Automatisches Differenzieren zum Lösen nichtlinearer Gleichungssysteme in der Ausgleichungsrechnung

Jens Kersten und Christian Clemen

Zusammenfassung

Die Bestimmung von partiellen Ableitungen ist integraler Arbeitsschritt der Ausgleichungsrechnung. Die Anwendung der hier behandelten Differentiationsarithmetik ermöglicht eine exakte und automatische Bestimmung von Werten partieller Ableitungen beliebiger Ordnung simultan zur Berechnung von Funktionswerten ohne die analytische Form dieser zu kennen. Der vorliegende Beitrag behandelt die Anwendung des automatischen Differenzierens (AD) für Aufgaben der Ausgleichungsrechnung im Gauß-Markov-Modell und wendet sich vor allem an Personen, die Software für die Lösung von Ausgleichungsproblemen entwickeln. Der Vorteil von AD ist, dass sich der Programmieraufwand auf die Implementierung des funktionalen Modells beschränkt, da die Linearisierung durch eine eigenständige Algebra automatisch realisiert wird. Dadurch wird die Softwareentwicklung schneller und robuster.
Als Alternative zum bekannten Gauß-Newton-Verfahren lässt sich das Newton-Raphson-Verfahren (NR) mit AD einfach implementieren und anwenden. Es wird gezeigt, wie mittels NR die Anwendung robuster M-Schätzer ohne das übliche Verfahren der »Regewichtung« erfolgen kann. Eine simulierte Ausgleichungsrechnung mit drei robusten Schätzern demonstriert Relevanz und Mehrwert des Einsatzes von AD.

Summary

The computation of partial derivatives is an integral step in adjustment calculus. The application of the here discussed differential arithmetic enables the exact and automated determination of the values of partial derivatives of arbitrary order. This is done simultaneously during the computation of function values without knowledge about the corresponding equations. This article focuses on the application of automatic differentiation (AD) in the context of adjustment calculus in Gauss-Markov-models and therefore addresses to software engineers. The advantage of AD is the restriction of programming efforts to the implementation of the functional model, since linearization is done using an independent algebra. Hence, software engineering gets faster and more robust.
The Newton-Raphson method (NR) is an alternative to the well-known Gauss-Newton method and can easily be implemented and applied using AD. It is shown how NR can be used to apply robust M-estimators without the common approach of »re-weighting«. A simulated example estimation with three robust estimators demonstrates the relevance and value of AD.

Schlüsselwörter: Automatisches Differenzieren, Ausgleichungsrechnung, Newton-Raphson-Verfahren, M-Schätzer

1 Einleitung

Das Differenzieren mathematischer Funktionen ist Handwerkszeug. Es wird beispielsweise für die Lösung überbestimmter Gleichungssysteme mit dem bekannten Gauß-Newton-Verfahren angewendet. Die Berechnung der Ableitung $f'(x)$ einer Funktion $f(x)$ an der Stelle x erfolgt üblicherweise entweder auf analytischem oder numerischem Weg. Die Bestimmung des symbolischen Ausdrucks einer analytischen Ableitung – entweder manuell oder mit Software – ist mitunter sehr komplex und zeitaufwändig. Die gewonnenen symbolischen Ausdrücke werden anschließend üblicherweise in Programmquellcode überführt, was fehleranfällig ist und eine aufwändige Kontrolle der Ergebnisse (z. B. durch Vergleich mit der numerischen Ableitung) erforderlich macht. Das numerische Ableiten beruht dagegen auf dem Differenzenquotienten

$$f'(x) = lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}, \tag{1}$$

wobei das Intervall Δx zu definieren ist und starken Einfluss auf die Güte des Ergebnisses hat. Rundungsfehler sind hier aufgrund der begrenzten Rechengenauigkeit sowie des Verfahrensfehlers üblich. Einerseits kann Δx nicht beliebig klein, andererseits darf es auch nicht zu groß gewählt werden, da die daraus resultierende Sekante der Funktion wohlmöglich stark von der eigentlich gesuchten Tangente abweicht.

Automatisches Differenzieren (AD) – auch Algorithmisches Differenzieren – ist dagegen nicht sehr verbreitet, wenngleich es auf einfachen algebraischen Regeln und noch einfacheren Implementierungen beruht. AD ermöglicht den gänzlichen Verzicht der Bestimmung und Codierung von Ableitungsformeln bei gleichzeitiger Vermeidung von Rundungsfehlern. Die Bestimmung partieller Ableitungen $f^i(x)$ beliebiger Ordnung i erfolgt hier exakt, automatisch und simultan zu der Berechnung der Funktionswerte $f(x)$.

Inspiriert von verschiedenen Anwendungsbeispielen (z. B. in Brückner et al. 2006) sowie dem Vorschlag von Jerell (1989), Minimierungsaufgaben unter Einsatz von AD zu lösen, haben die Autoren in den Studien (Kersten 2008) sowie (Clemen 2010) untersucht, wie AD gewinnbringend für Aufgaben der Ausgleichungsrechnung eingesetzt werden kann. Im vorliegenden Beitrag werden die Ergebnisse dieser Arbeiten zusammengefasst. Insbesondere wird der Einsatz von AD für Minimierungsprobleme mit dem Newton-Raphson-Verfahren als Alternative zum bekannten Gauß-Newton-Verfahren behandelt. Weiter-

hin wird gezeigt, wie mittels AD robuste Schätzmethoden einfach umgesetzt werden können.

Der Artikel ist wie folgt gegliedert: In Abschnitt 2 werden die mathematischen Grundlagen der Differentiationsarithmetik behandelt. In Abschnitt 3 werden Aspekte der Implementierung von AD in C++ erläutert. In Abschnitt 4 wird der Einsatz von AD für Aufgaben der Ausgleichungsrechnung – insbesondere für robuste M-Schätzer – beschrieben. In Abschnitt 5 wird die vorgestellte Methodik anhand eines Beispiels demonstriert. In Abschnitt 6 werden eine Zusammenfassung und ein Ausblick auf mögliche künftige Arbeiten gegeben.

2 Mathematische Grundlagen

Die Differentiationsarithmetik (Rall 1986) vereint sowohl den Wert einer Funktion $f(x)$, als auch die dazugehörige Ableitung $f'(x)$ in einer Variable. Dazu können, äquivalent zu den komplexen Zahlen, geordnete Paare von Zahlen als Grundelement definiert werden

$$U=(u,u'),V=(v,v'),U,V\in \mathrm{R}^2. \tag{2}$$

Basierend auf den bekannten Regeln des analytischen Differenzierens können für diese Elemente beispielsweise folgende Regeln für mathematische Elementaroperationen definiert werden:

1. Addition: $U+V=(u,u')+(v,v')=(u+v,u'+v')$
2. Subtraktion: $U-V=(u,u')-(v,v')=(u-v,u'-v')$
3. Multiplikation: $U\cdot V=(u,u')\cdot(v,v')=(u\cdot v,u'\cdot v+u\cdot v')$
4. Division:
$$U\cdot V=(u,u')/(v,v')=\left(\frac{u}{v},(u'\cdot v-u\cdot v')/v^2\right). \tag{3}$$

Die Herleitung neuer Regeln erfolgt durch Einführen einer »infinitesimalen Einheit« d, für die $d^2=0$ gilt, sodass $U=u+u'd$. Für die Multiplikation folgt damit beispielsweise durch Ausmultiplizieren:

$$\begin{aligned}U\cdot V&=(u+u'd)\cdot(v+v'd)\\&=uv+uv'd+u'dv+d^2u'v'\\&=uv+(uv'+u'v)d.\end{aligned} \tag{4}$$

Die erste Komponente uv im Ergebnis von (4) resultiert dabei aus den Regeln der reellen Arithmetik und die zweite Komponente resultiert aus den dazu korrespondierenden Ableitungsregeln.

Ein Zahlenpaar $U=(u,u')\in \mathrm{R}^2$ wird Element der Menge D genannt, wobei das reelle Zahlensystem R in D enthalten ist: $c=(c,0)$, $C\in \mathrm{R}$. Eine Funktion $f:\mathrm{R}\to\mathrm{R}$ kann mit der Differentiationsarithmetik zu einer Funktion $f:\mathrm{D}\to\mathrm{D}$ erweitert werden. Dazu muss lediglich jeder Wert x durch $X=(x,1)$ und jede Konstante c durch $C=(c,0)$ ersetzt werden, was den allgemeinen Regeln

$$\frac{d}{dx}x=1,\quad \frac{d}{dx}c=0 \tag{5}$$

der Differenzierung von Variablen und Konstanten entspricht.

Beispielhaft sei die Funktion

$$f(x)=\frac{x^2+2x-3}{x+2}=\frac{(x-1)\cdot(x+3)}{x+2} \tag{6}$$

angeführt (Rall 1986). Der Ausdruck für die entsprechende Ableitung ist gegeben mit

$$f'(x)=1+\frac{3}{(x+2)^2}, \tag{7}$$

womit sich für $x=3$ die folgenden Werte ergeben

$$f(3)=\frac{12}{5},\ f'(3)=1+\frac{3}{25}=\frac{28}{25}. \tag{8}$$

Wird in die Funktion nun $X=(3,1)$ und für die Konstanten $C=(c,0)$ eingesetzt, erhält man durch Anwendung von (3)

$$\begin{aligned}f((3,1))&=\frac{((3,1)-(1,0))\cdot((3,1)+(3,0))}{((3,1)+(2,0))}\\&=\frac{(12,8)}{(5,1)}=\left(\frac{12}{5},\frac{28}{25}\right).\end{aligned} \tag{9}$$

Der Funktionswert und die Ableitung sind direkt berechnet worden. Ein analytischer Ausdruck für $f'(x)$ wurde dabei nicht benötigt.

Der Schlüssel für die Verwendung von AD für beliebig komplexe Funktionen in Computerprogrammen liegt bei der Anwendung der Kettenregel, nach der für eine Verkettung $f(x)=u(v(x))$ die Ableitung durch das Produkt der äußeren Ableitung u' und der inneren Ableitung v', also durch $f'(x)=u'(v(x))\cdot v'(x)$, gebildet wird. Diese Regel kann rein mechanisch auf beliebige differenzierbare Funktionen angewendet werden. Zum Beispiel:

$$sin(U)=sin((u,u'))=(sin(u),u'\cos(u)). \tag{10}$$

Da jeder differenzierbare Formelausdruck eine Komposition von arithmetischen Operatoren und elementaren Funktionen ist, ist eine Zerlegung in eine Sequenz von elementaren Ausdrücken immer möglich.

Eine Erweiterung auf Funktionen, die von n Variablen abhängig sind, ist möglich, wobei der Infinitesimalteil dann ein n-dimensionaler Vektor ist. Darüber hinaus können auf diese Weise Ableitungen beliebiger Ordnung automatisch berechnet werden.

3 Realisierung von AD in C++ Computerprogrammen

Mathematische Funktionen beliebiger Komplexität werden in Computerprogrammen in eine Sequenz von Elementaroperationen $\{f_i\}_{i=1,\ldots,n}$ zerlegt, wobei f_i jeweils von den Zwischenergebnissen $\tau_1,\ldots,\tau_{i-1}$ abhängt, sodass $\tau_i = f_i(\tau_1,\ldots,\tau_{i-1})$ gilt. Als Beispiel sei eine Funktion $y = f(x_1,x_2)$ gegeben (Griewank 2000):

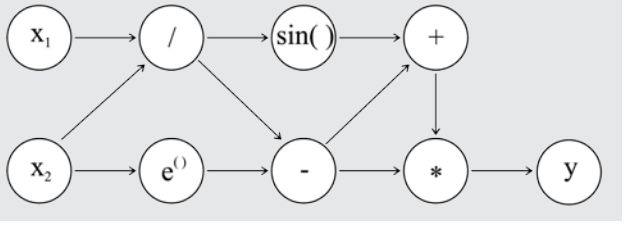
$$y = \left[sin\left(\frac{x_1}{x_2}\right) + \frac{x_1}{x_2} - e^{x_2} \right] \cdot \left[\frac{x_1}{x_2} - e^{x_2} \right], \qquad (11)$$

wobei $\tau_{-1} = x_1 = 1{,}5$ und $\tau_0 = x_2 = 0{,}5$. Für die Berechnung von y wird der Ausdruck (11) beim Berechnungsablauf in eine Sequenz (Tab. 1) zerlegt. Diese Sequenz kann durch einen Berechnungsgraphen *(computational graph)* dargestellt werden (Abb. 1). Um redundante Knoten im Graphen zu vermeiden, darf jede Variable nur einmal deklariert werden. Durch diese Zerlegung ist eine Form erreicht, die den mechanischen Einsatz der Kettenregel in Computerprogrammen begünstigt, da hierzu lediglich alle Variablen $x_i \in \mathrm{R}$ durch Zahlentupel $U_i = (u,u') \in \mathrm{R}^2$ ersetzt werden müssen. Die simultane Berechnung von Funktionswerten sowie von partiellen Ableitungen unter Anwendung der Kettenregel wird prinzipiell auf zweierlei Arten umgesetzt:

1. *Forward*-Methode: Für jede Variable werden entlang der Berechnungssequenz simultan zu den Zwischenergebnissen auch die partiellen Ableitungen nach den gewünschten Variablen berechnet und mitgeführt.
2. *Backward*-Methode: Bei der initialen Berechnung des Funktionswertes wird der Berechnungsablauf gespeichert und anschließend wird der Berechnungsgraph rückwärts traversiert.

Tab. 1: Berechnungssequenz für Gleichung (11)

τ_{-1}	=	x_1	=	1,5000		
τ_0	=	x_2	=	0,5000		
τ_1	=	τ_{-1}/τ_0	=	1,5000 / 0,5000	=	3,0000
τ_2	=	$\sin(\tau_1)$	=	sin (3,0000)	=	0,1411
τ_3	=	e^{τ_0}	=	$e^{0{,}5000}$	=	1,6487
τ_4	=	$\tau_1 - \tau_3$	=	3,0000 – 1,6487	=	1,3513
τ_5	=	$\tau_2 + \tau_4$	=	0,1411 – 1,3513	=	1,4924
τ_6	=	$\tau_5 \cdot \tau_4$	=	1,4924 · 1,3513	=	2,0167
y	=	τ_6	=	2,0167		



Kersten 2008

Abb. 1: Berechnungsgraph für Gleichung (11)

Anhand der Ableitung $\partial y / \partial x_1$ der Funktion (11) wird im Folgenden die Forward-Methode erläutert. In jedem Schritt der Funktionsberechnung in Tab. 1 wird für jeden Wert τ_i simultan der Wert der Ableitung $\dot{\tau}_i = \partial \tau_i / \partial x_1$ mitgeführt[1]. Da x_1 eine unabhängige Variable ist und x_2 als konstant angesehen wird, folgt für die Initialisierung $\dot{x}_1 = \dot{\tau}_{-1} = 1{,}0$ und $\dot{x}_2 = \dot{\tau}_0 = 0{,}0$. Durch Anwendung der Kettenregel ergibt sich zu jedem τ_i ein korrespondierender Wert $\dot{\tau}_i = \partial \tau_i / \partial x_1$, wie beispielsweise für $\tau_1 = \tau_{-1} / \tau_0$:

$$\begin{aligned} \dot{\tau}_1 &= \left(\frac{\partial \tau_1}{\partial \tau_{-1}}\right) \cdot \dot{\tau}_{-1} + \left(\frac{\partial \tau_1}{\partial \tau_0}\right) \cdot \dot{\tau}_0 \\ &= \frac{1}{\tau_0} \cdot \dot{\tau}_{-1} - \left(\frac{\tau_{-1}}{\tau_0}\right) \cdot \frac{1}{\tau_0} \cdot \dot{\tau}_0 = (\dot{\tau}_{-1} - \tau_1 \cdot \dot{\tau}_0) / \tau_0 \\ &= (1{,}0 - 3{,}0 \cdot 0{,}0) / 0{,}5 = 2{,}0 . \end{aligned} \qquad (12)$$

Die auf diese Weise durch ein kleines Vielfaches der ursprünglichen Rechenoperationen erweiterte Abfolge von Berechnungen ist in Tab. 2 dargestellt. Für eine bessere Lesbarkeit sind Zwischenergebnisse und die dazugehörige partielle Ableitung jeweils paarweise grau oder weiß gekennzeichnet.

Die Erweiterung einer Funktion $f: \mathrm{R} \to \mathrm{R}$ zu $f: \mathrm{D} \to \mathrm{D}$ kann in der Programmiersprache C++ mit Hilfe der Bibliothek FADBAD++ (Bendtsen und Stauning 1996) realisiert werden. Die Grundidee von FADBAD++ ist, dass für die AD neue Datentypen definiert werden. Die Grundelemente (2) werden hier durch die Datentypen für die *Forward*-(`FDouble`) sowie *Backward*-Methode (`BDouble`) definiert. Jede Variable U dieses Typs besteht aus dem skalaren Realteil u (z.B. vom Datentyp `double`), einem Vektor u' mit den Infinitesimalteilen sowie weiteren Variablen. Sind Ableitungen höherer Ordnung gewünscht, muss der Infinitesimalteil u' wiederum vom Datentyp `FDouble` oder `BDouble` sein. Für eine Ableitung der Ordnung p ergeben sich aufgrund der zwei möglichen Methoden 2^p Kombinationen von Datentypen. Für eine partielle Ableitung zweiter Ordnung ($p = 2$) im Forward-Modus gilt dann: Die Variable U besteht aus einem Realteil u und einem Vektor u' mit den partiellen Ableitungen erster Ordnung. Diese partiellen Ableitungen sind wiederum Variablen vom Typ `FDouble`. Sie beinhalten ihrerseits einen Realteil mit dem jeweiligen Wert der partiellen Ableitung erster Ordnung sowie einen Infinitesimalteil. Der Infinitesimalteil ist ein Vektor mit den Werten der Ableitungen zweiter Ordnung.

Nach Bendtsen und Stauning (1996) ist die *Backward*-Methode schneller für die Ableitung von wenigen abhängigen Variablen nach vielen Inputvariablen. Soll eine

1 Der hochgestellte Punkt bei $\dot{\tau}_i$ bezeichnet in diesem Artikel allgemein den mathematischen Wert der partiellen Ableitung, nicht – wie in der Physik üblich – die Ableitung »nach der Zeit«.

Tab. 2: Berechnungssequenz für die Forward-Methode (Griewank 2000)

τ_1	=	x_1	=	1,5000		
$\dot{\tau}_{-1}$	=	$\dot{x}_1$	=	1,0000 (Initialisierung als Variable)		
τ_0	=	x_2	=	0,5000		
$\dot{\tau}_0$	=	$\dot{x}_2$	=	0,0000 (Initialisierung als Konstante)		
τ_1	=	τ_{-1}/τ_0	=	1,5000 / 0,5000	=	3,0000
$\dot{\tau}_1$	=	$(\dot{\tau}_{-1} - \dot{\tau}_1 \cdot \dot{\tau}_0)/\tau_0$	=	1,0000 / 0,5000 (Quotientenregel)	=	2,0000
τ_2	=	$\sin(\tau_1)$	=	sin (3,0000)	=	0,1411
$\dot{\tau}_2$	=	$\cos(\tau_1) \cdot \dot{\tau}_1$	=	−0,9900 · 2,0000 (Kettenregel)	=	−1,9800
τ_3	=	e^{τ_0}	=	$e^{0,5000}$	=	1,6487
$\dot{\tau}_3$	=	$\tau_3 \cdot \dot{\tau}_0$	=	1,6487 · 0,0000	=	0,0000
τ_4	=	$\tau_1 - \tau_3$	=	3,0000 − 1,6487	=	1,3513
$\dot{\tau}_4$	=	$\dot{\tau}_1 - \dot{\tau}_3$	=	2,0000 − 0,0000	=	2,0000
τ_5	=	$\tau_2 + \tau_4$	=	0,1411 − 1,3513	=	1,4924
$\dot{\tau}_5$	=	$\dot{\tau}_2 + \dot{\tau}_4$	=	−1,9800 + 2,0000	=	0,0200
τ_6	=	$\tau_5 \cdot \tau_4$	=	1,4924 · 1,3513 (Produktregel)	=	2,0167
$\dot{\tau}_6$	=	$\dot{\tau}_5 \cdot \tau_4 + \tau_5 \cdot \dot{\tau}_4$	=	0,0200 · 1,3513 + 2,0000 · 1,4924	=	3,0118
y	=	τ_6	=	2,0167		
$\dot{y}$	=	$\dot{\tau}_6$	=	3,0118		

große Menge von abhängigen Variablen nach einigen wenigen unabhängigen Variablen abgeleitet werden, ist die *Forward*-Methode zu bevorzugen. Die Untersuchungen von Kersten (2008) haben darüber hinaus gezeigt, dass bei Ableitungen zweiter Ordnung die Kombinationen *Backward-Backward* und *Backward-Forward* am effizientesten sind.

Die obige Klassenschablone wird vom Programmierer mit dem gewünschten Datentyp spezialisiert. Sind partielle Ableitungen erster Ordnung gewünscht, erfolgt dies mit dem Datentyp `double` (Listing 2).

Für das Newton-Raphson-Verfahren (siehe Abschnitt 4) werden jedoch auch Ableitungen zweiter Ordnung benötigt. Dafür wird die Klassenschablone auf Typebene mit dem Datentyp der ersten Ableitung wie folgt spezialisiert (Listing 3).

Die Programmierung mit AD ist besonders elegant, wenn die Programmiersprache die Möglichkeit zur *Operatorüberladung* anbietet. Der Compiler (oder Interpreter) prüft bei der Operatorüberladung, ob für einen bestimmten, nicht-elementaren Datentyp, andere Anweisungen für die arithmetischen Operatoren, beispielsweise +, −, · und \, programmiert worden sind. Im Fall der AD werden für die neuen Datentypen `FDouble` (Datentyp für die *Forward*-Methode, siehe Listing 1) und `BDouble` (Datentyp für die *Backward*-Methode) sämtliche elementaren Rechenoperatoren und Funktionen, wie z. B. +, −, ·, \ sowie typische Funktionen wie sin(·), entsprechend der differentiellen Arithmetik neu definiert.

Die Rechenoperatoren können dann im Sinne von (3) auf Variablen der neuen Datentypen der FADBAD++ Bib-

```
1  template <class T>
2  class FTypeName
3  {
4  public:
5     T v;                          // Realteil
6     T *g;                         // Array mit Infinitesimalteilen
7     int gsize;                    // Größe dieses Feldes
8     void diff(int idx, int n); // Definition der abhängigen Variablen
9     T& x();                       // Ausgabe des Realteils
10    T& d(int i);                  // Ausgabe der gewünschten Ableitung
11
12    // weitere Methoden
13 };
```

Listing 1: FADBAD++ Typdeklaration für den Forward-Modus

```
1  typedef FTypeName<double> FDouble; // Forward
2  typedef BTypeName<double> BDouble; // Backward
```

Listing 2: Typspezialisierung der Datentypschablone für Ableitungen erster Ordnung

```
1  // Forward-Forward:
2  typedef FTypeName<FDouble> FF;
3  // Forward-Backward:
4  typedef FTypeName<BDouble> FB;
5  // Backward-Backward:
6  typedef BTypeName<BDouble> BB;
7  // Backward-Forward:
8  typedef BTypeName<FDouble> BF;
```

Listing 3: Verschiedene Typspezialisierungen für Ableitungen zweiter Ordnung

```
template <class T>
FTypeName<T> operator/ (const FTypeName<T>& a, const FTypeName<T>& b)
{
    // Division der Realteile
    FTypeName<T> c(a.v/b.v);

    // Allokieren des Gradienten
    c.touchg(a.gsize);

    // Anwendung der Quotientenregel
    for (int i=0;i<a.gsize;i++) c.g[i]=(a.g[i]-c.v*b.g[i])/b.v;

    // Rückgabe
    return c;
}
```

Listing 4: Überladung des Operators / für die Differentiationsarithmetik

```
    // Deklaration der Variablen
    FDouble x1,x2,y;

    // Kennzeichnen der unabhängigen Variablen
    x1.diff(0,2);    // Differenziere nach x1..
    x2.diff(1,2);    //..und nach x2

    // Berechnung des Quotienten
    y = x1/x2;

    // Ausgabe (keine Berechnung!)
    std::cout<<y.x()<<endl;          // Funktionswert
    std::cout<<y.d(0).x()<<endl;     // Wert der Ableitung nach x1
    std::cout<<y.d(1).x()<<endl;     // Wert der Ableitung nach x2
```

Listing 5: Anwendung der Differentiationsarithmetik

liothek angewendet werden. Die Überladung des Operators für die Division erfolgt beispielhaft in Listing 4. Die Anwendung der Arithmetik ist in Listing 5 für die Division exemplarisch dargestellt.

Die Programmierung von Softwarebibliotheken für das Automatische Differenzieren ist natürlich auch mit anderen Programmiersprachen, z. B. Python (Walter 2013) oder Java (Pham-Quang 2012) möglich.

4 Einsatz von AD für Aufgaben der Ausgleichungsrechnung

4.1 Gauß-Newton und Newton-Raphson

Die unbekannten Parameter eines überbestimmten Gleichungssystems im Gauß-Markov-Modell werden üblicherweise durch die Methode der kleinsten Quadrate geschätzt. Die gewichtete Quadratsumme der Verbesserungen wird hier gemäß der Vorschrift

$$\Omega = \boldsymbol{v}^T \boldsymbol{P} \boldsymbol{v} \rightarrow min \tag{13}$$

minimiert, wodurch die bestmögliche und wahrscheinlichste Schätzung $\hat{\boldsymbol{x}}$ der gesuchten Parameter $\boldsymbol{x}$ erreicht wird (siehe z. B. Jäger et al. 2005). Das in der Geodäsie wohl bekannteste Verfahren zur Lösung dieses Minimierungsproblems ist das Gauß-Newton-Verfahren (GN). Bei diesem iterativen und lokalen Verfahren werden die Verbesserungsgleichungen des funktionalen Modells in jedem Iterationsschritt vor der Bildung der Normalgleichungen mittels Taylorapproximation erster Ordnung an der Stelle $\boldsymbol{x}_i$ linearisiert und anschließend im Sinne von Kriterium (13) exakt gelöst. Die auf diese Weise geschätzten Parameter werden dann wiederum in der nächsten Iteration eingesetzt. Es ist allerdings zu beachten, dass GN auf die Methode der kleinsten Quadrate festgelegt ist und somit keine alternativen Kriterien, wie z. B. robuste Schätzer, Anwendung finden können. Die sukzessive Eliminierung von fehlerhaften Beobachtungen erfolgt hier üblicherweise mit Verfahren der Regewichtung (Caspary 2013). Allerdings: »Bei dem in der Literatur oftmals vorgeschlagenen Lösungsweg der ›iterativen regewichteten L2-Schätzung‹ lässt sich bereits an einfachen numerischen Beispielen zeigen, dass der Einsatz dieser Methode zu falschen Ergebnissen führen kann« (Neitzel 2004).

Das Newton-Raphson-Verfahren (NR) ist eine Alternative zu GN. In den Arbeiten von Teunissen (1990) und Lohse (1994) wird dieses und weitere Verfahren im Kontext der Lösung nicht-linearer Ausgleichungsprobleme untersucht. Prinzipiell wird bei NR im ersten Schritt die gewünschte Zielfunktion $\boldsymbol{\Omega}(\boldsymbol{x})$ aus den Verbesserungsgleichungen formuliert. Für die gesuchten Schätzwerte $\hat{\boldsymbol{x}}$ muss die Forderung eines lokalen Extremums

$$\nabla \boldsymbol{\Omega}(\hat{\boldsymbol{x}}) = \frac{\partial \boldsymbol{\Omega}}{\partial \hat{x}_i} = 0, i = 1, \ldots, n, \tag{14}$$

erfüllt werden, wobei n die Anzahl der unbekannten Parameter beschreibt. Zur Lösung von Gleichungssystem (14)

kann eine Iterationsvorschrift bestimmt werden, indem die Gleichungen mittels Taylorapproximation erster Ordnung linearisiert werden. Für die differentiellen Zuschläge $\Delta\boldsymbol{x}$ zu den aktuellen Schätzwerten $\hat{\boldsymbol{x}}_{i-1}$ folgt dann für Iteration i der Ausdruck

$$\left(\hat{\boldsymbol{x}}_i - \hat{\boldsymbol{x}}_{i-1}\right) = \Delta\boldsymbol{x} = \left(-\nabla^2\boldsymbol{\Omega}\left(\hat{\boldsymbol{x}}_{i-1}\right)\right)^{-1} \cdot \nabla\boldsymbol{\Omega}\left(\hat{\boldsymbol{x}}_{i-1}\right), \tag{15}$$

wobei der Vektor $\nabla\boldsymbol{\Omega}\left(\hat{\boldsymbol{x}}_{i-1}\right)$ die partiellen Ableitungen erster Ordnung und die symmetrische Hesse-Matrix $\nabla^2\boldsymbol{\Omega}\left(\hat{\boldsymbol{x}}_{i-1}\right)$ die partiellen Ableitungen zweiter Ordnung enthält.

Auch NR ist ein lokales und iteratives Verfahren, bei dem Näherungswerte benötigt werden. Ein wichtiges Kriterium bei nicht-linearen Ausgleichungsproblemen ist die Konvergenzgeschwindigkeit. Eine lineare Konvergenz liegt vor, wenn der maximale Zuschlag zwischen zwei Iterationen bei jedem Iterationsschritt um einen konstanten Faktor $0 < c < 1$ kleiner wird. Es sind weniger Iterationsschritte nötig, wenn der Zuschlag bei jedem Iterationsschritt quadratisch abnimmt. NR weist, sobald die Startwerte im Konvergenzbereich der Lösung liegen, eine quadratische Konvergenz auf, wohingegen bei GN im Allgemeinen mit linearer Konvergenz zu rechnen ist (Madsen et al. 2004). In beiden Fällen wird bei linearen Ausgleichungsproblemen mit der Methode der kleinsten Quadrate die exakte Lösung in einem Iterationsschritt bestimmt.

In Kersten (2008) wurde neben dem Newton-Raphson-Verfahren auch das schneller konvergierende Halley-Verfahren (siehe z.B. Schwetlick 1979) implementiert und getestet. Die automatische Berechnung der hier zusätzlich benötigten partiellen Ableitungen dritter Ordnung hat jedoch zu teilweise extrem langen Rechenzeiten geführt.

4.2 Anwendungspotenziale für AD in der Ausgleichungsrechnung

Unabhängig von der eigentlichen Methode zur Lösung von überbestimmten Gleichungssystemen kann AD für die automatische und exakte Berechnung der Werte von partiellen Ableitungen beliebig komplexer Funktionen angewendet werden. Unter Verwendung der Bibliothek FADBAD++ müssen dazu lediglich die Verbesserungsgleichungen programmiert werden – die elementaren Rechenoperationen der Differentiationsarithmetik sind in der Softwarebibliothek definiert.

Bei GN kann AD gewinnbringend eingesetzt werden, um die Werte der partiellen Ableitungen der Verbesserungsgleichungen im Rahmen der Linearisierung automatisch zu bestimmen, wodurch der Aufwand vor allem für komplexe Probleme – beispielsweise einer Bündelblockausgleichung – stark reduziert wird und Fehler vermieden werden können. Clemen (2010) nutzt AD für die Berechnung partieller Ableitungen für Ausgleichungsprobleme im Kontext von 3D-Gebäudemodellen.

Die Bestimmung der Werte der partiellen Ableitungen bis zur zweiten Ordnung durch AD im Rahmen von NR ist ebenfalls für beliebig komplexe Problemstellungen automatisch und ohne Mehraufwand möglich. Dies wird in Kersten (2008) untersucht und demonstriert. Der entscheidende Vorteil von NR gegenüber GN liegt darin, dass NR nicht auf die Methode der kleinsten Quadrate beschränkt ist. Unabhängig von der Wahl des Minimierungskriteriums ändert sich der NR-Lösungsalgorithmus nicht, wodurch die Anwendung robuster Schätzer (Andrews et al. 1972, Huber 1981) ermöglicht wird. In der Geodäsie sowie in verwandten Wissenschaften wurden robuste Schätzer in den letzten Jahrzehnten ausgiebig untersucht und angewendet (Krarup et al. 1980, Harvey 1993, Yang 1999, Domingo 2001, Wieser und Brunner 2001). Insbesondere die Klasse der robusten M-Schätzer ist von großem Interesse (Huber 1981, Hampel et al. 1986, Wiśniewski 2014). Die Anwendung von M-Schätzern in der geodätischen Ausgleichungsrechnung wurde beispielsweise in (Krarup und Kubik 1983, Gui und Zhang 1998) untersucht. Im Folgenden wird daher die Anwendung von AD für robuste M-Schätzer diskutiert.

5 Beispielhafte Anwendung von AD für robuste M-Schätzer

5.1 Programmierung der M-Schätzer mit AD

Die Verlustfunktion ist definiert als die Abweichung einer Schätzfunktion vom gesuchten optimalen Schätzergebnis und eignet sich daher zur Bewertung dieser (Niemeier 2002). Die Minimierung der Summe der Verlustfunktion

$$\rho\left(v_i\right) = \frac{1}{2}v_i^2 \tag{16}$$

führt zur Methode der kleinsten Quadrate. Die Einflussfunktion $\Psi\left(v\right) = \partial\rho\left(v\right)/\partial v$ spiegelt die Eigenschaften eines Schätzers äquivalent zur Verlustfunktion wider. Im Fall der L^2-Norm ist Ψ eine Gerade und somit unbeschränkt (Abb. 2). Nach Caspary (1988, 1996) sollten robuste Schätzverfahren überall einen beschränkten Einfluss von einzelnen Beobachtungen auf das Schätzergebnis aufweisen. Der mittlere Bereich der Einflussfunktion sollte dabei der Methode der kleinsten Quadrate entsprechen. Robuste M-Schätzer – entwickelt aus der Verallgemeinerung der Maximum-Likelihood-Methode – zeichnen sich durch eine solche beschränkte Einfluss- und Ψ-Funktion aus. Durch die Wahl einer alternativen Verlustfunktion werden hier die Eigenschaften des Schätzverfahrens beeinflusst. Im Folgenden werden drei ausgewählte M-Schätzer vorgestellt.

Die Verlust- sowie Ψ-Funktion des *modifizierten Huber-Schätzers* ist wie folgt definiert

$$\rho(v)=\begin{cases}\frac{1}{2}\cdot v^2 & , \quad |v|\le k\\ k\cdot|v|-\frac{1}{2}k^2 & , \quad k<|v|\le m\\ k\cdot m-\frac{1}{2}k^2 & , \quad |v|>m\end{cases} \tag{17}$$

$$\Psi(v)=\begin{cases}v & , \quad |v|\le k\\ k\cdot sign(v) & , \quad k<|v|\le m\\ 0 & , \quad |v|>m\end{cases} \tag{18}$$

wobei die Tuningkonstanten z. B. mit k = 1,5 und m = 3,0 angesetzt werden können. Abb. 3 verdeutlicht, dass die oben genannten Eigenschaften eines robusten Schätzers erfüllt sind. Während für Verbesserungen im Bereich $m \ge |v| > k$ ein linearer Einfluss vorliegt, haben Verbesserungen $|v| > m$ keinerlei Einfluss mehr auf das Schätzergebnis.

Ein ähnliches Verhalten weisen die Schätzer *Hampel*

$$\rho(v)=\begin{cases}\frac{1}{2}\cdot v^2 & , \quad |v|\le a\\ a\cdot|v|-\frac{1}{2}a^2 & , \quad a<|v|\le b\\ a\cdot b-\frac{a^2}{2}+\frac{a}{2}\cdot(c-a)\cdot\left[1-\left(\frac{c-|v|}{c-b}\right)^2\right] & , \quad b<|v|\le c\\ a\cdot b-\frac{a}{2}+\frac{a}{2}\cdot(c-b) & , \quad |v|>c\end{cases} \tag{19}$$

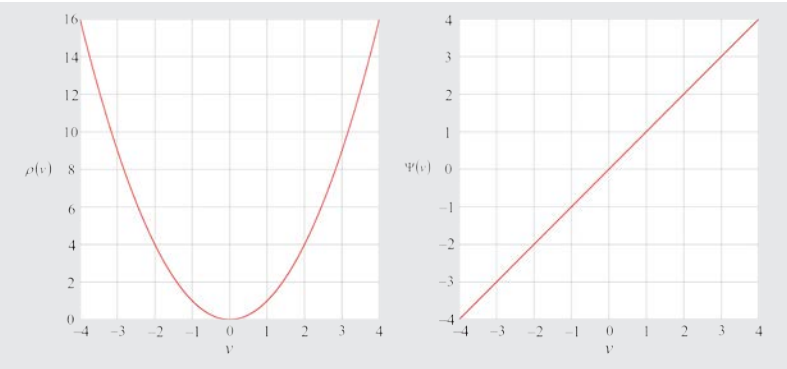


Abb. 2: Verlust- und Ψ-Funktion für die L^2-Norm

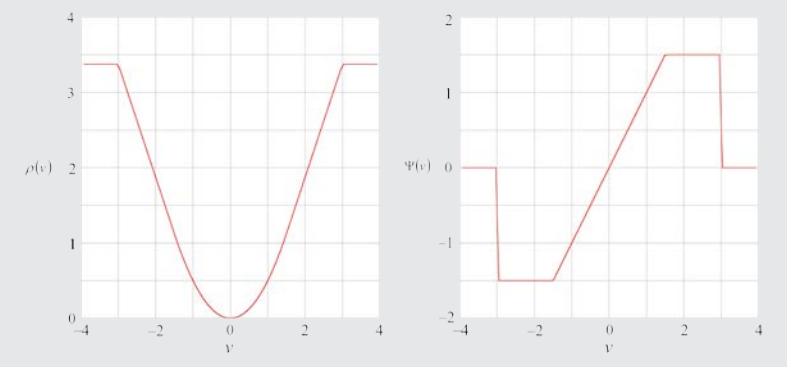


Abb. 3: Verlust- und Ψ-Funktion für den modifizierten Huber-Schätzer

$$\Psi(v)=\begin{cases}v & , \quad |v|\le a\\ a\cdot sign(v) & , \quad a<|v|\le b\\ a\cdot\frac{c-|v|}{c-b}\cdot sign(v) & , \quad b<|v|\le c\\ 0 & , \quad |v|>c\end{cases} \tag{20}$$

beispielsweise mit a = 2, b = 4 und c = 8, sowie Welsch

$$\rho(v)=\frac{1}{2}\cdot a_W{}^2\cdot\left(1-e^{\left(-(v/a_W)\right)^2}\right),\quad \Psi(v)=v\cdot e^{-(v/a_W)^2} \tag{21}$$

auf.

5.2 Programmierung der M-Schätzer mit AD

Bezüglich der Umsetzung von M-Schätzern in C++ ändert sich am eigentlichen Algorithmus zum Lösen des Minimierungsproblems mittels NR nichts. Zusätzlich zum implementierten funktionalen Zusammenhang muss lediglich mit sehr wenig Aufwand die Verlustfunktion $\rho(v_i)$ dem Schätzer entsprechend im Quellcode angepasst werden. Für die Berechnung unterschiedlicher Zielfunktionen in einem Programm eignet sich eine Interface-Klasse (Listing 6), die dann entsprechend des Schätzers spezialisiert wird (Listing 7).

Anhand der definierten Objektklasse (z. B. `HuberK_Mod`) wird dann zur Laufzeit der Berechnungen entschieden, welche Methode für die Berechnung der Zielfunktion mittels `CalcEstimFunktn` angewendet wird.

Als Grundlage für die robuste Parameterschätzung ist eine einheitlich skalierte Form der Verbesserungen erforderlich. Anders als beim Gauß-Newton-Verfahren wird hier jedoch von nichtlinearen Beobachtungsgleichungen ausgegangen bzw. auf eine Linearisierung dieser verzichtet. Eine Homogenisierung des funktionalen und stochastischen Modells (siehe z. B. Jäger 2005) führt unter Transformation der Verbesserungen $\boldsymbol{v}$ auf unkorrelierte und gleich genaue Beobachtungen $\bar{\boldsymbol{l}}$ mit der Varianz $\sigma_{\bar{l}}^2 = 1{,}0$. Die Homogenisierung erfolgt durch Linksmultiplikation des funktionalen Modells mit der Transformationsmatrix $\boldsymbol{C}_{ll}^{-1/2}$, also

$$\bar{\boldsymbol{v}} = \boldsymbol{C}_{ll}^{-1/2}\cdot\boldsymbol{v} = \boldsymbol{P}^{1/2}\cdot\boldsymbol{v}\,. \tag{22}$$

5.3 Simulation: Ausgleichende Gerade

Um Relevanz und Effektivität von NR unter Verwendung von AD zu demonstrieren, sei hier das Beispiel der ausgleichenden Gerade aus Kersten (2008) angeführt. Bei der Anwendung der Methode der kleinsten Qua-

```
// Interface-Klasse für die Schätzfunktionen
class IEstimation
{
 public:
     // Methode zur Berechnung der Schätzfunktion
     virtual void CalcEstimFunktn()=0;
     // Namen der Schätzfunktion erfragen
     virtual string getName()=0;
    // Veränderung der Tuningparameter
     virtual void setTuningParam(double t1, double t2, double t3)=0;
};
```

Listing 6: Interface-Klasse für die Schätzfunktionen

```
// Spezialisierung der Interface-Klasse: Modifizierter Huber-Schätzer
class HuberK_Mod: public IEstimation
{
public:

    HuberK_Mod(){k = 1.5; m = 4.0;};

    // Berechnung der Zielfunktion für den modifizierten Huber-Schätzer
    void CalcEstimFunktn()
    {
         vtPv = 0; // Initialisierung von vtPv (Zielfunktion)

         // Berechnung von vtPv für die n Beobachtungsgleichungen
         for (int i = 0; i < n; i++ )
         {
            // Verbesserung für Beobachtung i mittels Unbekannten
            FDOUBLE v = getResidual();
            if ( v < 0 ) v *= -1; // Betrag der Verbesserung

            v *= sqrt(P(i)); // Homogenisierung mit Gewicht P nach (23)

            // Quadratischer Teil der Schätzfunktion
            if ( v <= k ) vtPv += pow(v,2);
            // Schätzfunktion für linearen Teil
            else if ( v <= m && v > k )  vtPv += 2 * k * v -  pow( k, 2 );
            // v hat keinen Einfluss auf das Ergebnis mit
            else vtPv += 2 * k * m - pow( m, 2 );
 }
    }

    string getName(){return »HuberK_Mod«;};

    void setTuningParam(double t1, double t2, double t3){k = t1; m = t2;};

protected:
    // Tuningkonstanten:
    double k;
    double m;
    // Ergebnis der Schätzfunktion
    FDOUBLE vtPv;
};
```

Listing 7: Funktion zur Berechnung der Zielfunktion für den modifizierten Huber-Schätzer mit dem FADBAD++ Datentyp FDOUBLE

drate wird aufgrund der Linearität dieses Problems die Lösung mittels NR und GN in einem Iterationsschritt geschätzt. Es sei darauf hingewiesen, dass dies für alternative Schätzer nicht mehr zutrifft und somit auch für lineare Probleme Näherungswerte erforderlich werden.

Zu fehlerfrei bekannten Zeitpunkten t wurden die Werte y gemessen (vgl. Tab. 3). Der Verlauf der Messungen soll nun mittels ausgleichender Gerade

$$y = f(t) = a_1 \cdot t + a_0 \qquad (23)$$

approximiert werden. Die Messungen sind gleich genau und nicht korreliert. Für den mittleren Gewichtseinheitsfehler einer Beobachtung mit dem Gewicht $p = 1$ wird $\sigma_{0,a\,priori} = 1$ angenommen. Es handelt sich um ein lineares Problem mit zwei unbekannten Parametern, woraus bei $n = 17$ Beobachtungen eine Redundanz von $r = 15$ folgt.

Mit der Methode der kleinsten Quadrate (sowohl mit NR als auch mit GN) wurden die Parameter $a_0 = 3{,}7162$ und $a_1 = 1{,}2517$ geschätzt. Die Messwerte wurden nun durch fünf grobe Fehler f verfälscht (Tab. 4).

Die Ergebnisse der L^2-Norm sowie der Schätzer Welsch, Mod.-Huber und Hampel mit diesen fehlerbehafteten Beobachtungen sind in Tab. 5 zusammen gefasst. Dabei beschreibt dx_{L^2} die Abweichung der Ergebnisse zum Schätzergebnis mit der Methode der kleinsten Quadrate ohne grobe Fehler, $|v_f - f|$ den Betrag der Differenz einer Verbesserung und dem korrespondierenden hinzugefügten Fehler, v_f die Verbesserungen der fehlerbehafteten Beobachtungen und v_{max} die jeweils zwei größten Verbesserung der fehlerfreien Beobachtungen (Verschmierungseffekt).

Anhand der Werte in Tab. 5 wird ersichtlich, dass der negative Einfluss der groben Fehler auf das Schätzergebnis für die L^2-Norm am größten ist. Mit dem Welsch-Schätzer konnte ein Ergebnis geschätzt werden, welches dem Ergebnis mit fehlerfreien Beobachtungen am nächsten kommt. Der Einfluss v_{max} der Fehler auf die korrekten Beobachtungen wurde von diesem Schätzer am stärksten reduziert. In Abb. 4 sind die Ergebnisse der vier Schätzer dargestellt. Die grüne Gerade ist dabei jeweils das Ergebnis aus der L^2-Norm Schätzung ohne grobe Beobachtungsfehler.

Tab. 3: Messwerte

Zeit t [s]	1,00	1,25	1,50	1,75	2,00	2,25
y	5,10	5,20	5,90	6,10	6,40	6,50
Zeit t [s]	2,50	2,75	3,00	3,25	3,50	3,75
y	6,60	6,90	7,20	7,70	7,90	8,30
Zeit t [s]	4,00	4,25	4,50	4,75	5,00	
y	8,50	9,10	9,40	10,00	10,20	

Tab. 4: Für die Simulation wurden fünf Messwerte verfälscht.

Zeit t [s]	1,25	1,50	3,00	4,74	5,00
Verfälschung f	+9,00	+10,00	+3,00	−7,00	−10,00

Tab. 5: Ergebnisse mit fünf groben Fehlern

Schätzer/ Tuning-konstanten	dx_{L^2}	v_f		$\lvert v_f - f \rvert$	v_{max}	
		t [s]	Wert		t [s]	Wert
L^2	da_0 = 7,582 da_1 = −2,415	1,25 1,50 3,00 4,74 5,00	−4,302 −6,306 −2,432 2,650 5,133	4,698 3,694 0,568 4,350 4,867	1,00 4,50	5,103 −3,458
Mod. Huber k = 4 m = 6	da_0 = 1,489 da_1 = −0,535	1,25 1,50 3,00 4,74 5,00	−8,045 −9,579 −2,885 5,469 8,441	0,955 0,421 0,115 1,531 1,559	4,25 4,50	−0,956 −1,090
Hampel a = 2,8 b = 5 c = 7	da_0 = 0,401 da_1 = −0,080	1,25 1,50 3,00 4,74 5,00	−8,565 −9,985 −2,609 6,535 9,625	0,435 0,015 0,391 0,465 0,375	2,75 2,50	0,432 0,411
Welsch a_W = 2,985	da_0 = 0,106 da_1 = −0,003	1,25 1,50 3,00 4,74 5,00	−8,763 −10,164 −2,673 6,606 9,716	0,237 0,164 0,327 0,394 0,284	2,50 2,75	0,330 0,329

5.4 Diskussion der Ergebnisse

Die Simulation demonstriert, dass durch die Anwendung des Newton-Raphson-Verfahrens die Nutzung verschiedener Schätzer möglich wird, ohne dabei den eigentlichen Minimierungsalgorithmus zu verändern. Wenngleich die Bildung der Zielfunktion hier schnell zu sehr komplexen und langen mathematischen Ausdrücken führen kann, ist dies für den Anwender und Programmierer nicht relevant. Es muss für ein beliebiges Ausgleichungsproblem lediglich der ohnehin herzuleitende funktionale Zusammenhang in Form von Verbesserungsgleichungen implementiert werden. Die Berechnung von partiellen Ableitungen bis zur zweiten Ordnung im Rahmen der Linearisierung der Zielfunktion erfolgt dank AD exakt und vollautomatisch. Implementierungsfehler können somit an dieser Stelle gänzlich ausgeschlossen werden. Das Newton-Raphson-Verfahren wurde von den Autoren unter Verwendung der Bibliothek FADBAD++ in C++ umgesetzt, wodurch eine schnelle und unkomplizierte Modifikation der Zielfunktion ermöglicht wird. In diesem Artikel werden beispielhaft Ergebnisse ausgewählter M-Schätzer für das Problem der ausgleichenden Geraden dargestellt und verglichen.

Im Gegensatz zur L^2-Norm sind für alternative Schätzer – unabhängig davon, ob das Problem linearer oder nichtlinearer Natur ist – Näherungswerte und eine iterative Schätzung erforderlich. Die Wahl der Tuningparameter ist nicht trivial: damit eine robuste Lösung mittels NR gefunden werden kann, ist es erforderlich, dass die Startwerte für die zu schätzenden Parameter auf Funktionswerte der Verlustfunktion $\rho(v)$ im nicht-linearen Teil dieser führen. So müssen die Startwerte für den modifizierten Huber-Schätzer (17) auf Funktionswerte $\rho(v)$ führen, welche innerhalb des Intervalls $[0,k]$ liegen. Ist diese

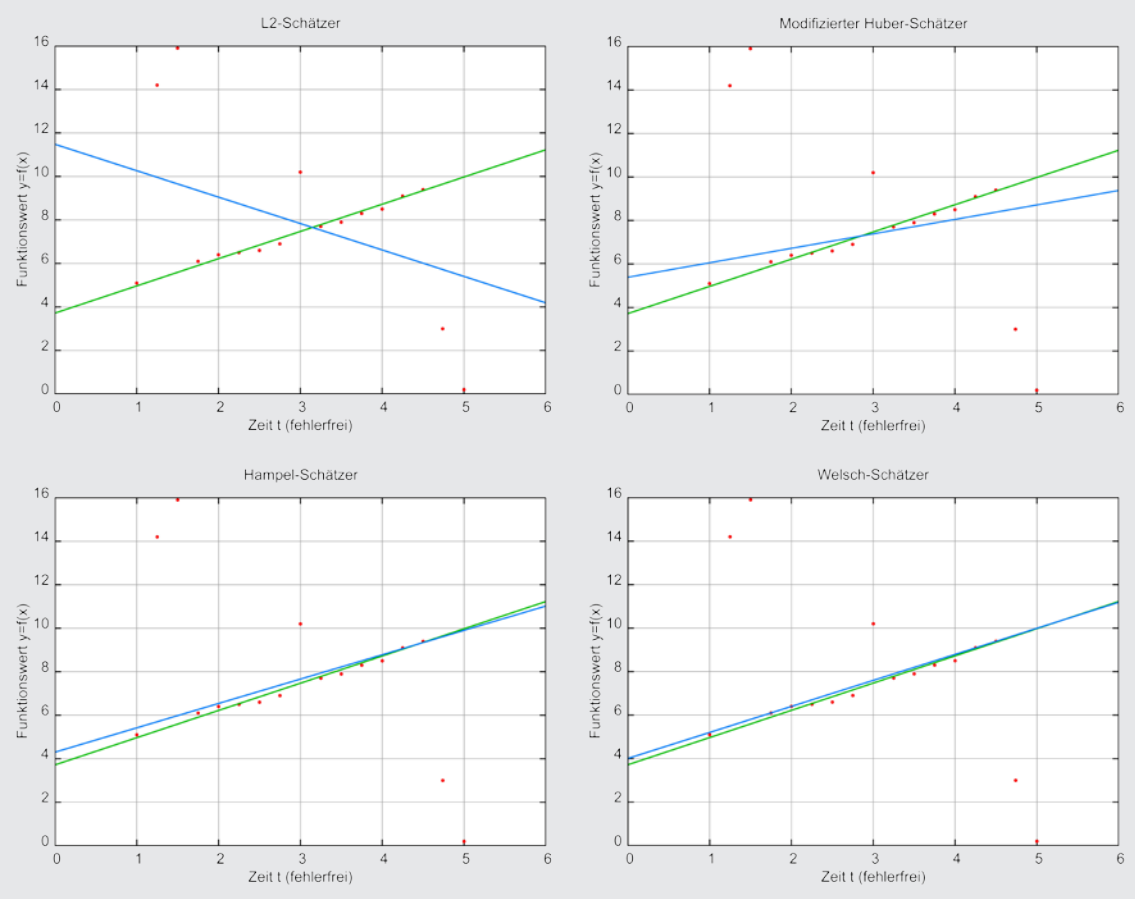


Abb. 4: Visueller Vergleich der vier Lösungen (blau) mit der L^2-Lösung ohne fehlerhafte Beobachtungen (grün)

Bedingung nicht eingehalten, kann aufgrund der lokalen Linearität der Verlustfunktion für Werte $v_i > k$ mit diesem Verfahren keine Lösung gefunden werden. Bei »ausreichend guten« Näherungswerten können die Tuningparameter mit den vorgeschlagenen Standardwerten angesetzt werden. Meist haben grobe Fehler jedoch auch schlechte Näherungswerte zur Folge, sodass es zu einer Überlagerung dieser beiden Effekte kommt. In diesem Fall ist ein iteratives Vorgehen erforderlich, bei dem zunächst die Tuningparameter so groß gewählt werden, dass eine Lösung gefunden werden kann. Anhand der Größe ihrer Verbesserung oder anhand der Berechnung der normierten Verbesserungen können dann fehlerhafte Beobachtungen erkannt und ausgeschaltet werden. Um dieses iterative Vorgehen sowie die Wahl von Tuningparametern zu vermeiden, wird die Anwendung komplett nicht-linearer Schätzer, wie Biweight, Fair, Krarup oder Welsch nahegelegt.

Anhand der Hesse-Matrix $\nabla^2\boldsymbol{\Omega}(\hat{\boldsymbol{x}}_{i-1})$ in (15) kann im Rahmen von NR eine Aussage über die Qualität des Schätzergebnisses getroffen werden. Nach Schwetlick (1979) ist im Rahmen der Lösung einer Minimierungsaufgabe erforderlich, dass die Hesse-Matrix symmetrisch und positiv semidefinit ist. Sind diese Voraussetzungen nicht erfüllt, ist die zu minimierende Zielfunktion in der Umgebung der aktuellen Schätzwerte $\hat{\boldsymbol{x}}_{i-1}$ nicht konvex, was zu einem Verlust von Lösungen führen kann.

6 Zusammenfassung und Ausblick

Basierend auf Clemen (2006, 2010) und Kersten (2008) wurde im vorliegenden Beitrag beschrieben, wie AD gewinnbringend für Aufgaben der Ausgleichungsrechnung eingesetzt werden kann. Die automatische und exakte Bestimmung von Werten partieller Ableitungen beliebig komplexer Verbesserungsgleichungen ist für eine schnelle Softwareentwicklung sehr hilfreich. Mit dem Newton-Raphson-Verfahren ist eine sehr interessante Alternative zu GN gegeben, da hier die Linearisierung erst nach der Bildung der Minimierungsfunktion erfolgt. Der Algorithmus für die Lösung des Minimierungsproblems ändert sich demnach – unabhängig von der Wahl der Minimierungsfunktion – nicht. Dies führt dazu, dass auch robuste Schätzer leicht programmiert werden können. Die Bestimmung des Gradienten sowie der Hesse-Matrix für das Newton-Raphson-Verfahren nach der Formulierung der Minimierungsfunktion ist mit AD selbst für extrem komplexe funktionelle Zusammenhänge unproblematisch. Ist der generelle Ablauf der Ausgleichung einmal implementiert, beschränkt sich der Entwicklungsaufwand auf die ohnehin herzuleitenden Verbesserungsgleichungen.

Die Simulation in Abschnitt 5.3 verdeutlicht den Mehrwert des Einsatzes von AD bei NR. Allerdings wird ebenfalls deutlich, dass die Anwendung von robusten Schätzern hier nur mit Einschränkungen möglich ist. Prinzipiell ist die Lösung eines überbestimmten Gleichungssystems nur dann möglich, wenn die aktuellen Schätzwerte $\hat{\boldsymbol{x}}_i$ zu

Verbesserungen $\boldsymbol{v}_i$ führen, welche in einen nicht-linearen Teil der Verlustfunktion $\rho(v)$ fallen. Aus diesem Grund ist auch eine Schätzung mittels L^1-Norm mit dem vorgestellten Ansatz nicht möglich. Komplett nicht-lineare Verlustfunktionen, wie die des Welsch-Schätzers, sind daher in diesem Kontext zu bevorzugen. Unabhängig davon, dass dieser Schätzer im vorliegenden Experiment das beste Ergebnis erzielt hat, ist hier aufgrund der nichtlinearen Schätzfunktion die mitunter schwierige Wahl des Tuningparameters vergleichsweise einfach. Die Untersuchung der automatisierten Anpassung von Tuningparametern könnte darüber hinaus Gegenstand zukünftiger Arbeiten sein.

Der Nutzen von AD wird noch deutlicher, wenn die Ausgleichungsrechnung mit komplexen Beobachtungsgleichungen programmiert wird oder andere Optimierungsverfahren zum Einsatz kommen. Die Anwendung und das Verhalten der hier vorgestellten Methodik in diesem Kontext sollten daher in weiterführenden Arbeiten untersucht werden. Mit anderen Softwarebibliotheken (siehe z. B. Hogen 2014) kann die Berechnungsgeschwindigkeit großer Optimierungsaufgaben noch gesteigert werden.

Literatur

Andrews, D. F., Bickel, P. J., Hampel, F. R., Huber, P. J, Rogers, W. H., Tukey, J. W.: Robust estimates of location. Princeton University, Press, Princeton, pp. 372, 1972.

Bendtsen, C., Stauning, O.: FADBAD: a Flexible C++ Package for Automatic Differentiation. Technical Report IMM-REP-1996-17, Department of Mathematical Modelling, Technical University of Denmark, 1996.

Brückner, H. M., Corliss, G., Hovland, P., Naumann, U., Norris, B. (Hrsg): Automatic Differentiation: Applications, Theory and Implementations. Springer, Berlin, 2006.

Caspary, W.: Fehlerverteilungen, Methode der kleinsten Quadrate und robuste Alternativen. ZfV, S. 113: 123–133, 1988.

Caspary, W.: Anmerkungen zum robusten Schätzen. Allgemeine Vermessungsnachrichten, No. 7, S. 287–289, 1996.

Caspary, W.: Fehlertolerante Auswertung von Messdaten: Daten- und Modellanalyse, robuste Schätzung. Oldenbourg Wissenschaftsverlag, München, 2013.

Clemen, C.: Anwendung der FADBAD++ Bibliothek zum automatischen Differenzieren. In: Clemen, C. (Hrsg.), Tagungsband Entwicklerforum Geoinformationstechnik, S. 87–97, Shaker-Verlag, Aachen, 2006.

Clemen, C.: Ein geometrisch-topologisches Informationsmodell für die Erfassung und Validierung von flächenparametrisierten 3d-Gebäudemodellen. Dissertation, Deutsche Geodätische Kommission bei der Bayerischen Akademie der Wissenschaften, Reihe C, 647, München, 2010.

Domingo, A. M.: Aplicación de las técnicas de estimación robusta en algunos problemas fotogramétricos. Topografía y Cartografía, 103, pp. 41–47, 104, pp. 48–57 and 105, pp. 30–37, 2001.

Griewank, A.: Evaluating Derivatives, Principles and Techniques of Algorithmic Differentiation. Soc. for Industrial and Applied Math, Philadelphia, PA, USA, 2000.

Gui, Q., Zhang, J.: Robust biased estimation and its applications in geodetic adjustment. J Geod 72, pp. 430–435, 1998.

Hampel, F. R., Ronchetti, E. M., Rousseuw, P. J., Stahel, W. A.: Robust statistics. The approach based on influence functions. Wiley, New York, 1986.

Harvey, B. R.: Survey network adjustments by the L1 method. Australian Journal of Geodesy, Photogrammetry and Surveying, 59, pp. 39–52, 1993.

Hogan, R. J.: Fast Reverse-Mode Automatic Differentiation using Expression Templates in C++. In: ACM Transactions on Mathematical Software, Vol. 40, Article 26, 2014.

Huber, P. J.: Robust statistics. John Wiley & Sons, New York, p. 328, 1981.

Jäger, R., Müller, T., Saler, H., Schwäble, R.: Klassische und robuste Ausgleichungsverfahren. Wichmann, Heidelberg, 2005.

Jerell, M. E.: Function minimization and automatic differentiation using C++. In: Conference proceedings on Object-oriented programming systems, languages and applications, pp. 169–173, ACM Press, New York, 1989.

Kersten, J.: Anwendung der Differentiations-Arithmetik für Aufgaben der Ausgleichungsrechnung. Diplomarbeit, Technische Universität Berlin, 2008. www.uni-weimar.de/fileadmin/user/fak/medien/professuren/Computer_Vision/Downloads/extern/DiplomarbeitJensKersten.pdf.

Krarup, T., Juhl, J., Kubik, K.: Götterdämmerung over least squares adjustment. In: Proc. 14th Congress of the International Society of Photogrammetry, Vol B3, pp. 369–378, Hamburg, 1980.

Krarup T., Kubik, K.: The Danish method; experience and philosophy. Deutsche Geodätische Kommission, München, Reihe A, Heft 7, pp. 131–134, 1983.

Lohse, P.: Ausgleichungsrechnung in nichtlinearen Modellen. DGK, Reihe C, Heft Nr. 429, 1994.

Madsen, K., Nielsen, H. B., Tingleff, O.: Methods for non-linear least squares problems. Technical Report, Informatics and Mathematical Modelling, Technical University of Denmark, 2004.

Neitzel, F.: Identifizierung konsistenter Datengruppen am Beispiel der Kongruenzuntersuchung geodätischer Netze. Dissertation, Deutsche Geodätische Kommission bei der Bayerischen Akademie der Wissenschaften, Reihe C, 565, München, 2004.

Niemeier, W.: Ausgleichungsrechnung. de Gruyter, Berlin, New York, 2002.

Pham-Quang, P., Delinchant B.: Java Automatic Differentiation Tool Using Virtual Operator Overloading. In: Recent Advances in Algorithmic Differentiation, Lecture Notes in Computational Science and Engineering, Springer, Heidelberg, 2012.

Rall, L. B.: The arithmetic of differentiation. Mathematics Magazine 59, pp. 275–282, 1986.

Schwetlick, H.: Numerische Lösung nichtlinearer Gleichungen. Deutscher Verlag der Wissenschaften, Berlin, 1979.

Teunissen, P.: Nonlinear least squares. Manuscripta Geodaetica 15, pp. 137–150, 1990.

Walter, S. F., Lehmann L.: Algorithmic differentiation in Python with AlgoPy. In: Journal of Computer Sciences, Vol. 4, Issue 5, pp. 334–344, 2013.

Wieser, A., Brunner, F. K.: Robust estimation applied to correlated GPS phase observations. In: Carosio A., Kutterer H. (Eds.) Proc. First Int. José Luis Berné Valero – Sergio Baselga Moreno Robust estimation in geodetic networks Symposium on Robust Statistics and Fuzzy Techniques in Geodesy and GIS. ETH Zürich, pp. 193–198, 2001.

Wiśniewski, Z.: M-estimation with probabilistic models of geodetic observations. Journal of Geodesy, pp. 941–957, 2014.

Yang, Y.: Robust estimation of geodetic datum transformation. Journal of Geodesy, 73, pp. 268–274, 1999.

Anschrift der Autoren

Dr.-Ing. Jens Kersten
Bauhaus-Universität Weimar
Fakultät Medien, Bereich Computer Vision
Bauhausstraße 11, 99423 Weimar
jens.kersten@uni-weimar.de
Tel.: +49 3643 58-3730, Fax: +49 3643 58-3728

Prof. Dr. -Ing. Christian Clemen
HTW Dresden
Fachgebiet CAD und Virtual Reality, Fakultät Geoinformation
Friedrich-List-Platz 1, 01069 Dresden
christian.clemen@htw-dresden.de

Dieser Beitrag ist auch digital verfügbar unter www.geodaesie.info.