David Wladimir Zakrevskyy/ 70351576

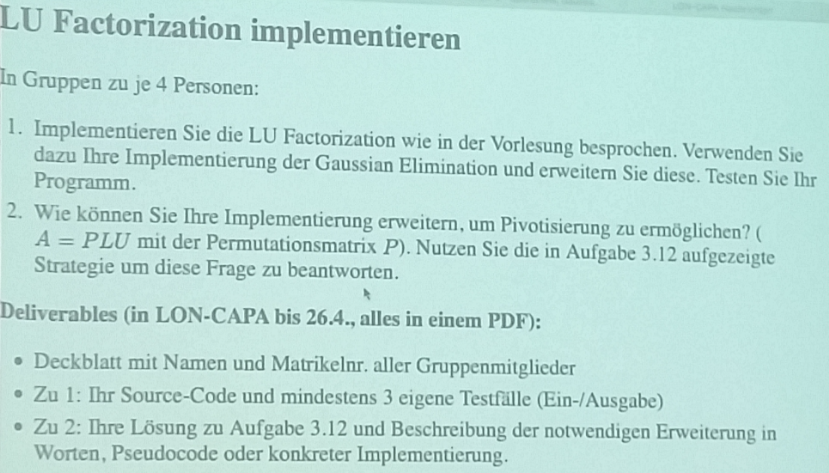
w.zakrevskyy@ostfalia.de

Gruppenmitgileder

Anton Zaiser/Mtr. 70466135  
Lisa Manthai/Mtr. 70466023  
Marc Fiedler /Mtr. 70484207

Numerik LU - Zerlegung

Abgabe zu 26.4.2016



Aufgabe 1

Implementierung sind unter <https://github.com/davidwowa/NumericJava/tree/master/src/de/wdz/numeric/matrix/gj/core> zu finden.

Erweiterung von Gauß zu LU:

1. Gauß-Algorithmus wurde in Forward- und Backward-substitution gesplittet. (Vorher war alles in einer Methode)
2. Es wurde eine Liste erstellt in der die Permutationsmatrizen gespeichert werden
3. Während des Ablaufen von Forward-substituion werden die Permutationsmatrizen gespeichert in der Liste
4. Für Berechnung der U-Matrix werden einfach die Permutationsmatrizen bei jedem Schritt auf einander multipliziert
5. Für Berechnung der L-Matrix werden die S bzw. E Matrizen invertiert und diese dann auch auf einander multipliziert

Die Implementierungen sind auf folgenden Seiten zu finden.

|  |  |
| --- | --- |
| Gauß | U |
| **public** **void** forwardSubstitution(**double**[][] A, **double**[][] B, **double**[][] inverse) {  // iterate over current pivot row p  **for** (**int** p = 0; p < A.length; p++) {  // pivot code here  **if** (A[p][p] != 0.) {  **double** s = 1. / A[p][p]; // s <- 1/u\_pp  A = scale(p, s, A); // Yp <- s \* Yp  B = scale(p, s, B);  inverse = scale(p, s, inverse);  **for** (**int** r = p + 1; r < A.length; r++) {// Eliminate from future  **if** (A[r][p] != 0.) {  s = (-1.) \* A[r][p];  A = addRows(p, r, s, A);// scale row p by s and add to  // row r  B = addRows(p, r, s, B);  inverse = addRows(p, r, s, inverse);  …  // Backward-Substituion | **public** **void** U(**double**[][] A) {  // iterate over current pivot row p  **for** (**int** p = 0; p < A.length; p++) {    // scale row p to make element at (p, p) equal one  **if** (A[p][p] != 0.) {  **double** s = 1. / A[p][p]; // s <- 1/u\_pp  A = scale(p, s, A); // Yp <- s \* Yp  **double**[][] sMatrix = getIdentityMatrix(A.length);  sMatrix = scale(p, s, sMatrix);  getMatrixList().add(sMatrix);  **for** (**int** r = p + 1; r < A.length; r++) {// Eliminate from future  // if (A[r][p] != 0.) {  s = (-1.) \* A[r][p];  A = addRows(p, r, s, A);// scale row p by s and add to  // row r  **double**[][] el = build\_e\_l\_Matrix(A.length, r);  **double**[][] ek = build\_e\_k\_Matrix(A.length, p);  **double**[][] elMatrix = eliminate(s, el, ek);  getMatrixList().add(elMatrix);  … |

L-Methode

|  |
| --- |
| **public** **void** L() {  List<**double**[][]> matrixListWithInverses = **new** ArrayList<>();  **if** (getMatrixList() != **null** && !getMatrixList().isEmpty()) {  **for** (**double**[][] currentMatrix : matrixList) {  **double**[][] inverse = getIdentityMatrix(currentMatrix.length);  forwardSubstitution(currentMatrix, currentMatrix, inverse);  backwardSubstitution(currentMatrix, currentMatrix, inverse);  matrixListWithInverses.add(inverse);  }  } **else** {  System.***out***.println("ERROR: \t\t\t matrix list is emtpy or null");  }  **if** (!matrixListWithInverses.isEmpty()) {  **double**[][] currentResult = matrixListWithInverses.get(matrixListWithInverses.size() - 1);  **for** (**int** i = matrixListWithInverses.size() - 1; i > 1; i--) {  currentResult = mult(currentResult, matrixListWithInverses.get(i));  System.***out***.println("\t\t\t result");  printMatrix(currentResult);  }  System.***out***.println("L");  printMatrix(currentResult);  } **else** {  System.***out***.println("ERROR: \t\t\t list with inverses is emtpy");  }  } |

Aufgabe 2

Implementierung von Pivot.

Algorithmus in Buch zeigt die Stelle wo Pivotisierung eingesetzt werden kann. Folgender Code wurde in diesem Beispiel verbaut:

|  |
| --- |
| // pivot code here  **int** max = p;  **for** (**int** i = 0; i < B.length; i++) {  **if** (Math.*abs*(A[i][p]) > Math.*abs*(A[max][p])) {  max = i;  }  }  // permutate rows  **int**[] sigma = getSigma(max, p, B.length);  A = permute(sigma, A);  B = permute(sigma, B); |

Aufgabe 3.12

1. Die Matrix L kann während des Ablaufs von Algorithmus S oder E sein, da S eine „diagonale“ Matrix ist wird durch diese die Matrix U angenähert.
2. Erster Satz wird schon durch den Algorithmus vorgegeben. (Zweiter Satz) Durch auf multiplizieren von S und E wird auch die Matrix L angenähert(Vorher die Inversen bestimmen).
3. Durch Elimination wird ein Schritt schon vorgegeben.
4. Da die Elimination und Permutiation voneinander unabhängig ausgeführt werden kann gilt der Beschriebene Satz.

Anhang

Beispiel 1

|  |
| --- |
| Coefficient matrix A  Row 1: 4.619355 6.6288204 -5.189272 2.1269035 2.7483473  Row 2: -3.8189888 1.0087395 -7.6598682 1.9509048 5.6306925  Row 3: -3.3356323 -4.944477 -2.2962165 2.2607145 9.696831  Row 4: 9.656389 7.5836506 -9.538363 8.824982 -6.480465  Row 5: -4.500921 -3.741281 -7.422057 -2.6404858 -7.0679674  b = 24.558529 11.175966 37.413776 -0.8937969 -80.15143  Decomposed matrix LU  Row 1: 9.656389 7.5836506 -9.538363 8.824982 -6.480465  Row 2: -0.3954883 4.007984 -11.432179 5.441082 3.0677445  Row 3: -0.46610805 -0.051517297 -12.456921 1.7532192 -9.930522  Row 4: 0.47837293 0.7487573 -0.63687897 -5.0521903 -2.7731104  Row 5: -0.34543267 -0.5800513 0.9811678 -1.3350756 15.278912  x = 1.0000005 1.9999993 2.9999995 3.9999998 5.0 |

Beispiel 2, 3

|  |  |
| --- | --- |
| <---permute---->  1.0 1.0 -2.0 ;  0.0 1.0 -1.0 ;  3.0 -1.0 1.0 ;  with sigma  [2, 1, 0]  result matrix  3.0 -1.0 1.0 ;  0.0 1.0 -1.0 ;  1.0 1.0 -2.0 ;  <---scale---->  3.0 -1.0 1.0 ;  0.0 1.0 -1.0 ;  1.0 1.0 -2.0 ;  with factor 0.3333333333333333 and row 0  result matrix  1.0 -0.3333333333333333 0.3333333333333333 ;  0.0 1.0 -1.0 ;  1.0 1.0 -2.0 ;  <---scale---->  1.0 0.0 0.0 ;  0.0 1.0 0.0 ;  0.0 0.0 1.0 ;  with factor 0.3333333333333333 and row 0  result matrix  0.3333333333333333 0.0 0.0 ;  0.0 1.0 0.0 ;  0.0 0.0 1.0 ;  <---add rows in matrix---->  1.0 -0.3333333333333333 0.3333333333333333 ;  0.0 1.0 -1.0 ;  1.0 1.0 -2.0 ;  row to scale with factor -0.0  [1.0, -0.3333333333333333, 0.3333333333333333]  row to add (first row is scaled)  [-0.0, 0.0, -0.0] + [0.0, 1.0, -1.0]  result matrix  1.0 -0.3333333333333333 0.3333333333333333 ;  0.0 1.0 -1.0 ;  1.0 1.0 -2.0 ;  <---eliminate--->  identity matrix  1.0 0.0 0.0 ;  0.0 1.0 0.0 ;  0.0 0.0 1.0 ;  with factor -0.0  with vector e\_l  0.0 ;  1.0 ;  0.0 ;  with vector e\_k  1.0 0.0 0.0 ;  =  0.0 0.0 0.0 ;  0.0 0.0 0.0 ;  0.0 0.0 0.0 ;  <---add rows in matrix---->  1.0 -0.3333333333333333 0.3333333333333333 ;  0.0 1.0 -1.0 ;  1.0 1.0 -2.0 ;  row to scale with factor -1.0  [1.0, -0.3333333333333333, 0.3333333333333333]  row to add (first row is scaled)  [-1.0, 0.3333333333333333, -0.3333333333333333] + [1.0, 1.0, -2.0]  result matrix  1.0 -0.3333333333333333 0.3333333333333333 ;  0.0 1.0 -1.0 ;  0.0 1.3333333333333333 -2.3333333333333335 ;  <---eliminate--->  identity matrix  1.0 0.0 0.0 ;  0.0 1.0 0.0 ;  0.0 0.0 1.0 ;  with factor -1.0  with vector e\_l  0.0 ;  0.0 ;  1.0 ;  with vector e\_k  1.0 0.0 0.0 ;  =  0.0 0.0 0.0 ;  0.0 0.0 0.0 ;  -1.0 0.0 0.0 ;  <---permute---->  1.0 -0.3333333333333333 0.3333333333333333 ;  0.0 1.0 -1.0 ;  0.0 1.3333333333333333 -2.3333333333333335 ;  with sigma  [0, 2, 1]  result matrix  1.0 -0.3333333333333333 0.3333333333333333 ;  0.0 1.3333333333333333 -2.3333333333333335 ;  0.0 1.0 -1.0 ;  <---scale---->  1.0 -0.3333333333333333 0.3333333333333333 ;  0.0 1.3333333333333333 -2.3333333333333335 ;  0.0 1.0 -1.0 ;  with factor 0.75 and row 1  result matrix  1.0 -0.3333333333333333 0.3333333333333333 ;  0.0 1.0 -1.75 ;  0.0 1.0 -1.0 ;  <---scale---->  1.0 0.0 0.0 ;  0.0 1.0 0.0 ;  0.0 0.0 1.0 ;  with factor 0.75 and row 1  result matrix  1.0 0.0 0.0 ;  0.0 0.75 0.0 ;  0.0 0.0 1.0 ;  <---add rows in matrix---->  1.0 -0.3333333333333333 0.3333333333333333 ;  0.0 1.0 -1.75 ;  0.0 1.0 -1.0 ;  row to scale with factor -1.0  [0.0, 1.0, -1.75]  row to add (first row is scaled)  [-0.0, -1.0, 1.75] + [0.0, 1.0, -1.0]  result matrix  1.0 -0.3333333333333333 0.3333333333333333 ;  0.0 1.0 -1.75 ;  0.0 0.0 0.75 ;  <---eliminate--->  identity matrix  1.0 0.0 0.0 ;  0.0 1.0 0.0 ;  0.0 0.0 1.0 ;  with factor -1.0  with vector e\_l  0.0 ;  0.0 ;  1.0 ;  with vector e\_k  0.0 1.0 0.0 ;  =  0.0 0.0 0.0 ;  0.0 0.0 0.0 ;  0.0 -1.0 0.0 ;  <---permute---->  1.0 -0.3333333333333333 0.3333333333333333 ;  0.0 1.0 -1.75 ;  0.0 0.0 0.75 ;  with sigma  [0, 2, 1]  result matrix  1.0 -0.3333333333333333 0.3333333333333333 ;  0.0 0.0 0.75 ;  0.0 1.0 -1.75 ;  <---scale---->  1.0 -0.3333333333333333 0.3333333333333333 ;  0.0 0.0 0.75 ;  0.0 1.0 -1.75 ;  with factor -0.5714285714285714 and row 2  result matrix  1.0 -0.3333333333333333 0.3333333333333333 ;  0.0 0.0 0.75 ;  -0.0 -0.5714285714285714 1.0 ;  <---scale---->  1.0 0.0 0.0 ;  0.0 1.0 0.0 ;  0.0 0.0 1.0 ;  with factor -0.5714285714285714 and row 2  result matrix  1.0 0.0 0.0 ;  0.0 1.0 0.0 ;  -0.0 -0.0 -0.5714285714285714 ; | <---permute---->  1.0 2.0 3.0 4.0 ;  5.0 6.0 7.0 8.0 ;  9.0 1.0 2.0 3.0 ;  with sigma  [2, 1, 0]  result matrix  9.0 1.0 2.0 3.0 ;  5.0 6.0 7.0 8.0 ;  1.0 2.0 3.0 4.0 ;  <---scale---->  9.0 1.0 2.0 3.0 ;  5.0 6.0 7.0 8.0 ;  1.0 2.0 3.0 4.0 ;  with factor 0.1111111111111111 and row 0  result matrix  1.0 0.1111111111111111 0.2222222222222222 0.3333333333333333 ;  5.0 6.0 7.0 8.0 ;  1.0 2.0 3.0 4.0 ;  <---scale---->  1.0 0.0 0.0 ;  0.0 1.0 0.0 ;  0.0 0.0 1.0 ;  with factor 0.1111111111111111 and row 0  result matrix  0.1111111111111111 0.0 0.0 ;  0.0 1.0 0.0 ;  0.0 0.0 1.0 ;  <---add rows in matrix---->  1.0 0.1111111111111111 0.2222222222222222 0.3333333333333333 ;  5.0 6.0 7.0 8.0 ;  1.0 2.0 3.0 4.0 ;  row to scale with factor -5.0  [1.0, 0.1111111111111111, 0.2222222222222222, 0.3333333333333333]  row to add (first row is scaled)  [-5.0, -0.5555555555555556, -1.1111111111111112, -1.6666666666666665] + [5.0, 6.0, 7.0, 8.0]  result matrix  1.0 0.1111111111111111 0.2222222222222222 0.3333333333333333 ;  0.0 5.444444444444445 5.888888888888889 6.333333333333334 ;  1.0 2.0 3.0 4.0 ;  <---eliminate--->  identity matrix  1.0 0.0 0.0 ;  0.0 1.0 0.0 ;  0.0 0.0 1.0 ;  with factor -5.0  with vector e\_l  0.0 ;  1.0 ;  0.0 ;  with vector e\_k  1.0 0.0 0.0 ;  =  0.0 0.0 0.0 ;  -5.0 0.0 0.0 ;  0.0 0.0 0.0 ;  <---add rows in matrix---->  1.0 0.1111111111111111 0.2222222222222222 0.3333333333333333 ;  0.0 5.444444444444445 5.888888888888889 6.333333333333334 ;  1.0 2.0 3.0 4.0 ;  row to scale with factor -1.0  [1.0, 0.1111111111111111, 0.2222222222222222, 0.3333333333333333]  row to add (first row is scaled)  [-1.0, -0.1111111111111111, -0.2222222222222222, -0.3333333333333333] + [1.0, 2.0, 3.0, 4.0]  result matrix  1.0 0.1111111111111111 0.2222222222222222 0.3333333333333333 ;  0.0 5.444444444444445 5.888888888888889 6.333333333333334 ;  0.0 1.8888888888888888 2.7777777777777777 3.6666666666666665 ;  <---eliminate--->  identity matrix  1.0 0.0 0.0 ;  0.0 1.0 0.0 ;  0.0 0.0 1.0 ;  with factor -1.0  with vector e\_l  0.0 ;  0.0 ;  1.0 ;  with vector e\_k  1.0 0.0 0.0 ;  =  0.0 0.0 0.0 ;  0.0 0.0 0.0 ;  -1.0 0.0 0.0 ;  <---permute---->  1.0 0.1111111111111111 0.2222222222222222 0.3333333333333333 ;  0.0 5.444444444444445 5.888888888888889 6.333333333333334 ;  0.0 1.8888888888888888 2.7777777777777777 3.6666666666666665 ;  with sigma  [0, 1, 2]  result matrix  1.0 0.1111111111111111 0.2222222222222222 0.3333333333333333 ;  0.0 5.444444444444445 5.888888888888889 6.333333333333334 ;  0.0 1.8888888888888888 2.7777777777777777 3.6666666666666665 ;  <---scale---->  1.0 0.1111111111111111 0.2222222222222222 0.3333333333333333 ;  0.0 5.444444444444445 5.888888888888889 6.333333333333334 ;  0.0 1.8888888888888888 2.7777777777777777 3.6666666666666665 ;  with factor 0.18367346938775508 and row 1  result matrix  1.0 0.1111111111111111 0.2222222222222222 0.3333333333333333 ;  0.0 0.9999999999999999 1.0816326530612244 1.163265306122449 ;  0.0 1.8888888888888888 2.7777777777777777 3.6666666666666665 ;  <---scale---->  1.0 0.0 0.0 ;  0.0 1.0 0.0 ;  0.0 0.0 1.0 ;  with factor 0.18367346938775508 and row 1  result matrix  1.0 0.0 0.0 ;  0.0 0.18367346938775508 0.0 ;  0.0 0.0 1.0 ;  <---add rows in matrix---->  1.0 0.1111111111111111 0.2222222222222222 0.3333333333333333 ;  0.0 0.9999999999999999 1.0816326530612244 1.163265306122449 ;  0.0 1.8888888888888888 2.7777777777777777 3.6666666666666665 ;  row to scale with factor -1.8888888888888888  [0.0, 0.9999999999999999, 1.0816326530612244, 1.163265306122449]  row to add (first row is scaled)  [-0.0, -1.8888888888888886, -2.043083900226757, -2.197278911564626] + [0.0, 1.8888888888888888, 2.7777777777777777, 3.6666666666666665]  result matrix  1.0 0.1111111111111111 0.2222222222222222 0.3333333333333333 ;  0.0 0.9999999999999999 1.0816326530612244 1.163265306122449 ;  0.0 2.220446049250313E-16 0.7346938775510208 1.4693877551020407 ;  <---eliminate--->  identity matrix  1.0 0.0 0.0 ;  0.0 1.0 0.0 ;  0.0 0.0 1.0 ;  with factor -1.8888888888888888  with vector e\_l  0.0 ;  0.0 ;  1.0 ;  with vector e\_k  0.0 1.0 0.0 ;  =  0.0 0.0 0.0 ;  0.0 0.0 0.0 ;  0.0 -1.8888888888888888 0.0 ;  <---permute---->  1.0 0.1111111111111111 0.2222222222222222 0.3333333333333333 ;  0.0 0.9999999999999999 1.0816326530612244 1.163265306122449 ;  0.0 2.220446049250313E-16 0.7346938775510208 1.4693877551020407 ;  with sigma  [0, 2, 1]  result matrix  1.0 0.1111111111111111 0.2222222222222222 0.3333333333333333 ;  0.0 2.220446049250313E-16 0.7346938775510208 1.4693877551020407 ;  0.0 0.9999999999999999 1.0816326530612244 1.163265306122449 ;  <---scale---->  1.0 0.1111111111111111 0.2222222222222222 0.3333333333333333 ;  0.0 2.220446049250313E-16 0.7346938775510208 1.4693877551020407 ;  0.0 0.9999999999999999 1.0816326530612244 1.163265306122449 ;  with factor 0.9245283018867926 and row 2  result matrix  1.0 0.1111111111111111 0.2222222222222222 0.3333333333333333 ;  0.0 2.220446049250313E-16 0.7346938775510208 1.4693877551020407 ;  0.0 0.9245283018867925 1.0 1.0754716981132078 ;  <---scale---->  1.0 0.0 0.0 ;  0.0 1.0 0.0 ;  0.0 0.0 1.0 ;  with factor 0.9245283018867926 and row 2  result matrix  1.0 0.0 0.0 ;  0.0 1.0 0.0 ;  0.0 0.0 0.9245283018867926 ; |