Terascale alliance 19th June 2008

Likelihood fits – Advanced algorithms and tools

Miniworkshop on Statistical Tools

Volker Blobel – Universität Hamburg

---

1. Introduction

Likelihood fits in physics analysis: two main steps

- Construction of the Likelihood function (⇒ **part 2**), and
- Minimization of the Likelihood function (Optimization) (⇒ **part 3**)

The construction of the Likelihood function requires knowledge of the physics concepts and of details of the statistics of the measurement process.

- **Training + Optimized packages are required!**

Some recent books on Optimization:

Jorge Nocedal and Stephen J. Wright, **Numerical Optimization**, Springer Series in Operations Research (1999)

J. Frederic Bonnans, J. Charles Gilbert, Claude Lemaréchal, Claudia A. Sagastizábal, **Numerical Optimization** – Theoretical and Practical Aspects, Springer (2006)

Per Christian Hansen, **Rank-Deficient and Discrete Ill-posed Problems**, Siam (1998)

Optimization with bound constraints, equality constraints, and inequality constraints is treated.

. . . the world of HEP . . .

Some concepts/notions/strategies, which are popular in HEP, are not mentioned in the text books:

- χ^2 minimization
- ". . . we are using an iterative method . . . " (often extremely slow steepest descent method)
- determination of parameter dependence by grid interpolation
- simulated annealing, for continuous problems
- inequality constraints (bounds) treated by transformation (MINUIT)
- straight-line fits with MINUIT, higher-order polynomial fit without orthogonalization
- "fit uncertainties", "fit errors"; individual χ^2 contributions to . . . , unbiased residuals . . .
- "inappropriateness of using $\Delta\chi^2 = 1$ *(for error estimation)* . . . used $\Delta\chi^2 = 50$ for 90 % confidence level"
- proofs and error estimates by Monte Carlo simulation, experimental Mathematics
- unsmearing bin-by-bin correction of measured distributions
- ". . . solution of linear system of equations requires inversion . . . "
- ". . . in 4200 unknowns: computational infeasible; even worse, non-linear fit won't converge . . . ".

2. Likelihood function for optimal physics analysis

The construction of ML functions $\mathcal{L}$ is sometimes done in packages, but more often done individually for a certain analysis. An inappropriate construction may lead to a **bias** in the result.
Negative log-likelihood function $F(\boldsymbol{x})$, requiring minimization w.r.t. the unknowns $\boldsymbol{x}$

$$F(\boldsymbol{x}) = -\log \mathcal{L}(\boldsymbol{x}) \qquad \text{(corresponding to } \tfrac{1}{2} \text{ “}\chi^2(\boldsymbol{x})\text{”)}$$

allows combination of least squares and maximum-likelihood expressions.

Recommended principles:

- **leave data $\boldsymbol{y}_j$ (e.g. counts) unchanged (no corrections to the data)**
- **compare unchanged data to expectation $\boldsymbol{t}_j$, including all corrections, like normalization uncertainty, background.**

- Do not destroy ideal statistic, e.g. Poisson statistic for counts.
- Do not introduce correlations between data points from systematic effects by data corrections.
- If possible, use parametrization with (*almost*) orthogonal parameters: e.g. σ in $N/(\sqrt{2\pi}\sigma)\exp(-(\mu-y)^2/(2\sigma^2))$ instead of $N\exp(-(\mu-y)^2/(2\sigma^2))$.
- Use correct statistical expressions for densities in maximum likelihood fits. Example: Normalization uncertainty $\Rightarrow$.

Normalization uncertainty

The normalization factor is represented by $\alpha = 1 \pm \varepsilon$, i.e. ε is the relative normalization error.

From a paper: "...the following definition for the effective χ^2 is adopted:

$$\chi^2_{\text{global}} = \sum_n w_n \chi^2_n(x) \qquad \chi^2_n(x) = \left(\frac{\alpha_n - 1}{\varepsilon_n}\right)^2 + \sum_\ell \left(\frac{\boxed{\alpha_n y_{n\ell}} - t_{n\ell}(x)}{\sigma^y_{n\ell}}\right)^2$$

(n labels different experiments).

The normalization parameter α_n should be applied as a factor to the expection $t_{n\ell}(x)$, instead of the data $y_{n\ell}$; otherwise a normalization bias is introduced.

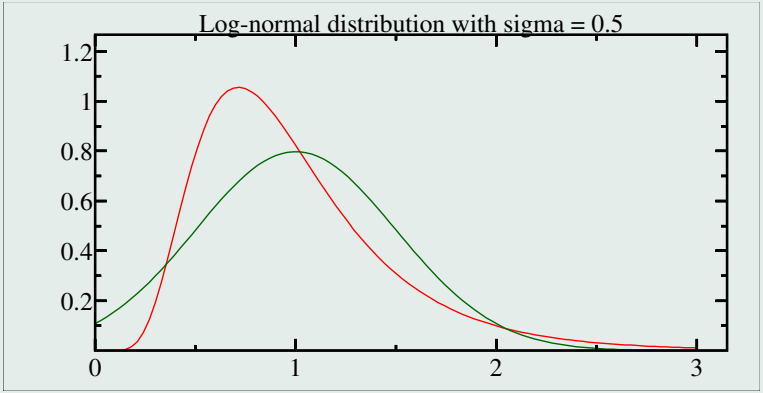
$$\alpha_n y_{n\ell} - t_{n\ell}(x) \qquad \Rightarrow \qquad y_{n\ell} - \alpha_n t_{n\ell}(x)$$

...and what about weight factor w_n?

Normalization error = uncertainty in the factor, used to convert *event numbers* to *cross sections*. Because of its origin – product of many factors, each with some uncertainty – the factor perhaps follows a **log-normal distribution** due the multiplicative central limit theorem.

For a log-normal distribution of a random variable α with $E[\alpha] = 1$ and standard deviation of ε the contribution to the ML-function is

$$\Delta F^{\text{norm}} = \tfrac{1}{2} \ln \alpha \left(3 + \frac{\ln \alpha}{\ln (1+\varepsilon^2)} \right)$$
$$\to \tfrac{1}{2} \frac{(\alpha - 1)^2}{\varepsilon^2} \quad \text{for small } \varepsilon$$



The normal and the log-normal distribution, both with mean 1 and standard deviation $\varepsilon = 0.5$.

Log-normal density of a random factor $\alpha = 1 \pm \varepsilon$:

$$p(\alpha) = \exp \left[-\tfrac{1}{2} \ln \alpha \left(3 + \frac{\ln \alpha}{\ln (1+\varepsilon^2)} \right) + \frac{\varepsilon^2}{8} - \ln \varepsilon - \ln \sqrt{2\pi} \right]$$

The objective function in least squares is the sum of **squares** of scaled residuals z, with larger influence for larger residuals (outliers) $\Rightarrow$ the presence of outliers in the data will deteriorate the fit result.

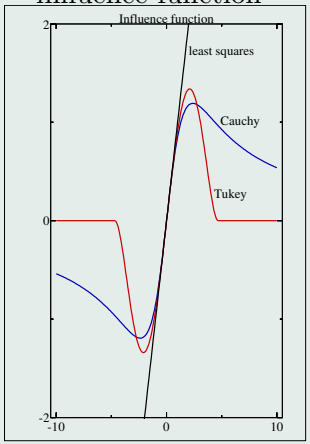
The **square** is replaced in M-estimates by a dependence with **reduced influence** for larger residuals (non-gaussian maximum likelihood).

Least squares $F(\boldsymbol{x}) = \frac{1}{2}\sum_i z_i^2$ $\qquad z_i = \left(\frac{y_i - t_i}{\sigma_i}\right)$

e.g. Cauchy $\frac{1}{2}z_i^2 \rightarrow \frac{c^2}{2}\ln\left(1 + (z/c)^2\right)$

	$\rho(z) = -\ln \mathrm{pdf}(z)$	influence function $\psi(z) = \mathrm{d}\rho(z)/\mathrm{d}z$	add. weight $\omega(z) = \psi(z)/z$
Least squares	$= \frac{1}{2}z^2$	$= z$	$= 1$
Cauchy($c = 2.3849$)	$= \frac{c^2}{2}\ln\left(1 + (z/c)^2\right)$	$= \frac{z}{1 + (z/c)^2}$	$= \frac{1}{1 + (z/c)^2}$
Huber $\begin{cases}\text{if } \lvert z\rvert \le c = 1.345\\ \text{if } \lvert z\rvert > c = 1.345\end{cases}$	$= \begin{cases}z^2/2\\ c\,(\lvert z\rvert - c/2)\end{cases}$	$= \begin{cases}z\\ c\cdot\mathrm{sign}\,(z)\end{cases}$	$= \begin{cases}1\\ c/\lvert z\rvert\end{cases}$

influence function



scaled residual z

Nuisance parameters and profile likelihood

Statistical analysis of subset of parameters, e.g. one or two or ... parameters requires minimization w.r.t. to all other parameters (called nuisance parameters) of problem,
i.e. eliminating the other parameters, but accounting for the extra uncertainty due to the *fitted* nuisance parameters.

One-parameter analysis: minimize $F(x_1, \boldsymbol{x}_r)$ for set of fixed values for x_1 w.r.t. all other parameters $\boldsymbol{x}_r$ (option MINOS in MINUIT):

$$P(x_1) = \min_{\boldsymbol{x}_r} F(x_1, \boldsymbol{x}_r)$$

Two-parameter analysis: minimize $F(x_1, x_2, \boldsymbol{x}_r)$ for fixed pairs (x_1, x_2) w.r.t. all other parameters $\boldsymbol{x}_r$ (option CONTUR in MINUIT):

$$P(x_1, x_2) = \min_{\boldsymbol{x}_r} F(x_1, x_2, \boldsymbol{x}_r))$$

Partitioning: Solution for subset $\boldsymbol{x}_1$ by partitioning of $\boldsymbol{x}$ into $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ and elimination of $\boldsymbol{x}_2$
$\Rightarrow$ (next part).

- Construct an **Objective Function** $F(\boldsymbol{x})$ from statistical considerations, depending on the parameter n-vector $\boldsymbol{x}$, to be minimized. $\boldsymbol{x} \in \mathcal{R}^n$ with start value $\boldsymbol{x}_0$.
- **Step 1:** construct quadratic model $\widetilde{F}_k(\boldsymbol{d}) = F_k + \boldsymbol{d}^{\mathrm{T}}\boldsymbol{\nabla}F_k + \frac{1}{2}\boldsymbol{d}^{\mathrm{T}}\boldsymbol{C}_k\boldsymbol{d}$

  function value: F_k gradient: $\boldsymbol{\nabla}F_k$ sec. der. matrix (Hessian): $\boldsymbol{C}_k \approx \boldsymbol{\nabla}^2 F_k$ at $\boldsymbol{x}_k$
- **Step 2:** determine minimum of quadratic model function: step vector $\boldsymbol{d}_k$

  solve $\boldsymbol{C}_k\,\boldsymbol{d}_k = \boldsymbol{b}_k \equiv -\boldsymbol{\nabla}F_k$ for $\boldsymbol{d}_k$ and update: $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + 1\cdot\boldsymbol{d}_k$
- repeat steps 1 and 2 (i.e. iterate) for non-linear problems (only single step for $\boldsymbol{C}$ = constant)
- covariance matrix of parameter n-vector $\boldsymbol{x}$ is given by inverse matrix: $\boldsymbol{B} = \boldsymbol{C}^{-1}$
  (i.e. error propagation) (equivalent to "$\Delta\chi^2 = 1$")

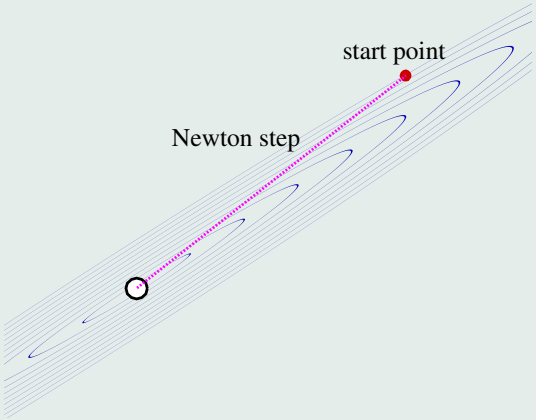
Improved convergence by

line-search $\min \varphi(\alpha) = F(\boldsymbol{x}_k + \alpha \cdot \boldsymbol{d}_k)$ w.r.t. factor α

or trust-region $\min \widetilde{F}_k(\boldsymbol{d}) = F_k + \boldsymbol{d}^{\mathrm{T}}\boldsymbol{\nabla}F_k + \frac{1}{2}\boldsymbol{d}^{\mathrm{T}}\boldsymbol{C}_k\boldsymbol{d}$ w.r.t. $\boldsymbol{d}$ with constraint $\|\boldsymbol{d}\| \le \Delta_k$

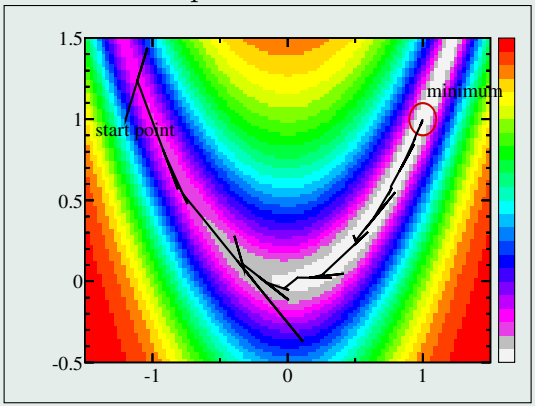
"... extremely important, and might supersede line-searches, sooner or later."

. . . large correlation



Quadratic function is minimized by a single NEWTON step; steepest-descent would be inefficient due to high correlation.

. . . non-quadratic behaviour



Non-quadratic function: minimization requires to follow very narrow curved valley.

- NEWTON method based on second-derivative matrix has *quadratic convergence* rate.
- Simple methods like steepest-descent with only *linear convergence* rate are slow; convergence may not occur as the iteration stagnates (can be misinterpreted as indication for convergence).

Derivative calculation

Gauss-Newton methods, to calculate gradient and Hessian approximation of the log ML-function $F(\boldsymbol{x})$ from *first* derivatives of the parametrization function f:
(only 6 statements to accumulate $\boldsymbol{b}$ and $\boldsymbol{C}$)

- Least squares: fit of function f to data $\boldsymbol{y}$ (Gauss-Newton, f'' terms ignored)

$$\boldsymbol{b}_j = \sum_i w_i \left(\frac{\partial f_i}{\partial x_j}\right)(y_i - f_i) \qquad \boldsymbol{C}_{jk} \approx \sum_i w_i \left(\frac{\partial f_i}{\partial x_j}\right)\left(\frac{\partial f_i}{\partial x_k}\right)$$

- ML-estimation (Poisson): fit of (bin-integrated) function f to counts y_i (f'' terms ignored)

$$\boldsymbol{b}_j = \sum_i \left(\frac{\partial f_i}{\partial x_j}\right)\frac{y_i - f_i}{f_i} \qquad \boldsymbol{C}_{jk} \equiv (\boldsymbol{\nabla}^2 F_{jk}) \approx \sum_i \left(\frac{y_i}{f_i^2}\right)\left(\frac{\partial f_i}{\partial x_j}\right)\left(\frac{\partial f_i}{\partial x_k}\right)$$

numerical differentiation: by finite-differences in a number of repeated function evaluations;

first der.: $2n$ second der.: $n(n-1)/2 \propto n^2$ function evaluations

automatic differentiation: takes as input a computer code able to calculate a function value – produces as output another computer code able to calculate the derivatives;

symbolic differentiation: algebraic specification of the function is manipulated to produce new algebraic expressions for the derivatives.

Solution by partitioning

Interest in subset $\boldsymbol{x}_1$ of total parameter vector $(\boldsymbol{x}_1, \boldsymbol{x}_2)$

Partitioning the matrix equation $\boldsymbol{C}\boldsymbol{x} = \boldsymbol{b}$ ($\boldsymbol{C}$ symmetric, $\boldsymbol{C}_{11}$ and $\boldsymbol{C}_{22}$ are square matrices):

$$\left(\begin{array}{c|c} \boldsymbol{C}_{11} & \boldsymbol{C}_{12} \\ \hline \boldsymbol{C}_{12}^{\mathrm{T}} & \boldsymbol{C}_{22} \end{array}\right) \left(\begin{array}{c} \boldsymbol{x}_1 \\ \hline \boldsymbol{x}_2 \end{array}\right) = \left(\begin{array}{c} \boldsymbol{b}_1 \\ \hline \boldsymbol{b}_2 \end{array}\right)$$

with $\boldsymbol{C}_{12} \neq 0$. Solve matrix equation for $\boldsymbol{x}_1$, e.g. by inversion:

$$\left(\quad \boldsymbol{C}_{11} - \boldsymbol{C}_{12}\boldsymbol{C}_{22}^{-1}\boldsymbol{C}_{12}^{T} \quad\right) \left(\quad \boldsymbol{x}_1 \quad\right) = \left(\quad \boldsymbol{b}_1 - \boldsymbol{C}_{12}\left[\boldsymbol{C}_{22}^{-1}\ \boldsymbol{b}_2\right] \quad\right)$$

$$\text{covariance matrix} \quad \boldsymbol{V}_{11}(\boldsymbol{x}_1) = \left(\quad \boldsymbol{C}_{11} - \boldsymbol{C}_{12}\boldsymbol{C}_{22}^{-1}\boldsymbol{C}_{12}^{T} \quad\right)^{-1}$$

Note: "local solution" $\boldsymbol{x}_2^* \equiv \left[\boldsymbol{C}_{22}^{-1}\ \boldsymbol{b}_2\right]$

Applications:

- Track detector alignment (Millepede): simultaneous fit of alignment parameters ($\boldsymbol{x}_1$) and an unlimited number of parameters of tracks ($\boldsymbol{x}_2$, many).
- Barlow fraction fit of histograms, using MC histograms with limited statistics.

MINUIT: "used by people in many fields, many applications; lifetime long compared with computers, accelerators" [MINUIT Upgrade Project 2002]

From the MINUIT User's Guide (2004):

"...suited to handle difficult problems, including those which may require guidance in order to find the correct solution."

"...not intended for the repeated solution of identical parametrized problems ..."

"...no limit on the number of parameters ...however the 'technological" limitations of MINUIT can be seen around a maximum of 15 free parameters ..."

number of parameters	small number of cases	large number of cases
small number	MINUIT	–
derivatives available	[MINUIT]	L-BFGS, N
constraints required	Newton + Lagrange	
large number	L-BFGS	–
ill-conditioned	Regularization	

Additional methods needed for special problems:
large number of parameters, large number of cases, constraints, ill-conditioned problems.

Symmetric matrices

Space $\propto n^2$ and cpu-time $\propto n^3$ for **standard** (in-place) solution of a matrix equation of dimension n:

dimension n	space		cpu-time	
10	0.8	kB	10^{-5}	sec
1 000	8	MB	10	sec
100 000	80	GB	120	days

Method acceptable for n up to 1 000 (or even 10 000 – few hours cpu-time), if matrix condition is sufficient, and robust algorithm is used.

Sparse matrix: large matrices are often *sparse* – can be stored in $\ll n^2$ locations, but inverse of sparse matrix is full matrix!

Generalized residual minimization: (related to method of conjugate gradients);

system of linear equations $\boldsymbol{C}\,\boldsymbol{x} = \boldsymbol{y}$ equivalent to $\min ||\boldsymbol{C}\,\boldsymbol{x} - \boldsymbol{y}||_2$

$\boldsymbol{C}$ may be indefinite, very large and sparse. Needs only product of sparse matrix $\boldsymbol{C}\times$ vector, e.g. MINRES*. Example: $n = 50\,558$ with cpu-time of < 3 minutes instead of 15 days.

Band matrix, arrow matrix, skyline matrix: cpu-time only $\propto n^1$ by Cholesky method with decomposition $\boldsymbol{C} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^{\mathrm{T}}$, forward $\boldsymbol{L}\boldsymbol{z} = \boldsymbol{y}$ and backward solution $\boldsymbol{L}^{\mathrm{T}}\boldsymbol{x} = \boldsymbol{D}^{-1}\boldsymbol{z}$. Even elements of inverse matrix within band can be calculated fast.

Specialized parametrization of problem: $\Rightarrow$ e.g. band matrix msc track fit by broken lines (large nr of parameters) 10 $\times$ faster than Kalman.

*) C. C. Paige and M. A. Saunders (1975), Solution of sparse indefinite systems of linear equations, SIAM J. Numer. Anal. 12(4), pp. 617-629.

`www.stanford.edu/group/SOL/software/minres.html`

Quasi-Newton algorithm (only gradient required)

Invented in the mid 1950's (physicist W.C. Davidon – unpublished)

Start with simple (diagonal) matrix $\boldsymbol{B}_0$ ($\boldsymbol{C}_0^{-1}$), calculate gradient compute step $\boldsymbol{d}$ by product

$$\boldsymbol{d} = -\boldsymbol{B}_k \times \boldsymbol{\nabla} F_k$$
$$\text{line-search} \quad \min_\alpha \varphi(\alpha) \equiv F(\boldsymbol{x}_k + \alpha \boldsymbol{d}) \qquad \Rightarrow \qquad \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha \boldsymbol{d}\ ,$$

calculate differences $\boldsymbol{s}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k \qquad \boldsymbol{y}_k = \boldsymbol{\nabla} F_{k+1} - \boldsymbol{\nabla} F_k$, and update matrix $\boldsymbol{B}_k$ such that $\boldsymbol{B}_{k+1}\boldsymbol{y}_k = \boldsymbol{s}_k$ (BFGS updating formula, 1970)

$$\boldsymbol{B}_{k+1} = \left(\boldsymbol{I} - \rho_k \boldsymbol{s}_k \boldsymbol{y}_k^{\mathrm{T}}\right) \boldsymbol{B}_k \left(\boldsymbol{I} - \rho_k \boldsymbol{y}_k \boldsymbol{s}_k^{\mathrm{T}}\right) + \rho_k \boldsymbol{s}_k \boldsymbol{s}_k^{\mathrm{T}} \qquad \text{with } \rho_k = \frac{1}{\boldsymbol{y}_k^{\mathrm{T}} \boldsymbol{s}_k}$$

$\Rightarrow$ correct inverse Hessian $\boldsymbol{B}$ reached after n (linear independent) steps for quadratic problem - but may diverge for small $\boldsymbol{y}_k^{\mathrm{T}} \boldsymbol{s}_k$!

Requires modern line-search algorithm with sufficient decrease in function value and slope (strong Wolfe conditions) (J.J. Moré and D.J. Thuente (1994)) to have self-correcting properties. (1976)

$\Rightarrow$ a minimization algorithm, that needs only the gradient (no second derivatives).

Limited memory BFGS (L-BFGS)

BFGS algorithm requires matrix with $\propto n^2$ locations:

What to do, if the number of parameters is 100 000 or larger?

Note: the inverse Hessian $\boldsymbol{B}_k$ is used only in the product $\boldsymbol{d} = -\boldsymbol{B}_k \times \boldsymbol{\nabla} F_k$

BFGS: $\boldsymbol{B}_k$ determined by $\boldsymbol{B}_0$ and k pairs $\{\boldsymbol{s}_0, \boldsymbol{y}_0\} \ldots \{\boldsymbol{s}_{k-1}, \boldsymbol{y}_{k-1}\}$.

In J. Nocedals "Limited memory" algorithm (1980) the vector $\boldsymbol{d}$ is calculated from

L-BFGS: diagonal $\boldsymbol{B}_0$ and the last m pairs $\{\boldsymbol{s}_{k-m}, \boldsymbol{y}_{k-m}\} \ldots \{\boldsymbol{s}_{k-1}, \boldsymbol{y}_{k-1}\}$.

with fixed $m = 7 \ldots 29$, i.e. space $\propto n^1$, **no matrix storage** required! Algorithm often more efficient than original BFGS algorithm, and requires e.g. only 60 Mbyte for 100 000 parameters.

" . . . algorithm is excellent, it is at present the best choice, often the only possible one, for large-scale problems . . . "

Comparison LVMINI/MINUIT

Minimization package LVMINI, using L-BFGS, developed for up to several 100 000 parameters, needs gradient ∇F; error calculation included.

	Straight-line least squares fit			Rosenbrock fct	
	LVMINI	MINUIT	MINUIT-MINOS	LVMINI	MINUIT
no gradient	-	131	392	-	286
gradient	12 + 4	70	159	71	144
+ diagonal	10 + 4				

2-parameter fits;

in some straight-line fits MINUIT parabolic error and correlation wrong.

	LVMINI	MINUIT	MINUIT-MINOS
no gradient	-	170	638
gradient	14 + 8	118	289
+ diagonal	11 + 8		

4-parameter fits:
Gaussian histogram peak + background

Initial values		Number of function evaluation LVMINI	MINUIT	MINUIT-MINOS
good	no gradient		2074	22 417
	gradient	558 + 40	–	–
	+ diagonal	29 + 40		
bad	no gradient		[2865]	[17 199]
	gradient	601 + 40	–	–
	+ diagonal	104 + 40		

20-parameter fits:
five peaks, described by Students distribution, plus linear background.

Note: errors calculated by LVMINI are as accurate as MINOS errors in this test. Number in [ ] = minimum not found.

Ill-posed least squares: $\boldsymbol{A}\boldsymbol{x} \cong \boldsymbol{y}$ (with weakly defined degrees of freedom)

Idea of regularization: control the *norm of the residuals* and the *norm of the solution vector* $\boldsymbol{x}$, simultaneously, by adding regularizing term with regularization parameter $\tau > 0$:

Thikhonov-Phillips: $F_\tau(\boldsymbol{x}) = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|^2 + \tau \|\boldsymbol{L}\boldsymbol{x}\|^2$ with $\boldsymbol{L} = \boldsymbol{I}$ or second-derivative term

Matrix equation: $\left(\boldsymbol{C} + \tau \boldsymbol{L}^{\mathrm{T}}\boldsymbol{L}\right)\boldsymbol{x} \equiv \left(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{V}_y^{-1}\boldsymbol{A} + \tau \boldsymbol{L}^{\mathrm{T}}\boldsymbol{L}\right)\boldsymbol{x} = \boldsymbol{A}^{\mathrm{T}}\boldsymbol{V}_y^{-1}\boldsymbol{y} \equiv \boldsymbol{b}$ to be solved

Solution with orthogonalization: eigenvalues λ_j and eigenvectors $\boldsymbol{u}_j$

Fourier coefficients $c_j = \dfrac{1}{\sqrt{\lambda_j}}\left(\boldsymbol{b}^{\mathrm{T}}\boldsymbol{u}_j\right)$ independent, with error = 1

Solution $\boldsymbol{x}_\tau = \displaystyle\sum_{j=1}^{n} f_j \frac{1}{\sqrt{\lambda_j}} c_j \boldsymbol{u}_j$ with filter factor $f_j = \left(\dfrac{\lambda_j}{\lambda_j + \tau}\right)$

Regularization parameter τ appears only in filterfactor: scan over a large τ range with selection of τ, that minimizes the **global correlation** between the parameters.

Summary

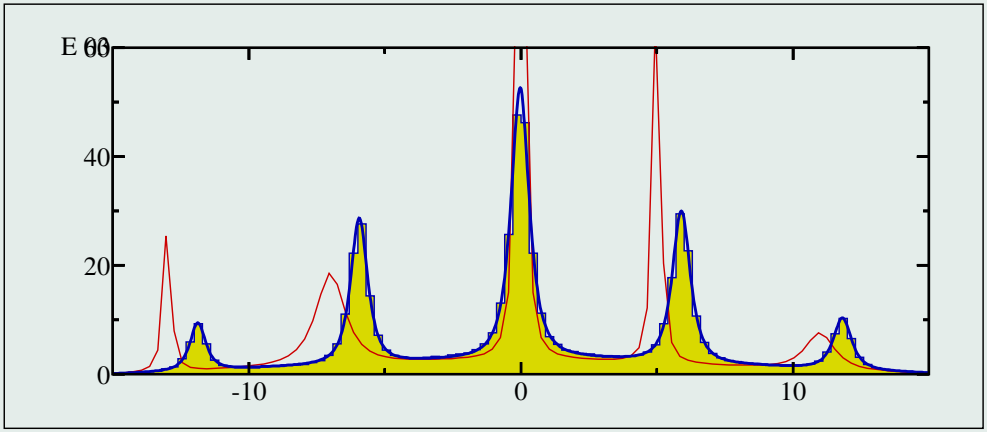
Additional methods needed for different problems: for example

- large number of parameters up to 10^5;
- large number of cases;
- solution of ill-posed problems (e.g. unfolding).

New methods developed during the last 40 years: for example

- BFGS with modern line-search (Wolfe conditions);
- limited memory BFGS for large-scale optimization;
- regularization methods with stable solutions

Initial values		Number of function evaluation LVMINI	MINUIT	MINUIT-MINOS
good	no gradient		2074	22 417
	gradient	558 + 40	–	–
	+ diagonal	29 + 40		
bad	no gradient		[2865]	[17 199]
	gradient	601 + 40	–	–
	+ diagonal	104 + 40		



Five peaks, described by Students distibution, plus linear background. Initial parameter values (bad) corresponds to red line; blue line is fit result. Number in [ ] = minimum not found.

Contents