Universität zu Köln Wintersemester 2016/2017

Dr. M. Lanser 21. Dezember 2016

10. Übung zur Algorithmischen Mathematik und Programmieren

Hinweis: Beachten Sie alle Abgabeformalitäten, die auf dem ersten Übungszettel angegeben wurden.

Programmieraufgabe 1: (10 Punkte)
Implementieren Sie die LR-Zerlegung einer Matrix A mittels Gauß-Elimination mit Spalten-Pivotsuche; siehe Algorithmus 0.1 für einen Pseudocode einer mögliche Implementierung. Ihre Implementierung sollte nur mit sparse-Matrizen (dünnbesetzten Matrizen) arbeiten. Nutzen Sie also in Matlab *sparse* statt *zeros*, *speye* statt *eye*, etc. Testen Sie die Implementierung am Besten mit der dünnbesetzten Matrix

$$A = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}$$

in verschiedene Größen. Überprüfen Sie die Korrektheit der Software, indem Sie $||PA - LR||_1$ oder $||PA - LR||_\infty$ berechnen und ausgeben. Beachten Sie dabei, dass Sie, falls Sie A bei der Elimination überschreiben, eine Kopie des ursprünglichen A aufbewahren; siehe auch Algorithmus 0.1.
Optional: Optimieren Sie die LR-Zerlegung für sparse-Matrizen, indem Sie Ansätze nutzen wie z. B.: *kji*-Variante, Vektorisierung der inneren Schleifen, oder die Ausnutzung des Nicht-Null-Musters der Matrix (kann mit *find* ermittelt werden). Der schnellste Code in jeder Übungsgruppe gewinnt etwas "Nutzloses".

Sparse-Matrizen in MATLAB

- $sparse(n, m)$ erstellt leere $n \times m$ sparse-Matrizen
- $speye(n, n)$ erstellt $n \times n$ sparse Identität
- $A = full(A)$ wandelt A in dicht besetzte Matrix um
- $[i, j, val] = find(A)$ gibt das Nichtnullmuster von A zurück
- $spy(A)$ plottet ein Bild der Nichtnull-Struktur von A

Algorithmus 0.1 *LR-Zerlegung für sparse-Matrizen mit Pivotsuche in kij-Form*

Gegeben: *Invertierbare sparse-Matrix* $A \in \mathbb{K}^{n\times n}$

Gesucht: *Faktoren P (Permutationsmatrix), L (linke untere Dreiecksmatrix) und R (rechte obere Dreiecksmatrix) mit $PA = LR$*

Setze: $P = I$ *(Matlab: $P = speye(n, n)$)*

Setze: $L = 0$ *(Matlab: $L = sparse(n, n)$)*

Speichere Kopie: $A_1 = A$

for $k = 1 : n - 1$

  Setze: $m = k$

  for $i = k + 1 : n$

    if $(|a_{ik}| > |a_{mk}|)$

      Setze: $m = i$

    end

  end

  tausche: *Zeilen m und k in A und P und L*

  for $i = k + 1 : n$

    $l_{ik} = \frac{a_{ik}}{a_{kk}}$

    for $j = k + 1 : n$

      $a_{ij} = a_{ij} - l_{ik} \cdot a_{kj}$

    end

  end

end

Setze: *R auf den obere Dreiecksteil von A inklusive Diagonale (Matlab: $R = triu(A)$)*

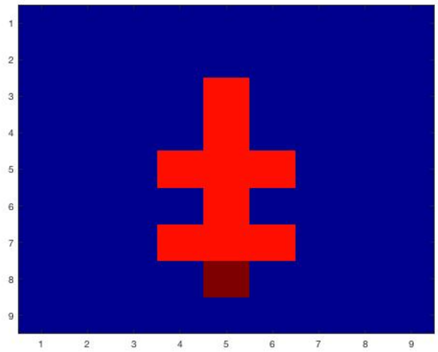
Setze: $L = L + Id$

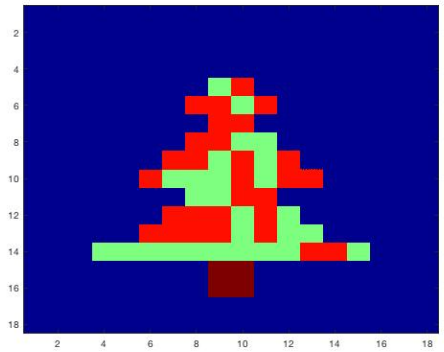
Überprüfe: $PA_1 = LR$

Frohe Weihnachten und einen guten Rutsch!

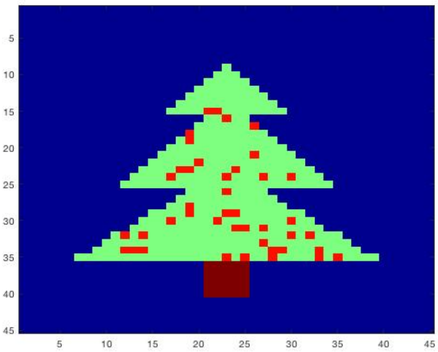
Die Funktion *xmas_tree.m* steht auf der Homepage zum Download zur Verfügung! Ab der Auflösung 45×45 sieht es auch nach was aus. Die roten Kleckse sind sogenannte *random*-Weihnachtsbaumkugeln.

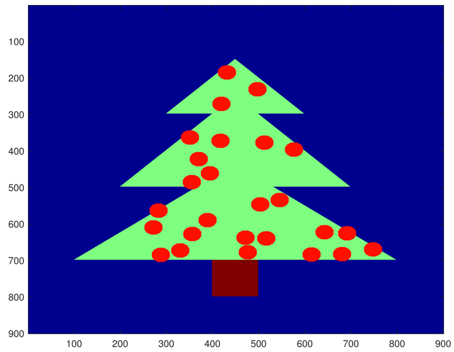
xmas_tree(9) und *xmas_tree*(18):





xmas_tree(45) und *xmas_tree*(900):





Abgabedatum: 12.01.2017, 12 Uhr in den Kasten im Mathematischen Insitut