# Data Reduction Notes

Tobias Marriage

Feb 2011

## 1 Probability Distributions

Generically, a probability distribution corresponds to a function $p(x)$ which defines the likelihood for random variable $x$. **The likelihood that for which the probability of the random variable falling in a small interval $dx$ centered on the value $x_0$ is $p(x_0)dx$.** The distribution describes the *probability density* which, when integrated, yields the probability of $x$ falling between any two values:

$$P(x \in [x_0, x_1]) = \int_{x_0}^{x_1} p(x)dx. \tag{1}$$

Furthermore the distribution is normalized such that

$$P(x \in [-\infty, \infty]) = \int_{-\infty}^{\infty} p(x)dx = 1. \tag{2}$$

The **mean** $\mu$ of the distribution is defined as its first moment $\langle x \rangle$,

$$\mu \equiv \int_{-\infty}^{\infty} xp(x)dx, \tag{3}$$

and the **variance** $\sigma^2$ of the distribution is defined as its second moment $\langle x^2 \rangle$,

$$\sigma^2 \equiv \int_{-\infty}^{\infty} (x-\mu)^2 p(x)dx. \tag{4}$$

The probability distributions of course have third moments (skew, $\langle x^3 \rangle$), and fourth moments (kurtosis, $\langle x^4 \rangle$) which are useful but do not enter the everyday life of the experimenter. The mean describes the center of the distribution while the variance describes the width of the distribution. Another important quantity is the **standard deviation** $\sigma$ which is the square root of the variance.

### 1.1 Gaussian Distribution

The Gaussian (or normal) distribution is the most used distribution in experimental science. The Gaussian in fully described by a mean $\mu$ and a variance $\sigma^2$. The functional form of the Gaussian distribution is

$$p(x; \mu, \sigma) = \frac{1}{\sigma\sqrt{(2\pi)}} \exp\left[-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right]. \tag{5}$$

### 1.1.1 Standard Gaussian Distribution

By redefining variables $z = (x-\mu)/\sigma$ in Eq 5, we obtain the Standard Gaussian Distribution ($\mu = 1$, $\sigma = 1$):

$$p(z) = \frac{1}{\sigma\sqrt{(2\pi)}} \exp\left(-\frac{1}{2}\frac{z^2}{\sigma^2}\right). \tag{6}$$

The ability to define a Standard Gaussian by a simple linear transformation of the random variable has an significant implication for simulations (see section 6).

# 2 Errors

## 2.1 Errors on Raw Data

### 2.1.1 Counts

In some experiments, the data are counts. For instance in the classical Rutherford experiment, alpha particles were counted. When studying nuclear processes, product gamma rays may be counted. When studying galaxy populations, you may bin (count) galaxies in brightnesses intervals. In these cases, the data follow Poisson statistics: for a given count N, $\sigma_N^2 = N$.

# 3 Linear Modeling

Let $\vec{d}$ be an $n$-long vector of data that we model with a linear model $M\vec{p}$, where *vecp* is an $m$-long vector of parameters and $M$ is an $n \times m$ projection matrix which oper parameters $p$ into the data space. Furthermore, let the noise covariance matrix $N$ be an $n \times n$ diagonal matrix with the data's variance $\vec{\sigma^2}$ on the diagonal:

$$N = \begin{bmatrix} \sigma_1^2 & & 0 \\ & \ddots & \\ 0 & & \sigma_n^2 \end{bmatrix}. \tag{7}$$

The goodness of fit for this model is evaluated using the chi-squared statistic:

$$\chi^2(\vec{p}) = (\vec{d} - M\vec{p})^T N^{-1} (\vec{d} - M\vec{p}) = \sum_i \frac{(d_i - [M\vec{p}]_i)^2}{\sigma_i^2} \tag{8}$$

The quantity is $\vec{d} - M\vec{p}$ is called the "residual". The chi-squared statistic is the square of this residual over the variance. As the sum in Eq 8 suggests, for properly estimated errors $\sigma_i^2$, a good fit minimizes chi-squared and has a value close to $n$, the so-called number of "degrees of freedom".

In order to minimize the chi-squared with respect to the linear model parameters, we simply set the derivative of Eq 8 with respect to these parameters to zero:

$$\begin{aligned} \frac{d\chi}{d\vec{p}} &= 2M^T N^{-1}(\vec{d} - M\vec{p}) = 0 \\ \vec{p} &= (M^T N^{-1} M)^{-1} M^T N^{-1} \vec{d} \end{aligned} \tag{9}$$

# 4 Non-linear Modeling

To interpret a dataset with measurements $\{d_i\}$, we have a model which, given $n$ model parameters $\vec{p}$, gives corresponding values $\{m_i(\vec{p})\}$ . The chi-squared for this model is

$$\chi^2(\vec{p}) = \sum_i \frac{[d_i - m_i(\vec{p})]^2}{\sigma_i^2}, \tag{10}$$

where $\sigma_i^2$ is the estimate for the variance on $d_i$.

The errors on $d_i$ are assumed uncorrelated. For linear models, the chi-squared function was a quadratic in the parameters, so we can write

$$\chi^2(\vec{p}) = \chi^2(\vec{p_0}) + \frac{d\chi^2}{d\vec{p}}(\vec{p_0})(\vec{p} - \vec{p_0}) + \frac{1}{2}(\vec{p} - \vec{p_0})^T \frac{d^2\chi^2}{d^2\vec{p}}(\vec{p_0})(\vec{p} - \vec{p_0}), \tag{11}$$

where $\vec{p_0}$ is an arbitrary starting point in parameter space. The minimizing "jump" in parameter space, $\delta\vec{p} = (\vec{p_{min}} - \vec{p_0})$, is given by

$$\begin{aligned} 0 &= \frac{d\chi^2}{d\vec{p}}(\vec{p_0}) + \frac{d^2\chi^2}{d^2\vec{p}}(\vec{p_0})\delta\vec{p} \\ \delta\vec{p} &= -\left(\frac{d^2\chi^2}{d^2\vec{p}}(\vec{p_0})\right)^{-1} \frac{d\chi^2}{d\vec{p}}(\vec{p_0}) \end{aligned} \tag{12}$$

Note that in general the first derivative of chi-squared is a vector (the gradient), and the second derivative is a matrix (the hessian).

Many times, however, the chi-squared function is not purely quadratic. This is the case for non-linear models. In these cases, the above prescription for a purely quadratic function can fail. In these cases, we must follow our nose, potentially taking many steps around parameter space to find the minimum of the chi-squared. One easy prescription is to follow the gradient of chi-squared. The step in parameter space is then

$$\delta\vec{p} = -\vec{C}\frac{d\chi^2}{d\vec{p}}(\vec{p_0}), \tag{13}$$

where $\vec{C}$ is a constant vector of length $n$ (same as $\vec{p}$). In general, we can use both step formulae (Eq 12 and Eq 13) to explore the parameter space in search of the best fit values.

## 4.1 Gradient and Hessian

The next step towards an algorithm for non-linear model fitting is constructing the gradient vector and hessian matrix of the chi-squared function (Eq 10).The gradient is straight forward. The $j^{th}$ element of this vector is

$$\frac{\partial\chi^2}{\partial p_j}(\vec{p_0}) = -2\sum_i \frac{[d_i - m_i(\vec{p_0})]}{\sigma_i^2}\left(\frac{\partial m_i}{\partial p_j}(\vec{p_0})\right). \tag{14}$$

And the $(j,k)^{th}$ element of the hessian is

$$\begin{aligned}\frac{\partial^2\chi^2}{\partial p_j \partial p_k}(\vec{p_0}) &= -2\sum_i \sigma_i^{-2}\left(\frac{\partial m_i}{\partial p_j}(\vec{p_0})\right)\left(\frac{\partial m_i}{\partial p_k}(\vec{p_0})\right) + \frac{[d_i - m_i(\vec{p_0})]}{\sigma_i^2}\left(\frac{\partial^2 m_i}{\partial p_j \partial p_k}(\vec{p_0})\right) \\ &\approx -2\sum_i \sigma_i^{-2}\left(\frac{\partial m_i}{\partial p_j}(\vec{p_0})\right)\left(\frac{\partial m_i}{\partial p_k}(\vec{p_0})\right). \end{aligned} \tag{15}$$

The second term in the sum on the first line tends to cancel out for a good model near the chi-squared minimum: the quantity $d_i - m_i(\vec{p}_0)$ will be randomly positive and negative across $d_i$ such that its sum over $i$ should be small (Why isn't this the case for Eq 14? My hypothesis is that both of these terms are small near the minimum and it is the first term in the hessian which drives the final steps of the search.). The second term is also supposedly a numerical nuisance, so we drop it for the effective expression of the hessian.

## 4.2 Levenburg-Marquardt

The final step towards solving a non-linear least squares system is providing a prescription for stepping through parameter space and in particular choosing the constant in Eq 13. The Levenberg-Marquardt approach is the classic way to do this [1]. In this approach the constant for gradient decent is chosen to be proportional to the inverse of the second derivative with respect to the parameter in question. Following this prescription, Eq 12 and Eq 13 can be written

$$-\begin{bmatrix} \frac{\partial^2\chi^2}{\partial p_1 \partial p_1} & \cdots & \frac{\partial^2\chi^2}{\partial p_1 \partial p_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2\chi^2}{\partial p_n \partial p_1} & \cdots & \frac{\partial^2\chi^2}{\partial p_n \partial p_n} \end{bmatrix} \begin{bmatrix} \delta p_1 \\ \vdots \\ \delta p_n \end{bmatrix} = \begin{bmatrix} \frac{\partial\chi^2}{\partial p_1} \\ \vdots \\ \frac{\partial\chi^2}{\partial p_n} \end{bmatrix}, \tag{16}$$

$$\lambda\begin{bmatrix} \frac{\partial^2\chi^2}{\partial p_1 \partial p_1} & & 0 \\ & \ddots & \\ 0 & & \frac{\partial^2\chi^2}{\partial p_1 \partial p_1} \end{bmatrix} \begin{bmatrix} \delta p_1 \\ \vdots \\ \delta p_n \end{bmatrix} = \begin{bmatrix} \frac{\partial\chi^2}{\partial p_1} \\ \vdots \\ \frac{\partial\chi^2}{\partial p_n} \end{bmatrix}, \tag{17}$$

where $\lambda$ is a constant. Eq 16 and Eq 17 specify, respectively, the quadratic (Eq 12) and gradient (Eq 13) contributions to $\delta\vec{p}$. These equations can be rewritten as a single operation on the gradient

$$\begin{bmatrix} \delta p_1 \\ \vdots \\ \delta p_n \end{bmatrix} = \left(\begin{bmatrix} \frac{\partial^2\chi^2}{\partial p_1 \partial p_1}(-1+\lambda) & \cdots & \frac{\partial^2\chi^2}{\partial p_1 \partial p_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2\chi^2}{\partial p_n \partial p_1} & \cdots & \frac{\partial^2\chi^2}{\partial p_n \partial p_n}(-1+\lambda) \end{bmatrix}\right)^{-1} \begin{bmatrix} \frac{\partial\chi^2}{\partial p_1} \\ \vdots \\ \frac{\partial\chi^2}{\partial p_n} \end{bmatrix}, \tag{18}$$

Setting the matrix on the right hand side to $\boldsymbol{\alpha}(\vec{p},\lambda)$, the expression simplifies to

$$\delta\vec{p} = \boldsymbol{\alpha}(\vec{p},\lambda)^{-1} \bigtriangledown_p \chi^2. \tag{19}$$

As a prescription for choosing $\lambda$ and stepping around parameter space, here is a suggestion from the ever useful Numerical Recipes books [2]

1. Solve Eq 19 for $\delta\vec{p}$.
2. If $\chi^2(\vec{p}+\delta\vec{p}) > \chi^2(\vec{p})$, do *not* set $\vec{p} = \vec{p}+\delta\vec{p}$ (reject the step) and increase $\lambda$ by a factor of 10. Return to start.
3. If $\chi^2(\vec{p}+\delta\vec{p}) < \chi^2(\vec{p})$, set $\vec{p} = \vec{p}+\delta\vec{p}$ and decrease $\lambda$ by a factor of 10. Return to start.

You stop iterating this sequence when the quantity $\chi^2(\vec{p}+\delta\vec{p}) - \chi^2(\vec{p})$ is significantly less than one.

# 5 Error Propagation

Let the variance on a dataset $\{d_i\}$ ($n$ long) be given by $\{\sigma_i^2\}$. The variance of a function of the data $p(\{d_i\})$ is given by

$$\sigma_p^2 = \sum_i \left|\frac{\partial p}{\partial d_i}\right|^2 \sigma_i^2 \tag{20}$$

A corollary of this is that, for functions linear in $\{d_i\}$, the fractional variance ($\sigma_p^2/p^2$) of the function is the quadrature sum of the fractional variance of the data. This is corollary does not hold for non-linear functions.

Eq 20 can be written in vector form as

$$\sigma_p^2 = \vec{\nabla}_d p \cdot N \cdot \vec{\nabla}_d p, \tag{21}$$

where $N$ is given by Eq 7. More generally we can define the covariance matrix of a set of $m$ dependent variables (e.g., model parameters) as

$$C = \vec{\nabla}_d \vec{p} \cdot N \cdot \vec{\nabla}_d \vec{p} \tag{22}$$

where $\vec{\nabla}_d \vec{p}$ has dimensions of $(n \times m)$. Referring to Eq 9 for an expression for $\vec{p}$, the covariance matrix takes the form

$$C = (M^T N^{-1} M)^{-1}. \tag{23}$$

This expression can be directly related to the Hessian of the chi-squared in Eq 8:

$$\frac{d\chi^2}{d^2\vec{p}} = 2M^T N^{-1} M \equiv 2\alpha. \tag{24}$$

Thus half the Hessian (defined here as $\alpha$) is the inverse of the covariance matrix. For a non-linear model which is well approximated by linear terms near the minimum of the chi-squared function, the inverse of the $\alpha$ matrix is the formal covariance. This quantity can be calculated at the end of a non-linear model fit in order to give errors – it is often returned by algorithms based on the Levenburg-Marquardt chi-squared minimization.

Generally near the minimum, the chi-squared function can approximated by a second order function:

$$\Delta\chi^2 = \delta\vec{p} \cdot \alpha \cdot \delta\vec{p} \tag{25}$$

Table 1: Chi-squared values corresponding to confidence levels (CL) for various numbers of degrees of freedom.

| C.L. | 1 dof | 2 dof | 3 dof |
|---|---|---|---|
| 68% | 1.00 | 2.30 | 3.53 |
| 95.4% | 4.00 | 6.17 | 8.02 |

where we have used the definition of the matrix $\alpha$ from Eq 24, $\Delta\chi^2 = \chi^2 - \chi^2_{min}$ and $\delta\vec{p} = \vec{p} - \vec{p}_{min}$. In order to find the uncertainty in a parameter $\delta p_1$, we allow this parameter to take an arbitrary value and minimize over the other parameters. This latter condition implies that the gradient of $\chi^2$ will be zero along directions besides that corresponding to $\delta p_1$.

$$\frac{d\Delta\chi^2}{d\vec{p}} = \alpha \cdot \delta\vec{p} = \begin{bmatrix} c \\ \vdots \\ 0 \end{bmatrix} \tag{26}$$

Furthermore, for a single degree of freedom ($\delta p_1$), the change in chi-square (Eq 25) corresponding to the 68% confidence interval is $\Delta\chi^2 = \delta\vec{p} \cdot \alpha \cdot \delta\vec{p} = 1$. This condition constrains the value of $c$ in Eq 26 such that

$$\delta\vec{p}(68\%) = C \begin{bmatrix} 1/\sqrt{C_{11}} \\ \vdots \\ 0 \end{bmatrix}, \tag{27}$$

where we have used the identity $C = \alpha^{-1}$. It then follows that the standard deviation of parameter $p_1$ is $\sqrt{C_{11}}$.

More generally, one can plot the confidence contours of multiple parameters. Given the constraints of the printed page, two parameter error ellipses are most frequently used and provide insight to degeneracy among parameters. For $\nu$ parameters, one constructs a $\nu \times \nu$ sub-covariance matrix by extracting the intersections of rows and columns associated with the parameters of interested. Then one plots chi-squared contours corresponding to the appropriate confidence intervals. See Table 1.

# 6 Monte Carlo Simulations

Monte Carlo simulations are simulations of experiments. Monte Carlo is a famous gambling venue, and this name must have been chosen because these simulations probe out the probabilities associated with experiments.

- Random (or pseudorandom) data are generated based on either predictions/models or statistical properties (i.e., errors) derived from existing samples.
- These simulated data are then reduced to derive model parameters.
- This prescription is repeated many times to explore the space of model parameters allowed by the input errors.

Monte Carlos are useful for planning an experiment. For a given experimental configuration (e.g., integration time, distribution of measurements, etc), we can use Monte Carlos to predict the resulting constraints on models. For instance, in the Muon Lifetime experiment one can use Monte Carlos to estimate how many decay events are needed to constrain the Muon lifetime to percent level. Or with the Rutherford scattering, one could use a Monte Carlo to decide how long to integrate at each angle to get acceptable errors on the differential crossection. In a similar way, Monte Carlos are also useful for propagating errors to models once the dataset is taken.

## 6.1 Uniform Random Deviates

It's impossible for computer programs to generate truly random sequences of numbers (random deviates) for a Monte Carlo. Instead, there are algorithms to generate **pseudorandom numbers** given an arbitrary seed. The seed can be provided from the user or pulled from the lower order digits of the computer clock. A standard method for creating a pseudorandom sequence of integers is the multiplicative congruential method for which the $n^{th}$ random integer in the sequence is

$$x_n = (ax_{n-1}) \bmod M. \tag{28}$$

Where $a$ and $M$ are carefully chosen integers and $x_0$ the seed. The mod$M$ operation is the modulo with respect to M. This generates at most an $M$ element sequence of numbers. *Numerical Recipes* suggests $a = 16807$ and $M = 2^{32} - 1$ [2]. These pseudorandom numbers can be normalized such that they fall in the unit interval simply by dividing by $M$.

$$x'_n = \frac{(ax_{n-1}) \bmod M}{M}. \tag{29}$$

The numbers then approximate (for large $M$) random variables drawn from a uniform distribution on the unit interval:

$$p(x) = \begin{cases} 1, & \text{if } x \in [0,1) \\ 0, & \text{if } x \notin [0,1) \end{cases}. \tag{30}$$

This distribution has mean $\mu = 0.5$ and variance $\sigma^2 = 1/12$ (see Eq 4). Our approximation is characterized by $\mu = 0.5$ and $\sigma^2 = 1/12 - 1/6M$.

## 6.2 Gaussian Random Deviates

Our experimental errors are generally approximated by the Gaussian distribution (Eq 5). We therefore would like an algorithm that generates random deviates from this distribution. In particular it would be nice to have an algorithm which generates deviates from a Standard Gaussian distribution (Eq 6). Deviates from the standard distribution can be scaled to any Gaussian distribution with by the transformation $z = \sigma x + \mu$.

We are helped here by the **central limit theorem** which states that the sum of $N$ random samples from any distribution tends to a Gaussian distribution for large $N$. Consider the uniform random deviates generated by the algorithm in Eq 29 and drawn from the

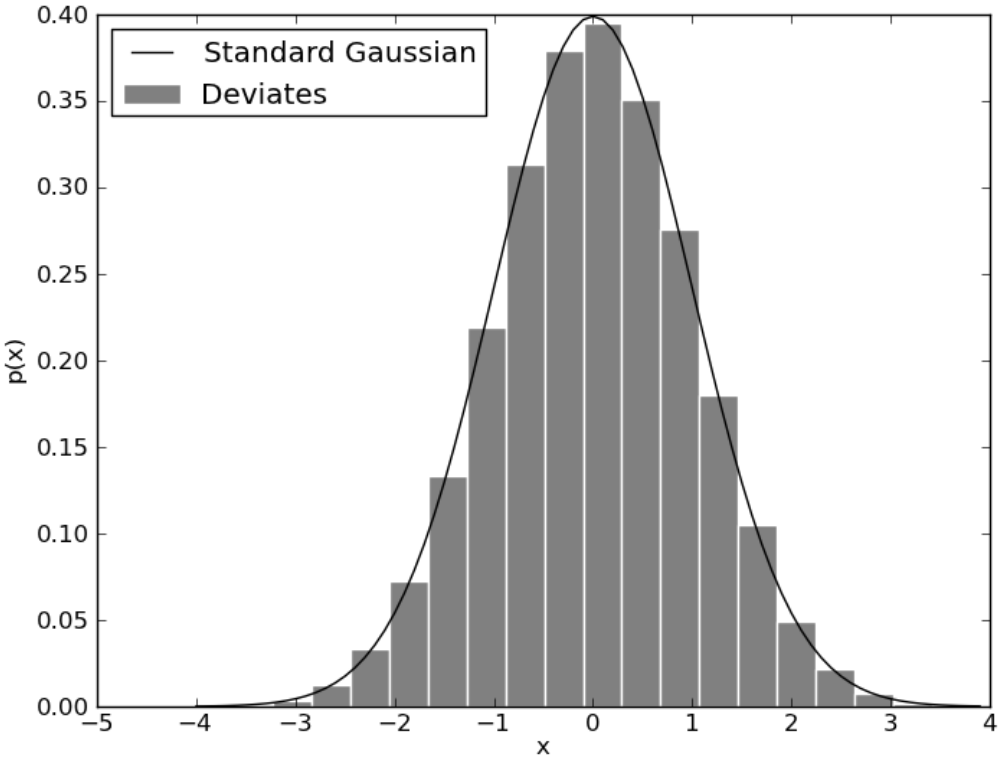


Figure 1: Gaussian Deviates. The normalized histogram derives from 100,000 random deviates derived as in Eq 31 with $N = 12$.

distribution 30. For $N$ uniform deviates thus defined, we can approximate a Gaussian deviate by

$$x_G = \sum_{i=1}^{N} x_i - N/2. \tag{31}$$

The distribution describing $x_G$ has mean $\mu = 0.$ and variance $\sigma^2 = N/12$ (recall the uniform distribution has $\sigma^2 = 1/12$). A convenient choice the approximate a Standard Gaussian distribution is then $N = 12$.

Of course there are packaged routines for computing deviates. For instance, the numerical python package has numpy.random.uniform() and numpy.random.normal().

# References

[1] Kenneth Levenberg. "A Method for the Solution of Certain Non-Linear Problems in Least Squares". The Quarterly of Applied Mathematics 2: 164-168 (1944).

[2] William Press et al. *Numerical Recipes in C*. Cambridge: Cambridge University Press, 1992.