Astrophysics with the Computer

# Bayesian Estimation of Parameters:

# Measuring the Radio Flux from the Moon

Joachim Köppen Strasbourg 2011

**The astrophysics background:**

The Moon, like every other body with some temperature, emits radio waves: the thermal motions of the electrons in its material cause the emission of a blackbody spectrum, which extends well into the radio range. At frequencies of 10 GHz, the present ordinary equipment for the reception of satellite television is sensitive enough to measure this signal from the Moon, and thus permit to determine the temperature on the lunar surface. Already in 1946 with the first radio telescopes it was discovered that the temperature averaged over the lunar disk varies with the lunar phase, and that the Moon is brightest in the radio range not at Full Moon, but about five days later. The reason is that the surface is not composed of solid rock, which heats up quickly due to the solar radiation, but instead of pebbles, dust, and small rocks – which is called regolith.

One way to observe the Moon with a radio telescope is to perform a drift scan: the telescope is pointed at a position where the Moon will be several minutes later, and thus one records the rise and fall of the signal as the Moon passes through the antenna beam. Because the sensitivity of the antenna drops on either side of the central direction, one obtains a bell-shaped curve when the power is plotted as a function of time. Due to the Earth's rotation a celestial source moves across the sky with a fixed and well-known speed (0.25°/min for a source at the celestial equator), thus one can relate time to angle in the sky. For a point source, the shape of this curve can be used to determine the antenna pattern. In particular the angular width at half maximum power (the Half-Power Beam-Width) is useful to characterize an antenna, as it gives its angular resolution. Common parabolic dishes have HPBW between 1° and 3°, appreciably wider than the Moon.

Along with the observation of a flux calibrator one measures the height of this curve above the background signal from the empty sky, and thus determines the observed radio flux from the Moon. However, thermal emission is a random process, and thus the measured signal fluctuates. But even more important is the noise produced in the radio receiver, as the lunar signal is not very strong. All this causes the individual measurements to fluctuate strongly about some average curve. Hence, we need to extract from a noisy curve the best estimate of the height of the underlying curve from the drift scan. By the way, this situation of having to dig out a weak signal from noise is common in science, because we all try to use the available equipment as best as possible. Once the easy strong sources are done, we always try use a new facility close to its limits to detect weaker sources!

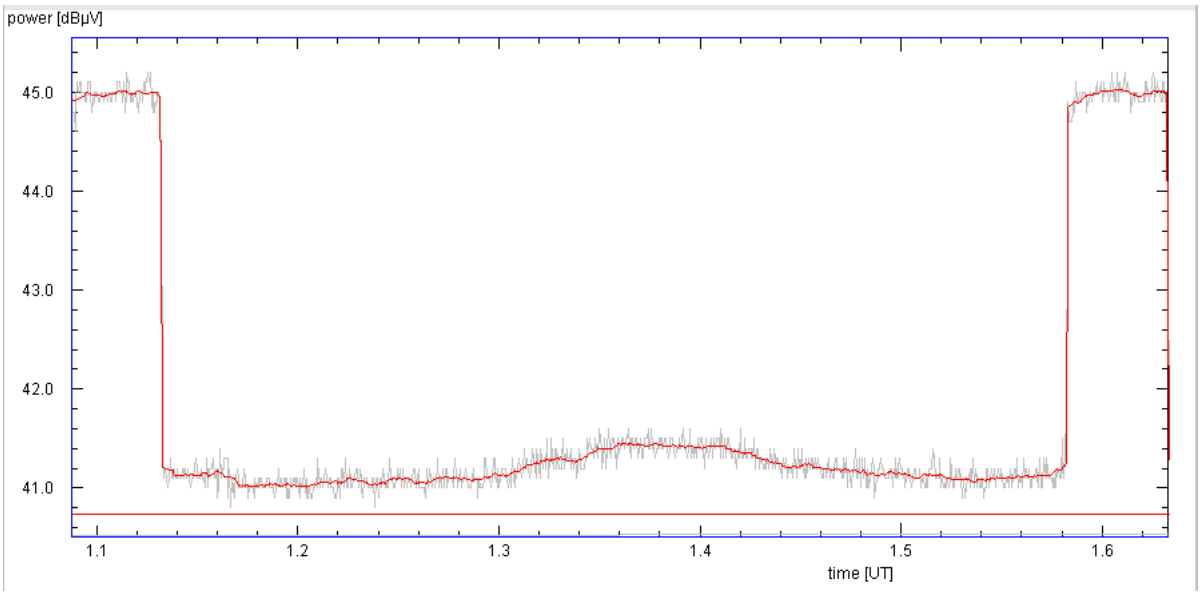


**Figure 1**: Data from a drift scan of the Moon observed with the ESA-Dresden radio telescope. The received power is shown as a function of time. Before and after the observation are the records of observations of the flux calibrator, the Holiday inn hotel building, which presents the emission of a blackbody of about 290 K.

One almost intuitive way to deal with the problem is to smooth the noisy curve, thus to average over several points. There is some arbitrariness, though: if we smooth too little, the result will still be subject to noise, but if we smooth too strongly, we wash out the structure. Here, this would be the fact that the signal rises and falls …

In this exercise we want to use a more clever approach, which tries to extract as much information as is possible from the measurements: we compare the data with a model whose known parameters are kept fixed, but for the unknown parameters we compute the probability distributions.

## The basic concept:

Let us derive the formulae from a basic consideration: consider one single measurement of some parameter *p* in the presence of some noise. If the true value is $p_0$ then the measured value is

$$p = p_0 + s$$

where *s* is the noise on the measurement. Suppose that this noise follows a Gaussian distribution with some amplitude σ. Then the probability to measure *p* given that the true value is $p_0$ is

$$P(p|\, p_0) = \frac{1}{\sigma\sqrt{\pi}} \exp\left(-\frac{(p-p_0)^2}{2\sigma^2}\right)$$

This quantity is also called the **likelihood**.

If we now consider the comparison of our noisy data points y(i) – the signal y at time instant i – with the corresponding predictions y0(i) from the model curve. We compute then the joint probability for that every datum y is from the true value y0. As the noise at each instant is independent of the noise at another instant, this joint likelihood is the product of the individual likelihoods

$$P(y \ldots |y_0 \ldots) = P(y(1)\,|y_0(1)) * P(y(2)\,|y_0(2)) * \ldots * P(y(n)\,|y_0(n))$$

or

$$P(y \ldots \mid y_0 \ldots) = \frac{1}{\sigma^n \sqrt{\pi}^n} \exp\left(-\frac{\sum (y(i) - y_0(i))^2}{2\sigma^2}\right)$$

where the sum over all squared deviations of the measurements from the prediction goes over all points from 1 to *n*. Note that this sum resembles the r.m.s. devation.

Now the model curve is determined by some parameters: for example if we assume a Gaussian curve, too (a good first approximation for the antenna pattern), it depends on the angular deviation *x* from the centre, but its shape is characterized by its width $\sigma_A$ and the height *h* of the maximum:

$$y_0(x;\ \sigma_A, h) = \frac{h}{\sigma_A \sqrt{\pi}} \exp\left(-\frac{x^2}{2{\sigma_A}^2}\right)$$

Thus, we can identify the above likelihood as the likelihood for the set of chosen model curve parameters:

$$P(y \ldots \mid \sigma_A, h, \sigma) = \frac{1}{\sigma^n \sqrt{\pi}^n} \exp\left(-\frac{\sum (y(i) - y_0(x(i);\ \sigma_A, h))^2}{2\sigma^2}\right)$$

Please note that the likelihood also depends on the assumed amplitude of the noise.

As the likelihood is a probabilistic measure of the distance of the observed data set from the assumed model predictions, its computation as a function of the model parameters gives us the probability for the model parameters. If we scan through the parameter space, and search the maximum of the likelihood, we get the best fit – in the sense that with the obtained parameters this particular model is best able to reproduce the observed data, and under the taken assumption (Gaussian noise, Gaussian shape of the model curve).

In a proper Bayesian approach one also needs to specify the apriori probabilities for the parameters to arrive at the probability for the model, given the matching of the data … For simplicity we shall assume uniform distributions for all parameters, as we know nothing about these parameters.

## The utility of this approach:

For Gaussian noise, the described approach is identical to the well-known chi-squared approach, which weeks to minimize the r.m.s. deviation. But viewing the problem in terms of probabilities offers one advantage: we can combine the results in order to ask different questions.

The direct outcome of the above formulae is the likelihood density for the three parameters

$$P(y \ldots \mid \sigma_A, h, \sigma)$$

but we have to search in a 3D space! If we have any knowledge of the antenna's HPBW, say, from other measurements, we do not need to vary this parameter and thus we are interested only in the 2D distribution

$$P(y \ldots \mid h, \sigma;\ \sigma_A)$$

Now the search is in two dimensions only, which is faster!

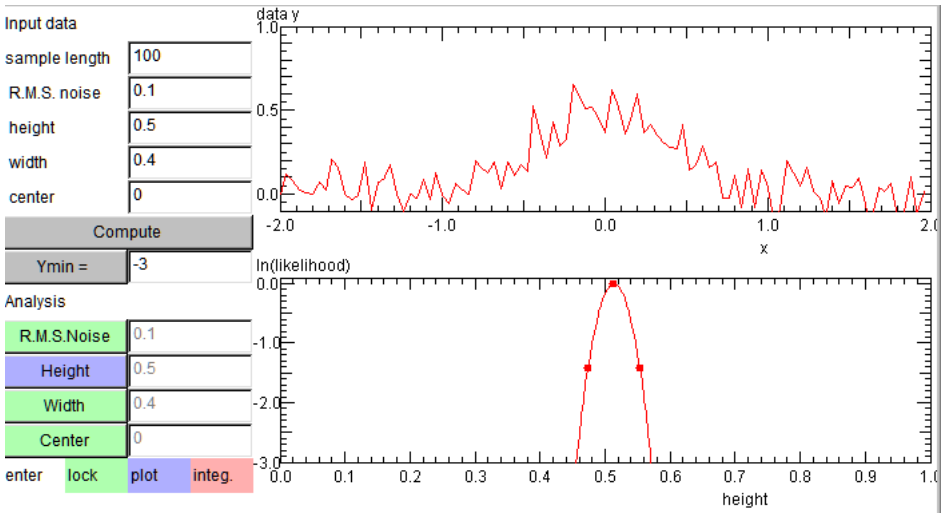


**Figure 2**: Screenshot of the simulation applet http://astro.u-strasbg.fr/~koppen/GaussFit/ . The top panel shows the simulated data width the parameters given on the left hand side. The bottom panel is the likelihood distribution for the parameter h, if all other parameters are known and equal to those of the original model. Note that every click on the Compute button or change of parameters will cause the generation of a fresh data set. Thus the likelihood distribution will also vary in position and shape!

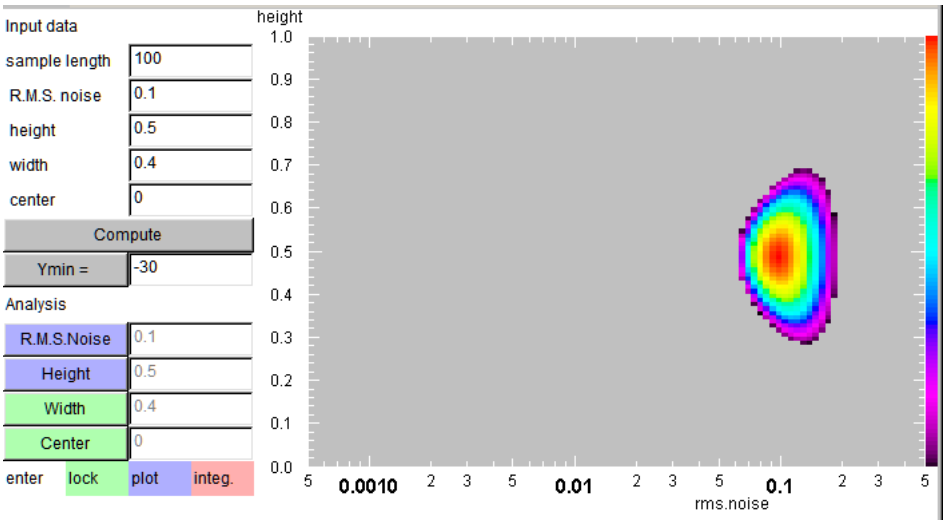


**Figure 3**: Screenshot of the simulation applet, showing the results of scanning two parameters, while keeping the others constant.

But as our principal interest is the height of the peak power during the drift scan, the knowledge of the amplitude of the noise is without relevance! So let us ask the question: what is the likelihood for the parameter h irrespective of the noise amplitude? In other words we look for the joint likelihood for any value of the noise level ... which is nothing but the sum over all individual likelihoods. As we consider here a continuous parameter, we need to integrate over its range of values

$$P(y\ldots|h;\ \sigma_A,\sigma) = \int P(y\ldots|h,\sigma;\ \sigma_A)\ d\sigma$$

This gives a curve with a single peak for the best value of h.

## Confidence regions:

Apart from the knowledge of the value of the best parameter it is also very useful to give a range of values with a certain value for its probability. For a one-dimensional distribution this is an interval which contains, say 90 percent of the probability to find the true parameter. Therefore we integrate the distribution over the entire range of values for the parameter, creating a table of

$$P(x) = \frac{\int_{pmin}^{x} P(\ldots|p')dp'}{\int_{pmin}^{pmax} P(\ldots|p')dp'}$$

the probability normalized to the entire range. From this table, one may find the median by searching for $P(x_{50}) = 0.5$, and the limits of the 90 percent confidence interval at $P(x_5) = 0.05$ and $P(x_{95}) = 0.95$. As the table gives only a finite number of values, it is useful to do a linear interpolation between neighbouring points to find the exact $x$-values.

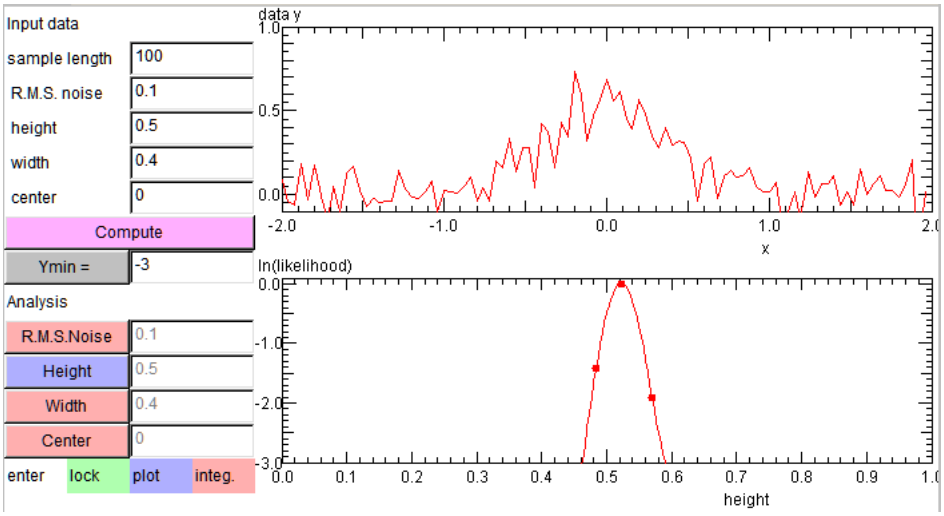


**Figure 4**: Screenshot of the simulation applet, showing the results of scanning in one parameter, but integrating over the other three. The three dots on the curve mark the median value and the borders of the 90% confidence interval.

In two dimensions one would do the analogue procedure, seeking the contour line which encloses 90 percent of the probability … but this is a bit tedious to realize with programs like `gnuplot`.

## More issues of realism:

To represent the real data, there are other problems: we do not know precisely the time of the transit of the Moon, i.e. the time for the peak power. Hence, we must add this as an additional free parameter. As this is not an essential one, we shall integrate the likelihoods over its range of values.

In the data from the ESA-Dresden radio telescope the recorded values of the power are not the linear power, but values given in decibel dB. Thus, we have to convert them into linear values by

$$P = 10^{P[dB]/10}$$

The reference value (for 1µV) is irrelevant here, since all values are given with the same reference.

A third aspect is the fact that the recorded powers are given only in 0.1 dB steps. This finite discretization step size has technical reasons, as it is the resolution of the analogue-to-digital converter in the receiver. We shall not deal with this here!

### Some practical aspects:

The choice of the parameter ranges is best done after some experimentation with real data: We do not want to make these ranges too large, because then most of the volume represents unsuitable models and poor fits. On the other hand, they should not be too small, as one could miss the best fit! Since we have quite a large number of data points, the likelihood distributions will be quite narrow, so a reasonable choice could be that we should resolve the maximum with say one tenth of the number of grid points in a parameter range. With present computers we can use 100 points for each parameter, and up to 4 parameters, and still get a good turn-around time.

The raw data is given in a text file, every line of which constitutes one measurement. The first number is the time in UTC in hh:mm:ss.sss format. This string needs to be converted in a sensible number. Since the duration of a drift scan is about 30 min, it appears to be reasonable to use minutes as units of time, and perhaps use only the time difference from the first instant. The next two numbers are the azimuth and elevation of the telescope, which are of no interest here; ignore them. The last number is the measured power, in dBµV. This means that it has to be converted into linear values, before we can use it.

Some observational data can be found at http://astro.u-strsbg.fr/~koppen/ue7e/archive.html#moon

### Another Example:

Linear regression is a straight-forward technique to find the best straight line to match some given data set. The formulae for the best parameter can be found in any relevant handbook. We may also give a formulation in terms of Bayesian estimation. Now the model is simply

$$y_0(x;\ m, b\ ) =\ mx + b$$

The parameters are the slope *m* and the offset *b*. The likelihood is given by

$$P(y\ldots|m, b, \sigma) = \frac{1}{\sigma^n\sqrt{\pi}^n}\exp(-\frac{\sum(y(i) - (mx(i) - b\ ))^2}{2\sigma^2})$$

The outcome should be the same as given by the conventional formulae. Thus, they present a nice test case with which we can test our program.

## How to test the program:

The verification is quite simple: Generate a data set composed of the values *y* of a linear function of the abscissae *x*, but add Gaussian noise to the ordinates. For this purpose, we need to generate random number distributed like a Gaussian function. One possibility is to search for such a generator on the internet; the other choice is to apply the transformation method (described in *Numerical Receipes*, available in the M2 library) to convert a uniform random number into some with any given distribution function.

## Orthogonal Regression:

Linear regression assumes that the noise is present in the ordinates only. But when you look for a correlation between two noisy data sets, you look for the straight line that minimizes the average distance of the scattered points to that line. In our Bayesian approach we simply reformulate the likelihood function. Since the perpendicular distance of a point (*x*,*y*) from a straight line y=mx+b is

$$d = \frac{|y - mx - b|}{\sqrt{1 + m^2}}$$

The likelihood has the slightly modified form:

$$P(y \ldots | m, b, \sigma) = \frac{1}{\sigma^n \sqrt{\pi^n}} \exp(-\frac{\sum (y(i) - (mx(i) - b))^2}{2\sigma^2 (1 + m^2)})$$

## Nonlinear Regression:

If we want to find the best fit, given any function *y=g(x; a, b, c)* with some parameters *a, b, c* that describe the curve's form, we insert this into the likelihood, assuming Gaussian noise on the ordinates:

$$P(y \ldots | a, b, c, \sigma) = \frac{1}{\sigma^n \sqrt{\pi^n}} \exp(-\frac{\sum (y(i) - g(x(i); a, b, c))^2}{2\sigma^2})$$

## Here are my suggestions that you will never follow:

- Before actually writing the program, do make a flow chart diagram in order to understand the sequence of what is computed; a diagram of the program structure and the data structure to find out, how the loops and iterations are nested, which data from earlier parts you need at each section, which kind of vectors and arrays you are going to need. This may seem bureaucratic, boring or even old fashioned, but **don't start typing anything, before you are absolutely clear about what you plan to do**. Otherwise you may really end up wasting much time in trying to find the logical errors, loopholes, and cul-de-sacs of your hasty programming. Save yourself the frustration, disappointment, and anger!
- **General Program Planning**: It is a good idea to lay out the program as general as possible. This makes it easier to include other effects, or to try out other situations. E.g. keep the number of levels as a parameter which is assigned a value in the main program, rather than being specified everywhere in the loop limits. Checking the program for other numbers of

levels can thus be done easily at any time. Or one can try out how many levels are really necessary for a particular model and accuracy.

- **Modular Construction**: It is also a good idea to break up the program into mathematically or physically sensible units. This allows a better testing of these individual modules - and most of the time is spent in tracing an error - a more flexible use of them for other purposes, and their exchange against improved or alternative methods, improved data, other physical processes, etc. For example, if the integration routine is contained in an independent unit, one simply exchanges this against a more sophisticated method, if need arises, but without the trouble of having to change the program at a dozen places. For testing, a simple main program has to be written which supplies the necessary input data to this unit.
- **Check Everything by Hand**: Often, we underestimate our ingenuity to make small logical mistakes or simple typing errors, which may cause faulty results. The worst mistakes are those which produce results that look as one would expect them to be. Be happy if probabilities are negative … then you know for sure that there is an error! Take the trouble of check everything the program does, until you are sure it does only what you want it to do. In programs about physical things, basic physics must be obeyed: conservation of particles, energy, etc. Also, all the simple and limiting cases which we do understand must be reproduced accurately.
- **Be Highly Skeptic of anything the program produces**
- **Be Careful with the indices**: (I,J) is easily confused with (J,I), so is I and 1. If one tries to find the origin of some error in the results, one may **never** notice in the program's listing that the two letters were interchanged. Thus, just merely writing (J,K) is preferable to (I,J).
- When you are making tests, and later running the program for various situations and parameters, try to keep a careful written record of what you do, noting input parameters and results. This will make it easier for you later to compare results with earlier ones, in case you have to hunt for an error that has crept in yesterday when you “just changed a few things, almost nothing --- but the program doesn't work any more".