Unterschrift des Betreuers

Diplomarbeit

Discussion, simulation and foundations of experimental tests of phenomena with superimposed signals by the example of time dependent radioactive decay

ausgeführt am
Atominstitut
der Technischen Universität Wien

unter der Anleitung von

Em.Univ.Prof. Dipl.-Ing. Dr.techn. Gerald Badurek
und
Ass.Prof. Dipl.-Ing. Dr.techn. Erwin Jericha

durch
Lukas Császár, BSc.
Rainergasse 31/28
1040 Wien

Datum

Unterschrift Student

Kurzfassung

Um in den von einem realen Messaufbau gelieferten Daten einen Effekt geringer Amplitude vom statistischen und systematischen Untergrund zu isolieren, ist, vor allem wenn beide Beiträge von vergleichbarer Größenordnung sind, die Kenntnis über den systematischen Untergrund eine wesentliche Voraussetzung. Ist das eigentliche Signal besonders klein, so verschwindet es mitunter vollständig im Untergrund.

Um die Möglichkeiten derartiger Messsituationen zu untersuchen wurde eine Computersimulation geschrieben, die es dem Nutzer ermöglicht, Datensätze zu simulieren als stammten sie aus einem realen Versuchsaufbau. Dabei können die vorherrschenden physikalischen Gesetzmäßigkeiten naturgemäß nach Belieben vorgegeben werden. Anhand der Simulationsergebnisse kann untersucht werden, ob und unter welchen Bedingungen das tatsächlich vorhandene Messsignal statistisch signifikant nachweisbar ist. Dazu lassen sich in der Simulation auch verschiedene Analysemethoden implementieren und anwenden.

Als Arbeitsbeispiel wurde die Idee zeitabhängigen radioaktiven Zerfalls (E. Fischbach, J. H. Jenkins, P. A. Sturrock et al.) herangezogen und durch ein einfaches physikalisches Modell – eine Modifikation des allseits bekannten exponentiellen Zerfallsgesetzes um eine zeitabhängige Oszillation der Zerfallskonstante – beschrieben. Mittels der durch die Simulation erzeugten Datensätze wurden Analysemethoden wie Autokorrelation, Glättung und verschiedene Arten der Normierung auf ihre Tauglichkeit, die eingeführte Oszillation zu detektieren, getestet.

Es stellte sich heraus, dass nur eine Kombination aller möglichen Analysemethoden zu einem sinnvollen Ergebnis führen kann. Es ist damit aber gelungen, Oszillationen mit einer relativen Amplitude von bis zu 10^{-7} nachzuweisen.

Die entstandene Simulation und die theoretische Arbeit können als Grundlage für andere Anwedungsbereiche dienen.

Abstract

In order to distinguish effects of small amplitude from the statistical and systematic background in the data gathered by a real measurement setup, knowledge about the systematic background is essential, even more so if both contributions (measurement signal and background signal) are of comparable magnitude. Tiny signals might become even completely obscured.

To allow for investigation of such measurement situations, a computer simulation framework was established. It enables the user to generate data sets similar to those obtained by a real experimental setup. Naturally, the applied physical laws in such a computer program can be arbitrary. The results of the simulation are used to investigate under which circumstances an actual effect can be determined with statistical significance. In order to do that, various analysis methods can be implemented and used in the framework.

Time dependent radioactive decay (E. Fischbach, J. H. Jenkins, P. A. Sturrock et al.) serves as an example to establish the contents of this work. On the basis of a simple model of such "new physics" – given by the introduction of a time dependent decay constant into the well-known exponential law of decay – it was attempted to isolate the introduced oscillation by means of several analysis methods like autocorrelation, smoothing and different kinds of normalization.

It turned out that only a combination of all available methods can provide usable results. That way it was possible to detect oscillations with a relative amplitude of down to 10^{-7}.

From such tests one can derive conclusions to support real experiments and characterize essential parameters. The flexibility of the framework allows the application to other physical models as well.

Contents

List of Figures

1. Introduction

In every real measurement situation the experimenter is confronted with data in which the actual measurement signal is superimposed on a statistical and systematic background. The modeling of said background is always of great importance when judging the significance of the data. If both contributions in the signal (data of interest and background) are of the same order of magnitude a proper analysis can be difficult – all the more, the less pronounced the amplitudes of the measurement signal.

In some extreme cases, the measurement signal is apparently invisible and entirely obscured by statistical fluctuation and the systematic background. However, if the investigated signal is periodic or the nature of the systematic background known, this can help (or even make possible at all) to differentiate the signal components.

The thesis in hand aims to develop analysis methods in order to find such periodic signals of small amplitude and isolate them from a dominant background. This is achieved by theoretical considerations and by means of an extendable computer simulation framework. The latter provides the means to generate and analyze simulated measurement data.

The key idea of the simulation involves the generation of synthetic data containing a small, obscured signal the nature of which is defined by an implemented, physical model. By a set of analysis methods (which are also part of the simulation framework) it is then attempted to detect this signal content. From such tests one can derive conclusions to support real experiments and characterize essential parameters.

Oscillations of order 10^{-3} in the nuclear decay rates of some investigated nuclides as reported by E. Fischbach, J. H. Jenkins, P. A. Sturrock et al. may serve as an interesting example to build the theoretical and programming work on. The signal of interest would be a tiny sinusoidal oscillation of the activity superimposed on an exponential background with large statistical fluctuations, since radioactive decay is a random process.

They support the hypothesis that there is some unknown solar influence on nuclear decay rates, which becomes visible through annual oscillations varying with Earth-Sun distance [1]. These findings mainly revolve around two sets of data, one from the Brookhaven National Laboratory concerning the decay rate of a Si-32 sample recorded from 1982 until 1989 and one from the Physikalisch-Technische Bundesanstalt where (among others) a Ra-226 sample was monitored from 1983 until 1998 [2, 3]. Furthermore, perturbations in a Mn-54 sample's decay rate coinciding with severe solar flares and storms in December 2006 strengthen their postulation of a solar influence on nuclear decay processes [4]. Several more findings of other groups are listed in [5] as is a table of experiments exhibiting time dependent decay rates.

They attribute the observed effects to a solar influence. Seasonal changes in the Earth-Sun distance due to the orbit's eccentricity could therefore account for annually varying decay rates. Additionally, all observed effects concern β-decays or were β-decay related (such as β-decaying daughter products in equilibrium with their parents), suggesting a possible influence of solar neutrinos [5]. They even discuss the potential existence of a new field/particle; the neutrello [6]. α-decay seems to be entirely unaffected ([5] and references therein).

Also, oscillations of other frequencies have been found in the data sets, which were linked to possible internal solar activities [7].

Naturally, their inference did not go unchallenged, since such findings speak against our understanding of radioactive decay and many other researchers opposed these claims [8–11].

Quoting [10]: *When radioactivity was first being studied in the early 1900s, the question was raised if environmental factors could change the speed at which radioactive atoms decay. Several experiments conducted by Earnest Rutherford, Pierre and Marie Curie, and others found that the* *decay constant had no measurable influence, within measurement uncertainties, to environmental conditions, including temperature, pressure, chemical composition and magnetic fields. Recently, starting in 2009, a series of studies and measurements have been conducted to re-investigate whether the decay constant is truly constant. This recent work has been focused on Earth-Sun distance variation and solar flare activity and how they might influence the decay constant.*

Quantum mechanics does predict that under under special circumstances, decay of unstable quantum states, including unstable nuclear ground states, is not always exponential [...]. *It has also been demonstrated experimentally that the radioactive decay constant can be influenced by external forces for the electron capture decay mode* [...], *this is a well-understood consequence of changes to the phase space* [...].

The main criticism of the conclusions drawn by Jenkins and Fischbach has been that the data which they based their conclusions on were acquired mostly by old technology detector systems (gas filled detectors) which were not in climate controlled rooms, nor were the environmental conditions measured during the time of radiation measurements. Jenkins et al. concludes that "there is an obvious need for more experiments examining a wide variety of detector technologies and a large array of isotopes utilizing appropriate controls, including directly measuring temperature, pressure and humidity effects on the detector systems in environmental chambers".[1]

Dedicated high-precision experiments and investigations in the references listed above find no evidence for sought oscillations. For instance [11]: *The most stable activity measurements of alpha, beta-minus, electron capture, and beta-plus decaying sources set an upper limit of 0.0006 % to 0.008 % to the amplitude of annual oscillations in the decay rate. Oscillations in phase with Earth's orbital distance to the Sun could not be observed within a 10^{-6} to 10^{-5} range of precision. There are also no apparent modulations over periods of weeks or months.*

The publications mentioned above were discussed within the neutron and quantum physics group of the Atominstitut and E. Jericha decided in 2012 to launch perpetual data mining using a spare germanium photon detector "Polygam" of the institute without any further planning. Polygam was set up in a measuring room in the second basement of the Atominstitut, TU Wien, which it shares with two other detectors, the latter of which is low level shielded, and a mass spectrometer. Polygam is regularly used in the frame of the neutron physics laboratory course, two weeks per semester.

Until now, this issue has been subject of [13] and [14], however, the thesis in hand did not use the mentioned resources, as an "unspoiled" approach seemed appropriate. Furthermore, these theses do not provide a description of the investigated effect and focus on the evaluation of data gathered by the aforementioned experiment.

Early parts of the thesis dealt with finding ways of refining the available setup. Since, very quickly, the focus of the work shifted to the establishment of the theoretical background and the simulation and analysis framework, this thesis will only very briefly touch on several aspects of a possible experimental setup.

Section 2 Physics and math (p. 15) covers simple physical models of (non-)standard radioactive decay, provides general information on the measuring process and introduces the mathematical analysis tools.

Section 3 Simulation (p. 47) discusses the design and contents of the simulation framework and how to use and possibly extend it.

Section 4 Application (p. 73) combines the two preceding chapters and tests the methods and ideas introduced in the second chapter by using the established framework.

Section 5 Experimental setup (p. 103) briefly discusses the experimental setup and points out possible improvements.

[1] Jenkins et al. present a detailed analysis of the detector responses to environmental influences and conclude that these effects are unlikely to explain the observed periodic fluctuations [12].

Finally, section 6 Conlusion and outlook (p. 111) provides a collection of results from section 4 Application (p. 73), briefly discusses other possible fields of application and gives a list of items of future work.

2. Physics and math

This section is devoted to the presentation of physical and mathematical concepts necessary to describe the issues of this thesis.

2.1. Radioactive decay

The work takes place in the frame of radioactive decay of nuclei. It is thus essential to be well informed about its phenomena and laws.

Radioactivity is the property of certain nuclei (so called radionuclides) to *spontaneously*, i.e. independent from external influences), disintegrate/convert (radioactive decay) into other nuclei. In the process, energy in the form of particles – α-, β- or γ-decay – is released.[2]

The latter, electromagnetic radiation from γ-decay, is subject of the thesis in hand. Gamma spectroscopy examines the characteristic energy of the released radiation (photons) and thus allows to study the decay process.

Since the experiment and research described follow the tracks of the assumption that there are still aspects about the decay process to be unveiled, it will be referred to as "standard decay" in the following.

2.1.1. The law of radioactive decay

The following ordinary differential equation was found to describe the effect in question:

$$\frac{\mathrm{d}N}{\mathrm{d}t} = -\lambda N \quad , \tag{2.1}$$

where $N = N(t)$ is the current quantity of nuclei, t is the time and λ is a constant factor – the decay constant with dimension $[\lambda] = 1/\mathrm{s}$. The minus sign is used since the number of nuclei has to decrease over time. Its reciprocal

$$t_\lambda = 1/\lambda \tag{2.2}$$

is called lifetime ($[t_\lambda] = \mathrm{s}$).

Solving the equation results in what is known as the law of radioactive decay, namely

$$N(t) = N_0\,\mathrm{e}^{-\lambda t} \quad , \tag{2.3}$$

where $N_0 = N(t_0) = N(t = 0)$ is the number of nuclei existing at the beginning.

The time $t = t_{1/2}$, after which only half of the initially existing nuclei remain, is called half-life. With $N(t_{1/2}) = 1/2 \cdot N_0$ we find

$$\lambda = \frac{\ln 2}{t_{1/2}} \tag{2.4}$$

to link decay constant and half-life, of which the latter is commonly found in literature.

The statistical process of radioactive decay follows a Poisson distribution. For a large number of nuclei, the process can be described by a Gaussian normal distribution with mean $\mu(t) = N(t)$ and standard deviation $\sigma(t) = \sqrt{N(t)}$. The details of this statistical treatment will be extensively covered in section 2.3.4 Statistical description of radioactive decay (p. 30).

[2] Freely translated from [15].

2.1.2. Remaining, decaying and decayed nuclei – basic formulas

As explained above, the ansatz of a simple differential equation of first order in time and its solution (in combination with a starting condition) leads us to a function, which describes the number of *remaining* nuclei.

$$\boxed{\frac{\mathrm{d}N(t)}{\mathrm{d}t} \propto N(t) \quad \xrightarrow[\text{equation}]{\text{solve differential}} \quad N(t) \text{ with } N(t=0) = N_0} \tag{I}$$

Let us remember the just introduced law of standard decay (2.3) from the last section at this point. Whereas it only provides the number of *remaining* nuclei in the sample, it is useful to think about the mathematical description from another "perspective".

We want to find the total amount of nuclei N_d which already *decayed* at a given time and therefore state a (very intuitive) fact: The sum of already decayed and still remaining nuclei must always remain constant; N_0, to be exact.

$$N(t) + N_\mathrm{d}(t) \equiv N_0 = \text{const} \quad \forall t \geq t_0$$

This quickly leads to

$$\boxed{N_\mathrm{d}(t) = N_0 - N(t)} \tag{II}$$

and

$$N_\mathrm{d}(t) = N_0\left(1 - \mathrm{e}^{-\lambda t}\right) \tag{2.5}$$

in terms of standard decay.

If we want to describe the decay process at a certain (infinitesimal) time rather than over a time interval (the previous two formulas gave us information about the time interval $t - t_0$), we naturally define

$$\boxed{n(t) = \frac{\mathrm{d}N_\mathrm{d}(t)}{\mathrm{d}t} = -\frac{\mathrm{d}N(t)}{\mathrm{d}t} \propto N(t) \ , \quad \text{where} \quad \operatorname{sgn} n(t) \overset{!}{=} 1 \quad \forall t \geq t_0} \ . \tag{III}$$

as the decay rate, thus, the number of currently *decaying* nuclei.[3] Care must be taken to guarantee the easily overlooked condition of a constant sign!

$$\operatorname{sgn} n(t) \overset{!}{=} 1 \quad \forall t \geq t_0 \tag{IIIa}$$

A change of sign – once a "perspective" has been chosen – would indicate "a wrong direction" of the process, which is against physical laws. For instance, a negative sign in (III) would describe decayed nuclei reassembling!

We decide on the "perspective of the experimenter" to properly depict the experimental setup; using the expressions of standard decay, we obtain in this case

$$n(t) = \lambda N_0 \,\mathrm{e}^{-\lambda t} = \lambda N(t) \ . \tag{2.6}$$

[3] It is important to be aware of the frame of reference at this point. We define the decay rate from the perspective of the experimenter. We see the decay rate as the change of *decayed* nuclei, a growing function, thus we define as in (III) and the function must have a positive range. If we were to express the decay rate in the sample's frame, we would ask in what manner the number of *remaining* nuclei changes and consequently find

$$n(t) = \frac{\mathrm{d}N(t)}{\mathrm{d}t} \ , \quad \text{where} \quad \operatorname{sgn} n(t) \overset{!}{=} -1 \quad \forall t \geq t_0 \ ,$$

since it decreases.

Note that (III) is, in fact, the definition of activity

$$A(t) := -\frac{\mathrm{d}N(t)}{\mathrm{d}t} = \lambda N_0\,\mathrm{e}^{-\lambda t} = A_0\,\mathrm{e}^{-\lambda t} \quad , \tag{2.7}$$

the unit of which is $[A] = 1\,\text{Becquerel} = 1\,\text{Bq} = 1/\text{s}$, where an activity of $A = 1\,\text{Bq}$ means one decay process per second. $A_0 := \lambda N_0$ is the initial activity. Another common unit of activity is $1\,\text{Curie} = 1\,\text{Ci} = 3.7 \cdot 10^{10}\,\text{Bq}$.[4]

Math already suggests to find the number of nuclei which *decayed during a time interval* $[t_0, t_0{+}\Delta t]$ through integration after

$$\boxed{\Delta N(\Delta t; t_0) = \int_{t_0}^{t_0+\Delta t} n(t)\,\mathrm{d}t} \quad , \tag{IV}$$

or more exactly

$$\Delta N(\Delta t; t_0) = \int_{t_0}^{t_0+\Delta t} n(t)\,\mathrm{d}t = \int_{t_0}^{t_0+\Delta t} \frac{\mathrm{d}N_\mathrm{d}(t)}{\mathrm{d}t} = N_\mathrm{d}(t_0 + \Delta t) - N_\mathrm{d}(t_0) \tag{IVa}$$

$$\equiv -\int_{t_0}^{t_0+\Delta t} \frac{\mathrm{d}N(t)}{\mathrm{d}t} = N(t_0) - N(t_0 + \Delta t) \quad . \tag{IVb}$$

For standard decay we obtain

$$(\text{IVb}) = N_0\,\mathrm{e}^{-\lambda t_0}\left(1 - \mathrm{e}^{-\lambda \Delta t}\right) \quad . \tag{2.8}$$

Defining $t_0 = 0$ and $\Delta t \to t$ results precisely in case (II), namely

$$\Delta N(\Delta t \to t; t_0 = 0) \equiv N_\mathrm{d}(t) = N_0\left(1 - \mathrm{e}^{-\lambda t}\right) \quad .$$

Obviously, equations (I) to (IV) are profoundly intertwined. It is important to see the four introduced descriptions of radioactive decay – namely, remaining $N(t)$, decayed $N_\mathrm{d}(t)$, decaying (decay rate) $n(t)$ and "discretely decayed" $\Delta N(\Delta t; t_0)$ nuclei – as different points of view on *one* process.

Figure 1 shows the introduced functions.

2.2. Models – postulated influences and their mathematical description

2.2.1. New physics?

In contradiction to the preliminary findings, as presented in the introduction, stands the postulation of a time dependence of what we know as the decay *constant*, thus

$$\lambda \overset{?}{=} \lambda(t) \quad .$$

Since a sufficiently accurate model was found to describe the observed (macroscopic) nuclear decay processes, the scientific community focused on working off the heap of data which was now to be gathered; such as measuring half-lives, mapping the different nuclides on the well

[4] How N_0 can be calculated from a given A_0 or a measurement is pictured in 4.1.3 Magnitudes and values of introduced variables; exemplary case (p. 75).

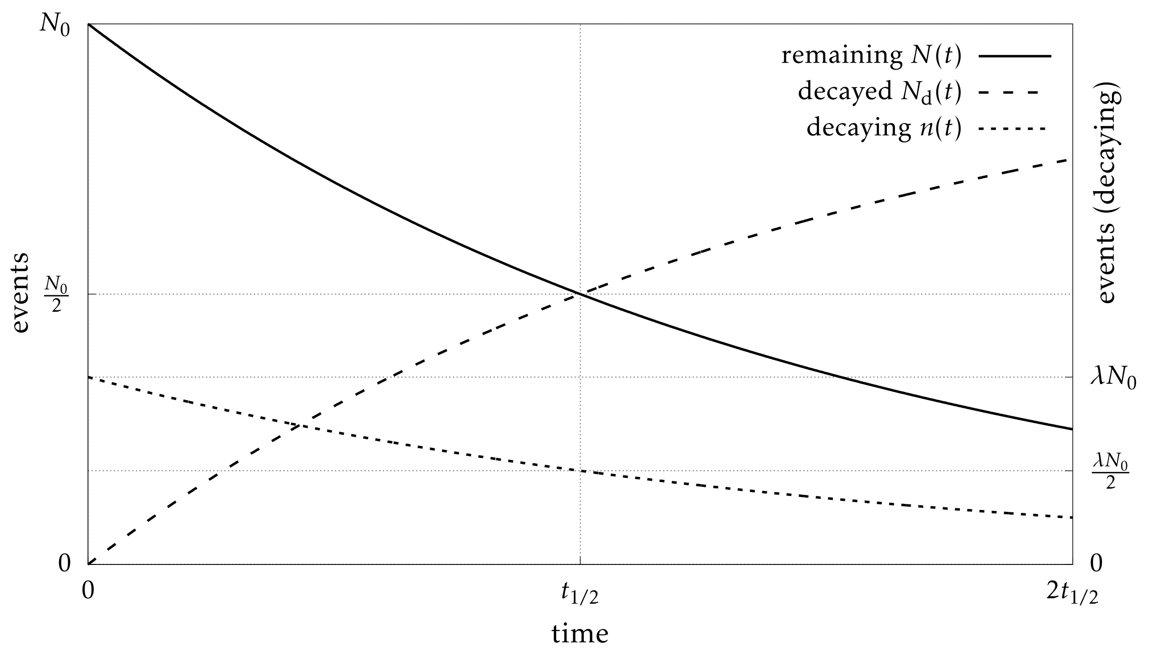


Figure 1 This figure shows the graphs of the remaining, decayed and decaying nuclei (standard decay). It is clearly visible that $n(t)$ is mathematically identical to $N(t)$. $N(t)$ and $N_0(t)$ add up to N_0 at all times. The $\Delta N(t;\Delta t)$ graph (not depicted), where now Δt is a constant parameter and $t_0 \to t$ is the variable, would lie between 0 for $\Delta t = 0$ and $N(t)$ for $\Delta t \to \infty$ and is again mathematically equivalent to $N(t)$ and differs only by a constant pre-factor. It is identical to $n(t)$ for $\Delta t = -\ln(1-\lambda)/\lambda$.

known chart of nuclides and expanding the knowledge of nuclear physics rather than diligently observing a, seemingly, entirely random process in order to find tiny but regular variations, the assumption of which going without any obvious indication. In short: There was too much to do, one would not even expect this kind of periodic fluctuation and, apart from that, it was irrelevant.

Quoting from the introduction of [16]: *In a recent paper, early data from a sample of Cs-137 on board the* MESSENGER *spacecraft enroute to Mercury were analyzed to set limits on a possible solar influence on nuclear decay rates. This work was motivated by the suggestion put forward in a recent series of papers which cite evidence for a drop in the count rate of Mn-54 during a solar flare, for a correlation between decay rates of various isotopes and Earth-Sun distance, and for periodicities in decay rate data associated with solar rotation. Although the suggestion of a solar influence on nuclear decay rates has been challenged by the apparent absence of decay anomalies in some isotopes that have been studied, and by a recent reactor experiment, there is no* a priori *reason to assume that all isotopes should be equally sensitive to a putative solar influence, or that the antineutrinos produced in reactors would be the dominant agents through which a solar influence would be exerted. As we have noted elsewhere, the very same properties of decaying nuclides that are responsible for the broad range of observed half-lives (e.g., nuclear and atomic wave-functions, Q-values, selection rules) would likely render nuclides sensitive in different degrees to a putative solar influence.*

The suggestion that the sun is responsible for variations in decay rates can be tested directly by studying the decay rates of appropriate nuclides located on spacecraft traveling through the solar system.

Assuming the sun being origin of new aspects of nuclear decay, it could contribute in two different ways: Irregular activities (solar flares for instance) and a permanent influence due to an unknown mechanism.

The first would probably cause temporary fluctuation, reflected in varying count rates of a measured spectrum.

The latter would cause a change of the decay rate possibly in dependence of the distance between the sun (or, in fact, any other star) and the position of the affected nucleus. Earth's orbit around the sun as well as its rotation would both cause a varying distance. The resulting effect would thus be twofold.

Fischbach et al. propose that neutrinos could be responsible this effect. The Triga reactor of the Atominstitut, for example, is a neutrino source and must therefore be taken into account. If the influence of neutrinos from the reactor is stronger than that of the sun (or other neutrino sources) one should be able to detect their impact by comparison of measured variations and reactor runtime.

However those hypothetical effects work, their influence must be so small that they do not obviously show up in data sets already gained in the course of the exploration of nuclear decay. Fischbach et al. suppose the effects to show up with a relative magnitude of 10^{-3}!

In order to describe the effects in question, we have to modify the standard model of radioactive decay and find appropriate versions of formulas (I) to (IV).

All models will introduce a well-defined periodical fluctuation of a certain amplitude ε into the signal. Depending on the ansatz of the respective model, the meaning of this parameter can vary.

Not all models presented in this section were actually simulated and investigated.

2.2.2. General sinusoidal Sun influence

A possible spatial dependence of $\lambda = \lambda(r)$ due to the sun's influence can be transformed into a time dependence, considering that Earth orbits the sun in a well understood manner.

$$\lambda(r) = \lambda(r(t)) = \lambda(t)$$

This movement involves a varying distance that will be described by a periodical function. Furthermore, the Earth's rotation will cause variations of the distance between the sun and the experiment as well, however being of the magnitude of Earth's diameter, which is negligible in comparison to the distance of approximately one astronomical unit[5].

A very general description according to [16] reads

$$\lambda(t) = \lambda_0 + \lambda_1(\vec{r}, t) \ ,$$

where λ_0 represents the intrinsic contribution to the decay rate of the unstable nuclei, along with a possible time dependent background arising from new interactions. Here $\lambda_1(\vec{r}, t) \ll \lambda_0$ characterizes the anomalous position and time dependent contribution to the decay rate, assumed, in this case, to arise from the sun. [...] *there is evidence using terrestrial radioactive samples suggesting that decay rates are correlated with sample-Sun separation. To study this effect, we will assume a specific phenomenological form for $\lambda_1(\vec{r}, t)$ given by*

$$\lambda_1(\vec{r}, t) = \lambda_0 \, \xi^{(n)} \left(\frac{r_0}{r(t)} \right)^n \ ,$$

where $n \in \mathbb{N}$, $r = |\vec{r}|$ is now the sample-Sun separation distance, $r_0 = 1\,\mathrm{AU}$ and $\xi^{(n)}$ is a composition dependent dimensionless parameter characterizing the strength of the decay anomaly for a specific nucleus. The form of [the equation] *is designed to encompass a broad range of theories. If the decay*

[5] 1 astronomical unit = 1 AU = 149 597 870 km $\approx 1.5 \cdot 10^8$ m [17].

anomalies are caused by a flux of neutrinos from the sun or by an inverse-square law field, then one expects $n = 2$.[6]

2.2.3. Sinusoidal, time dependent summand

General description

The following ansatz assumes that a certain percentage $0 < \varepsilon \ll 1$, of the remaining nuclei shows sinusoidal decay, whereas a fraction of $(1-\varepsilon)$ follows the unmodified law of decay. As a consequence, we obtain simply a modified version of the standard decay function for remaining nuclei (in accordance with (I)), which reads as follows:

$$N(t) := (1-\varepsilon)\cdot N_0\,\mathrm{e}^{-\lambda t} + \varepsilon\cos(\omega t+\varphi)\cdot N_0\,\mathrm{e}^{-\lambda t} \quad ,$$

where ω is the frequency of the periodic variation (and is correlated with the orbit of the earth) and φ is a phase shift.[7] The frequency is proportional to the reciprocal value of the effect's cycle duration T, thus

$$\omega = \frac{2\pi}{T} \quad , \quad [\omega] = \mathrm{rad/s} \quad . \tag{2.9}$$

T will be of the order of a year for the orbit and of a day for the rotation respectively.

Note that in this ansatz the decay constant is $\lambda = \mathrm{const}$, but a time dependent, periodical summand is added to factor in the assumed influence of the sun.

A more general version of this "brute-force" ansatz reads

$$N(t) := N_0\,\mathrm{e}^{-\lambda t}\cdot[1+\eta\varepsilon+\varepsilon\cos(\omega t+\varphi)] \quad , \tag{2.10}$$

where

$$\eta = \begin{cases} +1 & \text{inhibiting} \\ \;\;0 & \text{alternating} \\ -1 & \text{inducing} \end{cases} \quad .$$

For $\eta = -1$ we obtain the case presented before, in which a fraction of $(1-\varepsilon)$ nuclei show sinusoidal decay, which matches an inducing effect, the maximum values being the ones predicted by the standard decay equation. If this seems counter-intuitive, remember that $N(t)$ provides the number of *remaining* nuclei: In this case, this number is equal or less than the standard values and a diminished number of remaining nuclei means that the effect has induced decay processes. Analogously, $\eta = +1$ would describe an inhibiting effect, where the lowest occurring values are that of standard decay. The case of alternately inducing and inhibiting decay about the values corresponding to the standard case is given by $\eta = 0$. In all cases, ε is the amplitude of the postulated oscillation, the order of magnitude of which determines directly that of the effect. At the same time, it denotes which percentage of nuclei show a change in their disintegration behavior (i.e. their activity).

This model will be referred to as the "sinusoidal summand" model in the course of this text.

Formulas

The common differential equation already introduced,

$$\frac{\mathrm{d}N(t)}{\mathrm{d}t} = -\lambda N(t) \quad ,$$

[6] The mathematical models described throughout the thesis in hand developed *independently* from the cited papers. They were, on purpose, read *after* the formalism and simulation were written to avoid any bias!

[7] The choice of the cosine is due to consistency reasons with the next presented model.

underlies this description – its solution is just modified!

In order to satisfy the initial condition $N(t=0)=N_0$, we must also modify the integration constant.

$$N(t=0) = [1+\eta\varepsilon+\varepsilon\cos(\varphi)]\, N_0$$

leads to

$$N_0 \to \frac{N_0}{1+\eta\varepsilon+\varepsilon\cos(\varphi)} =: N_0(\varphi) \ ,$$

which yields the first equation:

$$\begin{aligned}
\text{(I)} \quad \to \quad N(t) &= \frac{N_0}{1+\eta\varepsilon+\varepsilon\cos(\varphi)}\, \mathrm{e}^{-\lambda t}\, [1+\eta\varepsilon+\varepsilon\cos(\omega t+\varphi)] \\
&= N_0(\varphi)\, \mathrm{e}^{-\lambda t}\, [1+\eta\varepsilon+\varepsilon\cos(\omega t+\varphi)] \qquad (2.11)
\end{aligned}$$

If we didn't want to neglect the earth's rotation compared to the orbit of the earth around the sun, we would choose a modification somewhat like

$$N(t) := N_0\, \mathrm{e}^{-\lambda t} \cdot [1+\eta(\varepsilon_{\text{orbit}}+\varepsilon_{\text{rot}}) + \varepsilon_{\text{orbit}}\cos(\omega_{\text{orbit}} t+\varphi_{\text{orbit}}) + \varepsilon_{\text{rot}}\cos(\omega_{\text{rot}} t+\varphi_{\text{rot}})] \ ,$$

where $\varepsilon_{\text{orbit}} \gg \varepsilon_{\text{rot}}$ but again $\varepsilon_{\text{orbit}}+\varepsilon_{\text{rot}} \ll 1$.

We find the other functions according to their definitions:

$$\text{(II)} \quad \to \quad N_{\text{d}}(t) = N_0 \Big[1-(1+\eta\varepsilon+\varepsilon\cos(\varphi))^{-1}\, \mathrm{e}^{-\lambda t}\, [1+\eta\varepsilon+\varepsilon\cos(\omega t+\varphi)]\Big] \qquad (2.12)$$

$$\text{(III)} \quad \to \quad n(t) = \lambda N_0(\varphi)\mathrm{e}^{-\lambda t}\Big[[1+\eta\varepsilon+\varepsilon\cos(\omega t+\varphi)] + \frac{\omega\varepsilon}{\lambda}\sin(\omega t+\varphi)\Big] \qquad (2.13)$$

$$\begin{aligned}
\text{(IVb)} \quad \to \quad \Delta N(\Delta t; t_0) &= N(t_0) - N(t_0+\Delta t) \qquad (2.14) \\
&= N_0(\varphi)\mathrm{e}^{-\lambda t_0}\Big[[1+\eta\varepsilon+\varepsilon\cos(\omega t_0+\varphi)] - \\
&\quad - \mathrm{e}^{-\lambda\Delta t}[1+\eta\varepsilon+\varepsilon\cos(\omega(t_0+\Delta t)+\varphi)]\Big]
\end{aligned}$$

Simplification

The phase φ defines the starting point in the effect's cycle. However, this information is neither of importance in the description of the model nor in the simulations or any real measurement.[8] Merely when fitting to a set of experimental data, a phase variable (fit parameter) has to be introduced to allow for a proper fit result. Additionally, it can be argued that the knowledge about Earth's position with respect to aphelion and perihelion allows to determine the approximate current phase, assuming that in the perihelion the effect is the greatest.

With these considerations in mind we will set $\varphi := 0$ for future reference. This leaves the trigonometric functions unchanged in that the phase variable can be re-introduced at any time without modification. This choice of φ introduces the negligible error of $[1+\eta\varepsilon+\varepsilon\cos(0)]^{-1} \approx 1$, because ε is by definition very small[9], which simplifies $N_0(\varphi) \approx N_0$. This approximation is supported by the fact, that uncertainties in the determination of N_0 will most probably always obscure any resulting errors. (In the simulations only the order of magnitude of N_0 is taken into

8 In theory, it can be determined in a normalized signal as shown in 2.4.3 Standard decay normalization (p. 43).

9 This is actually a very important point: If ε wasn't very small, it would have already been discovered.

account anyway.)

$$\text{(I)} \quad \rightarrow \quad N(t) = N_0\,\mathrm{e}^{-\lambda t}\,[1+\eta\varepsilon+\varepsilon\cos(\omega t)] \tag{2.15}$$

$$\text{(II)} \quad \rightarrow \quad N_\mathrm{d}(t) = N_0\Big[1-\mathrm{e}^{-\lambda t}\,[1+\eta\varepsilon+\varepsilon\cos(\omega t)]\Big] \tag{2.16}$$

$$\text{(III)} \quad \rightarrow \quad n(t) = \lambda N_0\,\mathrm{e}^{-\lambda t}\left[[1+\eta\varepsilon+\varepsilon\cos(\omega t)]+\frac{\omega\varepsilon}{\lambda}\sin(\omega t)\right] \tag{2.17}$$

$$\begin{aligned}\text{(IVb)} \quad \rightarrow \quad \Delta N(\Delta t;t_0) &= N(t_0)-N(t_0+\Delta t) \\ &= N_0\,\mathrm{e}^{-\lambda t_0}\Big[[1+\eta\varepsilon+\varepsilon\cos(\omega t_0)]- \\ &\quad -\mathrm{e}^{-\lambda\Delta t}[1+\eta\varepsilon+\varepsilon\cos(\omega(t_0+\Delta t))]\Big]\end{aligned} \tag{2.18}$$

2.2.4. Sinusoidal, time dependent decay constant

General description

In this ansatz of the differential equation modeling decay, the decay "constant" is composed of a static and an oscillating contribution

$$\lambda(t) = \lambda_0(1+a\sin(\omega t+\varphi)) \approx \lambda_0 \ , \tag{2.19}$$

where λ_0 represents the standard decay constant and $a \ll 1$ is a dimensionless factor ($[a]=1$), resembling the strength of the postulated effect.[10] This mathematical ansatz introduces both induction and inhibition of disintegration by its very nature. The other introduced variables are analogous to the model described above.

In this model, the free parameter a has some significant physical meaning and is an attribute of the sample. The magnitude of the resulting effect (deviation from the expectations of the model of standard decay) will still be given by the quantity ε, however, in this model, it *results* from the physical model, i.e. $\varepsilon = \varepsilon(\lambda_0, a, \omega)$, whereas in the preceding model it was predefined; in direct comparison, it is important to beware $a \neq \varepsilon$. Mind also that a itself is not directly accessible by the experimenter – its nature is entirely unclear!

This model will be referred to as the "sinusoidal decay constant" model.

Formulas

The differential equation is then (check against (I))

$$\frac{\mathrm{d}N(t)}{\mathrm{d}t} = -\lambda(t)N(t) = -[\lambda_0(1+a\,\sin(\omega t+\varphi)]N(t) \ . \tag{2.20}$$

The difference to the summand model lies in the implication of the modified decay constant $\lambda(t)$ in the ansatz. Solving leads to a new solution, which already includes the idea of the new model. In contrast, the properties of the preceding model have just been tinkered into the solution of the unmodified standard model. Hence, the present model is somewhat more profound. We will also see that its equations resemble those of the standard description more convincingly. The solution of (2.20) is

$$N(t) = \mathrm{e}^{-\lambda_0\int(1+a\sin(\omega t+\varphi))\,\mathrm{d}t} = C\,\mathrm{e}^{-\lambda_0 t}\,\mathrm{e}^{\lambda_0\frac{a}{\omega}\cos(\omega t+\varphi)} \ .$$

[10] Of course, a definition like $\lambda(t) = \lambda_0 + a\sin(\omega t+\varphi)$, where $a \ll \lambda_0$ and $[a] = 1/s$ would also be possible. However, the chosen approach, yields a deeper intertwining of λ_0 and a and makes the latter a dimensionless, relational quantity.

To satisfy the boundary condition $N(t=0)=N_0$, we find

$$N(t=0) = C\,\mathrm{e}^{\lambda_0 \frac{a}{\omega}\cos(\varphi)} = N_0$$

and thus, solving for the integration constant C,

$$C = N_0\,\mathrm{e}^{-\lambda_0 \frac{a}{\omega}\cos(\varphi)} =: N_0(\varphi)\ .$$

This gives

$$\begin{aligned}(\mathrm{I}) \quad \rightarrow \quad N(t) &= N_0\,\mathrm{e}^{-\lambda_0 \frac{a}{\omega}\cos(\varphi)}\,\mathrm{e}^{-\lambda_0 t}\,\mathrm{e}^{\lambda_0 \frac{a}{\omega}\cos(\omega t+\varphi)} \\ &= N_0(\varphi)\,\mathrm{e}^{-\lambda_0 t}\,\mathrm{e}^{\lambda_0 \frac{a}{\omega}\cos(\omega t+\varphi)}\ . \end{aligned} \tag{2.21}$$

Again we calculate the different functions to describe the introduced model:

$$(\mathrm{II}) \quad \rightarrow \quad N_\mathrm{d}(t) = N_0\left[1-\mathrm{e}^{-\lambda_0 \frac{a}{\omega}\cos(\varphi)}\,\mathrm{e}^{-\lambda_0 t}\,\mathrm{e}^{\lambda_0 \frac{a}{\omega}\cos(\omega t+\varphi)}\right] \tag{2.22}$$

$$(\mathrm{III}) \quad \rightarrow \quad n(t) = \lambda_0\,[1+a\sin(\omega t+\varphi)]\cdot N_0(\varphi)\,\mathrm{e}^{-\lambda_0 t}\,\mathrm{e}^{\lambda_0 \frac{a}{\omega}\cos(\omega t+\varphi)} = \lambda(t)\,N(t) \tag{2.23}$$

$$\begin{aligned}(\mathrm{IVb}) \quad \rightarrow \quad \Delta N(\Delta t; t_0) &= N(t_0) - N(t_0+\Delta t) \\ &= N_0(\varphi)\,\mathrm{e}^{-\lambda_0 t_0}\left[\mathrm{e}^{\lambda_0 \frac{a}{\omega}\cos(\omega t_0+\varphi)} - \mathrm{e}^{-\lambda_0 \Delta t}\,\mathrm{e}^{\lambda_0 \frac{a}{\omega}\cos(\omega(t_0+\Delta t)+\varphi)}\right]\end{aligned} \tag{2.24}$$

The amplitude of the sine has to be very small, $\lambda_0\,a/\omega \ll 1$, otherwise the effect would have been described already – which makes this relation a promising candidate for $\varepsilon(\lambda_0, a, \omega)$.

Simplification

By the use of the same arguments as before, we can set the phase argument $\varphi := 0$. The term $\exp(-\lambda_0\,a/\omega\cos(0)) \approx 1$ allows for the approximation $N_0(\varphi) \approx N_0$. This simplifies the equations as follows:

$$(\mathrm{I}) \quad \rightarrow \quad N(t) = N_0\,\mathrm{e}^{-\lambda_0 t}\,\mathrm{e}^{\lambda_0 \frac{a}{\omega}\cos(\omega t)} \tag{2.25}$$

$$(\mathrm{II}) \quad \rightarrow \quad N_\mathrm{d}(t) = N_0\left[1-\mathrm{e}^{-\lambda_0 t}\,\mathrm{e}^{\lambda_0 \frac{a}{\omega}\cos(\omega t)}\right] \tag{2.26}$$

$$(\mathrm{III}) \quad \rightarrow \quad n(t) = \lambda_0\,[1+a\sin(\omega t)]\cdot N_0\,\mathrm{e}^{-\lambda_0 t}\,\mathrm{e}^{\lambda_0 \frac{a}{\omega}\cos(\omega t)} = \lambda(t)\,N(t) \tag{2.27}$$

$$\begin{aligned}(\mathrm{IVb}) \quad \rightarrow \quad \Delta N(\Delta t; t_0) &= N(t_0) - N(t_0+\Delta t) \\ &= N_0\,\mathrm{e}^{-\lambda_0 t_0}\left[\mathrm{e}^{\lambda_0 \frac{a}{\omega}\cos(\omega t_0)} - \mathrm{e}^{-\lambda_0 \Delta t}\,\mathrm{e}^{\lambda_0 \frac{a}{\omega}\cos(\omega(t_0+\Delta t))}\right]\end{aligned} \tag{2.28}$$

The equations look very similar to the equations from the standard decay; even more so than those of the sinusoidal summand.

Effective amplitude ε of the oscillation

We already found $\varepsilon := \lambda_0\,a/\omega$ as one possible candidate for the effective amplitude of the postulated oscillation, which should be visible in the measured/simulated data, eventually. Considering the mathematical structure of equation (2.21), this assumption seems intuitive.

Let us examine the general term $f(t;\xi) = \exp(\xi\cos(\omega t))$. A quick plot will show that $f(t;\xi)$ takes the characteristics of a cosine for fairly large values of ξ already. A Taylor expansion for small values $\xi \approx 0$, namely

$$f(t;\xi)\Big|_{\xi_0\approx 0} = \sum_n \frac{1}{n!}\frac{\partial^n}{\partial \xi^n} f(t;\xi_0)(\xi-\xi_0)^n = 1 + \xi\cos(\omega t) + \mathcal{O}(\xi^2)\ , \tag{2.29}$$

where $\partial^n/\partial\xi^n f(t;\xi) = \xi^n \cos^n(\omega t)$, performed for the corresponding term in (2.21) yields

$$N(t) \approx N_0(\varphi)\,\mathrm{e}^{-\lambda_0 t}\left[1 + \lambda_0 \frac{a}{\omega}\cos(\omega t + \varphi)\right] \ ,$$

which is exactly the result (2.11) from the sinusoidal summand model in the case of $\eta = 0$.[11] This is consistent with the assumption of disintegration being both induced and inhibited in the current model and comparison of these two equations indeed surmounts the relationship

$$\varepsilon = \lambda_0 \frac{a}{\omega} \ll 1 \ . \tag{2.30}$$

This finding also justifies the general ansatz (2.19), since (2.30) links all important quantities defining the model; namely the standard decay constant λ_0, the dimensionless factor a, the effect's frequency ω and the observable effective amplitude ε. This relation allows to estimate the value for a, which otherwise remains inaccessible to the experimenter. And still $\varepsilon \ll 1$ holds, because if not, I would not need to write any of this.

2.2.5. Inducing and/or inhibiting effect in sinusoidal models

The second model (sinusoidal decay constant) assumes the effect in question to *induce* and *inhibit* decay equally, since $\lambda(t) \propto \sin(\omega t)$. The first model (sinusoidal summand) can describe an inducing, an inhibiting or *alternating* influence on the disintegration rate via the parameter η. For $\eta = 0$ the two models coincide in the first approximation.

To describe inducing influence exclusively also in the second model, one would need to shift the sine

$$\frac{\sin(\omega t) + 1}{2}$$

instead of a mere $\sin(\omega t)$. Analogously, one could state

$$\frac{\sin(\omega t) - 1}{2}$$

to examine inhibiting influence only.[12]

Generally, we could also postulate a mixture of the two, in that we assume the effect in question to induce decay at the maximum amplitude of $\theta^+ > 0$ and at the same time act inhibiting at the minimum $\theta^- < 0$. We seek to find the description of a sine shaped influence satisfying the mentioned bounding conditions (extrema).

[11] This finding was the motivation to choose a cosine function in the ansatz of the former model.

[12] Mind that simply taking the absolute value $\pm|\sin(\omega t)|$ or writing $\pm\sin^2(\omega t)$ would alter the shape of the influence!

The ansatz

$$\lambda(t) \propto \frac{\sin(\omega t) + \tilde{\theta}}{C}$$

and the bounding conditions

$$\text{Maximum:} \quad \frac{1+\tilde{\theta}}{C} = \theta^+ \ , \ \theta^+ > 0$$
$$\text{Minimum:} \quad \frac{-1+\tilde{\theta}}{C} = \theta^- \ , \ \theta^- < 0$$

lead to

$$\tilde{\theta} = -\frac{1+\theta}{1-\theta} \ , \ \theta = \frac{\theta^+}{\theta^-} \quad \text{and}$$
$$C = -\frac{2}{\theta^- - \theta^+} \ .$$

A phase shift of $\varphi = -\arcsin\tilde{\theta}$ will enforce $\lambda(0) = 0$. See figure 2.

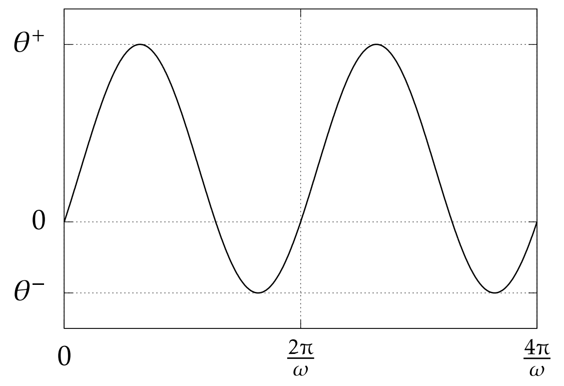


Figure 2 This figure shows a shifted sine curve for given θ^+, θ^- and ω.

2.2.6. Discrete Sun influence (solar activities)

Another idea is to expect irregular sun activities, like solar flares, to have an influence on decay processes as in [4]. In comparison with known dates of solar flares, one could try to find correlations events in the data.

An analytical model could use Dirac functions, Gaussians or typical window functions to describe a time-limited influence. However, Dirac functions may not properly introduce possible "precursor" signals before the actual solar event into to the model as described in [4].

2.2.7. Rectangular (neutrino) influence

Given that neutrinos of the reactor at the Atominstitut have an effect on decaying nuclei in that they either induce or inhibit disintegration, the measured spectrum has to show a rectangular shaped variation. The periodicity of the variations would coincide with the reactor runtime, documented in the reactor journal. Thus, the fundamental mathematical shape of the adapted formula, analogously to (2.10), would read

$$N(t) = N_0 \, e^{-\lambda t} \left[1 + \eta\varepsilon \Big(\Theta(t - t_{\text{on}}) \Theta(-(t - t_{\text{off}})) \Big) \right] \ ,$$

where

$$\eta = \begin{cases} +1 & \text{inhibiting} \\ -1 & \text{inducing} \end{cases}$$

and Θ is the Heaviside theta function. The time dependence of this effect lies in the time dependent reactor activity, actually – thus, it is, similar to the sinusoidal distant dependent effects, not time dependent by nature.

Since the reactor is generally scheduled to run from Monday to Friday, 8am to 3:45pm, the correct formula would look very cluttered. Instead, the calculations will be performed with general t_{on} and t_{off} to represent any rectangular influence.

We find

$$\text{(I)} \quad \rightarrow \quad N(t) = N_0\,\mathrm{e}^{-\lambda t}\Big[1+\eta\varepsilon\Big(\Theta(t-t_{\text{on}})\Theta(-(t-t_{\text{off}}))\Big)\Big] \tag{2.31}$$

$$\text{(II)} \quad \rightarrow \quad N_{\text{d}}(t) = N_0\Big[1-\mathrm{e}^{-\lambda t}\Big[1+\eta\varepsilon\Big(\,\Theta(t-t_{\text{on}})\,\Theta(-(t-t_{\text{off}}))\,\Big)\Big]\Big] \tag{2.32}$$

$$\text{(III)} \quad \rightarrow \quad n(t) = \lambda N(t) - \eta\varepsilon N_0\,\mathrm{e}^{-\lambda t}\Big(\delta(t-t_{\text{on}})-\delta(t-t_{\text{off}})\Big) \tag{2.33}$$

$$\begin{aligned}\text{(IV)} \quad \rightarrow \quad \Delta N(\Delta t;t_0) = N_0\,\mathrm{e}^{-\lambda t}\Bigg[&\lambda\Big(\Delta t+\eta\varepsilon\int_{t_0}^{t_0+\Delta t}\Theta(t-t_{\text{on}})\Theta(-(t-t_{\text{off}}))\,\mathrm{d}t\Big)\\ &-\eta\varepsilon\int_{t_0}^{t_0+\Delta t}\Big(\delta(t-t_{\text{on}})-\delta(t-t_{\text{off}})\Big)\,\mathrm{d}t\Bigg]\end{aligned} \tag{2.34}$$

The Dirac delta functions in (2.33) become 1 each if t_{on} or t_{off} respectively lie within $[t_0, t_0+\Delta t]$, which corresponds to switching the effect on or off. The first integral in (2.34) becomes 1 for the intersection of $[t_0, t_0+\Delta t]$ and $[t_{\text{on}}, t_{\text{off}}]$, whereas the second integral, as above, becomes 1 for each Dirac function lying in $[t_0, t_0+\Delta t]$.

The relationships

$$\frac{\mathrm{d}}{\mathrm{d}t}\Theta(at-b) = a\,\delta(at-b)$$
$$\delta(t) = \delta(-t)$$

were used.

Mind that the use of Heaviside theta and Dirac delta functions involve an idealization of the real process in which the reactor start-up and shut-down times may even surpass Δt.

Similar to the first sine model, this model simply modifies the solution of standard decay. An ansatz introducing a reactor run time dependent decay "constant", much like the second sine model, could be formulated as well.

2.3. Mathematical discussion of the measurement process

This section will briefly explain the principle of measurement and which formulas will be used to compute data in the simulation described in section 3 Simulation (p. 47). Furthermore, the statistical treatment of radioactive decay processes will be presented and the explained issues will be put into the light of a real measurement and concluded for practical use.

2.3.1. Principle of measurement

In a real measurement situation a detector cannot measure for an infinitesimal time interval $\mathrm{d}t$. In order to obtain usable results, the detector needs to measure throughout a certain time interval

$$[t_i, t_i+\Delta t] =: [t_i, t_{i+1}] \quad \text{for} \quad \Delta t = \text{const} \quad \text{and} \quad [\Delta t] = 1\,\text{s} \tag{2.35}$$

and sum up all recorded events (photons of decayed nuclei hitting the detector) to output a tuple of values

$$\Big(t_{i+1}\,,\ \Delta N(t_i;\Delta t),\ \sqrt{\Delta N(t_i;\Delta t)}\Big)\ , \tag{2.36}$$

which holds the time t_{i+1} after the measurement interval Δt, the number of events registered by the detector during $[t_i, t_{i+1}]$ and the statistical uncertainty of that number.

This principle of measurement requires a discretization of time:

$$t \quad \rightarrow \quad t_{i+1} = t_i + \Delta t \quad \text{or} \quad t_i = t_0 + i\Delta t\ , \quad i \in \mathbb{N}_0\ . \tag{2.37}$$

Let us, at this point, refer to equation (IV)

$$\Delta N(\Delta t; t_0) = \int\limits_{t_0}^{t_0+\Delta t} n(t)\,\mathrm{d}t = N(t_0) - N(t_0 + \Delta t)$$

to get rid of confusion about the nomenclature. The formula describes the amount of decayed nuclei in dependence of the measurement time Δt for a given starting time t_0; it is therefore natural to write $\Delta N(\Delta t; t_0)$. We chose the notation $\Delta N(t_i; \Delta t)$ for the context of measuring, since it emphasizes the role of t_i as the variable (an arbitrary time during the measurement) instead of the constant Δt. This makes (IV):

$$\Delta N(\Delta t; t_0) \quad \rightarrow \quad \Delta N(t_i; \Delta t) = \sum_{j=1}^{k} n(t_{i,j})\Delta t_{\mathrm{min}} = N(t_i) - N(t_i + \Delta t) = N(t_i) - N(t_{i+1}) \ , \tag{2.38}$$

where the integral became a sum due to the discretization. For Δt we find in this context

$$\Delta t = k \cdot \Delta t_{\mathrm{min}} \quad \text{where} \quad t_{i,j} = t_i + j\Delta t_{\mathrm{min}} \quad \text{and} \quad k, j \in \mathbb{N} \ .$$

In this context, Δt_{min} resembles the smallest step size possible (or reasonable) – the "infinitesimal step in discrete time", so to say. This seemingly strange definition is necessary when approaching computer simulations, where we could define a small yet reasonable $\Delta t_{\mathrm{min}} := 1\,\mathrm{s}$, for instance. A single measurement therefore lasts k steps of length Δt_{min}, which is exactly the upper limit in the sum.

The whole measurement is characterized by the measurement duration $\mathscr{T}([\mathscr{T}] = 1\,\mathrm{s})$, which is ideally a whole-number multiple of Δt.

Note that Δt must never be of the same magnitude as the cycle duration T but considerably smaller to not obscure the periodicity. The Nyquist theorem states

$$\Delta t < T/2 \ , \tag{2.39}$$

however, in reality, the integration time has to (and will) be way shorter.[13]

In (2.38), because of the discretization, we found two expressions for $\Delta N(t_i; \Delta t)$. The former integration – now the sum – over the activity

$$\Delta N(t_i; \Delta t) = \sum_{j=1}^{k} n(t_{i,j})\Delta t_{\mathrm{min}} \tag{2.40}$$

and, on the other hand, the use of the stem function

$$\Delta N(t_i; \Delta t) = N(t_i) - N(t_i + \Delta t) \ . \tag{2.41}$$

The difference between methods (2.40) and (2.41) may not be immediately clear, but will be pointed out in section 3.3.3 Types of simulating (p. 53).

The actual setup of the experiment and a slightly more detailed description of how the detector works is pictured in section 5 Experimental setup (p. 103).

[13] Too large values for Δt will very likely render an analysis futile as they can obscure the periodicity of a signal.

2.3.2. Basic concepts of statistics

This section briefly lists well-known formulas for the basic statistical handling of values with uncertainty. It mainly relies on [18] and supplementary internet research. Section 2.3.4 Statistical description of radioactive decay (p. 30) covers the probability distributions used to describe radioactive decay.

The mean value $\overline{x}$ of N acquired values x_i is

$$\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \ .$$

For an infinite number of measurements $N \to \infty$ the mean value approaches the expectation value $\mathrm{E}[x]$ of the underlying probability distribution. The combined squared deviation from the mean value gives the (empirical) standard deviation σ_x according to

$$\sigma_x = \sqrt{\frac{1}{N-1}\sum_i (x_i - \overline{x})^2},$$

where the prefactor $1/N$ instead of $1/(N-1)$ is used if the population is known. The variance σ_x^2 is the standard deviation squared. Mind that mean value and standard deviation are of the same dimension.

The variance $\mathrm{Var}[x]$ of a stochastic variable x originating from a probability distribution is

$$\mathrm{Var}[x] = \mathrm{E}\left[x^2\right] - \mathrm{E}[x]^2$$

and we also find for the standard deviation

$$\mathrm{Std}[x] = \sqrt{\mathrm{Var}[x]} \ .$$

Mind that in a real measurement situation the experimenter always deals with $\overline{x}$ and σ_x!

Under linear transformations we find for the expectation value and the variance

$$\mathrm{E}[c \cdot x + d] = c \cdot \mathrm{E}[x] + d \quad \text{and} \quad \mathrm{Var}[c \cdot x + d] = |c|^2 \cdot \mathrm{Var}[x] \ . \tag{2.42}$$

In many cases one needs information about expectation/mean value and the standard deviation while the data is still recorded or one needs to avoid buffering large amounts of data. Constantly updated values $\sum_{i=1}^{N} x_i$ and $\sum_{i=1}^{N} x_i^2$ allow for a continuous calculation of the mean value

$$\overline{x}\Big|_N = \frac{1}{N}\sum_{i=1}^{N} x_i$$

and the empirical standard deviation[14] according to

$$\sigma_x\Big|_N = \sqrt{\frac{1}{N-1}\left[\left(\sum_{i=1}^{N} x_i^2\right) - \frac{1}{N}\left(\sum_{i=1}^{N} x_i\right)^2\right]} \ . \tag{2.43}$$

[14] This relation was taken from [19] and checked by personal calculation.

Given that $y = y(x_i)$ is a function of multiple values x_i, all coming with an uncertainty σ_i, one needs a way to treat the combination of said uncertainties. The Gaussian uncertainty propagation is an essential tool for the statistical treatment of such a function y

$$\sigma_y = \sqrt{\sum_i \left(\frac{\partial y}{\partial x_i}\right)^2 \sigma_i^2} \ , \tag{2.44}$$

where the uncertainties σ_i are statistically independent.

Generally, if one wishes to combine multiple values x_i but weigh their influence individually, one computes the weighted average

$$\overline{x} = \frac{\sum_i x_i g_i}{\sum_i g_i} \ , \tag{2.45}$$

where the g_i are the individual weights. The respective standard deviation is calculated via

$$\sigma_x = \sqrt{\frac{\sum_i g_i \sigma_i^2}{(\sum_i g_i) - 1}} \ ,$$

where $\sigma_i = \overline{x} - x_i$.

In the case of a stochastic variable with given individual uncertainties σ_i one defines

$$g_i = \frac{1}{\sigma_i^2} \quad \text{and} \quad \sigma_x = \frac{1}{\sqrt{\sum_i g_i}} = \frac{1}{\sqrt{\sum_i \frac{1}{\sigma_i^2}}} \ . \tag{2.46}$$

2.3.3. Computing the initial number of nuclei from measurement data

The initial number of nuclei $N(t = t_0) = N_0$ is not an absolute value, because, in fact, the reference time t_0 is arbitrary. In case of a measurement situation one would naturally choose it to be at the beginning of the measurement process and define $t_0 := 0$.

"Measuring" means sampling the graph of $\Delta N(t_i; \Delta t)$ at discrete times t_i. According to (2.37) and $t_0 := 0$, we find $t_i = i\Delta t$. Hence, we can plug into (2.41) and get

$$\Delta N(t_i; \Delta t) = N(t_i) - N(t_i + \Delta t) = N_0 \, \mathrm{e}^{-\lambda t_i} \left(1 - \mathrm{e}^{-\lambda \Delta t}\right) \ .$$

This expression can be rearranged and we find for the initial number of nuclei

$$N_0 = \frac{\Delta N(t_i; \Delta t)}{(1 - \mathrm{e}^{-\lambda \Delta t})} \mathrm{e}^{\lambda t_i} = \frac{\Delta N(t_i; \Delta t)}{(1 - \mathrm{e}^{-\lambda \Delta t})} \mathrm{e}^{\lambda i \Delta t} \ . \tag{2.47}$$

If the values $\Delta N(t_i; \Delta t)$ (short-hand notation ΔN_i for now) are drawn from, for instance, a data file, it would be plausible to use all available data points to increase accuracy and calculate an average value $\overline{N_0}$ for the initial number of nuclei from all resulting $N_{0,i}$.

Let $B_i := (1 - \mathrm{e}^{-\lambda \Delta t}) \mathrm{e}^{-\lambda i \Delta t}$ and thus $N_{0,i} = \Delta N_i / B_i$, then using (2.42), $\sigma_{\Delta N_i} = \sqrt{\Delta N_i}$ and (2.46) will provide

$$\sigma_{N_{0,i}}^2 = \frac{\Delta N_i}{B_i^2} \quad \text{and} \quad g_i = \frac{1}{\sigma_{N_{0,i}}^2} = \frac{B_i^2}{\Delta N_i} \ .$$

Plugging into (2.45) will provide

$$\overline{N_0} = \frac{\sum_i N_{0,i} \, g_i}{\sum_i g_i} = \frac{\sum_i N_{0,i} \, g_i}{\sum_i g_i} = \frac{\sum_i \frac{\Delta N_i}{B_i} \frac{B_i^2}{\Delta N_i}}{\sum_i \frac{B_i^2}{\Delta N_i}} = \frac{\sum_i B_i}{\sum_i \frac{B_i^2}{\Delta N_i}} \ . \tag{2.48}$$

For the standard deviation we obtain

$$\sigma_{\overline{N_0}} = \frac{1}{\sqrt{\sum_i g_i}} = \frac{1}{\sqrt{\sum_i \frac{B_i^2}{\Delta N_i}}} \tag{2.49}$$

by using (2.46).[15] [16]

2.3.4. Statistical description of radioactive decay

If not stated otherwise, the contents of this section rely on mathematical definitions and connections given by the respective Wikipedia articles and on personal communication with Erwin Jericha and Rudolf Frühwirth.

Let us recall the empirical law of radioactive decay (2.3)

$$N(t) = N_0 \, \mathrm{e}^{-\lambda t}$$

and the definition of the activity (2.7)

$$A(t) := -\frac{\mathrm{d}N(t)}{\mathrm{d}t} := n(t) = \lambda N_0 \, \mathrm{e}^{-\lambda t} = A_0 \, \mathrm{e}^{-\lambda t} \quad .$$

The law of decay can also be derived by simple statistical argumentation and without knowledge about the underlying physical mechanisms.

Let us assume (a purely statistical assumption about this stochastic, i.e. random, process) that the probability of a nucleus to decay during a time interval δ is constant. Given that δ is small enough, the probability of a decay during that time is $\lambda\delta$, where λ (the decay constant) is a constant rate specific to the type of nucleus and regardless of the time it has existed. The probability for a nucleus *not* to decay, $1 - \lambda\delta$, during the time $t = k\delta$ is then $(1 - \lambda\delta)^k$. For very small $\delta \to 0$ we obtain

$$\lim_{\delta\to 0}(1-\lambda\delta)^k = \lim_{\delta\to 0}(1-\lambda\delta)^{t/\delta} \equiv \lim_{\delta\to 0} \mathrm{e}^{\ln(1-\lambda\delta)^{t/\delta}} = \lim_{\delta\to 0} \mathrm{e}^{t\ln(1-\lambda\delta)/\delta} \quad .$$

Taking the limit by applying l'Hôspital's rule yields

$$\lim_{\delta\to 0} \exp\left(\frac{\frac{\mathrm{d}}{\mathrm{d}\delta}\ln(1-\lambda\delta)}{\frac{\mathrm{d}}{\mathrm{d}\delta}\delta} t \right) = \lim_{\delta\to 0} \exp\left(\frac{-\lambda}{1-\lambda\delta} t \right) = \mathrm{e}^{-\lambda t} \quad ,$$

which is the probability of a nucleus not to decay. Starting from a given number N_0 of nuclei, the number of untransformed nuclei is in fact given by $N(t) = N_0 \exp(-\lambda t)$. (q.e.d.) [20]

The observation of decay events is carried out via counting processes (registration of decay products by a detector), thus we are more interested in a probability distribution of the expected number of decay events in a certain time, than in their temporal distribution.

The decay of a single nucleus is a simple Bernoulli process with probability $p = 1 - \mathrm{e}^{-\lambda t}$ as we just calculated; p gives the probability of the nucleus to decay within time t. For a number of nuclei N, the same situation can be described by a binomial distribution $\mathscr{B}(N,p)$, which factors in that the events are discrete and independent from each other, that all nuclei have the

[15] While testing it seemed that this method would yield results too low such that the standard deviation was too small to include the true value. This can happen in case of non-standard decay – turning non-standard physics off yielded results well in accordance with the true value for N_0 and of great accuracy!

[16] For reviewing the actual code in the simulation framework it helps to know that $B_i \equiv$ `factor`, $\sum_i B_i \equiv$ `facSum` and $\sum_i B_i^2/\Delta N_i \equiv$ `fac2sum`.

same (constant) decay probability λt and that it is unknown which of the nuclei decayed. The probability density function (PDF) reads

$$\mathscr{B}(x \mid N, p) = \binom{N}{x} p^x (1-p)^{N-x} \quad \text{with} \quad \binom{N}{x} = \frac{N!}{x!(N-x)!} \quad ,$$

where $p = 1 - \mathrm{e}^{-\lambda t}$ is the same as above and x denotes the number of registered events. The distribution therefore provides the probability of registering x events for given N and p. The expectation value and variance read as follows (the standard deviation is $\mathrm{Std}[x] = \sqrt{\mathrm{Var}[x]}$):

$$\mathrm{E}[x] = Np \qquad \mathrm{Var}[x] = Np(1-p)$$

The correct description of the statistics of radioactive decay follows a binomial distribution, however, it can be approximated by a Poisson distribution $\mathscr{P}(x \mid \kappa) \approx \mathscr{B}(N, p)$, if the observation time t is small compared to the life time, $\lambda t = 1/t_\lambda \cdot t \ll 1$, and if N is large compared to the number of detected events x, $x \ll N$, which means that the sample has a low activity. In terms of the symbols above, the same conditions can be written as $N \to \infty$ and $p \to 0$. The approximation is valid, if the limits are approached such that $Np = \mathrm{const} := \kappa$. The PDF of the Poisson distribution (a distribution of discrete events in continuous time) reads

$$\mathscr{P}(x \mid \kappa) = \frac{(Np)^x}{x!} \mathrm{e}^{-Np} = \frac{\kappa^x}{x!} \mathrm{e}^{-\kappa} \quad , \tag{2.50}$$

where yet again x denotes the number of detected events. One finds:

$$\mathrm{E}[x] = \kappa = Np \qquad \mathrm{Var}[x] = \kappa = Np$$

Since N is usually in the magnitude of 10^{10} and higher, the Poisson distribution is a good description of the phenomenon.

It is important to recall that all introduced variables $p = p(t; \lambda)$, $N = N(t)$ and $A = A(t)$ are actually time dependent and that the presented Poisson approximation is only valid, if these quantities remain quasi-constant during the observation time t, which is true under the aforementioned conditions. The presented distributions provide the probability of detecting x events within that time t, but are not explicitly time dependent. In reality, at each point in time, the situation is described by a new distribution, because the parameters changed, even if just ever so slightly. This is illustrated in figure 3 for the Gaussian distribution which will be introduced in the next paragraph.

The well known Gaussian distribution[17] $\mathscr{G}(\mu, \sigma)$ can be used for further simplification and is described by two parameters: mean value μ and variance σ^2 (σ is the standard deviation). The general PDF reads

$$\mathscr{G}(x \mid \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \mathrm{e}^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad .$$

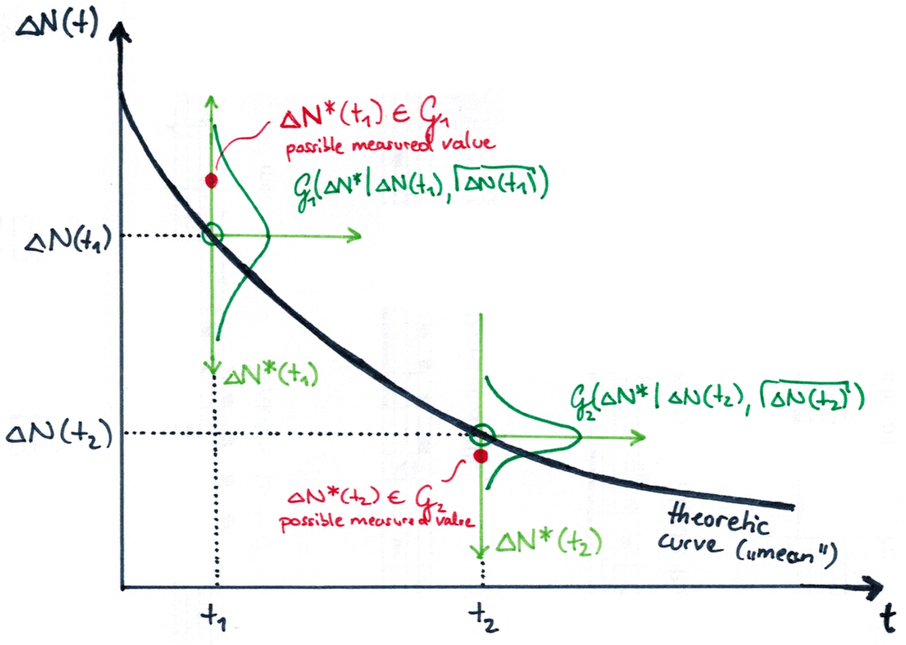
Already for values $\kappa \approx 35$ [18], the Poisson distribution $\mathscr{P}(\kappa)$ can be approximated by a Gaussian distribution according to

$$\mathscr{P}(x \mid \kappa) \approx \frac{1}{\sqrt{2\pi\kappa}} \mathrm{e}^{-\frac{(x-\kappa)^2}{2\kappa}} = \mathscr{G}(x \mid \kappa, \sqrt{\kappa})$$

and comparison yields

$$\mu = \sigma^2 = \kappa \quad \text{and thus} \quad \sigma = \sqrt{\kappa} \quad .$$

[17] To avoid confusion with the number of nuclei, we refrain from using the usual symbol $\mathcal{N}$ (“normal distribution”).



$\Delta N(t)$
$\Delta N^*(t_1) \in \mathcal{G}_1$
possible measured value
$\mathcal{G}_1(\Delta N^* | \Delta N(t_1), \sqrt{\Delta N(t_1)})$
$\Delta N(t_1)$
$\Delta N^*(t_1)$
$\Delta N(t_2)$
$\mathcal{G}_2(\Delta N^* | \Delta N(t_2), \sqrt{\Delta N(t_2)})$
$\Delta N^*(t_2) \in \mathcal{G}_2$
possible measured value
$\Delta N^*(t_2)$
theoretic curve („mean")
t_1
t_2
t

Figure 3 This sketch shows the principle of the connection between the law of decay, the probability distribution of the physical process of decay and the results of an experiment. The "theoretic" line (black) represents the findings of the law of decay (decayed nuclei in a time interval, in this case; we have refrained from including Δt into the nomenclature here). At each point in time t_i there is a probability distribution (green) with mean and standard deviation dictated by the theoretical expectation value $\Delta N(t_i)$. These factor in the physical (unpredictable) behavior of the decaying sample. In an actual simulation/experiment, the obtained value (red) at each measurement time t_i originates from its respective distribution.

One finds for the Gaussian distribution:

$$\mathrm{E}[x] = \mu = \kappa \ , \quad \mathrm{Var}[x] = \sigma^2 = \kappa \quad \text{and} \quad \mathrm{Std}[x] = \sqrt{\mathrm{Var}[x]} = \sigma = \sqrt{\kappa}$$

Mind that this distribution describes *continuous* events in continuous time despite the discrete nature of a collection of nuclei. Furthermore, note that also the empirical law of decay does not factor in their discreteness either.

When plugging into $\kappa = Np$, we obtain a term which is similar to (2.8)

$$\kappa = Np = N(1 - \mathrm{e}^{-\lambda t}) \mathrel{\hat{=}} \Delta N = \int_0^t A(t')\,\mathrm{d}t' = \overline{A}_t t \ , \tag{2.51}$$

where we used the mean value theorem to obtain a mean activity $\overline{A}_t$ for the time interval t.[18] Now the relations for the Gaussian distribution can be rewritten as

$$\mathcal{G}\left(x \mid \mu = \Delta N, \sigma = \sqrt{\Delta N}\right) = \frac{1}{\sqrt{2\pi\Delta N}} \mathrm{e}^{-\frac{(x-\Delta N)^2}{2\Delta N}} \tag{2.52}$$

and

$$\mathrm{E}[x] = \mu = \Delta N \ , \quad \mathrm{Var}[x] = \sigma^2 = \Delta N \quad \text{and} \quad \mathrm{Std}[x] = \sqrt{\mathrm{Var}[x]} = \sigma = \sqrt{\Delta N} \ . \tag{2.53}$$

[18] The gentle reader might argue that (2.51) rather resembles expression (2.5) for the number of decayed nuclei N_d than equation (2.8) (as claimed in the text), which denotes the number of decay events during a time interval Δt. This is true, but, for the sake of clarity, the present notation was chosen for the rest of this section, because ΔN really is the true quantity of interest. The integral limits in (2.51), which cause the potential confusion, are just due to an arbitrary shift of the reference time, which causes $\Delta t \to t$, in order to make the expression simpler.

This notation builds a bridge between the theoretical, statistical description and a quantity, which can be accessed in the experiment, namely the number of decayed nuclei ΔN.

The expression $\kappa = Np = \overline{A}_t t$ for the mean activity in (2.51) can also be used to rewrite the Poisson distribution (2.50):

$$\mathscr{P}(x \mid \kappa) = \frac{\kappa^x}{x!}\,\mathrm{e}^{-\kappa} = \frac{(\overline{A}_t t)^x}{x!}\,\mathrm{e}^{-\overline{A}_t t} = \mathscr{P}(x \mid \overline{A}_t t) \tag{2.54}$$

The PDF has now an explicit time dependence and we can use it to determine the (probability of the) time until x arrivals have happened.

When we want to find the probability P of having the k-th event in a time interval $[t, t+\delta]$, where δ is so small that the chance of having more than one event during that interval is negligible, then we need to find the respective PDF $f_k(t)$ such that

$$f_k(t)\,\delta = P(t \le k\text{-th event} \le t+\delta) \ , \tag{2.55}$$

which can be written as the product of the probabilities of having $k-1$ events during $[0,t]$ and one during $[t, t+\delta]$;

$$P(t \le k\text{-th event} \le t+\delta) = P(0 \le k-1 \text{ events} \le t) \cdot P(t \le \text{one event} \le t+\delta)$$

Using (2.54) and the interpretation of $\overline{A}_t$ as an average event rate/"intensity", we can write

$$P(0 \le k-1 \text{ events} \le t) = \mathscr{P}(x = k-1 \mid \overline{A}_t t) = \frac{(\overline{A}_t t)^{k-1}}{(k-1)!}\,\mathrm{e}^{-\overline{A}_t t}$$
$$P(t \le \text{one event} \le t+\delta) = \overline{A}_t \delta \ ,$$

which yields, after plugging into (2.55),

$$f_k(t)\,\delta = \frac{(\overline{A}_t t)^{k-1}}{(k-1)!}\,\mathrm{e}^{-\overline{A}_t t} \cdot \overline{A}_t \delta \ ,$$

which simplifies to

$$f_k(t) = \frac{\overline{A}_t^{\,k}\, t^{k-1}}{(k-1)!}\,\mathrm{e}^{-\overline{A}_t t} \ .$$

This is ($k \longrightarrow x$) the Erlang distribution $\mathscr{ERL}(\overline{A}_t, x)$ with PDF

$$\mathscr{ERL}(t \mid \overline{A}_t, x) = \frac{\overline{A}_t^{\,x}\, t^{x-1}}{(x-1)!}\,\mathrm{e}^{-\overline{A}_t t}$$

with

$$\mathrm{E}[t] = \frac{x}{\overline{A}_t} \quad \text{and} \quad \mathrm{Var}[t] = \frac{x}{\overline{A}_t^{\,2}}$$

and it describes the probability distribution of the time t until the x-th event at constant average rate $\overline{A}_t$. For $x = 1$, it turns into the exponential distribution $\mathscr{E}(\overline{A}_t)$

$$\mathscr{ERL}(t \mid \overline{A}_t, x = 1) = \frac{\overline{A}_t^{\,1}\, t^0}{(0)!}\,\mathrm{e}^{-\overline{A}_t t} = \overline{A}_t\,\mathrm{e}^{-\overline{A}_t t} = \mathscr{E}(t \mid \overline{A}_t)$$

with

$$\mathrm{E}[t] = \frac{1}{\overline{A}_t} \quad \text{and} \quad \mathrm{Var}[t] = \frac{1}{\overline{A}_t^{\,2}} \ ,$$

which was already implied in the equation for the activity $A(t) \propto \lambda\,\mathrm{e}^{-\lambda t}$.[19] Thus, the time until the first event and between two consecutive events (since the events are independent from each other) is distributed exponentially [21].

Let us summarize the findings of this section: The correct statistical description of the radioactive decay of a number of nuclei N is that of a (discrete) binomial distribution $\mathscr{B}(N,p)$. Since it will be always the decay products that will be detected (not the remaining nuclei), the probability parameter p of the distribution will be that of the description of decayed nuclei, thus $p = 1-\exp(-\lambda t)$. This probability follows from the empirical law of decay, but can also be derived from statistical considerations. Under the condition that the observation time is small compared to the life-time of the nuclei and that the number of nuclei N is large compared to the number of detected events x, which is the same as demanding low activity (or a quasi-constant average decay rate), the binomial distribution can be approximated by a Poisson distribution $\mathscr{B}(N,p) \approx \mathscr{P}(\kappa = Np)$ (Poisson approximation). In reality, the quantities $p = p(t;\lambda)$, $N = N(t)$ (and as a consequence any derived quantity such as $\Delta N = \Delta N(t)$ and $\overline{A}_t = \overline{A}_t(t)$) are *not* constant, which means that all distributions, whether they describe a time dependency or not, are, as a whole, time dependent and change at every moment in time. However, the statistical treatment as provided above is valid as long as these quantities change slow enough. The Poisson distribution can then, for a number κ sufficiently large, be approximated by a Gaussian distribution, which is very easy to handle. By the relation $Np = \Delta N = \overline{A}_t t$ – where ΔN denotes the number of detected decay products and $\overline{A}_t$ the average constant decay rate/activity during the observation time –, the distributions can be rewritten to either describe counting statistics or temporal aspects of the same process. In the most practical form, we find $\mathscr{P}(Np) \approx \mathscr{G}(\Delta N, \sqrt{\Delta N})$. The Gauss approximation set aside, the radioactive decay of a large sample is a discrete Poisson process, so the number of events is Poisson distributed and the time between two consecutive events is exponentially distributed.

Conclusion for the application

Generally, the law of decay provides the statistical *mean* value $N(t)$ of remaining nuclei after a given time t. This principle also holds true for any equation derived from it, such as, for instance, the quantity $\Delta N(t;\Delta)$, which is the number of decay events during a given time interval. For reasons of readability we will write $\Delta N(t)$ until the end of this section.

In the case of a real measurement (denoted by an asterisk), in which the observable is ΔN instead of N, the actual number of decayed nuclei $\Delta N^*(t)$ originates from the Poisson distribution $\mathscr{P}(N,p)$ of that time t. As already pointed out above, the state of the system is actually described by a new distribution at every step t along the time axis! Approximating the Poisson by a Gaussian distribution and using $Np = \Delta N$, as displayed in the preceding section, this can be expressed by

$$\Delta N \to \Delta N(t) \quad \text{and thus} \quad \mathscr{G}\left(\Delta N, \sqrt{\Delta N}\right) \to \mathscr{G}\left(\Delta N(t), \sqrt{\Delta N(t)}\right) \;,$$

since $p = p(t;\lambda)$ and $N = N(t)$ are changing with every decay process. Examination of a real measured or simulated signal would show a multitude of data points (values $\Delta N^*(t_i)$), distributed about the corresponding theoretical curve (or a fit), following the statistics of a Gaussian distribution. A measured or simulated data point is a random value from this very distribution:

$$\Delta N^*(t) \in \mathscr{G}\left(\Delta N(t), \sqrt{\Delta N(t)}\right)$$

You may want to go back to figure 3 at this point.

[19] Considering that the time until the decay of a single nucleus is exponentially distributed, the Erlang distribution (as the probability distribution which describes x decays) can be obtained by convolution of x exponential distributions of the same constant rate.

The nature of a Gaussian distribution demands that 68.3% of all "real" data points $N^*(t)$ (with their own uncertainties $\sqrt{N^*(t)}$) lie within the error bar of the theoretical value $\mu(t) = N(t)$ (error $\sigma(t) = \sqrt{N(t)}$).

Relative error

We define the difference $\Delta x = \tilde{x} - x$ between a measured (approximated) value x and the exact value $\tilde{x}$ (where $x \approx \tilde{x}$) as the absolute error. We obtain the relative error δ_x by dividing the absolute error by the exact value, thus

$$\delta_x = \frac{|\Delta x|}{\tilde{x}} \approx \frac{|\Delta x|}{x} \quad .$$

Considering the fact that the problem is described by a Gaussian distribution $\mathscr{G}(\mu, \sigma)$ and (2.53), we can set $x = \mu = \Delta N$ and $\Delta x = \sigma = \sqrt{\Delta N}$ and obtain

$$\delta_N = \frac{\sigma}{\mu} = \frac{\sqrt{\Delta N}}{\Delta N} = \frac{1}{\sqrt{\Delta N}} \quad . \tag{2.56}$$

As a result, the relative error decreases for larger ΔN.

If we demand the relative error to be $\delta_{\Delta N} \leq \alpha$, we can solve for ΔN

$$\Delta N \geq \frac{1}{\alpha^2} \tag{2.57}$$

to obtain the minimum number of detected nuclei ΔN (and thus, dependent from the measurement situation values for the initial number of nuclei N_0, integration time Δt etc.) to achieve a relative error of α.

2.3.5. Accuracy and resolution; problems

Two substantial quantities come along with a measurement: Measuring accuracy and resolution.

We expect to have to look for an effect of very low amplitude, thus we have to guarantee high measuring accuracy. Secondly, we need to achieve a time resolution sufficient to make sought effects visible in time units possibly down to hours (see section 4.1.3 Magnitudes and values of introduced variables; exemplary case (p. 75)).[20]

Short things short: We demand maximum accuracy at maximum resolution. Unfortunately, they depend inversely on each other.

According to (2.56), the relative error decreases for an increasing number of registered decayed nuclei ΔN. Three main parameters have an impact on the magnitude of the signal: Measuring time interval Δt, initial number of nuclei N_0 and the decay constant λ; the last two being inherent with the respective sample and best described by its activity as defined in (2.7).

Considering $\Delta N(t; \Delta t) = N_0 \exp(-\lambda t)[1 - \exp(-\lambda \Delta t)]$, which is (IV) for standard decay, we conclude: The number of counts, and thus accuracy, increases with increasing Δt and increasing N_0. The role of λ is a bit more complex; for large λ, the exponential decay is more rapid due to $\exp(-\lambda t)$, but at the same time the term $[1 - \exp(-\lambda \Delta t)]$ increases. The value of λ determines the time curve of ΔN – high values of λ will lead to high initial values of ΔN but a rapid decrease, whereas lower values for λ will yield a smaller but more steady ΔN over time.

To achieve maximum resolution, the time interval Δt of integrating events by the detector has to be kept *low*, according to e.g. (2.39), which already opposes one of the statements above.

[20] If one ignores the earth's contribution, much higher values for Δt can be chosen and resolution is not a major problem anymore.

The severity of this restriction depends heavily on the magnitude of T; for large values of T the choice of Δt is not too restricted.

Eventually, coming to a compromise is inevitable: The measuring time has to be long enough to ensure sufficient accuracy (given a limited number of samples with strictly determined quantities) and, at the same time, be short enough to guarantee appropriate resolution to allow the sought periodic fluctuations to become visible in the data.

Special care has to be taken in the case in which the setup heavily focuses on sample attributes to achieve desired accuracy and resolution! If the total measuring time $\mathscr{T}$ is very long, for instance, the sample must retain its properties in the course the whole measurement. A sample with a short half-life may not live long enough to achieve a sufficiently high count rate at the end of a long term measurement. Long half-lives will lead to a less pronounced exponential shape of the signal, which could facilitate later analysis.

In what fashion the parameters influence each other is briefly discussed in 4.1.2 A set of rules (p. 73).

2.3.6. The solid angle in a real measurement setup

Although it is not the purpose of this chapter to deal with the intricacies of a real measurement setup, there is one important aspect when considering such a setup and the particular detector in use.

While talking of ΔN or whatever description of the number of nuclei, events etc. we may choose, we must not ignore the fact that not *all* of the photons resulting from radioactive decay in the sample will hit the detector and be recorded, but only a part of them determined by the constant cross sectional area F of the detection unit and the distance d between the detection unit and the sample. The sample radiates into all directions isotropically so the fraction $\Omega(d)$ of decay products reaching the detector is the ratio of F to the surface area of the sphere $4\pi d^2$:

$$\Omega(d) = \frac{1}{4\pi}\frac{F}{d^2} \quad . \tag{2.58}$$

This is actually the two-dimensional solid angle S/r^2, where S is the surface segment of a sphere and r is its radius. Since the detection unit's diameter is small, we can approximate $S \approx F$. Furthermore, $r = d$ in terms of above.

A detector usually also has an energy dependent efficiency $\epsilon(E)$ which further reduces the yield of the detector.

As a consequence, ΔN and therefore the value N_0, which we use in our calculations, are but a fraction $f(E,d)$ of the *real* amount of nuclei $\mathcal{N}(t)$ and $\mathcal{N}_0$ remaining/decaying,

$$N(t) = \underbrace{\epsilon(E)\cdot\Omega(d)}_{=:f(E,d)}\cdot\mathcal{N}(t) = f(E,d)\cdot\mathcal{N}_0\,\mathrm{e}^{-\lambda t} \quad , \tag{2.59}$$

which may neither change anything about the validity of what has been displayed so far nor about the error handling[21], but reduces the magnitude of the effective count rate; a fact that was not yet considered in the discussion in section 2.3.5 Accuracy and resolution; problems (p. 35)!

[21] For a solid angle of 4π, we have N and $\sigma_N = \sqrt{N}$. For $4\pi/n$, we obtain $N_i = N/n$, where $i = 1,\ldots,n$ and thus $\sigma_{N_i} = \sqrt{N_i} = \sqrt{N/n}$. Summing up all "solid angle elements" yields $\sum_{i=1}^n N_i = \sum_{i=1}^n N/n = N/n \sum_{i=1}^n 1 = N/n\cdot n = N$. Using simple error propagation, the error is given by $\sigma_{\sum_i N_i} = \sqrt{\sum_{i=1}^n [\partial(\sum_{i=1}^n N_i)/\partial N_i \cdot \sigma_{N_i}]^2} = \sqrt{\sum_{i=1}^n [1\cdot\sigma_{N_i}]^2} = \sqrt{\sum_{i=1}^n \sqrt{N/n}^2} = \sqrt{\sum_{i=1}^n N/n} = \sqrt{N/n \sum_{i=1}^n 1} = \sqrt{N/n\cdot n} = \sqrt{N} \;\rightarrow\; \sigma_N = \sigma_{\sum_i N_i}$ q.e.d.

2.4. Mathematical analysis tools

This section will provide the mathematical/statistical means to, on the one hand, isolate possible traces of the postulated effects in available spectra, and to, on the other hand, estimate the magnitudes of all important parameters in order to obtain reasonable results.

The actual usage of the presented methods will be primarily covered in section 3.5 Implementation of the analysis classes (p. 66) in greater detail.

2.4.1. Autocorrelation

All models including a periodic term obviously generate functions for remaining, decaying and decayed nuclei which show the same periodicity as well and its frequency ω does not change. Such a periodicity in the change of count rates will also show up in the data! A powerful tool to find periodicity in data sets or signals[22] is autocorrelation, which is the cross-correlation of the signal with itself. Its definition reads as follows:

$$\Psi(\tau) = \frac{1}{\mathscr{T}} \int_0^{\mathscr{T}} N(t) N(t+\tau) \, \mathrm{d}t \ ,$$

where τ is the time shift or so-called lag, which describes at which "distance" the signal $N(t)$ is overlapping with itself. The time-discrete variant is

$$\Psi_{\hat{\tau}} = \sum_{i=1}^{M} N_i N_{i+\hat{\tau}} \ , \tag{2.60}$$

where $\tau = \hat{\tau}\Delta t$ and $\hat{\tau} \in \mathbb{N}_0$ and M equals the number of included data points.[23]

The self-similarity is obviously the highest, when the lag is $\tau = 0$

$$\max \Psi(\tau) = \Psi(0) \ ,$$

which is commonly used for normalization

$$\psi(\tau) := \frac{\Psi(\tau)}{\Psi(0)} \ , \tag{2.61}$$

since the autocorrelation function usually attains fairly high values.

The autocorrelation function is a function of the lag τ where $[\tau] = 1\,\mathrm{s}$ or $\hat{\tau}$ where $[\hat{\tau}] = 1$ respectively.

Periodicity

For our application, the following property of the autocorrelation function $\Psi(\tau)$ is crucial: If the signal $N(t) = N(t+nT)$ is periodical, then the autocorrelation as well is, $\Psi(\tau) = \Psi(\tau + nT)$, and both share the same periodicity T.

The periodical maxima $\Psi(\tau_k)$ of the autocorrelation function at the respective lags τ_k coincide with the periodicity T in that $T = \tau_{k+1} - \tau_k$ or $T = \tau_1$ for the first maximum.[24] In our specific case of an exponential decay modulated with a periodical interference, the maxima at τ_k will

[22] In the course of this explanation, $N(t)$ will be used to represent the signal, however, any other function can be used, of course.

[23] Furthermore, the symbol N is not even correct at this point, as it should rather be $\Delta N(t_i;\Delta t)$! For the sake of simplicity we will write N regardless and introduce a more precise notation when necessary.

[24] Note that this is *not* the maximum at lag $\tau = 0$!

only be local and shrink with increasing index k. This periodicity might even only be visible to the human eye.

For a time-discrete autocorrelation function as in (2.60), we obtain the periodicity T via

$$T = \underbrace{(\tau_{k+1} - \tau_k)}_{=:\Delta\tau} = \tau_1 = \hat{\tau}_1 \Delta t \ , \tag{2.62}$$

where Δt shall be seen as in (2.37) and $\hat{\tau}$ as in (2.60). We obtain the corresponding frequency by plugging into (2.9):

$$\omega = \frac{2\pi}{\Delta\tau} = \frac{2\pi}{\tau_1} = \frac{2\pi}{\hat{\tau}_1 \Delta t}$$

Subtleties in the application

Recall equation (2.60) in which M was the number of included data points, which can, in general, differ from the number of *available* data points M_{tot}! Under the premise of comparability, we can demand the number of included data points to remain constant

$$M = \text{const}$$

for every lag while computing the autocorrelation function, which inevitably causes at times the exclusion of large ranges of available data!

The maximum lag at the highest number of included data points M we can achieve is

$$\hat{\tau}_{\max} = \frac{M_{\text{tot}}}{2} \ , \tag{2.63}$$

because any lag $\tau > \tau_{\max}$ would surpass the number of available data points M_{tot}. This definition automatically implies

$$M = \frac{M_{\text{tot}}}{2} = \hat{\tau}_{\max} \tag{2.64}$$

as well.

A lag of

$$\tau_{\max} \overset{!}{>} T$$

is necessary in order to find at least one significant maximum in the autocorrelation function of a periodical signal with cycle duration T, which accounts for the problem mentioned above. Using $\tau = \hat{\tau}\Delta t$, this becomes

$$\hat{\tau}_{\max} > \frac{T}{\Delta t}$$

and plugging into (2.63) yields

$$\frac{T}{\Delta t} < \frac{M_{\text{tot}}}{2} \ .$$

The number of available data points M_{tot} for a given total measurement duration $\mathscr{T}$ and integration time Δt can be calculated via

$$M_{\text{tot}} = \frac{\mathscr{T}}{\Delta t} \ .$$

Combining the last two equations, the demanded total measurement duration is

$$\mathscr{T} > 2T \ . \tag{2.65}$$

Since the phase of the effect is not known upon start of measurement, even

$$\mathscr{T} \approx 4T$$

might be indicated in order to assure that the periodicity of the autocorrelation function is very clearly visible (even more so because of the exponential decay)![25]

At this point it is good to remember (2.39) $\Delta t \ll T/2$ from the previous chapter to prevent the periodic fluctuation from becoming obscured, although it is not a matter of autocorrelation but only the fault of the experimenter in that he chose an improper Δt and stripped the signal from any oscillation.

Direction of the lag

The definition of the autocorrelation can be slightly altered in that the lag is pointing "backwards", namely (in comparison to (2.60))

$$\begin{aligned} \Psi_{\hat{\tau}}^{-} &= \sum_{i=1}^{M} N_{M+1-i}\, N_{(M+1-i)-\hat{\tau}} \\ &\hat{=} \sum_{i=M}^{1} N_i\, N_{i-\hat{\tau}} \ , \end{aligned} \tag{2.66}$$

where the first line describes in a mathematically correct manner what the second shows intuitively: A decrementing sum and a lag reaching "backwards" in that the M included data points will be summed up, starting with the smaller values, to avoid numerical precision problems in the sum.[26]

Errors in autocorrelation

The calculation of the error in the autocorrelation is not trivial. We would need to compute the propagation of the uncertainty of the sum of products of two values at a time, both with a given uncertainty. Since the autocorrelation overlays the input signal with itself, the "two" signals are not independent of each other. Thus, one would need to consider correlation terms in the error propagation. The variables in such an analysis would be the N_i, though, which makes the procedure rather complicated.

Assuming that there are no correlation effects to be considered and using "simple" error propagation, we can estimate a resulting uncertainty for the function values of the autocorrelation.

According to (2.60), we can calculate the autocorrelation function value $\Psi_{\hat{\tau}}$ for a given lag $\hat{\tau}$ by

$$\Psi_{\hat{\tau}} = \sum_i N_i N_{i+\hat{\tau}} := \sum_i f_i(\hat{\tau}) \ ,$$

where every $N_k = N_k^* \pm \sigma_k$ is given with a measured value N_k^* and uncertainty σ_k and $f_i(\hat{\tau}) = f_i^* \pm \sigma_{f_i}$ shall denote the individual summed up products. According to Gaussian error propagation, the resulting error can be obtained by (2.44), thus

$$f_i^* = N_i^* N_{i+\hat{\tau}}^* \quad \text{and} \quad \sigma_{f_i} = f_i^* \sqrt{\left(\frac{\sigma_i}{N_i}\right)^2 + \left(\frac{\sigma_{i+\hat{\tau}}}{N_{i+\hat{\tau}}}\right)^2} \ .$$

The error of the sum is then

$$\Psi_{\hat{\tau}} = \sum_i f_i^* \pm \sqrt{\sum_i \sigma_{f_i}^2}$$

[25] Actually, the phase could be estimated by taking the current position of Earth in its orbit between perihelion and aphelion into account.

[26] In all comparisons that were performed, both methods of summation yielded the same results.

and plugging in yields

$$\Psi_{\hat{\tau}} = \Psi^*_{\hat{\tau}} \pm \sigma_{\Psi_{\hat{\tau}}} = \sum_i N^*_i N^*_{i+\hat{\tau}} \pm \sqrt{\sum_i \left(N^*_i N^*_{i+\hat{\tau}}\right)^2 \left[\left(\frac{\sigma_i}{N_i}\right)^2 + \left(\frac{\sigma_{i+\hat{\tau}}}{N_{i+\hat{\tau}}}\right)^2\right]} \tag{2.67}$$

for the estimated error of the autocorrelation.

Comparison with the empirical calculation of the autocorrelation's expectation values and standard deviations (see figure 4) according to (2.43) shows that (2.67) is indeed a very generous estimation of the correct uncertainty. It ignores obvious correlation effects which apparently *minimize* the actual uncertainty. At large times t the results converge.

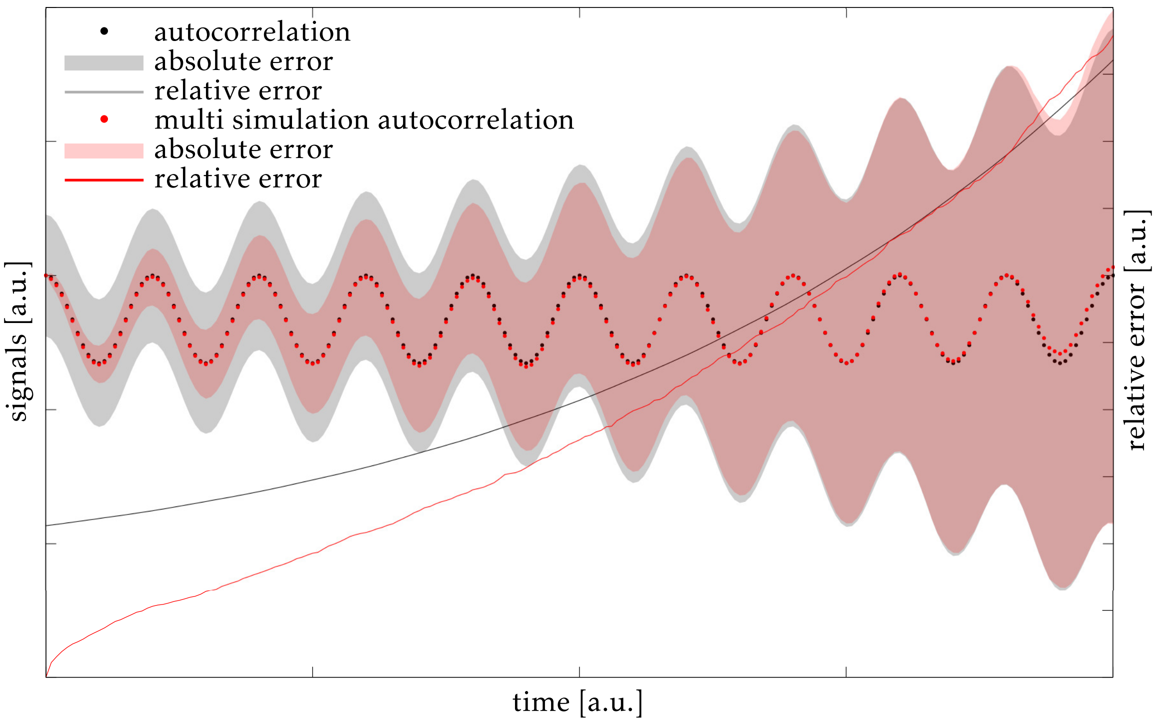


Figure 4 Error calculation of the autocorrelation. The red curves are the empirical results computed by 1001 simulations and (2.43). The black curves are the obtained via one autocorrelation simulation and the error was estimated via (2.67).

2.4.2. Standard decay subtraction

Any oscillation of amplitude ε in a simulated/measured signal is obscured by the exponential background and its large statistical uncertainty due to the nature of the decay process. The exponential component of the signal surmounts the oscillating one by many orders of magnitude and significantly complicates the analysis or might even render it futile.

One approach to get rid of said background is what we might call "standard decay subtraction". By knowing what kind of sample is used in the measurement process, one also knows what statistical mean background is to be expected in the signal. Simply subtracting these values from the signal should expose the deviations from the expected values – be it statistical (as inherent with the phenomenon) or periodic fluctuations – and make the autocorrelation analysis more sensible.

The general ansatz for the modified signal ΔN_{sub} reads

$$\Delta N_{\mathrm{sub}}(t) := \Delta N(t;\Delta t) - \Delta N_{\mathrm{std}}(t) \ , \tag{2.68}$$

where ΔN is the measured/simulated non-standard signal and ΔN_{std} are the analogous standard decay expectation values as in (2.8). Both are of the same oder of magnitude, which is expected to make the deviations visible.

This procedure, however, has two disadvantages: First of all, as we will see, an exponential term $\propto \exp(-\lambda t)$ remains, albeit of smaller amplitude, which complicates analysis. Secondly, the relative error drastically increases by this procedure, since a simple constant shift of the data does not affect the uncertainty of the values according to (2.42). Since ΔN_{sub} fluctuates about zero, the relative error will attain extreme values.

In the following, the expressions for ΔN_{sub} for the examined models will be presented.

Sinusoidal summand model

For the the sinusoidal summand model, using (2.18), the subtracted signal reads

$$\begin{aligned}\Delta N_{\mathrm{sub}}(t) &:= \Delta N(t;\Delta t) - \Delta N_{\mathrm{std}}(t)\\ &= N_0\,\mathrm{e}^{-\lambda t}\Big[1+\eta\varepsilon+\varepsilon\cos(\omega t) - \mathrm{e}^{-\lambda\Delta t}\Big(1+\eta\varepsilon+\varepsilon\cos(\omega(t+\Delta t))\Big)\Big] - N_0\,\mathrm{e}^{-\lambda t}\Big(1-\mathrm{e}^{-\lambda\Delta t}\Big)\\ &= N_0\,\mathrm{e}^{-\lambda t}\Big[1+\eta\varepsilon+\varepsilon\cos(\omega t) - \mathrm{e}^{-\lambda\Delta t}\Big(1+\eta\varepsilon+\varepsilon\cos(\omega(t+\Delta t))\Big) - 1 + \mathrm{e}^{-\lambda\Delta t}\Big]\end{aligned}$$

For $\lambda\Delta t \to 0$ the term $\exp(-\lambda\Delta t) \approx 1$ can be approximated and the expression simplifies to

$$\Delta N_{\mathrm{sub}}(t) \approx N_0\,\mathrm{e}^{-\lambda t}\Big[\varepsilon\Big(\cos(\omega t) - \cos(\omega(t+\Delta t))\Big)\Big] \ .$$

Since Δt is chosen by the experimenter and ω assumed to be constant and evaluated through the data, we can write $\omega\Delta t = \phi = \text{const}$ and obtain

$$\Delta N_{\mathrm{sub}}(t) \approx N_0\,\mathrm{e}^{-\lambda t}\Big[\varepsilon\Big(\cos(\omega t) - \cos(\omega t+\phi)\Big)\Big] \ .$$

Sinusoidal decay constant model

For the model of a sinusoidal decay constant, using (2.28), the subtracted signal reads

$$\begin{aligned}\Delta N_{\mathrm{sub}}(t) &:= \Delta N(t;\Delta t) - \Delta N_{\mathrm{std}}(t)\\ &= N_0\,\mathrm{e}^{-\lambda_0 t}\Big[\mathrm{e}^{\lambda_0\frac{a}{\omega}\cos(\omega t)} - \mathrm{e}^{-\lambda_0\Delta t}\,\mathrm{e}^{\lambda_0\frac{a}{\omega}\cos(\omega(t+\Delta t))}\Big] - N_0\,\mathrm{e}^{-\lambda_0 t}\Big(1-\mathrm{e}^{-\lambda_0\Delta t}\Big)\\ &= N_0\,\mathrm{e}^{-\lambda_0 t}\Big[\mathrm{e}^{\lambda_0\frac{a}{\omega}\cos(\omega t)} - \mathrm{e}^{-\lambda_0\Delta t}\,\mathrm{e}^{\lambda_0\frac{a}{\omega}\cos(\omega(t+\Delta t))} - 1 + \mathrm{e}^{-\lambda_0\Delta t}\Big]\\ &= N_0\,\mathrm{e}^{-\lambda_0 t}\Big[\mathrm{e}^{\varepsilon\cos(\omega t)} - \mathrm{e}^{-\lambda_0\Delta t}\,\mathrm{e}^{\varepsilon\cos(\omega(t+\Delta t))} - 1 + \mathrm{e}^{-\lambda_0\Delta t}\Big] \ ,\end{aligned}$$

where we used (2.30) in the last step. Expanding the terms $\propto \exp(\varepsilon)$ according to (2.29) yields

$$\Delta N_{\text{sub}}(t) \approx N_0\, e^{-\lambda_0 t}\left[\Big(1+\varepsilon\cos(\omega t)\Big) - e^{-\lambda_0 \Delta t}\Big(1+\varepsilon\cos(\omega(t+\Delta t))\Big) - 1 + e^{-\lambda_0 \Delta t}\right] \ ,$$

which again, for $\lambda_0 \Delta t \to 0$ and $\omega\Delta t = \phi = \text{const}$, simplifies to

$$\Delta N_{\text{sub}}(t) \approx N_0\, e^{-\lambda_0 t}\left[\varepsilon\Big(\cos(\omega t) - \cos(\omega t + \phi)\Big)\right] \ .$$

The result for the models is the same in the first approximation, both oscillating about zero. However, in addition to the problem of the relative error getting out of hand, the subtracted signal still includes an exponential contribution/envelope.

Error propagation

When computing the subtracted signal for a given set of $\Delta N(t_i;\Delta t) =: \Delta N_i$ and $\sigma_{\Delta N_i}$ from a measurement, it is likely that N_0 has to be calculated from the available data. The error propagation is not trivial in this case. The values $N_{\text{std},i}$ will be calculated after $\overline{N_0}$ and $\sigma^2_{\overline{N_0}}$ was computed according to (2.48):

$$N_{\text{std},i} = \overline{N_0}\, e^{-\lambda t_i}$$

Simple error propagation yields

$$\sigma^2_{N_{\text{std},i}} = e^{-2\lambda t_i}\, \sigma^2_{\overline{N_0}} \ ,$$

where $\sigma^2_{\overline{N_0}}$ is calculated via (2.49). According to the rules of error propagation, (2.68) has to be derived with respect to the two variables ΔN_i and $N_{\text{std},i}$, since both have uncertainties. Mind that the standard values depend on the weighted mean number of initial nuclei, which itself is dependent on ΔN_i!

$$\begin{aligned}
\frac{\partial}{\partial(\Delta N_i)}\left(\Delta N_i - \overline{N_0}\, e^{-\lambda t_i}\right) := D_{\Delta N_i} &= 1 - \frac{\partial}{\partial(\Delta N_i)}\left(\overline{N_0}\, e^{-\lambda t_i}\right) \\
&= 1 - e^{-\lambda t_i}\, \frac{\partial}{\partial(\Delta N_i)}\left(\frac{\sum_k B_k}{\sum_k \frac{B_k^2}{\Delta N_k}}\right) \\
&= 1 - e^{-\lambda t_i}\left(\sum_k B_k\right)\cdot \frac{\partial}{\partial(\Delta N_i)}\left(\sum_k \frac{B_k^2}{\Delta N_k}\right)^{-1} \\
&= 1 - e^{-\lambda t_i}\left(\sum_k B_k\right)\cdot(-1)\left(\sum_k \frac{B_k^2}{\Delta N_k}\right)^{-2} \frac{\partial}{\partial(\Delta N_i)}\left(\sum_k \frac{B_k^2}{\Delta N_k}\right) \\
&= 1 + e^{-\lambda t_i}\left(\sum_k B_k\right)\cdot\left(\sum_k \frac{B_k^2}{\Delta N_k}\right)^{-2} (-1)\left(\frac{B_i^2}{\Delta N_i^2}\right) \\
&= 1 - e^{-\lambda t_i}\, \frac{\sum_k B_k}{\left(\sum_k \frac{B_k^2}{\Delta N_k}\right)^2}\left(\frac{B_i^2}{\Delta N_i^2}\right)
\end{aligned}$$

where we used the fact that in the sum only the term where $k = i$ gives a non-zero contribution. We also find

$$\frac{\partial}{\partial(\Delta N_{\text{std},i})}\left(\Delta N_i - N_{\text{std},i}\right) := D_{N_{\text{std},i}} = -1 \ .$$

Thus, we find for the variance of the subtracted signal value $\Delta N_{\mathrm{sub},i} = \Delta N_i - N_{\mathrm{std},i}$:

$$\sigma^2_{\Delta N_{\mathrm{sub},i}} = D^2_{\Delta N_i}\, \sigma^2_{\Delta N_i} + D^2_{N_{\mathrm{std},i}}\, \sigma^2_{N_{\mathrm{std},i}}$$

The error propagation for the sinusoidal summand model is not provided, since it is not used in the quantitative analysis.

2.4.3. Standard decay normalization

Similar to the just introduced modification is the standard decay normalization. Much like the standard decay subtraction method it aims at stripping the signal from the exponential background and it even avoids the major disadvantages of said method.

As its name suggests, the definition of the normalized signal ΔN_{norm} reads

$$\Delta N_{\mathrm{norm}}(t) := \frac{\Delta N(t;\Delta t)}{N_{\mathrm{std}}(t)} \quad , \tag{2.69}$$

where again ΔN is the measured/simulated non-standard signal and N_{std} are the standard decay expectation values.

This procedure eliminates all time dependent terms $\propto \exp(-\lambda t)$, which leads to a constant amplitude of any visible oscillation. Furthermore, the relative error remains the same, because of (2.42). It is the total extinction of the time dependent exponential contribution that will render this analysis method very useful when estimating the effect's amplitude ε and frequency ω.

Note that the label *normalization* is a little bit misleading as one would expect the resulting signal to oscillate around one, however, the signal is normalized to *standard* decay values in order to expose ε! Effect magnitudes as given in the references mentioned in the introduction are given for the normalization with respect to ΔN_{std}.

In the following, the expressions for ΔN_{norm} for the examined models will be presented.

Sinusoidal summand model

For the the sinusoidal summand model, using (2.18), the normalized signal reads

$$\begin{aligned}
\Delta N_{\mathrm{norm}}(t) &:= \frac{\Delta N(t;\Delta t)}{N_{\mathrm{std}}(t)} \\
&= \frac{N_0\,\mathrm{e}^{-\lambda t}\Big[1 + \eta\varepsilon + \varepsilon\cos(\omega t) - \mathrm{e}^{-\lambda\Delta t}\Big(1 + \eta\varepsilon + \varepsilon\cos(\omega(t+\Delta t))\Big)\Big]}{N_0\,\mathrm{e}^{-\lambda t}} \\
&= 1 + \eta\varepsilon + \varepsilon\cos(\omega t) - \mathrm{e}^{-\lambda\Delta t}\Big(1 + \eta\varepsilon + \varepsilon\cos(\omega(t+\Delta t))\Big)
\end{aligned}$$

With the same arguments as before we set $\lambda\Delta t \to 0$ and $\omega\Delta t = \phi = \mathrm{const}$ and obtain

$$\Delta N_{\mathrm{norm}}(t) \approx \varepsilon\Big(\cos(\omega t) - \cos(\omega t + \phi)\Big) \ .$$

Sinusoidal decay constant model

For the model of a sinusoidal decay constant, using (2.28), the normalized signal reads

$$\begin{aligned}
\Delta N_{\mathrm{norm}}(t) &:= \frac{\Delta N(t;\Delta t)}{N_{\mathrm{std}}(t)} \\
&= \frac{N_0\,\mathrm{e}^{-\lambda_0 t}\Big[\mathrm{e}^{\lambda_0\frac{a}{\omega}\cos(\omega t)} - \mathrm{e}^{-\lambda_0\Delta t}\,\mathrm{e}^{\lambda_0\frac{a}{\omega}\cos(\omega(t+\Delta t))}\Big]}{N_0\,\mathrm{e}^{-\lambda_0 t}} \\
&= \mathrm{e}^{\lambda_0\frac{a}{\omega}\cos(\omega t)} - \mathrm{e}^{-\lambda_0\Delta t}\,\mathrm{e}^{\lambda_0\frac{a}{\omega}\cos(\omega(t+\Delta t))} \\
&= \mathrm{e}^{\varepsilon\cos(\omega t)} - \mathrm{e}^{-\lambda_0\Delta t}\,\mathrm{e}^{\varepsilon\cos(\omega(t+\Delta t))} \quad ,
\end{aligned}$$

where we used again (2.30) in the last step. Expanding the terms $\propto \exp(\varepsilon)$ according to (2.29) yields

$$\Delta N_{\mathrm{norm}}(t) \approx \Big(1 + \varepsilon\cos(\omega t)\Big) - \mathrm{e}^{-\lambda_0 \Delta t}\Big(1 + \varepsilon\cos(\omega(t+\Delta t))\Big) \ , \tag{2.70}$$

which as usual, for $\lambda\Delta t \to 0$ and $\omega\Delta t = \phi = \mathrm{const}$, simplifies to

$$\Delta N_{\mathrm{norm}}(t) \approx \varepsilon\Big(\cos(\omega t) - \cos(\omega t + \phi)\Big) \ . \tag{2.71}$$

Yet again, the results for both models are similar in the first approximation. Additionally, they differ from the findings in the case of standard decay subtraction merely by a pre-factor. The normalization is using N_{std} instead of ΔN_{std} since the latter would contribute a factor $1/[1-\exp(-\lambda\Delta t)]$ to the amplitude of the normalized signal.

We will see in section 4.2.4 Retrieving the effect's frequency ω and amplitude ε (p. 87) how the amplitude ε and the frequency ω of the effect can be derived from that relation.

Error propagation

For computing the normalized signal for a given set of $\Delta N(t_i;\Delta t) =: \Delta N_i$ and $\sigma_{\Delta N_i}$ from a measurement, we proceed analogously to the case of the subtracted signal.

We derive (2.69) with respect to the two variables ΔN_i and $N_{\mathrm{std},i}$ for the error propagation.

$$\begin{aligned}
\frac{\partial}{\partial(\Delta N_i)}\left(\frac{\Delta N_i}{\overline{\overline{N_0}}\,\mathrm{e}^{-\lambda t_i}}\right) := D_{\Delta N_i} &= \frac{1}{\overline{\overline{N_0}}\,\mathrm{e}^{-\lambda t_i}} + \Delta N_i \frac{\partial}{\partial(\Delta N_i)}\left(\frac{1}{\overline{\overline{N_0}}\,\mathrm{e}^{-\lambda t_i}}\right) \\
&= \frac{1}{\overline{\overline{N_0}}\,\mathrm{e}^{-\lambda t_i}} + \Delta N_i\,\mathrm{e}^{\lambda t_i}\frac{\partial}{\partial(\Delta N_i)}\left(\frac{\sum_k \frac{B_k^2}{\Delta N_k}}{\sum_k B_k}\right) \\
&= \frac{1}{\overline{\overline{N_0}}\,\mathrm{e}^{-\lambda t_i}} + \frac{\Delta N_i\,\mathrm{e}^{\lambda t_i}}{\sum_k B_k}\cdot(-1)\frac{B_i^2}{\Delta N_i^2} \\
&= \mathrm{e}^{\lambda t_i}\left(\frac{1}{\overline{\overline{N_0}}} - \frac{1}{\Delta N_i}\frac{B_i^2}{\sum_k B_k}\right) \ ,
\end{aligned}$$

where we used the fact that in the sum only the term where $k = i$ gives a non-zero contribution. We also find

$$\frac{\partial}{\partial(\Delta N_{\mathrm{std},i})}\left(\frac{\Delta N_i}{N_{\mathrm{std},i}}\right) := D_{N_{\mathrm{std},i}} = -\frac{\Delta N_i}{N_{\mathrm{std},i}^2} \ .$$

Thus, we find for the variance of the normalized signal value $\Delta N_{\mathrm{norm},i} = \Delta N_i / N_{\mathrm{std},i}$:

$$\sigma^2_{\Delta N_{\mathrm{norm},i}} = D^2_{\Delta N_i}\,\sigma^2_{\Delta N_i} + D^2_{N_{\mathrm{std},i}}\,\sigma^2_{N_{\mathrm{std},i}}$$

The error propagation for the sinusoidal summand model is not provided, since it is not used in the quantitative analysis.

2.4.4. Smoothing and filtering

An operation mathematically very similar to the autocorrelation is the convolution of two functions $N(t)$ and $g(t)$, denoted by the symbol $(N * g)(t)$. Its definition reads as follows:

$$(N * g)(t) = \int_{-\infty}^{\infty} N(\tau)g(t-\tau)\,\mathrm{d}\tau \tag{2.72}$$

Note that the argument t of the convolution function is the same as in $N(t)$, and the lag τ is the integration variable as opposed to the autocorrelation function, where τ becomes the argument of the resulting function $\Psi(\tau)$.

The time-discrete version of (2.72) is

$$(N * g)_i = \sum_{k=1}^{M} N_k \, g_{i-k} \ , \tag{2.73}$$

where $k\Delta t \mathrel{\hat{=}} \tau$ and $k \in \mathbb{N}_0$ and M equals the number of included data points.[27]

The uncertainty propagation is pretty forward since g is analytical. The errors of the products are calculated using (2.42) and the errors of the sum via (2.44). Thus we find

$$(N * g)_i = \sum_k N_k \, g_{i-k} \pm \sqrt{\sum_k g_{i-k}^2 \, \sigma_i^2} \ .$$

The convolution can be understood as the product of two functions or the moving average of $N(t)$, weighted by the function $g(t)$.

In system theory, $N(t)$ is an input signal and $g(t)$ the so-called system response function of a linear system, which contains its complete information (it is the system's response to an input Dirac pulse). The convolution function is then the output signal corresponding to $N(t)$ after going through the system.

The response function must be normalized, otherwise it distorts the output signal.

By choosing a suitable response function g, one can smooth the input signal, i.e. remove high-frequency components. In this context, g might rather called the kernel. In image processing, so-called Gaussian smoothing is very common. In this case, g is just a normalized Gauss function

$$g(t;\sigma_t) = \frac{1}{\sqrt{2\pi}\sigma_t} \, \mathrm{e}^{-\frac{t^2}{2\sigma_t^2}} \ . \tag{2.74}$$

The convolution of two functions $N(t)$ and $g(t)$ as in (2.72) can be written as the product of their respective Fourier transforms $\widehat{N}(\omega)$ and $\widehat{g}(\omega)$ in Fourier space

$$(N * g)(t) = \widehat{N}(\omega) \cdot \widehat{g}(\omega) \ ,$$

where $\widehat{N}(\omega)$ is called the Fourier spectrum of $N(t)$ and $\widehat{g}(\omega)$ the frequency response of $g(t)$. This is the formalism of filtering.

Obviously, smoothing and filtering are profoundly entangled and can not be separated from each other – they are two ways of describing the same process within two different domains. In the time domain, the meaning of the signal $N(t)$ might be more tangible for us and we consider the convolution with a kernel $g(t)$ as some kind of averaging (smoothing). In Fourier space, the frequency response $\widehat{g}(\omega)$ of a kernel, a filter, is very graphic and we can quickly understand, how the frequency components of a signal ($\widehat{N}(\omega)$ – which might not be as conceivable as $N(t)$) are modified by it.

Let $\widehat{N'}(\omega) = \widehat{N}(\omega)\widehat{g}(\omega)$ be the modified spectrum of the original signal $N(t)$. Then the inverse Fourier transformation of $\widehat{N'}(\omega)$ back into the time domain, denoted by $N'(t)$ – the filtered signal – is equal to the smoothed signal $(N * g)(t)$ obtained by convolution with the kernel $g(t)$!

[27] For the sake of readability, we use N for an arbitrary signal instead of ΔN, which would be the correct nomenclature for the signal in our case.

Now, being aware of the influence of the application of a smoothing kernel on the frequency spectrum of the input signal, one should investigate its frequency response.

Looking at the well-known solution

$$\mathrm{e}^{-at^2} \xrightarrow[\text{transform}]{\text{Fourier}} \frac{\pi}{a}\,\mathrm{e}^{-\frac{\pi^2\omega^2}{a}}$$

shows that the Fourier transform of a Gaussian is again a Gaussian and comparison against (2.74) yields

$$a = \frac{1}{2\sigma_t} \quad \text{and} \quad \frac{\pi^2}{a} = \frac{1}{2\sigma_\omega} \ .$$

Eliminating a in both equations provides us with a relation between the standard deviations of the kernel σ_t and the respective frequency response σ_ω:

$$\sigma_t = \frac{1}{2\pi\,\sigma_\omega} \tag{2.75}$$

From this we can draw the conclusion that smoothing with a Gaussian kernel will essentially function as a low-pass filter (since its frequency response has the shape of a Gaussian as well), gradually suppressing all frequency components about $\omega \approx \sigma_\omega$ and higher.

Since the discretization due to the nature of the data acquisition $N(t_i)$ in discrete time intervals Δt in combination with the inherent, strong statistical fluctuation introduces "artificial" frequency components, smoothing (i.e. low-pass filtering) the signal with a proper choice of σ_t can get rid of the noise introduced by the statistical fluctuations and simplify eventual fitting processes.[28]

2.4.5. Accumulation

It is perfectly possible to accumulate k data points of a measurement with count numbers ΔN_k and integration time Δt to one new data point ΔN after the time $k\Delta t$. The accumulated count number is then just

$$\Delta N = \sum_k \Delta N_k \ .$$

The resulting error can be easily calculated by using (2.44); the derivatives are simply $\frac{\partial \Delta N}{\partial(\Delta N_k)} = 1$ and using $\sigma_k^2 = \Delta N_k$ (refer to (2.53)) will lead to

$$\sigma_k = \sqrt{\sum_i \frac{\partial \Delta N}{\partial(\Delta N_k)}\sigma_k^2} = \sqrt{\sum_k \Delta N_k} = \sqrt{\Delta N} \ .$$

This shows that subsequent accumulation does not affect the uncertainty of the results. This means that accumulation of a signal can be performed such that the investigated phenomenon is still visible but at the highest possible accuracy.

[28] We are talking about low-pass filtering because we expect the frequency of the effect to be very low (cycle duration of approximately one year).

3. Simulation

The main part of this thesis was to create a computer program – a simulation written in C++. Now that the physical and mathematical issues of radioactive decay have been discussed, the computer simulation can be explained in great detail.

3.1. Requirements and goal

3.1.1. Primary requirements and goal

The simulation should meet two primary requirements.

Firstly, the generation of a signal in tuples of values $\left(t_{i+1}\,,\ \Delta N(t_i;\Delta t),\ \sqrt{\Delta N(t_i;\Delta t)}\right)$ like a real measurement would provide them in the case of non-standard decay and secondly, it should feature the means to analyze them in order to find the anomalies. In short:

- signal generation
- signal analysis

The goal is to find out *if*, *how*, *when* and under which *circumstances* we can find *what kind* of effect of *which magnitude* in a set of data. Therefore, the second point is, in fact, the most important one. The generation of signals is necessary only because we have to test if the analysis methods can find an effect of which we *know* it is there because we "made" it.

3.1.2. Secondary requirements

The simulation was created in compliance with additional requirements no less important. The separation is solely motivated by the fact that the "primary" demands formulate the objective of the program, whereas the "secondary" conditions will specify the architecture of the program itself.

The computer program is designed to be a flexible "framework", thus it is essential to make it adaptable, flexible and expandable in order to develop it further. The desired modularity demands to follow the principle of loose coupling and the separation of concerns as much as possible, both of which are often referred to in object orientated programming. The former term means that all components of the software should depend as little as possible on each other and should function independently. The latter term implies that every specific task should be performed by a dedicated component of the software.

A basic framework in order to provide file I/O and documentation facilities needs to be established.

Data needs to be distributed in a simple and organized fashion and it is necessary to provide the means to implement different physical models with little effort.

A generic mechanism needs to provide a simulated signal depending on the given input parameters and chosen model.

Once a signal is created, the framework needs to provide the means to perform various analyses and enable the user to add new methods easily.

3.2. Structure

The aforementioned requirements impose a certain structure on the program, which is the object of this section.

3.2.1. Schematic description of the framework

First of all, we certainly need to separate the implementation of a physical **model** from the "dumb" computing work, the act of **simulation**. The model will contain the physical information (formulas, parameters, ...), which shall be used for computation, whereas the simulation just *uses* said formulas to create (with the aid of a random number generator) a signal it knows nothing about, though. Loose coupling and the presence of multiple models motivate the implementation of a simulation component, which can handle *any* model!

Comparing to "reality", the model plays the role of the actual physics involved, the simulation represents the collection of data as in the course of a real measurement.[29] Both pivot on a radioactive sample, which is inherent to the model.[30] A third component, **input data**, consists of merely more than organized collections of data in order to represent sample and parameter attributes.

All three components are (and should be!) independent from each other! A model doesn't need information about the implementation of a sample, it just has to provide mathematical instructions representing whatever alternative physics we seek to describe. Then again, the simulation shall not care *which* model it performs thousands of calculations for. Its job is to take a model, compute data and save it to a file

This data will be used by a fourth component, which is charged with data **analysis**. Since there may be a manifold of available analysis methods, it's good to have a solid base structure for this kind of tasks.

To make all these parts work together seamlessly and consistently, a collection of **utility** components provides file management, file documentation, data distribution and conversion functions.

The interconnection of the components of the framework is schematically depicted in figure 5.

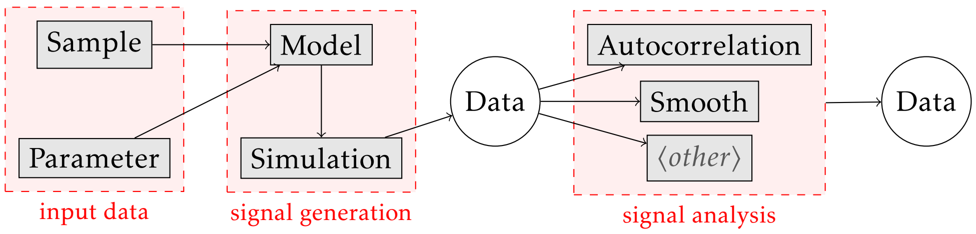


Figure 5 Schematic illustration of the program components divided into three main task groups (red). The utility classes are not included in this picture.

[29] Actually, the engaged hardware can be simulated as well – the complexity of this part of the simulation is arbitrary, abundant and will not be covered in the course of this thesis.

[30] A future design may change that fact in order to enable the model to deal with more than one sample at a time.

3.2.2. Fundamental implementation

The introduced modules will be implemented through respective classes. Especially the model and the simulation class have to work together in that the simulation class has to know about *how* to create a signal according to any given model. The model class is obviously independent from the simulation class and the analysis classes are not concerned with the rest.

The following explanation is no exhaustive description of the implementation, but it lists all defined classes, gives a picture of the role and functionality of every component and how they fulfill the stated requirements. For a more detailed description see section 3.4 Implementation of the framework (p. 54) and refer to the full program code in the CD attachment of this thesis.

Utility classes

The core of the framework is given by several utility classes, which simplify and standardize the basic mechanisms of the program.

The `File` class keeps file data like path, name and extension and allows for basic modifications like adding pre- and post-fixes. It is used to have a consistent format for file names and to easily distribute file references (as they are the source of data) to the different components of the program.

The `FileDoc` (file documentation) class is a very simple utility class which is used in every non-utility class for the purpose of documentation. It provides simple means to add information to it which will be printed into every output file. Every class apart from the utility classes are supposed to use this facility. In that, every file containing data also contains information about the chosen parameters, model etc.

The `data3` struct is a POD ("plain old data") C-style structure in order to standardize the format of the data used in the simulation framework. It represents the triple $\left(x, f(x), f_{\text{error}}(x)\right)$ and is nothing but a data container with no specific functionality whatsoever, but it enables concise implementations like `std::vector<data3> result_;`.

The `Data` class is an important class charged with data distribution throughout the components of the simulation and with file I/O. Every read or write process is managed by this class, which guarantees consistency and compatibility throughout. It keeps track of accessed files in order to avoid overwrites, provides static getters to "globally" define and distribute information like comment character (for in-file documentation) or data tag (to mark the beginning of the data in a file for read-out) and supplies a static method to write data to a file. These methods are implemented static so that other classes can access its functionality. The class makes data it manages accessible read-only (by providing a constant reference to the data) to avoid redundancy and of course uses the aforementioned classes.

The `RandomNumbers` class is an abstract, static class, which can not be instantiated but consists only of public, static members. Its purpose is to provide the means to generate pseudo-random numbers by a pseudo-random number engine to any class calling its methods. It may provide pseudo-random numbers originating from a normal distribution and offers a mechanism to seed the engine and return the value of said seed.

The `Analysis` class is an important abstract base class which provides the basic functionality of every analysis method to be implemented. That way, the demand of consistency throughout all analysis methods is accommodated. Furthermore, this keeps the actual analysis classes simple and slender, as they have to implement the new functionality only. Because of that, this class has to be inherited and no objects of `Analysis` can be instantiated. It accesses the data to be evaluated by a read-only reference provided by a `Data` object and uses the static interface of the latter to make its results available.

Input data classes

The input data classes collect and logically capsule information about samples and parameters of the different models. They are used to keep the respective model abstractions tidy. Since most of the actual values are of such intangible magnitudes, a set of conversion functions is provided and eases their handling.

The `Sample` class is the object-orientated representation of a physical nuclide.

The `Parameter` class is an abstract parent class which has to be inherited for every created model and will hold the parameters defined by the respective model.

A set of conversion functions is defined in the `conv` namespace, which will convert years or months to seconds or half-lives into decay constants, for example.

Model classes

The `Model` class is also an abstract base class of which no objects can be created; it must be inherited. Each implementation of a derived child class represents a physical model.

Every model class will provide a `Calculation()` function (virtual and null-defined in the parent class) featuring equations (I) to (III) of the respective model which will be called by the simulation component. To address the different equations in a descriptive way, an enumeration class `CalcType` was introduced.

Furthermore, the parent contains a `Sample` object and the children add their individual `Parameter` child class object.[31]

Simulation class

The `Simulation` class is the working horse of the framework and probably its most crucial component. It is responsible for generating signal data either by calculating the statistical mean values according to the equations provided by a model or by additionally using pseudo-random numbers in order to simulate of a real measurement. For this purpose, the random numbers must originate from a normal distribution – a feature which is provided by the `RandomNumbers` class. Furthermore, it documents the current seed in the file documentation, which makes every simulation run entirely reproducible.

Much like the experimenter, the user can set the total measurement time $\mathscr{T}$ and integration time Δt. By design and definition, these input parameters are of integer type and of unit seconds. This makes seconds the ("quantized") unit of time throughout the framework and accounts for consistency, simplicity and any possible errors due to machine epsilon.

To fulfill its task in the most coherent way, it makes use of almost all utility classes.

3.2.3. Style choice, naming conventions and coding standards

This section briefly presents the style choices, naming conventions and coding standards in this project.

As in all complex projects, the KISS principle is of greatest importance: "Keep it short and

[31] The parent does not include a `Parameter` base class object, because a `child` is only compatible with a specific, individually created `Parameter` child.

simple". In the course of its creation, the current version of the framework underwent numerous, sometimes very substantial, changes in the attempt to achieve both simplicity and flexibility in the most elegant fashion.

Also, the order of "make it work, make it nice, then make it fast" in terms of priority shall be kept in mind to never sacrifice flexibility and simplicity for speed in a clear case of premature optimization. This application is anything but time-critical.

In the light of separation of concerns and encapsulation, the classes should be split up into several header and implementation files. Currently, each group (the utility classes, the input data classes, the model classes and the simulation class) have their own translation unit. The analysis methods are defined in isolated units.

In order to guarantee traceability and reproducibility, every file created by the framework shall provide thorough documentation which can be easily achieved by the `FileDoc` class.

All code in the header files must be commented in Doxygen syntax to allow for the automatic generation of an exhaustive documentation. The structural means of Doxygen (grouping, cross-referencing, member and parameter referencing, LaTeX interpreting etc.) should be generously used. The member fields and methods are mostly documented inline via `\\\<`, members are referenced by `#` and function parameters by `\a`. Documenting without using line-breaks is of advantage, because the code stays compact when auto-wrapping is disabled in the IDE.

The implementation files will only include comments concerning the actual coding and thus be C++-style. The generated documentation will include all files nonetheless. Implementation documentation should always answer the question "Why?". Decent code answers the questions "What?" by sophisticated naming and "How?" by decent implementation by itself in most cases.

The conceptual structure and implementation of the framework is covered in this text. To keep this description free from clutter and distracting details, it is important to document the source code in great detail. The combination of these two sources – concept and overview in the text, details in the code (header files) – should enable the reader to comprehend, work with, refine and advance the framework.

A thorough output via the standard `std::cout` stream in order to inform the user about the current tasks in progress is mandatory. Tab stops should be used to indicate the hierarchy of the current process. For example:

```
Creating Standard Decay Normalization object (source file: D:/path/to/file.dat)
    Received 43830 data points.
    Performing standard decay normalization...
        (Calculating N0 from the data.)
        ...done.
    Requesting to print results to file...
        Printing data to file (D:/path/to/file.dat)...
        ...done.
```

Variables and functions are written lower camel case.[32] Member variables are indicated by a trailing underscore: `classMemberVariable_`. Function names should always hint at what they do or provide by the use of meaningful prefixes like "set", "get", "is" etc. Also variable names should tell as much about them as possible.

Class names and enumerations are upper camel case, structs are lower camel case.

Constructor arguments should always have the same name as the respective member fields

[32] Camel case means that the first letter of every word the variable/function name consists of is capitalized. Lower camel case means that the very first letter is printed lowercased; `justLikeThis`. In upper camel case also the first character is capitalized; `LikeInThisExample`.

without the trailing underscore.

In case of function parameters, the required unit should be given in the parameter name offset by an underscore, e.g. `void setFrequency (double frequency_inOneOverSeconds);` as the setter for a class member `frequency_`.

Children of the `Parameter` class start with a capital `P`, e.g. `PSineSummand`, and children of the `Model` class with capital `M`, e.g. `MSineSummand`. This way, classes of the same model can carry the same name.

The implementations of the framework follow the principle of "const correctness". This means that every method, which does not alter the state of the calling object, should be declared constant. Applied consistently together with sound naming conventions and meaningful variable/function names, const correctness can produce very clean code, which automatically tells the user exactly what is happening. Prime example of const methods are getters, for instance.

Trivial functions like getters and setters are implemented in the respective header file.

Operator overloads are declared as member functions in case of unary operators and in case of binary operators when one of objects is changed (e.g. `operator+=`). If a binary operator treats both objects equally (e.g. `operator+`, `operator==`, ...) it is implemented as a non-member function; in that case it may be friend of the operand's type.

Furthermore, there should always be provided a set of related operators, if meaningful (e.g. `operator==` and `operator!=`, `operator+=` and `operator+`, ...). This can be achieved very effortlessly since related operators to be implemented will just forward their argument to the defined one in a proper manner.

Enumerations are implemented as `enum class`es and a string representation function should be provided.

3.3. Random numbers, normal distributed values and approaches of simulating

3.3.1. Random numbers – Mersenne Twister engine

This short section is devoted to the description of the engaged pseudo-random number engine/generator (P-RNG). The simulation uses the `std::mersenne_twister_engine` [22] with its standard instantiation `std::mt19937`.[33] It is defined in `<random>`. A random number can be generated via:

```
#include <random>
#include <ctime>
std::mt19937 generator(std::time(NULL));      //seeds the P-RNG
double random = generator();                  //returns a random number
```

The argument `std::time(NULL)` returns the current system time and functions as a so-called "seed". Any P-RNG provides a recurring series of numbers, which will eventually be repeated. The quality of the engine determines how "random" the numbers appear and after how many numbers the series repeats itself. The seed defines the entry point in the series. If no seed is provided, the engine will use a default starting point during instantiation. Be aware that each definition with the same (or no) seed will result in the same sequence of "random" numbers!

The `<chrono>` library features a system clock with much higher resolution, which avoids the problem of calling `std::time(NULL)` in such rapid succession that it returns the same value through multiple calls. The code would then look like

```
#include <random>
#include <chrono>
```

[33] A 64 bit version `std::mt19937_64` is available as well.

```
auto time = std::chrono::high_resolution_clock::now();      //get time from system clock
std::mt19937 generator(time.time_since_epoch().count());    //seeds the P-RNG
double random = generator();                                //returns a random number
```

3.3.2. Gaussian normal distributed values

A Gaussian normal distributed random number `value` $\in \mathscr{G}(\mu,\sigma)$ can be obtained by using the introduced random number engine in combination with the `std::normal_distribution` template from `<random>`. It demands two parameters for initialization, namely mean μ and standard deviation σ. In accordance with section 2.3.4 Statistical description of radioactive decay (p. 30), we pass $\mu = \Delta N$ and $\sigma = \sqrt{\Delta N}$, where $\Delta N = \Delta N(t_i;\Delta t)$, every time we need a random value. From the created distribution, the P-RNG picks a value, which is then returned. Every new random number at a different time t_i demands the creation of a new Gaussian distribution with the updated attributes.

Implementation can be accomplished as pictured:

```
#include <random>
std::mt19937 generator();    //assumed to be seeded as described in preceding section
normal_distribution<double> distribution(⟨ΔN⟩, ⟨√ΔN⟩);
double value = distribution(generator);      //random number from the distribution
```

3.3.3. Types of simulating

As held out in section 2.3.1 Principle of measurement (p. 26), more precisely in equations (2.40)

$$\Delta N(t_i;\Delta t) = \sum_{j=1}^{k} n(t_{i,j})\Delta t_{\text{min}} \quad \text{where} \quad \Delta t = k\cdot\Delta t_{\text{min}} \ , \quad t_{i,j} = t_i + j\Delta t_{\text{min}} \quad \text{and} \quad k,j\in\mathbb{N}$$

and (2.41)

$$\Delta N(t_i;\Delta t) = N(t_i) - N(t_i+\Delta t) \ ,$$

there are two types of simulating.

The first one splits a measurement integration interval Δt. The calculation algorithm for decaying nuclei will be called for every single one of the k steps of size Δt_{min}, where $\Delta t = k\cdot\Delta t_{\text{min}}$. The results will be summed up and returned as $\Delta N(t_i;\Delta t)$.

The description and "measurement" of continuous disintegration of nuclei according to (III) is attempted by choosing a reasonably small ("infinitesimal") Δt_{min}, e.g. $\Delta t_{\text{min}} = 1\,\text{s}$, and performing k "measurements" until the time Δt is completely covered.

The second approach calculates the result by using the stem functions straightforwardly, which means that the calculation algorithm according to (I) is called twice per measurement integration interval.

In pseudo-code, both types would look somewhat like:

```
//summation
double result;  //ΔN(t_i;Δt)
for (int j = 1; j <= k; j++)     //delta_t = k·delta_t_min
    result += calculateDecaying(t_i+j*delta_t_min)*delta_t_min;

//stem function
double result;
result = calculateRemaining(t_i) - calculateRemaining(t_i+delta_t);
```

It is important to understand the difference between the two solutions.

The first performs a number of measurements in a very small time interval and sums them up. It is monitoring the activity of the sample, so to say; for a low activity and very small Δt_{min} even one decay after another. In practice, it is indeed possible to measure in “list mode”, which gathers a time stamp for every detected decay process, potentially creating huge data files.

The second solution describes the detector measuring during Δt and providing the number of registered events *afterwards*. However, it delivers no information about *when* the individual counts were detected.

As explained in section 2.4.5 Accumulation (p. 46), there is no difference in statistics between the two methods, thus the second method[34] was implemented in the framework for reasons of simplicity.

The value `result` in the code example above corresponds to the theoretical values given by the law of decay. If we wanted to really simulate a measurement, we needed to – as explained in section 2.3.4 Statistical description of radioactive decay (p. 30) – use values from the right normal distribution. Thus, the code example would read:

```
//summation
double result;  //ΔN(t_i;Δt)
for (int j = 1; j <= k; j++) {  //delta_t = k · delta_t_min
    //calculate theory value
    double theory = calculateDecaying(t_i+j*delta_t_min)*delta_t_min;
    //create corresponding normal distribution (refer to preceding section)
    std::normal_distribution<double> dist(theory, sqrt(theory));
    result += dist();    //get a random value from that distribution (pseudo code; P-RNG missing)
}

//stem function
//calculate theory value
double theory = calculateRemaining(t) - calculateRemaining(t+delta_t);
//create corresponding normal distribution (refer to preceding section)
std::normal_distribution<double> dist(theory, sqrt(theory));
double result = dist(); //get a random value from that distribution (pseudo code; P-RNG missing)
```

3.4. Implementation of the framework

This section will discuss the implementation of the program components in greater detail. Refer to section 3.2 Structure (p. 47) to get familiar with the structure of the framework; specifically subsection 3.2.2 Fundamental implementation (p. 49) offers a short overview of the available classes. The actual (commented) source code can be found in the attachment of this thesis. A documentation can additionally be generated by Doxygen.

Note that the `std::` namespace is set by a `using` directive in the implementation files.

The code examples given in the following section are copies of the contents of the header files (class declarations) and some exemplary sections from the implementation files. All code has been stripped from documentation comments to maximize readability; explanations are given in the accompanying text.

The code inserts should only “list” the declared members, methods and the design of the class. Due to the sophisticated naming conventions all elements are, to a high extent, self-explaining. Together with the explanations and the full source code (containing full documentation) side by side, the reader will be able to comprehend the implementation of the framework.

[34] Actually, this approach is “unrealistic” in that it is impossible to determine the exact number of *remaining* nuclei (which is what using the stem function means) at the start and the beginning of each measurement. However, it is possible in the simulation and valid.

Any parameters of functions and constructors may be omitted in the accompanying texts for reasons of readability. Refer to the code examples printed at the beginning of each section to find the parameters of the declared methods. Since member fields are always furnished with a trailing underscore_, it is easy to distinguish between member variables and function parameters.

While describing a specific class in the following sections, references to fields of other classes not yet in focus will be prefixed with the respective class namespace in order to make the distinction clear.

3.4.1. `File` class

The `File` class is part of the utility classes and it used to manage files. It is defined in the files `utility.h` and `utility.cpp` respectively.

```
//from utility.h
class File {
private:
    std::string path_;
    std::string basename_;
    std::string extension_;

    std::string temporary_;
public:
    File() {}
    File(std::string path, std::string basename, std::string extension);
    friend bool operator==(const File&, const File&);

    void setPath(std::string path);
    void setBasename(std::string basename);
    void setExtension(std::string extension);

    void addPrefix(std::string prefix);
    void addSuffix(std::string suffix);
    bool isModified() const;
    void resetTemporary();

    std::string getPath() const;
    std::string getFilename() const;
    std::string getPathFilename() const;
    File        getFile() const;
};
bool operator==(const File& lhs, const File& rhs);
bool operator!=(const File& lhs, const File& rhs);
```

The `temporary_` member holds a modified version of `basename_` when `addPrefix()` or `addSuffix()` are called. It is possible to call them multiple times – `temporary_` gets updated every time.

The `getFile()` method returns a new `File` object, the member `basename_` of which contains the old basename including any temporary changes. The `temporary_` variable of the new object is reset to `""`. This is referred to as "baking" in the documentation.

The `get`-functions are implemented in the source file, since they perform error checking and the constructor of the class uses said functions to assign the member fields.

Any given `path_` must already exist in the file system; this class does *not* create files or directories.

3.4.2. `FileDoc` class

The `File` class is part of the utility classes and provides a standardized way to add information to any given output `File`. It is defined in the files `utility.h` and `utility.cpp` respectively.

```
//from utility.h
class FileDoc {
private:
    std::string fileComment_;

public:
    FileDoc();
    FileDoc(std::string comment);
    friend bool operator== (const FileDoc&, const FileDoc&);
    FileDoc& operator+=(const FileDoc& rhs);

    void addComment(std::string comment, std::string lineEnd="\n");
    void addCommentTitle(std::string comment);
    template<typename T> void addCommentKeyVal(std::string key, T value) {
        std::stringstream stream;
        stream.precision(15);
        stream << "\t" << key << ": " << value;
        addComment(stream.str(), "\n");
        return;
    }
    void appendString(std::string comment);
    void setFileCommentString(const std::string& comment) {fileComment_=comment;}
    void resetFileCommentString();
    std::string getString() const {return fileComment_;}
};
bool operator== (const FileDoc& lhs, const FileDoc& rhs);
bool operator!= (const FileDoc& lhs, const FileDoc& rhs);
FileDoc operator+ (FileDoc lhs, const FileDoc& rhs);
```

All documentation, obtained as a string variable by calling `getString()`,[35] is stored in the `fileComment_` member. The `addComment⟨...⟩` functions add informative text to the information string with different formatting (they all call `addComment()` though). Note that `addCommentKeyVal()` is implemented as a template function which can handle any standard type.

`appendString()` appends a string without any formatting. Using `setFileCommentString()` overwrites contents of `fileComment_` and `resetFileCommentString()` sets it to `""`!

Every non-utility class contains a `FileDoc` object and must document the parameters (value and unit) it uses!

Here is an example of how such a documentation might look like. The comment character is `#` and the `#BEGIN#` tag marks the beginning of the data; both have been received by static methods of the `Data` class as will be pointed out there.

```
#Model info:                        // addCommentTitle("Model info");
#***********
#   model: Sinosoidal ⟨...⟩          // addCommentKeyVal("model", "Sinusoidal ⟨...⟩");
#   formulas:                       // addComment("formulas:");
#       N_0*exp(-lambda_0*t)*⟨...⟩   // addComment("\t\tN_0*exp(-lambda_0*t)*⟨...⟩");
#       ⟨...⟩
#   ⟨...⟩

#Sample:                             // addCommentTitle("Sample");
#*******
#   name: test sample               // addCommentKeyVal("test sample", name);
#   N(t=0): 100000000               // addCommentKeyVal("N(t=0)", n0);
```

[35] This function name was chosen in case the internal structure of `FileDoc` gets more complex.

```
#   half life (s): 693147180.559945 // etc.
#   decay constant (1/s): 1e-009
#   ⟨...⟩

#BEGIN#
⟨Data⟩
```

3.4.3. `data3` struct

The `data3` struct is part of the utility classes and standardizes the data format $\left(x, f(x), f_{\text{error}}(x)\right)$ to be used throughout the whole program. Every input and output data is expected to have this structure. It is defined in the files `utility.h` and `utility.cpp` respectively.

```
//from utility.h
struct data3 {
    double x;
    double y;
    double err;
};
bool operator== (const data3& lhs, const data3& rhs);
bool operator!= (const data3& lhs, const data3& rhs);
```

This allows for compact definitions like `std::vector<data3> data_;`. The contents of this POD struct objects can be accessed simply via the member accessor “`.`”.

3.4.4. `Data` class

The `Data` class is part of the utility classes and exclusively manages file I/O and data distribution. It uses the more basic utility classes. It is defined in the files `utility.h` and `utility.cpp` respectively.

```
//from utility.h
class Data {
private:
    static std::vector<File> fileList_;

    File file_;
    FileDoc fileDoc_;
    std::vector<data3> data_;

public:
    Data(File file);
    Data(File file, FileDoc fileDoc, std::vector<data3> data);
    ~Data();
    friend bool operator== (const Data&, const Data&);

    static std::string dataIndicator() {return "#BEGIN#";}
    static std::string commentCharacter() {return "#";}

    static bool printDataToFile (const File& file, const FileDoc fileDoc,
        const std::vector<data3>& data);

    File getFile() const {return file_.getFile();}
    FileDoc getFileDoc() const {return fileDoc_;}
    const std::vector<data3>& getData() const {return data_;}
};
bool operator== (const Data& lhs, const Data& rhs);
bool operator!= (const Data& lhs, const Data& rhs);
```

The static `fileList_` member stores every `File` the `Data` class reads from (via the constructor `File(File)`) or writes to (via `printDataToFile()`). This prevents the class from overwriting already existing files. If a destination file is in the `fileList_`, the data will *not* be saved and the function returns false. Upon destruction of a Data object, its file_ member will be removed from the fileList_. Of course, the class can only monitor File objects it registered. It will not perform any file checking in the file system.

The data tag and the comment character which we encountered in the example of the FileDoc class were provided by dataIndictor() and commentCharacter(), which are called by the FileDoc ::addComment-methods (hence the static modifier). This allows for flexibility and eliminates redundancy.

The printDataToFile() method is used to print a given set of data to a given file, obligatorily adding the respective FileDoc. Since it is static, it doesn't matter whether a Data object is making a call or any other class which wishes to store data. This design guarantees a uniform data and file format. Thus, also external data can be saved by this method and thus be used with classes of the framework.

The Data(File) constructor is used when the source of the data is a File already existing in the file system. If the file doesn't exist, a zombie object is created whose FileDoc_ member is set to "NO FILE FOUND!". If the destructor tries to delete a File from the fileList_, it first checks whether the calling object is a zombie by comparing its fileDoc_ contents. If that is the case, the file name is not deleted from the list, because it may be in use by a valid object by now.

The Data(File, FileDoc, std::vector<data3>) constructor is used whenever data generated by the program needs to be written to a file not yet existing or needs to be distributed. The constructor assigns the members naturally, does not yet add the File object to the fileList_, though. To save the data to a file, printDataToFile() has to be called.

Both constructors copy the provided data, either from the source file or the "source variable" – hence, the Data object is always the owner of the data and does not depend on the source after successful data acquisition.

By the getData() getter, a Data object provides the data it received as a constant reference. Thus, the data is not copied but only read by the receiving class. This prevents unnecessary copying and allows one Data object to make its data available to multiple other components without the danger of data corruption. For the same reason, care must be taken when a Data object gets destructed. This might lead to invalid references in the current version.

3.4.5. RandomNumbers class

The RandomNumbers class is part of the utility classes and an abstract (private constructor), static class which provides pseudo-random numbers to other classes. It is defined in the files utility.h and utility.cpp respectively.

```
class RandomNumbers {
private:
    static std::mt19937 generator_;

    RandomNumbers() {}

public:
    static int getNewSeed();
    static bool testSeeder();
    static void seed(int seed);

    static void getTestData (File file, int noValues);

    static double randomValue();
    static double normalDistValue (double mean, double stddev);
};
```

Together with the Model and the Simulation class, the RandomNumbers class could be considered the heart of the framework. Its very profound task is to provide pseudo-random numbers originating from a certain distribution, namely the normal distribution. To do this, it uses a static instance of Mersenne Twister engine (described in section 3.3.1 Random numbers – Mersenne Twister engine (p. 52)) and the std::normal_distribution class template (described in section 3.3.2 Gaussian normal distributed values (p. 53)), which are both part of the standard library. Pseudo-random number generators (P-RNGs) always provide a repeating series of numbers. The quality of the engine determines the non-repetitive length of the series. The so-called "seed" is the entry point in the series. Thus, the class also provides a mechanism to seed the P-RNG.

A couple of static fields manage the random number generation.
std::mt19937 generator_ is the Mersenne Twister, defined in the <random> library. In order to generate pseudo-random numbers it needs to be seeded. This mechanism is implemented in the getNewSeed() method whose return value can be passed to the generator_ by the seed() method for initialization. It uses static variables inside the function to check whether the random devices have been initialized already and to output user information. The std::default_random_engine will be used for seeding. Since any non-random seeder will produce the same series of numbers with every start of the application, it itself needs to be seeded. This is done by the std::chrono class from the <chrono> library.

A random value can be obtained by calling randomValue().
As described in section 3.3.2 Gaussian normal distributed values (p. 53), the random numbers should originate from a normal distribution, though. The static normalDistValue() method uses the P-RNG and the std::normal_distribution template to return a random number originating from the distribution defined by mean and stddev. How this is achieved is covered in the linked section as well. For more details refer to the source code.

Upon initialization, the seed generation will be tested by the testSeeder() function, which compares two consecutive return values of getNewSeed() – if they are equal, it reports an error.

Because of its static public design, any class can use pseudo-random numbers without the need to introduce this functionality on its own.

3.4.6. Analysis class

The Analysis class is part of the utility classes and an abstract base class which must be inherited in order to implement an analysis method. It uses all the other utility classes. It is defined in the files utility.h and utility.cpp respectively.

```
//from utility.h
class Analysis {
protected:
    const File fileSource_;
    const FileDoc fileDocSource_;
    File file_;
    FileDoc fileDoc_;
    const std::vector<data3>& data_;

    std::vector<data3> result_;

    const std::string analysisName_;
    Analysis(const Data& data, std::string analysisName);

public:
    bool printToFile();
    File getFile() const {return file_.getFile();}
```

```
        Data toDataObject() const;
    };
```

Every Analysis object (every child, to be more exact) needs data to work with. This data will be provided by a Data object passed to the constructor. It assigns the constant reference to access the data and makes constant copies of the source File and FileDoc (members of the Data instance) objects to the corresponding member fields in order to preserve them for later (unmodified) use. Additionally, the fileSource_ will be copied to the file_ member, since the resulting file name after analysis should just be a modification of the source file name.

The reason for the FileDoc separation on the other hand is that the child classes may need to reset their own file documentation during their life time. If the old documentation was only included in the same non-constant member, they could delete the source documentation in the course of updating their own!

The analysisName_ member, set by the corresponding constructor argument, gives the name of the implemented analysis method which is used for documentation later.

The results of the analysis will be saved in the result_ vector, which can be saved by the Analysis::printToFile() method. It combines the old fileDocSource_ and the new fileDoc_ file documentations while adding basic contents (source file and analysis method name) before passing it to Data::printDataToFile(File, FileDoc, std::vector<data3>). The necessary file_ modification is task of the child's methods, of course, same as any additional documentation.

If one source signal should undergo several different analyses in a row, when not all generated data needs to be saved for later access or when one wants to avoid to multiply read data from a file which is already present in the running application, the toDataObject() method will create a Data object containing the result_ data and the prepared File and FileDoc as described for the printToFile() method. This avoids the slow file read-out.

If this method is used, it is important to assign to a variable in the same scope.

```
//pseudo-code class definitions; both shall feature a performAnalysis() method
class AnalysisONE : public Analysis;
class AnalysisTWO : public Analysis;

//usage
File file(⟨source file⟩);
Data source(file);          //receive source data from a source file

AnalysisONE one(source);    //provide source data
one.performAnalysis();      //do stuff
one.printToFile();          //save the results

//instead of (slow file read): Data data(one.getFile()):
Data data(one.getAnalysisDataObject());

AnalysisTWO two(data);
two.performAnalysis();
⟨...⟩
```

Mind that a construction like

```
AnalysisTWO two(one.getAnalysisDataObject());   //bad!
```

will lead to a segmentation fault during run-time, since the Data object passed as a parameter will only be available during this function call but not for any later access!

3.4.7. `Sample` class

The `Sample` class is part of the input data classes and depicts a radioactive nuclide. It is defined in the files `inputdata.h` and `inputdata.cpp` respectively.

```
//from inputdata.h
class Sample {
private:
    FileDoc fileDoc_;

    std::string name_;
    double n0_;
    double halflife_;
    double decayconst_;

public:
    Sample(std::string name, double n0, double halflife_inSeconds);
    ~Sample() {}

    FileDoc getFileDoc () const {return fileDoc_;}

    //getter
    double getN0 () const {return n0_;}
    double getHalflife () const {return halflife_;}
    double getDecayconst () const {return decayconst_;}
};
```

The class is pretty much self-explaining and straightforward. Like every other class it needs to provide documentation; this is handled by the constructor:

```
//from inputdata.cpp
Sample::Sample (string name, double n0, double halflife_inSeconds)
: name_(name), n0_(n0), halflife_(halflife_inSeconds)
{
    decayconst_ = log(2)/halflife_inSeconds;

    fileDoc_.addCommentTitle("Sample");
    fileDoc_.addCommentKeyVal("name", name_);
    fileDoc_.addCommentKeyVal("N(t=0)", n0_);
    fileDoc_.addCommentKeyVal("half life (s)", halflife_);
    fileDoc_.addCommentKeyVal("decay constant (1/s)", decayconst_);
}
```

3.4.8. `Parameter` class

The `Parameter` class is an abstract base class and part of the input data classes and collects the parameters for a specific physical model. It is defined in `inputdata.h`.

```
class Parameter {
protected:
    FileDoc fileDoc_;

    Parameter() {}
    ~Parameter() {}

public:
    FileDoc getFileDoc() const {return fileDoc_;}
};
```

This class is barely more than the shadow of a POD class. It essentially just enforces documentation and its children will add member fields for the necessary parameters, adding documentation to the `fileDoc_` member and implementing simple getter methods. Mind that this documentation should only provide the units and values of the parameters by calling

FileDoc::addCommentKeyVal() (in the constructor) – the description of the *meaning* of the parameters is task of the Model classes.

In general, the Parameter classes hold all parameters that describe the effect in question, especially those which can be evaluated by the experimenter. In some cases, there might be other parameters whose values can only be defined in conjunction with the engaged sample. Such quantities will be part of the respective Model class for conceptual (i.e. the quantity is an attribute of the sample) and practical (i.e. parameters of the sample are needed for calculation) reasons. The 2.2.4 Sinusoidal, time dependent decay constant (p. 22) model with its fundamental parameter a is an example for such a case.

3.4.9. Conversion functions

Together with the input data classes comes a set of conversion functions, collected in a namespace called conv. It seeks to facilitate the usage of the other components and the handling of all the parameters in use which oftentimes are of such different magnitude. Implemented are:

```
//from inputdata.h
namespace conv {

int yearsToSec(double years);
int weeksToSec(double weeks);
int daysToSec(double months);
int hoursToSec(double hours);
double perToFrequ(double periodicity_in_seconds);       // cycle dur. (s) to angular frequ. (rad/s)
double frequToPer(double frequency_in_radOverSeconds);// angular frequ. (rad/s) to cycle dur. (s)
double DCtoHL(double decayconst_in_oneOverSeconds);    // decay constant (1/s) to half-life (s)
double HLtoDC(double halflife_in_seconds);             // half-life (s) to decay constant (1/s)

}
```

The ⟨...⟩ToSec() functions return integer values, which already illustrates the paradigm of using seconds as the smallest unit of time in the simulation. These conversion functions call each other, round their calculated value via std::round() defined in <cmath> and type-cast it to an integer before returning it. Only the yearsToSec() function uses the following definition: $1\,\text{a} = 365.24219878\,\text{d}$. [23]

3.4.10. Model class

The Model class is an abstract base class and defines the implementation of a physical model used as a basis for signal generation. It is defined in the files model.h and model.cpp respectively.

```
//from model.h
class Model {
protected:
    FileDoc fileDoc_;

    std::string name_;
    std::string descr_;

    std::string formulaRem_;
    std::string formulaDeced_;
    std::string formulaDecing_;

    Sample sample_;

    Model (const Sample& sample);
    ~Model () {}

    void aboutBasic();
    virtual void setStrings () = 0;

public:
    virtual void aboutDetailed () = 0;
    FileDoc getFileDoc() const {return fileDoc_;}

    virtual double calculation (CalcType type, const int t) = 0;
};
```

The most important part of the class is the pure virtual `calculation()` function, implementing the basic formulas (I) to (III), which are unique for every model and thus every derived child class. Choosing between the equations by the `CalcType` enumeration, and providing an integer time argument `t`, this function returns the corresponding theoretical value. It is implemented via a switch-case control sequence. The definition of the enumeration reads:

```
//from model.h
enum class CalcType {
    remaining,
    decayed,
    decaying
};

std::string getCalcType (CalcType type);
```

and it includes a string representation function declaration for documentation purposes.

For standard decay one would implement the `calculation()` function in the following way:

```
//from model.cpp
double MExponential::calculation (CalcType type, const int t) {
    double n0 = sample_.getN0();
    double lambda = sample_.getDecayconst();

    switch (type) {
    case CalcType::remaining:
        return n0 * exp(-lambda * t);
    case CalcType::decayed:
        return n0 * ( 1 - exp(-lambda * t) );
    case CalcType::decaying:
        return lambda * n0 * exp(-lambda * t);
    default:
        return -666;
    }
}
```

Note that before the switch-case query all used variables are locally stored.[36] At the same time, they are renamed to resemble the symbols in the equations so that comparison and error checking becomes more intuitive; a very recommended approach, because their original meaning gets clear thanks to their assignment anyway.

Apart from the self-explaining members `fileDoc_`, `name_` and `descr_`, this parent class contains three other string variables (`formula`$\langle X \rangle$`_`), which are assigned the string representations of the basic formulas (I) to (III) to be included in the documentation. For this purpose, the child class implements the pure virtual `setStrings()` function – in which *all* string members are set – to keep the constructor free from clutter. For example

```
//from model.cpp
void MExponential::setStrings() {
```

[36] This might be a place for performance optimization, since this function will be called very often by the `Simulation` class.

```
        name_ = "Exponential decay";
        descr_ = "This model describes the standard exponential decay.";
        formulaRem_ = "N(t) = N_0*exp(-lambda*t)";
        formulaDeced_ = "N_d(t) = N_0*(1 - exp(-lambda*t))";
        formulaDecing_ = "n(t) = lambda*N_0*exp(-lambda*t)";
    }
```

for standard decay. It is advised to choose variable names like `lambda` and `omega` in the formula strings instead of `decayconst` and `frequency` as in the class definitions. Whereas in the latter case it is important to be informed about the *meaning* of the symbols (hence the naming), the purpose of the former is to quickly transcribe the equations to paper to study them.

The base class method `aboutBasic()` is already implemented and adds the strings defined via the `setStrings()` function to the `fileDoc_` in a consistent manner.

The pure virtual method `aboutDetailed()` is implemented by the child class and contains a call to `Model::aboutBasic()` and a description (including dimensions) of the introduced model parameters (unlike the corresponding `Parameter` class which adds the concrete values and units). Here is one example from a non-standard model:

```
    void MSineSummand::aboutDetailed () {
        Model::aboutBasic();      //include basic model info; calls setStrings() of child

        //add child-model-specific info
        fileDoc_.addComment("\tparameters:");
        fileDoc_.addComment("\t\tamount (epsilon), [dimensionless] --- describes the ⟨...⟩");
        fileDoc_.addComment("\t\tinfluence (eta), [dimensionless] --- defines the ⟨...⟩");
        fileDoc_.addComment("\t\tperiodicity (T), [time] --- is the periodicity of ⟨...⟩");
        ⟨...⟩

        //add sample and parameter info afterwards
        fileDoc_ += sample_.getFileDoc();
        fileDoc_ += parameters_.getFileDoc();

        return;
    }
```

As the example shows, the `aboutDetailed()` method also includes the `FileDoc` info of the `Sample` and an inherited `Parameter` member. The latter is included in the child class. Since a child of a `Model` can only work with one specific child of `Parameter`, the `Model` parent class does *not* include a generic parent class `Parameter parameter_;` member field.

3.4.11. `Simulation` class

The `Simulation` class provides the means to generate a theoretical or simulated signal (as a real measurement would provide it), according to the chosen model, which may introduce standard or non-standard physics. It is defined in the files `simulation.h` and `simulation.cpp` respectively.

```
    //from simulation.h
    class Simulation {
    private:
        File file_;
        FileDoc fileDoc_;

        int deltaT_;
        int timeTot_;

        Model& model_;

        std::vector<data3> result_;
```

```
public:
    Simulation(Model& mod, File file, int dt_inSeconds, int t_inSeconds);
    ~Simulation() {}

    void simulate(bool simulate, int seed = 0);
    void justPlot(CalcType type, bool simulate, int seed = 0);

    bool printToFile();

    File getFile() const {return file_.getFile();}

    Data toDataObject() const;
};
```

By using the `RandomNumbers` and `Model` classes, the `Simulation` class enables the user to generate signals equal to those an experimenter could find when performing a measurement.

It uses the facilities of the `RandomNumbers` class and documents the results of its P-RNG tests.

The class has a `model_` member defining the physical equations that will be used for simulation. By design, it includes all necessary data and documentation. Mind that `Model&` is compatible with every child of `Model`, which is the conceptual motivation of the separation of the `Simulation` and the `Model` class and deriving from the latter to implement different physical models.

There are two members corresponding to Δt and $\mathscr{T}$, namely `deltaT_` and `timeTot_` respectively. They are of `int` type, which makes time in the simulation "quantized". The unit of time throughout the simulation is seconds. `timeTot_` will be truncated to a whole-number multiple of `time_` by an integer division and "re-multiplication".

Similar to the `Analysis` class, the `Simulation` class features a `std::vector<data3> result_` vector which will hold the generated signal, a `printToFile()` method, which will use the interface supplied by the `Data` class to save the generated signal to a file and a `toDataObject()` method for the same reason as presented in the context of the `Analysis` class. Both functions will combine the `model_`'s and the `Simulation` object's `FileDoc` instances. They need to be kept separate, otherwise the `Simulation` object's documentation could not be updated throughout its lifetime.

The `simulate()` method performs the actual signal generation and simulates a real measurement (i.e. uses random numbers), if the parameter `simulate` is true. Otherwise it uses the statistical mean values, which means that it actually just plots the graph of (IVb). Because of the findings in 3.3.3 Types of simulating (p. 53), this is the only way of signal generation provided.

If `simulate` is true and `seed` is 0, the seeding mechanism of `RandomNumbers` will be used to kick off the P-RNG. If `seed` is different from 0, this value will be used to seed the engine, which makes every simulation run exactly reproducible.

The `justPlot()` method works almost exactly the same as the `simulate()` method but includes an additional function parameter of type `CalcType`. The function is used to generate data containing values that would just plot equations (I) to (III) depending on the `calc` argument. In case of very large values `timeTot_` ($\mathscr{T}$), the method restricts itself to 10 000 values.

Both methods, `simulate()` and `justPlot()`, clear the `result_` vector when being called in order to allow for multiple simulation runs with different simulation settings (but constant `time_` and `deltaT_`).[37] In order to keep every result, the `printToFile()` or `getSimulationDataObject()`

[37] This is because these two parameters are not included in the file name modifications, thus changing just the two would result in file overwrites.

methods have to be called before every new call of `simulate()` or `justPlot()`.

They also reset and then update the `file_` and `fileDoc_` members accordingly. If the `simulate` argument is true, they add the post-fix `simulated`, otherwise `theory`. Additionally, the `justPlot()` method adds the string representation of the chosen `CalcType` as an additional post-fix.

3.5. Implementation of the analysis classes

This section explains how the methods presented in 2.4 Mathematical analysis tools (p. 37) are implemented by inheriting from the abstract `Analysis` base class documented in section 3.4.6 `Analysis` class (p. 59).

3.5.1. `Autocorr`(elation) class

The `Autocorr` class performs the autocorrelation of a signal. It is defined in the files `autocorrelation.h` and `autocorrelation.cpp` respectively. Its mathematical description can be found in 2.4.1 Autocorrelation (p. 37).

```
//from autocorrelation.h
class Autocorr : public Analysis {
private:
    int noTotal_;
    int noUsed_;
    int lagMax_;

    void initializeParameters();

    double autocorrelationValue (AucoType direction, int lag);

public:
    Autocorr (const Data& data);
    void autocorrelation (AucoType direction, bool normalize = true);
};
```

`noTotal_` is the total number of data points M_{tot} provided by the parent object, `noUsed_` is the number of data points M used in every step of the integration and `lagMax_` is the maximum lag $\hat{\tau}_{\text{max}}$ available in the resulting autocorrelation function.

The values for the member variables are calculated according to (2.63) and (2.64) by the `initializeParameters()` method which is called by the constructor.

The `autocorrelationValue()` method returns the autocorrelation function value $\Psi(\hat{\tau})$ for a given lag $\hat{\tau}$ using an implementation of (2.60).

`autocorrelation()` loops through all possible lags (calling the `autocorrelationValue()` method) and adds the results to the `Analysis::result_` vector, which is the autocorrelation function table $\Psi(\hat{\tau})$. If the parameter `normalize` is set to true, the result will be (2.61), the normalized autocorrelation function $\psi(\hat{\tau})$.

The enumeration `AucoType`

```
//from autocorrelation.h
enum class AucoType {
    forwards,
    backwards
};

std::string getAucoType (AucoType type);
```

allows the user to choose the direction in which the summation should be performed in `autocorrelationValue()`. Since the function values for small lags are significantly higher, integration back to front might minimize numerical summation errors. Although not dedicatedly investigated it seems that both methods yield the same results.

The first column of the `Analysis::result_` holds the dimensionless lag $\hat{\tau}$. In order to compute the corresponding periodicity T of a periodic autocorrelation function, a multiplication with Δt (evident from the file documentation) according to (2.62).

An automated lag estimation algorithm is not implemented and not trivial, since in some cases the autocorrelation function will show an exponential decrease. In addition to the statistical fluctuation and small amplitudes of the possible effect, this shape of the curve makes it hard for an automated algorithm to find the periodicity, although it might be obvious to the human eye.

For quantitative analysis one should probably restrict oneself to autocorrelation signals with no exponential contribution.

Generally, fitting by using starting values gained from an analysis “by hand”, could lead to quantitative estimations.

The class adds the prefix `auto` to the file name and, depending on the chosen options, concatenates `N` for a normalized autocorrelation and `Backw` in case of backward summation.

3.5.2. `StdDecaySub`(traction) class

The `StdDecaySub` class performs the standard decay subtraction of a signal. It is defined in the files `stddecaysub.h` and `stddecaysub.cpp` respectively. Its mathematical description can be found in 2.4.2 Standard decay subtraction (p. 41).

```
//from stddecaysub.h
class StdDecaySub : public Analysis {
public:
    StdDecaySub(const Data& data);

    void standardDecaySubtraction(const double decayconst_inOneOverSec);
    void standardDecaySubtraction(const double n0,
                                  const double decayconst_inOneOverSec);
    void old_standardDecaySubtraction();
};
```

This class is fairly simple. It computes the subtracted signal according to (2.68). In its discrete form, the formula reads

$$\Delta N_{\text{sub}}(t_{i+1}) = \Delta N(t_i;\Delta t) - \Delta N_{\text{std}}(t_i;\Delta t)$$

since the data is provided in tuples $\left(t_{i+1},\ \Delta N(t_i;\Delta t),\ \sqrt{\Delta N(t_i;\Delta t)}\right)$ and the subtraction of $\Delta N_{\text{std}}(t_i;\Delta t)$ allows to factor the term out as in (refer to 2.4.2 Standard decay subtraction (p. 41))

$$\underbrace{N_0\,\mathrm{e}^{-\lambda t_i}\cdot\Big(\dots\Big)}_{\Delta N(t_i;\Delta t)} - \underbrace{N_0\,\mathrm{e}^{-\lambda t_i}(1-\mathrm{e}^{-\lambda\Delta t})}_{\Delta N_{\text{std}}(t_i;\Delta t)} \ .$$

Depending on the chosen function overload, the value for N_0, which is needed for the calculation, is either passed to the method by the user (if N_0 is known or taken from a fit) or computed automatically. In the latter case, the method calculates N_0 according to (2.47) for all available data points and uses the arithmetic mean value for the calculation of the standard decay values, which are used for subtraction.

The reference point t_0 – and the resulting N_0 – is of course at the start of the measurement.

The `old_standardDecaySubtraction()` method implements the original (deprecated) subtraction method presented later in 4.2.1 Increasing sensitivity by standard decay subtraction/normalization (p. 77) for compatibility reasons.

Depending on the chosen method, the class adds the prefix `subGetN0`, `subCalcN0` or `subOLD` to the file name.

The uncertainty propagation as given in the respective theory chapter is not yet implemented in the framework and needs to be tested.

3.5.3. `StdDecayNorm`(alization) class

The `StdDecayNorm` class performs the standard decay normalization of a signal. It is defined in the files `stddecaynorm.h` and `stddecaynorm.cpp` respectively. Its mathematical description can be found in 2.4.3 Standard decay normalization (p. 43).

```
//from stddecaynorm.h
class StdDecayNorm : public Analysis {
public:
    StdDecayNorm(const Data& data);

    void standardDecayNormalization(const double decayconst_inOneOverSec);
    void standardDecayNormalization(const double n0,
                                    const double decayconst_inOneOverSec);
};
```

This class is fairly simple. It computes the normalized signal according to (2.69). In its discrete form, the formula reads

$$\Delta N_{\text{norm}}(t_{i+1}) = \frac{\Delta N(t_i;\Delta t)}{N_{\text{std}}(t_i)} \quad ,$$

since the data is provided in tuples $\left(t_{i+1},\ \Delta N(t_i;\Delta t),\ \sqrt{\Delta N(t_i;\Delta t)}\right)$ and only the division by $N_{\text{std}}(t_i) = N_{\text{std}}(t_{i+1} - \Delta t)$ allows to cancel the prefactor as in (refer to 2.4.3 Standard decay normalization (p. 43))

$$\frac{N_0\, \mathrm{e}^{-\lambda t_i} \cdot \left(\dots\right)}{N_0\, \mathrm{e}^{\lambda t_i}} \quad \begin{array}{l} \big\}\, \Delta N(t_i;\Delta t) \\ \big\}\, N_{\text{std}}(t_i) \end{array} \quad .$$

Depending on the chosen function overload, the value for N_0, which is needed for the calculation, is either passed to the method by the user (if N_0 is known or taken from a fit) or computed automatically. In the latter case, the method calculates N_0 according to (2.47) for all available data points and uses the arithmetic mean value for the calculation of the standard decay values, which are used for normalization.

The reference point t_0 – and the resulting N_0 – is of course at the start of the measurement.

Depending on the chosen method, the class adds the prefix `normGetN0` or `normCalcN0` to the file name.

3.5.4. `Smooth` class

The `Smooth` class tries to even out the statistical fluctuations of a given signal. It is defined in the files `smooth.h` and `smooth.cpp` respectively. Its mathematical description can be found in 2.4.4 Smoothing and filtering (p. 44).

```
class Smooth : public Analysis {
private:

public:
    Smooth(const Data& data);

    std::vector<double> kernel_;
```

```
        void printCurrentKernelTable(File file);

        void setKernelGauss(int sigma_inUnitsDeltaT, int range_inUnitsSigma);
        void setKernel⟨OtherKernel⟩(⟨...⟩);

        bool smooth();
    };
```

This class convolves the input signal with a given kernel, thus providing a corresponding weighed average. The member `kernel_` holds the look-up table of the chosen kernel. It is set by calling one of the `setKernel⟨...⟩()`. Every one of them should reset the current state of `Analysis::fileDoc_` and add its own documentation. The `printCurrentKernelTable()` method prints the current kernel look-up table to a given file and adds the current contents of `fileDoc_`.

The implemented `setKernelGauss()` method computes a simple Gaussian kernel by sampling (2.74). Its standard deviation `sigma_inUnitsDeltaT` ($\sigma_{\Delta t}$) is passed in units Δt, where $1\Delta t$ equals the distance between two data points in a typical signal. The width of the kernel, i.e. how many sigmas width should be taken into account for smoothing, is determined by the `range_inUnitsSigma` (n_σ) parameter. The resulting size of the look-up table is then $2(\sigma_{\Delta t} n_\sigma) + 1$ elements/data points (the plus one is due to the "zero position" of the bell curve). Normalization is achieved by dividing every value in the look-up table by the sum over all its entries!

Care must be taken when choosing these parameters. The convolution can only compute values where the kernel fits in the data completely. This means that the resulting function will be shortened by $\sigma_{\Delta t} n_\sigma$ data points on each side. If there are only few data points available – due to a large Δt or small $\mathscr{T}$ – an unsuitable choice of especially n_σ can render the analysis impossible. A value of $n_\sigma = 2$ seems proper in almost all cases. $\sigma_{\Delta t}$ must be chosen according to the desired grade of smoothing.

New smoothing kernels may be added to this class at any time. In that case mind the resulting `setKernel⟨...⟩()` methods would also need to assign different `file_` prefixes to avoid overwrites.

It adds the prefix `smooth`$\langle\sigma_{\Delta t}\rangle$`x`$\langle n_\sigma\rangle$ to the file name.

3.5.5. `Accumulation` class

The `Accumulation` class adds up multiple data points of a given file. This reduces the number of available points but also decreases the relative error of each data point. It is defined in the files `accumulation.h` and `accumulation.cpp` respectively. Its mathematical description can be found in 2.4.5 Accumulation (p. 46).

```
class Accumulation : public Analysis {
public:
    Accumulation(const Data& data);

    void accumulate(int accumulate_inUnitsDeltaT);
};
```

The principle and the implementation of this class are pretty straightforward: The parameter n = `accumulate_inUnitsDeltaT` defines how many data points (spaced in steps Δt) shall be summed up to one new resulting point. The class checks if any data points at the end will be dismissed as a consequence and informs the user.

It adds the prefix `acc`$\langle n\rangle$ to the file name.

3.6. Using and extending the framework

This section will briefly describe how to implement new physical models, new analysis classes and provide a check list. Also, the code of the standard decay model is presented and as well as a general code example.

Knowing the structure of the simulation and how the different classes are meant to interact as described in sections 3.2 Structure (p. 47) and 3.4 Implementation of the framework (p. 54) will render the actual usage and extension of the framework pretty straightforward. It may help to follow the implementation of an already existing model while reading this explanations. Mind also the established coding style described in 3.2.3 Style choice, naming conventions and coding standards (p. 50).

3.6.1. Guide to implement a new model

The models implemented in the course of this work are all defined/implemented in the `model.h/*.cpp` files. The respective parameter classes are located in `inputData.h/*.cpp`. As long as the file structure of the simulation is not changed, it stands to reason to add new models in these files.

- Implement the parameter class for the new model by inheriting from `Parameter` and adding all necessary model parameters as member variables and provide getter methods. Mind that these do not include sample parameters or quantities that follow from the combination of model and sample parameters (such as a in the sinusoidal decay constant model), which will be part of the model class.

- In the constructor perform basic error checking, assign the parameters and use the parent's `fileDoc_` object for proper documentation of the actual values (using `FileDoc::addCommentKeyVal()`. Take care to ask the user for only a minimum number of intuitive input parameters (such as T instead of ω; and definitely not both) and use the conversion functions of the framework to derive other quantities. Add new conversion functions to the framework if necessary.

- Subclass the `Model` parent class for the new model and define a constructor accepting a `Sample` and a newly created parameter object. Add member fields for possible additional parameters and provide getter functions.

- Implement the virtual `setStrings()` method and assign the parent member strings: `name_` and `descr_` (description) plus the formulas `formula`⟨...⟩`_` like `"N(t) = N_0*exp(-lambda*t)"`.

- Implement the virtual `aboutDetailed()` method: First, this method must call the parent's `aboutBasic()` method, which provides standard documentation. Then use the parent's `FileDoc` object to add a qualitative explanation of all parameters via `FileDoc::addComment()`. Finally, append the `FileDoc` members of the `sample_` and the `parameters_` object as in `fileDoc_ += sample_.getFileDoc()` etc. This provides decent documentation (basic model description, parameter description, actual values for all parameters).

- In the `Model` child class constructor just assign the members and call the `setStrings()` and the `aboutDetailed()` methods.

- Now implement the parent's `calculation()` function by using a switch-case query for the three different calculation types `CalcType`. It may be helpful to define dummy variables like `double lambda = sample_.getDecayconst();` etc. in order to increase readability in the implementation of the formulas.

3.6.2. Guide to implement a new analysis method

Each analysis class is implemented in its own translation unit - create new header and implementation files.

- Define the new analysis class by deriving the `Analysis` parent. Create a constructor accepting only the parameters demanded by the parent constructor. Basic documentation is performed by the parent constructor automatically. The constructor should *not* be used for any class managing purposes.
- Define all necessary member fields and methods to provide the desired functionality. Take care to provide decent user feedback and documentation! Unless the analysis method is very simple, use dedicated methods to gather and process user input (like picking a kernel in the `Smooth` class). Such functions then also deal with file documentation (`fileDoc_`) and output file naming (`file_`); always reset those objects before assigning values so that the same analysis object can be used for multiple sets of input parameters. Pick file name prefixes in the style of the already existing analysis classes and add the most important values to make the files distinguishable.

It is advised to skim through the code of the analysis classes already implemented in order to see how different design problems are tackled.

3.6.3. Benefits of the design

This seemingly pedantic coding style and design provides major advantages and should therefore be pursued dedicatedly.

Decent user feedback will make the procedures and progress obvious to the user. This will also help to identify errors in the implementation or usage.

The ample documentation effort results in a rich data file preamble collecting all necessary information about the data in the file. Thus, every single file will provide comprehensive information about the model and parameters in use and about all steps of processing that were applied to the data.

The file name conventions allow the user to draw the most important information about the current data from the file name itself. For instance, the file name

`autoN_normCalcN0_acc336_A_simulated.dat`

will tell the user that the data set `A` was simulated using the random number engine. The original signal was accumulated with $n = 336$ and then normalized where N_0 was computed from the data. The current file contains the normalized autocorrelation of said data.

3.6.4. Code example

The following code example shows how one can generate a simulated measurement signal, smooth it and then perform an autocorrelation analysis.

```
//INPUT DATA
//create base file
File file("./data/", "example", "dat");

//create a sample
double N0 = 1E9;
double halflife = conv::yearsToSec(10.54);
Sample ba133("Ba-133", N0, halflife);
```

```
//create a parameter set
double eps = 1E-4;
double perio = conv::yearsToSec(1.0);
PSineLambda param(eps, perio);

//create model
MSineLambda model(ba133, param);

//measurement parameters
int deltaT = conv::daysToSec(1);
int tTotal = conv::yearsToSec(4);


//SIMULATION
//create simulation
Simulation sim(model, file, deltaT, tTotal);
sim.simulate(true);
sim.printToFile();


//ANALYSIS
//load data of simulation
Data dataSim(sim.getFile());     //reads from file
//or
Data dataSim = sim.toDataObject();

//normalize signal
StdDecayNorm norm(dataSim);
norm.standardDecayNormalization(sample.getDecayconst());
norm.printToFile();

//smooth the normalized signal
Data normData(norm.getFile());     //reads from file
//or
Data normData = norm.toDataObject();
Smooth smoothNorm(normData);
smoothNorm.setKernelGauss(14, 2);     //σ_Δt = 14Δt = 14 d
smoothNorm.smooth();
smoothNorm.printToFile();

smoothNorm.setKernelGauss(28, 2);     //σ_Δt = 28Δt = 28 d
smoothNorm.smooth();
smoothNorm.printToFile();             //will be printed to a new file

//autocorrelation of smoothed normalized signal
Data smoothNormData = norm.toDataObject();  //or...
Autocorr autoSmoothNorm(smoothNormData);
autoSmoothNorm.autocorrelation(AuCoType::forwards, true);
autoSmoothNorm.printToFile();
```

4. Application

This section will cover the findings gained by the usage of the methods presented in 2 Physics and math (p. 15) and the simulation and analysis framework pictured in 3 Simulation (p. 47). It will evaluate the "pros and cons" of the introduced tools and try to derive "rules" for their usage. The goal is to work out how the different methods and parameters intertwine and how these insights might guide the ambitious experimenter when designing an experimental setup dedicated to track down effects similar to what is covered in this thesis and beyond.

By the very nature of the problem, all investigations and solution approaches are more or less reverse-engineered. In order to test them against the influence of bias, a series of blind tests is performed. Additionally, the nature of data originating from simulations of *standard* decay has to be studied thoroughly before jumping to any conclusions about an ostensibly visible, unknown effect.

4.1. Summary of the theoretical and mathematical findings

This section is a direct reference to the contents of chapter 2 Physics and math (p. 15) and will gather the most important presented issues and draw theoretical/mathematical conclusions which will help to learn about the relations between the manifold of (free) parameters and physical values.

4.1.1. The model of choice

Two models to introduce a sinusoidal oscillation in the standard equation of radioactive decay were established in sections 2.2.3 Sinusoidal, time dependent summand (p. 20) and 2.2.4 Sinusoidal, time dependent decay constant (p. 22).

The sinusoidal summand model was a blatant and physically "unmotivated" approach in order to achieve the desired behavior and could be understood very easily. It was used to establish first results, to test analysis methods (as pointed out in the later section 4.2.1 Increasing sensitivity by standard decay subtraction/normalization (p. 77)) and to "get a feeling" for the effect in question.

Its evolution after developing some understanding for the nature of such signals lead to the sinusoidal decay constant model, which introduced the sinusoidal modification in a physically reasoned way, namely an oscillation inherent with the decay "constant" of the sample as in (2.19), which also leads to a new differential equation. Where the effect's amplitude ε was the main parameter in the first model, its role as a (merely) effective quantity detectable by an experiment became clear in the second, in which the profound variable is, in fact, a, describing the variation of the decay constant itself. The cause of such a quantity is entirely unclear, but, according to this model, the connection between ε (the amplitude of the oscillation in the measurement signal) and a is given by (2.30).

This discussion clearly concludes the sinusoidal decay constant model to be the model of choice to describe radioactive decay with a sinusoidal influence.

As demonstrated in the sections linked above, Taylor expansion shows that both models are equivalent in the first order. This, in retrospect, justifies the analyses performed by using the sinusoidal summand model.

4.1.2. A set of rules

After receiving a set of data through measurement (or simulation), our primary goal is to find a recurring oscillation of cycle duration T (and thus frequency ω according to (2.9)) in the signal.

For an estimated cycle duration and magnitude of the effect

$$T := \text{const} \quad \text{and} \quad \varepsilon := \text{const} \ ,$$

we can find restrictions for other parameters.

Because of (2.39), we automatically obtain an upper limit for integration time Δt, namely

$$\Delta t \ll T/2 \ .$$

Assuming ε to be the amplitude of the effect (as in the sinusoidal summand model for instance), a maximum relative error $\delta_{\Delta N} \leq \alpha$ is required in order isolate the effect in the data,

$$\delta_{\Delta N} \leq \alpha = \alpha(\varepsilon) \ ,$$

where $\alpha = \alpha(\varepsilon)$ directly depends on the effect's amplitude (both are relative values) and $\alpha < \varepsilon$.

Equation (2.57) reminds us of the fact that a desired accuracy of α yet demands a minimum count number ΔN in accordance with

$$\Delta N \geq \frac{1}{\alpha^2} \ ,$$

which must be true during the whole measurement time $\mathscr{T}$. Since the relative error rises over time (because $N(t)$ drops), this is guaranteed if (refer to (2.8))

$$\Delta N(\mathscr{T} - \Delta t; \Delta t) \geq \frac{1}{(c\varepsilon)^2} \ . \tag{4.1}$$

Mind (2.65), which imposes a *lower* limit for the measurement time

$$\mathscr{T} > 2T \ .$$

If we could simply choose a sample to fulfill the stated estimations, we would require (rearranging (4.1) after plugging in (2.8)) an initial number of nuclei (for $\lambda = \text{const}$)

$$N_0 \geq \frac{1}{\alpha^2} \frac{e^{\lambda \mathscr{T}}}{e^{\lambda \Delta t} - 1} \ .$$

An analogous solution for λ is not analytically possible. For a given N_0 or total measurement duration $\mathscr{T}$, we can solve (4.1) for Δt and obtain, assuming $\mathscr{T} - \Delta t \approx \mathscr{T}$,

$$\Delta t \geq \frac{1}{\lambda} \ln\left(1 + \frac{e^{\lambda \mathscr{T}}}{(\alpha^2 N_0)}\right) \ .$$

Mind that, taking (2.59) into account, we actually obtain

$$\mathcal{N}_0 \geq \frac{N_0}{f(E,d)}$$

in all the relations above for the sample in use!

Number of counts

To raise the number of counts to the value demanded by the preceding conditions (in case of a given sample where N_0 and λ are not free to choose), we can increase Δt because of (IV) as long as (2.39) is not violated. Additionally, we can decrease the distance d between the detector and the sample to increase the count rate in accordance to (2.58).

Notes
This set of rules was established for a given cycle duration T and effect magnitude ε but is not limited to these assumptions. All formulas can be rearranged to fit any other set of "starting parameters". For example, considering ε unknown, all relations could be used to find out which magnitude of a possible effect could theoretically be observable for given values of N_0, λ, c, ...

In section 4.2.5 Effect magnitude and uncertainty estimation (p. 90) the relation between measuring accuracy and the magnitude of the effect will be further discussed.

4.1.3. Magnitudes and values of introduced variables; exemplary case

Discussion of the parameters of the effect
The distance to the sun (as a possible parameter for the sun-induced influence) is twofold. We have to distinguish the contribution of the earth's orbit around the sun from the rotation of the earth itself. The distance r can be the parameter itself or it can be parameterized via time $r = r(t)$.

The distance between Earth and Sun varies between approximately $152 \cdot 10^6$ km (aphelion) and $147 \cdot 10^6$ km (perihelion) [17] – on average $r_0 \approx 1\,\mathrm{AU} \approx 150 \cdot 10^6$ km. The spatial parameter of the first component is therefore $r_{\mathrm{orbit}} \approx 5 \cdot 10^6\,\mathrm{km} = 5 \cdot 10^9$ m, since only the change of distance may lead to a hypothetical, observable change in activity. The cycle duration is in the magnitude of years or months; $T_{\mathrm{orbit}} \approx 1\,\mathrm{a} \approx 3.15 \cdot 10^7$ s.

The second contribution results in a spacial parameter of the magnitude of the earth's diameter d. The experiment is either $r_0 - d/2$ away from the sun or $r_0 + d/2$, where $r_{\mathrm{earth}} = d \approx 12\,760\,\mathrm{km} \approx 1.3 \cdot 10^7$ m, which is the important magnitude. This variation occurs on a daily basis, hence $T_{\mathrm{earth}} \approx 24\,\mathrm{h} \approx 86\,400$ s.

We obtain $\omega_{\mathrm{orbit}} \approx 2 \cdot 10^{-7}$ rad/s and $\omega_{\mathrm{earth}} \approx 7.3 \cdot 10^{-5}$ rad/s.

Mind that r_{orbit} exceeds r_{earth} by more than two orders of magnitude! The contribution of the earth's rotation shall be neglected in the first approximation, provided that the effect depends on the change of distance only.

contribution	r [m]	T [s]	ω [rad s^{-1}]
orbit	$5.0 \cdot 10^9$	$3.15 \cdot 10^7$	$2.0 \cdot 10^{-7}$
Earth	$1.3 \cdot 10^7$	$8.60 \cdot 10^4$	$7.3 \cdot 10^{-5}$

Naturally, this discussion is a matter of pure speculation. The nature of such an effect is entirely unknown and hypothetical. This was merely an attempt to construct an exemplary case to be used for demonstration purposes somewhat inspired by [1].

Exemplary case: Ba-133
The current setup (described in 5.2 Current setup (p. 105)) uses a Barium Ba-133 sample; thus it will be used as an example. Its attributes are listed according to [24]:

	Ba-133
half-life	$t_{1/2} = 10.540(6)\,\mathrm{a} = 332.61(19) \cdot 10^6$ s
decay constant	$\lambda = 2.0840(12) \cdot 10^{-9}\,\mathrm{s}^{-1}$
number of nuclei (current sample)	$\mathcal{N}_0 \approx 10^{13}$

The value of $\mathcal{N}_0$ was estimated according to the following considerations:

It can be calculated via a provided initial activity of $\mathcal{A}_{\mathrm{init.}} \approx 10\,\mu\mathrm{Ci} = 3.7 \cdot 10^5$ Bq on $t_{\mathrm{init.}} = 01.10.1988$. From then until $t_0 = 01.01.2015$, $\Delta t \approx 8.284 \cdot 10^8$ s passed. Using (2.7) we obtain

$$\mathcal{A}(t_0) = \mathcal{A}_{\mathrm{init.}}\, \mathrm{e}^{-\lambda \Delta t}$$

and then

$$\mathcal{N}(t_0) = \mathcal{A}(t_0)/\lambda \ .$$

We define according to the current reference time t_0: $\mathcal{N}(t_0) \equiv \mathcal{N}_0$ and $\mathcal{A}(t_0) \equiv \mathcal{A}_0$. In this case, we obtain $\mathcal{N}_0 \approx 3.2 \cdot 10^{13} \pm 5.7 \cdot 10^{6}$.

An alternative approach uses one of the latest measurement results: $\Delta N(t_0;\Delta t) \approx 250\,000$ where $\Delta t = 20\,\mathrm{min} = 1200\,\mathrm{s}$ (dead time not taken into account). We can use (2.47) and obtain $N_0 \approx 9.997 \cdot 10^{10} \pm 3.2 \cdot 10^{5}$.

We define according to the current reference time t_0: $N(t_0) \equiv N_0$ and $A(t_0) \equiv A_0$. Note that this approach provides N_0 and A_0 and *not* $\mathcal{N}_0$ and $\mathcal{A}_0$! A realistic combined efficiency $f(E,d) \approx 1\,\%$ would yield $\mathcal{N}_0 \approx 10^{13}$, again, in the same order of magnitude.

General discussion for the present case

If we recall the considerations of the preceding section and follow the established rules we can estimate magnitudes of all involved parameters. The following figures relate to the described Ba-133 sample and the model of the sinusoidal summand (2.2.3 Sinusoidal, time dependent summand (p. 20)) under negligence of the earth's contribution and $f(E,d)$.

Let's summarize the given parameters:

$$\begin{array}{ll} \lambda = 2.084 \cdot 10^{-9}\,\mathrm{s}^{-1} & T = 3.15 \cdot 10^{7}\,\mathrm{s} = 1\,\mathrm{a} \\ \mathcal{N}_0 = 10^{13} \rightarrow N_0 \approx 10^{11} & \omega = 2.0 \cdot 10^{-7}\,\mathrm{rad/s} \end{array}$$

The following conditions apply:

$$\mathscr{T} \geq 2T = 6.3 \cdot 10^{7}\,\mathrm{s} = 2\,\mathrm{a} \quad \text{and} \quad \Delta t_{\max} < \frac{T}{2} = 1.575 \cdot 10^{7}\,\mathrm{s} = 0.5\,\mathrm{a} \ ,$$

where the latter wont ever be transgressed in a real measurement, if T is expected to be in the magnitude of months. Considering $\mathscr{T} = 2\,\mathrm{a} = \mathrm{const}$ for now leaves Δt and the relative error α to be the free parameters.

$$\Delta t(\alpha;\mathscr{T} = 2\,\mathrm{a}) \geq \frac{1}{\lambda} \ln\left(1 + \frac{\mathrm{e}^{\lambda\mathscr{T}}}{\alpha^2 N_0}\right) = \left\{ \begin{array}{ccc} \alpha & \Delta t\ [\mathrm{s}] & \Delta t\ [\mathrm{min}] \\ \hline 0.010 & 54.72 & 0.91 \\ 0.005 & 218.87 & 3.65 \\ 0.001 & 5471.66 & 91.19 \end{array} \right.$$

delivers the minimum Δt required to achieve the demanded accuracy α. Errors in the magnitude of $\alpha = 0.0001$ yield values of $\Delta t = 1\,\mathrm{week}$. Since the half-life of Ba-133 is so long, changes of $\mathscr{T}$ have almost no effect as proven by:

$$\Delta t(\mathscr{T}, \alpha = 0.001) \geq \frac{1}{\lambda} \ln\left(1 + \frac{\mathrm{e}^{\lambda\mathscr{T}}}{\alpha^2 N_0}\right) = \left\{ \begin{array}{cc} \mathscr{T}\ [\mathrm{a}] & \Delta t\ [\mathrm{s}] \\ \hline 1 & 5124.39 \\ 2 & 5472.48 \\ 3 & 5844.22 \end{array} \right.$$

We can also estimate the possible α for a given Δt and $\mathscr{T} = 2\,\mathrm{a}$.

$$\alpha(\Delta t) = \sqrt{\frac{1}{N_0} \frac{\mathrm{e}^{\lambda\mathscr{T}}}{\mathrm{e}^{\lambda\Delta t} - 1}} = \left\{ \begin{array}{cc} \Delta t\ [\mathrm{s}] & \alpha \\ \hline 55 & 0.0100 = 1.00\,\% \\ 10^3 & 0.0023 = 0.23\,\% \\ \Delta t_{\max} & 1.85 \cdot 10^{-5} = 0.00185\,\% \end{array} \right.$$

Figure 6 shows the presented connections.

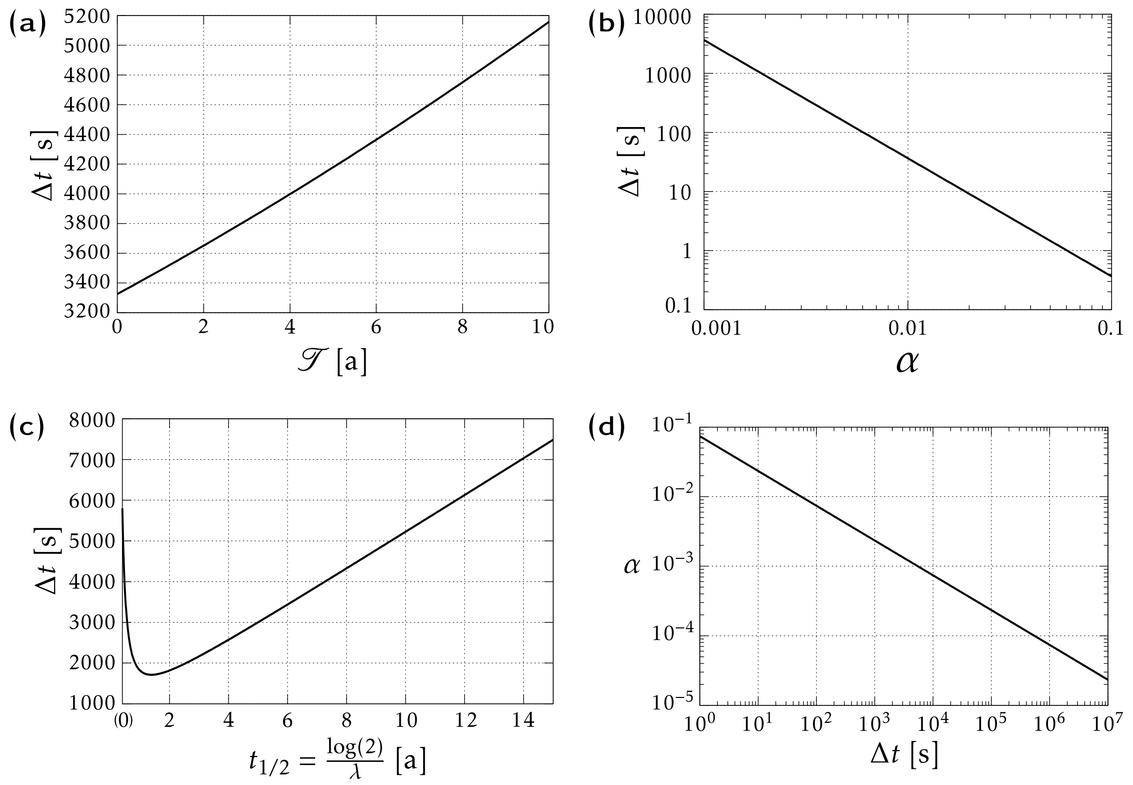


Figure 6 These plots show the functional connections of several attributes of a sample (Ba-133) and experimental parameters; $N_0 = 10^{11}$, $\lambda = 2.084 \cdot 10^{-9}\,\mathrm{s}^{-1}$ and $\mathscr{T} = 6.3 \cdot 10^7\,\mathrm{s}$ (if not stated otherwise). (a) Necessary integration time to maintain a certain error ($\alpha = 0.001$) throughout the whole measurement for a given total measurement duration; $\Delta t(\mathscr{T})$. (b) Necessary integration time for a given error throughout the whole measurement; $\Delta t(\alpha)$. (c) Necessary integration time to maintain a certain error ($\alpha = 0.001$) throughout the whole measurement for a given half-life; $\Delta t(\lambda)$. One can see very clearly that if the half-life is too short, the sample can not deliver enough counts (in the end of the measurement) because it disintegrates too fast. On the other hand, if its half-life is too long it takes very long to get enough counts. (d) Resulting error for a given integration time; $\alpha(\Delta t)$.

4.2. Effectiveness of the analysis methods – subtleties in the application of the simulation

This section describes all the findings which rose up during the implementation and usage of the simulation framework. It will therefore reference the contents of chapter 3 Simulation (p. 47) but implicitly also those of chapter 2 Physics and math (p. 15).

Generally, as the findings of section 4.3 Blind tests (p. 91) will show, it stands to reason to use only as much processing as necessary in order to avoid systematic errors.

4.2.1. Increasing sensitivity by standard decay subtraction/normalization

In all subsequent figures, the graphs were drawn with lines according to the underlying data points in order to guide the eye and to emphasize the quality of what is discussed hereafter.

As can also be seen in the second column (ψ) of figure 7, the autocorrelation function does reveal only effects of unrealistically pronounced amplitude. However, by subtracting the statistical mean values (standard decay) corresponding to the engaged sample (N_0, λ) from the data obtained through the measurement *before* performing the autocorrelation analysis, the sensibility of the latter can potentially be increased! The same holds true for the standard decay normalization method. Both tools are presented in sections 2.4.2 Standard decay subtraction (p. 41) and 2.4.3 Standard decay normalization (p. 43) and are, in fact, so effective that an autocorrelation analysis might not be even necessary.

The first discussion of these findings was made by using an early version of the standard decay subtraction. By that time, it was not yet mathematically formulated in the way it is now and the standard decay normalization was not introduced at all. Although every step along this time line improved the results, the old tests still picture the effectiveness of standard decay subtraction (or normalization) pretty well.

The original subtraction method reverse engineered a theoretical initial number of nuclei N_0' corresponding to one data pair of the measurement $\Delta N(t_i;\Delta t)$:

$$\Delta N(t_i;\Delta t) \overset{!}{\approx} N'(t_i) := N_0' \mathrm{e}^{-\lambda \cdot i \Delta t}$$

Rearranging yields

$$N_0' = \Delta N(t_i;\Delta t) \cdot \mathrm{e}^{\lambda \cdot i \Delta t}$$

and by setting $i = 1$, without loss of generality, one obtains

$$N_0' = \Delta N(t_1;\Delta t) \cdot \mathrm{e}^{\lambda \Delta t} \quad . \tag{4.2}$$

For every t_i in the data set one would now subtract the value $N'(t_i)$ from the measured value $\Delta N(t_i;\Delta t)$ and use the results for autocorrelation analysis.

$$\Delta N'(t_i;\Delta t) := \Delta N(t_i;\Delta t) - N'(t_i)$$

The results of this treatment, shown in the third column (ψ') of figure 7, speak for themselves. Normal autocorrelation, second column (ψ), fails already at high magnitudes of the effect.

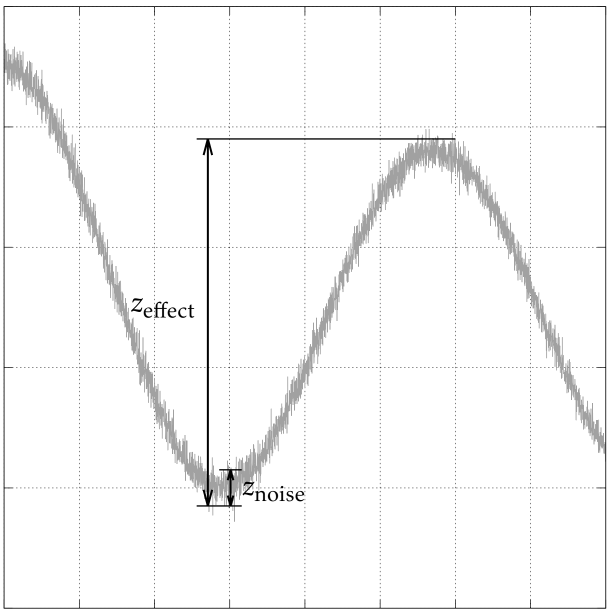
We define the mean relative error $\overline{\delta}$ of a measured number of counts

$$\overline{\delta} := \frac{\delta_{\min} + \delta_{\max}}{2} \quad , \tag{4.3}$$

using the boundary values of the respective measurement according to (2.56). We also introduce a signal-to-noise ratio (SNR), which is defined as the ratio of the effect amplitude z_{effect} to the amplitude of the noise z_{noise} in a given graph according to

$$\text{SNR} := \frac{z_{\text{effect}}}{z_{\text{noise}}} \quad . \tag{4.4}$$

The higher the value, the more pronounced the oscillation of the signal over the noise. A "smooth" graph will be denoted by SNR = ∞.



For simplicity, a fictional test sample ($t_{1/2} = 60 - 5000\,\mathrm{d}$, $N_0 = 10^5 - 10^{12}$) was used to explore the potential sensitivity of the procedure.

Simulations using the model of a sinusoidal summand (2.2.3 Sinusoidal, time dependent summand (p. 20)) were performed for constant $T = 20\,\mathrm{d}$ and $\mathscr{T} = 80\,\mathrm{d}$. As experimenting with different sets of parameters in the given ranges (see appendix A CD Attachment (p. 117)) shows, effects of the order $\varepsilon \approx 10^{-7}$ in the sinusoidal summand model are potentially visible as can be seen in figure 8! Mind that a statement about the magnitude of ε is only of use real use, if parameters like N_0 etc. are given. For instance, an absurdly high value for N_0 might render extremely low amplitudes visible, however, is not practically feasible.

A long half-life $t_{1/2}$ with respect to Δt and $\mathscr{T}$ can result in a quasi-constant background signal, but will also increase the relative error, which yet again might call for a greater $\mathcal{N}_0$.

Figure 9 compares the steps in the mentioned development of methods illustrating the respective advantages. All methods aim at eliminating the exponential contribution in the signal and reveal the (possible) oscillation.

As described in the theory chapter 2.4.2 Standard decay subtraction (p. 41), the subtracted signal will oscillate around zero and symmetrically decrease by an exponential envelope. The deprecated version of the subtraction algorithm, denoted by primed symbols, distorts the signal (although achieving astonishing results despite being very simple) in that it introduces an "exponential slope". The normalized signal removes any trace of the exponential characteristics successfully.

The disadvantage of maintaining an exponential contribution in both of the subtraction methods (it can also be seen in the third row of figure 8 or in the other panels as an exponential envelope/decay) is that it makes fitting complicated and error-prone and even leads to the autocorrelation failing its task.

As can be seen in the figure as well, autocorrelation seems to strangely fail sometimes; for instance in this case of a normalized signal, although the oscillation is clearly visible to the (blind) naked eye. The reason for that is unclear. Even more so why it was successful in the case of the very similar subtracted signal that, actually, should be even harder to analyze because of the exponential decay. On the other hand, we find examples of successful autocorrelation analysis of a normalized signal in 4.3.3 Comparison of the methods (p. 92). A quick test by examining the Fourier transform of the normalized signal showed a distinct peak at the effect's frequency.

On a side note, when performing standard decay subtraction or normalization, oscillations are usually already clearly visible in the modified signal (as shown in figure 9)! However, there are cases in which an autocorrelation analysis comes in handy as shown in the next paragraph.

Simulation with Ba-133

Figure 10 shows one exemplary simulation for a "real" measurement situation as described in 4.1.3 Magnitudes and values of introduced variables; exemplary case (p. 75). Furthermore, it is a text book example of the successful use of autocorrelation in order to verify an oscillation in a (subtracted) signal.

As shown before, a manifold of parameter configurations can provide workable results regarding the signal analysis. Thus, one would need to choose among them a set of parameters, which provides the best possible overall results (relative error, SNR etc.). It has yet to be further investigated, which parameter combinations interlock in what fashion. From the ample list of configurations given in the attachment, one may derive relationships between the different parameters and their magnitudes.

Behaviour of the subtraction methods

As already pointed out above, the autocorrelation might fail its job on signals with a pronounced (exponential) slope – and even in situations where it clearly should not. Also, the different methods of modifying the original signal yield similar results but differ in the details.

An error in the implementation of the deprecated subtraction algorithm while testing and comparing the mechanisms demonstrated how even small changes can impair the effectiveness of the results. During programming, two mistakes were made in the calculation of N_0' which would read `N0′ = data[0].dN * exp(lambda*dt)` in pseudo code according to (4.2): Once the factor $\exp(\lambda\Delta t)$ was missing – `N0′ = data[0].dN` referred to as `INDEX0` – and the other time the wrong index was used accidentally – `N0′ = data[1].dN` referred to as `INDEX1`.

Figure 11 shows the differences of the results. `INDEX1` comes with the largest deviation from the correct graph $\Delta N'$ and `INDEX0` is between the two. The "more correct" the calculated N_0' the less the exponential slope. A wrong choice of N_0' cannot not sufficiently suppress the exponential contribution. The current implementation ΔN_{sub} oscillates around zero as expected and shows the already discussed exponential envelope.

The autocorrelation functions emphasize the exponential contributions even more. One can clearly see that signals with a strong exponential decay slope will most likely provide no usable autocorrelation signal.

Interestingly enough, the shapes of the curves in figure 8 vary greatly although they use the same subtraction method. This might be due to different combinations of the parameters. A similar effect is described in 4.3.5 Parameter estimation by reconstruction (p. 98) where the qualitative characteristics of a normalized and a subtracted signal are reversed.

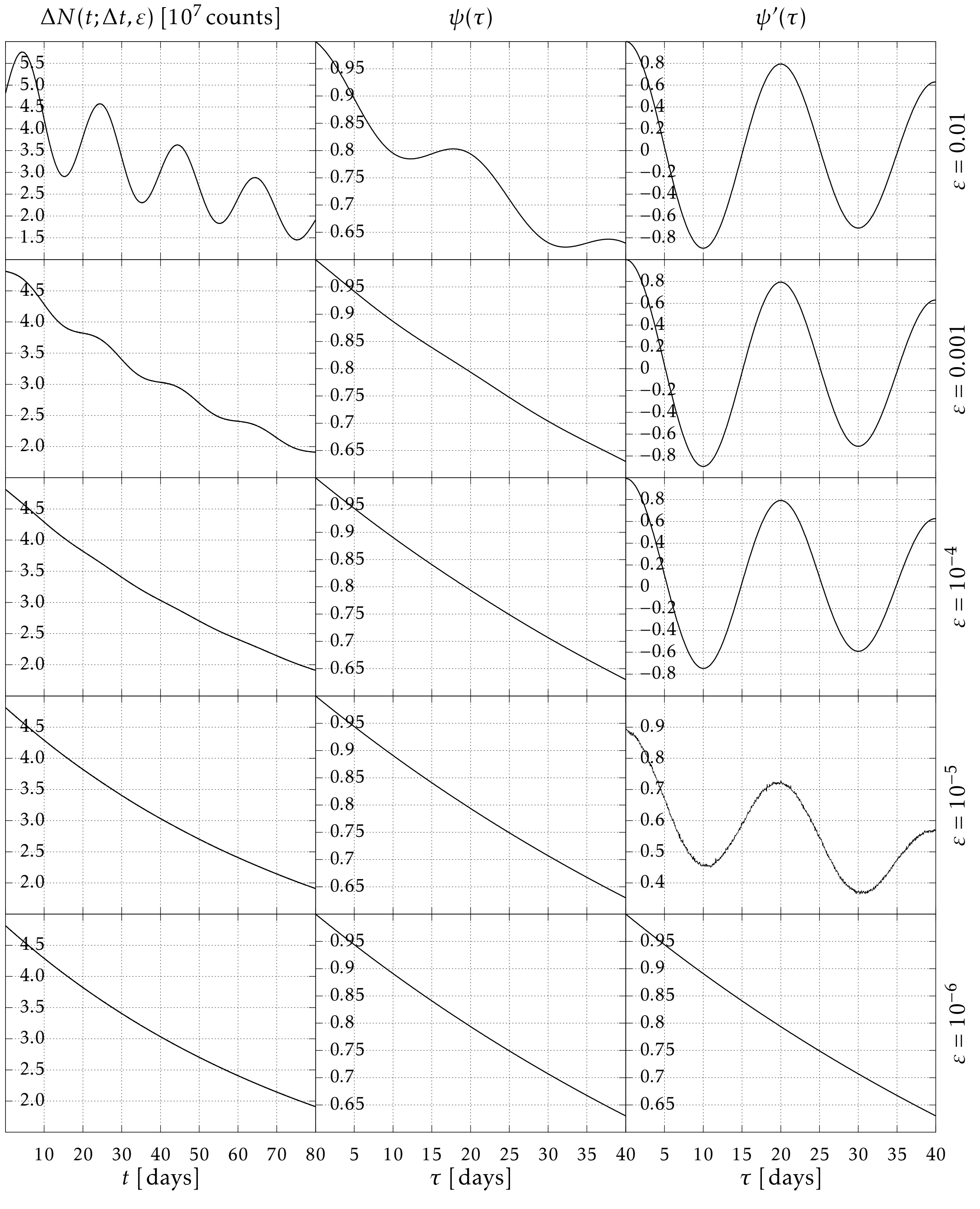


Figure 7 This plot shows several signals of varying oscillation amplitude ε (printed on the right hand-side) and constant periodicity $T = 20\,\mathrm{d}$. The first column represents the data of a measurement (created with the simulation program), the second column shows the normalized autocorrelation function and the third column pictures the normalized autocorrelation function resulting after standard decay subtraction. The periodicity of the signal is clearly visible and can be read directly from the plot. Mind that the abscissas of the autocorrelation plots show the real lag $\tau = \hat{\tau}\Delta t$ already. $N_0 = 10^{11}$, $t_{1/2} = 60\,\mathrm{d}$, $\mathscr{T} = 80\,\mathrm{d}$ and $\Delta t = 1\,\mathrm{h}$.

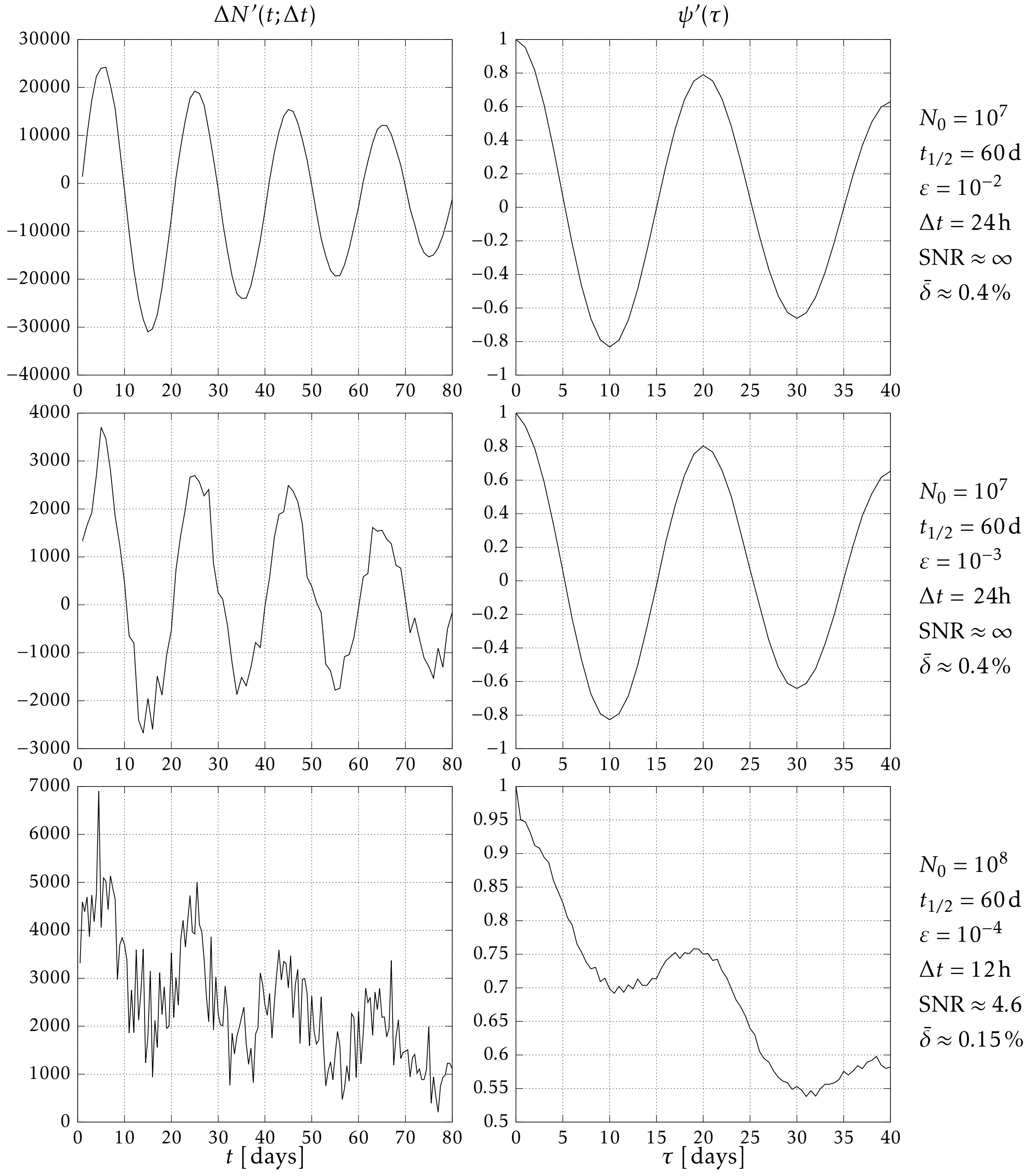
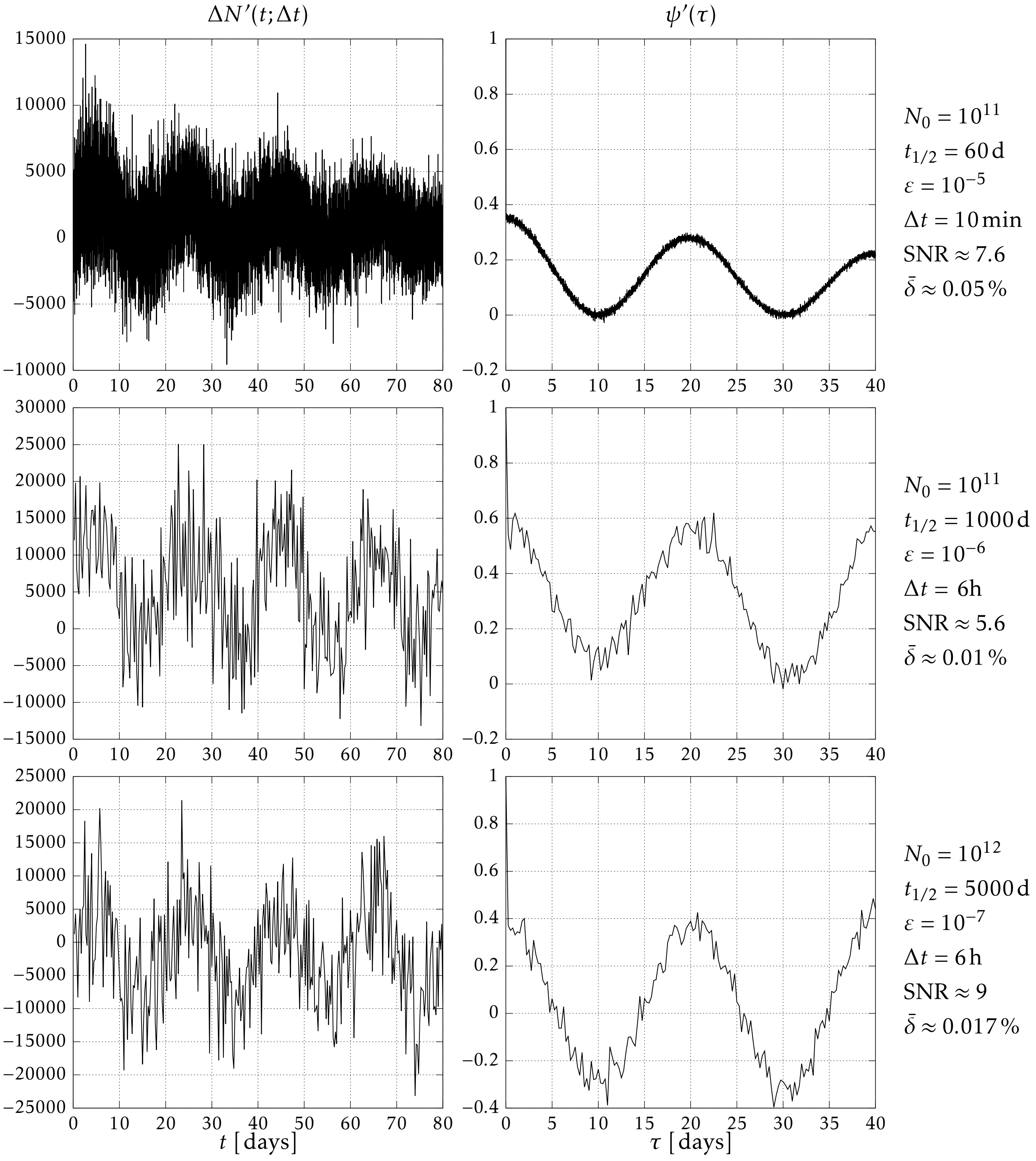


Figure 8 This graph shows plots for effect amplitudes ε the range of 10^{-2} to 10^{-7} with different sample and measurement parameters. The effect periodicity $T = 20\,\text{d}$ and the total measurement duration $\mathscr{T} = 80\,\text{d}$ remain the same. The left column shows the signal after standard decay subtraction was performed, the second column provides the resulting autocorrelation function. In every graph, the periodicity is clearly visible, ...



Continued Figure 8 …even in the ones of tiniest effect amplitude ε. Many more parameter configurations provide workable results. The periodicity is also visible in the ΔN_{sub} graphs, although the noise is less pronounced in the ψ' graphs. The values for SNR and $\bar{\delta}$ are roughly estimated "by hand" in accordance with (4.4) and (4.3); they correspond to the ψ' graphs. Additionally, the condition (2.65) can be clearly seen. Long half-lives ($t_{1/2} \gg T$) suppress the exponential character of the graphs.

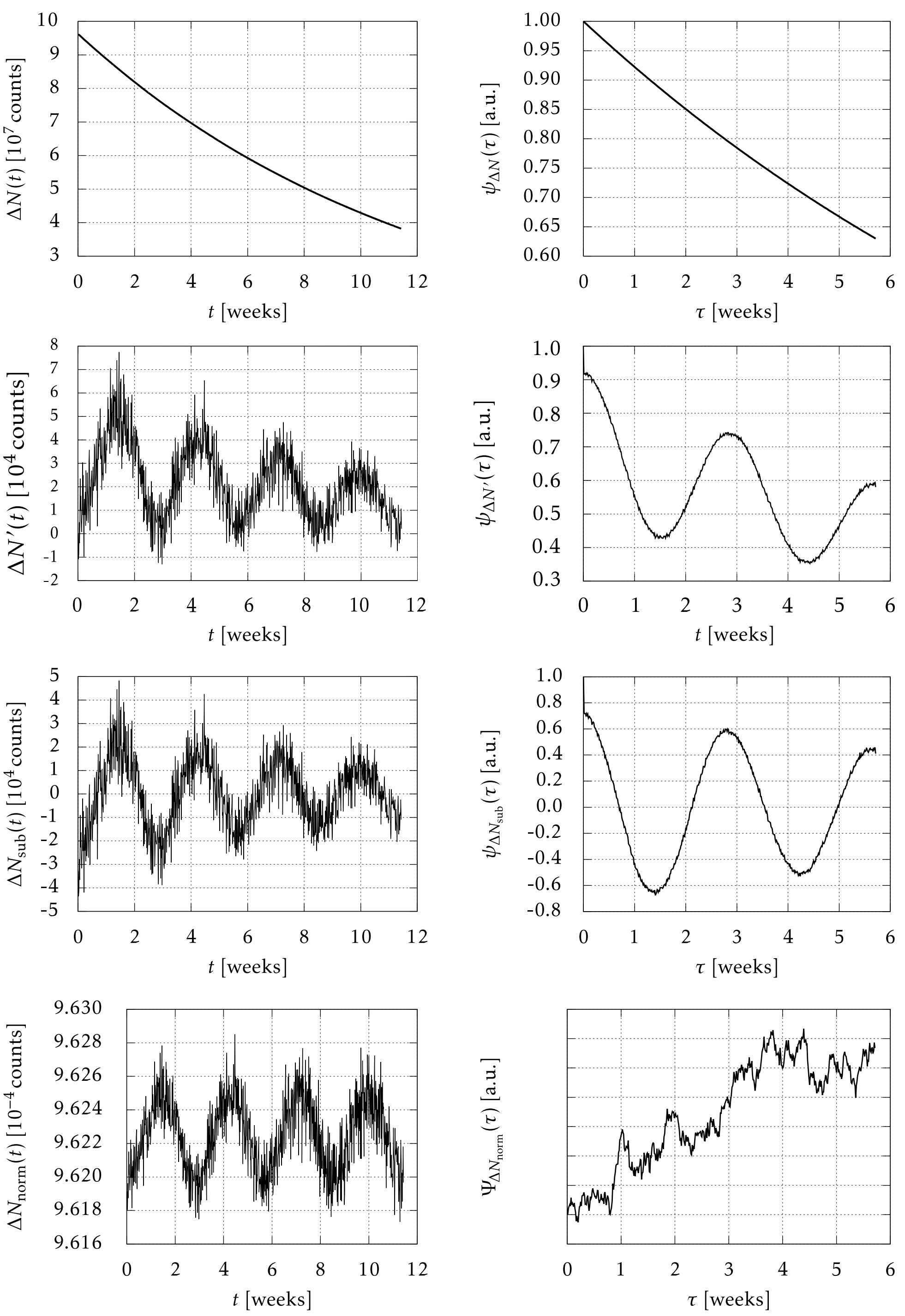


Figure 9 These plots demonstrate the different "standard decay procedures" in the left column and the resulting autocorrelation fuction in the right column. $N_0 = 10^{11}$, $t_{1/2} = 60\,\mathrm{d}$, $\varepsilon = 10^{-5}$, $T = 20\,\mathrm{d}$, $\mathscr{T} = 80\,\mathrm{d}$ and $\Delta t = 2\,\mathrm{h}$. The first row shows the original signal. The second row shows the original subtraction method. Its disadvantage lies in the"exponential slope" which also influences the autocorrelation function. The more pronounced the exponential decay is, the more ineffective the autocorrelation analysis becomes. The third row demonstrates the current implementation of the standard decay subtraction. The exponential contribution is that of an envelope. As expected, the curve oscillates about zero. Finally, in the last row the standard decay normalization is presented. The exponential characteristics have vanhished and the amplitude shifted in to an entirely different order of magnitude (which will be used for the estimation of ε). However, it inexplicably spoils the autocorrelation analysis in this case (the range of values is so small that it cannot be usefully displayed). The reason for that is unknown.

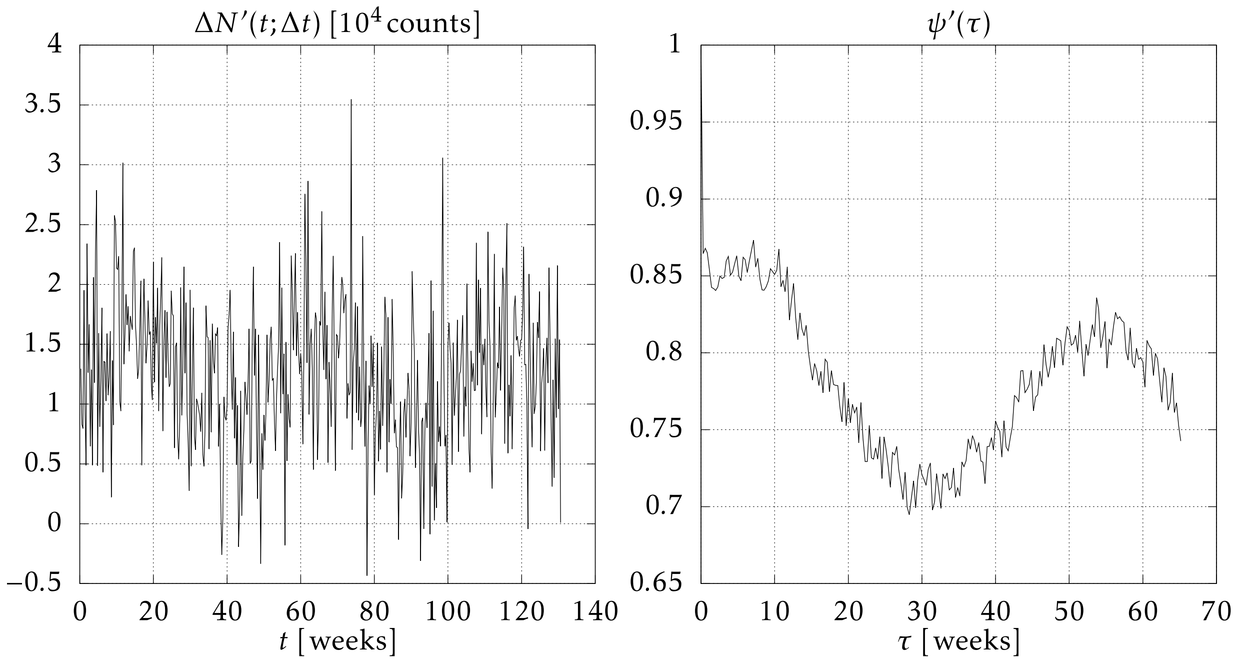


Figure 10 This figure shows the standard decay subtraction graph ΔN_{sub} and the resulting autocorrelation function ψ' for a Ba-133 sample. The parameters are: $N_0 = 10^{11}$, $t_{1/2} = 10.54\,\mathrm{a}$; $\varepsilon = 10^{-6}$, $T = 1\,\mathrm{a}$; $\Delta t = 48\,\mathrm{h}$, $\mathscr{T} = 2.5\,\mathrm{a}$. It might be one of the cases in which the presence of an effect needed to be assured by autocorrelation. This result is especially noteworthy considering how poorly the autocorrelation analysis fails in the example in figure 9.

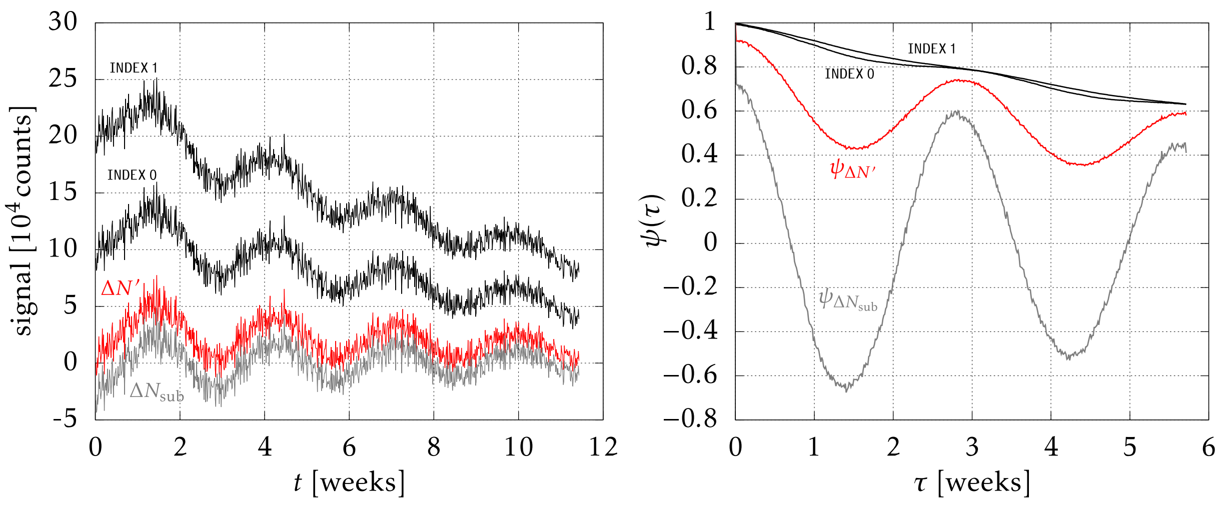


Figure 11 This plot compares several versions of the standard decay subtraction procedure; namely the deprecated $\Delta N'$ and the current subtraction ΔN_{sub}. The left graph provides the subtracted signals, the right plot the corresponding autocorrelation functions. The same exemplary parameters as in the other comparison plot are used: $N_0 = 10^{11}$, $t_{1/2} = 60\,\mathrm{d}$, $\varepsilon = 10^{-5}$, $T = 20\,\mathrm{d}$, $\mathscr{T} = 80\,\mathrm{d}$ and $\Delta t = 2\,\mathrm{h}$. The upper two curves (black) resulted from programming errors described in the main text. The red curve resembles the correct algorithm and the gray curve gives the current implementation. One can clearly see how the characteristics of the exponential contribution differ.

4.2.2. Problems in the autocorrelation analysis

Application to subtracted/normalized signals

Autocorrelation of an unprocessed signal is futile. Standard decay subtraction/normalization, as extensively discussed in the preceding section, can produce signals for successful analysis. Ususally, an oscillation is already visible in these source signals, but autocorrelation might enhance the SNR and thus improve fit results or might find correlations which are not even visible in the subtracted/normalized signal. Refer to figure 10, for instance.

Yet there are cases in which the autocorrelation analysis fails miserably and the reason for that is not clear. Refer to the last panel in figure 9. These problems seem to arise with normalized signals mostly, which might be due to their small values.

Problems in the automated analysis of the autocorrelation function

The autocorrelation function of a processed signal can come with various exponential contributions. The implementation of an automated algorithm designed to find a periodicity in the given signal is not trivial, even though a possible oscillation might be even visible to the naked eye. In addition to the statistical fluctuation, this shape of the curve makes it hard for an automated algorithm to find a periodicity. This is especially the case, when the exponential decrease is so pronounced that $\forall i: \ \psi(t_{i+1}) < \psi(t_i)$, even though there *is* a clear sinusoidal oscillation. Of course, also fitting to such a curve is not trivial - the exponential artifacts would need to be modeled via additional fit parameters.

Immense measurement duration

One major disadvantage of the autocorrelation is the fact that the maximum lag reaches only half-way into the data according to (2.63). This means that minimum measurement time for a successful autocorrelation analysis is twice the cycle duration of the effect according to (2.65). In case of an annual oscillation this would mean a *minimum* measurement time of two years!

Considering the immense measurement duration, it is highly advised (in a real setup) to perform multiple measurements in parallel.

Smoothing/Accumulation

The autocorrelation of a lightly smoothed/accumulated signal may deliver better results, since it reduces the amount of noise in the signal.

Fourier analysis

In cases in which the autocorrelation fails to deliver usable results, a Fourier analysis of the processed signal could reveal a periodicity. Such an analysis tool is not implemented in the framework, yet a quick test in Matlab for the presented case showed a distinct peak at the effect's frequency in the spectrum.

Improper integration time

If the integration time Δt is too long, an oscillation might still be visible, but the maxima/minima get shifted and will not provide the correct lag in the autocorrelation, because the step size on the $\hat{\tau}$-axis is too big. A value for Δt corresponding to the Nyquist frequency is a very unrealistic choice, anyway. In general, it is a question of initial phase whether the maximum/minimum of a possible sinusoidal oscillation appears exactly on the data grid. A value for Δt way below $\mathscr{T}/4$ might be appropriate so that the maxima and minima of possible sinusoidal oscillations appear on the data grid, independent from the phase of the effect's cycle. Such considerations might become important for large Δt since the autocorrelation reduces the number of available points.

4.2.3. Comparing the effects of smoothing and accumulation

In this section the benefits of smoothing (refer to 2.4.4 Smoothing and filtering (p. 44)) a given signal versus accumulating it (refer to 2.4.5 Accumulation (p. 46)) will be presented.

In order to justify comparison, the signal was smoothed/accumulated such that the respective first data point would lie at the same point in time, thus

$$n_{\Delta t} = \sigma_{\Delta t} n_\sigma + 1 \quad .$$

Smoothing yields a lower relative error at the cost of truncation on both ends of the signal. Since, however, the accumulated signal does not provide direct information about the time prior to its first data point either, this "disadvantage" remains without consequence. In contrast to the accumulation procedure which might discard at most $n_{\Delta t}$ data points at the end of the signal, the smoothing algorithm uses all available data points. Furthermore, the smoothing process leaves the position of the graph unchanged unlike the accumulation process, which shifts the signal height into another order of magnitude. See figure 12.

It turned out, however, that strong smoothing alters the effective amplitude of the signal – an effect which needs to be considered if the signal should be used for further analysis.

This most probably is due to the equivalence of smoothing and filtering. As explained in the theory chapter, smoothing is the same as to low-pass filtering. The process removes spectral components of high frequency which weakens the signal. The higher the grade of smoothing (i.e. the lower the critical filter frequency), the more frequency components are removed and the more pronounced the impact on total amplitude.

Accumulation on the other hand seems to not distort the signal. These findings are documented in section 4.3 Blind tests (p. 91).

4.2.4. Retrieving the effect's frequency ω and amplitude ε

All the presented techniques can and should be engaged in order to isolate any traces of a possible effect. The two parameters of interest are its frequency ω and amplitude ε and all available means should be used to evaluate the present data.

In theory, the autocorrelated signal will provide the experimenter with the frequency of the effect. Usually, it will already become visible in the signal obtained by standard decay subtraction or normalization respectively. Fitting will provide a value for ω. In order to retrieve an estimation for the effect's amplitude, we will use exactly these signals and investigate relations (2.70)

$$\Delta N_{\text{norm}}(t) \approx \Big(1 + \varepsilon\cos(\omega t)\Big) - \mathrm{e}^{-\lambda_0 \Delta t}\Big(1 + \varepsilon\cos(\omega t + \phi)\Big)$$

and (2.71)

$$\Delta N_{\text{norm}}(t) \approx \varepsilon\Big(\cos(\omega t) - \cos(\omega t + \phi)\Big)$$

further.

We would like to extract ε from the normalized signal ΔN_{norm}. It is only available as a data function, but the underlying analytical expression is well defined. So far, the amplitude of the signal is $\xi \neq \varepsilon$ due to interference of the two cosine terms. However, since $\omega = \text{const}$, there is no beat in the signal and since the phase difference $\phi = \omega\Delta t = \text{const}$ is known ($\Delta t = \text{const}$ is chosen by the experimenter) and constant and ω will be drawn from analysis, the signal is sinusoidal and oscillating with the same frequency. There is only a modulation of the effect's amplitude ε by a constant factor and a phase shift χ. It can thus be rewritten in the very simple form of a sine (at first we consider (2.71))

$$\Delta N_{\text{norm}}(t) := \xi \cdot \sin(\omega t + \chi) \quad ,$$

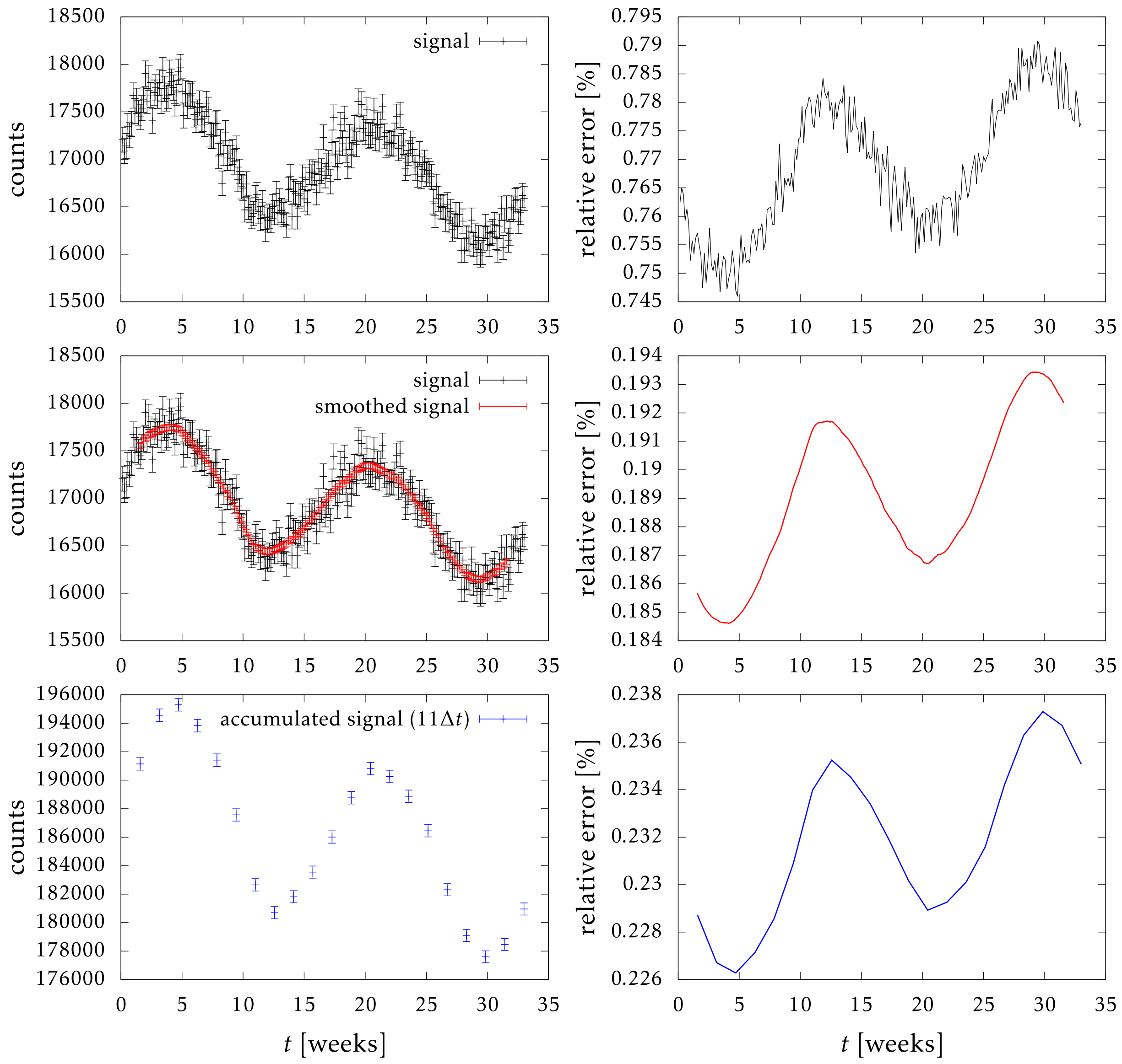


Figure 12 This figure shows a simulated signal $\Delta N(t_i;\Delta t)$, where $N_0 = 10^8$, $\lambda = 2\cdot10^{-9}\,\mathrm{s}^{-1}$, $T = 10^7\,\mathrm{s}$ and $\Delta t = 86400\,\mathrm{s}$, and its smoothed ($\sigma_{\Delta t} = 5$ and $n_\sigma = 2$) and accumulated ($n_{\Delta t} = \sigma_{\Delta t} n_\sigma + 1 = 11$) version. Next to them are their respective relative errors. One can clearly see that any treatment of the two ameliorates the relative error, however, the best results are achieved by smoothing.

which can be shown by the following addition theorem:

$$\cos(x) - \cos(y) = -2\sin\left(\frac{x+y}{2}\right)\sin\left(\frac{x-y}{2}\right)$$

Setting $x = \omega t$ and $y = \omega t + \phi$ we obtain (including the prefactor ε)

$$-2\varepsilon\sin\left(\frac{\omega t + \omega t + \phi}{2}\right)\sin\left(\frac{\omega t - \omega t - \phi}{2}\right)$$

and using $\sin(-x) = -\sin(x)$ we find

$$2\varepsilon\sin\left(\frac{\phi}{2}\right)\cdot\sin\left(\omega t + \frac{\phi}{2}\right) \ .$$

Comparison yields then

$$\xi = 2\varepsilon\sin\left(\frac{\phi}{2}\right) \quad \text{and} \quad \chi = \frac{\phi}{2} \quad \text{for (2.71).} \tag{4.5}$$

With the same arugments we can also find an alternate form of (2.70), unveiling the impact of the interference between the cosine terms. Expansion of the product in (2.70) yields

$$\varepsilon\cos(\omega) - \varepsilon\,\mathrm{e}^{-\lambda_0\Delta t}\cos(\omega t + \phi) + \left(1 - \mathrm{e}^{-\lambda_0\Delta t}\right)$$

and we generalize by writing

$$A\cos(\omega t) + B\cos(\omega t + \phi) + C \ ,$$

where $A = \varepsilon$, $B = -\varepsilon\exp(-\lambda_0\Delta t)$ and $C = 1 - \exp(-\lambda_0\Delta t)$. In analogy to the case before we formulate the desired function

$$\xi\cdot\sin(\omega t + \chi) + \zeta \ .$$

Using $\cos(x+y) = (\cos x\cos y - \sin x\sin y)$ and equating the coefficients provides the following relations between the coefficients A, B, C and ξ, χ, ζ:

$$\begin{aligned} A + B\cos\phi &= \xi\sin\chi \\ -B\sin\phi &= \xi\cos\chi \\ C &= \zeta \end{aligned}$$

Adding the first two equations squared and using $(\sin^2 x + \cos^2 x) = 1$ yields

$$\xi = \sqrt{A^2 + B^2 + 2AB\cos\phi} \ .$$

By simply rearranging them we obtain

$$\xi = \frac{A + B\cos\phi}{\sin\chi} = -\frac{B\sin\phi}{\cos\chi} \ .$$

Dividing the first by the second equation and using $(\sin x/\cos x) = \tan x$, we find an equation for the new phase χ:

$$\chi = \tan^{-1}\left(-\frac{A + B\cos\phi}{B\sin\phi}\right)$$

Plugging in A, B and C we find

$$\xi = \varepsilon\sqrt{1 + \mathrm{e}^{-2\lambda_0\Delta t} - 2\,\mathrm{e}^{-\lambda_0\Delta t}\cos\phi} \ , \quad \chi = \tan^{-1}\left(\frac{1 - \mathrm{e}^{-\lambda_0\Delta t}\cos\phi}{\mathrm{e}^{-\lambda_0\Delta t}\sin\phi}\right)$$
$$\text{and} \quad \zeta = 1 - \mathrm{e}^{-\lambda\Delta t} \quad \text{for (2.70)} \ . \tag{4.6}$$

If χ is drawn from the data, it might be of advantage to use

$$\xi = \varepsilon \frac{\mathrm{e}^{-\lambda_0 \Delta t} \sin\phi}{\cos\chi}$$

instead.

Setting $A := \varepsilon$, $B := -\varepsilon$ and $C := 0$ according to (2.71), we obtain exactly the results of (4.5).[38]

In a measurement/simulation, Δt is known and ξ and ω ($\phi = \omega\Delta t$) are drawn from the data.[39] The effect's amplitude ε can then be easily calculated and, in case of the sinusoidal decay "constant" model, the parameter a determined via (2.30)

$$a = \frac{\varepsilon\omega}{\lambda_0} \quad .$$

Rearranging to obtain ε from (4.6) yields

$$\varepsilon = \xi \left(1 + u^2 - 2u\cos\phi\right)^{-1/2} \quad \text{with} \quad u = \mathrm{e}^{-\lambda_0 \Delta t} \quad .$$

In order to perform an uncertainty calculation according to (2.44) one needs the derivatives with respect to ξ and ϕ, where the latter reads

$$\frac{\partial\varepsilon}{\partial\phi} = -\xi u \sin\phi \left(1 + u^2 - 2u\cos\phi\right)^{-3/2} \quad .$$

4.2.5. Effect magnitude and uncertainty estimation

In section 4.1.2 A set of rules (p. 73) we found relations in very simple fashion, showing how the manifold of parameters affect each other and pose criteria to be fulfilled in order to meet certain requirements. Among others, a desired relative error of α was introduced, which was related to the effect's amplitude ε and the accuracy with which it can be detected. This was a very superficial discussion and now that we have decided for a concrete model in 4.1.1 The model of choice (p. 73) and can retrieve the effect's amplitude from a normalized signal as in 4.2.4 Retrieving the effect's frequency ω and amplitude ε (p. 87), a brief subsequent discussion is due.

As explained in 2.2.4 Sinusoidal, time dependent decay constant (p. 22), ε is actually just an intermediate quantity linking the intrinsic model parameter a and the quantity ξ, which is the amplitude in the normalized signal. This value ξ is detectable in the experiment and links almost all parameters characterizing the effect, the sample and the experiment itself: $\xi = \xi(\varepsilon, \omega, \lambda, \Delta t)$ as given in (4.6). For these two reasons the detectable amplitude ξ will be in focus of this discussion.

The SNR introduced in 4.2.1 Increasing sensitivity by standard decay subtraction/normalization (p. 77) ultimately determines if an oscillation becomes visible in the normalized signal. Recalling (4.4)

$$\mathrm{SNR} := \frac{z_\text{effect}}{z_\text{noise}}$$

we set $z_\text{effect} = \xi$ and $z_\text{noise} = \sigma_{\Delta N_\text{norm}}$ and obtain

$$\frac{z_\text{effect}}{z_\text{noise}} = \frac{\xi}{\sigma_{\Delta N_\text{norm}}} =: c \quad .$$

[38] For the proof we need $1 - \cos x = 2\sin^2\left(\frac{x}{2}\right)$ and $\frac{1-\cos x}{\sin x} = \tan\left(\frac{x}{2}\right)$.

[39] The amplitude ξ must be taken from ΔN_norm whereas ω might be obtained by fitting to either ΔN_norm or smoothed/autocorrelated versions of the signal.

Rearranging yields

$$c\,\sigma_{\Delta N_\mathrm{norm}} = \xi \quad ,$$

providing the amplitude ξ with an accuracy of c sigmas.

An inverse definition

$$\mathrm{NSR} := \frac{z_\mathrm{noise}}{z_\mathrm{effect}}$$

brings a slightly different interpretation. Assigning the same variables this becomes

$$\frac{\sigma_{\Delta N_\mathrm{norm}}}{\xi} =: \beta \quad .$$

The nature of β is that of a relative uncertainty most closely resembling the parameter α, where

$$\beta = \frac{1}{c} \quad .$$

However, in contrast to α, β relates to ΔN_norm instead of ΔN.

Mind that $\sigma_{\Delta N_\mathrm{norm}}$ is actually time and sample dependent and that – whatever requirements are formulated on the basis of this parameter – it needs to retain its minimum value until the end of the whole measurement.

Furthermore, $\sigma_{\Delta N_\mathrm{norm}}$ may be roughly approximated by

$$\sigma_{\Delta N_\mathrm{norm}} \approx \frac{\sigma_{\Delta N}}{N_\mathrm{std}} = \frac{\sqrt{\Delta N}}{N_\mathrm{std}} \quad .$$

Mind also here the time dependence.

Additionally, β must not be confused with $\delta_{\Delta N_\mathrm{norm}} = \sigma_{\Delta N_\mathrm{norm}}/\Delta N_\mathrm{norm}$, which is the relative uncertainty of the normalized signal.

4.3. Blind tests

This section will document the blind tests which were performed to test the presented methods.

4.3.1. Motivation and design

While creating and testing all the presented models and analysis methods, the investigated effects and test/debug scenarios were available in all their detail. This means that at all times, the expected results were known; a situation bringing a lot of bias!

The concept of a blind test is to eliminate said bias and investigate *unknown* signals.

Erwin Jericha used the simulation framework to create a series of test signals and deleted any documentation about the introduced effect from the data files. A copy of the original files was saved for later comparison against the obtained results.

The following sections are devoted to the quantitative comparison of the analysis methods and will draw conclusions for their usage.

4.3.2. Evaluation of analysis methods based on the results of a blind test

Next to the elimination of expectation there is another huge advantage to the idea of such tests. Naturally, the *true* values of the quantities in question are known! Thus, the most accurate method to retrieve those values can be picked by comparison.

As in all experiments and simulations there are two types of errors: statistical and systematic errors. Obviously, for the analysis of a given signal, only methods with minor to no systematic errors should be used. Among those one wishes to pick the ones of highest statistical quality. If one needs to use

A set of simple, straight-forward selection rules was established to eliminate faulty methods and rank the remaining ones by their statistical accuracy in the course of a blind test.

Every fitted or calculated quantity brings a result/mean value μ_i and an uncertainty σ_i.[40] The *true* value of interest is denoted by $\tilde{\mu}$ and is exact.

First attempts of working with the various methods showed that some of them provide high statistical accuracy but introduce a notable systematic error. Evaluation of the given uncertainty alone is thus no sufficient selection criterion. First, the deviations of the acquired values from the true value, the absolute errors, are computed:

$$\Delta\mu_i = \left|\tilde{\mu} - \mu_i\right| \tag{4.7}$$

The relative error is defined as

$$\delta_{\mu_i} = \frac{\Delta\mu_i}{\tilde{\mu}} \quad . \tag{4.8}$$

Now, only methods which find the true value within one standard deviation are accepted, thus, they need to fulfill the criterion

$$\Delta\mu_i \leq 1 \cdot \sigma_i \quad . \tag{4.9}$$

The set of accepted methods can now be ranked by their statistic quality, the best being $\min\{\sigma_i\}$. In case of a number of methods providing the same uncertainty σ, they can be ranked by the deviation of their respective mean values from the true value. The method with $\min\{\Delta\mu_i\}$ is the most accurate among them.

It is possible that a result of sufficient accuracy can emerge from two input values with uncertainties although one of them might not meet one of the upper criteria. It makes sense to further evaluate also the disqualified methods according to the level of their correctness. The criterion

$$r_{\mu,i} := \frac{\Delta\mu_i}{\sigma_i} \tag{4.10}$$

provides how many standard deviations the calculated is off the real value. Depending on this ratio one can decide whether a method should still be (under reserve) accepted nontheless. It is just (4.9) where $r_{\mu,i} = 1$. Mind that judging from this ratio alone can be very misleading! Accpeted methods according to (4.9) need to be ranked by the criteria mentioned above. (4.10) only gives a general notion about the method's accuracy.

4.3.3. Comparison of the methods

The contents and ideas presented above will be exemplified on the basis of such a blind test – namely blind test "A".

After a standard decay normalization, which provided a value for N_0, an oscillation was undoubtedly visible in the signal. Fits were performed to this normalized signal as well as to differently processed versions of it like the respective autocorrelation function or differently

[40] The fitting was performed with Gnuplot 5.2 using the basic syntax `fit` ⟨*fit function*⟩"⟨*source file*⟩" `using 1:2:3 yerr via` ⟨*fit parameters*⟩. ⟨*fit function*⟩ is a user-defined function using the free fit parameters ⟨*fit parameters*⟩ and the entries of the third column of the data file (uncertainty) are used as weights. The deviations of the listed fit results are Gnuplot's "asymptotic standard errors" and are only useful for qualitative purposes, since they are considered "over-optimistic". Thus, the given deviations do not determine confidence levels [25].

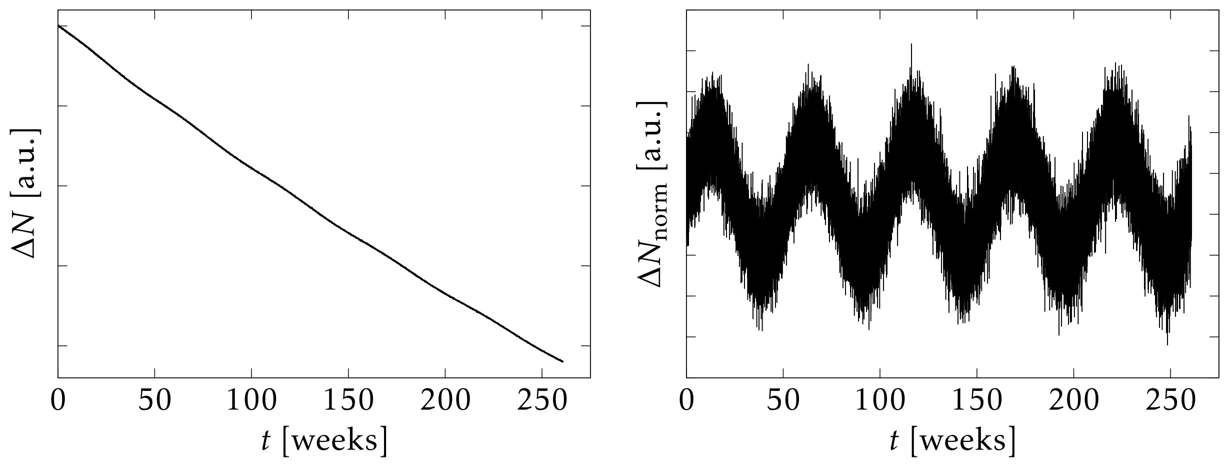


Figure 13 Original (left) and normalized (right) signal of blind test "A". The oscillation is clearly visible in the normalized signal and already faintly suggested by the original signal. This speaks for a quite pronounced amplitude of a possible effect.

smoothed or accumulated versions.

Normalized signals can provide information about amplitude ε, frequency ω and phase χ of the effect and so do accumulated and smoothed signals. An autocorrelation signal will just provide the frequency of the effect. The model parameter is then calculated by using the amplitude value gained from the signal before autocorrelation.

The known parameters of this blind test (seed: 1316160400) were:

$$\begin{aligned} \mathscr{T} &= 5\,\mathrm{a} \\ \Delta t &= 3600\,\mathrm{s} = 1\,\mathrm{h} \\ \lambda &= 2.08396 \cdot 10^{-9}\,\mathrm{s}^{-1} \end{aligned}$$

From these,

$$\begin{aligned} u &= \mathrm{e}^{-\lambda \Delta t} = 0.999992498 \\ \zeta &= 1 - u = 7.5022278 \cdot 10^{-6} \end{aligned}$$

can be calculated analytically.

Before presenting the numbers, the most important findings shall be discussed:

First of all, it is very important that a clear evaluation of techniques is only possible if the exact result is known, which is never the case in a real measurement situation. Thus, the results gained by the *least* amount of processing involved should be given the most weight.

Secondly, one needs to be aware of what parameters a given analysis method might distort. Smoothing, for instance, potentially increases the accuracy of a frequency fit, but quickly spoils the results of an amplitude fit.

The following processing was performed:

- (1) standard decay normalization,
- (2) autocorrelation of the normalized signal,[41]

[41] Note that, unlike in section 4.2.1 Increasing sensitivity by standard decay subtraction/normalization (p. 77), the autocorrelation of the normalized signal worked just fine.

- (3) (4) smoothing with two values $\sigma_{\Delta t}$ where $n_\sigma = 2$ and respective
- (5) (6) autocorrelation of the smoothed signals,
- (7) (8) normalization of accumulated versions of the original signal for two values $n_{\Delta t}$ and respective
- (9) (10) autocorrelation of the accumulated, normalized signals.

The true values (denoted by a tilde) of the parameters of interest read:

$$\begin{aligned}\widetilde{N}_0 &= 10^{12}\\ \tilde{\varepsilon} &= 10^{-5}\\ \widetilde{\omega} &= 1.9910638 \cdot 10^{-7}\,\mathrm{rad\,s^{-1}}\\ \tilde{a} &= 9.554215 \cdot 10^{-4}\end{aligned}$$

Of course they were looked up *after* the calculations. The number of initial nuclei is computed in the course of the normalization and reads $N_0 = 1.00000558 \cdot 10^{12} \pm 1.89 \cdot 10^6$.[42] Fitting to the normalized signal, we obtain (in brackets the coefficient of variation, i.e. the relative standard deviation σ/μ, is given)

$$\begin{aligned}\varepsilon_1 &= (1.0017 \pm 0.0028) \cdot 10^{-5}\ (0.28\,\%)\\ \omega_1 &= (1.9910 \pm 0.0006) \cdot 10^{-7}\,\mathrm{rad\,s^{-1}}\ (0.03\,\%)\\ a_1 &= (9.569 \pm 0.027) \cdot 10^{-4}\ (0.28\,\%)\end{aligned}$$

The frequencies ω_i and the error are taken directly from the Gnuplot fit, whereas the amplitude ε_i and the model parameter a_i are computed via (4.6) and (2.30). The necessary values for that (like ξ_i and $\phi_i = \omega_i \Delta t$) are also computed or taken from the fit. Consistency is indicated by the fit result $\zeta_1 = (7.502156 \pm 0.000014) \cdot 10^{-6}$ (0.0002 %) – this held true for all cases.

Autocorrelation yielded

$$\begin{aligned}\omega_2 &= (1.989517 \pm 0.000024) \cdot 10^{-7}\,\mathrm{rad\,s^{-1}}\ (0.001\,\%)\\ a_2 &= (9.563 \pm 0.027) \cdot 10^{-4}\ (0.28\,\%)\ .\end{aligned}$$

Smoothing with $\sigma_{\Delta t,3} = 168 \mathrel{\hat{=}} 7\,\mathrm{d}$ yields[43]

$$\begin{aligned}\varepsilon_3 &= (9.9391 \pm 0.0011) \cdot 10^{-6}\ (0.01\,\%)\\ \omega_3 &= (1.991051 \pm 0.000024) \cdot 10^{-7}\,\mathrm{rad\,s^{-1}}\ (0.001\,\%)\\ a_3 &= (9.496 \pm 0.001) \cdot 10^{-4}\ (0.01\,\%)\end{aligned}$$

and smoothing with $\sigma_{\Delta t,4} = 672 \mathrel{\hat{=}} 4\,\mathrm{weeks}$ results in

$$\begin{aligned}\varepsilon_4 &= (9.13186 \pm 0.00045) \cdot 10^{-6}\ (0.005\,\%)\\ \omega_4 &= (1.990844 \pm 0.000012) \cdot 10^{-7}\,\mathrm{rad\,s^{-1}}\ (0.0006\,\%)\\ a_4 &= (8.72382 \pm 0.00043) \cdot 10^{-4}\ (0.005\,\%)\ ,\end{aligned}$$

[42] The sinusoidal contribution distorts the accuracy of the result. For standard decay signals, the computed values for N_0 are consistent.

[43] The mathematical sign "$\hat{=}$" is used because the two sides are not of equal dimension, but it means (in this case) that using the smoothing class with a parameter value of 168 corresponds to averaging over a time window of 7 days for the given Δt.

which brings in already quite considerable deviations from the true values of $\tilde{\varepsilon}$ and $\tilde{a}$. Notice that, at the same time, the ostensible accuracy increased by up to three orders of magnitude! Also the results for the autocorrelated signals show "wrong" results of high certainty.

$$\begin{aligned}\omega_5 &= (1.989541 \pm 0.000022) \cdot 10^{-7}\,\mathrm{rad\,s^{-1}}\ (0.001\,\%)\\ a_5 &= (9.489 \pm 0.001) \cdot 10^{-4}\ (0.01\,\%)\end{aligned}$$

$$\begin{aligned}\omega_6 &= (1.98945 \pm 0.00019) \cdot 10^{-7}\,\mathrm{rad\,s^{-1}}\ (0.001\,\%)\\ a_6 &= (8.71773 \pm 0.00044) \cdot 10^{-4}\ (0.007\,\%)\ .\end{aligned}$$

Accumulation is not an analysis method in that sense and does not process the data other than simulating a higher integration time Δt by summation providing the statistical advantages of such a measurement. It turns out indeed that it does not introduce the disadvantages of the smoothing procedure presented above. To make the procedures comparable, $n_{\Delta t,7} = 336$ was chosen such that $n_{\Delta t} \mathrel{\hat{=}} \sigma_{\Delta t} n_\sigma$, meaning that both graphs cover the same time range. Mind that $\Delta t_{\mathrm{acc}} = n_{\Delta t}\Delta t$ has to be used for further calculations with the accumulated signal.

$$\begin{aligned}\varepsilon_7 &= (9.994 \pm 0.023) \cdot 10^{-6}\ (0.23\,\%)\\ \omega_7 &= (1.99096 \pm 0.00051) \cdot 10^{-7}\,\mathrm{rad\,s^{-1}}\ (0.03\,\%)\\ a_7 &= (9.548 \pm 0.022) \cdot 10^{-4}\ (0.23\,\%)\end{aligned}$$

For $n_{\Delta t,8} = 1344$ we obtain

$$\begin{aligned}\varepsilon_8 &= (9.993 \pm 0.027) \cdot 10^{-6}\ (0.27\,\%)\\ \omega_8 &= (1.99071 \pm 0.00063) \cdot 10^{-7}\,\mathrm{rad\,s^{-1}}\ (0.03\,\%)\\ a_8 &= (9.546 \pm 0.026) \cdot 10^{-4}\ (0.27\,\%)\ .\end{aligned}$$

The results of the autocorrelation are:

$$\begin{aligned}\omega_9 &= (1.98947 \pm 0.00045) \cdot 10^{-7}\,\mathrm{rad\,s^{-1}}\ (0.02\,\%)\\ a_9 &= (9.541 \pm 0.022) \cdot 10^{-4}\ (0.23\,\%)\end{aligned}$$

$$\begin{aligned}\omega_{10} &= (1.9891 \pm 0.0011) \cdot 10^{-7}\,\mathrm{rad\,s^{-1}}\ (0.05\,\%)\\ a_{10} &= (9.538 \pm 0.027) \cdot 10^{-4}\ (0.28\,\%)\ .\end{aligned}$$

Applying the criterion (4.9) qualifies the following methods, ranked by increasing standard deviation and (if necessary) increasing absolute error:

- ε_7, ε_8 and ε_1
- ω_3, ω_7, ω_1 and ω_8
- a_7, a_8, (a_2), (a_9), a_1 and (a_{10})

The results in brackets denote that the respective value was calculated by using a value not on the list of accepted methods (see below).

As we can see, the autocorrelation (2) (5) (6) (9) (10) never yields satisfying results for the effect's frequency ω. Thus, it should only be engaged in cases where a periodicity can not be assuredly detected in the signals before autocorrelation analysis. However, (9) and (10) almost fulfill the criteria since $r_{\omega,9} = 3.54$ and $r_{\omega,10} = 1.79$.

Interestingly enough though, exactly those "faulty" frequencies obtained from autocorrelation signals allowed for proper determination of the model parameter a, except for those of the smoothed signals (5) and (6). While $r_{a,5} = 65.22$ and $r_{a,6} = 1901.10$, the other ratios for the autocorrelation methods are well $r_{a,i} < 1$ ($i = 2, 9, 10$). This is probably by virtue of the parameter ε being of sufficient accuracy: $r_{\varepsilon,1(2)} = 0.61$, $r_{\varepsilon,7(9)} = 0.26$ and $r_{\varepsilon,8(10)} = 0.26$ as opposed to $r_{\varepsilon,3(5)} = 55.36$ and $r_{\varepsilon,4(6)} = 1929.2$.[44] Mind also that the inaccuracy of the frequency in these cases, as mentioned before, is not strikingly bad; $r_{\omega,9} = 3.54$ and $r_{\omega,10} = 1.79$ almost fulfill the selection criterion, merely $r_{\omega,2} = 64.45$ is quite high but presumably compensated for by $\varepsilon_{1(2)}$.

Soft smoothing (3) might be acceptable for frequency fitting ($\Delta\omega_3 < \sigma_{\omega,3}$, $r_{\omega,3} = 0.53$). In this particular case it even yielded the best result for the frequency approximation, although the error introduced in the amplitude is already so considerable ($r_{\varepsilon,3(5)} = 55.36$) that the result was not fulfilling the criterion anymore ($r_{a,3} = 58.22$). The autocorrelation (5) of that signal worsened the results ($r_{\omega,5} = 69.22$, $r_{a,5} = 65.22$).

Heavy smoothing (4) brings unacceptable distortion to the amplitude evaluation ($r_{\varepsilon,4(6)} = 1929.2$) which renders the final result useless ($r_{a,4} = 1931.15$), although the frequency deviation is not affected that badly ($r_{\omega,4} = 18.32$). Interestingly, the autocorrelation signal (6) slightly augmented the results in this case ($r_{\omega,6} = 8.49$, $r_{a,6} = 1901.1$).

These figures support the findings of section 4.2.3 Comparing the effects of smoothing and accumulation (p. 87) in that smoothing affects the amplitude of the processed signal.

Although various combinations of methods (even with "insufficient" interim results) yield values fulfilling condition (4.9), the results obtained by a pronounced normalized signal or by accumulation and normalization should be considered with highest priority. This is also surmounted by the fact that methods (1), (7) and (8) are the only ones providing *all* parameters while satisfying the selection criteria. Furthermore, the autocorrelated (9) and (10) also use signal accumulation and miss the criteria only very narrowly.[45]

Generally, the relative errors (4.8) consistently follow (with minor deviations) the discussed ranking of methods.

The relative errors δ_{ε_i} range from $\delta_{\varepsilon_7} = 0.06\,\%$ to $\delta_{\varepsilon_3} = 0.61\,\%$; only method (4) delivers the worst result of $\delta_{\varepsilon_{4(6)}} = 8.68\,\%$ by virtue of the influence of the applied smoothing.

The results for ω are all extraordinarily good judging from the relative errors alone: We find values from $\delta_{\omega_3} = 0.0006\,\%$ to $\delta_{\omega_{10}} = 0.0986\,\%$. Note that, however, only four of them fulfill criterion (4.9).

Since the parameter a is derived from both ε and ω, the relative errors unsurprisingly vary from $\delta_{a_7} = 0.07\,\%$ to $\delta_{a_5} = 0.68\,\%$ with the expected exception of $\delta_{a_4} \approx \delta_{a_6} \approx 8.7\,\%$.

After all it needs to be emphasized that all methods (apart from the obvious exception of the smoothed signal in terms of amplitude) deliver results with a relative error well below 1 %, even though a considerable number of them fails to fulfill the criterion (4.9), which is used to judge their ostensible systematic accuracy.

The hand-written full evaluation of blind test "A" can be found in appendix C Evaluation notes for blind test "A" (p. 119).

[44] The notation $r_{\varepsilon,1(2)}$ just emphasizes that method (2) does not provide a value for ε and that the value from method (1) is used instead as would seem natural; etc.

[45] In comparison to unweighted fitting (provided in the CD attachment), the results changed only very slightly, often yielding almost exactly the same values. The "ranking" of the methods is not changed either, apart from the a values, however, the presented qualitative findings remain valid. Interestingly, the results obtained by methods using the autocorrelation improved slightly when compared to unweighted fitting, which encourages to further investigate the uncertainty propagation of the autocorrelation analysis method. Also, the accumulation results seem to have slightly improved. In contrast, the results of the purely normalized signal worsened ever so slightly.

4.3.4. Characteristics of standard decay

When searching for "non-standard" physics, one has to be very careful not to be fooled by expectations. Thus, it is very important to know the characteristics of standard decay and how such a signal looks like when the presented analysis methods are applied. Blind test "B" was assumed to and turned out to be a standard signal. Figure 14 demonstrates the effects of smoothing and accumulating such a signal. Additionally, the respective autocorrelation functions are given. Especially the smoothed signal might put oneself into assuming a deterministic oscillation. However, the irregularities in periodicity and particularly the great fluctuation in amplitude speak against such a finding.

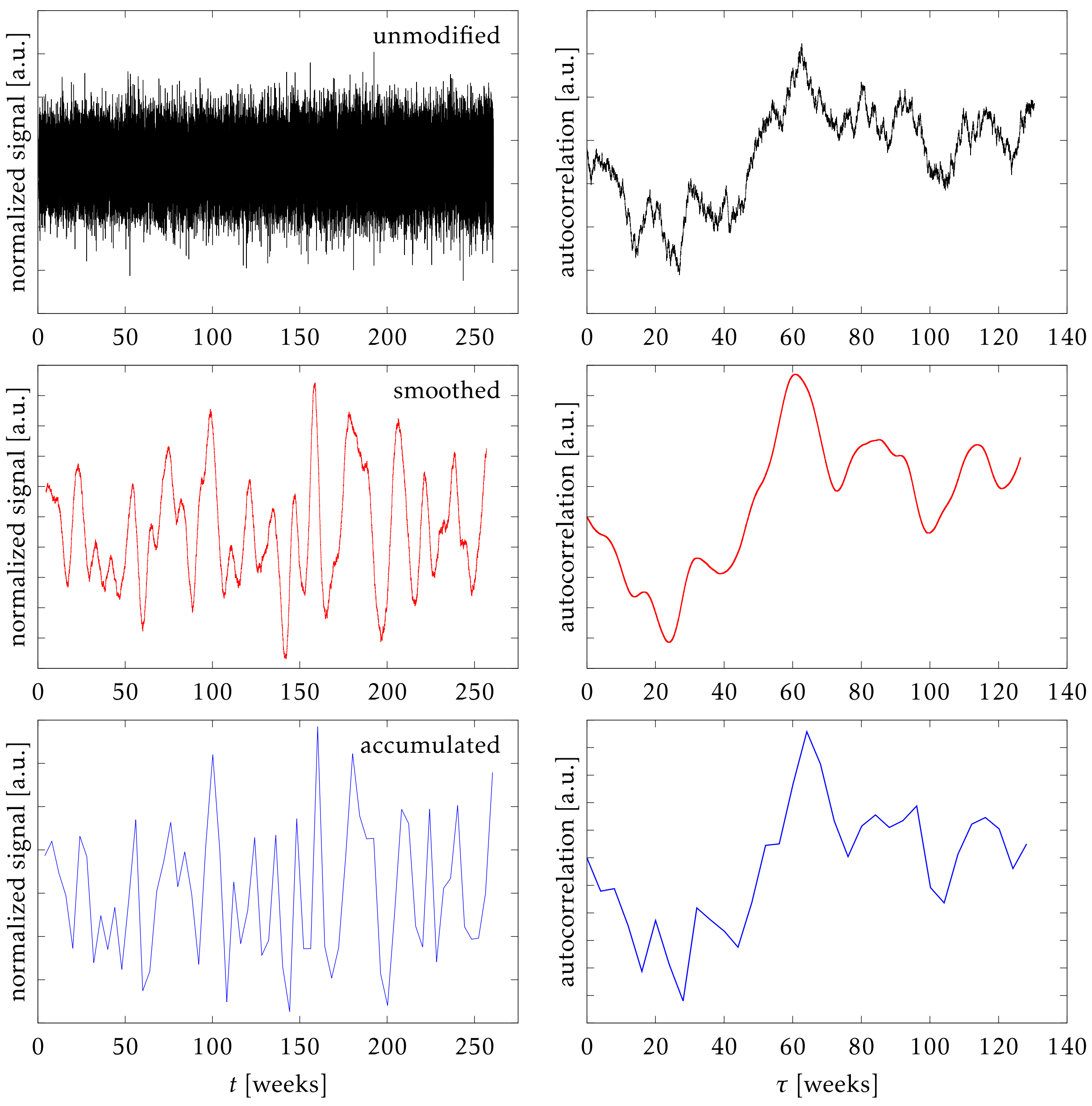


Figure 14 This figure shows a series of processings of a standard decay signal (blind test "B"). The left column shows the normalized signal, a smoothed version ($\sigma_{\Delta t} = 336$, $n_\sigma = 2$) and the normalization of an accumulated version ($n_{\Delta t} := \sigma_{\Delta t} n_\sigma + 1 = 673$). The right column depicts the respective autocorrelation function. $\Delta t = 3600\,\mathrm{s}$ and after accumulation $n_{\Delta t}\Delta t = 2419200\,\mathrm{s}$. The data points were connected because only the characteristics of the graphs are of interest.

4.3.5. Parameter estimation by reconstruction

As luck would have it, another interesting strategy can be demonstrated on the basis of blind test "C": The reconstruction of the source signal by using the simulation framework in order to estimate the effect parameters. If the analysis of the given signal does not allow for comprehensive parameter determination, missing values may be narrowed down roughly by "sculpting" an artificial signal by varying the unassigned parameters.

The known parameters of this blind test were like in blind test "A" (seed: 561038530):

$$\begin{aligned}\mathscr{T} &= 5\,\mathrm{a}\\ \Delta t &= 3600\,\mathrm{s} = 1\,\mathrm{h}\\ \lambda &= 2.08396\cdot 10^{-9}\,\mathrm{s}^{-1}\end{aligned}$$

From these,

$$\begin{aligned}u &= \mathrm{e}^{-\lambda\Delta t} = 0.999992498\\ \zeta &= 1-u = 7.5022278\cdot 10^{-6}\end{aligned}$$

can be calculated analytically.

The first trials showed that neither the subtracted or the normalized signal displayed any sign of periodic fluctuation and nor did the autocorrelation function of the subtracted signal. Merely the autocorrelation graph of the normalized source signal might give a hint on a obscured periodicity. The mentioned graphs are depicted in figure 15.

However, upon accumulation of the source signal, the autocorrelation of both the subtracted and the normalized signals revealed a periodicity. Interestingly enough, the one of the subtracted signal provided the most workable result with no exponential slope in contrast to the normalized signal.

The source signal was accumulated with $n_{\Delta t} = 336$. A fit to the autocorrelation function of the afterwards subtracted signal yielded

$$\omega_1 = (2.009 \pm 0.033)\cdot 10^{-7}\,\mathrm{rad\,s}^{-1}\ (1.64\,\%)\ .$$

The subtraction algorithm provided a value for the initial number of nuclei: $N_0 \approx 10^{12}$. Accumulation with $n_{\Delta t} = 672$ gave analogously

$$\omega_2 = (2.00 \pm 0.03)\cdot 10^{-7}\,\mathrm{rad\,s}^{-1}\ (1.5\,\%)\ .$$

In both autocorrelation functions the first value (which is always $\psi(0) = 1$) was set to `NaN` in order to facilitate the fitting process.

Even though the autocorrelation analysis provided a value for the effect's frequency, the normalized signal did not show a clear oscillation. In a daring attempt to retrieve a value for ε from it nonetheless, a fit was performed with fixed ω_1. The resulting value

$$\varepsilon_1 = (5.60 \pm 0.086)\cdot 10^{-5}\ (15\,\%)$$

seemed totally wrong (judging from experience of the former blind tests).

Another approach proved successful: Knowing N_0 and ω, the only missing parameter was ε. The simulation framework was now used – with constant seed 561038530 – to create test signals with the known input parameters ($\Delta t' = 336\,\Delta t$ according to the accumulation which revealed the signal's periodicity) but varying ε.

Starting from a relatively high value at which the oscillation was clearly visible in both the subtracted and normalized signal, ε was reduced step by step until any trace of the oscillation disappeared. This was the case at $\varepsilon \approx 4.0 \cdot 10^{-7}$ below which the oscillation could not be unequivocally seen in the processed source signal anymore (but indeed in the autocorrelation plots). This naturally marked the upper limit of ε, since in the studied blind test the processed signals did not show any clear sign of oscillation either but the autocorrelation plots did indeed.

The lower limit was determined by steadily reducing the value of ε until even the autocorrelation functions did not show a clear sign of periodic oscillation. This was the case at $\varepsilon \approx 1.0 \cdot 10^{-7}$. Refer to figure 17.

Especially for finding the lower limit, all four graphs ΔN_{sub}, ΔN_{norm}, ψ_{sub} and ψ_{norm} (which did not show the pronounced slope) were inspected.[46]

This procedure provided an estimated range of

$$\varepsilon \approx (1.0 - 4.0) \cdot 10^{-7} \quad .$$

The true values read:

$$\begin{aligned} \widetilde{N}_0 &= 10^{12} \\ \tilde{\varepsilon} &= 10^{-7} \\ \widetilde{\omega} &= 1.9910638 \cdot 10^{-7}\,\text{rad}\,\text{s}^{-1} \\ \tilde{a} &= 9.554215 \cdot 10^{-6} \end{aligned}$$

The estimation of ω is very good and ε is also surprisingly accurate (given that it could not be taken from the source signal).

These simulations were performed with $\Delta t'$ directly. Instead, simulation with Δt and subsequent accumulation would have been the approach to depict the original situation more correctly. This method introduced a slight exponential slope in the autocorrelation of the normalized signal (in accordance with the findings of the original examination), however, rendered the autocorrelation of the subtracted signal slightly more workable.

In retrospect, I would have gone down as far as $\varepsilon \approx 8 \cdot 10^{-8}$ in estimating the lower boundary.

Figure 16 shows the accumulated, normalized/subtracted source signals and the respective autocorrelation functions. The frequency of the effect was taken from the autocorrelation graph of the subtracted signal.

Also, in this case happened what became apparent in various situations of testing: The quality of the results varies. It can happen that, with the same set of parameters, one simulation shows a known periodicity and the next one simply does not as demonstrated in figure 18. This is a strong argument supporting the need to perform multiple measurements in parallel, since very subtle effects may not always become visible.

[46] For instance, I would have set the lower limit only to $\varepsilon = 1.5 \cdot 10^{-7}$ judging from ψ_{sub} alone.

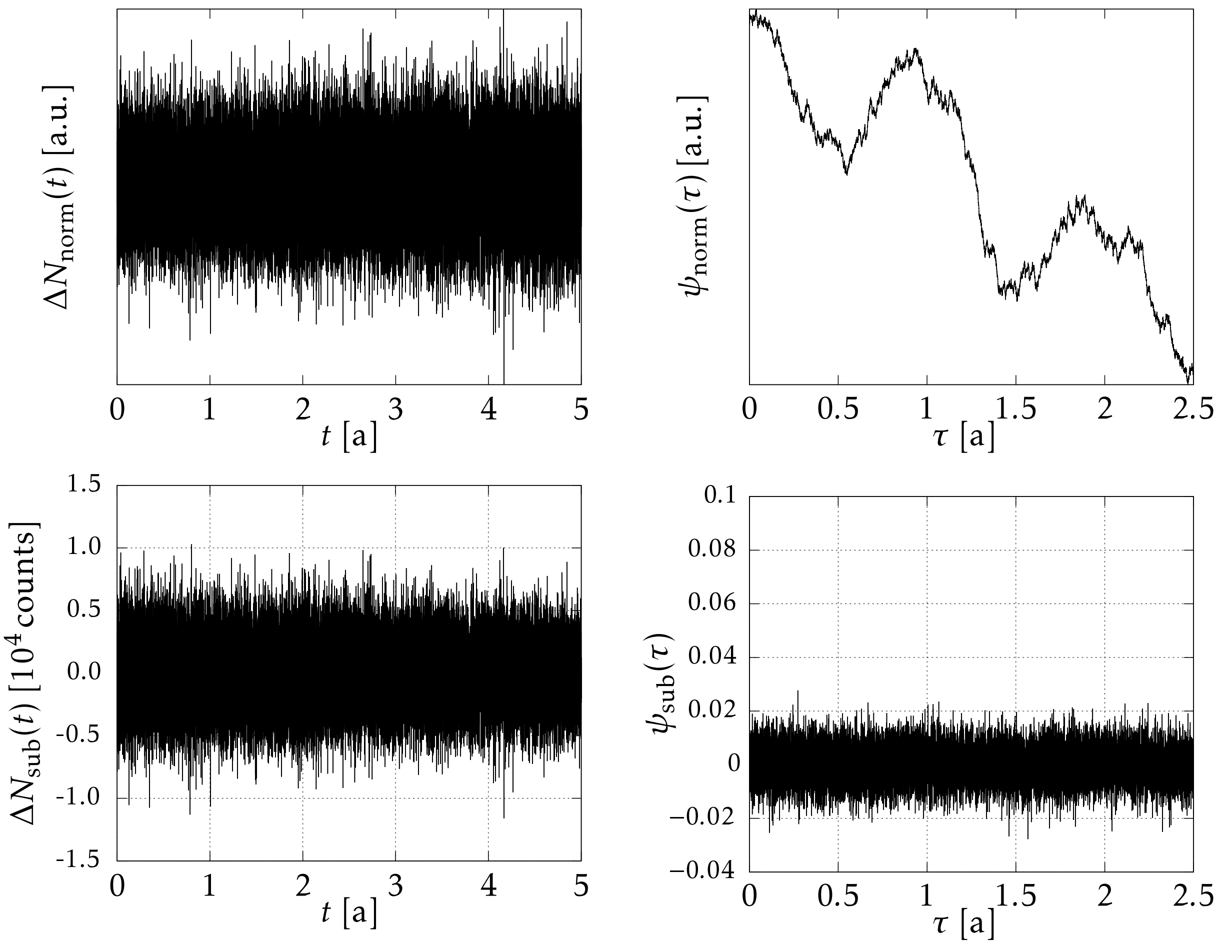


Figure 15 This figure shows the normalized/subtracted source signal of blind test "C" in the left column and the corresponding autocorrelation functions in the right column. The graphs look very similar and do not give away a possible obscured oscillation except for the autocorrelation graph of the normalized signal (top right), which suggests a periodicity. The difference in the two autocorrelation graphs is striking, given that the respective underlying signals show no visible difference in their quality. It is also noteworthy that the character of $\psi_{\mathrm{norm}}(\tau)$ (compare with figure 16) does not significantly change throughout the signal processing. The data point $\psi_{\mathrm{sub}}(0)$=1 is cut off at 0.1 for reasons of presentability.

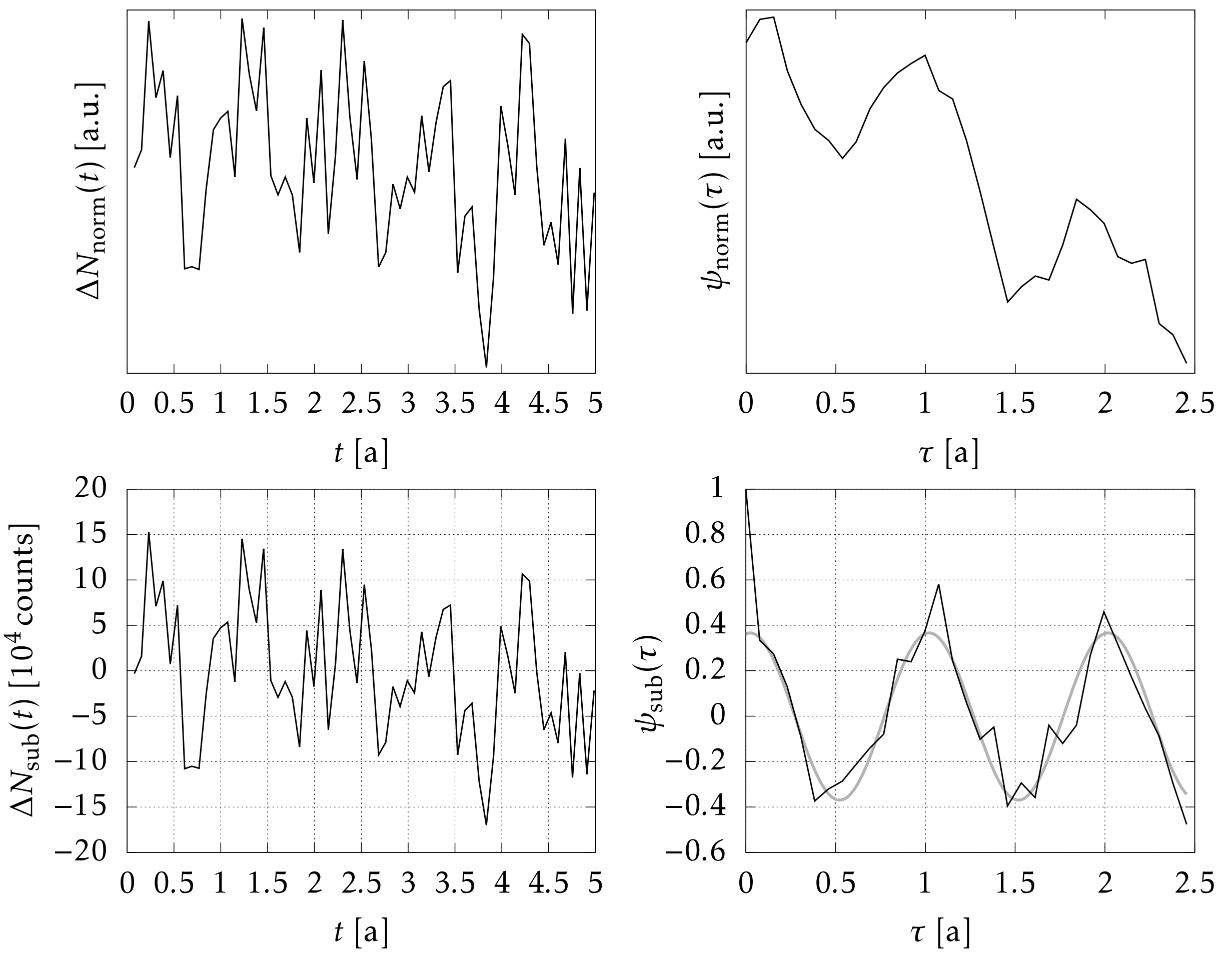


Figure 16 The left panels show the accumulated ($n_{\Delta t} = 672$) and normalized/subtracted source signal of blind test "C". They do not give away the hidden periodicity $\widetilde{T} = 1\,\mathrm{a}$ which is clearly visible in the autocorrelation graph of the subtracted signal (bottom right), though. The gray curve is the sinusoidal fit for which the value $\psi(0) = 1$ was omitted and from which ω was taken. The autocorrelation graph of the normalized signal (top right) suggests the oscillation, but shows a slope.

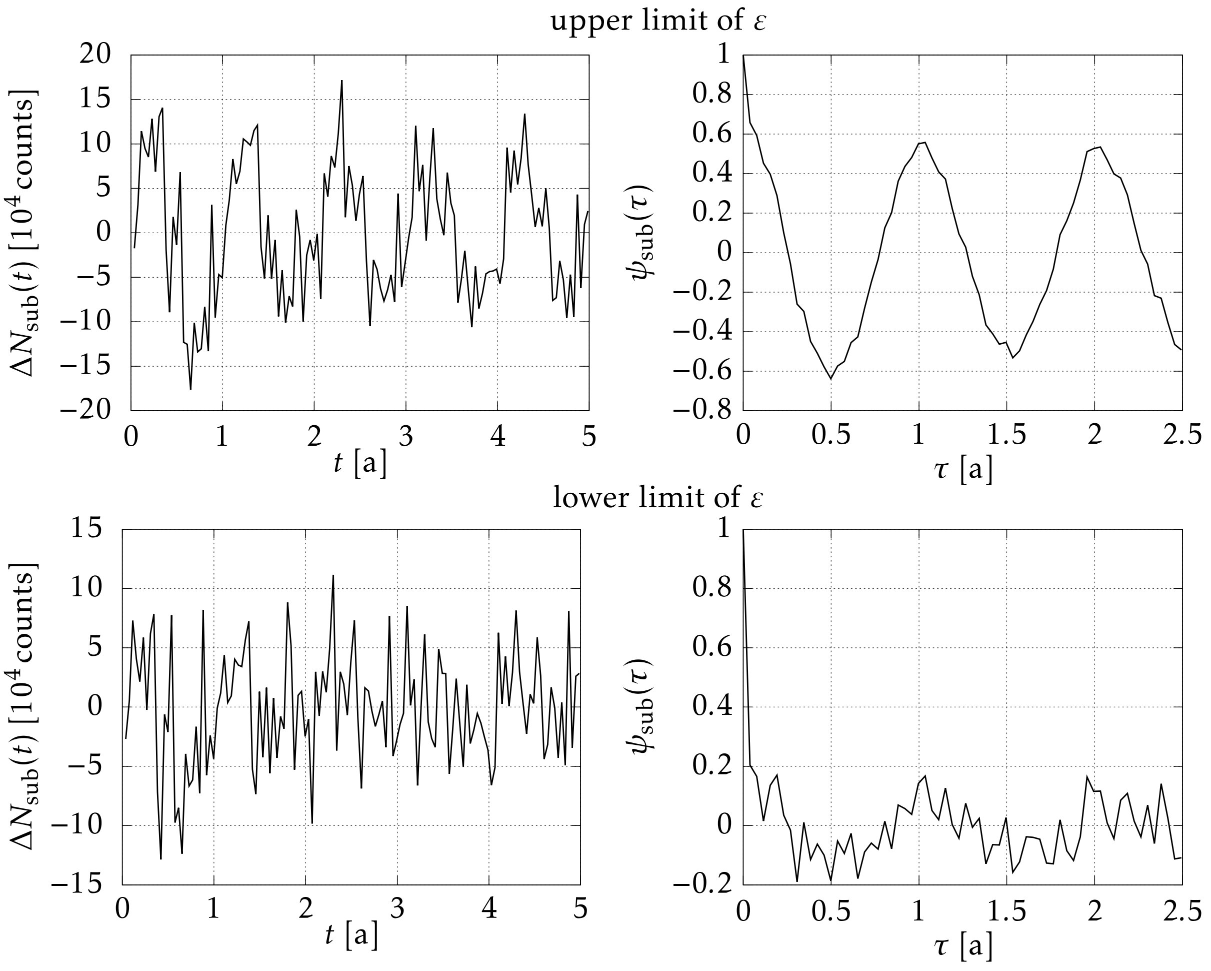


Figure 17 This figure shows exemplary plots of the ε estimation procedure (blind test "C") described in the main text. The first row shows graphs for the upper limit of $\varepsilon \approx 4.0 \cdot 10^{-7}$ – the subtracted signal starts to get obscured, however, the oscillation in autocorrelation function is obvious as expected. The second row shows graphs for the lower limit of $\varepsilon \approx 1.0 \cdot 10^{-7}$ – the subtracted signal shows no clear oscillation anymore and also the oscillation of the autocorrelation graph is rather subtle.

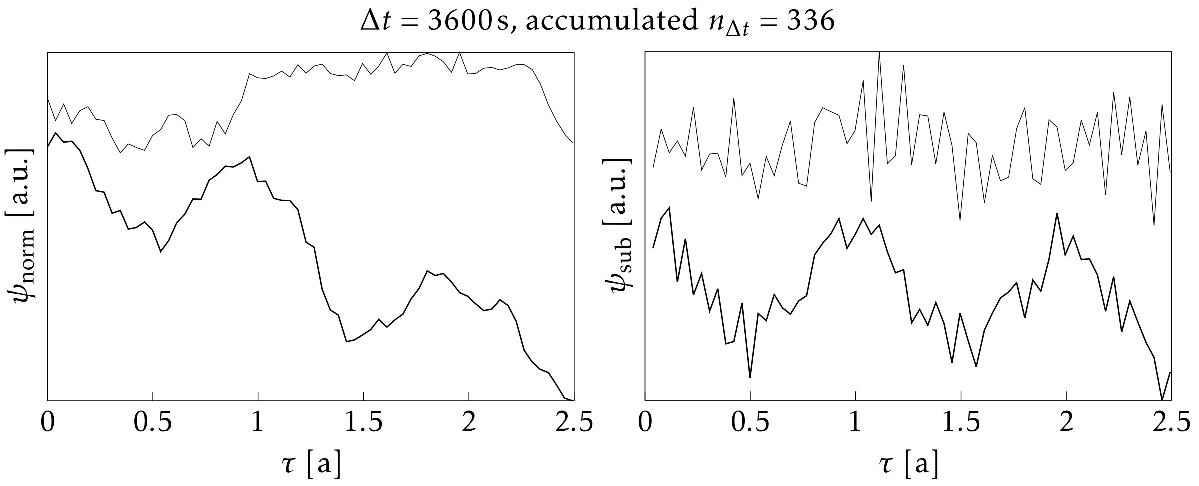


Figure 18 This plot shows the autocorrelation functions (shifted along the y axis for better presentation) for two simulation runs. Both graphs use the same parameters N_0 and ω obtained by the reconstruction and $\varepsilon = \tilde{\varepsilon} = 10^{-7}$. The lower (thick) graphs use the original seed of blind test "C". The upper (thin) ones are generated with another seed (1800565643). Where the oscillation is clearly visible in the original data, it is invisible in the newly generated data set. This shows that two simulations with the same parameters can lead to very different results.

5. Experimental setup

This section will discuss the experimental setup at the Atominstitut which was "spontaneously" installed to investigate the effects presented in [1] etc.

As already pointed out in the introduction it is important to take note of the fact that the experiment was not actually *designed* and therefore cannot achieve results of demanded accuracy.

Early parts of the thesis dealt with finding ways of refining the available setup and performing a small series of measurements. However, since, very quickly, the focus of the work shifted to the establishment of the theoretical background and the establishment of the simulation framework, this section touches on several aspects of a possible experimental setup and its improvements only very briefly. The actual measuring process and the evaluation of the gathered data was *not* part of the thesis.

The contents of this chapter rely mostly on personal communication with Dieter Hainz and Erwin Jericha and very basic introductory experimental work under their guidance.

5.1. Remarks on a real measurement

This section will briefly point out some basic principles about measuring activity of a radioactive sample. Generally, a detector detects the decay products of a radioactive decay process. Disintegration can happen via α-, β- and γ-decay. Different detection techniques are used for different particles. The detector used in this setup is a photon detector. As shown in [10] and [11] a manifold of measurement techniques is available and should be exploited when investigating such a controversial effect. Even more so considering the fact stressed by [16] that there is no reason to assume that all nuclides should be equally affected.

5.1.1. The detector and further influences

The detector is the crucial, non-trivial component in a *real* setup. In the simulation, its principle of operation has been entirely ignored! Apart from the mentionable fact that it

- comes with an inherent instability and thus inaccuracy (to a certain extent),
- wears out after a certain time (and needs to be maintained, vacuum),
- shows an energy dependent efficiency $\epsilon(E)$,
- saturates at very high count rates and
- demands continuous cooling, which automatically implies a constant interference with the setup,

the devices necessary for measuring (all coming with their own errors) and the working principle of the detector itself contain a significant effect: dead time.

As opposed to the so-called live time, the dead time is the time during which a detector did *not* register any events. Live and dead time put together will sum up to the real time (measurement duration).

A parameter fundamentally affecting the resulting dead time is the so-called shaping time of the spectroscopic amplifier, which is set by the experimenter. Every photon the detector registers triggers an electrical signal, which has to be amplified and formed into a pulse whose height corresponds to the energy of the detected photon. The time to build up the pulse is exactly the introduced shaping time. The longer the shaping time, the better the resolution. Note, though, that no signal can be processed during (at least) the defined shaping time of a preceding count. Thus, a high shaping time may ignore a high number of counts if the sample is very active.

Furthermore, the effect of saturation has to be considered for very high count rate; a state in which the detector provides no usable signal, due to a too large number of "incoming" events.

As pointed out in section 2.3.6 The solid angle in a real measurement setup (p. 36) the positioning of the sample and the detector relative to each other has a major impact on the count rate. In case of a germanium detector the area of the detecting unit is that of the semiconductor crystal's cross-section. The efficiency is determined by the length of the crystal (high-energy photons may pass the entire crystal before depositing their energy).

The back end of the measurement machinery is formed by a computer program, which evaluates the count numbers of a specific energy interval, so-called γ-lines, which are characteristic for the deployed sample(s). Those lines are the subject of interest. However, the lines of a certain energy range have to be separated from other lines in their vicinity (which should not be a big problem, if proper samples have been chosen) and from the Compton background (due to Compton scattering of the incident photons which leave the crystal before their total energy is deposited), whose intensity grows with the growing energy of a γ-line.

More precisely speaking, a measurement provides a histogram of events assigned to a set of channel numbers. Those channel bins represent the range of detectable γ-energies. The number of counts of several adjacent channels add up to the total number of events ascribed to a γ-line (including background). The area of this line (above the background) represents the γ-intensity in question.

Gaining triplets of data, namely

$$(\text{energy of } \gamma\text{-line, peak area, measuring time}) \ ,$$

is the task of an evaluation software for which it uses its *own* algorithms. Although this does not directly effect the detector, it shall be mentioned in this context.

Right at this point, the simulation described in section 3 Simulation (p. 47) cues in and provides data of an imaginary measuring cycle (the peak areas are represented by the count numbers ΔN).

Generally, all devices and the detector shall be called the hardware, next to the software and the sample.

5.1.2. List of parameters

Concluding the previous chapters, a setup offers several degrees of freedom, some of which can be controlled by the experimenter (marked with ∘), divided into roughly three main groups:

- sample (activity)
  - ∘ decay constant λ
  - ∘ order of $\mathcal{N}_0$
- detector and hardware
  - ∘ shaping time
  - – dead time
  - – efficiency $\epsilon(E)$
  - – accuracy of the detector
  - – stability (linear, fluctuating, . . .)
- measurement (resulting in a data set for analysis)

   ◦ yield (count number ΔN) in dependence of the distance d between detector and sample (factored in by $\Omega(d)$)
   ◦ measurement duration $\mathscr{T}$
   ◦ integration time interval Δt
   ◦ accuracy via current count number ΔN
- (software and algorithms)

5.2. Current setup

5.2.1. General description, signal chain

A measurement set up consists of three important parts: The sample, the hardware and the software. The hardware includes the detector, whose running requires a high voltage supply and constant cooling, and all devices to amplify and transform the analog signal of the detector into a digital signal. The signal is fed to a computer, on which a piece of software evaluates the data. The sample is mounted in front of the detector, both sitting on a rack, encased in an absorbing lead shielding.

The detector has a built-in pre-amplifier. The jagged signal of a registered count provided by it then gets shaped and boosted by the main amplifier. The duration of building the pulse is called shaping time – any event during the shaping time will be ignored. The higher the shaping time, the higher the resolution but the the more events are potentially lost. The analog signal is then converted to a digital signal. The multi channel analyzer then associates the pulse height with a channel. The former is a measure of the energy of the original photon which caused the signal. A sprectrum is thus a scaled probability distribution of the pulse heights.

A software evaluates the registered countrates. An energy calibration is necessary to associate a channel number with an energy value. It is performed with the aid of a calibrated test sample, the energies of the detected pulses of which are known.

The setup is shown in figure 19.

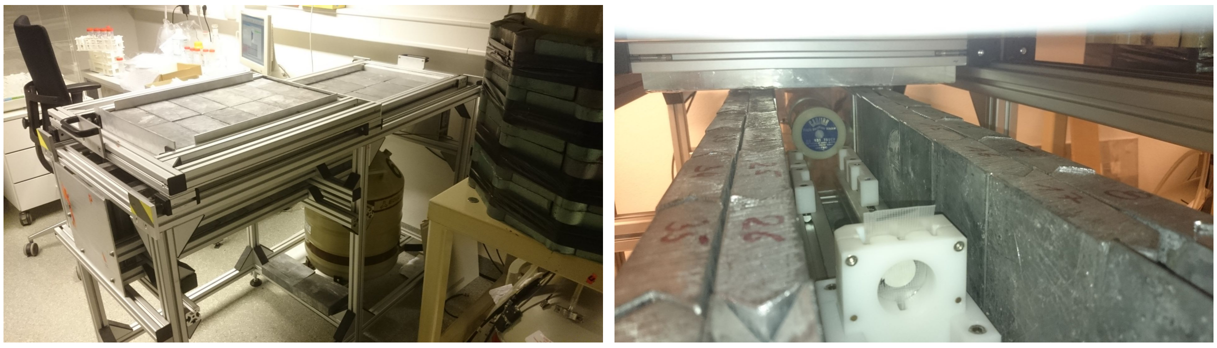


Figure 19 The current setup is pictured on the left side. The right image shows an inside view of the setup. The sample is mounted in the front, the detector window is visible in the rear.

5.2.2. Equipment used (samples, hardware and software)

Samples

The samples currently at our disposal are:

- Ba-133
- Am-241

- Cs-137
- Co-60

One sample at a time can be mounted in a custom made plastic mount. There are three slots in different distances from the detector for the mount to be placed in.

Hardware

The setup installed in 2012 uses the following hardware components:

- high-purity germanium solid-state photon detector[47] [26]
- built-in preamplifier
- liquid nitrogen cooling (refilled once a week)
- high voltage supply "ORTEC 454 Bias Supply" at $U = -2.9\,\mathrm{kV}$
- amplifier "ORTEC 570 Amplifier"
- analog-to-digital converter (ADC) "CANBERRA FAST ADC 8077"
- Multi Channel Analyzer (MCA-527)[48] [27]
- the personal computer of Fred Feuerstein (ATI 112-39) with an ancient interface necessary to communicate with our punch card like output card insulated with a high-end A4 paper sheet, constantly losing track of time in its longing for the bygone days before the Gregorian calendar... Joking aside: The setup *really* needs a "better" computer (one that features a USB port higher than version 1.0, if possible).

During maintenance work the high voltage supply "ORTEC 660 Dual Power Supply" was replaced with the device listed above. Be sure to never abruptly switch on or off the high voltage supply!

Software

The evaluation software in use is GENIE 2000 [28, 29].

It is necessary to perform an energy calibration in the course of which the 8192 channels of the MCA are associated with a certain energy. One can select regions of interest (ROI) around a peak in the spectrum. These ROIs will be included in the automatically generated evaluation report. Said report features information about the number of counts etc. at specific energies.

For the time of a computer damage, alternative software was used: WINSPEC FOR INSPECTORS (`*.spe` files). However, it turned out that it is insufficient for our purposes. Among other issues, the energy calibration parameters provided by the program greatly differed from those used for spectrum evaluation by the program itself! The program's algorithms seemed too incomprehensible and unstable. The measurements took place during a damage of the evaluation computer. In my personal opinion, the measurements and the software should be paid no attention at all. Efforts of combining the interim measurements with the past and future measurements would be a precarious task and are utterly futile.[49]

New gear

In the meantime, the institute afforded a new piece of hardware: A digital signal analyzer "CANBERRA DSA-LX" replacing the aforementioned high voltage supply, the amplifier, the ADC and the MCA. This also required a software update to GENIE 2000 V3.3.

[47] The data sheet of this approximately 30 year old detector is not available anymore.

[48] This particular device could also function as an amplifier and ADC, but is used as a MCA only (`direct` mode).

[49] The damage lasted from 16.06.14 until 08.07.14. No important measurements were performed. The detector was serviced shortly afterwards.

5.2.3. Detector service

After the computer damage, a detector service was performed in the time from 18.07.14 to 23.07.14. At first, the detector had to slowly warm up until it reached room temperature. Then it was heated to $T_C \approx 50\,°C$ crystal temperature ($50\,V \rightarrow T_{\text{heat}} \approx 80\,°C$) and evacuated to a pressure of $p \approx 2 \cdot 10^{-6}$ mbar over three days.

5.3. Problems and possible solutions

This section is devoted to the description of several design flaws and possible solutions, which must be considered carefully in the course of a new setup. High-precision measurements have been performed by [8] and [10] with astonishing effort. Considering the design of their experiments it is unimaginable to allow for high-precision analysis of the data gathered by the current setup.

Lead shielding

The shielding consists of rectangular lead elements which can be assembled according to the requirements of the experimental setup. Currently, it is arguably not meeting these requirements and does not provide low-level shielding. The radiation source and the detector are framed by said lead elements and covered with a sliding lid that also has a lead inlay. There is a 8 cm measuring gap between the "lead-walls" and the cover of the setup. This all round gap could be a problem.

I propose to either

⇒ close the gap with additional lead elements, or

⇒ increase the height of the lead shielding to a reasonable extent.

Nearby experiments and room access

Out of necessity, the experiment shares the room with multiple other experiments but their influence on our own measurements hasn't yet been estimated. The closest experiment – which uses radiation sources as well – is installed less than one meter away. Considering the very possibly insufficient shielding, a resulting effect can't be ruled out and has to be taken into account. Additionally, other experimenters enter the room randomly to maintain their own setups. Neither their actions, nor the resulting influence on our measurements can be captured.

⇒ In the best case, a separate room for the experiment is provided. At least the distance of the experiments has to be increased.

⇒ No matter if further separation is feasible or not, a "room journal" should be installed to backtrack events, if necessary.

⇒ For some reason we frequently found the setup opened several centimeters. We need to install a clasp to prevent that from happening.

Background noise

All the conditions mentioned above yet only strengthen the need for the – until now omitted – gathering of background noise data.

⇒ Collection of data with and without a sample is essential to judge over the impact of each of the possible sources of disturbances. The latter would give an impression of the background noise.

Triga reactor activity – neutrino interaction

Neutrinos – as they are produced by reactors – that originate from the spatially separated Triga reactor could effect the decay of the nuclei in the used samples. The distance of the reactor and the shielding provided by the facility are known thanks to construction plans. Knowledge about the precise run time of the reactor is essential, though.

$\Rightarrow$ The reactor journal has to be at our disposal during and after the evaluation of the data.

$\Rightarrow$ The influence of reactor neutrinos on the measured spectrum should be examined.

Cooling

The nitrogen cooling system has to be maintained on a regular basis to provide constant thermal conditions. The interval of this maintenance work has to be set with respect to the answer to the following question:

$\Rightarrow$ What difference of the temperature does show up (and how) in the obtained data?

Vibrations

The distance between sample and detector plays an important role. It has to be examined how far vibrations and convulsions cause changes of said distance and thus falsify the measured spectrum.

$\Rightarrow$ Evaluation of the solid angle change of the detector depending on changes of the distance between detector and sample.

$\Rightarrow$ Actively disturbing the setup by vibrations and examining the impact.

Monitoring

The detector feeds a computer via an interface with the obtained data. The clock of this computer tends to gain time continuously. Furthermore, there's a room thermometer to monitor the thermal conditions of the room. This device is not yet connected to the computer system.

$\Rightarrow$ It has to be decided whether (and how) this time difference is taken into account.

$\Rightarrow$ The temperature device has to be connected to the monitoring system.

$\Rightarrow$ Ambient pressure and humidity monitoring needs to be added.

Sample mounting

Currently, there is one custom made sample mount (plastic), which can hold one sample at a time and be inserted into three fixed slots. In a measurement with two samples, the second was stuck on the mount with duct tape.

$\Rightarrow$ There has to be built a sample mount which can hold more than one sample.

$\Rightarrow$ There should be made at least a second mount. If the plans of the old mount do not exist anymore, a new one should be designed.

Effect of various influences Before a new setup is established we should use the opportunity to do measurements under various disturbances and evaluate their impact on the resulting spectra. Since, according to [30], a cathode ray tube monitor can effect a measurement, it was replaced with an LCD screen on 23.07.14.

Additionally, measurements with an external oscilloscope have been made, showing intense noise – up to half the height of a regular pulse – which we could not backtrack. Shortly after, a detector service was performed. All cable connections should be thoroughly checked in a new setup! So should the sensitivity to vibrations by touching the detector etc.

5.4. Running the measurements

The longtime measurements were performed with $\Delta t = 20\,\text{min} = 1200\,\text{s}$. A Ba-133 sample was engaged.

Starting a measurement is fairly simple. The Genie 2000 can show the measuring process in real time or be programmed to measure for a certain period of time. Of course, the software is a very powerful tool!

5.4.1. Starting measurements

Apart from controlling via the software, a measurement can be launched via command line options. A `*.bat` file to start a measurement could look like:

```
FOR /L %%J IN (1 1 ⟨end index⟩) DO (
STARTMCA DET:POLYGAM /REALPRESET=⟨seconds (integer)⟩
WAIT DET:POLYGAM /ACQ
MOVEDATA DET:POLYGAM ⟨full/path/to/destination/filename without extension⟩%%J.CNF
)
```

`REALPRESET` is the measurement real time Δt (real time = live time + dead time). `%%J` represents the running index and enables the start of several measurements in a row. It can be included into the file name (note that there are no preceding zeros – for a consistent file name scheme multiple loops are necessary). Note that Windows paths are written `\`.

5.4.2. File formats, report

The `*.CNF` files hold the channel data of the measurement and can be evaluated by the Genie 2000 software. A report – an analysis of the registered events – can be output, which provides information about the gamma lines in the spectrum, the energy and the peak area. To obtain the report one has to run:

```
⟨path/to/Genie 2000's/⟩Analyze.exe ⟨path/to/source file⟩.CNF
    /SEQ=⟨path/to/Genie 2000's/⟩Peak_sho1.asf > ⟨path/to/destination file⟩.TXT
```

An exemplary report is provided in appendix B Genie 2000 sample report (p. 117).

6. Conlusion and outlook

6.1. Summary of findings

This thesis used the example of time dependent radioactive decay due to a postulated influence of the sun on nuclear decay rates ([1, 4] etc.) as an example to investigate by theoretical discussion and computer simulation to which extent (periodic) measurement signals of small amplitude can be isolated from a dominant (statistical and systematical) background.

Although the experiment pictured in 5 Experimental setup (p. 103) launched by Erwin Jericha originally aimed at refuting claims about such effects, this work is not trying to surmount arguments in favor of or to disprove such an effect by any means. It merely provides a framework which can be used and extended to investigate phenomena of such or similar nature. Considering the efforts being made in order to test these assumptions ([8–11] etc.) it is unlikely that data from this setup can deliver results of statistical significance. Nonetheless it may prove interesting to analyze these data using the framework – something which lies beyond the scope of this work.

In this text, simple physical models were discussed and implemented in the simulation framework. Several analysis methods were introduced, programmed and used on data sets which were created using the simulation.

The manifold of free parameters – the attributes of the sample, the parameters of the physical model and the measurement – makes it very hard to find all relations between them; especially because they often are of extreme (and extremely different) magnitudes packed in rather complicated equations.

The most suitable model to describe sinusoidal influence of the sun turned out (in the frame of this thesis) to be the sinusoidal decay constant model, which is used most of the time.

Where the standard decay subtraction method can help to find oscillations in a signal, only the standard decay normalization method can provide values for the effect's amplitude ε.

Interestingly, the autocorrelation (which at first seemed to be the ultimate tool for finding such hidden periodic fluctuations) sometimes inexplicably fails in the most obvious cases of periodicity in a normalized signal. The reason for that is unclear.

Yet again, this strengthens the following inference which came up multiple times during the theoretical, programming and testing work: All available (and new) tools should be used to analyze a signal; there is no such thing as an analysis recipe, it is often a matter of trial and error. Automatized analysis, at this point, seems to be highly unrealistic.

The analysis of blind tests showed that the means presented in this thesis can potentially provide good results and find amplitudes of $\varepsilon \approx 10^{-7}$. However, such figures always depend on other parameters such as N_0, λ etc.

Testing of and working with the analysis classes showed that it is best to reduce the signal processing to a minimum since it potentially introduces systematic errors. Furthermore, in case of a real measurement, multiple measurements should be performed in parallel due to the long measurement duration. It can be demonstrated (multiple simulations with unchanged parameters but different P-RNG seeds) that the results can look very differently in case of extreme parameter values (such as very low ε, for instance).

6.2. Other applications

As mentioned before, time dependent radioactive decay was merely an example to build this work on. Other situations in which the experimenter is investigating a small (periodic) signal superimposed on a dominant background may also profit from this framework. Its extensibility allows for the implementation of other physical models.

For instance, the analysis of interference spectra of low contrast in neutron interferometry experiments [31] may provide application for the framework.

Another example could be the analysis of high-resolution measurements of the neutron intensity of the “white beam” at the Triga reactor at the Atominstitut, which were performed by time-resolved neutron detection with diamond detectors.

6.3. Possible next steps...

After concluding the work and findings of the thesis in hand there remains a manifold of tasks to be tackled in the future. This section will briefly list possible next steps in the continuation of this work.

6.3.1. Physics and math

- The models of “discrete” and rectangular influences have not been implemented yet. It stands to reason to include them in the simulation, since the former represents the effects discussed in [4] and the latter describes the Triga reactor of the Atominstitut as a possible source of disturbance in any real measurement at the facility.
- The findings in [1] might encourage to pursue the influence of a free phase parameter in the models.
- New mathematical analysis tools should be implemented. Especially the autocorrelation needs to be improved: At the time being it is unclear why it sometimes fails in finding obvious oscillations in a signal of very small values. Additionally, more sophisticated implementations (such as in Matlab) might circumvent the restriction $\hat{\tau}_{\text{max}} = M_{\text{tot}}/2$.
- Fourier analysis of a measurement signal can also reveal periodicities in their spectrum.

6.3.2. Simulation

The simulation (as every computer program in the past, present or future) can be enhanced in many regards. First, some design issues will be listed.

- The models, parameters and utility classes are not consistently separated in their own translation units. It stands to reason to capsule the core of the framework (being the abstract parent classes, the sample class, the conversion functions and the utility classes) in their own translation units. New components like actual models or analysis methods should then be added in their own units (such as currently the analysis classes) as well.
- The aforementioned separation and structure might be supported by dedicated namespace declarations.
- The current documentation mechanism should be preserved, however, until now, the data columns of the files have no table header. Although the ample documentation within the file leaves no doubt about its contents, such table headers would indeed be “nice” to have. The implementation is not trivial, since it would demand some kind of updatable member field in the `FileDoc` class (also taking care to delete old table headers in chains of processing steps) *and* a mechanism to adapt to other data formats than the `data3` structure.
- So far, quantities like λ were assumed to be exact in the simulation! In fact, they are not. A new version of the simulation framework should introduce new datatype structures allowing to assign uncertainties to a given quantity. Operator overloads could provide basic error propagation calculations. Mind that by introducing such uncertainties any error propagation relations can become by far more complicated.

- In the course of major changes to the framework, a versioning convention should be included into the source files. Additionally, the version number should then be documented in the data file documentation in order to avoid compatibility and reproducibility issues.
- The option to pass `Data` objects to an `Analysis` object instead of providing a `File` object was not yet satisfactorily accounted for in the user feedback of the simulation.
- One might introduce a new abstract `Simulation` parent class to allow for more flexibility in the choice of physical models and calculation types. In that case, `class DecaySim : Simulation` could be the simulation class for time dependent radioactive decay models, for instance.

Possible improvements of the actual contents include:

- A mechanism to switch to Poisson distributed pseudo-random numbers in the `Simulation` class (or `RandomNumbers` class respectively) as soon as the given mean value falls below a critical value.
- The complete uncertainty propagation of the standard decay subtraction is not yet implemented and needs to be coded and tested. Given is only an approximation of the uncertainty.
- The findings in [1] encourage the implementation of a free phase parameter in the models.
- A sliding, non-weighted average could be included as a new kernel in the `Smooth` class.
- A graphical interface for real-time adjustment of the parameters of (standard) decay might help to understand the relations between them on a deeper level and is very much encouraged. This does not need to be a direct part of the framework!
- A graphical user interface in order to use the framework more quickly in the analysis process would be a major step, however, demands a lot of work and dedication.

6.3.3. Application

A lot more of the available blind tests should be analyzed in order to understand the possibilities and restrictions of the framework and the investigated problem even better.

The subtraction and normalization processing should also be tested with a value for the decay constant λ obtained from a fit, because using the correct value for λ in this tests might bias the results. Right now, only N_0 can be calculated from the data.

6.3.4. Experimental setup

After thorough investigation and proper enhancement of the current setup, new long-time measurements could be launched.

References

[1] J. H. Jenkins et al. “Evidence of correlations between nuclear decay rates and Earth–Sun distance”. In: *Astroparticle Physics* 32 (2009), pp. 42–46. DOI: 10.1016/j.astropartphys.2009.05.004.

[2] P. A. Sturrock et al. “Power Spectrum Analysis of Physikalisch-Technische Bundesanstalt Decay-Rate Data: Evidence for Solar Rotational Modulation”. In: *Solar Physics* 267 (2010), pp. 251–265. DOI: 10.1007/s11207-010-9659-4.

[3] D.E. Alburger, G. Harbottle, and E.F. Norton. “Half-life of ^{32}Si”. In: *Earth and Planetary Science Letters* 78 (1986), pp. 168–176. DOI: 10.1016/0012-821x(86)90058-0.

[4] J. H. Jenkins and E. Fischbach. “Perturbation of nuclear decay rates during the solar flare of 2006 December 13”. In: *Astroparticle Physics* 31 (2009), pp. 407–411. DOI: 10.1016/j.astropartphys.2009.04.005.

[5] J. H. Jenkins et al. *Analysis of Experiments Exhibiting Time-Varying Nuclear Decay Rates: Systematic Effects or New Physics?* 2011. arXiv: 1106.1678v1 [nucl-ex].

[6] E. Fischbach, J. H. Jenkins, and P. A. Sturrock. *Evidence for Time-Varying Nuclear Decay Rates: Experimental Results and Their Implications for New Physics*. 2011. arXiv: 1106.1470v1 [nucl-ex].

[7] E. Fischbach et al. *Evidence for Solar Influences on Nuclear Decay Rates*. 2010. DOI: 10.1142/9789814327688_0033. arXiv: 1007.3318v1 [hep-ph].

[8] E. Bellotti et al. *Search for the time dependence of the ^{137}Cs decay constant*. 2012. DOI: 10.1016/j.physletb.2012.02.083. arXiv: 1202.3662v1 [nucl-ex].

[9] J.C. Hardy, J.R. Goodwin, and V.E. Iacob. “Do radioactive half-lives vary with the Earth-to-Sun distance?” In: *Applied Radiation and Isotopes* 70 (2012), pp. 1931–1933. DOI: 10.1016/j.apradiso.2012.02.021.

[10] B. Goddard et al. “Experimental setup and commissioning baseline study in search of time-variations in beta-decay half-lives”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 812 (2016), pp. 60–67. DOI: 10.1016/j.nima.2015.12.026.

[11] S. Pommé et al. “Evidence against solar influence on nuclear decay constants”. In: *Physics Letters B* 761 (2016), pp. 281–286. DOI: 10.1016/j.physletb.2016.08.038.

[12] J. H. Jenkins, D. W. Mundy, and Ephraim Fischbach. “Analysis of environmental influences in nuclear half-life measurements exhibiting time-dependent decay rates”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 620 (2010), pp. 332–342. DOI: 10.1016/j.nima.2010.03.129.

[13] S. Lindner. “Untersuchung der zeitlichen Stabilität einer Ba-133-Quelle”. Atominstitut TU Wien, 2015.

[14] S. Lindner. “Statistische Analyse der Gammasprektren einer Ba-133-Quelle mit R”. Atominstitut TU Wien, 2013.

[15] Dudenverlag, ed. *Das Lexikon für Österreich in 20 Bänden*. "Radioaktivität". German. Vol. 14. 20 vols. 2006.

[16] D. E. Krause et al. “Searches for solar-influenced radioactive decay anomalies using Spacecraft RTGs”. In: *Astroparticle Physicsu* 36 (2012), pp. 51–56. arXiv: 1205.7015v1 [astro-ph.SR].

[17] Dudenverlag, ed. *Das Lexikon für Österreich in 20 Bänden*. "Apsiden", "Astronomische Einheit". German. Vol. 1. 20 vols. 2006.

[18] I. G. Hughes and T. P. A. Hase. *Measurements and their Uncertainties A practical guide to modern error analysis*. Oxford University Press, 2010.

[19] Wikipedia. *Stichprobenvarianz (Schätzfunktion) – Stichprobenstandardabweichung*. URL: https://de.wikipedia.org/wiki/Stichprobenvarianz_(Sch%C3%A4tzfunktion)#Stichprobenstandardabweichung (visited on 12/02/2017).

[20] S. Meyer and E. Schweidler. *Radioaktivität*. Naturwissenschaft und Technik in Lehre und Forschung. B.G. Teubner, 1927.

[21] J. Tsitsiklis. *Probabilistic Systems Analysis And Applied Probability – Poisson Process I*. MIT Open Courseware. 2011. URL: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-041-probabilistic-systems-analysis-and-applied-probability-fall-2010/video-lectures/lecture-14-poisson-process-i/ (visited on 12/02/2017).

[22] The C++ Resources Network 2016. *Class template reference: std::mersenne_twister_engine*. 2016. URL: http://www.cplusplus.com/reference/random/mersenne_twister_engine/ (visited on 12/02/2017).

[23] IAEA - Nuclear Data Services. *Live Chart of Nuclides (Guide)*. URL: https://www-nds.iaea.org/relnsd/vcharthtml/guide.html#gs_hs (visited on 12/02/2017).

[24] Laboratoire National Henri Becquerel. *Nucléide - Lara: Library for gamma and alpha emissions (keyword Ba-133)*. 2016. URL: http://www.nucleide.org/Laraweb/ (visited on 12/02/2017).

[25] T. Williams and C. Kelley. *gnuplot 5.2 – an interactive plotting program (manual)*. 2017.

[26] EG&G Ortec. *Gamma-X High-Purity Germanium Coaxial Solid-State Photon Detector System. Operators Manual.*

[27] GBS Elektronik. *MCA-527 Digital Multi-Channel Analyzer. User Manual*. 2012.

[28] Canberra Industries. GENIE 2000 *Spectroscopy Software. Customization Tools*. 2006.

[29] Canberra Industries. GENIE 2000 *Spectroscopy Software. Operations*. 2016.

[30] G. Gilmore and J. D. Hemingway. *Practical Gamma-Ray Spectrometry*. John Wiley & Sons, 1995.

[31] H. Rauch et al. "Low-contrast and low-counting-rate measurements in neutron interferometry". In: *Phys Rev A* 42 (1990), pp. 3726–3732. DOI: 10.1103/physreva.42.3726.

A. CD Attachment

This thesis includes a CD attachment containing:

- the C++ code (header and implementation files) of the simulation framework (refer to 3 Simulation (p. 47)),
- the Qt Creator project files,
- the Doxygen documentation (html) and the Doxygen file for its creation,
- the simulation results referred to in 4.2.1 Increasing sensitivity by standard decay subtraction/normalization (p. 77),
- the weighted and unweighted fit results referred to in 4.3.3 Comparison of the methods (p. 92) and
- more blind test data for further evaluation.

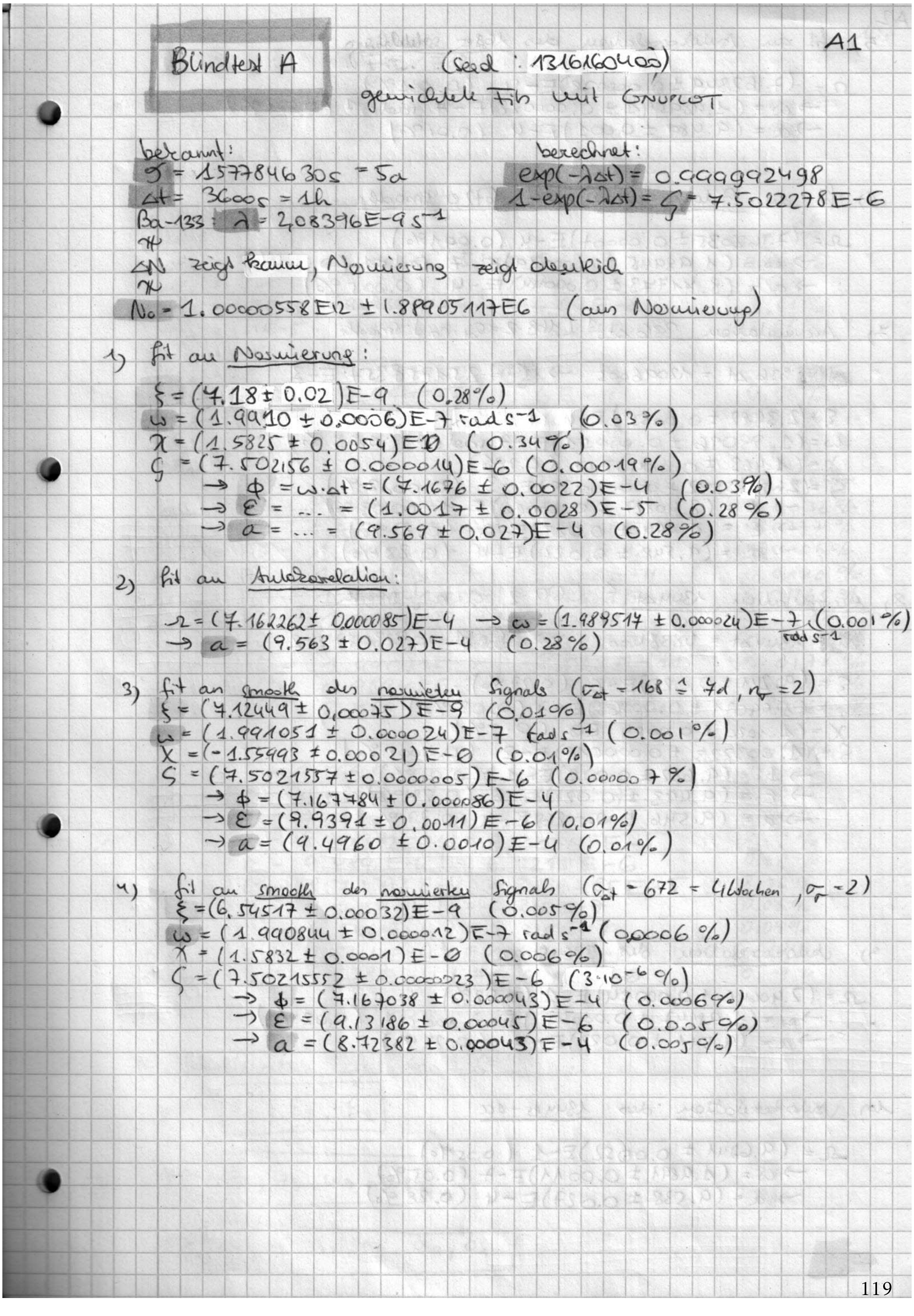
Please contact `lukas_csaszar@yahoo.de` if the attachment is missing.

B. Genie 2000 sample report

```

*************************************************************************
*****          G A M M A - S P E K T R U M - A N A L Y S E        ******
*************************************************************************


Dateiname:C:\GENIE2K\SHAPINGTIME_DISTANCE\ST_10\ST_10_D_GROSS_1.CNF

Bericht erstellt am             : 28.07.2014 16:00:10

Probentitel                     : Sample title.
Probenbeschreibung              :
Proben-Identifikation           :
Probentyp                       :
Probengeometrie                 :

Peaksuch-Empfindlichkeit        :    3.00
Bereich Peaksuche (Kanaele)     :    1 - 65535
Bereich Peakflaechen (Kanaele)  :    1 - 65535
Identifizierungs-Energietoleranz:    1.000 keV

Probenmenge                     :  1.000E+000 Unit

Probe entnommen am              :
Messung gestartet am            : 28.07.2014 15:00:42

Dead-time                       :        0.3 Sekunden
Real-time                       :      600.0 Sekunden

Live-time                       :  99.94 %


              Benutzte Energie-Kalibrierung vom     : 28.07.2014
              Benutzte Effizienz-Kalibrierung vom   : ??????????
              Effizienz ID                          :

```

```


--- <Next Page> ---

Peakanalyse-Bericht              28.07.2014 16:00:11             Seite  2


**************************************************************************
*****          P E A K A N A L Y S E  - B E R I C H T               *****
**************************************************************************

     Detektorname    POLYGAM
     Probentitel:    Sample title.
     Peakanalyse durchgefuehrt am: 28.07.2014 16:00:11
     Peakanalyse Anfangskanal:        1
     Peakanalyse Endkanal:         8192

   Peak  ROI  ROI    Peak    Energie  FWHM      Net Peak   Net Flaeche   Untergrund
    Nr. Anf. Ende    Lage     (keV)   (keV)     Flaeche      Fehler         Flaeche

 M  1    56-   84    62.42    27.93   2.39         43.32    7.50          17.86
 m  2    56-   84    74.90    32.06   2.40        146.36   12.61          63.50
    3   152-  165   157.92    59.54   1.31        470.48   30.69         170.52
    4   201-  210   205.54    75.29   1.16        110.50   24.29         190.50
 M  5   229-  249   233.85    84.66   1.31         87.24   12.36          95.03
 m  6   229-  249   243.99    88.02   1.31        187.99   16.15         109.35
    7   340-  354   347.08   122.13   1.51         68.86   19.35         106.14
    8  1155- 1169  1161.51   391.67   0.92         57.25   12.44          33.75
    9  1969- 1987  1977.17   661.60   1.88        232.36   18.51          32.64
   10  2685- 2700  2692.09   898.20   0.55         51.31    9.91          15.69
   11  3512- 3533  3523.50  1173.35   1.68        200.49   14.88           5.51
   12  3995- 4015  4004.94  1332.68   2.12        154.33   14.35          14.67

M = Erster Peak in einer Multiplett-Region
m = Weiterer Peak in der Multiplett-Region
F = Gefittetes Singlet

Fehler in  1.000 Sigma angegeben
```

C. Evaluation notes for blind test “A”



A1

Blindtest A (Seed : 1316160400)

gewichtete Fits mit GNUPLOT

bekannt:

$\tau = 157784630\,s = 5a$

$\Delta t = 3600\,s = 1h$

Ba-133 : $\lambda = 2{,}08396E{-}9\,s^{-1}$

χ

ΔN zeigt kaum, Normierung zeigt deutlich

χ

$N_0 = 1.00000558E12 \pm 1.89905117E6$ (aus Normierung)

berechnet:

$\exp(-\lambda\Delta t) = 0.999992498$

$1-\exp(-\lambda\Delta t) = \zeta = 7.5022278E{-}6$

1) fit an Normierung:

$\xi = (7.18 \pm 0.02)E{-}9$ (0.28%)

$\omega = (1.9910 \pm 0.0006)E{-}7\ rad\,s^{-1}$ (0.03%)

$\chi = (1.5825 \pm 0.0054)E10$ (0.34%)

$\zeta = (7.502156 \pm 0.000014)E{-}6$ (0.00019%)

$\rightarrow \phi = \omega\cdot\Delta t = (7.1676 \pm 0.0022)E{-}4$ (0.03%)

$\rightarrow \varepsilon = \ldots = (1.0017 \pm 0.0028)E{-}5$ (0.28%)

$\rightarrow a = \ldots = (9.569 \pm 0.027)E{-}4$ (0.28%)

2) fit an Autokorrelation:

$\Omega = (7.162262 \pm 0.000085)E{-}4 \rightarrow \omega = (1.989517 \pm 0.000024)E{-}7\ rad\,s^{-1}$ (0.001%)

$\rightarrow a = (9.563 \pm 0.027)E{-}4$ (0.28%)

3) fit an smooth des normierten Signals ($\tau_{\Delta t} = 168 \mathrel{\hat{=}} 7d$, $\tau_\sigma = 2$)

$\xi = (7.12449 \pm 0{,}00075)E{-}9$ (0.01%)

$\omega = (1.991051 \pm 0.000024)E{-}7\ rad\,s^{-1}$ (0.001%)

$\chi = (-1.55993 \pm 0.00021)E{-}0$ (0.01%)

$\zeta = (7.5021557 \pm 0.0000005)E{-}6$ (0.000007%)

$\rightarrow \phi = (7.167784 \pm 0.000086)E{-}4$

$\rightarrow \varepsilon = (9.9391 \pm 0.0011)E{-}6$ (0.01%)

$\rightarrow a = (9.4960 \pm 0.0010)E{-}4$ (0.01%)

4) fit an smooth des normierten Signals ($\sigma_{\Delta t} = 672 = 4$ Wochen, $\sigma_\sigma = 2$)

$\xi = (6.54517 \pm 0.00032)E{-}9$ (0.005%)

$\omega = (1.990844 \pm 0.000012)E{-}7\ rad\,s^{-1}$ (0.0006%)

$\chi = (1.5832 \pm 0.0001)E{-}0$ (0.006%)

$\zeta = (7.50215552 \pm 0.00000023)E{-}6$ ($3\cdot 10^{-6}$ %)

$\rightarrow \phi = (7.167038 \pm 0.000043)E{-}4$ (0.0006%)

$\rightarrow \varepsilon = (9.13186 \pm 0.00045)E{-}6$ (0.005%)

$\rightarrow a = (8.72382 \pm 0.00043)E{-}4$ (0.005%)

A2

5) fit an <u>Autokorrelation</u> des <u>168er smooths</u>

$\Omega = (7.162349 \pm 0.000078)E-4 \quad (0.001\%)$
$\rightarrow \omega = (1.989541 \pm 0.000022)E-7$ rad s^{-1} (0.001%)
$\rightarrow a = (9.489 \pm 0.001)E-4 \quad (0.01\%)$

6) fit an <u>Autokorrelation</u> des <u>672er smooths</u>

$\Omega = (7.16203 \pm 0.00007)E-4 \quad (0.001\%)$
$\rightarrow \omega = (1.98945 \pm 0.00019)E-7$ rad s^{-1} (0.001%)
$\rightarrow a = (8.41773 \pm 0.00044)E-4 \quad (0.005\%)$

7) <u>Accumulation 336 Δt</u> $\hat{=}$ $[168 \cdot 2 = \sigma_{\Delta t} \cdot n_\sigma]$-smooth

$\Delta t' = 336\,\Delta t = 1209600s \rightarrow \varsigma = 2.51758357E-3$

$\xi = (2.3982 \pm 0.0055)E-6 \quad (0.23\%)$
$\omega = (1.99096 \pm 0.00051)E-7$ rad s^{-1} (0.03%)
$\chi = (1.463 \pm 0.004)E0 \quad (0.3\%)$
$\varsigma = (2.5175592 \pm 0.0000039)E-3 \quad (0.0002\%)$
$\rightarrow \phi = (2.40827 \pm 0.00062)E-1 \quad (0.03\%)$
$\rightarrow \varepsilon = (9.994 \pm 0.023)E-6 \quad (0.23\%)$
$\rightarrow a = (9.548 \pm 0.022)E-4 \quad (0.23\%)$

8) <u>Accumulation 1344 Δt</u> $\hat{=}$ $[672 \cdot 2 = \sigma_{\Delta t} n_\sigma]$-smooth

$\Delta t' = 1344\,\Delta t = 4838400 \rightarrow \varsigma = 1.00323687E-2$

$\xi = (9.211 \pm 0.025)E-6 \quad (0.27\%)$
$\omega = (1.99071 \pm 0.00063)E-7 \quad (0.03\%)$
$\chi = (1.1020 \pm 0.0054)E0 \quad (0.5\%)$
$\varsigma = (1.003223 \pm 0.000018)E-2 \quad (0.0002\%)$
$\rightarrow \phi = (9.632 \pm 0.003)E-1 \quad (0.03\%)$
$\rightarrow \varepsilon = (9.993 \pm 0.027)E-6 \quad (0.27\%)$
$\rightarrow a = (9.546 \pm 0.026)E-4 \quad (0.27\%)$

9) <u>Autokorrelation</u> des <u>336er-acc</u>

$\Omega = (2.40646 \pm 0.00054)E-1 \quad (0.02\%)$
$\rightarrow \omega = (1.98947 \pm 0.00045)E-7$ rad s^{-1} (0.02%)
$\rightarrow a = (9.541 \pm 0.022)E-4 \quad (0.23\%)$

10) <u>Autokorrelation</u> des <u>1344er-acc</u>

$\Omega = (9.6241 \pm 0.0052)E-1 \quad (0.05\%)$
$\rightarrow \omega = (1.9891 \pm 0.0011)E-7 \quad (0.05\%)$
$\rightarrow a = (9.538 \pm 0.027)E-4 \quad (0.28\%)$

Auflösung: $\tilde{N}_2 = 1E12$
$\tilde{\varepsilon} = 1E{-}5$
$\tilde{\omega} = 1.9910638E{-}7$ rad s^{-1}
$\tilde{\alpha} = 9.554215E{-}4$

Analyse:

- $\Delta\mu_i =: |\tilde{\mu} - \mu_i| \Rightarrow$

		$\delta_\varepsilon =: \frac{\Delta\varepsilon}{\tilde{\varepsilon}}$
$\Delta\varepsilon_1$	$= \lvert 1E{-}5 - 1.0017E{-}5 \rvert = 1.7E{-}8$	0.17 %
$\Delta\varepsilon_2$	= /	
$\Delta\varepsilon_3$	$= \lvert 1E{-}5 - 9.9391E{-}6 \rvert = 6.09E{-}8$	0.61 %
$\Delta\varepsilon_4$	$= \lvert 1E{-}5 - 9.13186E{-}6 \rvert = 8.6814E{-}7$	8.68 %
$\Delta\varepsilon_5$	= /	
$\Delta\varepsilon_6$	= /	
$\Delta\varepsilon_7$	$= \lvert 1E{-}5 - 9.994E{-}6 \rvert = 6E{-}9$	0.06 %
$\Delta\varepsilon_8$	$= \lvert 1E{-}5 - 9.993E{-}6 \rvert = 7E{-}9$	0.07 %
$\Delta\varepsilon_9$	= /	
$\Delta\varepsilon_{10}$	= /	

		$\delta_\omega =: \frac{\Delta\omega}{\tilde{\omega}}$
$\Delta\omega_1$	$= \lvert \tilde{\omega} - 1.9910E{-}7 \rvert = 6.38E{-}12$	0.0032 %
$\Delta\omega_2$	$= \lvert \tilde{\omega} - 1.989517E{-}7 \rvert = 1.5468E{-}10$	0.0777 %
$\Delta\omega_3$	$= \lvert \tilde{\omega} - 1.991051E{-}7 \rvert = 1.28E{-}12$	0.0006 %
$\Delta\omega_4$	$= \lvert \tilde{\omega} - 1.990844E{-}7 \rvert = 2.198E{-}11$	0.0110 %
$\Delta\omega_5$	$= \lvert \tilde{\omega} - 1.988541E{-}7 \rvert = 1.5228E{-}10$	0.0765 %
$\Delta\omega_6$	$= \lvert \tilde{\omega} - 1.98945E{-}7 \rvert = 1.6138E{-}10$	0.0811 %
$\Delta\omega_7$	$= \lvert \tilde{\omega} - 1.99096E{-}7 \rvert = 1.038E{-}11$	0.0052 %
$\Delta\omega_8$	$= \lvert \tilde{\omega} - 1.99071E{-}7 \rvert = 3.538E{-}11$	0.0178 %
$\Delta\omega_9$	$= \lvert \tilde{\omega} - 1.98947E{-}7 \rvert = 1.5938E{-}10$	0.0800 %
$\Delta\omega_{10}$	$= \lvert \tilde{\omega} - 1.9891E{-}7 \rvert = 1.9638E{-}10$	0.0986 %

		$\delta_a = \Delta a / \tilde{a}$
Δa_1	$= \lvert \tilde{a} - 9.569E{-}4 \rvert = 1.4785E{-}6$	0.15 %
Δa_2	$= \lvert \tilde{a} - 9.563E{-}4 \rvert = 8.785E{-}7$	0.09 %
Δa_3	$= \lvert \tilde{a} - 9.4960E{-}4 \rvert = 5.8215E{-}6$	0.61 %
Δa_4	$= \lvert \tilde{a} - 8.72382E{-}4 \rvert = 8.30395E{-}5$	8.69 %
Δa_5	$= \lvert \tilde{a} - 9.489E{-}4 \rvert = 6.5215E{-}6$	0.68 %
Δa_6	$= \lvert \tilde{a} - 8.71773E{-}4 \rvert = 8.36485E{-}5$	8.76 %
Δa_7	$= \lvert \tilde{a} - 9.548E{-}4 \rvert = 6.215E{-}7$	0.07 %
Δa_8	$= \lvert \tilde{a} - 9.546E{-}4 \rvert = 8.215E{-}7$	0.09 %
Δa_9	$= \lvert \tilde{a} - 9.541E{-}4 \rvert = 1.3215E{-}6$	0.14 %
Δa_{10}	$= \lvert \tilde{a} - 9.538E{-}4 \rvert = 1.6215E{-}6$	0.17 %

- behalte alle $\Delta\mu_i < 1.0\tilde{\sigma}$:

$\Rightarrow \boxed{\varepsilon_1, \varepsilon_7, \varepsilon_8}$

$\Rightarrow \boxed{\omega_1, \omega_3, \omega_7, \omega_8}$

$\Rightarrow \boxed{a_1, a_2, a_7, a_8, a_9, a_{10}}$

A4

• sortiere nach aufsteigendem σ_i (je kleiner desto besser)

$\Rightarrow \boxed{\varepsilon_7, \varepsilon_8, \varepsilon_1}$ $\qquad$ $(\dots) \rightsquigarrow$ kommt nicht in allen Parametern vor.

$\Rightarrow \boxed{\omega_3, \omega_7, \omega_1, \omega_8}$

$\Rightarrow a_7 \hat{=} (a_9),\ a_8,\ a_1 \hat{=} (a_2) \hat{=} (a_{10})$

• sortiere nach aufsteigendem $\Delta\mu_i$ (je kleiner desto besser)

$\Rightarrow \boxed{a_7,\ a_8,\ (a_2),\ (a_9),\ a_1,\ (a_{10})}$

• bewerte die Methoden über $r_{\mu_i} =: \Delta\mu_i / \sigma_i$

abgelehnt

$r_{\varepsilon 3,5} = 55.36$	$r_{\omega 2} = 64.45$	$r_{a3} = 58.22$
$r_{\varepsilon 4,6} = 1929.2$	$r_{\omega 4} = 18.32$	$r_{a4} = 1931.15$
	$r_{\omega 5} = 69.22$	$r_{a5} = 65.22$
	$r_{\omega 6} = 8.49$	$r_{a6} = 1901{,}10$
	$r_{\omega 9} = 3.54$	
	$r_{\omega 10} = 1.79$	

#

akzeptiert

$r_{\varepsilon 1,2} = 0.61$	$r_{\omega 1} = 0.11$	$r_{a1} = 0.55$
$r_{\varepsilon 7,9} = 0.26$	$r_{\omega 3} = 0.53$	$r_{a2} = 0.33$
$r_{\varepsilon 8,10} = 0.26$	$r_{\omega 7} = 0.20$	$r_{a7} = 0.28$
	$r_{\omega 8} = 0.56$	$r_{a8} = 0.32$
		$r_{a9} = 0.60$
		$r_{a10} = 0.60$

Autokorrelation :	2, 5, 6, 9, 10
Accumulation :	7, 8, 9, 10
Smooth :	3, 4, 5, 6