Rakettitieteen jalanjäljillä
Osa 7: Moniulotteista optimointia

Sampo Niskanen

Rakettien simulointiohjelmistoja käytetään usein tavoitellessa jotain haluttuja raketin lento-ominaisuuksia. Tyypillisesti raketin halutaan lentävän mahdollisimman korkealle, mutta yhtä hyvin voidaan tavoitella suurta lentonopeutta, mahdollisimman pitkää lentoaikaa tai laskeutumista mahdollisimman lähelle laukaisupistettä.

Monesti raketin parantelu tehdään yrityksen ja erehdyksen periaattein — testaillaan käsin mikä toimii ja mikä ei. Tehokkaina numeronmurskaimina tietokoneet tarjoaisivat kuitenkin oivan apuvälineen kokeiluiden automatisointiin. Esimerkiksi RockSim tarjoaa mahdollisuuden optimoida raketin massaa saavuttaakseen mahdollisimman suuren lentokorkeuden.

Raketin optimaalisen massan etsiminen on suoraviivainen yksiulotteinen optimointitehtävä: etsi se massa, jolla lentokorkeus saavuttaa suurimman arvonsa. RockSim ottaa syötteenä minimi- ja maksimimassan ja simuloi raketin lennon tasavälein määrätyn monessa pisteessä näiden väliltä. Näin saa piirrettyä kuvaajan siitä, miten raketin massa vaikuttaa lentokorkeuteen.

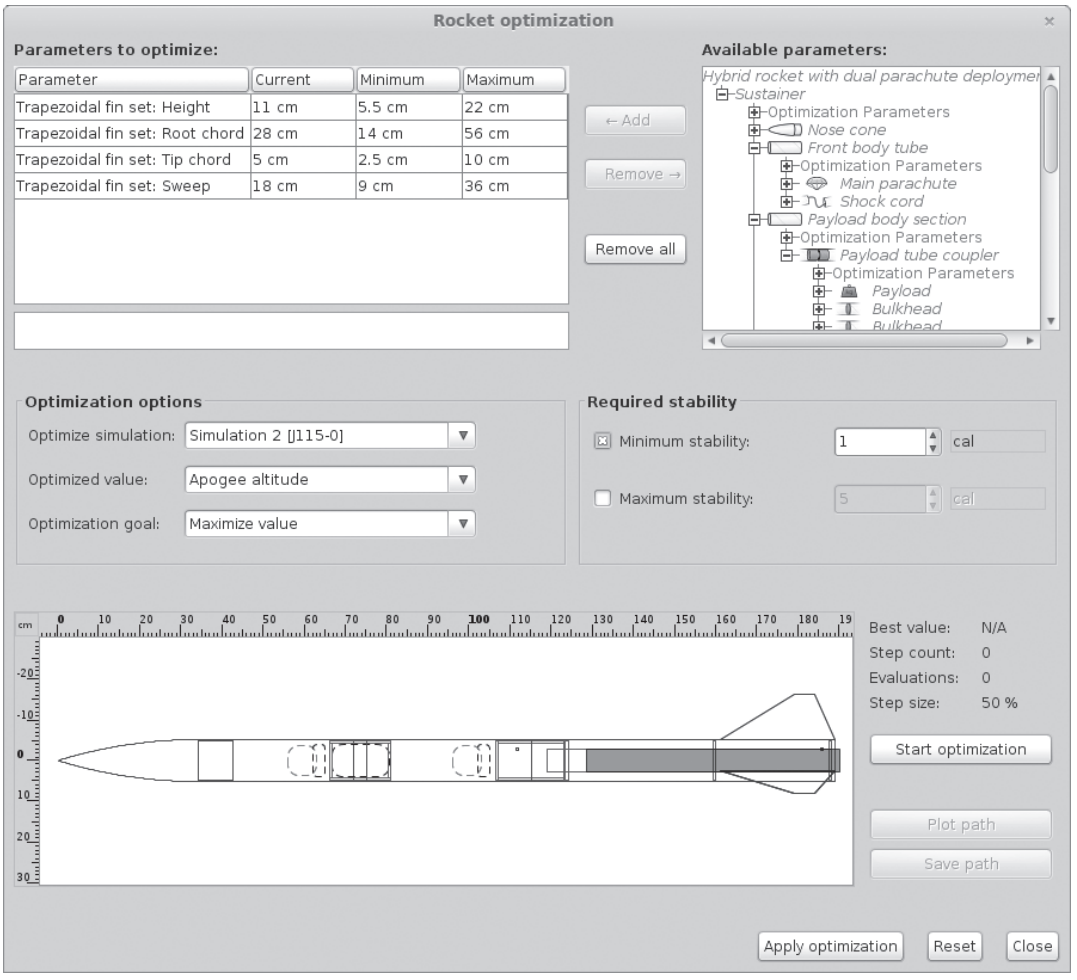
Vaikka lentokorkeuden optimointi massan suhteen on selkeä ja ehkä yleisin käyttötapaus, vaikuttaa monet muut asiat raketissa usein enemmän kuin pelkkä massa. Matemaattisena henkilönä rajoitukset tiettyihin muuttujiin tuntui hölmöltä, ja halusin mahdollistaa monenlaisen optimoinnin.

Raketin yleinen optimointi

Yleistetysti optimointitehtävä voidaan määritellä niin, että jotain määrää syötearvoja muuttamalla halutaan jonkin loppuarvo saada mahdollisimman suureksi, pieneksi tai lähelle jotain arvoa. Lisäksi ratkaisulle voidaan asettaa jotain reunaehtoja. Näin lähestyin ongelmaa OpenRocketissakin. Käyttäjä voisi määritellä muutettavat parametrit raketissa sekä optimointitavoitteen. Reunaehtoina voi määritellä minimi- ja maksimimäärät raketin vakaudelle.

Jokainen muutettava parametri, optimointiarvo ja reunaehto on toteutettu erillisenä plugin-tyyppisenä koodina, joka muokkaa tai laskee raketista jotain arvoja. Näin ollen niitä pystyy helposti kirjoittamaan lisää. Kuva 1 näyttää OpenRocketin optimointi-ikkunan, jossa Haisunäätä-raketin siivekkeen muotoa optimoitaisiin lentokorkeuden suhteen. Tämä on neliulotteinen optimointitehtävä — optimoinnin kohteena on siivekkeen juuren pituus, kärjen pituus, korkeus ja kulma.

Matemaattisesti minkä hyvänsä optimointitehtävän voi muuntaa tehtäväksi funktion minimiarvon etsimisestä. Funktion maksimiarvon etsintä on sama kuin funktion vastaluvun minimin etsintä. Määrätyn arvon tavoittelu puolestaan on sama kuin kyseisestä arvosta etäisyyden minimointi.



Kuva 1: OpenRocketin optimointi-ikkuna ja Haisunäätä-raketti.

Reunaehdot pystytään myös muotoilemaan minimoimistehtävään. Jos jokin reunaehto ei täyty, mitataan ”etäisyys” reunaehdon täyttymisestä, eli miten paljon reunaehtoa rikotaan. Funktioon lisätään sitten jokin niin iso sakkofunktio, että se dominoi optimoitavaa funktiota. Tällöin optimointi ensisijaisesti etsii reunaehdot täyttävää aluetta, ja vasta tämän jälkeen itse funktion minimiarvoa.

Yksiulotteinen optimointi

Yksiulotteisessa optimoinnissa muuttuvia parametreja on vain yksi. Optimointiin on olemassa lukemattomia eri menetelmiä, joilla on erilaisia vahvuuksia. Monet hyödyntävät funktion derivaattoja tai jatkuvuutta minimin etsinnässä. Jos optimoitava funktio on esimerkiksi raketin lentokorkeus, ei derivaattojen laskeminen ole erityisen helppoa, eikä funktio ole välttämättä jatkuva tai tasainen. Niinpä valitsin menetelmän, joka toimii puhtaasti yksittäisiä arvoja laskemalla ja niitä toisiinsa vertailemalla.

Kultaisen leikkauksen menetelmä on tällainen [1]. Siinä hakualueen sisältä lasketaan jokaisella askeleella yksi piste lisää, minkä perusteella hakualuetta rajataan pienemmäksi. Pisteiden sijainti on valittu niin, että aiemmin lasketut pisteet toimivat lähtöpisteinä seuraavassa askeleessa. Kun hakualue on saatu rajattua riittävän pieneksi, on optimointi valmis.

On tärkeää huomata, että lähestulkoon kaikki optimointialgoritmit olettavat, että optimoitava funktio on konveksi, eli että sillä on määrätty minimikohta, jonka kummankin puolen funktio ainoastaan kasvaa. Kuva 2 esittää funktion, jolla on kaksi (lokaalia) minimiä. Jos optimointi aloitetaan koko funtion alueelta, on mahdollista, että algoritmi päätyy vasemmanpuoleiseen lokaaliin minimiin, eikä globaaliin minimiin. Jos taas hakualue rajataan merkittyyn konveksiin alueeseen, on globaalin minimin löytäminen taattua.

Moniulotteinen optimointi

Yksi yleinen tapa tehdä moniulotteista optimointia on käydä eri ulottuvuuksia vuorotellen läpi, ja optimoida vuorotellen kutakin akselia pitkin yksiulotteisena optimointitehtävänä jotakin soveltuvaa algoritmia käyttäen. Jos kokonaisfunktio on konveksi, löytää algoritmi lopulta globaalin minimin.

Tietokoneet sisältävät nykyään kuitenkin vähintään kaksi, usein neljä prosessoria, joilla pystyy laskemaan yhtaikaa. Useimmat optimointialgoritmit eivät pysty hyödyntämään näitä ylimääräisiä prosessoreita mitenkään. Tämän vuoksi etsin algoritmia, joka toimisi rinnakkaistetusti ja tukeutuisi ainoastaan funktion yksittäisiin arvoihin. Löysin tällaisen algoritmin Dennisin ja Torczonin kehittämänä [2]. Algoritmi lähtee määrätystä hakualueen määrittävästä geometrisestä muodosta, jota se alkaa rajata. Algoritmi pystyy ennustamaan, missä pisteissä funktion arvoa todennäköisesti tullaan tarvitsemaan, joten näitä pystytään laskemaan ennakkoon rinnakkain. Tämä nopeuttaa optimointia moniytimisillä tietokoneilla.

Kaikkiaan yleinen optimointityökalu on todella tehokas työkalu, mutta sen kanssa pitää olla myös varovainen. Sen voi liian helposti ajatella tekevän kaiken työn puolestasi. Kuitenkin jos sille antaa liikaa vapausasteita, se helposti tuottaakin järjettömiä tuloksia johtuen esimerkiksi rajallisesta mallinnustarkkuudesta. Käyttäjän pitää osata asettaa rajoituksia ja nähdä mikä on järkevä lopputulos ja mikä ei.

Jos optimointi yleisesti kiinnostaa, suosittelen lämpimästi tutustumaan kirjaan Nonlinear programming [1], joka löytyy myös SATS:n kirjastosta. □

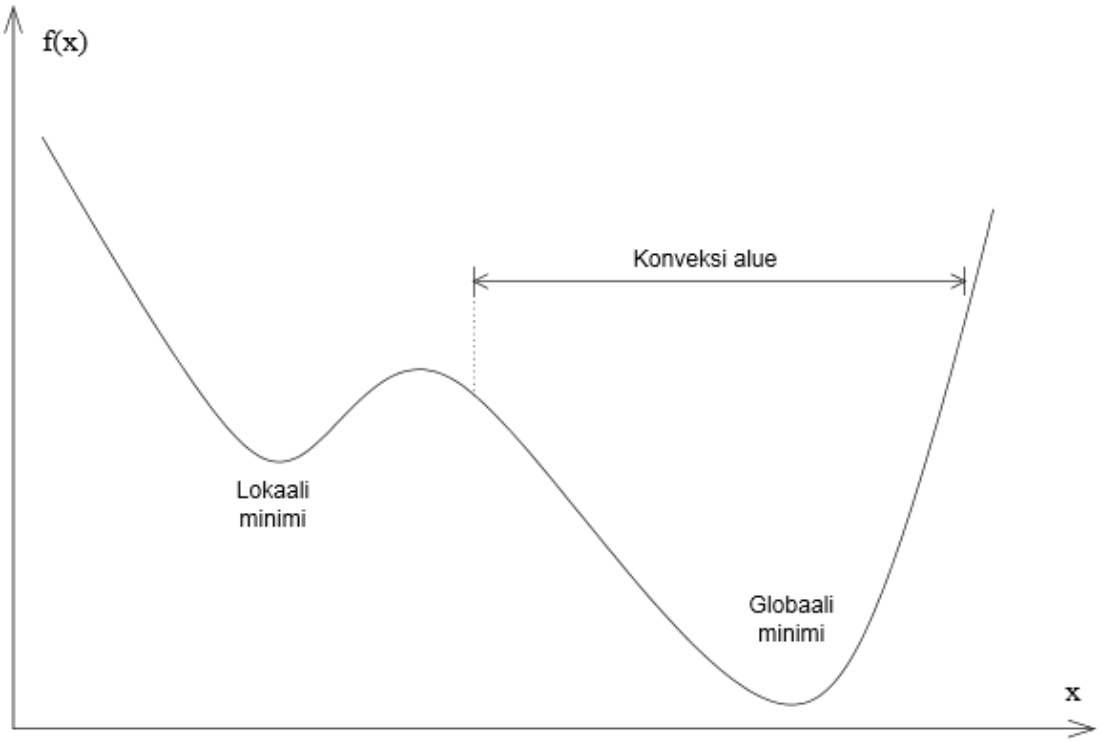
Viitteet:

[1] Nonlinear programming: Theory and Algorithms, 2nd ed., Bazaraa, Sherall, Shetty, 1993, s. 270.

[2] Direct search methods on parallel machines, Dennis, Torczon, SIAM Journal on Optimization, Vol. 1, Issue 4, 1991, s. 448-474.

Artikkelisarjan aiemmissa osissa olen kertonut OpenRocket-ohjelmiston kehitysvaiheista ja raketin aerodynaamisten ominaisuuksien laskennasta. Myöhemmissä osissa on luvassa tietoa mm. tulevaisuuden suunnitelmista.

OpenRocket-ohjelmiston sekä diplomityön saa ladattua osoitteesta http://openrocket.sourceforge.net/ Artikkelisarjan aiemmat osat on luettavissa SATS:n sivuilta osoitteesta http://www.sats-saff.fi/



Kuva 2: Optimoitavan funktion terminologiaa.