



TECHNISCHE  
UNIVERSITÄT  
WIEN

N U M E R I K P R O J E K T

**Titel**  
**ggf. mehrzeilig**

ausgeführt am

Institut für  
Analysis und Scientific Computing  
TU Wien

unter der Anleitung von

**Name des Betreuers**

durch

**Markus Rinke**

Matrikelnummer: 1402581

**Stefan Schrott**

Matrikelnummer: 1607388

Wien, am 21. Januar 2018

## Inhaltsverzeichnis

|  |          |
|--|----------|
| <b>1 Grundlagen</b>  | <b>1</b> |
| <b>2 Implementierung von Aufgabe a</b>                     | <b>2</b> |
| 2.1 Tests . . . . .  | 2        |
| <b>3 Implementierung von Aufgabe b Version 1</b>           | <b>2</b> |
| 3.1 Tests . . . . .  | 2        |
| <b>4 Implementierung von Aufgabe b Version 2</b>           | <b>2</b> |
| 4.1 Tests . . . . .  | 2        |
| <b>5 Implementierung von adaptiver Schrittweite</b>        | <b>2</b> |
| 5.1 Tests . . . . .  | 2        |
| <b>6 Implementierung von Niveaulinien</b>                  | <b>2</b> |
| 6.1 Problemstellung und Idee der Implementierung . . . . . | 2        |
| 6.2 Details der Implementierung . . . . .                  | 3        |
| 6.3 Tests . . . . .  | 4        |
| <b>7 Anhang: Code-Listings</b>                             | <b>5</b> |

## 1 Grundlagen

Die Grundlage für die folgenden Überlegung ist der Hauptsatz über implizite Funktionen im Spezialfall von Funktionen  $F : A \times B \rightarrow \mathbb{R}$ , wobei  $A$  und  $B$  der Einfachheit halber offene Intervalle seien.

**Satz:** Seien  $a < b$  sowie  $c < d \in \mathbb{R}$  und  $F : (a, b) \times (c, d) \rightarrow \mathbb{R}$  stetig differenzierbar. Seien  $x_0 \in (a, b)$  und  $y_0 \in (c, d)$ , sodass  $F(x_0, y_0) = 0$  und  $\frac{\partial F}{\partial y}(x_0, y_0) \neq 0$ .

Dann existieren  $a_0, b_0 \in \mathbb{R}$  mit  $a < a_0 < x_0 < b_0 < b$  und eine stetig differenzierbare Funktion  $f : (a_0, b_0) \rightarrow \mathbb{R}$  mit  $f(x_0) = y_0$ , sodass

$$\forall x \in (a_0, b_0) : F(x, f(x)) = 0$$

und

$$\forall x \in (a_0, b_0) : f'(x) = -\frac{\frac{\partial F}{\partial x}(x, f(x))}{\frac{\partial F}{\partial y}(x, f(x))}. \quad (1)$$

**Beweis:** Unter den gegebenen Voraussetzungen ist der Hauptsatz über implizite Funktionen anwendbar und liefert Umgebungen  $U$  von  $x_0$  und  $V$  von  $y_0$  und eine Funktion  $f : U \rightarrow V$  mit den geforderten Eigenschaften. Da  $x_0$  ein innere Punkt von  $U$  ist, enthält  $U$  ein Intervall  $(a_0, b_0)$  mit den geforderten Eigenschaften.

Die Umgebung  $V \subseteq \mathbb{R}$  in der Zielmenge von  $f$  kann durch ganz  $\mathbb{R}$  ersetzt werden, da wir nur behauptet haben, dass  $y = f(x)$  eine Lösung von  $F(x, \cdot) = 0$  ist, allerdings nicht dass diese eindeutig ist. ■

**Satz:** Sei unter den Voraussetzungen des vorherigen Satz  $F$  zwei mal stetig differenzierbar.

Dann ist  $f \in C^2((a_0, b_0))$  mit  $f''(x) =$

$$\frac{-\frac{\partial^2 F}{\partial^2 x}(x, f(x)) \left(\frac{\partial F}{\partial y}(x, f(x))\right)^2 + 2\frac{\partial^2 F}{\partial x \partial y}(x, f(x)) \frac{\partial F}{\partial x}(x, f(x)) \frac{\partial F}{\partial y}(x, f(x)) - \frac{\partial^2 F}{\partial^2 y}(x, f(x)) \left(\frac{\partial F}{\partial x}(x, f(x))\right)^2}{\left(\frac{\partial F}{\partial y}(x, f(x))\right)^3}.$$

Außerdem gilt:

$$\forall x \in (a_0, b_0) \exists \xi \in (x_0, x) \cup (x, x_0) : f(x) = y_0 + \frac{\frac{\partial F}{\partial x}(x_0, y_0)}{\frac{\partial F}{\partial y}(x_0, y_0)}(x - x_0) + \frac{f''(\xi)}{2}(x - x_0)^2.$$

**Beweis:** Aus  $F \in C^2$  folgt mit der Kettenregel und Einsetzen der Darstellung (1) für  $f'$ :

$$\begin{aligned} \frac{d}{dx} \left( \frac{\partial F}{\partial x}(x, f(x)) \right) &= \left( \frac{\partial^2 F}{\partial^2 x}(x, f(x)), \frac{\partial^2 F}{\partial x \partial y}(x, f(x)) \right) \cdot \begin{pmatrix} 1 \\ f'(x) \end{pmatrix} \\ &= \frac{\partial^2 F}{\partial^2 x}(x, f(x)) - \frac{\partial^2 F}{\partial x \partial y}(x, f(x)) \frac{\frac{\partial F}{\partial x}(x, f(x))}{\frac{\partial F}{\partial y}(x, f(x))}. \end{aligned}$$

Für  $\frac{d}{dx} \left( \frac{\partial F}{\partial y}(x, f(x)) \right)$  erhält man analog eine ähnliche Darstellung. Damit kann man den Ausdruck (1) mithilfe der Quotientenregel differenzieren und erhält durch Erweitern mit  $\frac{\partial F}{\partial y}(x, f(x))$  obige Darstellung für  $f''$ .

Die zweite Aussage folgt aus dem Satz von Taylor und der Tatsache, dass  $f''$  als Komposition stetiger Funktionen stetig ist. ■

## 2 Implementierung von Aufgabe a

### 2.1 Tests

## 3 Implementierung von Aufgabe b Version 1

### 3.1 Tests

## 4 Implementierung von Aufgabe b Version 2

### 4.1 Tests

## 5 Implementierung von adaptiver Schrittweite

### 5.1 Tests

## 6 Implementierung von Niveaulinien

### 6.1 Problemstellung und Idee der Implementierung

Die bisherigen Algorithmen finden Paare  $(x_i, y_i)_{i=1, \dots, N}$ , sodass für  $F(x_i, y_i) = 0$  für  $i = 1, \dots, N$  und stellen damit die Nullstellenmenge von  $F$  (oder nur einen Teil davon) näherungsweise graphisch dar.

Im Folgenden sind  $c_1, \dots, c_k \in \mathbb{R}$  gegeben und es sollen für  $j = 1, \dots, k$  die Teilmengen von  $\{(x, y) \in \mathbb{R}^2 : F(x, y) = c_j\}$  graphisch dargestellt werden.

Grundsätzlich ist dieses Problem einfach auf die vorherigen Algorithmen zurückzuführen, indem man die Nullstellenmengen der Funktionen  $F_j(x, y) := F(x, y) - c_j$  graphisch darstellt.

Bei den vorherigen Algorithmen musste ein Startwert  $(x_0, y_0) \in \mathbb{R}^2$  übergeben werden, für den gilt  $F(x_0, y_0) = 0$ , also müsste in diesem Fall  $k$  Startwerte  $(x_j, y_j) \in \mathbb{R}^2$  übergeben werden, sodass

$$F(x_j, y_j) = c_j \quad j = 1, \dots, k.$$

Dies stellt sich in der Praxis als sehr benutzerunfreundlich heraus, da die Gleichungen  $F(x_j, y_j) = c_j$  im Allgemeinen nicht einfach zu lösen sind.

Aus diesem Grund wurde ein Algorithmus implementiert, der in einem gegebenen Intervall  $[a, b] \times [c, d] \subseteq \mathbb{R}^2$  entsprechende  $(x_j, y_j)$  sucht und anschließend für  $j = 1, \dots, k$  einen der vorherigen Algorithmen mit der Funktion  $F_j$  und den Startwerten  $(x_j, y_j)$  aufruft.

Der wesentliche Schritt ist also, nach Möglichkeit Nullstellen von  $F_j$  in  $[a, b] \times [c, d]$  zu finden. Das Newton-Verfahren im  $\mathbb{R}^n$  steht hier nicht zur Verfügung, da nur es für Funktionen  $G : \mathbb{R}^n \rightarrow \mathbb{R}^n$  anwendbar ist. Wegen der Regularitätsforderung an die Jacobi-Matrix von  $G$ , ist es auch nicht möglich etwa  $G(x, y) := \begin{pmatrix} F(x, y) \\ 0 \end{pmatrix}$  oder  $G(x, y) := \begin{pmatrix} F(x, y) \\ F(x, y) \end{pmatrix}$  zu setzen und damit das Newton-Verfahren zu verwenden.

Es wird daher folgende Strategie verwendet:

- Sei  $m := \begin{pmatrix} m_x \\ m_y \end{pmatrix} := \begin{pmatrix} (a+b)/2 \\ (c+d)/2 \end{pmatrix}$ . Berechne  $F(m)$ . Falls  $F(m) = 0$  sind wir fertig, falls  $F(m) < 0$  betrachte  $-F$ . Wir können also im folgenden annehmen  $F(m) > 0$ .
- Werte mithilfe geeigneter Schleifen  $F$  an verschiedenen  $(x, y) \in [a, b] \times [c, d]$  aus, bis  $(x, y)$  mit  $F(x, y) \leq 0$  gefunden wird. Tritt dies nicht ein, bricht der Algorithmus an der Stelle ohne Ergebnis ab. Ist  $F(x, y) = 0$  sind wir fertig. Wir können also im Folgenden annehmen, dass  $F(x, y) < 0$  ist.
- Sei nun  $\Psi : [0, 1] \rightarrow \mathbb{R}^2 : t \mapsto \begin{pmatrix} m_x \\ m_y \end{pmatrix} + t \begin{pmatrix} x - m_x \\ y - m_y \end{pmatrix}$ . Dann ist  $G := \Psi \circ F : [0, 1] \rightarrow \mathbb{R}$  stetig mit  $G(0) > 0$  und  $G(1) < 0$ . Mithilfe des Bisektionsverfahrens kann man eine Nullstelle  $t_0$  von  $G$  finden.
- Dann ist  $\Psi(t_0) \in [a, b] \times [c, d]$  eine Nullstelle von  $F$ .

Diese Strategie hat in unseren Tests immer die Nullstellen gefunden. Nullstellen die gleichzeitig Extremstellen der Funktion  $F$  sind, können damit nur durch großen Zufall gefunden werden, da die Funktion bei ihnen keinen Vorzeichenwechsel macht. Das ist kein großer Mangel, da diese Nullstellen aber uninteressant sind, denn dort ist  $\frac{\partial F}{\partial x} = 0$  und  $\frac{\partial F}{\partial y} = 0$ , was sie als Startwerte eher unbrauchbar macht.

## 6.2 Details der Implementierung

Es wurde also eine Funktionen der Art

`nivlines (F, dFx, dFy, Z, A, B, C, D, Steps, StepWidth)`

implementiert. Dabei sind:

- $Z$  ein Vektor ist, der die Funktionswerte enthält, zu denen Niveaulinien geplottet werden sollen. Bezeichne  $k$  im Folgenden die Länge von  $Z$ .
- $A, B, C, D$  jeweils Vektoren der Länge  $k$ , sodass ein Startwert für die Niveaulinie zu  $Z(j)$  im Intervall  $[A(j), B(j)] \times [C(j), D(j)]$  gesucht wird. Alternativ können auch Skalare übergeben werden, die wie Vektoren mit konstanten Einträgen behandelt werden.
- **Steps** und **StepWidth** sind ebenfalls Vektoren der Länge  $k$  oder Skalare, die die Schritztanzahl bzw. Schrittweite übergeben.

Die Implementierung der Funktion sieht dann im Wesentlichen (Assertions etc. wurden im

Listing weggelassen) so aus:

```

1 function [ X ,Y ] =nivlines4 (F, dFx, dFy, Z, A, B, C, D, Steps,
   StepWidth)
2
3 X = cell(k,1);
4 Y = cell(k,1);
5
6 X0=zeros(1,k);
7 Y0=zeros(1,k);
8
9 for j = 1:k
10     X{j}=zeros(Steps(j)+1,1);
11     Y{j}=zeros(Steps(j)+1,1);
12     Fj = @(x,y)F(x,y) - Z(j);
13     [X0(j),Y0(j),err]=findZero(Fj,A(j),B(j),C(j),D(j));
14
15     if err ~= 0 % kein Startwert gefunden
16         X{j}=zeros(0); %leerer Vektor, damit nichts geplottet
   wird
17         Y{j}=zeros(0);
18     else
19         [X{j},Y{j}] = implicitCurveXXX( Fj, dFx, dFy, X0(j), Y0(
   j), Steps(j), StepWidth(j) );
20     end
21 end
22 end

```

Listing 1: Ich bin ein Beispiel-Lisitng

Da Niveaulinien zu unterschiedlichen Funktionswerten sehr unterschiedlich lang sein können, ist es nicht sinnvoll, alle das die  $x$ - bzw.  $y$ -Werte der Punkte für die einzelnen Niveaulinien in Matrix  $X \in \mathbb{R}^{k \times \maxSteps}$  zu schreiben. Stattdessen bietet sich ein cell-Arrays an, der  $k$  Vektoren der Länge  $Steps$  enthält. Der Zugriff auf die einzelnen Vektoren erfolgt durch  $X\{j\}$ .

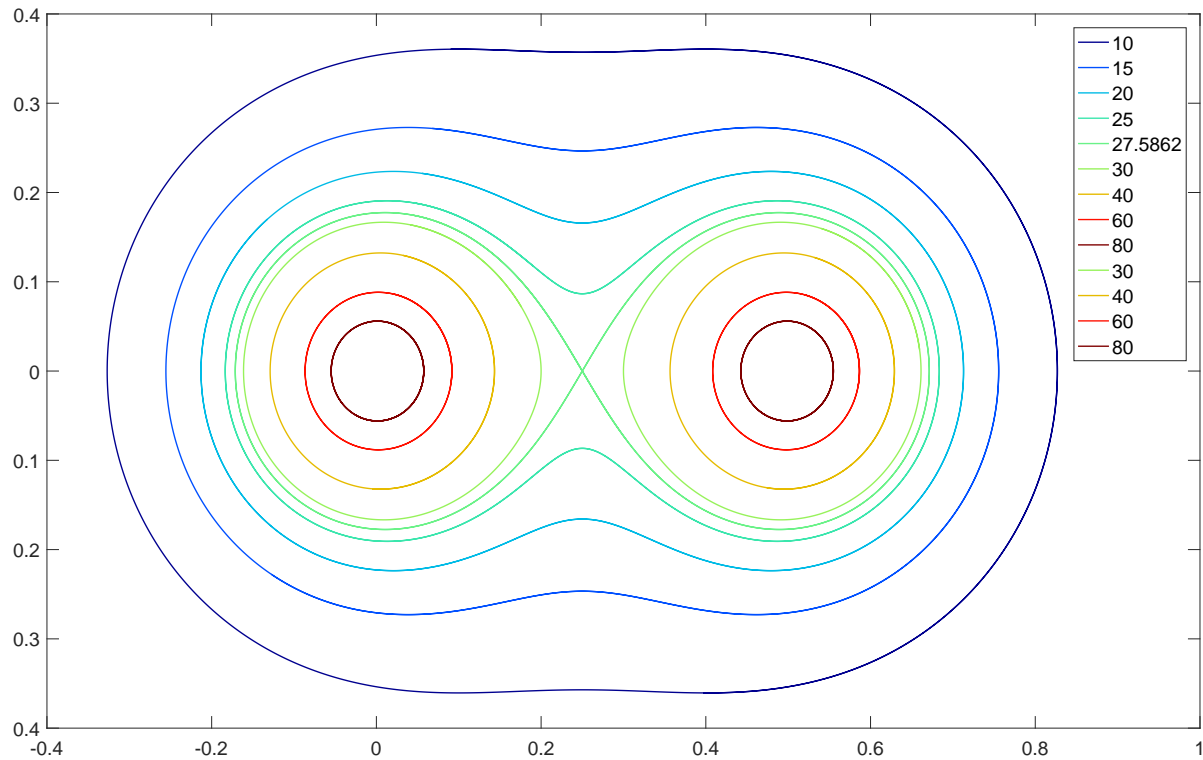
### 6.3 Tests

Sei

$$F(x, y) := \frac{1}{x^2 + y^2 + 10^{-2}} + \frac{1}{(x - 0.5)^2 + y^2 + 10^{-2}}$$

Sei  $Z := (10, 15, 20, 25, 800/29, 30, 40, 60, 80, 30, 40, 60, 80)$  der Vektor der Funktionswerte, für die Niveau-Linien geplottet werden sollen. Für alle Werte wurden Startpunkte im Intervall  $[0, 1/4] \times [0, 1]$  gesucht, für jene Werte, die im Vektor  $Z$  doppelt vorkommen, wurde zusätzlich im Intervall  $[1/2, 3/4] \times [0, 1]$  nach einem Startwert gesucht. Die Motivation für die Auswahl des Wertes  $800/29$  ist, dass  $F(1/4, 0) = 800/29$  und  $DF(1/4, 0) = (0, 0)$ .

Die Schrittweite betrug  $2 \cdot 10^{-3}$  die Schrittzahl 2000 für die ersten fünf Niveaulinien bzw. 500 für die Restlichen.



## 7 Anhang: Code-Listings

```

1 function [x0,y0,err] = findZero (F, a, b, c, d)
2 % finde (x0,y0) in [a,b]x[c,d] mit F(x0,y0)=0
3
4 mx = (a+b)/2;
5 my = (c+d)/2;
6
7 if isZero(F(mx,my))
8     x0y0 = [mx,my];
9 else
10     if F(mx,my) > 0
11         x0y0 = findZero2 (F,a,b,c,d);
12     else
13         x0y0 = findZero2 (@(x,y)-F(x,y),a,b,c,d);

```

```
14     end
15 end
16 x0=x0y0(1);
17 y0=x0y0(2);
18
19 if isZero(F(x0,y0))
20     err=0;
21 else
22     err=1;
23 end
24 end
25
26 function [X0Y0,err] = findZero2 (F, a, b, c, d)
27 % finde (x0,y0) in [a,b]x[c,d] mit F(x0,y0)=0
28 % fuer den Spezialfall F(mx,my) > 0
29
30 mx = (a+b)/2;
31 my = (c+d)/2;
32
33
34 %finde Funktionswert kleiner null, der nach mglichkeit nahe an
    mx,my ist
35 [x0y0,err]=findNegVal(F,a,b,c,d,4,20);
36
37 if err==1
38     [x0y0,err]=findNegVal(F,a,b,c,d,4,99); %99 statt 100 um
        andere Funktionswerte zu treffen
39 end
40 if err==1
41     X0Y0=[0,0];
42     return; %kein vorzeichen welchsel, also wird es nix
43 end
44
45
46
47 %transformiere auf Funktion F(Psi)=G: [0,1]-> R
48 Psi1= @(t) mx + t*(x0y0(1)-mx);
49 Psi2= @(t) my + t*(x0y0(2)-my);
50 G = @(t) F(Psi1(t),Psi2(t));
51
52
53 %finde Nullstelle von G in [0,1]
54 t0 = bisection(G,0,1);
55
56 %transfomiere Nullstelle in [0,1] zurck auf NSt in R^2
```



```
57 XOY0 = [mx+t0*(x0y0(1)-mx),my+t0*(x0y0(2)-my)];
58
59 end
60
61
62 function [x0y0, err] = findNegVal(F,a,b,c,d,k,n)
63 % n anzahl der einzelnen zerteilung
64 % k anzahl der intervallverkleinerungen
65
66 mx = (a+b)/2;
67 my = (c+d)/2;
68 err = 0;
69
70
71 for j=k:-1:1
72     [x0y0, err2] = findNegVal2(F,mx-(b-a)/2^j,mx+(b-a)/2^j,my-(d
73     -c)/2^j,my+(d-c)/2^j,n);
74     %suche_in = [[mx-(b-a)/2^j,mx+(b-a)/2^j],[my-(d-c)/2^j,my+(d
75     -c)/2^j]]
76     if err2==0
77         return;
78     end
79 end
80 %wenn man bis daher kommt wurde nix gefunden
81 warning('gar keine NSt gefunden');
82 err=1;
83 x0y0=[0,0];
84
85 end
86
87 function [x0y0,err]=findNegVal2(F,a,b,c,d,n)
88 % n gibt die Feinheit der Suche an: [a,b] resp [c,d] wird in ca
89     2n
90 %intervalle zerlegt
91
92 err=0;
93
94 mx = (a+b)/2;
95 my = (c+d)/2;
96
97 dx = (b-a)/(2*n);
98 dy = (d-c)/(2*n);
99
100 for j=-n:n
101     for k=-n:n
```

```
99         %[mx+j*dx,my+k*dy]
100         if F(mx+j*dx,my+k*dy) < 0
101             x0y0 = [mx+j*dx,my+k*dy];
102             return;
103         end
104     end
105 end
106
107 %wenn wir bis daher kommen waren wir erfolglos
108 x0y0=[0,0];
109 err=1;
110 %warning('jetzt keine NSt gefunden');
111
112 end
```

Listing 2: Implementierung der Nullstellensuche im  $\mathbb{R}^2$