02.09.12

# TIETORAKENTEIDEN HARJOITUSTYÖ Elokuu 2012

## Toteutusdokumentti

**Tekijä:** Juho Eskelinen (013466770)
Helsingin yliopisto - Tietojenkäsittelytieteen laitos
**Ohjaaja:** Kristiina Paloheimo

**Ohjelman rakenne:**

Luokassa Matrices on matriisien luontiin ja tulostukseen liittyvät metodit. Matriiseille suoritettavat operaatiot ovat luokassa MatrixOperations ja LU-hajotelman luonti sovelluksineen tapahtuu luokan LU avulla. Metodit käyttävät virhetilanteiden kuvaamiseen muutamia Exception-luokan alaluokkia, jotka on koottu pakettiin MatriisiKirjasto.virheet.

**Aika – ja tilavaativuudet:**

Mikäli muuta ei mainita, n on neliömatriisin sivun pituus, n ja m ovat yleisen matriisin sivujen pituudet ja m ja p ovat mahdollisen toisen matriisin sivujen pituudet.

Transpoosi:

- Aika: O(n*m). Kaikki alkiot käsitellään kerran.
- Tila: O(n*m). Muodostettavan matriisin vaatima tila.

Matriisien yhteenlasku:

- Aika: O(n*m). Molempien matriisien alkiot käsitellään kertaalleen.
- Tila: O(n*m). Tulosmatriisin takia.

Vakiolla kertominen:

- Aika: O(n*m). Kopion luominen ja operaation suorittaminen vievät molemmat tämän ajan.
- Tila: O(n*m). Tulosmatriisin takia.

Matriisien kertolasku:

- Aika: O(n*m*p). Transpoosin luonti tapahtuu ajassa O(m*p), joten se ei muuta aikavaativuutta. Uloin silmukka on O(n), keskimmäinen O(p) ja sisin O(m), jolloin kokonaisaikavaativuus on näiden tulo.
- Tila: O(n*p + m*p). Tulosmatriisin vie n*p ja transpoosi (m*p)

LU-hajotelman luominen (LU:n konstruktori):

- Aika: $O(n^3)$. Permutaatiomatriisin luominen on O(n) ja syötematriisin

kopioiminen on $O(n^2)$. Varsinainen hajotelman muodostaminen sisältää kolme sisäkkäistä silmukkaa, joista kukin on $O(n)$ eli kaiken kaikkiaan $O(n^3)$ ja dominoi siis aikavaativuutta.

- Tila: $O(n^2)$. Hajotelman sijoittamiseen ja säilyttämiseen. Lisäksi pienempänä myös permutaatioiden säilyttämiseen $O(n)$.

Determinantti:

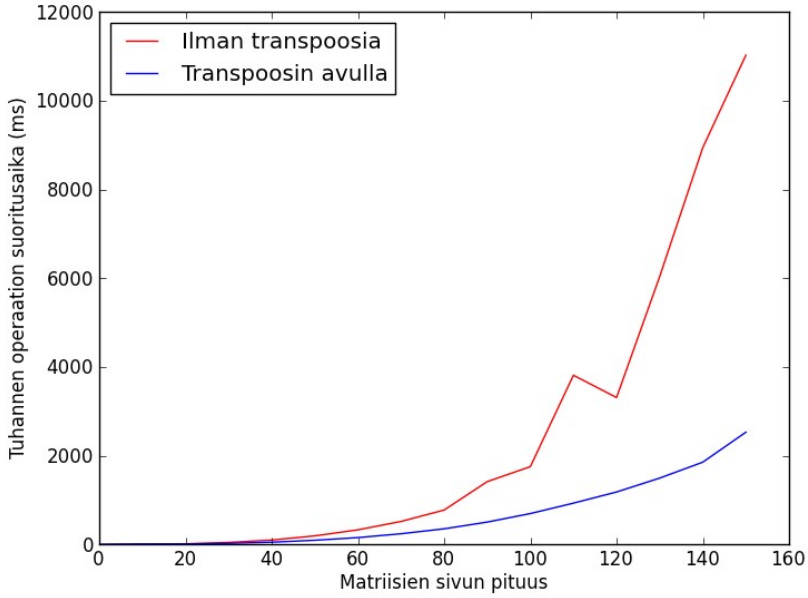
- Aika: $O(n)$, lisäksi hajotelman luominen $O(n^3)$. Sisältää itse vain $O(n)$-silmukan.
- Tila: $O(1)$.

Yhtälöryhmien ratkaiseminen:

- Aika: $O(n^2)$, lisäksi hajotelman luominen $O(n^3)$. Metodi sisältää kaksi $O(n^2)$-silmukkaa.
- Tila: $O(n)$. Tulosvektoria varten.

Monien metodien tilavaativuus voitaisiin muuttaa $O(1)$:ksi, mikäli tulokset sijoitettaisiin syötematriisiin. Halusin kuitenkin säilyttää alkuperäiset matriisit sellaisenaan, jolloin aputilaa tarvitaan. Tilantarve on kuitenkin aina samaa suuruusluokkaa syötteen kanssa.

Suoritin O-luokan sisällä tapahtuvaa optimointia matriisikertolaskun kohdalla. Otan jälkimmäisestä matriisista ensin transpoosin, jolloin siitä voidaan lukea sarakkeiden sijaan rivejä. Tämä nopeuttaa operaatiota huomattavasti, kuten kuvasta 1 nähdään. Alle 10x10-matriisien kohdalla transpoosin laskeminen voi tosin hidastaa laskutoimitusta hieman, mutta ei ratkaisevasti.




*Kuva 1. Matriisikertolaskun nopeus valmiiksi lasketun transpoosin avulla ja ilman sitä.*

**Parannettavaa tai lisättävää:**

LU-hajotelman avulla saisi tehtyä myös käänteismatriisin muodostamisen. Muilla hajotelmilla on tietyissä tilanteissa parempia ominaisuuksia, mutta LU on helpompi toteuttaa ja varsin numerisesti stabiili.

Monet metodit soveltuisivat sisäkkäisten silmukoidensa takia rinnakkaistettaviksi. Tämän kurssin puitteissa en kuitenkaan lähtenyt tutustumaan näin monimutkaiseen täysin tuntemattomaan Javan osaan. Lisäksi Java ei ole välttämättä mielekkäin kieli hyvin raskaaseen laskentaa, oli rinnakkaistamista tai ei.

Kauhean monimutkaiseen optimointiin en myöskään lähtenyt ja Javan liikuteltavan luonteen takia niiden toimivuudesta kaikilla alustoilla ei ole takeita.

**Lähteet:**

LU-hajotelmia lukuunottamatta, algoritmit perustuvat suoraan matemaattisiin esityksiin operaatioille ja niiden toteuttamiseen omien tietojenkäsittelytietojeni pohjalta. LU-hajotelmien muodostamiseen sovelluksineen sain apua seuraavista kirjoista: Matemaattiseen taustaa ja yleisen tason toteukseen Cormen et al. Introduction to Algorithms 3rd edition, sekä käytännön toteutukseen Press et al. Numerical Recipes 3rd edition. En kuitenkaan toteuttanut sokeasti Numerical Recipesin pohjalta, sillä sen optimoinnit eivät välttämättä toimi eri kielellä ja vaikeuttavat lähdekoodin luettavuutta.