaikeuksia tehtävän ratkaisemisessa, vaikkakin alkuarvaus $\mathbf{x}_1^1$ tuotti tuloksen.

Taulukko 8.2: *Rajoitteellisten optimointitehtävien ratkaisuohjelmistojen vertaamisesta saatuja tuloksia (kappale 8.7). Piste $\mathbf{x}^f$ on ohjelmiston antama ratkaisupiste. Lisäksi on kerrottu iteraatioiden ja funktiokutsujen määrä sekä kulutettu CPU-aika sekunteina.*

Rutiini	$f(\mathbf{x}^f)$	*Iter.*	*Funkt.*	*CPUs*
Alkuarvaus $\mathbf{x}_1^1$				
GAMS/Minos	0.131946	6+3616	7805	150
DNCONG (IMSL)	0.131946	24	54	9
E04UCF (NAG)	0.131946	44	105	60
Alkuarvaus $\mathbf{x}_2^1$				
GAMS/Minos	—			
DNCONG (IMSL)	0.131946	207	556	74
E04UCF (NAG)	0.131946	482	907	615

8.8 Ohjelmistoja

Matlabin Optimization Toolbox tarjoaa mahdollisuuden pienten ja keskisuurten rajoitteellisten optimointitehtävien ratkaisemiseen (taulukko B.3 sivulla 209). Aliohjelmakirjastot IMSL ja NAG sisältävät hyvin dokumentoituja ja melko luotettavia ratkaisurutiineja. Netlib-verkkoarkiston TOMS-algoritmeista löytyy käyttökelpoisia rutiineja (taulukko B.6 sivulla 214).

SQP-menetelmää käytetään mm. Matlabin Optimization Toolboxissa [Gra93, HHL+03] ja IMSL-aliohjelmakirjastossa [IMS]. FSQP-ohjelmisto sisältää SQP-algoritmin toteutuksen min-max -optimointitehtävien ratkaisemiseen (kappale 9.2).

Ohjelmistoista löytyy lisätietoja esimerkiksi IMSL- ja NAG-aliohjelmakirjastojen käsikirjoista [IMS, Num] ja optimointia käsittelevistä oppikirjoista [Sca85, DS83, GMW81]. Hyvä katsaus löytyy teoksesta *Optimization Software Guide* [MW93].

8.9 Lisätietoja

Teokset *Practical Optimization* [GMW81] ja *Practical Methods of Optimization* [Fle87] tarjoavat käytännöllisen ja perusteellisen johdatuksen epälineaariseen optimointiin. Myös teosta *Introduction to Non-linear Optimization* [Sca85] voi käyttää lähdeteoksena. Epälineaarista optimointia on kä-

sitelty myös teoksissa [MS78, Mie98, PSU88]. Katsausartikkeleista [CGT94, AD94, Wri92a] saa lisätietoa aiheesta.

Artikkelissa [GMSW84a] on esitelty SQP-menetelmien taustaa ja vertailtu eri tyyppisiä toteutuksia. Menetelmän kuvaus löytyy teoksista [GMW81, Fle87, Mie98, Sca85, SP89]

Projektiomatriisien käyttöä ja projektiomenetelmiä on käsitelty viitteissä [GMSW84b, Sca85, Mie98]. Täydennetyn Lagrangen funktion menetelmää on esitelty teoksissa [Fle87, Sca85, MS87]. Epälineaarisia rajoitteita on käsitelty teoksissa [GMW81, Fle87, Sca85, Wri92a, Mie98]. Sakkofunktioiden käyttöä on analysoitu viitteissä [Wri92a, MW].

9 Erikoismenetelmiä

Tässä luvussa sekä luvuissa 10 ja 11 esitellään erikoismenetelmiä, joita voi yrittää käyttää perinteisillä menetelmillä vaikeasti ratkaistaviin tehtäviin. Aluksi käsitellään globaalin optimin löytämisen problematiikkaa sekä eräitä erikoistyyppisiä tehtäviä.

9.1 Globaali optimointi

Epälineaaristen optimointitehtävien ratkaisualgoritmit löytävät lokaalin minimin, eikä globaalisuutta yleensä voida osoittaa, ellei kohdefunktiolle ja käyvälle alueelle voida todistaa konveksisuutta tai vastaavaa ominaisuutta. Yleensä ratkaisualgoritmit löytävät lähellä alkuarvausta $\mathbf{x}^1$ olevan minimipisteen käyttäen iteratiivista menetelmää

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \lambda_k \mathbf{s}^k,\ \lambda_k > 0,\ k = 1, 2, \ldots, \tag{9.1}$$

kunnes iteraatio suppenee eli esimerkiksi $\|\mathbf{s}^k\| \to 0$. Jos sekä kohdefunktio että käypä alue ovat konvekseja, on löydetty lokaali minimi myös globaali (lause 3.4.5 sivulla 67).

Globaalissa optimoinnissa etsitään optimointitehtävän pienintä paikallista minimipistettä. Siten kyseessä on tehtävä, joka on huomattavasti vaikeampi kuin yksittäisen lokaalin minimipisteen hakeminen.

Perinteisistä lokaaleja minimiä hakevista menetelmistä löytävät todennäköisimmin globaalin minimin erilaiset sakkofunktiomenetelmät, mutta ne voivat vaatia runsaasti työtä sakkoparametrien valitsemiseksi. Toisaalta voidaan kokeilla useita eri lähtöpisteitä (esimerkiksi satunnaisesti valittuja), jolloin saadaan tilastollista aineistoa globaalin optimin löytämisen varmistamiseksi.

Huomautus 9.1.1 Mikään optimointimenetelmä ei voi taata globaalin minimin löytymistä mille tahansa funktiolle. Jos globaalissa optimointitehtävässä on yli 20 muuttujaa, voi olla ettei probleemalle löydy luotettavaa ratkaisumenetelmää.

Optimoitavan funktion täytyy kuulua johonkin sopivan ”kauniisti” käyttäytyvään funktioluokkaan (esimerkkinä polynomit), jotta voitaisiin luottaa

optimointimenetelmän antamaan arvioon globaalille minimille. Esimerkiksi funktion jatkuvuus on luonnollinen oletus. Eräs tyypillinen funktioluokka ovat Lipschitz-jatkuvat funktiot, joille on olemassa vakio $L > 0$ siten, että kaikilla $\mathbf{x}$ ja $\mathbf{y} \in \mathcal{F} \subset \mathbb{R}^n$ pätee

$$|f(\mathbf{x}) - f(\mathbf{y})| \leq L \|\mathbf{x} - \mathbf{y}\|. \tag{9.2}$$

Kuitenkin globaalin optimiratkaisun löytämiseen tarvitaan liian paljon laskentatyötä, jotta mikään "varma menetelmä" voisi olla käytännöllinen. Yleensä käytetään menetelmiä, joissa pyritään löytämään joukko (ei kaikkia!) lokaaleja minimejä, ja valitaan näistä pienin.

Esimerkki 9.1.1 Skalaarifunktiot

$$\begin{aligned} f_1(x) &= x^3, & f_2(x) &= x^3 - 3x, \\ f_3(x) &= \cos x, & f_4(x) &= x^2 - \cos x, \\ f_5(x) &= \exp(-x) \end{aligned}$$

ovat eri syistä vaikeita optimoida. Funktiot f_1 ja f_2 eivät ole alhaalta rajoitettuja, funktiot f_3 ja f_4 sisältävät jaksollisen komponentin, ja funktio f_5 saavuttaa minimin äärettömyydessä.

Globaalin optimoinnin menetelmät voidaan jakaa kahteen luokkaan: deterministisiin ja tilastollisiin menetelmiin. Monet menetelmät käyttävät deterministisiä menetelmiä tilastollisen tai heuristisen menetelmän osana.

Toistuvasti käynnistetyissä menetelmissä (multistart) 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.

Menetelmän valintaan vaikuttavat myös globaalin optimoinnin tavoitteet. Jos optimoitavan systeemin sisäinen käyttäytyminen on tuntematonta, ei voida taata optimin löytämistä (minimipiste voi sijaita miten pienessä osassa ratkaisuavaruutta tahansa). Toisaalta jos tehtävän dimensio on hyvin iso, voi globaalin optimin hakeminen viedä hyvin paljon resursseja.

Joskus on tärkeää löytää lokaalit minimit globaalin optimin 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.

Kannattaa myös miettiä, tarvitaanko takeita globaalin optimin löytymiselle. Konvekseille optimointitehtäville (esimerkiksi lineaarinen optimointi) voidaan taata löydetyn lokaalin minimin globaalisuus, mutta käytännön tehtävät eivät välttämättä ole konvekseja. Toisinaan kuitenkin riittää tarpeeksi hyvän lokaalin minimin löytyminen, jolloin ei välttämättä tarvita globaalia optimointimenetelmää.

Myös optimointimallin muodostaminen vaikuttaa tehtävän ratkaisemiseen: jos malli vastaa vain karkeasti todellista tilannetta, ei tarkan globaalin optimiratkaisun löytämisestä ole välttämättä mitään hyötyä.

Esimerkki 9.1.2 Haetaan maksimi funktiolle (kuva 9.1)

$$f(\mathbf{x}) = 3\,(1-x_1)^2\,\exp(-x_1^2-(x_2+1)^2) - 10\,(\frac{x_1}{5} - x_1^3 - x_2^5)$$
$$\times \exp(-x_1^2-x_2^2) - \frac{1}{3}\exp(-(x_1+1)^2 - x_2^2) + 10.$$

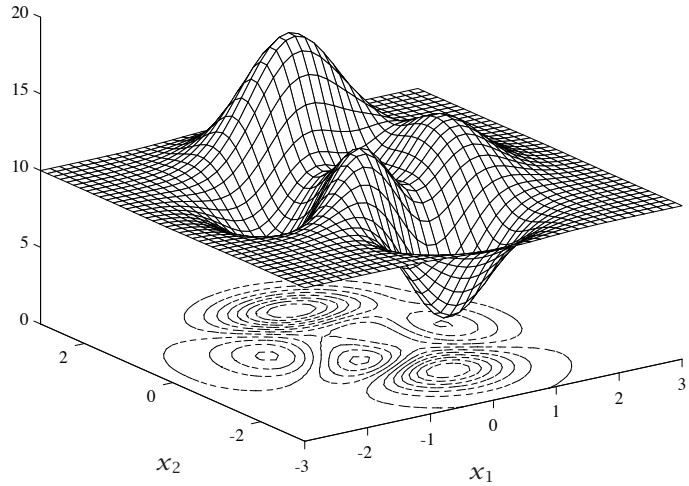
Funktiolla on kolme lokaalia maksimia. Likimääräisen globaalin maksimin löytämiseen lokaalilla optimointimenetelmällä tarvitaan hyvä alkuarvaus. Globaali maksimi voidaan löytää mm. tilastollisia menetelmiä käyttävillä ohjelmistoilla. Globaali optimipiste on $(-0.0106, 1.5803)^T$ ja funktion maksimiarvo on likimäärin 18.1.

Kuvassa 9.2 on yritetty ratkaista käytettyä testitehtävää Matlabin Optimization Toolboxin rutiinilla `fminu`, joka perustuu BFGS-sekanttimenetelmään. Tehtävässä minimoitiin funktiota $-f(\mathbf{x})$. Minimoinnissa käytettiin neljää eri alkuarvausta, joista algoritmi suppeni lokaaleihin optimipisteisiin. Globaali optimikohta $\mathbf{x}^* = (-0.0093, 1.5814)^T$ löytyi sopivilla alkuarvauksilla. Lisäksi löytyi lokaali optimikohta $\mathbf{x} = (-0.4600, -0.6292)^T$.

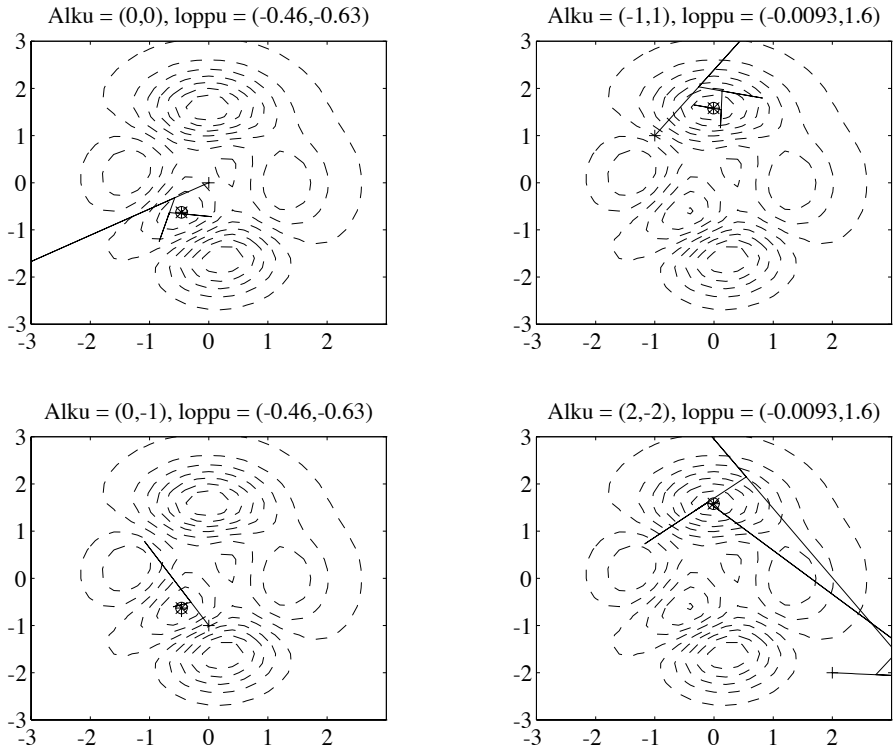
Matlabin BFGS-rutiini suppenee siis kohti lokaaleja minimejä. Mikäli alkuarvaus on kyllin lähellä lokaalia minimikohtaa, löydetään se varsin nopeasti, mutta mitään takeita minimikohdan globaalisuudesta ei voida esittää.

9.1.1 Tilastolliset hakumenetelmät

Tilastollisissa hakumenetelmissä generoidaan hakusuuntia sopivan (usein heuristisen) strategian mukaisesti. Menetelmille ei voida taata stabiilisuutta muuten kuin tilastollisessa mielessä (suppenemiseen tarvitaan äärettömän



Kuva 9.1: *Kuvan funktiolla on kolme lokaalia maksimia.*



Kuva 9.2: *Useita lokaaleja maksimipisteitä sisältävän funktion optimointi Matlabin Optimization Toolboxilla.*

monta hakua). Toisaalta eräissä tilastollisissa menetelmissä voivat kohdefunktio ja rajoitteet olla jopa epäjatkuvia.

Algoritmi 9.1.1 (Tilastollisen menetelmän yleiskuvaus)

1: Valitse alkuarvaus $\mathbf{x}^1 \in \mathcal{F} \subset \mathbb{R}^n$ ja aseta $k = 1$.

2: Muodosta otantajakauma g_k ja valitse piste $\mathbf{y}$ otantajakaumasta.

3: Aseta $\mathbf{x}^{k+1} = S(\mathbf{x}^k, \mathbf{y})$.

4: Laske $f(\mathbf{x}^{k+1})$. Jos arvo on kyllin hyvä, lopetetaan.

5: Aseta $k = k + 1$. Mene vaiheeseen 2.

Tässä esitetyssä algoritmissa voidaan valita otantajakauman tiheysfunktioksi g_k esimerkiksi tasajakauma, mikäli käypä alue $\mathcal{F}$ on rajoitettu. Jos rajoitteet ovat muotoa $\mathbf{x}^l \le \mathbf{x} \le \mathbf{x}^u$ (laatikkorajoitteet), on otantajakaumasta helppo muodostaa näytepisteitä $\mathbf{y}$.

Valintafunktio $S(\mathbf{x},\mathbf{y})$ määrää menetelmän käyttäytymisen otantajakauman ohella. Seuraavassa muutamia vaihtoehtoja:

$$\begin{aligned} S(\mathbf{x},\mathbf{y}) &= \lambda\mathbf{x} + (1-\lambda)\mathbf{y}, \\ &\quad \text{missä } \lambda = \arg\text{minimi}_{\lambda\in[0,1]} f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}), \\ S(\mathbf{x},\mathbf{y}) &= \begin{cases} \mathbf{x}, & \text{jos } f(\mathbf{x}) \le f(\mathbf{y}), \\ \mathbf{y}, & \text{jos } f(\mathbf{x}) > f(\mathbf{y}). \end{cases} \end{aligned}$$

Valintafunktiolle S asetetaan usein ehto

$$f(S(\mathbf{x},\mathbf{y})) \le \min\{\, f(\mathbf{x}), f(\mathbf{y}) \,\}. \tag{9.3}$$

Olkoon m_k tiheysfunktiota g_k vastaava todennäköisyysmitta. Olkoon joukko $\mathcal{R} \subset \mathcal{F}$ nollasta poikkeava todennäköisyysmitan mielessä, jolloin pätee $\mathrm{m}_k(\mathcal{R}) > 0$. Jos kaikilla tällaisilla joukoilla pätee

$$\prod_{k=1}^{\infty} (1 - \mathrm{m}_k(\mathcal{R})) = 0, \tag{9.4}$$

suppenee algoritmin 9.1.1 tuottama funktion f arvojen jono $\{f(\mathbf{x}^k)\}$ tilastollisesti kohti globaalia minimiä, eli pätee

$$\lim_{k\to\infty} \mathrm{P}(|f(\mathbf{x}^k) - f(\mathbf{x}^*)| \le \epsilon) = 1 \tag{9.5}$$

kaikilla $\epsilon > 0$.

Jos joukko $\mathcal{F}$ on rajoitettu ja otantajakauma g_k on tasajakauma kaikilla k, pätee yhtälö (9.4). Tällainen menetelmä siis suppenee kohti globaalia minimiä todennäköisyydellä yksi. Toisaalta tällainen algoritmi on erittäin tehoton, sillä se ei hyödynnä mitenkään löydettyjen pisteiden tuottamaa kuvaa kohdefunktion arvojen jakautumasta.

Monissa käytännön tehtävissä (esimerkiksi kemian epälineaariset optimointitehtävät) tilastolliset menetelmät ovat osoittautuneet tehokkaiksi. Teho riippuu käytännössä siitä, miten onnistuneesti hakustrategia käy läpi ratkaisuavaruutta.

9.2 Min-max -optimointitehtävät

Äärellisellä min-max -optimointitehtävällä tarkoitetaan seuraavaa ongelmaa:

$$\text{minimi}_{\mathbf{x}} \max_i \{f_i(\mathbf{x}),\ i = 1,\ldots,m\} \text{ siten että } \mathbf{x} \in \mathcal{F}\,, \tag{9.6}$$

missä funktiot $f_i : \mathbb{R}^n \mapsto \mathbb{R}$ ovat jatkuvasti differentioituvia. Määritellään

$$\varphi(\mathbf{x}) = \max_i \{f_i(\mathbf{x}),\ i = 1,\ldots,m\}. \tag{9.7}$$

Funktiota $\varphi(\mathbf{x})$ käytetään useissa menetelmissä *ansiofunktiona*. Funktio φ on yleisessä tapauksessa ei-differentioituva pisteissä, joissa maksimiarvon saavuttaa useampi kuin yksi funktio f_i.

Eräs vaihtoehtoinen tapa ratkaista tehtävä (9.6) on muuntaa se muotoon

$$\underset{\mathbf{x}\in\mathbb{R}^{n+1}}{\text{minimi}}\ x_{n+1},\ \text{kun}\ f_i(\mathbf{x}) \le x_{n+1},\ i = 1,\ldots,m. \tag{9.8}$$

Kyseessä on differentioituva rajoitettu optimointitehtävä. Ongelmana on tietenkin epälineaaristen rajoitteiden käsittely.

9.3 Epäsileä optimointi

Tarkastellaan lähemmin kappaleessa 3.4 esiteltyjä konveksisuuden määritelmiä ja ominaisuuksia. Seuraava lause mahdollistaa konveksisuuden käytön epäsileissä optimointitehtävissä (non-smooth optimization).

Lause 9.3.1 Olkoon $f(\mathbf{x})$ konveksi funktio, joka on määritelty ei-tyhjässä konveksissa joukossa $\mathcal{F} \subset \mathbb{R}^n$. Jos $\mathbf{x}^0$ on joukon $\mathcal{F}$ sisäpiste, löytyy vektori $\mathbf{d} \in \mathbb{R}^n$ siten, että

$$f(\mathbf{x}) \ge f(\mathbf{x}^0) + \langle \mathbf{d}, \mathbf{x} - \mathbf{x}^0 \rangle \text{ kaikilla } \mathbf{x} \in \mathcal{F}. \tag{9.9}$$

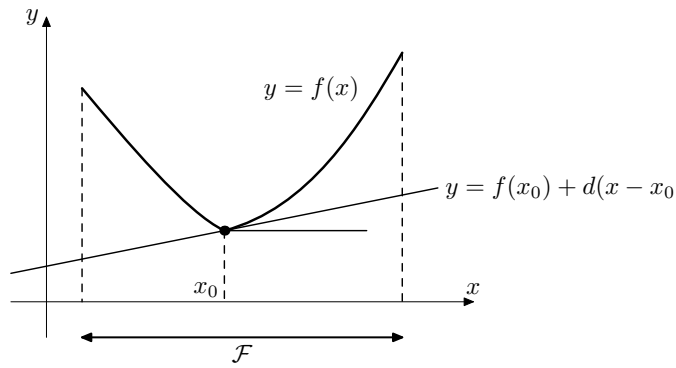
Konkaaville funktiolle pätee tietenkin $\le$ yhtälössä (9.9).

Todistus: Vertaa lauseeseen 3.4.1 sivulla 65. Lausetta on havainnollistettu kuvassa 9.3. Lisätietoja löytyy tämän luvun kirjallisuusluettelossa mainituista teoksista. □

Määritelmä 9.3.1 (Aligradientti ja alidifferentiaali) Edellä määriteltyä vektoria $\mathbf{d}$ kutsutaan funktion $f(\mathbf{x})$ *aligradientiksi* (subgradient) pisteessä $\mathbf{x}^0$ (kuva 9.3). Kaikkien aligradienttien joukkoa

$$\partial_c f(\mathbf{x}) = \{\mathbf{d} \in \mathbb{R}^n \mid f(\mathbf{x}) \ge f(\mathbf{x}^0) + \langle \mathbf{d}, \mathbf{x} - \mathbf{x}^0 \rangle \text{ kaikilla } \mathbf{x} \in \mathcal{F}\} \tag{9.10}$$

kutsutaan funktion $f(\mathbf{x})$ *alidifferentiaaliksi*. Jos funktion f ensimmäiset osittaisderivaatat ovat olemassa pisteessä $\mathbf{x}^0$, koostuu alidifferentiaali yhdestä vektorista $\{\nabla f(\mathbf{x})\}$.



Kuva 9.3: *Skalaarifunktiolle $f(x)$ pisteestä x_0 löytyvän aligradientin d määrittelemä suora $y = f(x_0) + d(x - x_0)$, $x, x_0 \in \mathcal{F} \subset \mathbb{R}$.*

Konveksisuus ja alidifferentiaali voidaan liittää toisiinsa seuraavasti:

Lause 9.3.2 Jos funktio $f : \mathbb{R}^n \mapsto \mathbb{R}$ on konveksi, pätee kaikilla $\mathbf{y} \in \mathbb{R}^n$:

$$f(\mathbf{y}) = \max\{f(\mathbf{x}) + \langle \mathbf{d}, \mathbf{y} - \mathbf{x} \rangle \mid \mathbf{x} \in \mathbb{R}^n, \mathbf{d} \in \partial_c f(\mathbf{x})\}. \tag{9.11}$$

Kohdefunktion ja käyvän alueen konveksisuutta käyttäen voidaan esittää seuraava lause:

Lause 9.3.3 Olkoon $f(\mathbf{x})$ konveksi funktio ei-tyhjässä konveksissa joukossa $\mathcal{F} \subset \mathbb{R}^n$. Olkoon optimointitehtävänä

$$\operatorname*{minimi}_{\mathbf{x}} f(\mathbf{x}), \text{ kun } \mathbf{x} \in \mathcal{F} \subset \mathbb{R}^n.$$

Piste $\mathbf{x}^*$ on tämän tehtävän optimiratkaisu, jos ja vain jos funktiolla $f(\mathbf{x})$ on pisteessä $\mathbf{x}^*$ aligradientti $\mathbf{d}$, jolle pätee:

$$\langle \mathbf{d}, \mathbf{x} - \mathbf{x}^* \rangle \geq 0 \text{ kaikilla } \mathbf{x} \in \mathcal{F}.$$

Alidifferentiaalia käyttämällä voidaan johtaa optimointimenetelmiä, jotka soveltuvat epäsileisiin tapauksiin. Epäsileisiin tehtäviin soveltuu mm. aliohjelmakirjasto NSOLIB [Mäk93].

9.4 Ohjelmistoja

Tilastollista globaalin optimoinnin hakumenetelmää käyttää mm. Smin1-ohjelmisto, joka kykenee käsittelemään myös laatikko- ja epälineaarisia rajoitteita (liite B). Netlibistä löytyvä TOMS-algoritmi 667 (SIGMA) perustuu puolestaan stokastisten differentiaaliyhtälöiden käyttöön. Differentiaaliyhtälöihin perustuvaa lähestymistapaa on hyödynnetty myös neuroverkkojen käyttämisessä optimointiin [CU93].

Ohjelmisto FSQP (feasible sequential quadratic programming) ratkaisee min-max -optimointitehtävän käyttäen toistettua kvadraattista optimointia (kappale 8.5). FSQP kykenee käsittelemään yleisiä epälineaarisia rajoitteita lineaaristen rajoitteiden ohella. Lisäksi FSQP:n tuottamat iteraatiopisteet toteuttavat optimointitehtävän epäyhtälörajoitteet sekä lineaariset yhtälörajoitteet. Lisätietoja löytyy liitteestä B.

Myös Matlabin Optimization Toolbox osaa ratkaista min-max -optimointitehtäviä (taulukko B.3 sivulla 209).

9.5 Lisätietoja

Eräitä globaalin optimoinnin menetelmiä on tutkittu artikkelissa *A Stochastic Approach to Global Optimization* [KBT84]. Teoksesta *Towards Global Optimization 2* [DS78] löytyy katsausluonteisia artikkeleita. Teokset

[MN93, Mäk90, Mäk93] esittelevät mm. epäsileiden tehtävien ratkaisemisessa käytettäviä menetelmiä. Epäsileää optimointia on käsitelty myös teoksissa [Roc93, PSU88, Zow84].

Neuroverkkojen käyttämistä optimointiin esittelee teos *Neural Networks for Optimization and Signal Processing* [CU93]. Väliaritmetiikkaa on käsitelty mm. raportissa [McK93].

Luvuissa 10 ja 11 on esitelty geneettisten algoritmien ja jäähdytysmenetelmien käyttöä globaaliin optimointiin.

10 Geneettiset algoritmit

Geneettisiä algoritmeja voi yrittää käyttää eräisiin vaikeasti ratkaistaviin optimointitehtäviin. Menetelmän tutkimiseen ja havainnollistamiseen käytetään Matlabia.

10.1 Kombinatorinen optimointi

Kombinatoriset optimointitehtävät ovat usein vaikeita ratkaista. Näissä tehtävissä on tyypillisesti luokkaa $n!$ tai 2^n eri ratkaisumahdollisuutta, kun n on muuttujien lukumäärää kuvaava kokonaisluku.

Tyypillinen esimerkki vaikeasta kokonaislukutehtävästä on *kauppamatkustajan ongelma* (kappale 10.3), jossa täytyy valita reitti joka käy kerran kussakin n:stä kaupungista. Tästä tehtävästä on monia muunnelmia. Perustehtävänä on valita alkioiden järjestys siten, että kohdefunktio minimoituu tai maksimoituu.

Kombinatorisen optimointitehtävän parhaan mahdollisen ratkaisun eli globaalin optimin löytäminen on vaikeaa. Samaan luokkaan voi laskea kuuluviksi monet epälineaariset useita lokaaleja minimejä sisältävät globaalit optimointitehtävät. Tällaisissa tehtävissä joudutaan usein turvautumaan heuristisia lähestymistapoja hyödyntäviin likimääräisiä ratkaisuja etsiviin menetelmiin.

10.1.1 Ratkaisumenetelmiä

Tässä oppaassa käsitellään kahta vaikeiden globaalien tai kombinatoristen optimointitehtävien likimääräiseen ratkaisemiseen soveltuvaa menetelmätyyppiä: *geneettisiä algoritmeja* (tämä luku) ja *jäähdytysmenetelmiä* (luku 11). Muitakin heuristisia menetelmiä on olemassa, esimerkiksi *tabu search* [Moo93]. Kaikki nämä menetelmät vaativat tehtäväkohtaista virittelyä pyrittäessä tehokkuuteen ja luotettavuuteen.

Geneettisiin algoritmeihin (genetic algorithms eli GA) liittyvät myös käsitteet *evoluutiostrategiat* (ES), evolutionary computing (EC) ja genetic programming (GP).

ES-menetelmät muistuttavat geneettisiä algoritmeja mutta ne on alunperin kehitetty jatkuvaparametriseen optimointiin [HB92]. EC on yleisnimi laskennalle, jossa simuloidaan evoluutiota. Geneettinen ohjelmointi eli GP tarkoittaa tietokoneohjelmien kehittelyä geneettisten algoritmien avulla: ohjelmakoodeja yhdistellään, muunnellaan ja valitaan, kunnes tehtävälle löydetään riittävän hyvä koodaus.

10.2 Optimointi geneettisellä algoritmilla

Geneettiset algoritmit ovat luonnon evoluutiomekanismeja ”matkivia” optimointimenetelmiä. Geneettiset algoritmit eivät vaadi kohdefunktion tai rajoitteiden jatkuvuutta, joten niitä voidaan käyttää perinteisillä menetelmillä vaikeasti ratkaistaviin ongelmiin. Lisäksi menetelmä soveltuu luonnostaan rinnakkaislaskentaan.

Geneettisiä algoritmeja on käytetty tehtäviin, joissa ratkaisuavaruus on hyvin suuri (esimerkiksi isot kombinatoriset tehtävät) ja likimääräinen eli ”kyllin hyvä” optimi riittää ratkaisuksi.

Geneettiset algoritmit sopivat siis tilanteisiin, joissa kohtuullisen hyvän ratkaisun saaminen on tärkeää eikä niinkään tarkat ratkaisut. Toisaalta menetelmistä ei ole vielä yleiskäyttöön, vaikkakin kokeiluluontoisia ohjelmistoja jo löytyy (liite B). Ongelman koodaus sopivaan esitysmuotoon sekä ratkaisualgoritmin heuristiikka vaikuttaa suuresti ratkaisemisen tehokkuuteen.

Geneettisiä algoritmeja ei kannata käyttää (ainakaan ainoana menetelmänä) perinteisillä algoritmeilla tehokkaasti ratkaistaviin tehtäviin, esimerkiksi jatkuvasti differentioituviin optimointitehtäviin: mikäli saatavilla on hyvä alkuarvaus, perinteiset menetelmät ovat yleensä paljon tehokkaampia. Toisaalta jos optimointitehtävässä on epäjatkuvuuksia ja paljon lokaaleja minimejä, geneettiset algoritmit voivat olla käyttökelpoisia.

10.2.1 Geneettisten algoritmien rakenne

Geneettisessä algoritmissa koodataan ratkaisuavaruuden vektorit sopivalla tavalla ”geneettiseksi koodiksi” eli ”yksilöksi” $\mathbf{x}$ (jonoksi bittejä tai liukulukuja tietokoneen muistissa). Täten esimerkiksi $\mathbf{x} \in \{0,1\}^n$ tai $\mathbf{x} \in \mathbb{R}^n$.

Seuraavassa on esitetty yksinkertainen geneettinen algoritmi:

Algoritmi 10.2.1 (Geneettinen algoritmi)

luo alkupopulaatio $\mathcal{P}_1 = \{\mathbf{x}^i\}$, $i = 1, \ldots, l$
for $k = 1, 2, \ldots, m$
  risteytä populaation $\mathcal{P}_k$ yksilöitä keskenään
  generoi mutaatioita populaatiossa $\mathcal{P}_k$
  if löytyy tarpeeksi hyvä ratkaisu $\mathbf{x} \in \mathcal{P}_k$
    aseta $\mathbf{x}^* = \mathbf{x}$ ja lopeta iterointi

```
    else
        valitse populaatiosta P_k parhaat yksilöt populaatioksi P_{k+1}
    end
end
```

Alkupopulaatioksi $\mathcal{P}_1$ valitaan edustava joukko yksilöitä eli parametrivektoreita $\{\mathbf{x}^i\}$, $i = 1, \ldots, l$. Tämän jälkeen ”risteytetään” kaksi sopivalla mekanismilla valittua yksilöä ja muutetaan saatua jälkeläistä satunnaisen ”mutaation” avulla. Populaatiosta voidaan karsia kullakin iteraatiokierroksella huonosti optimointitehtävän ratkaisuksi sopivat yksilöt.

Parametriavaruuden alkioita $\mathbf{x}$ voi verrata kromosomeihin Kukin arvo x_j, $j = 1, \ldots, n$ (bitti tai reaaliluku) vastaa suunnilleen geeniä tai geeniryhmää. Ratkaisuehdokkaiden vertaamiseksi tarvitaan parametrivektoreille *hyvyysfunktio* $f(\mathbf{x})$, jota pyritään maksimoimaan ja jonka arvot ovat positiivisia.

Geneettisten algoritmien soveltaminen optimointiin perustuu valintamekanismin ja risteytysmekanismin käyttämiseen. Toisin sanoen populaatiota ohjataan kohti parempia ratkaisuja. Toisaalta satunnainen muuntelu (mutaatiot) pyrkii estämään jäämisen lokaaliin minimiin.

10.3 Kauppamatkustajan ongelma

Kauppamatkustajan ongelmalla tarkoitetaan tehtävää, jossa on etsittävä edullisin suljettu reitti, joka kulkee kaikkien annettujen ”kaupunkien” kautta. Kaupunkin väliset yhteydet ja etäisyydet ovat tiedossa, joten kyseessä on tyypillinen kombinatorinen tehtävä. Tehtävää on käsitelty hyvin monissa kombinatorista optimointia käsittelevissä teoksissa. Seuraavassa esitetään matemaattinen malli kauppamatkustajan ongelmalle.

Esimerkki 10.3.1 Olkoon annettu n kaupunkia ja symmetrinen $n \times n$ -kokoinen matriisi D, jonka alkiot kuvaavat etäisyyksiä kaupunkiparien (a, b) välillä, $a, b \in \{1, 2, \ldots, n\}$. Reitillä tarkoitetaan suljettua polkua, joka käy kussakin kaupungissa täsmälleen kerran.

Tehtävän ratkaisuavaruutena on kaupunkien syklisten permutaatioiden π joukko, jossa on $(n-1)!$ eri mahdollisuutta. Kustannusfunktiona on reitin pituus eli

$$f(\pi) = \sum_{i=1}^{n} d_{i,\pi(i)}\,, \tag{10.1}$$

missä $\pi(i)$ tarkoittaa kaupungin i seuraajaa reitillä.

Seuraavassa esitetään eräs tapa koodata kauppamatkustajan ongelma geneettisen algoritmin vaatimaan muotoon. Keskeinen vaatimus on, että kahden parametrivektorin (kaksi eri reittiä) yhdistäminen tuottaa käypiä parametrivektoreita (uudet reitit).

Valitaan jokin reitti, joka kulkee kaikkien kaupunkien kautta. Pidetään tätä ”vakiojärjestyksenä”, esimerkiksi (a, b, c, d, e).

Mikä tahansa reitti, esimerkiksi (a, c, e, d, b) koodataan seuraavasti: poistetaan vakiojärjestyksessä olevasta listasta kaupungit sitä mukaan kun ne esiintyvät reitissä ja otetaan ylös kukin esiintymispaikka vakiolistassa:

Kaupunki	*Vakiolista*	*Järjestysnro*
a	(a, b, c, d, e)	1
c	(b, c, d, e)	2
e	(b, d, e)	3
d	(b, d)	2
b	(b)	1

Siten reitin (a, c, e, d, b) koodivektori on $(1, 2, 3, 2, 1)$. Tällä tavalla koodattujen reittien risteytykset ovat myös reittejä! Risteytys voidaan tehdä seuraavasti:

1. valitaan yksilöt $\mathbf{x}$ ja $\mathbf{y}$,
2. arvotaan katkaisukohta $l \in [1, 2, \ldots, n-1]$, missä n on koodivektorin komponenttien lukumäärä,
3. otetaan jälkeläisiksi vektorit $(x_1, x_2, \ldots, x_l, y_{l+1}, \ldots, y_n)$ ja $(y_1, y_2, \ldots, y_l, x_{l+1}, \ldots, x_n)$.

Esimerkiksi koodivektorit $(1, 2, 3, 2, 1)$ ja $(1, 1, 1, 2, 1)$ voisivat risteytyä seuraavasti:

- valitaan katkaisukohta $l = 3$ (pituus $n = 5$),
- uudet alkiot ovat $(1, 2, \mid 1, 2, 1)$ ja $(1, 1, \mid 3, 2, 1)$, missä $\mid$ kuvaa käytettyä katkaisukohtaa.

Periytyminen voidaan järjestää monella tavalla, esimerkiksi siten että kunkin sukupolven ”paras” yksilö risteytetään satunnaisesti valitun toisen yksilön kanssa. Usein risteytettävät alkiot valitaan todennäköisyydellä, joka on suoraan verrannollinen kunkin alkion hyvyysfunktion arvoon.

Mutaatio voidaan saada aikaan muuttamalla koodivektorin $\mathbf{x}$ jokin komponentti x_i toiseksi. Käytettäessä edellä kuvattua koodausta valitaan koodivektorin i:nnen komponentin uusi arvo joukosta $\{1, 2, \ldots, n + 1 - i\}$.

10.4 Geneettisen algoritmin toimintaperiaatteet

Geneettisen algoritmin perusidea on niin sanottu *rakennuspalikkahypoteesi* (building block hypothesis): hyvät parametrivektorit sisältävät osaratkaisuja, joita yhdistämällä on mahdollista löytää vielä parempia ratkaisuja. Geneettisen algoritmin tehokkuus riippuu ongelman koodaustavasta ja toisaalta sopivan kokoisen populaation valinnasta.

Geneettisiä algoritmeja analysoidaan tutkimalla ”hyvien” koodivektoreiden ominaisuuksia. Yleensä oletetaan, että käytetään *binääristä koodausta*, jolloin kukin koodivektori $\mathbf{x}$ on muotoa

$$\mathbf{x} = (x_1, x_2, \ldots, x_n), \text{ missä } x_i \in \{0,1\},\ i = 1, \ldots, n.$$

Binäärikoodaus on usein edullisin tapa koodata tehtävä, sillä tällöin tarvitaan pienin määrä merkkejä tehtävän koodaamiseen.

Koodivektorien samankaltaisuutta voidaan havainnollistaa *skeemojen* eli *mallien* (schema) avulla. Skeema on merkkijono

$$\mathcal{H} = (h_1, h_2, \ldots, h_n), \quad h_i \in \{0, 1, *\},\ i = 1, \ldots, n, \tag{10.2}$$

missä symboli $*$ vastaa sekä nollaa että ykköstä. Skeema $(0, *, 1, 1, *)$ vastaa koodivektoreiden joukkoa

$$\{(0,0,1,1,0),\ (0,0,1,1,1),\ (0,1,1,1,0),\ (0,1,1,1,1)\}.$$

Binääristen koodivektoreiden joukkoa siis vastaa 3^n erilaista skeemaa, kun n on koodivektorin pituus. Täten tietty binäärinen koodivektori on 2^n:n erilaisen skeeman erikoistapaus, esimerkiksi koodivektori $(0,1)$ vastaa skeemoja

$$\{(*,*),\ (0,*),\ (*,1),\ (0,1)\}.$$

Laskettaessa koodivektorille $\mathbf{x}$ hyvyysfunktion arvo $f(\mathbf{x})$ saadaan samalla tietoa koodivektoria vastaavien skeemojen hyvyydestä. Tätä näkökulmaa kutsutaan geneettisten algoritmien *implisiittiseksi rinnakkaisuudeksi*.

Geneettisissä algoritmeissa valitaan populaation alkioita niiden hyvyyden perusteella. Olkoon $m(\mathcal{H}^k)$ tiettyä skeemaa $\mathcal{H}$ vastaavien alkioiden lukumäärä populaatiossa iteraatiolla k. Oletetaan että koodivektori $\mathbf{x}^j$ tulee valituksi todennäköisyydellä $f(\mathbf{x}^j)/\sum_{i=1}^{n} f(\mathbf{x}^i)$ eli mitä suurempi on hyvyysfunktion arvo, sitä todennäköisempää on kyseisen koodivektorin valinta. Populaation keskimääräinen hyvyys on $\overline{f} = \sum_{i=1}^{n} f(\mathbf{x}^i)/n$, joten skeemaa $\mathcal{H}$ vastaavien alkioiden lukumäärälle iteraatiolla $k+1$ saadaan yhtälö

$$m(\mathcal{H}^{k+1}) = m(\mathcal{H}^k)\frac{f(\mathcal{H})}{\overline{f}}. \tag{10.3}$$

Täten normaalia parempi skeema saa seuraavalla iteraatiolla enemmän itseään vastaavia alkioita.

Pelkkä valinta ei kuitenkaan riitä hyvien ratkaisuvektoreiden löytämiseksi, sillä se ei tuota uusia ratkaisuvektoreita. Tähän tarvitaan risteytystä, jossa yhdistetään populaation alkioita uusiksi alkioiksi. Seuraavassa tarkastellaan yhteen katkaisukohtaan perustuvaa risteytystä.

Olkoon $\mathbf{x}$ populaation alkio ja sitä vastaavat kaksi skeemaa $\mathcal{H}_1$ ja $\mathcal{H}_2$ seuraavat:

$$\begin{aligned}
\mathbf{x} &= (1,0,1,\,|\,0,0,1),\\
\mathcal{H}_1 &= (*,0,*,\,|\,*,*,1),\\
\mathcal{H}_2 &= (*,*,*,\,|\,0,0,*).
\end{aligned}$$

Pystyviiva | tarkoittaa valittua katkaisukohtaa. Mikäli alkion $\mathbf{x}$ kanssa risteytettävä populaation alkio ei ole sama kuin $\mathbf{x}$ komponenttien x_2 ja x_6 osalta, eivät jälkeläiset enää vastaa skeemaa $\mathcal{H}_1$, sillä skeeman osat $(*, 0, *|$ ja $|*, *, 1)$ sijoittuvat eri jälkeläisiin. Toisaalta skeema $\mathcal{H}_2$ tulee säilymään, sillä sen komponentti $|0, 0, *)$ siirtyy sellaisenaan jälkeläiseen.

Jotta edellä esitettyä huomiota voitaisiin käyttää analyysissä, tarvitaan *määrittelypituuden* (definition length) käsite: määrittelypituus $\delta(\mathcal{H})$ on skeeman ensimmäisen ja viimeisen kiinnitetyn komponentin välimatka.

Esimerkkitapauksessa saadaan $\delta(\mathcal{H}_1) = \delta(*, 0, *, *, *, 1) = 4$ sekä $\delta(\mathcal{H}_2) = \delta(*, *, *, 0, 0, *) = 1$.

Risteytyskohdan mahdolliset arvot ovat $1, 2, \ldots, n-1$ eli tässä tapauksessa eri arvoja on 5 kappaletta. Jos risteytyskohta valitaan satunnaisesti tasajakaumasta, tuhoutuu skeema $\mathcal{H}_1$ todennäköisyydellä $\delta(\mathcal{H}_1)/5 = 4/5$ ja skeema $\mathcal{H}_2$ todennäköisyydellä $\delta(\mathcal{H}_2)/5 = 1/5$. Täten skeeman säilymistodennäköisyydelle P_s voidaan laskea alaraja

$$P_s \geq 1 - P_c \frac{\delta(\mathcal{H})}{n-1}, \tag{10.4}$$

missä P_c on risteytymisen todennäköisyys. Yhdistämällä lausekkeet (10.3) ja (10.4) saadaan seuraava arvio skeeman $\mathcal{H}$ edustajien lukumäärälle sukupolvessa $k+1$:

$$m(\mathcal{H}^{k+1}) \geq m(\mathcal{H}^k) \frac{f(\mathcal{H})}{\overline{f}} \left(1 - P_c \frac{\delta(\mathcal{H})}{n-1}\right). \tag{10.5}$$

Täten tiettyä skeemaa vastaavien alkioiden määrän kehitys riippuu siitä, onko kyseinen skeema keskimääräistä parempi, sekä siitä, kuinka suuri sen määrittelypituus on. Parhaiten selviävät siis skeemat, joiden määrittelypituus on lyhyt ja jotka vastaavat hyviä populaation alkioita.

Tässä ei otettu huomioon mutaation vaikutusta, joka pienentää jonkin verran tietyn skeeman säilymistodennäköisyyttä. Olkoon $o(\mathcal{H})$ skeeman $\mathcal{H}$ *kertaluku* (order) eli kiinnitettyjen komponenttien lukumäärä. Esimerkiksi skeemalle $\mathcal{H}_1$ saadaan $o(*, 0, *, *, *, 1) = 2$.

Jos mutaation todennäköisyys on P_m, saadaan yhdistämällä edellä käydyt tarkastelut seuraava lause:

Lause 10.4.1 Skeeman $\mathcal{H}$ lukumäärälle $m(\mathcal{H})$ populaatiossa iteraatioilla k ja $k+1$ pätee

$$m(\mathcal{H}^{k+1}) \geq m(\mathcal{H}^k) \frac{f(\mathcal{H})}{\overline{f}} \left(1 - P_c \frac{\delta(\mathcal{H})}{n-1} - o(\mathcal{H}) P_m\right). \tag{10.6}$$

Lausetta 10.4.1 voidaan pitää geneettisten algoritmien peruslauseena. Geneettisen algoritmin asymptoottiselle suppenemiselle voidaan esittää lause:

Lause 10.4.2 Olkoon geneettisellä algoritmilla voimassa ominaisuudet

1. Populaatioiden jono $\mathcal{P}_1, \mathcal{P}_2, \ldots$ on *monotoninen*:

$$\max_{\mathbf{x} \in \mathcal{P}_{k+1}} f(\mathbf{x}) \geq \max_{\mathbf{x} \in \mathcal{P}_k} f(\mathbf{x}) \text{ kaikilla } k = 1, 2, \ldots \tag{10.7}$$

2. Ratkaisuvektori $\mathbf{y}$ on *saavutettavissa* ratkaisuvektorista $\mathbf{x}$ käsin käyttämällä peräkkäisiä mutaatio- ja risteytysoperaatioita.

Tällöin globaali optimi löydetään todennäköisyydellä 1 eli

$$\lim_{k\to\infty} \mathrm{P}(\mathbf{x}^* \in \mathcal{P}_k) = 1. \tag{10.8}$$

Lauseessa 10.4.2 maksimoidaan hyvyysfunktion f arvoa. Monotonisuus saadaan aikaan esimerkiksi *elitismillä*: paras alkio säilyy aina seuraavalle iteraatiokierrokselle. Saavutettavuus voidaan taata esimerkiksi nollasta poikkeavalla mutaatiotodennäköisyydellä. Tällöin globaali maksimi löytyy melkein aina eli todennäköisyydellä yksi, kun geneettistä algoritmia iteroidaan kyllin kauan.

10.5 Globaali optimointi

Geneettisissä algoritmeissa prosessoidaan yleensä diskreeteistä ”kromosomeista” koostuvia alkioita. Keskeinen osa globaaliin optimointiin käytettyä geneettistä algoritmia onkin hyvän koodauksen löytäminen ratkaistavalle optimointitehtävälle.

Seuraavassa haetaan kuvassa 9.1 sivulla 156 esitetyn funktion globaali maksimi geneettisellä algoritmilla käyttäen apuna Matlab-ohjelmistoa.

Muuttujat x ja $y \in [-3, 3]$ koodataan kumpikin erikseen n:ksi bitiksi. Tämä koodaus ei välttämättä ole tehokkain, mutta se valitaan tässä yksinkertaisuuden vuoksi.

Listaus 10.1: *Bittikoodauksen muuttaminen reaaliluvuiksi.*

```
function [x,y] = gaDecode(popul)
% Lasketaan alkioiden ja kromosomien lkm
[npop bits] = size(popul);
bits = bits/2;
% Haetaan alkioiden x- ja y-komponentit
xv = popul(:,1:bits);
yv = popul(:,(bits+1):(2*bits));
% Muutetaan binääriluvut reaaliluvuiksi
code = 2.^(0:(bits-1));
r = 3.0;
s = 2*r/(2^bits-1);
for i = 1:npop
  x(i) = s*sum(xv(i,1:bits).*code) - r;
  y(i) = s*sum(yv(i,1:bits).*code) - r;
end
```

Listauksen 10.1 funktio `gaDecode` muuntaa riveittäin reaaliluvuiksi populaation kromosomit sisältävän $m \times n$ -kokoisen Matlab-taulukon `popul`.

Käytetty kromosomien talletustapa on varsin tuhlaileva, sillä kukin kromosomi (bitti) vie yhden reaaliluvun tilan. Toisaalta Matlab ei suoraan tue bittitason operaatioita.

Taulukon `popul` rivit tulkitaan populaation alkioiksi. Ensimmäiset $k = n/2$ bittiä ovat x-koordinaatti ja loput y-koordinaatti. Kromosomeja vastaavien bittien $x_i \in \{0,1\}, i = 1,\ldots,k$ muunnokseen reaaliluvuksi $y \in [-r,r]$ käytetään kaavaa

$$y = (\sum_{i=1}^{k} \frac{2r}{2^k - 1} x_i 2^{i-1}) - r. \tag{10.9}$$

Listauksessa 10.2 esitetty funktio `gaFun` laskee populaation alkioiden arvot sekä bittiesitystä vastaavat x- ja y-koordinaatit.

Listaus 10.2: *Kohdefunktion arvon laskeva Matlab-funktio gaFun.*

```
function [z, x, y] = gaFun(population)
% Lasketaan funktion arvo sekä x ja y
[x y] = gaDecode(population);
z = 10 + 3*(1-x).^2.*exp(-(x.^2)-(y+1).^2) - 10*(x/5 - ...
    x.^3 - y.^5).*exp(-x.^2-y.^2) - 1/3*exp(-(x+1).^2-y.^2);
```

Luodaan satunnainen kolmen alkion populaatio (kromosomien määrä on 8) ja lasketaan hyvyysfunktion `gaFun` arvot:

```
>> popul = rand(3, 8) > 0.5
popul =
     0     1     1     1     1     0     1     0
     1     0     1     1     0     1     0     0
     1     0     1     1     0     1     1     1
>> [z x y] = gaFun(popul)
z =
   10.0774    9.9823   10.0118
x =
    2.6000    2.2000    2.2000
y =
   -1.0000   -2.2000    2.6000
```

Funktion `gaFun` arvot ovat kaikki positiivisia, jolloin niitä voidaan käyttää suoraan hyvyysfunktion arvoina. Muussa tapauksessa arvot pitäisi normeerata. Usein sopivasti valittu normeeraus myös nopeuttaa geneettisen algoritmin suppenemista.

10.5.1 Vanhempien valinta, mutaatiot ja risteyttäminen

Geneettisessä algoritmissa tuotetaan ”jälkeläisiä” sopivalla mekanismilla valituille ”vanhemmille”. Tässä esimerkissä käytetään *rulettipöytämekanismia*: jokaisen alkion todennäköisyys päästä vanhemmaksi on suoraan verrannollinen alkion hyvyysfunktion arvoon (joka on > 0). Olkoon `values` vektori,

Listaus 10.3: *Matlab-funktio gaSim. Jokaisella iteraatiolla korvataan koko populaatio ja käytetään elitismiä. Lisäksi kerätään tilastotietoja.*

```
function [pop,values,stats]=gaSim(fun,initpop,crossprob, ...
  mutprob,gNr)

pop=initpop;               % Alkupopulaatio
[npop bits]=size(pop);     % Alkioiden ja bittien lkm
funstring=[fun,'(pop)'];   % Funktion arvot alussa
[values popx popy]=eval(funstring);
% Haetaan parhaat yksilöt
maxval=max(values);
bestgen=pop(find(values == maxval),:);
disp(['best value: ',num2str(maxval)]);

for i=1:gNr
  valuesum=cumsum(values);
  maxsum=valuesum(npop);
  stats(i,:)=[maxval mean(values) std(values)];
  newpop=[];
% Risteytetään alkioita npop/2 kertaa
  for j=1:floor(npop/2);
    for k=1:2
      tmp=find(valuesum>=maxsum*rand);
      pnt(k)=tmp(1);
    end
    if rand>=crossprob                % Tehdään risteytys
      place=round((bits-1)*rand);
      child(1,:)=[pop(pnt(1),1:place), ...
                  pop(pnt(2),(place+1):bits)];
      child(2,:)=[pop(pnt(2),1:place), ...
                  pop(pnt(1),(place+1):bits)];
    else
       child=pop(pnt,:);
    end
    child=xor(child,mutprob >= rand(2,bits)); % Mutaatiot
    newpop=[newpop; child];
  end
  pop=newpop;
  [values popx popy]=eval(funstring);
  maxvalNew=max(values);              % Paras arvo
  if maxvalNew>maxval                 % Löytyikö parempia
    bestgen=pop(find(values==maxvalNew),:);
    maxval=maxvalNew;
  else                                % Korvataan huonoin
    minindex=find(values==min(values));
    pop(minindex(1),:)=bestgen(1,:);
  end
  disp(['best value: ',num2str(maxval)]);
end
```

joka sisältää alkioiden hyvyydet. Seuraava koodi valitsee rulettipöytämekanismilla kaksi vanhempaa:

```
valuesum = cumsum(values);
maxsum = valuesum(length(values));
for k = 1:2
  tmp = find(valuesum >= maxsum*rand);
  pnt(k) = tmp(1);
end
```

Funktio `cumsum` laskee kumulatiiviset summat ja `find` palauttaa tosia arvoja vastaavat vektorin indeksit.

Seuraava rutiini (listaus 10.3) tekee risteyksen yllä valituille vanhemmille käyttäen yhtä katkaisukohtaa. Lisäksi tuotetaan mutaatioita pienellä todennäköisyydellä `mutprob`:

```
place=round((bitlen-1)*rand);
child(1,:)=[pop(pnt(1),1:place), pop(pnt(2),(place+1):bits)];
child(2,:)=[pop(pnt(2),1:place), pop(pnt(1),(place+1):bits)];
child=xor(child, mutprob >= rand(2,bits));
```

Uusi populaatio voidaan tuottaa joko yksi risteytys kerrallaan (steady-state replacement) tai korvaamalla koko populaatio (generational replacement). Lisäksi voidaan käyttää *elitismiä*: paras alkio säilyy aina seuraavaan sukupolveen. Tämä mekanismi saattaa toisinaan johtaa liian aikaiseen suppenemiseen kohti lokaalia maksimia.

Listauksessa 10.3 on esitetty algoritmi, jossa korvataan koko populaatio ja käytetään elitismiä: huonoin uusi alkio korvataan edellisen sukupolven parhaalla.

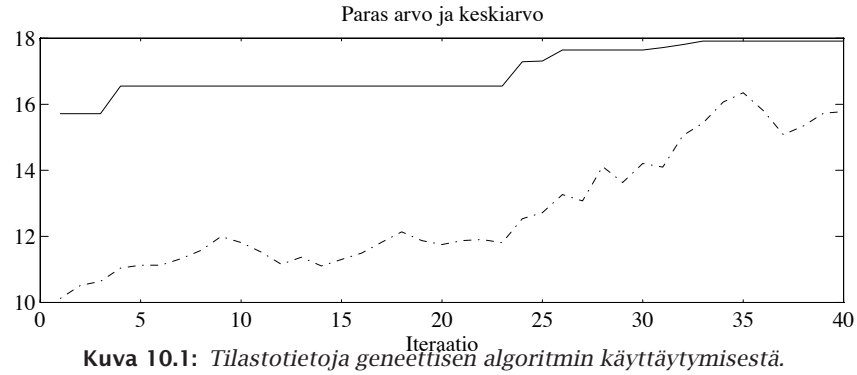
10.6 Geneettisen algoritmin simulointi

Seuraavassa simuloidaan geneettistä algoritmia listauksen 10.3 funktiolla `gaSim`:

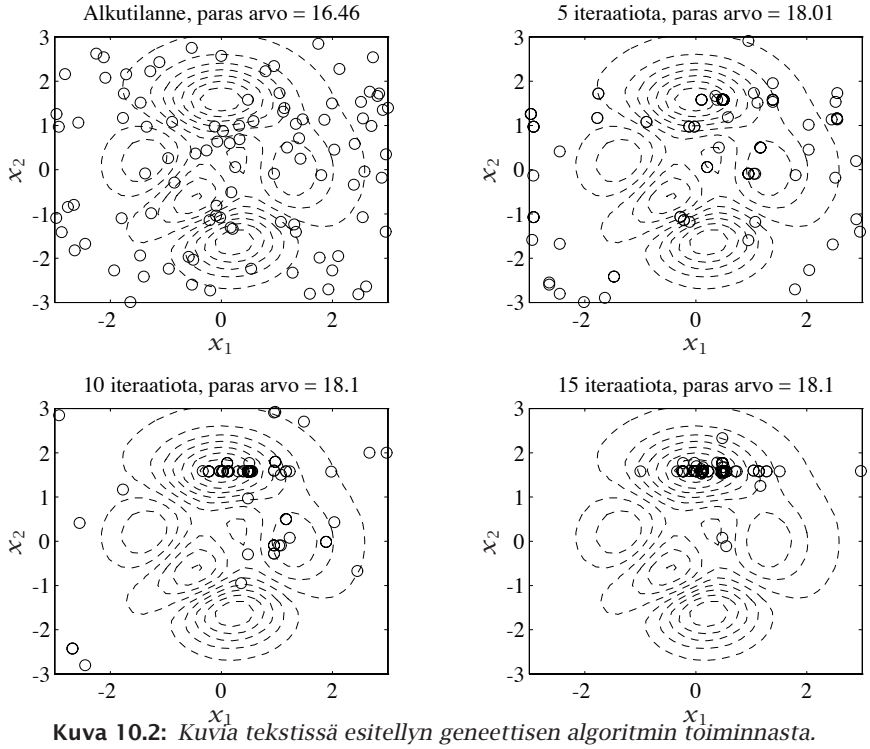
```
npop = 50; nbits = 20;
rand('seed',sum(100*clock));
initpop = rand(npop, nbits) > 0.5;
crossP = 0.8; mutP = 0.01; genNr = 40;
[pop endval stats] = gaSim('gaFun',initpop,crossP,mutP,genNr);
```

Kuvassa 10.1 esitetyssä simulaatiossa populaatioon kuului 50 alkiota, kukin sisälsi 20 bittiä informaatiota. Kullakin iteraatiolla (yhteensä 40) korvattiin koko populaatio risteyttämällä alkioita keskenään. Lisäksi käytettiin elitismiä eli kunkin kierroksen paras alkio säilyi seuraavalle kierrokselle. Risteytymistodennäköisyys oli 80% ja bittimutaation todennäköisyys 1%. Kuvasta nähdään, että parhaan alkion arvo parani hyppäyksittäin ja keskiarvossa oli runsaasti tilastollista vaihtelua.

Kuvassa 10.2 kuului populaatioon 100 alkiota, joista kukin sisälsi 20 bittiä informaatiota. Kullakin iteraatiolla korvattiin koko populaatio risteytyksil-



Kuva 10.1: *Tilastotietoja geneettisen algoritmin käyttäytymisestä.*



Kuva 10.2: *Kuvia tekstissä esitellyn geneettisen algoritmin toiminnasta.*

lä, lisäksi käytettiin elitismiä. Populaation alkiot sijoittuvat varsin nopeasti maksimikohdan ($x = -0.0093, y = 1.5814, z = 18.106$) läheisyyteen. Lisäksi havaitaan, että x- ja y-akselien suunnissa maksimista löytyy runsaasti populaation alkioita. Tämä johtuu käytetystä koodauksesta (x- ja y-komponentit peräkkäin) ja siitä, että risteytys säilyttää hyviksi havaittuja koodivektorien palasia.

10.7 Lisätietoja

Kombinatoristen optimointitehtävien taustaa ja ratkaisumenetelmiä on käsitelty teoksessa *Combinatorial Optimization* [PS82]. Aihepiiriä on käsitelty myös teoksissa [Dav87, Dav91, Gol89, vLA88].

Tässä luvussa käsiteltiin vain aivan yksinkertaisimpia geneettisten algoritmien ominaisuuksia. Kirjallisuusluettelon teoksista [Dav87, Dav91, Gol89] saa lisätietoja aiheesta. Sarjajulkaisussa *Complex Systems* on ajankohtaisia raportteja ja tutkimustuloksia geneettisistä algoritmeista. Usenet-uutisryhmästä `comp.ai.genetic` löytyy asiaan liittyviä viestejä.

Geneettisiä algoritmeja voi tutkia Matlabin lisäksi erikoisohjelmistoilla kuten Genesis ja OOGA [Dav91]. Myös esimerkiksi Mathematica-ohjelmistoa voi käyttää pienimuotoisiin kokeiluihin [Fre93]. Toisaalta esimerkiksi Fortrantai C-kielellä kirjoitetut erikoisohjelmistot ovat tehokkaampia isoissa tehtävissä.

Geneettisiä algoritmeja on käytetty muilla menetelmillä vaikeasti ratkaistavissa optimointitehtävissä [Gol89]. Vaihtoehtoisia menetelmiä GA:lle ovat mm. perinteinen polytooppihaku, tilastolliset menetelmät kuten Smin1 sekä aligradienttimenetelmät. Myös seuraavassa luvussa esiteltävät jäähdytysmenetelmät ovat eräs vaihtoehto.

Kauppamatkustajan ongelman ratkaisua geneettisillä algoritmeilla on käsitelty teoksissa [Dav91, Dav90]. Kappaleessa 10.3 esitettyä koodaustapaa on käytetty mm. synkrotronisäteilylähteen optimoinnissa [HR93]. Muita esimerkkejä löytyy mm. teoksista [Dav91, Dav87, Gol89, Ala92].

Evoluutiostrategioita on käsitelty teoksessa [HB92] ja geneettistä ohjelmointia käsitellään teoksessa [Koz92].

CSC:n julkaisemista oppikirjasta *Fortran 90/95* löytyy erään geneettisen algoritmin toteutus. Tämä menetelmä sisältää myös seuraavassa luvussa esiteltävien jäähdytysmenetelmien piirteitä.

11 Jäähdytysmenetelmät

Seuraavassa esitellään jäähdytysmenetelmien toimintaperiaatteet ja annetaan joitakin tuloksia menetelmien käyttäytymisestä sekä muutamia sovelluskohteita. Lisäksi luvussa ratkaistaan epälineaarinen pienimmän neliösumman minimointitehtävä käyttämällä jäähdytysmenetelmään perustuvaa koodia.

11.1 Jäähdytysmenetelmät optimoinnissa

Jäähdytysmenetelmät (simulated annealing eli SA, suora suomennos ”simuloitu mellotus”) ovat tilastollisesta fysiikasta (jossa käytetään *Metropoliksen 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

$$P = \frac{1}{1 + \exp(\Delta f_{ij}/T)} \approx \exp(-\Delta f_{ij}/T), \tag{11.1}$$

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.

Perinteisistä SA-algoritmeista on myös kehitetty nopeutettuja versioita. Jäähdytysmenetelmiin voidaan lukea kuuluviksi myös eräät neuroverkkojen (neural networks) mahdollistamat optimointimenetelmät.

11.2 Jäähdytysmenetelmän toiminta

Jäähdytysmenetelmässä matkitaan materiaalifysiikassa paljon tutkittua sulan aineen hitaan jäähdyttämisen ongelmaa:

1. Kuumennetaan systeemiä, kunnes hiukkaset voivat järjestyä satunnaisesti (metallin sulattaminen).
2. Jäähdytetään systeemiä kyllin hitaasti, jolloin hiukkaset asettuvat matalan energian tiloihin (metallihilan kiteytyminen).

Sovellettaessa jäähdytysmenetelmää optimointiin vastaa systeemin tilaa jokin ratkaisuehdokas (äärellisdimensioinen vektori). Ehdokas uudeksi ratkaisuksi saadaan muuttamalla satunnaisesti nykyistä ratkaisuehdokasta hiukan eli valitsemalla jokin ”lähellä” oleva ratkaisuehdokas nykyisen sijaan.

Alussa systeemin ”lämpötila” on korkea eli voidaan valita nykyistä huonompiakin ratkaisuvaihtoehtoja suurella todennäköisyydellä. Lämpötilaa lasketaan hiljalleen, jolloin muutoksia tapahtuu vähemmän ja vähemmän, kunnes löydetään ”jähmettynyt” ratkaisu.

Jäähdytysmenetelmän toteutuksessa täytyy huomioida seuraavat tekijät:

1. käytetty *vierailujakauma* (visiting distribution), joka kuvaa todennäköisyyttä siirtymiselle nykyisestä ratkaisuvektorista muutettuun ratkaisuvektoriin,
2. *jäähdytysaikataulu* (cooling schedule), joka määrää millä tavoin lämpötilaa muutetaan jäähdytyksen aikana,
3. *hyväksymistodennäköisyydet* (acceptance probabilities), jotka yleensä riippuvat lämpötilasta T,
4. *tasapainotilan tuottaminen* kussakin lämpötilassa.

Vierailujakaumalla tarkoitetaan nykyiselle ratkaisuvektorille tehtävien muutosten todennäköisyysjakaumaa. Diskreeteissä optimointitehtävissä vierailujakauma on diskreetti ja voi pysyä samana koko jäähdytyksen ajan. Mahdollisten tilasiirtymien joukkoa kutsutaan ratkaisuvektorin *naapuristoksi* (neighbourhood). Reaalimuuttujien optimointitehtävissä vierailujakauma on jatkuva ja riippuu jäähdytysmenetelmän kunkin hetkisestä lämpötilasta: lämpötilan laskiessa todennäköisyysjakaumasta generoitujen askelten pituudet yleensä lyhenevät.

Esimerkki 11.2.1 Kauppamatkustajan ongelmaa ratkottaessa (esimerkki 10.3.1 sivulla 164) voidaan käyttää naapuristoa, joka syntyy vaihtamalla reitin kaksi kaupunkia keskenään. Esimerkiksi reitistä (a, c, e, d, b) saadaan reitti (a, d, e, c, b) vaihtamalla kaupunkien c ja d järjestykset reitissä.

Jäähdytysmenetelmän tehokkuus riippuu jäähdytysnopeuden valinnasta. Globaali optimi löydetään varmasti vain äärettömän hitaalla jäähdytyksellä. Tehtävä on myös koodattava sopivaan esitysmuotoon. Lisäksi jäähdytysmenetelmän suppenemisnopeuden analysointi on melko hankalaa (kappale 11.3).

Jäähdytysmenetelmällä optimoitava funktio voi olla epäjatkuva. Lisäksi jäähdytysmenetelmä tuottaa vaikeissa kombinatorisissa (2^n tai $n!$ vaihtoehtoa) optimointitehtävissä yleensä kohtuullisen hyviä ratkaisuja.

11.2.1 Perusalgoritmi

Olkoon tehtävänä minimoida alhaalta rajoitettua funktiota $f(\mathbf{x})$. Olkoon $\Delta f_{ij} = f(\mathbf{x}^j) - f(\mathbf{x}^i)$ kustannusten muutos, kun valitaan vektori $\mathbf{x}^j$ nykyisen ratkaisuehdokkaan $\mathbf{x}^i$ sijaan. Vektorin $\mathbf{x}^j$ hyväksymistodennäköisyys riippuu jäähdytysmenetelmässä kustannusten muutoksesta Δf_{ij}.

Algoritmi 11.2.1 (Jäähdytysmenetelmä)

valitse alkukonfiguraatio $\mathbf{x}^1$ ja korkea lämpötila $T(1)$
aseta $k = 1$ ja $t = 1$
repeat
  repeat
    hae uusi tilavektori $\mathbf{y}$ muuttamalla vektoria $\mathbf{x}^k$ satunnaisesti
      käytetyn vierailujakauman mukaisesti
    laske kohdefunktion muutos $\Delta f = f(\mathbf{y}) - f(\mathbf{x}^k)$
    if $\Delta f \leq 0$
      $\mathbf{x}^{k+1} = \mathbf{y}$
    else
      aseta todennäköisyydellä $e^{-\Delta f / T(t)}$
        $\mathbf{x}^{k+1} = \mathbf{y}$ ja muuten $\mathbf{x}^{k+1} = \mathbf{x}^k$
    end
    $k = k + 1$
  until lähestytään tasapainotilaa
  pienennä lämpötilaa: $T(t + 1) \in [0, T(t))$
  aseta $t = t + 1$
until systeemissä ei tapahdu muutoksia

Ratkaisuvektorin satunnaisella muutoksella tarkoitetaan siirtymistä johonkin naapurikonfiguraatioon käytetyn vierailujakauman mukaisesti. Keskeinen osa toimivaa jäähdytysmenetelmää onkin sopivan rakenteen löytäminen naapuristolle, josta uusi ratkaisuehdokas valitaan.

Esitetyssä algoritmissa käytettiin eksponenttifunktiosta saatavaa todennäköisyyttä (ns. Boltzmannin jäähdytys). Alussa keinotekoinen lämpötila $T(t)$ on suuri, jolloin todennäköisyys valita nykyistä huonompi ratkaisu on suuri. Lämpötilaa $T(t)$ pienennettäessä muutoksia tapahtuu yhä harvemmin, kunnes toivottavasti päädytään tehtävän globaaliin minimiin.

Jäähdytysmenetelmissä pyritään saavuttamaan mahdollisimman nopea jäähdytysnopeus ja samalla välttämään paikalliseen minimiin jäämistä. Nämä pyrkimykset ovat vastakkaisia toistensa kanssa: mitä suurempaa luotettavuutta menetelmältä halutaan, sitä hitaammaksi jäähdytysnopeus on valittava.

11.3 Kombinatoriset optimointitehtävät

Oletetaan, että optimoitava tilavektori $\mathbf{x}$ saa arvoja äärellisestä joukosta $\mathcal{R}$, jonka alkioiden lukumäärä olkoon $r < \infty$. Kohdefunktio f on siten kuvaus $\mathcal{R} \mapsto \mathbb{R}$. Seuraavassa otetaan lisäksi käyttöön indeksit i ja j siten, että tilavektorit $\mathbf{x}^i$ ja $\mathbf{x}^j$ kuuluvat joukkoon $\mathcal{R}$.

Jäähdytysmenetelmän toimintaa voi analysoida mm. Markovin ketjujen avulla. Jäähdytysmenetelmässä muutetaan hiukan nykyistä tilavektoria sopivasta jakaumasta peräisin olevalla todennäköisyydellä $P_{ij}(k-1,k)$. Tämä ehdollinen todennäköisyys kuvaa iteraatiolla k tapahtuvan tilasiirtymän $\mathbf{x}^i \to \mathbf{x}^j$ todennäköisyyttä. Olkoon vektori $\mathbf{x}^k$ systeemin tilavektori k:n peräkkäisen tilasiirtymän jälkeen.

Tilasiirtymiä kuvaa $r \times r$ -kokoinen matriisi $P(k-1,k)$. Tilasiirtymien todennäköisyydet riippuvat lämpötilaparametrin T arvosta. Jos T on vakio, syntyvä Markovin ketju on homogeeninen (P ei riipu k:sta) ja matriisi P voidaan määritellä seuraavasti:

$$P_{ij}(T) = \begin{cases} G_{ij}(T)A_{ij}(T) & \text{kaikilla } i \neq j, \\ 1 - \sum_{l=1, l\neq i}^{r} G_{il}(T)A_{il}(T) & \text{kun } i = j. \end{cases} \tag{11.2}$$

Tässä G_{ij} kuvaa tilavektorin $\mathbf{x}^j$ generoitumisen todennäköisyyttä, kun alkuperäinen tilavektori on $\mathbf{x}^i$. Matriisi A_{ij} puolestaan kuvaa siirtymän $\mathbf{x}^i \to \mathbf{x}^j$ hyväksymistodennäköisyyttä.

Määritelmän (11.2) perusteella matriisi $P(T)$ on *stokastinen matriisi*, sillä $\sum_j P_{ij}(T) = 1$ kaikilla i. Edellisen tarkastelun perusteella voidaan jäähdytysmenetelmät jakaa kahteen luokkaan:

1. Homogeeniset algoritmit: tilasiirtymät tapahtuvat aina vakiolämpötilassa T.

2. Epähomogeeniset algoritmit: lämpötilaa T muutetaan tilasiirtymien välillä.

Jäähdytysmenetelmä löytää tehtävän globaalin minimin, mikäli K:n siirtymän jälkeen pätee

$$\mathrm{P}(\mathbf{x}^K \in \mathcal{R}^*) = 1, \tag{11.3}$$

missä $\mathcal{R}^*$ globaalien minimipisteiden joukko.

Pyrittäessä lähelle optimaalista ratkaisua voidaan joutua käymään läpi eksponentiaalinen määrä siirtymiä. Isoissa tehtävissä tähän kuluu enemmän aikaa kuin kaikkien eri ratkaisuvaihtoehtojen luettelemiseen.

Jos ratkaisun ei tarvitse olla mielivaltaisen lähellä optimia, voi jäähdytysmenetelmän tarvitsemien siirtymien lukumäärä kasvaa polynomisesti, joten ne voivat olla käyttökelpoisia kombinatoristen optimointitehtävien likimääräisiä ratkaisumenetelmiä. Polynominen laskenta-aika saadaan valitsemalla sellainen jäähdytysnopeus, että tilojen tasapainojakaumaa saadaan approksimoitua käyttämällä sopivan pientä määrää tilasiirtymiä. Takeita globaalin optimin löytymiselle ei voida antaa, mutta todennäköisyys ”kohtuullisen hyvän” ratkaisun löytymiselle on kuitenkin riittävä moniin käytännön tehtäviin.

11.4 Globaali optimointi

Edellisessä kappaleessa analysoitiin jäähdytysmenetelmän toimintaa diskreettien optimointitehtävien tapauksissa. Jatkuvissa optimointitehtävissä tilanne on toinen, koska nykyisestä konfiguraatiosta voidaan päästä uuteen konfiguraatioon vaikkapa lisäämällä kuhunkin parametrivektorin komponenttiin x_i reaaliluku Δx_i. Täten vierailujakauma on jatkuva ja menetelmän analyysiä varten täytyy ensin valita menetelmässä käytetty jakaumafunktio.

Eräs mahdollisuus on valita vierailujakaumaksi Gaussin jakauma, jolloin vierailujakauman tiheysfunktiolle g pätee

$$\mathrm{g}(\Delta\mathbf{x}, t) = \frac{\exp(-\|\Delta\mathbf{x}\|^2)}{(2\pi T(t))^{n/2}} = \prod_{i=1}^{n} \frac{\exp(-(\Delta x_i)^2)}{\sqrt{2\pi T(t)}}, \tag{11.4}$$

missä n on tehtävän dimensio. Tiheysfunktio on normeerattu eli pätee

$$\int_{\delta_1=-\infty}^{\infty} \cdots \int_{\delta_n=-\infty}^{\infty} \mathrm{g}(\delta, t)\, d\delta_1 \cdots d\delta_n = 1,$$

missä arvot $\delta_i,\ i = 1, \ldots, n$ ovat tilavektorin komponentteihin lisättyjä askeleita. Tätä jakaumaa käytettäessä on osoitettu, että jäähdytysaikatauluksi on syytä valita

$$T(t) = \frac{T(1)}{\ln t}, \tag{11.5}$$

missä aloituslämpötilan $T(1)$ tulee olla riittävän suuri. Kyseessä on siis logaritminen jäähdytysaikataulu, joka on varsin hidas.

Seuraavassa hahmotellaan jäähdytysmenetelmän analyysiä jatkuvaparametrisessa tapauksessa. Suppenemistodistuksessa lasketaan todennäköisyydet sille, ettei globaalin optimipisteen $\mathbf{x}^*$ läheisyydessä käydä jäähdytyksen kuluessa.

Olkoon joukko $U(\mathbf{x}^*)$ n-ulotteinen pallo, jonka halkaisija on d ja keskipiste $\mathbf{x}^* \in \mathbb{R}^n$. Otetaan käyttöön vierailujakauman tiheysfunktiota $\mathrm{g}(\Delta\mathbf{x}, t)$ vastaava todennäköisyysmitta m_t, jolloin $\mathrm{m}_t(U(\mathbf{x}^*))$ on vierailutodennäköisyys joukossa $U(\mathbf{x}^*)$ ajanhetkellä t. Tällöin todennäköisyys olla joukon $U(\mathbf{x}^*)$ ulkopuolella on $1 - \mathrm{m}_t(U(\mathbf{x}^*))$. Asetetaan

$$\prod_{t=1}^{\infty}(1 - \mathrm{m}_t(U(\mathbf{x}^*))) = 0. \tag{11.6}$$

Ottamalla kummaltakin puolelta logaritmi ja arvioimalla tulosta Taylorin sarjakehitelmällä saadaan yhtäpitävä ehto

$$\sum_{t=1}^{\infty} \mathrm{m}_t(U(\mathbf{x}^*)) = \infty. \tag{11.7}$$

Käyttämällä vierailujakaumana Gaussin jakaumaa ja logaritmista jäähdytysaikataulua saadaan arvio [SH87]:

$$m_t(U(\mathbf{x}^*)) \approx \frac{\exp(-d^2/T(t))}{T(t)^{-n/2}}, \tag{11.8}$$

mistä saadaan edelleen arvio

$$\sum_{t=1}^{\infty} \mathrm{m}_t(U(\mathbf{x}^*)) \geq \sum_{t=1}^{\infty} \exp(-\ln t) = \sum_{t=1}^{\infty} \frac{1}{t} = \infty. \tag{11.9}$$

Täten menetelmälle saadaan jonkinlainen varmuus suppenemisesta (ns. heikko suppeneminen eli suppeneminen tilastollisessa mielessä), vaikkakin suppenemiseen vaaditaan äärettömän pitkä jäähdytys.

Nopeassa jäähdytysmenetelmässä (fast annealing eli FA [SH87]) korvataan vierailujakauma (11.4) Cauchyn jakaumalla

$$\mathrm{g}(\Delta\mathbf{x}, t) = \frac{T(t)}{(\|\Delta\mathbf{x}\|^2 + T(t)^2)^{(n+1)/2}}, \tag{11.10}$$

joka tuottaa suuremmalla todennäköisyydellä pitkiä askeleita. Tällöin jäähdytysnopeudeksi voidaan valita

$$T(t) = \frac{T(1)}{t} \tag{11.11}$$

ja saadaan vastaava suppenemistulos kuin edellä.

Muitakin vaihtoehtoja vierailujakauman valitsemiseen on. Ohjelmisto ASA (adaptive simulated annealing) minimoi laatikkorajoitteet $\mathbf{x}^l \leq \mathbf{x} \leq \mathbf{x}^u$ sisältävää optimointitehtävää, ja uusi iteraatiopiste saadaan seuraavasti:

$$x_i^{k+1} = x_i^k + y_i(x_i^u - x_i^l),\ i = 1, \ldots, n, \tag{11.12}$$

missä satunnaismuuttujan $y_i \in [-1,1],\ i = 1,\ldots,n$ tiheysfunktio on

$$g(\mathbf{y},t) = \prod_{i=1}^{n} \frac{1}{2(|y_i| + T_i(t))\ln(1+1/T_i(t))} = \prod_{i=1}^{n} g_i(y_i,t). \tag{11.13}$$

Siis tiheysfunktio on tulo alkioittaisista tiheysfunktioista $g_i(y_i,t)$. Menetelmässä pidetään yllä erillisiä vierailujakauman lämpötiloja $T_i(t)$ kaikille optimoitaville muuttujille x_i. Uusia alkioita x_i generoidaan niin kauan kunnes yhtälö (11.12) tuottaa laatikkorajoitteet toteuttavan vektorin $\mathbf{x}$ alkion. Menetelmässä tarvittavat satunnaismuuttujat y_i voidaan generoida tasajakautuneiden satunnaismuuttujien $r_i \in [0,1]$ avulla:

$$y_i = \mathrm{sgn}(r_i - \frac{1}{2})\, T_i(t) \left((1 + \frac{1}{T_i(t)})^{|2r_i - 1|} - 1 \right).$$

Tälle vierailujakaumalle voidaan käyttää jäähdytysaikataulua

$$T_i(t) = T_i(1)\ \exp(-c_i\, t^{1/n}), \tag{11.14}$$

missä $c_i > 0$ on menetelmän parametri. Jäähdytysnopeus riippuu tehtävän dimensiosta n: mitä suurempi on n, sitä hitaammin täytyy lämpötilaa laskea. ASA-ohjelmisto tutkii aika ajoin löydetyn minimiarvon herkkyyttä kunkin muuttujan x_i suhteen ja muuttaa lämpötiloja $T_i(t)$ tämän mukaisesti.

11.5 Pienimmän neliösumman sovitus

Seuraavassa ratkaistaan epälineaarinen pienimmän neliösumman minimointitehtävä käyttämällä Netlib-verkkoarkistosta löytyvää ohjelmistoa. Käytettyä menetelmää on kuvattu viitteissä [GFR94, CMMR87].

Tässä menetelmässä korvataan vierailujakauma askelpituuksilla s_i, joita muutetaan suuremmiksi tai pienemmiksi sen mukaan kuinka usein tilavektori $\mathbf{x} + \mathbf{s}$ hyväksytään uudeksi iteraatiopisteeksi. Menetelmä käyttää eksponentiaalista jäähdytysaikataulua $T(t+1) = cT(t)$, missä $c \in (0,1)$. Tällaista aikataulua kutsutaan usein *pikajäähdytykseksi* (simulated quenching). Algoritmissa pienennetään lämpötilaa hyvin nopeasti, eivätkä edellä esitetyt suppenemistulokset ole enää voimassa.

Esimerkki 11.5.1 Sovitetaan epälineaariseen malliin

$$\begin{aligned} y \;=\; & f(t,\mathbf{a}) + \epsilon = a_1 + a_2 t + a_3 t^2 + a_4 \exp(-(t-a_5)^2/(2a_6^2)) \\ & + a_7 \exp(-(t-a_8)^2/(2a_9^2)) + \epsilon \end{aligned} \tag{11.15}$$

mittausdata $(t_i, y_i),\ i = 1,\ldots,m$. Mallissa esiintyvä ϵ on virhetermi. Minimointitehtäväksi saadaan

$$\underset{\mathbf{a}}{\text{minimi}}\ F(\mathbf{a}) = \sum_{i=1}^{m} (y_i - f(t_i,\mathbf{a}))^2,\ \text{kun}\ \mathbf{a} \in \mathbb{R}^9, \tag{11.16}$$

eli kyseessä on yhdeksän vapaan muuttujan epälineaarinen minimointitehtävä.

Datapisteitä oli käytettävissä 200 kappaletta ja alkuarvauksena oli $\mathbf{a} = \mathbf{1}$. Sovitukselle annettiin laatikkorajoitteet $-5 \leq a_i \leq 10,\ i = 1, \ldots, 9$. Todelliset parametrit olivat

$$\mathbf{a} = (0.5, -0.4, 0.1, 6.0, -1.0, 1.1, 5.0, 2.1, 0.9)$$

ja mittausdata oli generoitu lisäämällä tarkasta mallista saatuihin y-arvoihin tasajakautuneita satunnaislukuja väliltä $[-0.5, 0.5]$.

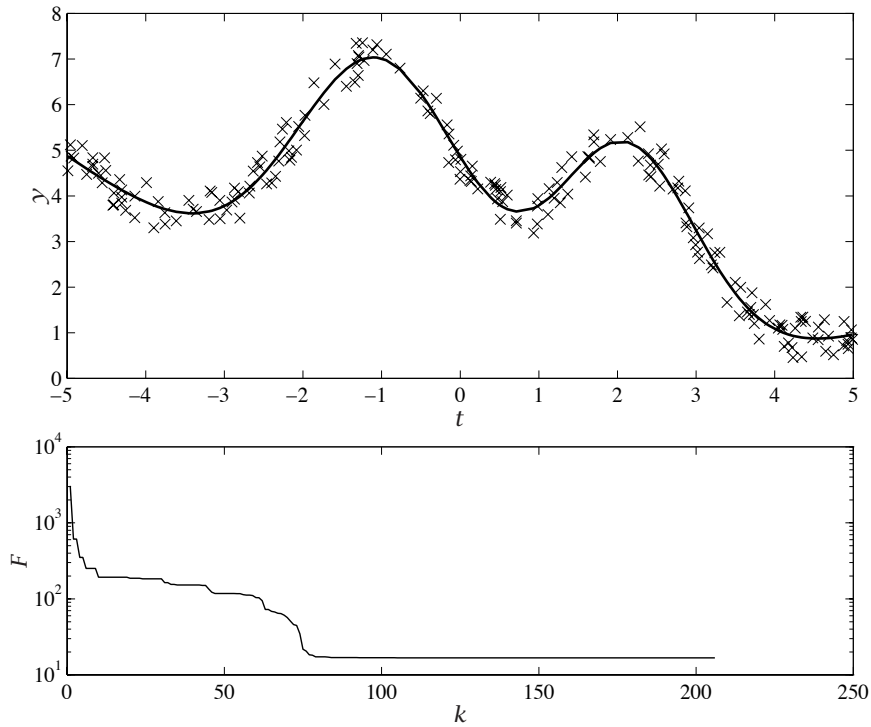
Mallin parametrien hakemiseen käytettiin Netlib-verkkokirjastosta löytyvää jäähdytysmenetelmään perustuvaa Fortran-koodia `simann.f` (versio 3.2).

Jäähdytysnopeudeksi asetettiin $c = 0.9$ ja alkulämpötilaksi 500. Askelpituuksiksi asetettiin alussa arvot $s_i = 2$. Kussakin lämpötilassa tehtiin 900 tilasiirtymää tasapainotilaan pääsemiseksi. Tällöin sovitukseen tarkkuudella 10^{-6} tarvittiin $900 \times 206 = 185400$ funktion arvon laskemista. Residuaaliksi lopussa saatiin $F(\mathbf{a}^*) \approx 16.7$.

Edellä kuvatulla tavalla saadaan mallin parametreiksi

$$\mathbf{a}^* \approx (0.87, -0.39, 0.08, 5.7, -0.99, 1.1, 4.7, 2.1, 0.89).$$

Alkuperäisillä parametrien arvoilla saadaan residuaali $F(\mathbf{a}) \approx 17.1$ eli huonompi kuin jäähdytysmenetelmän tuottamalla tuloksella. Menetelmä toimii varsin



Kuva 11.1: *Pienimmän neliösumman minimointitehtävässä käytetty data on merkitty symbolilla × ja sovitettu malli paksulla viivalla. Alemmassa kuvassa on esitetty virheen pieneneminen iteraatioiden funktiona.*

hyvin ottaen huomioon mittausdatan häiriöisyyden. Toisaalta mittauspisteitä oli käytettävissä kohtalaisen paljon.

Kuvassa 11.1 on esitetty mittausdata ja mittausdataan sovitettu malli. Sovituksesta saatua mallia kuvaa paksulla piirretty käyrä. Alemmassa kuvassa on esitetty minimointitehtävän kohdefunktion $F(\mathbf{a})$ arvon kehitys iteraatioiden funktiona logaritmisella asteikolla. Jäähdytyksen alkuvaiheissa pysähdytään joksikin aikaa paikalliseen minimiin, kunnes löydetään lopullista mallia vastaavat parametrien arvot.

Tämän tyyppisessä epälineaarisessa pienimmän neliösumman tehtävässä kannattaa pyrkiä hyödyntämään kaikkea tutkittavaan ilmiöön liittyvää tietoa. Tässäkin tapauksessa olisi ollut mahdollista antaa kohtuullisen hyvä alkuarvaus tarkastelemalla kuvassa 11.1 esitettyä mittausdataa. Suoraviivainen optimointi ei myöskään välttämättä anna parasta mahdollista ilmiötä kuvaavaa mallia. Tuloksien järkevyys on aina syytä tarkistaa.

11.6 Jäähdytysmenetelmän käyttökohteet

Jäähdytysmenetelmiä voi yrittää käyttää tehtäviin, joissa kohdefunktio on epäjatkuva tai epäsileä ja vahvasti epälineaarinen. Jos on mahdollista käyttää jotain tehokasta sileiden tai epäsileiden tehtävien ratkaisuohjelmistoa, kannattaa niin tehdä. Toisaalta jäähdytysmenetelmä voi olla käyttökelpoinen esimerkiksi isoissa kombinatorisissa tehtävissä, joissa kohtuullisen hyvä ratkaisu on riittävä.

Mikäli ratkaistavana on suuri kombinatorinen optimointitehtävä, josta on löydettävissä sopiva struktuuri (geneettinen koodaus tai jäähdytysmenetelmän naapuristorakenne), voi menetelmien tehokkuus olla varsin hyvä likimääräistä optimia haettaessa. Tyypillinen sovelluskohde on esimerkiksi kauppamatkustajan ongelma, vaikkakaan menetelmien tehokkuus hyvin suurissa tehtävissä ei ole erityisen kehuttava.

Yleisalgoritmeiksi ei jäähdytysmenetelmistä vielä ole ja toimivan energiafunktion, vierailujakauman ja jäähdytysnopeuden löytäminen voi vaatia monia kokeilukertoja. Lisäksi uuden ratkaisuehdokkaan generointimenetelmä vaikuttaa usein suuresti tehokkuuteen.

Jäähdytysmenetelmä saattaa perusmuodossaan olla hyvin hidas menetelmä, kuten luvussa 10 esitellyt geneettiset algoritmitkin. Kummankin menetelmän käyttökelpoisuus riippuu paljon käytetystä koodauksesta sekä tietenkin tehtävän rakenteen soveltumisesta ratkaisumenetelmän käyttöön.

Jos tehtävällä on monia paikallisia minimejä, voi koettaa useiden alkuarvausten käyttöä tai jotain monipuolisempaa globaalin optimoinnin menetelmää. Yleensä näissä on yhdistetty paikalliseen optimointiin tarkoitettu menetelmä ja jokin globaali strategia (luku 9).

Esimerkki 11.6.1 Käytännön esimerkki jäähdytysmenetelmien käytöstä verrattuna muihin menetelmiin on suprajohteiden Ginzburg-Landau -yhtälöiden ratkaiseminen. Artikkelissa [DGR90] on ratkaistu pieniä kaksiulotteisia tehtäviä jäähdytysmenetelmällä. Artikkelissa [WH91] on käytetty jyrkimmän suunnan menetelmää, jolla on voitu tehostaa ratkaisemista. Artikkelissa [GSB+92] on käytetty gradienttimenetelmiä, jolloin on voitu ratkaista vaikeampia ja suurempia tehtäviä. Uusimpia tuloksia on esitelty artikkelissa [JP93], jossa on ratkottu kolmiulotteista tehtävää käyttäen katkaistua Newtonin menetelmää.

Jäähdytysmenetelmällä saatiin tässä tehtävässä ensimmäiset tulokset ja voitiin osoittaa optimointiin perustuvan menetelmän käyttökelpoisuus. Suuremmissa tehtävissä käytettiin puolestaan gradienttimenetelmiä, joiden avulla ratkaisemiseen tarvittava aika saatiin hyväksyttävälle tasolle.

11.7 Lisätietoja

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. Artikkelissa [BL93] on käsitelty jäähdytysmenetelmien rinnakkaistamista. Jäähdytysmenetelmiin perustuvista ohjelmistoista on kerrottu liitteessä B.

Boltzmannin jäähdytystä on käsitelty teoksissa [Ing93, IR92]. Metropoliksen algoritmia on käsitelty teoksissa [Kan93, MRR+53]. Teoksissa [AK89, vLA88, OvG89] on esitetty asymptoottisia ($k \to \infty$) suppenemistuloksia jäähdytysmenetelmille.

CSC:n julkaisemista oppikirjasta *Fortran 90/95* löytyy erään sellaisen geneettisen algoritmin toteutus, joka sisältää myös jäähdytysmenetelmien piirteitä.

12 Derivaattojen laskeminen

Eräs tyypillisimpiä virheitä optimointitehtävien ratkaisussa on väärin lasketut funktion derivaatat. Derivaattojen oikeellisuus kannattaa aina varmistaa, oli sitten kyseessä differenssiarvio tai analyyttisesti lasketut derivaatat. Seuraavassa on vertailtu eräitä tapoja laskea tarvittavat derivaatat.

12.1 Gradienttimenetelmät optimoinnissa

Jos optimointitehtävän ratkaisemiseen käytetään luvussa 6 esiteltyjä gradienttimenetelmiä, tarvitaan arviot funktion derivaatoille esimerkiksi laskettaessa gradienttivektori tai Hessen matriisi. Tyypillinen gradienttimenetelmän iteraatio on seuraavaa muotoa:

Algoritmi 12.1.1 (Gradienttimenetelmän iteraatio)

Ratkaise hakusuunta $\mathbf{p}^k$ yhtälöryhmästä $H_k\mathbf{p}^k = -\nabla f(\mathbf{x}^k)$.
Hae uusi piste ratkaisuavaruudesta: $\mathbf{x}^{k+1} = \mathbf{x}^k + \lambda_k\mathbf{p}^k$, $\lambda > 0$.
Tutki optimaalisuutta: jos $\|\nabla f(\mathbf{x}^{k+1})\| \leq \epsilon$, voit lopettaa.

Tässä iteraatiossa tarvitaan kohdefunktion gradientin arvot $\nabla f(\mathbf{x}^k)$. Jos kyseessä on Newtonin menetelmä, tarvitaan myös kohdefunktion toisia derivaattoja Hessen matriisin $H_k = \nabla\nabla^T f(\mathbf{x}^k)$ muodostamiseen. Sekanttimenetelmissä tarvitaan vain kohdefunktion gradienttivektorit. Toisaalta Hessen matriisia voi arvioida analyyttisesti laskettujen gradienttivektoreiden avulla.

12.1.1 Suuret tehtävät

Derivaattojen laskeminen suurten tehtävien tapauksessa on usein aikaavievää ja hankalaa. Jos käytetään käsin laskettuja analyyttisiä derivaattalausekkeita, niiden tarkistaminen vie myös oman aikansa. Eräs mahdollinen tapa tarkistaa derivaattalausekkeet on käyttää monista ohjelmistoista löytyviä

tarkistusrutiineja, jotka vertaavat derivaattakoodin tuottamia arvoja differenssiarvioista saataviin arvoihin. Kuitenkaan tämän tyyppinen vertailu ei voi löytää kaikkia mahdollisia virheitä.

Seuraavassa esiteltyjä symbolienkäsittelyjärjestelmien ja automaattiseen derivointiin kehitettyjen välineiden tarjoamia mahdollisuuksia kannattaa käyttää apuna suurten tehtävien käsittelyssä. Isoissa tehtävissä voi usein hyödyntää tehtävän erikoisominaisuuksia (symmetrisyys ja rakenteellisuus), jolloin derivaatat voidaan usein laskea huomattavasti tehokkaammin.

Myös kappaleessa 12.3 esiteltäviä differenssiapproksimaatioiden laskukaavoja voidaan käyttää suurissa tehtävissä. Jos Hessen matriisi on harva, voidaan sen rakennetta hyödyntää differenssiarvioita laskettaessa (esimerkki 12.3.1 sivulla 191).

12.1.2 Symbolinen derivointi

Mikäli optimointitehtävän ratkaisemiseen tarvittavat derivaatat voidaan laskea analyyttisesti, kannattaa se useimmiten tehdä. Jos optimoitava funktio on esitettävissä symbolisessa muodossa, voi derivaatat laskea symbolienkäsittelyjärjestelmillä (esimerkiksi Macsyma, Maple tai Mathematica) ja tulostaa vaikkapa Fortran-koodina tiedostoon.

Listaus 12.1: *Mathematica-rutiinit symbolisessa muodossa esitettyjen funktioiden derivaattojen laskemiseksi.*

```
Grad[f_,var_List] := D[f, #]& /@ var;
Div[F_List, var_List] := Inner[D, F, var, Plus];
Lap[f_, var_List] := Div[Grad[f, var], var];
Hesse[f_, var_List] := Outer[D, Grad[f, var], var];
Jac[F_List, var_List] := Outer[D, F, var];
```

Esimerkki 12.1.1 Listauksessa 12.1 on esitetty muutamia Mathematica-rutiineita funktioiden derivaattojen laskemiseen symbolisesti.

Rutiini `Grad` laskee gradienttivektorin, `Div` laskee divergenssin, `Lap` toimii Laplace-operaattorina, `Hesse` laskee Hessen matriisin ja `Jac` Jacobin matriisin. Vastaavat rutiinit voi kirjoittaa muillakin symbolienkäsittelyjärjestelmillä.

Lasketaan Rosenbrockin funktion gradientti ja Hessen matriisi Mathematicalla käyttäen rutiineita `Grad` ja `Hesse`. Olkoot Mathematica-rutiinit tiedostossa `Derivaatat.m`.

```
In[1]:= <<Derivaatat.m

In[2]:= f = 100 (x2 - x1^2)^2 + (1 - x1)^2
                2          2      2
Out[2]= (1 - x1)  + 100 (-x1  + x2)

In[3]:= gf = Simplify[Grad[f,{x1,x2}]]
                                3                          2
Out[3]= {-2 + 2 x1 + 400 x1  - 400 x1 x2, 200 (-x1  + x2)}

In[4]:= Hf = Simplify[Hesse[f,{x1,x2}]]
```

```
                                                  2
Out[4]= {{2 + 1200 x1  - 400 x2, -400 x1}, {-400 x1, 200}}
```

Lausekkeiden tulostamisessa voi myös käyttää apuna Mathematican muotoilukomentoja `FortranForm`, `TableForm` ja `TeXForm`, vaikkakin näissä rutiineissa on joitakin heikkouksia, eivätkä ne sellaisinaan riitä kaikkiin käyttötarkoituksiin.

Esimerkki 12.1.2 Listauksessa 12.2 on esimerkki derivaattojen laskemisesta käyttäen Maplen `linalg`-paketista löytyviä rutiineja. Aluksi lasketaan funktiolle

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} \cos(x_1 x_2) \\ \sin(x_1 + x_2) \end{pmatrix}$$

Jacobin matriisi ja divergenssi. Tämän jälkeen sovelletaan funktioon

$$f(\mathbf{x}) = x_1^3 \left(x_2^2 + 2\right)$$

gradienttioperaattoria, Laplace-operaattoria ja Hessen matriisin laskevaa rutiinia `hessian`.

Listaus 12.2: *Derivaattojen laskeminen Maplella symbolisessa muodossa esitetyille funktioille. Tässä käytetään Maplen mukana tulevasta `linalg`-paketista löytyviä rutiineja.*

```
with(linalg):

# Lasketaan vektoriarvoiselle funktiolle
#   Jacobin matriisi ja divergenssi
x := vector(2);
F := vector([ cos(x[1]*x[2]), sin(x[1]+x[2])]);
jac := jacobian(F,x);
div := diverge(F,x);

# Skalaariarvoisen funktion gradientti,
# Laplace-operaattori ja Hessen matriisi
f := (x[1]**3)*(2 + x[2]**2);
gradientti := grad(f,x);
lap := diverge(grad(f,x),x);
hesse := hessian(f,x);
```

Isoissa tehtävissä kannattaa tietenkin hyödyntää Hessen matriisin symmetrisyyttä laskemisessa ja tallettamisessa.

Suoraviivainen symbolienkäsittelyjärjestelmien käyttö voi tuottaa huomattavia vaikeuksia gradienttivektorin sekä Hessen ja Jacobin matriisien muodostamisessa. Jos kohdefunktio koostuu n:stä termistä, on muodostetussa derivaattalausekkeessa ennen sievennystä tyypillisesti luokkaa n^2 termiä. Täten monimutkaisten lausekkeiden suora symbolinen derivointi tuottaa hyvin

hankalasti käsiteltäviä lausekkeita. Symbolienkäsittelyjärjestelmiä voi myös yrittää käyttää syntyvien lausekkeiden sieventämiseen, mutta tulos ei välttämättä ole kovin hyvä ja vie usein paljon laskenta-aikaa. Lisäksi kohde- tai rajoitefunktioiden symbolisen lausekkeen esittäminen voi olla mahdotonta tai erittäin vaikeaa.

Derivaattojen laskemisen lisäksi voi symbolienkäsittelyjärjestelmiä siis käyttää saatujen lausekkeiden sieventämiseen ja optimointiin. Toinen lähestymistapa on *automaattinen derivoiminen* (kappale 12.2), jossa derivoidaan aliohjelmia. Lopputuloksena voidaan tuottaa esimerkiksi Fortran- tai C-koodia.

Automaattisen derivoinnin etuna on saadun koodin tiiviys ja tehokkuus. Lisäksi useat menetelmät tuottavat samalla virhearvion aliohjelmakoodin laskemalle funktion arvolle.

Kehitteillä on myös ympäristöjä, joissa voidaan linkittää symbolienkäsittelyjärjestelmä ja numeerinen laskentaympäristö. Tällaisia järjestelmiä on rakennettu mm. Mathematican ja Maplen ympärille.

12.2 Automaattinen derivoiminen

Automaattisella derivoinnilla (automatic differentiation, algorithmic differentiation) tarkoitetaan aliohjelman (siis tietokoneella laskettavan koodin) muodossa annetun funktion symbolista derivoimista.

Seuraavissa esimerkeissä käytetään hyväksi Maple-ohjelmistoa automaattisen derivoinnin havainnollistamiseksi.

Esimerkki 12.2.1 Lasketaan funktion

$$f(x) = c\,(a-x)^3 + (a-x)^2\,(b-x)^2$$

arvo käyttäen Maple-ohjelmiston ohjelmointikieltä:

```
f := proc(x)
  local s1,s2;
  s1 := a - x;
  s2 := b - x;
  c*s1^3 + (s1*s2)^2
end;
```

Tämä ohjelmakoodi voidaan ”derivoida” käyttäen ketjusääntöä

$$\frac{\partial f(x(u))}{\partial u} = \frac{\partial f(x(u))}{\partial x}\frac{\partial x(u)}{\partial u}. \tag{12.1}$$

Maplen komennolla `D(f)` saadaan funktion derivaatan $g(x) = df(x)/dx$ laskeva koodi:

```
g := proc(x)
  local s1,s2;
  s1 := a-x;
```

```
  s2 := b-x;
  -3*c*s1^2-2*s1*s2^2-2*s1^2*s2
end;
```

Tätä tapaa ohjelmakoodin derivoimiseksi kutsutaan *eteenpäin suuntautuvaksi* (forward mode). Pyrkimyksenä olisi, että tuotettu koodi olisi mahdollisimman tiivistä ja tehokasta. Tehokkuuden kannalta tulisi useassa eri paikassa koodia esiintyvät identtiset lausekkeet laskea vain yhteen kertaan käyttäen tilapäismuuttujia.

Edellä saatu koodi ei kuitenkaan ole kovin tehokas. Käyttämällä Maplen rutiinia `optimize` funktiokoodin optimointiin saadaan seuraava tulos:

```
> with(codegen):
> optimize(D(f));
proc(x)
local s1, s2, t1, t4;
    s1 := a - x; s2 := b - x; t1 := s1^2; t4 := s2^2;
    -3*c*t1 - 2*s1*t4 - 2*s2*t1
end proc
```

Tämä derivaatan laskeva koodi on jo varsin tehokas. Luonnollisesti monimutkaisemmissa lausekkeissa ja ohjelmakoodeissa on edellä kuvatun kaltainen koodin virittely tärkeämpää kuin tässä yksinkertaisessa esimerkissä.

Esimerkki 12.2.2 Seuraavassa on esimerkki kahden muuttujan funktion derivoimisesta. Tämä esimerkki ei toimi Maplen uusimmilla versiolla, sillä niiden mukana ei enää tule Maplen *share library* -arkistoa.

Maple-ohjelmalla kirjoitetusta Rosenbrockin funktion arvon laskevasta koodista voidaan generoida gradienttivektorin ja Hessen matriisin alkiot laskeva rutiini seuraavasti:

```
> with(share):
> read ``.sharename.`/autodiff/GRADIENT.m`;
> F := proc(x,y) 100*(y - x^2)^2 + (1 - x)^2 end;
F := proc(x,y) 100*(y-x^2)^2+(1-x)^2 end
> G := GRADIENT(F);
G := proc(x,y)
  RETURN(-400*(y-x^2)*x-2+2*x,200*y-200*x^2)
end
> H := GRADIENT(G);
H := proc(x,y)
  RETURN(1200*x^2-400*y+2,-400*x,-400*x,200)
end
```

Tässä käytettiin Maplen share library -rutiinia `GRADIENT` automaattiseen derivointiin, sillä esimerkissä 12.2.1 käytetty funktio `D` laskee derivaatan vain yhden muuttujan suhteen.

Automaattisessa derivoinnissa pyritään mahdollisimman tiiviiden ja tehokkaasti laskettavien lausekkeiden muodostamiseen. Edellinen esimerkki oli varsin yksinkertainen ja siinä selvittiin ilman paikallisia tilapäismuuttujia.

Esimerkki 12.2.3 Seuraavassa lasketaan funktion

$$f(x,y) = \sum_{i=0}^{n} a_i x \exp\left(-(y-p_i)/s_i^2\right)$$

gradienttivektori. Funktion arvon laskeva ohjelmakoodi sisältää silmukan.

```
> F := proc(x,y,a,p,s,n) local i,t;
  t := 0;
  for i from 0 to n do
    t := t + a[i]*x*exp(-(y-p[i])/s[i]^2);
  od;
  RETURN(t)
end:
> G := GRADIENT(F,[x,y]);
G := proc(x,y,a,p,s,n)
  local dt,i,t;
      dt := array(1 .. 2);
      dt[1] := 0;
      dt[2] := 0;
      t := 0;
      for i from 0 to n do
          dt[1] := dt[1]+a[i]*exp(-(y-p[i])/s[i]^2);
          dt[2] := dt[2]-a[i]*x/s[i]^2*
                   exp(-(y-p[i])/s[i]^2);
          t := t+a[i]*x*exp(-(y-p[i])/s[i]^2)
      od;
      RETURN(dt[1],dt[2])
  end
```

Tämä esimerkki ei toimi Maplen uusimmilla versiolla, sillä niiden mukana ei enää tule Maplen *share library* -arkistoa.

Funktion `GRADIENT` tuottama ohjelmakoodi laskee osittaisderivaatat muuttujien `x` ja `y` suhteen. Syntynyttä ohjelmakoodia voisi vielä nopeuttaa. Silmukassa laskettavat paikallisen muuttujan `t` arvot ovat tarpeettomia derivaattojen laskemisen kannalta.

12.3 Differenssiarvioiden käyttö

Käytettäessä derivaatoille differenssiarvioita on syytä huomioida numeerisen tarkkuuden vaikutukset arvioiden laskemiseen. Useat valmisohjelmistot (esimerkiksi aliohjelmakirjastot IMSL ja NAG) osaavat arvioida haluttaessa derivaattoja differensseillä, mutta tähän kuluu jonkin verran ylimääräistä laskutyötä. Monet ohjelmistot sisältävät myös derivaattojen oikeellisuutta tarkistavia rutiineja.

Huomautus 12.3.1 Askelpituuden h valinnalla on suuri merkitys differenssiarvion tarkkuuden kannalta: liian pieni askelpituus tuottaa numeerisesti

epätarkkoja tuloksia ja liian suuri antaa epätäsmällisen arvion derivaatalle. Mikäli numeroarvot ovat suuruusluokkaa 1, hyvä ensimmäinen arvio askelpituudelle on $h = \sqrt{\epsilon_M}$ (ϵ_M on konevakio), joka on 32-bittisessä aritmetiikassa $\approx 10^{-4}$.

12.3.1 Gradienttivektorin differenssiarviot

Funktion $f(\mathbf{x})$ gradienttivektorille $\nabla f(\mathbf{x}) = (n_1(\mathbf{x}), \ldots, n_n(\mathbf{x}))^T$ voidaan tehdä differenssiarviot:

$$\begin{cases} n_i = \dfrac{f(\mathbf{x} + h_i\mathbf{e}^i) - f(\mathbf{x})}{h_i}, & \text{eteenpäin laskettu differenssi,} \\ n_i = \dfrac{f(\mathbf{x}) - f(\mathbf{x} - h_i\mathbf{e}^i)}{h_i}, & \text{taaksepäin laskettu differenssi,} \\ n_i = \dfrac{f(\mathbf{x} + h_i\mathbf{e}^i) - f(\mathbf{x} - h_i\mathbf{e}^i)}{2h_i}, & \text{keskeisdifferenssi.} \end{cases} \tag{12.2}$$

Tässä $\mathbf{e}^i$ on i:s yksikkövektori: $\mathbf{e}^i = (0, \ldots, 0, e_i = 1, 0, \ldots, 0)^T$. Parametri h_i on suuntaan $\mathbf{e}^i$ valittu askelpituus.

Keskeisdifferenssin virhe on luokkaa $\mathcal{O}(h_i^2)$ ja muiden arvioiden virhe on luokkaa $\mathcal{O}(h_i)$.

12.3.2 Hessen matriisin differenssiarviot

Toisten derivaattojen arvioiminen differensseillä on melko epätarkkaa. Mikäli derivaattoja ei voi laskea tarkasti, on useimmiten tehokkainta käyttää menetelmää, jossa ei tarvita toisten derivaattojen differenssiarvioita (esimerkiksi sekanttimenetelmät). Seuraavassa kuitenkin esitellään muutamia tapoja laskea differenssiarvio Hessen matriisille. Käyttökelpoinen menetelmä on gradienttivektoria käyttävä Hessen matriisin arvio, johon perustuu *diskreetti Newtonin menetelmä*.

Jos Hessen matriisille $H(\mathbf{x}) = \nabla\nabla^T f(\mathbf{x})$ halutaan differenssiarvio, voidaan matriisin alkioille H_{ij} käyttää esimerkiksi kaavoja

$$H_{ii} = \frac{f(\mathbf{x} + h\mathbf{e}^i) + f(\mathbf{x} - h\mathbf{e}^i) - 2f(\mathbf{x})}{h^2}, \tag{12.3}$$

$$H_{ij} = \frac{f(\mathbf{x} + h\mathbf{e}^i + h\mathbf{e}^j) - f(\mathbf{x} + h\mathbf{e}^i) - f(\mathbf{x} + h\mathbf{e}^j) + f(\mathbf{x})}{h^2}. \tag{12.4}$$

Tässä on oletettu yksinkertaisuuden vuoksi, että differenssiaskel h on sama kaikkiin suuntiin $\mathbf{e}^i$.

Jos funktion gradientti $\nabla f(\mathbf{x})$ voidaan laskea analyyttisesti, Hessen matriisin $H(\mathbf{x}) = (H_{ij})$ sarakekomponenteille $H_j = (H_{1j}, H_{2j}, \ldots, H_{nj})^T$ voi kokeilla gradienttia käyttävää differenssiapproksimaatiota

$$H_j = \frac{\nabla f(\mathbf{x} + h\mathbf{e}^j) - \nabla f(\mathbf{x})}{h}. \tag{12.5}$$

Jos optimointitehtävässä esiintyvä Hessen matriisi on harva, voidaan sille laskea differenssiarvio hyvin tehokkaasti. Hyvinkin suuria Hessen matriiseja voidaan arvioida muutaman sopivissa pisteissä lasketun gradienttivektorin avulla. Seuraavassa on tästä esimerkki.

Esimerkki 12.3.1 *Diskreetissä Newtonin menetelmässä* arvioidaan Hessen matriisia differenssiarvioiden avulla. Mikäli Hessen matriisi on harva ja sen rakenne on säännöllinen, voidaan differenssiarvio laskea hyvin tehokkaasti käyttämällä sopivissa pisteissä laskettua gradienttivektoria.

Jos funktio $f(\mathbf{x})$ on kvadraattinen, pätee Taylorin sarjakehitelmän perustella

$$\nabla f(\mathbf{x}+\mathbf{s}) = \nabla f(\mathbf{x}) + H(\mathbf{x})\mathbf{s}. \tag{12.6}$$

Olkoon askel $\mathbf{s} \in \mathbb{R}^n$ pieni, jolloin kahdesti differentioituvalle yleiselle fuktiolle f pätee edellisen yhtälön perusteella

$$H(\mathbf{x})\mathbf{s} \approx \nabla f(\mathbf{x}+\mathbf{s}) - \nabla f(\mathbf{x}). \tag{12.7}$$

Valitsemalla askel $\mathbf{s}$ sopivasti voidaan Hessen matriisia siis arvioida gradienttivektorien avulla.

Olkoon laskettavana Rosenbrockin n-ulotteisen funktion Hessen matriisi, joka koostuu diagonaalilla olevista kooltaan 2×2 lohkoista eli on muotoa

$$H = \begin{pmatrix} \times & \times & & & & \\ \times & \times & & & & \\ & & \times & \times & & \\ & & \times & \times & & \\ & & & & \times & \times \\ & & & & \times & \times \end{pmatrix}. \tag{12.8}$$

Lasketaan vektorit $\mathbf{y}^1$ ja $\mathbf{y}^2 \in \mathbb{R}^n$:

$$\mathbf{y}^i = \frac{1}{h\|\mathbf{z}^i\|}\left(\nabla f(\mathbf{x}+h\mathbf{z}^i) - \nabla f(\mathbf{x})\right), \quad i = 1,2, \tag{12.9}$$

missä $\mathbf{z}^1 = (1,0,1,0,\dots)^T$ ja $\mathbf{z}^2 = (0,1,0,1,\dots)^T$. Merkitään vektorin $\mathbf{y}^1$ alkioita $(y_1^1, y_2^1, \dots, y_n^1)$ ja vastaavasti vektorin $\mathbf{y}^2$ alkioita. Vertaamalla kaavaa 12.9 kaavaan 12.5 havaitaan, että pätee esimerkiksi

$$H_{11} \approx y_1^1, \quad H_{12} = H_{21} \approx y_1^2, \quad H_{22} \approx y_2^2 \quad \text{jne.}$$

Täten kaikki Hessen matriisin alkiot saadaan arvioitua gradienttivektorien $\nabla f(\mathbf{x})$, $\nabla f(\mathbf{x}+h\mathbf{z}^1)$ ja $\nabla f(\mathbf{x}+h\mathbf{z}^2)$ avulla riippumatta tehtävän dimensiosta n.

12.3.3 Jacobin matriisin differenssiarviot

Funktion $\mathbf{f}(\mathbf{x})$ Jacobin matriisille $J(\mathbf{x})$ voidaan laskea differenssiarviot soveltamalla gradienttivektorin differenssikaavoja (12.2). Seuraavassa esimerkkejä differenssiarvioista $J(\mathbf{x}) = (J_{ij})$:

$$\begin{cases} J_{ij} = \dfrac{f_i(\mathbf{x}+h_j\mathbf{e}^j) - f_i(\mathbf{x})}{h_j}, & \text{eteenpäin laskettu differenssi,} \\ J_{ij} = \dfrac{f_i(\mathbf{x}+h_j\mathbf{e}^j) - f_i(\mathbf{x}-h_j\mathbf{e}^j)}{2h_j}, & \text{keskeisdifferenssi.} \end{cases} \tag{12.10}$$

Jacobin matriisin differenssiarvioita voi tarvita esimerkiksi menetelmissä, joissa käytetään rajoitteiden $\mathbf{h}(\mathbf{x}) = \mathbf{0}$ ja $\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$ derivaattoja.

12.4 Ohjelmistoja

Edellä on esitelty Mathematica- ja Maple-ohjelmistojen käyttöä symboliseen ja automaattiseen derivointiin. Muutkin kehittyneet matemaattiset symbolienkäsittelyjärjestelmät tarjoavat vastaavanlaisia mahdollisuuksia [Sof94, KN94].

ADIFOR on uusi Fortran-koodin käsittelyyn perustuva automaattisen derivoinnin ohjelmisto, joka on kehitetty erityisesti suurten tehtävien tarpeisiin [BCC+92].

Teoksessa *Optimization Software Guide* [MW93] on esitelty muutamia automaattiseen derivointiin soveltuvia ohjelmistoja (esimerkiksi JAKEF, joka perustuu Fortran-aliohjelmakirjastojen käyttöön).

Opas *Matemaattiset ohjelmistot* [HHL+03] esittelee joukon CSC:ssä käytettävissä olevia ohjelmistoja. Hyviä vinkkejä Mathematica-ohjelmiston käyttöön löytyy teoksista [Mae90, Wol91]. Lisäksi IMSL- ja NAG-aliohjelmakirjastot ja Netlib-verkkoarkisto sisältävät rutiineita differenssiarvioiden laskemiseen ja derivaattojen tarkistamiseen.

12.5 Lisätietoja

Kokoelmateoksessa *Automatic Differentiation of Algorithms* [GC91] on runsaasti symboliseen ja automaattiseen derivointiin liittyviä artikkeleita. Automaattista derivoimista on käsitelty artikkeleissa [Mon94, GvHM91, Iri91]. Mathematica-ohjelmiston käyttöä on esitelty raporteissa [Sof93a, Sof94, Sof93b]. Automaattiseen derivointiin liittyviä ohjelmistoja on esitelty viitteissä [Abb94, Pel94, Gom90, Cap92]. Lisätietoa automaattisesta derivoinnista löytyy www-osoitteesta `http://www.autodiff.org/`.

Differenssiarvioiden laskemisen perusteet on esitetty teoksessa *Numerical Methods for Unconstrained Optimization and Nonlinear Equations* [DS83] ja diskreettiä Newtonin menetelmää on käsitelty kirjoissa *Practical Optimization* [GMW81] ja *Introduction to Non-linear Optimization* [Sca85].

13 Yksiulotteinen optimointi ja luottamusalue

Tässä luvussa kerrotaan epälineaarisen optimoinnin viivahaku- ja luottamusaluemenetelmien taustasta sekä menetelmien merkityksestä laskennan tehokkuuden kannalta. Lisäksi kerrotaan yhden muuttujan optimointitehtävien ratkaisumenetelmistä.

13.1 Viivahaku- ja luottamusaluealgoritmit

Ratkaistaessa jatkuvasti differentioituvaa, epälineaarista optimointitehtävää

$$\operatorname*{minimi}_{\mathbf{x}} f(\mathbf{x}),\ \text{kun}\ \mathbf{x} \in \mathcal{F} \subset \mathbb{R}^n\,, \tag{13.1}$$

käytetään yleensä iteratiivista ratkaisumenetelmää, jossa valitaan askel $\mathbf{s}^k$ siten, että funktion arvo pienenee uudessa iteraatiopisteessä $\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{s}^k$ (luku 6).

Viivahakualgoritmeissa (line search) löydetään uusi iteraatiopiste kulkemalla hakusuuntaan $\mathbf{p}^k \in \mathbb{R}^n$, kunnes löydetään uusi iteraatiopiste:

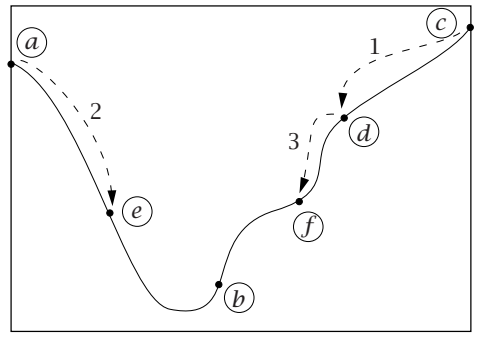
$$\mathbf{x}^{k+1} = \mathbf{x}^k + \lambda_k \mathbf{p}^k, \quad \lambda > 0\,. \tag{13.2}$$

Hiukan toisenlaiseen lähestymistapaan perustuvia *luottamusaluealgoritmeja* (trust region) on käsitelty sivulta 199 alkaen.

Kummankin tyyppiset menetelmät ovat osoittautuneet käyttökelpoisiksi vaikeidenkin optimointitehtävien ratkaisemisessa. Luottamusaluemenetelmiä on käytetty mm. rajoitteettomassa optimoinnissa ja epälineaaristen yhtälöryhmien ratkaisussa. Viivahakumenetelmille puolestaan on löydetty runsaasti käyttökohteita mm. rajoiteellisten optimointitehtävien ratkaisemisessa.

13.2 Yksiulotteinen optimointi

Olkoon minimoitava funktio $f(x)$ jatkuva skalaarifunktio $\mathbb{R} \mapsto \mathbb{R}$. Paikallisen minimin sijainti on tiedossa, kun on löydetty kolme pistettä $a < b < c$ siten, että $f(b) < f(a)$ ja $f(b) < f(c)$. Kuvassa 13.1 on esitetty skalaarifunktion minimoinnin periaate.



Kuva 13.1: *Skalaarifunktion minimin hakeminen.*

Aluksi minimikohtaa rajaavat pisteet (a, b, c). Tämän jälkeen funktion arvo lasketaan pisteessä d, jolla voidaan korvata piste c jne. Kullakin askeleella valitaan jatkoon keskipiste, jossa funktion arvo on pienempi kuin sitä reunustavissa pisteissä. Iteraation 3 jälkeen minimi on pisteiden (e, b, f) välissä.

CSC:n oppaassa *Numeeriset menetelmät* [HKR93] on kerrottu mm. *kultaisen leikkauksen menetelmän*, *Brentin menetelmän* ja *Newtonin menetelmän* käytöstä skalaarifunktion minimointiin.

Skalaarifunktion minimoinnissa on Newtonin menetelmän iteraatioaskel

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}. \tag{13.3}$$

Newtonin menetelmä suppenee neliöllisesti optimipisteen läheisyydessä, mutta ilman varmistuksia se on epäluotettava törmättäessä toisen derivaatan nollakohtaan.

Kultaisen leikkauksen menetelmä on yksinkertainen vaikkakin hitaampi kuin Newtonin menetelmä, eikä se vaadi kohdefunktiolta differentioituvuutta. Erityyppiset interpolaatiomenetelmät (esimerkiksi Brentin menetelmä) ovat luotettavampia kuin Newtonin menetelmä ja lähes yhtä tehokkaita, varsinkin jos kohdefunktion derivaatat ovat kohtuullisen työläitä laskea.

13.3 Viivahakualgoritmit

Epälineaaristen optimointitehtävien *viivahakuun* perustuvat ratkaisualgoritmit ovat perusrakenteeltaan seuraavia:

Algoritmi 13.3.1 (Viivahakumenetelmät)

1: Hae käypä alkuarvaus $\mathbf{x}^1 \in \mathcal{F}$. Aseta $k = 1$.

2: Jos olet kyllin lähellä optimia, lopeta iterointi.

3: Valitse käypä hakusuunta $\mathbf{p}^k \in \mathbb{R}^n$, jolle pätee $f(\mathbf{x}^k + \lambda\mathbf{p}^k) < f(\mathbf{x}^k)$
siten, että $\mathbf{x}^k + \lambda\mathbf{p}^k \in \mathcal{F}$, $\lambda > 0$.

4: Viivahaku: etsi askelpituus $\lambda_k = \arg\text{minimi}_{\lambda>0} f(\mathbf{x}^k + \lambda\mathbf{p}^k)$
siten, että $\mathbf{x}^k + \lambda_k\mathbf{p}^k \in \mathcal{F}$.

5: Aseta $\mathbf{x}^{k+1} = \mathbf{x}^k + \lambda_k\mathbf{p}^k$ ja $k = k + 1$. Mene vaiheeseen 2.

Viivahakualgoritmeja on kehitetty hyvin monenlaisiin tarkoituksiin, ja esimerkiksi funktion arvon laskemiseen kuluvan työn määrä vaikuttaa tehokkaimman menetelmän valintaan. Lisäksi optimointialgoritmit asettavat vaatimuksia viivahaun tarkkuudelle.

Takautuvat viivahakualgoritmit ovat yleisimmin käytettyjä viivahakumenetelmiä ja ne ovat osoittautuneet luotettaviksi ja joustaviksi. Yksinkertaisimmillaan kyseessä on *Armijo-tyyppinen viivahaku*, jossa valitaan askelpituudeksi 1, mikäli kohdefunktion arvo pienenee riittävästi, ja muuten pienennetään rekursiivisesti askelpituutta vakiokertoimella $\tau \in (0, 1)$.

13.3.1 Viivahakumenetelmien toteutus

Viivahakua käyttävissä optimointialgoritmeissa valitaan aluksi hakusuunta ja käytetään tämän jälkeen jotain sopivaa menetelmää minimin löytämiseksi kuljettaessa kyseiseen suuntaan. Menetelmästä riippuen voidaan käyttää *epätarkkaa hakua* tai *tarkkaa hakua*. Esimerkiksi liittogradienttimenetelmissä on viivahaun tarkkuudella ainakin teoriassa suuri merkitys suppenemisen kannalta, kun taas sekanttimenetelmiä käytettäessä viivahaun tarkkuus voi olla usein melko pieni.

Olkoon vektori $\mathbf{p} \in \mathbb{R}^n$ suunta, johon funktion arvo vähenee, joten tällöin pätee $\langle \nabla f(\mathbf{x}), \mathbf{p} \rangle < 0$. Viivahakualgoritmit pyrkivät löytämään äärellisellä määrällä askeleita pisteen, joka toteuttaa sopivat ehdot koskien kohdefunktion arvoja ja suuntaderivaattaa hakusuoralla. Viivahakualgoritmeissa asetetaan hyväksyttävälle askelpituudelle λ esimerkiksi *riittävän pienentymisen ehto*:

$$f(\mathbf{x} + \lambda\mathbf{p}) \leq f(\mathbf{x}) + \mu\lambda\langle \nabla f(\mathbf{x}), \mathbf{p} \rangle, \tag{13.4}$$

sekä esimerkiksi *kaarevuusehto*:

$$|\langle \nabla f(\mathbf{x} + \lambda\mathbf{p}), \mathbf{p} \rangle| \leq \eta |\langle \nabla f(\mathbf{x}), \mathbf{p} \rangle|, \tag{13.5}$$

missä $0 < \mu < \eta < 1$. Kannattaa huomata, että kun suunta $\mathbf{p}$ on funktion vähenemissuunta pisteessä $\mathbf{x}$, pätee esimerkiksi $\langle \nabla f(\mathbf{x}), \mathbf{p} \rangle < 0$. Ehto (13.4) takaa funktion arvon riittävän pienentymisen ja ehto (13.5) takaa, ettei olla liian kaukana funktion minimistä suoralla $\mathbf{x} + \lambda\mathbf{p}$.

Käytettäessä esimerkiksi ehdot (13.4) ja (13.5) toteuttavaa viivahakualgoritmia voidaan monille ratkaisumenetelmille todistaa suppenemistuloksia.

Viivahaun tarkkuuden määrittelemiseen voidaan käyttää myös yhtälöä

$$|\langle \nabla f(\mathbf{x}+\lambda\mathbf{p}), \mathbf{p}\rangle| \leq -\eta\langle \nabla f(\mathbf{x}), \mathbf{p}\rangle\,. \tag{13.6}$$

Kun kerroin η on pieni, on kyseessä *tarkka viivahaku* (accurate line search). Kun $\eta = 0$, on kyseessä *eksakti viivahaku*.

Kirjallisuudessa viitataan viivahakualgoritmien yhteydessä usein myös *Wolfen ehtoihin*, jotka ovat

$$f(\mathbf{x}+\lambda\mathbf{p}) \;\leq\; f(\mathbf{x}) + \sigma_1\lambda\langle \nabla f(\mathbf{x}), \mathbf{p}\rangle, \tag{13.7}$$

$$\langle \nabla f(\mathbf{x}+\lambda\mathbf{p}), \mathbf{p}\rangle \;\geq\; \sigma_2\langle \nabla f(\mathbf{x}), \mathbf{p}\rangle, \tag{13.8}$$

missä $0 < \sigma_1 < \sigma_2 < 1$. Ensimmäinen epäyhtälö takaa riittävän vähennyksen funktion arvolle ja toinen riittävän vähennyksen suuntaderivaattaan.

13.3.2 Takautuvat viivahakualgoritmit

Takautuvalla viivahakualgoritmilla (backtracking line search) tarkoitetaan menetelmää, jossa hyväksytään askelpituus $\lambda_1 = 1$, jos ehto (13.7) toteutuu, ja muussa tapauksessa kokeillaan yhä lyhyempää askelpituutta. *Puhtaasti takautuvissa* viivahakualgoritmeissa (esimerkiksi *Armijo-viivahaku*) valitaan uudeksi koeaskeleeksi λ_{k+1} vakio-osa τ edellisestä askeleesta λ_k, esimerkiksi $\lambda_{k+1} = 0.1\,\lambda_k$. Mikäli askelpituus $\lambda_1 = 1$ hyväksytään usein, on Armijo-tyyppinen viivahakualgoritmi varsin tehokas.

Seuraavassa esitetään monipuolinen neliölliseen ja kuutiolliseen interpolaatioon perustuva viivahakualgoritmi:

Algoritmi 13.3.2 (Takautuva viivahakumenetelmä)

1: Valitse suunta $\mathbf{p}$, siten että $\langle \nabla f(\mathbf{x}), \mathbf{p}\rangle < 0$. Valitse parametri $\sigma_1 < 0.5$, esimerkiksi $\sigma_1 = 10^{-4}$.

2: Aseta $\lambda = 1$.

3: Laske piste $\mathbf{x}^+ = \mathbf{x} + \lambda\mathbf{p}$. Jos yhtälön (13.7) ehto toteutuu, palauta $\mathbf{x}^+$ ja lopeta viivahaku.

4: Jos askelpituus λ on liian pieni, palauta $\mathbf{x}$ ja lopeta iterointi virheeseen.

5: Pienennä parametrin λ arvoa kertoimella, joka on välillä $[0.1, 0.5]$.

- **(1)** Ensimmäisellä hakukerralla valitse uusi askelpituus λ_+ siten, että se minimoi kvadraattisen funktion, joka sovitetaan dataan $f(\mathbf{x})$, $\nabla f(\mathbf{x})$ ja $f(\mathbf{x}^+)$. Uuden askelpituuden λ_+ tulee olla ≥ 0.1. Aseta $f_o = f(\mathbf{x}^+)$ ja $\lambda = \lambda_+$.
- **(2)** Myöhemmillä hakukerroilla valitse uusi askelpituus λ_+ siten, että se minimoi kuutiollisen funktion, joka sovitetaan dataan $f(\mathbf{x})$, $\nabla f(\mathbf{x})$, $f(\mathbf{x}^+)$ ja f_o. Uuden askelpituuden λ_+ tulee olla välillä $[0.1\lambda, 0.5\lambda]$. Aseta $\lambda = \lambda_+$.

6: Palaa vaiheeseen 3.

Algoritmi 13.3.2 toteuttaa Wolfen ensimmäisen ehdon (13.7). Teoksesta *Numerical Methods for Unconstrained Optimization and Nonlinear Equations* [DS83] löytyy funktion $f(\mathbf{x})$ suuntaderivaattaa käyttävä algoritmi, joka toteuttaa myös ehdon (13.8).

Esimerkki 13.3.1 Minimoidaan kaksiulotteista Rosenbrockin funktiota

$$f(\mathbf{x}) = 100\left(x_2 - x_1^2\right)^2 + (1 - x_1)^2$$

pisteestä $\mathbf{x}^k = (-1.2, 1.0)^T$ suuntaan $\mathbf{p}^k = -\nabla f(\mathbf{x}^k) = (215.6, 88.0)^T$:

$$\lambda_k = \arg \underset{\lambda>0}{\text{minimi}}\ f(\mathbf{x}^k + \lambda \mathbf{p}^k). \tag{13.9}$$

Tehtävässä on siis kyse tyypillisestä n-ulotteisen optimointitehtävän viivahausta. Listauksessa 13.1 on esitetty Mathematicalla toteutettu viivahakurutiini `LinSearch`. Algoritmi löytyy mm. teoksesta [DS83]. Lisäksi käytetään luvussa 12 esitettyä rutiinia `Grad` funktion gradientin laskemiseen.

```
In[1]:= <<Derivaatat.m; <<LinSearch.m

In[2]:= f[x1_,x2_] = 100 (x2 - x1^2)^2 + (1 - x1)^2;

In[3]:= gf[x1_,x2_] = Simplify[Grad[f[x1,x2],{x1,x2}]];

In[4]:= x0 = {-1.2, 1};

In[5]:= fx0 = Apply[f, x0]

Out[5]= 24.2

In[6]:= gx0 = Apply[gf, x0]; p0 = - gx0

Out[6]= {215.6, 88.}

In[7]:= maxstep = 100; steptol = 10^(-8);

In[8]:= LinSearch[f, x0, p0, maxstep, steptol, fx0, gx0]

Out[8]= {{-0.964472, 1.09613}, 6.61237, 0}
```

Viivahaun optimipisteeksi saatiin siis $(-0.96, 1.10)^T$ ja funktion arvoksi kyseisessä pisteessä saatiin 6.6.

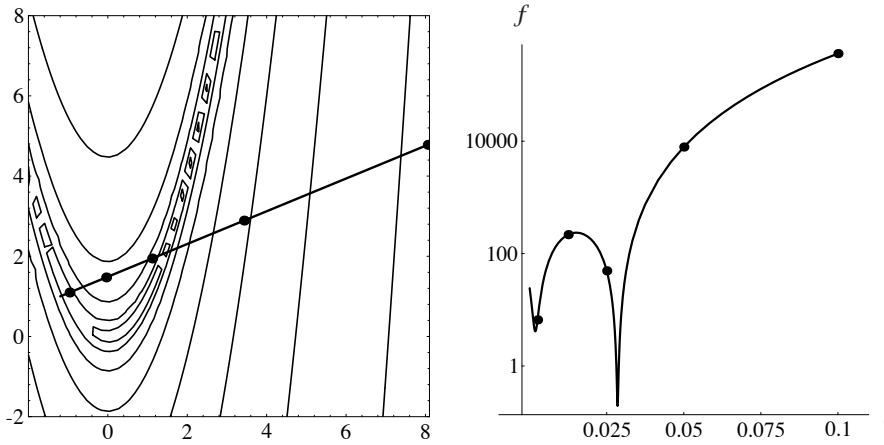
Kuvassa 13.2 on esitetty takautuvan viivahakualgoritmin toiminta. Listauksen 13.1 viivahakualgoritmi kokeilee tässä tapauksessa askelpituuksia

$$\lambda = \{1,\ 0.1,\ 0.05,\ 0.025,\ 0.0125,\ 0.00254392\},$$

joista viimeinen hyväksytään. Kuvaan 13.2 ei ole otettu ensimmäistä eli kaikkein pisintä askelta. Käytetty viivahakualgoritmi ei huomaa kohdan $\lambda = 0.025$ vieressä olevaa minimikohtaa.

Listaus 13.1: *Neliöllistä ja kuutiollista interpolaatiota käyttävä Mathematicalla ohjelmoitu viivahakurutiini* `LinSearch`.

```
LinSearch[f_,x0_,p0_,maxstep_,steptol_,fx0_,gx0_] :=
Block[{x = x0, p = p0, alphatol = 10^(-4), RCODE = -1,
       steplen, linegrad, fx, lambdamin, lambda, lmb2,
       lmbtmp, fx2, temp1, temp2, lmbdiff, discr, a, b},
  steplen = Sqrt[Apply[Plus, p^2]];
  If[steplen > maxstep,
    p = p*maxstep/steplen; steplen = maxstep];
  linegrad = Apply[Plus, gx0*p];
  lambdamin = steptol/Max[Abs[p]/Max[Abs[x],1]];
  lambda = 1.0;
  While[RCODE < 0,
    x = x0 + lambda*p; fx = Apply[f, x];
    If[fx <= (fx0 + alphatol * lambda * linegrad),
      (* tarpeeksi hyvä ratkaisu löytyi *)
      RCODE = 0,
      If[lambda < lambdamin,
        (* ei löytynyt x0:sta poikkeavaa ratkaisua *)
        x = x0; RCODE = 1,
        If[lambda == 1.0,
          (* laske uusi lambda ensimmäisessä vaiheessa *)
          lmbtmp = -linegrad/
                      (2*(fx - fx0 - linegrad)),
          (* laske uusi lambda myöh. käyttöä varten *)
          temp1 = fx - fx0 - lambda*linegrad;
          temp2 = fx2 - fx0 - lmb2*linegrad;
          lmbdiff = 1/(lambda-lmb2);
          a = lmbdiff*(temp1/lambda^2-temp2/lmb2^2);
          b = lmbdiff*(lambda*temp2/lmb2^2-
              lmb2*temp1/lambda^2);
          discr = b^2 - 3.0*a*linegrad;
          If[a == 0.0,
            lmbtmp = -linegrad/(2.0*b),
            lmbtmp = (Sqrt[discr] - b)/(3.0*a)];
          If[lmbtmp > 0.5*lambda, lmbtmp = 0.5*lambda]
        ]
      ]
    ];
    If[RCODE < 0, fx2 = fx; lmb2 = lambda;
      lambda = Max[lmbtmp, 0.1*lambda]]
  ];
  {x, fx, RCODE}  (* palauta arvot *)
]
```



Kuva 13.2: *Viivahakualgoritmin toiminta minimoitaessa kaksiulotteista Rosenbrockin funktiota. Vasemmalla on funktion esitys tasa-arvokäyrien avulla ja oikealla funktion arvot viivahakuun käytetyllä puolisuoralla.*

13.4 Luottamusalueen käyttö

Viivahakualgoritmeissa valitaan hakusuunta esimerkiksi Newtonin menetelmällä, ja tämän jälkeen haetaan yksidimensioisella viivahaulla funktion (likimääräinen) minimikohta kyseiseen suuntaan. *Luottamusaluealgoritmeissa* (trust-region methods) pyritään käyttämään tehokkaammin hyväksi kohdefunktion paikallista kvadraattista mallia uuden hakupisteen löytämisessä.

Luottamusaluealgoritmeissa askeleeksi $\mathbf{s}^k \in \mathbb{R}^n$ valitaan funktion $f(\mathbf{x})$ kvadraattisen mallin

$$G(\mathbf{x}^k + \mathbf{s}^k) = f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{s}^k \rangle + \frac{1}{2}\langle \mathbf{s}^k, H(\mathbf{x}^k)\mathbf{s}^k \rangle \tag{13.10}$$

minimoiva vektori $\mathbf{s}^k$, joka ratkaisee likimäärin tehtävän

$$\underset{\mathbf{s}^k}{\text{minimi}}\ G(\mathbf{x}^k + \mathbf{s}^k),\ \text{kun}\ \|D_k\mathbf{s}^k\| \leq \Delta, \tag{13.11}$$

missä parametri $\Delta > 0$ on luottamusalueen säde. Diagonaalimatriisi D_k toimii skaalaustekijänä. Matriisi H_k on kohdefunktion Hessen matriisi tai sen arvio.

Seuraavassa on esitetty luottamusaluealgoritmin yleisrakenne:

Algoritmi 13.4.1 (Luottamusaluealgoritmi)

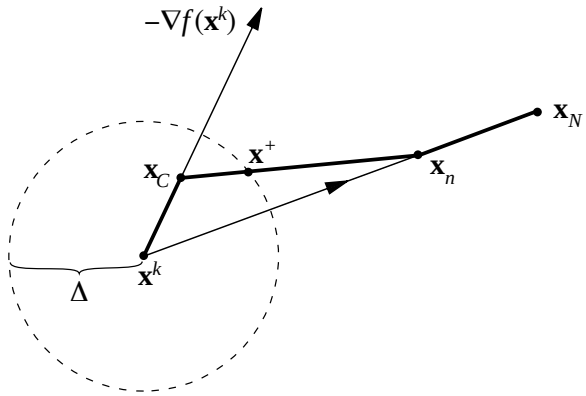
valitse alkuarvaus $\mathbf{x}^1$
laske $f(\mathbf{x}^1)$, $\nabla f(\mathbf{x}^1)$ ja $H_1 = \nabla\nabla^T f(\mathbf{x}^1)$
for $k = 1, 2, \ldots$
    ratkaise $\mathbf{s}^k$ likimäärin yhtälöstä (13.11)

laske $r_k = \dfrac{G(\mathbf{x}^k) - G(\mathbf{x}^k + \mathbf{s}^k)}{f(\mathbf{x}^k) - f(\mathbf{x}^k + \mathbf{s}^k)}$
laske uusi arvio luottamusalueen säteelle Δ
if askel $\mathbf{s}^k$ on hyväksyttävissä: $r_k \geq \eta > 0$
$\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{s}^k$
else
$\mathbf{x}^{k+1} = \mathbf{x}^k$
end
if $\|\mathbf{x}^{k+1} - \mathbf{x}^k\| > \epsilon$
laske $f(\mathbf{x}^{k+1})$, $\nabla f(\mathbf{x}^{k+1})$ ja $H_{k+1} = \nabla\nabla^T f(\mathbf{x}^{k+1})$
end
until $\|\mathbf{x}^{k+1} - \mathbf{x}^k\| \leq \epsilon$

Esimerkki 13.4.1 Netlib-verkkoarkistosta löytyvän Minpack-ohjelmiston sisältämää *dogleg*-luottamusaluealgoritmia on käsitelty teoksessa [DS83]. Kuvassa 13.3 on havainnollistettu menetelmän perusideaa.

Piste $\mathbf{x}^k$ on nykyinen iteraatiopiste ja skalaari $\Delta > 0$ on luottamusalueen koko. Piste $\mathbf{x}_C$ on *Cauchy-piste* eli jyrkimmältä suuntavektorilta haettu kvadraattisen mallin minimipiste. Piste $\mathbf{x}_N$ on puolestaan Newton-askeleen tuottama uusi iteraatiopiste. Piste $\mathbf{x}_n$ on pisteiden $\mathbf{x}^k$ ja $\mathbf{x}_N$ välissä.

Tässä luottamusaluealgoritmissa saadaan uusi iteraatiopiste murtoviivalta, joka yhdistää pisteet $\mathbf{x}^k$, $\mathbf{x}_C$, $\mathbf{x}_n$ ja $\mathbf{x}_N$. Jos luottamusalue on pieni, kuljetaan negatiivisen gradientin suuntaan, ja jos luottamusalue on suuri, kuljetaan Newton-askeleen suuntaan.



Kuva 13.3: *Minpack-ohjelmiston* dogleg*-käyrään perustuva luottamusaluemenetelmä.*

13.5 Lisätietoja

Viivahaku- ja luottamusaluealgoritmeista löytyy lisätietoa teoksista [DS83, Sca85, KMN89, GMW81]. Lähdekoodeja löytyy mm. TOMS-algoritmeista. *Numerical Recipes* -teokset [PTVF92a, PTVF92b] sisältävät yksinkertaisen viivahakualgoritmin kuvauksen ja lähdekoodin.

Wolfen ehtoja on esitelty viitteissä [BN89, DS83, Noc92, OR70]. Artikkelissa [Noc92] on esitetty katsaus uusimpiin rajoitteetonta optimointia koskeviin tuloksiin. Viivahaun kaarevuusehtoja on käsitelty teoksissa [GMW81, MW93]. Sisäpistemenetelmien viivahakuun liittyviä erikoiskysymyksiä on tarkasteltu artikkelissa [MW].

Luottamusaluemenetelmiä on esitelty viitteissä [GMW81, DS83]. Teoksessa *Practical Methods of Optimization* [Fle87] on esitelty sekä viivahaku- että luottamusaluemenetelmiä.

Liitteet

A Optimoinnin peruskirjallisuutta

Optimoinnista on saatavilla runsaasti opetukseen ja itseopiskeluun sopivaa kirjallisuutta. Seuraavassa esitellään teoksia, joista on hyötyä niin optimoinnin teorian opiskelussa kuin käytännön tehtävien ratkaisemisessakin.

Seuraavassa esiteltyjen kirjojen sisältö painottuu jatkuvasti derivoituvien ja äärellisulotteisten optimointitehtävien ratkaisemiseen. Globaalin optimoinnin problematiikkaa ei useimmissa tässä esitellyissä teoksissa käsitellä, vaikka mukana on kaksi yleisiä heuristiikkoja esittelevää teosta.

A.1 Englanninkielisiä perusteoksia

Scientific Computing — An Introductory Survey [Hea97]

Michael Heathin teos on monipuolinen ja pätevä johdatus numeeriseen laskentaan. Vaikka teos esittelee suuren joukon aihepiirejä lineaarialgebrasta satunnaislukuihin, päästään eri osa-alueilla varsin pitkälle.

Teos sisältää runsaasti esimerkkejä ja eritasoisia harjoitustehtäviä, joten se soveltuu hyvin opiskeluun ja opetukseen. Teoksessa on 200 harjoitustehtävää sekä lisäksi 180 tietokoneella ratkaistavaksi tarkoitettua tehtävää.

Teos esittelee monipuolisesti tärkeimmät optimointitehtävien ratkaisualgoritmit. Kirjassa ei juuri käsitellä optimoinnin teoreettisia perusteita vaan keskitytään ratkaisumenetelmien esittelyyn ja käyttöesimerkkeihin. Esiteltävät menetelmät ovat luotettavia ja tehokkaita. Lisäksi teos tuo esille menetelmien toteutuksia eri ohjelmistoissa ja algoritmikokoelmissa.

Practical Optimization Methods With Mathematica Applications [Bha00]

Teos on käytännönläheinen johdatus optimointitehtävien muodostamiseen ja ratkaisuun. Teoksessa on paljon esimerkkejä ja graafisia havainnollistuksia.

Kirjan sisältämien runsaiden havainnollistusten toteuttamiseen käytetään Mathematica-ohjelmistoa. Kirjan mukana on CD-ROM, joka sisältää Mathematica-koodeja ja esimerkkejä optimointitehtävien ratkaisemisesta. CD:llä on myös toteutuksia kirjassa esitellyistä algoritmeista. Nämä sopivat lähin-

nä opetuskäyttöön. Tosin Mathematican tehokas käyttö opiskelussa edellyttää jonkintasoista aiempaa tutustumista ohjelmistoon.

Kirjassa käsitellään jonkin verran (tosin hiukan ylimalkaisesti) myös optimoinnin teoreettisia perusteita sekä erilaisten optimointialgoritmien käyttäytymistä.

Teoksen esitystapa ei ole aina niin täsmällistä ja yleispätevää kuin toivoisi. Käytännönläheisyyteen ja konkreettisuuteen pyrkiminen on paikoin viety liiankin pitkälle. Eräät huomautukset ja varoitukset ovat epätäsmällisiä liittyen usein johonkin konkreettiseen esimerkkiin. Hiukan syvällisemmällä ja eksaktimmalla esitystavalla teoksesta saisi luotettavamman oppikirjan.

Teoksessa käytetään varsin suuri sivumäärä lineaaristen optimointitehtävien ratkaisemiseen. Esitellyiksi tulevat niin simplex-menetelmä kuin sisäpistemenetelmätkin. Teoksessa käsitellään myös rajoitteellisten tehtävien kehittyneempiä ratkaisumenetelmiä.

Teoksessa on runsaasti harjoitustehtäviä, jotka usein liittyvät jonkin konkreettisen optimointitehtävän muodostamiseen ja ratkaisemiseen. Paino on hyvin selvästi käytännössä: eri tilanteissa vastaan tulevien optimointitehtävien ratkaisemisessa.

Nonlinear Programming: Theory and Algorithms [BSS93]

Teos on perusteellinen johdatus optimoinnin matemaattisiin perusteisiin ja tärkeimpiin optimointialgoritmeihin. Teoksesta on muodostunut klassikko optimoinnin perusoppikirjana. Puolet teoksesta on varattu optimointiteorian esittämiseen ja toinen puoli perusalgoritmeille.

Teos sisältää runsaasti harjoitustehtäviä ja soveltuu hyvin optimointiteoriaa korostavan yliopistokurssin materiaaliksi. Optimointitehtävien muodostaminen käytännön tapauksissa jää vähemmälle painotukselle.

Vaikka teos on alunperin julkaistu jo vuonna 1979, puolustaa sen toinen painos yhä paikkaansa optimointiopin perusesityksenä ja käsikirjana. Teoksen selkeä esitystapa, lukuisat havainnollistukset ja monet harjoitustehtävät tekevät siitä yhä hyvän vaihtoehdon. Uusimpia optimoinnin teorian ja käytännön oivalluksia ei teoksesta löydy, joten sitä on hyvä täydentää muilla lähteillä.

Linear and Nonlinear Programming [NS96]

Teoksen painopiste on optimointialgoritmien esittelyssä ja analyysissä. Optimoinnin peruskäsitteet ja johdatus teoriaan esitetään tiiviisti. Myös tarvittavat esitiedot löytyvät tiivistelminä teoksesta: lineaarialgebra, tietokonearitmetiikan perusteet ja algoritmien skaalautuminen.

Yllättävänkin suuri osa teoksesta käsittelee lineaaristen optimointitehtävien ratkaisumenetelmiä. Teos esittelee sekä klassisen simplex-menetelmän että nykyaikaisia sisäpistemenetelmiä. Epälineaarisen optimoinnin perusalgoritmit esitellään suppeammin. Mukana on silti myös edistyneempää materiaalia kuten katkaistu Newtonin menetelmä ja muita muistinkäytöltään tehokkaita menetelmiä.

Teos sisältää esimerkkejä algoritmien käytöstä tehtävien ratkaisemiseen.

Myös harjoitustehtäviä on annettu. Optimointitehtävien muodostamiseen liittyvää problematiikkaa käsitellään varsin suppeasti.

Teos on pätevä ja melko ajantasainen johdatus optimointiin, vaikkakaan sen painotukset eivät sovellu kaikkiin tarpeisiin. Teoreettiset perusvalmiudet on ehkä opiskeltava muista lähteistä, samoin optimointitehtävien muodostamiseen liittyvät käytännön taidot.

How to Solve It: Modern Heuristics [MF99]

Michalewicz ja Fogel ovat kirjoittaneet katsaustyyppisen esityksen ongelmanratkaisuun. Teos on samalla johdatus uusien, heurististen menetelmien käyttöön mm. optimointitehtävien ratkaisemisessa.

Monet teoksen esimerkeistä ovat ymmärrettävissä vähäisillä taustatiedoilla ja haastavat siten kenet tahansa pohtimaan ongelman ratkaisua.

Koska Michalewicz ja Fogel ovat tutkineet geneettisten algoritmien käyttöä ongelmanratkaisuun, on teoksessa vahva painotus evoluutioalgoritmien soveltamisella erityyppisiin tehtäviin, etupäässä erilaisiin optimointiongelmiin. Toisaalta teos sisältää myös ajan tasalla olevan johdatuksen muihin paljon käytettyihin heuristiikkoihin: jäähdytysmenetelmään, tabu-hakuun, neuroverkkoihin ja sumeaan logiikkaan. Ajantasaisuus on toisaalta myös teoksen ongelma: paikka paikoin tekstissä on luettelomaisuutta, enkä yhtään ihmettelisi, jos monet esitellyistä heuristiikoista menettäisivät suosiotaan lähivuosien aikana.

Monet esimerkeistä ovat irrallisia erikoistapauksia, eivätkä välttämättä auta reaalimaailman ongelmien ratkaisemisessa. Teoksesta ei ole kovin paljon apua reaalimaailman ongelman muuntamiseen ratkaistavissa olevaksi matemaattiseksi probleemaksi. Teoksen ansiot ovat toisaalla: teos on monipuolinen ja kiinnostava katsaus uusiin tietokonepohjaisiin hakualgoritmeihin.

An Introduction to Genetic Algorithms [Mic98]

Mitchellin kirja on mielestäni paras saatavilla oleva johdatus geneettisten algoritmien (GA) käyttöön. Teos on varsin tiivis, eikä ylimääräistä painolastia ole mukana.

Kirjassa käsitellään GA-menetelmien taustaa ja merkitystä. Lisäksi teos esittelee menetelmien sovelluksia ja käyttöä mallintamisessa. Mitchell esittelee GA-menetelmien käyttöä optimoinnin lisäksi säätötehtävissä ja koneoppimisessa. Kirjassa on myös hyvä katsaus GA-menetelmien teoreettisiin perusteisiin. Teos antaa myös hyviä ideoita menetelmien jatkokehittelyn pohjaksi.

A.2 Suomenkielisiä teoksia

CSC on julkaissut tämän oppaan lisäksi useita käsikirjoja, joissa käsitellään optimointia [HHL+02, Haa01, HHL+03]. Lisäksi muun muassa Jyväskylän yliopistossa on julkaistu asiaa käsittelevä luentomoniste [Mie98].

B Ohjelmistoja

Seuraavassa esitellään CSC:ssä käytettävissä olevia optimointitehtävien käsittelyyn soveltuvia ohjelmistoja. Tarkempia kuvauksia löytyy liitteen lopussa olevan kirjallisuusluettelon teoksista.

B.1 CSC:n ohjelmistotuki ja neuvontapalvelut

Tässä luvussa kerrotaan eräistä optimointitehtävien ratkaisemiseen soveltuvista ohjelmistoista. Ohjelmistojen käsikirjoihin voi tutustua CSC:ssä ja eräitä käsikirjoja voi tarvittaessa saada myös lainaksi. Kaikkien ohjelmistojen käyttöä ei voida tukea yhtä hyvin ja siksi joidenkin ohjelmistojen asiakastuki rajoittuu lähinnä niiden asennukseen CSC:n koneille (taulukko B.2).

CSC järjestää koulutustilaisuuksia eräiden ohjelmistojen käytöstä. Lisäksi CSC järjestää eri sovellusalojen koulutusta, esimerkiksi numeeristen menetelmien, visualisoinnin tai ohjelmankehityksen alueilta. CSC:n toimintaa on esitelty liitteessä E.

B.2 Optimointitehtävien ratkaisuohjelmistot

Taulukossa B.1 on lueteltu erityyppisistä optimointitehtävistä käytettyjä lyhenteitä.

Taulukossa B.2 on listattu joidenkin optimointiohjelmistojen tärkeimmät ominaisuudet. 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 käsin [HHL+03]. Tässä oppaassa esitellään lähinnä ohjelmistoja, jotka on saatavissa Unix-ympäristöön. Macintosh- ja PC-tietokoneilla voi ohjelmatarjonta olla osin erilaista.

Optimointikoodeja löytyy myös Netlibistä (sivu 214). Optimointitehtävien ratkaisumoduuleita löytyy lisäksi eri alojen sovellusohjelmista, mutta näitä ei seuraavassa käsitellä.

Taulukko B.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
NLEQ	Epälineaariset yhtälöt ja yhtälöryhmät
DNLP	Ei-differentioituva tai epäjatkuva (discontinuous) epälineaarinen optimointi
LSQ	Lineaariset pienimmän neliösumman tehtävät
NLSQ	Epälineaariset pienimmän neliösumman tehtävät

Taulukko B.2: *CSC:n tarjoamia optimointiin soveltuvia ohjelmistoja. Ajan tasalla olevaa tietoa ohjelmistoista saa liitteen E yhteystietojen avulla.*

Ohjelmisto	*Tehtävät*	*Käyttöliittymä*
Lamps	LP, MIP	Interaktiivinen
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
Minpack	NLSQ, NLEQ	Aliohjelmakirjasto
Smin1	NLP, DNLP	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

Mallinnuskieli GAMS on kehitetty erityisesti taloustieteen optimointitehtävien ratkaisemiseen [BKM88]. GAMSin mallikirjastosta löytyy esimerkkejä erityyppisten tehtävien ratkaisemisesta.

Aliohjelmakirjastot IMSL ja NAG sisältävät testattuja ja tehokkaita Fortran-rutiineita, joiden käyttö on melko vaivatonta käsikirjojen ja esimerkkiohjelmien avulla [HHL+03].

Monilla ratkaisuohjelmistoilla on rajoituksia ongelman koon suhteen, ja tehokkuus voi kärsiä dramaattisesti tehtävän dimension kasvaessa. Jos tehtävässä on epälineaarinen kohdefunktio ja epälineaariset rajoitusehdot, tehtävän ratkaiseminen tehokkaasti voi vaatia paljon tutkimustyötä. Oikean menetelmän valintaan kannattaa siis kiinnittää huomiota. Epäjatkuvuuksista selviävät lähinnä vain tilastollisiin menetelmiin perustuvat ohjelmistot sekä ”raakaa voimaa” käyttävät menetelmät, jotka syövät runsaasti tietokoneen laskenta-aikaa.

Uusia jäähdytysmenetelmiin, geneettisiin algoritmeihin tai neuroverkkojen (neural nets) simulointiin perustuvia koodeja on jonkin verran jo saatavilla. Joissakin tehtävissä näillä menetelmillä on saavutettu hyviä tuloksia.

B.3 Yleiskäyttöiset ohjelmistot

Seuraavassa esitellään yleiskäyttöisiä ohjelmistoja kuten matriisikieli Matlab, symbolienkäsittelyjärjestelmät Mathematica ja Maple sekä tilastolliset ohjelmistot SAS ja Splus. Lisäksi kerrotaan mallinnuskieli GAMSista ja NAG- ja IMSL-aliohjelmakirjastoista.

B.3.1 Matlab

Matlab (matrix laboratory) on nimensä mukaisesti matriisien numeeriseen käsittelyyn kehitetty ympäristö [HHL+03, Kiv91]. Matlabin rutiinikokoelma Optimization Toolbox sisältää joukon optimointirutiineita, joita voi käyttää mm. eri menetelmien testaamiseen ja erilaisiin kokeiluihin [Gra93]. Joitakin Matlabin Optimization Toolboxin rutiineista on lueteltu taulukossa B.3. Lisätietoja saa Matlabin sisältä komennolla `help`. Matlab-istunnon voi lopettaa komennolla `quit`.

B.3.2 Mathematica ja Maple

Symbolienkäsittelyjärjestelmät Mathematica ja Maple ovat pienten mallien käsittelyssä ja erilaisissa kokeiluissa hyvin käyttökelpoisia, mutta niitä ei voi sellaisinaan suositella isojen tehtävien toistuvaan ratkaisemiseen. Mathematicassa ja Maplessa on rutiineita pienten LP-tehtävien ja rajoitteettomien epälineaaristen optimointitehtävien ratkaisemiseen numeerisesti.

Taulukko B.3: *Eräitä Matlabin työkalupakkiin* Optimization Toolbox *kuuluvia rutiineita. Lisätietoja saa komennolla* `help` *rutiinin_nimi. Käypä alue* $\mathcal{F}_1$ *toteuttaa rajoitteet* $\{A\mathbf{x} \le \mathbf{b},\ A_{eq}\mathbf{x} = \mathbf{b}_{eq},\ \mathbf{l} \le \mathbf{x} \le \mathbf{u} \mid \mathbf{x} \in \mathbb{R}^n\}$. *Käypä alue* $\mathcal{F}_2$ *toteuttaa edellisten rajoitteiden lisäksi rajoitteet* $\{\mathbf{g}(\mathbf{x}) \le \mathbf{0},\ \mathbf{h}(\mathbf{x}) = \mathbf{0}\}$. *Rutiinilla* `optimset` *voi määritellä optimointirutiinien asetuksia.*

Tehtävä	*Matem. esitys*	*Funktiokutsu*	*Menetelmä*
Skalaari-funktio	$\min f(x)$, $x \in [a,b] \subset \mathbb{R}$	`fminbnd`	Interpolaatio ja kultainen leikkaus
Lineaarinen optimointi	$\min \mathbf{c}^T\mathbf{x}$, $\mathbf{x} \in \mathcal{F}_1$	`linprog`	Simplex-algoritmi tai sisäpistemenet.
Kvadraatt. optimointi	$\min \frac{1}{2}\mathbf{x}^T H\mathbf{x} + \mathbf{c}^T\mathbf{x}$, $\mathbf{x} \in \mathcal{F}_1$	`quadprog`	Projektiomenet. (active set)
Rajoitteeton optimointi	$\min f(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^n$	`fminunc`	BFGS, DFP tai jyrkin suunta
Rajoitteeton optimointi	$\min f(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^n$	`fminsearch`	Polytooppihaku
Rajoitteellinen optimointi	$\min f(\mathbf{x})$, $\mathbf{x} \in \mathcal{F}_2$	`fmincon`	SQP-menet.
Monitavoite-optimointi	$\min_{\mathbf{x},\gamma} \mathbf{f}(\mathbf{x}) - \mathbf{w}\gamma \le \mathbf{g}$	`fgoalattain`	SQP ja ansiofunktio
Minimax-optimointi	$\min \max_{1\le i\le p} f_i(\mathbf{x})$, $G(\mathbf{x}) \le \mathbf{0},\ \mathbf{x} \in \mathbb{R}^n$	`fminimax`	SQP ja ansiofunktio
Epälineaarinen pns-tehtävä	$\min \sum_{i=1}^m f_i(\mathbf{x})^2$, $\mathbf{x} \in \mathbb{R}^n$	`lsqnonlin`	Levenberg-Marquardt tai Gauss-Newton
Epälineaariset yhtälöt	$\mathbf{f}(\mathbf{x}) = \mathbf{0},\ \mathbf{f} : \mathbb{R}^n \mapsto \mathbb{R}^n$	`fsolve`	Levenberg-Marquardt tai Gauss-Newton

Mathematican käyttöä kuvaillaan mm. teoksissa *Mathematica, A System for Doing Mathematics by Computer* [Wol91], *Johdatus Mathematica-ohjelmiston käyttöön* [Lau92] ja *Mathematica-opas* [Rus93a]. Maplen alkeisoppaita ovat mm. *MAPLE, First Leaves, A Tutorial* [CGG+] ja *The Maple Handbook* [Red92].

B.3.3 Mallinnuskieli GAMS

GAMS (General Algebraic Modeling System) on mallinnuskieli ja ratkaisuohjelmisto lineaarisille, epälineaarisille ja sekalukuongelmille. GAMS soveltuu käytettäväksi mm. taloustieteen ja operaatiotutkimuksen mallinnustehtävissä. GAMSin käyttöopas on *GAMS: A User's Guide* [BKM88]. Lisäksi CSC on julkaissut teoksen *GAMS-ohjelmiston pikaopas* [Haa93]. Www-osoitteesta

```
http://www.csc.fi/oppaat/gams/
```

löytyy kurssimateriaalia GAMSin käytöstä.

Ohjelmistoon kuuluu laajahko mallikirjasto (noin sata mallia mm. kansantaloustieteen ja operaatiotutkimuksen aloilta), joita voi käyttää hyväksi omassa mallinnustyössä. Mallikirjastosta löytyy esimerkkejä LP-, NLP- ja MIP-tehtävistä sekä mm. kuljetus- ja verkkoprobleemien esittämisestä ja ratkaisusta GAMS-kielellä.

B.3.4 Tilasto-ohjelmistot SAS ja Splus

SAS/OR on operaatiotutkimuksen tarpeisiin kehitetty ohjelmistopaketti, joka osaa käsitellä mm. LP-, verkko- ja kuljetustehtäviä. SAS/STAT puolestaan on kattavimpia tilastoanalyysipaketteja ja sisältää mm. useita rutiineita pienimmän neliösumman tehtävien ratkaisemiseen.

SAS-ohjelmiston käyttöoppaita on useita, joista mainittakoon *SAS/OR User's Guide* [SASa], *SAS/STAT User's Guide* [SASb] ja *SAS/OR-opas* [Rus93b].

Splus on puolestaan modulaarinen Unix-ympäristön tilastollinen ohjelmisto ja sisältää rutiineja mm. LSQ-tehtäviin.

SAS- ja Splus-ohjelmistoja on esitelty CSC:n oppaassa *Matemaattiset ohjelmistot* [HHL+03].

Taulukko B.4: *IMSL-aliohjelmakirjaston (versio 3.0) rutiineja optimointitehtävien ratkaisuun.*

Tehtävä	*Rutiinit*	*Menetelmä*
Skalaarifunktio	`uvmif uvmid`	Interpolaatio ym.
Skalaarifunktio	`uvmgs`	Kultainen leikkaus
Lineaarinen optimointi	`dlprs`	Simplex-menetelmä
Kvadraattinen optimointi	`qprog`	*Dual quadratic programming*
Rajoittamaton NLP	`uminf uming`	Sekanttimenetelmä
Rajoittamaton NLP	`umidh umiah`	Modifioitu Newton
Rajoittamaton NLP	`umcgf umcgg`	Liittogradienttimenetelmä
Rajoittamaton NLP	`umpol`	Polytooppihaku
Laatikkoraj. NLP	`bconf bcong`	Sekanttimenetelmä
Laatikkoraj. NLP	`bcodh bcoah`	Modifioitu Newton
Laatikkoraj. NLP	`bcpol`	Polytooppihaku
Lin. raj. NLP	`lconf lcong`	Aktiivijoukko ja BFGS
Rajoitteellinen NLP	`nconf ncong`	SQP-menetelmä
NLSQ	`unlsf unlsj`	Levenberg-Marquardt
Laatikkoraj. NLSQ	`bclsf bclsj`	Levenberg-Marquardt

B.3.5 Aliohjelmakirjastot IMSL ja NAG

IMSL- ja NAG-aliohjelmakirjastot ovat keskenään kilpailevia hyvin dokumentoituja matemaattisia aliohjelmakirjastoja. IMSL:n vahvuus on tilastollisessa analyysissä ja NAGin lineaarialgebrassa. Kummassakin kirjastossa on kattava joukko Fortran-rutiineita tavallisimpien optimointitehtävien ratkaisemi-

Taulukko B.5: *NAG-aliohjelmakirjaston (Mark 16) rutiineja optimointitehtävien ratkaisuun.*

Tehtävä	*Rutiinit*	*Menetelmä*
Skalaarifunktio	`e01abf` `e01bbf`	Interpolaatio
Lineaarinen optimointi	`e04mff`	Aktiivijoukkomenetelmä
Kvadraattinen optimointi	`e04nff`	Aktiivijoukkomenetelmä
NLSQ sekä QP	`e04ncf`	Aktiivijoukkomenetelmä
Rajoittamaton NLP	`e04ccf`	Polytooppihaku
Rajoittamaton NLP	`e04dgf`	Raj. muistin sekanttimenet.
Laatikkoraj. NLP	`e04kad` `e04kaf`	Sekanttimenetelmä
Laatikkoraj. NLP	`e04kcf`	Sekanttimenetelmä
Laatikkoraj. NLP	`e04laf` `e04lbf`	Modifioitu Newton
Laatikkoraj. NLP	`e04kdf`	Modifioitu Newton
Rajoitteellinen NLP	`e04ucf`	SQP-menetelmä
NLSQ	`e04hef` `e04hff`	Gauss-Newton
NLSQ	`e04gdf` `e04gef`	Gauss-Newton
NLSQ	`e04gbf` `e04gcf`	Sekanttimenetelmä
NLSQ	`e04fcf` `e04fdf`	Gauss-Newton (ei deriv.)
Rajoitteellinen NLSQ	`e04upf`	SQP-menetelmä
Sekalukutehtävät	`h02bbf`	Branch and bound
Sijoittelutehtävät	`h03abf`	Erikoismenetelmä

seen. Korkeatasoisista käsikirjoista löytyy ohjeet algoritmin valitsemiseen ja tehtävän ratkaisemiseen.

NAG-aliohjelmakirjaston optimointitehtävien ratkaisurutiinit sijaitsevat pääosin luvussa E04. IMSL:n optimointirutiineja löytyy puolestaan kirjastoista IMSL/MATH ja IMSL/STAT.

Taulukoissa B.4 ja B.5 on lueteltu IMSL- ja NAG-aliohjelmakirjastojen optimointirutiineita. Katso lisätietoa käsikirjoista [IMS, Num].

B.4 Lineaarinen, kvadraattinen ja sekalukuoptimointi

Internet-verkosta löytyvä artikkeli *Linear Programming FAQ* (liite E) on hyvä katsaus saatavilla oleviin lineaarisen ja sekalukuoptimoinnin ohjelmistoihin. Seuraavassa mainittavien erityisohjelmistojen lisäksi Matlab, Maple, SAS ja GAMS soveltuvat LP-tehtävien ratkaisemiseen.

B.5 Epälineaarinen optimointi

Internet-verkosta löytyvä artikkeli *Nonlinear Programming FAQ* (liite E) sisältää epälineaarisen optimoinnin ohjelmistokatsauksen.

B.5.1 Minos

Minos on Stanfordin yliopistossa kehitetty varsin monipuolinen optimointiohjelmisto, joka kuitenkaan ei sisällä varsinaista käyttöliittymää [MS87]. Minos-ohjelmistosta on hankittu akateeminen käyttölisenssi CSC:hen.

Minos sisältää simplex-menetelmän ja sekanttimenetelmän sekä projisoidun Lagrangen funktion menetelmät. Minos osaa lukea MPS-muodossa esitettyjä LP-malleja; epälineaariset mallit esitetään Fortran-aliohjelmina.

B.5.2 FSQP

Aliohjelmakirjasto FSQP (Feasible Sequential Quadratic Programming) on rajoitteita sisältävien epälineaaristen min-max -optimointitehtävien ratkaisuohjelmisto [HHL+03]. Optimoitavien funktioiden ja rajoitusehtojen tulee olla jatkuvasti differentioituvia [ZT95]. FSQP:sta on saatavissa Fortran-kielinen versio nimeltä FFSQP ja C-kielinen ohjelmisto nimeltä CFSQP.

Kvadraattisten osatehtävien ratkaisijana ohjelmistoon kuuluu QLD-rutiini, joka on Klaus Schittkowskin kehittämä QP-ratkaisija. FSQP on saatavilla vapaasti akateemiseen käyttöön.

B.6 Pienimmän neliösumman tehtävät

Pienimmän neliösumman tehtäviä voi ratkoa Matlabilla sekä IMSL- ja NAG-aliohjelmakirjastoilla. Myös symbolienkäsittelyjärjestelmiä kuten Mathematica ja Maple voi käyttää apuna. Myös TOMS-algoritmeista löytyy rutiineja PNS-tehtäviin. Lineaarisiin tehtäviin soveltuu esimerkiksi aliohjelmakirjasto Lapack tai Matlab-ohjelmisto.

B.6.1 Minpack

Minpack on Netlibistä saatava klassinen kokoelma Fortran-rutiineja pienimmän neliösumman tehtävien ja epälineaaristen yhtälöryhmien ratkaisemiseen. Minpackin sisältämiä rutiineita vastaavia algoritmeja löytyy myös NAG- ja IMSL-aliohjelmakirjastoista.

Minpackista on kehitteillä myös uudempi versio, jota kutsutaan nimellä Minpack-2. Tällä hetkellä ohjelmistosta on saatavissa jonkin verran puut-

teellinen ja huonosti dokumentoitu esiversio. Minpack-2:n tavoitteena on parantaa suurten pienimmän neliösumman tehtävien ratkaisemista.

B.6.2 ODRPACK

Netlibin hakemistosta `odrpack` löytyvä ODRPACK-ohjelmisto on nimenomaan epälineaariseen mallin sovitukseen kehitetty ohjelmisto. Netlibistä löytyy lisäksi ohjelmiston laajahko käsikirja.

B.7 Globaali optimointi

Artikkeli *Nonlinear Programming FAQ* sisältää joitakin viittauksia globaalin optimoinnin ohjelmistoihin. Seuraavassa esiteltävien ohjelmistojen lisäksi TOMS-algoritmi 667 (SIGMA) soveltuu globaaliin optimointiin.

B.7.1 Geneettiset algoritmit

Geneettisiin algoritmeihin liittyvää tietoa löytyy WWW-palvelimesta *Genetic Algorithms Archive* osoitteesta `http://www.aic.nrl.navy.mil/galist/`.

B.8 Lineaarialgebra

Lineaarialgebran operaatioiden luotettava ja tehokas toteuttaminen on välttämätöntä optimointitehtävän ratkaisualgoritmin toiminnan kannalta. Seuraavassa kerrotaan tärkeimmistä lineaarialgebran ohjelmistopaketeista.

B.8.1 Lapack

Lapack-ohjelmistoa on kuvattu viitteissä [HHL+03, ABB+92]. Lapack on erittäin hyödyllinen lineaarialgebran operaatioita sisältävien ohjelmistojen rakennuspalikkana. Lapackin käsikirja löytyy WWW-järjestelmästä osoitteesta

```
http://http.ucar.edu/SOFTLIB/LAPACK.html
```

Tässä kirjassa on käytetty Lapack-ohjelmistoa mm. taulukossa 6.3 sivulla 111 esitetyssä Fortran-koodissa. Lapackista on tulossa versio myös rinnakkaiskoneille (ScaLapack).

B.8.2 Harvat matriisit

Harvoja matriiseja käsittelevät ohjelmistot Sparse, Sparse-BLAS ja Y12M löytyvät Netlibistä. Saatavilla on lisäksi ohjelmistot YSMP (Yale Sparse Matrix Package) ja LASPack (ANSI C -kielinen). Myös esimerkiksi Harwell-aliohjelmakirjasto sisältää rutiineita harvojen matriisien käsittelyyn.

Taulukko B.6: *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	Epälineaarinen optimointi
739	Rajoitteeton optimointi
744	Globaali rajoitteinen optimointi
745	Fortran 90 -versio algoritmista 630
746	Automaattinen derivointi (Fortran)
750	Kauppamatkustajan ongelma
754	Kvadraattiset sijoittelutehtävät
755	Automaattinen derivointi (C/C++)
765	Suuren rajoitteettomat tehtävät (STENMIN)
768	Epälineaariset yhtälöryhmän ja pienimmän neliösumman tehtävät (TENSOLVE)
774	Laatikkorajoitteisten optimointitehtävien generointi
778	Suurten laatikkorajoitteisten optimointitehtävien ratkaiseminen (L-BFGS-B)
811	Epäsileiden optimointitehtävien ratkaiseminen (NDA)
813	Konveksien rajoitettujen tehtävien ratkaiseminen (SPG)

B.9 Netlibin sisältämiä ohjelmistoja

Netlib-ohjelmistoarkistosta www-osoitteesta `http://www.netlib.org` löytyy optimointitehtävien ratkaisemiseen käyttökelpoisia ohjelmistoja hakemistoista `minpack`, `matlab`, `opt/praxis`, `opt/subplex`, `opt/ve08` ja `slatec`. Myös Lapack ja eräät muut aiemmin tässä luvussa esitellyt ohjelmistot löytyvät Netlibistä.

B.9.1 TOMS-algoritmit

Taulukossa B.6 on lueteltu Netlibin hakemistosta `toms` (ACM Transactions on Mathematical Software) löytyviä TOMS-algoritmeja. Lisätietoja löytyy sarjajulkaisusta *ACM Transactions on Mathematical Software*. Arkiston käyttöä on kuvattu CSC:n oppaassa *Matemaattiset ohjelmistot* [HHL+03].

TOMS-algoritmit löytyvät Netlib-verkkoarkistosta osoitteesta

```
http://www.netlib.org/toms/
```

B.10 Oppaan esimerkkiohjelmat

Tämän teoksen esimerkkiohjelmat on tarkoitettu lähinnä demonstraatiotarkoituksiin eli niitä ei ole kehitetty vaativiin laskentatehtäviin. Tätä varten on tarjolla runsaasti valmisohjelmistoja. Käytännön tehtävissä kannattaa aina käyttää monipuolisia ja luotettavia ohjelmistoja tai esimerkiksi Netlibin sisältämiä laajalti käytettyjä lähdekoodeja.

Mahdollisista esimerkkiohjelmien ongelmista kannattaa raportoida sähköpostiosoitteeseen `Juha.Haataja@csc.fi`. Kirjan esimerkkiohjelmia löytyy Funetin ftp-arkistosta osoitteesta

```
ftp://ftp.funet.fi/pub/sci/math/optopas/
```

sekä www-osoitteesta `http://www.csc.fi/math_topics/opt/`.

B.11 Lisätietoja

CSC:n opas *Matemaattiset ohjelmistot* [HHL+03] kuvaa eräitä yleisesti käytettyjä matemaattisia ohjelmistoja. Www-osoitteesta `http://gams.nist.gov/` löytyy kätevä ohjelmisto-opas *Guide to Available Mathematical Software*.

Tässä teoksessa annetut ohjelmien käyttöesimerkit ovat jossain määrin riippuvaisia CSC:n koneympäristöstä (Unix-käyttöjärjestelmä, Fortran-kääntäjän versio jne.) sekä CSC:n tarjoamista sovellusohjelmistoista. Unix-ympäristössä työskentelyssä auttaa esimerkiksi opas *CSC:n palvelinympäristö* [JKKR03].

CSC:n yhteystiedot löytyvät liitteestä E.

C Vektoriavaruus, normi ja sisätulo

C.1 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 summa-alkio 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 C.1.1 Tutuin esimerkki vektoriavaruuksista lienee n-komponenttisten reaalisten vektorien joukko $\mathbb{R}^n$, kun laskutoimitukset määritellään komponenteittain ja nolla-alkioksi sovitaan nollavektori $\mathbf{0} = (0, 0, \ldots 0)$. 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 C.1.2 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:

$$\|A\| = \max_{\|x\|=1} \|Ax\|, \tag{C.1}$$

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

$$\begin{aligned} \|A\|_1 &= \max_j \sum_i |a_{ij}|, \\ \|A\|_2 &= \sigma_1 = \sqrt{\lambda(A^T A)}, \\ \|A\|_\infty &= \max_i \sum_j |a_{ij}|, \end{aligned} \tag{C.2}$$

missä σ_1 on matriisin A suurin *singulaariarvo* eli matriisin $A^T A$ isoimman ominaisarvon neliöjuuri (katso myös sivua 29). Symmetriselle matriisille 2-normi on sama kuin itseisarvoiltaan suurin ominaisarvo.

Joskus käytetään myös Frobeniuksen normia, jossa lasketaan neliöjuuri kaikkien matriisin alkioiden neliöiden summasta:

$$\|A\|_F = \sqrt{\sum_j \sum_i |a_{ij}|^2}. \tag{C.3}$$

Huomaa, että Frobeniuksen normia ei voida johtaa mistään vektorinormista.

C.2 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: $\langle \mathbf{x}, \mathbf{x} \rangle \geq 0 \quad$ kaikilla $\mathbf{x} \in V$.
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

$$\langle f, g \rangle = \int_{\Omega} f(\mathbf{x}) g(\mathbf{x}) \, d\mathbf{x}, \tag{C.4}$$

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 C.2.1 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

$$\begin{aligned} \langle \sqrt{n + 1/2} P_n, \sqrt{m + 1/2} P_m \rangle &= \int_{-1}^{1} \sqrt{n + 1/2} \sqrt{m + 1/2} P_n(x) P_m(x) \, dx \\ &= \delta_n^m. \end{aligned}$$

D Liukulukuesitys

D.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

$$(0.0001100110011\ldots)_2 \approx (0.000110011)_2 = 0.099609375.$$

D.2 Liukulukuesityksen virheet

Määrittelemme virheen alkuperäisen reaaliluvun x ja tietokoneen esitysmuodon (yleensä liukulukuesityksen) x_f avulla seuraavasti:

$$|x_f - x| \quad \textit{absoluuttinen virhe}, \tag{D.1}$$

$$\frac{|x_f - x|}{|x|} \quad \textit{suhteellinen virhe}. \tag{D.2}$$

Esimerkki D.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:

$$C = \frac{\left| \frac{f(x_f) - f(x)}{f(x)} \right|}{\left| \frac{x_f - x}{x} \right|}.$$

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

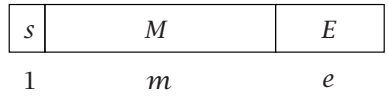
$$\epsilon_{a/b} \leq \frac{\epsilon_b |a_f| + \epsilon_a |b_f|}{b|b_f|} \approx \frac{\epsilon_b |a_f| + \epsilon_a |b_f|}{|b_f|^2}.$$

D.3 IEEE-aritmetiikka

Nollasta poikkeava luku x esitetään IEEE-aritmetiikan mukaisessa liukulukuesityksessä muodossa

$$x = (-1)^s M \cdot 2^E. \tag{D.3}$$

Kuva D.1 havainnollistaa liukuluvun binääristä talletusmuotoa tietokoneessa.



Kuva D.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.

D.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 D.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 D.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 D.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 D.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.

D.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 D.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 D.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 $\sqrt{\mathrm{OFL}}$ aiheuttavat ylivuodon. Täten laskukaavaa ei voi käyttää esimerkiksi 32-bittisessä aritmetiikassa suuruusluokkaa 10^{19} suuremmille luvuillelle ($\mathrm{OFL} \approx 10^{38}$).

Parempi tapa laskea itseisarvo on

$$|x + yi| = \begin{cases} |x|\sqrt{1 + (y/x)^2}, & \text{kun } |x| \geq |y|, \\ |y|\sqrt{1 + (x/y)^2}, & \text{kun } |x| < |y|. \end{cases} \tag{D.4}$$

Ennen laskuja on tarkistettava, että kompleksiluvun x- ja y-komponentit ovat $\neq 0$.

D.6 Virheiden kasautuminen

Pitkissä peräkkäisissä laskutoimituksissa voivat virheet kasautua jopa niin, että tarkkuus menetetään täysin.

Esimerkki D.6.1 Tutkimme eksponenttifunktion laskemista sarjakehitelmällä

$$e^x = 1 + x + \frac{x^2}{2!} + \cdots = \sum_{i=0}^{\infty} \frac{x^i}{i!} \tag{D.5}$$

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.

D.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.

E 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)
- *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:n palvelinympäristö* (Manta Jääskeläinen, Sirpa Kotila ja Tiina Kupila-Rantala, toim.; CSC, 2003)
  `http://www.csc.fi/oppaat/palvelimet/`
- *CSC User's Guide* (Kupila-Rantala, toim.; CSC, 2000)
  `http://www.csc.fi/oppaat/cscuser/`
- *Fortran 90/95* (Haataja, Rahola ja Ruokolainen; CSC, 2001)
  `http://www.csc.fi/oppaat/f95/`
- *Käytännön ohjeita ohjelmankehitykseen* (Haataja, toim.; CSC, 2003)
  `http://www.csc.fi/oppaat/ohjelmointi/`
- *Rinnakkaisohjelmointi MPI:llä* (Haataja ja Mustikkamäki; CSC, 2001)
  `http://www.csc.fi/oppaat/mpi/`

Löydät lisätietoja CSC:n julkaisuista www-osoitteesta

`http://www.csc.fi/lehdet/index.phtml.fi`

CSC:n oppaita voit tilata www-osoitteesta

`http://www.csc.fi/lehdet/oppaat.phtml.fi`

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.

[Abb94] P. C. Abbott. InterCall - Linking Mathematica and Numerical Routines. Teoksessa: Heikki Apiola, Marko Laine ja Esko Valkeila, toim., *Proceedings of the Workshop on Symbolic and Numeric Computing*, sivu 85. Rolf Nevanlinna Institute, 1994. Research Reports B10.

[AD94] Natalia Alexandrov ja J. E. Dennis, Jr. Multilevel Algorithms for Nonlinear Optimization. Raportti, NASA Langley Research Center, 1994.

[AK89] Emile Aarts ja Jan Korst. *Simulated Annealing and Boltzmann Machines*. Wiley, 1989.

[Ala92] Jarmo T. Alander, toim. *Geneettiset algoritmit – Genetic Algorithms*. TKK, Tietotekniikan osasto, 1992. Seminaarijulkaisu.

[AMO91] Ravindra K. Ahuja, Thomas L. Magnanti ja James B. Orlin, Some Recent Advances in Network Flows, *Siam Review*, 2/1991, Volume 33, sivut 175–219.

[BCC+92] Christian Bischof, Alan Carle, George Corliss, Andreas Griewank ja Paul Hovland, ADIFOR—Generating Derivative Codes from Fortran Programs, *Scientific Programming*, 1992, Volume 1, sivut 11–29.

[BGL+92] Robert E. Bixby, John W. Gregory, Irwin J. Lusting, Roy E. Marsten ja David F. Shanno, Very Large-Scale Linear Programming: A Case Study in Combining Interior Point and Simplex Methods, *Operations Research*, 5/1992, Volume 40, sivut 885–897.

[Bha00] M. Asghar Bhatti. *Practical Optimization Methods With Mathematica Applications*. Springer Verlag, 2000.

[BKM88] Anthony Brooke, David Kendrick ja Alexander Meeraus. *GAMS—A User's Guide*. The Scientific Press, 1988.

[BL85] A. Buckley ja A. Lenir, Algorithm 630. BBVSCG — A Variable-Storage Algorithm for Function Minimization, *ACM Transactions on Mathematical Software*, 2/1985, Volume 11, sivut 103–119.

[BL93] Nicolas Boissin ja Jean-Luc Lutton, A parallel simulated annealing algorithm, *Parallel Computing*, 1993, Volume 19, sivut 859–872.

[BN89] Richard H. Byrd ja Jorge Nocedal, A Tool for the Analysis of Quasi-Newton Methods with Application to Unconstrained Minimization, *SIAM Journal on Numerical Analysis*, 3/1989, Volume 26, sivut 727–739.

[Bor92] Brian Borchers. *Improved Branch and Bound Algorithms for Integer Programming*. PhD thesis, Rensselaer Polytechnic Institute, Troy, New York, 1992.

[BSS93] Mokhtar S. Bazaraa, Hanif D. Sherali ja C. M. Shetty. *Nonlinear Programming: Theory and Algorithms*. Wiley, 1993.

[BT88] Dimitri P. Bertsekas ja John N. Tsitsiklis. *Parallel and Distributed Computation*. Prentice-Hall, 1988.

[Cap92] Patrick Capolsini. *MacroC: C code generation within Maple*. INRIA, Université de Nice, 1992.

[CGG+] Bruce W. Char, Keith O. Geddes, Gaston H. Gonnet, Michael B. Monagan ja Stephen M. Watt. *MAPLE, First Leaves, A Tutorial — Introduction to Maple*. Waterloo Maple Publishing.

[CGT94] A. R. Conn, N. I. M Gould ja Ph. L. Toint. Large-scale nonlinear constrained optimization: a current survey. Raportti, CERFACS, 1994. Report TR/PA/94/03.

[CMMR87] A. Corana, M. Marchesi, C. Martini ja S. Ridella, Minimizing Multimodal Functions of Continuous Variables with the "Simulated Annealing" Algorithm, *ACM Transactions on Mathematical Software*, 3/1987, Volume 13, sivut 262–280.

[Col84] Thomas F. Coleman. *Large Sparse Numerical Optimization*. Springer-Verlag, 1984. Lecture Notes in Computer Science 165.

[CU93] A. Cichocki ja R. Unbehauen. *Neural Networks for Optimization and Signal Processing*. Wiley, 1993.

[Dav87] Lawrence Davis. *Genetic algorithms and simulated annealing*. Pitman, 1987.

[Dav90] Yuval Davidor. *An Intuitive Introduction to Genetic Algorithms as Adaptive Optimizing Procedures*. The Weismann Institute of Science, 1990.

[Dav91] Lawrence Davis, toim. *Handbook of Genetic Algorithms*. Van Hostrand Reinhold, 1991.

[DGR90] M. M. Doria, J. E. Gubernatis ja D. Rainer, Solving the Ginzburg-Landau equations by simulated annealing, *Physical Review B*, 10/1990, Volume 41, sivu 6335.

[DS78] L. C. W. Dixon ja G. P. Szegö, toim. *Towards Global Optimization 2*. North-Holland, 1978.

[DS83] J. E. Dennis, Jr ja Robert B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, 1983.

[ES91] Elizabeth Eskow ja Robert B. Schnabel, Algorithm 695. Software for a New Modified Cholesky Factorization, *ACM Transactions on Mathematical Software*, 3/1991, Volume 17, sivut 306–312.

[EWK90] Lars Eldén ja Linde Wittmeyer-Koch. *Numerical Analysis: An Introduction*. 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. 1992.

[Fle87] R. Fletcher. *Practical Methods of Optimization*. Wiley, 1987.

[FM68] Anthony V. Fiacco ja Garth P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. Wiley, 1968.

[Fra87] Joel Franklin, Convergence in Karmarkar's Algorithm for Linear Programming, *SIAM Journal on Numerical Analysis*, 4/1987, Volume 24.

[Fre93] James Freeman, Simulating a Basic Genetic Algorithm, *The Mathematica Journal*, 2/1993, Volume 3, sivut 52–56.

[GC91] Andreas Griewank ja George F. Corliss, toim. *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*. SIAM, 1991.

[GFR94] Goffe, Ferrier ja Rogers, Global Optimization of Statistical Functions with Simulated Annealing, *Journal of Econometrics*, 1/2/1994, Volume 60, sivut 65–100.

[GMS+86] P. E. Gill, W. Murray, M. A. Saunders, J. A. Tomlin ja M. H. Wright. On Projected Newton Barrier Methods for Linear Programming and an Equivalence to Karmarkar's Projective Method. Teoksessa: *Mathematical Programming*, nr. 36, sivut 183–209. 1986.

[GMSW84a] Philip E. Gill, Walter Murray, Michael A. Saunders ja Margaret H. Wright. Model Building and Practical Aspects of Nonlinear Programming. Teoksessa: Klaus Schittkowski, toim., *Computational Mathematical Programming*. Springer-Verlag, 1984. NATO ASI Series.

[GMSW84b] Philip E. Gill, Walter Murray, Michael A. Saunders ja Margaret H. Wright, Procedures for Optimization Problems with a Mixture of Bounds and General Linear Constraints, *ACM Transactions on Mathematical Software*, 3/1984, Volume 10, sivut 282–298.

[GMW81] Philip E. Gill, Walter Murray ja Margaret H. Wright. *Practical Optimization*. Academic Press, 1981.

[GMW91] Philip E. Gill, Walter Murray ja Margaret H. Wright. *Numerical Linear Algebra and Optimization. Volume 1*. Addison-Wesley, 1991.

[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/`.

[Gom90] Claude Gomez. *Macrofort: A FORTRAN code generator in Maple*. INRIA, 1990. Technical Report 119.

[Gon92] Clovis C. Gonzaga, Path-following methods for linear programming, *SIAM Review*, 2/1992, Volume 34, sivut 167–224.

[Gra93] Andrew Grace. *Optimization Toolbox For Use with MATLAB*. The MathWorks, Inc, 1993.

[GSB+92] J. Garner, M. Spanbauer, R. Benedek, K. J. Strandburg, S. Wright ja P. Plassmann, Critical fields of Josephson-coupled superconducting multilayers, *Physical Review B*, 14/1992, Volume 45, sivu 7973.

[GvHM91] Victor V. Goldman, Johan A. van Hulzen ja Jan H. J. Molenkamp. Efficient Numerical Program Generation and Computer Algebra Environments. Teoksessa: Andreas Griewank ja George F. Corliss, toim., *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, sivu 74. SIAM, 1991.

[GvL89] Gene H. Golub ja Charles F. van Loan. *Matrix Computations*. The Johns Hopkins University Press, 1989.

[Haa93] Juha Haataja. *GAMS-ohjelmiston pikaopas*. CSC, 1993. 35 s.

[Haa94] Juha Haataja. *Suurten optimointitehtävien ratkaiseminen*. Lisensiaattityö, TKK, 1994. 142 s.

[Haa01] Juha Haataja. *GAMS-ohjelmiston pikaopas*. CSC, 2001. `http://www.csc.fi/oppaat/gams/`.

[Haa02] Juha Haataja, toim. *Alkuräjähdyksestä kännykkään — näkökulmia laskennalliseen tieteeseen*. CSC, 2002. Www-osoite `http://www.csc.fi/oppaat/lask.tiede/`.

[HB92] Frank Hoffmeister ja Thomas Bäck. Genetic Algorithms and Evolution Strategies: Similarities and Differences. Raportti, Systems Analysis Reseach Group, University of Dortmund, BOX 50 05 00, D-4600 Dortmund 50, Germany, 1992. Technical Report No. SYS – 1/92.

[Hea97] Michael T. Heath. *Scientific Computing — An Introductory Survey*. McGraw-Hill, 1997.

[Hes80] Magnus Hestenes. *Conjugate Direction Methods in Optimization*. Springer-Verlag, 1980. Applications of Mathematics 12.

[HH91] Günther Hämmerlin ja Karl-Heinz Hoffman. *Numerical Mathematics*. Springer-Verlag, 1991.

[HHL+02] Juha Haataja, Jussi Heikonen, Yrjö Leino, Jussi Rahola, Juha Ruokolainen ja Ville Savolainen. *Numeeriset menetelmät käytännössä*. CSC, 2002.

[HHL+03] Juha Haataja, Jussi Heikonen, Esa Lammi, Yrjö Leino, Mikko Lyly ja Ville Savolainen, toim. *Matemaattiset ohjelmistot*. CSC, 2003. Www-osoite `http://www.csc.fi/oppaat/mat.ohj/`.

[HJKR02] Juha Haataja, Jari Järvinen, Jari Koponen ja Peter Råback, toim. *Laskennallinen tuotekehitys: suunnittelun uusi ulottuvuus*. CSC, 2002. Www-osoite `http://www.csc.fi/oppaat/tuotekehitys/`.

[HKR93] Juha Haataja, Juhani Käpyaho ja Jussi Rahola. *Numeeriset menetelmät*. CSC, 1993. 236 s.

[HM01] Juha Haataja ja Kaj Mustikkamäki. *Rinnakkaisohjelmointi MPI:llä*. CSC, 2001.

[Hoc91] Leslie M. Hocking. *Optimal Control, An Introduction to the Theory with Applications*. Clarendon Press, Oxford, 1991.

[HPY91] C. Han, P. M. Pardalos ja Y. Ye. On Interior-Point Algorithms for Some Entropy Optimization Problems. Raportti, Department of Computer Science, The Pennsylvania State University, 1991. CS-91-02.

[HR93] Juha Haataja ja Matti Ryynänen, Synkrotronisäteilylähteen optimointi geneettisellä algoritmilla, *Supermenu*, 4/1993, sivut 12–15.

[HRR01] Juha Haataja, Jussi Rahola ja Juha Ruokolainen. *Fortran 90/95*. CSC, 2001.

[HS84] Daniel P. Heyman ja Matthew J. Sobel. *Stochastic Models in Operations Research, Volume II — Stochastic Optimization*. McGraw-Hill, 1984.

[HT02] Jari P. Hämäläinen ja Hannu Turpeinen. Virtauslaskentaa virtuaaliympäristössä. Teoksessa: Juha Haataja, Jari Järvinen, Jari Koponen ja Peter Råback, toim., *Laskennallinen tuotekehitys: suunnittelun uusi ulottuvuus*, sivu 116. CSC, 2002. Www-osoite `http://www.csc.fi/oppaat/tuotekehitys/`.

[IMS] IMSL Inc. *MATH/LIBRARY*.

[Ing93] Lester Ingber, Simulated annealing: Practice versus theory, *Mathematical and Computer Modelling*, 11/1993, Volume 18, sivut 29–57.

[IR92] Lester Ingber ja Bruce Rosen, Genetic Algorithms and Very Fast Simulated Reannealing: A Comparison, *Mathematical and Computer Modelling*, 11/1992, Volume 16, sivut 87–100.

[Iri91] Masao Iri. History of Automatic Differentiation and Rounding Error Estimation. Teoksessa: Andreas Griewank ja George F. Corliss, toim., *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, sivut 3–16. SIAM, 1991.

[JKKR03] Manta Jääskeläinen, Sirpa Kotila ja Tiina Kupila-Rantala, toim. *CSC:n palvelinympäristö*. CSC, 2003. Www-osoite `http://www.csc.fi/oppaat/palvelimet/`.

[JP93] Mark T. Jones ja Paul E. Plassmann, Computation of Equilibrium Vortex Structures for Type-II Superconductors, *The International Journal of Supercomputer Applications*, 2/1993, Volume 7, sivut 129–143.

[Kan93] Kari Kankaala. *Monte Carlo Simulations: Methodology and Application to H/W(100)*. PhD thesis, Tampere University of Technology, CSC, 1993.

[Kap88] J. N. Kapur. *Mathematical Modelling*. Wiley, 1988.

[Kar84] N. Karmarkar. A New Polynomial-time Algorithm for Linear Programming. Teoksessa: *Combinatorica*, nr. 4, sivut 373–395. 1984.

[Kar95] Hannu Karttunen. *Datan käsittely*. CSC, 1995. 224 s.

[KBT84] A. H. G. Rinnooy Kan, C. G. E. Boender ja G. Th. Timmer. A Stochastic Approach to Global Optimization. Teoksessa: Klaus Schittkowski, toim., *Computational Mathematical Programming*. Springer-Verlag, 1984. NATO ASI Series.

[KH80] Kennington ja Helgason. *Algorithms for Network Programming*. Wiley, 1980.

[Kiv84] Simo K. Kivelä. *Matriisilasku ja lineaarialgebra*. Otakustantamo, 1984.

[Kiv91] Simo K. Kivelä. *MATLAB-opas*. Otakustantamo, 1991.

[KLT03] Tamara G. Kolda, Robert Michael Lewis ja Virginia Torczon, Optimization by direct search: New perspectives on some classical and modern methods, *SIAM Review*, 3/2003, Volume 45, sivut 385–482.

[KMN89] David Kahaner, Cleve Moler ja Stephen Nash. *Numerical Methods and Software*. Prentice-Hall, 1989.

[KN94] G. Keady ja G. Nolan. Production of Argument SubPrograms in the AXIOM – NAG link: examples involving nonlinear systems. Teoksessa: Heikki Apiola, Marko Laine ja Esko Valkeila, toim., *Proceedings of the Workshop on Symbolic and Numeric Computing*. Rolf Nevanlinna Institute, 1994. Research Reports B10.

[Koz92] John R. Koza. *Genetic Programming: On Programming Computers by Means of Natural Selection and Genetics*. MIT Press, 1992.

[Lau92] Petri Laukkanen. *Johdatus Mathematica-ohjelmiston käyttöön*. Jyväskylän yliopisto, Matematiikan laitos, 1992.

[LMS92a] Irvin J. Lustig, Roy E. Marsten ja David F. Shanno. Computational Experience with a Globally Convergent Primal-Dual Predictor-Corrector Algorithm for Linear Programming. Raportti, September 1992.

[LMS92b] Irvin J. Lustig, Roy E. Marsten ja David F. Shanno. Interior Point Methods for Linear Programming: Computational State of the Art. Raportti, December 1992.

[Lok77] Olli Lokki. *Matemaattinen ohjelmointi II*. OtaData ry, 1977.

[LR88] F. A. Lootsma ja K. M. Ragsdell, State-of-the-art in parallel nonlinear optimization, *Parallel Computing*, 1988, Volume 6, sivut 133–155.

[Lue69] David G. Luenberger. *Optimization by Vector Space Methods*. Wiley, 1969.

[Luk92] Ladislav Lukšan. Computational Experience with Known Variable Metric Updates. Raportti, Institute of Computer Science, Academy of Sciences of the Czech Republic, 1992. Technical report No. V-534.

[MAB89] Nezam Mahdavi-Amiri ja Richard H. Bartels, Constrained Nonlinear Least Squares: An Exact Penalty Approach with Projected Structured Quasi-Newton Updates, *ACM Transactions on Mathematical Software*, 3/1989, Volume 15, sivut 220–242.

[Mae90] Roman Maeder. *Programming in Mathematica*. Addison-Wesley, 1990.

[Mäk90] Marko M. Mäkelä. *Nonsmooth Optimization — Theory and Algorithms with Applications to Optimal Control*. University of Jyväskylä, Department of Mathematics, 1990.

[Mäk93] Marko M. Mäkelä. *Issues of Implementing a Fortran Subroutine Package NSOLIB for Nonsmooth Optimization*. University of Jyväskylä, Department of Mathematics, Laboratory of Scientific Computing, 1993. Report 5/93.

[McK93] Ken McKinnon. Discrete and Continuous Global Optimization. Raportti, August 1993. Department of Mathematics and Statistics, University of Edinburgh.

[Meg89] Nimrod Megiddo, toim. *Progress in Mathematical Programming. Interior-Point and Related Methods*. Springer-Verlag, 1989.

[Mey84] R. R. Meyer. Network Optimization. Teoksessa: Klaus Schittkowski, toim., *Computational Mathematical Programming*. Springer-Verlag, 1984. NATO ASI Series.

[MF99] Zbigniew Michalewicz ja David B. Fogel. *How to Solve It: Modern Heuristics*. Springer Verlag, 1999.

[Mic98] Melanie Michell. *An Introduction to Genetic Algorithms*. MIT Press, 1998.

[Mie98] Kaisa Miettinen. *Optimointi*. Lecture Notes 33. University Jyväskylä, Department of Mathematics, 1998.

[Mie99] Kaisa Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, 1999.

[MN93] Marko M. Mäkelä ja Pekka Neittaanmäki. *Nonsmooth Optimization: Analysis and Algorithms with Applications to Optimal Control*. World Scientific, 1993.

[MNV82] Matti Mäkelä, Olavi Nevanlinna ja Juhani Virkkunen. *Numeerinen matematiikka*. Gaudeamus, 1982.

[Mon94] Michael B. Monagan. Computing Partial Derivatives Using Algorithmic Differentiation in Maple. Teoksessa: Heikki Apiola, Marko Laine ja Esko Valkeila, toim., *Proceedings of the Workshop on Symbolic and Numeric Computing*. Rolf Nevanlinna Institute, 1994. Research Reports B10.

[Moo93] Edward L. Mooney. Local Search Algorithms. Raportti, Industrial and Management Engineering Department, Montana State University, 1993.

[MRR+53] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller ja E. Teller, Equation of State Calculations by Fast Computing Machines, *Journal of Chemical Physics*, 1953, Volume 21, sivut 1087–1092.

[MS78] Bruce A. Murtagh ja Michael A. Saunders. *Large-Scale Linearly Constrained Optimization*. Department of Operations Research, Stanford University, 1978. Reprint of Math. Prog. **14** (1978) 41-72.

[MS87] Bruce A. Murtagh ja Michael A. Saunders. *Minos 5.1 User's Guide*. Department of Operations Research, Stanford University, 1987. Technical Report SOL 83-20R.

[MS89] Raino Mäkinen ja Kimmo Salmenjoki. *Numeeriset menetelmät*. Jyväskylän yliopisto, 1989. Matematiikan laitos.

[MW93] Jorge J. Moré ja Stephen J. Wright. *Optimization Software Guide*. SIAM, 1993. Frontiers in applied mathematics 14.

[MW] Walter Murray ja Margaret H. Wright. Line Search Procedures for the Logarithmic Barrier Function. Raportti, Department of Operations Research, Stanford University.

[Noc92] Jorge Nocedal. Theory of algorithms for unconstrained optimization. Teoksessa: *Acta Numerica*, sivut 199–242. 1992.

[NS96] Stephen G. Nash ja Ariela Sofer. *Linear and Nonlinear Programming*. McGraw-Hill, 1996.

[Num] Numerical Algorithms Group, Ltd. *NAG Fortran Library Manual, Mark 16*.

[OR70] J. M. Ortega ja W. C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press, 1970.

[Öst92] Ralf Östermark. *Parametric stability of interior point methods for linear programming*. Företagsekonomiska Instituten, Åbo Akademi, 1992. Ser. A:380.

[OvG89] R. H. J. M. Otten ja L. P. P. P. van Ginneken. *The Annealing Algorithm*. Kluwer Academic Publishers, 1989.

[Pel94] Pirkka Peltola. Generating FORTRAN argument subprograms with ASP-FOR - a collection of Maple functions. Teoksessa: Heikki Apiola, Marko Laine ja Esko Valkeila, toim., *Proceedings of the Workshop on Symbolic and Numeric Computing*, sivu 33. Rolf Nevanlinna Institute, 1994. Research Reports B10.

[Pis84] Sergio Pissanetzky. *Sparse Matrix Technology*. Academic Press, 1984.

[PR91] Manfred Padberg ja Giovanni Rinaldi, A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems, *SIAM Review*, 1/1991, Volume 33, sivut 60–100.

[PS82] Christos H. Papadimitriou ja Kenneth Steiglitz. *Combinatorial Optimization. Algorithms and Complexity.* Prentice-Hall, 1982.

[PSU88] A. L. Peressini, F. E. Sullivan ja J. J. Uhl, Jr. *The Mathematics of Nonlinear Programming*. Springer-Verlag, 1988.

[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. 2nd edition.

[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. 2nd edition.

[PV90] P. M. Pardalos ja S. A. Vavasis. Quadratic programming with one negative eigenvalue is NP-hard. Raportti, Department of Computer Science, The Pennsylvania State University, 1990. CS-90-41.

[PY91] P. M. Pardalos ja Y. Ye. Solving Some Engineering Problems Using an Interior-Point Algorithm. Raportti, Department of Computer Science, The Pennsylvania State University, 1991. CS-91-04.

[Rao79] S. S. Rao. *Optimization: Theory and Applications*. Wiley, 1979.

[Red92] Darren Redfern. *The Maple Handbook*. Springer-Verlag, 1992.

[Roc93] R. Tyrrell Rockafellar, Lagrange Multipliers and Optimality, *SIAM Review*, 2/1993, Volume 35, sivut 183–238.

[Roh95] Jii Rohn. NP-Hardness Results for Some Linear and Quadratic Problems. Raportti, Institute of Computer Science, Academy of Sciences of the Czech Republic, 1995. Technical report No. 619.

[Rus93a] Heikki Ruskeepää. *Mathematica-opas (Versio 2.2)*. Turun yliopisto, Sovellettu matematiikka, 1993. 178 sivua.

[Rus93b] Heikki Ruskeepää. *SAS/OR-opas (Versio 6.07)*. Turun yliopisto, Sovellettu matematiikka, 1993. 101 sivua.

[Sal75] Harvey M. Salkin. *Integer Programming*. Addison-Wesley, 1975.

[SASa] SAS Institute Inc. *SAS/OR User's Guide*.

[SASb] SAS Institute Inc. *SAS/STAT User's Guide*. Volume 1/Volume 2.

[Sca85] L. E. Scales. *Introduction to Non-linear Optimization*. Macmillan, 1985.

[SF92] Tamar Schlick ja Aaron Fogelson, TNPACK — A Truncated Newton Minimization Package for Large-Scale Problems: I. Algorithm and Usage, *ACM Transactions on Mathematical Software*, 1/1992, Volume 18, sivut 46–70.

[SH87] Harold Szu ja Ralph Hartley, Fast Simulated Annealing, *Physics Letters A*, 3,4/1987, Volume 122, sivut 157–162.

[Sof93a] Mark Sofroniou, Extending the Built-in Format Rules, *The Mathematica Journal*, 3/1993, Volume 3, sivut 74–80.

[Sof93b] Mark Sofroniou. *Format.m: A Usage Manual*. Loughborough University of Technology, 1993.

[Sof94] Mark Sofroniou. An Efficient Symbolic-Numeric Environment Using Mathematica. Teoksessa: Heikki Apiola, Marko Laine ja Esko Valkeila, toim., *Proceedings of the Workshop on Symbolic and Numeric Computing*. Rolf Nevanlinna Institute, 1994. Research Reports B10.

[SP80] David F. Shanno ja Kang Hoh Phua, Remark on algorithm 500: minimization of unconstrained multivariate functions, *ACM Transactions on Mathematical Software*, 1980, Volume 6, sivut 618–622.

[SP89] David F. Shanno ja Kang Hoh Phua, Numerical Experience with Sequential Quadratic Programming Algorithms for Equality Constrained Nonlinear Programming, *ACM Transactions on Mathematical Software*, 1/1989, Volume 15, sivut 49–63.

[Ste86] R. E. Steuer. *Multiple Criteria Optimization*. Wiley, 1986.

[Tah75] Hamdy A. Taha. *Integer programming: theory, applications, and computations*. Academic Press, 1975.

[Tah87] Hamdy A. Taha. *Operations Research*. Macmillan, 1987.

[Tod95] Michael J. Todd. Potential-Reduction Methods in Mathematical Programming. Raportti, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, January 1995.

[Ves95] K. Veseli , Finite Catenary and the Method of Lagrange, *Siam Review*, 2/1995, Volume 37, sivut 224–229.

[vLA88] P. J. M. van Laarhoven ja E. H. L. Aarts. *Simulated Annealing: Theory and Applications*. D. Reidel Publishing Company, 1988.

[Vos93] Heinrich Voss. *Iterative Methods for Linear Systems of Equations*. Jyväskylän yliopisto, Matematiikan laitos, 1993. Lecture notes 27.

[WH91] Z. D. Wang ja C.-R. Hu, Numerical relaxation approach for solving the general Ginzburg-Landau equations for type-II superconductors, *Physical Review B*, 21/1991, Volume 44, sivu 11918.

[Wol59] P. Wolfe, The simplex method for quadratic programming, *Econometrica*, 1959, Volume 27, sivut 382–398.

[Wol91] Stephen Wolfram. *Mathematica, A System for Doing Mathematics by Computer*. Addison-Wesley, 1991. Second Edition.

[Wri92a] Margaret H. Wright. Interior methods for constrained optimization. Teoksessa: *Acta Numerica*, sivut 341–407. 1992.

[Wri92b] Stephen J. Wright, An Interior Point Algorithm for Linearly Constrained Optimization, *SIAM Journal on Optimization*, 4/1992, Volume 1.

[Wri95] Margaret H. Wright. Direct search methods: Once scorned, now respectable, 1995. Www-osoite `http://citeseer.nj.nec.com/wright95direct.html`.

[Wria] Margaret H. Wright. Some Linear Algebra Issues in Large-Scale Optimization. Raportti. NATO Advanced Study Institute, August 3-14, 1992.

[Wrib] Stephen J. Wright. Interior Point Methods for Optimal Control of Discrete-Time Systems. Raportti, Mathematics and Computer Science Division, Argonne National Laboratory.

[Ye] Yinyu Ye. Interior-Point Algorithms for Quadratic Programming. Raportti, Department of Management Sciences, University of Iowa.

[Zow84] J. Zowe. Nondifferentiable Optimization. Teoksessa: Klaus Schittkowski, toim., *Computational Mathematical Programming*. Springer-Verlag, 1984. NATO ASI Series.

[ZT95] Jian L. Zhou ja Andrè L. Tits. *User's Guide for FFSQP Version 3.5: A FORTRAN Code for Solving Constrained Nonlinear (Minimax) Optimization Problems, Generating Iterates Satisfying All Inequality and Linear Constraints*. Electrical Engineering Department and Institute for Systems Research, University of Maryland, 1995. Systems Research Center TR-92-107r4.

Hakemisto

S

T

U

V

W

Y

Ä