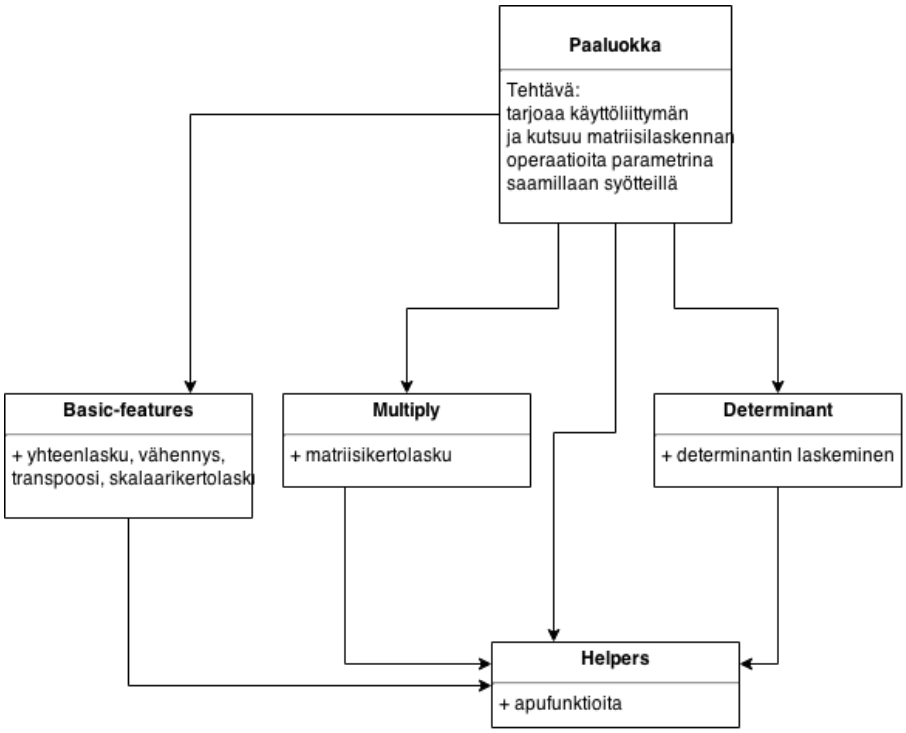
Samu Tamminen 013976176

Toteutusdokumentti

Ohjelman rakenne

Ohjelma on toteutettu funktionaalista ohjelmointiparadigmaa hyödyntäen. Funktioilla ja kokoelmilla ei ole tilaa olioiden ja luokkien tavoin, vaan funktiot saavat syötteenä parametreja ja palauttavat aina paluuarvon. Ohjelma koostuu viidestä kokoelmasta funktiota. Pyrin jakamaan funktiot sopiviin kokonaisuuksiin olio-ohjelmoinnista tutulla tavalla tarkastelemalla luokkien vastuuta. Samassa kokoelmassa olevat funktiot tulisi liittyä jotenkin toisiinsa.

Paaluokka on ohjelman ainoa luokka, joka luodaan JVM-kääntämistä varten. Paaluokan funktiot tulostavat tulosteita, eli niillä on “sivuvaikutuksia”. Pääluokka tarjoaa komentoriville tekstikäyttöliittymän ja se kutsuu tarvittavia funktioita eri funktio-kokoelmista. Helpers-kokoelma tarjoaa sellaisia apufunktioita matriisien käsittelyyn, jotka itsessään eivät ole matriisilaskennan operaatioita ja joita monet funktiot hyödyntävät. Kokoelmassa basic-features on yksinkertaisimmat matriisilaskennan operaatiot(lisäys, vähennys, transpoosi, skalaarikerroin).



Kokoelma multiply koostuu kahdesta toiminnallisuudesta; raa’an voiman matriisikertolaskusta ja hieman tehokkaammasta strassenin algoritmistä sekä näiden apufunktioista.
Determinant-kokoelmassa tarjotaan determinantin laskemiseen hitaampi ja nopeampi toteutus. Nopeampi toteutus ei toimi oletetusti, vaan joillain syötteillä tehdään jakolasku luvulla 0, mistä seuraa JVM-poikkeus.

Puutteet ja parannusehdotukset

LU-determinant funktio muuntaa ensin LU-decompose funktiolla matriisin hajotelmaksi doolittle-algoritmilla. Toteutukseni algoritmille ei toimi kuten pitäisi, kuten yllä kuvailin. Myönnän, etten hallitse työkaluna käyttämääni ohjelmointikieltä Clojurea (vielä) yhtä hyvin kuin Javaa, enkä ajan loppuessa ehtinyt debugata tai uudelleenkirjoittaa algoritmia Clojurella. Tutkiessani tapausta kokeilin toista vastaavaa algoritmia Java:lla ja se toimi myös ongelmaksi muodostuneilla syötteillä. Demossa Kristiina kehotti liittämään Java-toteutuksen liitteeksi, joten liitin tämän tiedoston jatkoksi.

Testausta varten olisin voinut aluksi kerätä ison varaston testattavaa dataa, jonka avulla olisin voinut rakentaa ohjelmaa TDD-menetelmin. Nyt aikaa kului paljon yksikkötestien kirjoittamiseen itse toiminnallisuuden ollessa jo valmis. Joskus huomasin erheellisen toiminnan vasta testejä kirjoittaessa. Myös matriiseja generoiva apuohjelma olisi ollut alusta alkaen avuksi.

Kurssin loppua kohden minulla tuli kiire. Toteutin toiminnallisuuksia ja kirjoitin koodia alusta asti, mutta hieman verkkaisesti. Olisi pitänyt alussa nakuttaa algoritmit ja testit valmiiksi ja keskittyä sitten paranteluihin.

```
public static double[][] LUdecomposition(double[][] A) {
        double[][] LU = A.clone();
        int m = A.length;
        int n = m;
        int[] piv = new int[m];
        for (int i = 0; i < m; i++) {
            piv[i] = i;
        }
        double[] LUrivit;
        double[] LUsarakkeet = new double[m];

        for (int j = 0; j < n; j++) {
            for (int i = 0; i < m; i++) {
                LUsarakkeet[i] = LU[i][j];
            }
            for (int i = 0; i < m; i++) {
                LUrivit = LU[i];
                int kmax = Math.min(i, j);
                double s = 0.0;
                for (int k = 0; k < kmax; k++) {
                    s += LUrivit[k] * LUsarakkeet[k];
```

```
            }
            LUrivit[j] = LUsarakkeet[i] -= s;
        }
        int p = j;
        for (int i = j + 1; i < m; i++) {
            if (Math.abs(LUsarakkeet[i]) > Math.abs(LUsarakkeet[p])) {
                p = i;
            }
        }
        if (p != j) {
            for (int k = 0; k < n; k++) {
                double t = LU[p][k];
                LU[p][k] = LU[j][k];
                LU[j][k] = t;
            }
            int k = piv[p];
            piv[p] = piv[j];
            piv[j] = k;
        }
        if (j < m & LU[j][j] != 0.0) {
            for (int i = j + 1; i < m; i++) {
                LU[i][j] /= LU[j][j];
            }
        }
    }
    return LU;
}
```