Karlsruher Institut für Technologie

FAKULTÄT FÜR PHYSIK

PHYSIKALISCHES PRAKTIKUM FÜR FORTGESCHRITTENE

PRAKTIKUM MODERNE PHYSIK

Gruppe Nr. 102 Kurs: **Mo1** ~~**Mo2**~~ **Mi3** WS 23/24

zutreffendes bitte ankreuzen — aktuelles Semester angeben

Versuch: Winkelkorrelation

Namen: Carl Jarschke ()

Jona Umlauf ()

Assistent: Svenja Heyns

durchgeführt am: 13.11.2023

Protokollabgabe am: 27.11.2023

vom Betreuer auszufüllen

Note gesamt [+] [0] [-]

Anerkannt: _____

(Datum Unterschrift)

Datum Rückgabe: _____

Bemerkung:

Inhaltsverzeichnis

1 Ziel des Versuchs, Theoretische Grundlagen

1.1 Radioaktive Zerfälle

Es gibt verschiedene Arten von radioaktiven Zerfällen, ihnen allen ist gemein, dass die Baryonenzahl (Protonen, Neutronen, etc) und die Ladungen erhalten sind.

1.1.1 α Zerfall

Beim α-Zerfall zerfällt ein Teilchen unter Aussendung eines α-Quants (^4He-Kern), dabei reduziert sich die Ordungszahl des Kerns um 2 und die Massenzahl um 4.
Aufgrund der erhaltenen Größen, erhält das *alpha*-Teilchen eine diskrete Energie, woraus eine monochromatische Strahlung resultiert.
Zustande kommt dieser Effekt durch den quantenmechanischen Tunneleffekt. Dieser sorgt dafür, dass Teilchen, welche mit einer Potentialwand konfrontiert werden eine endliche Wahrscheinlichkeit besitzen hinter der Potentialwand zu gefunden zu werden. Dies führt dazu, dass das *alpha*-Teilchen die starke Wechselwirkung des Kerns überwinden kann.

1.1.2 β Zerfall

Sogenante β Zerfälle beschreiben die Klassen von radioaktiven Zerfallsprozessen, bei denen die Kernladungszahl sich um eins erhöht, und die Massenzahl konstant bleibt.
Dabei unterscheidet man drei verschiedene Formen.

β^--Zerfall

Der β^- -Zerfall besteht aus dem Zerfall eines Neutrons in ein Proton, ein Elektron und ein Elektron-Antineutrino, wobei das Proton die Kernladungszahl um 1 erhöht und Elektrion und Antineutrino emittiert werden. In ihrem Fall wirkt die starke Kernkraft nicht, die bei α Zerfall das α-Teilchen daran hindert sich (in den meisten Fällen) vom Atomkern zu trennen, da es sich bei ihnen um Leptonen handelt. Aus diesem Grund können Elektron und Antineutrion einfach emittiert werden.[Wika]

β^+-Zerfall

Bei dieser Zerfallssorte wird ein Proton in ein Neutron umgewandelt, wobei ein Positron und ein Elektronneutrino entstehen, die ausgesendet werden. Dabei bleibt die Massenzahl des Kerns erhalten, die Kernladungszahl verringert sich jedoch um 1.

Elektroneneinfang

Der Elektroneneinfang ist ein Prozess bei dem genau wie beim β^+-Zerfall ein Proton in ein Neutron umgewandelt wird, das hier entstehende Positron wird allerdings mit einem kernnahen Elektron vernichtet und so ein Neutrino erzeugt und emittiert.
Dadurch, dass das Elektron mit dem Positron vernichtet wird, entsteht an seiner Stelle eine

Leerstelle, deren Energieniveau niedriger ist, als die Energieniveaus anderer Elektronen in kernferneren Hüllen.
Dieser Unterschied in den Energien sorgt dafür, dass ein Elektron aus einem höher energetischen Niveau den Platz des vernichteten Elektrons besetzt. Die dabei frei werdende Energie wird in Form eins Photons ausgesendet. Diese Art von Strahlung wird auch charakteristische Röntgenstrahlung genannt.

1.1.3 γ Zerfall

Wenn der α- oder β-Zerfall den Kern nicht in einen Grundzustand übergehen lässt, sondern in einen angeregten Zustand, geht dieser unter Emission von γ-Strahlung (Photonen) in den Grundzustand über.

1.2 Zerfall von ^{60}Co

Der Zerfall von ^{60}Co erfolgt in 99,88 % der Fällen in zwei Schritten. Zuerst zerfällt das radioaktive Isotop in einem β^- Zerfall in einen angeregten Zustand von ^{60}Ni (Spin(4^+)) um dann unter Aussendung eines oder zweier Gammaquanten in den Grundzustand von ^{60}Ni über zu gehen. [Wikb]

1.3 Winkelkorrelation

Die Aussendung elektromagnetischer Strahlung bei einem Übergang aus einem J_1 in den J_2 Zustand ist isotrop, wenn gilt:

- Die $2J_1 + 1$ Unterzustände sind gleich besetzt
- alle zwischen J_1, m_1 und J_2, m_2 möglichen Übergänge werden beobachtet.

Dabei ist es wichtig, dass nicht die Einzelintensitäten isotrop sind, sondern erst ihre Überlagerungen.

$$G_{Ges} = \sum_i W_i d\Omega \tag{1.1}$$

Um diese Isotropie aufzuheben gibt es verschiedene Möglichkeiten. Eine von ihnen ist die Winkelkorrelation.
Dabei wird eine gleichzeitige Messung zweier Übergänge durchgeführt, die aufeinander folgen. Es wird also zum Beispiel der Übergang von J_1 zu J_3 über den Zustand J_2 beobachtet. Wenn nun das erste Quant nachgewiesen wird, kann der Zwischenzustand J_2 nicht mehr besetzt sein, wodurch die Isotropie für das zweite Quant aufgehoben ist.
Für das zu untersuchende Element führt dies aufgrund der ausgesendeten Quadrupolstrahlung und dem Zwischenzustand J_2 mit Spin = 2 dazu, dass die Korrelationsfunktion $K(\theta)$:

$$K(\theta) = \frac{d\sigma(\theta)}{d\Omega} / \frac{d(90^\circ}{d\Omega}$$

wird zu

$$K(\theta) = 1 + a_2 \cos^2\theta + a_4 * \cos^4\theta, \tag{1.2}$$

wobei laut Theorie $a_2 = \frac{1}{8}$ und $a_4 = \frac{1}{24}$ gilt. Für die Anisotropie A_n der Abweichung der Winkelkorrelation von einer isotropen Verteilung gilt:

$$A_n = \frac{d\sigma(180°)/d\Omega - d\sigma(90°)/d\Omega}{d\sigma(90°)/d\Omega} \tag{1.3}$$

$$A_n = K(180°) - 1 \tag{1.4}$$

1.4 Koinzidenz

Zwei Phänomene sind koinzident, wenn sie gleichzeitig (in diesem Experiment gleichbedeutend mit im einem gewissen Zeitintervall (Koinzidenzzeit)) stattfinden. Bei der Messung der Winkelkorrelation ist dies wichtig, da aus Koinzidenz darauf geschlossen werden kann, dass zwei Phänomene die gleiche Ursache besitzen können. Ist dies der Fall, so wird von echter Koinzidenz gesprochen.
Allerdings kann es auch zufällig passieren, dass zwei Phänomene zur selben Zeit geschehen, ohne dass es zwischen ihnen eine Verbindung gibt. Dies nennt sich zufällige Koinzidenz

1.4.1 Echte Koinzidenz

Für die Anzahl auftretender echter Koinzidenzen N_k gilt:

$$N_k = \epsilon_1 \epsilon_2 A f(\theta) \tag{1.5}$$

wobei $f(\theta) \approx 1$ und die A Aktivität sind und ϵ_i Wahrscheinlichkeiten der Detektoren auszuschlagen sind.

1.4.2 Zufällige Koinzidenz

Für die zufällige Koinzidenz existiert eine quadratische Abhängigkeit von der Aktivität der Probe:

$$N_Z = \tau_A \epsilon_1 \epsilon_2 A^2 = \tau_A N_1 N_2 \tag{1.6}$$

Wobei es sich bei τ_A um die Auflösungszeit des Detektorsystems handelt.
Aus den zwei Formeln für die verschiedenen Koinzidenzen lässt sich das Verhältnis der beiden berechnen

$$\frac{N_k}{N_Z} = \frac{f(\theta)}{A\tau_A} \approx \frac{1}{A\tau_A}. \tag{1.7}$$

Aufgrund dieses Zusammenhangs mit der Aktivität der Quelle A ist es nicht sinnvoll eine Quelle mit zu hoher Aktivität zu wählen, da bei höherer Aktivität mehr zufällige Koinzidenzen auftreten und so die Messung der echten Koinzidenz zunehmend verfälscht wird.

1.4.3 Koinzidenzzeit

Die Koinzidenzzeit ist die Zeit, innerhalb derer die zwei Ereignisse als gleichzeitig, betrachtet werden. Wird sie zu groß gewählt steigt die Zahl der zufälligen Koinzidenzen, ist sie zu klein werden zu wenige echte Koinzidenzen erkannt.

1.5 Koinzidensschaltung

Wichtig für die Messung der Koinzidenz ist nicht die Höhe der Peaks, sondern die zeitliche Auflösung der Messung, damit das gleichzeitige Geschehen zweier Messungen erfasst werden kann.
Um dies zu erreichen wird auf eine Vor- und Nachverstärkung verzichtet und direkt ein Diskriminator an den Detektor angeschlossen.
Der Vorverstärker dient dazu, die ankommenden Signale für die Diskriminatorstufe vorzubereiten, damit diese sie richtig analysieren kann.
Der Nachverstäker tut selbiges nach der Diskriminatorstufe.
Dies dient zur Reduktion des Hintergrundrauschens und einer besseren Auflösung der Messung was die Impulshöhe anbelangt
Allerdings verschlechtert es die zeitliche Auflösung, weshalb die Verstärker für das Experminent nicht benutzt werden.
Ein Diskriminator wird dazu verwendet die gesuchten Impulse vom Hintergrundrauschen zu trennen, indem er beim überschreiten einer bestimmten voreingestellten Schwelle ein Signal triggert. Da es aber aufgrund dieses Triggervorgangs in den beiden Detektoren und aufgrund von Laufzeitunterschieden in der Signalverabreitung der beiden Detektoren zu einer unterschiedlichen Messzeit für zwei koinzidente Ereignisse kommen kann, ist zwischen den Zähler und den Diskriminator eine Verzögerung eingebaut, welche diese Unterschiede korrigiert. Indem sie die früher ankommende Messung gegenüber der späteren verzögert, sodass zur selben Zeit an den Detektoren angekommene Ereignisse auch zur selben Zeit gemessen werden.
Um die Koinzidenz zu bestimmen, ist noch ein dritter Zähler vonnöten, dieser schlägt aus, wenn innerhalb der Koinzidenzzeit bei beiden Detektoren ein Ereignis registriert wurde.

1.6 Poissonverteilung

Bei der Poissonverteilung handelt es sich um eine diskrete Wahrscheinlichkeitsverteilung bei der nur Werte größer gleich 0 angenommen werden.

$$P_\lambda(k) = \frac{\lambda^k}{k!}\dot{e}^{-k} \tag{1.8}$$

$$k \in \mathbb{N} \tag{1.9}$$

Die Poissonverteilung modelliert die Anzahl an Ergebnissen/Ereignissen, die bei konstanter Rate in einem festen Zeitintervall eintreten. Dies ist immer der Fall, wenn man radioaktive Strahlung detektiert, welche näherungsweise eine konstante Rate/ Aktivität aufweist. Dadurch eignet sie sich gut für das Experiment.

2 Experimenteller Aufbau

Der Aufbau besteht aus einer Strahlungsquelle (^{60}Co)und zwei Detektoren, von denen der eine um die Strahlungsquelle rotieren kann.
Dieser wird für die verschiedenen Messungen auf 0, 45 und 90 Grad ausgelenkt. (Da sich die beiden Detektoren bei 0° genau gegenüberstehen (180 °) entspricht dies einem Winkel zwischen den Detektoren von (180 °, 135 ° und 90 °)
Die Detektoren sind mit einem Koninzidenzmesser verbunden, dessen Daten mithilfe des Rechners und eines Skriptes aufgenommen werden. Die daraus gespeicherten Binärdaten werden mit einem weiteren Skript in ein csv-File übersetzt. Siehe 5
Es wurde mit Natrium 22 kalibriert, dessen emittiertes Gammaquant eine Energie von 1274.5 keV (siehe [Lei]) besitzt.

3 Durchführung

3.1 Kalibrierung mit ^{22}Na

Zuerst werden für beide Detektoren gleichzeitig eine Messreihe von ^{22}Na aufgenommen. Dabei werden die Detektorspannungen so gewählt, dass die Peakhöhe der charakteristischen Peaks von ^{22}Na, also die Anzahl der gemessenen Ereignisse bei charakteristischer Strahlungsenergie, ungefähr gleich sind. Die im Spektrum zu erkennenden Peaks werden dann durch Anpassung an eine Gauß-Funktion bestimmt und die die Kanalnummern werden durch eine lineare Anpassung an die theoretischen Energien der charakteristischen Peaks durch die Strahlungsenergie ersetzt. Zur Energiekalibrierung werden die Werte der Photopeaks von ^{22}Na, die bei 1274.5 keV und 511 keV [Lei] liegen. Somit sind für jeweils beide Detektoren die Kanäle durch die zugehörigen Energien der Gammastrahlung dargestellt.

3.2 Messung der Strahlung für verschiedene Gradzahlen

Mithilfe des verschiebbaren Armes wird der zweite Detektor gedreht. Dabei wird der Arm auf 0, 45 und 90 Grad gestellt, und jeweils zweimal 400 s gemessen. Dieser Ablauf wird zweimal durchlaufen um statistische Fehler zu minimieren.
Um den Winkel so präzise wie möglich einzustellen wird ein gerades Objekt benutzt um die Kante des Detektors mit der Winkelkante auszurichten. In unserem Fall handelte es sich um den Block eines Lenovo Netzstecker. Die Messdaten werden mithilfe eines Skriptes aufgenommen.

3.3 Messung der Hintergrundstrahlung

Damit die Strahlung der Probe richtig gemessen werden kann, muss eine Nullkorektur durchgeführt werden, bei der die Strahlung des Hintergrundes von der Strahlung der Proben abgezogen wird.
Dazu werden je zwei Messungen von 400 s für 0° und 90° durchgeführt. Das Weglassen der 45° Messung ist möglich, da der Hintergrund isotrop sein sollte, da er keine konkreten Strahlungsquelle besitzt und somit auch keine unterschiedlichen Verteilungen im Raum vorliegen sollten. Dies lässt sich mit den Messungen von 90° und 0° überprüfen.

4 Auswertung, Fehlerrechnung und Diskussion der Messergebnisse

Die Daten werden im folgenden mit zwei verschiedenen Methoden ausgewertet und Verglichen. Zuerst werden die Anzahlen der erfolgten Ereignisse/Energien über gleiche Messreihen aufsummiert und somit eine einzelne Zählrate für gleiche Messungen bestimmt. Als zweite Methode werden alle Messreihen einzeln ausgewertet und mehrere Zählraten bestimmt. Über statistischen Mittelwert und Standardabweichung werde die einzelnen Ergebnisse in einem finalen Ergebnis dargestellt.
Generell wird für den Versuch die Zählrate $R(\Theta)$ der einzelnen Winkel bestimmt um diese dann auf 90° zu normieren und die Korrelationsfunktion für 135° und 180° zu bestimmen. Diese werden benutzt um die Koeffizienten a_2 und a_4 der Korrelationfunktion zu bestimmen. Zusätzlich geben die Werte eine Aussage über die Anisotropie A_n aus Gleichung 1.4. Die statischen Fehler aus den Zählungen werden gemäß der Poisson-Verteilung $\sigma = \sqrt{N}$ bestimmt. Die statistischen und systematischen Fehler werden dann über alle Rechnungen hinweg mit gauß'scher unkorrelierter Fehler-Bestimmung fortgepflanzt.

Benötigte Formeln:

$$R(\Theta) = \frac{N_K - N_{K,U} - N_Z}{(N_1 - N_U) \cdot (N_2 - N_U)} \tag{4.1}$$

$$\Delta R(\Theta) = \sqrt{\left(\frac{\partial R}{\partial N_K} \cdot \Delta N_K\right)^2 + \left(\frac{\partial R}{\partial N_{K,U}} \cdot \Delta N_{K,U}\right)^2 + \left(\frac{\partial R}{\partial N_Z} \cdot \Delta N_Z\right)^2 + \left(\frac{\partial R}{\partial N_i} \cdot \Delta N_i\right)_i^2 + \left(\frac{\partial R}{\partial N_U} \cdot \Delta N_U\right)^2} \tag{4.2}$$

wobei N_K: Anzahl der Koinzidenzen, $N_{K,U}$: Anzahl der Koinzidenzen in der Untergrund-Messung, N_Z: Anzahl der zufälligen Koinzidenzen und N_i: Anzahl der Ereignisse am Detektor i.

$$K(\Theta) = \frac{R(\Theta)}{R(90°)} \tag{4.3}$$

$$\Delta K(\Theta) = \sqrt{\left(\frac{\partial K}{\partial R(\Theta)} \cdot \Delta R(\Theta)\right)^2 + \left(\frac{\partial K}{\partial R(90°)} \cdot \Delta R(90°)\right)^2} \tag{4.4}$$

$$a_2 = 4 \cdot K(135°) - K(180°) - 3 \tag{4.5}$$

$$a_4 = -4 \cdot K(135°) + 2 \cdot K(180°) + 2 \tag{4.6}$$

$$\Delta a_{2,4} = \sqrt{\left(\frac{\partial a_{2,4}}{\partial K(135°)} \cdot \Delta k(135°)\right)^2 + \left(\frac{\partial a_{2,4}}{\partial K(180°)} \cdot \Delta k(180°)\right)^2} \tag{4.7}$$

Messung der zufälligen Koinzidenz Um die zufällige Koinzidenz zu erfassen, werden die Messungen der Proben benutzt, allerdings wird eine Zeitverschiebung eingebaut, damit wahre Koinzidenz ausgeschlossen werden kann (die zeitgleich wahrgenommenen Messungen sind nicht im Intervall gemessen worden, dass als zeitgleich identifiziert wurde). Da es sich bei zufälligen Koinzidenzen um ein statistisches Phänomen handelt, kann so auf die ungefähre Häufigkeit zufälliger Koinzidenzen während der Messungen geschlossen werden.

Untergrund-Messung: Bei der Messung des Untergrundes wurde in dieser Durchführung des Experiments unabhängig vom Winkel eine einzelne Koinzidenz pro Reihe gemessen. Hierbei handelt es sich um eine zufällige Koinzidenz. Diese wird wie in Gleichung 4.1 beschrieben von den gemessenen Koinzidenzen der relevanten Messreihen abgezogen.

4.1 Auswertung über die Summe äquivalenter Messreihen

In dieser Auswertung wird die Anzahl der Ereignisse des selben Winkels addiert. Die Untergrundmessung wurde normiert, indem die Anzahl der Ereignisse gemittelt wird. Die gemittelte Messreihe wird dann mit der Anzahl der aufsummierten Messreihen multipliziert. In dieser Durchführung des Versuchs wurde 100 ns als Koinzidenzzeit gewählt. Für zufällige Koinzidenzen werden die wahren Koinzidenzen grob lokalisiert und dann außerhalb dieses Intervalls nach zusätzlichen zufälligen Koinzidenzen abgetastet.

Tabelle 4.1: Summierte Ereignisse und Koinzidenzen

	180°	Δ_{180}	135°	Δ_{135}	90°	Δ_{90}
N_K	277	17	310	18	257	16
N_1	175758	419	200840	448	201170	449
N_2	184422	429	174677	418	178729	423
N_Z	2	1	1	1	4	2
N_U	4	2	4	2	4	2

Über Gleichung 4.1 bis Gleichung 4.6 werden die gesuchten Zählraten, Korrelationskoeffizienten und die Anisotropie bestimmt mit den dazugehörigen Fehlern.

Tabelle 4.2: Koinzidenzraten

Θ	R(Θ) in $10^{-9}\frac{1}{s}$	Δ_R in $10^{-9}\frac{1}{s}$
180°	8.73	0.55
135°	9.07	0.54
90°	7.22	0.47

Tabelle 4.3: Korrelationskoeffizienten und Anisotropie

	Wert	Δ
$K(135°)$	1.26	0.09
$K(180°)$	1.21	0.09
$a2$	0.81	0.38
$a4$	-0.60	0.41
An	0.21	0.09

Diese Ergebnisse befinden sich in der Größenordnung der Literaturwerte ($a_2 = \frac{1}{8}$,$a_4 = \frac{1}{24}$). Die Abweichungen sind jedoch ausreichend stark, dass sich die Literaturwerte nicht in dem Fehlerbereich der statistischen Unsicherheit befindet. Grund dafür müssen systematische Unsicherheiten im Aufbau sein. Grund dafür könnte Unsicherheiten beim Aufbau der Messapperatur sein, wie z.B. Unsicherheit in der Ausrichtung der Detektoren, unterschiedliche Abstände der Detektoren zur radioaktiven Quelle etc.

4.2 Auswertung der einzelnen Messreihen

Bei dieser Methode werden die einzelnen Messreihen analog zur Summe ausgewertet. Zusätzlich wird zuletzt über die verschiedenen Ergebnisse gemittelt und der Fehler des Mittelwerts bestimmt. Dabei wird der Fehler der Einzelwerte fortgepflanzt und ähnlich zu Standardabweichung wird der Fehler auf den Mittelwert bestimmt. In dieser Durchführung wurde jede Messreihe vier mal aufgenommen. Dabei jeweils zwei direkt hintereinander (Nummerierung: 1.1,1.2;2.1,2.2). Dabei ergibt sich:

Tabelle 4.4: Korrelationkoeffizienten der einzelnen Messreihen

	Wert 1.1	$\Delta_{1.1}$	Wert 1.2	$\Delta_{1.2}$	Wert 2.1	$\Delta_{2.1}$	Wert 2.2	$\Delta_{2.2}$
$K(135°)$	1.03	0.18	1.18	0.16	1.07	0.18	1.98	0.23
$K(180°)$	1.23	0.20	0.95	0.18	1.17	0.17	1.67	0.22
a_2	-0.11	0.75	0.78	0.68	0.10	0.73	3.25	0.95
a_4	0.34	0.83	-0.83	0.75	0.07	0.79	-2.58	1.02
An	0.23	0.20	-0.05	0.18	0.17	0.17	0.67	0.22

Tabelle 4.5: Mittelwerte der Korrelationskoeffizienten und der Anisotropie

	Wert	Δ
a_2	1.01	0.78
a_4	-0.75	0.85
An	0.26	0.19

Im Vergleich zur Auswertung über die Summe gibt es auch hier eine starke Abweichung von den Theoriewerten. Jedoch sind die Fehlerintervalle größer und das Fehlerintervall für a_4 beinhaltet nun den theoretischen Wert. Grund dafür ist, dass durch Summieren der einzelnen Messreihen, der Poisson-Fehler überhand nimmt und kleine Fehler durch die Wurzel großer Zahlen produziert. Während sich die Fortgepflanzten Fehler der einzelnen Messreihen in bei Bildung des Mittelwertes statistisch größere Fehler abbildet. Der theoretische Wert für a_2 liegt hingegen knapp außerhalb des Fehlerintervalls. Die Anisotropie ist ähnlich zur vorherigen Auswertung. Daraus lässt sich schlussfolgern, dass längere Messreihen nicht bessere Ergebnisse liefern als die durchgeführten Messreihen. Um die Winkelkorrelation signifikanter zu bestimmen müsste der Detektoraufbau präziser konfiguriert werden. Vorallem der Abstand zwischen Detektor und radioaktiver Quelle wurde bei dieser Durchführung nicht angepasst.

4.3 Auflösungszeit der Koinzidenz

Koinzidenzen manifestieren sich als gleichzeitige oder zeitlich korrelierte Ereignisse, deren präzise zeitliche Auflösung von entscheidender Bedeutung ist. Um die Auflösungszeit zu bestimmen wird, analog zur ersten Auswertungsmethode, die Anzahl der Koinzidenzen, Ereignisse und zufälligen Koinzidenzen aufsummiert. Nach dem blauen Buch [Wol23] folgt für die Auflösungszeit:

$$\tau_A = \frac{N_Z}{N_1 \cdot N_2} \tag{4.8}$$

Somit ergibt sich:

$$\tau_a \approx 4.66ns$$

Bei der Auswertung wird ein Koinzidenzintervall von 100 ns eingestellt. Der Wert von rund 4.66 ns hat sich bereits in der Datenauswertung beim Testen der Koinzidenzzeit wiedergefunden.

Bei einer Koinzidenzzeit von 8 ns wurde keine Veränderung der Koinzidenzanzahl gefunden. D.h. dass Koinzidenzen unter 8 ns in den Messungen nicht mehr von einander unterscheidbar waren, was für eine Auflösungsgrenze des Detektors spricht. Dementsprechend erfüllt die ermittelte Auflösungszeit den während der Auswertung aufgestellten Hypothesen.

Quellen

[Lei] https://www.leifiphysik.de/kern-teilchenphysik/radioaktivitaet-fortfuehrung/aufgabe/beta-plus-zerfall-von-natrium-22, letzter Zugriff 12.11.2021

[Wika] https://de.wikipedia.org/wiki/Betastrahlung#Beta-Minus-Zerfall_(%CE%B2%E2%88%92, letzter Zugriff 20.11.2023

[Wikb] https://en.wikipedia.org/wiki/Cobalt-60, letzter Zugriff 12.11.2021

[Wol23] Wolf, Joachim: *Einführung in das Kern- und Teilchenphysikalische Praktikum.* 17. November 2023. Karlsruhe, 2023. – 271 S.

5 Quellcode

In [ ]:
```
import os
import struct
import numpy as np
import matplotlib.pyplot as plt
import scipy.optimize as opt
import pandas as pd
```

def read_data(infile): with open(infile, 'rb') as f: nevent = 0 notEOF = True max_nevent = 1000000000 xList, eventList, tList, tDAQ, = [], [], [], [] while notEOF and nevent < max_nevent: # read header: record, boardID, channel, pattern, event, time_ns s = f.read(24) if len(s) != 24: notEOF = False break record, boardID, channel, pattern, event, time_ns = struct.unpack("

Read first waveform file (channel 0)

In [ ]:
```
f0 = "wave0_Kalibrierung.csv"
xList0, eventList0, tList0,  tDAQ0 = np.loadtxt(f0, unpack=True, skiprows=1, delimiter=',')
```

Read second waveform file (channel 1)

In [ ]:
```
f1 = "wave1_Kalibrierung.csv"
xList1, eventList1, tList1,  tDAQ1 = np.loadtxt(f1, unpack=True, skiprows=1, delimiter=',')
```

Plot histogram

In [ ]:
```
x0, bins0, patch = plt.hist(xList0, bins=1000, range=(0,15000))
bins0 = bins0[:-1]
plt.xlabel('channel number')
plt.ylabel('events PMT 0')
plt.axis([0, 15000, 0, 400])
plt.grid(True)
plt.show()

x1, bins1, patch = plt.hist(xList1, bins=1000, range=(0,15000))
bins1 = bins1[:-1]
plt.xlabel('channel number')
plt.ylabel('events PMT 1')
plt.axis([0, 13000, 0, 400])
plt.grid(True)
plt.show()
```

Write the histogrammed data to .csv files

In [ ]:
```
np.savetxt("kalibrierung_0.csv", np.transpose(np.array((bins0, x0))), delimiter=",", header="bins, counts")
```

In [ ]:
```
np.savetxt("kalibrierung_1.csv", np.transpose(np.array((bins1, x1))), delimiter=",", header="bins, counts")
```

TODO: calibration

- energy = m*channel + b
- fit the two photo-peaks in the Na-22 spectrum with a Gaussian/quadratic shape (be careful, which are the right peaks)
- from that, deduce the index channel number corresponding to 60% of the second photo-peak of the Co-60 spectrum
- do this separately for both detectors

In [ ]:
```
########################################################################
# TO DO: define a gaussian function with the normalization factor A #
########################################################################
def gauss(x,mu ,sigma,A):
    return A*1/(sigma*np.sqrt(2*np.pi))*np.exp(-0.5*((x-mu)/sigma)**2)
```

This is a simple code block to fit a Gaussian to a part of the spectrum.

It should be carefully read to understand what the code does and results of the fit should not be used without reading the documentation of the fitting function and all of its return values.

Detector 1

In [ ]:
```
n_start_1, n_stop_1 = int(3000/15),int(5000/15)
n_start_2, n_stop_2 = int(8500/15),int(10500/15)

print(len(bins0))
print(len(x0))
sample_x_1 = bins0[n_start_1:n_stop_1]
```

```
sample_y_1 = x0[n_start_1:n_stop_1]
sample_x_2 = bins0[n_start_2:n_stop_2]
sample_y_2 = x0[n_start_2:n_stop_2]


plt.figure(figsize=(12,7))
plt.plot(bins0[10:,],x0[10:,])
plt.plot(sample_x_1,sample_y_1)
plt.plot(sample_x_2,sample_y_2)

param_1 = [4000,500,200]
param_2 = [9500,1000,25]

fit_result_1 = opt.curve_fit(gauss, sample_x_1, sample_y_1, param_1)
fit_result_2 = opt.curve_fit(gauss, sample_x_2, sample_y_2, param_2)

print("### First Peak")
print(f"mu = {str(round(fit_result_1[0][0],1))} +- {str(round(np.sqrt(fit_result_1[1][0][0]),1))}")
print(f"sigma = {str(round(fit_result_1[0][1],1))}  +- {str(round(np.sqrt(fit_result_1[1][1][1]),1))}")
print(f"A = {str(round(fit_result_1[0][2],1))} +- {str(round(np.sqrt(fit_result_1[1][2][2]),1))}")
print("")
print("### Second Peak")
print(f"mu = {str(round(fit_result_2[0][0],1))} +- {str(round(np.sqrt(fit_result_2[1][0][0]),1))}")
print(f"sigma = {str(round(fit_result_2[0][1],1))} +- {str(round(np.sqrt(fit_result_2[1][1][1]),1))}")
print(f"A = {str(round(fit_result_2[0][2],1))} +- {str(round(np.sqrt(fit_result_2[1][2][2]),1))}")

fit_y_1 = [gauss(x,fit_result_1[0][0],fit_result_1[0][1],fit_result_1[0][2]) for x in sample_x_1]
fit_y_2 = [gauss(x,fit_result_2[0][0],fit_result_2[0][1],fit_result_2[0][2]) for x in sample_x_2]

plt.plot(sample_x_1, fit_y_1)
plt.plot(sample_x_2, fit_y_2)
plt.axis([0, 12000, 0, 300])
plt.show()
```

Detector 2

In [ ]:

```
n_start_3, n_stop_3 = int(2500/15),int(4500/15)
n_start_4, n_stop_4 = int(8000/15),int(9500/15)

sample_x_3 = bins1[n_start_3:n_stop_3]
sample_y_3 = x1[n_start_3:n_stop_3]
sample_x_4 = bins1[n_start_4:n_stop_4]
sample_y_4 = x1[n_start_4:n_stop_4]

plt.figure(figsize=(12,7))
plt.plot(bins1[10:,],x1[10:,])
plt.plot(sample_x_3,sample_y_3)
plt.plot(sample_x_4,sample_y_4)

param_3 = [4000,500,200]
param_4 = [9500,1000,25]

fit_result_3 = opt.curve_fit(gauss, sample_x_3, sample_y_3, param_3)
fit_result_4 = opt.curve_fit(gauss, sample_x_4, sample_y_4, param_4)

print("### First Peak")
print(f"mu = {str(round(fit_result_3[0][0],1))} +- {str(round(np.sqrt(fit_result_3[1][0][0]),1))}")
print(f"sigma = {str(round(fit_result_3[0][1],1))}  +- {str(round(np.sqrt(fit_result_3[1][1][1]),1))}")
print(f"A = {str(round(fit_result_3[0][2],1))} +- {str(round(np.sqrt(fit_result_3[1][2][2]),1))}")
print("")
print("### Second Peak")
print(f"mu = {str(round(fit_result_4[0][0],1))} +- {str(round(np.sqrt(fit_result_4[1][0][0]),1))}")
print(f"sigma = {str(round(fit_result_4[0][1],1))} +- {str(round(np.sqrt(fit_result_4[1][1][1]),1))}")
print(f"A = {str(round(fit_result_4[0][2],1))} +- {str(round(np.sqrt(fit_result_4[1][2][2]),1))}")

fit_y_3 = [gauss(x,fit_result_3[0][0],fit_result_3[0][1],fit_result_3[0][2]) for x in sample_x_3]
fit_y_4 = [gauss(x,fit_result_4[0][0],fit_result_4[0][1],fit_result_4[0][2]) for x in sample_x_4]

plt.plot(sample_x_3, fit_y_3)
plt.plot(sample_x_4, fit_y_4)
plt.axis([0, 12000, 0, 300])
#plt.show()
```

Do the energy calibration to determine the 60% (maybe higher is better?) threshold

In [ ]:

```
channel_peak_1 = fit_result_1[0][0]
channel_peak_2 = fit_result_2[0][0]
```

```
E_1, E_2 = 0.511, 1.2745 #Energies in MeV

E_Co = 1.3325

m1 = (E_2 - E_1)/(channel_peak_2 - channel_peak_1)
b1 = (channel_peak_2 * E_1 - channel_peak_1 * E_2)/(channel_peak_2 - channel_peak_1)

print(f"m1 = {round(m1,7)} MeV/channel")
print(f"b1 = {round(b1,4)} MeV")

ch_threshold1 = (0.6 * E_Co - b1)/m1

print(f"60% of the second peak's energy equals to channel number {ch_threshold1}")
print()

channel_peak_3 = fit_result_3[0][0]
channel_peak_4 = fit_result_4[0][0]

#E_1, E_2 = 1.173, 1.332 #Energies in MeV

#E_Co = 1.332

m2 = (E_2 - E_1)/(channel_peak_4 - channel_peak_3)
b2 = (channel_peak_4 * E_1 - channel_peak_3 * E_2)/(channel_peak_4 - channel_peak_3)

print(f"m2 = {round(m2,7)} MeV/channel")
print(f"b2 = {round(b2,4)} MeV")

ch_threshold2 = (0.6 * E_Co - b2)/m2

print(f"60% of the second peak's energy equals to channel number {ch_threshold2}")
```

In [ ]:
```
def lin(x,m,b):
    return m*x+b
```

Detector 2

In [ ]:
```
plt.figure(figsize=(12,7))
plt.plot(lin(bins1[10:,],m2,b2),x1[10:,])
plt.plot(lin(sample_x_3,m2,b2),sample_y_3)
plt.plot(lin(sample_x_4,m2,b2),sample_y_4)


plt.plot(lin(sample_x_3,m2,b2), fit_y_3)
plt.plot(lin(sample_x_4,m2,b2), fit_y_4)
plt.xlabel('E in MeV')
#plt.axis([0, 12000, 0, 300])
plt.show()
```

Detector 1

In [ ]:
```
plt.figure(figsize=(12,7))
plt.plot(lin(bins0[10:,],m1,b1),x0[10:,])
plt.plot(lin(sample_x_1,m1,b1),sample_y_1)
plt.plot(lin(sample_x_2,m1,b1),sample_y_2)

plt.plot(lin(sample_x_1,m1,b1), fit_y_1)
plt.plot(lin(sample_x_2,m1,b1), fit_y_2)
plt.xlabel('E in MeV')
#plt.axis([0, 12000, 0, 300])
plt.show()
```

Search for coincidences in the Co-60 data

In [ ]:
```
f0 = "wave0_co_0g_1_2.csv"
xList0, eventList0, tList0,  tDAQ0 = np.loadtxt(f0, unpack=True, skiprows=1, delimiter=',')
f1 = "wave1_co_0g_1_2.csv"
xList1, eventList1, tList1,  tDAQ1 = np.loadtxt(f1, unpack=True, skiprows=1, delimiter=',')
```

In [ ]:
```
x0, bins0, patch = plt.hist(xList0, bins=1000, range=(0,15000))
bins0 = bins0[:-1]
plt.xlabel('channel number')
plt.ylabel('events PMT 0')
plt.axis([0, 15000, 0, 400])
```

```
plt.grid(True)
plt.show()

x1, bins1, patch = plt.hist(xList1, bins=1000, range=(0,15000))
bins1 = bins1[:-1]
plt.xlabel('channel number')
plt.ylabel('events PMT 1')
plt.axis([0, 13000, 0, 400])
plt.grid(True)
plt.show()
```

In [ ]:
```
plt.figure(figsize=(12,7))
plt.plot(lin(bins0[10:,],m1,b1),x0[10:,])
#plt.plot(lin(sample_x_1,m1,b1),sample_y_1)
#plt.plot(lin(sample_x_2,m1,b1),sample_y_2)

#plt.plot(lin(sample_x_1,m1,b1), fit_y_1)
#plt.plot(lin(sample_x_2,m1,b1), fit_y_2)
plt.xlabel('E in MeV')
#plt.axis([0, 12000, 0, 300])
plt.show()
```

In [ ]:
```
x=np.linspace(-2000,15000)
plt.plot(x,lin(x,m1,b1))
plt.xlabel('channel')
plt.ylabel('Energie in MeV')
plt.hlines(0.6*E_Co,-2000,14000,label='0.6 E_Co',color='r')
plt.hlines(E_Co,-2000,14000,label='E_Co',color='g')
plt.legend()
```

In [ ]:
```
print(len(xList0), len(tDAQ0), len(tDAQ1))

coinc0, eCoinc0, tCoinc0 = [], [], []
coinc1, eCoinc1, tCoinc1 = [], [], []
dtCoinc, eSum = [], []

if len(xList0) != len(xList1):
    print("Warning: different event numbers")

for i in range(len(xList0)):
    if tDAQ0[i] != tDAQ1[i]:
        print("different times in event ",i)

    # uncomment the following line to calculate random coincidences
    #if abs(tList0[i] - tList1[i]) > '<TO DO: start time for arbitrary coincidence>' and abs(tList0[i] - tList1[i]) < '<T
    #if abs(tList0[i] - tList1[i]) > 200 and abs(tList0[i] - tList1[i]) < 300 and xList0[i]>int(ch_threshold1) and xList1
    #if abs(tList0[i] - tList1[i]) < '<TO DO: write time in seconds for true coincidence>'  and xList0[i]>int(ch_threshol
    if abs(tList0[i] - tList1[i]) < 100 and xList0[i]>int(ch_threshold1) and xList1[i]>int(ch_threshold2):

        coinc0.append(eventList0[i])
        eCoinc0.append(xList0[i])
        tCoinc0.append(tList0[i])

        coinc1.append(eventList1[i])
        eCoinc1.append(xList1[i])
        tCoinc1.append(tList1[i])

        dtCoinc.append(tList0[i]-tList1[i])
        eSum.append(xList0[i]+xList1[i])

nCoinc = len(eCoinc0)

n0 = len([x for x in xList0 if x>int(ch_threshold1)]) # total number of events above the threshold
n1 = len([x for x in xList1 if x>int(ch_threshold2)])

print(f"Number of events PMT 0:      {n0}")
print(f"Number of events PMT 1:      {n1}")
print(f"Number of coincident events: {nCoinc}")
```

Resolving coincidence time

In [ ]:
```
```

In [ ]:
```
plt.hist(dtCoinc, bins=20, range=(-200,200))
plt.axis([-200, 200, 0, 200])
```

```
plt.grid(True)
plt.show()
```

Daten generieren

Funktionen definieren

In [ ]:

```
def koinzidenzen(file0,file1,dt,dtstart):
    xList0, eventList0, tList0,  tDAQ0 = np.loadtxt(file0, unpack=True, skiprows=1, delimiter=',')
    xList1, eventList1, tList1,  tDAQ1 = np.loadtxt(file1, unpack=True, skiprows=1, delimiter=',')

    coinc0, eCoinc0, tCoinc0 = [], [], []
    coinc1, eCoinc1, tCoinc1 = [], [], []
    dtCoinc, eSum = [], []


    if len(xList0) != len(xList1):
        print("Warning: different event numbers")

    for i in range(len(xList0)):
        if tDAQ0[i] != tDAQ1[i]:
            print("different times in event ",i)

        #koinzidenzen
        if abs(tList0[i] - tList1[i]) < dt and xList0[i]>int(ch_threshold1) and xList1[i]>int(ch_threshold2):
            coinc0.append(eventList0[i])
            eCoinc0.append(xList0[i])
            tCoinc0.append(tList0[i])

            coinc1.append(eventList1[i])
            eCoinc1.append(xList1[i])
            tCoinc1.append(tList1[i])

            dtCoinc.append(tList0[i]-tList1[i])
            eSum.append(xList0[i]+xList1[i])

    nCoinc = len(eCoinc0)

    n0 = len([x for x in xList0 if x>int(ch_threshold1)]) # total number of events above the threshold
    n1 = len([x for x in xList1 if x>int(ch_threshold2)])

    res=[]
    res.append([nCoinc,n0,n1])

    coinc0, eCoinc0, tCoinc0 = [], [], []
    coinc1, eCoinc1, tCoinc1 = [], [], []
    dtCoinc, eSum = [], []

    if len(xList0) != len(xList1):
        print("Warning: different event numbers")

    for i in range(len(xList0)):
        if tDAQ0[i] != tDAQ1[i]:
            print("different times in event ",i)

        # uncomment the following line to calculate random coincidences
        #if abs(tList0[i] - tList1[i]) > '<TO DO: start time for arbitrary coincidence>' and abs(tList0[i] - tList1[i]) <
        if abs(tList0[i] - tList1[i]) > dtstart and abs(tList0[i] - tList1[i]) < dtstart+dt and xList0[i]>int(ch_threshol
            coinc0.append(eventList0[i])
            eCoinc0.append(xList0[i])
            tCoinc0.append(tList0[i])

            coinc1.append(eventList1[i])
            eCoinc1.append(xList1[i])
            tCoinc1.append(tList1[i])

            dtCoinc.append(tList0[i]-tList1[i])
            eSum.append(xList0[i]+xList1[i])

    nCoinc = len(eCoinc0)

    n0 = len([x for x in xList0 if x>int(ch_threshold1)]) # total number of events above the threshold
    n1 = len([x for x in xList1 if x>int(ch_threshold2)])

    res.append([nCoinc,n0,n1])
```

```
    return res
```

In [ ]:
```
def datensatz(names,dt,t):
    zeroCo,zeroRand=[],[]
    tmp0,tmp1=[],[]
    for i in names:
        tmp0,tmp1=koinzidenzen("wave0_"+i,"wave1_"+i,dt,t)
        zeroCo.append(tmp0)
        zeroRand.append(tmp1)
    return zeroCo,zeroRand
```

In [ ]:
```

```

In [ ]:
```
names_0d=['co_0g_1_1.csv','co_0g_1_2.csv','co_0g_2_1.csv','co_0g_2_2.csv']
names_45d=['co_45g_1_1.csv','co_45g_1_2.csv','co_45g_2_1.csv','co_45g_2_2.csv']
names_90d=['co_90g_1_1.csv','co_90g_1_2.csv','co_90g_2_1.csv','co_90g_2_2.csv']
names_Hintergrund=['Hintergrund_0g_1.csv','Hintergrund_0g_2.csv','Hintergrund_90g_1.csv','Hintergrund_90g_2.csv']
```

In [ ]:
```
dt,t=100,1000
co_background,rand_background=datensatz(names_Hintergrund,dt,t)
co0d,rand0d=datensatz(names_0d,dt,t)
co45d,rand45d=datensatz(names_45d,dt,t)
co90d,rand90d=datensatz(names_90d,dt,t)
```

In [ ]:
```
#BACKGROUND
Nkb=np.array(co_background)[:,0]
print(Nkb)
Nb_val=np.array(co_background)[:2,1:]
Nb=np.sum(Nb_val)/2
sigma_Nb=np.sqrt(Nb)
print(Nb)

#Normiereung der Hintergrundmessungen für 0 Grad

print('kaum unterschied zwischen 90° und 180° -> 1 Koinzidenz pro Mesreihe aus Untergrund')
Nkb=1
#Mitteln der Anzahl der Zähler
```

In [ ]:
```
print(co0d)
print(np.sum(co0d,axis=0))
```

Auswertung über die Summe

In [ ]:
```
def datasum(x):
    return np.sum(x,axis=0)
```

In [ ]:
```
def rate(x,Nb):
    return (x[0]-x[3]-x[4])/((x[1]-4*Nb)*(x[2]-4*Nb))
```

In [ ]:
```
def sigma_rate(x,sx,Nb):
    return np.sqrt(
        ((1/((x[1]-4*Nb)*(x[2]-4*Nb)))*sx[0])**2+
        (-1/(((x[1]-4*Nb)*(x[2]-4*Nb)))*sx[3])**2+
        (-1/(((x[1]-4*Nb)*(x[2]-4*Nb)))*sx[4])**2+
        (((-x[0]+x[3]+x[4])/((x[1]-4*Nb)**2 *(4*Nb-x[2])))*sx[1])**2+
        ((x[0]-x[3]-x[4])/((x[1]-4*Nb)**2 *(x[2]-4*Nb)**2)*(x[1]-2*Nb+x[2])*sigma_Nb)**2+
        (((-x[0]+x[3]+x[4])/((x[1]-4*Nb)*(x[2]-4*Nb)**2))*sx[2])**2
        )
```

In [ ]:
```
def korrandval(r1,r2,r3,sr1,sr2,sr3):
    K135=r2/r3
    K180=r1/r3
    sK135=np.sqrt((sr2/r3)**2+(sr3/r2)**2)
    sK180=np.sqrt((sr1/r3)**2+(sr3/r1)**2)

    a2=4*K135-K180-3
    sa2=np.sqrt(16*sK135**2+sK180**2)
    a4=-4*K135+2*K180+2
    sa4=np.sqrt(16*sK135**2+4*sK180**2)
    An=K180-1
    sAn=sK180
    return [K135,K180,a2,a4,An],[sK135,sK180,sa2,sa4,sAn]
```

In [ ]:
```
N_0=datasum(co0d)
N_Z_0=datasum(rand0d)[0]
N_U=4

N_45=datasum(co45d)
N_Z_45=datasum(rand45d)[0]

N_90=datasum(co90d)
N_Z_90=datasum(rand90d)[0]

a=np.append(N_0,[N_Z_0,N_U])
b=np.append(N_45,[N_Z_45,N_U])
c=np.append(N_90,[N_Z_90,N_U])
sa=np.sqrt(a).round()
sb=np.sqrt(b).round()
sc=np.sqrt(c).round()
```

In [ ]:
```
df={'180°':a,r'$\Delta_{180}$':sa,'135°':b,r'$\Delta_{135}$':sb,'90°':c,r'$\Delta_{90}$':sc}
df=pd.DataFrame(df)
df.index = [r'$N_K$',r'$N_1$',r'$N_2$',r'$N_Z$',r'$N_U$']
display(df)
df.to_latex()
```

In [ ]:
```
ra=rate(a,Nb)
sra=sigma_rate(a,sa,Nb)
rb=rate(b,Nb)
srb=sigma_rate(b,sb,Nb)
rc=rate(c,Nb)
src=sigma_rate(c,sc,Nb)

dfr={r'$\Theta$':['180°','135°','90°'],r'R($\Theta$) in $10^{-9}$':np.array([ra,rb,rc])*10**9,r'$\Delta_R$ in $10^{-9}$':
dfr=pd.DataFrame(dfr).round(2)
dfr.to_latex()
display(dfr)
dfr.to_latex()
#print(ra,rb,rc)
#<print(sra,srb,src)
```

In [ ]:
```
dat,sdat=korrandval(ra,rb,rc,sra,srb,src)
dfres=pd.DataFrame({'Wert':dat,r'$\Delta$':sdat},index=[r'K(135°)',r'K(180°)',r'a2',r'a4',r'An']).round(2)
display(dfres)
dfres.to_latex()
```

Einzelne Auswertung

In [ ]:
```
val=[[],[],[],[],[],[],[],[],[],[],[],[],[]]
data_max=[[co0d,rand0d],[co45d,rand45d],[co90d,rand90d]]
#print(data_max[0][0])
#print(co0d)

for a in range(3):
    for i in range(4):
        #print(data_max[a][1][i][0])
        val[4*a+i]=np.append(data_max[a][0][i],[data_max[a][1][i][0],1])

val=np.array(val)
print(val)
sval=np.sqrt(val)

rates=[]
srates=[]
for i in range(len(val)):
    rates.append(rate(val[i],Nb/4))
    srates.append(sigma_rate(val[i],sval[i],Nb/4))
#print(rates,srates)

dat,sdat=korrandval(ra,rb,rc,sra,srb,src)
```

In [ ]:
```
endval,sendval=[],[]
for i in range(4):
    tmp,stmp=korrandval(rates[i],rates[i+4],rates[i+8],srates[i],srates[i+4],srates[i+8])
    endval.append(tmp)
    sendval.append(stmp)
endval=np.array(endval)
```

```
sendval=np.array(sendval)
print(endval)
```

In [ ]:
```
endval[:,0]
```

In [ ]:
```
dfres=pd.DataFrame({'Wert 1.1':endval[0],r'$\Delta_{1.1}$':sendval[0],'Wert 1.2':endval[1],r'$\Delta_{1.2}$':sendval[1],'
display(dfres)
dfres.to_latex()
```

In [ ]:
```
print('a2',np.mean(endval[:,2]).round(2),np.sqrt(1/4*np.sum(sendval[:,2]**2)).round(2))
print('a4',np.mean(endval[:,3]).round(2),np.sqrt(1/4*np.sum(sendval[:,3]**2)).round(2))
print('An',np.mean(endval[:,4]).round(2),np.sqrt(1/4*np.sum(sendval[:,4]**2)).round(2))
```

Auflösungszeit

In [ ]:
```
x=np.sum(np.array(val),axis=0)
#print(x)
result=x[3]/(x[1]*x[2])
print(result)
sresult=np.sqrt( (x[3]*10**9/(x[1]**2 * x[2])*np.sqrt(x[1]))**2 +(x[3]/(x[2]**2 * x[1])*np.sqrt(x[2]))**2+ (1/(x[1] * x[2
print(sresult)
```