LINEAARISEN YHTÄLÖRYHMÄN RATKAISEMINEN CHOLESKI'N MENETELMÄLLÄ

JORMA KÖLIÖ

Rakenteiden Mekaniikka Vol. 5 No 4 1972 ss. 430-438; Rakenteiden Mekaniikan Seura, Helsinki

JOHDANTO

Numeerisissa tietokonemenetelmissä joudutaan usein ratkaisemaan yhtälöryhmiä, joissa tuntemattomien lukumäärä nousee satoihin, jopa tuhansiin. Tällöin kerroinmatriisin tallettamistapa ja laskutoimitusten järjestely vaikuttavat ratkaisevasti sekä laskuaikaan että tarvittavaan muistikapasiteettiin.

Seuraavassa oletetaan, että kerroinmatriisi on positiivisesti definiitti, symmetrinen ja sisältää nollasta poikkeavia termejä vain suhteellisen lähellä diagonaalia. Näillä edellytyksillä laskuaikaa ja vaadittavaa muistitilaa voidaan säästää huomattavasti käytettäessä muuttuvaa nauhanleveyttä Choleski'n menetelmän yhteydessä.

CHOLESKI'N MENETELMÄN PERIAATE

Kaava

$$A \cdot X = C \qquad (1)$$

esittää lineaarista yhtälöryhmää, jossa A on symmetrinen positiivi-

sesti definiitti neliömatriisi, X tuntematon vektori ja C tunnettu vektori. Matriisi A voidaan jakaa kaavan

$$U^T \cdot U = A \qquad (2)$$

mukaisesti matriisituloksi, jossa U on yläkolmiomatriisi ja U^T sen transpoosi. Otetaan käyttöön apuvektori $Y = U \cdot X$, joka ratkaistaan välituloksena yhtälöstä

$$U^T \cdot Y = C \qquad (3)$$

Tuntematon vektori X voidaan tämän jälkeen ratkaista yhtälöstä

$$U \cdot X = Y \qquad (4)$$

Tarvittavat laskutoimitukset suoritetaan seuraavien kaavojen mukaisesti (summauksia ei lasketa ylärajan ollessa alarajaa pienempi).

$$u_{ii} = \sqrt{a_{ii} - \sum_{m=1}^{i-1} u_{mi}^2} \qquad (5)$$

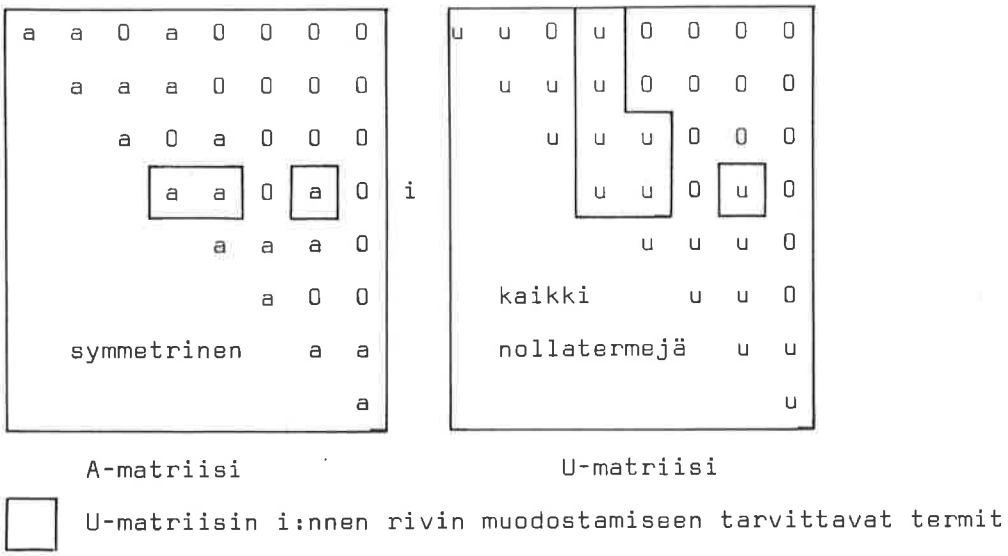
$$u_{ij} = (a_{ij} - \sum_{m=1}^{i-1} u_{mi} \cdot u_{mj})/u_{ii} \quad ; j > i \qquad (6)$$

$$u_{ij} = 0 \quad ; j < i \qquad (7)$$

$$y_i = (c_i - \sum_{m=1}^{i-1} u_{mi} \cdot y_m)/u_{ii} \qquad (8)$$

$$x_i = (y_i - \sum_{m=i+1}^{n} u_{im} \cdot x_m)/u_{ii} \qquad (9)$$

Yhtälöistä (5) ja (6) käy ilmi, että termi u_{ij} on nolla vain, jos kaikki a_{mj} ($m = 1...i$) ovat nollia. Toisin sanoen matriisit A ja U ovat muodoltaan samanlaisia, mutta jokaisella pystyrivillä ylimmän nollasta eroavan a-termin ja lävistäjän väliset nollatermit muuttuvat U-matriisissa nollasta poikkeaviksi. Mikäli RLI(i) on U-matriisin i:nnen vaakarivin nauhanleveys (diagonaalista), niin edellä sanotusta seuraa myös: $RLI(i) \geq RLI(i-1)-1$ ($i = 2...n$).



Kuva 1. Vastaavat A- ja U-matriisit.

INDEKSOINNISTA

Kaavat (5-9) on indeksoitu kaksidimensioisina, mutta tehokkain tilansäästö saavutetaan kerroinmatriisin yksidimensioisella (pystyriveittäin alhaalta ylös) indeksoinnilla. Mikäli RLJ(m) on m:nnen pystyrivin termien lukumäärä (diagonaalilta lukien), termin ij indeksi on $\sum_{m=1}^{j-1} RLJ(m) + j - i + 1$.

Muodostettaessa U-matriisia ratkaistu u-termi kirjoitetaan välittömästi vastaavan a-termin päälle, joten A-matriisinkin indeksoinnissa otetaan mukaan ne nollatermit, joiden paikalle tulevat u-termit ovat nollasta poikkeavia. Tällöin molempien matriisien indeksointi on identtinen ja voidaan käyttää samaa työtilaa.

RATKAISUN KULKU

Ratkaisun ensimmäisessä vaiheessa A-matriisi kolmioidaan vastaavaksi U-matriisiksi (kuva 1, kaavat (2), (5), (6) ja (7)). Kolmiointi tapahtuu riveittäin lävistäjältä lähtien. Termien paikallistamiseksi tarvitaan taulukot RLI(1:NE) ja RLJ(1:NE) sekä aputaulukot IND(1:MAX) ja JX(1:MAX). Edelliseen lasketaan kulloinkin käsiteltävänä olevan vaakarivin termien indeksit ja jälkimmäiseen vastaavien pystyrivien järjestysnumerot. MAX on suurin nauhanleveys ja NE yhtälöiden lukumäärä.

Toisessa vaiheessa ratkaistaan eteenpäinsijoituksella välitulosvektori Y ja lopuksi takaisinsijoituksella tuntematon vektori X (kaavat (8) ja (9)). Vektoreille X ja Y sekä oikean puolen vakiovektorille C käytetään samaa työtilaa X(1:NE). Useampien oikeiden puolien ratkaiseminen yhtäaikaisesti ei tässä menetelmässä sanottavasti nopeuta ratkaisua.

MUISTITILA JA TUKIMUISTIN KÄYTTÖ

Liitteissä (1) ja (2) esitetyt Algol-kieliset ratkaisuproseduurit (laadittu tietokoneelle ICL 4-50) toimivat yksinomaan keskusmuistia käyttäen. Kumpikin vaatii muuttujia varten muistitilan ∿SIZ + 2NE + 2MAX (SIZ on kerroinmatriisin koko).

Tukimuisteja käytettäessä keskusmuistitilaa voidaan vielä huomattavasti säästää. Mikäli kolmiointi suoritetaan riveittäin, matriisin vaatima työtila on WSP ≤ (1 + MAX)·MAX/2 ja lisäksi tarvitaan tilaa taulukkojen RLI, RLJ, IND ja JX termeille. Välittömästi vaakarivin i kolmioinnin jälkeen pystyrivi i kirjoitetaan tukimuistiin ja

seuraava vaakarivi ottaa edellisen aseman, toisin sanoen ko. vaakarivin yläpuolinen kolmio liikkuu alaspäin.

Ratkaisun toisessa vaiheessa u-termit luetaan pystyriveittäin takaisin keskusmuistiin Y-vektorin muodostamiseksi. Vain i:s pystyrivi sekä sitä vastaavat edellä ratkaistut y-termit tarvitaan y_i:tä varten.

Laskettaessa X-vektoria U-matriisi on luettava takaperin keskusmuistiin. Jälleen vain i:s vaakarivi sekä edellä ratkaistut x-termit tarvitaan x_i:tä varten. Koska U on indeksoitu pystyriveittäin, on kuitenkin tarkoituksenmukaisinta pitää muistissa hetkittäin ko. vaakarivin alapuolinen kolmio.

Yleensä tehtävissä, joissa tukimuisteja joudutaan käyttämään, kerroinmatriisi on jo sinänsä pitkällisten laskutoimitusten tulos. Tällöin on tärkeää muodostaa kukin vaakarivi vuorollaan valmiiksi mahdollisimman aikaisessa vaiheessa kolmiointia varten ja näin säästää ylimääräisiä siirtoja tukimuistiin ja takaisin.

Kolmiointi voidaan suorittaa pienemmässäkin keskusmuistitilassa kuin edellä on esitetty, koska tietyn termin kolmioimiseksi tarvitaan vain osa kahden pystyrivin termeistä kerrallaan. Samoin tilaa voidaan säästää ratkaisuvaiheessa. Tällöin kuitenkin laskuaika kasvaa ratkaisevasti lisääntyvien siirto-operaatioiden vuoksi ja muodostuu helposti käyttöä rajoittavaksi tekijäksi.

CHOLESKI'N MENETELMÄ GAUSSIN MENETELMÄÄN VERRATTUNA

Gaussin eliminointimenetelmässä joudutaan kolmiointivaiheessa oikean puolen vakiovektorit redusoimaan kerroinmatriisin kolmioinnin yhteydessä. Menetelmä on ilmeisesti tehokkain tunnetuista, kun

yhtälöryhmä ratkaistaan vain yhdelle vakiovektorille tai vektorit voidaan kehittää yhtäaikaisesti.

Suurissa yhtälöryhmissä vakiovektoreiden tallettaminen asettaa muistitilapulmia, tai vakiovektoreita ei lainkaan voida muodostaa samanaikaisesti. Tällöin Gaussin menetelmässä yhtälöiden kerrointermit on joko talletettava kolmiointivaiheessa (muodostavat nauhamatriisin) tai kehitettävä uudelleen kolmioidusta matriisista.

Choleski'n menetelmässä tätä ei tarvita, koska kolmiointi ei vaikuta vakiovektoreihin. Voidaan siis välttää kerrointermien tallettaminen ja vakiovektoreiden redusointi. Sen sijaan joudutaan kehittämään välitulosvektori Y.

Menetelmien tehokkuutta on asiaa testaamatta vaikea verrata toisiinsa. Kuitenkin Choleski'n menetelmä tarjoaa selväpiirteisesti kahteen (oik. kolmeen) osaan jaettavan ratkaisun ja on näin ollen ohjelmointiteknisesti edullinen varsinkin tukimuistiversiona.

YHTEENVETO

Muuttuvaa nauhanleveyttä käytettäessä voidaan tietokoneen muistitilaa vielä huomattavasti säästää verraten vakionauhanleveyden käyttöön. Eräässä laskemassani esimerkissä kerroinmatriisin koko SIZ oli eri menetelmiä käyttäen:

	SIZ	Suhde edelliseen %
Täydellinen neliömatriisi	99856	-
Symmetria huomioon otettuna	50086	50.7
Nauhanleveys vakio	15800	31.6
Nauhanleveys muuttuva	11094	70.2

Tässä kerroinmatriisi oli koottu paikallista miniminauhanleveyttä silmälläpitäen, joten nauha oli suhteellisen tasalevyinen. Tehtävissä, joissa nauhanleveys vaihtelee paljon, säästö on huomattavasti suurempi.

Ratkaisuohjelma muodostuu hieman monimutkaisemmaksi ja näin ollen myös hitaammaksi kuin vakionauhanleveydellä toimiva. Koska laskutoimitusten lukumäärä tietyllä rivillä kasvaa verrannollisena termien lukumäärän neliöön, voidaan kokonaislaskuajassa kuitenkin saavuttaa merkittävä säästö.

Vakiovektoreiden riippumattomuus kolmioinnista on tärkeä etu probleemoissa, joissa kerroinmatriisi on pitkien laskutoimitusten tulos ja joissa vakiovektoreita ei voida muodostaa samanaikaisesti kerroinmatriisin kolmioimisen kanssa (esim. ratkaisun virhearviointi ja iteraationa tapahtuva tarkennus).

KIRJALLISUUTTA

[1] Jenkins W.M., Matrix & Digital Computer Methods in Structural Analysis. McGraw-Hill, Lontoo 1969, s. 170 ja 171, 174...176.

[2] Jennis A. & Tuff A.D. & Reid J.K., Large Sparse Sets of Linear Equations. Lontoo 1971, s. 97...104.

[3] Rubinstein M.F., Structural Systems: Statics, Dynamics and Stability, Englewood Cliffs. Prentice-Hall, s. 152...155.

Jorma Köliö, dipl.ins.

```
'BEGIN'
    'PROCEDURE' FACT(U,RLI,RLJ,MAX,NE);
    'VALUE' MAX,NE;
    'REAL''ARRAY' U;
    'INTEGER''ARRAY' RLI,RLJ;
    'INTEGER' MAX,NE;
    'BEGIN'
        'REAL' UH,C;
        'INTEGER' I,J,R,S,T,W,WU,A;
        'INTEGER''ARRAY' IND,JX(/1:MAX/);
        'FOR'I:=1'STEP'1'UNTIL'NE'DO'
        'BEGIN'
            'IF'I=1'THEN' W:=1 'ELSE' W:=W+RLJ(/I-1/);
            S:=1;J:=RLI(/I/);WU:=W;
            'FOR'T:=2'STEP'1'UNTIL'J'DO'
            'BEGIN' WU:=WU+RLJ(/I+T-2/);
                'IF' T-RLJ(/I+T-1/) <=0 'THEN'
                'BEGIN'
                    S:=S+1;
                    IND(/S/):=WU+T-1;
                    JX(/S/):=I+T-1;
                'END';
            'END' SUBSCRIPTS FOR ROW I ARE FORMED;
            UH:=0;J:=RLJ(/I/)-1;
            'FOR'T:=1'STEP'1'UNTIL'J'DO' UH:=UH+U(/W+T/)**2;
            C:=U(/W/):=SQRT(U(/W/)-UH);
            'FOR'T:=2'STEP'1'UNTIL'S'DO'
            'BEGIN' A:=IND(/T/);WU:=RLJ(/JX(/T/)/)-JX(/T/)+I-1;
                'IF' J<WU 'THEN' WU:=J;
                UH:=0;
                'FOR'R:=1'STEP'1'UNTIL'WU'DO'
                UH:=UH+U(/W+R/)*U(/A+R/);
                U(/A/):=(U(/A/)-UH)/C;
            'END' ROW I IS FORMED;
        'END' I LOOP;
    'END' PROCEDURE FACT;
```

U(1:SIZ) on kerroinmatriisi 1-dim. taulukkona
RLI(1:NE) on vaakarivien pituudet
RLJ(1:NE) on pystyrivien pituudet
SIZ on kerroinmatriisin koko
MAX on suurin vaakarivin pituus
NE on yhtälöiden lukumäärä

Huom: $SIZ = \sum_{m=1}^{NE} RLJ(m)$

ko. rivin läv. termin indeksi

ko. rivin muiden termien indeksit

lävistäjätermin kolmiointi

rivin muiden termien kolmiointi

Matriisin kolmiointiproseduuri

Liite 1.

```
'PROCEDURE' BACK(U,RLI,RLJ,MAX,NE,SIZ,X);
'VALUE' MAX,NE,SIZ;
'REAL''ARRAY' U,X;
'INTEGER''ARRAY' RLI,RLJ;
'INTEGER'MAX,NE,SIZ;
'BEGIN'
    'REAL' UH;
    'INTEGER''ARRAY' IND,JX(/1:MAX/);
    'INTEGER' I,S,T,P,W,WU;
    X(/1/):=X(/1/)/U(/1/);W:=1;
    'FOR'I:=2'STEP'1'UNTIL'NE'DO'
    'BEGIN'
        UH:=0;P:=RLJ(/I/);W:=W+P;
        'FOR'S:=1'STEP'1'UNTIL'P-1'DO'
        UH:=UH+U(/W-S+1/)*X(/I+S-P/);
        X(/I/):=(X(/I/)-UH)/U(/W-P+1/);
    'END' Y VECTOR IS FORMED;
    W:=SIZ-RLJ(/NE/)+1;
    X(/NE/):=X(/NE/)/U(/W/);
    'FOR'I:=NE-1'STEP'-1'UNTIL'1'DO'
    'BEGIN'
        WU:=W:=W-RLJ(/I/); UH:=0; P:=RLI(/I/); S:=1;
        'FOR'T:=2'STEP'1'UNTIL'P'DO'
        'BEGIN' WU:=WU+RLJ(/I+T-2/);
            'IF' T-RLJ(/I+T-1/)<=0'THEN'
            'BEGIN'
                S:=S+1;
                IND(/S/):=WU+T-1;
                JX(/S/):=I+T-1;
            'END';
        'END' SUBSCRIPTS FOR ROW I ARE FORMED;
        'FOR'T:=2'STEP'1'UNTIL'S'DO'
        UH:=UH+U(/IND(/T/)/)*X(/JX(/T/)/);
        X(/I/):=(X(/I/)-UH)/U(/W/);
    'END' X VECTOR IS FORMED;
'END' PROCEDURE BACK;
```

X(1:NE) on työtila vektoreille C, Y ja X
muut parametrit kuten edellä

y_1:n muodostaminen

y_i:n muodostaminen (i = 2...NE)

x_{NE}:n muodostaminen

ko. rivin läv. termin indeksi

rivin muiden termien indeksit

x_i:n muodostaminen (i = NE-1...1)

Ratkaisuproseduuri

Liite 2.