TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

AAPO HONKAKUNNAS
TRUST REGION -OPTIMOINTIALGORITMI

Kandidaatintyö

Tarkastaja: Simo Ali-Löytty
7.5.2018

TIIVISTELMÄ

AAPO HONKAKUNNAS: Trust region -optimointialgoritmi
Tampereen teknillinen yliopisto
Kandidaatintyö, 22 sivua, 3 liitesivua
Huhtikuu 2018
Teknis-luonnontieteellinen koulutusohjelma
Pääaine: Matematiikka
Tarkastaja: Simo Ali-Löytty
Avainsanat: optimointi, trust region, dog-leg, algoritmit

Trust region- eli luottamusaluealgoritmit ovat iteratiivisiä optimointialgoritmeja. Niiden toiminta perustuu siihen, että minimoitavaa funktiota approksimoidaan kvadraattisella mallifunktiolla jonkin luottamusalueen sisällä. Mallifunktio pyritään minimoimaan luottamusalueen sisällä ja saatu ratkaisu hyväksytään uudeksi iteraatioaskeleeksi, mikäli se tuottaa riittävästi vähenemää funktiossa. Samalla myös luottamusalueen kokoa muutetaan.

Mallifunktion minimointia luottamusalueella kutsutaan algoritmin osaongelmaksi. Tässä työssä käsitellään kahta ratkaisutapaa, Cauchyn pistettä sekä dog-leg -metodia, jotka hakevat approksimoidun ratkaisun osaongelmalle, sekä lisäksi eksaktin ratkaisun etsimistä. Myös algoritmin suppeneminen näitä tapoja käyttäen todistetaan.

Työssä myös toteutettiin itse MATLAB-ohjelmistolla luottamusaluealgoritmi, jonka toimintaa testattiin minimoimalla sillä Rosenbrockin funktiota. Algoritmin toimintaa iteraatioprosessin aikana tutkittiin ja vertailun vuoksi sama optimointitehtävä ratkaistiin myös viivahakuun perustuvalla algoritmilla.

SISÄLLYS

1. Johdanto 1

2. Trust region -algoritmi 3

2.1 Minimikohdista 3

2.2 Trust region -algoritmin rakenne 5

2.3 Osaongelman ratkaiseminen 6

2.3.1 Dog-leg-menetelmä 8

2.3.2 Eksakti ratkaisu 9

2.4 Algoritmin suppeneminen 12

3. Algoritmin käytännön toiminta 15

3.1 Testausongelman esittely 15

3.2 Algoritmin toiminta suorituksen aikana 16

4. Yhteenveto 20

Lähteet 21

Liite A: Algoritmin MATLAB-koodi

LYHENTEET JA MERKINNÄT

A^T	matriisin A transpoosi
A^{-1}	matriisin A inverssi
A^+	matriisin A pseudoinverssi
B	mallifunktion Hessen matriisi tai sen approksimaatio
β	vakio
$c(x)$	rajoitefunktio arvolla x
δ	positiivinen vakio
Δ	luottamusalueen säde
ϵ	positiivinen vakio
$f(x)$	funktio f arvolla x
g	mallifunktion gradientti
γ	dog-leg -polun parametri
I	identiteettimatriisi
$\mathcal{L}$	Lagrangen funktio
λ	vakioarvo
k	iteraatiokerran järjestysluku ($k = 0, 1, \ldots$)
K	konveksi lukujoukko
m	kvadraattinen mallifunktio
μ	matriisin positiivisen definiittiyden korjauskerroin
η	algoritmin parametri
Ω	optimointiongelman määrittelyjoukko
$\mathbb{R}$	reaalilukujen joukko
$\mathbb{R}^n$	n-ulotteinen reaaliavaruus
ρ	iteraatioaskeleen todellisen ja ennustetun vähenemän suhde
s	iteraatioaskel
$\sigma(A)$	matriisin A ominaisarvo
τ	vakio
x^*	minimiarvo
∇f	funktion f gradientti
$\nabla^2 f$	funktion f Hessen matriisi
$\|\cdot\|$	matriisin tai vektorin euklidinen normi

1. JOHDANTO

Matemaattinen optimointi on monikäyttöinen ja laaja matematiikan osa-alue. Optimoinnin tavoitteena on löytää suotuisin vaihtoehto useiden vaihtoehtojen joukosta, ja onkin luontevaa ajatella, että jokainen ihminen optimoi jollain tasolla joka päivä. Myös elinkeinoelämässä optimoinnin käyttö on jatkuvaa esimerkiksi kustannustehokkainta toimintamallia etsittäessä. Tässä työssä tarkoituksena on selvittää ja esitellä trust region -tyyppisten optimointialgoritmien toimintaa.

Matemaattisessa optimoinnissa tavoitteena on löytää kohdefunktiolle paras mahdollinen ratkaisu mahdolliset rajoittavat ehdot huomioonottaen. Jos kohdefunktio ilmaisee saavutettavaa hyötyä, pyritään löytämään mahdollisimman suuren arvon tuottava ratkaisu ja vastaavasti funktion kuvatessa haittaa etsitään pienintä arvoa. Näitä toimenpiteitä kutsutaan maksimoinniksi ja minimoinniksi. [14] Matemaattisesti kohdefunktion $f : \mathbb{R}^n \to \mathbb{R}$ minimointia merkitään kirjoittamalla

$$\min_{x \in \Omega} \quad f(x) \tag{1.1}$$

jossa $f(x)$ on minimoitava funktio, x on muuttuja, jonka suhteen minimointi suoritetaan [19]. Maksimointitehtävätkin on tapana merkitä yhtälön (1.1) kuvaamalla tavalla minimointitehtäviksi. Tällöin kohdefunktion etumerkki vaihdetaan ja haluttu maksimi löydetään minimoimalla. Useissa tapauksissa minimointitehtävään liittyy myös rajoitteita joukkoon Ω. Rajoitteet ilmoitetaan rajoitefunktioina $c(x)_i$ ja rajoitteet yleensä monimutkaistavat ongelman ratkaisemista. Rajoitefunktiot ovat joko yhtälö- tai epäyhtälörajoitteita ja usein samassa tehtävässä on molempia. Rajoitettuun optimointiin voidaan soveltaa rajoittamattoman optimoinnin menetelmiä tietyin reunaehdoin. [19] Tässä kandidaatintyössä keskitytään käsittelemään rajoittamattomia optimointitehtäviä.

Hyvin usein optimointitehtävien kohdefunktiot ovat hankalasti käsiteltäviä ja suuridimensioisia, mistä johtuen minimointi suoralla laskulla ei ole mahdollista. Suoran laskun sijaan ratkaisun saamisessa hyödynnetään iteratiivisia optimointialgoritmeja, joissa funktiosta saatavien tietojen avulla edetään kohti minimikohtaa. Käytössä olevat algoritmit eroavat toisistaan suuresti esimerkiksi lähestymistavaltaan, var-

muudeltaan tai laskuteholtaan ja erilaisilla ominaisuuksilla varustettuja algoritmeja käytetään eri käyttötarkoituksiin. Minimin löytämiseen iteroimalla pyrkivät algoritmit voidaan jakaa lähestymistapansa pohjalta kahteen luokkaan: viivahaku- ja luottamusaluealgoritmeihin (trust region). [19] Tässä työssä keskitytään tutkimaan luottamusaluealgoritmien toimintaa.

Luottamusaluealgoritmien idea pohjautuu siihen, että iteraatioaskeleen pituus on rajoitettu. Ensimmäinen tähän lähestymistapaan viittaava algoritmi on Levenbergin vuonna 1944 kehittämä [9]. Tässä algoritmissa pyrittiin erilaisilla toimenpiteillä stabiloimaan systeemiä pienimmän neliösumman ongelman ratkaisemisessa. Levenbergin algoritmia veivät 1960-luvun alussa eteenpäin Morrison [12] ja Marquardt [10], joiden työn seurauksena syntyi Levenberg-Morrison-Marquardt-algoritmi, jota voidaan pitää luottamusaluealgoritmien eräänlaisena prototyyppinä [18]. Morén 1980-luvun alussa julkaisema artikkeli [13] toi luottamusaluealgoritmeja tunnetummaksi ja vakiinnutti luottamusalue-termin käytön yleisessä terminologiassa.

Tässä työssä esitellään ensin algoritmin teoreettinen tausta. Luvussa 2.2 käsitellään algoritmin toiminta ideatasolla sekä esitellään esimerkkialgoritmi, luvussa 2.3 esitellään tapoja ratkaista luottamusalueen osaongelma ja luvussa 2.4 todistetaan algoritmin suppeneminen. Teoreettisen käsittelyn jälkeen itse MATLAB-ohjelmistolla toteutettua algoritmia testataan Rosenbrockin funktioon. Luvussa 3.2 tarkastellaan algoritmin suoriutumista funktion minimoinnissa sekä vertaillaan suoritusta viivahakualgoritmin suoritukseen.

2. TRUST REGION -ALGORITMI

2.1 Minimikohdista

Optimointialgoritmeillä pyritään löytämään funktion globaali minimikohta.

Määritelmä 2.1.1. Funktion globaali minimikohta x^* määrittelyjoukossaan on piste, joka täyttää ehdon $f(x^*) \leq f(x)$ kaikilla määrittelyjoukon arvoilla x. Minimikohta on lokaali, jos ehto täyttyy jossain määrittelyjoukon ympäristössä $\|x - x^*\| < \delta$, kun $\delta > 0$.
Kriittinen piste on piste, jossa funktio voi saavuttaa lokaalin tai globaalin ääriarvon. Kriittisiä pisteitä ovat funktion gradientin nollakohdat, määrittelyjoukon reunat sekä epäjatkuvuuskohdat. [3]

Algoritmin saavutettua minimikohdan x^* ei kuitenkaan voida olla varmoja, onko saavutettu kohta globaali vai lokaali minimi. Tämä johtuu siitä, että etenkin monimutkaisten funktioiden kohdalla ei voida arvioida varmasti funktion käyttäytymistä. Algoritmi on löytänyt ratkaisun, mutta ei voida tietää, onko algoritmin kattamien alueiden ulkopuolella pienempiä arvoja. Monilla algoritmeilla ei pystytäkään ratkaisemaan kuin lokaali minimikohta. Algoritmin rakentamiseksi vaaditaan tietoa minimikohdan ominaisuuksista. Funktion minimikohdalle onkin olemassa joukko differentiaalilaskennasta pohjautuvia ehtoja, joiden avulla voidaan arvioida pistettä [7, Theorem 3.1, 3.2 ja 3.4]. Globaalin minimikohdan löytäminen helpottuu mikäli tarkasteltava funktio on konveksi [14, Luku 2.1].

Määritelmä 2.1.2. Joukko $K \subset \mathbb{R}^n$ on konveksi, jos jokaisella muuttujan $x, y \in K$ ja vakion $\lambda \in [0, 1]$ arvolla pätee ehto $\lambda x + (1 - \lambda)y \in K$. Funktio $f : K \to \mathbb{R}$ on konveksi, jos jokaisella muuttujan $x, y \in K$ ja vakion $\lambda \in (0, 1)$ arvolla pätee ehto

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y). \tag{2.1}$$

Konveksissa joukossa määritelty funktio $f : K \to \mathbb{R}$ on konkaavi, jos $-f$ on konveksi.

Pistejoukko $\lambda x + (1 - \lambda)y$ jollain vakion $\lambda \in [0, 1]$ voidaan ajatella pisteiden x ja y

välisenä janana. Näin ollen konveksi joukko on joukko, jossa jokainen joukon kahden pisteen väliin piirretty jana pysyy joukon sisällä. Konveksissa funktiossa taas kahden funktion kuvaajalla olevan pisteen väliin piirretty jana on aina kuvaajan yläpuolella. Näihin ominaisuuksiin perustuen konvekseille funktioille voidaan todistaa minimikohdan paikallisuutta koskeva lause.

Lause 2.1.1. *Jos funktio $f : K \to \mathbb{R}$ on konveksi, mikä tahansa sen paikallinen minimi x^* on funktion globaali minimi. Jos f on derivoituva, on mikä tahansa sen kriittinen piste x^* globaali minimi.*

Todistus. [14, Luku 2.1] Tehdään vastaoletus, että minimi x^* on funktion paikallinen, muttei globaali minimi. Tällöin voidaan aina valita piste $z \in \mathbb{R}^n$, joka täyttää ehdon $f(z) < f(x^*)$. Kun tutkitaan pisteiden x^* ja z välistä janaa d

$$d = \lambda z + (1 - \lambda)x^*, \quad \lambda \in [0, 1] \tag{2.2}$$

huomataan Määritelmän 2.1.2 funktion f konveksisuusehdon perusteella, että

$$f(d) \leq f(\lambda z + (1 - \lambda)x^*) < f(x^*). \tag{2.3}$$

Näin ollen missä tahansa pisteen x^* ympäristössä $\mathcal{N}$ on osanen janaa d. Täten on aina olemassa piste $d \in \mathcal{N}$, jolle pätee epäyhtälö (2.3). Näin ollen pisteen x^* on pakko olla globaali minimi.

Lauseen toinen osa todistetaan valitsemalla piste z samalla tavalla kuin yllä ja tekemällä vastaoletus, että piste x^* ei ole globaali minimi. Muokataan janan d lausetta muotoon

$$d = \lambda z + (1 - \lambda)x^* = x^* + \lambda(z - x^*). \tag{2.4}$$

Koska funktio f on derivoituva ja konveksi, voidaan janalle d esittää ensimmäisen kertaluvun Taylorin approksimaation mukaisesti

$$\begin{aligned}(\nabla f(x^*))^T(z - x^*) &= \frac{\mathrm{d}}{\mathrm{d}\lambda} f(x^* + \lambda(z - x^*)) \\ &= \lim_{\lambda \to 0} \frac{f(x^* + \lambda(z - x^*)) - f(x^*)}{\lambda}.\end{aligned} \tag{2.5}$$

Funktion f konveksisuuden nojalla voidaan yhtälöön (2.5) soveltaa konveksisuusehtoa (2.1) ja saadaan epäyhtälö

$$\begin{aligned}\lim_{\lambda \to 0} \frac{f(x^* + \lambda(z - x^*)) - f(x^*)}{\lambda} &\leq \lim_{\lambda \to 0} \frac{\lambda f(z) + (1 - \lambda)f(x^*) - f(x^*)}{\lambda} \\ &= f(z) - f(x^*).\end{aligned} \tag{2.6}$$

Todistuksen alussa valittiin piste z niin, että $f(z) < f(x^*)$, joten yhtälön (2.6) perusteella

$$\nabla f(x^*)^T = f(z) - f(x^*) < 0. \tag{2.7}$$

Nyt $\nabla f(x^*) \neq 0$, joten piste x^* ei voi olla kriittinen piste. □

Konveksien funktioiden minimoiminen on siis helpompaa kuin ei-konveksien. Tästä syystä konveksisuutta pyritään hyödyntämään optimointialgoritmeja rakennettaessa, ja usein esimerkiksi kohdefunktiota mallinnetaan konvekseilla funktioilla iteraatioaskelten minimoinnin helpottamiseksi. [1]

2.2 Trust region -algoritmin rakenne

Sekä viivahaku- että luottamusaluealgoritmit pyrkivät ratkaisemaan minimointiongelman mallintamalla kohdefunktiota $f(x)$ kullakin iteraatioaskeleella k jollain yksinkertaisemmalla mallifunktiolla $m_k(x)$. Eri algoritmeissa lähestymistapa tähän mallifunktioon on kuitenkin erilainen. Siinä missä viivahakualgoritmit etsivät mallifunktiosta suunnan seuraavalle iteraatiolle, rajataan luottamusaluealgoritmeissa tietyn kokoinen alue senhetkisen iteraatiopisteen ympärille ja minimoidaan mallifunktio tämän alueen sisällä. Algoritmi onkin saanut nimensä siitä, että siinä luodaan mallifunktio, johon luotetaan vain tietyllä alueella Ω_k. Tätä aluetta kutsutaan luottamusalueeksi. Luottamusalueen sisältä saadun minimiarvon perusteella suoritetaan arviointi mallin osuvuudesta kohdefunktioon. Tämän arvioinnin perusteella valitaan seuraavan iteraation aloituspiste sekä luottamusalueen koko. [3, Luku 1]

Luottamusaluealgoritmissa siis kullekin iteraatioaskeleelle ratkaistaan optimointitehtävä

$$\min_{x \in \Omega_k} \quad m_k(x). \tag{2.8}$$

Yleisesti mallifunktiona käytetään Taylorin sarjan mukaista kvadraattista approksimaatiota. Merkitään seuraavaa iteraatioaskelta s_k ja approksimaatiota pisteen x_k ympärillä

$$m_k(s_k) = f_k + g_k^T s_k + \frac{1}{2} s_k^T B_k s_k. \tag{2.9}$$

Tässä f_k on funktion f arvo $f(x_k)$, g_k on funktion f gradientti $\nabla f(x_k)$ ja matriisi B_k on Hessen matriisi, eli funktion f kaikki toiset derivaatat sisältävä symmetrinen matriisi tai sen approksimaatio [19]. Näin ollen yhtälön (2.8) mukainen osaongelma saadaan muotoon

$$\min_{\|s_k\|_2 \leq \Delta_k} \quad f_k + g_k^T s_k + \frac{1}{2} s_k^T B_k s_k. \tag{2.10}$$

Yhtälössä (2.10) luottamusalue on määritelty euklidisen normin $\|\cdot\|_2$ avulla niin,

että luottamusalueena on x_k-keskinen n-ulotteinen pallo, jonka säde on Δ_k. Luottamusalue voidaan määritellä myös muiden normien avulla [14, Luku 4.5], mutta tässä työssä pitäydytään euklidisen normin käytössä ja jatkossa merkitäänkin $\|\cdot\|_2 = \|\cdot\|$. Yhtälöä (2.10) kutsutaan luottamusalueen osaongelmaksi ja sen ratkaisu on laskennallisesti ja suorituskyvyllisesti vaativin osuus luottamusaluealgoritmin toiminnassa. Luvussa 2.3 esitellään tapoja osaongelman ratkaisemiseksi.

Tärkeä osa luottamusaluealgoritmin toimintaa on saadun osaongelman ratkaisun s_k arviointi. Arvioimalla saadaan tietoa siitä, kuinka hyvin malli kuvaa kohdefunktiota, ja tämän tiedon perusteella voidaan tehdä muutoksia luottamusalueen kokoon. Mallin kuvatessa kohdefunktiota hyvin voidaan aluetta suurentaa ja vastaavasti päinvastaisessa tapauksessa pienentää. Suurempi luottamusalue antaa mahdollisuuden ottaa pidempiä iteraatioaskelia, mikä nopeuttaa algoritmin suppenemista. [3]

Määritellään arviointia varten funktion ja mallifunktion arvoja hyödytävä suhde ρ_k, jota käytetään yleisesti osaongelman ratkaisun arvioinnissa.

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{m_k(0) - m_k(s_k)} = \frac{\text{todellinen vähenemä}}{\text{ennustettu vähenemä}} \tag{2.11}$$

Algoritmille annetaan parametrejä, ja ajon aikana suhteen ρ_k arvon perusteella päätetään seuraavan iteraatioaskeleen luottamusalueesta ja lähtöpisteestä [14, Luku 4].

Havainnollistetaan algoritmin toimintaa Algoritmin 1 esimerkkialgoritmilla. Algoritmi on asetettu jatkamaan, kunnes kohdefunktion gradientin normi $\|g_k\| < \epsilon$, jollakin valitulla arvolla $\epsilon > 0$. Lisäksi on valittu luottamusalueen sädettä säätävien parametrien $\eta_i, i = 1, 2, \ldots$ arvoiksi esimerkki yleisesti käytössä olevista arvoista. [2, 14] Muokkaamalla parametrien arvoja voidaan vaikuttaa sen nopeuteen tai saada erilaisia suppenemistuloksia [6].

2.3 Osaongelman ratkaiseminen

Luottamusalueen osaongelman (2.10) ratkaisemiseksi on olemassa useita erilaisia lähtökohtia. Ongelman ratkaiseminen eksaktisti vaatii paljon laskentatehoa esimerkiksi suoritettavien matriisihajotelmien takia ja tästä syystä monissa luottamusaluealgoritmeissa osaongelma ratkaistaankin approksimoiden. Approksimoitu ratkaiu vähentää kohdefunktion arvoa vähemmän kerralla kuin eksakti, mutta samoja suppenemistuloksia voidaan johtaa sekä approksimoidulle että eksaktille ratkaisulle. Yleisesti käytetty esimerkki tällaisesta approksimoidusta ratkaisusta, joka on riittävä globaalia suppenemista varten on Cauchyn piste s_k^C. [14, Luku 4.1]

Algoritmi 1 Esimerkki luottamusaluealgoritmista

Olkoon $\eta_1 = 0{,}001$ $\eta_2 = 0{,}25$ $\eta_3 = 0{,}75$ $\eta_4 = 0{,}25$ $\eta_5 = 2$ ja $\epsilon = 0{,}001$
Asetetaan $x_0 \in \mathbb{R}^n$, $\Delta_0 = 1$ ja $\Delta_{\max} = 4$
Ratkaistaan kohdefunktion f gradientti $\nabla f(x_0)$ ja matriisi B_0
while $\|g_k\| < \epsilon$ **do**
  Ratkaistaan s_k osaongelmasta (2.10)
  Ratkaistaan ρ_k yhtälön (2.11) tapaan
  if $\rho_k < \eta_2$ **then**
    $\Delta_{k+1} = \eta_4 \Delta_k$ ▷ Pienennetään luottamusaluetta
  else
    if $\rho_k > \eta_3$ ja $\|s_k\| = \Delta_k$ **then**
      $\Delta_{k+1} = \min\{\eta_5 \Delta_k, \Delta_{max}\}$ ▷ Suurennetaan luottamusaluetta
    else
      $\Delta_{k+1} = \Delta_k$
    end if
  end if
  if $\rho > \eta_1$ **then**
    $x_{k+1} = x_k + s_k$ ▷ Hyväksytään ratkaisu uudeksi iteraatiopisteeksi
  else
    $x_{k+1} = x_k$ ▷ Hylätään saatu ratkaisu
  end if
  Ratkaistaan $\nabla f(x_{k+1})$ ja B_{k+1}
end while

Määritelmä 2.3.1. Cauchyn piste s_k^C on minimi kohdefunktion negatiivisen gradientin $-g_k$ suunnassa.

$$s_k^C = -\tau_k \Delta_k \frac{g_k}{\|g_k\|} \quad , \tau_k = \begin{cases} 1 & \text{jos } g_k^T B_k g_k \leq 0 \\ \min\{\|g_k\|^3/(\Delta_k g_k^T B_k g_k), 1\} & \text{muulloin} \end{cases} \tag{2.12}$$

Cauchyn piste saavutetaan siis kulkemalla negatiivisen gradientin suuntaisesti. Mikäli matriisi B_k ei ole positiivisesti definiitti ($a^T B_k a > 0$ kaikilla vektoreilla $a \neq 0$, joille tulo on määritelty), mallifunktio vähenee monotonisesti ja näin ollen osaongelman ratkaisu saavutetaan luottamusalueen rajalla. Matriisin B_k ollessa positiivisesti definiitti, on mallifunktiolla minimiarvo. Tällöin τ_k saadaan ratkaistua derivaatan nollakohdassa ja arvo valitaan sen mukaan, onko minimi luottamusalueen sisällä vai ei. [14]

Tässä alaluvussa käsitellään approksimoivaa dog-leg-menetelmää sekä sivutaan eksaktia ratkaisua hakevaa menetelmää. Molemmat tuottavat paremman ratkaisun kuin s_k^C ja ovat siten suppenevia globaalisti [19].

2.3.1 Dog-leg-menetelmä

Dog-leg-menetelmä pyrkii parantamaan Cauchyn pisteen tarjoamaa ratkaisua. Cauchyn pisteen laskemisessa jää kokonaan mallifunktion Hessen matriisin tarjoama kaarevuusinformaatio hyödyntämättä, kun minimoimissuuntana käytetään negatiivisen gradientin suuntaa. Jos matriisi B_k on positiivisesti definiitti, saadaan m_k minimoitua laskemalla Newtonin menetelmän mukainen askel s_k^B [7, s.72]

$$s_k^B = -B_k^{-1} g_k. \tag{2.13}$$

Jos askel s_k^B on luottamusalueen sisällä, on s_k^B osaongelman ratkaisu. Luottamusalueen ollessa pieni, ei kaarevuudella ehdi olla kovin suurta vaikutusta mallifunktion käyttäytymiseen ja näin ollen negatiivista gradienttia pitkin saatu ratkaisu on osuva. Dog-leg-menetelmällä yhdistetään nämä kaksi eri suuntaa ja osaongelman ratkaisu saadaan näiden lineaarikombinaationa. [4]

Negatiivisen gradientin suuntainen vektori $s_k^U = -\tau_k g$ saadaan ratkaisemalla τ_k mallifunktion derivaatan nollakohdasta. Sijoitetaan s_k^U yhtälöön (2.9)

$$\begin{aligned} m(\tau) &= f + g^T(-\tau g) + \frac{1}{2}(-\tau g)^T B(-\tau g) \\ &= f - \tau g^T g + \frac{\tau^2}{2} g^T B g. \end{aligned} \tag{2.14}$$

Derivoidaan (2.14) vakion τ suhteen ja asetetaan derivaatta nollaksi. Vakio τ voidaan ratkaista

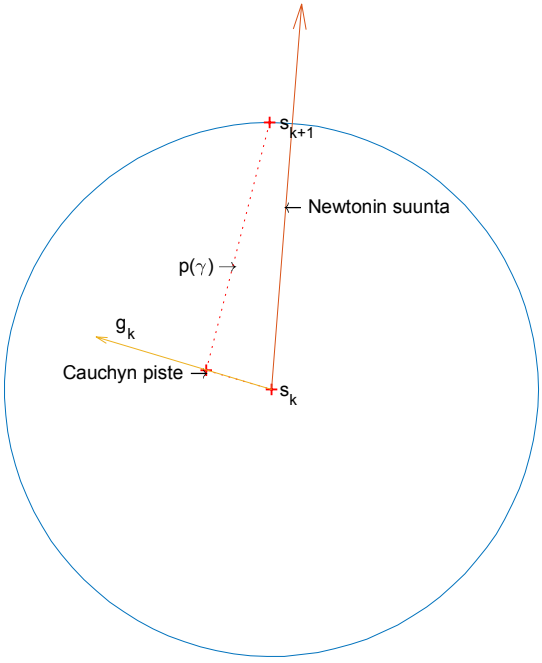
$$m\prime(\tau) = -g^T g + \tau g^T B g \implies \tau = \frac{g^T g}{g^T B g} \tag{2.15}$$

Dog-leg -menetelmän lineaarikombinaatiossa käytettävät vektorit ovat siis $s_k^U = -\frac{g_k^t g_k}{g_k^t B_k g_k} g_k$ ja $s_k^B = -B_k g_k$. Määritellään vakion γ avulla dog-leg -menetelmässä kuljettu polku $\tilde{p}(\gamma)$ [3, Luku 7.5.3]. Polkua $\tilde{p}(\gamma)$ on kuvattu myös kuvassa 2.1.

$$\tilde{p}(\gamma) = \begin{cases} \gamma s_k^U & 0 \leq \gamma \leq 1 \\ s_k^U + (\gamma - 1)(s_k^B - s_k^U) & 1 \leq \gamma \leq 2 \end{cases} \tag{2.16}$$

Polun normi $\|\tilde{p}(\gamma)\|$ on kasvava muuttujan γ suhteen ja vastaavasti mallifunktio on vähenevä polkua pitkin [14, Luku 4.1]. Näin ollen ratkaisuksi s_k voidaan valita s_k^B, jos $\|s_k^B\| \leq \Delta_k$, ja Cauchyn piste s_k^C, jos $\|s_k^U\| \geq \Delta_k$. Muissa tapauksissa ratkaisu löytyy polulta $\tilde{p}(\gamma)$ luottamusalueen rajalla. Tällöin ratkaisu s_k saadaan ratkaisemalla γ yhtälöstä

$$\|s_k^U + (\gamma - 1)(s_k^B - s_k^U)\|^2 = \Delta_k^2. \tag{2.17}$$



Kuva 2.1 *Dog-leg-askeleen ratkaisemiseen tarvittavat vektorit sekä niiden avulla ratkaistu polku* $\tilde{p}(\gamma)$

Kuvaan 2.1 on piirretty luottamusalue, sekä dog-leg-askeleen ratkaisemisessa tarvittavat vektorit. Kuvan tilanteessa gradientin suunnasta ratkaistu Cauchyn piste s_k^C on alueen sisäpuolella, joten voidaan päästä parempaan iteraatioon käyttämällä myös Newtonin suuntaa s_k^B. Uusi iteraatio s_{k+1} löytyy siis vektorin $s_k^B - s_k^U$ kohdatessa luottamusalueen rajan.

Dog-leg-menetelmä on tehokas laskutehon suhteen, sillä siinä joudutaan tekemään matriisihajotelmia ainoastaan inverssin B_k^{-1} laskemisessa. Ongelma menetelmässä on se, että se vaatii matriisin B_k olevan positiivisesti definiitti. Tämä ehto saadaan poistumaan käyttämällä esimerkiksi kaksiulotteisen aliavaruuden minimointimenetelmää [16].

2.3.2 Eksakti ratkaisu

Osaongelman ratkaisemiseksi eksaktisti tarvitaan dog-leg-menetelmää enemmän välivaiheita ja monimutkaisempi laskualgoritmi. Hakemalla eksaktia ratkaisua saadaan kuitenkin kvadraattisen mallifunktion tarjoama informaatio kaarevuudesta paremmin hyödynnettyä. [11] Lähtökohtana ratkaisuun on Newtonin menetelmän kaltaisen askeleen (2.13) hakeminen muokkaamalla matriisia B_k niin, että matriisi on positiivisesti definiitti ja sillä on minimi luottamusalueella. Gay [5] ja Sorensen [17] ovat

johtaneet lauseen, jonka avulla voidaan etsiä osaongelman optimaalista ratkaisua.

Lause 2.3.1. *Askel s^* on osaongelman 2.10 globaali ratkaisu, jos ja vain jos s^* on käypä ja on olemassa luku $\mu \geq 0$, joka toteuttaa ehdot*

$$(B + \mu I)s^* = -g \tag{2.18a}$$

$$\mu(\Delta - \|s^*\|) = 0 \tag{2.18b}$$

$$(B + \mu I) \geq 0, \tag{2.18c}$$

missä merkintä $(B + \mu I) \geq 0$ tarkoittaa, että matriisi on positiivisesti semidefiniitti ($z^T M z \geq 0$ kaikilla vektoreilla z)

Todistus. [14, Luku 4.3]Oletetaan ensin, että luku $\mu \geq 0$ toteuttaa ehdon (2.18). Tällöin s^* on kvadraattisen funktion

$$\hat{m}(s) = g^T s + \frac{1}{2} s^T (B + \mu I)s = m(s) + \frac{\mu}{2} s^T s \tag{2.19}$$

globaali minimi, kun $m(s)$ on muotoa (2.9) oleva mallifunktio [14, Lemma 4.7]. Koska $\hat{m}(s) \geq \hat{m}(s^*)$, pätee myös

$$\begin{aligned} m(s) + \frac{\mu}{2} s^T s &\geq m(s^*) + \frac{\mu}{2}(s^*)^T s^* \\ m(s) &\geq m(s^*) + \frac{\mu}{2}((s^*)^T s^* - s^T s). \end{aligned} \tag{2.20}$$

Koska $\mu(\Delta - \|s^*\|) = 0$, saadaan normin ominaisuuksien avulla $\mu(\Delta^2 - (s^*)^T s^*)$ ja voidaan sijoittaa edelliseen

$$m(s) \geq m(s^*) + \frac{\mu}{2}(\Delta^2 - (s^*)^T s^*). \tag{2.21}$$

Näin ollen, kun luku $\mu \geq 0$, pätee $m(s) \geq m(s^*)$ kaikilla sellaisilla askelilla s, joiden normi $\|s\| \leq \Delta$. Näin ollen s^* on globaali minimi.

Vastakkaisesti oletaan, että s^* on globaali minimi ja osoitetaan, että on olemassa luku $\mu \geq 0$, joka täyttää ehdon (2.18). Tutkitaan tapauksia, joissa $\|s^*\| < \Delta$ ja $\|s^*\| = \Delta$. Kun $\|s^*\| < \Delta$, on s^* mallifunktion $m(s)$ minimi luottamusalueen sisällä ja näin ollen

$$\nabla m(s^*) = Bs^* + g = 0 \quad \text{, kun } B \geq 0. \tag{2.22}$$

Ehto 2.18 siis pätee, kun $\mu = 0$.

Kun taas $\|s^*\| = \Delta$, täyttyy ehto (2.18b) välittömästi. Tällöin s^* on funktion $m(s)$ minimi luottamusalueen rajalla. Tällaista rajoitettua minimointiongelmaa voidaan

käsitellä Lagrangen menetelmällä. Voidaan siis löytää luku μ, jonka arvolla Lagrangen funktiolla

$$\mathcal{L}(s,\mu) = m(s) + \frac{\mu}{2}(s^T s - \Delta^2) \tag{2.23}$$

on kriittinen piste pisteessä s^*. Lasketaan Lagrangen funktion osittaisderivaatta askeleen s suhteen ja kriittisessä pisteessä s^* tämä on nolla.

$$\frac{\partial \mathcal{L}}{\partial s}(s^*) = g + Bs^* + \mu s^* = 0 \tag{2.24}$$

Tästä seuraa, että $(B + \mu I)s^* = -g$, mikä toteuttaa ehdon (2.18a).

Nyt $m(s) \geq m(s^*)$ kaikkialla luottamusalueen rajalla, eli $s^T s = (s^*)^T s^* = \Delta^2$. Tällaisille vektoreille s pätee

$$m(s) \geq m(s^*) + \frac{\mu}{2}((s^*)^T s^* - s^T s) \tag{2.25}$$

ja jos määritellään gradientti ehdon (2.18a) mukaisesti, saadaan järjestelemällä termit

$$\frac{1}{2}(s - s^*)^T (B + \mu I)(s - s^*) \geq 0. \tag{2.26}$$

Tämä osoittaa, että matriisi $(B + \mu I)$ on positiivisesti semidefiniitti.

Todistetaan vielä, että luku $\mu \geq 0$. Oletetaan ehdot (2.18a,c) päteviksi pisteessä s^*, joka minimoi yhtälön (2.19) mukaisen kvadraattisen funktion. Tällöin yhtälö (2.20) pätee. Tehdään vastaoletus, että vain arvot $\mu < 0$ toteuttavat ehdot (2.18a,c).

Nyt yhtälön (2.20) mukaan $m(s) \geq m(s^*)$, kun $\|s\| \geq \|s^*\| = \Delta$. Koska s^* minimoi funktion $m(s)$ luottamusalueella, seuraa nyt edellisestä huomiosta, että s^* on globaali ja rajoittamaton minimi. Täten $Bp = -g$ ja B on positiivisesti semidefiniitti [14, Lemma 4.7]. Seuraa ristiriita, koska ehdot (2.18a,c) toteutuvat arvolla $\mu = 0$. Näin ollen on oltava $\mu \geq 0$. □

Lauseen 2.3.1 mukaisen ratkaisun löytäminen onnistuu etsimällä oikea luku μ, joka toteuttaa lauseen ehdot. Valtaosassa tapauksista luku μ saadaan ratkaisuna yhtälöstä

$$\frac{1}{\|(B + \mu I)^{-1} g\|} - \frac{1}{\Delta} = 0. \tag{2.27}$$

Hankaluuksia aiheuttaa tilanne, jossa ratkaistu $\mu = -\sigma(B)$ luvun $\sigma(B)$ ollessa matriisin B pienin ominaisarvo. Jos $\sigma(B) = 0$, voidaan osaongelma ratkaista pseudoinverssin avulla, jolloin $s_k = -B_k^+ g$. Mikäli $\mu = -\sigma(B) > 0$, voidaan käyttää erilaisia tekniikoita varotoimenpiteinä, jotta saadaan rajattua arvoja käyviksi. [19]

2.4 Algoritmin suppeneminen

Luottamusaluealgoritmin rakentamisessa on tavoitteena saada iteraatioaskelten s_k arvo suppenemaan kohti kohdefunktion globaalia minimiä. Tällöin myös jokaisella iteraatiokerralla lasketun gradientin g_k arvo lähestyy nollaa. Luvussa 2.3 on käsitelty kolmea eri tapaa ratkaista uusi iteraatioaskel s_k osaongelmasta (2.10) ja näistä kaikkein pienimmän vähenemän tuottaa Cauchyn piste s_k^C. Cauchyn pisteessä saadulle ratkaisulle pätee epäyhtälö

$$m_k(0) - m_k(s_k^C) \geq c_1 \|g_k\| \min\left(\Delta_k, \frac{\|g_k\|}{\|B_k\|}\right), \tag{2.28}$$

missä vakion $c_1 \in [0, 1]$ arvo on $c_1 = \frac{1}{2}$. [14, Luku 4.2] Koska dog-leg -menetelmällä sekä eksaktia ratkaisua hakevalla algoritmillä saadut ratkaisut ovat parempia tai vähintäänkin samoja kuin s_k^C, pätee epäyhtälö (2.28) myös niillä saaduille ratkaisuille. Ehdolla (2.28) on tärkeä rooli suppenemisen tutkimisessa, sillä sen täyttäville ratkaisuille voidaan todistaa globaali suppeneminen. [17]

Määritellään seuraavaa lausetta varten tasojoukko $S := \{x | f(x) \leq f(x_0)\}$ sekä avoin ympäristö tässä joukossa $S(R_0) = \{x | \|x - y\| < R_0, \text{ jollain } y \in S\}$. Yleisemmän tarkastelun vuoksi muokataan myös askeleen s_k pituutta koskevaa rajoitusta sellaiseksi, että jollain vakion $\theta \geq 1$ arvolla $\|s_k\| \leq \theta\Delta$.

Lause 2.4.1. *Olkoon Algoritmin 1 parametri $\eta_1 > 0$. Oletetaan, että matriisi B_k on rajoitettu johonkin vakioon β, jolloin $\|B_k\| \leq \beta$. Oletetaan myös, että kohdefunktio f on alhaalta rajoitettu tasojoukossa S sekä Lipschitz-jatkuvasti derivoituva jossain ympäristössä $S(R_0), R_0 > 0$. Jos kaikki algoritmin tuottamat osaongelman ratkaisut s_k toteuttavat ehdon (2.28) sekä ehdon $\|s_k\| \leq \theta\Delta$, niin*

$$\lim_{k \to \infty} g_k = 0. \tag{2.29}$$

Todistus. [14] Käsitellään iteraatiota indeksillä m, jonka gradientti $g_m \neq 0$. Koska gradientti g on Lipschitz-jatkuva, merkitään Lipschitzin vakiota β_1, ja voidaan esittää

$$\|g(x) - g_m\| \leq \beta_1 \|x - x_m\| \tag{2.30}$$

kaikissa ympäristön $S(R_0)$ pisteissä $x \in S(R_0)$. Määritellään luvut ϵ ja R sellaisiksi, että

$$\epsilon = \frac{1}{2}\|g_k\| \qquad R = \min\left(\frac{\epsilon}{\beta_1}, R_0\right)$$

Pallo $\mathcal{B}(x_m, R) = \{x | \|x - x_m\| < R\}$ sisältyy kokonaan alueeseen $S(R_0)$, joten gradientin Lipschitz-jatkuvuus pätee myös pallon sisällä. Näin ollen kaikille pallon

sisällä oleville arvoille $x \in \mathcal{B}(x_m, R)$ pätee

$$\|g(x)\| \geq \|g_m\| - \|g(x) - g_m\| \geq \frac{1}{2}\|g_m\| = \epsilon. \tag{2.31}$$

Jos koko jono $\{x_k\}, k \geq m$ kuuluisi palloon $\mathcal{B}(x_m, R)$, pätisi $\|g_k\| \geq \epsilon > 0$. Näin ei kuitenkaan ole, vaan jono poistuu pallosta $\mathcal{B}(x_m, R)$ jollain indeksillä $k \geq m$ [14, Theorem 4.5 todistus]

Olkoon indeksi $l \geq m$ sellainen, että askel x_{l+1} on ensimmäinen pallon $\mathcal{B}(x_m, B)$ ulkopuolella oleva askel. Silloin $\|g_k\| \geq \epsilon$ indekseillä $m, m+1, \ldots, l$ ja ehdon (2.28) perusteella voidaan kokonaisvähenemä kirjoittaa kunkin askeleen vähenemien summana

$$\begin{aligned} f(x_m) - f(x_{l+1}) &= \sum_{k=m}^{l} f(x_k) - f(x_{k+1}) \\ &\geq \sum_{k=m, x_k \neq x_{k+1}}^{l} \eta_1 [m(0) - m(s_k)] \\ &\geq \sum_{k=m, x_k \neq x_{k+1}}^{l} \eta c_1 \epsilon \min\left(\Delta_k, \frac{\epsilon}{\beta}\right). \end{aligned} \tag{2.32}$$

Summassa on otettu huomioon vain ne indeksit, joilla $x_k \neq x_{k+1}$, eli vain hyväksytyt askeleet on laskettu mukaan. Jos $\Delta_k \leq \frac{\epsilon}{\beta}$ kaikilla indekseillä $m, m+1, \ldots, l$, saadaan (2.32) muotoon

$$f(x_m) - f(x_{l+1}) \geq \eta_1 c_1 \epsilon \sum_{k=m, x_k \neq x_{k+1}}^{l} \Delta_k \geq \eta_1 c_1 \epsilon R = \eta_1 c_1 \epsilon \min\left(\frac{\epsilon}{\beta_1}, R_0\right). \tag{2.33}$$

Jos taas $\Delta_k > \frac{\epsilon}{\beta}$ jollain indeksillä $m, m+1, \ldots, l$, pätee

$$f(x_m) - f(x_{l+1}) \geq \eta_1 c_1 \epsilon \frac{\epsilon}{\beta}. \tag{2.34}$$

Koska jono $\{f(x_k)\}_{k=0}^{\infty}$ on vähenevä ja rajoitettu, lähestyy se jotain arvoa f^*

$$f(x_k) \to f^*. \tag{2.35}$$

Näin voidaan (2.33) ja (2.34) perusteella voidaan esittää

$$\begin{aligned} f(x_m) - f^* &\geq f(x_m) - f(x_{l+1}) \\ &\geq \eta_1 c_1 \epsilon \min\left(\frac{\epsilon}{\beta}, \frac{\epsilon}{\beta_1}, R_0\right) \\ &= \frac{1}{2}\eta_1 \|g_m\| \min\left(\frac{\|g_m\|}{2\beta}, \frac{\|g_m\|}{2\beta_1}, R_0\right) > 0. \end{aligned} \tag{2.36}$$

Koska $f(x_m) \to f^*$, pitää $f(x_m) - f^* \to 0$. Näin on oltava $g_m \to 0$. Tämä todistaa lauseen. $\square$

Tässä tapauksessa vaadittiin, että Algoritmin 1 parametri $\eta_1 > 0$, jolloin jokaiselta uudelta askeleelta s_k vaaditaan, että kohdefunktio $f(x)$ vähenee. Algoritmissä 1 voidaan asettaa myös $\eta = 0$, jolloin vähenemistä ei vaadita. Tällöin kuitenkin suppenemistulos on heikompi ja muotoa [6].

$$\liminf_{k\to\infty}\{g_k\} = 0. \tag{2.37}$$

Luottamusaluealgoritmi joka tapauksessa suppenee globaalisti, jos osaongelman ratkaisut vähenevät riittävästi.

3. ALGORITMIN KÄYTÄNNÖN TOIMINTA

Algoritmin käytännön toimintaa havainnollistettiin toteuttamalla Algoritmin 1 mukainen luottamusaluealgoritmi MATLAB-ohjelmistolla. Toteutetun algoritmin toimintaa testattiin ratkaisemalla sen avulla optimointitehtävä ja keräämällä tietoja ratkaisun vaiheista. Luvussa 3.1 esitellään tehtävä ja algoritmi ja luvussa 3.2 käsitellään algoritmin toimintaa ratkaisun aikana.

3.1 Testausongelman esittely

Optimointialgoritmien testaukseen käytetään yleisesti funktioita, joiden tiedetään käyttäytyvän erityisen haastavalla tavalla optimoidessa. Testausfunktioilla voi olla esimerkiksi useita lokaaleja minimejä tai satulapisteitä, joihin algoritmi voi harhautua globaalin minimin sijaan. [8] Tässä työssä testausfunktiona käytettiin Rosenbrockin funktiota

$$f : \mathbb{R}^2 \to \mathbb{R} \quad f(x, y) = (1 - x)^2 + 100(y - x^2)^2 \tag{3.1}$$

joka kuuluu yleisimpiin optimointialgoritmien testauksessa käytettyihin funktioihin. Funktion kuvaaja kolmiulotteisena on esitetty kuvassa 3.1 ja sen globaali minimi on pisteessä $x^* = [1, 1]^T$ ja $f(x^*) = 0$. Funktiosta tekee käytetyn testiongelman se, että sen minimi on kapeassa ja pitkässä parabolisessa laaksossa, mikä saattaa harhauttaa algoritmin suppenemaan väärässä kohdassa laaksoa [15]. Alkuarvauksena käytettiin pistettä $x_0 = [2{,}15;\ 0{,}88]^T$, joka on mielivaltaisesti valittu piste selkeästi laakson ulkopuolelta.

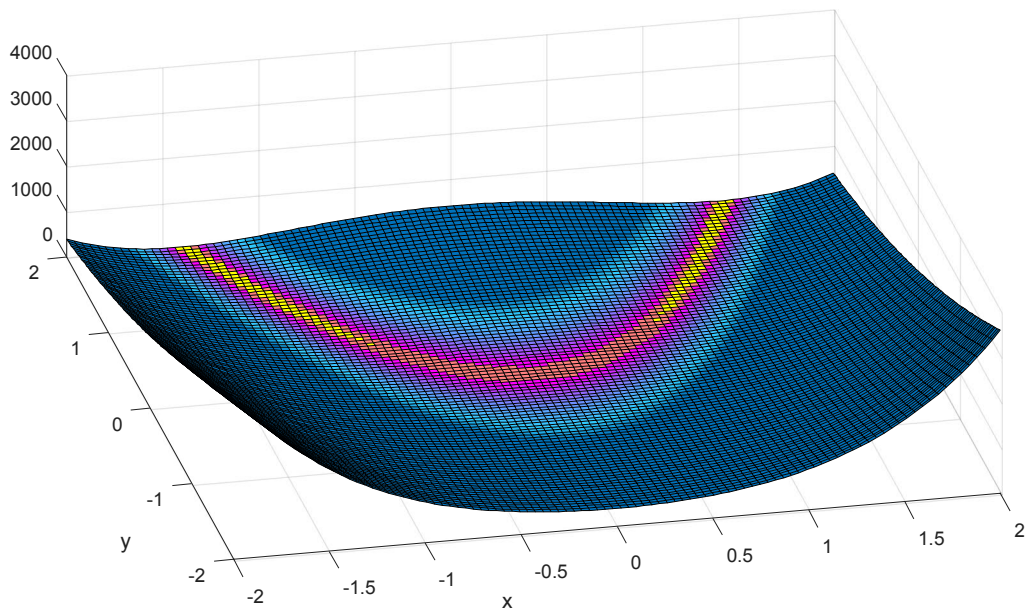
Testauksessa käytetty algoritmi toteutettiin käyttämällä osaongelman ratkaisussa luvussa 2.3.1 esiteltyä dog-leg -menetelmää. Algoritmissa käytettyjen parametrien η_i ja ϵ arvot valittiin samoin kuin Algoritmissa 1 ja luottamusalueen säteen Δ_0 arvoksi valittiin $\Delta_0 = 1$. Algoritmin nopeampaa suorittamista varten laskettiin lisäksi funktion gradientti $\nabla f(x, y)$ ja Hessen matriisi $\nabla^2 f(x, y)$

$$\nabla f(x, y) = \begin{bmatrix} 2x - 400x(y - x^2) - 2 \\ 200y - 200x^2 \end{bmatrix} \quad \text{ja } \nabla^2 f(x, y) = \begin{bmatrix} 1200x^2 - 400y + 2 & -400x \\ -400x & 200 \end{bmatrix}$$

Gradientin ja Hessen matriisin avulla voidaan vielä perustella, että funktion globaali minimi $x^* = [1, 1]^T$. Gradientin nollakohta saadaan selvitettyä ratkaisemalla ensin alempi yhtälö ja sijoittamalla tämä ylempään.

$$\begin{aligned} 200y - 200x^2 = 0 &\iff y = x^2 \\ 2x - 400x(y - x^2) - 2 = 2x - 2 = 0 &\iff x = 1 \Rightarrow y = 1 \end{aligned} \tag{3.2}$$

Tässä pisteessä laskettu Hessen matriisi $\begin{bmatrix} 802 & -400 \\ -400 & 200 \end{bmatrix} > 0$, koska kaikki sen ominaisarvot ovat positiivisia. Näin ollen kyseessä on globaali minimi.



Kuva 3.1 *Rosenbrockin funktio esitettynä kolmiulotteisesti*

Toteutetun algoritmin MATLAB-koodi on esitetty liitteessä 1. Algoritmi laskee funktion, gradientin ja Hessen matriisin arvot kussakin pisteessä kahvafunktioiden avulla, ja dog-leg -askeleen laskemisessa käytettävä yhtälö (2.17) on purettu osiin ja ratkaistu toisen asteen yhtälön ratkaisukaavalla.

3.2 Algoritmin toiminta suorituksen aikana

Algoritmi suoritti optimoinnin käyttäen 15 iteraatiota, jonka jälkeen se lopetti toimintansa, koska $\|g_k\| < 0{,}001$. Tällöin algoritmin ratkaisema globaali minimipiste

x^*_{tr} ja funktion arvo $f(x^*_{tr})$ olivat

$$x^*_{tr} = \begin{bmatrix} 1{,}00000005946 \\ 1{,}00000011012 \end{bmatrix} \quad f(x^*_{tr}) = 1{,}19402 \times 10^{-14},$$

ja aikaa oli kulunut 0,123 sekuntia. Algoritmi suppenee samaan ratkaisuun $x^* = (1, 1)$ mistä tahansa alkuarvauksesta.

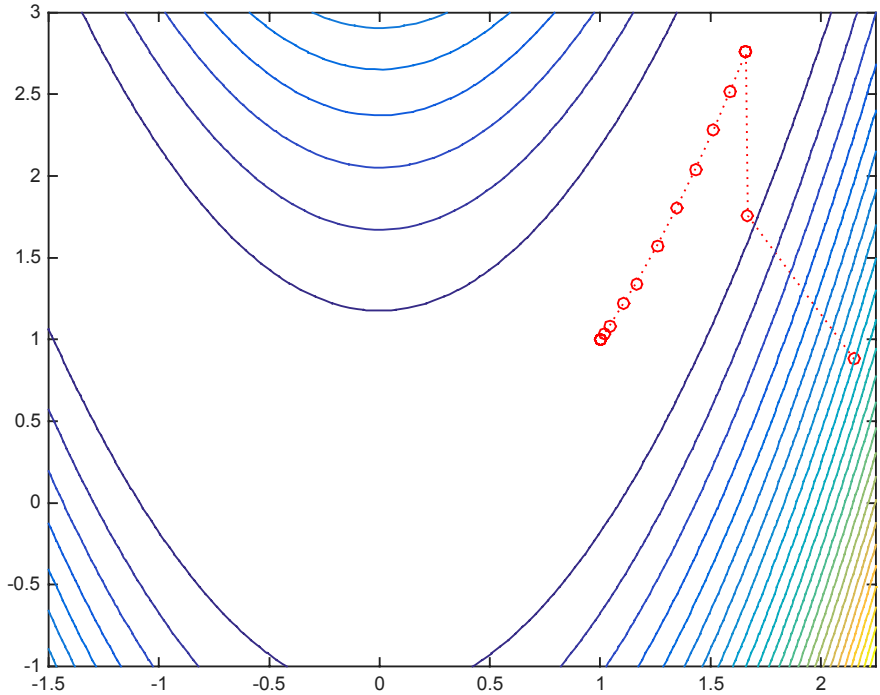
Iteraatioprosessin kulkua on kuvattu taulukossa 3.1 sekä kuvassa 3.2. Taulukossa 3.1 on esitetty luottamusalueen säde Δ_k, funktion arvo $f(x_k)$ sekä etäisyys ratkaisusta $\|x_k - x^*\|$ viidelle ensimmäiselle iteraatioaskeleelle sekä siitä eteenpäin joka toiselle askeleelle. Kuvaan 3.2 on piirretty kohdefunktio kaksiulotteisten tasa-arvokäyrien avulla sekä iteraatiopisteiden muodostama polku, jota pitkin algoritmi saavutti ratkaisun.

Taulukko 3.1 *Luottamusalueen säde Δ_k, kohdefunktion arvo $f(x_k)$ ja iteraatiopisteen etäisyys todellisesta minimistä $\|x_k - x^*\|$ eri iteraatioaskelilla*

k	Δ_k	$f(x_k)$	$\|x_k - x^*\|$
0	1	$1{,}420\times10^{3}$	1,1562
1	1	105,809	1,0091
2	1	0,4373	1,8765
3	0,25	0,8254	1,8765
4	0,25	0,3469	1,6270
6	0,25	0,1886	1,1280
8	0,25	0,0718	0,6286
10	0,5	0,0120	0,2397
12	2	$3{,}6787\times10^{-4}$	0,0408
14	4	$1{,}0139\times10^{-8}$	$2{,}1132\times10^{-4}$
15	4	$1{,}194\times10^{-14}$	$1{,}2524\times10^{-7}$

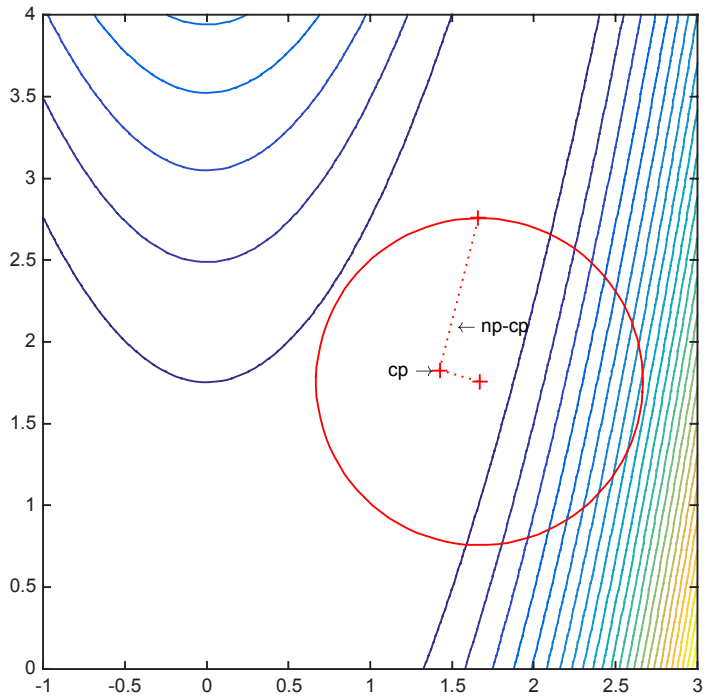
Tarkastelemalla taulukkoa 3.1 ja kuvaa 3.2 tarkemmin, voidaan todeta, että algoritmi saavutti kohdefunktion laakson helposti, mutta sen jälkeen laskettua askelta ei hyväksytty uudeksi iteraatioksi, sillä funktion arvo kasvoi eikä laskenut. Luottamusaluetta pienentämällä löydettiin kuitenkin käypä askel ja sen jälkeen algoritmi saavutti minimipisteen ympäristön suoraviivaisesti. Minimipisteen ympäristössä algoritmi iteroi luottamusaluettaan kasvattaen, kunnes $\|g_k\| < 0{,}001$ ja algoritmi hyväksyi ratkaisun.

Käsitellään vielä algoritmin tapaa muodostaa dog-leg -ratkaisu käytännön tilanteessa. Kuvassa 3.3 on esitetty iteraatioaskeleella $k = 2$ laskettu Cauchyn piste (cp) sekä Newtonin pisteen ja Cauchyn pisteen välinen jana (np-cp). Koska Cauchyn piste $[1{,}4235;\ 1{,}8294]^T$ on luottamusalueen sisällä, ei sitä hyväksytty ratkaisuksi, vaan



Kuva 3.2 *Iteraatioaskeleet kuvattuna kohdefunktion kaksiulotteisia tasa-arvokäyriä vasten*

ratkaisua parannettiin liikkumalla Newtonin pisteen suuntaan, kunnes luottamusalueen raja tuli vastaan. Näin ratkaisuksi saatiin $x_2 = [1{,}6609;\ 2{,}7562]^T$.



Kuva 3.3 *Algoritmin muodostama dog-leg -askel, kun* $k = 2$

Vertailun vuoksi samaa ongelmaa yritettiin ratkaista myös suhteellisen yksinkertaisella viivahakuun perustuvalla algoritmilla. Algoritmissa viivahaun askeleen pituuden määrittämiseen käytettiin Armijo-ehtoa. Algoritmin MATLAB-koodi on suoraan lähteestä [2, s.27]. Myös viivahakualgoritmi suppeni aina oikeaan ratkaisuun, mutta se käytti ratkaisun löytämiseksi todella paljon enemmän iteraatioita. Taulukkoon 3.2 on kerätty tietoja eri alkuarvauksilla suoritettujen molempien algoritmien suorituksista.

Taulukko 3.2 *Luottamusalue- ja viivahakualgoritmeilla ratkaistujen optimointitehtävien alkuarvaukset x_0, ratkaisujen etäisyydet globaalista minimistä $\|x^*_{tr} - x^*\|$, käytettyjen iteraatioiden määrä n_{iter} sekä algoritmin suorituksessa kulunut aika t sekunteina.*

	Luottamusaluealgoritmi			Viivahakualgoritmi		
x_0	$\|x^*_{tr} - x^*\|$	n_{iter}	$t(s)$	$\|x^*_{tr} - x^*\|$	n_{iter}	$t(s)$
$[2{,}15;\ 0{,}88]^T$	$1{,}2524 \times 10^{-7}$	15	0,123	0,00154	5535	0,135
$[-0{,}75;\ 0{,}25]^T$	$7{,}3828 \times 10^{-5}$	22	0,02	$1{,}613 \times 10^{-5}$	13441	0,292
$[-1{,}80;\ 3{,}20]^T$	$1{,}054 \times 10^{-5}$	28	0,024	0,00154	7033	0,165
$[-1{,}10;\ -2{,}00]^T$	$6{,}0283 \times 10^{-6}$	23	0,022	0,000191	5161	0,124
$[-0{,}10;\ 1{,}25]^T$	$6{,}66 \times 10^{-8}$	24	0,033	0,000128	9009	0,176
$[3{,}33;\ 3{,}33]^T$	$2{,}5445 \times 10^{-6}$	19	0,012	0,00154	7807	0,160

Tarkastelemalla taulukon tuloksia huomataan, että vaikka iteraatioiden määrässä oli suuri ero, on ero algoritmien suoritusajoissa huomattavasti pienempi. Syy sille, että viivahaussa yhden iteraation suoritusaika on noin paljon pienempi on se, että viivahakualgoritmi ei käytä matriisihajotelmia, vaan seuraavan askeleen löytäminen perustuu pelkästään gradientin tutkimiseen. Hessen matriisin tuoman kaarevuusinformaation hyödyntäminen on samalla yksi iso syy sille, miksi luottamusaluealgoritmin laskema dog-leg-askel on paljon tehokkaampi. Toisaalta dog-leg-askeleen laskeminen vaatii reilusti enemmän laskentatehoa.

4. YHTEENVETO

Työssä tutkittiin luottamusaluealgoritmien toimintaa optimointiongelmissa sekä teoreettiselta että käytännölliseltä pohjalta. Algoritmin idea perustuu siihen, että kohdefunktiota approksimoidaan mallifunktiolla jollain ennalta määrätyllä luottamusalueella. Seuraava iteraatioaskel saadaan minimoimalla mallifunktio luottamusalueen sisällä, ja tämän jälkeen lasketulle askeleelle suoritetaan arviointi, jonka perusteella muokataan luottamusalueen kokoa ja päätetään hyväksytäänkö askel. Tässä työssä käsitellyissä tapauksissa mallifunktiona oli yhtälön (2.9) mukainen kvadraattinen approksimaatio ja muu algoritmin kulku tapahtui Algoritmin 1 kuvaamalla tavalla.

Algoritmin osoitettiin suppenevan, jos kullakin iteraatiokerralla laskettu uusi askel toteuttaa ehdon (2.28) ja näin ollen tuottaa riittävän vähenemän joka iteraatiolla. Funktion negatiivisen gradientin suuntainen minimi eli Cauchyn piste (2.12) on esimerkki riittävän vähenemän tuottavasta osaongelman ratkaisusta. Työssä esiteltiin myös kaksi muuta suppenevaa tapaa ratkaista luottamusalueen osaongelma: dog-leg -menetelmä (2.16), joka on Cauchyn pisteen tarjoamaa ratkaisua parantava menetelmä, sekä tapa etsiä eksaktia ratkaisua.

Käytännöllisessä osuudessa testattiin itse MATLAB-ohjelmistolla toteutettua algoritmia Rosenbrockin funktion (3.1) minimoinnissa. Algoritmi oli toteutettu käyttämällä osaongelman ratkaisuun dog-leg -menetelmää, ja se käytti 15 iteraatiota minimoinnin suorittamiseen. Algoritmin MATLAB-koodi on esitetty liitteessä 1.

Algoritmin toimintaa ongelman ratkaisemisen aikana tarkasteltiin lähemmin, ja taulukossa 3.1 sekä kuvassa 3.2 on esitetty iterointiprosessin kulkua. Vertailukohtana samaa ongelmaa yritettiin ratkaista myös lähdemateriaalista haetulla viivahakualgoritmilla, joka suoritti iteroinnin lähes yhtä nopeasti, mutta käyttäen paljon enemmän iteraatioaskelia. Näin todettiinkin, että tehty algoritmi on tehokas, mutta se vaatii myös jonkin verran laskutehoa. Tilanteesta riippuen kannattaa pohtia, millaista algoritmia optimointiongelman ratkaisussa käyttää.

LÄHTEET

[1] S. P. Boyd and L. Vandenberghe, Convex Optimization. Cambridge: Cambridge University Press, 2004.

[2] V. Buljak, Inverse Analyses with Model Reduction. Berlin: Springer, 2012.

[3] A. R. Conn, N. I. Gould, and P. L. Toint, Trust Region Methods. Philadelphia: SIAM, 2000.

[4] R. Fletcher, Practical Methods of Optimization. West Sussex: John Wiley and Sons, Inc, 2013.

[5] D. M. Gay, Computing Optimal Locally Constrained Steps, 2 SIAM Journal, pp. 185–197, 1981.

[6] N. Gould, D. Orban, A. Sartanaer, and T. Toint, Sensitivity of Trust-region Algorithms to Their Parameters, 3 4OR - A Quarterly Journal of Operations Research, pp. 227–241, 2005.

[7] T. Hämäläinen, Optimization, Tampereen teknillinen yliopisto, Tampere, 2015, luentomoniste.

[8] M. Jamil and Y. Xin-She, A Literature Survey of Benchmark Functions for Global Optimization Problems., 4 International Journal of Mathematical Modeling and Numerical Optimization, pp. 150–194, 2013.

[9] K. Levenberg, A Method for Solution of Certain Problems in Least Squares, 2 Q. J. Appl. Math., pp. 164–168, 1944.

[10] D. Marquardt, An Algorithm for Least-squares Estimation on Nonlinear Parameters, 11 SIAM J. Appl. Math., pp. 431–441, 1963.

[11] J. J. More and D. C. Sorensen, Computing a Trust Region Step, 4 SIAM Journal, pp. 553–573, 1983.

[12] D. D. Morrison, Methods for Nonlinear Least Squares Problems and Convergence Proofs, In: Lorell, J. and Yagi, F. (eds) Proceedings of the Seminar on Tracking Programs and Orbit Determination, pp. 1–9, 1960.

[13] J. Moré, Recent Developments in Algorithms and Software for Trust Region Methods, In: Bachem A., Korte B., Grötschel M. (eds) Mathematical Programming The State of the Art, pp. 258–287, 1983.

[14] J. Nocedal and S. J. Wright, Numerical Optimization. New York, NY: Springer, 2006.

[15] Y.-W. Shang and Y.-H. Qiu, A Note on the Extended Rosenbrock Function, 14 Evolutionary Computation, pp. 119–126, 2006.

[16] G. A. Shultz, R. B. Schnabel, and R. H. Byrd, A Family of Trust-Region-Based Algorithms for Unconstrained Minimization with Strong Global Convergence Properties, 22 SIAM Journal, pp. 47–68, 1985.

[17] D. C. Sorensen, Newton's Method with a Model Trust Region Modification, 19 SIAM Journal, pp. 409–426, 1982.

[18] Y.-x. Yuan, Recent Advances in Trust Region Algorithms, 151 Mathematical Programming, pp. 249–281, 2015.

[19] Y.-x. Yuan, A Review of Trust Region Algorithms for Optimization, 99 ICIAM, pp. 271–282, 2000.

LIITE A: ALGORITMIN MATLAB-KOODI

```
% Trust region -minimointialgoritmi dog-leg -menetelmällä

% Minimoitavana funktiona Rosenbrockin funktio.
% Määritellään kahvafunktiot rb, grad ja hesse, joiden avulla ...
    lasketaan
% Rosenbrockin funktion ja sen gradientin sekä Hessen matriisin arvo
% tarvittavissa pisteissä

rb = @(x) ((1-x(1)).^2) + (100*((x(2)-(x(1).^2)).^2));

grad = @(x) [2*x(1) - 400*x(1)*(- x(1)^2 + x(2)) - 2; - ...
    200*x(1)^2 + 200*x(2)];

hesse = @(x) [[1200*x(1)^2 - 400*x(2) + 2, -400*x(1)]; [ ...
    -400*x(1),    200]];

% Määritellään parametrit algoritmille
max_iter = 89; % Iteraatioiden maksimimäärä, jollei pysähdy sitä ...
    ennen
MINg_normi = 0.001; % Gradientin normin arvo, jonka jälkeen ...
    algoritmi pysäytetään
min_suhde = 0.001; % Suhteen arvo, joka vaaditaan ratkaisun ...
    hyväksymiseksi
TR_pienennys = 0.25; % Suhteen ollessa tämän alle pienennetään ...
    luottamusaluetta
pienennys = 0.25; % Kerroin, jolla pienennetään luottamusaluetta
TR_kasvatus = 0.75; % Suhteen ollessa tätä suurempi kasvatetaan ...
    luottamusaluetta
kasvatus = 2; % Kerroin, jolla kasvatetaan luottamusaluetta
max_TR = 4; % Luottamusalueen säteen maksimiarvo algoritmissä
vastaus = [1,1]';

% Annetaan alkuarvot
arvaus = [2.15, 0.88]'; % Alkuarvaus
TR_sade = 1; % Luottausalueen säteen alkuarvo
iter = 0;

% Lasketaan kohdefunktion gradientti ja Hessen matriisi
gk = grad(arvaus');
normi_g = norm(gk);
Bk = hesse(arvaus');

```

```
% Iterointisykli
while (normi_g > MINg_normi && iter < max_iter)
    iter = iter + 1;
    value = rb(arvaus'); % Funktion arvo pisteessä

    % Ratkaistaan osaongelma dog-leg -menetelmällä

    % Jos matriisi ei ole positiivisesfi definiitti, valitaan ...
        negatiivisen
    % gradientin suuntainen piste luottamusalueen rajalla
    if ((gk)'*Bk*gk <= 0 )
        kerroin = -TR_sade/normi_g;
        askel = kerroin * gk;
    else
        np = -Bk \ gk; % Lasketaan Newtonin piste
        normi_n = norm(np);

        % Jos newtonin askel on luottamusalueen sisällä, otetaan se
        % kandidaatiksi
        if (normi_n <= TR_sade)
            askel = np;

        else
            % Muussa tapauksessa lasketaan Cauchyn piste
            cp = -((gk'*gk)/(gk'*Bk*gk))*gk;
            normi_c = norm(cp);

            % Jos Cauchyn piste on luottamusalueen ulkopuolella, ...
                valitaan
            % pisteeksi negatiivisen gradientin ja alueen rajan ...
                leikkauspiste
            if (normi_c > TR_sade)
                kerroin = TR_sade / normi_g;
                askel = -kerroin * gk;
            else
                % Muussa tapauksessa muodostetaan dog-leg ...
                    -polku. Muokataan
                % yhtälöä 2.20 niin, että se voidaan ratkaista ...
                    toisen asteen
                % yhtälönä. Ratkaistaan alfa, joka vastaa ...
                    yhtälön kerrointa
                % (1-gamma)
                a = norm(np-cp)^2;
                b = cp'*(np-cp);
                c = norm(cp)^2 - TR_sade^2;
                D = b^2 - a*c;
                alfa = (-b + sqrt(D))/a;
```

```
                dogleg = cp + alfa * (np-cp);
                askel = dogleg;
            end
        end
    end
    % Arvioidaan ratkaisua
   malli_red = -(askel)'*gk - 0.5*(askel)'*Bk*(askel);
   funktion_arvo = rb(arvaus + askel);
   fun_red = value - funktion_arvo;
   suhde = fun_red/malli_red;

   % Muutetaan luottamusalueen kokoa ratkaisun arvioinnin ...
       perusteella
   %suhded = double(suhde);
   % Jos ratkaisu liian huono, pienennetään luottamusaluetta
   if (suhde < TR_pienennys)
       TR_sade = pienennys*TR_sade;
   else
       % Kasvatetaan luottamusaluetta, jos ratkaisu riittävän hyvä
       if (suhde > TR_kasvatus && TR_sade - norm(askel) >= ...
           1.0e-06*TR_sade)
           TR_sade = min([kasvatus*TR_sade, max_TR]);
       end
   end

   % Hyväksytään ratkaisu uudeksi iteraatiopisteeksi, jos ...
       vähenemistä
   % tapahtuu
   if (suhde > min_suhde)
       arvaus = arvaus + askel;
       % Ratkaistaan gradientti ja Hessen matriisi seuraavalle
       % iteraatiokierrokselle
       gk = grad(arvaus);
       Bk = hesse(arvaus);
       normi_g = norm(gk);
   end

end
```