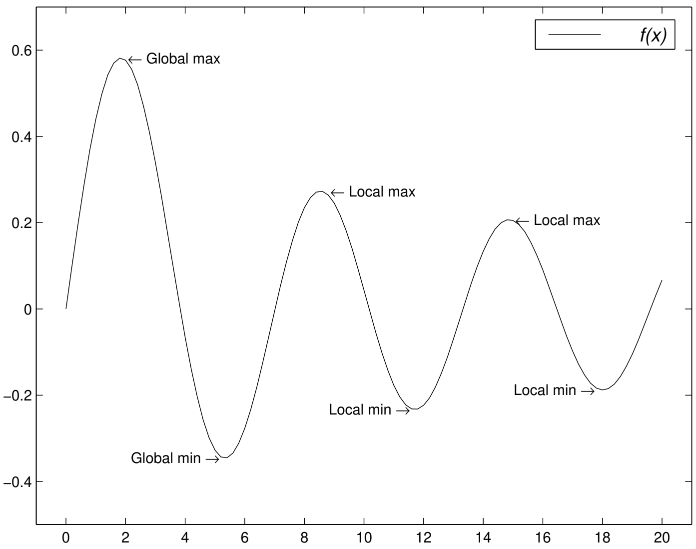
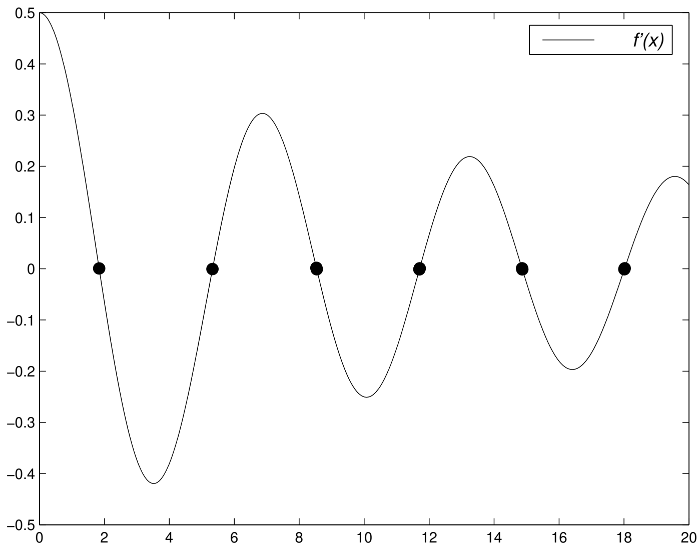
Luku 3

Optimointimenetelmien alkeita

Optimoinnilla tarkoitetaan "parhaan mahdollisen" etsimistä. Jotta eri mahdollisuuksia voidaan vertailla ja luokitella, tarvitaan sopiva mittari - kustannusfunktio, jonka ääriarvot (minimit ja maksimit) ovat juuri haluttuja optimipisteitä. Tällaisia pisteitä karakterisoivat *derivoituvan funktion* derivaatan nollakohdat. Tätä on havainnollistettu alla olevissa kuvissa:





3.1 Optimointitehtävä yleisessä muodossaan

Tarkastellaan (rajoittamatonta) minimointitehtävää, jonka yleinen muoto on

$$\text{minimoi} \quad \mathcal{J}(\mathbf{v}) \quad \text{missä } \mathbf{v} = \begin{bmatrix} \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_n \end{bmatrix} \in \mathbf{R}^n \tag{3.1}$$

ja $\mathcal{J}$ on kustannusfunktio $\mathbf{R}^n$:stä $\mathbf{R}$:ään. Oletetaan tästä lähtien, että $\mathcal{J}(\mathbf{v}) \geq 0$ kaikille $\mathbf{v} \in \mathbf{R}^n$. (Huomataan, että jos tehtävänä on maksimoida funktio $\mathcal{J}(v)$, niin on yhtäpitävää minimoida funktiota $-\mathcal{J}(v)$). Vektorin $\mathbf{v} \in \mathbf{R}^n$ komponentit $(\mathbf{v}_1, \ldots, \mathbf{v}_n)$ ovat tehtävään liittyvät varsinaiset *muuttujat* eli tuntemattomat, joiden arvoja olemme määräämässä.

Määritelmä 1. Piste $\mathbf{v}^*$ on tehtävän (3.1) *globaali minimi*, jos

$$\mathcal{J}(\mathbf{v}^*) \leq \mathcal{J}(\mathbf{v}) \quad \text{kaikilla } \mathbf{v} \in \mathbf{R}^n.$$

Määritelmä 2. Piste $\mathbf{v}^*$ on tehtävän (3.1) *lokaali minimi*, jos on olemassa $\delta > 0$ siten, että

$$\mathcal{J}(\mathbf{v}^*) \leq \mathcal{J}(\mathbf{v}) \quad \text{kaikilla } \mathbf{v} \in \mathbf{R}^n,$$

joille pätee $\|\mathbf{v} - \mathbf{v}^*\| \leq \delta$.

HUOM: vektoriavaruudessa $\mathbf{R}^n$ on käytössä *Euklidinen normi* $\|\mathbf{v}\| = \sqrt{\mathbf{v}^T\mathbf{v}} = \left(\sum_{i=1}^n \mathbf{v}_i^2\right)^{1/2}$.

Puhuttaessa *aidoista* lokaaleista tai globaalista minimistä, käytetään yo. määritelmissä aitoja epäyhtälöitä.

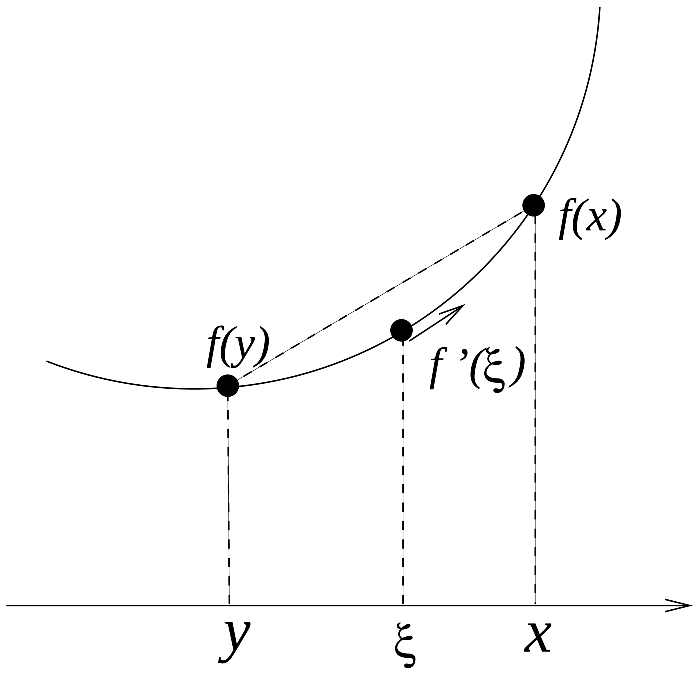
Muistellaan tässä yhteydessä vielä ns. Weierstrassin lausetta, joka antaa tietoa siitä, milloin minimiratkaisu on olemassa.

Lause 1. Jatkuvalle funktiolle $\mathcal{J} : \mathbf{R}^n \to \mathbf{R}$ on tehtävällä (3.1) olemassa minimiratkaisu $\mathbf{v}^*$.

Tällä kurssilla rajoitumme tarkastelemaan vain ns. gradientti-pohjaisia menetelmiä tehtävän (3.1) ratkaisemiseksi. Peruslähtökohtana on aluksi yleistää yksiulotteinen derivaatan määrittely moniulotteiseen avaruuteen $\mathbf{R}^n$, josta tehtävän ratkaisua ollaan hakemassa. Sopivan yleistyksen tarjoaa funktion $\mathcal{J}$ *gradientti* $\nabla\mathcal{J}$, jonka nollakohdista moniulotteisen funktion ääriarvoja voidaan hakea analogisesti 1d tapaukselle. Teoreettisesti toinen peruslähtökohta on mahdollisuus *approksimoida* funktiota $\mathcal{J}$ kyseisen gradientin avulla. Tässä yleistyy moniulotteiseen tapaukseen (joillekin?) tuttu yksiulotteinen *differentiaalilaskennan väliarvolause*:

$$f(x) = f(y) + f'(\xi)(x - y) \quad \text{jollekin } \xi \in (x, y)$$

jatkuvasti differentioituvalle funktiolle $f : \mathbf{R} \to \mathbf{R}$. Tätä on havainnollistettu alla olevassa kuvassa:



3.2 Gradienttimenetelmien teoreettiset perusteet

Funktio $\mathcal{J}$ on *differentioituva* ($\mathcal{J} \in C^1(\mathbf{R}^n)$) pisteessä $\mathbf{v}$, jos on olemassa vektori $\nabla\mathcal{J}(\mathbf{v}) \in \mathbf{R}^n$ ja funktio $\varepsilon : \mathbf{R}^n \to \mathbf{R}$ siten, että

$$\mathcal{J}(\bar{\mathbf{v}}) = \mathcal{J}(\mathbf{v}) + \nabla\mathcal{J}(\mathbf{v})^T(\bar{\mathbf{v}} - \mathbf{v}) + \|\bar{\mathbf{v}} - \mathbf{v}\|\varepsilon(\mathbf{v}, \bar{\mathbf{v}} - \mathbf{v}) \tag{3.2}$$

kaikilla $\bar{\mathbf{v}}$, missä $\nabla\mathcal{J}(\mathbf{v}) \in \mathbf{R}^n$ on $\mathcal{J}$:n *gradientti* pisteessä $\mathbf{v}$ ja $\varepsilon(\mathbf{v}, \bar{\mathbf{v}} - \mathbf{v}) \to 0$ kun $\bar{\mathbf{v}} \to \mathbf{v}$. Gradientin komponentteja ovat *osittaisderivaatat*

$$\nabla\mathcal{J}(\mathbf{v}) = \begin{bmatrix} \frac{\partial\mathcal{J}(\mathbf{v})}{\partial\mathbf{v}_1} \\ \vdots \\ \frac{\partial\mathcal{J}(\mathbf{v})}{\partial\mathbf{v}_n} \end{bmatrix},$$

ja gradientti määritellään siis *samassa tietorakenteessa* kuin muuttuja $\mathbf{v}$ itse eli $\mathbf{R}^n$:n vektorina. Geometriselta kannalta katsottuna gradienttivektori on funktion $\mathcal{J}$ *tangenttitason* suuntainen.

Funktio $\mathcal{J}$ on *kahdesti differentioituva* ($\mathcal{J} \in C^2(\mathbf{R}^n)$) pisteessä $\mathbf{v}$, jos on olemassa vektori $\nabla\mathcal{J}(\mathbf{v})$ ja symmetrinen $n \times n$ -matriisi $\mathbf{H}(\mathbf{v})$, niin sanottu *Hessen matriisi*, ja funktio $\varepsilon : \mathbf{R}^n \to \mathbf{R}$ siten, että

$$\mathcal{J}(\bar{\mathbf{v}}) = \mathcal{J}(\mathbf{v}) + \nabla\mathcal{J}(\mathbf{v})^T(\bar{\mathbf{v}} - \mathbf{v}) + \frac{1}{2}(\bar{\mathbf{v}} - \mathbf{v})^T\mathbf{H}(\mathbf{v})(\bar{\mathbf{v}} - \mathbf{v}) + \|\bar{\mathbf{v}} - \mathbf{v}\|^2\varepsilon(\mathbf{v}, \bar{\mathbf{v}} - \mathbf{v}), \tag{3.3}$$

missä $\varepsilon(\mathbf{v}, \bar{\mathbf{v}} - \mathbf{v}) \to 0$ kun $\bar{\mathbf{v}} \to \mathbf{v}$. Kahdesti differentioituvan funktion Hessen matriisi koostuu *toisen asteen osittaisderivaatoista* $\frac{\partial^2\mathcal{J}(\mathbf{v})}{\partial\mathbf{v}_i\partial\mathbf{v}_j}$ $(\mathbf{H}(\mathbf{v}) \simeq \nabla(\nabla\mathcal{J}(\mathbf{v}))^T)$:

$$\mathbf{H}(\mathbf{v}) = \begin{bmatrix} \frac{\partial^2\mathcal{J}(\mathbf{v})}{\partial\mathbf{v}_1^2} & \cdots & \frac{\partial^2\mathcal{J}(\mathbf{v})}{\partial\mathbf{v}_1\partial\mathbf{v}_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2\mathcal{J}(\mathbf{v})}{\partial\mathbf{v}_n\partial\mathbf{v}_1} & \cdots & \frac{\partial^2\mathcal{J}(\mathbf{v})}{\partial\mathbf{v}_n^2} \end{bmatrix}.$$

Jatkossa objektifunktion $\mathcal{J}$ Hessen matriisia merkitään $\mathbf{H}$:lla. Huomaa, että matriisin $\mathbf{H}$ symmetrisyys seuraa siitä, että $\frac{\partial^2\mathcal{J}(\mathbf{v})}{\partial\mathbf{v}_i\partial\mathbf{v}_j} = \frac{\partial^2\mathcal{J}(\mathbf{v})}{\partial\mathbf{v}_j\partial\mathbf{v}_i}$ kaikille $1 \leq i, j \leq n$ kun $\mathcal{J} \in C^2(\mathbf{R}^n)$.

Määritelmä 3. Olkoon funktio $\mathcal{J} : \mathbf{R}^n \to \mathbf{R}$. Vektori $\mathbf{d} \in \mathbf{R}^n$ on funktion $\mathcal{J}$ *laskeva suunta* pisteessä $\mathbf{v}^*$, jos on olemassa $\delta > 0$ siten, että

$$\mathcal{J}(\mathbf{v}^* + t\mathbf{d}) < \mathcal{J}(\mathbf{v}^*) \quad \text{kaikilla } t \in (0, \delta].$$

Lause 2. Olkoon funktio $\mathcal{J} : \mathbf{R}^n \to \mathbf{R}$ differentioituva pisteessä $\mathbf{v}^*$. Jos on olemassa suunta $\mathbf{d}$ siten, että $\nabla\mathcal{J}(\mathbf{v}^*)^T\mathbf{d} < 0$, niin $\mathbf{d}$ on funktion $\mathcal{J}$ laskeva suunta pisteessä $\mathbf{v}^*$.

Lause 3. Olkoon funktio $\mathcal{J} : \mathbf{R}^n \to \mathbf{R}$ differentioituva pisteessä $\mathbf{v}^*$. Jos $\mathbf{v}^*$ on lokaali minimipiste, niin $\nabla\mathcal{J}(\mathbf{v}^*) = 0$ eli $\mathbf{v}^*$ on funktion *kriittinen piste*.

Ylläolevassa *välttämättömässä* (jos ..., niin ...) optimaalisuusehdossa $\nabla\mathcal{J}(\mathbf{v}^*) = 0$ esiintyy siis kustannusfunktion gradienttivektori. Siksi sitä kutsutaan *ensimmäisen asteen optimaalisuusehdoksi*. Huomataan, että termi *ensimmäinen aste* kuvaa tässä yhteydessä oikeastaan kahtakin asiaa: gradientissa esiintyviä ensimmäisiä derivaattoja sekä kaavan (3.2) sisältämää virheen suuruutta, joka riippuu etäisyyden $\|\bar{\mathbf{v}} - \mathbf{v}\|$ ensimmäisestä potenssista.

Välttämättömiä ehtoja voidaan lausua myös funktion toisen asteen osittaisderivaatoista muodostuvan Hessen matriisin avulla. Tällaisia optimaalisuusehtoja kutsutaan *toisen asteen optimaalisuusehdoiksi*, ja tässä jälleen termi *toinen aste* viittaa sekä toisen asteen derivaattoihin että kaavan (3.3) sisältämään virheeseen, joka riippuu etäisyyden $\|\bar{\mathbf{v}} - \mathbf{v}\|$ neliöstä.

Lause 4. Olkoon funktio $\mathcal{J} : \mathbf{R}^n \to \mathbf{R}$ kahdesti differentioituva pisteessä $\mathbf{v}^*$. Jos $\mathbf{v}^*$ on lokaali minimipiste, niin $\nabla\mathcal{J}(\mathbf{v}^*) = 0$ ja Hessen matriisi $\mathbf{H}(\mathbf{v}^*)$ on positiivisesti semidefiniitti. Jos $\nabla\mathcal{J}(\mathbf{v}^*) = 0$ ja Hessen matriisi $\mathbf{H}(\mathbf{v}^*)$ on positiividefiniitti, niin $\mathbf{v}^*$ on *aito* lokaali minimipiste.

3.3 Perusalgoritmi

Gradientti-pohjainen optimointimenetelmä yleisen minimointitehtävän (3.1) ratkaisemiseksi noudattaa seuraavaa iteratiivista algoritmia:
Algoritmi 1.

1. Valitse aloituspiste $\mathbf{v}^0$. Aseta iteraatiolaskuri $k = 0$.
2. Generoi laskeva suunta $\mathbf{d}^k$.
3. Laske askelpituus t^k siten, että $\mathcal{J}(\mathbf{v}^k + t^k\mathbf{d}^k) < \mathcal{J}(\mathbf{v}^k)$.
4. Aseta $\mathbf{v}^{k+1} = \mathbf{v}^k + t^k\mathbf{d}^k$.
5. Lopetustesti. Jos jatketaan, aseta $k = k + 1$ ja mene kohtaan 2.

Algoritmin lopetustestina voi käyttää esim. seuraavia mittareita, jotka kaikki vaativat sopivan toleranssin $\varepsilon > 0$ asettamista etukäteen:

- **kriittinen piste:** $\|\nabla\mathcal{J}(\mathbf{v}^{k+1})\| \le \varepsilon$.
- **ratkaisun muutos:** $\|\mathbf{v}^{k+1} - \mathbf{v}^k\| = t^k\|\mathbf{d}^k\| \le \varepsilon$.
- **kustannuksen muutos:** $\dfrac{\mathcal{J}(v^{k+1}) - \mathcal{J}(\mathbf{v}^k)}{\max(\delta, |\mathcal{J}(\mathbf{v}^k)|, |\mathcal{J}(\mathbf{v}^{k+1}|)} \le \varepsilon, \quad$ missä $\delta > 0$.

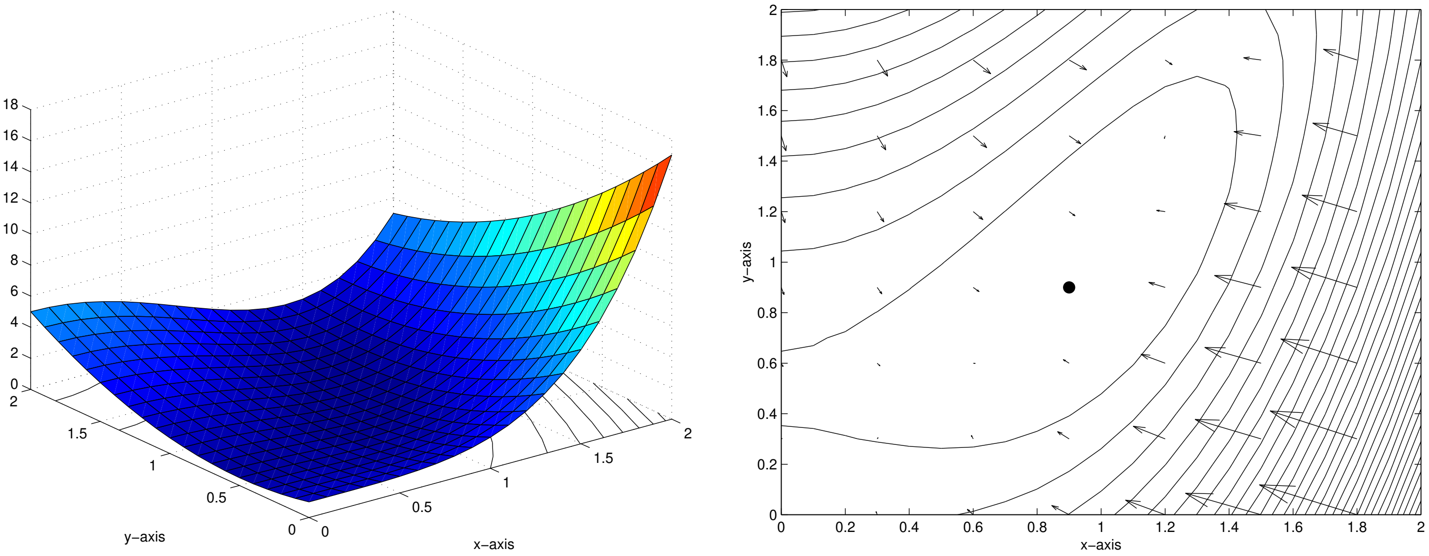
Useissa käytännön toteutuksissa askeleen 2. laskeva suunta $\mathbf{d}^k$ *normeerataan* asettamalla $\mathbf{d}^k = \mathbf{d}^k/\|\mathbf{d}^k\|$. Huomataan, että tämä vaatii oletuksen $\mathbf{d}^k \neq 0$, ja usein askeleen 5. lopetustesti suoritetaankin jo askeleen 2. yhteydessä, koska kaikki ylläannetut lopetuskriteerit "laukeavat" kun hakusuunnasta tulee tarpeeksi lyhyt. Algoritmista 1 voidaan vielä huomauttaa, että käytännössä askeleet 3. ja 4. tehdään yhtäaikaisesti eli uusi ratkaisun kandidaatti $\mathbf{v}^{k+1}$ päivitetään jo valmiiksi askelpituuden määrämisen yhteydessä turhien laskentaoperaatioiden välttämiseksi.

Ohitetaan tässä yhteydessä ns. *konvergenssitarkastelut* (löytääkö algoritmi minimin vai ei?) toteamalla, että jos pystymme joka iteraatiolla *aidosti* pienentämään alhaalta rajoitetun kustannusfunktion $\mathcal{J}$ arvoa, meidän *täytyy* päästä *lokaalin* "kuopan pohjalle" äärellisen monella iteraatiolla. Menetelmän tehokkuus määräytyy siitä, kuinka kauan tähän prosessiin kuluu aikaa. Siten suoritusajassa kertautuu yhden iteraation laskennallinen vaativuus niiden lukumäärän kanssa.

3.4 Askelpituuden määrääminen

Oletetaan, että laskeva suunta $\mathbf{d}^k$ on saatu generoitua kuvan 3.1 mukaisesti. Tarkastellaan seuraavaksi muutamaa yksinkertaista tapaa määrätä sopiva askelpituus t^k. Peruslähtökohtana on tarkastella laskevan suunnan suhteen yksiulotteista minimointitehtävää

$$\min_{t\in I} \mathcal{J}(\mathbf{v}^k + t\mathbf{d}^k) = j(t), \tag{3.4}$$



Kuva 3.1: Funktio ja sen laskevia suuntia.

missä I on askelpituudelle ennalta määrätty *hakuväli*, joksi yleensä valitaan $I = [0, 1]$ (konveksisuuden vuoksi).

Periaatteessa tehtävä (3.4) voidaan ratkaista millä tahansa yksiulotteisella minimointitekniikalla, joita ovat esimerkiksi erilaiset hakuväliä systemaattisesti pienentävät (puolitusmenetelmä, regula-falsi, kultainen leikkaus ...) menetelmät. Minimointimenetelmää valittaessa tulee kuitenkin huomioida se, että prosessi joudutaan toistamaan jokaisella iteraatiolla. Lisäksi yleensä monimutkaisten kustannusfunktioiden yhteydessä funktion arvon laskeminen annetussa pisteessä voi olla kallista eli paljon aikaavievää. Tämän vuoksi yleinen strategia on korvata tehtävän (3.4) tarkka ratkaiseminen jollakin yksinkertaisella menetelmällä, jolla saadaan nopeasti laskettua "tarpeeksi paljon" kustannusfunktion arvoa pienentävä seuraava ratkaisuehdokas $\mathbf{v}^{k+1}$.

Seuraavaksi käydään pintapuolisesti läpi muutama yksinkertainen ja yleensä kohtuullisen nopea perustekniikka askelpituuden $t^* = t^k$ määräämiseksi:

- **Kiinnitetty askelpituus:** Valitaan etukäteen jokin askelpituus $0 < t^* < 1$, jota käytetään koko ajan. Sopiva valinta riippuu täysin tehtävän luonteesta ja erityisesti suuntavektorin $\mathbf{d}^k$ "hyvyydestä", mutta usein käytettyjä arvoja ovat esimerkiksi valinnat $t^* = 0.01,\ 0.05,\ 0.1$.

- **Kvadraattinen interpolaatio:** Approksimoidaan funktiota j toisen asteen polynomilla $j(t) \simeq p(t) = a\,t^2 + bt + c$, jonka derivaatan nollakohdasta $p'(t) = 2at + b = 0$ saadaan askelpituuden arvo $t^* = -b/(2a)$ kun $a \neq 0$.

  Polynomin määräävien reaalilukujen a, b ja c etsimisessä käytetään yleensä kahta eri perusmenetelmää. Ensimmäisessä tavassa käytetään kustannusfunktiolle laskettuja arvoja kolmessa eri pisteessä, esimerkiksi

$$\begin{cases} t_0 = 0: & j_0 = \mathcal{J}(\mathbf{v}^k) \\ t_1 = \frac{1}{2}: & j_1 = \mathcal{J}(\mathbf{v}^k + \frac{1}{2} \cdot \mathbf{d}^k) \\ t_2 = 1: & j_2 = \mathcal{J}(\mathbf{v}^k + 1 \cdot \mathbf{d}^k) \end{cases}$$

  Toisen asteen polynomi, joka kulkee kolmen pisteen $(t_i, j_i),\ i = 1, 2, 3,$ kautta määrätään ratkaisemalla lineaarinen yhtälöryhmä

$$\begin{cases} a\,0^2 + b\,0 + c &= j_0 \\ a\left(\dfrac{1}{2}\right)^2 + b\,\dfrac{1}{2} + c &= j_1 \\ a\,1^2 + b\,1 + c &= j_2 \end{cases},$$

jonka ratkaisusta

$$\begin{cases} c &= j_0 \\ a &= 2(j_0 - 2j_1 + j_2) \\ b &= -3j_0 + 4j_1 - j_2 \end{cases}$$

saadaan $t^* = \frac{3j_0 - 4j_1 + j_2}{4(j_0 - 2j_1 + j_2)}$.

Jos pisteessä $\mathbf{v}^k$ on laskettu kustannusfunktion arvon $j_0 = \mathcal{J}(\mathbf{v}^k)$ lisäksi gradientti $\nabla\mathcal{J}(\mathbf{v}^k)$, voidaan polynomin määräämisessä käyttää myös tätä tietoa hyväksi. Tätä varten arvioidaan yksiulotteisen funktion $j(t)$ derivaattaa pisteessä $t_0 = 0$ kaavan (3.2) avulla:

$$\begin{aligned} j'(t_0) &= \lim_{t\to 0} \frac{\mathcal{J}(\mathbf{v}^k + t\mathbf{d}^k) - \mathcal{J}(\mathbf{v}^k)}{t} \\ &= \lim_{t\to 0} \frac{\mathcal{J}(\mathbf{v}^k) + \nabla\mathcal{J}(\mathbf{v}^k)^T(\mathbf{v}^k + t\mathbf{d}^k - \mathbf{v}^k) + \|\mathbf{v}^k + t\mathbf{d}^k - \mathbf{v}^k\|\varepsilon(\mathbf{v}^k, \mathbf{v}^k + t\mathbf{d}^k - \mathbf{v}^k) - \mathcal{J}(\mathbf{v}^k)}{t} \\ &= \lim_{t\to 0} \frac{t\nabla\mathcal{J}(\mathbf{v}^k)^T\mathbf{d}^k + t\|\mathbf{d}^k\|\varepsilon(\mathbf{v}^k, t\mathbf{d}^k)}{t} \\ &= \nabla\mathcal{J}(\mathbf{v}^k)^T\mathbf{d}^k. \end{aligned} \tag{3.5}$$

Kun nyt valitaan jokin $0 < t_1 \leq 1$ ja lasketaan sitä vastaava kustannusfunktion arvo $j_1 = \mathcal{J}(\mathbf{v}^k + t_1\mathbf{d}^k)$, saadaan polynomin kertoimet kaavoista ($j'(0) = b$ ja $at_1^2 + bt_1 + c = j_1$):

$$\begin{cases} c &= j_0 \\ b &= \mathcal{J}(\mathbf{v}^k)^T\mathbf{d}^k \\ a &= \dfrac{j_1 - bt_1 - c}{t_1^2} \end{cases}.$$

Lopuksi huomataan, että jos $a < 0$ niin kvadraattisen funktion kuvaaja aukeaa alaspäin, jolloin derivaatan nollakohta viittakiin maksimiin eikä minimiin. Jos näin käy, täytyy polynomin määräämisväliä $I = [0, 1]$ muuttaa, esimerkiksi voidaan valita $I = [0, \frac{1}{2}]$, jolloin funktion j arvo tarvitsee laskea yhdessä uudessa pisteessä $\tilde{t} = \frac{1}{4}$ ja prosessi (paraabelin määrääminen) voidaan toistaa alusta tällä uudella välillä.

- **Kuutiollinen interpolaatio:** Kuten kvadraattinen, mutta polynomiksi valitaan kolmannen asteen polynomi, jonka määräämiseksi tarvitaan funktion j arvot neljässä pisteessä tai funktion ja sen gradientin arvot kahdessa pisteessä (kaava (3.5) pätee myös muille pisteille kuin $t_0 = 0$). Muuten prosessi on vastaava kuin edellä.

Seuraava algoritmi, johon on lisätty kaikenlaisia hyödyllisiä tarkistuksia, suorittaa askelpituuden määräämisen kvadraattisen interpolaation avulla:

Algoritmi 2.

-1. Oletetaan, etta kutsussa aliohjelmaan välittyvät nykyinen vektori $\mathbf{v}$, suuntavektori $\mathbf{d}$ sekä kustannusfunktion arvo $j = \mathcal{J}(\mathbf{v})$ ja gradienttivektori $\mathbf{g} = \nabla \mathcal{J}(\mathbf{v})$.

0. Alusta parametrit $\varepsilon > 0$ (esim. $\varepsilon = 10^{-6}$) ja $c > 0$ (esim. $c = 0.05$), jotka kontrolloivat uuden pisteen hyväksymistoleranssia ja askelpituuden pienenemistä. Laske $acc = \varepsilon \|d\|$ ja aseta $t = 1$.

1. Laske $\mathbf{u} = \mathbf{v} + t\mathbf{d}$ ja $ju = \mathcal{J}(\mathbf{u})$. Jos $ju < j - t \cdot acc$, aseta $\mathbf{v} = \mathbf{u}$ ja $j = ju$. RETURN.

2. Laske $b = \mathbf{g}^T\mathbf{d}$ ja $a = ju - j - b \cdot t$. Jos $a < \varepsilon^2$, aseta $a = c \cdot t$, muuten aseta $a = -t^2 \cdot b/(2a)$. Jos $a < \varepsilon$ tai $a > (1-c)\,t$, aseta $a = c \cdot t$.

3. Laske $\mathbf{u} = \mathbf{v} + a\mathbf{d}$ ja $ju = \mathcal{J}(\mathbf{u})$. Jos $ju < j - a \cdot acc$, aseta $\mathbf{v} = \mathbf{u}$ ja $j = ju$ (askelpituudeksi t tuli a). RETURN.

4. Aseta $a = c \cdot a$. Laske $\mathbf{u} = \mathbf{v} + a\mathbf{d}$ ja $ju = \mathcal{J}(\mathbf{u})$. Jos $ju \geq j - a \cdot acc$ ja $a \geq \varepsilon$, toista askel 4.

5. Jos $a \geq \varepsilon$, aseta $\mathbf{v} = \mathbf{u}$ ja $j = ju$ (askelpituudeksi t tuli a), muuten $t = 0$. RETURN.

3.4.1 Laskevan suunnan määrääminen

Lauseesta 2 seuraa, että jos Algoritmi 1 ei ole vielä löytänyt kriittistä pistettä $\nabla \mathcal{J}(\mathbf{v}^k) = 0$, antaa $-\nabla \mathcal{J}(\mathbf{v}^k)$ halutunlaisen laskevan suuntavektorin:

$$-\nabla \mathcal{J}(\mathbf{v}^k)^T \nabla \mathcal{J}(\mathbf{v}^k) = -\|\nabla \mathcal{J}(\mathbf{v}^k)\|^2 < 0.$$

Edelleen voidaan osoittaa, että kyseinen vektori $-\nabla \mathcal{J}(\mathbf{v}^k)$ osoittaa kustannusfunktion *jyrkimmän laskun* suuntaan pisteessä $\mathbf{v}^k$ (eli suuntaan, jossa arvo pienenee nopeimmin) sisältäen siinä suhteessa parhaan mahdollisen informaation (siis suunnan, ei välttämättä pituuden suhteen). Kuvan 3.1 laskevat suunnat ovat muuten juuri havainnollistetun funktion (diskreettejä) negatiivia gradienttivektoreita.

Ns. *Newtonin menetelmässä*, jossa käytetään hyväksi Hessen matriisin $\mathbf{H}(\mathbf{v}^k)$ sisältämää toisen asteen informaatiota funktion $\mathcal{J}$ kaareutumisominaisuuksista, ratkaistaan suuntavektori $\mathbf{d}^k$ lineaarisesta yhtälöryhmästä

$$\mathbf{H}(\mathbf{v}^k)\mathbf{d}^k = -\nabla \mathcal{J}(\mathbf{v}^k) = -\mathbf{g}^k \tag{3.6}$$

(vrt. yksiulotteiseen iterointikaavaan $x^{k+1} = x^k - \frac{f'(x^k)}{f''(x^k)}$ yhtälön $f'(x) = 0$ ratkaisemiseksi). Jos matriisi $\mathbf{H}(\mathbf{v}^k)$ on positiividefiniitti (jolloin myös $[\mathbf{H}(\mathbf{v}^k)]^{-1}$ on SPD), saadaan määritelmän mukaisesti

$$\nabla \mathcal{J}(\mathbf{v}^k)^T \mathbf{d}^k = -{\mathbf{g}^k}^T [\mathbf{H}(\mathbf{v}^k)]^{-1} \mathbf{g}^k < 0$$

eli saatu suunta $\mathbf{d}^k$ on edelleen laskeva. Ongelmina Newtonin menetelmässä ovat vaikeiden ja monia muuttujia sisältävien kustannusfunktioiden tapauksessa Hessen matriisin $\mathbf{H}(\mathbf{v}^k)$ analyyttinen laskeminen sekä lineaarisen yhtälöryhmän muodostamisen ja ratkaisemisen vaatima suoritusaika.

Tämän vuoksi usein pyritäänkin vain muodostamaan jonkinlainen helppokäyttöinen aproksimaatio tarkan Hessen matriisin sijasta. Näitä menetelmiä kutsutaan yleisesti ns. *kvasi-Newton* menetelmiksi. Tällä hetkellä suosituin variantti on ns. BFGS-kaava (Broyden-Fletcher-Goldfarb-Shanno), jossa Hessen matriisin käänteismatriisia $\mathbf{D}^k = (\mathbf{H}(\mathbf{v}^k))^{-1}$ arvioidaan iteratiivisella kaavapläjäytyksellä

$$\mathbf{D}^{k+1} = \mathbf{D}^k + \left(1 + \frac{\mathbf{q}^T\mathbf{D}^k\mathbf{q}}{\mathbf{p}^T\mathbf{q}}\right)\frac{\mathbf{p}\,\mathbf{p}^T}{\mathbf{p}^T\mathbf{q}} - \frac{\mathbf{D}^k\mathbf{q}\,\mathbf{p}^T + \mathbf{p}\,(\mathbf{D}^k\mathbf{q})^T}{\mathbf{p}^T\mathbf{q}}, \tag{3.7}$$

missä

$$\begin{aligned} \mathbf{p} &= \mathbf{v}^{k+1} - \mathbf{v}^k, \\ \mathbf{q} &= \mathbf{g}^{k+1} - \mathbf{g}^k. \end{aligned} \tag{3.8}$$

Käytännössä yo. päivityskaava lisätään perusalgoritmiin omaksi askeleekseen, joka tapahtuu varsinaisen ratkaisun muuttamisen $\mathbf{v}^k \longrightarrow \mathbf{v}^{k+1}$ jälkeen. Menetelmä vaatii symmetrisen ja positiividefinittisen alkuarvauksen $\mathbf{D}^0$ antamisen, joksi yleensä valitaan identtinen matriisi $\mathbf{I}$. Lopuksi, koska kvasi-Newton kaavoissa Hessen matriisin käänteisen aproksimaatio kasautuu iteraatioiden aikana, tekniikasta käytetään yleensä versiota, jossa matriisi $\mathbf{D}^k$ alustetaan uudelleen $\mathbf{D}^0$:ksi jonkin sopivan iteraatiotoleranssin tullessa täyteen (esim. joka 20. iteraatio).

Mainitaanpa lopuksi lyhyesti vielä *Levenberg–Marquart* -niminen tekniikka ns. PNS-tehtävien (Pienimmän-Neliö-Summan) ratkaisemiseen. Optimointitehtävä on PNS-tehtävissä muotoa

$$\text{minimoi} \quad \mathcal{J}(\mathbf{v}) = \frac{1}{2}\sum_{i=1}^{m}\mathbf{e}_i(\mathbf{v})^2 = \frac{1}{2}\mathbf{E}(\mathbf{v})^T\mathbf{E}(\mathbf{v}), \quad \text{missä } \mathbf{E}(\mathbf{v}) = \begin{bmatrix} \mathbf{e}_1(\mathbf{v}) \\ \vdots \\ \mathbf{e}_m(\mathbf{v}) \end{bmatrix} \tag{3.9}$$

on ns. virhevektori. Suoralla laskulla nähdään, että funktion $\mathcal{J}(\mathbf{v})$ 1. derivaatta saadaan muotoon:

$$\nabla\mathcal{J}(\mathbf{v}) = \nabla(\frac{1}{2}\sum_i e_i(\mathbf{v})^2) = \sum_i \nabla e_i(\mathbf{v}) \cdot e_i(\mathbf{v}) = \mathbf{J}(\mathbf{v})^T\,\mathbf{E}(\mathbf{v}), \tag{3.10}$$

missä $\mathbf{J}(\mathbf{v})$ on ns. *Jacobin matriisi*

$$\mathbf{J}(\mathbf{v}) = \begin{bmatrix} \frac{\partial \mathbf{e}_1(\mathbf{v})}{\partial \mathbf{v}_1} & \cdots & \frac{\partial \mathbf{e}_1(\mathbf{v})}{\partial \mathbf{v}_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{e}_m(\mathbf{v})}{\partial \mathbf{v}_1} & \cdots & \frac{\partial \mathbf{e}_m(\mathbf{v})}{\partial \mathbf{v}_n} \end{bmatrix} \in \mathbf{R}^{m\times n}. \tag{3.11}$$

Nyt hakusuunta määrätään ratkaisemalla lineaarinen yhtälöryhmä

$$(\mathbf{J}(\mathbf{v}^k)^T\,\mathbf{J}(\mathbf{v}^k) + \mu^k\mathbf{I})\,\mathbf{d}^k = -\mathbf{J}(\mathbf{v}^k)^T\,\mathbf{E}(\mathbf{v}^k), \tag{3.12}$$

missä symmetrinen matriisi $\mathbf{J}(\mathbf{v}^k)^T\,\mathbf{J}(\mathbf{v}^k)$ muodostaa osan minimointitehtävän (3.9) Hessen matriisista (tämä nähdään laskemalla toinen derivaatta kaavan (3.10) avulla). Reaaliparametrin $\mu^k > 0$ tarkoituksena on toisaalta taata käännettävän matriisin ei-singulaarisuus ja toisaalta rajoittaa saatua askelpituutta. Parametrin valintaan ei puututa tämän enempää, mutta todetaan lopuksi kuitenkin, että μ^k:n järkevällä valinnalla voidaan joissain tapauksissa aikaansaada (rivakastikin) konvergoiva algoritmi, joka ei vaadi erillistä yksiulotteista minimointia.