Statistical Methods in Particle Physics

6. Method of Least Squares

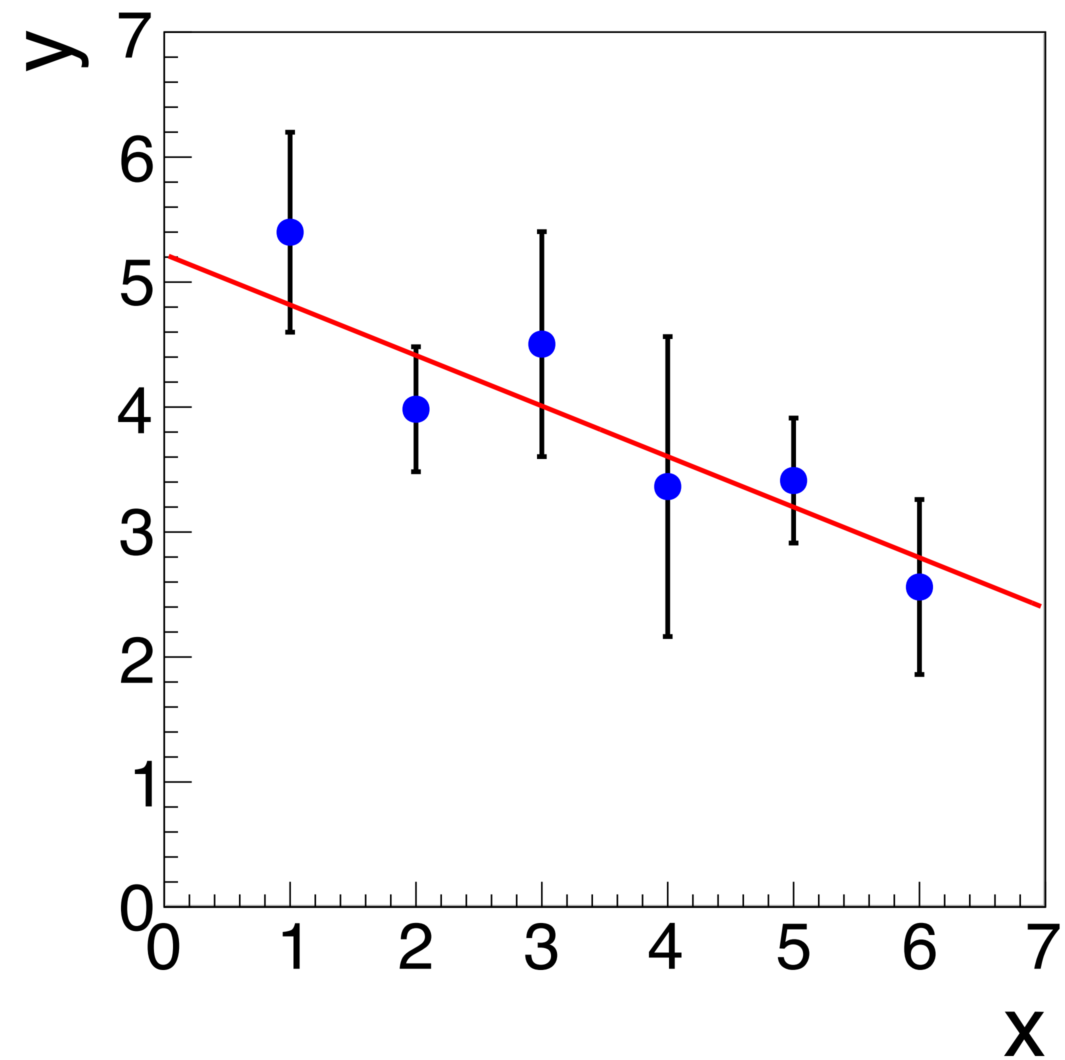
Heidelberg University, WS 2020/21

Klaus Reygers (lectures)
Rainer Stamen, Martin Völkl (tutorials)

Least squares from ML (1)

Consider n measured values $y_1(x_1)$, $y_2(x_2)$, … , $y_n(x_n)$ assumed to be independent Gaussian random variables with known variances:

$$V[y_i] = \sigma_i^2$$



Assume we have a function f with

$$E[y_i] = f(x_i; \vec{\theta})$$

We want to estimate $\vec{\theta}$

Likelihood function:

$$L(\vec{\theta}) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left[-\frac{1}{2}\left(\frac{y_i - f(x_i;\vec{\theta})}{\sigma_i}\right)^2\right]$$

Least squares from ML (2)

Log-likelihood function:

$$\ln L(\vec{\theta}) = -\frac{1}{2}\sum_{i=1}^{n}\left(\frac{y_i - f(x_i;\vec{\theta})}{\sigma_i}\right)^2 + \text{terms not depending on } \vec{\theta}$$

So maximizing the likelihood is equivalent to minimizing

$$\chi^2(\vec{\theta}) = \sum_{i=1}^{n}\left(\frac{y_i - f(x_i;\vec{\theta})}{\sigma_i}\right)^2$$

Minimizing χ^2 is called the method of least squares, goes back to Gauss and Legendre.

In other words, for Gaussian uncertainties the method of least squares coincides with the maximum likelihood method.

Minimization:

$$\frac{\partial \chi^2}{\partial \theta_j} = 0, \qquad j = 1, \ldots, m$$

Number of parameters

The χ^2 minimization is often done numerically, e.g., using the MINUIT code

https://en.wikipedia.org/wiki/MINUIT

Generalized least squares for correlated y_i

Suppose the y_i have a covariance matrix V and follow a multi-variate Gaussian:

$$g(\vec{y};\vec{\mu},V) = \frac{1}{(2\pi)^{n/2}|V|^{1/2}} \exp\left[-\frac{1}{2}(\vec{y}-\vec{\mu})^\mathsf{T} V^{-1}(\vec{y}-\vec{\mu})\right]$$

The generalized least-squares method then corresponds to minimizing:

$$\chi^2(\vec{\theta}) = (\vec{y}-\vec{f}(\vec{x};\vec{\theta}))^T V^{-1}(\vec{y}-\vec{f}(\vec{x};\vec{\theta}))$$

$$\vec{f}(\vec{x};\vec{\theta}) = (f(x_1;\vec{\theta}), \ldots, f(x_n;\vec{\theta}))$$

We can write this also as

$$\chi^2(\vec{\theta}) = \sum_{i,j}(y_i - f(x_i;\vec{\theta}))^T (V^{-1})_{ij}(y_j - f(x_j;\vec{\theta}))$$

Variance of the least squares estimator

Using

$$\chi^2(\vec{\theta}) = -2 \ln L(\theta) + \mathrm{const.}$$

we can use the result for the variance of the ML estimators and obtain

$$V[\widehat{\vec{\theta}}] \approx 2 \left[\left. \frac{\partial^2 \chi^2(\vec{\theta})}{\partial^2 \vec{\theta}} \right|_{\vec{\theta}=\widehat{\vec{\theta}}} \right]^{-1} \quad \text{i.e.} \quad (V^{-1}[\widehat{\vec{\theta}}])_{ij} = \frac{1}{2} \left. \frac{\partial^2 \chi^2(\vec{x};\vec{\theta})}{\partial \theta_i \partial \theta_j} \right|_{\vec{\theta}=\widehat{\vec{\theta}}}$$

Or determine 1σ uncertainties from the contour where

$$\chi^2(\vec{\theta}') = \chi^2_{\mathrm{min}} + 1$$

For z·σ uncertainties the condition is

$$\chi^2(\vec{\theta}') = \chi^2_{\mathrm{min}} + z^2$$

Linear least squares

Consider n data points y_i whose uncertainties and correlations are described by a covariance matrix V. The y_i are measured at points x_i.

We would like to fit a linear combination of m functions a_j(x) to the data:

$$f(x;\vec{\theta}) = \sum_{j=1}^{m} \theta_j a_j(x)$$

n data points y_i

m parameters θ_j

examples:

$f(x) = \theta_0 + \theta_1 x + \theta_2 x^2$

$f(x) = \theta_0 + \theta_1 \cos(x)$

The linear least squares problem can be solved in closed form:

Define $n \times m$ matrix A: $A_{i,j} = a_j(x_i)$ "design matrix"

Minimize $\chi^2 = (\vec{y} - A\vec{\theta})^\mathsf{T} V^{-1} (\vec{y} - A\vec{\theta}), \quad \vec{y} = (y_1, \ldots, y_n)$

best fit parameters:

$$\widehat{\vec{\theta}} = \underbrace{(A^\mathsf{T} V^{-1} A)^{-1}}_{\substack{\text{symmetric} \\ m\times m \text{ matrix}}} A^\mathsf{T} V^{-1} \vec{y}$$

covariance matrix of the parameters:

$$U = (A^\mathsf{T} V^{-1} A)^{-1}$$

Linear least squares: Derivation of the formula

$$\chi^2(\vec{\theta}) = (\vec{y} - A\vec{\theta})^\mathsf{T} V^{-1}(\vec{y} - A\vec{\theta}) = \vec{y} V^{-1}\vec{y} - 2\vec{y}^\mathsf{T} V^{-1} A\vec{\theta} + \vec{\theta}^\mathsf{T} A^\mathsf{T} V^{-1} A\vec{\theta}$$

Set derivatives w.r.t. θ_i to zero:

$$\nabla\chi^2 = -2(A^\mathsf{T} V^{-1}\vec{y} - A^\mathsf{T} V^{-1} A\vec{\theta}) = 0$$

$$\vec{\nabla}(\vec{a}^\mathsf{T} M\vec{x}) = M^\mathsf{T}\vec{a} \qquad \vec{\nabla}(\vec{x}^\mathsf{T} M\vec{x}) = (M^\mathsf{T} + M)\vec{x} \overset{M \text{ symm.}}{=} 2M\vec{x}$$

Solution:

$$\hat{\vec{\theta}} = (A^\mathsf{T} V^{-1} A)^{-1} A^\mathsf{T} V^{-1}\vec{y} \equiv L\vec{y}$$

Covariance matrix U of the parameters:

Here we use
$(XY)^\mathsf{T} = Y^\mathsf{T} X^\mathsf{T}$,
$[(A^\mathsf{T} V^{-1} A)^{-1}]^\mathsf{T} = (A^\mathsf{T} V^{-1} A)^{-1}$

$$\begin{aligned} U &= LVL^\mathsf{T} \\ &= (A^\mathsf{T} V^{-1} A)^{-1} A^\mathsf{T} V^{-1} V V^{-1} A (A^\mathsf{T} V^{-1} A)^{-1} \\ &= (A^\mathsf{T} V^{-1} A)^{-1} \end{aligned}$$

Non-linear least squares

[Non-linear least squares, Levenberg–Marquardt algorithm, Quasi-Newton method, BFGS method]

Use numerical minimization programs like MINUIT if the model is not linear in the parameters.

MINUIT's MIGRAD algorithm relies on gradients, it is based on the Davidon–Fletcher–Powell algorithm, a quasi-Newton method

Often used: Levenberg–Marquardt algorithm (see e.g. scipy.optimize.least_squares)

Choice of initial values of the fit parameters important to converge to the correct solution.

Often a numerical minimization program is also used in the linear case for convenience.

iminuit

iminuit is a Jupyter-friendly Python frontend to the *MINUIT2* C++ library.

https://iminuit.readthedocs.io/en/stable/

"Minuit2 has good performance compared to other minimisers, and it is one of the few codes out there which compute error estimates for your parameters."

Example: Straight line fit: $y = \theta_0 + \theta_1 \cdot x$ (1)

The conditions $d\chi^2/d\theta_0$ and $d\chi^2/d\theta_1$ give two linear equations with two variables which is easy to solve.

Here we use the general solution for linear least squares fits:

$$L = (A^T V^{-1} A)^{-1} A^T V^{-1} \qquad \widehat{\vec{\theta}} = L\vec{y}$$

$$A^T = \begin{pmatrix} 1 & 1 & \dots & 1 \\ x_1 & x_2 & \dots & x_n \end{pmatrix} \qquad \vec{\theta} = \begin{pmatrix} \theta_0 \\ \theta_1 \end{pmatrix} \quad V^{-1} = \begin{pmatrix} 1/\sigma_1^2 & & & \\ & 1/\sigma_2^2 & & \\ & & \ddots & \\ & & & 1/\sigma_n^2 \end{pmatrix}$$

$$A^T V^{-1} = \begin{pmatrix} 1/\sigma_1^2 & 1/\sigma_2^2 & \dots & 1/\sigma_n^2 \\ x_1/\sigma_1^2 & x_2/\sigma_2^2 & \dots & x_n/\sigma_n^2 \end{pmatrix}$$

$$A^T V^{-1} A = \begin{pmatrix} 1/\sigma_1^2 & 1/\sigma_2^2 & \dots & 1/\sigma_n^2 \\ x_1/\sigma_1^2 & x_2/\sigma_2^2 & \dots & x_n/\sigma_n^2 \end{pmatrix} \cdot \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix} = \begin{pmatrix} \sum_i \frac{1}{\sigma_i^2} & \sum_i \frac{x_i}{\sigma_i^2} \\ \sum_i \frac{x_i}{\sigma_i^2} & \sum_i \frac{x_i^2}{\sigma_i^2} \end{pmatrix}$$

Example: Straight line fit: $y = \theta_0 + \theta_1 \cdot x$ (2)

The 2 × 2 matrix is easy to invert:

shorthand notation for the sum

$$(A^\mathsf{T} V^{-1} A)^{-1} = \frac{1}{[1][x^2] - [x][x]} \begin{pmatrix} [x^2] & -[x] \\ -[x] & [1] \end{pmatrix} \qquad \text{where} \quad [z] := \sum_i \frac{z_i}{\sigma_i^2}$$

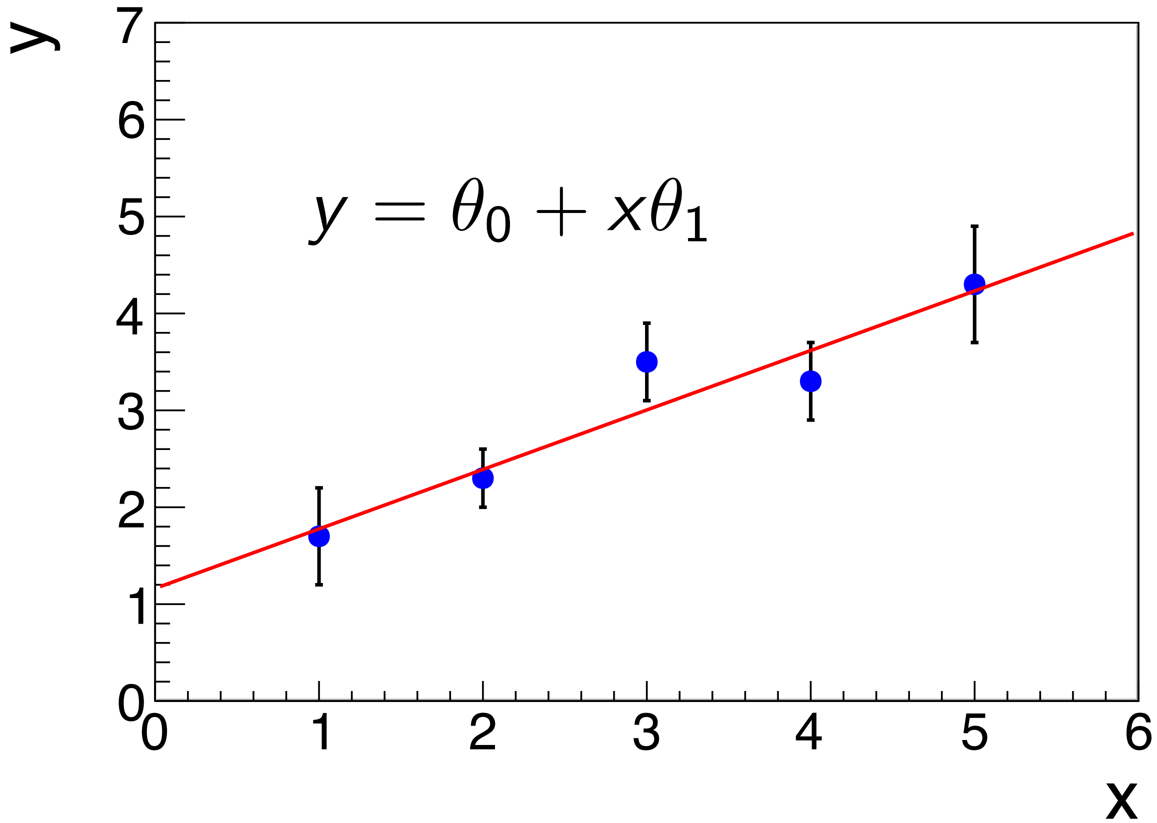
This gives:

$$\begin{aligned} L &= (A^\mathsf{T} V^{-1} A)^{-1} A^\mathsf{T} V^{-1} \\ &= \frac{1}{[1][x^2] - [x][x]} \begin{pmatrix} [x^2] & -[x] \\ -[x] & [1] \end{pmatrix} \cdot \begin{pmatrix} 1/\sigma_1^2 & 1/\sigma_2^2 & \dots & 1/\sigma_n^2 \\ x_1/\sigma_1^2 & x_2/\sigma_2^2 & \dots & x_n/\sigma_n^2 \end{pmatrix} \\ &= \frac{1}{[1][x^2] - [x][x]} \begin{pmatrix} [x^2]\frac{1}{\sigma_1^2} - [x]\frac{x_1}{\sigma_1^2} & \dots & [x^2]\frac{1}{\sigma_n^2} - [x]\frac{x_n}{\sigma_n^2} \\ -[x]\frac{1}{\sigma_1^2} + [1]\frac{x_1}{\sigma_1^2} & \dots & -[x]\frac{1}{\sigma_n^2} + [1]\frac{x_n}{\sigma_n^2} \end{pmatrix} \end{aligned}$$

We finally obtain:

$$\boxed{\hat{\theta}_0 = \frac{[x^2][y] - [x][xy]}{[1][x^2] - [x][x]} \qquad \hat{\theta}_1 = \frac{-[x][y] + [1][xy]}{[1][x^2] - [x][x]}} \qquad [xy] := \sum_i \frac{x_i y_i}{\sigma_i^2}$$

Example: Straight line fit: $y = \theta_0 + \theta_1 \cdot x$ (3)



x	y	σ_y
1	1.7	0.5
2	2.3	0.3
3	3.5	0.4
4	3.3	0.4
5	4.3	0.6

Fit result:

$$[z] := \sum_i \frac{z}{\sigma_i^2}$$

$$\hat{\theta}_0 = \frac{[x^2][y] - [x][xy]}{[1][x^2] - [x][x]} = 1.16207$$

$$\hat{\theta}_1 = \frac{-[x][y] + [1][xy]}{[1][x^2] - [x][x]} = 0.613945$$

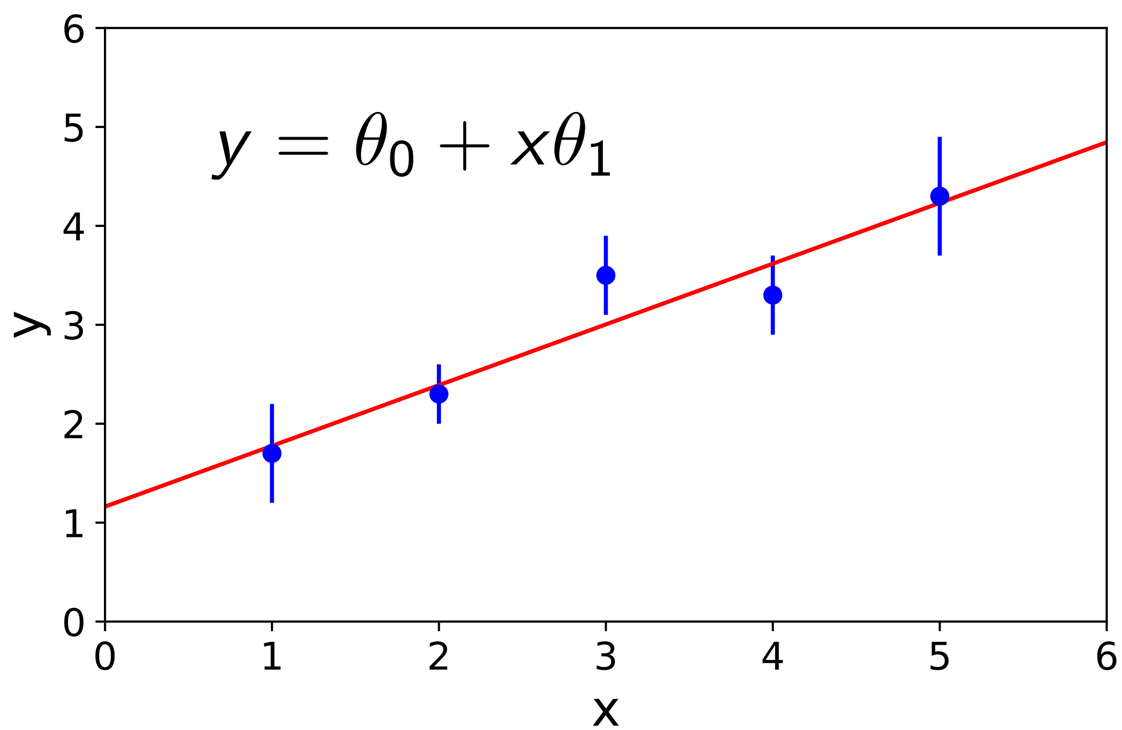
Covariance matrix of (θ_0, θ_1):

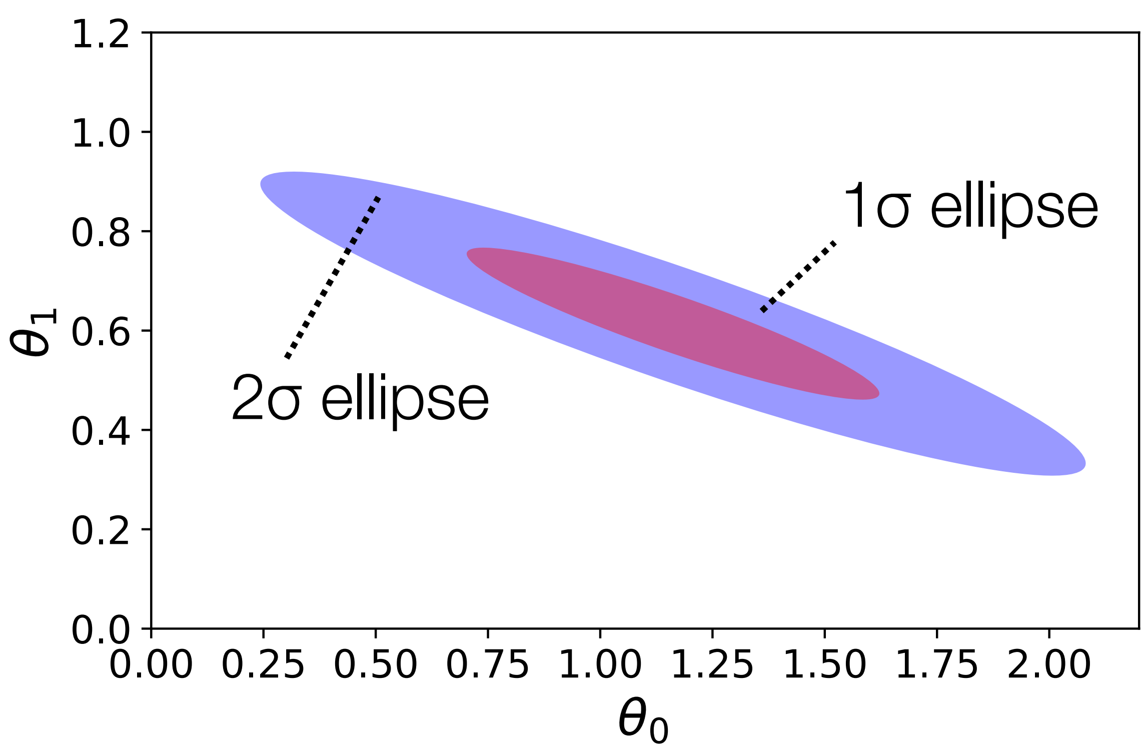
$$U = (A^\mathrm{T} V^{-1} A)^{-1}$$

$$= \begin{pmatrix} 0.211186 & -0.0646035 \\ -0.0646035 & 0.0234105 \end{pmatrix}$$

Straight line fit: Comparison to iminuit

[basic_chi2_fit_iminuit.ipynb]





FCN = 2.296				Ncalls = 30 (30 total)
EDM = 3.9e-23 (Goal: 0.0002)				up = 1.0
Valid Min.	Valid Param.	Above EDM		Reached call limit
True	True	False		False
Hesse failed	Has cov.	Accurate	Pos. def.	Forced
False	True	True	True	False

	Name	Value	Hesse Error	Minos Error-	Minos Error+
0	theta0	1.2	0.5		
1	theta1	0.61	0.15		

```
for p in m.parameters:
    print(f"{p} = {m.values[p]:.6f}" \
          f" +/- {m.errors[p]:.6f}")
```

```
theta0 = 1.162066 +/- 0.459550
theta1 = 0.613945 +/- 0.153005
```

```
# covariance matrix
print(m.np_covariance())
```

```
[[ 0.21118628 -0.06460344]
 [-0.06460344  0.02341046]]
```

Propagation of fit parameter uncertainties

$$y = \hat{\theta}_0 + \hat{\theta}_1 x \qquad \vec{J} = \begin{pmatrix} \frac{\partial y}{\partial \hat{\theta}_0} \\ \frac{\partial y}{\partial \hat{\theta}_1} \end{pmatrix} = \begin{pmatrix} 1 \\ x \end{pmatrix}$$

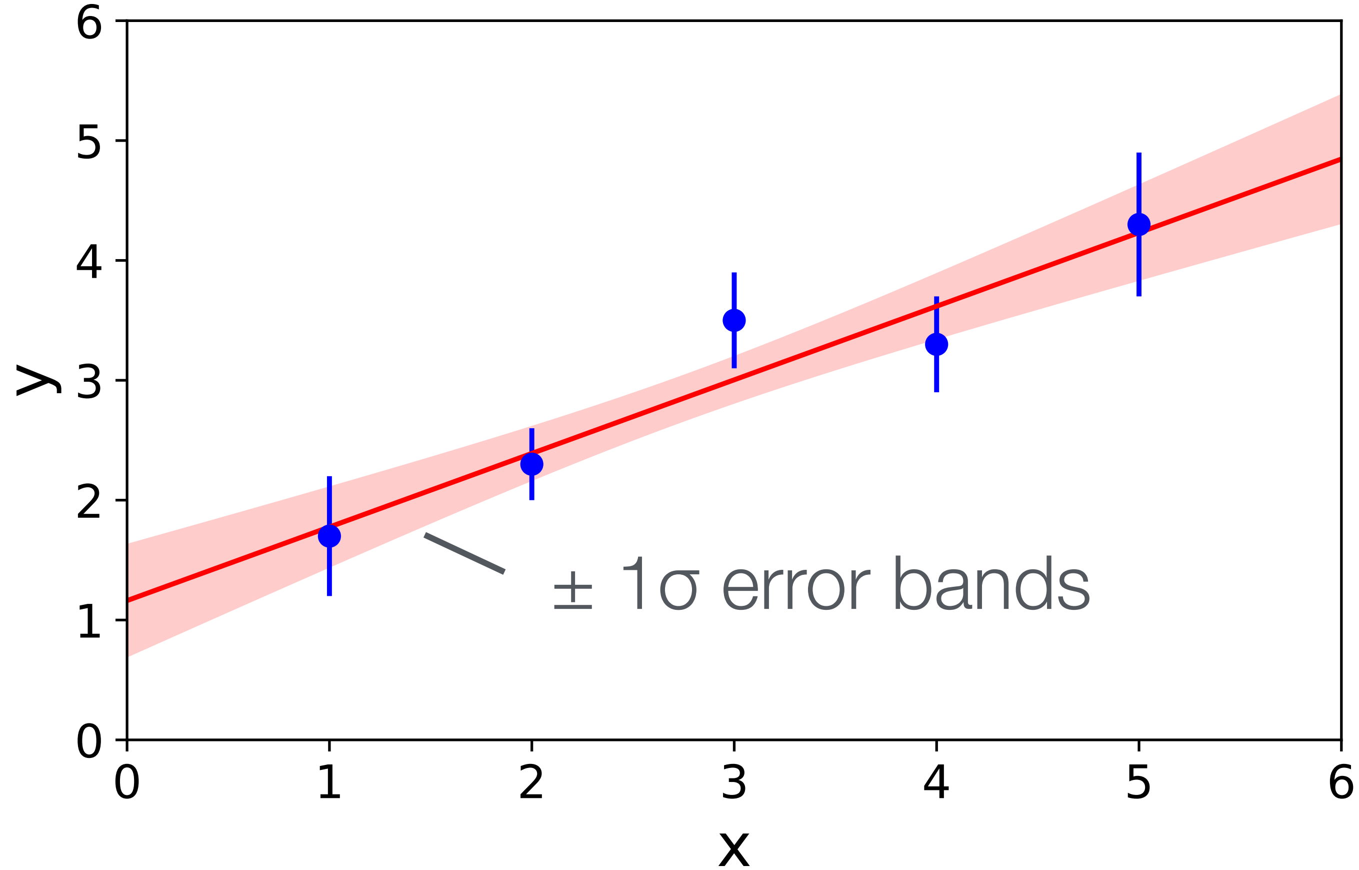
$$\sigma_y^2 = \vec{J}^\top U \vec{J} = \begin{pmatrix} 1 & x \end{pmatrix} \begin{pmatrix} \sigma_0^2 & \mathrm{cov}[\hat{\theta}_0, \hat{\theta}_1] \\ \mathrm{cov}[\hat{\theta}_0, \hat{\theta}_1] & \sigma_1^2 \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix}$$

$$= \begin{pmatrix} 1 & x \end{pmatrix} \begin{pmatrix} \sigma_0^2 + x\,\mathrm{cov}[\hat{\theta}_0, \hat{\theta}_1] \\ \mathrm{cov}[\hat{\theta}_0, \hat{\theta}_1] + x\sigma_1^2 \end{pmatrix}$$

$$= \sigma_1^2 x^2 + 2\mathrm{cov}[\hat{\theta}_0, \hat{\theta}_1] x + \sigma_0^2$$

Note:
correlation vanishes if you choose

$$y = \theta_0 + \theta_1(x - \langle x \rangle)$$



Least-squares fits to histograms

Consider histogram with k bins and n_i counts in bin i. If n_i is not too small one can use the Gaussian approximation of the Poisson distribution and apply the least-squares method:

Pearson's χ^2:
$$\chi^2(\vec{\theta}) = \sum_{i=1}^{k} \frac{(n_i - \nu_i(\vec{\theta}))^2}{\nu_i(\vec{\theta})}$$

Neyman's χ^2:
$$\chi^2(\vec{\theta}) = \sum_{i=1}^{k} \frac{(n_i - \nu_i(\vec{\theta}))^2}{n_i}$$

Problems arise in bins with few entries (typically less than 5), in particular in Neyman's χ^2.

Bins with zero entries are problematic, typically omitted from the fit
→ leads to biased fit results

Summary: Maximum Likelihood and χ^2 Method

Maximum likelihood method:

$$L(\vec{\theta}) = \prod_{i=1}^{n} f(x_i; \vec{\theta}) \qquad \frac{\partial \ln L}{\partial \theta_i} = 0, \quad i = 1, \ldots, m \quad \rightsquigarrow \quad \widehat{\vec{\theta}}$$

$$U[\widehat{\vec{\theta}}] = -H^{-1}, \; h_{ij} = \left. \frac{\partial^2 \ln L}{\partial \theta_i \partial \theta_j} \right|_{\widehat{\vec{\theta}}}, \quad H = (h_{ij}), \quad U = (u_{ij}), \quad u_{ij} = \mathrm{cov}[\hat{\theta}_i, \hat{\theta}_j]$$

covariance matrix of the estimated parameters θ_i

Least-squares method:

No correlations btw. the y_i:

$$\chi^2(\vec{\theta}) = -2 \ln L(\vec{\theta}) + \text{constant} = \sum_{i=1}^{n} \frac{(y_i - \mu(x_i; \vec{\theta}))^2}{\sigma_i^2}$$

With correlations btw. the y_i:

$$\chi^2(\vec{\theta}) = (\vec{y} - \vec{\mu}(\theta))^{\mathsf{T}} V^{-1} (\vec{y} - \vec{\mu}(\theta)), \quad V = (v_{ij}), \quad v_{ij} = \mathrm{cov}[y_i, y_j]$$

covariance matrix of the θ_i

$$\frac{\partial \chi^2}{\partial \theta_i} = 0, \quad i = 1, \ldots, m \quad \rightsquigarrow \quad \widehat{\vec{\theta}} \qquad U[\widehat{\vec{\theta}}] = 2H^{-1}, \; h_{ij} = \left. \frac{\partial^2 \chi^2}{\partial \theta_i \partial \theta_j} \right|_{\widehat{\vec{\theta}}}$$

Discussion of fit methods

[Wouter Verkerke, link]

Unbinned maximum likelihood fit (the best)

+ Don't need to bin data (no loss of information)
+ Works with multi-dimensional data
+ No Gaussian assumption
– No direct goodness of fit estimate
– Can be computationally expensive for large *n*
– Can't plot directly with data

Least-squares fit (the easiest)

+ fast, robust, easy
+ goodness of fit
+ can plot with data
+ works fine at high statistics
– data should be Gaussian
– misses information with feature size < bin size

Binned maximum likelihood fit in between