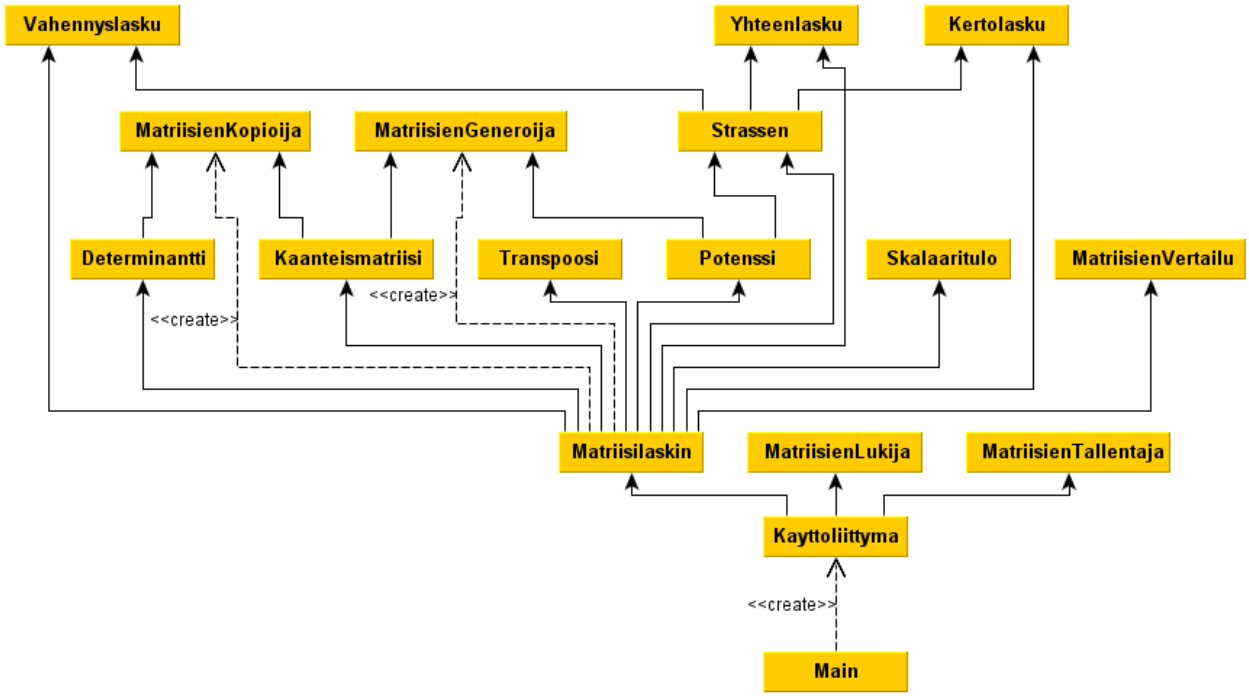
Matriisilaskimen yleisrakenne



Kuva 1. Matriisilaskimen luokkakaavio.

Kuvasta 1. nähdään matriisilaskin-ohjelman yleisrakenne ja luokkien väliset suhteet. Matriisilaskin-luokka kapseloi sisäänsä kaikki laskuoperaatiot, joita Kayttoliittyma-luokan kautta voidaan suorittaa. Kayttoliittyma-luokan kautta luetaan ja tallennetaan myös matriiseja.

Strassen

Strassenin algoritmi kertoo kaksi samankokoista neliömatriisia yhteen ja palauttaa niiden tulomatriisin. Jos matriisien rivit ja sarakkeet eivät valmiiksi ole kooltaan kahden potensseja, ne täytetään nollilla seuraavaan kahden potenssiin. Seuraavaksi ajetaan Strassenin rekursiivinen algoritmi, joka pilkkoo annetut neliömatriisit neljäksi yhtäsuureksi lohkomatriisiksi, muodostaa niiden avulla osituksen neliömatriisin vasemmalle ja oikealle ylä- ja alanurkalle ja laskee niillä rekursiivisesti 7 tulomatriisia, joiden avulla saadaan lopuksi koottua tulomatriisi, jonka jälkeen tarvittaessa poistetaan kootusta matriisista täytteenä olleet nollat, jolloin matriisin koko palautuu alkuperäiseksi

Optimoitu Strassenin algoritmi puolittaa aluksi neliömatriisin koon (ei neliömatriisia), jonka jälkeen tarkastaa mennäänkö 32:een tai sen alle, jolloin palauttaa tavallisella matriisikertolaskulla lasketun tulomatriisin. Tämä optimointi johtuu siitä, että pienillä matriiseilla Strassen paljon hitaampi kuin tavallinen matriisikertolasku. Tämän jälkeen alustaa neliömatriisien lohkomatriisit puolitetulla koolla ja sopivilla alkioilla, jotta muodostuu ositus neliömatriisin vasemmalle ja oikealle ylä- ja alanurkalle.

Determinantti

Determinantin algoritmi laskee parametrina annetun neliömatriisin determinantin. Osittaistuetun LU-hajotelman avulla muodostetaan LU-matriisiin sen ylä- sekä alakolmiomatriisi, missä päälävistäjän alkiot ovat yläkolmiomatriisin päälävistäjän alkiot. Jos LU-hajotelma epäonnistuu, olemme joutuneet tilanteeseen jossa jaetaan nollalla, joka on seuraus matriisin singulaarisuudesta eli epäsäännöllisyydestä. Tässä tilanteessa determinantin arvoksi palautetaan nolla. LU-hajotelman onnistuttua determinantti lasketaan LU-matriisin lävistäjäalkioiden tulona, jonka jälkeen se pyöristetään kahdeksan desimaalin tarkkuuteen. Lopuksi lisätään etumerkki rivinvaihtojen parillisuudesta riippuen, jonka jälkeen etumerkitty ja pyöristetty determinantti palautetaan.

Algoritmi käy järjestyksessä matriisin lävistäjäalkiot läpi etsien aluksi lävistäjäalkion sarakkeelle pivot-alkion. Pivot-alkio on lävistäjäalkiosta lähtien kyseisen sarakkeen itseisarvoltaan suurin alkio. Pivot-alkion etsintä estää nollalla jakamisen algoritmin myöhemmässä vaiheessa ja parantaa numeerista stabiilisuutta. Algoritmi palauttaakin falsen, jos pivot-alkion itseisarvoksi jää pienempi tai yhtäsuuri kuin 10^{-3}. Tällä optimoinnilla voidaan määrittää missä kulkee singulaarisuuden raja numeerisen tarkkuuden mielessä, joka tietysti riippuu siitä, minkälaisiin tapauksiin determinantin laskemista on määrä soveltaa. Kun löytyy itseisarvoltaan 10^{-3}:sta suurempi pivot-alkio, sen rivi ja lävistäjäalkion rivi vaihtavat paikkaa keskenään ja rivinvaihto-laskurin arvo kasvaa yhdellä, joka mahdollisesti vaikuttaa LU-matriisista laskettavan determinantin arvoon.

Viimeisessä vaiheessa hajotetaan matriisi kyseessä olevan lävistäjäalkion osalta LU-hajotelmalla. Tämä jakaa lävistäjäalkion sarakkeen alapuoliset alkiot lävistäjäalkiolla ja laskee Schurin komplementin arvot. Tähän mennessä päälävistäjän alapuolelle lävistäjäalkioon asti on löytynyt alakolmiomatriisin alkiot ja yläpuolelle päälävistäjän ja lävistäjäalkion rivin rajaamalle yläosalle yläkolmiomatriisin alkiot. Uusilla iteraatioilla algoritmi toimii edellisen iteraation lopussa lasketussa Schurin komplementissa eli lohkomatriisissa, joka on lävistäjäalkion sarakkeen ja rivin rajaama alue alaoikealla.

Käänteismatriisi

Käänteismatriisin algoritmi laskee parametrina annetun neliömatriisin käänteismatriisin. Annettu neliömatriisi A ja samankokoinen yksikkömatriisi I muodostavat laajennetun matriisin, johon sovelletaan skaalatulla osittaistuennalla toteutettua Gaussin eliminointi-menetelmää ja lopuksi takaisinsijoituksen avulla saadaan muodostettua haluttu käänteismatriisi, joka palautetaan.

Skaalatulla osittaistuennalla toteutetun LU-hajotelman avulla muodostetaan A-matriisiin sen ylä- sekä alakolmiomatriisi, missä päälävistäjän alkiot ovat yläkolmiomatriisin päälävistäjän alkiot. Rivinvaihtojen osalta myös I-matriisin rivit vaihdetaan keskenään. Seuraavaksi (mahdollisesti) rivimuunneltuun yksikkömatriisiin I tehdään suhteessa samat muutokset kuin mitä matriisiin A tehtiin LU-hajotelman osalta käyttäen apuna A:n alakolmiomatriisin arvoja ja eteenpäin sijoitusta. Tämän jälkeen Gaussin eliminointi on saatu päätökseen ja voidaan takaisinsijoittamalla laskea palautettava käänteismatriisi A:n yläkolmiomatriisin ja I:n sarakkeiden avulla.

Skaalatussa osittaistuennassa aluksi etsitään jokaisen rivin skaalaustekijä, joka on matriisin tutkittavana olevan rivin itseisarvoltaan suurin alkio. Tämän jälkeen etsitään parametrina annetun

matriisin A sarakkeelle pivot-rivi eli indeksi miltä riviltä löytyy sarakkeen pivot-alkio. Pivot-alkio on sarakeindeksin ja päälävistäjän leikkauksesta lähtien sarakkeen itseisarvoltaan suurin alkio suhteutettuna rivin skaalaustekijään.

Potenssiin korottaminen neliöimällä

Potenssiin korottaminen neliöimällä laskee rekursiivisesti parametrina annetun neliömatriisin potensseja haluttuun potenssiin asti. Koska matriisien potenssiin korottaminen on määritelty vain positiivisilla kokonaisluvuilla, ei negatiivisia potensseja hyväksytä. Nolla-potenssi palauttaa neliömatriisin kokoisen yksikkömatriisin. Parillisuudesta riippuen korottaa rekursiivisesti potenssiin kaksi käyttämällä Strassenin kertolaskua, parittomana palauttaa matriisilla kerrotun rekursioAlgon tuloksen. Lopuksi palauttaa neliömatriisin korotettuna haluttuun potenssiin.

Saavutetut aika- ja tilavaativuudet

Täysin rekursiivisen Strassenin algoritmin aikavaativuudeksi n x n matriiseille tavoiteltiin $O(n^{lb7}) = O(n^{2.8074})$ ja tilavaativuudeksi $O(n^2)$. Tähän aika- ja tilavaativuuteen päästiin, koska rekursioAlgo:ssa rekursiivisesti muodostetaan 7 tulomatriisia n/2 x n/2 kokoisista matriiseista ja niissä käytetään vakiomäärä $O(n^2)$ yhteen- ja vähennyslaskuja, jolloin rekursioyhtälön perusteella aikavaativuudeksi saadaan $T(n) = 7\ T(n/2) + O(n^2) = O(n^{lb7}) = O(n^{2.8074})$, missä $T(n) = O(1)$, kun $n = 1$. Tilaa per matriisi tarvitaan rekursiotasoittain $n^2/(4^{k-1})$, missä n on matriisin koko ja k on rekursiotason korkeus, joten tilavaativuudeksi per matriisi saadaan alla olevan summakaavan perusteella $O(n^2)$, joka on myös algoritmin tilavaativuus, koska matriiseja on vakiomäärä:

$$\sum_{k=1}^{\infty} \frac{n^2}{4^{k-1}} = \frac{4\,n^2}{3}$$

Lopulliseen toteutukseen jäi optimoitu Strassenin algoritmi, jonka lohkomatriiseiksi pilkkominen päättyy 64x64 tai pienempään matriisikokoon, jonka jälkeen kertolaskuvastuun ottaa tavallinen matriisien kertolasku. Tästä johtuen aikavaativuus 64x64 matriiseilla on tavallisen kertolaskun mukaisesti $O(n^3)$ ja tilavaativuus $O(n^2)$, ja ne ovat myös optimoidun Strassenin aika- ja tilavaativuudet. Tämä kuitenkin on hämäävää, koska käytännössä kuutiollinen aikavaativuus noin pienillä matriisikooilla ei ole mitään, joten keskimäärin isoimmilla matriiseilla päästään $O(n^{2.8074})$:n aikavaativuuteen.

Osittaistuetulla LU-hajotelmalla toteutetulle n x n matriisin determinantille tavoiteltu aikavaativuus oli $O(n^3)$ ja tilavaativuus $O(n^2)$. Tämä aika- ja tilavaativuus saavutettiin, koska muodostaLU:ssa oleva kolmen sisäkkäisen for-loopin pätkä on algoritmin aikavaativuudeltaan dominoivin osa ja matriisin kopiointi taas dominoi tilavaativuutta muiden muuttujien ollessa vakioisia.

Käänteismatriisin approksimointi n x n matriiseille toteutettiin skaalatulla osittaistuennalla varustetulla Gaussin eliminointi –menetelmällä, jonka tavoiteltu aikavaativuus oli $O(n^3)$ ja tilavaativuus $O(n^2)$. Tällä kertaa muodostaLU, suhteuta ja takaisinsijoita -metodeissa olevat kolmen sisäkkäisen for-loopin pätkät ovat algoritmin aikavaativuudeltaan dominoivimmat osat eli $O(n^3)$ aikavaativuus saavutettiin. Matriisien kopiointi ja yksikkömatriisin luominen dominoivat tilavaativuuden puolelta, joten myös tilavaativuus $O(n^2)$ saavutettiin.

Potenssiin korottamisen aika- ja tilavaativuustavoitteet neliöimällä n x n matriiseja k:nennelle potenssille olivat $O(n^3 lb(k))$ ja $O(n^2)$. Nämä aika- ja tilavaativuudet saavutettiin, koska kertolasku optimoidulla Strassenilla on $O(n^3)$ (vaikkakin isoimmilla matriiseilla keskimäärin $O(n^{2.8074})$) ja rekursiosyvyys on logaritminen potenssin suhteen, joten aikavaativuudeksi saadaan haluttu $O(n^3 lb(k))$ ja $O(n^2)$ tilavaativuus saavutetaan myös matriisin kertolaskun kopioinnin ja yksikkömatriisin luomisen dominoidessa tilavaativuutta.

Testausdokumentista voi lukea suorituskykyanalyysia verrattuna saavutettuihin aikavaativuuksiin.

Työn puutteet

Singulaaristen matriisien potenssiin korottaminen saattaa johtaa suurilla alkioilla numeeriseen epästabiilisuuteen determinanttia laskettaessa, joka johtaa matriisin determinantin arvon vääristymiseen ja ketjureaktiona mahdollistaa siis käänteismatriisin laskemisen. Kehitysehdotuksena tähän on soveltaa jotain osittaistuentaa kehittyneempää tuentamenetelmää.

Parannusehdotukset

Käänteismatriisi Strassenin algoritmilla

Choleskyn hajotelman avulla toteutettu tarkistus, onko matriisi positiivisesti definiitti

Householderin muunnoksella/peilauksella tai Givensin rotaatiolla toteutettu QR-hajotelma matriisien ominaisarvojen ja –vektorien laskemiseen

Mahdollisuus laskea myös kompleksiarvoisia matriiseja

Jotain älykkäämpää liukulukupyöristelyä alkioille ja determinantille

Parempi käyttöliittymä

Lähteet:

http://en.wikipedia.org/wiki/Strassen_algorithm

http://en.wikipedia.org/wiki/LU_decomposition

http://en.wikipedia.org/wiki/QR_decomposition

http://en.wikipedia.org/wiki/Cholesky_decomposition

http://en.wikipedia.org/wiki/Determinant#Calculation

http://en.wikipedia.org/wiki/Pivot_element

http://en.wikipedia.org/wiki/Gaussian_elimination

http://en.wikipedia.org/wiki/Invertible_matrix

http://en.wikipedia.org/wiki/Exponentiation_by_squaring

http://www.astro.utu.fi/edu/kurssit/f90/f90/finnish/pdf/numeigen.pdf

Cormen, Leiserson, Rivest, Stein: Introduction to Algorithms, 2nd edition

Cambridge University Press: Numerical Recipes in C: The Art of Scientific Computing, 2nd edition