Numerical Methods in Physics

Numerische Methoden in der Physik, 515.421.

Instructor: Ass. Prof. Dr. Lilia Boeri
Room: PH 03 090
Tel: +43-316-873 8191
Email Address: l.boeri@tugraz.at

Room: TDK Seminarraum

Time: 8:30-10 a.m.

Exercises: Computer Room, PH EG 004 F

http://itp.tugraz.at/LV/boeri/NUM_METH/index.html
(Lecture slides, Script, Exercises, etc).

TOPICS (this year):

- **Chapter 1: Introduction.** ✖
- **Chapter 2: Numerical methods for the solution of linear inhomogeneous systems.** ✖
- Chapter 3: Interpolation of point sets.
- **Chapter 4: Least-Squares Approximation.**
- **Chapter 5: Numerical solution of transcendental equations.**
- Chapter 6: Numerical Integration.
- Chapter 7: Eigenvalues and Eigenvectors of real matrices.
- **Chapter 8: Numerical Methods for the solution of ordinary differential equations: initial value problems.**
- Chapter 9: Numerical Methods for the solution of ordinary differential equations: marginal value problems.

Last week(29/10/2013)

- **Least Square Approximation:** Definition of the problem.
- **Statistical** distribution of **experimental data**: normal and Poisson distribution.
- **Statistical** properties of the **fitting parameters**.
- **Model** Functions with **Linear** parameters.
- **How is this implemented in practice?**

Least-Squares Approximation:

Mathematical Formulation: Given a set of n points (x_i/y_i), we wish to find a curve f(x) which approximates the points as closely as possible taking into account possible uncertainities due to measurement errors. We also would like to be able to assign different weights to the points through suitable weighting factors.

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

y_k	**Experimental values**
χ^2	**Weighted error sum**
$g_k > 0$	**Weighting factors (statistical)**
$f(x_k;\mathbf{a})$	**Model function**
$\mathbf{a} = a_1,\ldots,a_q$	**Model (fitting) parameters**

Statistics in the LSQ method:

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

The **weighting factors** g_k depend on the statistics of the experimental data sets y_k:

Normal distribution (analogic)

$$g_k = \frac{1}{\sigma_k^2}$$

$$P(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma}} \cdot \exp\left(-\frac{(x-\mu)^2}{2\sigma^2} \right)$$

Poisson distribution (digital)

$$g_k = \frac{1}{y_k}$$

$$P_{pois}(X = k) = f(k;\lambda) = \frac{\lambda^k e^{-\lambda}}{k!}$$

Normal matrix of the LSQ fit:

For the weighted error sum χ^2 we can define a **normal matrix** as:

$$[N]_{ij} = \frac{1}{2}\frac{\partial^2 \chi^2}{\partial \alpha_i \partial \alpha_j}$$

Its ***inverse*** is the ***covariance matrix***, which gives important information on the statistical properties of the fitting parameters:

$$C = N^{-1}$$

Covariance Matrix

$$\sigma_{a_i} = \sqrt{c_{ii}}$$

Standard deviation of the a_i's (fitting parameters)

$$r_{ij} = \frac{c_{ij}}{\sqrt{c_{ii} c_{jj}}}$$

Correlation coefficients for the a_i's: ($|cij| \leq 1$)

Measure how strongly the i-th and j-th parameter influence each other.

Model functions with linear parameters:

A model function with **linear parameters** has the form:

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

Here, $\phi_j(x)$ are **arbitrary** (linearly independent) **basis functions**. f(x,**a**) is ***linear*** in the ***fitting parameters***, not in x. The formula for χ^2 can be recast into a linear inhomogeneous problem:

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

$$\hat{A} \equiv \left[\alpha_{ij}\right] \qquad \alpha_{ij} = \sum_{k=1}^{n} g_k \varphi_i(x_k)\varphi_j(x_k) \qquad \beta_i = \sum_{k=1}^{n} g_k y_k \varphi_i(x_k)$$

It can be shown that **A** also coincides with the ***normal matrix*** of the problem.

How to implement LSQ with linear model parameters:

1) Input the experimental data set: x_k, y_k and the statistical weights g_k

2) Choose a set of **basis functions** $\phi_j(x)$: $f(x;\mathbf{a}) = \sum_{j=1}^{m} a_j \cdot \varphi_j(x)$

3) Construct the auxiliary linear problem: $\hat{A}\mathbf{a} = \beta$

$$\hat{A} \equiv \left[\alpha_{ij}\right] \qquad \alpha_{ij} = \sum_{k=1}^{n} g_k \varphi_i(x_k)\varphi_j(x_k), \qquad \beta_i = \sum_{k=1}^{n} g_k y_k \varphi_i(x_k)$$

4) Solve the linear problem (LU decomposition); Find optimal fitting parameters.

$$\mathbf{a}^{opt} = (a_1^{opt}, \ldots, a_q^{opt})$$

5) Calculate the covariance matrix (standard deviations of the fitting parameters).

6) Calculate the value of the optimal fitting function on the given data points: $f(x_k;\mathbf{a}^{opt})$

7) Evaluate the weighted error sum. $\chi^2 = \sum_{k=1}^{n} g_k \left[y_k - f(x_k;\mathbf{a})\right]^2$

This week(5/11/2013)

- **Least Square Approximation in practice.**
- Description of a program to perform the **LSQA** for model functions with **linear parameters**.
- Model functions with **non-linear parameters:** definition.
- Model functions with **non-linear** parameters: easy linearization **“tricks”**.

How to implement LSQ with linear model parameters:

1) Input the experimental data set: x_k, y_k and the statistical weights g_k
2) Choose a set of **basis functions** $\phi_j(x)$: $f(x;\mathbf{a}) = \sum_{j=1}^{m} a_j \cdot \varphi_j(x)$
3) Construct the auxiliary linear problem: $\hat{A}\mathbf{a} = \beta$

$$\hat{A} \equiv \left[\alpha_{ij}\right] \qquad \alpha_{ij} = \sum_{k=1}^{n} g_k \varphi_i(x_k)\varphi_j(x_k), \qquad \beta_i = \sum_{k=1}^{n} g_k y_k \varphi_i(x_k)$$

4) Solve the linear problem (LU decomposition); Find optimal fitting parameters.

$$\mathbf{a}^{opt} = (a_1^{opt}, \ldots, a_q^{opt})$$

5) Calculate the covariance matrix (standard deviations of the fitting parameters).
6) Calculate the value of the optimal fitting function on the given data points: $f(x_k;\mathbf{a}^{opt})$
7) Evaluate the weighted error sum. $\chi^2 = \sum_{k=1}^{n} g_k \left[y_k - f(x_k;\mathbf{a})\right]^2$

Program LFIT: input and output parameters

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

INPUT:

X,Y: vectors that contain the x_k, y_k for the LSQ fit.

SIG: vector with the standard deviations for the data (σ_k).

NDATA: Number of data points.

MA: Number of parameters in the model function (=q).

OUTPUT:

A: Vector which contains the optimized parameters for the model function ($\mathbf{a}^{opt}$).

YF: Vector which contains the values of the optimized fitting parametrs in x_k - $f(x_k)$.

COVAR: Covariance matrix.

CHISQ: Value of the weighted error sum for $\mathbf{a}=\mathbf{a}^{opt}$.

Program LFIT: structure of the program

1) Calculation of the matrix elements α_{ij} and of the components of the inhomogeneous vector β_i of the inhomogeneous vector. These quantities are stored in NORMAL(,) and BETA.

2) Calculation of the optimised fitting parameters and of the covariance matrix through the routines **LUDCMP** and **LUBKSB** (LU decomposition).

3) Calculation of the weighted error sum for the best-fit curve using the input x_k, y_k and the optimised fitting parameters.

External Functions/Routines:

- **FUNCS** (Contains the basis functions used for the optimal fit).
- **LUDCMP** and **LUBKSB:** routines used for the LU decomposition and for the inversion of the normal matrix -> covariance matrix.

```
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                                70
        COVAR(I,J):=LOES(I)
(return)
```

Initialize arrays to zero: normal (A) and beta (β).

Evaluate the components of A and beta:

solve $A \cdot x = \beta$ **->**

Get optimal values of the fitting parameters.

Use the optimal values of the fitting parameters

to evaluate χ^2

Calculate Covariance matrix:

$C = A^{-1}$.

```
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)
```

Evaluate the components of A and beta.

solve $A \cdot x = \beta$ -> Get optimal values of the fitting parameters.

$$\hat{A} \equiv [\alpha_{ij}] \qquad \alpha_{ij} = \sum_{k=1}^{n} g_k \varphi_i(x_k) \varphi_j(x_k) \qquad \beta_i = \sum_{k=1}^{n} g_k y_k \varphi_i(x_k)$$

Iterate on k=1,N_k **(NDATA)**

K=1(1)NDATA	
	FUNCS(X(K),AFUNC,MA)
	G:=1.0/SIG(K)/SIG(K)

Evaluate $\phi_i(x_k)$ **–** External call to **funcs**. Note that there are q=**MA** basis functions.

$$\varphi_i(x_k) \rightarrow \text{afunc(i)}$$

Evaluate the statistical weights $\mathbf{g_k=(1/\sigma_k)^2}$

$$g_k -> g$$

$$\hat{A} \equiv \left[\alpha_{ij}\right] \qquad \alpha_{ij} = \sum_{k=1}^{n} g_k \varphi_i(x_k)\varphi_j(x_k) \qquad \beta_i = \sum_{k=1}^{n} g_k y_k \varphi_i(x_k)$$

(Iterate on k=1,N_k **(NDATA))**

```
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)
```

Iterate on i=1,q **(NA)** – index for basis functions (rows).

Iterate on j=1,i **(NA)** – index for basis functions (columns).

The A matrix and β vector are constructed row by row for each x_k.

$$\alpha_{ij}(\mathrm{k}) = g_k \varphi_i(x_k)\varphi_j(x_k) \rightarrow \mathrm{normal(i,j)}$$

$$\beta_i(k) = g_k y_k \varphi_i(x_k) \rightarrow \mathrm{beta(i)}$$

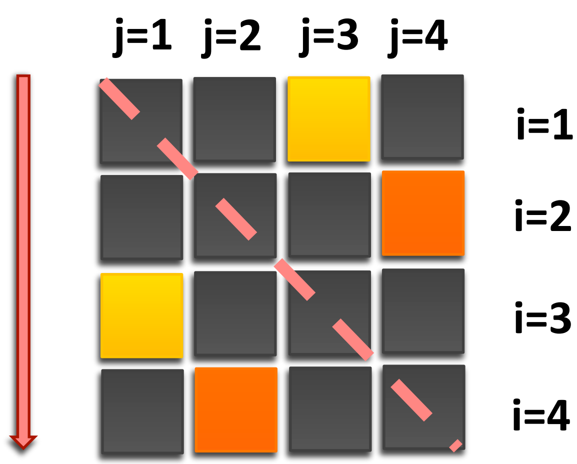
Up to here.

$$\sum_k f(k) \rightarrow \mathrm{new{=}old + new}$$

$$\hat{A} \equiv [\alpha_{ij}] \qquad \alpha_{ij} = \sum_{k=1}^{n} g_k \varphi_i(x_k) \varphi_j(x_k) \qquad \beta_i = \sum_{k=1}^{n} g_k y_k \varphi_i(x_k)$$

Fill up the A (normal) matrix, which has to be symmetric:

```
I=2(1)MA
    J=1(1)I-1
        NORMAL(J,I):=NORMAL(I,J)
```



i=2, q

j=1,i-1

We fill the lower half!

Solve the auxiliary problem: $\hat{A}\mathbf{a} = \beta$

```
LUDCMP(NORMAL,MA,INDX,D,KHAD)
LUBKSB(NORMAL,MA,INDX,BETA,LOES)
```

Using **LU decomposition** (see lect. 3, 15/10/2013 and script, chapter 2).

LUDCMP: The matrix NORMAL is decomposed into lower + upper triangular matrices; **NORMAL contains now the LU decomposition of A**.

LUBKSB: The linear inhomogeneous problem is solved with backward + forward recursion. **BETA** is the inhomogeneous vector, **LOES** is the solution.

$$\mathbf{a} = \mathbf{a}_{opt} = (a^1_{opt}, \ldots, a^q_{opt}) \leftrightarrow \text{LOES}$$

```
J=1(1)MA
    A(J):=LOES(J)
```

The **q (MA) optimal parameters are saved into the vector A(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)
```

Use the optimal values of the fitting parameters to evaluate χ^2

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

Optimal fitting function:

$$f(x;\mathbf{a}^{opt}) = \sum_{j=1}^{q} a_j^{opt} \cdot \varphi_j(x)$$

```
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)
```

Evaluate $\phi_i(x_k)$ **–** External call to **funcs**.

$$\varphi_i(x_k) \rightarrow \mathrm{afunc(i)}$$

Evaluate $f_j\,(x_k)$

$$f_j(x_k;\mathbf{a}^{opt}) = a_j^{opt} \cdot \varphi_j(x_k)$$

$$f_j(x_k;\mathbf{a}^{opt}) \rightarrow \mathrm{sum}$$

Iterate over j:

```
J=1(1)MA
    SUM:=SUM + A(J)*AFUNC(J)
YF(K):=SUM
CHISQ:=CHISQ + G*(Y(K)-SUM)*(Y(K)-SUM)
```

$$f(x_k;\mathbf{a}^{opt}) = \sum_{j=1}^{q} a_j^{opt}\cdot\varphi_j(x_k) = \sum_{j=1}^{q} f_j(x_k)$$

$$f(x_k;\mathbf{a}^{opt}) = \mathrm{SUM} \rightarrow \mathrm{YF(k)}$$

$$\chi_k^2 = g_k\left[y_k - f(x_k;\mathbf{a})\right]^2$$

Close the iteration on k:

$$\chi^2 = \sum_k \chi_k^2$$

Calculate the covariance matrix:

```
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)
```

70

$$C = N^{-1}$$

Using **LU decomposition** (see lect. 3, 15/10/2013 and script, chapter 2).

Possible uses: 2 Inversion of a matrix:

$$A \cdot X = I, X \equiv A^{-1}$$

$$A\begin{pmatrix} x_{11} \\ \dots \\ x_{n1} \end{pmatrix} = \begin{pmatrix} 1 \\ \dots \\ 0 \end{pmatrix} \dots \qquad A\begin{pmatrix} x_{1n} \\ \dots \\ x_{nn} \end{pmatrix} = \begin{pmatrix} 0 \\ \dots \\ 1 \end{pmatrix}$$

Structure chart — Inversion of a matrix

```
LUDCMP (A,N,INDX,D,KHAD)
J=1(1)N
    I=1(1)N
        VEKTOR(I):=0.0
    VEKTOR(J):=1.0
    LUBKSB (A,N,INDX,VEKTOR,X)
    I=1(1)N
        AINV(I,J):=X(I)
```

$$A \cdot X = I, X \equiv A^{-1}$$

The **inverse matrix** can be constructed as a collection of n column vectors:

$$\begin{pmatrix} x_{11} \\ \cdots \\ x_{n1} \end{pmatrix} \cdots \begin{pmatrix} x_{1j} \\ \cdots \\ x_{nj} \end{pmatrix} \cdots \begin{pmatrix} x_{1n} \\ \cdots \\ x_{nn} \end{pmatrix}$$

The same is true for the **identity matrix**, where the column vectors are:

$$\begin{pmatrix} 1 \\ \cdots \\ 0 \end{pmatrix} \cdots \begin{pmatrix} 0 \\ 1 \\ x_{nj} \end{pmatrix} \cdots \begin{pmatrix} 0 \\ \cdots \\ 1 \end{pmatrix}$$

Only the j^{th} element is non-zero and equal to one.

$$A \cdot X = I, X \equiv A^{-1}$$

The **original problem** reduces to n equivalent vector problems:

$$A \cdot \mathbf{x}_j = \mathbf{b}_j$$

For the column vectors $\mathbf{x}_j$ and $\mathbf{b}_j$.

In this way the LU decomposition of A is performed only once, and one has to solve (**LUBKSB**) n times an inhomogeneous system, with a different $\mathbf{b}_j$.

```
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)
```

70

Run over columns:

Construct the inhomogeneous vector for the j-th column: $\boldsymbol{\beta}_j$

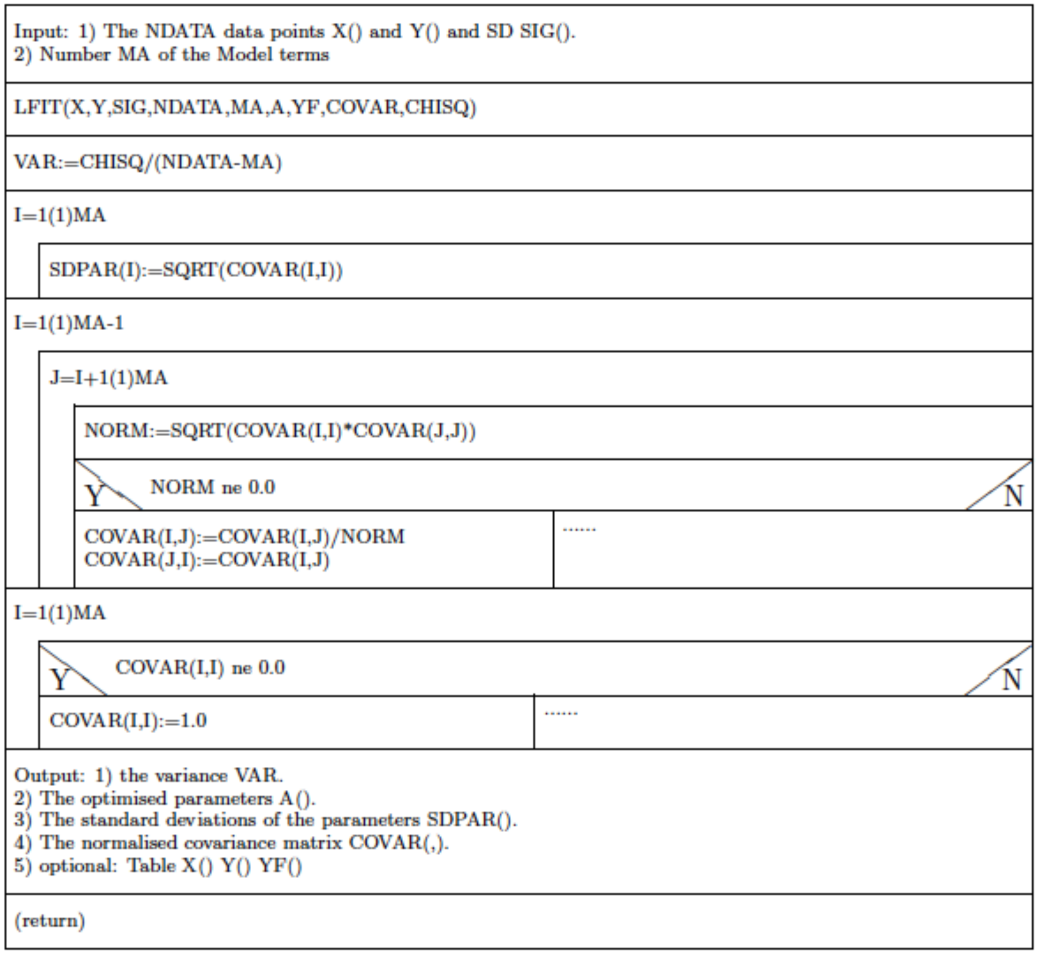
Solve the linear system $A\mathbf{x}_j=\boldsymbol{\beta}_j$

The column vector x_j is saved into the covariance matrix.

How to use LFIT?

Solving a LSQ problem in practice

Given a data set, we want to obtain the best fit with LSQ, and evaluate the quality of the fit (variance and standard deviation).



Normal matrix of the LSQ fit:

For the weighted error sum χ^2 we can define a **normal matrix** as:

$$[N]_{ij} = \frac{1}{2}\frac{\partial^2 \chi^2}{\partial \alpha_i \partial \alpha_j}$$

Its ***inverse*** is the ***covariance matrix***, which gives important information on the statistical properties of the fitting parameters:

$$C = N^{-1}$$

Covariance Matrix

$$\sigma_{a_i} = \sqrt{c_{ii}}$$

Standard deviation of the a_i's (fitting parameters)

$$r_{ij} = \frac{c_{ij}}{\sqrt{c_{ii}c_{jj}}}$$

Correlation coefficients for the a_i's: $(|cij|\leq 1)$

Measure how strongly the i-th and j-th parameter influence each other.

Variance and Standard Deviation:

$$V = \frac{\chi^2}{N - q}$$ **Variance**

$$\sigma_V = \sqrt{\frac{2}{N - q}}$$ **Standard Deviation**

q Number of **fitting parameters**

$N - q$ Number of **degrees of freedom**

Quality of the LSQ fit: In case of an **ideal model** for N>>1, V has approximately a **normal distribution** with E=1 and $\sigma=\sigma_V$. If V lies **significantly** outside the interval: $[1-\sigma_V, 1+\sigma_V]$ the fit is bad!

A practical example:

Let us imagine that we want to find the best fitting function for N_k=101 points, randomly generated around a polynomial curve, with a given (constant) standard deviation σ. The (third degree) polynomial is:

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

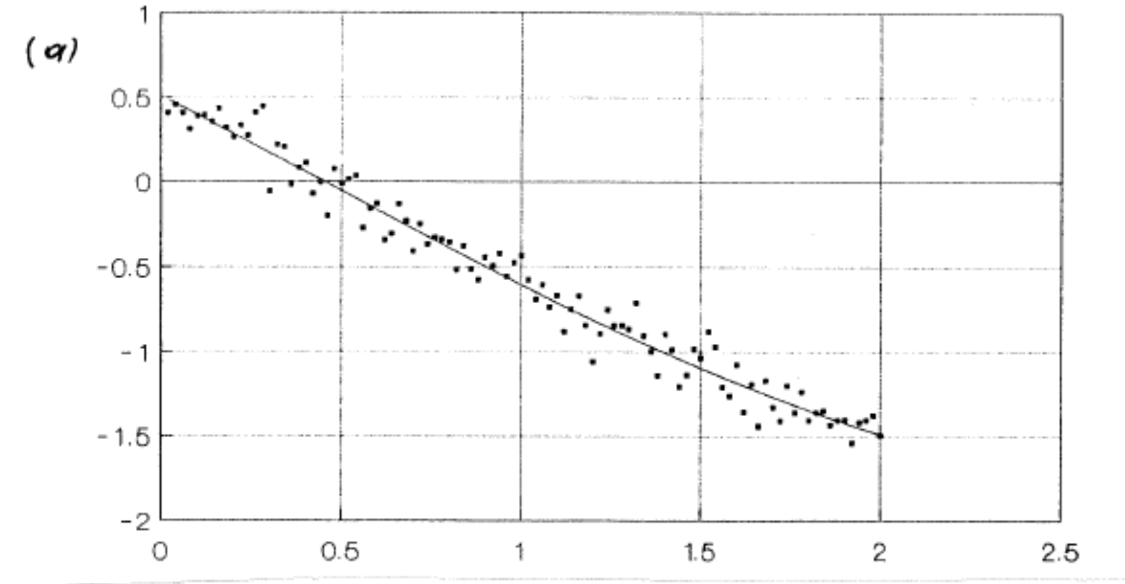
The optimal fit would be given by the linear model function (q=4):

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

With:

$$a_0 = 0.5 \qquad a_1 = -1 \qquad a_2 = -0.2 \qquad a_3 = 0.1$$

First data set (σ=0.1)



$a_0 = 0.5$	$a_1 = -1$	$a_2 = -0.2$	$a_3 = 0.1$	**Real**
$a_0 = 0.5097$	$a_1 = -1.1152$	$a_2 = -0.0517$	$a_3 = 0.053$	**Fit**

Table 4.1 (a):

MA	Variance	optimised parameters	Remarks
1	37.034	$a_1 = -0.5652 \pm 0.0100$	bad model
2	1.138	$a_1 = 0.4574 \pm 0.0198$	border value
		$a_2 = -1.0226 \pm 0.0171$	
3	1.032	$a_1 = 0.5307 \pm 0.0293$	
		$a_2 = -1.2449 \pm 0.0676$	good model
		$a_3 = 0.1111 \pm 0.0327$	
4	1.035	$a_1 = 0.5097 \pm 0.0384$	
		$a_2 = -1.1152 \pm 0.1670$	good model,
		$a_3 = -0.0517 \pm 0.1945$	bad statistics
		$a_4 = 0.0543 \pm 0.0639$	s. Fig.3.6
5	1.046	$a_1 = 0.5134 \pm 0.0469$	
		$a_2 = -1.1543 \pm 0.3284$	good model
		$a_3 = 0.0372 \pm 0.6726$	bad statistics
		$a_4 = -0.0151 \pm 0.5065$	
		$a_5 = 0.0174 \pm 0.1256$	

$$\sigma_V = \sqrt{\frac{2}{N-q}} \approx \sqrt{2} \cdot 10^{-1} \approx 0.14 \qquad V \pm \sigma_V = 0.86 - 1.14$$

Second data set (σ=0.025, improved statistics):

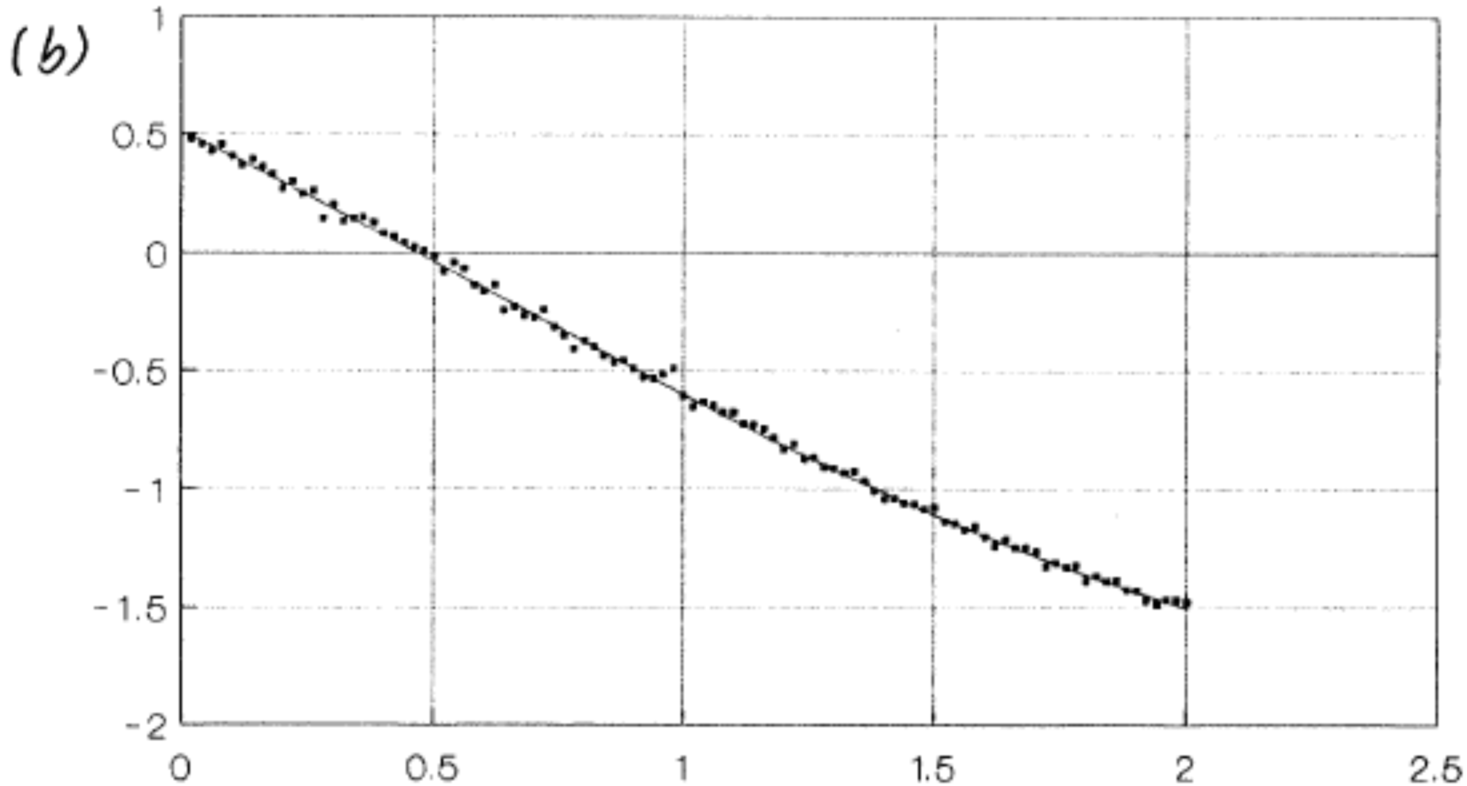


Table 4.1 (b):

MA	Variance	optimized parameters	Remarks
1	598.559	$a_1 = -0.5639 \pm 0.0025$	bad model
2	2.883	$a_1 = 0.4773 \pm 0.0049$	bad model
		$a_2 = -1.0413 \pm 0.0043$	
3	1.204	$a_1 = 0.5472 \pm 0.0073$	
		$a_2 = -1.2530 \pm 0.0169$	bad model
		$a_3 = 0.1059 \pm 0.0082$	
4	0.899	$a_1 = 0.5128 \pm 0.0096$	good model
		$a_2 = -1.0413 \pm 0.0417$	the fitting parameters have
		$a_3 = -0.1600 \pm 0.0486$	convenient σ's.
		$a_4 = 0.0886 \pm 0.0160$	see Fig.3.6
5	0.908	$a_1 = 0.5141 \pm 0.0117$	good model
		$a_2 = -1.0548 \pm 0.0821$	but some fitting parameters
		$a_3 = -0.1294 \pm 0.1681$	have a very bad statistics.
		$a_4 = 0.0647 \pm 0.1266$	'mixing of parameters'
		$a_5 = 0.0060 \pm 0.0314$	

$$\sigma_V = \sqrt{\frac{2}{N-q}} \approx \sqrt{2} \cdot 10^{-1} \approx 0.14$$

$$V \pm \sigma_V = 0.86 - 1.14$$

Functions with non-linear parameters

Linearization method (special cases) + Gauss Newton method

Model functions with non-linear parameters:

If we assume as fitting function an **exponential**:

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

We obtain for the weighted error sum:

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

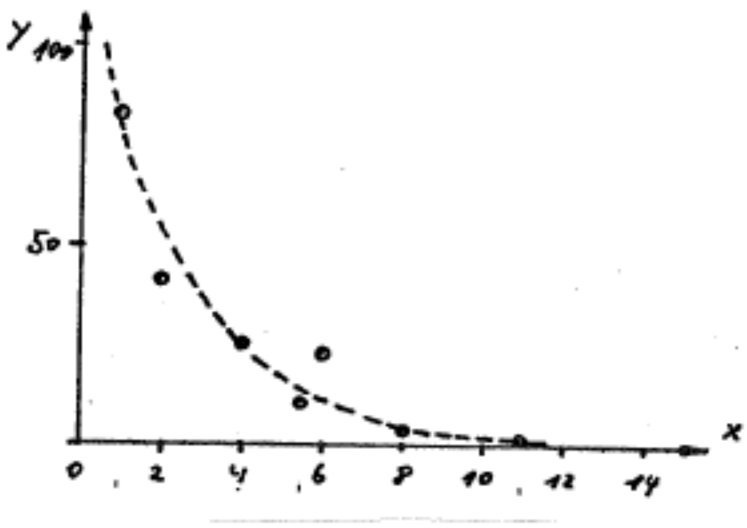
The **minimum condition** gives:

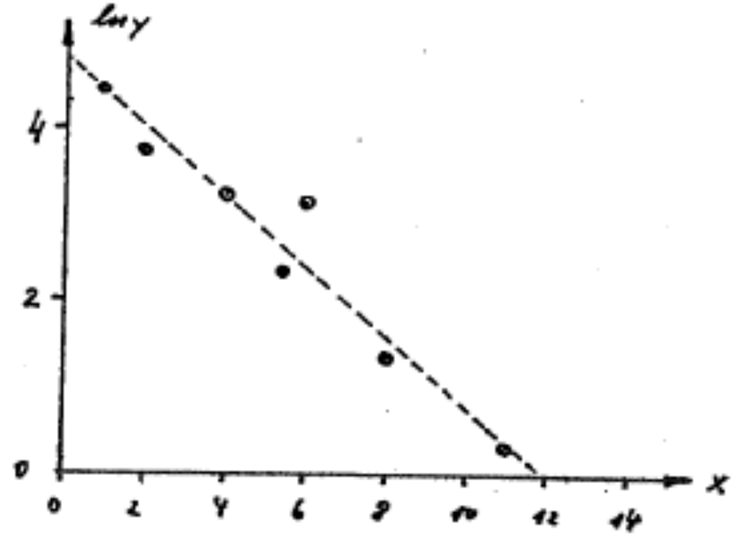
$$\begin{cases} a \cdot \sum_{k=1}^{n} g_k e^{-2bx_k} = \sum_{k=1}^{n} g_k y_k e^{-bx_k} \\ a \cdot \sum_{k=1}^{n} g_k e^{-2bx_k} = \sum_{k=1}^{n} g_k x_k y_k e^{-bx_k} \end{cases}$$

The system of equations is non-linear!

However, the exponential function is usually linearized using logarithms:

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





The set of equations for the a,b parameters is also linear:

$$\begin{pmatrix} n & \sum_k x_k \\ \sum_k x_k & \sum_k x_k^2 \end{pmatrix} \cdot \begin{pmatrix} \ln(a) \\ -b \end{pmatrix} = \begin{pmatrix} \sum_k \ln(y_k) \\ \sum_k x_k \ln(y_k) \end{pmatrix}$$

The solution is:

$$a = \exp\left[\left(\sum \ln y_k \cdot \sum x_k^2 - \sum x_k \cdot \sum x_k \ln y_k\right) / D\right]$$

$$b = -\left(n \cdot \sum x_k \ln y_k - \sum x_k \cdot \sum \ln y_k\right) / D$$

$$D = n \cdot \sum x_k^2 - \left(\sum x_k\right)^2$$

Remark: hidden correlations

Hidden correlations between the fitting parameters can occur when the parameters chosen to represent a function are not “really independent”.

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

$$\Leftrightarrow$$

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

This week(5/11/2013)

- **Least Square Approximation in practice.**
- Description of a program to perform the **LSQA** for model functions with **linear parameters**.
- Model functions with **non-linear parameters:** definition.
- Model functions with **non-linear** parameters: easy linearization **“tricks”**.