Rakenteiden Mekaniikka
Vol. 43, Nro 2, 2010, s. 94–126

Elementtimenetelmän lineaarisen yhtälösysteemin iteratiivisesta ratkaisusta

Juha Hartikainen ja Reijo Kouhia

Tiivistelmä. Artikkelissa käsitellään elementtimenetelmän h-versiossa syntyvän lineaarisen yhtälösysteemin iteratiivista ratkaisua. Tavanomaisimmat käytössä olevat Krylovin aliavaruusiteraatioihin perustuvat ratkaisualgoritmit esitetetään ja niiden valintaan liittyviä kysymyksiä pohditaan. Myös tavanomaisimmat pohjustinmenetelmät esitetään. Menetelmiä on vertailtu muutaman esimerkin avulla.

Avainsanat: Krylovin aliavaruus, iteraatio, gradienttimenetelmä, pohjustin, harva matriisi

Johdanto

Elementtimenetelmän h-versio on yksi yleisimmistä osittaisdifferentiaaliyhtälöiden numeerisista ratkaisutekniikoista, jossa diskretointi johtaa algebralliseen yhtälösysteemiin, jonka Jacobin matriisi on harva. Matriisi määritellään harvaksi, kun sen nollasta eroavien alkioiden lukumäärä on lineaarisesti verrannollinen yhtälöiden lukumäärään. Mallien kokojen kasvaessa suorien ratkaisijoiden vaatima muistitilan tarve ylittää helposti tietokoneen kapasiteetin tai ratkaisuajat kasvavat kohtuuttoman pitkiksi. Lineaarisen algebrallisen yhtälösysteemin iteratiiviset ratkaisijat ovat ideaalisia tälläisten harvojen systeemien ratkaisussa. Muita iteratiivisten ratkaisijoiden etuja suhteessa suoriin ratkaisijoihin on lueteltu seuraavassa [59, luku 1.2], [61, luku 1.1].

1. Systeemin koko kerroinmatriisia ei valttämättä tarvitse muodostaa. Elementtimenetelmässä matriisin ja vektorin kertolasku voidaan suorittaa elementtikohtaisesti.
2. Ratkaisusta saattaa olla jotain etukäteistietoa, jota voidaan hyödyntää iteraation aloitusvektorin muodostamisessa.
3. Iteraatio voidaan lopettaa, kunnes haluttu tarkkuus on saavutettu. Haluttu tarkkuustaso voidaan valita samalle tasolle elementtimenetelmäratkaisun diskretointivirheen kanssa.

On olemassa kuitenkin seikkoja, jotka suosivat suoria ratkaisijoita. Suorat menetelmät ovat luotettavuudessaan ylivoimaisia iteratiivisiin menetelmiin nähden. Niiden ratkaisuaika on helposti ennakoitavissa eikä se riipu tehtävään liittyvien parametrien valinnasta. Mikäli ratkaistavana on useita kuormitustapauksia, suorat ratkaisijat hyötyvät jo kertaalleen tehdyn hajotelman olemassaolosta. Iteratiivisista ratkaisijoista on myös kehitetty lohkoversioita, jotka iteroivat samanaikaisesti useita ratkaisuvektoreita. Näitä menetelmiä ei tässä kirjoituksessa kuitenkaan käsitellä.

Projektiomenetelmän periaate

Olkoon ratkaistavana lineaarinen algebrallinen yhtälösysteemi

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \tag{1}$$

jossa $\boldsymbol{A}$ on reaalinen $n \times n$ matriisi, $\boldsymbol{b}$ on kuormitusvektori ja $\boldsymbol{x}$ ratkaisuvektori.

Suurin osa käytännössä toimivista suurten lineaaristen systeemien iteratiivisista ratkaisumenetelmistä pohjautuu projektioprosessiin, jossa ratkaisua etsitään jostain aliavaruudesta, jonka ulottuvuus (dimensio) on huomattavasti ratkaistavan ongelman ulottuvuutta pienempi. Yhtälön (1) likiratkaisu $\tilde{\boldsymbol{x}}$ projektiomenetelmällä voidaan periaatteellisesti esittää seuraavasti [50, luku 5]:

$$\text{Etsi } \tilde{\boldsymbol{x}} \in \boldsymbol{x}_0 + \mathcal{K} \text{ siten, että } \boldsymbol{r} = \boldsymbol{b} - \boldsymbol{A}\tilde{\boldsymbol{x}} \perp \mathcal{L}, \tag{2}$$

jossa $\mathcal{K}$ ja $\mathcal{L}$ ovat m-ulotteisia $\mathbb{R}^n$:n aliavaruuksia ja $\boldsymbol{x}_0$ on alkuarvaus. Aliavaruutta $\mathcal{K}$ kutsutaan approksimaatioavaruudeksi tai etsintäavaruudeksi, kun taas aliavaruutta $\mathcal{L}$ nimitetään rajoiteavaruudeksi tai vasemmanpuoleiseksi aliavaruudeksi. Rajoitteita $\boldsymbol{r} \perp \mathcal{L}$ kutsutaan myös Petrovin-Galerkinin ehdoiksi.

Merkitään $\boldsymbol{V}$:llä $n \times m$ matriisia, jonka pystyvektorit $\boldsymbol{v}$ muodostavat aliavaruuden $\mathcal{K}$ kannan, ja vastaavasti $\boldsymbol{W}$:llä $n \times m$ matriisia, jonka pystyvektorit $\boldsymbol{w}$ muodostavat aliavaruuden $\mathcal{L}$ kannan. Tällöin ratkaisuyrite voidaan kirjoittaa muodossa

$$\tilde{\boldsymbol{x}} = \boldsymbol{x}_0 + \boldsymbol{V}\boldsymbol{y}, \tag{3}$$

jossa $\boldsymbol{y}$ on m-ulotteinen vektori. Petrovin-Galerkinin kohtisuoruusehdon soveltaminen yhtälöön (3) tuottaa $\boldsymbol{y}$:n ratkaisemiseksi yhtälösysteemin

$$\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V}\boldsymbol{y} = \boldsymbol{W}^T\boldsymbol{r}_0, \quad \text{jossa} \quad \boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0. \tag{4}$$

Mikäli $m \times m$ matriisi $\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V}$ on säännöllinen, saadaan ratkaisuyritteelle muoto

$$\tilde{\boldsymbol{x}} = \boldsymbol{x}_0 + \boldsymbol{V}(\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V})^{-1}\boldsymbol{W}^T\boldsymbol{r}_0. \tag{5}$$

Ratkaisun olemassaolo eli, että matriisi $\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{V}$ on säännöllinen, mielivaltaisille $\mathcal{K}$:n ja $\mathcal{L}$:n kannoille $\boldsymbol{V}$ ja $\boldsymbol{W}$ on taattu, mikäli [50, propositio 5.1]

1. $\boldsymbol{A}$ on positiivisesti definiitti ja $\mathcal{L} = \mathcal{K}$ tai
2. $\boldsymbol{A}$ on säännöllinen ja $\mathcal{L} = \boldsymbol{A}\mathcal{K}$.

Luokkaan 1 kuuluvia menetelmiä, joissa aliavaruudet $\mathcal{K}$ ja $\mathcal{L}$ ovat identtiset, kutsutaan ortogonaaliprojektiomenetelmiksi ja vastaavia rajoiteyhtälöitä Galerkinin ehdoiksi.

Havainnollistetaan iteraatioprosessia yksiulotteisten aliavaruuksien

$$\mathcal{K} = \text{span}\{\boldsymbol{v}\} \quad \text{ja} \quad \mathcal{L} = \text{span}\{\boldsymbol{w}\} \tag{6}$$

avulla. Tällöin uusi ratkaisuyrite on muotoa $\boldsymbol{x}_1 = \boldsymbol{x}_0 + \alpha\boldsymbol{v}$, jossa kertoimelle α Petrovin-Galerkinin ehto antaa ratkaisun

$$\alpha = \frac{\boldsymbol{w}^T\boldsymbol{r}_0}{\boldsymbol{w}^T\boldsymbol{A}\boldsymbol{v}}. \tag{7}$$

Tarkastellaan seuraavaksi matriisin $\boldsymbol{A}$ ominaisuuksista riippuen kolmea eri tapausta.

1. $\boldsymbol{A}$ on symmetrinen ja positiivisesti definiitti (SPD). Mikäli jokaisella askeleella valitaan $\boldsymbol{v} = \boldsymbol{r}$ ja $\boldsymbol{w} = \boldsymbol{r}$, saadaan gradienttimenetelmä, joka minimoi ratkaisuvektorin virheen $\boldsymbol{A}$-painotetun normin, n.s. energianormin avulla lausutun funktion

$$f(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}_*\|_A^2 = (\boldsymbol{x} - \boldsymbol{x}_*)^T \boldsymbol{A}(\boldsymbol{x} - \boldsymbol{x}_*) \tag{8}$$

jossa $\boldsymbol{x}_*$ on tarkka ratkaisu. Voidaan osoittaa, että algoritmi suppenee kohti tarkkaa ratkaisua lähdettäessä mielivaltaisesta aloitusvektorista. Virhevektorin $\boldsymbol{e}_k = \boldsymbol{x}_k - \boldsymbol{x}_*$ pienenemiselle voidaan $\boldsymbol{A}$-normissa mitattuna johtaa arvio [32, teoreema 2.2.2], [50, teor. 5.2]

$$\|\boldsymbol{e}_{k+1}\|_A \leq \frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}} \|\boldsymbol{e}_k\|_A = \frac{\kappa - 1}{\kappa + 1} \|\boldsymbol{e}_k\|_A, \tag{9}$$

jossa $\lambda_{\max}, \lambda_{\min}$ ovat matriisin $\boldsymbol{A}$ suurin ja pienin ominaisarvo ja $\kappa = \lambda_{\max}/\lambda_{\min}$ sen häiriöalttius.

2. $\boldsymbol{A}$ on vain positiivisesti definiitti. Mikäli $\boldsymbol{A}$ ei ole symmetrinen, mutta sen symmetrinen osa on positiivisesti definiitti, saadaan valinnalla $\boldsymbol{v} = \boldsymbol{r}$ ja $\boldsymbol{w} = \boldsymbol{A}\boldsymbol{r}$ iteraatio, jossa jokainen iteraatioaskel minimoi jäännösvektorin Euklidisen normin avulla lausutun funktion

$$f(\boldsymbol{x}) = \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|_2^2 \tag{10}$$

jäännöksen $\boldsymbol{r}$ suunnassa. Jäännösvektorin pienenemiselle voidaan johtaa arvio [32, teor. 2.2.2], [50, teor. 5.3]

$$\|\boldsymbol{r}_{k+1}\|_2 \leq \sqrt{1 - (\lambda_{\min}^{\mathrm{s}}/\sigma)^2} \|\boldsymbol{r}_k\|_2, \tag{11}$$

jossa $\lambda_{\min}^{\mathrm{s}}$ on $\boldsymbol{A}$:n symmetrisen osan pienin ominaisarvo ja $\sigma = \|\boldsymbol{A}\|_2$.

3. $\boldsymbol{A}$ on säännöllinen. Mikäli otaksutaan matriisin $\boldsymbol{A}$ olevan vain ei-singulaarinen, valinnat $\boldsymbol{v} = \boldsymbol{A}^T\boldsymbol{r}$ ja $\boldsymbol{w} = \boldsymbol{A}\boldsymbol{v}$ minimoivat jäännöksen (10) f:n gradientin vastakkaissuunnassa. Menettely on myös ekvivalentti gradienttimenetelmälle, kun sitä sovelletaan normaaliyhtälöön

$$\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b}. \tag{12}$$

Jos $\boldsymbol{A}$ on säännöllinen, on $\boldsymbol{A}^T\boldsymbol{A}$ symmetrinen ja positiivisesti definiitti ja iteraatio suppenee.

Krylovin aliavaruusiteraatiot

Krylovn aliavaruus

Tällä hetkellä tärkein menetelmäperhe suurten lineaaristen systeemien ratkaisemiseksi perustuu Krylovin[1] aliavaruuksien käyttöön [3, 50, 61]. m-ulotteisen Krylovin aliavaruuden virittävät vektorit, jotka saadaan rekursiivisesti kertomalla vektori $\boldsymbol{v}$ matriisilla $\boldsymbol{A}$:

$$\mathcal{K}_m(\boldsymbol{A}, \boldsymbol{v}) = \operatorname{span}\left\{\boldsymbol{v}, \boldsymbol{A}\boldsymbol{v}, \boldsymbol{A}^2\boldsymbol{v}, ..., \boldsymbol{A}^{m-1}\boldsymbol{v}\right\}.$$

[1] Alexei Nikolajevits Krylov (15.8.1863–26.10.1945), venäläinen laivanrakennusinsinööri, sovellettu matemaatikko. Hänen ominaisarvotehtävän ratkaisua käsittelevä venäjänkielinen artikkelinsa on vuodelta 1931 (AN SSSR, Otdel. mat. i estest. nauk., 1931, VII, Nr. 4, 491–539). Hänen lisäkseen tunnetaan ainakin kaksi kuuluisaa Krylov nimistä matemaatikkoa: N.M. Krylov (1875–1955) ja V.I. Krylov.

Jatkossa m-ulotteista Krylovin aliavaruutta $\mathcal{K}_m(\boldsymbol{A}, \boldsymbol{r}_0)$, jossa $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$ ja $\boldsymbol{x}_0$ on iteraation aloitusvektori, merkitään lyhyesti vain $\mathcal{K}_m$, mikäli sekaannuksen vaaraa ei ole. Keskeinen ongelma on $\mathcal{K}_m$:n kannan laskeminen. Kanta $\boldsymbol{r}_0, \boldsymbol{A}\boldsymbol{r}_0, \boldsymbol{A}^2\boldsymbol{r}_0, ..., \boldsymbol{A}^{m-1}\boldsymbol{r}_0$ ei ole numeerisesti mielekäs, sillä vektorit $\boldsymbol{A}^k\boldsymbol{r}_0$ osoittavat k:n kasvaessa yhä enemmän $\boldsymbol{A}$:n suurinta ominaisarvoa vastaavan ominaisvektorin suuntaan, jolloin pyöristysvirheiden johdosta niistä tulee lähes lineaarisesti riippuvia.

Rajoiteyhtälön muoto, t.s. avaruuden $\mathcal{L}_m$ valinta vaikuttaa ratkaisevasti iteraatioprosessiin. Yleisimmin käytössä olevat keinot perustuvat (i) ortogonaaliprojektioon $\mathcal{L}_m = \mathcal{K}_m$, (ii) minimijäännösstrategiaan $\mathcal{L}_m = \boldsymbol{A}\mathcal{K}_m$ tai (iii) siihen, että rajoiteavaruus valitaan $\boldsymbol{A}^T$:n avulla muodostetuksi Krylovin aliavaruudeksi $\mathcal{L}_m = \mathcal{K}_m(\boldsymbol{A}^T, \boldsymbol{r}_0)$.

Arnoldin algoritmi

Arnoldin algoritmia voidaan käyttää Krylovin aliavaruuden $\mathcal{K}_m$ ortogonaalisen kannan muodostamiseksi seuraavasti.

1. Valitse aloitusvektori $\boldsymbol{v}_1$ siten, että $\|\boldsymbol{v}_1\|_2 = 1$.
2. Toista, kun $j = 1, 2, \ldots, m$:
   (a) laske $h_{ij} = \boldsymbol{v}_i^T \boldsymbol{A}\boldsymbol{v}_j$, kun $i = 1, 2, \ldots, j$;
   (b) laske $\boldsymbol{s} = \boldsymbol{A}\boldsymbol{v}_j - \sum_{i=1}^{j} h_{ij}\boldsymbol{v}_i$;
   (c) laske $h_{j+1,j} = \|\boldsymbol{s}\|_2$;
   (d) jos $h_{j+1,j} = 0$, lopeta, muutoin jatka;
   (e) laske $\boldsymbol{v}_{j+1} = \boldsymbol{s}/h_{j+1,j}$.

Arnoldin algoritmissa vektorit $\boldsymbol{v}_j$ kerrotaan matriisilla $\boldsymbol{A}$ ja tuloksena syntyvä vektori $\boldsymbol{s}$ ortonormeerataan kaikkien edellisten vektoreiden $\boldsymbol{v}_j$ suhteen tavanomaisella Gramm-Schmidtin menetelmällä. Arnoldin algoritmi muodostaa matriisiin $\boldsymbol{A}$ liittyvän ylemmän $m+1 \times m$ Hessenbergin matriisin $\bar{\boldsymbol{H}}_m$

$$\bar{\boldsymbol{H}}_m = \begin{bmatrix} h_{11} & h_{12} & h_{13} & \cdots & h_{1,m-1} & h_{1,m} \\ h_{21} & h_{22} & h_{23} & \cdots & h_{2,m-1} & h_{2,m} \\ 0 & h_{32} & h_{33} & \cdots & h_{3,m-1} & h_{3m} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & h_{m-1,m} & h_{mm} \\ 0 & 0 & 0 & \cdots & 0 & h_{m+1,m} \end{bmatrix}. \tag{13}$$

Olkoon $\boldsymbol{V}_k = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k]$ Arnoldin algoritmin tuottamien ortonormaalien kantavektoreiden muodostama $n \times k$ matriisi. Tällöin voidaan helposti osoittaa, että

$$\boldsymbol{A}\boldsymbol{V}_m = \boldsymbol{V}_{m+1}\bar{\boldsymbol{H}}_m = \boldsymbol{V}_m\boldsymbol{H}_m + h_{m+1,m}\boldsymbol{v}_{m+1}\boldsymbol{i}_m^T, \tag{14}$$

$$\boldsymbol{V}_m^T\boldsymbol{A}\boldsymbol{V}_m = \boldsymbol{H}_m, \tag{15}$$

jossa $m \times m$ matriisi $\boldsymbol{H}_m$ saadaan matriisista $\bar{\boldsymbol{H}}_m$ poistamalla siitä alin vaakarivi ja vektori $\boldsymbol{i}_m$ on $m \times m$ identiteettimatriisin m:s pystyrivi.

Käytännössä Arnoldin algoritmi toteutetaan käyttäen modifioitua Grammin-Schmidtin menetelmää, Householderin algoritmia tai jotain muuta matemaattisesti ekvivalenttia mutta numeerisesti stabiilimpaa menetelmää. Arnoldin algoritmi käyttäen modifioitua Grammin-Schmidtin ortogonalisointia on seuraava.

1. Valitse aloitusvektori $\boldsymbol{v}_1$ siten, että $\|\boldsymbol{v}_1\|_2 = 1$.
2. Toista, kun $j = 1, 2, \ldots, m$:
   (a) laske $\boldsymbol{s} = \boldsymbol{A}\boldsymbol{v}_j$;

(b) toista, kun $i = 1, \ldots, j$:
- laske $h_{ij} = \boldsymbol{s}^T \boldsymbol{v}_i$,
- päivitä $\boldsymbol{s} = \boldsymbol{s} - h_{ij}\boldsymbol{v}_i$;

(c) laske $h_{j+1,j} = \|\boldsymbol{s}\|_2$;
(d) jos $h_{j+1,j} = 0$, lopeta, muutoin jatka;
(e) laske $\boldsymbol{v}_{j+1} = \boldsymbol{s}/h_{j+1,j}$.

Arnoldin algoritmi esitettiin 1951 ominaisarvotehtävän ratkaisuvaiheena redusoida täysi matriisi Hessenbergin matriisiksi [2]. Hessenbergin matriisin ominaisarvot aproksimoivat hyvin alkuperäisen matriisin ominaisarvoja, vaikka $m \ll n$. Tunnettu ominaisarvoratkaisija ARPACK [41] perustuu Arnoldin algoritmiin.

Arnoldin algoritmia voidaan käyttää hyväksi usealla eri tavalla. Esitetään seuraavassa kaksi vaihtoehtoista tapaa, joista toinen perustuu ortogonaaliprojektioon ja toinen jäännöksen minimointiin.

Täydellinen ortogonalisointialgoritmi – FOM

Ortogonaaliprojektiomenetelmä saadaan, kun valitaan $\mathcal{L} = \mathcal{K} = \mathcal{K}_m(\boldsymbol{A}, \boldsymbol{r}_0)$. Asettamalla $\boldsymbol{v}_1 = \boldsymbol{r}_0/\|\boldsymbol{r}_0\|_2$ ja merkitsemällä $\beta = \|\boldsymbol{r}_0\|_2$ saadaan

$$\boldsymbol{V}_m^T \boldsymbol{r}_0 = \beta\, \boldsymbol{V}_m^T \boldsymbol{v}_1 = \beta \boldsymbol{i}_1. \tag{16}$$

Systeemin (1) ratkaisun approksimaatio m-dimensioisessa aliavaruudessa $\boldsymbol{x}_0 + \mathcal{K}_m(\boldsymbol{A}, \boldsymbol{r}_0)$ on siten

$$\boldsymbol{x}_m = \boldsymbol{x}_0 + \boldsymbol{V}_m \boldsymbol{y}_m = \boldsymbol{x}_0 + \boldsymbol{H}_m^{-1}(\beta \boldsymbol{i}_1). \tag{17}$$

Tämä menetelmä tunnetaan kirjallisuudessa nimellä täydellinen ortogonalisointimenetelmä, josta käytetään lyhennettä FOM (Full Orthogonalization Method).

Iteraation suppenemista mitataan yleensä jäännösvektorin normista, jonka suora laskenta vaatisi matriisin ja vekrorin välisen tulon muodostamisen. Tältä voidaan kuitenkin välttyä, sillä on helppo osoittaa, että

$$\boldsymbol{r}_m = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_m = -h_{m+1,m} \boldsymbol{i}_m^T \boldsymbol{y}_m \boldsymbol{v}_{m+1}, \tag{18}$$

josta seuraa, että

$$\|\boldsymbol{r}_m\|_2 = h_{m+1,m} |\boldsymbol{i}_m^T \boldsymbol{y}_m|. \tag{19}$$

FOM:n vaatima työmäärä on kertaluokkaa $\mathcal{O}(m^2 n)$ ja muistitila luokkaa $\mathcal{O}(mn)$. Mikäli yhtälöryhmän koko on suuri, aliavaruuden kokoa m rajoittaa käytännössä muistitilan määrä. Muistin tarvetta voidaan vähentää, jos iteraatio aloitetaan periodisesti uudelleen tai jos ortogonalisointiprosessi katkaistaan tai suoritetaan epätäydellisesti. Uudelleenaloitettu FOM-iteraatio on periaatteeltaan seuraava.

FOM(m)

1. Muodosta $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$, $\beta = \|\boldsymbol{r}_0\|_2$, $\boldsymbol{v}_1 = \boldsymbol{r}_0/\beta$.
2. Aloittaen vektorista $\boldsymbol{v}_1$ muodosta Arnoldin algoritmilla Hessenbergin matriisi $\boldsymbol{H}_m$.
3. Ratkaise $\boldsymbol{y}_m = \boldsymbol{H}_m^{-1}\beta \boldsymbol{i}_1$ ja laske $\boldsymbol{x}_m = \boldsymbol{x}_0 + \boldsymbol{V}_m \boldsymbol{y}_m$. Lopeta, mikäli jäännös (19) on riittävän pieni. Muulloin jatka.
4. Aseta $\boldsymbol{x}_0 = \boldsymbol{x}_m$ ja palaa kohtaan 1.

Katkaistuja ja epätäydellisiä ortogonalisointialgoritmeja ei tässä esityksessä käsitellä.

Yleistetty pienimmän jäännöksen menetelmä – GMRES

Yksi tunnetuimmista ja käytetyimmistä epäsymmetristen lineaaristen yhtälösysteemien iteratiivisista ratkaisumenetelmistä on Saadin ja Schultzin 1986 esittämä yleistetty pienimmän jäännöksen menetelmä – GMRES (Generalized Minimal RESidual). Kuten FOM-menetelmä, GMRES etsii ratkaisua aliavaruudesta $\boldsymbol{x}_0 + \mathcal{K}_m(\boldsymbol{A}, \boldsymbol{r}_0)$. Tämän aliavaruuden mielivaltainen n-ulotteinen vektori $\boldsymbol{x}$ voidaan kirjoittaa m-ulotteisen vektorin $\boldsymbol{y}$ avulla muodossa

$$\boldsymbol{x} = \boldsymbol{x}_0 + \boldsymbol{V}_m \boldsymbol{y}, \tag{20}$$

ja jäännös eli residuaalivektori muodossa

$$\begin{aligned} \boldsymbol{r} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x} &= \boldsymbol{b} - \boldsymbol{A}(\boldsymbol{x}_0 + \boldsymbol{V}_m \boldsymbol{y}) = \boldsymbol{r}_0 - \boldsymbol{A}\boldsymbol{V}_m \boldsymbol{y} \\ &= \beta \boldsymbol{v}_1 - \boldsymbol{V}_{m+1} \bar{\boldsymbol{H}}_m \boldsymbol{y} = \boldsymbol{V}_{m+1}(\beta \boldsymbol{i}_1 - \bar{\boldsymbol{H}}_m \boldsymbol{y}). \end{aligned} \tag{21}$$

Koska $\boldsymbol{V}_{m+1}$:n pystyvektorit ovat ortonormaaleja, saadaan jäännöksen normille lauseke

$$f(\boldsymbol{y}) \equiv \|\boldsymbol{b} - \boldsymbol{A}(\boldsymbol{x}_0 + \boldsymbol{V}_m \boldsymbol{y})\|_2 = \|\beta \boldsymbol{i}_1 - \bar{\boldsymbol{H}}_m \boldsymbol{y}\|_2. \tag{22}$$

GMRES-approksimaatio m-ulotteisessa Krylovin aliavaruudessa $\boldsymbol{x}_0 + \mathcal{K}_m$ on siten

$$\boldsymbol{x}_m = \boldsymbol{x}_0 + \boldsymbol{V}_m \boldsymbol{y}_m, \tag{23}$$

jossa $\boldsymbol{y}_m$ minimoi yhtälössä (22) määritellyn funktion f.

GMRES-menetelmän muistitilan tarve ja laskentatyön määrä ovat samaa luokkaa FOM-menetelmän kanssa. Käytännössä iteraatio joudutaan uudelleenaloittamaan, jolloin saadaan GMRES(m)-algoritmi.

GMRES(m)

1. Muodosta $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$, $\beta = \|\boldsymbol{r}_0\|_2$, $\boldsymbol{v}_1 = \boldsymbol{r}_0/\beta$.
2. Aloittaen vektorista $\boldsymbol{v}_1$ muodosta Arnoldin algoritmilla $(m+1) \times m$ Hessenbergin matriisi $\bar{\boldsymbol{H}}_m$.
3. Ratkaise $\boldsymbol{y}_m = \text{argmin}\|\beta \boldsymbol{i}_1 - \bar{\boldsymbol{H}}_m \boldsymbol{y}\|_2$ ja laske $\boldsymbol{x}_m = \boldsymbol{x}_0 + \boldsymbol{V}_m \boldsymbol{y}_m$. Lopeta, mikäli jäännös on riittävän pieni. Muulloin jatka.
4. Aseta $\boldsymbol{x}_0 = \boldsymbol{x}_m$ ja palaa kohtaan 1.

Lanczosin menetelmä

Arnoldin algoritmi perustuu matriisin muuntamiseen Hessenbergin matriisiksi. Lanczosin menetelmässä matriisi redusoidaan tridiagonaalimatriisiksi. Tarkastellaan kuitenkin ensin symmetristä tapausta, jolloin Arnoldin algoritmin Hessenbergin matriisista $\boldsymbol{H}_m$ tulee tridiagonaalinen

$$\boldsymbol{T}_m = \boldsymbol{H}_m = \begin{bmatrix} \alpha_1 & \beta_2 & 0 & \cdots & 0 & 0 \\ \beta_2 & \alpha_2 & \beta_3 & \cdots & 0 & 0 \\ 0 & \beta_3 & \alpha_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \alpha_{m-1} & \beta_m \\ 0 & 0 & 0 & \cdots & \beta_m & \alpha_m \end{bmatrix}, \tag{24}$$

jossa on käytetty tavanomaista merkintää $\alpha_i = h_{ii}$, $\beta_i = h_{i-1,i}$. Lanczosin algoritmi kannan $\boldsymbol{v}_i$ ja tridiagonaalimatriisin $\boldsymbol{T}_m$ muodostamiseksi käytettäessä modifioitua Gramm-Schmidtin ortogonalisointia on seuraava.

1. Valitse aloitusvektori $\boldsymbol{v}_1$, $\quad \|\boldsymbol{v}_1\|_2 = 1$, aseta $\beta_0 = 0$ ja $\boldsymbol{v}_0 = \boldsymbol{0}$.
2. Toista $j = 1, 2, \ldots, m$;
   (a) laske $\boldsymbol{s} = \boldsymbol{A}\boldsymbol{v}_j - \beta_j \boldsymbol{v}_{j-1}$;
   (b) laske $\alpha_j = \boldsymbol{s}^T \boldsymbol{v}_j$;
   (c) päivitä $\boldsymbol{s} = \boldsymbol{s} - \alpha_j \boldsymbol{v}_j$;
   (d) laske $\beta_j = \|\boldsymbol{s}\|_2$; jos $\beta_j = 0$, lopeta;
   (e) laske $\boldsymbol{v}_{j+1} = \boldsymbol{s}/\beta_j$.

Lanczosin menetelmä symmetrisen lineaarisen systeemin $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ ratkaisuun voidaan siten periaatteessa esittää muodossa

$$\boldsymbol{x}_m = \boldsymbol{x}_0 + \boldsymbol{V}_m \boldsymbol{y}_m, \quad \boldsymbol{y}_m = \boldsymbol{T}_m^{-1}(\beta \boldsymbol{i}_1), \tag{25}$$

jossa $\boldsymbol{T}_m$ on tridiagonaalimatriisi $\boldsymbol{T}_m = \text{tridiag}(\beta_i, \alpha_i, \beta_{i+1})$ ja kanta $\boldsymbol{V}_m = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m]$ on muodostettu lähtien vektorista $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$ ja sen normista $\beta = \|\boldsymbol{r}_0\|_2$.

Liittogradienttimenetelmä

Liittogradientti- eli konjugaattigradienttimenetelmä (CG = Conjugate Gradient) on yksi tunnetuimmista iteratiivisista menetelmistä symmetrisen ja positiivisesti definiitin lineaarisen systeemin ratkaisemiseksi. Se on ortogonaaliprojektiomenetelmä, jossa ratkaisua etsitään Krylovin aliavaruudesta $\mathcal{K}_m(\boldsymbol{A}, \boldsymbol{r}_0)$ vaatimalla jäännöksen ortogonaalisuus tämän saman avaruuden suhteen. Täten uusi jäännös $\boldsymbol{r}_{i+1} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_{i+1}$ on ortogonaalinen Lanczosin prosessin generoimien vektoreiden $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_i$ virittämän aliavaruuden suhteen. Täten $\boldsymbol{r}_{i+1}$ on yhdensuuntainen vektorin $\boldsymbol{v}_{i+1}$ kanssa. Liittogradienttimenetelmä on siten läheistä sukua symmetriselle Lanczosin menetelmälle. Lähteessä [23, luku 11] liittogradienttimenetelmä on johdettu Lanczosin algoritmin avulla. Esitetään tässä menetelmän klassinen kehittely ilman todistuksia.

Hakusuunnan $\boldsymbol{d}_i$ avulla ratkaisuvektori voidaan kirjoittaa muodossa

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + a_i \boldsymbol{d}_i. \tag{26}$$

Lasketaan sitten uusi jäännösvektori

$$\boldsymbol{r}_{i+1} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_{i+1} = \boldsymbol{r}_i - a_i \boldsymbol{A}\boldsymbol{d}_i. \tag{27}$$

Koska jäännösvektorit ovat ortogonaalisia, on oltava

$$\boldsymbol{r}_i^T \boldsymbol{r}_{i+1} = 0, \tag{28}$$

jolloin kertoimelle a_i saadaan

$$a_i = \frac{\boldsymbol{r}_i^T \boldsymbol{r}_i}{\boldsymbol{r}_i^T \boldsymbol{A}\boldsymbol{d}_i}. \tag{29}$$

Uusi hakusuunta $\boldsymbol{d}_{i+1}$ on lineaarikombinaatio vektoreista $\boldsymbol{r}_{i+1}$ ja $\boldsymbol{d}_i$, joten se voidaan kirjoittaa muodossa

$$\boldsymbol{d}_{i+1} = b_i \boldsymbol{d}_i + \boldsymbol{r}_{i+1}. \tag{30}$$

Hakusuunnat konjugoivat, eli $\boldsymbol{d}_i^T \boldsymbol{A}\boldsymbol{d}_j = 0$, jos $i \neq j$. Täten kertoimelle b_i saadaan lauseke

$$b_i = -\frac{\boldsymbol{r}_{i+1}^T \boldsymbol{A}\boldsymbol{d}_i}{\boldsymbol{d}_i^T \boldsymbol{A}\boldsymbol{d}_i}. \tag{31}$$

Kertoimien a_i ja b_i laskemisessa voi välttää kahden pistetulon muodostamisen, sillä havaitsemalla yhteys

$$\boldsymbol{r}_i^T \boldsymbol{A}\boldsymbol{d}_i = (\boldsymbol{d}_i - b_{i-1}\boldsymbol{d}_{i-1})^T \boldsymbol{A}\boldsymbol{d}_i = \boldsymbol{d}_i^T \boldsymbol{A}\boldsymbol{d}_i \tag{32}$$

ja, että yhtälön (27) nojalla $\boldsymbol{A}\boldsymbol{d}_i = -a_i^{-1}(\boldsymbol{r}_{i+1} - \boldsymbol{r}_i)$, saadaan

$$b_i = \frac{\boldsymbol{r}_{i+1}^T(\boldsymbol{r}_{i+1} - \boldsymbol{r}_i)}{a_i \boldsymbol{d}_i^T \boldsymbol{A}\boldsymbol{d}_i} = \frac{\boldsymbol{r}_{i+1}^T \boldsymbol{r}_{i+1}}{\boldsymbol{r}_i^T \boldsymbol{r}_i}. \tag{33}$$

Lopuksi algoritmi voidaan kirjoittaa seuraavassa muodossa.

Liittogradienttimenetelmä (CG)

1. Laske $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$, aseta $\boldsymbol{d}_0 = \boldsymbol{r}_0$ ja laske $\tau_0 = \boldsymbol{r}_0^T \boldsymbol{d}_0$.
2. Iteroi $i = 0, 1, 2, ...$, kunnes menetelmä suppenee:
   (a) laske: $\boldsymbol{s} = \boldsymbol{A}\boldsymbol{d}_i, \quad a_i = \tau_i / \boldsymbol{d}_i^T \boldsymbol{s}$;
   (b) päivitä: $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + a_i \boldsymbol{d}_i, \quad \boldsymbol{r}_{i+1} = \boldsymbol{r}_i - a_i \boldsymbol{s}$;
   (c) laske: $\tau_{i+1} = \boldsymbol{r}_{i+1}^T \boldsymbol{r}_{i+1}, \quad b_i = \tau_{i+1}/\tau_i$;
   (d) päivitä: $\boldsymbol{d}_{i+1} = b_i \boldsymbol{d}_i + \boldsymbol{r}_{i+1}$.

Jokaisella iteraatioaskeleella on suoritettava yksi matriisin ja vektorin kertolasku, muodostettava kaksi pistetuloa ja kolme vektorin päivitysoperaatiota eli n.s. axpy-operaatiota (skalaari a kertaa vektori x + vektori y). Suppenemista voidaan seurata residuaalivektorin normin avulla. Iteraatio voidaan lopettaa, kun

$$\sqrt{\tau_i} \leq \epsilon_{\text{rel}} \sqrt{\tau_0} + \epsilon_{\text{abs}}, \tag{34}$$

jossa ϵ_{rel} ja ϵ_{abs} ovat iteraation suhteellinen ja absoluuttinen suppenemistoleranssi.

Liittogradienttimenetelmän suppenemisnopeudelle voidaan johtaa lauseke

$$\|\boldsymbol{x}_k - \boldsymbol{x}_*\|_A = \|\boldsymbol{e}_k\|_A \leq 2 \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^k \|\boldsymbol{e}_0\|_A, \tag{35}$$

jossa κ on matriisin $\boldsymbol{A}$ häiriöalttius. Huomaa myös, että $\|\boldsymbol{e}_k\|_A = \|\boldsymbol{r}_k\|_{A^{-1}}$. Residuaalin Euklidiselle normille ja virheen energianormille voidaan lisäksi johtaa relaatio

$$\frac{\|\boldsymbol{r}_k\|_2}{\|\boldsymbol{r}_0\|_2} \leq \frac{1}{\sqrt{\kappa}} \frac{\|\boldsymbol{e}_k\|_A}{\|\boldsymbol{e}_0\|_A}. \tag{36}$$

Mikäli matriisin häiriöalttius on suuri, jäännöksen normiin pohjautuva iteraation lopetusehto voi olla epäluotettava ja johtaa liian aikaiseen iteraation lopetuspäätökseen. Luotettavassa lopetusehdossa tulisi myös tutkia ratkaisuvektorin $\boldsymbol{x}$ muutosta.

Arvio (35) on varsin pessimistinen. Mikäli matriisin häiriöalttiuden lisäksi tiedetään jotain sen ominaisarvospektristä, arviota voidaan parantaa. Menetelmän suppenemista ja käyttäytymistä laskettaessa äärellisellä tarkkuudella on käsitelty esim. lähteissä [32, luvut 3 ja 4] sekä [45, luku 6.4]. Haluttaessa ratkaisun jäännöksen suhteellisen virheen olevan luokkaa ϵ, saadaan yhtälöstä (35) yläraja-arvio iteraatioiden lukumäärälle

$$k \sim \frac{1}{2}\sqrt{\kappa} \ln \frac{2}{\epsilon}. \tag{37}$$

Liittogradienttimenetelmälle voidaan johtaa useita erilaisia versioita, jotka ovat matemaattisesti ekvivalentteja, mutta voivat numeerisesti käyttäytyä hieman eri tavalla.

Liittogradienttimenetelmä on täysin luotettava vain mikäli $\boldsymbol{A}$ on SPD-matriisi. Kuten edellä esitetystä algoritmista havaitaan, voidaan sitä soveltaa myös indefiniittisiin tapauksiin. Tällöin on kuitenkin vaarana numeerinen epästabiilius, sillä kohdassa 2(a) parametrin a_i laskennassa voi nimittäjä olla miltei nolla. Symmetrisille indefiniiteille tapauksille on suositeltavaa käyttää luotettavampia menetelmiä kuten MINRES tai SYMMLQ [46]. Näitä algoritmejä ei kuitenkaan esitetä tässä kirjoituksessa.

Liittogradienttimenetelmän esittivät toisistaan riippumatta 1952 Lanczos [40] sekä Hestenes ja Stiefel [35]. Tällöin menetelmää pidettiin eräänä suorista ratkaisutekniikoista. Iteraatiovektoreiden ortogonaalisuuden häviäminen laskettaessa rajallisella tarkuudella havaittiin pian ja menetelmä hylättiin kahdeksi vuosikymmeneksi, kunnes 1970-luvun alussa havaittiin, että ortogonaalisuuden katoaminen ei estä menetelmän konvergenssiä [48]. Tästä alkoi menetelmään kohdistuva yhä lisääntyvä kiinnostus ja käyttö suurten lineaaristen systeemien ratkaisussa.

Kaksipuolinen Lanczosin menetelmä

Lanczosin menetelmä redusoi symmetrisen matriisin symmetriseksi tridiagonaalimatriisiksi ja generoi Krylovin aliavaruuden $\mathcal{K}_m(\boldsymbol{A}, \boldsymbol{v}_0)$ kantavektorit $\boldsymbol{v}_i$. Kaksipuolinen Lanczosin menetelmä redusoi epäsymmetrisen matriisin tridiagonaalimatriisiksi ja generoi edellämainitun aliavaruuden kantavektoreiden lisäksi avaruuden $\mathcal{K}_m(\boldsymbol{A}^T, \boldsymbol{v}_0)$ kantavektorit $\boldsymbol{w}_i$, jotka ovat toistensa suhteen bi-ortogonalisia, eli

$$\boldsymbol{w}_i^T \boldsymbol{v}_j = \delta_{ij}, \tag{38}$$

jossa δ_{ij} on Kroneckerin symboli. Lanczosin menetelmä kolmitermisen iteraatiokaavan vektoreiden $\boldsymbol{v}_i$ ja $\boldsymbol{w}_j$ sekä tridiagonaalimatrisiisin $\boldsymbol{T}$ muodostamiseksi ja on siten työmäärältään sekä muistitilan tarpeen kannalta optimaalinen. Bi-ortogonaalisuuden vuoksi se on myös luonteeltaan erilainen kuin Arnoldin algoritmi. Algoritmina Lanczosin bi-ortogonalisointimenetelmä on seuraava.

1. Valitse aloitusvektorit $\tilde{\boldsymbol{v}}_1, \tilde{\boldsymbol{w}}_1 \neq \boldsymbol{0}$ ja aseta $\boldsymbol{v}_0 = \boldsymbol{w}_0 = \boldsymbol{0}$.
2. Toista, kun $j = 1, 2, \ldots, m$:
   (a) laske $\rho_j = \tilde{\boldsymbol{w}}_j^T \tilde{\boldsymbol{v}}_j$, jos $\rho_j = 0$ lopeta; muulloin
   (b) valitse luvut β_j, γ_j siten, että $\beta_j \gamma_j = \rho_j$;
   (c) normeeraa $\boldsymbol{v}_j = \tilde{\boldsymbol{v}}_j / \gamma_j$ ja $\boldsymbol{w}_j = \tilde{\boldsymbol{w}}_j / \beta_j$;
   (d) laske $\alpha_j = \boldsymbol{w}_j^T \boldsymbol{A} \boldsymbol{v}_j$;
   (e) laske $\tilde{\boldsymbol{v}}_{j+1} = \boldsymbol{A}\boldsymbol{v}_j - \alpha_j \boldsymbol{v}_j - \beta_j \boldsymbol{v}_{j-1}$;
   (f) laske $\tilde{\boldsymbol{w}}_{j+1} = \boldsymbol{A}^T \boldsymbol{w}_j - \alpha_j \boldsymbol{w}_j - \gamma_j \boldsymbol{w}_{j-1}$.

Kertoimien β_j ja γ_j valinta vaiheessa (b) on mielivaltainen. Yleensä valitaan $\gamma_j = \sqrt{|\rho_j|}$, jolloin β_j ja γ_j ovat itseisarvoiltaan yhtä suuret.

Merkitään kantavektoreiden $\boldsymbol{v}_i$ ja $\boldsymbol{w}_i$ muodostamia $n \times m$-matriiseja $\boldsymbol{V}_m$:llä ja $\boldsymbol{W}_m$:llä. Tällöin iteraatiokaava voidaan kirjoittaa muodossa (vertaa Arnoldin algoritmiin kaavaan (14))

$$\boldsymbol{A}\boldsymbol{V}_m = \boldsymbol{V}_{m+1} \bar{\boldsymbol{T}}_m = \boldsymbol{V}_m \boldsymbol{T}_m + \gamma_m \boldsymbol{v}_{m+1} \boldsymbol{i}_m^T, \tag{39}$$

$$\boldsymbol{A}^T \boldsymbol{W}_m = \boldsymbol{W}_{m+1} \bar{\boldsymbol{T}}_m^T = \boldsymbol{W}_m \boldsymbol{T}_m^T + \beta_m \boldsymbol{w}_{m+1} \boldsymbol{i}_m^T, \tag{40}$$

$$\boldsymbol{W}_m^T \boldsymbol{A} \boldsymbol{V}_m = \boldsymbol{T}_m, \tag{41}$$

jossa $m \times m$ matriisit $\boldsymbol{T}_m = \text{tridiag}(\gamma_i, \alpha_i, \beta_{i+1})$ ja $\boldsymbol{T}_m^T$ saadaan matriiseista $\bar{\boldsymbol{T}}_m$ ja $\bar{\boldsymbol{T}}_m^T$ poistamalla niistä alin vaakarivi.

Kaksoisliittogradienttimenetelmä

Kaksipuoleista Lanczosin menetelmää käyttäen ratkaisu voidaan esittää periaatteessa muodossa

$$\boldsymbol{x}_m = \boldsymbol{x}_0 + \boldsymbol{V}_m \boldsymbol{y}_m, \tag{42}$$

jossa ratkaisuvektori $\boldsymbol{y}_m$ voidaan valita usealla eri tavalla. Yksi mahdollisuus on käyttää Petrovin-Galerkinin ehtoa ja pakottaa jäännökset ortogonaalisiksi vektoreita $\boldsymbol{w}_k, k = 1, \ldots, m$ vastaan, jolloin saadaan

$$\boldsymbol{W}_m^T \boldsymbol{r}_m = \boldsymbol{W}_m^T \boldsymbol{r}_0 - \boldsymbol{W}_m^T \boldsymbol{A} \boldsymbol{V}_m \boldsymbol{y}_m = \beta \boldsymbol{i}_1 - \boldsymbol{T}_m \boldsymbol{y}_m, \tag{43}$$

jossa $\beta = \|\boldsymbol{r}_0\|_2$. Yhtälösysteemi (43) voidaan ratkaista mikäli tridiagonaalimatriisi $\boldsymbol{T}_m$ on säännöllinen. On kuitenkin huomautettava, että $\boldsymbol{T}_m$ voi olla singulaarinen tai liki singulaarinen, vaikka Lanczosin prosessi olisikin hyvin määritelty, eli $\rho_j \neq 0$ kohdassa 2(a). Kaksoisliittogradienttimenetelmä voidaan johtaa kaksipuoleisesta Lanczosin menetelmästä samalla tavoin kuin liittogradienttimenetelmä symmetrisestä Lanczosin algoritmistä.

Kaksoisliittogradienttimenetelmä (Bi-CG)

1. Laske $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$; aseta $\boldsymbol{d}_0 = \boldsymbol{r}_0$;
   valitse $\tilde{\boldsymbol{r}}_0$ siten, että $\tau_0 = \tilde{\boldsymbol{r}}_0^T \boldsymbol{d}_0 \neq 0$, ja aseta $\tilde{\boldsymbol{d}}_0 = \tilde{\boldsymbol{r}}_0$.
2. Toista $i = 0, 1, 2, ...$, kunnes supennut:
   (a) laske: $\boldsymbol{s} = \boldsymbol{A}\boldsymbol{d}_i, \quad \tilde{\boldsymbol{s}} = \boldsymbol{A}^T \tilde{\boldsymbol{d}}_i, \quad a_i = \tau_i / \tilde{\boldsymbol{d}}_i^T \boldsymbol{s}$;
   (b) päivitä: $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + a_i \boldsymbol{d}_i, \quad \boldsymbol{r}_{i+1} = \boldsymbol{r}_i - a_i \boldsymbol{s}, \quad \tilde{\boldsymbol{r}}_{i+1} = \tilde{\boldsymbol{r}}_i - a_i \tilde{\boldsymbol{s}}$;
   (c) laske $\tau_{i+1} = \tilde{\boldsymbol{r}}_{i+1}^T \boldsymbol{r}_{i+1}, \quad \rho_i = \tau_{i+1} / \tau_i$;
   (d) päivitä $\boldsymbol{d}_{i+1} = \boldsymbol{r}_{i+1} + \rho_i \boldsymbol{d}_i, \quad \tilde{\boldsymbol{d}}_{i+1} = \tilde{\boldsymbol{r}}_{i+1} + \rho_i \tilde{\boldsymbol{d}}_i$.

Kaksoisliittogradienttimenetelmä on vaikeuksissa, mikäli sen pohjana oleva tridiagonaalimatriisi on huonokuntoinen. Menetelmässä on kaksi mahdollista kohtaa epäonnistua, joko kohdassa 2(a) sisätulo $\tilde{\boldsymbol{d}}^T \boldsymbol{A}^T \tilde{\boldsymbol{d}} \approx 0$ tai kohdassa 2(c) $\tilde{\boldsymbol{r}}^T \boldsymbol{r} \approx 0$. Parannettu versio menetelmästä on Freundin ja Nachtigalin 1991 esittämä kvasi-minimijäännösiteraatio QMR (Quasi-Minimal Residual) [28, 29]. Siinä vektoria $\boldsymbol{y}_m$ ei ratkaista yhtälöstä (43), vaan käytetään pienimmän neliön keinoa

$$\boldsymbol{y}_m = \text{argmin} \|\beta \boldsymbol{i}_1 - \bar{\boldsymbol{T}}_m \boldsymbol{y}\|_2, \tag{44}$$

sillä jäännös $\boldsymbol{r}_m$ voidaan lausua muodossa

$$\boldsymbol{r}_m = \boldsymbol{r}_0 - \boldsymbol{A}\boldsymbol{V}_m \boldsymbol{y}_m = \boldsymbol{V}_{k+1}(\beta \boldsymbol{i}_1 - \bar{\boldsymbol{T}}_m \boldsymbol{y}_m). \tag{45}$$

Jäännöksen normi toteuttaa siten ehdon

$$\|\boldsymbol{r}_m\|_2 \leq \|\boldsymbol{V}_{m+1}\|_2 \cdot \|\beta \boldsymbol{i}_1 - \bar{\boldsymbol{T}}_m \boldsymbol{y}\|_2. \tag{46}$$

Yhtälöllä (44) on aina ratkaisu riippumatta siitä, onko tridiagonaalimatriisi $\boldsymbol{T}_m$ singulaarinen. Täten QMR-menetelmän iteraatiot ovat hyvin määriteltyjä, jos taustalla oleva Lanczosin iteraatio onnistuu.

Kaksoisliittogradienttimenetelmä voidaan johtaa myös liittogradienttimenetelmänä laajennetulle symmetriselle systeemille

$$\begin{bmatrix} \boldsymbol{0} & \boldsymbol{A} \\ \boldsymbol{A}^T & \boldsymbol{0} \end{bmatrix} \begin{pmatrix} \tilde{\boldsymbol{x}} \\ \boldsymbol{x} \end{pmatrix} = \begin{pmatrix} \boldsymbol{b} \\ \tilde{\boldsymbol{b}} \end{pmatrix}, \tag{47}$$

jossa $\tilde{\boldsymbol{b}}$ on mielivaltainen vektori. Kaksoisliittogradienttimenetelmä ratkaisee siten samanaikaisesti kaksi systeemiä, jolloin puolet laskentatyöstä haaskaantuu tehtävään, jota ei välttämättä haluta ratkaista.

Niin Bi-CG kuin QMR menetelmät vaativat operoimista matriisin transpoosilla, joka ei ole aina helposti saatavilla. Sonneveld julkaisi 1989 kaksoisliittogradienttimenetelmästä johdetun version, jossa matriisin transpoosilla operoimista ei tarvita [55]. Menetelmä kulkee nimellä neliöity liittogradienttimenetelmä (CGS = Conjugate Gradient Squared) ja se suppenee kaksoisliittogradienttimenetelmään nähden kaksinkertaisella nopeudella, mikäli ratkaistava systeemi ei ole häiriöaltis (katso kuva 3b). Valitettavasti monissa tehtävissä CGS-menetelmä suppenee hyvin epätasaisesti, joka johtaa usein numeeriseen epästabiiliuteen. CGS-menetelmä vaatii kaksi matriisi–vektori-operaatiota ja kahden pistetulon muodostamisen iteraatioaskelta kohden. Se on siten verrattavissa kaksoisliittogradienttimenetelmän työmäärään.

Van der Vorst julkaisi 1992 [58] CGS-menetelmästä stabiloidun version artikkelissa, joka on ISI:n luokituksessa matematiikan alalla 1990-luvun viitatuin.[2] Tämä menetelmä tunnetaan lyhenteellä Bi-CGSTAB. Se voidaan tulkita kaksoisliittogradienttimenetelmän ja GMRES(1) menetelmien yhdistelmäksi. Kuten alla olevasta algoritmista voidaan havaita, vaatii Bi-CGSTAB-menetelmä iteraatioaskelta kohden kaksi matriisi–vektori-operaatiota ja neljä vektorien pistetuloa, mikä on kaksi pistetuloa enemmän kuin CGS-algoritmissa. Näiden operaatioiden lisäksi tarvitaan myös jäännöksen normin muodostaminen, mikäli suppenemista mitataan jäännöksestä.

Bi-CGSTAB on yksi suosituimmista iteraatiomenetelmistä epäsymmetrisen lineaarisen yhtälösysteemin ratkaisemiseksi. Se ei kuitenkaan konvergoi hyvin, mikäli matriisin ominaisarvot ovat kompleksisia. Tämä ongelma voidaan usein helposti poistaa käyttämällä pohjustinta. Menetelmästä on myös kehitetty kompleksista spektriä silmälläpitäen muunneltuja versioita [31, 54].

Stabiloitu kaksoisliittogradienttimenetelmä (Bi-CGSTAB)

1. Laske alkujäännös $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$; valitse $\tilde{\boldsymbol{r}}$; laske $\rho_0 = \tilde{\boldsymbol{r}}^T \boldsymbol{r}_0$; aseta $\boldsymbol{d}_0 = \boldsymbol{r}_0$.
2. Iteroi $i = 0, 1, 2, ...$, kunnes supennut:
   (a) laske: $\boldsymbol{v}_i = \boldsymbol{A}\boldsymbol{d}_i, \quad a_i = \rho_i / \tilde{\boldsymbol{r}}^T \boldsymbol{v}_i, \quad \boldsymbol{s} = \boldsymbol{r}_i - a_i \boldsymbol{v}_i$;
   (b) laske: $\boldsymbol{w} = \boldsymbol{A}\boldsymbol{s}, \quad \omega_i = \boldsymbol{w}^T \boldsymbol{s} / \boldsymbol{w}^T \boldsymbol{w}$;
   (c) päivitä: $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + a_i \boldsymbol{r}_i + \omega_i \boldsymbol{s}, \quad \boldsymbol{r}_{i+1} = \boldsymbol{s} - \omega_i \boldsymbol{w}$;
   (d) laske: $\|\boldsymbol{r}_{i+1}\|_2$;
   (e) laske: $\rho_{i+1} = \tilde{\boldsymbol{r}}^T \boldsymbol{r}_{i+1}$ and $b_{i+1} = (\rho_{i+1}/\rho_i)(a_i/\omega_i)$;
   (f) päivitä: $\boldsymbol{d}_{i+1} = \boldsymbol{r}_{i+1} + b_{i+1}(\boldsymbol{d}_i - \omega_i \boldsymbol{v}_i)$.

Koska kertoimien ρ ja ω on oltava nollasta eroavia, Bi-CGSTAB-iteraatiossa on kolme mahdollista virhetilannetta: $\tilde{\boldsymbol{r}}^T \boldsymbol{v}_i \approx 0, \boldsymbol{w}^T \boldsymbol{s} \approx 0$ tai $\tilde{\boldsymbol{r}}^T \boldsymbol{r}_i \approx 0$. Todennäköisyyttä kohdata näitä virhetilanteita voidaan pienentää valitsemalla $\tilde{\boldsymbol{r}}$ sopivasti tai käyttämällä hyvää pohjustinta. Menetelmän muunnokset [31, 54] ovat tässä suhteessa hivenen luotettavampia.

Muita menetelmä

Numeerisen matematiikan kirjallisuudessa on esitetty suuri joukko muitakin iteraatiomenetelmiä. Mainittakoon tässä Eirolan ja Nevanlinnan yksirangiseen päivitykseen perustu-

[2] `http://www.in-cites.com/papers/dr-henk-van-der-vorst.html`

va menetelmä [22] sekä Broydenin päivityskaavaan perustuvat menetelmät [19], joita on vertailtu GMRES-menetelmään lähteessä [62].

Pohjustetut iteraatiot

Periaate

Käytännön ongelmissa edellä esitetyt iteraatiot ovat sellaisenaan kuitenkin aivan liian hitaasti suppenevia. Suppenemista voidaan parantaa käyttämällä pohjustinoperaatiota. Esimerkiksi yhtälön (1) ekvivalentti pohjustettu systeemi on

$$\boldsymbol{M}_1^{-1}\boldsymbol{A}\boldsymbol{M}_2^{-1}\boldsymbol{y} = \boldsymbol{M}_1^{-1}\boldsymbol{b}, \tag{48}$$

jossa $\boldsymbol{M}_1$ ja $\boldsymbol{M}_2$ ovat vasemman- ja oikeanpuoleiset pohjustinmatriisit. Käytännössä tälläistä jakoa ei useinkaan tarvita vaan iteraatiomenetelmä johdetaan muodossa, jossa vain yksi pohjustinoperaatio on tarpeen. Yhtälö (48) antaa kuitenkin mahdollisuuden erilaisiin pohjustinstrategioihin. Operaattoreiden $\boldsymbol{M}^{-1}\boldsymbol{A}$, $\boldsymbol{A}\boldsymbol{M}^{-1}$ ja $\boldsymbol{M}_1^{-1}\boldsymbol{A}\boldsymbol{M}_2^{-1}$, jossa $\boldsymbol{M} = \boldsymbol{M}_1\boldsymbol{M}_2$, spektrit ovat identtiset, joten vasemman- ja oikeanpuoleisen tai jaetun pohjustuksen käytön voi olettaa tuottavan samankaltaisesti suppenevan iteraation. Huomattakoon, että ominaisarvospektri ei yksistään määrää konvergenssia [50].

Pohjustimen muodostaminen on optimointiongelma: sen pitäisi olla hyvä likiarvo $\boldsymbol{A}$:lle, nopeasti muodostettavissa ja sillä operoimisen olisi oltava mahdollisimman vähän laskentatyötä vaativa. Näitä toisilleen ristiriitaisia ominaisuuksia ei voida samanaikaisesti toteuttaa, vaan on tyydyttävä kompromisseihin.

Useimmille pohjustetuista iteraatiomenetelmistä pohjustinoperaation on oltava vakio. Muuttuvan pohjustimen mahdollistavia menetelmiä on myös kehitetty. Ehkä tunnetuin näistä on GMRES-menetelmän joustava muoto [45, luku 7.6], [50, luku 9.4].

Yleiskäyttöiset pohjustinstrategiat voidaan jakaa seuraaviin luokkiin [23]:

1. klassisiin iteraatioihin perustuvat pohjustimet, kuten Jacobi ja SSOR,
2. epätäydelliset harvat LU-hajotelmat, ILU (Incomplete LU) tai symmetrisille positiivisesti definiiteille systeemeille IC (Incomplete Cholesky),
3. polynomipohjustimet,
4. eksplisiittiset harvat käänteismatriisipohjustimet,
5. moniverkko- tai algebralliset monitasopohjustimet,
6. muut pohjustinstrategiat, esim. EBE (= Element-By-Element).

Käytännön rakenneanalyysin tehtävissä polynomi- ja klassisiin iteraatioihin perustuvat pohjustimet ovat liian tehottomia. Tarkastellaan seuraavaksi epätäydelliseen hajotelmaan, käänteismatriisiaproksimaatioon ja monitasomenetelmiin perustuvia pohjustimia.

Epätäydelliseen hajotelmaan perustuvat pohjustimet

Epätäydelliseen hajotelmaan perustuvat pohjustimet ovat ehkä kaikkein tunnetuimpia ja käytetyimpiä. Niistä on olemassa lukuisia erilaisia versioita – jo pelkästään, miten pohjustimen harvuusrakenne on määritelty. Yksinkertaisin strategia on käyttää hajotelman osille $\boldsymbol{L}$ ja $\boldsymbol{U}$ samaa harvuusrakennetta kuin matriisille $\boldsymbol{A}$. Tämä epätäydellinen hajotelma, joka tunnetaan lyhenteellä ILU(0) tai symmetrisille tapauksille IC(0), on helppo ohjelmoida ja nopea laskea. Mutta usein se johtaa hyvin karkeaan aproksimaatioon, minkä seurauksena iteraatio suppenee hitaasti. Voidaan osoittaa, että toisen kertaluvun elliptisille ongelmille

ILU(0)-pohjustettu iteraatio ei ole asymptoottisesti pohjustamatonta iteraatiota parempi. Tämä tarkoittaa sitä, että ILU(0)-pohjustinoperaation $\boldsymbol{M}^{-1}\boldsymbol{A}$ häiriöalttius on samaa luokkaa kuin itse matrisiin $\boldsymbol{A}$.

Useita vaihtoehtoisia ILU-hajotelmia on kehitetty, joissa harvuusrakenne määräytyy joko staattisesti tasoluvun mukaan tai dynaamisesti hajotelman muodostusvaiheessa jättämällä pienet alkiot pois pohjustimen harvuusrakenteesta. Meijerink ja Van der Vorst [44] osoittivat, että ILU-hajotelma on olemassa mielivaltaiselle harvuusrakenteelle, jos kerroinmatriisi on M-matriisi[3]. Näin on usein lämmönjohtumistehtävän kaltaisille ongelmille. Rakenteiden mekaniikan tehtävissä jäykkyysmatriisit eivät pääsääntöisesti kuitenkaan ole M-matriiseja.

Symmetristen ja positiivisesti definiittien matriisien tapauksessa voidaan epätäydellisen hajotelman olemassaolon varmistavat tekniikat luokitella kolmeen pääkategoriaan [8]. Yksinkertaisin tapa varmistaa epätäydellisen hajotelman olemassaolo on suorittaa hajotelma matriisille, jonka koko diagonaalia tai sen yksittäisiä alkioita on kasvatettu hajotelman tekovaiheessa. Toinen tapa on muodostaa epätäydellinen hajotelma matriisista $\boldsymbol{A}$ muodostetulle, sitä lähellä olevalle M-matriisille [53]. Vaikka näitä kahta menetelmää soveltaen epätäydellisen hajotelman teko onnistuu aina, on tuloksena usein huono pohjustin, jonka seurauksena kiihdytiniteraatio suppenee hitaasti. Kolmas, monimutkaisin ja yleisesti suositeltavin tapa on muotoilla hajotelma-algoritmi siten, että se tuottaa kaikille harvuusrakenteille SPD-pohjustimen. Tälläisiä pohjustimia ovat esittäneet mm. Ajiz ja Jennings [1], Tismenetsky [57] ja Kaporin [37]. Niiden ideaa voitaisiin nimittää stabiloiduksi hylkäysstrategiaksi. Näissä menetelmissä epätäydellinen Choleskyn hajotelmamatriisi $\boldsymbol{L}$ on häirityn matriisin $\boldsymbol{M} = \boldsymbol{A} + \boldsymbol{C}$ tarkka hajotelmamatriisi $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{L}^T$, ja jossa $\boldsymbol{C}$ on positiivisesti semidefiniitti "hylkäysmatriisi". Tämän hajotelman muodostaminen ei voi epäonnistua millään alakolmiomatriisin $\boldsymbol{L}$ harvuusrakenteella.

Kaporin [37] johtaa pohjustimen uuden hajotelmamuodon

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^T + \boldsymbol{L}\boldsymbol{R}^T + \boldsymbol{R}\boldsymbol{L}^T \tag{49}$$

avulla, jossa $\boldsymbol{R}$ on aito alakolmiomatriisi, joka on rakenteellisesti ortogonaalinen $\boldsymbol{L}$:n kanssa, eli $L_{ij}R_{ij} = 0$, $\forall\, i, j$. Hajotelma on olemassa mielivaltaiselle SPD-matriisille $\boldsymbol{A}$ ja mielivaltaiselle hajotelmamatriisin $\boldsymbol{L}$ harvuusrakenteelle. Mikäli lasketaan hajotelmaan (49) pohjautuvan pohjustimen virhematriisi, saadaan

$$\boldsymbol{E} = \boldsymbol{I} - \boldsymbol{L}^{-1}\boldsymbol{A}\boldsymbol{L}^{-T} = -\boldsymbol{R}^T\boldsymbol{L}^{-T} - \boldsymbol{L}^{-1}\boldsymbol{R}. \tag{50}$$

Ajizin ja Jenningsin pohjustin voidaan esittää muodossa [37]:

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^T - \boldsymbol{D} + \boldsymbol{R} + \boldsymbol{R}^T, \tag{51}$$

jossa $\boldsymbol{D}$ on ei-negatiivinen diagonaalimatriisi. Tämän pohjustimen virhematriisi on

$$\boldsymbol{E} = \boldsymbol{I} - \boldsymbol{L}^{-1}\boldsymbol{A}\boldsymbol{L}^{-T} = \boldsymbol{L}^{-1}(\boldsymbol{D} - \boldsymbol{R} - \boldsymbol{R}^T)\boldsymbol{L}^{-T}. \tag{52}$$

Vertaamalla Kaporinin pohjustimen virhematriisia (50) Ajizin ja Jenningsin vastaavaan (52) voidaan todeta, että virhematriisin alkiot ovat Ajizin ja Jenningsin pohjustimelle luokkaa $\|\boldsymbol{L}^{-1}\|^2\|\boldsymbol{R}\|$, kun taas Kaporinin pohjustimelle vain $\|\boldsymbol{L}^{-1}\|\|\boldsymbol{R}\|$. Valittavasti $\|\boldsymbol{L}^{-1}\|$

[3]Matriisi sanotaan olevan M-matriisi jos sen diagonaalin ulkopuoliset alkiot ovat ei-negatiivisia ja kaikki sen käänteismatriisin alkiot ovat positiivisia.

voi olla hyvin suuri, jos häiriöaltista matriisia $\boldsymbol{A}$ yritetään approksimoida tarkasti. Tällöin hylkäysparametrien pienentyessä $\|\boldsymbol{L}^{-1}\|$ lähestyy arvoa $\|\boldsymbol{A}^{-1/2}\|$. Kaporinin pohjustimen voi olettaa siten tuottavan huomattavasti nopeammin suppenevan iteraation. Tämä on myös selvästi nähtävissä kirjoituksen lopussa esitetyistä numeerisisista esimerkeistä.

Kaporinin pohjustimen haittapuolena on varsin suuri väliaikaisen mustitilan tarve ja sen vaatima suuri työmäärä. Tilantarvetta ja työmäärää voidaan kuitenkin hieman vähentää käyttämällä toista hylkäysparametria hajotelmavaiheen väliaikaisille suureille. Toisaalta, väliaikaisvaiheen pieninä hylättyjen alkioden vuoksi on suoritettava diagonaalinen kompensaatio Ajizin ja Jenningsin pohjustimen tapaan, jotta epätäydellinen hajotelma olisi olemassa. Tällöin pohjustimen $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{L}^T$ määrittelee yhtälön (49) sijasta lauseke

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^T - \boldsymbol{D} + \boldsymbol{L}\boldsymbol{R}^T + \boldsymbol{R}\boldsymbol{L}^T, \tag{53}$$

jossa $\boldsymbol{D}$ on ei-negatiivinen diagonaalimatriisi.

Epäsymmetrisessä tapauksissa ei ole menetelmää, joka takaisi hajotelman olemassaolon lukuunottamatta tapauksia, joissa kerroinmatriisi on M-matriisi. Saadin [49] kehittämä ILUT-pohjustin on varsin käyttökelpoinen elementtimenetelmissä syntyville epäsymmetrisille matriiseille. Siinä epätäydellisen hajotelman kokoa rajoitetaan tilaparametrilla p, joka tarkoittaa jokaiselle riville sallittavan lisäalkioiden maksimilukumäärää. Pohjustinta merkitään jatkossa ILUT(ψ, p), jossa ψ on hylkäysparametri.

Käänteismatrisiipohjustimet

Harvat käänteismatriisipohjustimet ovat hajotelmapohjustimia huomattavasti helpommin rinnakkaistettavissa. Näissä tekniikoissa muodostetaan eksplisiittisesti harva matriisi $\boldsymbol{M}^{-1}$, joka suoraan approksimoi käänteismatriisia $\boldsymbol{A}^{-1}$ [5, 33, 50]. Tällöin pohjustinoperaatio $\boldsymbol{M}^{-1}$ on vain matriisin kertominen vektorilla, joka on helposti rinnakkaistettavissa oleva operaatio. Toisaalta, tälläisen pohjustimen muodostaminen voi olla työlästä ja suppeneminenkin voi olla implisiittisiä menetelmiä hitaampaa.

Käänteismatriisipohjustintekniikat nojaavat otaksumaan, että on mahdollista muodostaa hyvä approksimaatio harvan matriisin $\boldsymbol{A}$ käänteimatriisille $\boldsymbol{A}^{-1}$, joka myös on *harva*. Hyvän ja harvan käänteismatriisiapproksimaation olemassaolo ei ole mitenkään taattu, sillä harvankin matriisin käänteismatriisi on yleensä täysi.

SPAI-menetelmänä[4] tunnetaan strategia, jossa harva matriisi $\boldsymbol{H} = \boldsymbol{M}^{-1} \approx \boldsymbol{A}^{-1}$ ratkaistaan rajoitetusta minimointiongelmasta

$$\min \ \|\boldsymbol{I} - \boldsymbol{A}\boldsymbol{H}\|_F \quad \text{rajoitteella } \boldsymbol{H} \in \mathcal{S}, \tag{54}$$

jossa $\mathcal{S}$ on harvoista matriiseista koostuva joukko. Matriisinormina käytetään Frobeniuksen normia

$$\|\boldsymbol{A}\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} |A_{ij}|^2} \tag{55}$$

tai jotain sen painotettua muotoa. Minimointiongelman (54) ratkaisu jakaantuu luonnollisesti n:n pienen lineaarisen systeemin ratkaisuun

$$\|\boldsymbol{I} - \boldsymbol{A}\boldsymbol{H}\|_F^2 = \sum_{k=1}^{n} \|\boldsymbol{i}_k - \boldsymbol{A}\boldsymbol{h}_k\|_2^2, \tag{56}$$

[4]SPAI = Sparse Approximate Inverse

jossa $\boldsymbol{h}_k$ on matriisin $\boldsymbol{H}$ k:s pystyrivi. Täten osatehtävät voidaan ratkaista rinnakkaislaskentana. Huomataan, että (54) muodostaa oikealta operoivan pohjustimen. Vasemmalta pohjustaville iteraatioille voidaan ratkaista ensin oikeanpuoleinen känteismatriisiapproksimaatio $\boldsymbol{A}$:n transpoosille, ja ottamalla sen transpoosi, sillä $\|\boldsymbol{I}-\boldsymbol{H}\boldsymbol{A}\|_F = \|\boldsymbol{I}-\boldsymbol{A}^T\boldsymbol{H}^T\|_F$.

On olemassa myös käänteismatriisipohjustintekniikoita, jotka muodostavat approksimaation käänteismatriisin hajotelmalle, esimerkiksi konjugointiin perustuva AINV [11] tai minimointiin perustuva SPAI-menettelyn kaltainen FSAI[5] [39]. Mikäli hajotelmaan perustuvaa käänteismatriisipohjustinta käytetään symmetriseen ja positiivisesti definiittiin ongelmaan, on pohjustimen symmetrian ja positiivisuuden toteuttaminen helppo taata, toisin kuin SPAI-tekniikoissa.

FSAI-menettely muodostaa alakolmiomatrisin $\boldsymbol{G}$ ratkaisemalla rajoitetun minimointiongelman

$$\min \ \|\boldsymbol{I} - \boldsymbol{G}\boldsymbol{L}\|_F \quad \text{rajoitteella } \boldsymbol{G} \in \mathcal{S}, \tag{57}$$

jossa $\boldsymbol{L}$ on $\boldsymbol{A}$:n Choleskyn hajotelman alakolmiomatriisi ja $\mathcal{S}$ on niiden alakolmiomatriisien joukko, jolla on tietty diagonaalin sisältävä harvuusrakenne. Käänteimatriisipohjustin on siten $\boldsymbol{H} = \boldsymbol{M}^{-1} = \boldsymbol{G}^T\boldsymbol{G}$.

Minimoitiongelma (57) voidaan ratkaista matriisia $\boldsymbol{L}$ tuntematta operoimalla pelkästään matriisilla $\boldsymbol{A}$ [39]. FSAI-pohjustimen ongelmana on $\boldsymbol{G}$:n harvuusrakenteen määrittäminen.

AINV-menetelmä [9] perustuu Grammin-Schmidtin $\boldsymbol{A}$-ortogonalisointiprosessiin eikä vaadi etukäteen määrättyä harvuusrakennetta. Mielivaltaisesta n-ulotteisesta lineaarisesti riippumattomasta vektorikannasta lähtien AINV-algoritmi muodostaa vektorijoukot $\{\boldsymbol{z}_i\}_{i=1}^n$ ja $\{\boldsymbol{w}_i\}_{i=1}^n$, jotka ovat $\boldsymbol{A}$-biortogonaalisia. Määritellään matriisit $\boldsymbol{Z} = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n]$ ja $\boldsymbol{W} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_n]$. Tällöin

$$\boldsymbol{W}^T\boldsymbol{A}\boldsymbol{Z} = \boldsymbol{D} = \operatorname{diag}(p_1, p_2, \ldots, p_n), \tag{58}$$

jossa $p_i = \boldsymbol{w}_i^T\boldsymbol{A}\boldsymbol{z}_i \neq 0$. Tästä saadaan $\boldsymbol{A}^{-1}$:n hajotelma

$$\boldsymbol{A}^{-1} = \boldsymbol{Z}\boldsymbol{D}^{-1}\boldsymbol{W}^T = \sum_{i=1}^{n} \frac{\boldsymbol{z}_i\boldsymbol{w}_i^T}{p_i}. \tag{59}$$

Jos $\boldsymbol{A}$-ortogonalisointiprosessia sovelletaan tavanomaisiin kantavektoreihin $\boldsymbol{i}_1, \ldots, \boldsymbol{i}_n$, havaitaan helposti, että $\boldsymbol{Z}$ ja $\boldsymbol{W}$ ovat yläkolmiomatriisit $\boldsymbol{Z} = \boldsymbol{U}^{-1}$ ja $\boldsymbol{W} = \boldsymbol{L}^{-T}$, jossa $\boldsymbol{U}$ ja $\boldsymbol{L}$ muodostavat $\boldsymbol{A}$:n kolmiohajotelman $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{U}$.

Pohjustinta muodostettaessa $\boldsymbol{Z}$ lasketaan epätäydellisesti poistamalla vektorin päivitysoperaatiossa pienet alkiot, jolloin $\bar{\boldsymbol{Z}} \approx \boldsymbol{Z}$, $\bar{\boldsymbol{W}} \approx \boldsymbol{W}$ ja $\bar{\boldsymbol{D}} \approx \boldsymbol{D}$. Täten käänteismatriisin hajotelmapohjustin on $\boldsymbol{H} = \boldsymbol{M}^{-1} = \bar{\boldsymbol{Z}}\bar{\boldsymbol{D}}^{-1}\bar{\boldsymbol{W}}^T$. Menetelmä soveltuu yhtälailla symmetriseen tapaukseen, jolloin $\boldsymbol{W} = \boldsymbol{Z}$. Menetelmästä on kehitetty SPD matriiseja varten stabiloitu muoto [11]. Numeeriset esimerkit [11, 12, 13] ovat osoittaneet menetelmän toimivan hyvin erilaisissa sovelluskohteissa.

Byckling ja Huhtanen [18] ovat esittäneet uuden tavan konstruoida harva käänteismatriisipohjustin, jonka avulla voidaan muodostaa myös tavanomainen hajotelmapohjustin.

Moniverkko- ja monitasopohjustimet

Klassiset iteraatiot kuten Jacobi ja Gauss-Seidel ovat sellaisenaan liian hitaita elementtimenetelmän suurten yhtälösysteemien ratkaisemisessa. Menetelmien ominaisuuksia tar-

[5]FSAI = Factorized Sparse Approximate Inverse

kemmin tutkittaessa ne ovat kuitenkin nopeita ratkaisun korkeataajuuksisisten komponenttien vangitsijoina. Ratkaisun matalataajuuksiset komponentit suppenevat hitaimmin ja tekevät kyseisten iteraatiomenetelmien käytön suurten elementtimenetelmässä syntyvien yhtälösysteemien ratkaisussa kannattamattomaksi.

Moniverkkoratkaisijan idea on käyttää useita diskretoinnin tasoja vangitsemalla ratkaisun matalataajuuksiset komponentit harvimmassa verkossa. Moniverkkoratkaisijaa voidaan käyttää lineaarisen systeemin ratkaisualgoritmina yksistäänkin eikä vain pohjustimena jollekin Krylovin aliavaruusiteraatiolle. Moniverkkoalgoritmin kaksi oleellista vaihetta ovat: (i) relaksaatiovaihe, jossa ratkaisun korkeataajuuksiset komponentit vangitaan yksinkertaisella iteraatiolla, esim. Jacobin tai Gauss-Seidelin iteraatiolla (relaksaatiota suoritetaan useilla diskretointitasoilla), ja (ii) korjausvaihe, jossa ratkaisun matalataajuuksiset komponentit vangitaan ratkaisemalla ongelma hyvin harvassa verkossa.

Moniverkkoalgoritmi esitetään useissa lähteissä (esim. [4, 56]) muodossa, jossa prosessi aloitetaan mielivaltaisesta alkuarvauksesta hienoimmassa verkossa relaksoiden ja siirtyen kohti karkeinta verkkoa, jossa suoritetaan korjaus ja palataan takaisin korjaten ja relaksoiden kohti hienointa verkkoa. Vieläkin tehokkaampi ja luonnollisempi tapa on johtaa hienoimman verkon iteraatiolle alkuarvaus karkeimman verkon ratkaisusta [47].

Yksinkertainen kaksiverkkoalgoritmi voisi olla seuraavanlainen.

1. Ratkaise systeemi harvassa verkossa $\boldsymbol{A}_1\boldsymbol{x}_1 = \boldsymbol{b}_1$.
2. Interpoloi hienon verkon arvot harvasta verkosta $\boldsymbol{x}_2 = \boldsymbol{F}\boldsymbol{x}_1$.
3. Suorita muutama relaksaatioiteraatio hienossa verkossa.
4. Muunna jäännös $\boldsymbol{r}_2 = \boldsymbol{f}_2 - \boldsymbol{A}_2\boldsymbol{x}_2$ harvempaan verkkoon $\boldsymbol{r}_1 = \boldsymbol{C}\boldsymbol{r}_2$.
5. Ratkaise harvan verkon korjaus $\boldsymbol{A}_1\Delta\boldsymbol{x}_1 = \boldsymbol{r}_1$.
6. Siirrä korjaus hienoon verkkoon ja lisää edellisiin hienon verkon arvoihin $\Delta\boldsymbol{x}_2 = \boldsymbol{F}\Delta\boldsymbol{x}_1, \quad \boldsymbol{x}_2^{\text{uusi}} = \boldsymbol{x}_2^{\text{vanha}} + \Delta\boldsymbol{x}_2$.
7. Siirry kohtaan 3.

Matriisi $\boldsymbol{F}$ on interpolaatio hienoon verkkoon ja vastaavasti $\boldsymbol{C}$ on "keskiarvoistus" harvaan verkkoon, joka on $\boldsymbol{F}$:n transpoosi. Huomattakoon, että kaksitasomenetelmä ei ole systeemin suoraa ratkaisua tehokkaampi menetelmä, mutta kelpaa moniverkkoalgoritmin idean esittelyyn.

Voidaan osoittaa, että moniverkkoalgoritmin suppenemisnopeus ei ole riippuvainen tehtävän koosta, mikä on suuri etu tavanomaisiin iteraatioihin verrattuna. Lisäksi moniverkkoalgoritmi on asymptoottisesti optimaalinen työmäärän suhteen.

Elementtimenetelmän yhteydessä on FETI-monitasoratkaisija (FETI = Finite Element Tearing and Interconnecting) saavuttanut suosiota [24, 25]. Siinä elementtiverkko "leikataan" alirakenteisiin ja nämä alirakenteet kytketään nurkistaan toisiin Lagrangen kertojien avulla. Se on erityisesti suunniteltu rinnakkaislaskentaan, sillä jokaisella iteraatiolla tarvitaan jokaisen alirakenneprobleeman ratkaisu ja Lagrangen kertojien ratkaisu. Alirakenneongelma ratkaistaan yleensä suoralla ratkaisijalla ja Lagrangen kertojat iteroidaan liittogradienttimenetelmällä.

Myös puhtaasti algebrallisia monitasomenetelmiä on kehitetty [17, 52].

Moniverkko- ja monitasoalgoritmit eivät välttämättä tarvitse kiihdytiniteraatiota, vaan ovat jo itsessään tehokkaita iteratiivisia menetelmiä lineaarisen systeemin ratkaisemiseksi.

Pohjustetut algoritmit

Pohjustinoperaation liittäminen edellä esitettyihin algoritmeihin on yksinkertainen muutos. Esitetään seuraavassa kolmen yleisimmän iteraation vasemmalta pohjustavat versiot.

Pohjustettu liittogradienttimenetelmäalgoritmi (PCG)

1. Muodosta pohjustin $\boldsymbol{M}$ (tai suoraan $\boldsymbol{M}^{-1}$).
2. Alusta $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$; suorita pohjustinoperaatio $\boldsymbol{d}_0 = \boldsymbol{M}^{-1}\boldsymbol{r}_0$; laske $\tau_0 = \boldsymbol{r}_0^T \boldsymbol{d}_0$.
3. Iteroi $i = 0, 1, 2, ...$, kunnes menetelmä suppenee:
   (a) laske: $\boldsymbol{s} = \boldsymbol{A}\boldsymbol{d}_i, \quad a_i = \tau_i / \boldsymbol{d}_i^T \boldsymbol{s}$;
   (b) päivitä: $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + a_i \boldsymbol{d}_i, \quad \boldsymbol{r}_{i+1} = \boldsymbol{r}_i - a_i \boldsymbol{s}$;
   (c) pohjusta: $\boldsymbol{z} = \boldsymbol{M}^{-1}\boldsymbol{r}_{i+1}$;
   (d) laske: $\tau_{i+1} = \boldsymbol{r}_{i+1}^T \boldsymbol{z}, \quad b_i = \tau_{i+1}/\tau_i$;
   (e) päivitä: $\boldsymbol{d}_{i+1} = \boldsymbol{z} + b_i \boldsymbol{d}_i$.

Edullinen tapa tutkia liittogradienttimenetelmän suppenemista perustuu skalaariin τ kohdassa 2(d). Se tosin mittaa jäännöksen suuruutta $\boldsymbol{M}^{-1}$-painotetussa normissa: $\sqrt{\tau} = (\boldsymbol{r}^T \boldsymbol{M}^{-1} \boldsymbol{r})^{1/2} = \|\boldsymbol{r}\|_{M^{-1}}$. Edellä esitetty algoritmi vaatii iteraatioaskelta kohden yhden pohjustinoperaation, yhden matriisin ja vektorin kertomisen, kaksi pistetuloa ja kolme vektorin päivitystä. Mikäli halutaan mitata todellisen jäännöksen pienenemistä, joudutaan iteraatioaskeleella laskemaan lisäksi jäännöksen normi.

Pohjustettu ja stabiloitu bi-konjugaatti gradienttimenetelmä

1. Muodosta pohjustin $\boldsymbol{M}$ (tai suoraan $\boldsymbol{M}^{-1}$).
2. Laske alkujäännös $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$; valitse $\tilde{\boldsymbol{r}}$; laske $\rho_0 = \tilde{\boldsymbol{r}}^T \boldsymbol{r}_0$; aseta $\boldsymbol{d}_0 = \boldsymbol{r}_0$.
3. Iteroi $i = 0, 1, 2, ...$, kunnes supennut:
   (a) pohjusta: $\boldsymbol{z} = \boldsymbol{M}^{-1}\boldsymbol{d}_i$;
   (b) laske: $\boldsymbol{v}_i = \boldsymbol{A}\boldsymbol{z}, \quad a_i = \rho_i / \tilde{\boldsymbol{r}}^T \boldsymbol{v}_i, \quad \boldsymbol{s} = \boldsymbol{r}_i - a_i \boldsymbol{v}_i$;
   (c) pohjusta: $\tilde{\boldsymbol{s}} = \boldsymbol{M}^{-1}\boldsymbol{s}$;
   (d) laske: $\boldsymbol{w} = \boldsymbol{A}\tilde{\boldsymbol{s}}, \quad \omega_i = \boldsymbol{w}^T \boldsymbol{s} / \boldsymbol{w}^T \boldsymbol{w}$;
   (e) päivitä: $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + a_i \boldsymbol{z} + \omega_i \tilde{\boldsymbol{s}}, \quad \boldsymbol{r}_{i+1} = \boldsymbol{s} - \omega_i \boldsymbol{w}$;
   (f) laske: $\|\boldsymbol{r}_{i+1}\|_2$;
   (g) laske: $\rho_{i+1} = \tilde{\boldsymbol{r}}^T \boldsymbol{r}_{i+1}$ ja $b_{i+1} = (\rho_{i+1}/\rho_i)(a_i/\omega_i)$;
   (h) päivitä: $\boldsymbol{d}_{i+1} = \boldsymbol{r}_{i+1} + b_{i+1}(\boldsymbol{d}_i - \omega_i \boldsymbol{v}_i)$.

Myös pohjustetussa Bi-CGSTAB-iteraatiossa on kolme mahdollista virhetilannetta: $\tilde{\boldsymbol{r}}^T \boldsymbol{v}_i \approx 0$, $\boldsymbol{w}^T \boldsymbol{s} \approx 0$ tai $\tilde{\boldsymbol{r}}^T \boldsymbol{r}_i \approx 0$. Paras tae välttää nämä virhetilanteet on käyttää hyvää pohjustinta.

Kirjallisuudessa suosittu valinta $\tilde{\boldsymbol{r}}$-vektorille on alkujäännös $\boldsymbol{r}_0$. Bulgakov [17] suosittelee käytettäväksi vektoria $\tilde{\boldsymbol{r}} = \boldsymbol{M}^{-1}\boldsymbol{r}_0$. Jos alkuarvauksena $\boldsymbol{x}_0$ käytetään nollasta eroavaa satunnaisvektoria, nämä kaksi vaihtoehtoista menettelyä tuottavat miltei identtisen iteraatiohistorian. Mikäli aloitusvektorina käytetään nollavektoria ja kuormitusvektorissa $\boldsymbol{b}$ on vain muutama nollasta eroava komponentti, valinta $\tilde{\boldsymbol{r}} = \boldsymbol{r}_0$ ei ole suositeltava. Käyttökelpoinen valinta on tällöin $\tilde{\boldsymbol{r}} = \boldsymbol{r}_0 + \boldsymbol{a}$, missä $\boldsymbol{a}$ on satunnaisesti valittu vektori.

Pohjustettu GMRES(m)

1. Muodosta pohjustin $\boldsymbol{M}$ (tai suoraan $\boldsymbol{M}^{-1}$).
2. Laske $\boldsymbol{r}_0 = \boldsymbol{M}^{-1}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0)$, $\beta = \|\boldsymbol{r}_0\|_2$, $\boldsymbol{v}_1 = \boldsymbol{r}_0/\beta$.
3. Muodosta m-ulotteinen kanta:
   (a) aloittaen vektorista $\boldsymbol{v}_1$ muodosta Arnoldin algoritmilla $(m+1) \times m$ Hessenbergin matriisi $\bar{\boldsymbol{H}}_m$;
   (b) ratkaise $\boldsymbol{y}_m = \text{argmin}\|\beta \boldsymbol{i}_1 - \bar{\boldsymbol{H}}_m \boldsymbol{y}\|_2$ ja laske $\boldsymbol{x}_m = \boldsymbol{x}_0 + \boldsymbol{V}_m \boldsymbol{y}_m$; lopeta, mikäli jäännös on riittävän pieni, muulloin jatka;
   (c) aseta $\boldsymbol{x}_0 = \boldsymbol{x}_m$ ja palaa kohtaan 3(a).

Vasemmalta pohjustava GMRES-iteraatio on siten suoraviivainen yleistys GMRES-menetelmästä. Oikealta pohjustava versio on esitetty esim. lähteessä [50, luku 9.3.2].

Esimerkkitapauksia

Laskennat on suoritettu CSC – Tieteen tietotekniikan keskus Oy:n HP ProLiant DL785 G5 laskentapalvelimen yhdellä neliytimisellä AMD Opteron 8360 SE suorittimella (kellotaajuus 2,5 GHz). Palvelimessa on kahdeksan suorittimen kesken jaettu 512 Gt keskusmuisti. Lineaarisen systeemin ratkaisualiohjelmat on käännetty Portland Groupin Fortran kääntäjällä käyttäen `-fastsse` -optimointivalitsinta lukuunottamatta muutamaa erikoistapausta.[6]

Iteratiivisten menetelmien laskenta-aikoja on vertailtu myös muutamaan suoran menetelmän ratkaisuaikoihin. Kaikki ajat on ilmoitettu sekunneissa. Symmetrisille systeemeille on tavanomaisen alkioittaisen aktiivisarakeratkaisijan [7, 36] (merkitty symbolilla SKY) lisäksi käytetty lohkomaista aktiivisarakeratkaisijaa [43] (SKYB). Lohkostrategiassa matriisi paloitellaan alimatriiseihin ja hajotelmavaiheen skalaarioperaatiot muunnetaan matriisioperaatioiksi, jolloin voidaan käyttää hyväksi 3-tason BLAS (matriisi–matriisi-operaatiot) aliohjelmia. Tällöin voidaan saavuttaa huomattava laskenta-ajan nopeutuminen erityisesti hierarkisen muistijärjestelmän koneissa.

Kaikissa esimerkeissä on SKYB-ratkaisijassa käytetty lohkon kokona 95×192 (diagonaalilohko 95×95), jotka mahtuvat hyvin laskentapalvelimen 1 Mt:n suuruiseen toisen tason välimuistiin.

Ennen iteratiivista ratkaisua yhtälösysteemi on skaalattu siten, että kaikki diagonaalialkiot ovat ykkösen suuruisia. Tämä menettely tunnetaan myös Jacobi-pohjustuksen nimellä. Suppenemista on mitattu vain jäännöksesta ja torelansseina on käytetty arvoja $\epsilon_{\text{rel}} = 10^{-7}$ ja $\epsilon_{\text{abs}} = 10^{-12}$.

Kaksiulotteinen lämmönjohtumisongelma

Tarkastellaan kaksiulotteista lämmönsiirtymisongelmaa

$$-\operatorname{div}(k \operatorname{grad} u) + \rho c \vec{v} \cdot \operatorname{grad} u = 0, \tag{60}$$

alueessa $(x, y) \in (0, L) \times (0, L)$ ja reunaehdoilla

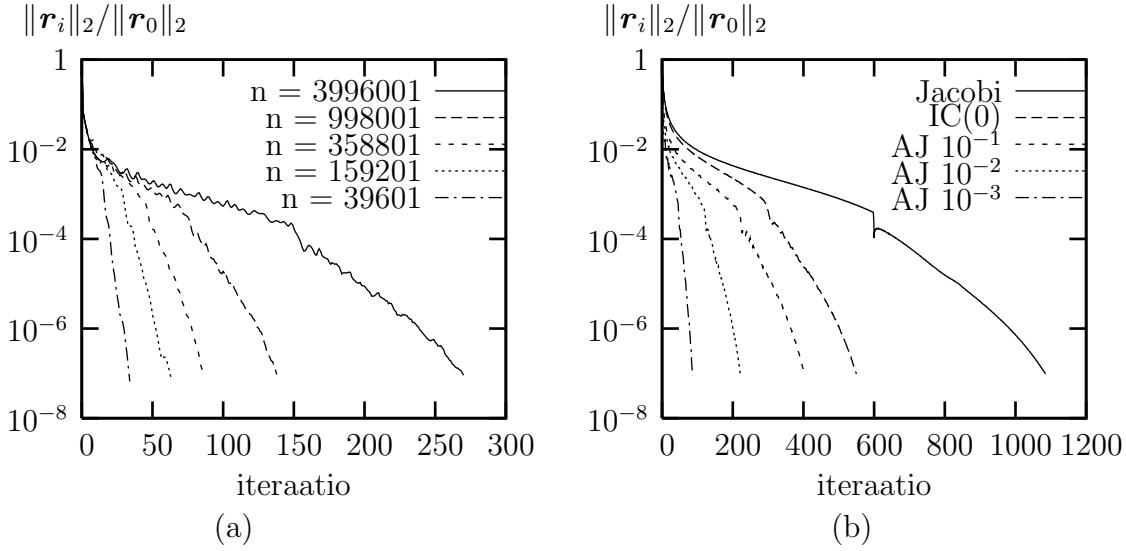
$$u = \begin{cases} 0, & \text{kun} \quad x = 0, \quad \text{ja} \quad y = 0, \\ (x/L)u_0, & \text{kun} \quad y = L, \\ (y/L)u_0, & \text{kun} \quad x = L. \end{cases} \tag{61}$$

Kulkeutumisen ja johtumisen voimakkuuseroa kuvaa Pécletin luku Pe, joka määritellään

$$\text{Pe} = \rho c |\vec{v}| L/k. \tag{62}$$

Kaikki materiaaliparametrit, lämmönjohtavuus k, tiheys ρ ja lämpökapasiteetti c ovat vakioita. Lisäksi virtausnopeus on otaksuttu vakioksi $\vec{v} = -(\cos\alpha\vec{i} + \sin\alpha\vec{j})\text{Pe}k/(\rho c L)$,

[6] Iteratiiviset ratkaisualgoritmit ovat pääosin tekijöiden ohjelmoimia lukuunottamatta GMRES-kiihdytintä ja ILUT-pohjustinta, jotka ovat Yousef Saadin SPARSKIT-ohjelmistosta, sekä SAINV- ja RIC2S-pohjustimia, jotka ovat Miroslav Tůman SPARSLAB-kokoelmasta. Yhtälöiden permutointiohjelmat ovat lähteistä [30, 42]



Kuva 1. Pohjustetun liittogradienttimenetelmän suppeneminen bilineaarisilla elementeillä diskretoidussa kaksiulotteisessa lämmönjohtumisongelmassa (Pe = 0). (a) Elementtiverkon tiheyden vaikutus iteraation suppenemiseen, kun pohjustimena on Ajizin ja Jenningsin RIC1 pohjustin ja hylkäysparametrilla arvo 10^{-3}, jonka tuottamassa pohjustimessa on 3,5–3,6 kertaa enemmän alkioita kuin itse matriisissa. (b) Pohjustimen vaikutus, kun pohjustimena on IC(0) tai RIC1 pohjustin kolmella eri hylkäysparametrin arvolla kun $n = 358801$ (600×600 -verkko).

jossa α on kulkeutumissuunnan ja $x-$akselin välinen kulma. Mikäli kulkeutumista ei ole (Pe = 0), puhtaan lämmönjohtumisongelman ratkaisu on $u(x, y) = (x/L)(y/L)u_0$, ja tällöin elementtimenetelmäratkaisu on tarkka.

Alue on jaettu tasaväliseen $k \times k$ -elementtiverkkoon. Mikäli lämmön kulkeutuminen otetaan huomioon, elementtimenetelmänä on käytetty Petrovin-Galerkinin tapaa ja elementtinä nelisolmuista bilineaarista elementtiä, jossa painofunktiot on muodostettu Brooksin ja Hughesin esittämällä tavalla [15].

Tarkastellaan ensin liittogradienttimenetelmän käyttäytymistä pelkän lämmönjohtumisongelman ratkaisussa. Kuvassa 1a on esitetty iteraation suppeneminen, kun pohjustimena on Ajizin ja Jenningsin kehittämä luotettava IC pohjustin hylkäysparametrin arvolla $\psi = 10^{-3}$, jolloin pohjustinmatriisin $\boldsymbol{M}$ alkioiden lukumäärä on noin 3,5-3,6 kertainen matriisin alkioiden lukumäärään verrattuna. Elementtinä on käytetty nelisolmuista bilineaarista elementtiä, jolloin matriisin $\boldsymbol{A}$ nollasta eroavien alkioiden lukumäärä $nz(\boldsymbol{A})$ on luokkaa $5n$, jossa $n = (k-1)(k-1)$. Iteraation suppenemisen hidastuminen ratkaistavan systeemin koon kasvaessa on selvästi nähtävissä.

Epätäydellisillä hajotelmapohjustimilla laskettaessa saadaan iteraatio suppenemaan verkosta riippumattomalla nopeudella, mikäli pohjustimen täyttymisaste suhteessa täydelliseen hajotelmaan on aina sama. Täydellisessä Choleskyn hajotelmamatriisissa $\boldsymbol{L}$ on alkioita luokkaa $n^{3/2}$. Esimerkiksi kuvan 1a 200 × 200 -elementtiverkon tapauksessa ($n =$ 39601) pohjustimessa on alkioita 8,7 % täydelliseen hajotelmamatriisiin verrattuna, kun vastaavasti 1000 × 1000 -verkon ($n = 998001$) tapauksessa luku on vain 1,8 %.

Laskentojen ratkaisuaikoja on esitetty taulukossa 1, jossa p-aika tarkoittaa pohjustimen muodostamisaikaa ja i-aika kiihdytiniteraation vaatimaa laskenta-aikaa. Pohjustimen parantamisen vaikutus iteraation suppenemiseen on esitetty kuvassa 1b ja vastaavia ratkaisuaikoja taulukossa 2. Ajizin ja Jenningsin pohjustimesta käytetään symbolia RIC1 ja Kaporinin stabiloidusta “toisen kertaluvun” pohjustimesta symbolia RIC2S lähteen [37]

Taulukko 1. Kaksiulotteisen lämmönjohtumisongelman (Pe = 0) ratkaisuaikoja Ajizin ja Jenningsin RIC1 pohjustinta käyttäen erilaisilla elementtiverkon tiheyksillä.

verkko k	n	ψ	pohj. tih.	it	p-aika	i-aika	p+i	SKYB	SKY
400	159201	$5 \cdot 10^{-4}$	4,4	51	0,6	4,6	5,2	11,5	17.9
600	358801	10^{-4}	8,0	37	2,3	11,8	14,1	53,9	103,2
1000	998001	$5 \cdot 10^{-5}$	10,7	48	13,6	62,7	76,3	447,9	6626,0*
1500	2247001	$5 \cdot 10^{-5}$	10,8	66	24,4	249,6	274,0	-	-
2000	3996001	$5 \cdot 10^{-5}$	10,8	94	71,9	422,7	493,6	-	-

* Ratkaisija-aliohjelma on käännetty `-O2` -optimointivalitsimella.

Taulukko 2. Kaksiulotteisen lämmönjohtumisongelman (Pe = 0) ratkaisuaikoja kolmea eri pohjustinta käyttäen, kun $n = 358801$ (600×600 -verkko).

pohjustin	pohj. tih.	ψ	it	p-aika	i-aika	p+i
*	-	-	1085	-	45,3	45,3
IC(0)	1,0	-	550	0,1	47,1	47,2
RIC1	1,0	10^{-1}	369	0,2	29,9	30,1
RIC1	1,6	10^{-2}	202	0,4	20,2	20,6
RIC1	3,6	10^{-3}	79	0,7	18,9	19,6
RIC1	8,0	10^{-4}	37	2,3	11,8	14,1
RIC1	14,6	$2 \cdot 10^{-5}$	20	5,0	10,6	15,6
RIC1	19,0	10^{-5}	15	7,4	12,5	19,9
RIC2S	2,0	10^{-2}	202	1,3	23,9	25,2
RIC2S	5,3	10^{-3}	37	8,6	14,0	22,6

* Pohjustamaton liittogradienttimenetelmä.

mukaisesti.

Mikäli alue diskretoidaan lineaarisilla kolmioelementeillä, saadaan kerroinmatriisi jonka ominaisarvot ja siten häiriöalttius voidaan laskea analyyttisesti. Häiriöalttius on

$$\kappa = \frac{1 + \cos(\pi/k)}{1 - \cos(\pi/k)}, \tag{63}$$

joka verkolla $k = 600$ antaa arvoksi $\kappa = 1,459 \cdot 10^5$. Tällöin arvio (37) antaa iteraatioiden määräksi 3211 ($\epsilon = 10^{-7}$). Pohjustamaton liittogradienttimenetelmä suppenee kuitenkin 1543:lla iteraatiolla. Suppenemiseen vaikuttavat häiriöalttiuden lisäksi myös ominaisarvospektrin muut kuin uloimmat ominaisarvot. Kyseisessä tehtävässä on symmetriasta johtuen suuri joukko kaksinkertaisia ominaisarvoja.

Tutkitaan vielä IC(0), RIC1 ja RIC2S pohjustetun liittogradienttimenetelmän suppenemista käytettäessä korkeampiasteisia elementtejä. Taulukossa 3 on esittetty tulokset laskettaessa kvadraattisilla 6-solmuisilla (P2) ja bikvadraattisilla 9-solmuisilla (Q2) elementeillä ja 300×300 -verkolla sekä bikuubisilla 16-solmuisilla (Q3) elementeillä ja 200×200 -elementtiverkolla. Tuntemattomien lukumäärä on kaikissa tapauksissa 358801. Havaitaan, että korkeamman asteen elementtejä käytettäessä kerroinmatriisissa on huomattavasti enemmän alkioita, vaikka tuntemattomia on sama määrä.

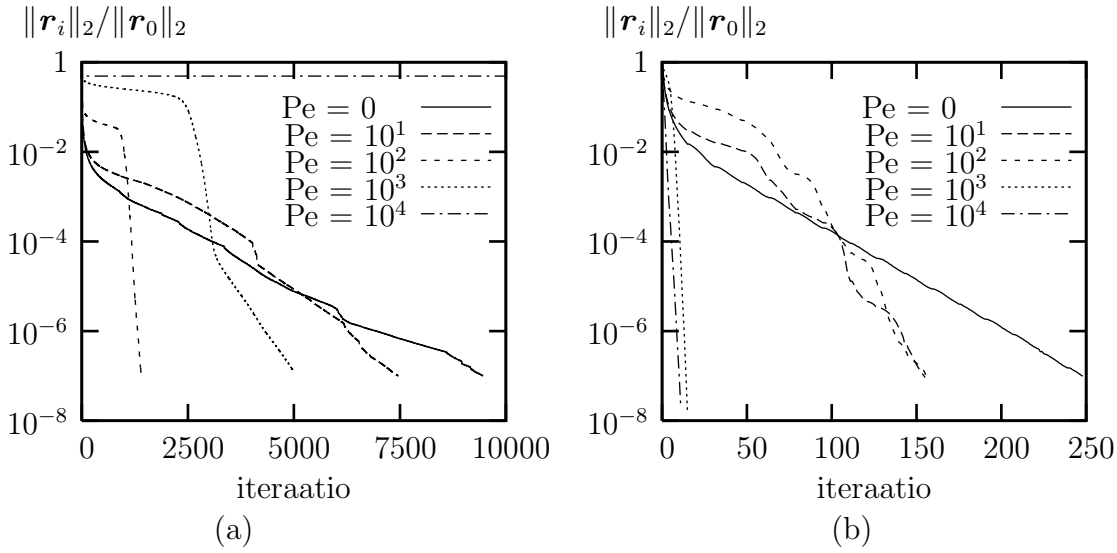
Taulukko 3. Kaksiulotteisen lämmönjohtumisongelman (Pe = 0) ratkaisuaikoja kolmea eri pohjustinta ja elementtityyppiä käyttäen, kun $n = 358801$ ja nollasta eroavien alkioiden lukumäärä elementeittäin: P2 – $nz(\boldsymbol{A}) = 2235027 \sim 6n$, Q2 – $nz(\boldsymbol{A}) = 3037841 \sim 8,5n$, Q3 – $nz(\boldsymbol{A}) = 4640485 \sim 13n$.

elementti	pohjustin	ψ	pohj. tih.	it	p-aika	i-aika	p+i	SKYB
P2	IC(0)	-	1,0	1373	0,0	218,2	218.2	142,8
	RIC1	10^{-2}	1,1	226	0,3	35,1	35,4	
	RIC1	10^{-3}	2,7	99	0,8	28,8	29,6	
	RIC2S	10^{-2}	1,3	127	4,3	21,6	25,9	
	RIC2S	10^{-3}	3,8	44	10,1	17,0	27,1	
Q2	IC(0)	-	1,0	2014	0,3	260,8	261,1	146,5
	RIC1	10^{-2}	1,0	236	0,5	38,7	39,2	
	RIC1	10^{-3}	2,3	97	1,2	26,3	27,5	
	RIC2S	10^{-2}	1,1	118	2,4	28,5	30,9	
	RIC2S	10^{-3}	3,3	38	17,4	16,9	34,3	
Q3	IC(0)	-	1,0	5102	0,3	1084,9	1085,2	260,3
	RIC1	10^{-2}	0,9	266	0,7	95,4	96,1	
	RIC1	10^{-3}	1,8	137	1,6	76,2	77,8	
	RIC2S	10^{-2}	0,9	134	3,0	38,2	41,2	
	RIC2S	10^{-3}	2,4	43	22,8	24,0	46,8	

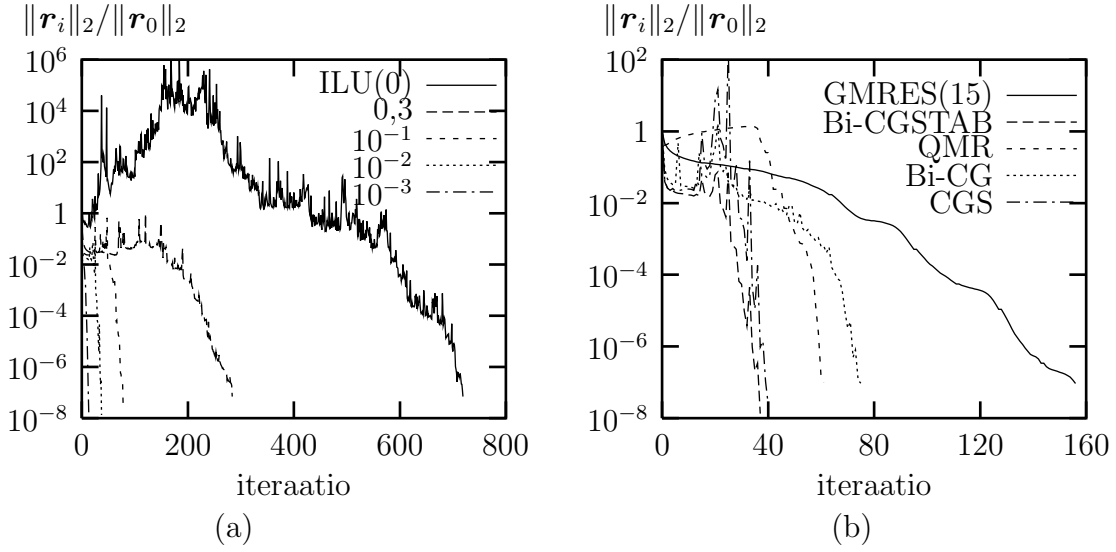
Kulkeutumisen huomioonottaminen johtaa epäsymmetriseen matriisiin. Kuvassa 2 on esitetty GMRES(15) menetelmän suppeneminen virtausnopeuden muuttuessa käyttäen joko yksinkertaista staattisen harvuusrakenteen ILU(0) tai dynaamisesti muodostettua ILUT(ψ, p) pohjustinta. Pelkkä lämmönjohtumisongelma (Pe = 0) osoittautuu yllättävän vaativaksi ongelmaksi. Kun Pécletin luku on 10^4 tai suurempi, ILU(0) pohjustetun GMRES(15) iteraation suppeneminen pysähtyy. Kulkeutumisen dominoimat tapaukset eivät näytä olevan ongelma ILUT(ψ, p) pohjustinta käytettäessä, vaan suppeneminen suurilla Pécletin luvun arvoilla tapahtuu hyvin nopeasti. Kuvan esimerkissä hylkäysparametrillä on arvo 10^{-2} ja tilaparametri $p = 100$, mikä johtaa pohjustimiin, joissa on noin kaksinkertainen määrä alkioita itse matriisiin verrattuna.

Kuten kuvasta 2 voidaan todeta, on pohjustimen vaikutus merkittävä. GMRES iteraation suppeneminen ILU(0) pohjustimella on yllättävän huono verrattaessa sitä Bi-CGSTAB menetelmään, kuva 3a. ILU(0) pohjustin vaatii miltei kymmenkertaisen määrän iteraatioita verrattuna sellaiseen ILUT pohjustimeen ($\psi = 10^{-1}$ kuvassa 3a), jossa on vain 20 % enemmän alkioita.

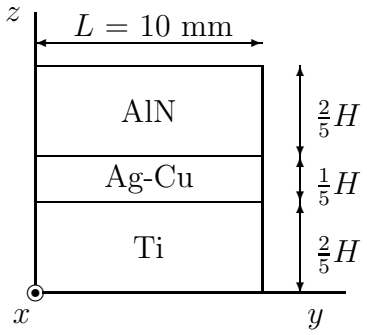
Kuvassa 3b on esitetty eri iteraatiomenetelmien suppeneminen käytettäessä ILUT pohjustinta, jossa on noin kaksinkertainen määrä alkioita itse matriisiin verrattuna ($\psi = 10^{-2}$). On huomattava, että GMRES iteraatio vaatii iteraatioaskelta kohti vain yhden matriisi-vektorikertomisen ja pohjustinoperaation, kun taas CGS ja Bi-CGSTAB vaativat näitä operaatioita kaksinkertaisen määrän. Bi-CG ja QMR menetelmissä kunkin iteraatioaskeleen pääoperaatiot ovat vektorin kertominen matriisilla ja sen transpoosilla sekä pohjustimella ja sen transpoosilla operoiminen. Tässä esimerkissä CGS ja Bi-CGSTAB ovat laskenta-ajassa mitattuna nopeimmat menetelmät.



Kuva 2. GMRES(15)-iteraation suppeneminen kaksiulotteisessa lämmönsiirtymisongelmassa erilaisilla Pécletin luvun Pe arvoilla, kun pohjustimena on (a) ILU(0) tai (b) ILUT(10^{-2},100) ja $n = 159201$ (400×400 -verkko). Virtaussuunnan ja x-akselin välinen kulma on $\alpha = 60°$.



Kuva 3. (a) Bi-CGSTAB menetelmän suppeneminen ILU(0) ja ILUT(ψ,100) pohjustimilla eri hylkäysparametrin ψ arvoilla. (b) Eri iteraatiomenetelmien suppeneminen ILUT(10^{-2},100) pohjustimella. Molemmissa esimerkeissä on Pécletin luku Pe = 100 ja tasainen 400×400 -elementtiverkko, $n = 159201$. Virtaussuunnan ja x-akselin välinen kulma on $\alpha = 60°$.



	Ti	Ag-Cu	AlN
E, [GPa]	118.6	94.4	315
ν, [-]	0.36	0.36	0.25
$10^6\alpha$, [-]	6.441	17.995	2.85

Kuva 4. Kolmen materiaalikerroksen kappale.

Taulukko 4. Kahden suoran ratkaisijan ratkaisuaikoja kahdelle lämpöjännitysongelmalle.

			SKYB [43]		SKY [7, 36]	
ongelma	n	rms-nauha	h-aika	t-aika	h-aika	t-aika
kolme mat. $20 \times 20 \times 20$	27777	1355	20,7	0,5	147,7	0,5
kolme mat. $40 \times 40 \times 40$	206757	5105	2151,5	8,4	-	-
kokillitela	230423	1552	277,2	7,4	-	-

Lämpöjännitysongelmia

Hopea-kupari juotteella (Ag-Cu) liitetyn keraamin (AlN) ja titaanilevyn (Ti) muodostama särmiö on esitetty kuvassa 4, jossa on annettu myös isotrooppisiksi ja lineaarisesti kimmoisiksi otaksuttujen kerrosten materiaaliparametrit. Kappale muodostuu alueesta $(x, y, z) \in (0, L)\times(0, L)\times(0, H)$. Materiaalin rajapinnat ovat xy-tason suuntaiset vaakatasot $z = 2H/5$ ja $z = 3H/5$. Kappale on tuettu siten, että jäykän kappaleen liike on estetty: siirtymät kaikkiin suuntiin on estetty pisteessä $(x, y, z) = (0, 0, 0)$, sekä x-suuntaan pisteessä (L, L, H), y-suuntaan pisteessä $(L, 0, 0)$ ja z-suuntaan pisteessä $(L, L, 0)$. Kuormituksena on tasainen lämpötilan muutos koko rakenteessa. Alueeseen on luotu tasavälinen kahdeksansolmuisista trilineaarisista elementeistä koostuva $k \times k \times k$ -elementtiverkko.

Taulukossa 4 on esitetty suorien ratkaisijoiden vaatima $\boldsymbol{LDL}^T$-hajotelman tekoaika (h-aika) ja kuormavekrorin redusointiin ja takaisinsijoitukseen kulunut laskenta-aika (t-aika). Taulukon luvuista havaitaan selvästi "dimensionaalisuuden kirous"– nauhanleveys kasvaa hyvin suureksi, jolloin tilantarve on suuri ja laskenta-aika kasvaa suhteessa $n^{2\frac{1}{3}}$.

Taulukoissa 5 ja 6 on esitetty pohjustetun liittogradienttimenetelmän vaatimia ratkaisuaikoja kolmella eri kappaleen paksuudella, jolloin elementtien sivusuhde muuttuu satakertaiseksi. Tästä on seurauksena suppenemisen hidastuminen kaikilla muilla paitsi Kaporinin RIC2S pohjustimella. Taulukoissa on annettu myös oikealle katsovan stabiloidun AINV menetelmän (SAINV) tulokset. Suppeneminen SAINV pohjustimella on RIC1 pohjustimen luokkaa, tosin pohjustimen muodostamisaika on huomattavasti suurempi. On syytä muistaa, että käänteismatriisipohjustimet ovat soveliaampia rinnakkaislaskennassa. SAINV pohjustimen tapauksessa vapausasteet on uudelleennumeroitu minimiastemenetelmän mukaisesti (MMD = Multiple-Minimum-Degree). SAINV pohjustin kärsii eniten elementin sivusuhdevääristymästä.

Iteratiiviset ratkaisijat ovat tiheämmän elementtiverkon tapauksessa kuitenkin kertaluokkaa nopeampia kuin suorat ratkaisijat. Kuten taulukon luvuista havaitaan, on

Taulukko 5. Kolmen materiaalin kappaleen lämpöjännitysongelman ratkaisuaikoja kolmea eri pohjustinta käyttäen, kun $n = 27777$ ($20 \times 20 \times 20$ -verkko).

H/L	pohjustin	ψ	pohj. tih.	it	p-aika	i-aika	p+i
1,0	RIC1	10^{-2}	0,8	215	0,3	8,1	8,4
	RIC1	$5 \cdot 10^{-3}$	1,0	188	0,7	12,7	13,4
	RIC1	10^{-3}	2,2	113	1,4	13,9	15,3
	RIC2S	10^{-2}	0,8	110	2,8	4,6	7,4
	RIC2S	$5 \cdot 10^{-3}$	1,2	84	5,4	5,5	10,9
	RIC2S	$2 \cdot 10^{-3}$	2,0	59	21,6	4,8	26,4
	SAINV	10^{-2}	1,2	199	6,9	9,8	16,7
0,1	RIC1	10^{-3}	1,1	415	0,7	23,2	23,9
	RIC1	10^{-4}	2,8	174	1,8	19,5	21,3
	RIC2S	10^{-3}	1,0	88	5,0	5,8	10,8
	RIC2S	10^{-4}	2,8	45	34,1	4,0	38,1
	SAINV	10^{-2}	1,6	354	9,6	31,0	40,6
0,01	RIC1	10^{-6}	3,1	395	2,2	35,6	37,8
	RIC2S	10^{-5}	1,4	376	7,5	20,4	27,9
	RIC2S	10^{-6}	2,7	71	19,7	5,5	25,2
	RIC2S	$7 \cdot 10^{-7}$	2,9	46	21,9	3,3	25,2
	SAINV	10^{-2}	7,2	1092	73,3	207,6	280,9

hylkäysparametrin valinta ongelmallista. Muuttamalla kappaleen sivusuhdetta kohti ohuempaa rakennetta, on hylkäysparametriksi valittava yhä pienempi luku, jotta saataisiin samankokoinen pohjustin.

Kaikissa edeltävissä esimerkeissä Kaporinin RIC2S pohjustimen väliaikaistulokset on laskettu tarkasti, eli hylkäysparametrilla ψ_2 on ollut arvona nolla. Käyttämällä tälle parametrille nollasta poikkeavaa arvoa pohjustimen muodostamisaikaa ja tilapäisen muistitilan tarvetta voidaan pienentää. Taulukon 6 esimerkeissä on väliaikaistulosten hylkäysparametri valittu väliltä $\psi/1000$ - $\psi/50$, jolloin pohjustimen teho on pysynyt hyvänä. Erityisesti on huomattava nopea suppeneminen vaikeimmassa tehtävässä, kun $H/L = 0,01$.

Toisessa lämpöjännitysongelmassa tarkastellaan soft-kalanterin kokillitelaa. Mallin data on saatu Pentti Varpasuolta (Fortum Nuclear Services Ltd.). Mallin elementtiverkko koostuu 69768 trilineaarisesta kahdeksansolmuisesta elementistä. Tuntemattomia on 230423. Mutta rakenteen hoikkuuden vuoksi jäykkyysmatriisin RMS-nauhanleveys on vain 1552, kun yhtälöt on uudelleennumeroitu tässä tapauksessa käänteistä Cuthill-McKee algoritmia paremman tuloksen antavalla Gibs-King algoritmilla. Mallissa on seitsemän eri materiaalia, joiden kimmokertoimet vaihtelevat välillä 110–186 GPa ja lämpöpitenemiskertoimet välillä $9,3 \cdot 10^{-6}$–$1,08 \cdot 10^{-5}$.

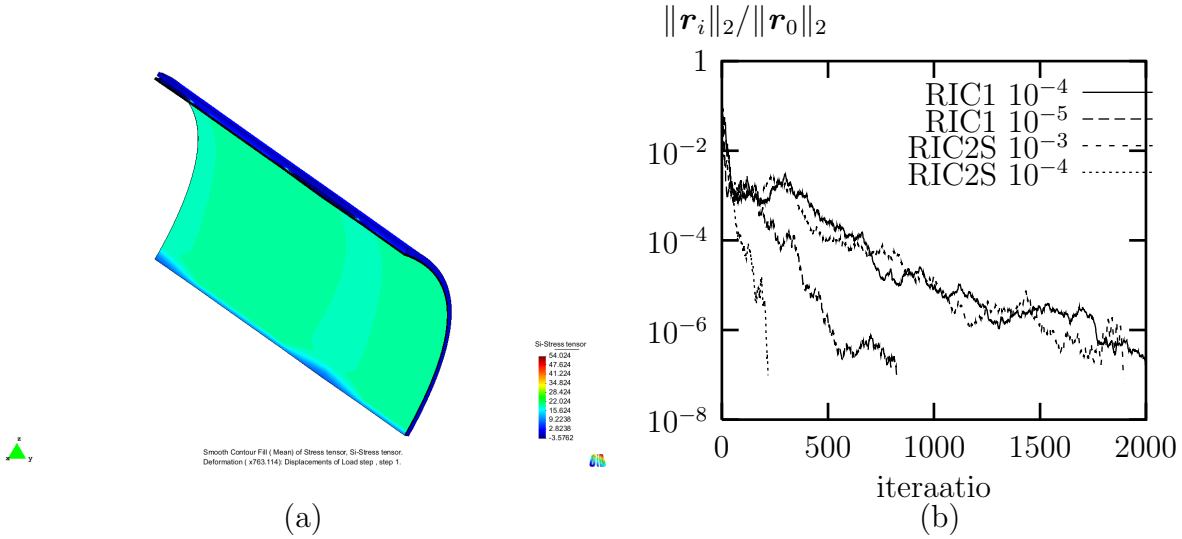
Probleema on varsin haastava pohjustetuille iteraatioille; katso taulukko 7 ja vertaa taulukkoon 4. Tämä johtuu elementtien suuresta sivusuhdepoikkeamasta; elementit ovat hyvin ohuita sivusuhdepoikkeaman ollessa pahimmillaan luokkaa $3 \cdot 10^{-3}$ ja parhaimmillaankin luokkaa $6 \cdot 10^{-2}$. Kaporinin stabiloitu RIC2S pohjustin on tässä esimerkissä huomattavasti tehokkaampi kuin Ajizin ja Jenningsin RIC1 pohjustin. Tosin sekin häviää ratkaisuajassa suoralle ratkaisijalle. Tiheämmän pohjustimen muodostamisaika on suoran ratkaisijan hajotelman tekoajan luokkaa.

Taulukko 6. Kolmen materiaalin kappaleen lämpöjännitysongelman ratkaisuaikoja kolmea eri pohjustinta käyttäen, kun $n = 206757$ ($40 \times 40 \times 40$ -verkko).

H/L	pohjustin	ψ	ψ_2	pohj. tih.	it	p-aika	i-aika	p+i
1,0	RIC1	$5 \cdot 10^{-3}$		1,0	477	9,6	127,7	137,3
	RIC1	10^{-3}		2,2	287	13,9	121,4	135,3
	RIC2S	$5 \cdot 10^{-3}$	10^{-4}	1,1	193	30,0	79,3	109,3
	RIC2S	10^{-3}	10^{-5}	2,8	102	307,8	61,4	369,2
	SAINV	10^{-2}		1,1	443	55,9	180,5	236,4
0,1	RIC1	10^{-3}		1,1	866	4,1	318,8	322,9
	RIC2S	$5 \cdot 10^{-3}$	10^{-5}	0,5	388	16,9	92,5	109,4
	RIC2S	10^{-3}	10^{-5}	1,0	166	22,1	61,5	83,6
	RIC2S	10^{-4}	10^{-7}	3,3	88	577,4	58,6	630,0
	SAINV	10^{-2}		1,6	847	106,1	455,7	561,8
0,01	RIC1	10^{-6}		3,6	1377	22,8	1261,1	1283,9
	RIC1	10^{-8}		9,0	182	112,5	341,8	454,3
	RIC2S	10^{-5}	10^{-7}	1,7	812	47,2	425,0	472,2
	RIC2S	10^{-6}	10^{-8}	3,2	80	66,6	52,9	119,5

Taulukko 7. Soft-kalanterin kokillitelan lämpöjännitysongelman ratkaisuaikoja kahta eri pohjustinta käyttäen, kun $n = 230423$ (69768 trilineaarisesta kahdeksansolmuisesta elementtiä).

pohjustin	ψ	ψ_2	pohj. tih.	it	p-aika	i-aika	p+i
RIC1	10^{-4}		3,3	2005	25,9	2360,4	2386,3
RIC1	10^{-5}		8,1	824	95,7	1851,1	1946,8
RIC1	10^{-6}		15,6	297	305,1	660,2	965,3
RIC2S	10^{-3}	10^{-6}	1,1	1864	59,4	779,6	839,0
RIC2S	$2 \cdot 10^{-4}$	10^{-6}	2,7	435	167,5	308,2	475,7
RIC2S	10^{-4}	10^{-6}	3,9	229	273,1	217,5	490,6



(a) (b)

Kuva 5. Soft-kalanterin kokillitelan lämpöjännitysongelman (a) geometria ja (b) iteraatioiden suppeneminen, kun $n = 230423$ (69768 trilineaarisesta kahdeksansolmuisesta elementtiä).

Elderin tehtävä

Elderin tehtävä on kaksiulotteinen vertailutesti, jossa simuloidaan nesteen lämpölaajenemisesta tai konsentraatioeroista aiheutuvan tiheysvaihtelun seurauksena syntyvää virtausilmiötä. Tarkastellaan kuvassa 6 esitetyssä suorakaiteenmuotoisessa veden kyllästämän huokoisen maan täyttämässä alueessa tapahtuvaa suolapitoisen veden virtauksen Elderin tehtävää [60]. Tehtävän lähteessä [34] esitetty matemaattinen malli koostuu huokosveden ja suolapitoisuuden jatkuvuusyhtälöistä

$$\eta \left(\kappa_{\mathrm{f}} \frac{\partial p}{\partial t} + \zeta_{\mathrm{c}} \frac{\partial C}{\partial t} \right) + \eta \left(\kappa_{\mathrm{f}} \operatorname{grad} p + \zeta_{\mathrm{c}} \operatorname{grad} C \right) \cdot \boldsymbol{J}_{\mathrm{f}} + \operatorname{div} \boldsymbol{J}_{\mathrm{f}} = 0, \tag{64}$$

$$\eta \frac{\partial C}{\partial t} + \operatorname{grad} C \cdot \boldsymbol{J}_{\mathrm{f}} + \left(\kappa_{\mathrm{f}} \operatorname{grad} p + \zeta_{\mathrm{c}} \operatorname{grad} C \right) \cdot \boldsymbol{J}_{\mathrm{c}} + \operatorname{div} \boldsymbol{J}_{\mathrm{c}} = 0, \tag{65}$$

sekä Darcyn ja Fickin yhtälöistä

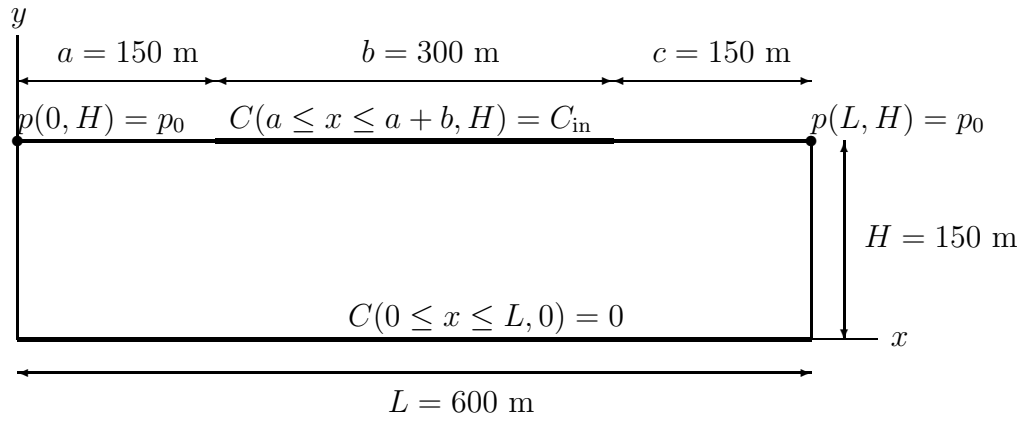
$$\boldsymbol{J}_{\mathrm{f}} = -\frac{k}{\mu} \left[\operatorname{grad} p - \left((1 - C) \bar{\rho}_{\mathrm{w}} + C \bar{\rho}_{\mathrm{c}} \right) \boldsymbol{g} \right], \tag{66}$$

$$\boldsymbol{J}_{\mathrm{c}} = -\eta D \left[a_1 \operatorname{grad} C - a_0 C (1 - C) (\bar{\rho}_{\mathrm{c}} - \bar{\rho}_{\mathrm{w}}) \boldsymbol{g} \right], \tag{67}$$

missä p on huokosveden paine, C huokosveden molaarinen suolapitoisuus, $\boldsymbol{J}_{\mathrm{f}}$ huokosveden Darcyn nopeus, $\boldsymbol{J}_{\mathrm{c}}$ suolaisuuden diffuusionopeus, η huokoisuus, $\kappa_{\mathrm{f}}(p)$ huokosveden kokoonpuristuvuus, $\zeta_{\mathrm{c}}(C)$ huokosveden tiheyden suolaisuuskerroin, k huokoisen aineen läpäisevyys eli permeabiliteetti, $\mu(C)$ huokosveden viskositeetti, $\bar{\rho}_{\mathrm{w}}(p, C)$ ja $\bar{\rho}_{\mathrm{c}}(p, C)$ puhtaan veden ja liuenneiden suolojen ominaistiheydet, $\boldsymbol{g}$ putoamiskiihtyvyys, D diffuusio-dispersiokerroin ja $a_1(p, C)$ ja a_0 materiaalikertoimia. Tehtävän parametrit on esitetty taulukossa 8 ja lähteessä [34].

Tehtävän oleelliset reunaehdot on esittey kuvassa 6 ja taulukossa 8; muutoin reunat ovat läpäisemättömiä. Alkuhetkellä huokosvesi on suolaton ja hydrostaattisessa tilassa: $C = 0$ ja $p = p_0 + \bar{\rho}_{\mathrm{w}} g (H - y)$. Lisäksi tehtävä on isoterminen.

Symmetriaa hyödyntäen ratkaisualueeksi valittiin tarkastelualueen vasen puolikas eli $(x, y) \in (0, L/2) \times (0, H)$, ja se diskretoitiin 343044 lineaarisella kolmioelementtillä, jolloin vapausasteiden lukumääräksi saatiin $n = 343643$. Epälineaarinen tehtävä ratkaistiin



Kuva 6. Elderin tehtävän ratkaisualue ja oleelliset reunaehdot; muutoin reunat ovat läpäisemättömiä.

Taulukko 8. Elderin tehtävän parametrejä.

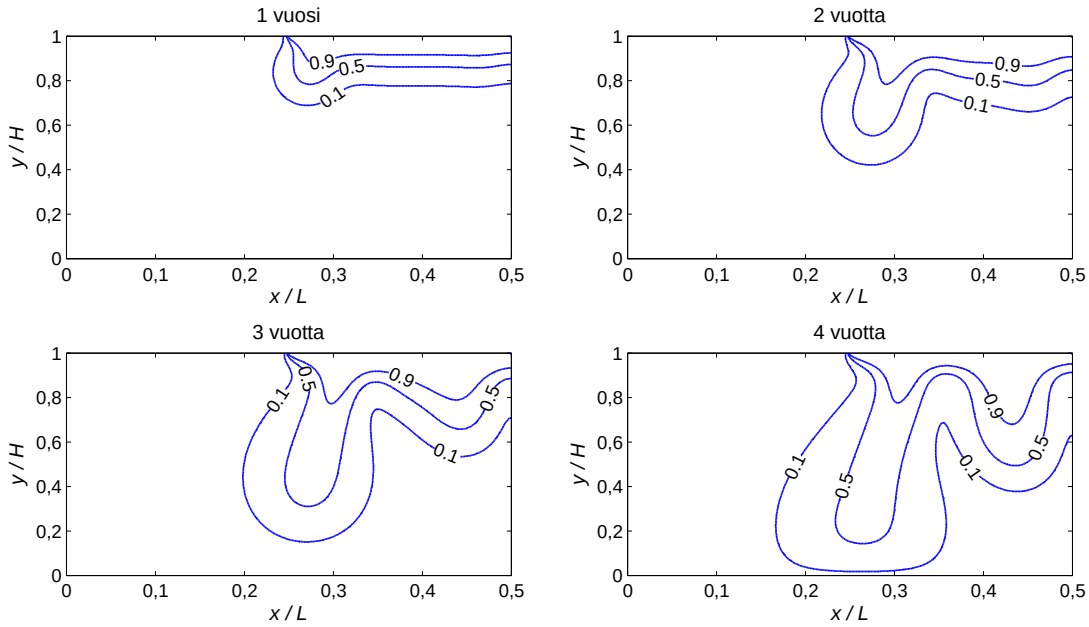
huokoisuus η, [-]	0,1
läpäisevyys k, [m^2]	$4{,}8 \cdot 10^{-13}$
dispersiokerroin D, [m^2/s]	$3{,}6 \cdot 10^{-6}$
molaarinen suolapitoisuus C_{in}, [-]	0,27*
painereunaehto p_0, [Pa]	101325
isoterminen lämpötila T, [°C]	10

* vastaa suolaisen veden tiheyttä 1200 kg/m^3

käyttäen Newtonin-Raphsonin menetelmää ja täysin implisiittistä yhden askeleen Eulerin menetelmää integroimalla 4 vuoden tarkastelujakso 80 aika-askeleella ja askeleen pituudella 0,05 vuotta. Linearisoitu systeemi ratkaistiin ILUT(ψ, 60)-pohjustetulla Bi-CGSTAB -menetelmällä erilaisilla hylkäysparametrien ψ arvoilla. Pohjustin päivitettiin jokaisella iteraatiolla.

Kuvassa 7 on esitetty huokosveden suolapitoisuuden kehittyminen vuoden välein. Tarkastelun alkuhetkellä olleen puhtaan huokosveden suolapitoisuus kasvaa vähitellen alueen yläreunan rajatulta pituudelta b olevasta lähteestä käynnistäen huokosveden tiheyseroista aiheutuvan virtauksen ja sen mukana tapahtuvan suolojen kulkeutumisen.

Laskentojen iteraatioiden tuloksia on esitetty taulukossa 9. Tulosten perusteella ratkaisuaika näyttäisi saavuttavan optimin, kun pohjustimen keskimääräinen tiheys on hieman yli 3, ja että tehtävän luonteesta johtuen yksittäisen linearisoidun tehtävän ratkaisun iteraatioiden määrä kasvaa selvästi, kun pohjustimen tiheys on pienempi kuin 2. Vertailun vuoksi yhden tehtävän suoran ratkaisijan [20] vaatima ratkaisuaika on noin 1800 s eli satakertainen iteratiiviseen ratkaisijaan nähden.



Kuva 7. Elderin tehtävän huokosveden suhteellinen suolapitoisuus pitoisuuden C_{in} suhteen määrittynä ajanhetkillä $t = 1$ vuosi, $t = 2$ vuotta, $t = 3$ vuotta ja $t = 4$ vuotta. Symmetrian vuoksi vain puolet ratkaisusta on esitetty.

Taulukko 9. Elderin tehtävän ILUT$(\psi, 60)$-pohjustetun Bi-CGSTAB -menetelmän ratkaisuaikoja erilaisilla hylkäysparametreilla, kun $n = 343643$ (343044 lineaarista kolmioelementtiä) ja aika-askelten lukumäärä 80. Taulukossa on esitetty linearisoitujen tehtävien ratkaisujen iteraatioiden keskiarvo (ka. it), yksittäisen tehtävän iteraatioiden maksimi määrä (maks. it), kaikkien iteraatioiden määrä yhteensä (yht. it), yksittäisen tehtävän pohjustimen keskimääräinen muodostamisaika (ka. p-aika) ja kiihdytiniteraation keskimääräinen laskenta-aika (ka. i-aika) sekä koko tehtävän kokonaisratkaisuaika (p+i).

ψ	pohj. tih.	ka. it	maks. it	yht. it	ka. p-aika	ka. i-aika	p+i
$1 \cdot 10^{-2}$	1,2	38	359	49608	0,7	18,6	28727
$5 \cdot 10^{-3}$	1,5	36	259	37209	0,9	22,0	23895
$1 \cdot 10^{-3}$	2,6	20	121	20367	2,0	19,2	21420
$5 \cdot 10^{-4}$	3,3	18	61	15529	2,7	14,9	15706
$1 \cdot 10^{-4}$	5,4	9	32	8617	6,5	12,9	18424
$5 \cdot 10^{-5}$	6,3	7	26	7247	9,0	10,1	20034
$1 \cdot 10^{-5}$	7,3	8	24	6427	12,3	12,9	20642
$1 \cdot 10^{-6}$	8,1	11	24	6171	16,7	18,8	20081

Taulukko 10. Iteratiiviset lineaarisen systeemin ratkaisijat kaupallisissa ohjelmissa.

ohjelma	iter. men.	pohjustin	soveltuvuus
ABAQUS V6.8	FETI		SPD ?
ANSYS R11.0	AML		symm., indef.
	CG	Jacobi, IC, ?	SPD
	QMR	?	symm., indef.
FINNSAP	CG	Jacobi, SSOR	SPD
		IC ja EBE tyyppinen IC	SPD
LUSAS	CG	IC	SPD
MSC.NASTRAN	CG	Jacobi, IC	SPD
	Bi-CG, CR	Jacobi	epäsym.
NISA	CG	IC	SPD
ROBOT	CG	IC, AML	SPD

AML = Algebraic Multi Level
CR = Conjugate Residual [50, luku 6.8]

Iteratiiviset ratkaisijat kaupallisissa rakenneanalyysiohjelmistoissa

Iteratiiviset lineaarisen systeemin ratkaisumenetelmät löytyvät nykyään miltei kaikkien rakenneanalyysiohjelmien valikoimasta. Taulukossa 10 on esitetty muutamassa yleisessä rakenneanalyysiohjelmassa olevia iteratiivisia ratkaisumenenetelmiä. Osa tiedoista perustuu vain ohjelmien manuaaleihin, joissa saattaa olla vähän teknistä tietoa käytetyistä algoritmeista ja terminologia voi olla harhaanjohtavaa. Yleisin pohjustintyyppi näyttäisi olevan epätäydellinen Cholesky (IC). Useimmissa ohjelmissa iteratiivisen ratkaisijan käyttö on rajoittunut symmetrisiin positiivisesti definiitteihin tapauksiin. Nastran-ohjelmassa on mahdollisuus liittää mukaan käyttäjän oma ratkaisija.

Lopuksi

Artikkelissa on esitetty katsaus elementtimenetelmässä syntyvän lineaarisen yhtälösysteemin iteratiivisiin ratkaisumenetelmiin. Erityisesti on keskitytty elementtimenetelmän h-versioon, jossa muodostuva lineaarinen yhtälösysteemi on harva. Yleisimmät Krylovin aliavaruusiteraatiot ja tavanomaisimmat yleiskäyttöiset pohjustinstrategiat on esitelty. Iteratiivisen ratkaisun tärkein osa on hyvän pohjustimen muodostaminen. Pohjustimen muodostamisessa tarvittavien kontrolliparametrien valinta vaatii kokemusta, sillä niihin vaikuttavat ongelman parametrit ja elementtien geometria.

Iteratiivisia menetelmiä voidaan menestyksellisesti käyttää myös p-elementtimenetelmässä sekä reunaelementtimenetelmässä, jossa syntyvä yhtälösysteemi on täysi ja epäsymmetrinen. Tällöin matriisi-vektorikertolaskun suorittaminen tapahtuu yleensä nopeaa multipoolimenetelmää tai nopeaa Fourier muunnosta käyttäen.

Kiitokset

Michele Benzille (Emory University, Atlanta USA) ja Miroslav Tůmalle (Institute of Computer Science, Academy of Sciences of the Czech Republic) opastuksesta pohjustettujen iteraatioiden maailmaan sekä Pentti Varpasuolle (Fortum Nuclear Services Ltd.) mahdollisuudesta käyttää kokillitelamallia testiesimerkkinä.

Viitteet

[1] M.A. Ajiz ja A. Jennings. A robust incomplete Cholesky conjugate gradient algorithm. *International Journal for Numerical Methods in Engineering*, 20:949–966, 1984.

[2] W.E. Arnoldi. The principle of minimized iteration in the solution of matrix eigenvalue problem. *Quarterly in Applied Mathematics*, 9:17-29, 1951.

[3] O. Axelsson. *Iterative Solution Methods*. Cambridge University Press, 1994.

[4] O. Axelsson, V.A. Barker, *Finite element solution of boundary value problems, theory and computation*, Academic Press, 1984.

[5] S.T. Barnard, L.M. Bernardo, ja H.D. Simon. An MPI implementation of the SPAI preconditioner on the T3E. Technical Report 40794, Lawrence Berkeley National Laboratory, Berkeley, CA 94720, USA, September 1997.

[6] R. Barrett, M. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, ja H. van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, 1994.

[7] K.J. Bathe. *Finite Element Procedures in Engineering Analysis*. Prentice-Hall, Englewood Cliffs, New Jersey, 1982.

[8] M. Benzi. Preconditioning techniques for large linear systems: a survey. *Journal of Computational Physics*, 182: 418–477, 2002.

[9] M. Benzi, C.D. Meyer, ja M. Tůma. A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM Journal on Scientific Computing*, 15(5):1135–1149, 1996.

[10] M. Benzi ja M. Tůma. A comparative study of sparse approximative inverse preconditioners. *Applied Numerical Mathematics*, 30(2-3):305–340, 1999.

[11] M. Benzi, J.K. Cullum ja M. Tůma. Robust approximative inverse preconditioning for the conjugate gradient method. *SIAM Journal on Scientific Computing*, 22(4):1318–1332, 2000.

[12] M. Benzi ja M. Tůma. Orderings for factorized sparse approximative inverse preconditioners. *SIAM Journal on Scientific Computing*, 21(5):1851–1868, 2000.

[13] M. Benzi, R. Kouhia ja M. Tůma. Stabilized and block approximate inverse preconditioners for problems in solid and structural mechanics. *Computer Methods in Applied Mechanics and Engineering*, 190(49-50):6533–6554, 2001.

[14] N. Bitoulas ja M. Papadrakakis. An optimized computer implementation of incomplete Cholesky factorization. *Computing Systems in Engineering*, 5(3):265–274, 1994.

[15] A.N. Brooks ja T.J.R. Hughes. Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 32(1-2):199–259, 1982.

[16] A.M. Bruaset. *A Survey of Preconditioned Iterative Methods*. Number 328 in Pitman Research Notes in Mathematics Series. Longman Scientific & Technical, 1995.

[17] V.E. Bulgakov. The use of the multi-level iterative aggregation method in 3-D finite element analysis of solid, truss, frame and shell structures. *Computers and Structures*, 63(5):927–938, 1997.

[18] M. Byckling ja M. Huhtanen. Approximate factoring of the inverse. *Numerische Mathematik*, julkaistu verkossa 16 lokatuuta 2010. DOI: 10.1007/s00211-010-0341-4.

[19] P. Deuflhard, R. Freund ja A. Walter. Fast secant methods for the iterative solution of large nonsymmetric linear systems. *Impact of Computing in Science and Engineering*, 2:244-276, 1990.

[20] G. Dhatt ja G. Touzot. *Une Présentation de la Méthode des Eléments Finis*. Maloine S.A. Editeur, 1984, 2. painos.

[21] J.J. Dongarra, I.S. Duff, D.C. Sorensen ja H.A. van der Vorst. *Numerical linear algebra for high-performance computers*. SIAM 1998.

[22] T. Eirola ja O. Nevanlinna. Accelerating with rank-one updates. *Linear Algebra and its Applications*, 121:511-520, 1989.

[23] T. Eirola ja O. Nevanlinna. *Numerical linear algebra; iterative methods*. Kurssin Mat-1.3651 luentomoniste, TKK, 2010.

[24] C. Farhat ja F.X. Roux. A method of finite element tearing and interconnecting and its parallel solution algorithm. *International Journal for Numerical Methods in Engineering*, 32(6): 1205–1227, 1991.

[25] C. Farhat, J. Mandel ja F.X. Roux. Optimal convergence properties of the FETI domain decomposition method. *Computer Method in Applied Mechanic and Engineering*, 115(3-4): 365–385, 1994.

[26] R.W. Freund. A transpose-free quasi-minimal residual algorihm for non-Hermitian linear systems. *SIAM Journal of Scientific Computing*, 14(2):470–482, 1993.

[27] R.W. Freund, G.H. Golub ja N.M. Nachtigal. Iterative solution of linear systems. *Acta Numerica*, 57–100, 1991.

[28] R.W. Freund ja N.M. Nachtigal. QMR: a quasi-minimal residual method for non-Hermitian linear systems. *Numerische Mathematik*, 60(1):315–339, 1991.

[29] R.W. Freund ja N.M. Nachtigal. QMRPACK: a package of QMR algorithms. *ACM Transactions on Mathematical Software*, 22(1):46–77, 1996.

[30] A. George ja J. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall, 1981.

[31] M.H. Gutknecht. Variants of Bi-CGSTAB for matrices with complex spectrum. *SIAM Journal on Scientific Computing*, 14(5):1020–1033, 1993.

[32] A. Greenbaum. *Iterative Methods for Solving Linear Systems*. SIAM, Frontiers in Applied Mathematics, Vol. 17, 1997.

[33] M.J. Grote ja T. Huckle. Parallel preconditioning with sparse approximate inverses. *SIAM Journal on Scientific Computing*, 18(3):838–853, 1997.

[34] J. Hartikainen, R. Kouhia ja T. Wallroth. Permafrost simulations at Forsmark using a numerical 2D thermo-hydro-chemical model. Technical Report SKB TR-09-17, 2010.

[35] M.R. Hestenes ja E.L. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49:409–436, 1952.

[36] T.J.R. Hughes. *The Finite Element Method – Linear Static and Dynamic Finite Element Analysis*. Prentice-Hall, 1987.

[37] I.E. Kaporin. High quality preconditioning of a general symmetric positive definite matrix based on its $U^TU + U^TR + R^TU$-decomposition. *Numerical Linear Algebra with Applications*, 5(6):483–509, 1998.

[38] C.T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*. SIAM, Frontiers in Applied Mathematics, Vol. 16, 1995.

[39] L.Y. Kolotilina ja A.Y. Yeremin. Factorized sparse approximate inverse preconditionings I. Theory. *SIAM Journal on Matrix Analysis and Applications*, 14:45–58, 1993.

[40] C. Lanczos. Solution of systems of linear equations by minimized iterations. *Journal of Research of the National Bureau of Standards*, 49:22–52, 1952.

[41] R.B. Lehoucq, D.C. Sorensen ja C. Yang. *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods* SIAM, 1998.

[42] J.G. Lewis. Implementation of the Gibbs-Poole-Stockmeyer and Gibbs-King algorithms. *ACM Transactions on Mathematical Software*. 8(2):180–189, 1982.

[43] O.A. Marques. A partitioned skyline LDL^T factorization. Technical Report TR/PA/93/53, CERFACS, 1993.

[44] J.A. Meijerink ja H.A. van der Vorst. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Mathematics of Computation*, 31:148–162, 1977.

[45] G. Meurant. *Computer solution of large linear systems*. North-Holland, 1999.

[46] C.C. Paige and M.A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12:617–629, 1975.

[47] J. Pitkäranta, Reuna-arvotehtävien moniverkkoratkaisijat - numeronmurskauksesta numeroiden pehmittelyyn, *Arkhimedes*, vol. 38, 1986, ss. 55-66.

[48] J.K. Reid. On the method of conjugate gradients for the solution of large sparse systems of linear equations. *Large Sparse Sets of Linear Equations*, J.K. Reid (ed.), 231–254, Academic Press, 1971.

[49] Y. Saad. ILUT: a dual treshold incomplete LU factorization. *Numerical Linear Algebra and Applications*, 1:387–402, 1994

[50] Y. Saad. *Iterative methods for sparse linear systems*. PWS Publishing Company, 1996.

[51] Y. Saad ja M.H. Schultz. GMRES: a generalized minimal residual algorithm for solving non-symmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7:856–869, 1986.

[52] Y. Saad ja B. Suchomel. ARMS: an algebraic recursive multilevel solver for general sparse linear systems. *Numerical Linear Algebra and Applications*, 9:359–378, 2002.

[53] P. Saint-Georges, G. Warzee, R. Beauwens ja Y. Notay. High-performance PCG solvers for FEM structural analysis. *International Journal for Numerical Methods in Engineering*, 39:1313–1340, 1996.

[54] G.L.G. Sleijpen ja D.R. Fokkema. Bi-CGSTAB(ℓ) for linear equations involving unsymmetric matrices with complex spectrum. *ETNA*, 1:11–32, 1993.

[55] P. Sonneveld. CGS, a fast Lanczos-type solver for nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 10:36–52, 1989.

[56] G. Strang, *Introduction to applied mathematics*, Wellesley-Cambridge Press, 1986.

[57] M. Tismenetsky. A new preconditioning technique for solving large sparse linear systems. *Linear Algebra and its Applications*, 154–156:331–402, 1991.

[58] H.A. van der Vorst. Bi-CGSTAB: a fast and smoothly convergent variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 13(2):631–644, 1992.

[59] H.A. van der Vorst. *Iterative Krylov methods for large linear systems.* Cambridge University Press, 2003.

[60] C.I. Voss ja W.R. Souza. Variable density flow and solute transport simulation of regional aquifers containing a narrow freshwater-saltwater transition zone. *Water Resources Research*, 23:1851–1866, 1987.

[61] H. Voss. *Iterative methods for linear systems of equations.* University of Jyväskylä, Department of Mathematics, lecture notes 27, 1993.

[62] C. Vuik ja H.A. van der Vorst. A comparison of some GMRES-like methods. *Linear Algebra and its Applications*, 160:131–162, 1992.

Juha Hartikainen, Reijo Kouhia
Aalto-yliopiston teknillinen korkeakoulu
Rakenne- ja rakennustuotantotekniikan laitos
PL 12100, 00076 Aalto
`etunimi.sukunimi@tkk.fi`