NUMERIKPROJEKT

Titel
ggf. mehrzeilig

ausgeführt am

Institut für
Analysis und Scientific Computing
TU Wien

unter der Anleitung von

Prof. Dr. Lothar Nannen

durch

Lukas Moser
Matrikelnummer: 1607333
Stefan Schrott
Matrikelnummer: 1607388

Wien, am 21. Januar 2018

Inhaltsverzeichnis

1 Grundlagen

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

Satz: Seien $a < b$ sowie $c < d \in \mathbb{R}$ und $F : (a,b) \times (c,d) \to \mathbb{R}$ 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 $f : (a_0, b_0) \to \mathbb{R}$ mit $f(x_0) = y_0$, sodass

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

und

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

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 $f : U \to V$ 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 Vorraussetzungen 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 Aufgabe a

Das Ziel dieser Aufgabenstellung ist es den Graphen, der durch die Nullstellenmenge einer Funktion, numerisch anzunähern. Im ersten Schritt wollen wir dazu einem Gitter $x_j = x_0 + j * h, j \in \{1...n\}$ entlang x folgen. Dazu soll die Funktion F auf der gesamten betrachteten Menge die Bedingungen des Hauptsatzes über implizite Funktionen erfüllen und $F(x_0, y_0) = 0$ sein. Damit wissen wir, dass $\forall j \in \{1...n\} : F(x_j, f(x_j) = 0)$ gilt. Um $y_n = f(x_n)$ zu berechnen betrachte man mittels Mittelwertsatz

$$f(x_{n+1}) = f(x_n) + f'(x_n) * h + r_n, |r_n| \leq \sup_{a,b \in [x_n, x_{n+1}]} \left|f'(a) - f'(b)\right| * h.$$

Demnach ist für ausreichend kleine Schrittweite das Restglied r_n ausreichend klein, so dass die Nullstelle für $F(x_n + 1, .)$ mittels Newtonverfahren von $f(x_n) + f'(x_n) * h$ aus gefunden werden kann.

2.1 Tests

3 Implementierung von Aufgabe b Version 1

3.1 Tests

4 Implementierung von Aufgabe b Version 2

Um $y_n = f(x_n)$ zu berechnen betrachte man mittels Taylorformel.

$$f(x_{n+1}) = f(x_n) + f'(x_n) * h + r, |r| \leq \left| \sup_{x \in [x_n, x_{n+1}]} \frac{f''(x)}{2} \right| * h^2$$

4.1 Tests

5 Implementierung von adaptiver Schrittweite

Der Einfachheit halber werden mögliche Strategien für adaptive Schrittweite zuerst an der Implementierung aus Aufgabe a getestet, da das Koordinatensystem dort noch fest ist. Dann werden sie, falls möglich und sinnvoll, auf den allgemeinen Fall ausgeweitet.

5.1 Mögliche Strategien für Aufgabe a

Es ergeben sich folgende mögliche Strategien

1. Versuchen, die Krümmung von f aus den letzten Punkten zu schätzen, und die Schrittweite bei großer Krümmung zu reduzieren
2. Die Krümmung im letzten Punkt explizit berechnen und die Schrittweite daran anpassen
3. Die Differenz von Prediktor und Korrektor betrachten und bei größerer Differenz die Schrittweite reduzieren
4. Die Anzahl der Schritte bis das Newton-Verfahren konvergiert

5.2 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

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

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 $G : \mathbb{R}^n \to \mathbb{R}^n$ anwendbar ist. Wegen der Regularitätsforderung an die Jacobi-Matrix von G, ist es auch nicht möglich etwa $G(x, y) := \binom{F(x,y)}{0}$ oder $G(x, y) := \binom{F(x,y)}{F(x,y)}$ zu setzen und damit das Newton-Verfahren zu verwenden.

Es wird daher folgende Strategie verwendet:

- Sei $m := \binom{m_x}{m_y} := \binom{(a+b)/2}{(c+d)/2}$. 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 $\Psi : [0, 1] \to \mathbb{R}^2 : t \mapsto \binom{m_x}{m_y} + t\binom{x-m_x}{y-m_y}$. Dann ist $G := \Psi \circ F : [0, 1] \to \mathbb{R}$ 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 unterinteressant 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)`]×[`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 Schrittanzahl bzw. Schrittweite übergeben.

Die Implementierung der Funktion sieht dann im Wesentlichen (Assertions etc. wurden im Listing weggelassen) so aus:

```
function [ X ,Y ] =nivlines4 (F, dFx, dFy, Z, A, B, C, D, Steps,
    StepWidth)

X = cell(k,1);
Y = cell(k,1);

X0=zeros(1,k);
Y0=zeros(1,k);

for j = 1:k
```

```
    X{j}=zeros(Steps(j)+1,1);
    Y{j}=zeros(Steps(j)+1,1);
    Fj = @(x,y)F(x,y) - Z(j);
    [X0(j),Y0(j),err]=findZero(Fj,A(j),B(j),C(j),D(j));

    if err ~= 0 % kein Startwert gefunden
        X{j}=zeros(0); %leerer Vektor, damit nichts geplotet
  wird
        Y{j}=zeros(0);
    else
        [X{j},Y{j}] = implicitCurveXXX( Fj, dFx, dFy, X0(j), Y0(
  j), Steps(j), StepWidth(j) );
    end
end
end
```

Listing 1: Ich bin ein Beispiel-Lisitng

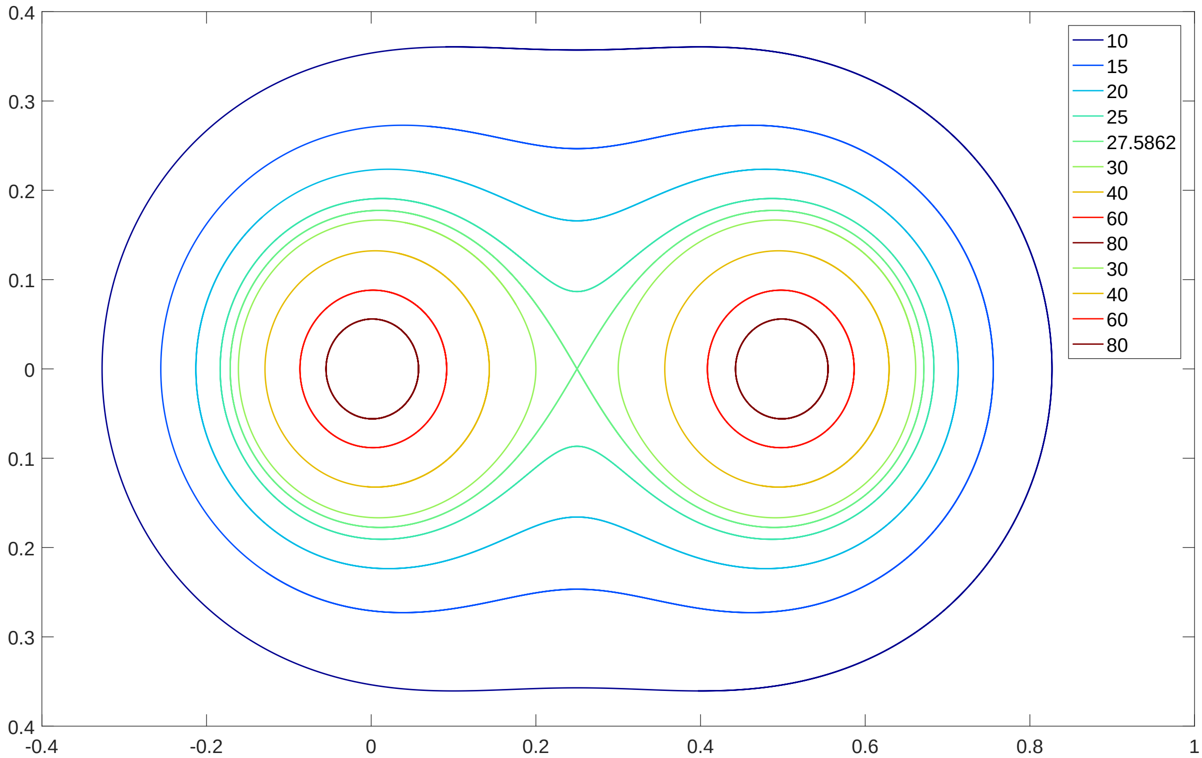
Da Niveaulinien zu unterschiedlichen Funktionswerten sehr unterschiedlich lang sein können, ist es nicht sinnvoll, alle das die x- bzw. y-Werte der Punkte für die einzelnen Niveaulinien in Matrix $X \in \mathbb{R}^{k \times maxSteps}$ zu schreiben. Stattdessen bietet sich ein cell-Arays an, der k Vektoren 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 Schrittanzahl 2000 für die ersten fünf Niveaulinien bzw. 500 für die Restlichen.



7 Anhang: Code-Listings

```
function [x0,y0,err] = findZero (F, a, b, c, d)
% finde (x0,y0) in [a,b]x[c,d] mit F(x0,y0)=0

mx = (a+b)/2;
my = (c+d)/2;

if isZero(F(mx,my))
    x0y0 = [mx,my];
else
    if F(mx,my) > 0
        x0y0 = findZero2 (F,a,b,c,d);
    else
        x0y0 = findZero2 (@(x,y)-F(x,y),a,b,c,d);
    end
end
x0=x0y0(1);
```

```
y0=x0y0(2);

if isZero(F(x0,y0))
    err=0;
else
    err=1;
end
end

function [X0Y0,err] = findZero2 (F, a, b, c, d)
% finde (x0,y0) in [a,b]x[c,d] mit F(x0,y0)=0
% fuer den Spezialfall F(mx,my) > 0

mx = (a+b)/2;
my = (c+d)/2;


%finde Funktionswert kleiner null
[x0y0,err]=findNegVal(F,a,b,c,d,4,20);

if err==1
    [x0y0,err]=findNegVal(F,a,b,c,d,4,99); %99 statt 100 um
   andere Funktionswerte zu treffen
end
if err==1
    X0Y0=[0,0];
    return; %kein vorzeichen welchsel, also wird es nix
end



%transformiere auf Funktion F(Psi)=G: [0,1]-> R
Psi1= @(t) mx + t*(x0y0(1)-mx);
Psi2= @(t) my + t*(x0y0(2)-my);
G = @(t) F(Psi1(t),Psi2(t));


%finde Nullstelle von G in [0,1]
t0 = bisection(G,0,1);

%transfomiere Nullstelle in [0,1] zurueck auf NSt in R^2
X0Y0 = [mx+t0*(x0y0(1)-mx),my+t0*(x0y0(2)-my)];

end

```

```
function  [x0y0, err] = findNegVal(F,a,b,c,d,k,n)
% n anzahl der einzelnen zerteilung
% k anzahl der intervallverkleinerungen

mx = (a+b)/2;
my = (c+d)/2;
err = 0;


for j=k:-1:1
    [x0y0, err2] = findNegVal2(F,mx-(b-a)/2^j,mx+(b-a)/2^j,my-(d-c)/2^j,my+(d-c)/2^j,n);
    %suche_in = [[mx-(b-a)/2^j,mx+(b-a)/2^j],[my-(d-c)/2^j,my+(d-c)/2^j]]
    if err2==0
        return;
    end
end
%wenn man bis daher kommt wurde nix gefunden
warning('gar keine NSt gefunden');
err=1;
x0y0=[0,0];

end

function [x0y0,err]=findNegVal2(F,a,b,c,d,n)
% n gibt die Feinheit der Suche an: [a,b] resp [c,d] wird in ca 2n
%intervalle zerlegt

err=0;

mx = (a+b)/2;
my = (c+d)/2;

dx = (b-a)/(2*n);
dy = (d-c)/(2*n);

for j=-n:n
    for k=-n:n
        %[mx+j*dx,my+k*dy]
        if F(mx+j*dx,my+k*dy) < 0
            x0y0 = [mx+j*dx,my+k*dy];
            return;
```

```
            end
        end
    end

    %wenn wir bis daher kommen waren wir erfolglos
    x0y0=[0,0];
    err=1;
    %warning('jetzt keine NSt gefunden');

end
```

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