Dr. R.Käppeli
M. Sprecher

Musterlösung 4

1. a)

```
function L = cholesky_dec(A);

% function L = cholesky_dec(A);
%
% Purpose: Cholesky-Zerlegung einer symmetrischen positiv definiten
%          Matrix A
%
% Input:  A ... Matrix
%
% Output: L ... untere (lower) Dreiecks-Matrix
%

  % A muss quadratisch sein!
  [m,n] = size(A);
  if ( m ~= n )
    error('A Matrix nicht quadratisch!!!');
  end

  % Speicherplatz fuer L
  L = zeros(n,n);

  % Cholesky-Zerlegung
  for i=1:n
    L(i,i) = sqrt(A(i,i));
    % Ueberpruefe, dass das Pivotelement positiv groesser eps ist!
    if ( L(i,i) <= eps )
      error('Pivotelement wurde <= 0!!!');
    end
    for j=i+1:n
      L(j,i) = A(j,i)/L(i,i);
      for k=i+1:j
        A(j,k) = A(j,k) - L(j,i)*L(k,i);
      end
    end
```

Bitte wenden!

```
end
```

b)

```
function x = cholesky_solve(L,b);

% function x = cholesky_solve(L,b);
%
% Purpose: loese lineares Gleichungs-System A x = b gegeben mit
%          Cholesky-Zerlegung der symmetrisch positiv definiten
%          Matrix A
%
% Input:  L ... untere (lower) Dreiecks-Matrix
%
% Output: x ... Loesung
%

% L muss quadratische Matrix sein!
  [mL,nL] = size(L);
  if ( mL ~= nL )
    error('L Matrix nicht quadratisch!!!');
  end
  n = nL; % da nL = mL!

% Vorwaerts einsetzen  L y = b --> y
  y = zeros(n,1); % Speicher fuer y
  for i=1:n
    y(i) = b(i);
    for j=1:i-1
      y(i) = y(i) - L(i,j)*y(j);
    end
    y(i) = y(i)/L(i,i);
  end

% Rueckwerts einsetzen L' x = y --> x
  x = zeros(n,1); % Speicher fuer x
  for i=n:-1:1
    x(i) = y(i);
    for j=i+1:n
      x(i) = x(i) - L(j,i)*x(j);
    end
    x(i) = x(i)/L(i,i);
  end
```

Siehe nächstes Blatt!

2. **a)** Es gilt $(A^TA)^T = A^T(A^T)^T = A^TA$, die Matrix A^TA ist also symmetrisch. Es gilt

$$x^TA^TAx = (Ax)^T(Ax) = \|Ax\|_2^2 \geq 0.$$

Zudem folgt, dass $x^TA^TAx = (Ax)^T(Ax) = 0$ genau dann wenn $\|Ax\|_2 = 0$. Dies ist genau dann der Fall wenn $Ax = 0$. Mit der Dimensionsformel folgt $dimKernA = n - dimBildA = n - n = 0$ und deshalb, dass $Ax = 0$ genau dann wenn $x = 0$. Die Matrix A^TA ist also positiv definit. Da A^TA symmetrisch und positiv definit ist, ist sie insbesondere invertierbar. Folglich besitzt die Normalengleichung $A^TAx = 0$ genau eine Lösung $\bar{x}$.
Aus denselben Gründen wie oben folgt, dass die Matrix AA^T symmetrisch und $x^TA^TAx \geq 0$ gilt. Da $dimKernA^T = m - n > 0$ existiert aber ein $x \in \mathbb{R}^m \backslash \{0\}$ mit $A^Tx = 0$. Es folgt $x^TAA^Tx = 0$. Die Matrix AA^T ist also **nicht** positiv definit.

b) Es gilt

$$(A(x-\bar{x}))^T(A\bar{x}-b) = (x-\bar{x})^TA^T(A\bar{x}-b) = (x-\bar{x})^T(A^TA\bar{x}-A^Tb) = (x-\bar{x})^T0 = 0.$$

Die Vektoren $A(x-\bar{x})$ und $A\bar{x}-b$ stehen also senkrecht. Nach Pythagoras gilt also

$$\|A\bar{x}-b\|_2^2 + \|A(x-\bar{x})\|_2^2 = \|A\bar{x}-b+A(x-\bar{x})\|_2^2 = \|Ax-b\|_2^2.$$

c) Aus b) folgt

$$\varphi(x) = \varphi(\bar{x}) + \frac{1}{2}\|A(x-\bar{x})\|_2^2 \geq \varphi(\bar{x}).$$

d) Wir erhalten

$$\frac{\partial\varphi}{\partial x_i} = \frac{1}{2}\left(\sum_{k=1}^{n}(A^TA)_{ik}x_k + \sum_{j=1}^{n}x_j(A^TA)_{ji}\right) - (A^Tb)_i.$$

Da A^TA symmetrisch ist gilt $(A^TA)_{ji} = (A^TA)_{ij}$ und daher

$$\begin{aligned}\frac{\partial\varphi}{\partial x_i} &= \sum_{k=1}^{n}(A^TA)_{ik}x_k - (A^Tb)_i \\ &= (A^TAx)_i - (A^Tb)_i.\end{aligned}$$

Also gilt $\nabla\varphi = A^TAx - A^Tb$.

3. **a)** Resultat

$$A = QR = \begin{pmatrix} 4/5 & -3/5 \\ 3/5 & 4/5 \end{pmatrix}\begin{pmatrix} 5 & -1 \\ 0 & 2 \end{pmatrix}$$

Bitte wenden!

b)

```
function [Q,R] = qr_mgs(A);

% function [Q,R] = qr_mgs(A);
%
% Purpose: QR-Zerlegung einer m x n Matrix mit modifiziertem Gram-Sc
%          Orthogonalisierungs-Verfahren
%
% Input:  A ... m x n Matrix
%
% Output: Q ... orthogonale/unitaere Matrix
%         R ... obere (upper) Dreiecks-Matrix
%

  % Groesse von A
  [m,n] = size(A);

  % Speicherplatz fuer Q & R
  Q = zeros(m,n);
  R = zeros(n,n);

  % modifizierter Gram-Schmidt Algorithmus
  for j=1:n
    v = A(:,j); % v beginnt als j-te Spalte von A und wird dann
                % sukzessive orthogonalisert auf die bereits
                % bekannten Spalten von Q
    for i=1:j-1
      % R(i,j) = Q(:,i)'*A(:,j); % vectorized version!
      R(i,j) = Q(1,i)*A(1,j);
      for k=2:m
        R(i,j) = R(i,j) + Q(k,i)*A(k,j);
      end
      % v = v - R(i,j)*Q(:,i); % vectorized version!
      for k=1:m
        v(k) = v(k) - R(i,j)*Q(k,i);
      end
    end
    R(j,j) = norm(v);
    Q(:,j) = v/(R(j,j) + eps);
  end
```

Siehe nächstes Blatt!

c) Es gilt $\left\| \begin{pmatrix} 4 \\ 3 \end{pmatrix} \right\|_2 = 5$ und daher

$$w = \begin{pmatrix} 4 \\ 3 \end{pmatrix} - 5 \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} -1 \\ 3 \end{pmatrix}$$

sowie

$$\tilde{w} = \begin{pmatrix} 4 \\ 3 \end{pmatrix} + 5 \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 9 \\ 3 \end{pmatrix} = 3 \begin{pmatrix} 3 \\ 1 \end{pmatrix}.$$

Es folgt

$$H = H(w) = \frac{1}{5} \begin{pmatrix} 4 & 3 \\ 3 & -4 \end{pmatrix}$$

und

$$\tilde{H} = H(\tilde{w}) = H(\frac{1}{3}\tilde{w}) = \frac{1}{5} \begin{pmatrix} -4 & -3 \\ -3 & 4 \end{pmatrix}.$$

Um die QR-Zerlegung zu erhalten setzt man $Q = H^T$ bzw. $Q = \tilde{H}^T$. Die Matrix R lässt sich mittels $R = Q^T A$ berechnen. Man beachte, dass die QR-Zerlegung bis auf Vorzeichen eindeutig ist.

d)

```
function [Q,R] = qr_householder(A);

% function [Q,R] = qr_householder(A);
%
% Purpose: QR-Zerlegung einer m x n Matrix mit Householder
%          Spiegelungen
%
% Input:  A ... m x n Matrix
%
% Output: Q ... orthogonale/unitaere Matrix
%         R ... obere (upper) Dreiecks-Matrix
%

  % Groesse von A
  [m,n] = size(A);

  % Speicherplatz fuer Q & R
  Q = zeros(m,m);
  R = zeros(m,n);

  % Speicherplatz fuer die Householder-Spiegelungs Vektoren
  U = zeros(m,n);

  % QR-Zerlegung mit Householder-Transformationen
```

Bitte wenden!

```
R = A;
for j=1:n
  w = R(j:m,j);
  % falls m < n ist, breche bei hier ab wenn j > m
  if ( j > m )
    break
  end
  % waehle Spiegelung mit 'kleinster' Wirkung
  if ( w(1) >= 0 )
    w(1) = w(1) - norm(w);
  else
    w(1) = w(1) + norm(w);
  end
  u = w/(norm(w) + eps); % normiere Spiegelungs Vektor
  U(j:m,j) = u; % speichere normierten Spieglungs Vektor in U
  % wende die Householder-Spiegelung auf A(j:m,j:n) an
  for i=j:n
    R(j:m,i) = R(j:m,i) - 2.*u*(u'*R(j:m,i));
  end
end

% bauen nun Q' auf
Q = eye(m) - 2.*U(:,1)*U(:,1)';
for i=2:n
  H = eye(m) - 2.*U(:,i)*U(:,i)';
  Q = H*Q;
end
Q = Q';
```

4. Die Matrix und die rechte Seite des Ausgleichsproblems sind

$$A = \begin{pmatrix} 1.1 \\ 1.9 \\ 2.7 \\ 4 \\ 5.2 \end{pmatrix} \quad \text{und} \quad b = \begin{pmatrix} 5 \\ 10 \\ 15 \\ 20 \\ 25 \end{pmatrix}.$$

Folgender Matlabcode liefert die Lösung (≈ 4.9864):

```
A=[1.1;1.9;2.7;4;5.2];
b=[5;10;15;20;25];
x=A\b
```