Prof. Dr. Stefan Funken
M.Sc. Andreas Bantle
Dipl.-Math. oec. Klaus Stolle

Universität Ulm
Institut für Numerische Mathematik
Sommersemester 2015

Numerische Analysis - Matlab-Blatt 2
Lösung

(Abgabe in den entsprechenden Matlab-Tutorien)

Aufgabe 8 *(Newton Verfahren)* (10 Punkte)

Gegeben sei die vektorwertige Funktion

$$f : \mathbb{R}^2 \to \mathbb{R}^2, \quad f(x) = \begin{pmatrix} x_1^2 + x_2^2 + 0.6x_2 - 0.16 \\ x_1^2 - x_2^2 + x_1 - 1.6x_2 - 0.14 \end{pmatrix}.$$

(i) Implementieren Sie das Newtonverfahren in MATLAB.

- Zur Auswertung der Funktion f schreiben Sie eine MATLAB-Funktion `f.m` die wiefolgt augerufen wird

  `function [y, dy] = f(x)`,

  wobei x, y Spaltenvektoren sind und dy die Jacobimatrix an der Stelle x ist.
- Als Abbruchkriterium benutzen Sie $\|d\|_2 <$ `tol`, mit Newtonrichtung d und Toleranz `tol` $= 10^{-10}$.

```
function [y, varargout] = f(x)

y = [x(1)^2 + x(2)^2 +         0.6*x(2) - 0.16;
     x(1)^2 - x(2)^2 + x(1) - 1.6*x(2) - 0.14];

if (nargout>1)
  dy = [2*x(1)     ,  2*x(2) + 0.6; ...
        2*x(1) + 1, -2*x(2) - 1.6];
  varargout = {dy};
end
```

(ii) Bestimmen Sie die Lösung x von $f(x) = 0$ für den Startwert $x^{(0)} = (0.6, 0.25)^T$.
Geben Sie in jeder Iteration folgende Größen aus

$x^{(k)}$:	aktuelle Iterierte;
$d^{(k)}$, $\|\|d^{(k)}\|\|_2$:	aktuelle Newtonrichtung mit euklidische Norm;
$e^{(k)} := \|\|x^{(k)} - x\|\|_2$:	Abstand der aktuellen Iterierten von der Lösung x.

```
clear all;
format short e;

% Toleranz
tol = 1e-10;
% Maximale Anzahl an Iterationen
maxIt = 100;

x = zeros(2,maxIt);
d = zeros(2,maxIt);

% Startwert
x(:,1) = [0.60; 0.25];
```

```
14
15 for k=1:maxIt
16   % Auswerten der Funktion und der Jacobimatrix
17   [y,dy] = f(x(:,k));
18
19   % Bestimme Newton Richtung
20   d(:,k) = dy \ y;
21
22   % Falls Norm der Richtung < Toleranz -> Stop
23   if (norm(d(:,k))<tol)
24     break;
25   end
26
27   % Bestimmung der naechsten Iterierten
28   x(:,k+1) = x(:,k) - d(:,k);
29 end
30
31 for i=1:k
32   eNorm(i) = norm(x(:,i)-x(:,k),2);
33   dNorm(i) = norm(d(:,i),2);
34 end
35
36 % Ausgabe Groessen fuer Aufgabenteil 2.2
37 [1:k; x(:,1:k); d(:,1:k); dNorm; eNorm]'
38
39 % Plot fuer Aufgabenteil 1.3
40 figure;
41 semilogy(eNorm);
42 grid on;
```

(iii) Plotten Sie den Fehler $e^{(k)}$ für jeden Iterationsschritt (Semilogarithmisch).

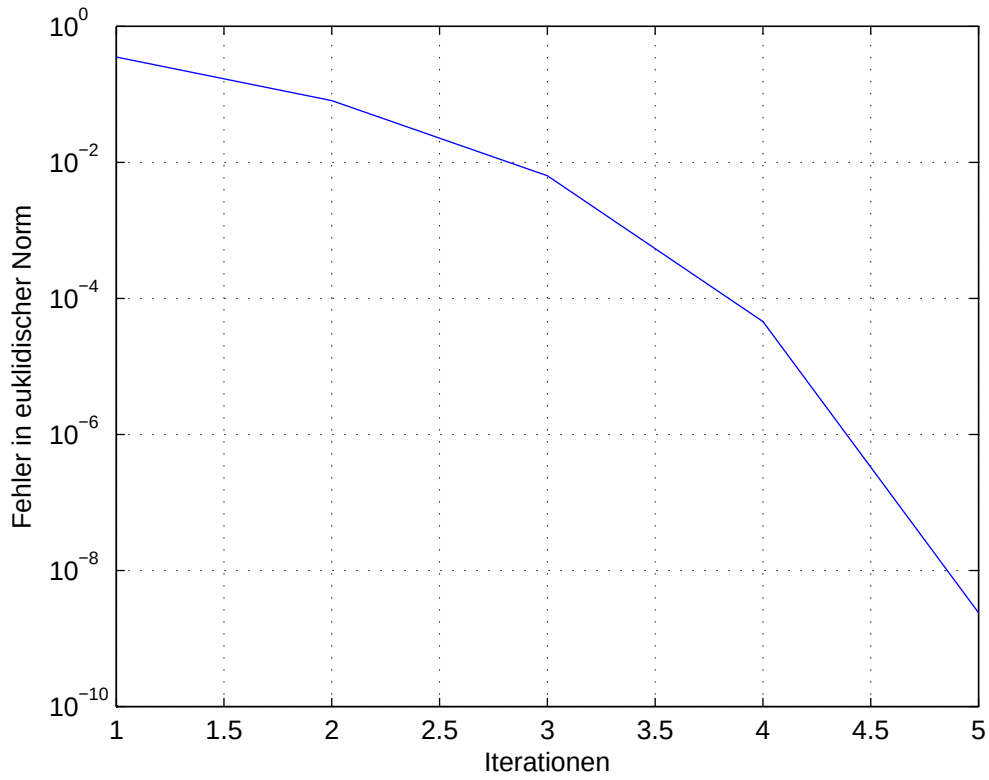

Abbildung 1: Fehlerplot der Funktion f

Erläuterungen:

(a) *Wie beim Newton in einer Dimension ist quadratisches Konvergenzverhalten zu erkennen (i.A nur lokal*

quadratisch)

(b) *Man beachte, dass nur die Y-Achse logarithmiert ist*

Aufgabe 3 *(Rosenbrock-Funktion)* (10 Punkte)

Gegeben sei die sogenannte Rosenbrock-Funktion

$$g : \mathbb{R}^2 \to \mathbb{R}, \quad g(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2.$$

(i) Zeichnen Sie die Niveaulinien im Gebiet $[-1.5, 1.5] \times [-1, 2]$.
(**Tipp:** Suchen Sie in der MATLAB-Hilfe die Funktion `contour`).

(ii) Lösen Sie das Minimierungsproblem $g(x) \to \min_x$, indem Sie die Sie mit Hilfe des Newtonverfahrens die Nullstelle von $0 = f(x) := \nabla g(x)$ bestimmen.

- Gehen Sie dabei vor wie in Aufgabe 2 (i) mit $x^{(0)} = (-1, 1)^T$.
- Plotten Sie die Iterierten $x^{(k)}$, $k = 0, 1, \ldots$ in den unter 1 erzeugten Graphen.

```
clear all;
clc;
clear all
format short e;

% Contourplot fuer Aufgabenteil 1
dx = 1e-2;
dy = 1e-2;
[X,Y] = meshgrid(-1.5:dx:1.5, -5:dy:2);
Z = rosen(X,Y);
contour(X,Y,Z,logspace(-2,3,20));
axis tight
colorbar
hold on

% Loesung mittels Newton Verfahren fuer den Gradienten
% Toleranz
tol = 1e-10;
% Maximale Anzahl an Iterationen
maxIt = 1e04;

% Startwert
x = [-1.5; 1.5];
plot(x(1),x(2),'ro');

for k=1:maxIt
  % Auswerten der Funktion und der Jacobimatrix
  [y,dy] = f_rosen(x);

  % Bestimme Newton Richtung
  d = dy \ y;

  % Falls Norm der Richtung < Toleranz -> Stop
  if (norm(d)<tol)
    break;
  end

  % Bestimmung der naechsten Iterierten
  x = x - d;
  plot(x(1),x(2),'ro');
  x
end
```

```
function [y, varargout] = f_rosen(x)

```

```
y = [-400*x(1)*(x(2)-x(1)^2) + 2*x(1) - 2;
     200*(x(2)-x(1)^2)];

if (nargout>1)
  dy = [1200*x(1)^2 - 400*x(2) + 2, -400*x(1);
        -400*x(1),         200];
  varargout = {dy};
end
```

```
function z = rosen(x,y)

z = 100*(y-x.^2).^2 + (1-x).^2;
```

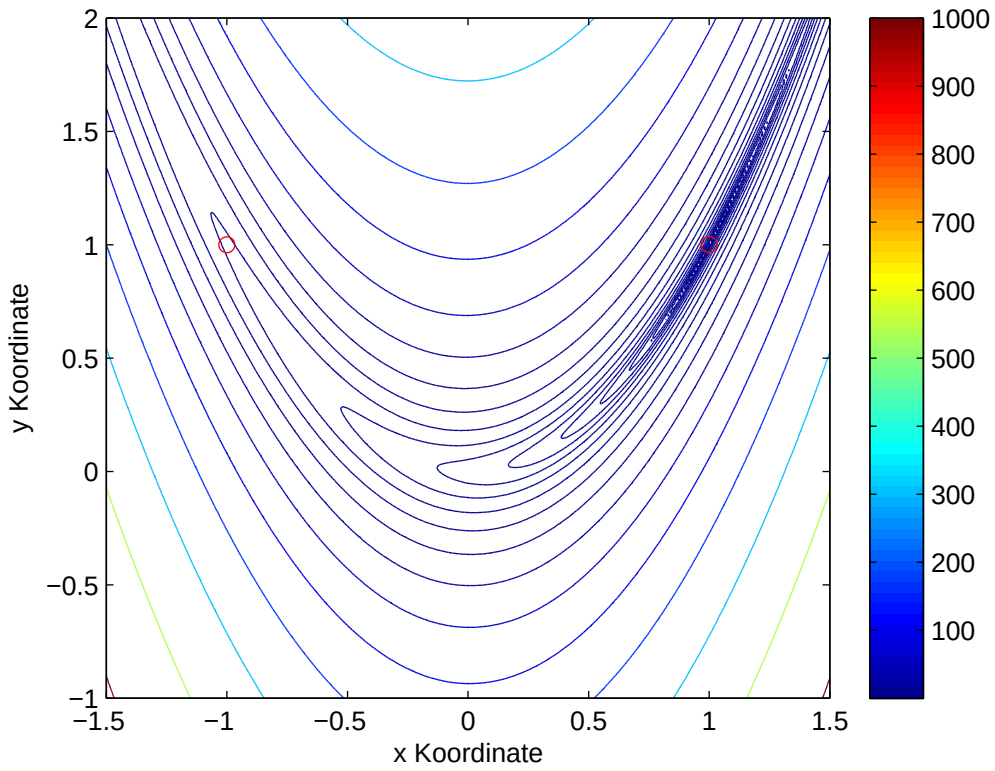

Abbildung 2: Höhenlinien und Iterationsverlauf

Erläuterungen:

(a) *Die Iterierten konvergieren gegen den Punkt (1,1).*

(b) *Die Forderung ist* $\Delta g = 0$ *ist nur ein notwendiges Kriterium für ein lokales Minimum (Analysis 2)*