NEWTONVERFAHREN

Nichtlineare Optimierung ohne Nebenbedingungen

Petkova Neli,Fallosch Nico,Kasic Ajla,Mittendorfer Patrick,Misurec Patrik

October 2015

Contents

1 Introduction

Das Newtonverfahren (auch Newton-Raphson gennant) ist ein Algorithmus zur Suche von Nullstellen. Es handelt sich dabei um ein Iterationsverfahren, welches sich pro Iteration der Nullstelle immer weiter annähert. Idealerweise wird ein Startpunkt bestimmt, der bereits möglichst nahe an der Nullstelle liegt. Die Funktion wird durch Ihre Tagente in diesem Punkt ersetzt, wobei der Schnittpunkt zwischen Tangente und x-Achse als neuer Ausgangspunkt für das Iterationsverfahren verwendet wird. Dieser Schritt wird analog solange durchgeführt, bis die gewünschte Rechengenauigkeit erreicht ist.

Durch Abwandlung der Formel $x_{k+1} = x_k - \frac{\mathrm{f}(x_k)}{\mathrm{f}'(x_k)}$ welche zur Bestimmung des Nullpunkts einer Funktion verwendet wird, lässt sich auch Minimumx bzw. Maximum der selben Funktion ermitteln. Da man hierbei eigentlich die Nullstelle der Ableitung sucht, ergibt sich für das Verfahren folgende Formel: $x_{k+1} = x_k - \frac{\mathrm{f}'(x_k)}{\mathrm{f}''(x_k)}$

Analog kann der Alogirthmus auch für mehrdimensionale Probleme angewandt werden. Die Formel hierfür lautet: $x_{k+1} = x_k - (H_f(x_k))^{-1}(\nabla F(x_k))'$

$x_{k+1} = x_k - \frac{\mathrm{f}(x_k)}{\mathrm{f}'(x_k)}$

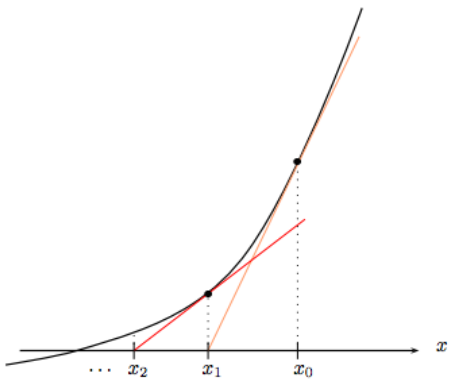
$x_{k+1} = x_k - \frac{\mathrm{f}'(x_k)}{\mathrm{f}''(x_k)}$

$\frac{\mathrm{f}'(x_k)}{\mathrm{f}''(x_k)} x_{k+1} = x_k - (H_f(x_k))^{-1}(\nabla F(x_k))'$

2 Theorie

2.1 Herleitung

2.1.1 Nullsetzen der Ableitung



Gegeben ist eine Funktion f(x). Angenommen $g(x) = f'(x)$.
Wir wählen einen beliebigen Startpunkt x_0 von g(x).

Dann legen wir die Tangente im Punkt x_0.
Allgemein lautet die Gleichung der Tangente y:

$y_T = m * x + b$

Die erste Ableitung einer Funktion (an Stelle x_0) gibt die Steigung der Tangente m an dieser Stelle an.

$\mathrm{m} = g'(x_0)$

m können wir gleich einsetzen. Wir erhalten folgende Gleichung:

$y_T = g'(x_0) * x + b$

Zur Berechnung von b benutzen wir die Eigenschaft der Tangente, dass y_T im Punkt x_0 gleich der Funktionswert im selben Punkt ist.

$g(x_0) = g'(x_0) * x_0 + b$

Für b erhalten wir:

$b = g'(x_0) * x_0 - g(x_0)$

Dann setzen wir den Wert von b in der Tangentengleichung:

$y_T = g'(x_0) * x + g'(x_0) * x_0 - g(x_0)$

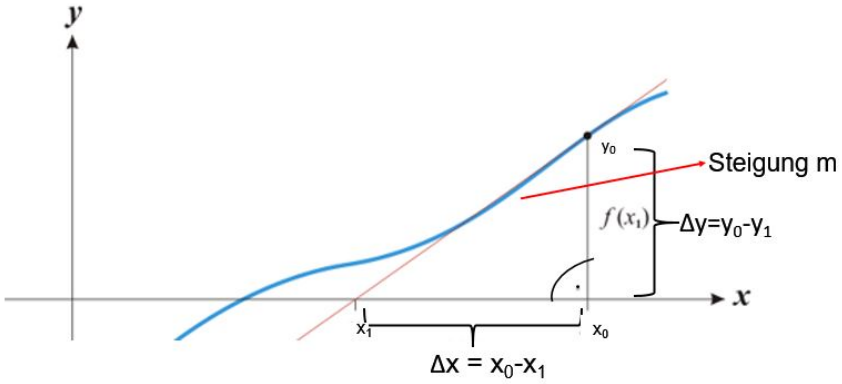
Wir klammern $g'(x_0)$ heraus und kriegen:

$y_T = g'(x_0) * (x - x_0) + g(x_0)$

Die Steigung der Tagente ist nichts anders als das Verhältnis der Differenzen der y- und x-Koordinaten von zwei zufällig ausgewählten Punkten von der Tangente. Wir nutzen aus, dass im Punkt x_0 der Tangentenwert gleich der Funktionswert ist.
(y-Koordinate von $x_0 = g'(x_0)$)
Als zweiten Punkt nehmen wir die Schnittstelle der Tangente mit der x-Achse x_1.
(y-Koordinate von $x_1 = 0$).

Für die Steigung der Tangente erhalten wir dann:

$m = \frac{\Delta g(x)}{\Delta x} = \frac{\mathrm{g}(x_0)-0}{(x_0-x_1)}$



$m = \frac{\Delta g(x)}{\Delta x} = \frac{\mathrm{g}(x_0)}{(x_0 - x_1)}$

Wie zuvor erwähnt, ist die Steigung der Tangente gleich der ersten Ableitung (m = $g'(x_0)$). Wir könnten die vorige Gleichung folgenderweise darstellen:

$g'(x_0) = \frac{\mathrm{g}(x_0)}{(x_0 - x_1)}$

Danach multipliziert man beide Seiten der Gleichung mit $(x_0 - x_1)$.

$g'(x_0) * (x_0 - x_1) = g(x_0)$

$x_0 - x_1 = \frac{\mathrm{g}(x_0)}{\mathrm{g}'(x_0)}$

Daher kommt raus:

$-x_1 = -x_0 + \frac{\mathrm{g}(x_0)}{\mathrm{g}'(x_0)}$

Man multipliziert die Gleichung mit (-1). Das ergibt:

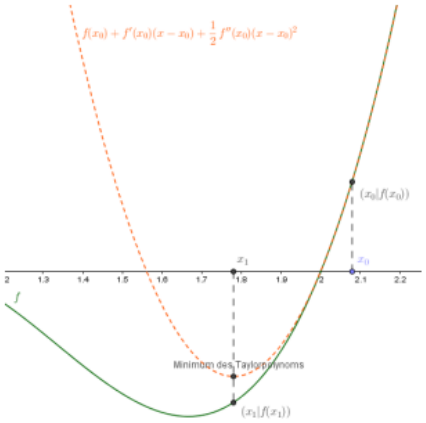
$x_1 = x_0 - \frac{\mathrm{g}(x_0)}{\mathrm{g}'(x_0)}$

Analog, lässt sich die Iterationsformel auf :

$x_{k+1} = x_k - \frac{\mathrm{g}(x_k)}{\mathrm{g}'(x_k)}$ herleiten. Bzw. lösen wir $g(x) = f'(x)$ auf.

$x_{k+1} = x_k - \frac{\mathrm{f}'(x_k)}{\mathrm{f}''(x_k)}$

2.1.2 Approximation durch quadratisches Polynom



Gegeben ist eine Funktion f(x).Das approximierende Polynom sei das Taylorpolynom 2.Grades. Die Taylorformel für ein Polynom 2.Grades lautet allgemein:

$$f(x) = f(x_0) + \frac{\mathrm{f}'(x_0)}{1!} * (x_1 - x_0) + \frac{\mathrm{f}''(x_0)}{2!} * (x_1 - x_0)^2$$

$$f(x) = f(x_0) + f'(x_0) * (x_1 - x_0) + \frac{\mathrm{f}''(x_0)}{2} * (x_1 - x_0)^2$$

Das lässt sich allgemein wie gefolgt darstellen:

$$f(x_k) = a * {x_k}^2 + b * x_k + c$$

Erste Ableitung $f'(x_k)$:

$$f'(x_k) = 2a * x_k + b$$

Zweite Ableitung $f''(x_k)$:

$$f''(x_k) = 2a$$

Daher erhalten wir für a und b:

$2a = f''(x_k)$ – von der ersten Ableitung

$b = -2a * x_k + f'(x_k)$ – von der zweiten Ableitung

Wir könnten a in $-2a * x$ wohl mit $f''(x)$ einsetzen:

$$b = -f''(x_k) * x_k + f'(x_k)$$

Den nächsten Punkt x_{k+1}, dass wir für die Iteration benutzen werden, ist das Minimum von $f(x)$. Extrempunkte berechnet man durch Setzten der ersten Ableitung auf 0. Von dem Vorzeichen der zweiten Ableitung lässt sich erkennen, ob der gefundene Extremum ein Minimum, ein Maximum (wenn die zweite Ableitung positiv ist) oder ein Wendepunkt (wenn $f''(x) = 0$ oder die Funktion nicht zweimal differenzierbar ist) ist. In unserem Fall sollte die zweite Ableitung negativ sein.

$f'(x_{k+1}) = 2a * x_{k+1} + b = 0$

$x_{k+1} = \frac{-b}{2a}$

Nun setzen wir a und b in x ein

$x_{k+1} = \frac{-b}{2a}$

$x_{k+1} = \frac{-(-f''(x_k)*x_k+f'(x_k))}{2*\frac{f''(x_k)}{2}} = \frac{f''(x_k)*x_k-f'(x_k)}{f''(x_k)} = x_k - \frac{f'(x_k)}{f''(x_k)}$

Nun kann man rekursiv fortsetzen:

$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$

2.2 Anwendungsbeispiel

Bei unserer Recherche konnten wir festellen, dass das Newtonverfahren meist nur in Kombination mit anderen Optimierungsverfahren eingesetzt wird. Begründet wird dies durch die bekannten Schwächen, welche das Newtonverfahren aufweist.

Angenommen (fiktives Beispiel!) ein Krankenhaus weißt folgenden funktionalen Zusammenhang zwischen der Einahme von Medikamten x und der Sterbewahrscheinlichkeit y auf:

$f(x) = x^5 - 3 \times x^3 + x^2 + 5$
$f'(x) = 5x^4 - 9x^2 + 2x$
$f''(x) = 20x^3 - 18x + 2$

Bei Anwendung des Newtonverfahrens mit dem Startwert $x = 2$ erhalten wir nach einigen Iterationsschritten das Optimum (Minimum) an der Stelle $x = 1.21$ mit dem Wert $y = 3.74$. Somit ist es möglich, die minimale Sterbewahrscheinlichkeit in Abhänigkeit der Medikamente zu berechnen.

n	x^n
1	2
2	1,619047619
3	1,376525059
4	1,252276875
5	1,215608655
6	1,212496881
7	1,21247531
8	1,212475309
9	1,212475309

2.3 Weiteres Beispiel

Wir möchten wieder überprüfen, ob wir mittels Newtonverfahren das Minimum der Funktion $f(x) = x^5 - 8x^2 - 1$ finden.

Die erste und zweite Ableitung der Funktion lauten:

$f'(x) = 5x^4 - 16x$
$f''(x) = 20x^3 - 16$

Als Startpunkt wählen wir $x_0 = 2$.
Dann setzen wir iterativ in der Formel ein:

n	x^n
1	2
2	1,666666667
3	1,511121857
4	1,475400801
5	1,473616925
6	1,473612599
7	1,473612599

Bei der 7 Iteration erkennen wir, das die Rechengenauigkeit = 0 ist, was heißt, dass wir das Minimum der Funktion gefunden haben.

2.4 Problemfall

Gegeben sei die Funktion $f(x) = \frac{x^4}{4} - x^2 + 2x$ und der Startpunkt $x_0 = 0$. Bei Anwendung des Newtonverfahrens zur Bestimmung des Minimus der Funktion ergeben sich folgende Iterationsschritte:

$f(x) = \frac{x^4}{4} - x^2 + 2x$
$f'(x) = x^3 - 2x + 2$
$f''(x) = 3x^2 - 2$

$x_0 = 0$

$x_1 = 0 - \frac{0^3 - 2 \times 0 + 2}{3 \times 0^2 - 2} = 1$

$x_2 = 1 - \frac{1^3 - 2 \times 1 + 2}{3 \times 1^2 - 2} = 0$

$x_3 = 0 - \frac{0^3 - 2 \times 0 + 2}{3 \times 0^2 - 2} = 1$

$x_4 = 1 - \frac{1^3 - 2 \times 1 + 2}{3 \times 1^2 - 2} = 0$

Wie man hier feststellen kann, alternieren die Werte zwischen 0 und 1, wodurch das Verfahren nicht zum Minimum der Funktion konvergiert. Das nennt man Osszilation. Das Problem hier liegt darin, dass der Startpunkt schlecht ausgewählt worden ist.

2.5 Gedämpftes Newtonverfahren

Beim gedämpften Newtonverfahren wird die Iterationsformel um einen Dämpfungsparameter λ erweitert, um so die Weite des Intervalles verändern zu können.

$$x_{k+1} = x_k - \lambda \times \frac{f'(x_k)}{f''(x_k)}$$

Beim Versuch die Funktion aus dem vorherigen Kapitel mit Hilfe des gedämpften Newtonverfahren unter Zuhilfename eines selbst geschrieben C++ Programmes zu lösen, ergaben sich folgende Ergebnisse:

$\lambda = 0.5, x_0 = 0 \rightarrow 64$ Schritte

$\lambda = 0.25, x_0 = 0 \rightarrow 245$ Schritte

$\lambda = 0.25, x_0 = 0 \rightarrow 47$ Schritte

$\lambda = 0.95, x_0 = 0 \rightarrow 39$ Schritte

$\lambda = 0.99, x_0 = 0 \rightarrow \infty$ Schritte

Daraus lässt sich schließen, dass die Gleichung unter Verwendung eines geeigneten λ und der Bedingung $x_0 = 0$ mittles gedämpften Newtonverfahrens lösbar ist. Beim Versuch λ nahe an eins zu wählen, konnten wir keine Ergebnisse erhalten.

3 Mehrdimensioneles Newtonverfahren

3.1 Herleitung

Das Newtonverfahren könnte ebenfalls auf mehrdimensionale Funktionen angewandt werden. Das mehrdimensionale Newtonverfahren setzt auch stetig differenzierbare (zweimal differenzierbare) Funktionen voraus. Darüber hinaus, damit Konvexität der Funktion sichergestellt werden kann, sollte die Hesse'sche Matrix immer positiv-definit.
Das Vorgehensweise ist wieder dasgleiche: Die mehrdimensionale Funktion wird duch ein quadratisches Polynom approximiert. Dann wird das Minimum der quadratischen Funktion gesucht und als nächsten Iterationswert gesetzt. Es wird so weit mit der Rekursion fortgegangen, bis die gewünsche Rechengenauigkeit erzielt wird.

Die Iterationsformel für das mehrdimensionale Newtonverfahren ist analag zu der für das eindimensionale.

$x_{k+1} = x_k - (H_F(x_k))^{-1}(\nabla F(x_k))'$

Die quadratische Funktion sieht im Allgemein so aus:

$h(x) = a + b'x + \frac{1}{2}x'Gx$

Die erste Ableitung oder der Gradientenvektor:

$\nabla h(x) = b' + x'G$

Die zweite Ableitung oder die Hesse'sche Matrix von $h(x)$:

$H_h(x) = G$

Die quadratische Funktion wird im Punkt x_k angenährt. Das heißt, dass der Funktionswert unserer mehrdimensionalen Funktion und der Funktionswert des quadratischen

Polynoms im Punkt x_k gleich sind. Das gilt natürlich auch für die Gradienten und die Hesse'schen Matrizen von beiden Funktionen im selben Punkt.

$F(x_k) = h(x_k)$
$\nabla F(x_k) = \nabla h(x_k)$
$H_F(x_k) = H_h(x_k)$

Das können wir ausnutzen und und in der obigen Gleichung $h(x_k)$ mit unserer Funktion $F(x_k)$ ersetzen:

$F(x_k) = a + b'x_k + \frac{1}{2}x_k' G x_k$
$\nabla F(x_k) = b' + x_k' G$
$H_F(x_k) = G$

G müssen wir gar nicht umformen, weil es gleich erkennbar ist, dass es gleich der Hesse'sche Matrix ist.
Wir brauchen nur noch b berechnen:

$$b' = \nabla F(x_k) - x_k' G'$$

Wir setzten gleich G mit $H_F(x_k)$ ein;

$$b' = \nabla F(x_k) - x_k' H_F(x_k)'$$

Da die Hesse'sche Matrix symmetrisch ist, wenn wir b' transponieren, bleibt $H_F(x_k)$ unverändert :

$$b = (\nabla F(x_k))' - H_F(x_k)x_k$$

Schließlich, um das Minimum x_{k+1} zu berechnen, müssen wir der Gradientvektor auf 0 setzen:

$$\nabla h(x_k) = 0$$

$$b' + x_{k+1}' G = 0$$

Daher erhalten wir für x_{k+1}:

$$x_{k+1} = -G^{-1}b$$

b setzen wir von oben ein:

$$x_{k+1} = -(H_F(x_k))^{-1}((\nabla F(x_k))' - H_F(x_k)x_k)$$

Wir kürzen den Bruch durch $H_F(x_k)$ und haben wir die Iterationsformel hergeleitet:

$x_{k+1} = x_k - (H_F(x_k))^{-1}(\nabla F(x_k))'$

3.2 Beispiel

Wir möchten die Funktion $F(x, y)$ minimieren.

$F(x, y) = 2x^2 + 6y^2 - 6x - 2y$

Als Startvektor wählen wir $(0, 0)'$ aus.

Wir berechnen zunächst mal die erste Ableitung unserer Funktion:

$\nabla F(x, y) = (4x - 6, 12y - 2)$
$H_F(x, y) = \begin{bmatrix} 4 & 0 \\ 0 & 12 \end{bmatrix}$
$H_F(x, y)^{-1} = \begin{bmatrix} \frac{1}{4} & 0 \\ 0 & \frac{1}{12} \end{bmatrix}$

$\surd$ Hesse'sche Matrix bekannt
$\surd$ Hesse'sche Matrix positiv definit => Funktion strikt konvex

Nun setzen wir rekursiv in der Iterationsformel ein:

$\begin{pmatrix} x_0 \\ y_0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$
$\begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} - \begin{bmatrix} \frac{1}{4} & 0 \\ 0 & \frac{1}{12} \end{bmatrix} * \begin{pmatrix} -6 \\ -2 \end{pmatrix} = \begin{pmatrix} \frac{3}{2} \\ \frac{1}{6} \end{pmatrix}$
$\begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = \begin{pmatrix} \frac{3}{2} \\ \frac{1}{6} \end{pmatrix} - \begin{bmatrix} \frac{1}{4} & 0 \\ 0 & \frac{1}{12} \end{bmatrix} * \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{3}{2} \\ \frac{1}{6} \end{pmatrix}$
Wir erkennen, dass $\begin{pmatrix} x_2 \\ y_2 \end{pmatrix}$ gleich zu $\begin{pmatrix} x_1 \\ y_1 \end{pmatrix}$ ist, und hören mit der Rekursion auf.
Zum Glück haben wir das Minimum $(\frac{3}{2}, \frac{1}{6})'$ erst bei der 2. Iteration gefunden.

4 Implementation

4.1 Tool für Problemfall

Um die in Kapitel 2.3 beschriebene "Problemgleichung" $f(x) = \frac{x^4}{4} - x^2 + 2x$ mit Startpunkt $x_0 = 0$ besser analysieren zu können, haben wir ein Testtool in C++ geschrieben. Das Tool wendet das gedämpfte Newtonverfahren an. Der Benutzer hat weiters die

Möglichkeit, den Startpunkt, λ und die maximale Anzahl an Iterationsschritten zu wählen. Bei Verwendung von $\lambda = 1$ entspricht das Verfahren dem regulären Newtonverfahren.

```
#include "stdafx.h"
#include <math.h>
#include <iostream>

double point = 0;
double lambda = 0.5;
int maxRuns = 500;

int main()
{
   //Calculator: f(x) = x^4/4-x+2x
   //öLsung mit ägedmpften Newtonverfahren

   //Info:
   //Wenn Lambda in starker äNhe von 1 ist, tritt ein endloser Zyklus ein↩
        vermeidet die Konvergenz


   char c;
   std::cout << "y for default values / n to input values" << std::endl;
   std::cin >> c;

   if (c == n )
   {
      std::cout << "enter startpoint (double): ";
      std::cin >> point;

      std::cout << std::endl << "enter lambda (double): ";
      std::cin >> lambda;

      std::cout << std::endl << "enter maxRuns int: ";
      std::cin >> maxRuns;

      std::cout << std::endl << std::endl;

   }
   for (int i = 0; i < maxRuns; i++)
   {
      if (point == point - lambda * ((pow(point, 3) - 2 * point + 2) / (3↩
          * (pow(point, 2)) - 2)))
      {
         std::cout << "Done after: " << i+1 << " runs" << std::endl;
         break;
      }
      point = point - lambda * ((pow(point, 3) - 2 * point + 2) / (3 * (↩
          pow(point, 2)) - 2));
      std::cout << point << std::endl;
   }
```

```
46     std::cin >> point;
47      return 0;
48 }
```

5 Homepage

Unsere HomePage ist unter http://ops-newton.github.io/Page aufrufbar.