Modeling of data

• The problem is roughly the following:

> We have a model that should describe the behavior or our experimental or simulation data and it has parameters the values of which we do not know. Using modeling of data we pursue to obtain values and error estimates for those parameters.

• If the model is obtained from a theory the parameters may have a physical meaning.
• In statistical modeling parameters do not necessarily have a clear interpretation.

• The basic approach is the following:

> We choose a figure-of-merit (**FOM**) function which tells us the difference between the data and the model (with a certain set of model parameters). The smaller the value of this function the better the model describes the data.

• We also should be able to obtain a statistical measure for the goodness of the fit and the uncertainties in the parameters.

• Assume we have M data points (x_i, y_i), $i = 1, 2, \ldots, M$. The model has N parameters.

$$y(x) = y(x; a_1, a_2, \ldots, a_N).$$

• The problem is thus:

> If we have the parameter set $(a_1, a_2, \ldots, a_N)$ what is the probability that it produces this particular data set (within some interval Δy)?

Modeling of data

- If we assume that the **data points are drawn from a Gaussian distribution** with standard deviation σ_i then we may estimate this probability as

$$P \propto \prod_{i=1}^{M} \left\{ \exp\left[-\frac{1}{2} \left(\frac{y_i - y(x_i)}{\sigma_i} \right)^2 \right] \Delta y \right\}.$$

- Maximizing this is equivalent to minimizing

$$\sum_{i=1}^{M} \left(\frac{y_i - y(x_i)}{\sigma_i} \right)^2 - N \ln \Delta y.$$

- Because N and Δy are constant (for this particular problem) we have to minimize the quantity

$$\chi^2 = \sum_{i=1}^{M} \left(\frac{y_i - y(x_i; a_1, a_2, \ldots, a_N)}{\sigma_i} \right)^2.$$

by varying the parameter set $(a_1, a_2, \ldots, a_N)$.

- χ^2 has the distribution $Q(\chi^2 | \nu)$ where ν is the number of degrees of freedom.

- In this case $\nu = M - N$.

Modeling of data

- $Q(\chi^2|\nu)$ tells us the **probability with which the value χ^2 can be exceeded by chance**.

  1. If the probability if very small, differences between the data and the model are not random fluctuations or the uncertainties of the data are too optimistic. Or the measurement errors may not be normally (Gaussian) distributed.

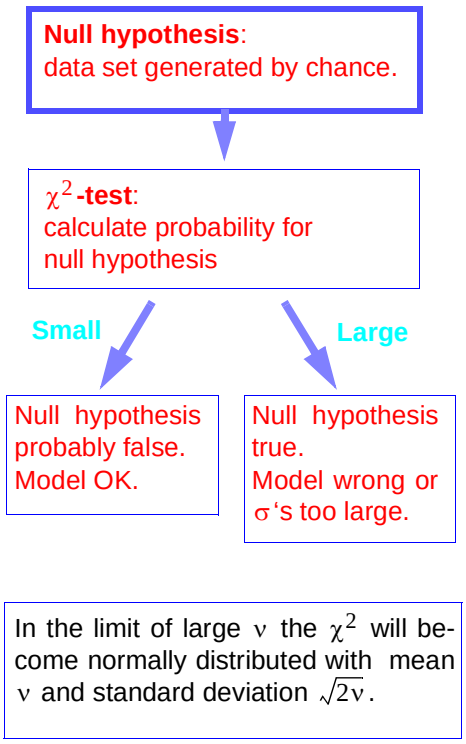
  2. If the probability is near unity the uncertainties of the data may be too large.

  3. Rule of thumb: a good fit gives $\chi^2 \approx \nu$.

- If we don't know the σ_i's we can get an estimate by

  1. Doing the fit by using a constant σ for all data points.
  2. Computing an estimate for σ_i's by

$$\sigma = \sum_{i=1}^{M} \frac{[y_i - y(x_i)]^2}{M}.$$



In the limit of large ν the χ^2 will become normally distributed with mean ν and standard deviation $\sqrt{2\nu}$.

Modeling of data

• By setting the derivative of χ^2 with respect to parameters a_k we get the equation that we must solve

$$\sum_{i=1}^{M} \left[\frac{y_i - y(x_i)}{\sigma_i^2}\right]\left[\frac{\partial y(x_i;(a_1, \ldots, a_k, \ldots, a_N))}{\partial a_k}\right] = 0.$$

- This is in general a set of M nonlinear equations for $(a_1, a_2, \ldots, a_N)$.

Modeling of data: line fitting

• Line fitting: Let's start with the most simple model (also called linear regression):

$$y(x) = y(x;a,b) = a + bx\,.$$

- One usually assumes that the values y_i have an uncertainty σ_i but x_i are accurate.

- The FOM function is now

$$\chi^2(a,b) = \sum_{i=1}^{M} \left(\frac{y_i - a - bx_i}{\sigma_i} \right)^2 .$$

- In the minimum its derivatives with respect to a and b are zero:

$$\frac{\partial \chi^2}{\partial a} = -2 \sum_{i=1}^{M} \frac{y_i - a - bx_i}{\sigma_i^2} = 0$$

$$\frac{\partial \chi^2}{\partial b} = -2 \sum_{i=1}^{M} \frac{x_i(y_i - a - bx_i)}{\sigma_i^2} = 0\,.$$

Modeling of data: line fitting

- Let's define the following sums

$$S \equiv \sum_{i=1}^{M} \frac{1}{\sigma_i^2}, \quad S_x \equiv \sum_{i=1}^{M} \frac{x_i}{\sigma_i^2}, \quad S_y \equiv \sum_{i=1}^{M} \frac{y_i}{\sigma_i^2}, \quad S_{xx} \equiv \sum_{i=1}^{M} \frac{x_i^2}{\sigma_i^2}, \quad S_{xy} \equiv \sum_{i=1}^{M} \frac{x_i y_i}{\sigma_i^2}.$$

- Now the equations can be written as

$$\begin{cases} aS + bS_x = S_y \\ aS_x + bS_{xx} = S_{xy} \end{cases}$$

- Now we get the solution into form

$$\Delta = SS_{xx} - S_x^2$$

$$a = \frac{S_{xx}S_y - S_x S_{xy}}{\Delta}.$$

$$b = \frac{SS_{xy} - S_x S_y}{\Delta}$$

Modeling of data: line fitting

- An estimate for the uncertainties of the parameters can be obtained

from the rule of the propagation of errors $\sigma_f^2 = \sum_{i=1}^{M} \sigma_i^2 \left(\frac{\partial f}{\partial y_i}\right)^2$:

$$\sigma_a^2 = \sum_{i=1}^{M} \sigma_i^2 \left(\frac{\partial a}{\partial y_i}\right)^2 = \sum_{i=1}^{M} \sigma_i^2 \left(\frac{S_{xx} - S_x x_i}{\sigma_i^2 \Delta}\right)^2 = \frac{S_{xx}}{\Delta}$$

$$\sigma_b^2 = \sum_{i=1}^{M} \sigma_i^2 \left(\frac{\partial b}{\partial y_i}\right)^2 = \sum_{i=1}^{M} \sigma_i^2 \left(\frac{S x_i - S_x}{\sigma_i^2 \Delta}\right)^2 = \frac{S}{\Delta}$$

- Note that these estimates assume that the coefficients a and b are uncorrelated.

- When there is error also in the values x_i the minimization gets more complicated.
  - FOM function is now computed as

$$\chi^2(a,b) = \sum_{i=1}^{M} \frac{(y_i - a - b x_i)^2}{\sigma_{y_i}^2 + b^2 \sigma_{x_i}^2}$$

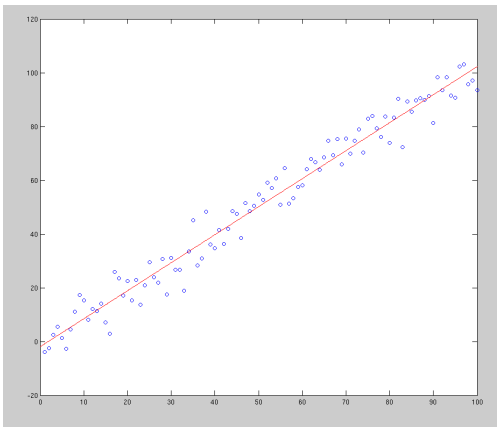
Note: $\mathrm{Var}(y_i - a - b x_i) = \mathrm{Var}(y_i) + b^2 \mathrm{Var}(x_i) = \sigma_{y_i}^2 + b^2 \sigma_{x_i}^2$.

  - Minimization is no more a linear problem but one has to use nonlinear minimization methods.

Modeling of data: line fitting

- Line fitting (or polynomial fitting in general) can be done in **Matlab** using the function **`polyfit`**:

```
>> load line_100_5_1.dat
>> x=line_100_5_1(:,1);
>> y=line_100_5_1(:,2);
>> plot(x,y,'o');
>> xf=0:0.1:100;
>> [p,s]=polyfit(x,y,1);
>> p
p = 1.0420   -1.9453
>> yf=polyval(p,xf);
>> hold
Current plot held
>> plot(xf,yf,'-r');
```



- GSL has also line fitting routines:

```
int gsl_fit_wlinear (const double *X, const size_t
  XSTRIDE, const double *W, const size_t WSTRIDE, const double
  *Y, const size_t YSTRIDE, size_t N, double *C0, double *
  C1, double *COV00, double *COV01, double *COV11, double *
  CHISQ)

This function computes the best-fit linear regression coefficients
(C0,C1) of the model Y = c_0 + c_1 X for the weighted datasets (X, Y),
two vectors of length N with strides XSTRIDE and YSTRIDE.  The vector
W, of length N and stride WSTRIDE, specifies the weight of each
datapoint. The weight is the reciprocal of the variance for each
datapoint in Y.

The covariance matrix for the parameters (C0, C1) is estimated from
weighted data and returned via the parameters (COV00, COV01, COV11).
The weighted sum of squares of the residuals from the best-fit line,
\chi^2, is returned in CHISQ.
```

Modeling of data: general linear fitting

• Now the linearity means *linearity of the model with respect to parameters* a_i.

- In general this can be expressed as

$$y(x) = \sum_{k=1}^{N} a_k X_k(x),$$

where $X_k(x)$ are arbitrary functions or **basis functions**. Note that they need not be linear in x.

- Our objective is to determine parameters a_i in such a way that the basis functions reproduce the data as accurately as possible:

$$y_i \approx \sum_{k=1}^{N} X_k(x_i) a_k$$

- Defining an $M \times N$ matrix $\mathbf{A}$ as $A_{ij} = X_j(x_i)$, denoting with vector $\mathbf{a}$ the N parameters a_i, and with vector $\mathbf{y}$ the M data values y_i we can express the least squares problem in matrix for as

$$\mathbf{Aa} = \mathbf{y}.$$

- Because this is an overdetermined system (more equations than unknowns, $M > N$) the equality should be understood in the least squares sense; or

$$\mathbf{Aa} \approx \mathbf{y},$$

meaning that the norm of the residual is minimized:

$$\min_{\mathbf{a}} \|\mathbf{Aa} - \mathbf{y}\|_p$$

- The most convenient norm in this case is the Euclidean norm i.e. $\| \|_2$.

Modeling of data: general linear fitting

- Actually the overdetermined problem may be more general than the one mentioned above.
- An example[1]:

A surveyor measures heights of mountains.

Measurement results are $h_1, h_2, h_3, d_{21}, d_{31}, d_{23}$.

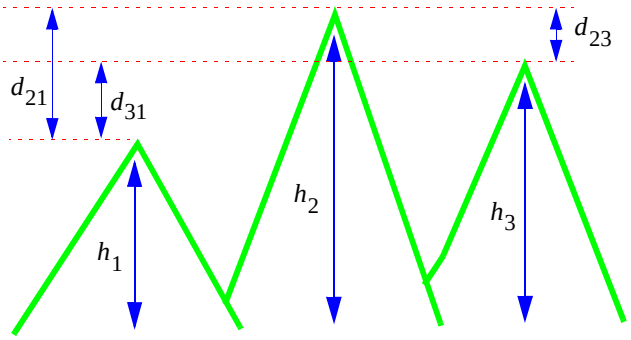
We must compute the best estimates of the heights x_1, x_2, x_3.

- We can write down the equations

$$\begin{cases} x_1 = h_1 \\ x_2 = h_2 \\ x_3 = h_3 \\ -x_1 + x_2 = d_{21} \\ -x_1 + x_3 = d_{31} \\ x_2 - x_3 = d_{23} \end{cases};$$

in matrix form:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ d_{21} \\ d_{31} \\ d_{23} \end{bmatrix}$$



Solution by Matlab

```
>> A=[1 0 0; 0 1 0; 0 0 1; -1 1 0; -1 0 1; 0 1 -1];
>> b=[310 520 405 205 120 100]';
x=A\b
x =
                        305
                        515
                        415

>> A*x-b
ans =
                         -5
                         -5
           9.99999999999994
                          5
          -10.0000000000001
        5.6843418860808e-14
```

Modeling of data: general linear fitting

- A simple example of a linear least squares problem is a polynomial of degree N as basis functions:

$$X_k(x) = x^{k-1}.$$

- The FOM function is now

$$\chi^2 = \sum_{i=1}^{M} \left[\frac{y_i - \sum_{k=1}^{N} a_k X_k(x_i)}{\sigma_i} \right]^2.$$

- This is more easily handled by the matrix notation:

$\mathbf{A}$ is an $M \times N$ matrix with elements $A_{ij} = \dfrac{X_j(x_i)}{\sigma_i}$

1. From M.T.Heath, *Scientific Computing: An Introductory Survey*, McGraw-Hill, 2002.

Modeling of data: general linear fitting

- Normally $\mathbf{A}$ has more rows than columns ($M > N$):

basis functions →

data points ↓

$$\begin{bmatrix} \frac{X_1(x_1)}{\sigma_1} & \frac{X_2(x_1)}{\sigma_1} & \frac{X_3(x_1)}{\sigma_1} & \cdots & \frac{X_N(x_1)}{\sigma_1} \\ \frac{X_1(x_2)}{\sigma_2} & \frac{X_2(x_2)}{\sigma_2} & \frac{X_3(x_2)}{\sigma_2} & \cdots & \frac{X_N(x_2)}{\sigma_2} \\ \frac{X_1(x_3)}{\sigma_3} & \frac{X_2(x_3)}{\sigma_3} & \frac{X_3(x_3)}{\sigma_3} & \cdots & \frac{X_N(x_3)}{\sigma_3} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{X_1(x_M)}{\sigma_M} & \frac{X_2(x_M)}{\sigma_M} & \frac{X_3(x_M)}{\sigma_M} & \cdots & \frac{X_N(x_M)}{\sigma_M} \end{bmatrix}$$

- Vector $\mathbf{b}$ with M elements is defined as

$$b_i = \frac{y_i}{\sigma_i}.$$

- Parameters a_i are denoted by vector $\mathbf{a}$ with N elements.

- By setting the partial derivatives with respect to parameters to zero we get the equations

$$\sum_{i=1}^{M} \frac{1}{\sigma_i^2} \left[y_i - \sum_{j=1}^{N} a_j X_j(x_i) \right] X_k(x_i) = 0, \qquad k = 1, \ldots, N.$$

Modeling of data: general linear fitting

- This can be written in the form

$$\sum_{j=1}^{N} \alpha_{kj} a_j = \beta_k,$$

where

$$\alpha_{kj} = \sum_{i=1}^{M} \frac{X_j(x_i) X_k(x_i)}{\sigma_i^2} \quad (\text{or } \alpha = \mathbf{A}^T\mathbf{A}), \quad \text{is an } N \times N \text{ matrix and}$$

$$\beta_k = \sum_{i=1}^{M} \frac{y_i X_k(x_i)}{\sigma_i^2} \quad (\text{or } \beta = \mathbf{A}^T\mathbf{b}), \quad \text{is a vector with } N \text{ elements.}$$

- Or in matrix form these **normal equations** are

$$\alpha\mathbf{a} = \beta$$

or

$$(\mathbf{A}^T\mathbf{A})\mathbf{a} = \mathbf{A}^T\mathbf{b}.$$

- This group of equations can be solved using the normal linear algebra methods (LU and back substitution).

Modeling of data: general linear fitting

- Let's denote the inverse matrix as $\mathbf{C} = \alpha^{-1}$.

- To estimate the uncertainties of the parameters a_j we can write the parameter as

$$a_j = \sum_{k=1}^{N} [\alpha]^{-1}_{jk} \beta_k = \sum_{k=1}^{N} C_{jk} \left[\sum_{i=1}^{M} \frac{y_i X_k(x_i)}{\sigma_i^2} \right].$$

- The estimated variance of a_j can be found as in the case of line fitting

$$\sigma^2(a_j) = \sum_{i=1}^{M} \sigma_i^2 \left(\frac{\partial a_j}{\partial y_i} \right)^2.$$

- Because α_{jk} is independent of y_i

$$\frac{\partial a_j}{\partial y_i} = \sum_{k=1}^{N} \frac{C_{jk} X_k(x_i)}{\sigma_i^2}.$$

Modeling of data: general linear fitting

- Finally we get

$$\sigma^2(a_j) = \sum_{k=1}^{N}\sum_{l=1}^{N} C_{jk}C_{jl}\left[\sum_{i=1}^{M}\frac{X_k(x_i)X_l(x_i)}{\sigma_i^2}\right]..$$

- The term in the brackets is $\alpha_{kl} = C_{kl}^{-1}$ and we obtain

$$\sigma^2(a_j) = C_{jj}\,.$$

- One can also show that the off-diagonal elements of the matrix C are the covariances of the parameters[1].

- Sometimes the matrix α is (nearly) singular.
  - In these cases we can not use the normal methods for solving the equation equations[2].

- Moreover, the normal equations have $\mathbf{A}^T\mathbf{A}$ as the multiplying matrix. One can show that the condition number κ of behaves as

$$\kappa(\mathbf{A}^T\mathbf{A}) = [\kappa(\mathbf{A})]^2.$$

(Remember that the condition number gives the error sensitivity of the linear problem $\mathbf{Ax} = \mathbf{b}$ $\frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}\|} \approx \kappa(\mathbf{A})\frac{\|\delta\mathbf{A}\|}{\|\mathbf{A}\|}$ and $\kappa(\mathbf{A}) = \|\mathbf{A}^{-1}\|\|\mathbf{A}\|$.)

1. See e.g. *Numerical Recipes*, paragraph 15.6
2. In this case—when the matrix is symmetric positive definite—the Cholesky factorization $\mathbf{A} = \mathbf{L}\mathbf{L}^T$, where $\mathbf{L}$ is a lower triangular matrix.

Modeling of data: general linear fitting

- A simple example:

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ \delta & 0 \\ 0 & \delta \end{bmatrix}, \qquad \mathbf{A}^T\mathbf{A} = \begin{bmatrix} 1+\delta^2 & 1 \\ 1 & 1+\delta^2 \end{bmatrix}.$$

- For $0 < \delta < \sqrt{\varepsilon}$ (with ε the machine epsilon $\Rightarrow 1+\delta^2 \approx 1$) we obtain for matrix

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix},$$

which is singular.

Modeling of data: general linear fitting

- Moreover, the condition number of matrices $\mathbf{A}$ and $\mathbf{A}^T\mathbf{A}$ are as below:

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ \delta & 0 \\ 0 & \delta \end{bmatrix} \qquad \mathbf{A}^+ = \frac{1}{\delta^2+2}\begin{bmatrix} 1 & 1/\delta+\delta & -1/\delta \\ 1 & -1/\delta & 1/\delta+\delta \end{bmatrix}$$

$$\|\mathbf{A}\|_2 = \sqrt{2+\delta^2} \qquad \|\mathbf{A}^+\|_2 = \frac{1}{\delta}$$

$$\Rightarrow \quad \kappa(\mathbf{A}) = \|\mathbf{A}\|_2\|\mathbf{A}^+\|_2 = \frac{\sqrt{2+\delta^2}}{\delta} \to \frac{\sqrt{2}}{\delta}, \text{ when } \delta \to 0.$$

$$\mathbf{B} \equiv \mathbf{A}^T\mathbf{A} = \begin{bmatrix} 1+\delta^2 & 1 \\ 1 & 1+\delta^2 \end{bmatrix} \qquad \mathbf{B}^+ = \frac{1}{\delta^2(2+\delta^2)}\begin{bmatrix} 1+\delta^2 & -1 \\ -1 & 1+\delta^2 \end{bmatrix}$$

$$\|\mathbf{B}\|_2 = 2+\delta^2 \qquad \|\mathbf{B}^+\|_2 = \frac{1}{\delta^2}$$

$$\Rightarrow \quad \kappa(\mathbf{B}) = \|\mathbf{B}\|_2\|\mathbf{B}^+\|_2 = \frac{2+\delta^2}{\delta^2} \to \frac{2}{\delta^2}, \text{ when } \delta \to 0.$$

Modeling of data: general linear fitting

- Here we have used $\mathbf{A}^+$ (an $N \times M$ matrix) which is so called *pseudo-inverse* of $\mathbf{A}$.
  - It is a generalization of matrix inverse for nonsquare matrices.
  - It has the following properties:

(i) $\mathbf{A}\mathbf{A}^+\mathbf{A} = \mathbf{A}$
(ii) $\mathbf{A}^+\mathbf{A}\mathbf{A}^+ = \mathbf{A}^+$
(iii) $(\mathbf{A}\mathbf{A}^+)^T = \mathbf{A}\mathbf{A}^+$
(iv) $(\mathbf{A}^+\mathbf{A})^T = \mathbf{A}^+\mathbf{A}$.

  - It is also the unique minimal norm solution to the problem

$$\min_{\mathbf{X} \in R^{N\times M}} \|\mathbf{A}\mathbf{X} - \mathbf{1}\|_F.$$

  - Pseudo-inverse can be calculated using the singular value decomposition of matrix $\mathbf{A}$:

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T,$$

$$\mathbf{A}^+ = \mathbf{V}\mathbf{S}^+\mathbf{U}^T,$$

$$\mathbf{S}^+ = \operatorname{diag}\left(\frac{1}{\sigma_1}, \frac{1}{\sigma_2}, \ldots, \frac{1}{\sigma_r}, 0, \ldots, 0\right) \in R^{N\times M},\ r = \operatorname{rank}(\mathbf{A}),$$

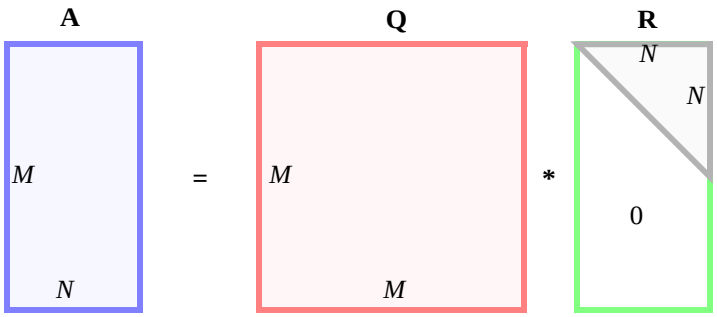
where σ_i are the singular values of $\mathbf{A}$ (see below).

Modeling of data: general linear fitting

- We can transform the least squares problem to a more stable form by QR decomposition:

  $\mathbf{A} = \mathbf{Q}\begin{bmatrix}\mathbf{R}\\ 0\end{bmatrix}$, where $\mathbf{Q}$ is an $M \times M$ orthogonal matrix (i.e. $\mathbf{Q}^T\mathbf{Q} = 1$) and $\mathbf{R}$ is an $N \times N$ triangular matrix.

- As a figure:



- The least squares problem is now

  $\mathbf{Q}\begin{bmatrix}\mathbf{R}\\ 0\end{bmatrix}\mathbf{x} \approx \mathbf{b}$.

- Now matrix $\mathbf{Q}$ is orthogonal, i.e. multiplying by it preserves the norm: $\|\mathbf{Qv}\|_2^2 = (\mathbf{Qv})^T\mathbf{Qv} = \mathbf{v}^T\mathbf{Q}^T\mathbf{Qv} = \mathbf{v}^T\mathbf{v} = \|\mathbf{v}\|_2^2$.
- Thus, the following equation is equivalent with the original one:

  $\mathbf{Q}^T\mathbf{Q}\begin{bmatrix}\mathbf{R}\\ 0\end{bmatrix}\mathbf{x} \approx \mathbf{Q}^T\mathbf{b} \quad \Rightarrow \quad \begin{bmatrix}\mathbf{R}\\ 0\end{bmatrix}\mathbf{x} \approx \mathbf{Q}^T\mathbf{b}$.

Modeling of data: general linear fitting

- Because of the triangular form of matrix $\begin{bmatrix}\mathbf{R}\\ 0\end{bmatrix}$ we can decompose the right hand side as $\mathbf{b} \approx \begin{bmatrix}\mathbf{c}_1\\ \mathbf{c}_2\end{bmatrix}$, where $\mathbf{c}_1$ is an N-vector and $\mathbf{c}_2$ an $M-N$-vector. The main point is that $\mathbf{c}_2$ does not depend on $\mathbf{x}$.

- The equation is now $\begin{bmatrix}\mathbf{R}\\ 0\end{bmatrix}\mathbf{x} \approx \begin{bmatrix}\mathbf{c}_1\\ \mathbf{c}_2\end{bmatrix}$.

- The residual norm becomes now:

  $$\|\mathbf{b}-\mathbf{Ax}\|_2^2 = \left\|\begin{bmatrix}\mathbf{c}_1\\ \mathbf{c}_2\end{bmatrix} - \begin{bmatrix}\mathbf{R}\\ 0\end{bmatrix}\mathbf{x}\right\|_2^2 = \|\mathbf{c}_1-\mathbf{Rx}\|_2^2 + \|\mathbf{c}_2\|_2^2.$$

- Because $\mathbf{c}_2$ does not depend on $\mathbf{x}$ we obtain the minimum by putting the first term to zero:

  $\mathbf{Rx} = \mathbf{c}_1$.

- The minimum norm is $\|\mathbf{c}_2\|_2^2$.
- In practice the QR decomposition is done by orthogonal transformations like Housholder reflections to columns of $\mathbf{A}$:

  $\mathbf{H}_N\mathbf{H}_{N-1}\cdots\mathbf{H}_2\mathbf{H}_1\mathbf{A} = \mathbf{R}$.

- The right hand size is transformed accordingly:

  $$\mathbf{H}_N\mathbf{H}_{N-1}\cdots\mathbf{H}_2\mathbf{H}_1\mathbf{b} = \begin{bmatrix}\mathbf{c}_1\\ \mathbf{c}_2\end{bmatrix}$$

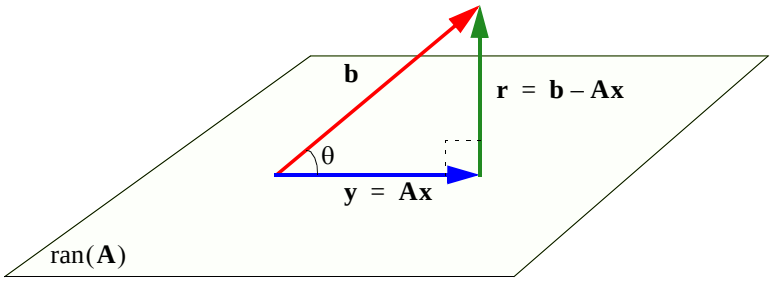
Modeling of data: general linear fitting

- Geometrical interpretation of the linear least squares problem in 3D/2D:

original equation $\mathbf{Ax} \approx \mathbf{b}$ or $\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \approx \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$

- The vector produced from transform $\mathbf{Ax}$ spans a subspace of R^3 (a plane; easily demonstrated with Matlab).
  - The *range* of matrix $\mathbf{A}$ is defined as the space spanned by the column vectors of $\mathbf{A}$ or

$$\text{ran}(\mathbf{A}) = \{\mathbf{y} \in R^3;\ \mathbf{y} = \mathbf{Ax};\ \mathbf{x} \in R^2\}.$$



Note that $\mathbf{y} \perp \mathbf{r}$ because:

$$(\mathbf{Ax})^T(\mathbf{b} - \mathbf{Ax}) = (\mathbf{Ax})^T\mathbf{b} - (\mathbf{Ax})^T\mathbf{Ax} = \mathbf{x}^T\mathbf{A}^T\mathbf{b} - \mathbf{x}^T\mathbf{A}^T\mathbf{Ax},$$

and the normal equation tells us that

$$(\mathbf{A}^T\mathbf{A})\mathbf{x} = \mathbf{A}^T\mathbf{b}$$

$$\Rightarrow \quad (\mathbf{Ax})^T(\mathbf{b} - \mathbf{Ax}) = 0$$

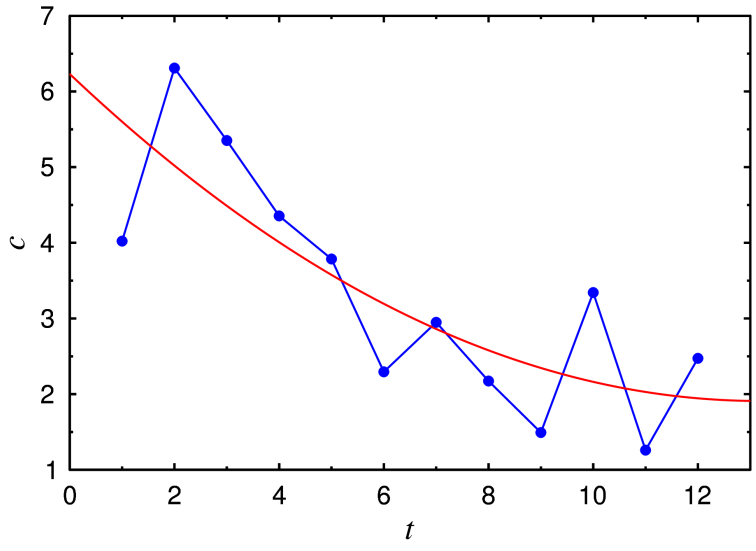
$$\cos\theta = \frac{\|\mathbf{y}\|_2}{\|\mathbf{b}\|_2}$$

One can show that the error in the solution behaves as

$$\frac{\|\Delta\mathbf{x}\|_2}{\|\mathbf{x}\|_2} \le \frac{\kappa(\mathbf{A})}{\cos\theta}\frac{\|\Delta\mathbf{b}\|_2}{\|\mathbf{b}\|_2}.$$

Modeling of data: general linear fitting

- A Matlab example[1]: water pumped through a container where dye is added. Concentration of dye as a function of time is measured. (blue curve). A second order polynomial is fitted to the data (red curve).



1. Data from example 6.4 in Kahaner, Moler, Nash: *Numerical Methods and Software*.

Modeling of data: general linear fitting

Original data:

```
    1.0000    4.0225
    2.0000    6.3095
    3.0000    5.3522
    4.0000    4.3553
    5.0000    3.7861
    6.0000    2.2947
    7.0000    2.9492
    8.0000    2.1732
    9.0000    1.4921
   10.0000    3.3424
   11.0000    1.2596
   12.0000    2.4732
```

Create the matrix A:

```
A=[t.^0 t.^1 t.^2]
A =
     1     1     1
     1     2     4
     1     3     9
     1     4    16
     1     5    25
     1     6    36
     1     7    49
     1     8    64
     1     9    81
     1    10   100
     1    11   121
     1    12   144
```

QR decomposition of A:

```
[R,c]=qrsteps(A,b)
R =
   -3.4641  -22.5167 -187.6388
         0   11.9583  155.4574
         0         0   36.5331
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
         0         0         0
c =
  -11.4922
   -3.9747
    0.9072
    0.3239
    0.2059
   -0.8364
    0.2649
   -0.0664
   -0.3049
    1.9857
    0.3411
    1.9906
```

Solve the equation:

```
y=R\c
y =
    6.2312
   -0.6552
    0.0248
```

Residual (transformed):

```
R*y - c
ans =
    0.0000
    0.0000
    0.0000
   -0.3239
   -0.2059
    0.8364
   -0.2649
    0.0664
    0.3049
   -1.9857
   -0.3411
   -1.9906
```

Residual (original):

```
A*y-b
ans =
    1.5783
   -1.2894
   -0.8631
   -0.3476
   -0.2101
    0.8993
   -0.0876
    0.4057
    0.8537
   -1.1799
    0.7691
   -0.5285
```

Matlab function **`qrsteps`** obtained from Cleve Moler: *Numerical Computing with Matlab*, http://www.mathworks.com/moler/chapters.html

Modeling of data: general linear fitting

- When the problem is ill-conditioned singular value decomposition of the matrix $\mathbf{A}$ may be of help.
- SVD of an $M \times N$ matrix $\mathbf{A}$ ($M \geq N$[1]) has the form

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T \text{ (or sometimes expressed in equivalent form } \mathbf{U}^T\mathbf{A}\mathbf{V} = \mathbf{S})$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal ($\mathbf{U}$ is $M \times M$, $\mathbf{V}$ is $N \times N$):

$$\mathbf{U}^T\mathbf{U} = \mathbf{U}\mathbf{U}^T = \mathbf{1}_M, \quad \mathbf{V}\mathbf{V}^T = \mathbf{V}^T\mathbf{V} = \mathbf{1}_N,$$

$$\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_M], \ \mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N]$$

and

$$\mathbf{S} = \begin{bmatrix} \sigma_1 & & 0 \\ & \sigma_2 & \\ & & \ldots \\ 0 & & \sigma_N \\ & 0 & \end{bmatrix} \quad (M \times N).$$

- Moreover, $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_N \geq 0$. σ_i are the singular values of $\mathbf{A}$.
- The smallest singular value σ_N is the distance (in the 2-norm) from $\mathbf{A}$ to the nearest degenerate (singular) matrix.
- The number of non-zero σ's is equal to the **rank** of the matrix.

  (Rank of an $M \times N$ matrix $\mathbf{A}$ is the dimension of its range. Range of a $\mathbf{A}$ is the set of all M-vectors $\mathbf{A}\mathbf{x}$ where $\mathbf{x}$ is an N-vector.)

1. This is not required for a SVD to exist. In least squares, however, we always have more equations than variables.

Modeling of data: general linear fitting

- If $\mathbf{A}$ is singular then at least $\sigma_N = 0$.
- SVD can tell many things about the matrix; assume we have, for matrix $\mathbf{A}$, $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > \sigma_{r+1} = \ldots = \sigma_N = 0$; i.e. all singular values from σ_{r+1} are zero.
- Then one can show that

$$\begin{cases} \text{rank}(\mathbf{A}) = r \\ \text{null}(\mathbf{A}) = \text{span}\{\mathbf{v}_{r+1}, \mathbf{v}_{r+2}, \ldots, \mathbf{v}_N\} \\ \text{ran}(\mathbf{A}) = \text{span}\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_r\} \end{cases}$$

$\text{ran}(\mathbf{A}) = \{\mathbf{y} \in R^M;\ \mathbf{y} = \mathbf{A}\mathbf{x};\ \mathbf{x} \in R^N\}$
(In some books range.)

$\text{rank}(\mathbf{A}) = \dim(\text{ran}(\mathbf{A}))$

$\text{null}(\mathbf{A}) = \{\mathbf{x} \in R^N;\ \mathbf{A}\mathbf{x} = 0\}$

- Various norm properties are related to SVD:

$$\|\mathbf{A}\|_F^2 = \sum_{i=1}^{p} \sigma_i^2 \qquad p = \min\{M, N\}$$

$$\|\mathbf{A}\|_2 = \sigma_1 \quad .$$

$$\min_{\mathbf{x} \neq 0} \frac{\|\mathbf{A}\mathbf{x}\|_2}{\|\mathbf{x}\|_2} = \sigma_N \qquad (M \geq N)$$

- Matrix $\mathbf{A}$ can be expressed as a **SVD expansion**:

$$\mathbf{A} = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$$

- In many cases $\mathbf{A}$ is not exactly singular but almost.
- In this case some of the singular values are small.
- The ratio σ_1 / σ_N is a similar measure of the singularity of the matrix as the condition number.

Modeling of data: general linear fitting

- To express this *near rank-deficiency* exactly the concept of ε-rank can be used

$$\text{rank}(\mathbf{A}, \varepsilon) = \min_{\|\mathbf{A}-\mathbf{B}\|_2 \leq \varepsilon} \text{rank}(\mathbf{B})$$

- If we define

$$\mathbf{A}_k = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^T, \text{ where } k < r = \text{rank}(\mathbf{A}),$$

then one can prove that

$$\min_{\text{rank}(\mathbf{B}) = k} \|\mathbf{A} - \mathbf{B}\|_2 = \|\mathbf{A} - \mathbf{A}_k\|_2 = \sigma_{k+1}.$$

- Also, one can show that if $r_\varepsilon = \text{rank}(\mathbf{A}, \varepsilon)$ then

$$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{r_\varepsilon} > \varepsilon \geq \sigma_{r_\varepsilon + 1} \geq \ldots \geq \sigma_p,\ p = \min\{M, N\}.$$

- With SVD one can define so called *pseudo-inverse* $\mathbf{A}^+$:

$$\mathbf{A}^+ = \mathbf{V}\mathbf{S}^+\mathbf{U}^T,$$

$$\mathbf{S}^+ = \begin{bmatrix} 1/\sigma_1 & & & 0 \\ & 1/\sigma_2 & & \\ & & \ldots & \\ 0 & & & 1/\sigma_r \\ & & 0 & \end{bmatrix} \in R^{N \times M},\ r = \text{rank}(\mathbf{A}).$$

- Now the LS solution of the equation $\mathbf{A}\mathbf{x} = \mathbf{b}$ can be expressed as $\mathbf{x}_{\text{LS}} = \mathbf{A}^+\mathbf{b}$.
- $\mathbf{A}^+$ is the solution for the minimization problem $\min_{\mathbf{X} \in R^{N \times M}} \|\mathbf{A}\mathbf{X} - \mathbf{1}_M\|_F$.

Modeling of data: general linear fitting

- SVD can be written in vector form as below

$$\mathbf{AV} = \mathbf{US} \quad \Rightarrow \quad \mathbf{A}\mathbf{v}_i = \sigma_i \mathbf{u}_i,$$

or

$$\mathbf{A}^T\mathbf{U} = \mathbf{V}\mathbf{S}^T \quad \Rightarrow \quad \mathbf{A}^T\mathbf{u}_i = \sigma_i \mathbf{v}_i.$$

- Compare these with the eigenvalue problem

$$\mathbf{A}\mathbf{x}_i = \lambda_i \mathbf{x}_i.$$

- SVD is the “eigenvalue problem of nonsquared matrices”.

Modeling of data: general linear fitting

- How is all this related to linear fitting?

- Well, first change the notation sligthly (*Bear with me!*):

Our data is (t_i, b_i), $i = 1, \ldots, M$

Basis functions are $\phi_j(t_i)$, $j = 1, \ldots, N$

Parameters are x_j, $j = 1, \ldots, N$

So, the approximation we seek is $b_i \approx x_1\phi_1(t_i) + x_2\phi_2(t_i) + \ldots + x_N\phi_N(t_i)$.

In matrix form $\mathbf{b} \approx \mathbf{A}\mathbf{x}$, where $A_{ij} = \phi_j(t_i)$.

- In other words we have a minimization problem: we have to find

$$\min_{\mathbf{x}} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 = \min_{\mathbf{x}} \left[\sum_{i=1}^{M} [(\mathbf{b} - \mathbf{A}\mathbf{x})]_i^2 \right]$$

or we have to minimize the residual.

Modeling of data: general linear fitting

- This norm can be written in terms of SVD of $\mathbf{A}$ as (remember $\mathbf{U}$ is orthogonal)

$$\|\mathbf{Ax}-\mathbf{b}\|_2 = \left\|\mathbf{USV}^T\mathbf{x}-\mathbf{b}\right\|_2 = \left\|\mathbf{U}^T(\mathbf{USV}^T\mathbf{x}-\mathbf{b})\right\|_2 = \left\|\mathbf{SV}^T\mathbf{x}-\mathbf{U}^T\mathbf{b}\right\|_2$$

- If we denote

$\mathbf{d} = \mathbf{U}^T\mathbf{b}$, $\mathbf{z} = \mathbf{V}^T\mathbf{x}$ we get

$$\|\mathbf{Ax}-\mathbf{b}\|_2^2 = \|\mathbf{Sz}-\mathbf{d}\|_2^2 = \left\|\begin{bmatrix}\sigma_1 z_1 - d_1\\ \sigma_2 z_2 - d_2\\ \dots\\ \sigma_N z_N - d_N\\ -d_{N+1}\\ -d_{N+2}\\ \dots\\ -d_M\end{bmatrix}\right\|_2^2 = (\sigma_1 z_1 - d_1)^2 + \dots + (\sigma_N z_N - d_N)^2 + d_{N+1}^2 + \dots + d_M^2.$$

- If none of the singular values is zero we can get the minimum of the above norm by setting

$$z_i = \frac{d_i}{\sigma_i}.$$

Modeling of data: general linear fitting

- This gives the norm its minimum value

$$\|\mathbf{Ax}-\mathbf{b}\|_2^2 = d_{N+1}^2 + \dots + d_M^2.$$

- If $\sigma_N = 0$ then any choice of z_N gives the same minimum residual

$$\|\mathbf{Ax}-\mathbf{b}\|_2^2 = d_N^2 + d_{N+1}^2 + \dots + d_M^2.$$

- This means that the least squares problem doesn't have a unique solution.

- Usual convention is to set $z_i = 0$ whenever $\sigma_i = 0$.

- Singular values σ_i are nonzero only if the basis functions are linearly independent.
- Near linear dependence between basis functions implies a singular value close to zero.

- Thresholds for the singular values can be used as the tolerance of the fit.
  - Any σ_i greater than the threshold is acceptable and the corresponding parameter value is computed from $z_i = d_i / \sigma_i$.
  - Any σ_i smaller than the threshold value is deemed as negligible and it set to zero.

- The only drawback of the SVD method is that it is slower than solving the normal equations.

Modeling of data: general linear fitting

• SVD has many other applications. See any textbook on numerical methods.

• SVD can be calculated by

Matlab: `[U,S,V]=svd(A);`

LAPACK:
```
SUBROUTINE DGESVD(JOBU,JOBVT,M,N,A,LDA,S,U,LDU,VT,LDVT,WORK,LWORK,INFO)
```

GSL:
```
int gsl_linalg_SV_decomp(gsl_matrix *A, gsl_matrix *V, gsl_vector *S,
                         gsl_vector *WORK)
```


Modeling of data: nonlinear fitting

• Now we generalize the fitting problem to models where the dependence on the parameters a_j is nonlinear.

• Due to nonlinearity in minimizing the χ^2 we have to resort to iterative methods.

- As in function minimization in general, we start with a initial guess for the parameter vector $(a_1, a_2, \ldots, a_N)$ and by using our algorithm proceed in the parameter space until the minimum of χ^2 has been reached.

- Following the principles developed in Chapter 8 the function can be approximated as quadratic near the minimum

$$\chi^2(\mathbf{a}) \approx \gamma - \mathbf{d}^T\mathbf{a} + \frac{1}{2}(\mathbf{a}^T\mathbf{D}\mathbf{a}),$$

where $\mathbf{d}$ is a vector with N elements and $\mathbf{D}$ is an $N \times N$ matrix.

- Minimum of this function is found with one single step

$$\mathbf{a}^{(i+1)} = \mathbf{a}^{(i)} + \mathbf{D}^{-1}[-\nabla\chi^2(\mathbf{a}^{(i)})].$$

- If the χ^2 function is not well approximated by a quadratic function we can use e.g. the steepest descent direction

$$\mathbf{a}^{(i+1)} = \mathbf{a}^{(i)} - \mu\nabla\chi^2(\mathbf{a}^{(i)}),$$

where the constant μ is small enough not to start going uphill.

Modeling of data: nonlinear fitting

- The difference between minimization in fitting and function minimization in general is that now we are able to compute both the gradient and the Hessian matrix. (We built the model, didn't we!)

- Our model is

$$y = y(x;\mathbf{a}),$$

where the dependence of y on $\mathbf{a}$ is no more linear.

- FOM function is now

$$\chi^2(\mathbf{a}) = \sum_{i=1}^{M} \left[\frac{y_i - y(x_i;\mathbf{a})}{\sigma_i} \right]^2$$

- Components of the gradient $\nabla\chi^2(\mathbf{a})$ are

$$\frac{\partial \chi^2}{\partial a_k} = -2 \sum_{i=1}^{M} \left[\frac{y_i - y(x_i;\mathbf{a})}{\sigma_i^2} \right] \left[\frac{\partial y(x_i;\mathbf{a})}{\partial a_k} \right], \quad k = 1, 2, \ldots, N.$$

- Elements of the Hessian matrix are

$$D_{kl} = \frac{\partial^2 \chi^2}{\partial a_k \partial a_l} = 2 \sum_{i=1}^{M} \frac{1}{\sigma_i^2} \left[\left(\frac{\partial y(x_i;\mathbf{a})}{\partial a_k} \right) \left(\frac{\partial y(x_i;\mathbf{a})}{\partial a_l} \right) - [y_i - y(x_i;\mathbf{a})] \left(\frac{\partial^2 y(x_i;\mathbf{a})}{\partial a_k \partial a_l} \right) \right].$$

Modeling of data: nonlinear fitting

- Let's drop those 2's by defining

$$\beta_k = -\frac{1}{2} \frac{\partial \chi^2}{\partial a_k}, \qquad \alpha_{kl} = \frac{1}{2} \frac{\partial^2 \chi^2}{\partial a_k \partial a_l}.$$

so that

$$\alpha = \frac{1}{2}\mathbf{D}.$$

- Now the iteration equation $\mathbf{a}^{(i+1)} = \mathbf{a}^{(i)} + \mathbf{D}^{-1}[-\nabla\chi^2(\mathbf{a}^{(i)})]$ can be written in the form

$$\sum_{l=1}^{N} \alpha_{kl} \delta a_l = \beta_k. \qquad \textbf{(1)}$$

where $\delta a_l = a_l^{(i+1)} - a_l^{(i)}$ is the movement in one iteration step.

- From equation (1) we solve δa_l and add it to the current position, thus giving us the new point.

- The steepest decent step takes the form

$$\delta a_l = \mu \beta_l. \qquad \textbf{(2)}$$

Modeling of data: nonlinear fitting

- Hessian matrix contains the second derivatives of χ^2 with respect to the parameters a_i.
- In most algorithms these terms are dropped.
  - This can be justified by observing that the coefficient of the term $\partial^2 y / \partial a_k \partial a_l$ is $[y_i - y(x_i)]$ which is essentially (for a successful model) the randomly distributed error of the data.
  - Because of this we can assume that these terms more or less cancel out each other.
  - In case of outliers keeping the second derivatives can make the algorithm unstable.

- So, we redefine matrix α:

$$\alpha_{kl} = 2 \sum_{i=1}^{M} \frac{1}{\sigma_i^2} \left(\frac{\partial y(x_i;\mathbf{a})}{\partial a_k} \right) \left(\frac{\partial y(x_i;\mathbf{a})}{\partial a_l} \right).$$

- Changing α this way does not change the end results; only the route in the $\mathbf{a}$ space that takes us to the minimum is changed.

- Note that dropping those second derivatives from the Hessian results in also by assuming that the model $y = y(x;\mathbf{a})$ is linear in $\mathbf{a}$.

Modeling of data: nonlinear fitting

- In the **Levenberg and Marquardt** (**LM**) method the steepest descent is used when far away from the minimum
- When the minimum of the χ^2 is approached the method gradually shifts to the (approximate) inverse Hessian method.

- The value of the constant μ in equation **(2)** is estimated by the Hessian matrix:
  - Dimension of μ is a_k^2.
  - The only element in the Hessian matrix α with this dimension is $1/\alpha_{kk}$.
  - Let's assume that this element tells us the 'scale' of the problem.
  - Moreover, in oder not to do too long jumps we divide it by a number $\lambda \gg 1$.
  - So we get the equation **(2)** to form

$$\delta a_l = \frac{1}{\lambda \alpha_{ll}} \beta_l. \qquad \textbf{(3)}$$

- Combining the inverse Hessian and SD methods goes like this:

  - Let's define the matrix α':

$$\alpha'_{jj} = \alpha_{jj}(1+\lambda)$$
$$\alpha'_{jk} = \alpha_{jk} \qquad (j \neq k)$$

  - Replace equations **(1)** and **(3)** by one equation

$$\sum_{l=1}^{N} \alpha'_{kl} \delta a_l = \beta_k. \qquad \textbf{(4)}$$

Modeling of data: nonlinear fitting

- If λ is large equation **(4)** approaches equation **(3)** while for small λ we obtain the Hessian equation **(1)**

- The **LM** algorithm is then the following (the initial parameter vector is $\mathbf{a}$)

1. Compute $\chi^2(\mathbf{a})$
2. Set a small value to λ. E.g. $\lambda = 0.001$.
3. Solve $\delta\mathbf{a}$ from equation **(4)** and compute $\chi^2(\mathbf{a}+\delta\mathbf{a})$.
4. If $\chi^2(\mathbf{a}+\delta\mathbf{a}) \geq \chi^2(\mathbf{a})$, increase λ ($\lambda \leftarrow 10 \cdot \lambda$) and go to step 3.
5. If $\chi^2(\mathbf{a}+\delta\mathbf{a}) < \chi^2(\mathbf{a})$ decrease λ: $\lambda \leftarrow \lambda/10$, update vector $\mathbf{a} \leftarrow \mathbf{a}+\delta\mathbf{a}$ and go to step 3.

- We also need a condition for stopping the iteration.
  - In practice stopping after χ^2 decreases only slightly (say 0.01) is a good measure of convergence.

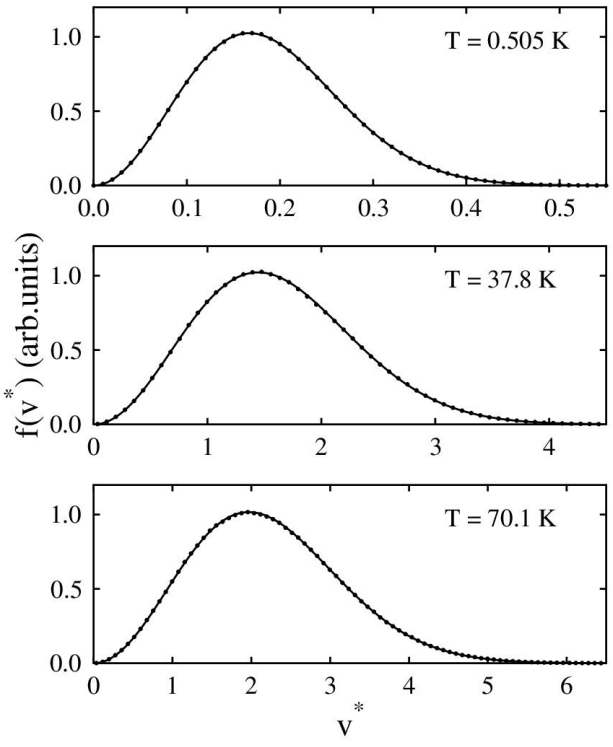
- After convergence the error estimates of the parameters can be computed from $\mathbf{C} = \alpha^{-1}$.

Modeling of data: nonlinear fitting

- As an example below are the fits of Maxwell-Boltzmann distribution

$$y = ax^2 e^{-x^2/(2b)}$$

to the data obtained from the molecular dynamics simulation of solid Lennard-Jones (in this case Ne) material.




```
vdfit> ../fitf/fitf vd70.dat boltzmann
-------------------------- fitf ---------------------------
Fitting function boltzmann to experimental file vd70.dat
      y=a*x**2*exp(-x**2/2.0/b)

> fitfunc  vd70.dat 2 1 1

    a             b           CHI^2          Lambda
 -----------   -----------   ------------ ------------
  1.0000000     1.0000000    0.16260163E-02 0.10000000E-02
  10.551674     5.1415530     19498.413      1000.0000
  10.550628     5.1404933     19498.392      100.00000
  10.540944     5.1300143     19498.190      10.000000
  10.509482     5.0350311     19496.523      1.0000000
  12.569750     4.3980820     19487.114     0.10000000
  19.954605     3.8127834     19471.618     0.10000000
  41.200786     3.2570494     19434.872     0.10000000
  111.62496     2.7146669     19334.445     0.10000000
  428.58148     2.1554226     18994.058     0.10000000
  2671.5811     1.5655949     17473.904     0.10000000
  17594.094     1.5383770     9299.2787     0.10000000
  48717.769     0.98988380    4880.1334     0.10000000E-01
  88756.809     1.0996447     193.74644     0.10000000E-02
  89438.680     1.0468608     2.1225936     0.10000000E-03
  89969.741     1.0425107     1.6102341     0.10000000E-04
  89974.933     1.0424922     1.6102060     0.10000000E-05
  89974.941     1.0424922     1.6102060     0.10000000E-06

    sigmaa        sigmab      CHI^2(ABS)
 -----------   -----------   ------------
  41.107902    0.24614145E-03  990.27666
```

Modeling of data: nonlinear fitting

- Implementations of the **LM** algorithm:

GSL: **`Derivative Solver: gsl_multifit_fdfsolver_lmsder`**
`This is a robust and efficient version of the Levenberg-Marquardt algorithm as implemented in the scaled LMDER routine in MINPACK. Minpack was written by Jorge J. More', Burton S. Garbow and Kenneth E. Hillstrom`

Matlab: **`X=LSQCURVEFIT(FUN,X0,XDATA,YDATA)`** `starts at X0 and finds coefficients X to best fit the nonlinear functions in FUN to the data YDATA (in the least-squares sense).`

SLATEC: **`DNLS1E:`** `The purpose of the routine is to minimize the sum of the squares of M nonlinear functions in N variables by a modification of the Levenberg-Marquardt algorithm.`

- **SLATEC** is a collection of Fortran (F77!) routines doing various kinds of numerical tasks.
  - It can be downloaded from www.netlib.org or www.csit.fsu.edu/~burkardt/f_src/slatec/slatec.html.
  - If you download it from the latter place you need also the F90 `f90split` program to split the one big file to separate subroutines (http://www.csit.fsu.edu/~burkardt/f_src/f90split/f90split.html).
  - Using the scripts you can create a library (.a file) that can be linked with your main program.

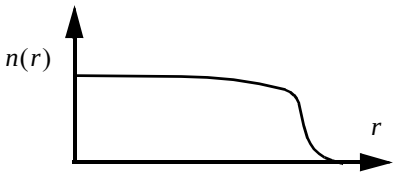
Modeling of data: nonlinear fitting

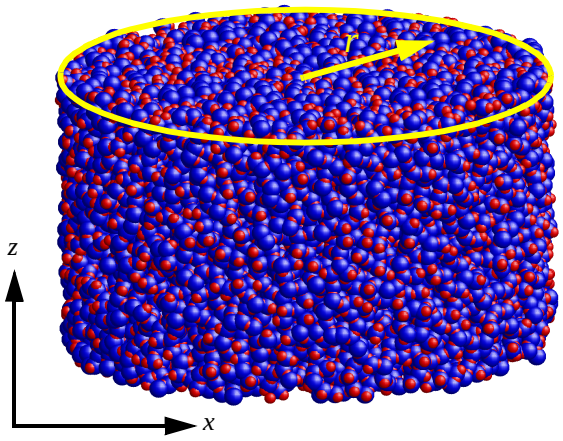
- Matlab Curve Fitting Toolbox is a nice tool for nonlinear fitting.

  - Example: In order to obtain a measure of a width of a stretched cylindrical SiO_2 beam fit a modified Fermi function to the atomic density as a function of distance from the center line if the beam:

$$n(r) = \frac{a - br}{e^{(r-c)/d} + 1}.$$

  (This width is in turn used to calculate the Poisson's ratio of the beam: $\nu = -\varepsilon_x / \varepsilon_z$.)



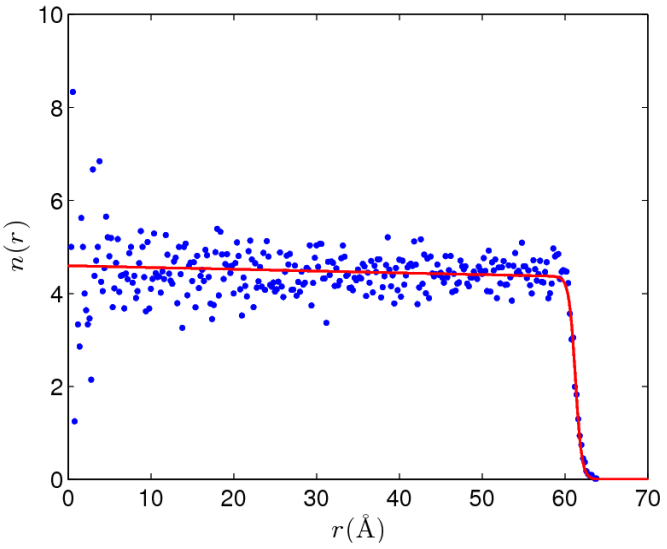


Modeling of data: nonlinear fitting

- Below is listed the Matlab script doing the fit and the results.

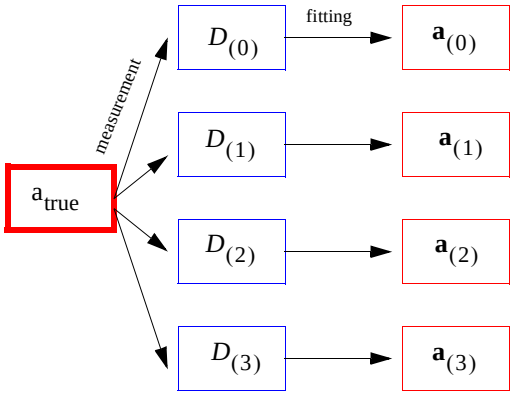
```
%
% Script fits a modified Fermi function
%
%                 a-b*r
%     d(r) = ----------------
%            exp((r-c)/d)+1
%
% to the radial distributions of atoms in a cylindrical beam.
%
% Antti Kuronen, 2009, antti.kuronen@helsinki.fi
%

n0=10;
g=fittype('(a-b*x)/(exp((x-c)/d)+1)','coeff',{'a','b','c','d'});
d=load('dist.dat');
[fp,gof,output]=...
fit(d(n0:end,1),d(n0:end,2),g,'Startpoint',[5 0.1 60 1]);
fprintf('a %g  b %g  c %g  d %g\n',fp.a,fp.b,fp.c,fp.d);

h=figure;
hold on;

h1=plot(d(:,1),d(:,2),'b.');
set(h1,'Marker','o');
set(h1,'MarkerSize',4);
set(h1,'MarkerFaceColor','blue');

h2=plot(fp);
set(h2,'LineWidth',2.0)
legend off;

ha=gca;
set(ha,'FontSize',14);
set(get(ha,'XLabel'),'String','{\it r} (Å)','FontSize',14);
set(get(ha,'YLabel'),'String','{\it n} ({\it r})','FontSize',14)
```

```
>> fitml
a 4.44564  b 9.88383e-05  c 61.2408  d 0.398287
```



Modeling of data: parameter errors

• According to the figure below we can think that there exists a set of true parameter values $\mathbf{a}_{\text{true}}$.



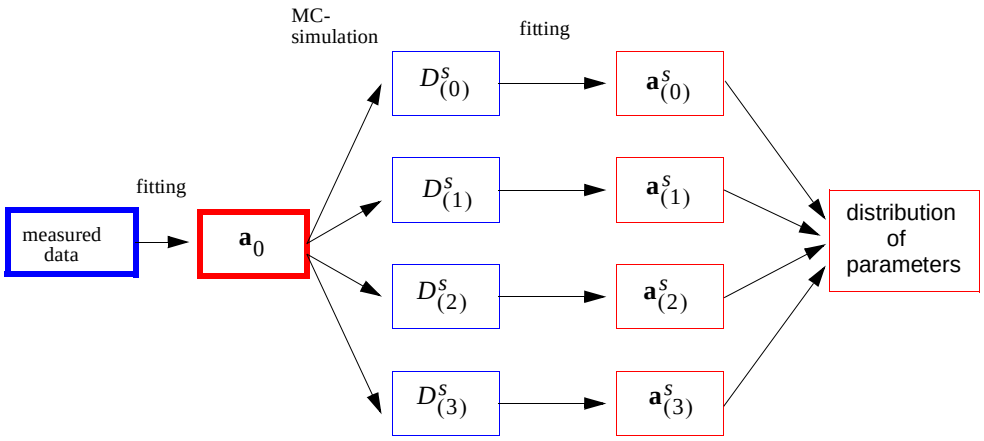
- Based on these parameters we can generate many data sets that fit the model but have random errors.

- This means that also the parameters $\mathbf{a}_{(i)}$ obtained by fitting individual data sets are different.

- You can also do many measurements to get error estimates of the parameters but this is in most cases not feasible.

Modeling of data: parameter errors

- What you can do is generate artificial data sets by using Monte Carlo: this is called the **bootstrap method**:



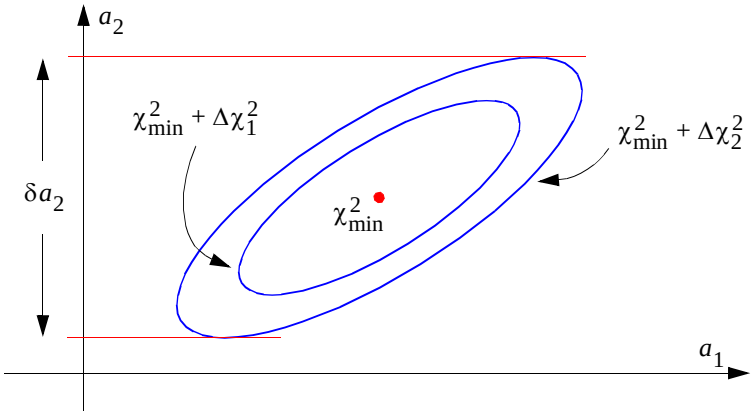
- Note that we must know the distribution of the errors in the data points in order to generate the data.

Modeling of data: parameter errors

• We can also use the function $\chi^2(\mathbf{a})$ as a basis for the error estimation.

- Confidence limits for the parameters can be computed as constant-χ^2 boundaries.
- Assume that the $\Delta\chi^2_{\min}$ has the χ^2 distribution with $M-N$ degrees of freedom. With a reasonable fit one can show that the parameters are normally distributed:

$$P(\delta\mathbf{a}) \propto \exp\left[\left(-\frac{1}{2}\right)((\delta\mathbf{a})^T \alpha \delta\mathbf{a})\right].$$



- Moreover, the quantity $\Delta\chi^2 = \chi^2(\mathbf{a}_{(j)}) - \chi^2(\mathbf{a}_{\text{true}})$ is distributed as a χ^2 distribution with M degrees of freedom. Here $\mathbf{a}_{\text{true}}$ is the true parameter vector and $\mathbf{a}_{(j)}$ one realization of it.
- Confidence region of one parameter a_i can be expressed in terms of the matrix $\mathbf{C} = \alpha^{-1}$ as

$$\delta a_i = \pm\sqrt{\Delta\chi^2}\sqrt{C_{ii}},$$

where $\Delta\chi^2$ is now the change in χ^2 that defines the confidence level.

- In the case of fitting by using SVD the elements of the matrix $\mathbf{C}$ are simply obtained as

$C_{jk} = \sum_{i=1}^{N} (V_{ji}V_{ki})/\sigma_i^2$, where σ_i is now the ith singular value.

- For determining **joint confidence regions** for more than one parameter see e.g. Numerical Recipes.

$\Delta\chi^2$ as a Function of Confidence Level and Degrees of Freedom

	ν					
p	1	2	3	4	5	6
68.3%	1.00	2.30	3.53	4.72	5.89	7.04
90%	2.71	4.61	6.25	7.78	9.24	10.6
95.4%	4.00	6.17	8.02	9.70	11.3	12.8
99%	6.63	9.21	11.3	13.3	15.1	16.8
99.73%	9.00	11.8	14.2	16.3	18.2	20.1
99.99%	15.1	18.4	21.1	23.5	25.7	27.8

From *Numerical Recipes*.

Modeling of data: data smoothing

• **Data smoothing** can be done by various means.

- By *fitting* a polynomial to the data set. Of course, the degree of the polynomial must be lower than the number of data points.

- By constructing a spline that has a restricted 'bending energy' (average curvature) but nonetheless goes near the data points. This can be accomplished by minimizing the following quantity

$$W = \rho\chi^2 + \int [S''(x)]^2 dx\,,$$

where χ^2 is the good old chi-squared and the last term gives the approximate average curvature.

- Parameter ρ controls the relative weight of the two terms:

with $\rho = 0$ we get a straight line

with $\rho \to \infty$ we get a normal spline interpolant.

Modeling of data: robust estimation

• In robust estimation we aim to diminish the effect of outlier points to the result of the fit.

- It might be that we know that the distribution of our data comprises of a narrow peak and a broad tail of outliers.
- The idea in robust estimation is to write the probability distribution not as

$$P \propto \prod_{i=1}^{M} \left\{ \exp\left[-\frac{1}{2}\left(\frac{y_i - y(x_i, \mathbf{a})}{\sigma_i} \right)^2 \right] \Delta y \right\}$$

but

$$P \propto \prod_{i=1}^{M} \{ \exp[-\rho(y_i, y(x_i, \mathbf{a}))] \Delta y \}\,.$$

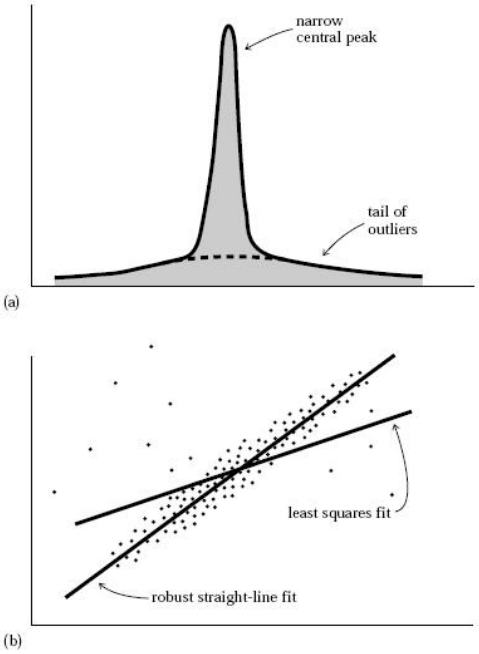
- In fitting we then want to minimize

$$\sum_{i=1}^{M} \rho(y_i, y(x_i, \mathbf{a}))\,.$$

- Often the argument of ρ is of the form $z \equiv \dfrac{y_i - y(x_i, \mathbf{a})}{\sigma_i}$.

- Defining $\psi(z) = \dfrac{d\rho(z)}{dz}$ we get the minimization equations as

$$\sum_{i=1}^{M} \frac{1}{\sigma_i} \psi\left(\frac{y_i - y(x_i, \mathbf{a})}{\sigma_i} \right) \frac{\partial y(x_i, \mathbf{a})}{\partial a_k} = 0\,, \quad k = 1, \ldots, N\,.$$



Numerical Recipes, Fig. 15.7.1

Modeling of data: robust estimation

- For normal distribution we get

$$\rho(z) = \frac{1}{2}z^2, \ \psi(z) = z.$$

- If we have errors distributed as double exponential

$$P \propto \prod_{i=1}^{M} \left\{ \exp\left[-\left| \frac{y_i - y(x_i, \mathbf{a})}{\sigma_i} \right| \right] \Delta y \right\}$$

we get

$$\rho(z) = |z|, \ \psi(z) = \mathrm{sign}(z).$$

- Sometimes a Lorentzian distribution is appropriate:

$$P \propto \prod_{i=1}^{M} \left\{ \frac{1}{1 + \frac{1}{2}\left(\frac{y_i - y(x_i, \mathbf{a})}{\sigma_i} \right)^2} \Delta y \right\},$$

which gives

$$\rho(z) = \ln(1 + z^2/2), \ \psi(z) = \frac{z}{1 + z^2/2}.$$