6. MC simulation of experimental data

[Numerical Recipes 15.6, own derivation]

Monte Carlo simulations can be a highly useful tool in the analysis of experimental data. The general idea is that doing experiments can be quite time-consuming, expensive, or both, so often one is limited in the amount of data points one can collect, as well as the number of data collection runs one can do.

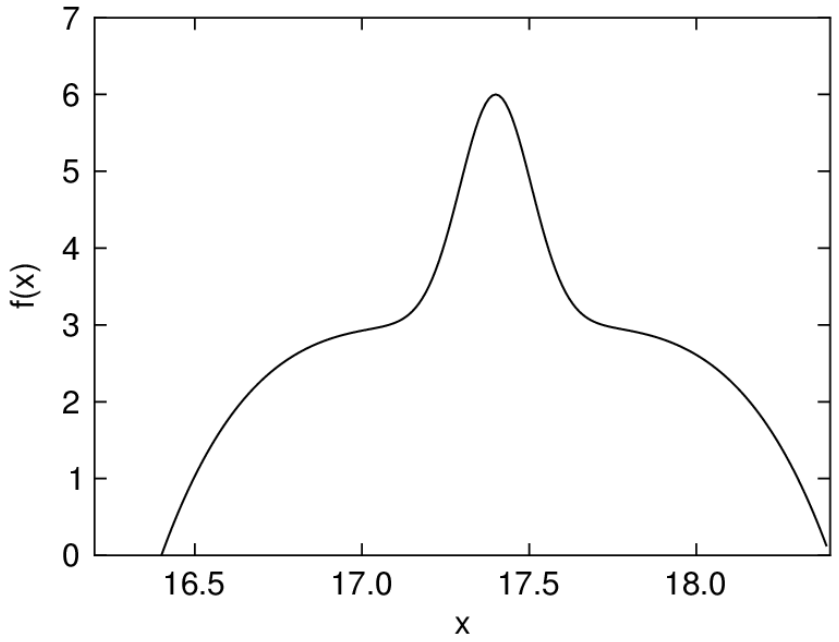
The way in which MC can be utilized for experiments depends strongly on the application. We will therefore first go through probably the simplest possible case, MC error analysis of a single parameter of a data set, then give an example of actual simulation of a physical process.

6.1. Error analysis of known distribution

Let us assume 3 things. **(A1)** we need to do a measurement of a distribution of known shape, but are limited in the number of statistics we can collect. **(A2)** there is a single relevant parameter we are interested in: the mean of the distribution. **(A3)** In measurements on different systems the mean will change position, but the shape and width of the distribution will always be the same.

In reality, one would in most cases probably need more than one parameter to characterize a distribution; for instance the distribution width often changes with the mean. But the basic idea of the MC error analysis will remain the same for many-dimensional parameter space, so we can limit ourselves to the one-parameter in this pedagogical explanation.

Say we know somehow (from previous measurements with very good statistics or theory) that the real distribution which would be obtained with infinite statistics looks as follows:



This actually is generated with `awk 'BEGIN for (x=13;x<=22;x+=0.01) f1=3*exp(-((x-17.4)/0.15)^ 2); f2=(3.0-3*(x-17.4)^ 4); f=f1+f2; if (f>0) print x,f,f1,f2; exit; ' > weirdo_distr` but knowledge of this is not necessary here.

We further assume there does exist a well-defined right answer, which could be obtained if we had infinetely good resolution and infinite statistics. The value of the right answer is called the **true value.** The true mean in the example data above is 17.4.

In the measurement, you will collect a finite number of counts for a number of bins in this distribution, and want to determine the mean and its error. Let us say you need to do a large

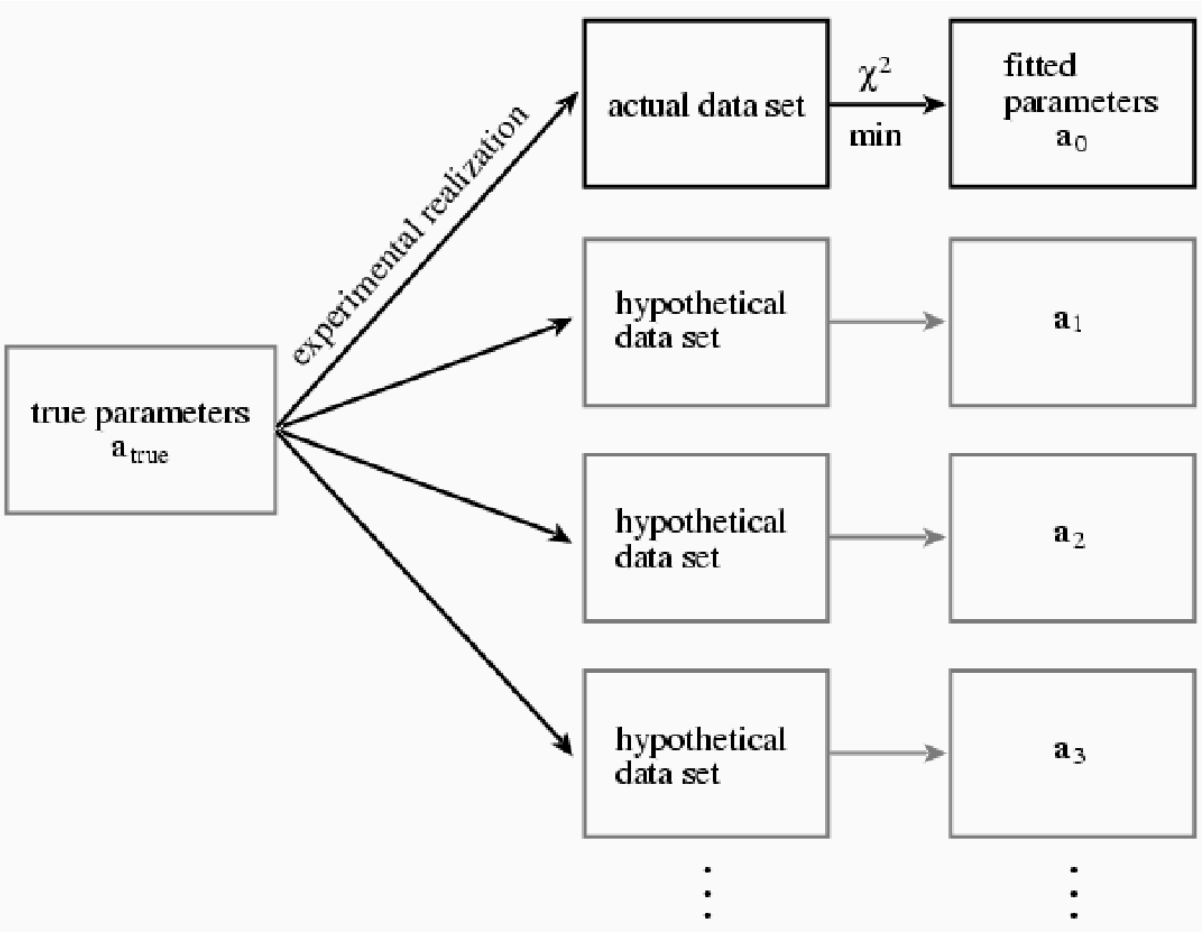
number of measurements, but can collect only 300 counts in each. This will lead to a fairly large error.

The distribution is so far from a Gaussian you do not have the nerve to simply calculate the error as one would for a Gaussian shape, and think you could get away with it.

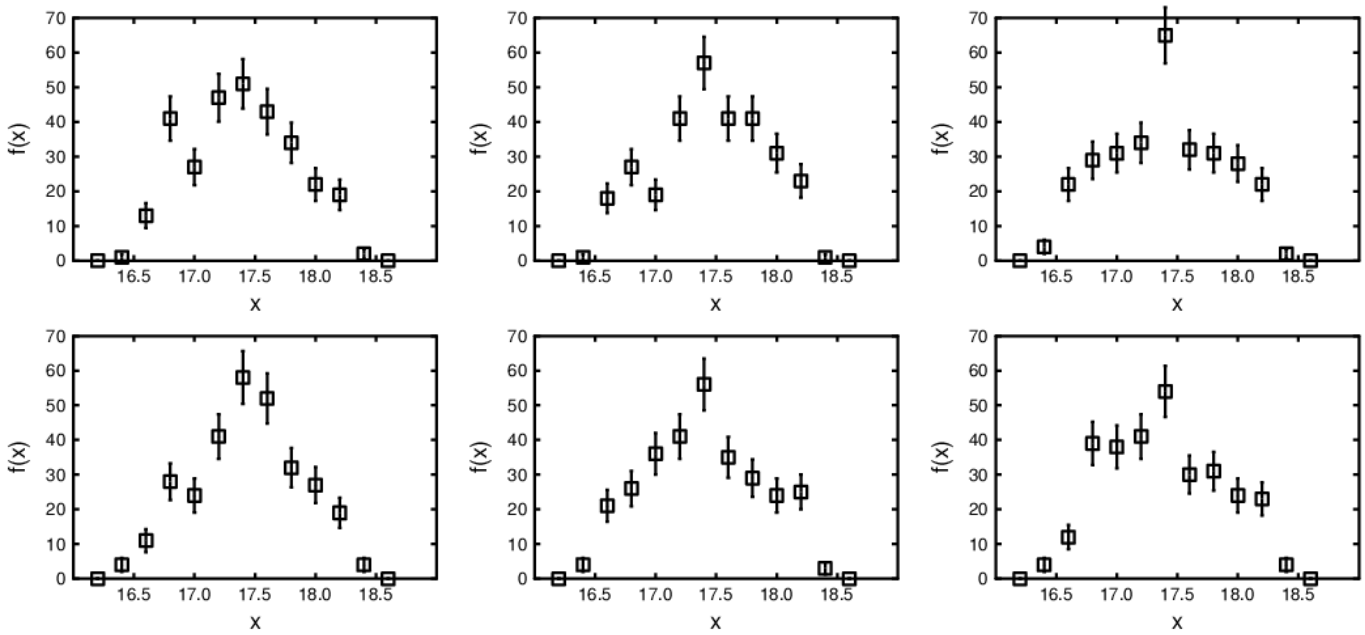
So how can we determine the error? We could try to analyze the distribution numerically or analytically. But for truly complicated and multidimensional distributions this is often not possible.

But one, quite straightforward approach is to use MC to simulate the counting experiment. We already know from the last section how to simulate random points collected in an arbitrary distribution (using the numerical-analytical approach or the hit-and-miss method). Now we can simply simulate 300 individual counts arriving randomly in the distribution $f(x)$ [1]. These artificial data sets are called **hypothetical data sets** or **synthetic data sets.**

[1] It is actually possible to do this much more efficiently by generating points with a Poisson distribution directly for each bin. But the approach in the text is simpler to understand and code and hence used here for educational purposes.



The result from 6 separate runs each generating 300 points looks as follows:



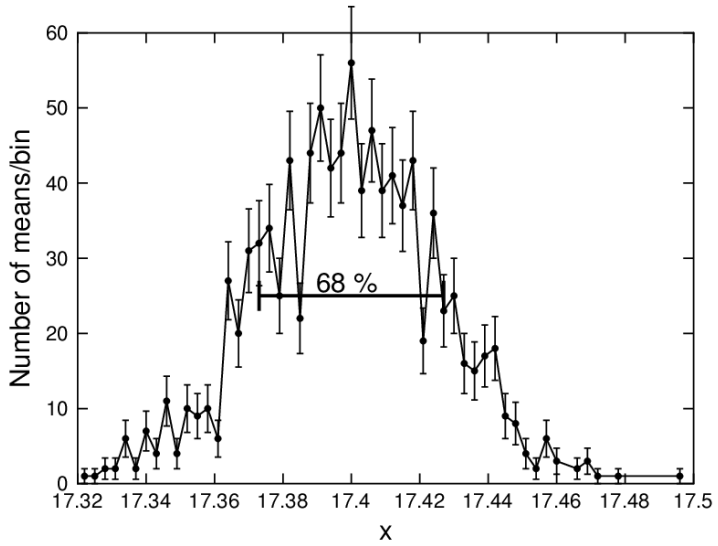
So the statistics is poor, and undoubtedly the error will be quite large.

But how does this help with determining the error? We have to return to the definition of a 1σ error. It is that 68.3 % of the measured values will be within 1σ of the true mean.

Now when we generate the data with MC we of course know the true mean, it is exactly 17.4 for this test case. Furthermore, we know (by our assumption A3) that the width and shape of the distribution is not affected by the exact location of the mean.

We can calculate the mean $\bar{x}_i$ for a a large number $i = 1 \ldots N$ of data sets generated with MC, then see what x range contains 68.3 % of the obtained means $\bar{x}_i$.

To do this, I repeated the generation of the 300-count data set 1000 times. This gave the list of means $\bar{x}_i, i = 1, \cdots, N$ with $N = 1000$. Then I generated a histogram **statistic of the means** $h_j(\bar{x}_i)$ by summing them in bins of width $\Delta\bar{x} = 0.003$, which gave the following distribution:

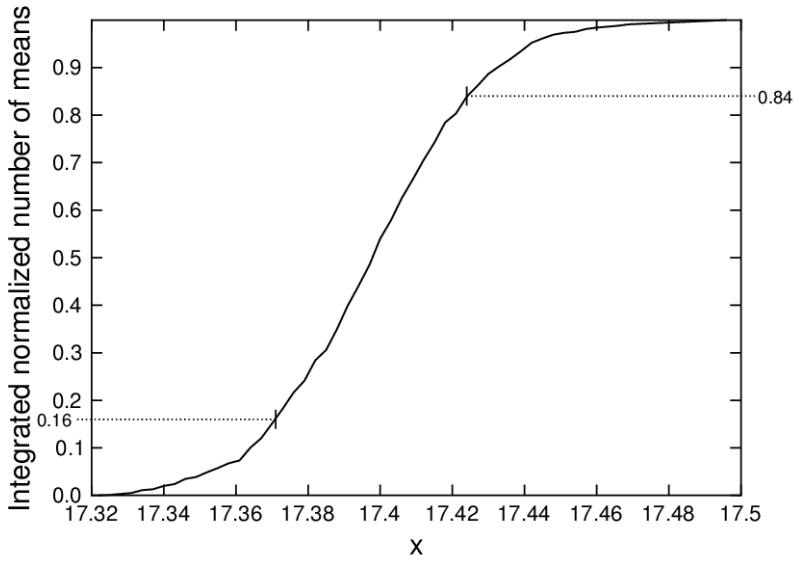


The distribution is centered around 17.4, as it should. To estimate the uncertainty, we want to

know in what x range 68.3% of the data points are located around the maximum. The easiest way to do this fairly accurately is to form the cumulative distribution function H of the means,

$$H_k = \sum_{j=1}^{k} h_j(\bar{x}_i)$$

To get the confidence interval, we simply need to find in what $\bar{x}$ region H_k is between 0.16 and 0.84, which can even be done visually accurately enough (an interpolation scheme would of course also be easy to devise):



So we find that 68 % of the means lie between 17.373 an 17.427. Hence the width of the confidence interval is 0.054, and the uncertainty of the data half of that, 0.027.

So with this straightforward MC method we have determined that the 1σ statistical uncertainty of our 300-count distributions is 0.027. Provided our starting assumption A3 (that the width and shape of the distribution does not change with the position of the mean) holds, we could then use this value of the uncertainty for all measurements in the series, giving the result as

$$\bar{x}_i \pm 0.027$$

Incidentally, calculating the error bar assuming Gaussian statistics gives errors of $0.026 - -0.028$ for the different distributions, so Gaussian error analysis happens to give almost the right answer here.

The discussion on this subsection can be summarized as the following algorithm (assuming we use the analytical- numerical approach to generate points distributed as f):

01 a° Read in known shape of distribution $f_i(x)$ with N_f points

01 b° Calculate "true mean" $\bar{x}_{\rm true}$ from $f_i(x)$

02 a° Select number of counts to generate N_c in one distribution

02 b° Select number of distributions to generate N_d

03° Form the cumulative distribution $F_k(x) = \sum_{i=1}^{N_f} f_i(x)$

04° Set $h_j = 0$ for all $j = 1, \ldots, N_h$, select $\Delta\bar{x}$

05° Do loop over distributions $i = 1, N_d$

  06° Set $D_k = 0$ for all $k = 1, \ldots, N_f$

  07° Do loop over counts $j = 1, N_c$

    08° Generate a uniformly distributed number $u = P_u(0, 1)$

    09° Find k such that $F_{k-1} < u \leq F_k$

    10° Generate one count: set $D_k = D_k + 1$

  11° End loop over counts $j = 1, N_c$

  12° Calculate weighted mean $\bar{x}$ of distribution D_k

  13° For position $j = \bar{x}/\Delta\bar{x}$ do $h_j = h_j + 1$

14° End loop over distributions $i = 1, N_d$

15° Form the cumulative distribution $H_k(x) = \sum_{i=1}^{k} h_j(x)$

16° Find the points $\bar{x}_l$ and $\bar{x}_u$ where $H_k(\bar{x}) = 0.16$ and $= 0.84$.

17° Report the lower bound uncertainty $\bar{x}_{\rm true} - \bar{x}_l$ and upper $\bar{x}_u - \bar{x}_{\rm true}$

In addition to the loops written out, at least steps 01 b° , 09° , 12° , 15° and 16° involve loops,

but since these are quite simple they are not written out. We have assumed for simplicity f_i and D_k have the same Δx, which they need not have – if N_c is low it is probably better to have D_k on a cruder grid than the read-in function.

Note that step 17° also allows for finding the non-symmetric uncertainty of a non-symmetric data distribution, i.e. an uncertainty of the form

$$12.9^{+1.4}_{-0.9}$$

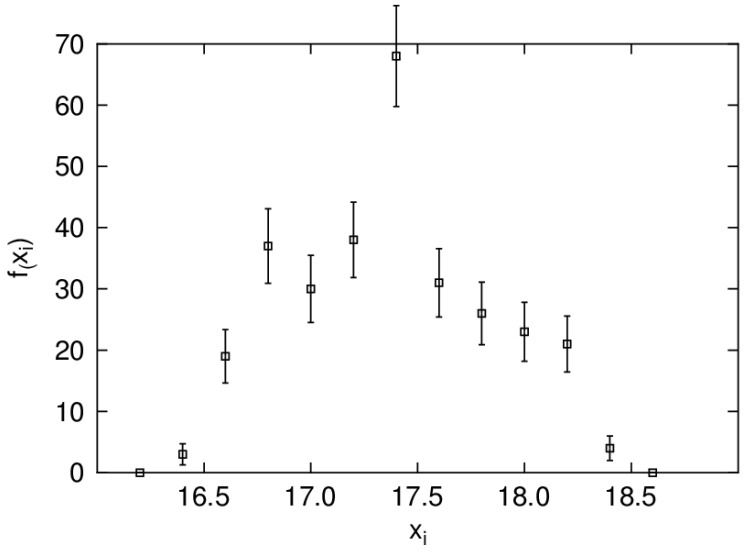
(the numbers here are just arbitrary examples).

Note that this example is almost certainly not the most efficient way of doing this. It is just (in the lecturer's subjective opinion) a simple and intuitively clear way, which is why it is presented here. But as long as we are dealing with only 1D distributions, the computational demands of all this is probably minimal, so there may not be any need to focus on making it more efficient.

6.2. Worst case: the bootstrap method

The above method has the obvious drawback that we need to know the shape of the distribution. Let's now look at the worst possible case: we do not know the shape of the distribution, we only have a single measurement giving a lineshape with poor statistics, and we can not get any more. We need to determine the mean of the data, and its uncertainty. And we do not understand the physical process producing the lineshape well enough that we could simulate the process producing the lineshape.

Let us say the data is in the form of points $f_i(x_i)$. It could look something like this:



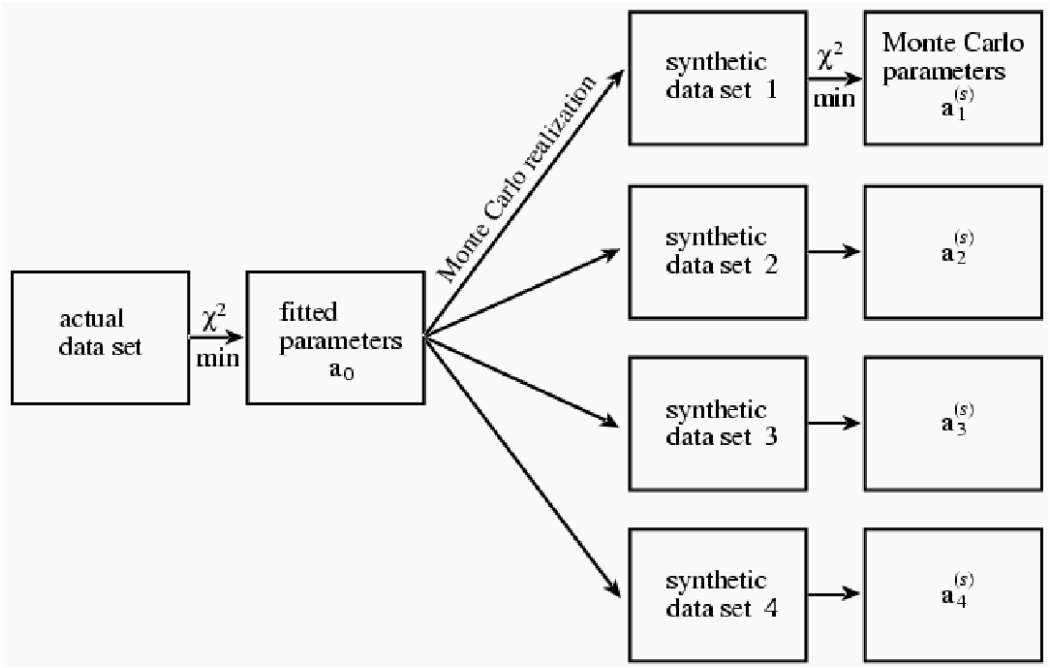
Determining the mean of the lineshape is of course no problem, this is just

$$\bar{x}_i = \frac{\sum_{i=1}^{N} f_i(x_i) x_i}{\sum_{i=1}^{N} f_i(x_i)}$$

provided the data is at even intervals in x_i. But what about the error? We could again use the standard equations to calculate the Gaussian statistics error, but is it possible to do anything else?

Offhand it would seem like the answer is no. But there is in fact an extremely dirty thing which can be carried out which actually seems to work quite well. This is repeating the process described

in the previous section, starting from the single data set itself! Doing this works exactly as in the method described above, the only difference is we start directly with the experimental distribution.



Since this method is pretty much like pulling oneself into the air from ones own bootstraps (a la Baron Münchausen), this is called the **bootstrap method.** According to Numerical Recipes, it has already been in use for some time, but has only recently started to become accepted by statisticians "when enough theorems have been proved".

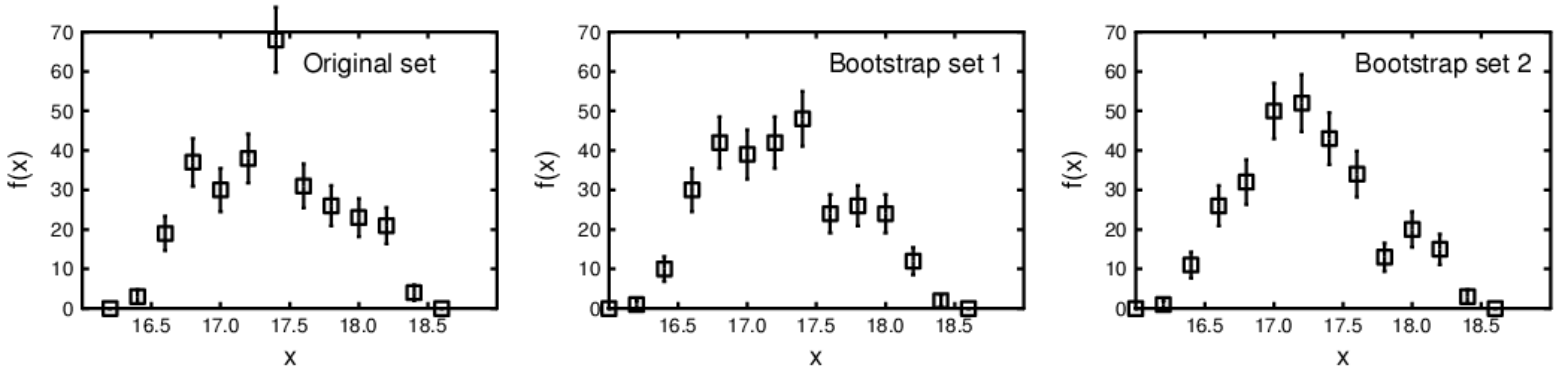
The apparent problem here is of course that offhand doing this would seem like circular deduction:

we use the same data first to calculate the average, then deduce the error. But this is not necessarily a real problem: the distribution after all in addition to having an average, contains also some information about its own width and shape, and the latter information is related to the error.

6.2.1. Test of bootstrap method

Since the method seems suspicious, we will now *test it* using the data generated in the previous section as a starting point. Since we there started with a known distribution, we could determine the error very reliably to 0.027. Now we can take one of the lousy, 300-count distributions and repeat the whole analysis based on that. If the bootstrap method works, it should give an answer close to 0.027.

Based on one 300-count distribution I generated new ones. Below is the original and two new examples:



The new distributions do resemble the original, as far as one can say based on the very poor statistics.

When I then ran the whole error analysis on the original data set (as described above), I got finally an uncertainty of the mean of 0.0285. This is almost exactly the same as above. To be sure this was not just fortuitous, I repeated the process starting from several different data sets, and the answers lied in the range $0.027 - -0.029$. So at least in this particular case the bootstrap method is actually amazingly good for estimating the error!

Note that in a run starting from a measured data set, the mean of the new data sets generated by MC will on average be the same as the mean of the original set. Despite the fact that the mean most likely does not match exactly the true mean, the estimate of the error is still valid. It is of course not possible to improve on the mean estimate using MC – that would really be getting something for nothing!

If this sounds like a paradox, thing of it like this: the information on the mean is easily evaluated from the original data set. The MC is just used to analyze how the width and shape of the distribution lead to an error.

From this example we see that the bootstrap method can work. However, there are some assumptions about the data that must be fulfilled. **The first** is that the data points should be independent: that if, the value of $f(x_{n+1})$ should not be directly correlated to the value of f_n. E.g. points in a spectrum are usually not correlated, but if you measure time-dependent data (say

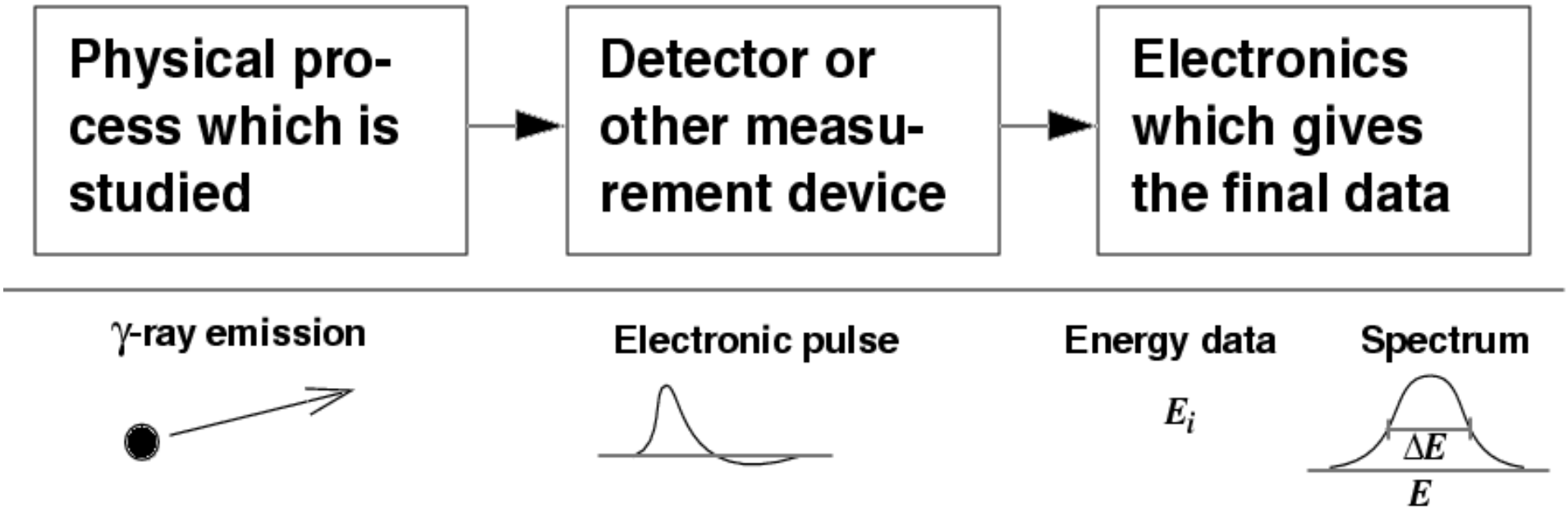
the motion of an oscillator) there may well be direct correlations. **The second** is that the points should be identically distributed with respect to the statistical analysis performed. This means that e.g. when you do the sum to get the mean, it does not matter in which order you sum the points. [Numerical Recipes ch. 15.6].

But provided this is true, the bootstrap method can often be the only possible decent way to analyze the error of a measured data set, especially if it clearly does not have a Gaussian-like shape.

6.3. Simulation of underlying physical process

A much more advanced means to get synthetic data sets is to actually simulate the underlying physical process and measurement which produces the data.

This is far more complicated than the above, since it requires good understanding of the physical process, and at least fair knowledge of every step in the measurement process. A simplified view of a measurement process is



The latter two boxes represent the measurement electronics.

What this means is probably easier to understand with a concrete example. The measured process could be nuclei emitting γ-rays. (this example is illustrated in the lower part of the figure). The detector could then e.g. be a solid-state γ particle detector giving an electronic pulse for each γ particle that passes through. The integral of the pulse is proportional to the energy of the particle. The electronics would then collect each pulse and integrate it to give a measurement of the energy, and collect statistics of all the pulses to give a spectrum.

This process is of course not exact, but there is an uncertainty associated with each stage, which has to be taken into account in the simulation of the data.

For synthetic data generation which can be compared with the real data, we have to be able to mimic all stages.

Fortunately for most well-established measurement devices the uncertainty is known. Either it has been provided by the device producer, or measured in the laboratory using some test system. Most often the instrumental uncertainty is simply a profile which will broaden the signal coming in. And most often this profile is of Gaussian shape. In case all instrumental responses are of this character, taking account of the measurement devices is not difficult – one simply has to convolute the result of the physical process with a composite profile which accounts for all the devices.

This leaves us still with the task of simulating the physical process itself. How to do this is of course entirely application-specific, and it is impossible to say anything general about how it should be done (except that it should be done right!).

But to give an idea of what this can be like, I will in the next section give a real-life example.

6.3.0.1. How can MC simulation of measurements be useful?

Generating synthetic data by simulation of the physical process **can be used to analyze uncertainties of the data**, in a similar manner as discussed in section .

But in addition to the statistical analysis, there can be other, even more important motivations for simulating the underlying process.

One of these is advance **analysis of what it is possible to measure**. Say for instance you want to measure an effect which is just at the limit of the resolution of the system, and you are uncertain of whether the number of statistics you can collect is enough to give a statistically significant answer. Then, if you are able to generate synthetic data sets based on the physical process, you can use the MC simulation to find out whether it is worth doing the measurement. Since in many cases measurements can be very expensive to do, and even require traveling halfway around the world, it can be very valuable to determine in advance whether it is even worth trying.

Related to this, you can also use MC simulation in advance to **determine the optimal conditions in which the measurement should be carried out**. This could e.g. be finding what experimental condition gives the best resolution. The "experimental condition" could be straightforward matters like the distance from the sample to the detector.

Another, a bit more complex use, is to **analyze how different parts of the process are reflected in the end result**. This can be important for understanding what parts in a complicated system are important. Simulations are highly versatile for doing this, since it is possible to do tricks like turning one part of the process off completely to see how this affects the end result.

Along similar lines, one can also attempt to determine why **experimental results are not as expected from theory.** You could carry out process simulations which use less approximations than the previous theory, trying to find out what could explain the unexpected feature.

Once the reason to the discrepancy has been found, the simulation can then be used to **determine the values of the previously missing quantities**.

This leads us to think about the ultimate use of simulations: **trying to figure out what the**

physical process is. Let's say you do not know for certain what physical process gives rise to the measured result, but can make one or more educated guesses. Then you could do process simulations based on the guesses, and attempt to find the one (hopefully only one!) which explains the measured result. This topic brings us beyond data analysis, since this starts to be the highest level of use of simulations: that of trying to find entirely new physical effects. While this topic certainly is fascinating, we will not talk more about it since it does not really belong to this section.

6.3.1. Example: simulation of GRID

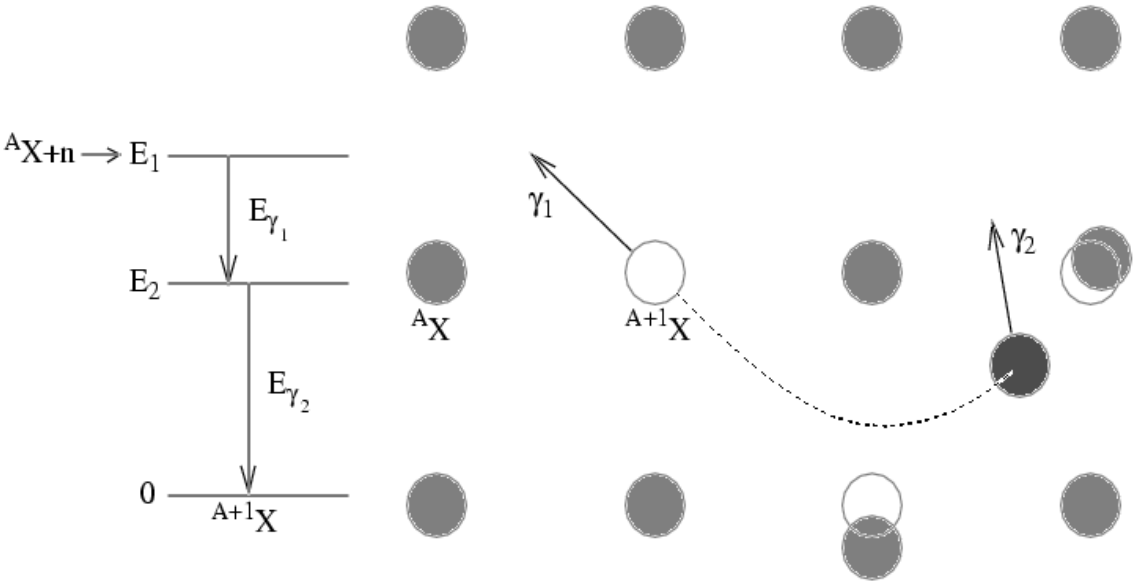
[Kai Nordlund, Master's Thesis, Univ. Helsinki 1993.]

The Gamma-ray induced Doppler broadening (GRID) process is an effect related to nuclei being subject to thermal neutron irradiation. The thermal neutrons are typically produced in a nuclear reactor after passing through a moderator).

Consider a stable nucleus ${}^{A}X$ which absorbs a thermal neutron:

$$ {}^{A}X + n \Rightarrow {}^{A+1}X $$

The product nucleus usually is in an excited nuclear state, lying at an energy E_1 above the ground state $E = 0$. The nuclei will eventually decay back to the ground state. But sometimes this occurs via an intermediate state E_2, and GRID deals exactly with this situation. In this case the nuclear energy diagram can be given as in the left part of this figure:



What will now happen is that due to the conservation of momentum the nucleus will receive a recoil energy from the emitted gamma particle of energy E_{γ_1}. This recoil velocity will be

$$v_r = \frac{E_{\gamma_1}}{mc}$$

After the nucleus has received this recoil energy (which typically is of the order of 100's of eV's), it will move in the crystal and collide with other atoms, until eventually slowing down to thermal energies (at 300 K about 1/40 eV/atom), see right part of figure.

In case the second γ particle γ_2 is emitted before the nucleus has stopped, its energy will be Doppler broadened by a factor of

$$\frac{\Delta E_\gamma}{E_{\gamma_2}} = \frac{v_{\gamma_2}}{c}$$

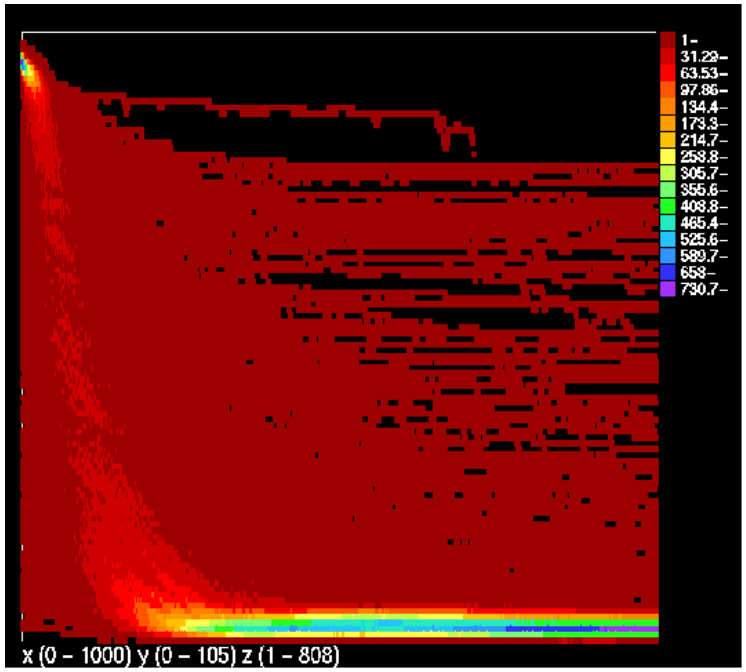
where v_{γ_2} is the velocity of the nucleus in the direction of the second γ decay. So if we now measure the second γ particle peak, its shape will be affected by the recoil energy.

The question we want to answer is, what is the shape of the Doppler broadened peak? The peak shape for a non-broadened decay would be close to a very narrow Gaussian (broadened by the instrumental resolution). To be able to predict the shape of the peak, we need to know the lifetime of the nuclear level, and how fast it slows down.

Predicting the slowing down can be done crudely using mean free path calculations, or much more accurately by molecular dynamics simulations. The use of MD for analysis of GRID data was initiated by A. Kuronen and J. Keinonen from our department around 1990, see e.g. Phys. Rev. Lett. **67** (1991) 3692. I came in the business in 1991.

From our MD simulations, we could predict the atom velocity as a function of time. For the best results, one needs to account for the whole distribution of velocities, which looks typically as follows.

The x axis is time, the y axis is the velocity compared to the original, and the color scale shows the number of recoils that have a given velocity at a given time. The color scale is logarithmic.



Clearly most atoms slow down at about the same rate (see the stronger red region in the left part of the figure). Hence as a first approximation we could use the average velocity as a function of time, $\bar{v}(t)$, for analysis of the GRID process. We know that the probability that the decay occurs in the time interval $[t, t + dt]$ decays exponentially as

$$\frac{e^{-t/\tau}}{\tau} dt$$

and we know how to generate random numbers in an exponential distribution. We can pick an arbitrary direction as the direction of the second γ ray, simply e.g. the $\mathbf{z}$ direction.

The quantity we want to calculate is the intensity

$$I(\Delta E_\gamma)$$

Then if we would want to get a first idea of how the γ peak is broadened, we could use the following MC approach to simulate the shape of the Doppler peak as a function of the lifetime τ:

0° Set lifetime τ and step width $\Delta(\Delta E_\gamma)$ of I array , zero I array.

1° Select a random direction in 3D $\mathbf{d}$ for a photon

2° Select a random lifetime for this particular photon: $t_p = -\tau \ln P(0,1)$

3° Find $\bar{v}(t_p)$.

4° Pick the projection of the velocity on the unit direction z: $v_z = \bar{v}(t_p)\mathbf{d} \cdot \hat{z}$.

5° Calculate $\Delta E_\gamma = E_{\gamma_2}\dfrac{v_z}{c}$

6° Get index in I array: $i_I = \dfrac{\Delta E_\gamma}{\Delta(\Delta E_\gamma)}$

7° Do $I(i_I) = I(i_I) + 1$

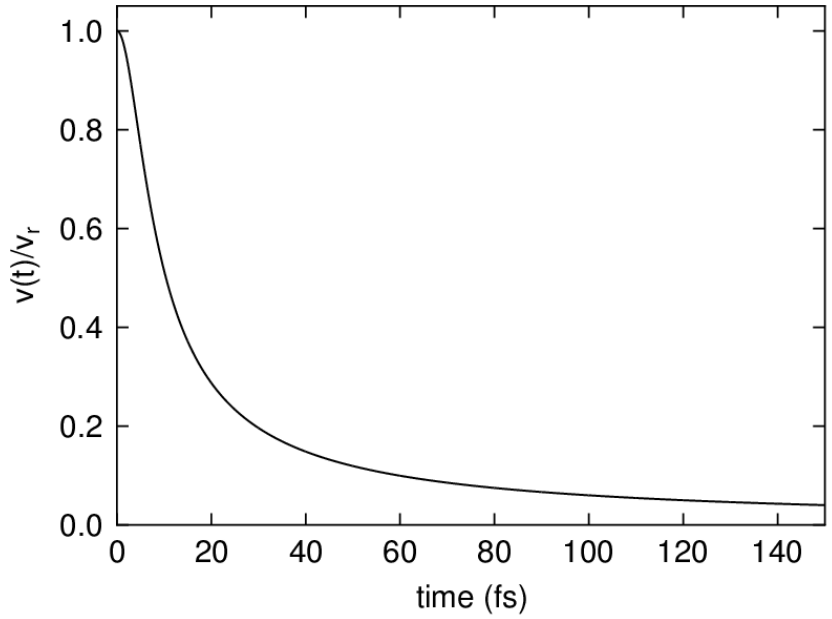
8° Return to step 1° .

This now neglects the instrumental and uncertainty principle broadening of the lineshape. These effects are important for comparison with experiments, but when we only want to understand the general behaviour they can be neglected - they will just broaden everything in the lineshape.

This is not complicated to code. For example, let us consider the decay scheme $8580 \rightarrow 2864 \rightarrow 0$ keV in ^{36}Cl. The lifetime was not known when we did the work, so we took that as a free parameter. From my experience with the MD simulations, I can say that the mean velocity relative to the initial one v_r can be estimated by something like

$$\frac{\bar{v}(t)}{v_r} = \cos\left(\tan^{-1}(t/t_s)\right) \quad \text{(Crude approximation only!)}$$

which is chosen so that it gives about the same shape as the mean in the previous figure. The characteristic slowing down time is of the order of $t_s = 10$ fs. Thus the average velocity in this approximation looks like



Here is an implementation of the algorithm (in awk, but almost identical to C). This could be easily optimized to become faster, but for clarity that has not been done.

```
# Select variable
tau=20;
# Set needed constants
pi=3.14159265358979; c=299792458; u=1.6605655e-27; eV=1.6021892e-19;
# Set constants for the particular transition and nucleus
Egamma1=(8580-2864)*1e3; Egamma2=(2864-0)*1e3;
m=36*u; ts=6.0;
# Select number of decays to simulate and number of elements in array
N=100000; Nel=100;

# Zero I array
for(i=-Nel-1;i<=Nel+1;i++) I[i]=0;

# Calculate recoil energy from first decay
vr=Egamma1*eV/(m*c)

# Calculate maximum possible Delta E from vr
DeltaEmax=Egamma2*(vr/c);

# Select step size for I array
DeltaDeltaEgamma=DeltaEmax/Nel;

# Main loop over decays
for (n=0;n<N;n++) {
        # Steps 1° and 2° : random velocity and lifetime
        theta = acos(1-2*rand());
```

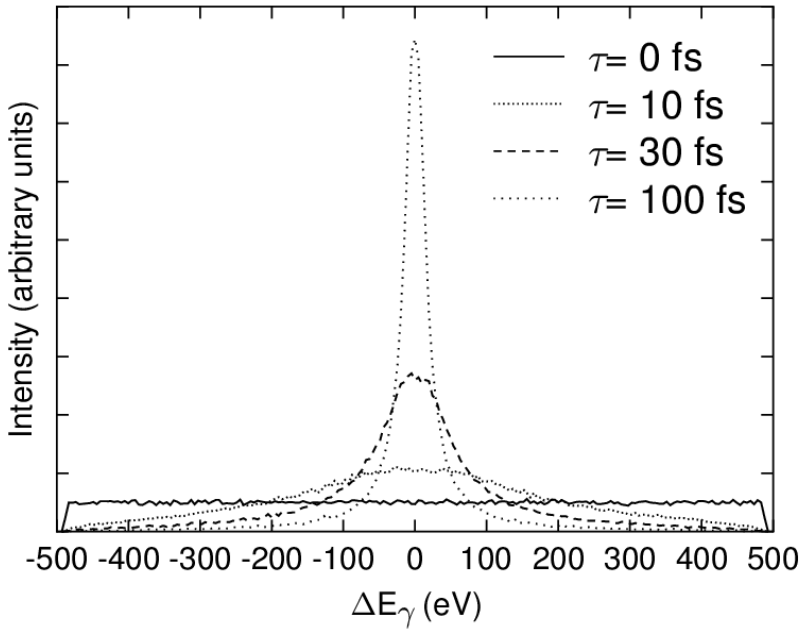
```
        phi = 2*pi*rand();
        tp = -tau*log(rand());

        # Steps 3°  - 5° : calculating the physical quantities
        v = vr*cos(atan2(tp/ts,1));
        vz = v*cos(theta);
        DeltaEgamma = Egamma2*(vz/c);

        # Steps 6°  - 7° : doing the statistics
        if (DeltaEgamma >=0) iI=int(DeltaEgamma/DeltaDeltaEgamma+0.5);
        else iI=int(DeltaEgamma/DeltaDeltaEgamma-0.5);
        I[iI]=I[iI]+1;
}
# Print out result
for(i=-Nel-1;i<=Nel+1;i++) printf("%10.2f %8d\n",i*DeltaDeltaEgamma,I[i]);

exit;
```

The results, for several different lifetimes, look as follows

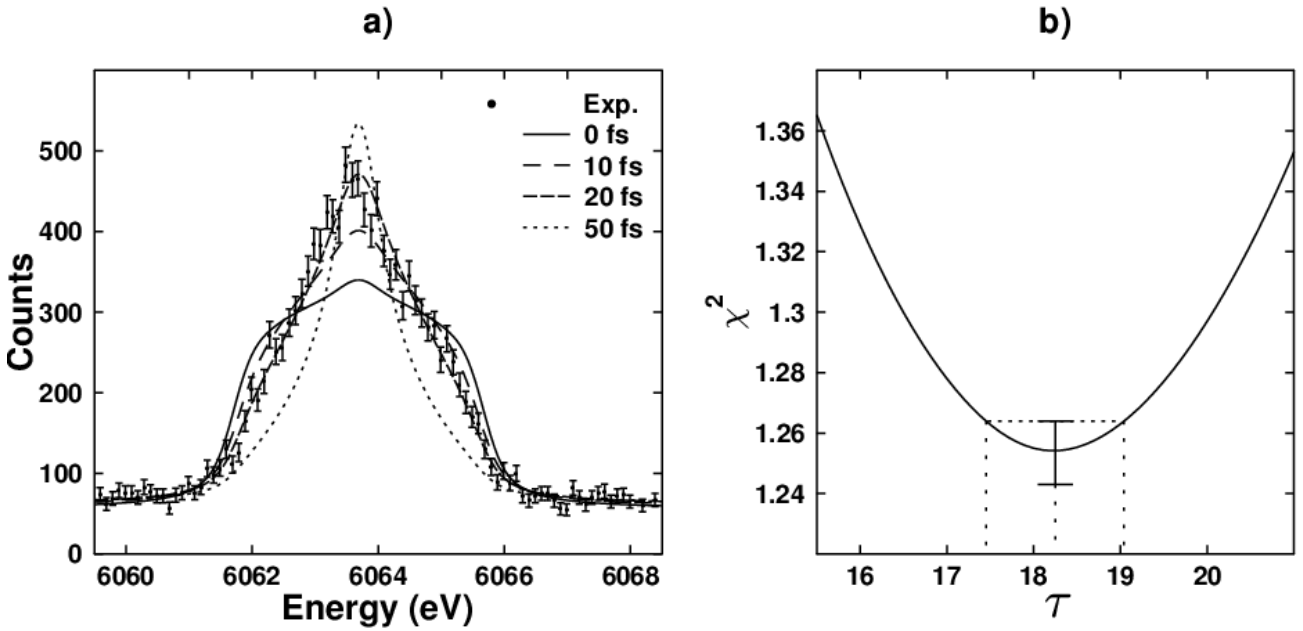


Now we clearly see the qualitative behaviour: for very short lifetimes the lineshape is box-like, because all velocity z components are equally probable. For longer lifetimes, as the ion slows down, the shape becomes more and more like the sharp peaks expected for normal emission without any Doppler broadening.

What we did back around 1990-1994 is derive the velocity distribution (not just the average) from MD simulations of the collisional process, and also add all the convolution terms of the measurement to be able to compare with experiments. We then MC-MD simulated data of different

lifetimes and compared to experimental data measured at the ILL in Grenoble. The simulated profile which fit the experiments best gave the lifetime of the nuclear level.

For this particular transition the comparison looked as follows (note that the energy scale on the x axis is wrong). The left part shows the fit of the simulated curves to the experimental data, the right side the minimization of the quality of the fit as a function of the lifetime.



In this case, the predicted lifetime would have been about 18 fs. In the end, after accounting for secondary intermediate decays as well, we obtained a lifetime of 21 ± 1 fs.

In all, we obtained lifetime values for tens of nuclear levels. In addition, we also used this method

for well-known lifetimes to examine the strength of the atomic interactions at intermediate energies, but that is a long story.

You do not fully have to understand this example. But hopefully it gave an idea of two things: how simple MC simulations (like the short code presented) can be useful for understanding what is going on, and how a more complex simulation scheme (the way of doing it all with MD and convoluting with the instrumental resolutions) can be used together with experiments to derive physical data.