Kapitel 4

Least-Squares Approximation

4.1 Das Grundproblem.

So leistungsfähig die Methode der Interpolation in vielen Fällen auch sein mag, gibt es doch sehr häufig Approximationsprobleme, bei welchen die gegebene Punktmenge keine glatte Funktion repräsentiert. Dies trifft vor allem dann zu, wenn die Stützpunktkoordinaten *statistisch fehlerbehaftet* sind, weil sie z.B. Ergebnisse experimenteller Messungen darstellen.

Betrachtet man beispielsweise die Bestimmung des Widerstandswertes R eines metallischen Leiters mittels der Versuchsanordnung Abb.4.1, so erhält man ein *Spannungs-Strom-Diagramm* der Art Abb.4.2.

Der Zusammenhang zwischen Spannung und Strom ist offenbar ein linearer: es gilt das *Ohm'sche Gesetz*

$$U = R * I$$

mit R als dem *elektrischen Widerstand*. An der grundsätzlichen Linearität zwischen U und I ändern auch die statistischen Abweichungen bzw. offensichtliche Fehlmessungen nichts!

Die in einem solchen Fall anzuwendende Approximationsfunktion muß also eine Gerade (ein Polynom ersten Grades) sein. Es ist klarerweise völlig sinnlos, hier eine Interpolationsfunktion dazu zu zwingen, genau durch alle Meßpunkte hindurchzugehen. Es ist im Gegenteil eine *Glättung* ohne übermäßige Berücksichtigung von Fehlmessungen erwünscht.

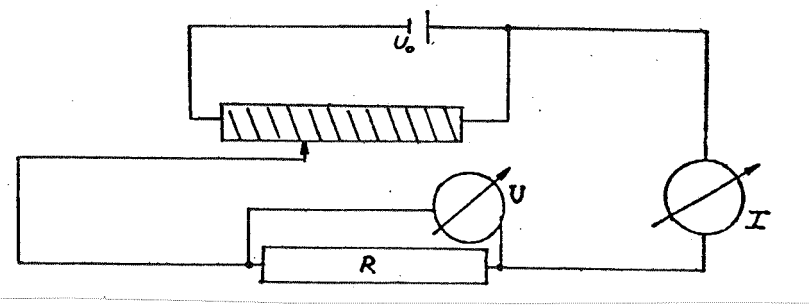


Abbildung 4.1: Versuchsanordnung: Bestimmung des elektrischen Widerstandes.

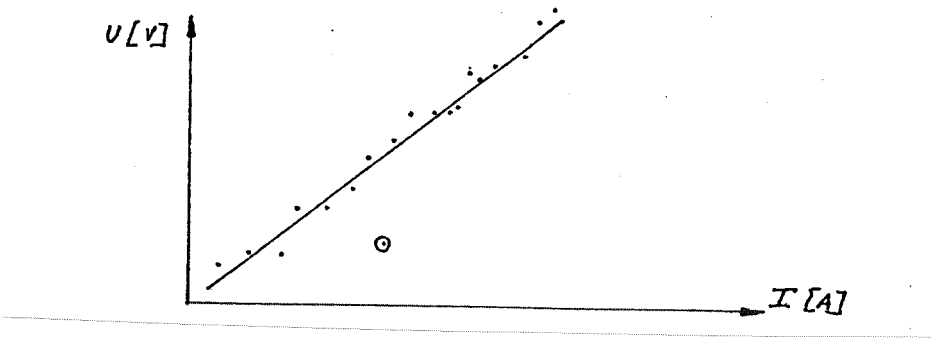


Abbildung 4.2: Ein Spannungs-Strom-Diagramm.

Abbildung 4.2: Ein Spannungs-Strom-Diagramm.

4.2 Mathematische Formulierung des Problems.

Es soll durch n gegebene Punkte $(x_i \mid y_i)$ eine Kurve $f(x)$ gelegt werden, die den Punkten möglichst nahekommt, wobei jedoch die auftretenden Meßunsicherheiten ausgeglichen werden sollen. Zusätzlich soll es noch möglich sein, den Einfluß der einzelnen Meßpunkte auf $f(x)$ durch Gewichtsfaktoren zu verändern.

Diese Forderungen lassen sich mathematisch folgendermaßen formulieren (Grundgleichung der Least-Squares (LSQ) Approximation):

$$\chi^2 = \sum_{k=1}^{n} g_k \left[y_k - f(x_k; \mathbf{a})\right]^2 \quad \rightarrow \quad \text{Minimum} \tag{4.1}$$

χ^2 ist die *gewichtete Fehlersumme*, $g_k > 0$ ist der *Gewichtsfaktor* des k-ten Punktes, und $f(x; \mathbf{a})$ ist die *Modellfunktion* mit den q *Modell- oder Fit-Parametern* $\mathbf{a} = a_1, a_2, \ldots, a_q$.

Die Summe der gewichteten Fehlerquadrate zwischen den gegebenen und den approximierten Funktionswerten soll ein Minimum werden. Die Bedeutung der Gewichtsfaktoren wird vor allem dann klar, wenn man sich für die statistische Auswertung von Punktmengen interessiert. Davon wird im nächsten Abschnitt dieses Kapitels die Rede sein.

Die Modellfunktion f dient in vielen Fällen nicht bloß dazu, 'eine schöne Kurve zu formen', sondern oft haben die Modellparameter auch eine konkrete *physikalische* Bedeutung.

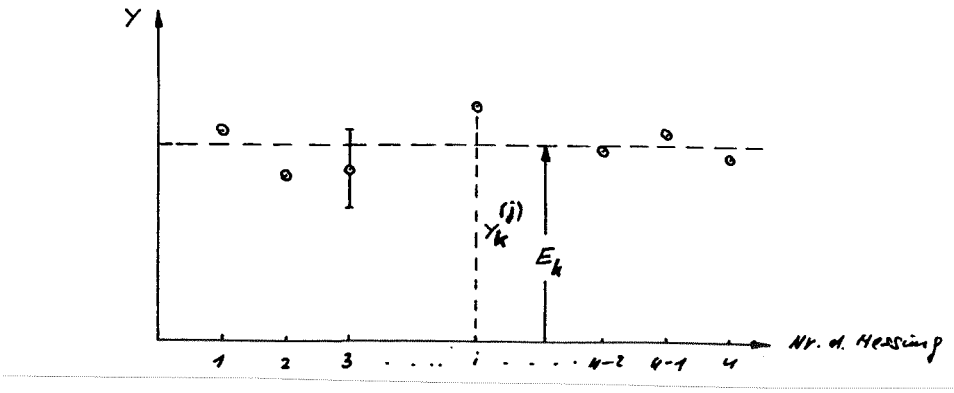


Abbildung 4.3: Erwartungswert von Einzelmessungen.

4.3 Die statistische Auswertung des Least-Squares Problems.

4.3.1 Grundbegriffe: Erwartungswert und Standardabweichung eines Meßwertes.

Die LSQ-Methode beruht auf dem Vergleich von (meist) experimentell ermittelten Meßwerten y_k mit den entsprechenden Modellwerten $f(x_k; \mathbf{a})$. Es ist evident,daß die Qualität der Ergebnisse (Fit-Parameter) sehr von der statistischen Qualität der y_k-Werte abhängen wird.

Angenommen, eine Messung wurde unter genau denselben Bedingungen n mal wiederholt. Als Meßwerte für $x = x_k$ wurden dabei die (i.a.) *verschiedenen* Werte $y_k^{(1)}, y_k^{(2)}, \ldots, y_k^{(n)}$ erhalten. Trägt man diese Werte in ein Diagramm ein (vgl. Abb.4.3), so erhält man eine Punktmenge, die um den *Erwartungswert* E_k oder *Mittelwert* der Messung verteilt ist:

$$E_k = \frac{1}{n} \sum_{j=1}^{n} y_k^{(j)} \tag{4.2}$$

Ein Maß für die *Streuung* der einzelnen Meßwerte um den Erwartungswert ist die *Standardabweichung*

$$\sigma_k = \left[\frac{\sum_{j=1}^{n} (y_k^{(j)} - E_k)^2}{n-1} \right]^{1/2} \tag{4.3}$$

In sehr vielen praktischen Fällen ist die *Wahrscheinlichkeit* P, mit der eine bestimmte Abweichung des Meßwertes vom Erwartungswert E auftritt, durch die *Wahrscheinlichkeitskurve*

$$P(y - E) = \frac{1}{\sqrt{2\pi}\sigma} \cdot \exp^{-(y-E)^2/(2\sigma^2)} \tag{4.4}$$

gegeben. Dies ist die sehr häufig vorkommende *Gauss'sche Normalverteilung* (Abb.4.4). Erwähnenswert ist in diesem Zusammenhang auch, daß $P(y - E)$ genau an den Stellen $y - E = \pm\sigma$ Wendepunkte hat.

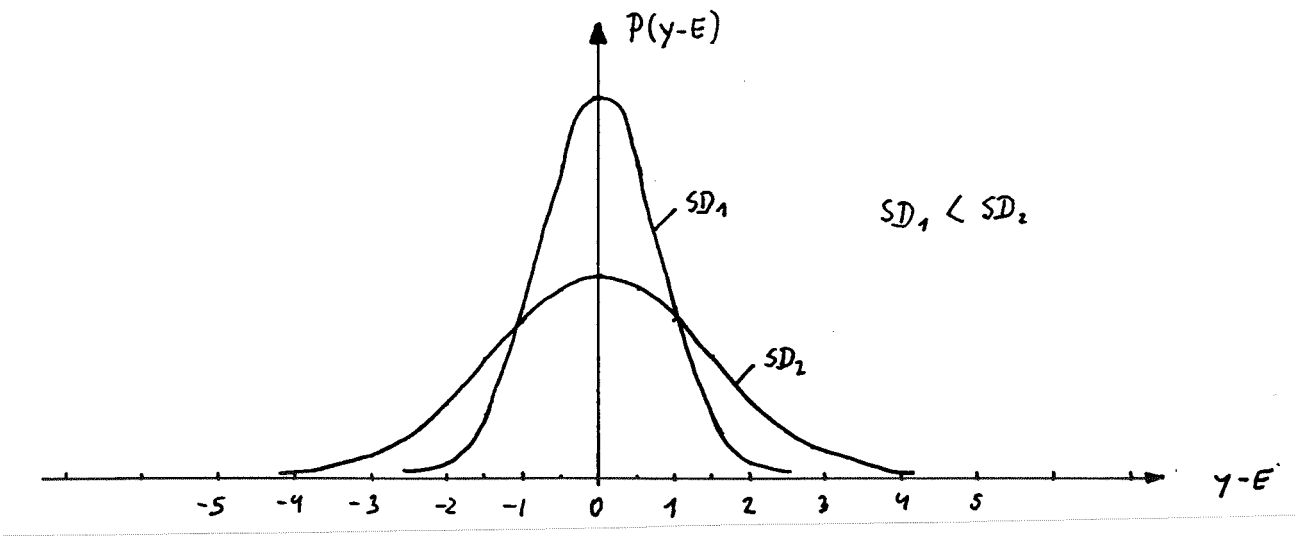


Abbildung 4.4: Die Gauss'sche Normalverteilung.

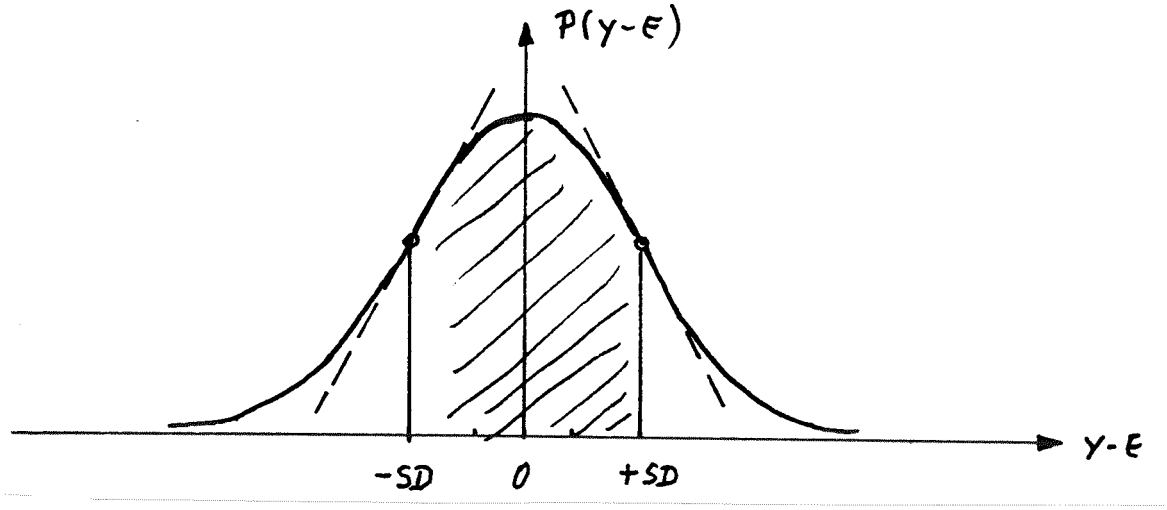


Abbildung 4.5: Zur geometrischen Bedeutung der Standardabweichung σ.

Der schraffierte Bereich in Abb.4.5 bedeutet offenbar jene Wahrscheinlichkeit, mit der ein Meßwert im Intervall $[E-\sigma, E+\sigma]$ liegt. Diese Wahrscheinlichkeit beträgt

$$\int_{-\sigma}^{+\sigma} d(y-E)\frac{1}{\sqrt{2\pi}\sigma}\exp^{(y-E)^2/(2\sigma^2)} = 0.685 \quad .$$

Bei Vorliegen einer Normalverteilung liegen also ca. 69 Prozent ($\approx 2/3$) aller Einzelwerte im Intervall $[E-\sigma, E+\sigma]$.

Normalverteilungen sind typisch für Meßvorgänge, bei denen die Meßgröße - zumindest innerhalb eines gewissen Intervalls - jeden beliebigen Wert annehmen kann (*analoge* Meßvorgänge). Die für *digitale* Meßvorgänge typische *Poisson-Verteilung* wird im nächsten Abschnitt diskutiert.

4.3.2 Berücksichtigung statistischer Größen im LSQ-Prozess.

Da diese LV. keine Statistik-Vorlesung ist, müssen die folgenden Überlegungen zwangsläufig ohne die an und für sich notwendigen Beweisführungen präsentiert werden.

- Die statistischen Unsicherheiten der y_k-Werte werden im LSQ-Prozess dadurch berücksichtigt, daß als Gewichtsfaktoren in (4.1) *die reziproken Quadrate der entsprechenden Standardabweichungen* genommen werden:

$$g_k \equiv \frac{1}{\sigma_k^2} \quad . \tag{4.5}$$

- Der LSQ-Prozess liefert die statistischen Erwartungswerte der Fit-Parameter. Die entsprechenden Standardabweichungen erhält man auf die folgende Weise:

  Man berechnet zuerst die sogenannte *Normalmatrix* N mit den Koeffizienten

$$[N]_{ij} = \frac{1}{2}\frac{\partial^2\chi^2}{\partial a_i \partial a_j} \quad . \tag{4.6}$$

  Durch Invertierung der Normalmatrix erhält man die *Kovarianzmatrix* C:

$$C = N^{-1} \tag{4.7}$$

  Die Kovarianzmatrix enthält zwei wichtige Informationen: einerseits stellen ihre Diagonalelemente die Quadrate der Standardabweichungen der Fit-Parameter dar, also

$$\sigma_{a_i} = \sqrt{c_{ii}} \quad , \tag{4.8}$$

  andererseits ergeben sich aus C die *Korrelationskoeffizienten* r_{ij} zwischen den Fit-Parametern:

$$r_{ij} = \frac{c_{ij}}{\sqrt{c_{ii}c_{jj}}} \quad . \tag{4.9}$$

Die r_{ij}-Werte, die stets im Intervall $[-1,+1]$ liegen, geben eine Information darüber, wie stark der i-te und der j-te Fit-Parameter einander beeinflussen.

- Berechnung der *Varianz* V bzw. deren Standardabweichung:

$$V = \frac{\chi^2}{n-q} \quad \text{und} \quad \sigma_V = \sqrt{\frac{2}{n-q}} \quad . \tag{4.10}$$

V ist im Falle eines *idealen Modells* und für $n >> 1$ approximativ normalverteilt mit dem Mittelwert 1 und der Standardabweichung σ_V. Die Größe $n-q$ (Anzahl der Stützpunkte minus Anzahl der Fit-Parameter) wird als die *Anzahl der Freiheitsgrade* bezeichnet.

Die Varianz eines Fits kann als Indikator für die Güte der verwendeten Modellfunktion dienen: wenn V signifikant außerhalb des Intervalles $[1-\sigma_V, 1+\sigma_V]$ liegt, paßt das Modell nicht gut zur gegebenen Punktmenge.

4.3.3 Die Bestimmung der Standardabweichungen der Einzelwerte.

Die statistische Auswertung eines LSQ-Problems setzt voraus, daß man die Standardabweichungen σ_k der Einzelwerte kennt. Woher bekommt man diese σ_k-Werte?

Für die Praxis sind vor allem die folgenden beiden Fälle von Bedeutung:

- Die Meßwerte sind um ihre Erwartungswerte normalverteilt:

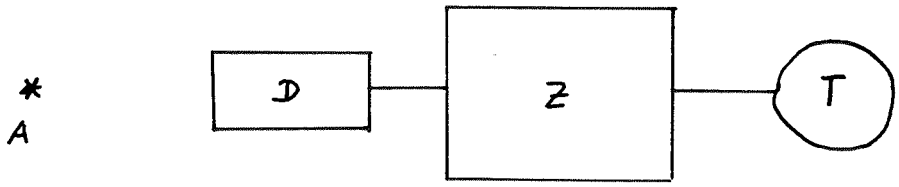
  Wie bereits erläutert, kommt diese Situation in der experimentellen Praxis sehr häufig vor. In diesem Fall gilt meistens wenigstens approximativ, daß die σ's *aller Meßwerte gleich sind*, daß also

$$\sigma_1 \approx \sigma_2 \approx \cdots \approx \sigma_n$$

  gilt. Diese repräsentative Standardabweichung kann dadurch bestimmt werden, daß man für einen ausgewählten x-Wert die Messung mehrmals wiederholt und das entsprechende σ mittels (4.3) berechnet.

- Wie bereits kurz erwähnt, gibt es aber noch eine andere wichtige Meßwert-Statistik, die bei allen Experimenten auftritt, die auf einer *Zählung von Ereignissen* beruhen (Digital-Messungen). In diesen Fällen sind die Einzelwerte um die jeweiligen Erwartungswerte *Poisson-verteilt*.

Zählexperimente spielen z.B. in der Kernphysik eine dominierende Rolle. Die Versuchsanordnungen entsprechen dabei dem Schema



Dabei könnte A ein radioaktives Präparat sein, D ein Strahlendetektor, Z ein Digitalzähler und T eine Uhr.

Ohne auf die Theorie der Poisson-Statistik näher einzugehen, soll hier nur die wichtigste Eigenschaft dieser Statistik erwähnt werden:

Bei einer Poisson-Verteilung ist für jeden Meßwert sein Erwartungswert E mit der entsprechenden Standardabweichung σ über die einfache Beziehung

$$\sigma = \sqrt{E}$$

verknüpft. Da man aber den Erwartungswert i.a. nicht kennt, nimmt man näherungsweise stattdessen den Meßwert selbst und erhält

$$\sigma \approx \sqrt{y} \quad .$$

In diesem Fall haben also die Gewichtsfaktoren beim LSQ-Verfahren die Form

$$g_k \approx \frac{1}{y_k} \quad . \tag{4.11}$$

4.4 Modellfunktionen mit linearen Parametern.

Unter einer *Modellfunktion mit linearen Parametern* versteht man eine Form

$$f(x;\mathbf{a}) = \sum_{j=1}^{m} a_j \cdot \varphi_j(x) \tag{4.12}$$

mit den beliebigen (linear unabhängigen) *Basisfunktionen* $\varphi_j(x)$. Die oft verwendete saloppe Bezeichung *lineare Modellfunktionen* ist mißverständlich: $f(x;\mathbf{a})$ muß nämlich keineswegs linear in x sein, sondern linear in den Fit-Parametern. Die vielzitierte *lineare Regression* hat als Modellfunktion

$$f(x; a_1, a_2) = a_1 + a_2\, x$$

eine Gerade. Es handelt sich dabei um eine sehr einfache Sonderform eines Modells mit linearen Parametern.

Setzt man ein Modell (4.12) in die LSQ-Grundgleichung (4.1) ein, so erhält man

$$\chi^2 = \sum_{k=1}^{n} g_k \left[y_k - \sum_{j=1}^{m} a_j \varphi_j(x_k) \right]^2 \quad \rightarrow \quad \text{Minimum!}$$

Die gewünschte Minimalisierung der Fehlersumme wird durch *Nullsetzen der partiellen Ableitungen von χ^2 nach den Modellparametern* **a** erreicht:

$$\frac{\partial \chi^2}{\partial a_i} = \sum_{k=1}^{n} g_k \varphi_i(x_k) \left[y_k - \sum_{j=1}^{m} a_j \varphi_j(x_k) \right] = 0 \quad i = 1, \ldots, m \quad .$$

Dies führt zu einem linearen, inhomogenen Gleichungssystem m-ten Grades für die a_i:

$$\sum_{j=1}^{m} a_j \sum_{k=1}^{n} g_k \varphi_i(x_k) \varphi_j(x_k) = \sum_{k=1}^{n} g_k y_k \varphi_i(x_k)$$

für alle $i = 1, \ldots, m$, also zu einem Problem

$$A \cdot \mathbf{a} = \beta$$

mit der symmetrischen, positiv-definiten Koeffizientenmatrix

$$A \equiv [\alpha_{ij}] \quad \text{mit} \quad \alpha_{ij} = \sum_{k=1}^{n} g_k \varphi_i(x_k) \varphi_j(x_k) \tag{4.13}$$

und dem inhomogenen Vektor

$$\beta_i = \sum_{k=1}^{n} g_k y_k \varphi_i(x_k) \quad . \tag{4.14}$$

Für lineare Modellfunktionen (4.12) ist die Aufstellung der Normalmatrix gemäß (4.6) sehr einfach; es gilt nämlich:

$$\frac{1}{2} \frac{\partial^2 \chi^2}{\partial a_i \partial a_j} = \sum_{k=1}^{n} g_k \varphi_i(x_k) \varphi_j(x_k) = \alpha_{ij} \quad . \tag{4.15}$$

4.4.1 Standardabweichungen der Fitparameter

Im Abschnitt 4.3.2 wurde (ohne Beweisführung) angegeben, daß sich die Standardabweichungen der Fitparameter aus den Diagonalelementen der Kovarianzmatrix ergeben [Glg. (4.8)]
Dieser Sachverhalt soll im folgenden an Hand eines sehr einfachen Modells demonstriert werden, nämlich mittels einer *linearen Regression* mit der Modellfunktion

$$f(x; a, b) = a + b\,x$$

Least-Squares-Formel:

$$\chi^2 = \sum_{k=1}^{n} g_k \, [y_k - a - b\,x_k]^2 \quad \rightarrow \quad \text{Minimum für} \quad a = a_{opt}, \quad b = b_{opt}\,.$$

Die Normalmatrix des Problems lautet nach Skriptum (4.6)

$$N = \begin{pmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{12} & \alpha_{22} \end{pmatrix}$$

mit

$$\alpha_{11} = \sum_{k=1}^{n} g_k \qquad \alpha_{12} = \sum_{k=1}^{n} g_k x_k \qquad \alpha_{22} = \sum_{k=1}^{n} g_k x_k^2$$

Die optimierten Parameter ergeben sich als Lösung des inhomogenen, linearen Gleichungssystems

$$N \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}$$

mit

$$\beta_1 = \sum_{k=1}^{n} g_k y_k \qquad \beta_2 = \sum_{k=1}^{n} g_k x_k y_k \quad .$$

Die Lösung dieses Gleichungssystems lautet

$$a = a_{opt} = \frac{\beta_1 \alpha_{22} - \beta_2 \alpha_{12}}{D} \qquad b = b_{opt} = \frac{\beta_2 \alpha_{11} - \beta_1 \alpha_{12}}{D} \tag{4.16}$$

mit D als der Determinanten der Normalmatrix

$$D = \alpha_{11}\alpha_{22} - \alpha_{12}^2 \quad .$$

Die *Kovarianzmatrix* C ist die Inverse der Normalmatrix, also

$$C = N^{-1} = \frac{1}{D} \begin{pmatrix} \alpha_{22} & -\alpha_{12} \\ -\alpha_{12} & \alpha_{11} \end{pmatrix}$$

Daraus erhält man unmittelbar die *Standardabweichungen der Fitparameter* als

$$\sigma_a^2 = \frac{\alpha_{22}}{D} \qquad \sigma_b^2 = \frac{\alpha_{11}}{D} \, . \tag{4.17}$$

Die Berechnung dieser Standardabweichungen kann aber auch auf eine andere Art geschehen, nämlich auf der Basis des Fehlerfortpflanzungsgesetzes (FFG).

Es ist klar, daß die optimierten Parameter a und b Funktionen aller $x-$ und $y-$Komponenten der gegebenen Punkte sind, also

$$a = a(x_1, \ldots, x_n; y_1, \ldots, y_n) \qquad b = b(x_1, \ldots, x_n; y_1, \ldots, y_n)$$

Nach dem FFG gilt nun für die Standardabweichungen von a und b

$$\sigma_a^2 = \sum_{l=1}^{n} \left[\left(\frac{\partial a}{\partial x_l} \right)^2 \sigma(x_l)^2 + \left(\frac{\partial a}{\partial y_l} \right)^2 \sigma(y_l)^2 \right]$$

und

$$\sigma_b^2 = \sum_{l=1}^{n} \left[\left(\frac{\partial b}{\partial x_l} \right)^2 \sigma(x_l)^2 + \left(\frac{\partial b}{\partial y_l} \right)^2 \sigma(y_l)^2 \right]$$

Dabei sind die $\sigma(x_l)$ und $\sigma(y_l)$ die Standardabweichungen (Fehler) der entsprechenden $x-$ bzw $y-$Komponenten. Nimmt man die $x-$Komponenten als

exakt an, gilt $\sigma(x_l) = 0$, und für die Fehler der $y-$Komponenten gilt nach Skriptum (4.5)

$$g_l = \frac{1}{\sigma(y_l)^2}$$

Es ergibt sich somit

$$\sigma_a^2 = \sum_{l=1}^{n} \frac{1}{g_l} \left(\frac{\partial a}{\partial y_l}\right)^2 \qquad \text{und} \qquad \sigma_b^2 = \sum_{l=1}^{n} \frac{1}{g_l} \left(\frac{\partial b}{\partial y_l}\right)^2$$

Die partiellen Ableitungen in diesen Gleichungen können nun aus (4.16) berechnet werden, und es ergibt sich

$$\sigma_a^2 = \sum_{l=1}^{n} \frac{1}{g_l} \left[\frac{1}{D}\left(\alpha_{22} g_l - \alpha_{12} g_l x_l\right)\right]^2$$

und

$$\sigma_b^2 = \sum_{l=1}^{n} \frac{1}{g_l} \left[\frac{1}{D}\left(-\alpha_{12} g_l + \alpha_{11} g_l x_l\right)\right]^2$$

Die nun folgenden Umformungen sind elementar:

$$\sigma_a^2 = \frac{1}{D^2} \sum_{l=1}^{n} g_l \left[\alpha_{22} - \alpha_{12} x_l\right]^2$$

$$= \frac{1}{D^2} \sum_{l=1}^{n} \sum_{k=1}^{n} \sum_{k'=1}^{n} g_l g_k g_{k'} \left[x_k^2 x_{k'}^2 - 2 x_l x_k^2 x_{k'} + x_l^2 x_k x_{k'}\right]$$

Durch geeignete Umbenennung der Summenindizes $l; k; k'$ ergibt sich daraus

$$\sigma_a^2 = \frac{1}{D^2} \sum_{l,k,k'} g_l g_k g_{k'} x_k^2 \left[x_{k'}^2 - x_{k'} x_l\right] = \frac{1}{D^2} \sum_{k=1}^{n} g_k x_k^2 \left[\sum_{l=1}^{n} g_l \sum_{k'=1}^{n} g_{k'} x_{k'}^2 - \left(\sum_{l=1}^{n} g_l x_l\right)^2\right]$$

Der Ausdruck in der eckigen Klammer in der letzten Zeile ist aber offenbar die Determinante D der Normalmatrix, und es ergibt sich schlußendlich

$$\sigma_a^2 = \frac{1}{D^2} D \sum_{k=1}^{n} g_k x_k^2 = \frac{\alpha_{22}}{D},$$

also dasselbe Ergebnis wie aus der Kovarianzmatrix (4.17), was zu beweisen war.

Eine ganz ähnliche Rechnung kann auch für σ_b durchgeführt werden.

4.4.2 Das Programm LFIT.

Quelle: [9], S.513ff, [10], S. 674ff mit Änderungen.

INPUT-Parameter:

X( ), Y( ): Koordinaten der Stützpunkte.

SIG( ): Standardabweichungen der y-Werte.

NDATA: Anzahl der Stützpunkte.

MA: Anzahl der Terme der linearen Modellfunktion.

OUTPUT-Parameter:

A( ): Feld mit den optimierten Fit-Parametern.

YF( ): Feld mit den Funktionswerten der Ausgleichskurve an den gegebenen x-Werten.

COVAR(,): Kovarianzmatrix.

CHISQ: die gewichtete Fehlersumme χ^2.

von LFIT benötigte Prozeduren:

FUNCS: In dieser Prozedur werden die in der Modellfunktion verwendeten Basisfunktionen definiert (s. Glg. (4.12)):

```
void funcs(double x, double afunc[], int ma)
{  ..... Definition lokaler Variabler .....
   afunc[1]=   .....  phi_{1}(x);
     .
     .
   afunc[ma] = .....  phi_{m}(x);
}
```

LUDCMP bzw. LUBKSB: Prozeduren zur Lösung des linearen Gleichungssystems [Glg.en (4.13),(4.14)] bzw. zur Invertierung der Normalmatrix, s. Kap. 2.

Programmstruktur:

1. Berechnung der Matrixkoeffizienten α_{ij} bzw. der Komponenten β_i des inhomogenen Vektors gemäß (4.13) und (4.14). Abspeicherung dieser Größen auf NORMAL(,) und BETA( ). Gemäß Glg.(4.15) stellt NORMAL gleichzeitig die Normalmatrix des Problems dar.

2. Berechnung der optimierten Fit-Parameter sowie der Kovarianzmatrix mittels der Prozeduren LUDCMP und LUBKSB.

3. Berechnung der Funktionswerte der Ausgleichskurve für die gegebenen x-Koordinaten und Berechnung der minimalen Fehlerquadratsumme.

Struktogramm 11 — LFIT(X,Y,SIG,NDATA,MA,A,YF,COVAR,CHISQ)

```
I=1(1)MA
    J=1(1)MA
        NORMAL(I,J):=0.0
    BETA(I):=0.0
K=1(1)NDATA
    FUNCS(X(K),AFUNC,MA)
    G:=1.0/SIG(K)/SIG(K)
    I=1(1)MA
        J=1(1)I
            NORMAL(I,J):=NORMAL(I,J) + G*AFUNC(I)*AFUNC(J)
        BETA(I):=BETA(I) + G*Y(K)*AFUNC(I)
I=2(1)MA
    J=1(1)I-1
        NORMAL(J,I):=NORMAL(I,J)
LUDCMP(NORMAL,MA,INDX,D,KHAD)
LUBKSB(NORMAL,MA,INDX,BETA,LOES)
J=1(1)MA
    A(J):=LOES(J)
CHISQ:=0.0
K=1(1)NDATA
    FUNCS(X(K),AFUNC,MA)
    G:=1.0/SIG(K)/SIG(K)
    SUM:=0.0
    J=1(1)MA
        SUM:=SUM + A(J)*AFUNC(J)
    YF(K):=SUM
    CHISQ:=CHISQ + G*(Y(K)-SUM)*(Y(K)-SUM)
J=1(1)MA
    I=1(1)MA
        BETA(I):=0.0
    BETA(J):=1.0
    LUBKSB(NORMAL,MA,INDX,BETA,LOES)
    I=1(1)MA
        COVAR(I,J):=LOES(I)
(return)
```

Anmerkungen zu LFIT:

1. Das Programm verlangt die Eingabe der Standardabweichungen σ_k der einzelnen y-Werte (das Feld SIG). Dazu gibt es, wie bereits im Abschnitt 4.3.3 dieses Kapitels erläutert, mehrere Möglichkeiten:

   - Stastistik der Stützpunkte unbekannt $\rightarrow$ alle $\sigma_k = 1$.
     In diesem Fall ist natürlich keine Aussage über die Varianz des Fits bzw. über die Statistik der optimierten Modellparameter möglich.
   - Statistik der normalverteilten Stützpunkte wird beschrieben durch eine typische Standardabweichung σ mit $\sigma_k \approx$ alle σ_k.
   - Die Stützpunkte sind Poisson-verteilt $\rightarrow \sigma_k \approx \sqrt{y}_k$.

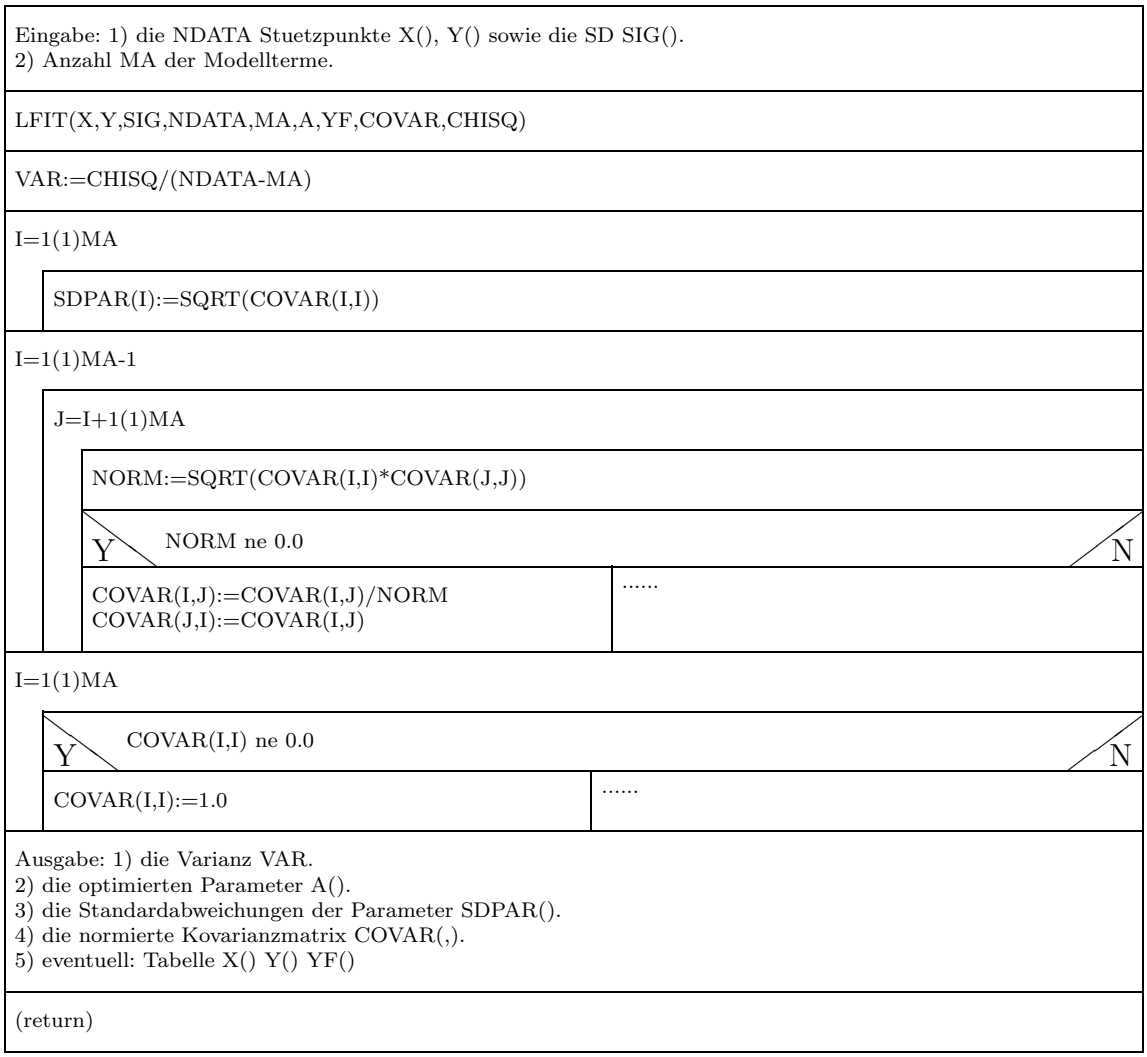
2. Das Programm liefert neben den optimierten Fit-Parametern auch noch die gewichtete Fehlersumme χ^2 und die Kovarianzmatrix. Daraus können bei Bedarf die Varianz des Fits sowie die Standardabweichungen bzw. die Korrelationskoeffizienten der gefitteten Parameter berechnet werden.

3. Es soll hier auch erwähnt werden, daß viele angebotene LSQ-Programme (z. B. die in den *Numerical Recipes* [9] und [10]) eine in der Praxis sehr brauchbare Option enthalten: es kann beim Aufruf des LSQ-Programms entschieden werden, welche der insgesamt MA Parameter der Modellfunktion tatsächlich optimiert (gefitted) werden sollen (Fit-Parameter), und welche während des gesamten LSQ-Prozesses unverändert bleiben sollen (fixe Parameter).

4.4.3 Anwendung von LFIT.

Die Anwendung der Prozedur LFIT kann z.B. im Rahmen des folgenden Hauptprogrammes erfolgen:

Programmschema — :



Ein Testbeispiel.

Für diesen Test wurden 101 Stützpunkte berechnet, die um ihre jeweiligen Erwartungswerte normalverteilt sind, wobei ein konstantes σ für alle Punkte angenommen wurde.

Die Erwartungswerte aller Punkte liegen exakt auf der Polynomkurve dritten Grades

$$y(x) = 0.5 - x - 0.2x^2 + 0.1x^3 \quad .$$

Die ideale Modellfunktion für dieses Problem ist also ein Polynom dritten Grades mit den Parametern $a_1, \ldots, a_4$:

$$f(x; \mathbf{a}) = \sum_{j=1}^{4} a_j x^{j-1}$$

Dementsprechend lautet die von LFIT aufgerufene Prozedur FUNCS z.B. in C wie folgt:

```
void funcs(double x, double afunc[], int ma)
{ int i;
  afunc[1]=1.0;
  for(i=2;i<=ma;i++) afunc[i]=afunc[i-1]*x;
}
```

Für den ersten Datensatz wurde eine Normalverteilung mit $\sigma = 0.1$ simuliert.

Interpretation der Tabelle 4.1 (a):

Je nach der Anzahl der Modellterme beträgt die Anzahl der Freiheitsgrade zwischen 96 und 100. Dementsprechend sollte sich für ein gutes Modell gemäß (4.10) eine Varianz zwischen ca. 0.86 und ca. 1.14 ergeben.

Wegen der relativ großen Streuung der Stützpunkte [vgl. Abb.4.6 (a)] ist eine Unterscheidung darüber, ob das ideale Modell ein Polynom 2. oder 3. Grades ist, nicht möglich! Diese Unsicherheit wirkt sich auch darin aus, daß die σ's der gefitteten Parameter ab MA=4 oft größer sind als die Beträge der entsprechenden Parameter-Mittelwerte, die auch für das an und für sich richtige Polynom 3. Grades sehr schlecht zu den simulierten Parametern a_j^s passen:

$$\begin{array}{ll} a_1^s = 0.5 & a_1^{Fit} = 0.5097 \\ a_2^s = -1.0 & a_2^{Fit} = -1.1152 \\ a_3^s = -0.2 & a_3^{Fit} = -0.0517 \\ a_4^s = 0.1 & a_4^{Fit} = 0.0543 \end{array}$$

Die Situation wird natürlich viel besser, wenn die Stützpunkte eine bessere Statistik haben. Für den zweiten Datensatz wurde angenommen: $\sigma = 0.025$.

Interpretation der Tabelle 4.1 (b):

Hier ist die Bestimmung des idealen Modells sehr viel leichter. Die Polynome nullten, ersten und zweiten Grades haben eine zu hohe Varianz, und erst ab MA$\leq$4 sind die Modellfunktionen als gut zu bezeichnen. Da aber für MA$>$4 einige Fit-Parameter 'in ihren σ's untergehen', ist das Polynom dritten Grades die beste Modellfunktion [s. Abb.4.7 (b)].

Tabelle 4.1 (a):

MA	Varianz	optimierte Parameter	Anmerkungen
1	37.034	$a_1 = -0.5652 \pm 0.0100$	schlechtes Modell
2	1.138	$a_1 = 0.4574 \pm 0.0198$	an der Grenze
		$a_2 = -1.0226 \pm 0.0171$	
3	1.032	$a_1 = 0.5307 \pm 0.0293$	
		$a_2 = -1.2449 \pm 0.0676$	gutes Modell
		$a_3 = 0.1111 \pm 0.0327$	
4	1.035	$a_1 = 0.5097 \pm 0.0384$	
		$a_2 = -1.1152 \pm 0.1670$	gutes Modell,
		$a_3 = -0.0517 \pm 0.1945$	schlechte Statistik
		$a_4 = 0.0543 \pm 0.0639$	s. Abb.4.6
5	1.046	$a_1 = 0.5134 \pm 0.0469$	
		$a_2 = -1.1543 \pm 0.3284$	gutes Modell
		$a_3 = 0.0372 \pm 0.6726$	schlechte Statistik
		$a_4 = -0.0151 \pm 0.5065$	
		$a_5 = 0.0174 \pm 0.1256$	

Tabelle 4.1 (b):

MA	Varianz	optimierte Parameter	Anmerkungen
1	598.559	$a_1 = -0.5639 \pm 0.0025$	schlechtes Modell
2	2.883	$a_1 = 0.4773 \pm 0.0049$	schlechtes Modell
		$a_2 = -1.0413 \pm 0.0043$	
3	1.204	$a_1 = 0.5472 \pm 0.0073$	
		$a_2 = -1.2530 \pm 0.0169$	schlechtes Modell
		$a_3 = 0.1059 \pm 0.0082$	
4	0.899	$a_1 = 0.5128 \pm 0.0096$	gutes Modell
		$a_2 = -1.0413 \pm 0.0417$	Fit-Parameter haben
		$a_3 = -0.1600 \pm 0.0486$	günstige σ's.
		$a_4 = 0.0886 \pm 0.0160$	siehe Abb.4.6
5	0.908	$a_1 = 0.5141 \pm 0.0117$	gutes Modell,
		$a_2 = -1.0548 \pm 0.0821$	aber manche Fit-Parameter
		$a_3 = -0.1294 \pm 0.1681$	haben sehr schlechte Statistik.
		$a_4 = 0.0647 \pm 0.1266$	'mixing of parameters'
		$a_5 = 0.0060 \pm 0.0314$	

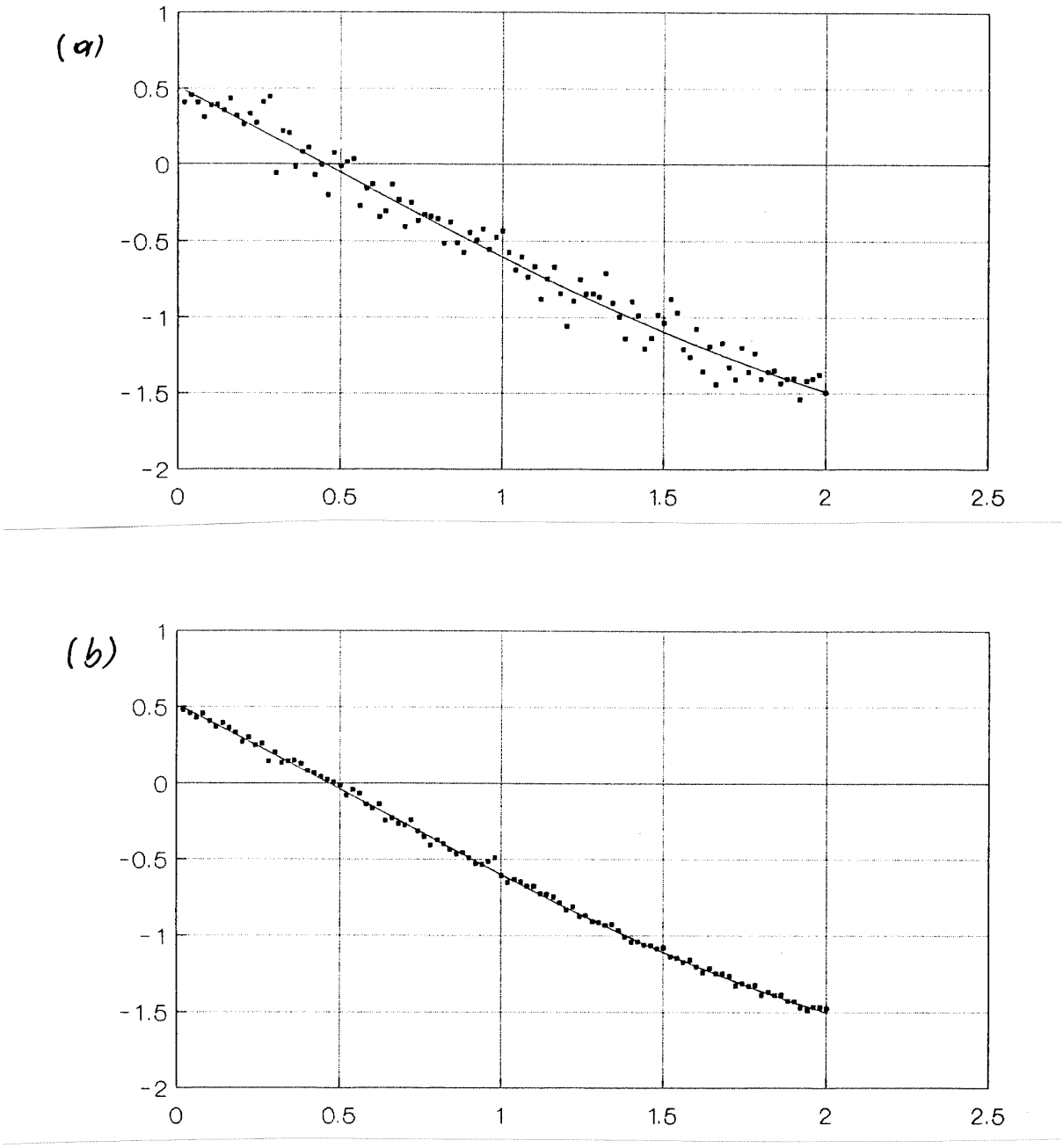


Abbildung 4.6: Lineare LSQ-Auswertung simulierter Datenwerte: (a) $\sigma = 0.1$, (b) $\sigma = 0.025$.

4.5 Modellfunktionen mit nicht-linearen Parametern.

4.5.1 Was sind nicht-lineare Parameter?

Setzt man die Modellfunktion

$$f(x; a, b) = a \cdot e^{-bx}$$

in die LSQ-Grundgleichung (4.1) ein, so erhält man

$$\chi^2 = \sum_{k=1}^{n} g_k \left[y_k - a \cdot e^{-bx_k} \right]^2 \quad \rightarrow \quad \text{Minimum!}$$

Geht man nun wie gewohnt vor, und differenziert nach dem Parameter a, so erhält man die bzgl. a lineare Gleichung

$$a \cdot \sum_{k=1}^{n} g_k e^{-2bx_k} = \sum_{k=1}^{n} g_k y_k e^{-bx_k} \quad .$$

Eine Ableitung nach b führt hingegen zu der nicht-linearen Gleichung

$$a \cdot \sum_{k=1}^{n} g_k x_k e^{-2bx_k} = \sum_{k=1}^{n} g_k x_k y_k e^{-bx_k} \quad .$$

Dementsprechend nennt man a einen *linearen* und b einen *nicht-linearen* Parameter der Modellfunktion $f(x)$.

Aus diesem einfachen Beispiel ist bereits zu ersehen, daß man bei nicht-linearen Modellfunktionen mit dem im Abschnitt 4.4 beschriebenen Verfahren nicht weiterkommt. Eine Möglichkeit der Behandlung nicht-linearer Modelle im Rahmen der LSQ Approximation stellt der im Abschnitt 4.5.3 behandelte *Gauss-Newton Formalismus* dar.

Vorher soll aber noch gezeigt werden, wie man bestimmte nicht-lineare Modellfunktionen auf einfache Weise linearisieren kann.

4.5.2 Linearisierung nicht-linearer Probleme.

Eine in der Praxis häufig vorkommende Meßpunkt-Verteilung ist in der Abb.4.7 dargestellt. Die $(x_k \mid y_k)$-Werte gehorchen offenbar einem *Exponentialgesetz*. Ihre Verteilung in einem halblogarithmischen $(x \mid \ln y)$-Koordinatensystem ist daher annähernd *linear*.

Für die so verteilten Punkte kommt als Modellfunktion offenbar

$$f(x; a, \lambda) = a \cdot e^{-\lambda x}$$

in Betracht, die im halblogarithmischen System die lineare Form

$$\ln y = \ln a - \lambda x$$

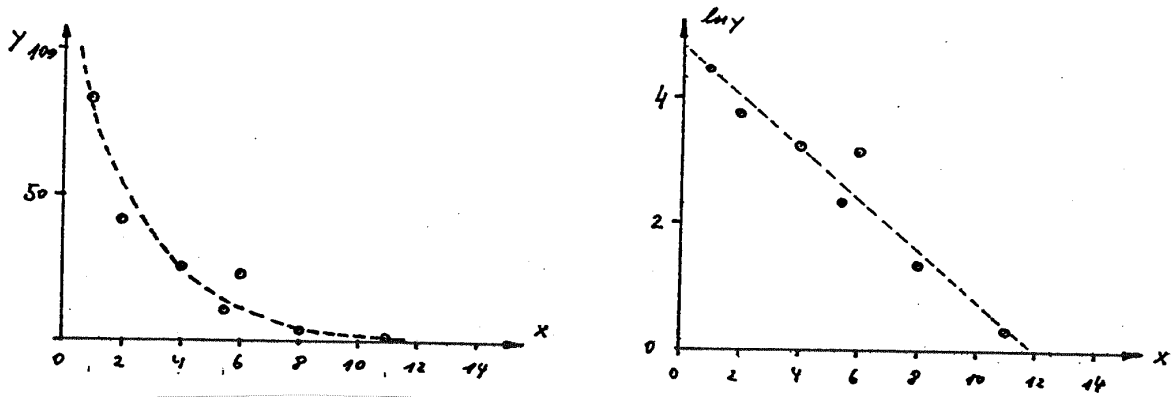


Abbildung 4.7: Nicht-lineare Modellfunktion (Exponentialfunktion) im normalen und halblogarithmischen Koordinatensystem.

hat. Das lineare Gleichungssystem für die beiden Fit-Parameter lautet gemäß den Gleichungen (4.13) und (4.14):

$$\begin{pmatrix} n & \sum_k x_k \\ \sum_k x_k & \sum_k x_k^2 \end{pmatrix} \cdot \begin{pmatrix} \ln a \\ -\lambda \end{pmatrix} = \begin{pmatrix} \sum_k \ln y_k \\ \sum_k x_k \ln y_k \end{pmatrix} \quad ,$$

wobei die Gewichtsfaktoren $g_k = 1$ gesetzt wurden!
Dieses System ist einfach zu lösen und man erhält

$$\begin{aligned} a &= \exp\left[\left(\sum \ln y_k \cdot \sum x_k^2 - \sum x_k \cdot \sum x_k \ln y_k\right)/D\right] \\ \lambda &= -\left(n \cdot \sum x_k \ln y_k - \sum x_k \cdot \sum \ln y_k\right)/D \\ D &= n \cdot \sum x_k^2 - \left(\sum x_k\right)^2 \end{aligned} \tag{4.18}$$

Dazu ein Beispiel: Die in Abb.4.7 dargestellten 7 Punkte haben die Koordinaten

x:	1.	2.	4.	5.5	6.	8.	11.
y:	83.2	41.7	25.1	10.5	22.9	3.8	1.4

Daraus erhält man unter Anwendung von (4.18)

$$a = 118.90 \quad \text{und} \quad \lambda = 0.398 \quad .$$

Die Ausgleichskurve lautet demnach

$$f(x) = 118.90\, e^{-0.398x}$$

(siehe die gestrichelten Kurven in Abb.4.7), und als Summe der Fehlerquadrate erhält man

$$\chi^2_{min} = 307.3 \quad .$$

Anmerkungen:

1. Diese 'minimale Fehlersumme' ist natürlich nur das Minimum der Summe der quadratischen Abweichungen zwischen den $\ln y_k$ und den $\ln f(x_k)$!

2. Ein großer Nachteil dieser Linearisierungs-Methode besteht auch darin, daß die Angabe von statistischen Gewichtsfaktoren meist nicht möglich ist.

3. Die exponentielle Modellfunktion ist ein gutes Beispiel für *physikalisch relevante* Modellparameter: so könnte $a \cdot \exp(-\lambda x)$ eine radioaktive Zerfallskurve mit a als Anfangsaktivität und λ als Zerfallskonstante darstellen.

In [7], S.330ff findet man eine Reihe weiterer Beispiele zur Linearisierung einfacher nicht-linearer Modellfunktionen.

Abschließend noch ein Hinweis: vermeiden Sie in Ihren Modellfunktionen 'versteckte Korrelationen' zwischen Fit-Parametern. So ist z. B. das Modell

$$f(x; a, b, c) = a\,\mathrm{e}^{-bx+c}$$

fehlerhaft, weil die Parameter a und c numerisch nicht getrennt werden können:

$$f(x; a, b, c) = \underbrace{a\,\mathrm{e}^{c}}_{=\text{ nur 1 Parameter}} \cdot\ \mathrm{e}^{-bx}\,.$$

4.5.3 Das Gauss-Newton- (GN-)Verfahren.

In all jenen Fällen, in denen eine Linearisierung der Modellfunktion im Sinne des Abschnitts 4.5.2 nicht möglich ist oder nicht gewünscht wird, empfiehlt sich die folgende Vorgangsweise:

Ausgangspunkt ist die völlig allgemeine Modellfunktion

$$f(x; a_1, a_2, \ldots, a_q) \tag{4.19}$$

mit den Parametern $\mathbf{a} \equiv a_1, a_2, \ldots, a_q$.
Die Minimalisierung der gewichteten Fehlersumme

$$\chi^2 = \sum_{k=1}^{n} g_k \left[y_k - f(x_k; \mathbf{a})\right]^2$$

kann nun durch den folgenden *iterativen Prozess* geschehen:

1. Man wählt einen Satz von Anfangswerten (*guessed values*) für die Parameter:
$$\mathbf{a} = \mathbf{a}^0 \quad .$$

2. Man macht eine *Taylor-Entwicklung der Modellfunktion* bzgl. der Parameter an der Stelle $\mathbf{a}^0$ und bricht nach dem linearen Term ab:
$$f(x; \mathbf{a}) \approx f(x; \mathbf{a}^0) + \sum_{l=1}^{q} \left(\frac{\partial f(x; \mathbf{a})}{\partial a_l}\right)_{\mathbf{a}=\mathbf{a}^0} \cdot (a_l - a_l^0) \quad .$$

3. Diese (näherungsweise) *linearisierte Modellfunktion* setzt man nun in die LSQ-Grundgleichung ein:
$$\chi^2 = \sum_{k=1}^{n} g_k \left[y_k - f(x_k; \mathbf{a}^0) - \sum_{l=1}^{q} \left(\frac{\partial f(x_k; \mathbf{a})}{\partial a_l}\right)_{\mathbf{a}=\mathbf{a}^0} \cdot (a_l - a_l^0)\right]^2 \quad ,$$
wobei für die folgenden Gleichungen die Abkürzungen
$$df_{k,l} \equiv \left(\frac{\partial f(x_k; \mathbf{a})}{\partial a_l}\right)_{\mathbf{a}=\mathbf{a}^0} \quad \text{sowie} \quad f_k \equiv f(x_k; \mathbf{a}^0)$$
verwendet werden.

4. Erst jetzt wird χ^2 nach den Modellparametern abgeleitet und die Ableitungen werden Null gesetzt:
$$\frac{\partial \chi^2}{\partial a_j} = -2 \sum_{k=1}^{n} g_k \left[y_k - f_k - \sum_{l=1}^{q} df_{k,l}(a_l - a_l^0)\right] \cdot df_{k,j} = 0$$
für $j = 1, \ldots, q$. Als Ergebnis erhält man ein lineares, inhomogenes Gleichungssystem für die q Ausdrücke $(a_l - a_l^0)$:
$$A \cdot (\mathbf{a} - \mathbf{a}^0) = \beta$$
mit
$$A = [\alpha_{ij}] \quad \alpha_{ij} = \sum_{k=1}^{n} g_k df_{k,i} df_{k,j} \quad \text{und} \quad \beta_i = \sum_{k=1}^{n} g_k (y_k - f_k) df_{k,i} \quad . \tag{4.20}$$

5. Die Lösungen dieses Systems, $(a_l - a_l^0)$, kann man nun als Differenzen zwischen den *guessed values* und *verbesserten Werten* der gesuchten Modellparameter ansehen:

$$a_l - a_l^0 \equiv \Delta a_l \rightarrow a_l^1 = a_l^0 + \Delta a_l \quad (l = 1, \ldots, q) \quad .$$

6. Mit diesem *verbesserten Satz von Parametern* wird nun der Ablauf $(2 \rightarrow 3 \rightarrow 4 \rightarrow 5)$ wiederholt. Auf diese Weise (Konvergenz des Verfahrens vorausgesetzt) ist es möglich, die Modellparameter *iterativ zu verbessern*:

$$a_l^t = a_l^{t-1} + \Delta a_l^t \quad (t = 1, 2, \ldots) \tag{4.21}$$

7. Wie bei jedem iterativen Verfahren braucht man auch hier ein *Abbruchkriterium*. In der einschlägigen Literatur werden eine Reihe von möglichen Kriterien diskutiert. Eines von ihnen (nicht immer das beste!) lautet:

   Die Iteration wird abgebrochen, wenn alle Parameter die relative Genauigkeitsabfrage

$$\mid a_l^t - a_l^{t-1} \mid < \mid a_l^t \mid \cdot \epsilon \tag{4.22}$$

   erfüllen (ϵ = Genauigkeit) oder wenn eine maximale Zahl von Iterationen überschritten wird.

8. Nach Erreichen der gewünschten Genauigkeit wird die Normalmatrix des Problems berechnet, deren i, j-ter Koeffizient gemäß (4.6) lautet:

$$\frac{1}{2}\frac{\partial^2 \chi^2}{\partial a_i \partial a_j} \quad .$$

   Mit einer nicht-linearen Modellfunktion erhält man

$$\frac{1}{2}\frac{\partial^2 \chi^2}{\partial a_i \partial a_j} = \sum_{k=1}^{n} g_k \left[\frac{\partial f(x_k; \mathbf{a})}{\partial a_i} \frac{\partial f(x_k; \mathbf{a})}{\partial a_j} - (y_k - f(x_k; \mathbf{a})) \frac{\partial^2 f(x_k; \mathbf{a})}{\partial a_i \partial a_j} \right] \quad .$$

   In den meisten Programmen der nicht-linearen LSQ-Approximation wird nun der zweite Term im obigen Klammerausdruck vernachlässigt. Dies ist dadurch gerechtfertigt, daß im Falle einer erfolgreichen Iteration die Differenz $[y_k - f(x_k; \mathbf{a})]$ i.a. klein sein wird. Man erhält also mit der auf Seite 115 definierten Abkürzung

$$\frac{1}{2}\frac{\partial^2 \chi^2}{\partial a_i \partial a_j} \approx \sum_{k=1}^{n} g_k df_{k,i} df_{k,j} \quad . \tag{4.23}$$

   In dieser Näherung ist also wieder (wie beim linearen Modell) die Normalmatrix identisch mit der Koeffizientenmatrix des LSQ Systems!

9. Invertierung der Normalmatrix = Kovarianzmatrix. Diese enthält wieder die statistische Information über den Fit und die Fit-Parameter.

4.5.4 Konvergenzprobleme beim GN-Verfahren. Die Variation von Marquardt.

Eine primäre Forderung für die praktische Verwendbarkeit numerischer Verfahren, insbesondere iterativer Verfahren, ist die nach deren *Stabilität*. Es muß gewährleistet sein, daß das Verfahren *konvergiert*, und zwar auch dann, *wenn die Anfangswerte der Iteration ungünstig gewählt wurden*.

Im folgenden Testbeispiel wird nun gezeigt, daß das GN-Verfahren diese Forderung keineswegs erfüllt.

Ausgangspunkt für diesen Test ist die Funktion

$$f(x) = 10e^{-3x} + 5e^{-x/2} \quad .$$

Dies entspricht der parametrisierten Funktion

$$f(x; a_1, a_2, a_3, a_4) = a_1 e^{-a_3 x} + a_2 e^{-a_4 x} \tag{4.24}$$

mit den (exakten) Parametern

$$a_1 = 10. \quad a_2 = 5. \quad a_3 = 3. \quad a_4 = 0.5 \quad .$$

Diese Funktion kann dazu verwendet werden, einen *Testdatensatz* für den GN-Algorithmus zu liefern:

```
10 Datenwerte:
           X              Y=F(X)
 1    .1000000E+01    .3530524E+01
 2    .2000000E+01    .1864185E+01
 3    .3000000E+01    .1116885E+01
 4    .4000000E+01    .6767378E+00
 5    .5000000E+01    .4104280E+00
 6    .6000000E+01    .2489355E+00
 7    .7000000E+01    .1509869E+00
 8    .8000000E+01    .9157819E-01
 9    .9000000E+01    .5554498E-01
10    .1000000E+02    .3368973E-01
```

Faßt man diese Daten als Stützpunkt-Koordinaten für eine LSQ-Problem mit (4.24) als Modellfunktion auf, so sollte das Iterationsergebnis des GN-Prozesses natürlich lauten:

$$a_1 \to 10. \quad a_2 \to 5. \quad a_3 \to 3. \quad a_4 \to 0.5$$

Im folgenden wird nun das Konvergenzverhalten eines Programms, das auf dem im Abschnitt 4.5.3 erläuterten Algorithmus beruht, untersucht. Zu diesem Zweck wurde von verschiedenen *guessed values* für die a_3 und a_4 ausgegangen, während für a_1 und a_2 stets die Startwerte 9. und 4. gewählt wurden.
Die Ergebnisse dieses Tests sind in der Abb.4.8 zusammengefaßt.

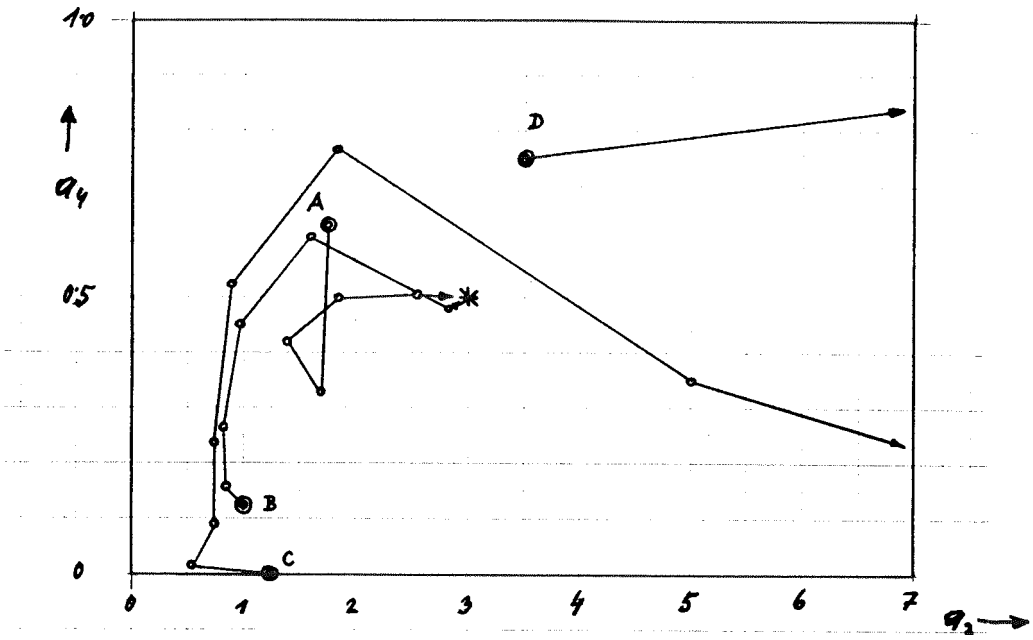


Abbildung 4.8: Konvergenzverhalten des Gauss-Newton Algorithmus.

Diese Testergebnisse sind natürlich sehr unbefriedigend! Es ist nämlich in der Praxis äußerst schwierig, den Konvergenzbereich abzuschätzen. Die Methode, die *guessed values* für die Parameter-Startwerte solange zu variieren, bis die Iteration endlich einmal klappt, ist inakzeptabel!

Einen Ausweg aus diesem Dilemma wurde 1963 von D.W. Marquardt gefunden [1]. Im folgenden werden die Ergebnisse seiner Untersuchungen präsentiert, ohne auf die mathematische Beweisführung einzugehen.

Ausgangspunkt für die *Variation des GN-Verfahrens nach Marquardt* ist das lineare Gleichungssystem (4.20):

$$\begin{pmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1q} \\ \alpha_{12} & \alpha_{22} & \cdots & \alpha_{2q} \\ . & . & & . \\ . & . & & . \\ . & . & & . \\ \alpha_{1q} & \alpha_{2q} & \cdots & \alpha_{qq} \end{pmatrix} \cdot \begin{pmatrix} \Delta a_1 \\ \Delta a_2 \\ . \\ . \\ . \\ \Delta a_q \end{pmatrix} = \begin{pmatrix} \beta_1 \\ \beta_2 \\ . \\ . \\ . \\ \beta_q \end{pmatrix}$$

In der Matrizen-Schreibweise sieht das so aus:

$$A \cdot \Delta\mathbf{a} = \beta$$

Gleichungssysteme dieser Art sind also im Rahmen des GM-Algorithmus zu lösen. *Nach der Variation von Marquardt löst man statt dessen das System*

$$(A + \lambda D) \cdot \Delta\mathbf{a} = \beta \quad , \tag{4.25}$$

wobei D eine Diagonalmatrix der Form

$$d_{ii} = \alpha_{ii} \quad (i = 1, \ldots, q) \tag{4.26}$$

[1] D.W. Marquardt, J. Soc. Ind. Appl. Math. **11**,431 (1963)

darstellt. Die Größe λ ist vorläufig noch unbestimmt.

Angenommen, der t-te Iterationsschritt während des GN-Prozesses habe die Fehlerquadratsumme χ_t^2 ergeben. Der darauffolgende Iterationsschritt liefere den Wert χ_{t+1}^2.
Es kann nun ohne weiteres geschehen, daß gilt:

$$\chi_{t+1}^2 > \chi_t^2 \quad .$$

Dies kann jedoch (vgl. den obigen Test, Startpunkt C) unter Umständen zu einer Divergenz des Iterationsprozesses führen.

Die allgemeingültige Aussage, die Marquardt streng mathematisch bewiesen hat, lautet nun:
Es ist immer möglich, die in (4.25) vorkommende Größe λ so zu wählen, daß gilt:

$$\chi_{t+1}^2 \leq \chi_t^2$$

Damit ist aber die Konvergenz des Verfahrens gesichert.

Um die Wirkung der Vergrößerung der (positiven) Größe λ zu demonstrieren, kann man von einem Gleichungssystem (4.25) ausgehen, wobei angenommen wird, daß das verwendete Modell nur 2 Parameter enthält:

$$\begin{pmatrix} \alpha_{11}(1+\lambda) & \alpha_{12} \\ \alpha_{12} & \alpha_{22}(1+\lambda) \end{pmatrix} \cdot \begin{pmatrix} \Delta a_1 \\ \Delta a_2 \end{pmatrix} = \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}$$

Die analytischen Lösungen dieses Systems lauten:

$$\Delta a_1(\lambda) = \frac{\beta_1\alpha_{22}(1+\lambda) - \beta_2\alpha_{12}}{\alpha_{11}\alpha_{22}(1+\lambda)^2 - \alpha_{12}^2} \quad \text{und} \quad \Delta a_2(\lambda) = \frac{\beta_2\alpha_{11}(1+\lambda) - \beta_1\alpha_{12}}{\alpha_{11}\alpha_{22}(1+\lambda)^2 - \alpha_{12}^2} \quad .$$

Das bedeutet:

- Für $\lambda \to 0$ erhält man die GN Methode.
- Für $\lambda \to \infty$ gehen Δa_1 und Δa_2 gegen Null!

Zusätzlich sieht man sofort, daß auch das Verhältnis der Korrektur-Komponenten, $\Delta a_1/\Delta a_2$, eine Funktion von λ ist. Das bedeutet, daß bei einer λ-Variation nicht nur die Korrekturen kleiner ('vorsichtiger') werden, sondern daß sich auch die Korrektur-Richtung ändert: die Umgebung des Ausgangspunktes wird abgesucht.

Man kann also zusammenfassen:

- *Eine Vergrößerung des Marquardt'schen Parameters λ führt zu einer Verkleinerung der Korrekturwerte für die Modellparameter und damit zu einer Verkleinerung der Konvergenzgeschwindigkeit.*
- *Andererseits garantiert jedoch eine (ausreichende) Vergrößerung von λ die monotone Abnahme der Fehlersummen*:

$$\chi_1^2 \geq \chi_2^2 \geq \chi_3^2 \cdots \geq \chi_t^2 \geq \chi_{t+1}^2 \cdots$$

Eine praktikable Methode, λ möglichst klein (um eine möglichst große Konvergenzgeschwindigkeit zu haben) und im Bedarfsfall doch groß genug zu halten (um die Konvergenz zu sichern), ist die sogenannte *Marquardt'sche Strategie*:

1. Start der Iteration mit einem kleinen (positiven, sonst beliebigen) Wert für λ (im folgenden Programm ist λ z.B. 0.0003).

2. Vor jedem neuen Iterationsschritt wird λ um einen fixen – ebenfalls beliebigen – Faktor verkleinert, um die Konvergenzgeschwindigkeit möglichst groß zu halten (im folgenden Programm um den Faktor 5).

3. Tritt der Fall $\chi^2_{t+1} > \chi^2_t$ auf, wird λ schrittweise um einen fixen Faktor (im folgenden Programm ebenfalls 5) erhöht, und die Auswertung des Gleichungssystems (4.25) wird solange wiederholt, bis $\chi^2_{t+1} \leq \chi^2_t$ erreicht wird. Erst dann wird

4. zum nächsten Iterationsschritt übergegangen.

Testet man den *GN-Algorithmus mit Marquardt-Variation*[2] unter Verwendung der Angaben des gegebenen Testproblems, so erhält man die in Abb.4.9 zusammengefaßten Ergebnisse. Wie man sieht, konvergiert das Verfahren *auch von jenen Startpunkten C und D aus*, bei welchen ohne die Marquardt-Variation eine Divergenz zu beobachten war (vgl. Abb.4.8). Für den Startpunkt D ist im folgenden auch der Rechenverlauf dargestellt, wobei jedes * eine Vergrößerung des Marquardt'schen Parameters λ bedeutet:

```
 Startpunkt D    OHNE MARQUARDT-VARIATION:
 ========================================
 t       chisq             a1             a2             a3             a4

   0   .368339E+01    .900000E+01    .400000E+01    .350000E+01    .750000E+00
                     Exekutionsabbruch wegen Exponenten-Overflow!

 Startpunkt D    MIT MARQUARDT-VARIATION:
 ========================================
 t       chisq             a1             a2             a3             a4

 0    .368339E+01       .900000E+01    .400000E+01    .350000E+01    .750000E+00
*
*
*
 1    .327945E+01       .116540E+02    .478102E+01    .326392E+01    .298864E+00
 2    .221123E+00       .139565E+02    .407836E+01    .262388E+01    .386921E+00
 3    .660544E-02       .776914E+01    .456550E+01    .253007E+01    .468859E+00
 4    .383322E-04       .750890E+01    .492223E+01    .264024E+01    .496633E+00
```

[2]Dieser Algorithmus wird in der Literatur auch als 'Levenberg-Marquardt Methode' bezeichnet

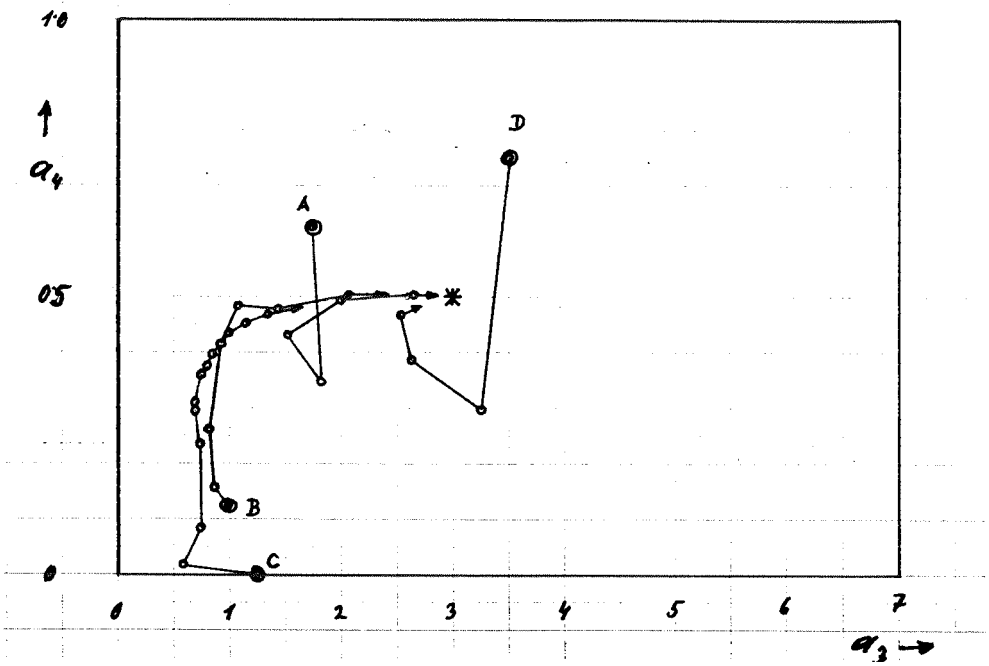


Abbildung 4.9: Konvergenzverhalten des Gauss-Newton-Marquardt Algorithmus.

```
*
 5    .708586E-05      .823696E+01    .497283E+01    .278430E+01    .498760E+00
*
 6    .320911E-05      .885081E+01    .498274E+01    .286453E+01    .499211E+00
 7    .315072E-05      .961475E+01    .499580E+01    .296028E+01    .499807E+00
 8    .101244E-06      .996144E+01    .499965E+01    .299644E+01    .499984E+00
 9    .147690E-10      .999937E+01    .499999E+01    .299994E+01    .500000E+00
10    .903999E-13      .100000E+02    .500000E+01    .300000E+01    .500000E+00
11    .340838E-13      .999999E+01    .500000E+01    .300000E+01    .500000E+00
```

4.5.5 Das Programm MRQMIN.

Quelle: [9], S.526f; [10], S. 683ff mit einigen Änderungen.

Das Programm MRQMIN (MaRQuardt MINimalisation) führt einen Iterationsschritt im Rahmen des Gauss-Newton-Marquardt Verfahrens durch.

INPUT-Parameter:

X( ),Y( ): Koordinaten der Stützpunkte.

SIG( ): Standardabweichungen der y-Werte.

NDATA: Anzahl der Stützpunkte.

MA: Anzahl der Parameter in der Modellfunktion.

A( ): Feld der Modellparameter (*guessed values*).

ALAMDA: Steuer-Parameter, Marquardt'scher Parameter λ.

OUTPUT-Parameter:

A( ): Feld der *optimierten* Fit-Parameter.

ALPHA(,): Koeffizientenmatrix des linearen Systems (4.20).

COVAR(,): Kovarianz-Matrix.

BETA(): inhomog. Vektor des linearen Systems (4.20).

CHISQ: gewichtete Fehlerquadratsumme χ^2.

OCHISQ: gewichtete Fehlersumme des vorherigen Iterationsschrittes.

ALAMDA: aktueller Marquardt'scher Parameter.

VMAR: logische Variable, die anzeigt, ob der Iterationsschritt im Sinne einer χ^2-Abnahme erfolgreich war (VMAR=false) oder nicht (VMAR=true).

Interne Felder:

NORMAL(,): Marquardt'sche Koeffizientenmatrix (4.25).

DA(): Korrekturvektor für die Parameter.

VEKTOR(), LOES() .

Benötigte Prozeduren:

MRQCOF: Berechnung der Matrix ALPHA und des Vektors BETA gemäß (4.19) sowie der gewichteten Fehlersumme CHISQ.

LUDCMP und LUBKSB: Lösung des linearen Gleichungssystems (4.25) bzw. Invertierung der Normalmatrix.

Programmstruktur:

1. Beginn der Iteration mit negativem ALAMDA. ALAMDA wird auf den Wert 0.0003[3] gesetzt, dann erfolgt der erste Aufruf von MRQCOF. Aus den dort berechneten Feldern ALPHA(,) und BETA( ) werden die Koeffizienten des Gleichungssytems (4.25) berechnet. Dieses System wird mittels LU-Decomposition gelöst.

2. Wenn ALAMDA Null ist (letzter Aufruf von MRQMIN!), wird durch Invertierung der Normalmatrix NORMAL(,) die Kovarianzmatrix COVAR(,) berechnet. Ende der Rechnung.

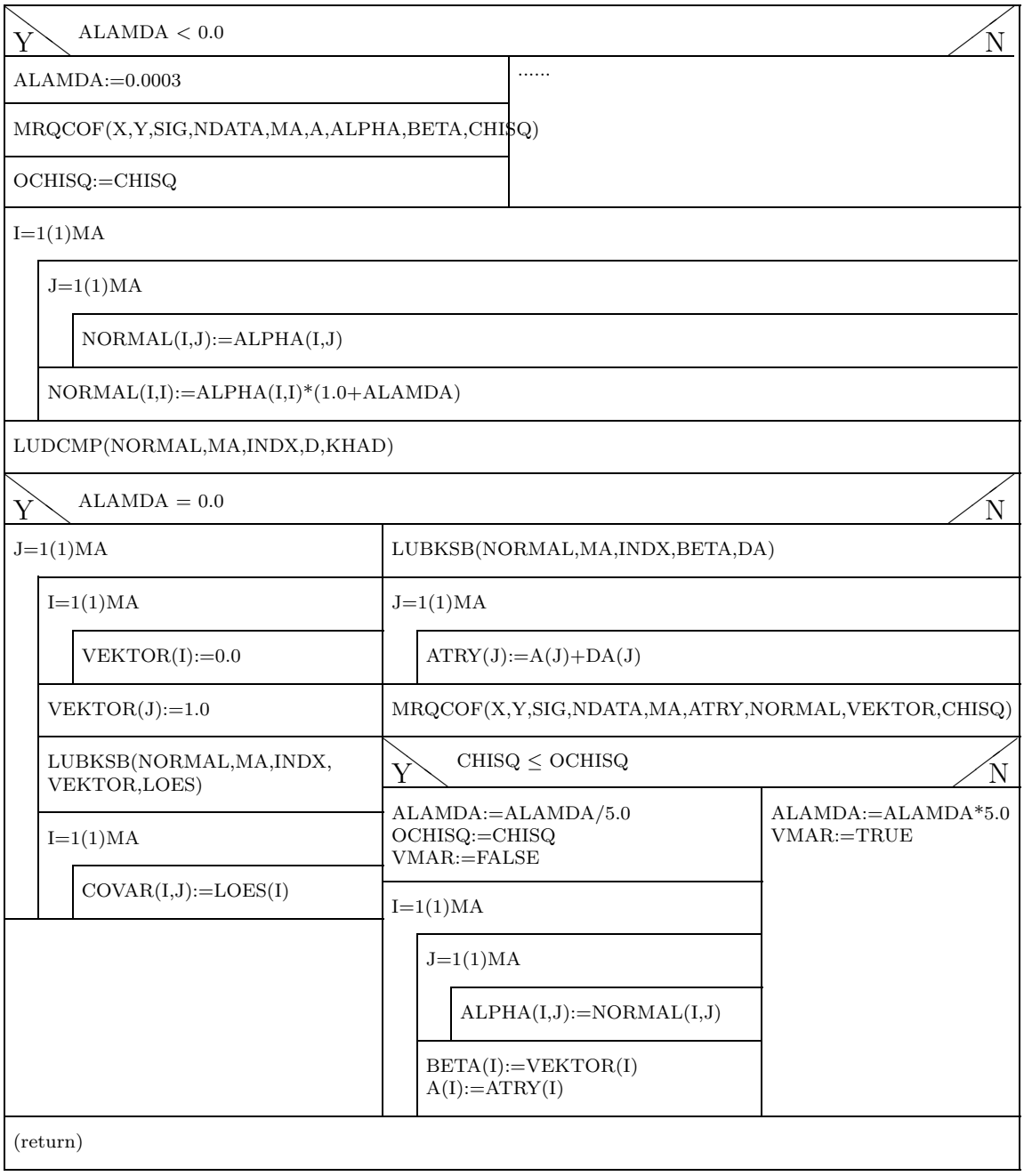
3. Wenn ALAMDA > Null, werden *versuchsweise* (ATRY) die verbesserten(?) Fit-Parameter berechnet. Neuerlicher Aufruf von MRQCOF.

[3]Dieser Startwert für λ ist ebenso wie der weiter unten angegebene Faktor (=5.) willkürlich und hat sich nur empirisch als praktikabel erwiesen.

4. Nun wird die neue, von MRQCOF gelieferte Fehlersumme CHISQ berechnet und mit der alten Fehlersumme OCHISQ verglichen:

   - CHISQ $\leq$ OCHISQ: Der Iterationsschritt war erfolgreich. Die logische Variable MVAR wird 'false' gesetzt, und ATRY( ) wird endgültig auf A( ) abgespeichert. ALAMDA wird durch den Faktor 5. dividiert.
   - CHISQ $>$ OCHISQ: Der Iterationsschritt war *nicht* erfolgreich. Die logische Variable wird 'true' gesetzt, und ALAMDA wird mit dem Faktor 5. multipliziert.

Struktogramm 12 — MRQMIN(X,Y,SIG,NDATA,MA,A,ALPHA,COVAR, BETA,CHISQ,OCHISQ,ALAMDA,VMAR)



4.5.6 Das Programm MRQCOF.

Quelle: [9], S.527f; [10], S. 687.

Das Programm MRQCOF (MaRQuardt COeFficients) berechnet die Matrix ALPHA und den Vektor BETA gemäß (4.20) sowie die gewichtete Fehlerquadratsumme χ^2.

INPUT-Parameter:

X( ),Y( ),SIG( ),NDATA,MA,A( ): siehe Beschreibung von MRQMIN, Abschnitt 4.5.5.

OUTPUT-Parameter:

ALPHA(,),BETA( ): Matrix ALPHA und Vektor BETA gemäß (4.20).

CHISQ: gewichtete Fehlerquadratesumme χ^2.

Benötigte Prozeduren:

FUNCS: Prozedur, die für das gegebene Argument X und die Modellparameter A( ) den Funktionswert Y der Modellfunktion sowie alle partiellen Ableitungen der Modellfunktion nach den Modellparametern berechnet. Diese MA Ableitungen werden im Feld DYDA( ) gespeichert.

Struktogramm 13 — MRQCOF(X,Y,SIG,NDATA,MA,A,ALPHA,BETA,CHISQ)

```
I=1(1)MA
    J=1(1)I
        ALPHA(I,J):=0.0
    BETA(I):=0.0
CHISQ:=0.0
K=1(1)NDATA
    FUNCS(X(K),A,YMOD,DYDA,MA)
    G:=1.0/SIG(K)/SIG(K)
    DY:=Y(K)-YMOD
    I=1(1)MA
        J=1(1)I
            ALPHA(I,J):=ALPHA(I,J) + G*DYDA(I)*DYDA(J)
        BETA(I):=BETA(I) + G*DY*DYDA(I)
    CHISQ:=CHISQ + G*DY*DY
I=2(1)MA
    J=1(1)I-1
        ALPHA(J,I):=ALPHA(I,J)
(return)
```

4.5.7 Anwendung von MRQMIN und MRQCOF

. — .

Eingabe: 1) die NDATA Stuetzpunkte X(), Y() sowie die Standardabweichungen SIG().
2) Anzahl MA der Modellparameter.
3) Vektor A() mit den guessed values fuer die Fit-Parameter.
4) Anzahl TMAX der maximalen Iterationsschritte.
5) relative Genauigkeit EPS der zu fittenden Parameter.

T:=1
ALAMDA:=–1.0

J=1(1)MA

AALT(J):=A(J)

MRQMIN(X,Y,SIG,NDATA,MA,A,ALPHA,COVAR,BETA,CHISQ,OCHISQ,ALAMDA,VMAR)

VMAR: Y / N

print '***' fuer Marquardt-Korrektur	print: T,CHISQ,A(...)
	DELMAX:=0.0
	J=1(1)MA
	DEL:=|(A(J)-AALT(J))/A(J)|
	DEL > DELMAX: Y / N
	DELMAX:=DEL | .....
	DELMAX < EPS: Y / N
	ALAMDA:=0.0 | J=1(1)MA
	MRQMIN(Parameter s.o.) | AALT(J):=A(J)
	1) Berechnung der Varianz. 2) Ber. der SD der Parameter. 3) Normierung der Kovar.matrix. 4) Ausgabe: ... (Ende der Rechnung) | T:=T+1

T > TMAX

Fehler: 'Genauigkeit nicht erreicht'

(Ende der Rechnung)

Ein Testbeispiel.

Gegeben sei ein radioaktives Präparat, das aus einem Gemisch von J radioaktiven Isotopen besteht. Beide Kernarten zerfallen, ausgehend von einer Anfangsaktivität A_j mit einer Halbwertzeit T_j, gemäß dem *radioaktiven Zerfallsgesetz*:

$$A_j = A_j e^{-\ln 2 \cdot t/T_j} \quad .$$

Die Gesamtaktivität der Quelle beträgt demnach

$$A(t) = \sum_{j=1}^{J} A_j e^{-\ln 2 \cdot t/T_j} \quad .$$

Die Messung geht nun wie folgt vor sich: Während einer fixen Zeitspanne Δ werden die von der Quelle herrührenden Zerfälle gezählt und das Ergebnis wird gespeichert. Danach wird die Zählung für eine zweite Zeitspanne Δ durchgeführt usw. Man erhält auf diese Weise eine Reihe von Zählwerten Z_k, wobei

$$Z_k = \int_{t=(k-1)\Delta}^{k\Delta} dt A(t) \quad (k = 1, 2, \ldots) \quad .$$

Ein solches Experiment ist natürlich ein typisches Zählexperiment d.h. die Meßwerte sind um ihre jeweiligen Erwartungswerte *poisson-verteilt* (s. Abschnitt 4.3.3).

Durch die Auswertung des obigen Integrals erhält man die Modellfunktion mit $2J$ Parametern

$$Z(k; A_1, \ldots, A_J, T_1, \ldots, T_J) = \sum_{j=1}^{J} \frac{A_j}{\ln 2} T_j \left(e^{+\Delta \ln 2/T_j} - 1 \right) e^{-\Delta \ln 2 \cdot k/T_j} \quad .$$

Die Ableitungen $\partial Z/\partial A_j$ bzw. $\partial Z/\partial T_j$ können daraus ohne Probleme berechnet werden, und die Prozedur FUNCS lautet in C-Version etwa folgendermaßen:

```
#define DELTA 15.0    // Konstante des Experiments
void funcs(double x, double a[], double *z, double dzda[], int ma)
// C-VERSION
{
  int   mterm,j,ind;
  double con,fac1,fac2,fac3,fac4;
  con=DELTA*log(2.0);
  mterm=ma/2;
  *z=0.0;
  for(j=1;j<=mterm;j++) {
    ind=mterm+j;
    fac1=con/a[ind];
    fac2=exp(fac1);
    fac3=fac2-1.0;
    fac4=exp(-fac1*x);

    dzda[j]=a[ind]*fac3*fac4/log(2.0);
    *z=*z + a[j]*dzda[j];
    dzda[ind]=a[j]/log(2.0)*fac4*(fac3*(1.0+fac1*x)-fac1*fac2);
  }
}
```

Wie Sie sehen, wird die Routine *funcs* für jeden x-Wert gesondert aufgerufen. Eine solche Vorgangsweise ist im Falle einer MATLAB-Realisierung

nicht optimal, weil die Fähigkeit dieser Sprache, sehr effektiv mit Vektoren umzugehen, nicht ausgenutzt wird. Besser ist es, wie im folgenden Beispiel dargestellt, die Eingabegröße x von vornherein als Vektor aller Werte x_i mit $i = 1, \ldots, NDATA$ vorzusehen:

```
  function [z,dzda] = funcs(x,a,ma,ndata);
% MATLAB-VERSION  x ist ein Vektor mit ndata Komponenten

  dzda=zeros(ma,ndata);
  delta=15.0;              % Messzeit
  con=delta*log(2);
  mterm=fix(ma/2);

    z=zeros(1,ndata);
    for j=1:mterm
      fac1=con/a(mterm+j);
      fac2=exp(fac1);
      fac3=fac2-1;
      fac4=exp(-fac1*x);

      dzda(j,:)=a(mterm+j)*fac3.*fac4/log(2);
      z=z+a(j)*dzda(j,:);
      dzda(mterm+j,:)=a(j)/log(2).*fac4.*(fac3*(1+fac1*x)-fac1*fac2);
    end
```

Nun die Angaben zu einem konkreten Beispiel:

Gemessen wurden die Zählraten eines aus zwei Komponenten bestehenden radioaktiven Präparates d.h. $J = 2$. Es wurden 40 Messungen durchgeführt, jede über einen Zeitraum von $\Delta = 15$ Sekunden.

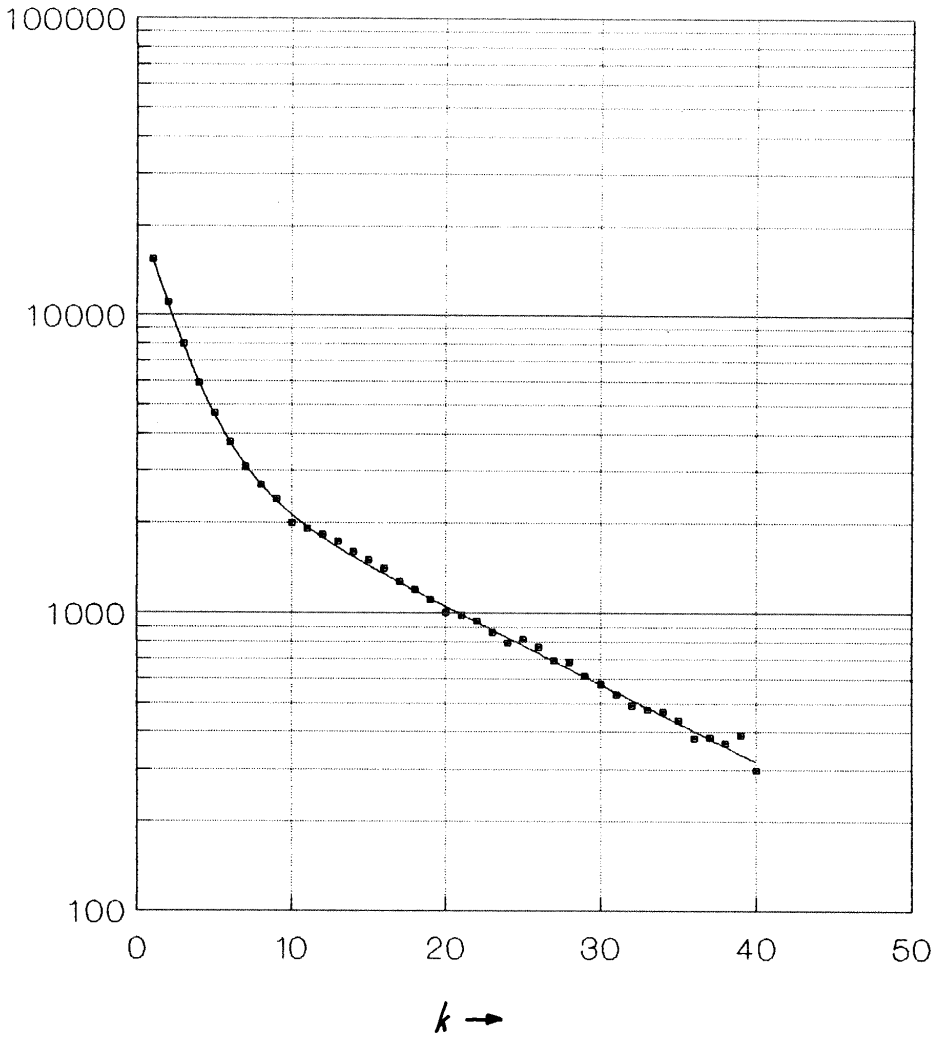
```
  40 Datenwerte:

   1  15376.0          21     981.0
   2  10903.0          22     939.0
   3   7950.0          23     857.0
   4   5865.0          24     790.0
   5   4653.0          25     814.0
   6   3721.0          26     766.0
   7   3089.0          27     691.0
   8   2683.0          28     681.0
   9   2396.0          29     614.0
  10   1992.0          30     576.0
  11   1910.0          31     529.0
  12   1820.0          32     488.0
  13   1726.0          33     472.0
  14   1600.0          34     464.0
  15   1495.0          35     434.0
```

```
 16   1410.0          36     380.0
 17   1271.0          37     382.0
 18   1197.0          38     365.0
 19   1106.0          39     387.0
 20   1004.0          40     296.0
```

Der Verlauf der Iteration mittels des Gauss-Newton-Marquardt-Verfahrens ist:

```
   t     chisq         A1        A2        T1        T2

   0   196876.304   2000.000   500.000   30.000   200.000
   1     1233.069    976.760   237.577   26.378   184.784
   2       45.645    998.414   229.228   22.949   172.341
   3       43.535   1005.360   226.315   23.159   173.250
   4       43.535   1005.458   226.349   23.153   173.245
   5       43.535   1005.457   226.348   23.153   173.246
Konvergenz wurde erreicht!

VARIANZ =  1.209  (sollte zwischen 0.764 und 1.236 liegen)

          Modellparameter        SD

  A1          1005.457          10.182
  A2           226.348           4.129
  T1(s)         23.153           0.353
  T2(s)        173.246           2.320

 Die Korrelationsmatrix:
            A1        A2        T1        T2

  A1      1.0000   -0.0494   -0.4642    0.0811
  A2     -0.0494    1.0000   -0.7345   -0.9370
  T1     -0.4642   -0.7345    1.0000    0.6405
  T2      0.0811   -0.9370    0.6405    1.0000
```

Zum Abschluß noch eine grafische Darstellung der gegebenen Meßpunkte und der gefitteten Modellfunktion:


100000
10000
1000
100
0
10
20
30
40
50
k →

4.6 Ergänzungen

Zwangsbedingungen

Im folgenden wird kurz die Vorgangsweise erläutert, wie man beim Least-Squares-Prozess verfährt, wenn die Modellparameter bestimmten linearen Zwangsbedingungen (*constraints*) genügen müssen. Solche Nebenbedingungen kommen in der Praxis relativ häufig vor.

Angenommen, man hat eine Modellfunktion mit q Parametern vorliegen. In diesem Fall kann es maximal $q-1$ Zwangsbedingungen geben. Unter der Voraussetzung, daß diese Bedingungen linear in den Parametern sind, können sie so geschrieben werden:

$$\sum_{l=1}^{q} \gamma_{t,l}\, a_l \stackrel{!}{=} \delta_t \qquad (t = 1, 2, \ldots, L < q)\,. \tag{4.27}$$

Es gibt also L Bedingungen, wobei die $\gamma_{t,l}$ und δ_t fixe Größen sind.

In diesem Fall hat die Least-Squares-Grundgleichung die Form

$$\chi^2 = \sum_{k=1}^{n} g_k \left[y_k - f(x_k; a_1, \ldots, a_q)\right]^2 + \sum_{t=1}^{L} \mu_t \left[\sum_{l=1}^{q} \gamma_{t,l}\, a_l - \delta_t\right] \quad \longrightarrow \quad \text{Min.}\,, \tag{4.28}$$

wobei die neuen Fit-Parameter μ_t (die sog. *Lagrange*-Parameter), die wie die übrigen Parameter behandelt werden, in vielen Fällen jedoch keine physikalische Bedeutung haben.

Least-squares when both variables have uncertainties

Ein wichtiges Problem bei der Auswertung experimenteller Daten mittels LSQ-Methoden besteht darin, daß in vielen Fällen nicht nur die y-Daten mit Meßfehlern behaftet sind, sondern auch die x-Daten: denken Sie etwa an die Messung irgendeiner physikalischen Größe in Abhängigkeit von der Zeit: in solchen Fällen wird in der Regel nicht nur die eigentliche Meßgröße, sondern auch die (auf der Abszisse aufgetragene) Zeit statistische Fehler aufweisen.

In derartigen Situationen verbietet sich die Anwendung von *Standard-LSQ-Methoden*, wie sie in den bisherigen Abschnitten dieses Kapitels verwendet wurden, und man sollte die sog. *effective variance method* einsetzen. Diese Methode werde ich im Rahmen dieser LV auf Basis der folgenden Publikation erläutern:

J. Orear, Am. J. Phys. **50**, 912 (1982)

4.7 Software-Angebot

Wie bei vielen Themen, ist das Problem bei der Internet-Suche nicht ein zu wenig, sondern ein zu viel: eine Eingabe des Stichwortes 'Levenberg-Marquardt' in die GOOGLE-Suchmaschine gibt über 6000 Eintragungen ...

- **FORTRAN**-User sollten in die bewährte NETLIB hineinschauen, insbesondere in das Programmpaket **ODRPACK**, das auf dem Levenberg-Marquardt-Verfahren beruht. Für Leute, die professionell mit diesem Programm arbeiten wollen, gibt es dort eine Menge Information, z.B. auf dem PS-File *guide.ps*.

- Für die **C**-User bin ich noch auf der Suche. Ich bin sicher, daß es auch sehr gute Levenberg-Marquardt-Realisationen in dieser Sprache gibt!

- *MATLAB* bietet eine Menge, sowohl für *linear* als auch für *non-linear fitting*. Ausführlich können Sie sich unter

  `www.mathworks.com/access/helpdesk/help/toolbox/optim/optim.shtml`

  über die *Optimization Toolbox* informieren. Durch Anklicken der Stichworte 'Gauss-Newton Method' bzw. 'Levenberg-Marquardt Method' erfahren Sie Interessantes zum diesen Themen.

- **Mathematica:** Fit[data,functions,variables]
  Ein Beispiel finden Sie im *Mathematica-Handbook* von S. Wolfram, S. 674.
  Im Paket *Statistics'NonlinearFit'* gibt es die einschlägigen Routinen *NonlinearFit* und *NonlinearRegress*.