# Numerik-Programmierabgabe: Blatt 6

Gruppe: 3

Martin Bieker
Julian Surmann

## 1 Funktion LU = LU_decompose(A)

Diese Funktion berechnet aus einer gegebenen Matrix A die LR-Zerlegung und gibt diese platzsparend in einer Matrix LU wieder zurück. Zunächst werden die Quotienten $q_i$ der Matrixelemente $A_{i-1,i}...A_{n,i}$ mit dem Pivotelement $A_{i,i}$ berechnet. Danach werden die Elemente der oberen Dreiecksmatrix bestimmt.

Listing 1: Skript - *LU_ decompose.m*

```
function LU = LU_decompose(A)

%Groesse der Matrix bestimmen

[m, n] = size(A);

if m ~= n
printf('Fehler!! A muss quadratisch sein.')
end

for i = [1:n-1]

% Zum Bestimmen der q-Werte wird die Aktuelle Alles unter der
%Aktuellen Spalte durch das Pivot-Element geteilt

q = A(i+1:n,i)/ A(i,i);

%Die Werte der i-ten Zeile werden mit q geichtet. Diese werden
%danch von der unteren Ecke der Matrix abgezogen.

A(i+1:n,i+1:n)= A(i+1:n,i+1:n)- q*A(i,i+1:n);
% Die Matrix L besteht aus den q-werten der einzelnen Schritte und
%wird in der Unteren Ecke von A gespeichert
A(i+1:n,i)= q;
end
LU = A;
end
```

## 2 Funktion `z = forward_solve(LU, b)`

Diese Funktion löst das LGS

$$L \cdot \vec{z} = \vec{b}$$

durch Vorwärtseinsetzen. Dabei wird ausgenutzt, dass die Elemente auf der Hauptdiagonalen der unteren Dreiecksmatrix gleich eins sind. Daher gilt für $z_i$:

$$z_i = x_i - \sum_{j=1}^{i-1} L_{i,j} \cdot b_j.$$

Listing 2: Skript - *forward_solve.m*

```
%Diese Funktion soll das LGS Lz= b loesen. Uebergeben wird
%die LR-Zerlegung einer Matrix und die rechte Seite b.
function z = forward_solve(LU, b)
    [m,n] = size(LU);
    z = zeros(n,1);
    if n ~= m
        printf('Fehler LU ist nicht quadratisch!')
    end
    for i = 1:n
        %Hier muss nicht am Ende durch das Diagonalelement geteilt
        %werden,da die Elemente der R-Matrix dort immer  1 sind.
        z(i) = b(i)-LU(i,1:i-1)*z(1:i-1);
    end
end
```

## 3 Funktion `x = backward_solve(LU, z)`

Als letzter Schritt zur Bestimmung des Lösungsvektors $\vec{x}$ ist das Gleichungssystem mit der Oberen Dreiecksmatrix

$$R \cdot \vec{x} = \vec{z}$$

zu lösen. Dazu werden die Schritte aus Abschnitt 2 rückwärts durchlaufen. Es gilt für das $i$-te Element von $\vec{x}$:

$$x_i = \frac{z_i - \sum_{j=i+1}^{n} A_{i,j} \cdot x_j}{A_{i,i}}$$

Listing 3: Skript - *backward_ solve.m*

```
%backward_solve.m
%Diese Funktion loesst das LGS Rx=z und liefert damit die entgueltige Loesung
%des LGS Ax=b

function x = backward_solve(LU, z)
    [m,n] = size(LU);
    x = zeros(n,1);
    if n ~= m
        printf('Fehler LU ist nicht quadratisch!')
    end
    %Rueckwerts einsetzen von n bis 1
    for i = n:-1:1
        %Hier muss am Ende durch das Diagonalelement geteilt werden
        %da die Elemente der R-Matrix dort nicht zwingend 1 sind.
        x(i) = (z(i) - LU(i,i+1:n)*x(i+1:n))/LU(i,i);
    end
end
```

## 4 Skript `LU_test.m`

Um die oben implementierten Funktionen zu testen, werden in diesem Skript 3 zufällige Matrizen und rechte Seiten verschiedener Größe erzeugt. Danach wird die LR-Zerlegung der Matrix bestimmt. Die Lösung des LGS wird dann mit Hilfe der Funktionen `forward_solve` und `backward_solve` bestimmt. Zum Vergleich werden die Gleichungssysteme auch mit dem in Octave integrierten Operator bestimmt.

Listing 4: Skript - *LU_ Test.m*

```
%LU_test.m
%Dieses Skript testet die Implemntierung der LR_Zerlegung mit
%Zufalls-Matrizen und Vektoren unterschiedlicher Grosse.

%die Funktionen mit mehreren LGS verschiedener Groessen testen
for n =[3,4,10]

%Zufaellige Matrizen und Vektoren erzeugen
A = rand(n);
b = rand(n,1);

%LR-Zerlegung erstellen

LU = LU_decompose(A);

%vorwaertseisetzen
z = forward_solve(LU,b);

%ruekwerts einsetzen

printf(['LR Zerlegung', '\n'])
x = backward_solve(LU,z)

%Zum Vergleich LSG direkt mit MATLAB loesen
printf(['Direkt mit MATLAB / Octave:', '\n'])
x_test = A\b
printf('######################\n')
end
```