Linear Mathematics Library

Table of Contents

by Jacek Langer

Hintergrund

Bei LGS handelt es sich um ein Schulprojekt, zum Lösen von Matrizen. Dabei findet der Gauß-Algorithmus Verwendung. Bei der Implementierung handelt es sich um eine Klassenbibliothek die eine Statische Methode bereitstellt. Es wird ein Argument erwartet und ein weiteres ist Optional zu Übergeben. Wird nur ein Argument übergeben wird aus der Übergebenen Matrix die Inverse Matrix gebildet. Wird ein Vector als zweites Argument übergeben wird die Angegebene Matrix anhand des Lösungsvektors gelöst.

Gauß Algorithmus zum Lösen von Matrizen

Beim Gaußschen Algorithmus zum lösen von Matrizen handelt es sich um ein Iteratives Verfahren zum Lösen von Matrizen anhand eines Lösungsvektors. Dabei wird eine Matrix mit einem Ausgangsvektor gleichgestellt und die einzelnen Zeilen der Matrix mit einander addiert bis die Matrix eine Einheitsmatrix darstellt. Dabei wird auf den Vektor dieselbe Operation angewendet wie auf die Matrix. Bei einer gelösten Matrix können die Werte für die einzelnen Unbekannten aus dem Lösungsvektor abgelesen werden.

Erste Iteration:

$$\begin{matrix} I \\ II \\ III \end{matrix}\left(\begin{array}{ccc|c} 5 & 3 & 8 & 7 \\ 15 & 8 & 18 & 5 \\ 10 & 10 & 20 & 9 \end{array}\right)\begin{matrix} \\ II - 3I \\ III - 2I \end{matrix}$$

$$\begin{matrix} I \\ II \\ III \end{matrix}\left(\begin{array}{ccc|c} 5 & 3 & 8 & 7 \\ 0 & -1 & -6 & -16 \\ 0 & 4 & 4 & -5 \end{array}\right)\begin{matrix} \\ \\ III + 4II \end{matrix}$$

Rückwärtige Iteration:

$$\begin{matrix} I \\ II \\ III \end{matrix}\left(\begin{array}{ccc|c} 5 & 3 & 8 & 7 \\ 0 & -1 & -6 & -16 \\ 0 & 0 & -2 & -21 \end{array}\right)\begin{matrix} I + 4III \\ II - 3III \\ \\ \end{matrix}$$

$$\begin{matrix} I \\ II \\ III \end{matrix}\left(\begin{array}{ccc|c} 5 & 3 & 0 & -77 \\ 0 & -1 & 0 & 47 \\ 0 & 0 & -2 & -21 \end{array}\right)\begin{matrix} I + 3II \\ \\ \\ \end{matrix}$$

Normalisieren der Einheitsmatrix

$$\begin{matrix} I \\ II \\ III \end{matrix}\left(\begin{array}{ccc|c} 5 & 0 & 0 & 64 \\ 0 & -1 & 0 & 47 \\ 0 & 0 & -2 & -21 \end{array}\right)\begin{matrix} I * 1/5 \\ II * -1 \\ III * -1/2 \end{matrix}$$

Lösungsvektor:

$$\begin{matrix} I \\ II \\ III \end{matrix}\left(\begin{array}{ccc|c} 1 & 0 & 0 & 64/5 \\ 0 & 1 & 0 & -47 \\ 0 & 0 & 1 & 21/2 \end{array}\right)\begin{matrix} x = \frac{64}{5} \\ y = -47 \\ z = \frac{21}{2} \end{matrix}$$

Anwendung

Diese Klassenbibliothek stellt eine Klasse bereit. Bei der initialisierung der Klasse wird eine Mehrdimensionales Array von Double Werten erwartet. Die angabe eines Lösungsvektors ist optional.

Wird kein Lösungsvektor angegeben kann die Matrix nicht gelöst werden, es steht lediglich die Methode bereit die Inverse der Matrix zu bilden. Die Matrix und der Lösungsvektor können nachträglich geändert werden.

Methoden:

- solve
- Invert

solve

Löst die Matrix mit Hilfe des angegebenen Vektors. Das Ergebnis dieser Operation ist ein Vektor.

Example 1. Matrix Lösen, vollständige Instanzierung

```
var matrix = arrayOf(doubleArrayOf(2.0, 4.0), doubleArrayOf(2.0, 3.0))

var vector = doubleArrayOf(5.0, 6.0)

var result = Gauss(matrix,vector).solve()
```

solve mit Parametern

Diese Funktion erwartet zwei Parameter. Eine Matrix und einen Vektor. Die Matrix und der Lösungsvektor werden neu gesetzt. Anschließend die Matrix gelöst.

Example 2. Matrix Lösen

```
var matrix = arrayOf(doubleArrayOf(2.0, 4.0), doubleArrayOf(2.0, 3.0))

var gauss = Gauss(matrix) var vector = doubleArrayOf(5.0, 6.0)

var result = gauss().solve(matrix,vector)
```

Invert

Generiert die Inverse der Matrix.

Example 3. Inverse einer Matrix generieren

```
var matrix = arrayOf(doubleArrayOf(2.0, 4.0), doubleArrayOf(2.0, 3.0))

var vector = doubleArrayOf(5.0, 6.0)

var inverse = Gauss(matrix,vector).invert()
```

Der ganaue Ablauf lässt sich hier finden.

Ausnahmen

Wird eine Matrix ohne Angabe eines Lösungsvektors gelöst wird eine `InvalidOperationException` geworfen. Stellt sich beim Lösen einer Matrix fest das es sich um eine unlösbare Matrix handelt wird eine `UnsolvableMatrixException` geworfen. Wird eine Matrix angegeben dessen Rang ungleich dem Rang des Lösungsvektors ist wird ebenfalls eine `UnsolvableMatrixException` geworfen.

Weitere Informationen finden sich hier

UML-Diagramme

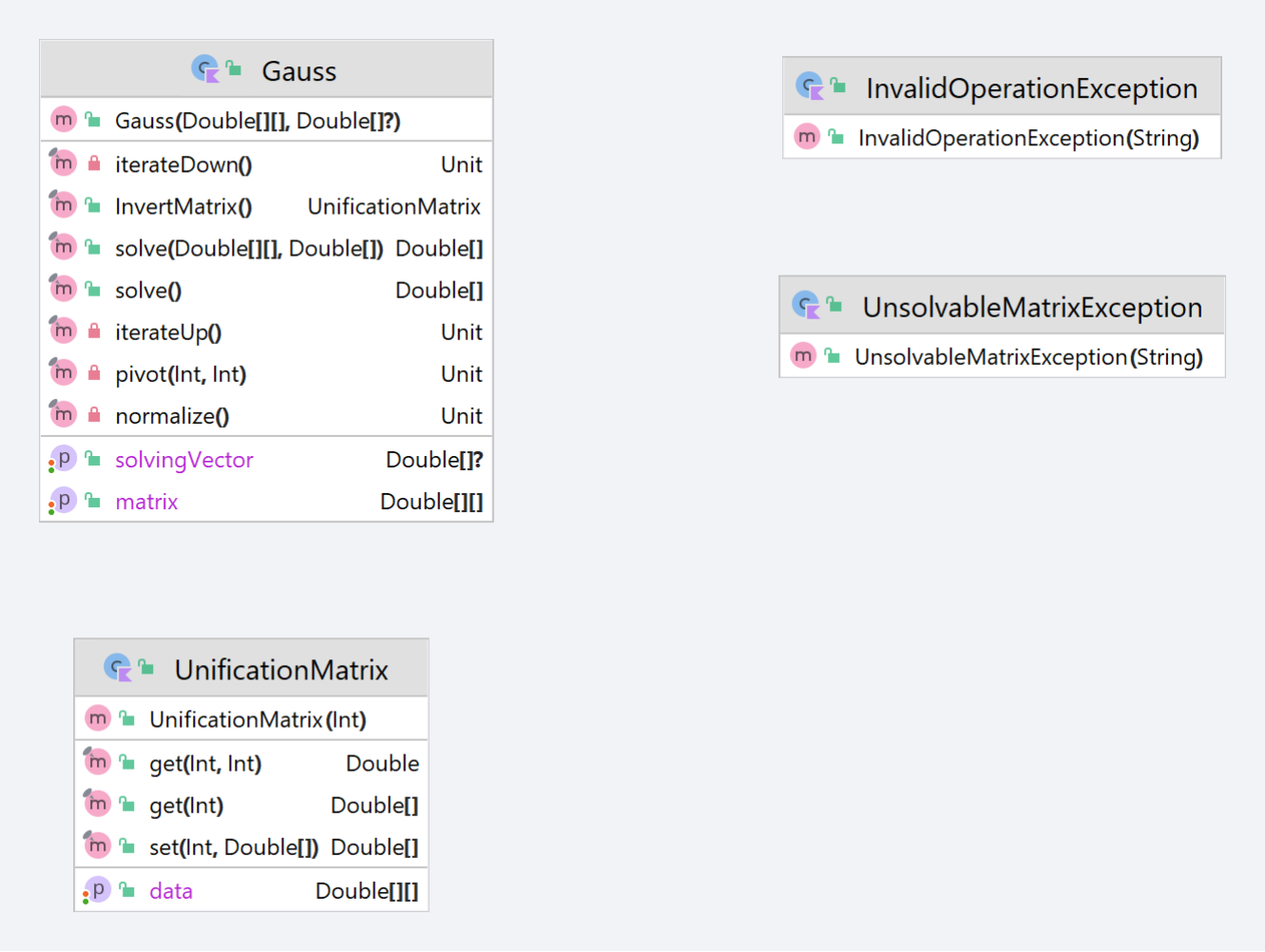
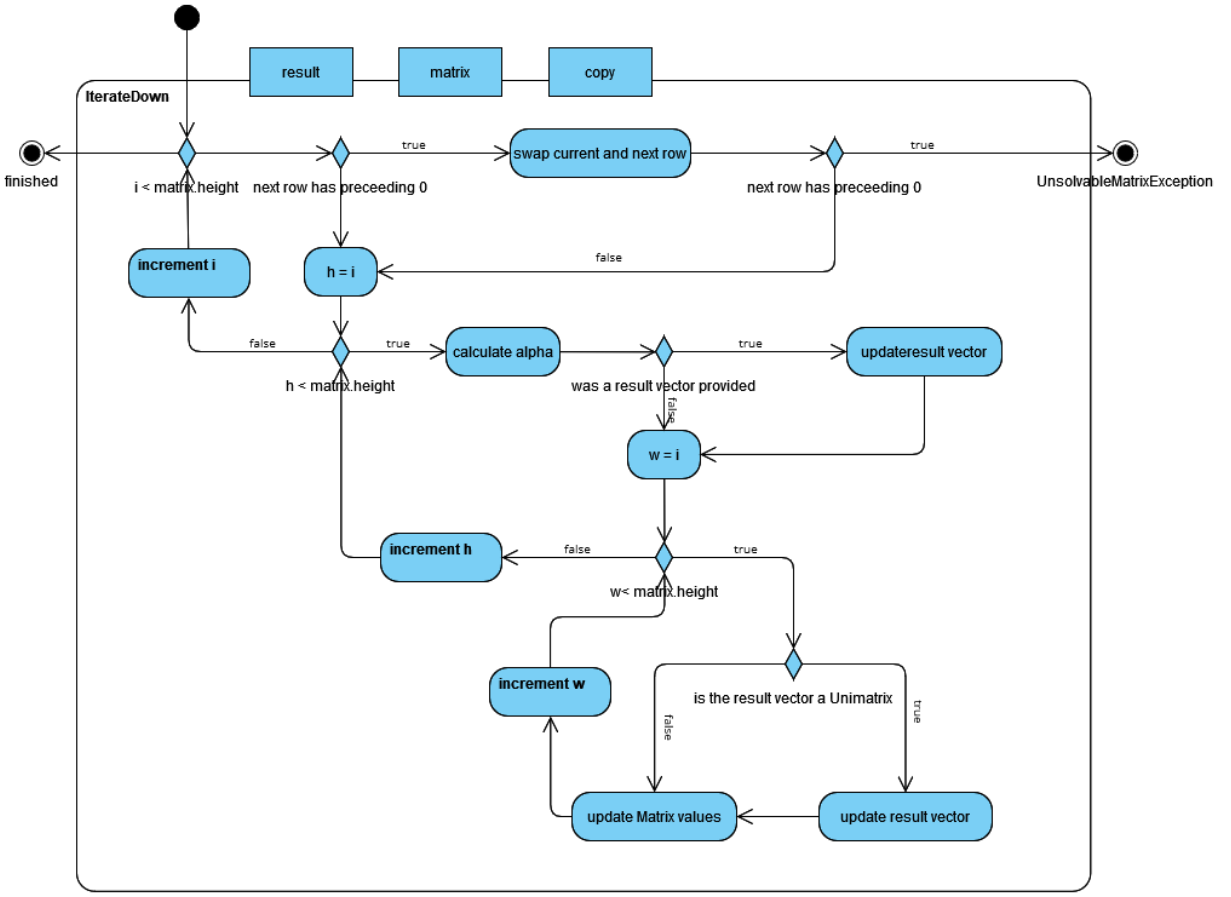


Figure 1. Klassendiagramm

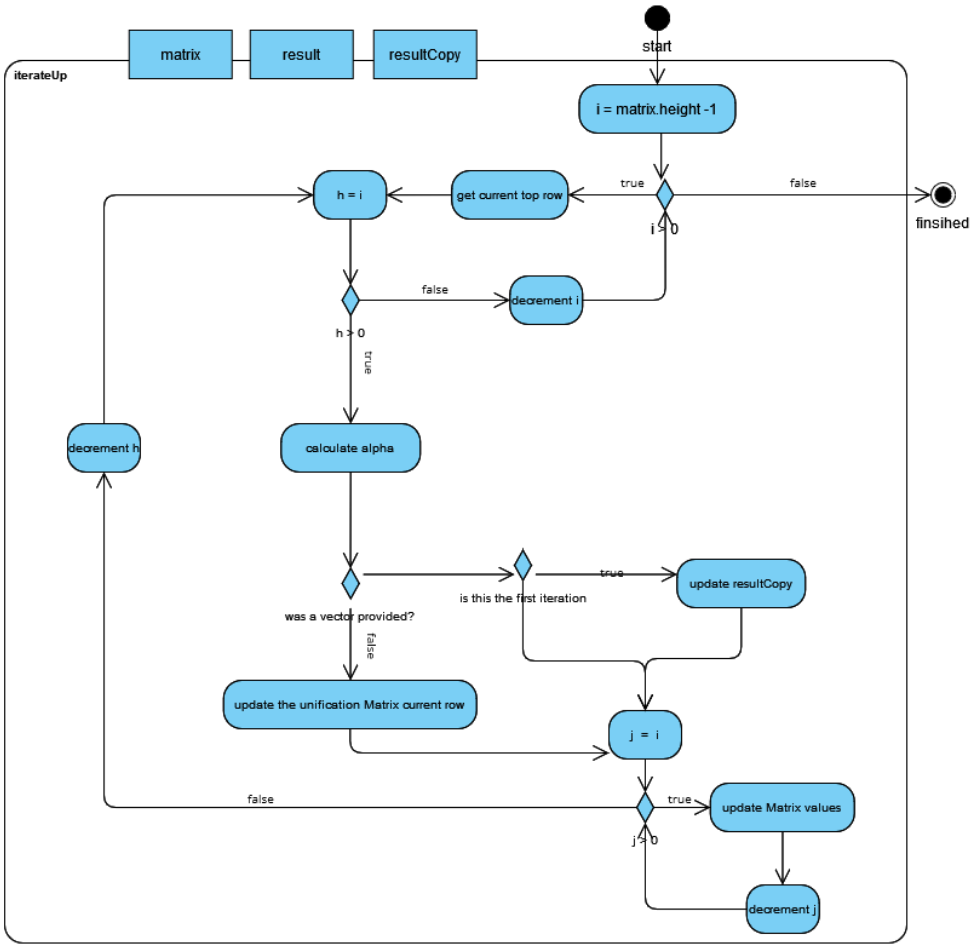
Visual Paradigm Online Free Edition



Visual Paradigm Online Free Edition

Figure 2. IterateDown-Methode

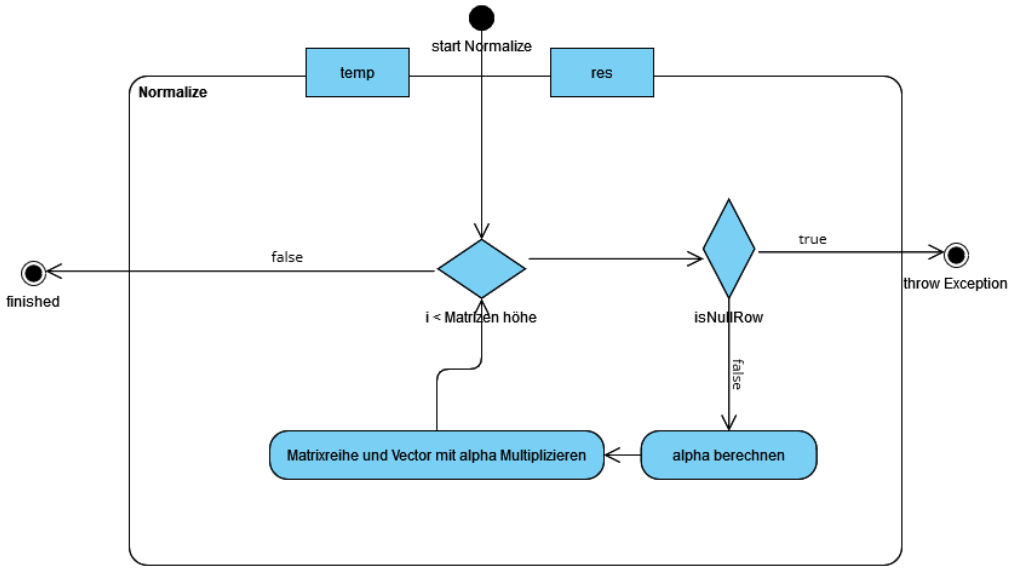
Visual Paradigm Online Free Edition



Visual Paradigm Online Free Edition

Figure 3. IterateUp-Methode

Visual Paradigm Online Free Edition



Visual Paradigm Online Free Edition

Figure 4. Normalize-Methode

Abhängigkeiten und Systemanforderungen

Dependency	Version
JVM	16
Kotlin	1.6.2
Java	17
Gradle	7.4.2
JUnit	5.6.0
GSON.[1]	2.9.0

[1] Testing only