TOTEUTUS

Ohjelman rakenne:

Ohjelma koostuu viidestä luokasta: App (main), Matrix, MatrixCalculator, LUDecomposition, sekä UI. App on main luokka, joka ei sisällä mitään mielenkiintoista. Matrix luokka on matriisin tietorakenne, jossa matriisi on esitetty kaksiulotteisena taulukkona. Taulukon lisäksi Matrix luokka sisältää tiedon rivien ja sarakkeiden määrästä, matriisien yksinkertaisimmat laskutoimitukset (yhteen- ja vähennyslasku, skalaarikertolaski ja transpoosi), sekä matriisin alkeisrivitoimitukset. Luokka MatrixCalculator sisältää monimutkaisemmat algoritmit, kuten Strassen algoritmilla toteutetun matriisien kertolaskun ja determinantin märityksen, sekä Gaussin-Jordanin eliminointimenetelmän. LUDecomposition luokkaan on toteutettu algoritmi LU decompositionin luomiseen, sekä algoritmi joka määrittää käänteismatriisin LU Decompositionin avulla. UI sisältää tekstipohjaisen käyttöliittymän. Käyttöliittymässä matriisien luominen on toteutettu niin, että kunkin operaation kohdalla tarkistetaan, että matriisit ovat oikeankokoisia operaatiota varten.

Aika- ja tilavaativuudet:

Matriisien yhteenlasku:

```
for i=1...m
   for j=1...n
      C[i][j] = A[i][j] + B[i][j]
```

Yhteenlaskun aikavaativuus on O(n), ja yhteenlaskujen määrä riippuu m:stä ja n:stä, joten aikavaativuudeksi tulee O(m*n). Tilavaativuus on myös O(m*n).

```
for i=1...m
   for j=1...n
      A[i][j] = x*A[i][j]
```

Skalaarilla kertomisessa on sama idea kuin yhteenlaskussa, eli aikavaativuus riippuu matriisin koosta. O(m*n).

Matriisien kertolasku on toteutettu Strassen algoritmilla, jonka asymptoottinen aikavaativuus $O(n^{2.807})$ ja tilavaativuus $O(n^2)$.

Käänteismatriisin sekä determinantin määrittämiseen käytetään LU decompositionia, jonka aikavaativuus on $O(n^3)$.

Gauss-Jordanin menetelmän aikavaativuus on $O(n^3)$.

Transpoosi:

```
for i=1...m
   for j=1...n
      A[i][j] = A[j][i]
```

Operaation A[i][j] = A[j][i] aikavaativuus on O(1), joten transpoosin aikavaativuus on tulee taas rivien ja sarakkeiden määrästä, eli O(m*n). Tilavaativuutena sama, koska tilaa vie vain matriisi.

Puutteet:

Jos laskinta lähtisi parantamaan, olisi yhtälöryhmien ratkaisu matriisien avulla helpohko toteuttaa jo toteutettujen ominaisuuksien avulla. Myös ominaisarvojen ja ominaisvektoreiden määritykset puuttuvat laskimesta. Koodin laatuun jäi myös parantamisen varaa.

Lähteet:
Introduction to Algorithms, 3rd edition
http://en.wikipedia.org/wiki/Gaussian_elimination
http://en.wikipedia.org/wiki/Strassen_algorithm
http://en.wikipedia.org/wiki/LU_decomposition
http://www.java2s.com/Code/Java/Collections-Data-Structure/LUDecomposition.htm