Fortgeschrittenen Praktikum II

Der Compton-Effekt

Wiebke Herzberg Kolja Glogowski

9. Mai 2005

Inhaltsverzeichnis

1. Aufgabenstellung

1. Aufnahme des direkten γ-Spektrums von ^{137}Cs mit dem NaI-Szintillator unter einem Winkel von 0°. Identifikation der Photolinie, Comptonkante und des Rückstreupeaks, sowie Eichung des MCA mit Hilfe einer weiteren Linie (^{22}Na-Quelle).

2. Aufnahme des direkten γ-Spektrums von ^{137}Cs mit dem Plastikszintillator, und Eichung des MCA mit den Compton-Kanten von ^{137}Cs und ^{22}Na.

3. Zeitlicher Abgleich von NaI- und Plastikszintillator.

4. Aufnahme des γ- und Elektronenspektrums bei Koinzidenz von Elektronen und γ-Quanten für verschiedene Streuwinkel. Dabei ist die Energieerhaltung beim Compton-Effekt zu verifizieren.

5. Messung des differentiellen Wirkungsquerschnitts für Compton-Streuung und Vergleich mit der Klein-Nishina-Formel.

2. Theoretische Grundlagen

2.1. Zerfallsarten

α-Zerfall

Beim α-Zerfall geht ein Kern aus einem angeregten Zustand unter Aussendung eines Helium-Kerns ($^4He^{2+}$) in einen stabileren Zustand über. Alle α-Teilchen eines bestimmten Übergangs weisen dieselbe Energie auf, da die beteiligten Niveaus diskret sind und nur ein Teilchen emittiert wird.

β^--Zerfall

Beim β^--Zerfall wird im Kern ein Neutron in ein Proton umgewandelt, wobei ein Elektron und ein Antineutrino emittiert werden.

β^+-Zerfall

Der β^+-Zerfall kommt in der Natur nur selten vor. Dabei wird ein Proton in ein Neutron umgewandelt, wobei ein Positron und ein Neutrino emittiert werden. Das Positron kann in Gegenwart von Materie jedoch nicht existieren: Es vereint sich mit einem Elektron zu einem Positronium-Atom, welches dann innerhalb kürzester Zeit unter Aussendung zweier Photonen einer Energie von je 511 keV zerfällt.

Elektroneneinfang (EC)

Der Elektroneneinfang ist eine Art der Radioaktivität, bei der der Atomkern ein Elektron aus der Hülle (meistens aus der K-Schale) einfängt. Dadurch wird ein Proton unter Aussendung eines Neutrinos in ein Neutron umgewandelt. Die in der Elektronenschale entstandene Lücke wird durch ein Elektron einer anderen Schale aufgefüllt und die überschüssige Energie des nachrückenden Elektrons in Form von Röntgenstrahlung oder Auger-Strahlung abgegeben. Der Vorgang ist ein Konkurrenzprozess zum β^+-Zerfall; er tritt auf, wenn nicht genügend Energie zur Abgabe eines Positrons verfügbar ist.

γ-Strahlung

Zusätzlich zur α- und β-Strahlung tritt beim radioaktiven Zerfall als Begleiterscheinung auch noch γ-Strahlung auf. Dabei handelt es sich um hochenergetische, elektromagnetische Strahlung mit sehr kurzen Wellenlängen von unter $10^{-9}m$.

Zur γ-Emission kommt es, wenn der Kern durch Elektroneneinfang, α- oder β-Zerfall in einen angeregten Zustand des Tochterkerns übergeht. Dieser zerfällt dann durch Emission von γ-Strahlung in energieärmere Zustände bis hin zum Grundzustand.

2.2. γ-Quellen im Versuch

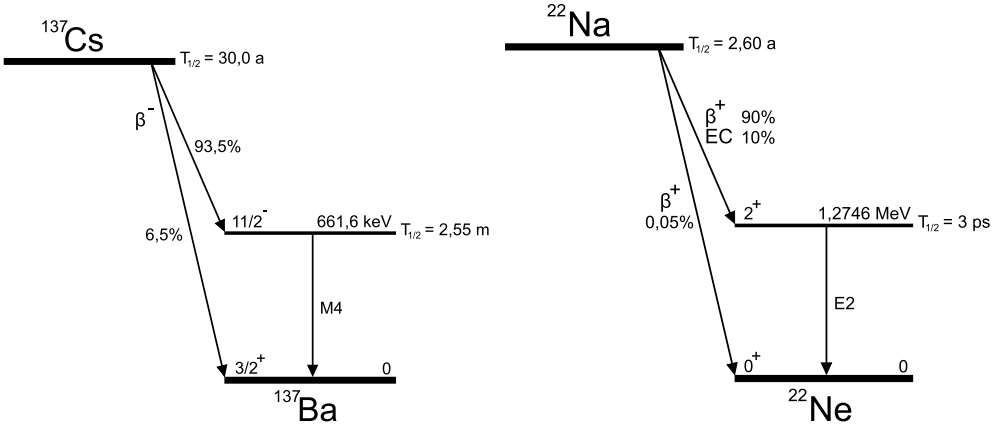

Abbildung 1: Zerfallsschemas von ^{137}Cs und ^{22}Na

Die im Versuch verwendete γ-Quelle besteht aus ^{137}Cs (siehe Abbildung 1). Dieses zerfällt durch β^--Zerfall zu 6,5% direkt in den Barium-Grundzustand und zu 93,5% in den angeregten 661,6 keV-Zustand von Barium. Aus dem angeregten Zustand geht der Kern durch Emission von γ-Strahlung mit einer Energie die seiner Anregungsenergie entspricht, also 661,6 keV, in den Grundzustand über. Außerdem wird im Versuch noch eine ^{22}Na-Quelle (siehe ebenfalls Abbildung 1) zwecks einer Eichung verwendet. ^{22}Na zerfällt zu fast 100% in den 1274,6 keV-Zustand von Neon. Dieser Zerfall geht allerdings zu 90% durch β^+-Zerfall (Aussendung zweier 511 keV-γ-Quanten durch Anihilation, siehe Abschnitt 2.1) und zu 10% durch Elektroneneinfang (EC) vonstatten. Nur ein sehr kleiner Bruchteil der ^{22}Na-Atome, nämlich 0,05%, geht direkt durch β^+-Zerfall in den Neon-Grundzustand über. Aus dem angeregten Neon-Zustand geht der Kern dann unter Emission von 1274,6 keV-γ-Strahlung auch in den Grundzustand über.

2.3. Wechselwirkung von γ-Strahlung mit Materie

In einem Szintillator der dem Nachweis von γ-Strahlung dient, können je nach Energie der Strahlung die drei folgenden Effekte auftreten.

Photoeffekt

Beim Photoeffekt überträgt ein in das Material eindringende γ-Quant seine gesamte Energie auf ein Elektron. Am wahrscheinlichsten ist dieser Effekt bei Elektronen der inneren Schalen. Das Elektron wird aus dem Atomverband herausgelöst und erhält eine kinetische Energie von:

$$E = h\nu - E_B \tag{1}$$

Dabei ist $h\nu$ die Energie des einfallenden γ-Quants und E_B die Energie die nötig ist um das Elektron herauszulösen. Die Lücke des herausgeschlagenen Elektrons wird durch ein nachrückendes Elektron wieder aufgefüllt, dadurch kommt es dann zu Röntgenstrahlung oder Auger-Effekt. Der Photoeffekt tritt am häufigsten auf bei γ-Strahlung niedriger Energie und Absorbermaterial das aus schweren Elementen besteht.

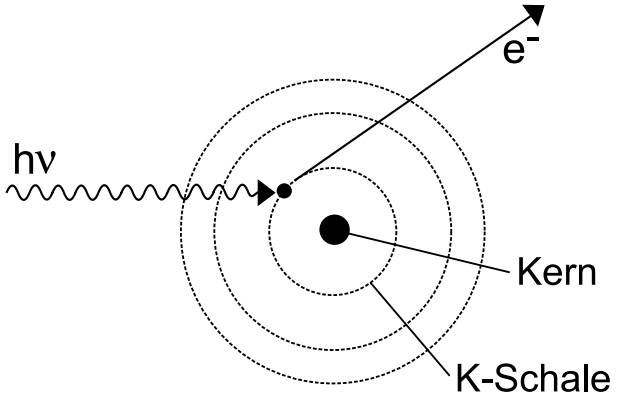

Abbildung 2: Photoeffekt

Paarbildung

Durch Wechselwirkung eines γ-Quants mit dem elektromagnetischen Feld eines Kerns oder eines Elektrons, kann es zur sogenannten Paarbildung kommen. Dafür ist allerdings eine γ-Energie von mehr als der doppelten Ruheenergie des Elektrons, also von über $2 \cdot 511keV$ nötig. Das γ-Quant wird dabei vollkommen absorbiert und es entsteht ein Positron-Elektron-Paar; die überschüssige Energie wird den beiden Teilchen, willkürlich verteilt, als kinetische Energie mitgegeben. Das Positron kann jedoch nicht allein existieren. Es vereinigt sich innerhalb kürzester Zeit mit einem Elektron und zerstrahlt dann in zwei γ-Quanten von je $511keV$, die dann durch Photo- oder Compton-Effekt absorbiert werden können. Der Paarbildungseffekt überwiegt bei hohen γ-Energien.

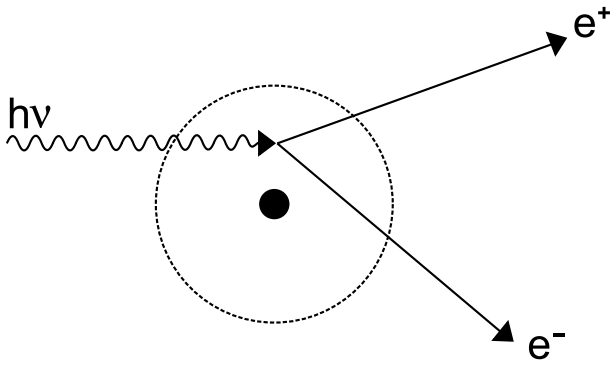

Abbildung 3: Paarbildung

Compton-Effekt

Als Compton-Effekt bezeichnet man die Streuung eines γ-Quants an einem nur leicht gebundenen Elektron. Dabei überträgt das γ-Quant einen Teil seiner Energie auf das Elektron und bewegt sich dann mit verminderter Energie, also größerer Wellenlänge weiter, wohingegen das Elektron an kinetischer Energie gewonnen hat. Das γ-Quant überträgt am meisten Energie auf das Elektron, wenn es gerade zurückgestreut wird, wenn also der Streuwinkel ϑ 180° beträgt. Mehr Energie kann durch Compton-Streuung nicht übertragen werden. Daher bricht an diesem Punkt im Energiespektrum die Compton-Verteilung ab, man nennt die Stelle auch Compton-Kante. Im Folgenden soll der Prozess der Comptonstreuung noch genauer untersucht werden.

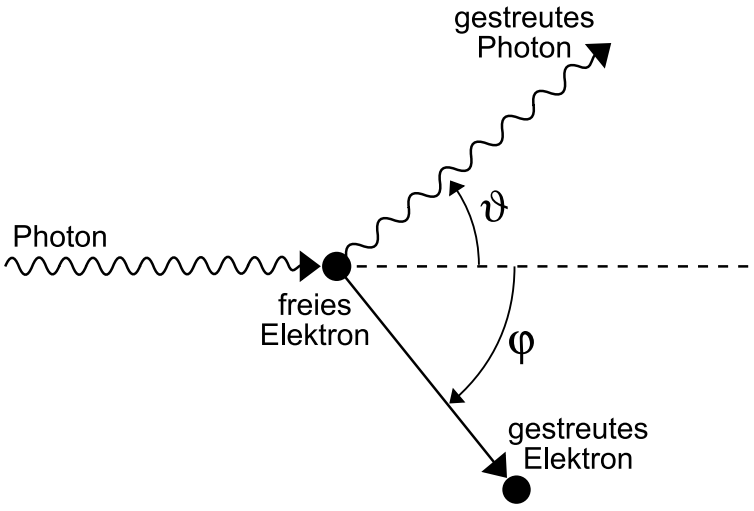

Abbildung 4: Compton-Streuung

2.4. Die Compton-Streuung

Da die Compton-Streuung in diesem Versuch die zentrale Rolle einnimmt, wird sie im Folgenden noch einmal genauer beschrieben. Das am Streuprozess beteiligte Photon habe vor der Streuung eine Energie von $E_\gamma = h\nu$ und einem Impuls von $\boldsymbol{p}_\gamma$, und nach der Streuung eine Energie $E'_\gamma = h\nu'$ und einen Impuls $\boldsymbol{p}'_\gamma$. Geht man nun von einem ruhenden, freien[1] Elektron aus, an dem die Streuung stattfindet, so besitzt es eine Ruheenergie $E_e = m_0c^2$ und den Impuls $\boldsymbol{p}_e = \boldsymbol{0}$ vor der Streuung, und eine Energie E'_e und den Impuls $\boldsymbol{p}'_e$ danach. Bei relativistischer Betrachtung gilt damit der Energiesatz:

$$\begin{aligned} E_\gamma + E_e &= E'_\gamma + E'_e \\ h\nu + m_0c^2 &= h\nu' + \sqrt{m_0^2c^4 + p_e'^2c^2}\ . \end{aligned} \tag{2}$$

Durch Umstellen und Quadrieren erhält man:

$$m_0^2c^4 + p_e'^2c^2 = h^2(\nu - \nu')^2 + 2h(\nu - \nu')m_0c^2 + m_0^2c^4\ .$$

Löst man nun nach dem Impuls des gestreuten Elektrons auf, so ergibt sich:

$$\boldsymbol{p}_e'^2 = \frac{h^2}{c^2}(\nu - \nu')^2 + 2h(\nu - \nu')m_0\ . \tag{3}$$

Betrachtet man weiter die Impulserhaltung, so gilt wegen $\boldsymbol{p}_e = \boldsymbol{0}$:

$$\boldsymbol{p}_\gamma = \boldsymbol{p}'_\gamma + \boldsymbol{p}'_e \tag{4}$$

Auflösen nach $\boldsymbol{p}'_e$ und Quadrieren liefert:

$$\boldsymbol{p}_e'^2 = \left(\boldsymbol{p}_\gamma - \boldsymbol{p}'_\gamma\right)^2 = \boldsymbol{p}_\gamma^2 - 2\,\boldsymbol{p}_\gamma \cdot \boldsymbol{p}'_\gamma + \boldsymbol{p}_\gamma'^2\ .$$

Mit den Beträgen der Photonen-Impulse

$$\left|\boldsymbol{p}_\gamma\right| = \frac{h\nu}{c} \quad \text{bzw.} \quad \left|\boldsymbol{p}'_\gamma\right| = \frac{h\nu'}{c}$$

und dem Winkel ϑ zwischen dem Impuls des einfallenden und des gestreuten Photons (siehe Abbildung 4) folgt:

$$\begin{aligned} \boldsymbol{p}_e'^2 &= \frac{h^2}{c^2}\nu^2 - \frac{2h^2}{c^2}\nu\nu'\cos\vartheta + \frac{h^2}{c^2}\nu'^2 \\ &= \frac{h^2}{c^2}\left(\nu^2 + \nu'^2 - 2\nu\nu'\cos\vartheta\right) \\ &= \frac{h^2}{c^2}\left(\left(\nu - \nu'\right)^2 + 2\nu\nu' - 2\nu\nu'\cos\vartheta\right) \\ &= \frac{h^2}{c^2}\left(\left(\nu - \nu'\right)^2 + 2\nu\nu'\left(1 - \cos\vartheta\right)\right)\ . \end{aligned} \tag{5}$$

[1] Bei Energien des Photons im Bereich von 100 keV kann die Bindungsenergie der äußeren Elektronen von einigen eV vernachlässigt, und die Elektronen als frei angesehen werden.

Aus Gleichungen (3) und (5) ergibt sich:

$$\frac{h^2}{c^2}\left(\nu-\nu'\right)^2+2h\left(\nu-\nu'\right)m_0 \;=\; \frac{h^2}{c^2}\left(\left(\nu-\nu'\right)^2+2\nu\nu'\left(1-\cos\vartheta\right)\right)$$

und damit:

$$\frac{\nu-\nu'}{\nu\nu'}=\frac{h}{m_0c^2}\left(1-\cos\vartheta\right) \;\Leftrightarrow\; \frac{1}{\nu'}-\frac{1}{\nu}=\frac{h}{m_0c^2}\left(1-\cos\vartheta\right)\ .$$

Mit dem Zusammenhang zwischen Frequenz und Wellenlänge

$$\nu=\frac{c}{\lambda} \quad \text{bzw.} \quad \nu'=\frac{c}{\lambda'}\ , \qquad \text{sowie} \quad \lambda_c=\frac{h}{m_0c}$$

erhält man somit für die Wellenlängenänderung $\Delta\lambda$:

$$\Delta\lambda=\lambda'-\lambda=\lambda_c\left(1-\cos\vartheta\right)\ , \tag{6}$$

wobei λ_c die sogenannte *Compton-Wellenlänge* des Elektrons ist. Man sieht hier, dass die Wellenlängenänderung $\Delta\lambda$ unabhängig von der Wellenlänge bzw. Energie des einfallenden Photons ist und nur vom Streuwinkel ϑ desselben abhängt. Die Änderung der Wellenlänge ist maximal, wenn das Photon genau entgegen der Einfallsrichtung, also um $\vartheta=180°$, gestreut wird. In diesem Fall gilt: $\Delta\lambda=2\lambda_c$.

Ausgehend von Gleichung (6) lässt sich die Energie der gestreuten Photonen bestimmen. Löst man nach λ' auf und bildet man den Kehrwert, so erhält man:

$$\frac{1}{\lambda'}=\frac{1}{\lambda+\lambda_c\left(1-\cos\vartheta\right)}\ .$$

Damit ergibt sich mit

$$E_\gamma=\frac{hc}{\lambda} \quad \text{bzw.} \quad E'_\gamma=\frac{hc}{\lambda'} \qquad \text{und der Abkürzung} \quad \alpha=\frac{E_\gamma}{m_0c^2}$$

für die Photonenenergie nach der Streuung:

$$E'_\gamma=\frac{E_\gamma}{1+\frac{E_\gamma}{m_0c^2}\left(1-\cos\vartheta\right)}=\frac{E_\gamma}{1+\alpha\left(1-\cos\vartheta\right)}\ . \tag{7}$$

Die kinetische Energie des Elektrons ergibt sich aus der Energiedifferenz des einlaufenden zum gestreuten Photon, also:

$$E'_e=E_\gamma-E'_\gamma=\frac{E_\gamma}{1+\frac{1}{\alpha(1-\cos\vartheta)}} \tag{8}$$

Wie erwartet, sieht man anhand von Gleichungen (7) und (8), dass die Energie des gestreuten Photons bei einem Streuwindel von $\vartheta=180°$ minimal und die auf das Elektron übertragene Energie somit maximal wird. Dabei gilt:

$$\left(E'_\gamma\right)_{min} \;=\; E'_\gamma\left(\vartheta=180°\right)=\frac{E_\gamma}{1+2\alpha} \tag{9}$$

$$\left(E'_e\right)_{max} \;=\; E'_e\left(\vartheta=180°\right)=\frac{E_\gamma}{1+\frac{1}{2\alpha}}\ . \tag{10}$$

Der theoretische Verlauf der Energien des gestreuten Photons und Elektrons in Abhängigkeit zum Streuwinkel ist in Abbildung 5 aufgetragen.

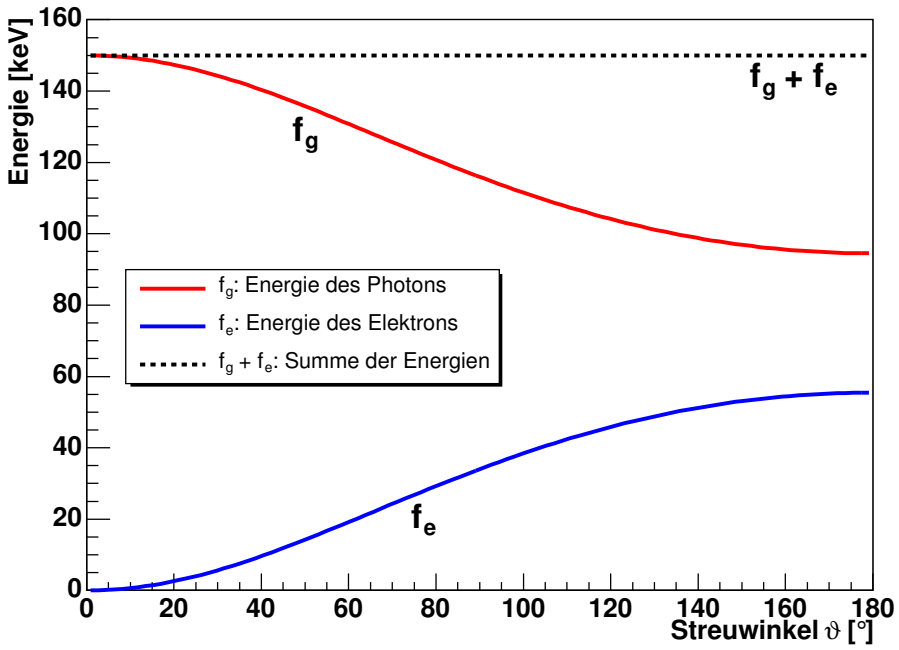

Abbildung 5: Energien der gestreuten Photonen und Elektronen bei Compton-Streuung mit Photonen der Energie $h\nu = 150\,\mathrm{keV}$ in Abhängigkeit zum Streuwinkel

2.5. Nachweis von γ-Strahlung

Bei γ-Strahlung handelt es sich um Strahlung von sehr hoher Energie die daher nur schlecht mit Materie wechselwirkt. Zum Nachweis dieser Strahlung verwendet man Szintillatoren. Man unterscheidet zwei Gruppen von Szintillatoren, anorganische und organische. Während beim organischen Szintillator hauptsächlich die einzelnen Moleküle für die Absorption des γ- Quants und die darauf folgende Lichtemission verantwortlich sind, finden beim anorganischen Szintillator diese Prozesse im Ionengitter eines Kristalls statt.

Der anorganische Szintillator

Ein anorganischer Szintillator besteht aus einem Kristall, der häufig auch mit Fremdatomen dotiert ist. Die einfallenden γ-Quanten geben ihre Energie an die Elektronen des Kristalls ab, die dadurch in das Leitungsband gehoben werden und an ihrer Stelle ein Loch hinterlassen. Rekombinieren diese Elektron-Loch-Paare, z.B. an durch die Fremdatome verursachten Fehlstellen, entstehen Licht-

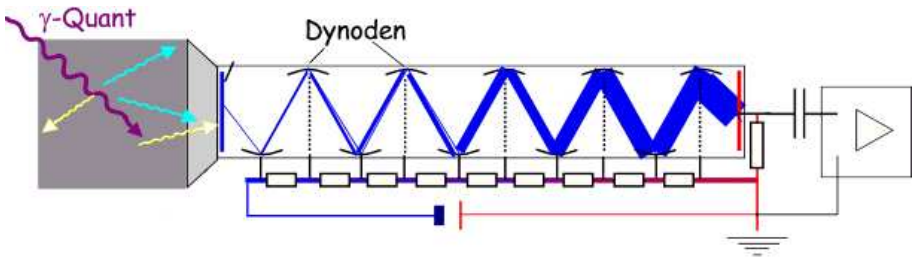

Abbildung 6: Schematische Darstellung eines Szintillationszählers[2]

blitze (Photonen) im nahen UV-Bereich. Je größer die Energie des einfallenden γ-Quants desto mehr Elektronen werden herausgelöst und desto mehr Photonen entstehen daher auch bei der Rekombination. Die Intensität des entstehenden Lichtblitzes ist also der Energie der einfallenden Strahlung proportional. Der Lichtblitz trifft dann auf die Kathode eines direkt anschließenden Photomultipliers, welcher das Signal verstärkt und messbar macht.

Der organische Szintillator

In organischen Szintillatoren beruht die Fluoreszenz auf der Anregung von Molekülen; die angeregten Molekülzustände senden dann bei ihrem Zerfall UV-Strahlung aus. Die Lichtausbeute organischer Szintillatoren ist wesentlich geringer als die der anorganischen Szintillatoren, weshalb sie sich nicht besonders gut zur Aufnahme von ganzen Spektren eignen, da ein großer Teil der γ-Strahlung gar nicht absorbiert wird. Dafür zeichnen sie sich durch eine kürzere Abklingzeit und damit auch eine hohe zeitliche Auflösung aus. Materialbedingt ist im organischen Szintillator die Wahrscheinlichkeit für das Auftreten der Comptonstreuung viel höher als die Wahrscheinlichkeit für den Photoeffekt, sodass man in einem mit dem organischen Szintillator aufgenommenen Spektrum fast keinen Photopeak, sondern nur eine deutliche Comptonkante sehen kann. Auch beim organischen Szintillator schließt sich gleich an den eigentlichen Szintillationskörper ein Photomultiplier an, der die emittierte UV-Strahlung in ein messbares Signal umwandelt.

Der Photomultiplier

Die aus einem Szintillator austretenden Photonen treffen anschließend auf die Kathode eines Photomultipliers (siehe Abbildung 6), wo sie durch Photoeffekt Elektronen herauslösen. Diese Elektronen werden dann innerhalb des Photomultipliers durch eine Spannung von Dynode zu Dynode beschleunigt wo sie immer mehr zusätzliche Elektronen herauslösen. Das Signal wird dadurch soweit

[2] Quelle: `http://www.roro-seiten.de/physik/nachweis/szintillationszahler.html`

verstärkt dass man schließlich, durch die an der Anode abfließenden Elektronen, einen messbaren elektrischen Impuls erhält.

2.6. Der differentielle Wirkungsquerschnitt

Um Streuversuche quantitativ zu beschreiben benötigt man den Begriff des Wirkungsquerschnitts. Man unterscheidet zwischen dem totalen und differentiellen Wirkungsquerschnitt. Der totale Wirkungsquerschnitt gibt die Wahrscheinlichkeit dafür an, dass ein eingestrahltes Teilchen durch das als Streuzentrum (Target) dienende Teilchen gestreut wird, also die Wahrscheinlichkeit für das Auftreten einer Wechselwirkung zwischen den beiden Teilchen. Er hat die Einheit einer Fläche und berechnet sich [2] durch:

$$\sigma_{\text{tot}} = \frac{R_{\text{streu}}}{R_{\text{ein}} \cdot \frac{N}{A}} \ , \tag{11}$$

dabei ist R_{streu} die Rate der gestreuten Teilchen, R_{ein} die Rate der einfallenden Teilchen, N die Anzahl der Streuzentren und A die bestrahlte Fläche.

Möchte man etwas über die Streuung in eine bestimmte Richtung, oder über die Streuung in Abhängigkeit der Energie erfahren so betrachtet man den differentiellen Wirkungquerschnitt. Er gibt die Wahrscheinlichkeit dafür an, dass ein Teilchen in ein Raumwinkelelement $\mathrm{d}\Omega$ gestreut wird und ist abhängig vom Streuwinkel ϑ, welcher wiederum von der Energie E und dem Stoßparameter b abhängt. Der Stoßparameter b ist der Abstand zwischen der Einfallsbahn des Teilchens und der Achse des Streuzentrums (siehe Abbildung 7). Werden die Teilchen in einen endlichen Raumwinkel $\Delta\Omega$ gestreut, so erhält man:

$$R_{\text{streu}} = R_{\text{ein}} \frac{N}{A} \Delta\Omega \left(\frac{\mathrm{d}\sigma}{\mathrm{d}\Omega} \right)_{\vartheta} \tag{12}$$

Für den differentiellen Wirkungsquerschnitt ergibt sich also:

$$\left(\frac{\mathrm{d}\sigma}{\mathrm{d}\Omega} \right)_{\vartheta} = \frac{R_{\text{streu}}}{R_{\text{ein}}} \frac{1}{\rho \cdot x} \frac{1}{\Delta\Omega} \ , \tag{13}$$

wobei hier die Flächendichte der Streuzentren N/A durch die Raumdichte ρ der Streuzentren mal die durchlaufene Strecke x des bestrahlten Materials ersetzt wurde. Durch das Integral über die volle Kugel:

$$\sigma_{\text{tot}} = \int \left(\frac{\mathrm{d}\sigma}{\mathrm{d}\Omega} \right)_{\vartheta} \mathrm{d}\Omega \tag{14}$$

erhält man wieder den totalen Wirkungsquerschnitt.

In diesem Versuch soll der Compton-Effekt, also die Streuung von γ-Quanten an näherungsweise freien, ruhenden Elektronen untersucht werden. Wir sind also am differentiellen Wirkungsquerschnitt für die Compton-Streuung interessiert. Klein

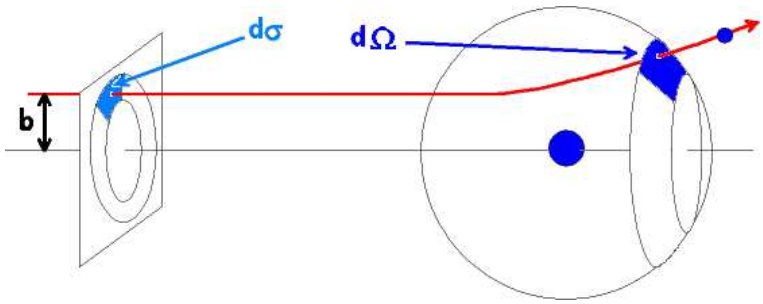

Abbildung 7: Der differentielle Wirkungsquerschnitt[3]

und Nishina erarbeiteten eine Formel welche die Energie der einfallenden Strahlung berücksichtigt. Für den Fall, dass die einfallende Strahlung linear polarisiert ist, ist der differentielle Wirkungsquerschnitt der Compton-Streuung durch die Klein-Nishina-Formel gegeben [1]:

$$\left(\frac{\mathrm{d}\sigma}{\mathrm{d}\Omega}\right)\Big|_{\mathrm{pol}} = \frac{1}{4} r_e^2 \left(\frac{E'_\gamma}{E_\gamma}\right)^2 \left(\frac{E_\gamma}{E'_\gamma} + \frac{E'_\gamma}{E_\gamma} + 4\cos^2\Theta - 2\right) \ , \tag{15}$$

wobei Θ der Winkel zwischen der Polarisation der einfallenden und der austretenden γ-Quanten ist. Ist die einfallende Welle unpolarisiert, so erhält man den differentiellen Wirkungsquerschnitt durch Bildung des Mittelwerts des obigen Ausdrucks über alle Winkel Θ. In Abhängigkeit des Streuwinkels ϑ ergibt sich damit:

$$\left(\frac{\mathrm{d}\sigma}{\mathrm{d}\Omega}\right)\Big|_{\mathrm{unpol}} = \frac{1}{2} r_e^2 \left(\frac{E'_\gamma}{E_\gamma}\right)^2 \left(\frac{E_\gamma}{E'_\gamma} + \frac{E'_\gamma}{E_\gamma} - \sin^2\vartheta\right) \ . \tag{16}$$

Drückt man nun noch die Energien der gestreuten γ-Quanten durch die Beziehung (7) aus, so erhält man:

$$\begin{aligned}\left(\frac{\mathrm{d}\sigma}{\mathrm{d}\Omega}\right)\Big|_{\mathrm{unpol}} \quad = \quad & \frac{1}{2} r_e^2 \Big\{ \left[1 + \alpha\left(1 - \cos\vartheta\right)\right]^{-3} \left[-\alpha\cos^3\vartheta \right. \\ & \left. + \left(\alpha^2 + \alpha + 1\right)\left(1 + \cos^2\vartheta\right) - \alpha\left(2\alpha + 1\right)\cos\vartheta\right] \Big\} \end{aligned} \tag{17}$$

[3] Quelle: `http://www.didaktik.physik.uni-erlangen.de/grundl_d_tph/`

3. Versuchsaufbau und Durchführung

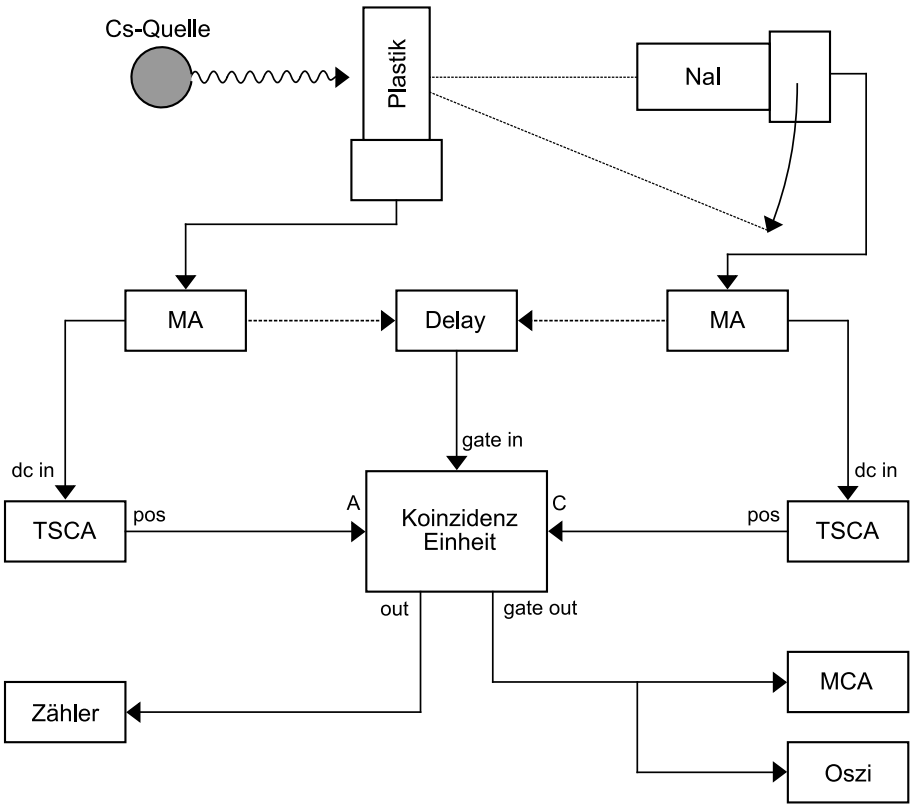

Abbildung 8: Versuchsaufbau

3.1. Aufbau

Zur Verfügung steht die γ-Quelle ^{137}Cs, ein organischer und ein anorganischer (NaI) Szintillator, sowie die zugehörige Nachweiselektronik mit Koinzidenzeinheit und Multichannelanalyser (MCA). Die eigentliche Compton-Streuung welche gemessen werden soll, findet im Plastikszintillator statt. Die dort gestreuten γ-Quanten werden mittels des anorganischen Szintillators für verschiedene Winkelpositionen gemessen. Da die Energieerhaltung verifiziert werden soll, müssen sowohl γ-Spektren mit dem NaI-Szintillator als auch Elektronenspektren mit dem Plastikszintillator aufgenommen werden. Das Signal des Szintillators mit dem ein Spektrum aufgenommen werden soll läuft zunächst in einen Hauptverstärker. Von dort aus wird das Signal aufgeteilt und zum Einen an einen TSCA weitergeleitet, wo das Signal der gewünschten Energie durch setzen eines Energiefensters heraussondiert und zusätzlich eine Verzögerung eingestellt werden kann; trifft ein Signal der gewünschten Energie ein, so sendet der TSCA einen Normpuls aus. Zum Anderen wird das Signal an ein Delay geschickt um die Verzögerung durch den TSCA auszugleichen, sodass das Signal welches dann von TSCA und Delay

an die Koinzidenzeinheit weitergegeben wird, dort gleichzeitig wieder zusammentrifft. Das Signal des jeweils anderen Szintillators wird zur Koinzidenz genutzt. Es wird zunächst auch in einem Hauptverstärker verstärkt, durchläuft dann einen TSCA und wird danach an die Koinzidenzeinheit weitergegeben. Beim TSCA muss die Verzögerung so gewählt werden, dass das Signal mit dem Signal des anderen Szintillators zeitlich abgeglichen ist, dass also zusammengehörige Pulse gleichzeitig bei der Koinzidenzeinheit angelangen. Das in die Koinzidenzeinheit integrierte Gate wird nur geöffnet wenn von beiden Szintillatoren gleichzeitig Signale eintreffen. In diesem Fall wird das Signal, welches nur über das Delay verzögert wurde, hindurch gelassen und an einen MCA abgegeben. Zusätzlich wird von der Koinzidenzeinheit ein Zähler angesteuert und ein Oszilloskop steht zur Verfügung um das Signal zu betrachten.

3.2. Durchführung

Als erstes machten wir uns mit der Messelektronik vertraut indem wir an verschiedenen Stellen, z.B. nach dem Hauptverstärker, das Oszilloskop dazuschalteten und die Form des auslaufenden Signals betrachteten. Nachdem wir geeignete Einstellungen gefunden hatten nahmen wir das γ-Spektrum von ^{137}Cs mit dem NaI-Szintillator und mit dem Plastikszintillator auf. Danach führten wir mittels einer weiteren Linie von ^{22}Na eine Energieeichung des MCA für beide Szintillatoren durch.

Als nächstes waren der NaI-Szintillator und der Plastikszintillator zeitlich abzugleichen. Dazu stellten wir das ^{22}Na-Präparat in die Mitte zwischen beide Szintillatoren. Bei ^{22}Na tritt ein Elektron-Positron-Vernichtungsprozess auf, bei dem zwei γ-Quanten mit je 511kev gleichzeitig in entgegengesetzter Richtung emittiert werden. Die Gleichzeitigkeit dieses Ereignisses wurde nun zum Abgleich der Szintillatoren benutzt. Wir betrachteten auf dem Oszilloskop das Signal beider Szintillatoren am Eingang zur Koinzidenzeinheit und brachten die Signale durch Variation der Verzögerung an den TSCAs zur Deckung. Somit kamen nun gleichzeitig stattfindende Ereignisse auch gleichzeitig bei der Koinzidenzeinheit an und konnten so für eine Koinzidenzschaltung verwendet werden. Nun war noch das analoge Signal aus dem Hauptverstärker einer der beiden Szintillatoren mit dem Gate der Koinzidenzeinheit abzugleichen, welches geöffnet wurde wenn Koinzidenz auftrat. Mit Hilfe des Delays konnte das analoge Signal so verzögert werden, dass es genau in das Gate hineinfiel.

Nun begannen wir mit der eigentlichen Messung und nahmen bei eingeschalteter Koinzidenz das γ-Spektrum mit dem NaI-Szintillator und das Elektronenspektrum mit dem Plastikszintillator für vier verschiedene Winkelpositionen auf. Danach führten wir nochmals eine Energieeichung durch. Schließlich nahmen wir noch eine Intensitätsmessung bei einem Winkel von 0° zur Bestimmung des differentiellen Wirkungsquerschnitts, sowie eine Messung des Untergrunds und eine Messung der zufälligen Koinzidenzen auf.

4. Auswertung

4.1. Aufgenommene Spektren und Energieeichung

Zur Energieeichung des MCA bezüglich der beiden Szintillationszähler wurde neben der ^{137}Cs-Quelle eine ^{22}Na-Quelle verwendet (zu den Quellen siehe auch Abschnitt 2.2). Bei den mit diesen Quellen aufgenommenen Spektren sind die in Tabelle 1 aufgeführten Photopeaks, Comptonkanten und Rückstreupeaks zu erwarten. Bei den aufgeführten Photopeaks handelt es sich um die Literatera-

Quelle	Energie [keV]	Typ	Szintillator
^{22}Na	1274,53	Photopeak	NaI
	1061,70	Comptonkante	NaI + Plastik
	510,999	Photopeak	NaI
	340,666	Comptonkante	NaI + Plastik
	212,834	Rückstreupeak	NaI
	170,333	Rückstreupeak	NaI
^{137}Cs	661,657	Photopeak	NaI
	477,334	Comptonkante	NaI + Plastik
	184,323	Rückstreupeak	NaI

Tabelle 1: Die theoretisch beobachtbaren Peaks

turwerte[4]. Die Comptonkanten, also die maximal auf ein Elektron übertragenen Streuenergien, wurden nach Gleichung (10) berechnet. Bei den Rückstreupeaks handelt es sich um γ-Quanten, die in den Detektorwänden um 180° gestreut werden, dadurch in den Szintillationskristall zurückkehren und dann als Photopeak detektiert werden. Die untere Energiegrenze der Rückstreuung kann man also aus der Energie des einfallenen Photons mit Hilfe von Gleichung (9) bestimmen.

Die mit jeweils beiden Quellen und Detektoren aufgenommenen Spektren sind in den Abbildungen 9 bis 12 dargestellt. Während die Photopeaks in den Spektren des NaI-Szintillators gut zu erkennen sind, sind diese in den mit dem Plastikszintillator aufgenommenen Spektren nicht sichtbar. Zur Energieeichung des Plastikszintillators bleiben also nur noch die zwei Comtonkanten der ^{22}Na- und die Comptonkante der ^{137}Cs-Quelle. Leider hielten wir uns bei der Versuchsdurchführung zu genau an die Angaben in der Anleitung und achteten bei der Na-Quelle nur darauf, dass die Comptonkante des 511 keV-Peaks im Messbereich des MCA zu sehen war. Daher blieb uns für die Eichung des Plastikszintillators nur noch jeweils eine Comptonkante pro Quelle, was einen vernünftigen Fit durch die ermittelten Kanalnummern der Peaks unmöglich machte. Daher ist davon auszugehen, dass diese Eichung zu einem erheblichen systematischen Fehler führt.

[4]Quelle: `http://www.nndc.bnl.gov/`

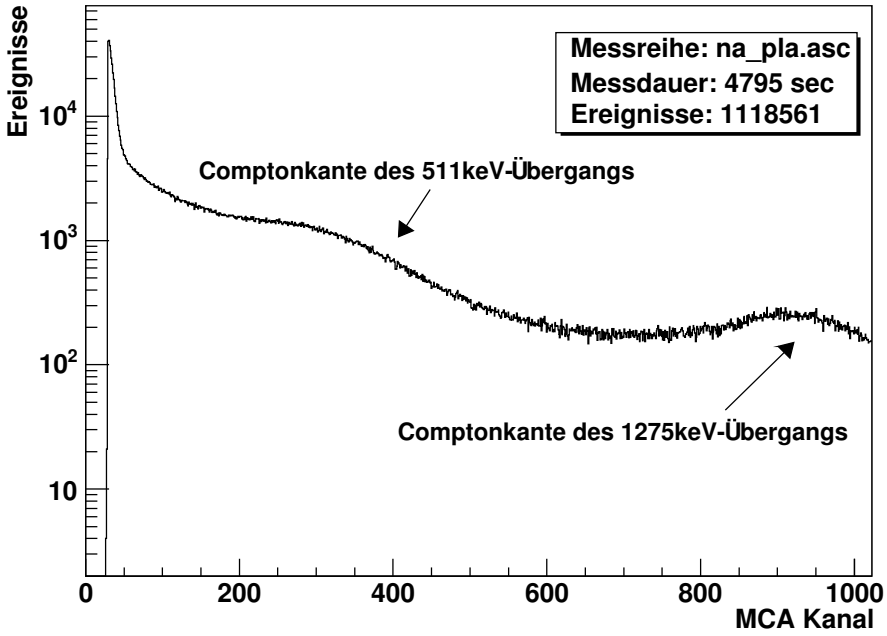

Abbildung 9: ^{22}Na-Spektrum des Plastikszintillators

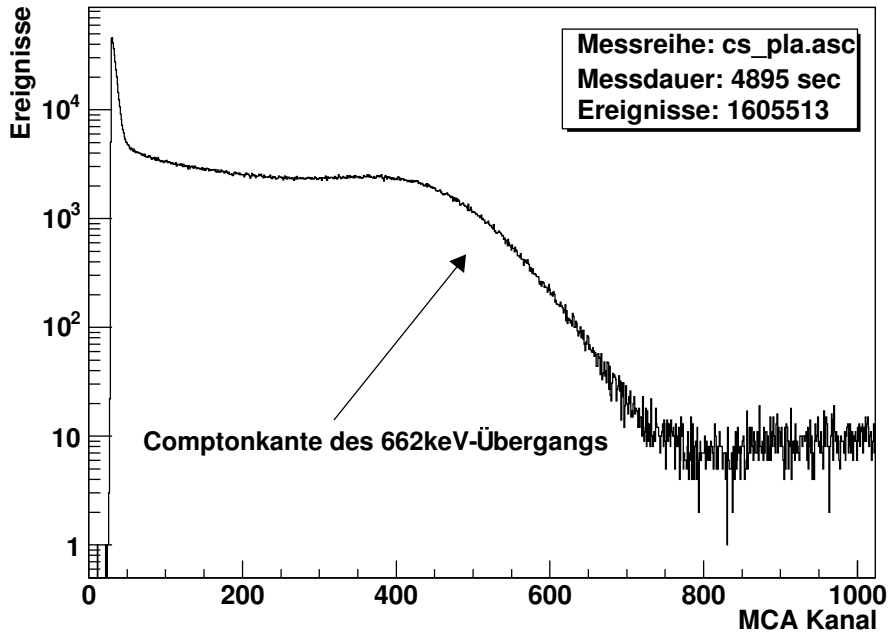

Abbildung 10: ^{127}Cs-Spektrum des Plastikszintillators

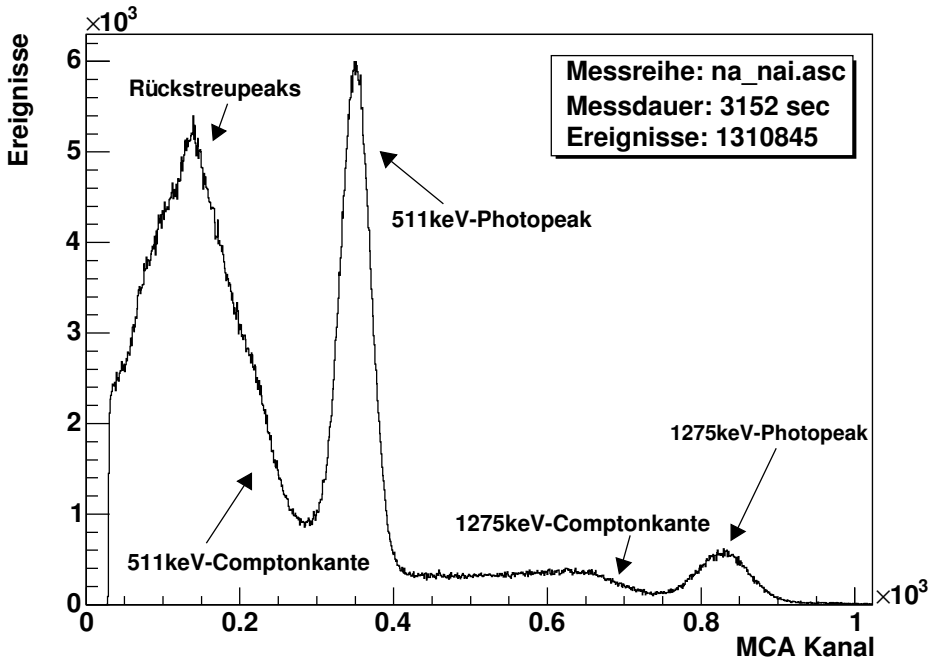

Abbildung 11: ^{22}Na-Spektrum des NaI-Szintillators

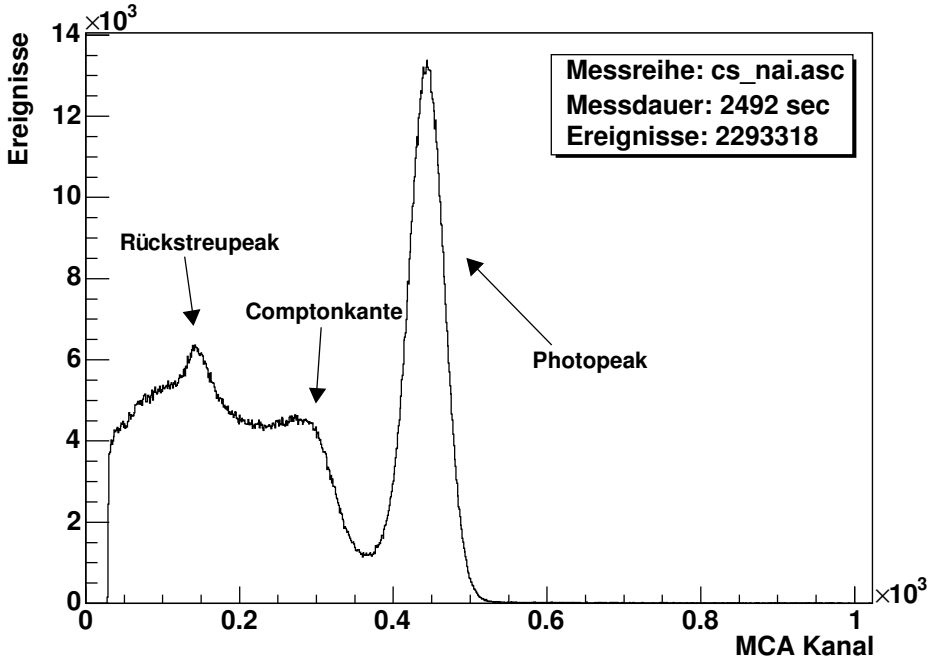

Abbildung 12: ^{127}Cs-Spektrum des NaI-Szintillators

Die Lage der Photopeaks bestimmten wir, indem wir Gaußfits durchführten. Zur Bestimmung der Comptonkante gingen wir zur Vereinfachung anstelle von der Klein-Nishina-Formel von einer Stufenfunktion aus und falteten diese mit einer Gaußverteilung. Das Ergebnis dieser Faltung ist die komplementäre Fehlerfunktion:

$$\operatorname{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2} \mathrm{d}t \ ,$$

welche wir dann zum Fitten der Comptonkanten verwendeten. Als Fehler verwendeten wir den Fehler des Fits, mindestens jedoch einen Fehler von einem MCA-Kanal. Die durchgeführten Fits sind im Anhang A.1 dargestellt; sie wurden an den vom Untergrund bereinigten Spektren durchgeführt.

Die durch die Fits ermittelten Kanäle trugen wir gegen die theoretischen Energien aus Tabelle 1 auf und erhielten so die Energie-Eichgeraden der beiden Szintillatoren. Die Eichgerade des Plastikszintillators ist in Abbildung 13 und die des NaI-Szintillators in Abbildung 14 dargestellt. Dabei wurden jeweils die Eichmessungen, welche wir zu Beginn durchgeführt hatten, sowie die Eichmessung vom Ende der Versuchsdurchführung gefittet. Man sieht, dass die beiden, nur aus zwei Messpunkten bestehenden, Messungen beim Plastikszintillator stark voneinander abweichen. Zur weiteren Auswertung verwendeten wir eine Eichgerade, bei welcher wir die Steigung und den Achsenabschnitt durch das gewichtete Mittel der Anfangs- und Endmessung bestimmten.

Für die Energie E in Abhängigkeit zum MCA-Kanal x:

$$E(x) = ax + b \tag{18}$$

erhielten wir für den Plastikszintillator:

$$a = (1{,}042 \pm 0{,}009)\,\mathrm{keV} \qquad \text{und} \qquad b = (-56{,}057 \pm 4{,}056)\,\mathrm{keV}$$

und für den NaI-Szintillator:

$$a = (1{,}581 \pm 0{,}003)\,\mathrm{keV} \qquad \text{und} \qquad b = (-41{,}245 \pm 1{,}436)\,\mathrm{keV}$$

Bei der im weiteren Verlauf der Auswertung verwendeten Umrechnung der Kanalnummer in Energie durch Gleichung (18) ist die Kanalnummer eine fehlerbehaftete Größe. Daher ist hier die im Folgenden verwendete[5] Fehlerfortpflanzung angegeben:

$$\begin{aligned} s_E &= \sqrt{\left(\frac{\partial E}{\partial a}\right)^2 s_a^2 + \left(\frac{\partial E}{\partial x}\right)^2 s_x^2 + \left(\frac{\partial E}{\partial b}\right)^2 s_b^2} \\ &= |ax| \sqrt{\left(\frac{s_a}{a}\right)^2 + \left(\frac{s_x}{x}\right)^2 + \left(\frac{s_b}{ax}\right)^2} \ . \end{aligned} \tag{19}$$

[5] siehe auch Quelltext in Anhang B.3

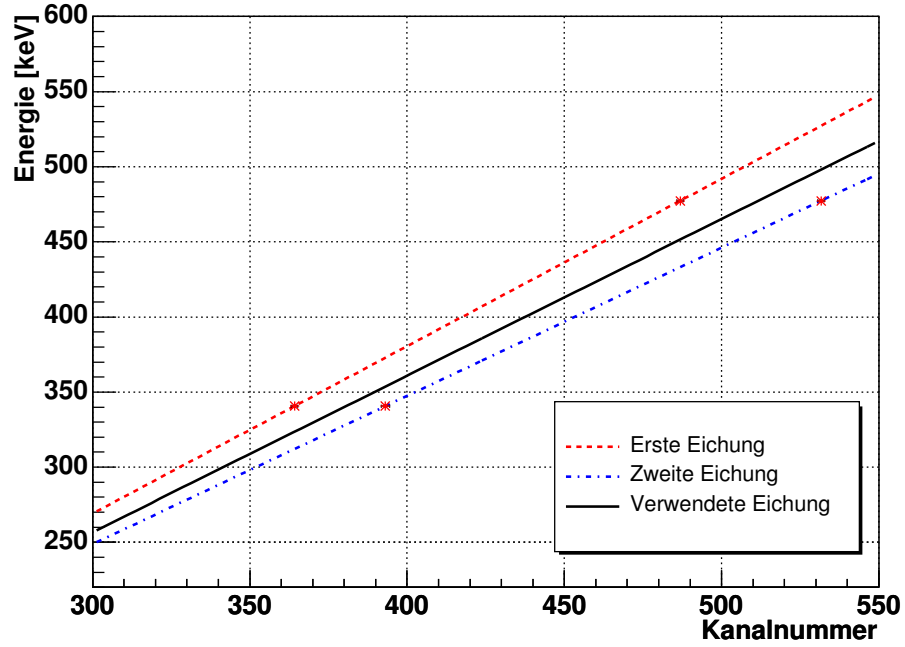

Abbildung 13: Energieeichung des Plastikszintillators

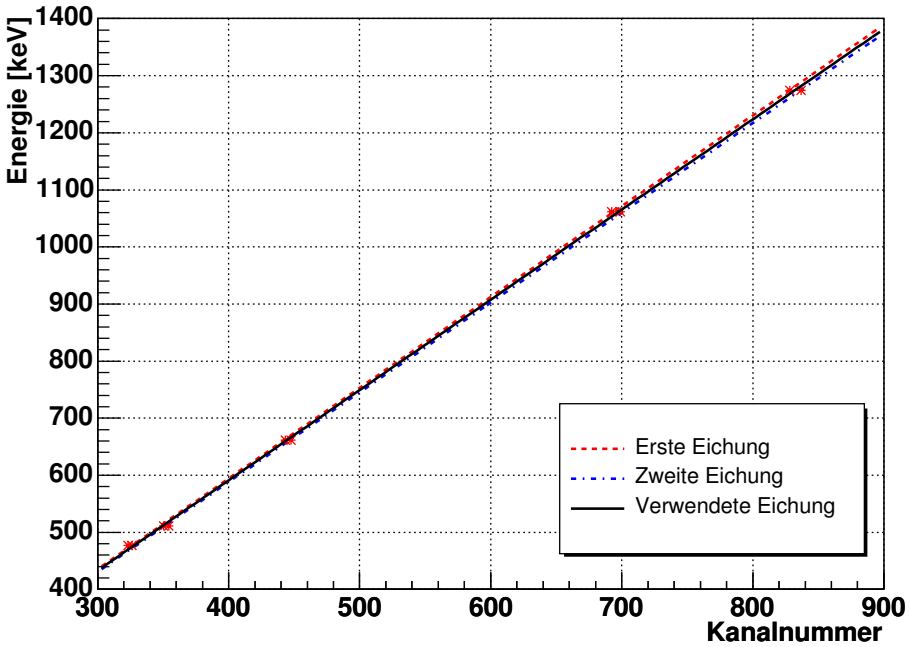

Abbildung 14: Energieeichung des NaI-Szintillators

Zur Bestimmung der Rückstreupeaks führten wir Gaußfits in den erwarteten Energiebereichen der mit dem NaI-Szintillator aufgenommenen Spektren durch (siehe Abbildungen 15 und 16).

Als Ergebnis erhielten wir für die Na-Quelle:

$$E_R^{Na} = (182{,}7 \pm 1{,}6)\,\text{keV}$$

und für die Cs-Quelle:

$$E_R^{Cs} = (188{,}8 \pm 1{,}6)\,\text{keV}\ .$$

Beide Werte liegen über den theoretischen Werten[6] von $E_R^{511} = 170{,}3\,\text{keV}$ für die 511 keV-Anihilationsstrahlung von ^{22}Na und $E_R^{662} = 184{,}3\,\text{keV}$ des 662 keV-Übergangs von ^{137}Cs. Während der im Cs-Spektrum bestimmte Peak noch relativ nah am theoretischen Wert liegt, weicht der Peak aus dem Na-Spektrum weit davon ab. Die Abweichung liegt zum einen daran, dass die genaue Lage des Rückstreupeaks von der Geometrie des Detektors abhängt und der theoretische Wert nur eine untere Schranke für die mögliche Energie des Rückstreupeaks darstellt. Zum Anderen handelt es sich speziell beim ermittelten Rückstreupeak im Na-Spektrum eigentlich um zwei Peaks[6]; nämlich den Rückstreupeak der 511 keV-Anihilationsstrahlung und den des 1275 keV-Übergangs, welcher eine etwas höhere theoretische Energie von $E_R^{1275} = 212{,}8\,\text{keV}$ besitzt.

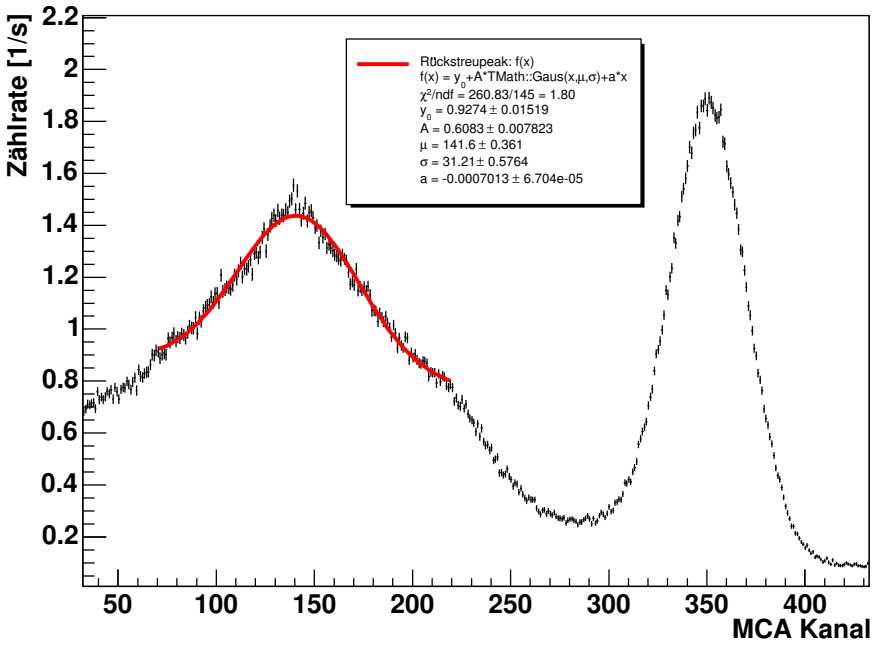

Abbildung 15: Rückstreupeak der ^{22}Na-Quelle im NaI-Szintillator

[6] siehe Tabelle 1

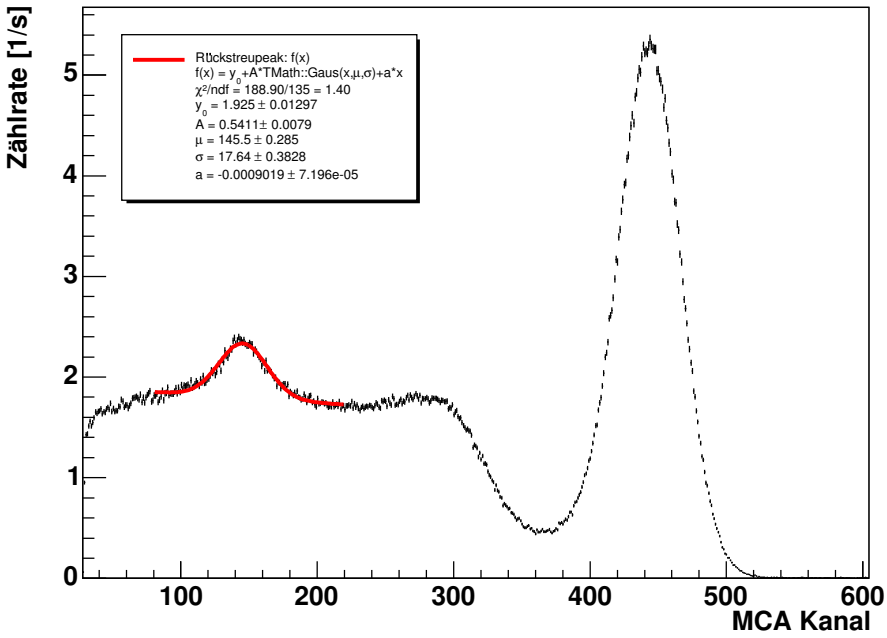

Abbildung 16: Rückstreupeak der ^{137}Cs-Quelle im NaI-Szintillator

4.2. Winkelabhängige Messung der Compton-Streuung

In diesem Versuchsteil sollte die Energieerhaltung beim Compton-Effekt verifiziert werden. Dazu nahmen wir die Energiespektren der gestreuten Photonen und die der Elektronen für verschiedene Streuwinkel auf, indem wir bei eingeschalteter Koinzidenzeinheit mit dem MCA das Spektrum des NaI-Szintillators bzw. des Plastikszintillators aufnahmen.

Zur Bestimmung der Lage der jeweiligen Energiepeaks führten wir bei jedem aufgenommenen Spektrum einen Gaußfit durch (siehe Anhang A.2). Die Ergebnisse für die verschiedenen Winkel sind zusammen mit den theoretischen Werten in Tabelle 2 aufgeführt. Dabei wurde zur Umrechnung der Kanäle in Energie Gleichung (18) mit den zugehörigen Koeffizienten für den NaI- bzw. Plastikszintillator verwendet, die Fehler der Messwerte wurden dabei nach Gleichung (19) aus den Fehlern der Fits und denen der Energieeichung fortgepflanzt. Zur Berechnung der theoretischen Streuenergien in Abhängigkeit zum Streuwinkel des Photons wurden die Gleichungen (7) und (8) aus Abschnitt 2.4 verwendet. Dabei ist mitberücksichtigt worden, dass es sich bei dem Winkel um eine fehlerbehaftete Größe handelt; wir schätzten den Fehler des Winkels auf $s_\vartheta = 2°$ ab. Leitet man Gleichungen (7) und (8) nach dem Winkel ϑ ab, so erhält man:

$$\frac{\partial E'_\gamma}{\partial \vartheta} = -\frac{\partial E'_e}{\partial \vartheta} = \frac{\alpha E_\gamma \sin\vartheta}{(1+\alpha(1-\cos\vartheta))^2} \ .$$

Streuart	Winkel [°]	Messwert [keV]	Theoriewert [keV]
Photonen	30 ± 2	$552{,}0 \pm 2{,}0$	$563{,}8 \pm 10{,}9$
	60 ± 2	$394{,}8 \pm 2{,}0$	$401{,}6 \pm 9{,}5$
	90 ± 2	$278{,}3 \pm 1{,}6$	$288{,}3 \pm 5{,}7$
	120 ± 2	$217{,}8 \pm 1{,}7$	$224{,}9 \pm 3{,}0$
Elektronen	30 ± 2	$48{,}6 \pm 4{,}2$	$97{,}8 \pm 10{,}9$
	60 ± 2	$216{,}8 \pm 4{,}7$	$260{,}0 \pm 9{,}5$
	90 ± 2	$342{,}0 \pm 5{,}4$	$373{,}3 \pm 5{,}7$
	120 ± 2	$406{,}2 \pm 5{,}9$	$436{,}8 \pm 3{,}0$

Tabelle 2: Gemessene und theoretische Werte für die Energien der gestreuten Photonen und Elektronen bei verschiedenen Streuwinkeln

Es ergibt sich also als Fehler für die theoretisch berechneten Werte:

$$s_{E'_\gamma} = s_{E'_e} = \sqrt{\left(\frac{\partial E'_\gamma}{\partial \vartheta}\right)^2 s_\vartheta^2} = \left|\frac{\alpha E_\gamma \sin\vartheta}{(1+\alpha(1-\cos\vartheta))^2}\right| s_\vartheta \ .$$

Weiter sind die Messwerte zusammen mit dem theoretischen Verlauf der Streuuenergien in Abbildung 17 dargestellt.

Vergleicht man die Messergebnisse mit den theoretischen Werten, so stellt man fest, dass die Messwerte zum Teil erheblich unter den Theoriewerten liegen. Während die gemessenen Photonenenergien noch innerhalb eines Fehlerintervalls von 1 bis 2σ mit den theoretischen Werten übereinstimmen, liegen die Elektronenenergien im Vergleich zur Theorie weit außerhalb des Fehlerbereichs. Da dabei alle Werte weit unterhalb des theoretischen Verlaufs liegen, muss es sich um einen gravierenden systematischen Fehler handeln, der höchstwahrscheinlich auf die sehr ungenaue, aus nur zwei Messpunkten bestehende, Energieeichung zurückzuführen ist.

Die Energieerhaltung bei der Compton-Streuung läßt sich also auf diese Weise leider nicht quantitativ verifizieren. Die in Tabelle 3 aufgeführten Summen der Photonen- und Elektronenenergien weichen zu stark von dem theoretischen Wert ab. Allerdings zeigt sich in Abbildung 17 deutlich eine Übereinstimmung

Winkel [°]	Summe der gemessenen Energien [keV]	Theoriewert [keV]
30	$600{,}6 \pm 4{,}7$	661,7
60	$611{,}6 \pm 5{,}1$	
90	$620{,}3 \pm 5{,}6$	
120	$623{,}9 \pm 6{,}2$	

Tabelle 3: Summe der Energien der gestreuten Photonen und Elektronen

der Messwerte mit dem theoretischen Verlauf, wenn man einen von dem systematischen Fehler herrührenden Offset in betracht zieht.

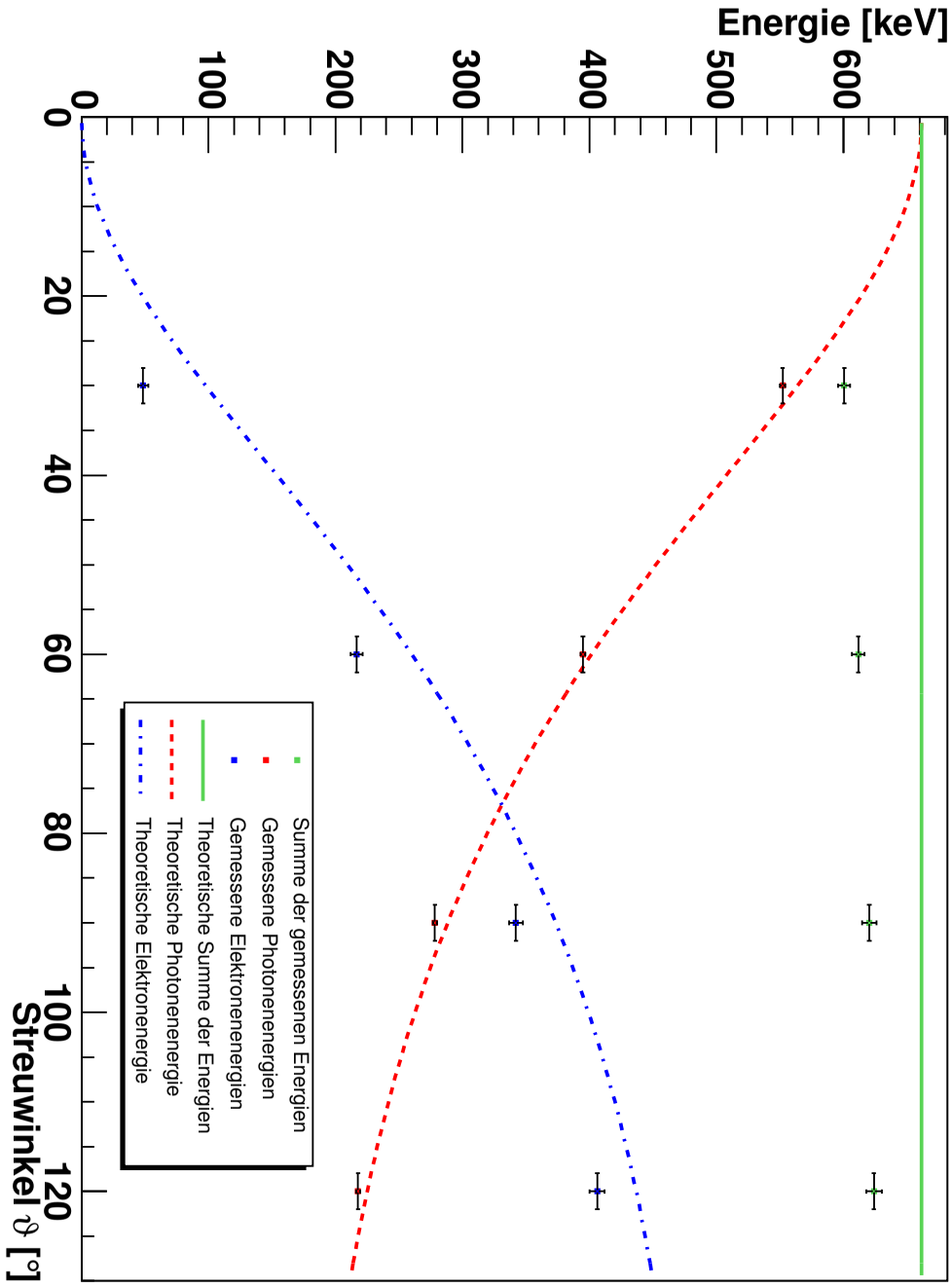

Abbildung 17: Gemessenen Photonen- und Elektronenenergien bei unterschiedlichen Streuwinkeln der Photonen, sowie der theoretische Verlauf der Energien

4.3. Differentieller Wirkungsquerschnitt

Den differentiellen Wirkungsquerschnitt bestimmten wir mit Hilfe von Gleichung (13):

$$\left(\frac{\mathrm{d}\sigma}{\mathrm{d}\Omega}\right)_{\vartheta} = \frac{R_{\text{streu}}}{R_{\text{ein}}} \frac{1}{\rho \cdot x} \frac{1}{\Delta\Omega} \ .$$

Um diese Formel verwenden zu können, müssen die Rate R_{ein} des einfallenden Strahls, die Rate R_{streu} der gestreuten γ-Quanten, der im Plastikszintillator zurückgelegte Weg x sowie der durch die gestreuten γ-Quanten bestrahlte Raumwinkel $\Delta\Omega$ bestimmt werden. Die Raumdichte ρ der Streuzentren (Elektronen) war mit $\rho = 3{,}4 \cdot 10^{23}\,\mathrm{cm}^{-3}$ angegeben.

Zur Bestimmung von R_{ein} gingen wir von der Intensitätsmessung bei einem Winkel von 0° aus. Von der daraus bestimmten Rate R_{int} zogen wir die Untergrundrate R_{ug} ab. Als resultierende Rate ergibt sich:

$$R'_{\text{int}} = R_{\text{int}} - R_{\text{ug}} \qquad \text{mit Fehler:} \quad s_{R'_{\text{int}}} = \sqrt{s^2_{R_{\text{int}}} + s^2_{R_{\text{ug}}}} \ ,$$

dabei berechnet sich der Fehler einer Zählrate $R = N/t$ durch:

$$s_R = |R| \sqrt{\left(\frac{s_N}{N}\right)^2} = \frac{N}{t} \ .$$

Weiter muss die Effizienz ε des NaI-Szintillators und die zuvor erfolgte Absorption im Plastikszintillator berücksichtigt werden. Die Effizienz und der Absorptionskoeffizient μ ließen sich aus Diagrammen aus dem Praktikumsordner ablesen (siehe Tabelle 4, Winkel 0°). Insgesamt berechneten wir also R_{ein} durch:

$$R_{\text{ein}} = \frac{R'_{\text{int}}}{\varepsilon}\, e^{\mu d} \ ,$$

wobei $d = 1\,\mathrm{cm}$ die Dicke des Plastiszintillators ist. Für den Fehler ergibt sich:

$$s_{R_{\text{ein}}} = |R_{\text{ein}}| \sqrt{\left(\frac{s_{R'_{\text{int}}}}{R'_{\text{int}}}\right)^2 + \left(\frac{s_\varepsilon}{\varepsilon}\right)^2}$$

dabei nahmen wir für den Fehler der Effizienz einen Ablesefehler von $s_\varepsilon = 0{,}02$ an. Für μ und d waren leider keine Fehler angegeben, weshalb wir sie nicht mitberücksichtigten.

Während bei der 0°-Messung der Weg x der γ-Quanten durch den Plastikszintillator genau der Dicke d des Szintillators entsprach, muss für die anderen Winkel die Winkelabhängigkeit des Wegs berücksichtigt werden. Da wir den Plastikstikszintillator immer so drehten, dass die Flächennormale mit dem halben Streuwinkel zusammenfiel, lässt sich der Weg durch den Plastikszintillator durch geometrische Überlegungen (siehe Abbildung 18) wie folgt berechnen:

$$x = \frac{d}{\cos\frac{\vartheta}{2}} \ .$$

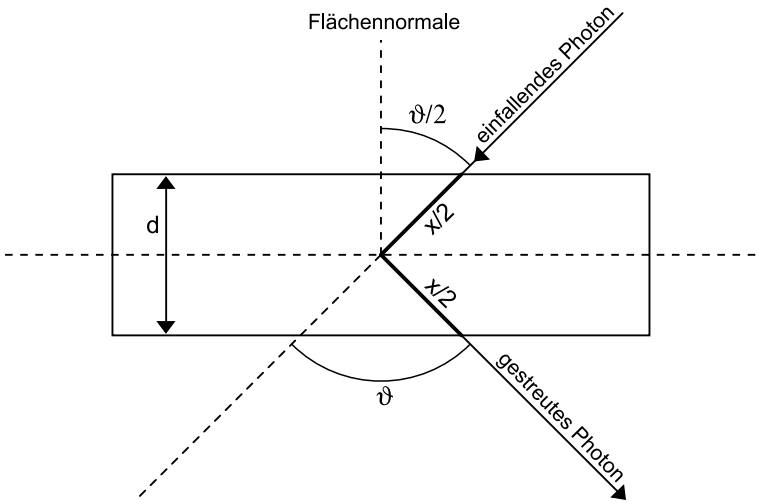

Abbildung 18: Weg der Photonen durch den Plastikszintillator

Mit einem Fehler des eingestellten Winkels von $s_\vartheta = 2°$ ergibt sich für den Fehler der Wegstrecke:

$$s_x = \sqrt{\left(\frac{\partial x}{\partial \vartheta}\right)^2 s_\vartheta^2} = \frac{x}{2} \tan\left(\frac{\vartheta}{2}\right) .$$

Um die Rate R_{streu} der gestreuten γ-Quanten zu bestimmen gingen wir ähnlich wie bei R_{ein} vor. Allerdings musste in diesem Fall sowohl die Rate R_{zuf} der zufälligen Koinzidenzen als auch die Rate R_{ugc} des Untergrunds bei eingeschalteter Koinzidenz von der eigentlichen Messung abgezogen werden. Damit ergibt sich als bereinigte gemessene Rate:

$$R'_{\text{mes}} = R_{\text{mes}} - R_{\text{zuf}} - R_{\text{ugc}}$$

und der Fehler:

$$s_{R'_{\text{mes}}} = \sqrt{s_{R_{\text{mes}}}^2 + s_{R_{\text{zuf}}}^2 + s_{R_{\text{ugc}}}^2}$$

Weiter muss auch hier die Effizienz ε des NaI-Szintillators berücksichtigt werden und die Möglichkeit, dass das gestreute γ-Quant im Plastikszintillator absorbiert werden kann. Dabei nehmen wir vereinfachend an, dass das γ-Quant auf halben Weg durch den Szintillator gestreut wird, wodurch es nur noch auf der zweiten Hälfte des Wegs, also über eine Strecke von $x/2$ absorbiert werden kann. Mit diesen beiden weiteren Korrekturen ergibt sich für die Rate der gestreuten γ-Quanten:

$$R_{\text{streu}} = \frac{R'_{\text{mes}}}{\varepsilon}\, e^{\mu x/2}$$

und für den Fehler:

$$s_{R_{\text{streu}}} = |R_{\text{streu}}| \sqrt{\left(\frac{s_{R'_{\text{mes}}}}{R'_{\text{mes}}}\right)^2 + \left(\frac{s_\varepsilon}{\varepsilon}\right)^2 + \left(\frac{\mu}{2} s_x\right)^2} .$$

Winkel ϑ [°]	Energie [keV]	Absorptionskoeff. μ [cm^{-1}]	Effizienz ε
0	662	0,089	0,427
30	564	0,093	0,465
60	402	0,108	0,564
90	288	0,121	0,691
120	225	0,132	0,797

Tabelle 4: Absorptionskoeffizient des Plastikszintillators und Effizienz des NaI-Szintillators in Abhängigkeit des Winkels bzw. der Energie des gestreuten Photons (aus Diagrammen abgelesen)

Die noch fehlende Größe $\Delta\Omega$ ist der der NaI-Detektorfläche A entsprechende Raumwinkel. Er lässt sich berechnen durch:

$$\Delta\Omega = \frac{A}{r^2} = \frac{\pi D^2}{4r^2} \ ,$$

wobei $r = 13{,}4 \pm 1\,\mathrm{cm}$ der Abstand zwischen Plastik- und NaI-Szintillator und $D = 3{,}8\,\mathrm{cm}$ der Durchmesser des NaI-Szintiallionskristalls ist. Für den Fehler des Raumwinkels ergibt sich damit:

$$s_{\Delta\Omega} = \sqrt{\left(\frac{\partial\Delta\Omega}{\partial r}\right)^2 s_r^2} = \left|2\,\Delta\Omega\,\frac{s_r}{r}\right| \ .$$

Mit den so berechneten Werten lässt sich nun mit Hilfe von Gleichung (13) der differentielle Wirkungsquerschnitt bestimmen. Der Fehler ergibt sich dabei einfach zu:

$$s_{\mathrm{d}\sigma/\mathrm{d}\Omega} = \left|\left(\frac{\mathrm{d}\sigma}{\mathrm{d}\Omega}\right)\right| \sqrt{\left(\frac{s_{R_{\mathrm{streu}}}}{R_{\mathrm{streu}}}\right)^2 + \left(\frac{s_{R_{\mathrm{ein}}}}{R_{\mathrm{ein}}}\right)^2 + \left(\frac{s_x}{x}\right)^2 + \left(\frac{s_{\Delta\Omega}}{\Delta\Omega}\right)^2}$$

Die für die gemessenen Winkel bestimmten differentiellen Wirkungsquerschnitte sind in Tabelle 5 aufgeführt und zusammen mit dem nach der Klein-Nishina-Formel berechneten theoretischen Verlauf in Abbildung 19 dargestellt.

Winkel ϑ [°]	diff. Wirkungsquers. [$10^{-30}\,\mathrm{m}^2$]	Klein-Nishina [$10^{-30}\,\mathrm{m}^2$]
30	$6{,}78 \pm 1{,}10$	5,12
60	$2{,}62 \pm 0{,}42$	2,20
90	$1{,}56 \pm 0{,}50$	1,30
120	$0{,}95 \pm 0{,}15$	1,16

Tabelle 5: Experimentell bestimmte und theoretisch berechnete Werte für den differentiellen Wirkungsquerschnitt

Die experimentellen Werte stimmen mit einer Abweichung von 1-2σ im Bereich ihrer Fehler relativ gut mit den theoretisch berechneten Werten überein. Allerdings scheint der Verlauf der Messwerte - sofern man das bei gerade mal 4 Messpunkten überhaupt beurteilen kann - nicht ganz der theoretischen Kurve

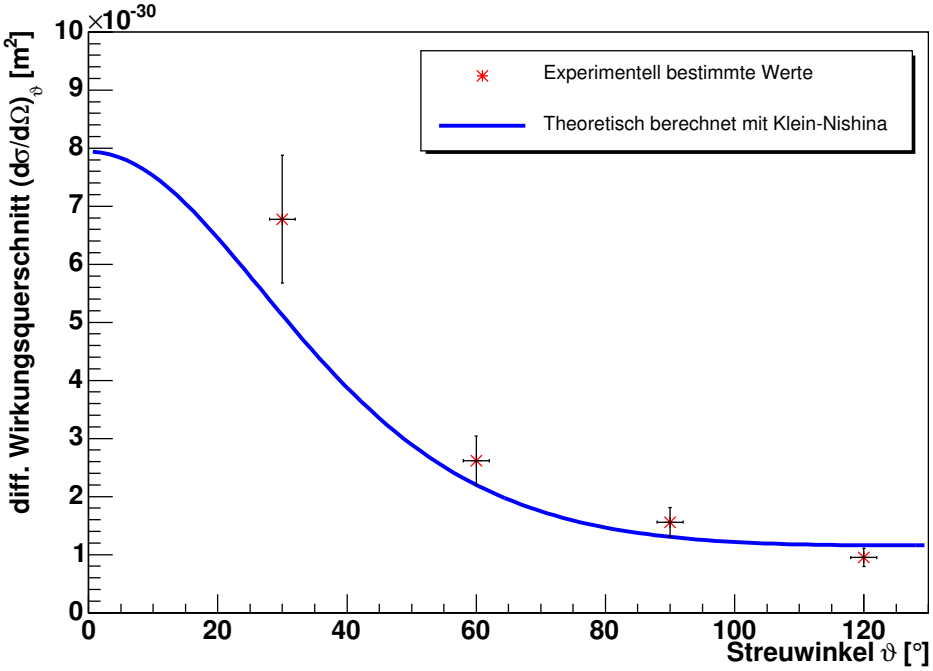

Abbildung 19: Experimentell bestimmter differentieller Wirkungsquerschnitt im Vergleich zur Theorie (Klein-Nishina-Formel)

zu folgen. Mit diesem Ergebnis lässt sich die Klein-Nishina-Formel für den differentiellen Wirkungsquerschnitt nur qualitativ bestätigen. Für eine quantitative Verifizierung wären mehr Messpunkte und genauere Messungen notwendig.

Besonders schade ist es, dass der Abstand r zwischen Plastik- und NaI-Szintillator nur sehr ungenau zu messen war, da dessen Fehler erheblich zu den Fehlern der Messwerte beiträgt. Ein weiterer Fehler, der auch signifikant sein könnte ist die Messzeit. Der MCA lieferte als Zeit nur die verstrichene Zeit (*real time*) und nicht die echte Messzeit (*live time*), bei welcher auch die Totzeit des Detektors berücksichtigt wird. Besonders stark wirkt sich dieser Fehler z.B. bei den Untergrundmessungen aus, bei denen die Totzeit des Detektors aufgrund der wenigen Ereignisse die eigentliche Messzeit praktisch nicht beeinflusst, während sie bei hohen Raten und Energien zu einer starken Differenz von gemessener und verstrichener Zeit führt.

5. Zusammenfassung

Bei der Energieeichung des MCA bezüglich des Plastik- und der NaI-Szintillators ergab sich für die Umrechnungen $E(x) = ax + b$ von Kanal zu Energie:

$$\begin{aligned} \text{Plastik:} \quad & a = (1{,}042 \pm 0{,}009)\,\text{keV}\ , \quad b = (-56{,}057 \pm 4{,}056)\,\text{keV} \\ \text{NaI:} \quad & a = (1{,}581 \pm 0{,}003)\,\text{keV}\ , \quad b = (-41{,}245 \pm 1{,}436)\,\text{keV}\ . \end{aligned}$$

Da beim Plastikszintillator nur zwei Messpunkte zur Verfügung standen, ist diese Eichung sehr ungenau. Man muss von einem noch größeren Fehler als dem schon beachtlichen relativen Fehler von fast 10% ausgehen, was vermutlich zu dem ausschlaggebenden systematischen Fehler der winkelabhängigen Messung der Streuenergien führte.

Bei der winkelabhängigen Messung der Streuenergien war die Energieerhaltung der Compton-Streuung zu verifizieren. Wir erhielten folgende Ergebnisse:

Streuart	Winkel [°]	Messwert [keV]	Theoriewert [keV]
Photonen	30 ± 2	$552{,}0 \pm 2{,}0$	$563{,}8 \pm 10{,}9$
	60 ± 2	$394{,}8 \pm 2{,}0$	$401{,}6 \pm 9{,}5$
	90 ± 2	$278{,}3 \pm 1{,}6$	$288{,}3 \pm 5{,}7$
	120 ± 2	$217{,}8 \pm 1{,}7$	$224{,}9 \pm 3{,}0$
Elektronen	30 ± 2	$48{,}6 \pm 4{,}2$	$97{,}8 \pm 10{,}9$
	60 ± 2	$216{,}8 \pm 4{,}7$	$260{,}0 \pm 9{,}5$
	90 ± 2	$342{,}0 \pm 5{,}4$	$373{,}3 \pm 5{,}7$
	120 ± 2	$406{,}2 \pm 5{,}9$	$436{,}8 \pm 3{,}0$

Dabei stimmen die Energien der gestreuten Photonen im Bereich von 1-2 Standardabweichungen mit den theoretisch berechneten Werten überein. Die Energien der Elektronen jedoch liegen alle weit unter den theoretischen Werten. Wie bereits erwähnt, ist dieser systematische Fehler vermutlich auf die sehr ungenaue Eichung des Plastikszintillators zurückzuführen. Entsprechend stark weichen daher auch die Summen der Streuenergien von dem theoretischen Wert ab. Die Energieerhaltung kann also nur qualitativ und unter Annahme eines von der Energieeichung verursachten Offsets der Elektronenenergien bestätigt werden.

Bei der experimentellen Bestimmung des Wirkungsquerschnitts ergaben sich folgende Werte:

Winkel ϑ [°]	diff. Wirkungsquers. $[10^{-30}\,\text{m}^2]$	Klein-Nishina $[10^{-30}\,\text{m}^2]$
30	$6{,}78 \pm 1{,}10$	5,12
60	$2{,}62 \pm 0{,}42$	2,20
90	$1{,}56 \pm 0{,}50$	1,30
120	$0{,}95 \pm 0{,}15$	1,16

Diese stimmen im Bereich von 1-2 Standardabweichungen mit den nach der Klein-Nishina-Formel berechneten Werten überein. Allerdings lässt hier die geringe

Zahl von Messpunkten keine quantitative Bestätigung der Theorie zu. Die Messwerte folgen zwar in etwa dem Verlauf der theoretischen Kurve, besitzen aber zum Teil sehr hohe Fehler und besonders die Randwerte scheinen nicht besonders gut zur theoretischen Klein-Nishina-Kurve zu passen.

Insgesamt hätten die Ergebnisse vermutlich dadurch verbessert werden können, wenn bei der Eichung des Plastikszintillators alle drei möglichen Comptonkanten verwendet worden wären. Weiter führt die Verwendung der verstrichenen Zeit (*real time*) anstelle der echten Messzeit (*live time*) bei der Berechnung von Zählraten und der Berücksichtigung der aufgenommenen Untergrundspektren zu erheblichen Fehlern, die durch einen anderen MCA oder eine andere Software hätten vermieden werden können. Neben den ungenauen Zählraten spielte bei der Bestimmung des Wirkungsquerschnitts auch noch der Fehler des Abstands zwischen Plastik- und NaI-Szintillator eine signifikante Rolle. Der Abstand war leider nicht genau bestimmbar, da die genaue Lage des Kristalls im Detektor nicht bekannt war.

In beiden Fällen, der Energieerhaltung und dem Wirkungsquerschnitt, ist die Verifizierung der Theorie nicht ganz so gut geglückt, ein Zusammenhang ist jedoch qualitativ zu erfassen.

Literatur

[1] Thomson and compton scattering of gamma-radiation. In *Praktikumsordner, FP II: Compton-Effekt*.

[2] German Hacker. *Grundlagen der Teilchenphysik*. Physikalisches Institut der Universität Erlangen, `http://www.didaktik.physik.uni-erlangen.de/grundl_d_tph/`, 2003.

A. Weitere Fits

A.1. Energieeichung

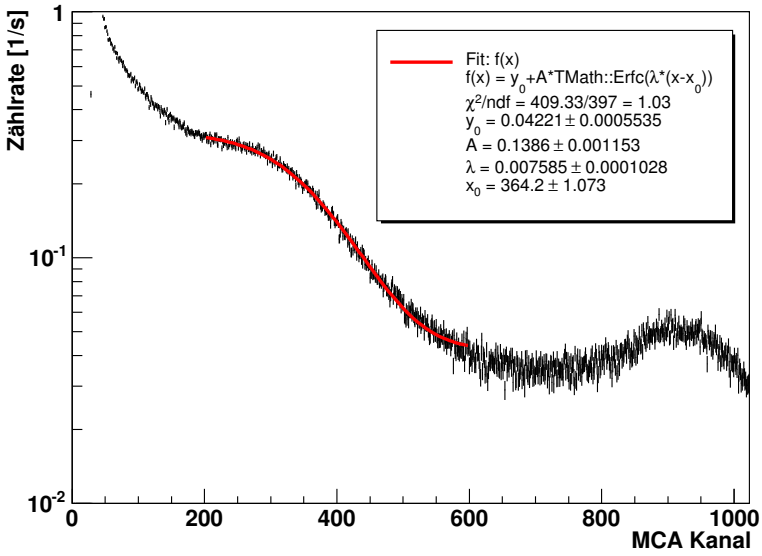

Abbildung 20: Fit der Comptonkante der ^{22}Na-Quelle im Spektrum des Plastikszintillators (erste Aufnahme)

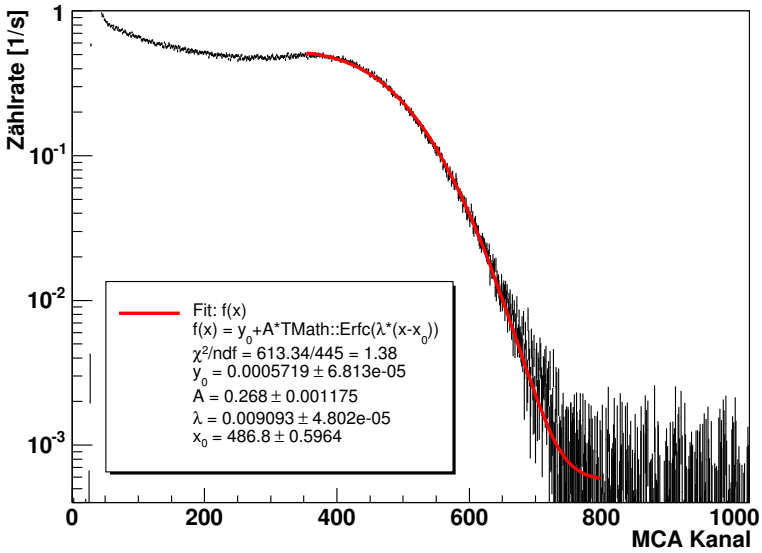

Abbildung 21: Fit der Comptonkante der ^{137}Cs-Quelle im Spektrum des Plastikszintillators (erste Aufnahme)

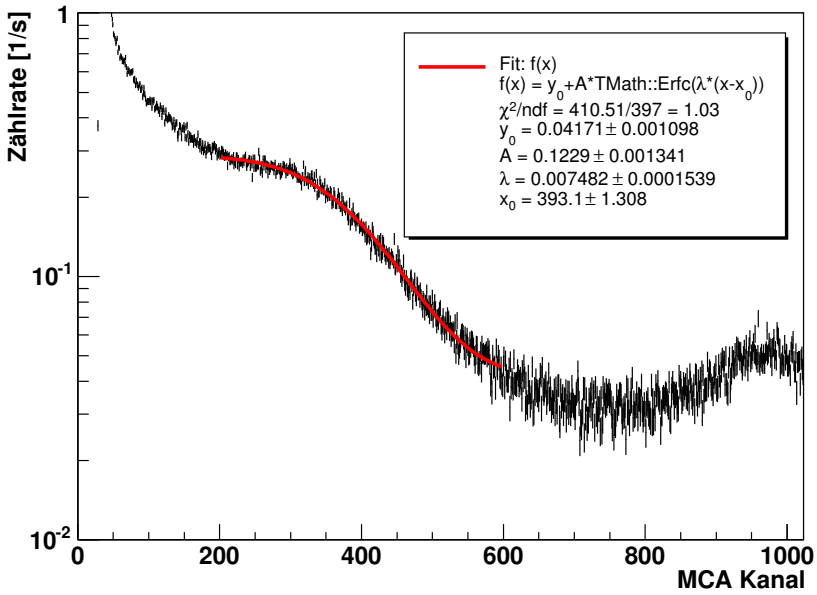

Abbildung 22: Fit der Comptonkante der ^{22}Na-Quelle im Spektrum des Plastikszintillators (zweite Aufnahme)

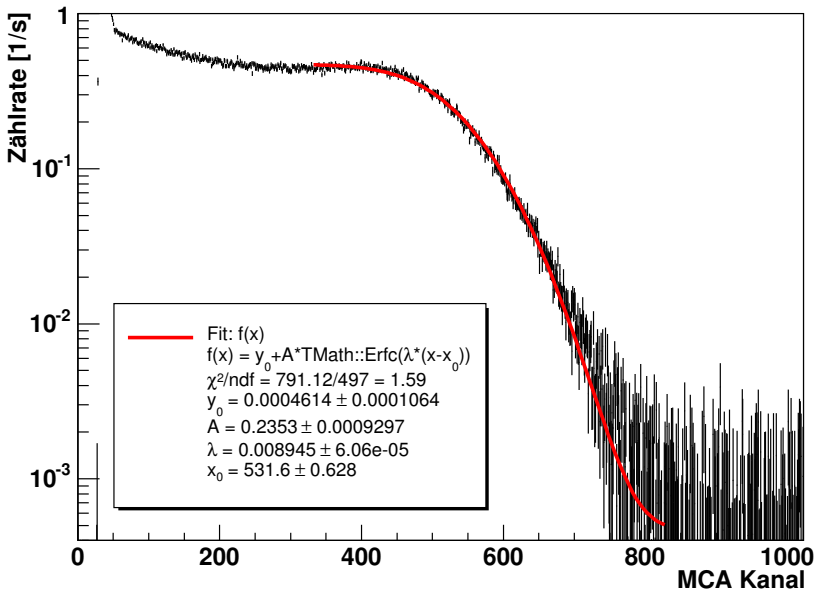

Abbildung 23: Fit der Comptonkante der ^{137}Cs-Quelle im Spektrum des Plastikszintillators (zweite Aufnahme)

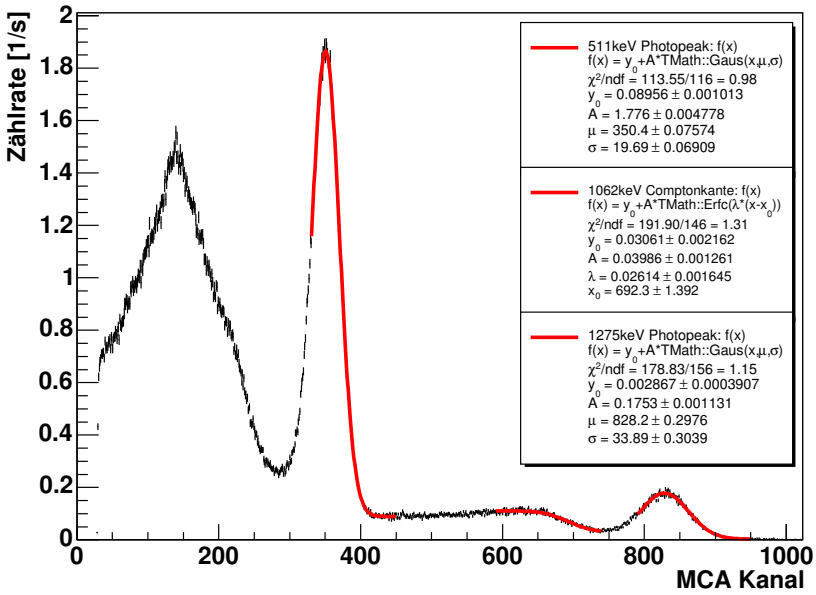

Abbildung 24: Fit der Comptonkanten und der Photopeaks der ^{22}Na-Quelle im Spektrum des NaI-Szintillators (erste Aufnahme)

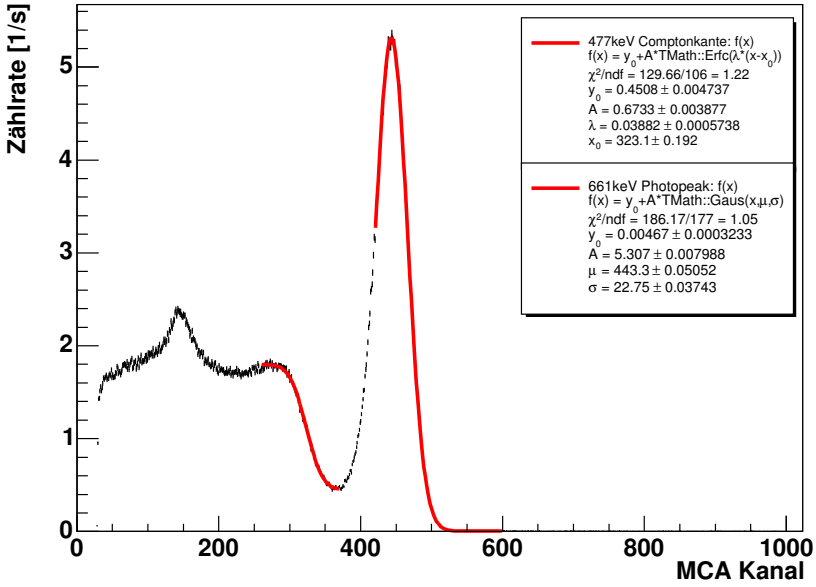

Abbildung 25: Fit der Comptonkante und des Photopeaks der ^{137}Cs-Quelle im Spektrum des NaI-Szintillators (erste Aufnahme)

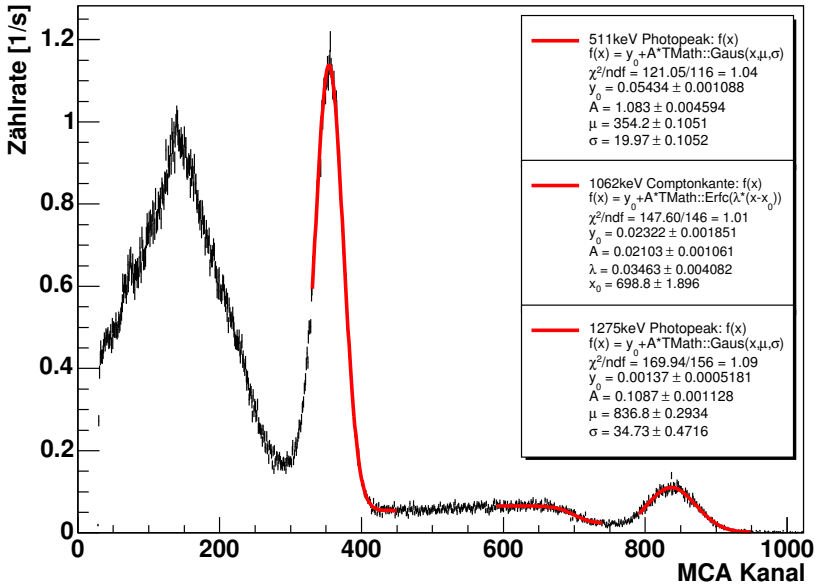

Abbildung 26: Fit der Comptonkanten und der Photopeaks der ^{22}Na-Quelle im Spektrum des NaI-Szintillators (zweite Aufnahme)

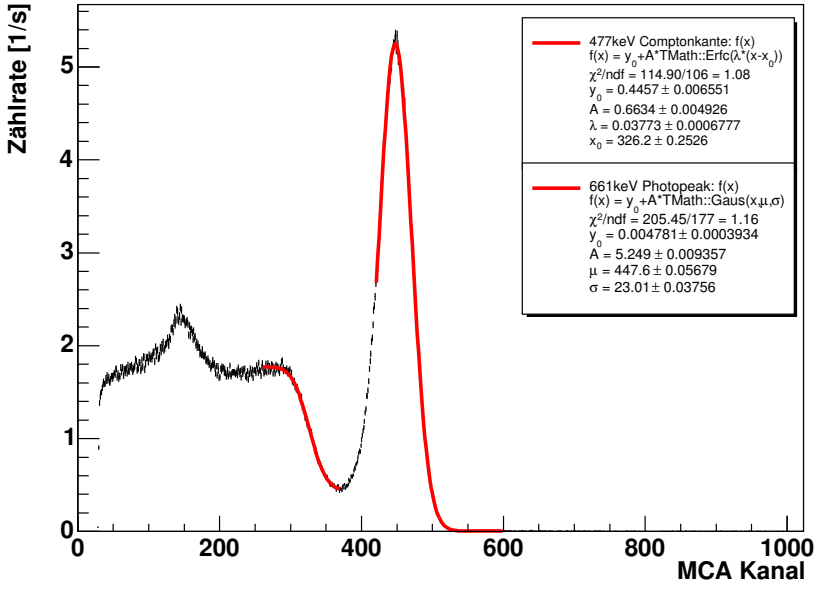

Abbildung 27: Fit der Comptonkante und des Photopeaks der ^{137}Cs-Quelle im Spektrum des NaI-Szintillators (zweite Aufnahme)

A.2. Winkelabhängige Messung

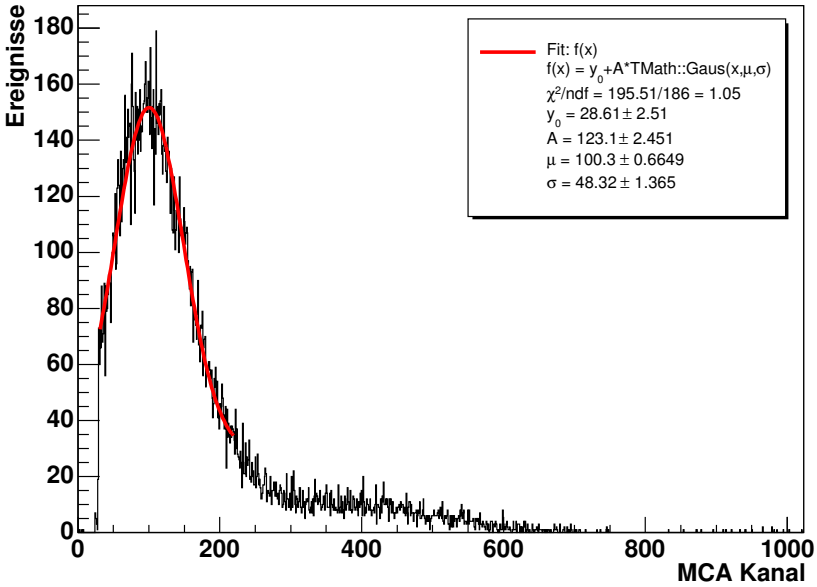

Abbildung 28: Energie der Elektronen bei 30°-Streuung der γ-Quanten

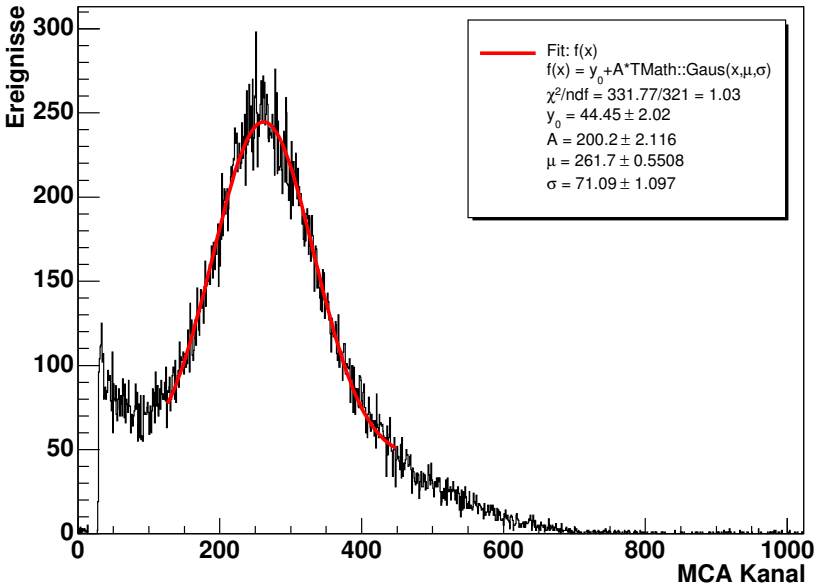

Abbildung 29: Energie der Elektronen bei 60°-Streuung der γ-Quanten

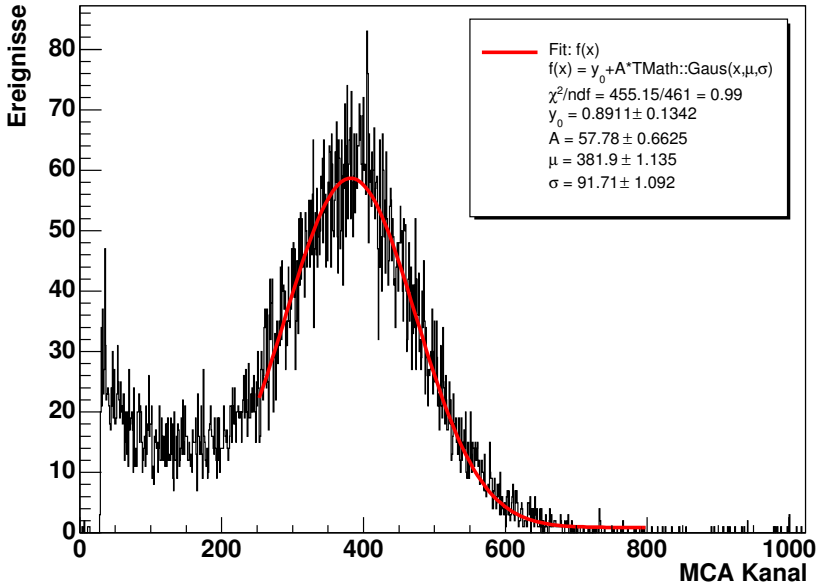

Abbildung 30: Energie der Elektronen bei 90°-Streuung der γ-Quanten

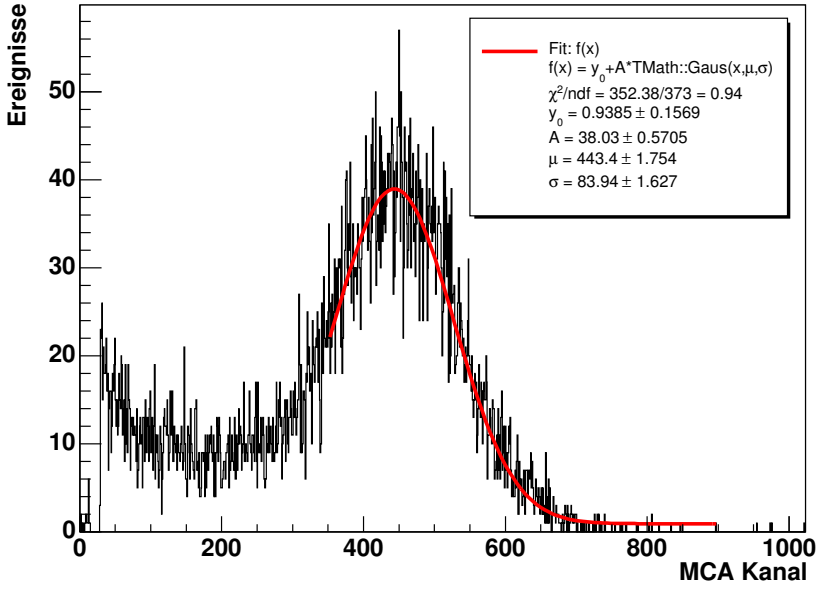

Abbildung 31: Energie der Elektronen bei 120°-Streuung der γ-Quanten

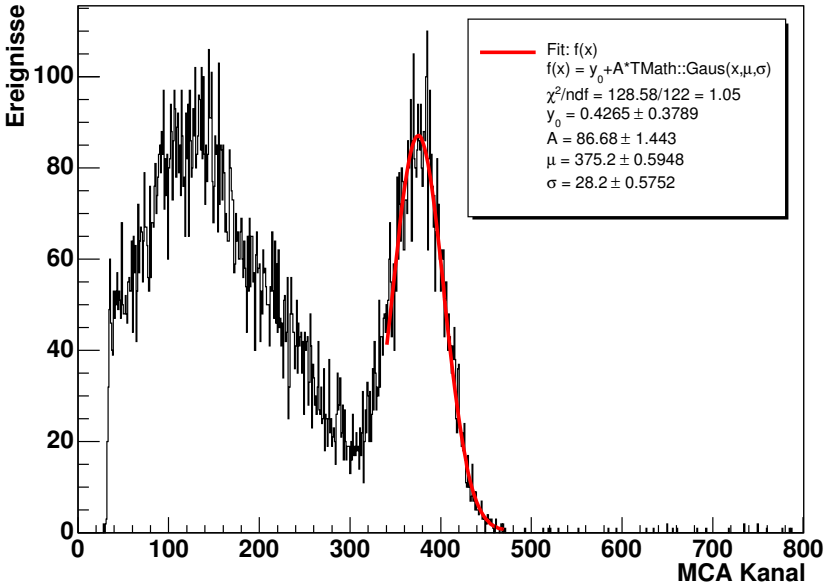

Abbildung 32: Energie der γ-Quanten nach einer Streuung von 30°

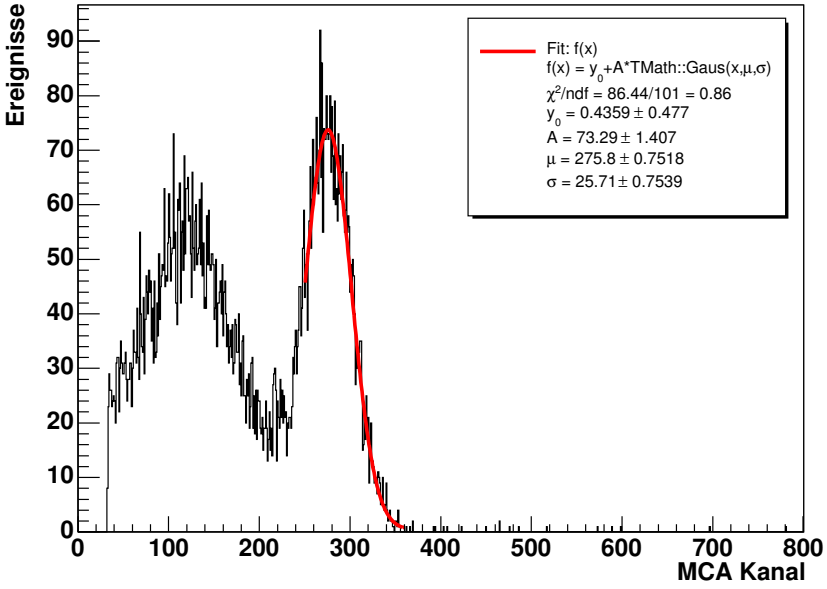

Abbildung 33: Energie der γ-Quanten nach einer Streuung von 60°

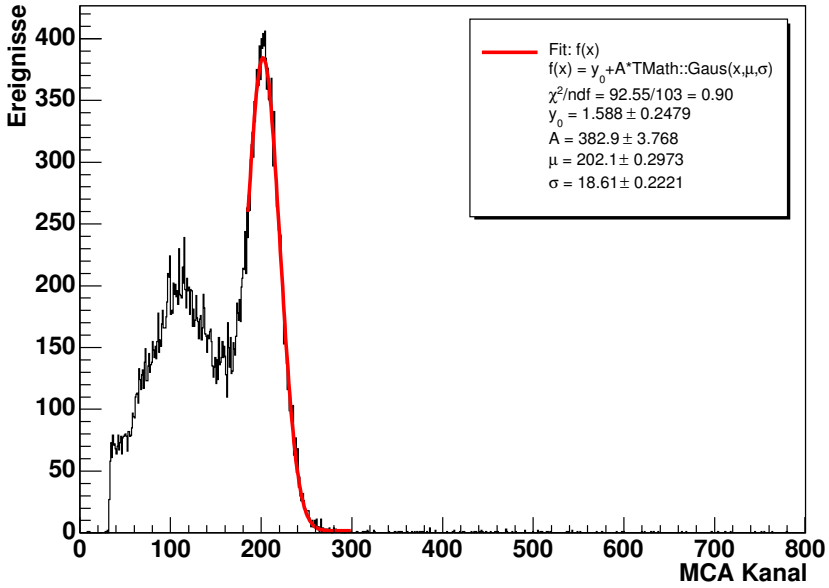

Abbildung 34: Energie der γ-Quanten nach einer Streuung von 90°

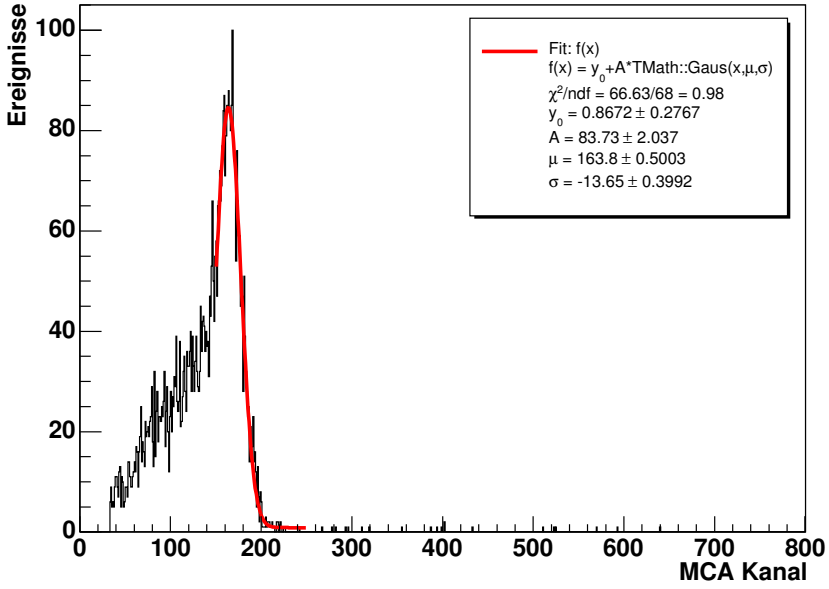

Abbildung 35: Energie der γ-Quanten nach einer Streuung von 120°

B. Quelltexte[7]

Wir verwendeten zur Auswertung die Programmiersprache Python mit dem Datenanalyse Framework ROOT.

B.1. Eichung des Plastikszintillators (`eeich_pla.py`)

```
#!/usr/bin/python
# -*- coding: iso-8859-1 -*-

import shelve
from array import array
from tools import gew_mittel
from mca_messung import Messung
from konst import Q, keV, Ena_anihi_compt, Ecs_compt
from fit_tools import create_fit_legend, print_fit_result
from ROOT import gROOT, TH1F, TCanvas, TF1, TGraphErrors, TMath, TLegend

gROOT.SetStyle('Plain')


class Eichung:
    def __init__(self, name, na_file, na_params, na_range, cs_file,
                 cs_params, cs_range, ug_file = 'ug_pla.asc'):

        # Na-Spektrum
        self.mna = mna = Messung(na_file)
        mna.entferne_untergrund(Messung(ug_file))
        mna.konvertiere_in_rate()
        hna = mna.histo; hna.SetMinimum(1e-2); hna.SetMaximum(1)
        mna.draw(draw_infos=False); mna.canvas.SetLogy()

        # 341 keV Comptonkante
        fstr = '[0]+[1]*TMath::Erfc([2]*(x-[3]))'
        fna = self.fna = TF1('f'+na_file, fstr, na_range[0], na_range[1])
        fna.SetLineColor(2)

        params = [
            (0, 'y_{0}', na_params[0]),
            (1, 'A', na_params[1]),
            (2, '#lambda', na_params[2]),
            (3, 'x_{0}', na_params[3]) ]

        for i, pn, pv in params:
            fna.SetParName(i, pn)
            fna.SetParameter(i, pv)

```

[7] Siehe `http://www.physik.uni-freiburg.de/~kolja/fp2/compton/`

```
        hna.Fit(fna, 'RQ')
        na_peak = (fna.GetParameter(3), fna.GetParError(3))

        print_fit_result(fna, show_fstr=False); print
        lna = self.lna = create_fit_legend(
            fna, lpos = (0.46, 0.56, 0.88, 0.87))
        lna.Draw()
        mna.canvas.Update()

        # Cs-Spektrum
        mcs = self.mcs = Messung(cs_file)
        mcs.entferne_untergrund(Messung(ug_file))
        mcs.konvertiere_in_rate()
        hcs = mcs.histo; hcs.SetMinimum(4e-4); hcs.SetMaximum(1)
        mcs.draw(draw_infos=False); mcs.canvas.SetLogy()

        # 477 keV Comptonkante
        fstr = '[0]+[1]*TMath::Erfc([2]*(x-[3]))'
        fcs = self.fcs = TF1('f'+cs_file, fstr, cs_range[0], cs_range[1])
        fcs.SetLineColor(2)

        params = [
            (0, 'y_{0}', cs_params[0]),
            (1, 'A', cs_params[1]),
            (2, '#lambda', cs_params[2]),
            (3, 'x_{0}', cs_params[3]) ]

        for i, pn, pv in params:
            fcs.SetParName(i, pn)
            fcs.SetParameter(i, pv)

        hcs.Fit(fcs, 'RQ+')
        cs_peak = (fcs.GetParameter(3), fcs.GetParError(3))

        print_fit_result(fcs, show_fstr=False); print
        lcs = self.lcs = create_fit_legend(
            fcs, lpos = (0.14, 0.15, 0.55, 0.46))
        lcs.Draw()
        mcs.canvas.Update()

        # Eichfit
        peaks = [na_peak, cs_peak]
        ch = array('d', [p[0] for p in peaks])
        sch = array('d', [max(1, p[1]) for p in peaks])
        E = array('d', [Ena_anihi_compt/keV, Ecs_compt/keV])
        g = self.graph = TGraphErrors(len(ch), ch, E, sch)

        fe = self.fe = TF1('fe_'+name, '[0]*x + [1]')
        for i, pn in enumerate(['a', 'b']):
            fe.SetParName(i, pn)
            fe.SetParameter(i, 1)

```

```
93          g.Fit(fe, 'Q0+')
94          print_fit_result(fe)
95
96          self.a = (fe.GetParameter(0), fe.GetParError(0))
97          self.b = (fe.GetParameter(1), fe.GetParError(1))
98
99
100 print 'Eichung 1:'
101 e1 = Eichung(
102     'Eichung1',
103     'na_pla.asc',  (0.05, 0.1, 0.01, 350), (200, 600),
104     'cs_pla2.asc', (0.05, 0.2, 0.01, 500), (352, 800) )
105
106 print '\nEichung 2:'
107 e2 = Eichung(
108     'Eichung1',
109     'na_plax.asc', (0.05, 0.1, 0.01, 350), (200, 600),
110     'cs_plax.asc', (0.05, 0.2, 0.01, 500), (330, 830) )
111
112 f1, g1, f2, g2 = e1.fe, e1.graph, e2.fe, e2.graph
113
114
115 print '\nErgebnisse:'
116 print 'a1 = %.3f +- %.3f, b1 = %.3f +- %.3f' % (e1.a + e1.b)
117 print 'a2 = %.3f +- %.3f, b2 = %.3f +- %.3f' % (e2.a + e2.b)
118
119 a = gew_mittel([e1.a, e2.a])
120 b = gew_mittel([e1.b, e2.b])
121 print 'a  = %.3f +- %.3f, b  = %.3f +- %.3f' % (a + b)
122
123
124 c = TCanvas('eeich_pla', 'Energieeichung, Plastik')
125 c.SetGrid()
126
127 h = TH1F('h', ';Kanalnummer;Energie [keV]', 1, 300, 550)
128 h.SetStats(0); h.SetMinimum(220); h.SetMaximum(600)
129 h.Draw()
130
131 f1.SetRange(300, 550)
132 f1.SetLineColor(2); f1.SetLineStyle(2); f1.SetLineWidth(2)
133 f1.Draw('same')
134
135 f2.SetRange(300, 550)
136 f2.SetLineColor(4); f2.SetLineStyle(4); f2.SetLineWidth(2)
137 f2.Draw('same')
138
139 for gi in [g1, g2]:
140     gi.SetMarkerColor(2); gi.SetMarkerStyle(3); gi.SetMarkerSize(.8)
141     gi.Draw('P')
142
143 f = TF1('f', '[0]*x + [1]', 300, 550)
144 f.SetParameter(0, a[0]); f.SetParameter(1, b[0])
```

```
145 f.SetLineWidth(2)
146 f.Draw('same')
147
148 lg = TLegend(.57, .15, .88, .36)
149 lg.SetFillColor(0)
150 lg.SetHeader('')
151 lg.AddEntry(f1, 'Erste Eichung', 'l')
152 lg.AddEntry(f2, 'Zweite Eichung', 'l')
153 lg.AddEntry(f, 'Verwendete Eichung', 'l')
154 lg.AddEntry(f, '', '')
155 lg.Draw()
156
157 # Speichere Daten
158 d = shelve.open('results.out')
159 d['eeich_pla.a'] = (a[0]*keV, a[1]*keV)
160 d['eeich_pla.b'] = (b[0]*keV, b[1]*keV)
161 d.close()
```

B.2. Eichung des NaI-Szintillators (`eeich_nai.py`)

```
1  #!/usr/bin/python
2  # -*- coding: iso-8859-1 -*-
3
4  import shelve
5  from array import array
6  from tools import gew_mittel
7  from mca_messung import Messung
8  from konst import Q, keV, Ena, Ena_compt, Ena_anihi, Ecs, Ecs_compt
9  from fit_tools import create_fit_legend, print_fit_result
10 from ROOT import gROOT, TH1F, TCanvas, TF1, TGraphErrors, TMath, TLegend
11
12 gROOT.SetStyle('Plain')
13
14
15 class Eichung:
16     def __init__(self, name,
17                  na_file,
18                  na_par1, na_rg1,
19                  na_par2, na_rg2,
20                  na_par3, na_rg3,
21                  cs_file,
22                  cs_par1, cs_rg1,
23                  cs_par2, cs_rg2,
24                  ug_file = 'ug_nai.asc'):
25
26         # Parameternamen fuer Gauss- und Erfc-Fits
27         gauss_params = [(0, 'y_{0}'), (1, 'A'), (2, '#mu'), (3, '#sigma')]
28         erfc_params  = [(0, 'y_{0}'), (1, 'A'), (2, '#lambda'), (3, 'x_{0}')]
29
30         # Na-Spektrum
```

```
31          mna = self.mna = Messung(na_file)
32          mna.entferne_untergrund(Messung(ug_file))
33          mna.konvertiere_in_rate()
34          hna = mna.histo; hna.SetMinimum(0)
35          mna.draw(draw_infos=False)
36
37          # 511keV Photopeak
38          fstr = '[0]+[1]*TMath::Gaus(x,[2],[3])'
39          fna1 = self.fna1 = TF1('f1'+na_file, fstr, na_rg1[0], na_rg1[1])
40          fna1.SetLineColor(2)
41
42          for i, pn in gauss_params:
43              fna1.SetParName(i, pn)
44              fna1.SetParameter(i, na_par1[i])
45
46          hna.Fit(fna1, 'RQ')
47          na_peak1 = (fna1.GetParameter(2), fna1.GetParError(2))
48          print_fit_result(fna1, show_fstr=False); print
49          mna.canvas.Update()
50
51          # 1062keV Compton-Kante
52          fstr = '[0]+[1]*TMath::Erfc([2]*(x-[3]))'
53          fna2 = self.fna2= TF1('f2'+na_file, fstr, na_rg2[0], na_rg2[1])
54          fna2.SetLineColor(2)
55
56          for i, pn in erfc_params:
57              fna2.SetParName(i, pn)
58              fna2.SetParameter(i, na_par2[i])
59
60          hna.Fit(fna2, 'QR+')
61          na_peak2 = (fna2.GetParameter(3), fna2.GetParError(3))
62          print_fit_result(fna2, show_fstr=False); print
63          mna.canvas.Update()
64
65          # 1275keV Photopeak
66          fstr = '[0] + [1]*TMath::Gaus(x,[2],[3])'
67          fna3 = self.fna3 = TF1('f3'+na_file, fstr, na_rg3[0], na_rg3[1])
68          fna3.SetLineColor(2)
69
70          for i, pn in gauss_params:
71              fna3.SetParName(i, pn)
72              fna3.SetParameter(i, na_par3[i])
73
74          hna.Fit(fna3, 'QR+')
75          na_peak3 = (fna3.GetParameter(2), fna3.GetParError(2))
76          print_fit_result(fna3, show_fstr=False); print
77          mna.canvas.Update()
78
79          # Erzeuge Legenden
80          lna1 = self.lna1 = create_fit_legend(
81              fna1, fit_name = '511keV Photopeak',
82              lpos = (0.589, 0.655, 0.889, 0.886))
```

```
83          lna1.Draw()
84
85          lna2 = self.lna2 = create_fit_legend(
86              fna2, fit_name = '1062keV Comptonkante',
87              lpos = (0.589, 0.435, 0.889, 0.666))
88          lna2.Draw()
89
90          lna3 = self.lna3 = create_fit_legend(
91              fna3, fit_name = '1275keV Photopeak',
92              lpos = (0.589, 0.215, 0.889, 0.446))
93          lna3.Draw()
94
95
96          # Cs-Spektrum
97          mcs = self.mcs = Messung(cs_file)
98          mcs.entferne_untergrund(Messung(ug_file))
99          mcs.konvertiere_in_rate()
100         hcs = mcs.histo; hcs.SetMinimum(0)
101         mcs.draw(draw_infos=False)
102
103         # 477keV Comptonkante
104         fstr = '[0]+[1]*TMath::Erfc([2]*(x-[3]))'
105         fcs1 = self.fcs1 = TF1('f1'+cs_file, fstr, cs_rg1[0], cs_rg1[1])
106         fcs1.SetLineColor(2)
107
108         for i, pn in erfc_params:
109             fcs1.SetParName(i, pn)
110             fcs1.SetParameter(i, cs_par1[i])
111
112         hcs.Fit(fcs1, 'QR')
113         cs_peak1 = (fcs1.GetParameter(3), fcs1.GetParError(3))
114         print_fit_result(fcs1, show_fstr=False); print
115         mcs.canvas.Update()
116
117         # 661keV Photopeak
118         fstr = '[0]+[1]*TMath::Gaus(x,[2],[3])'
119         fcs2 = self.fcs2 = TF1('f2'+cs_file, fstr, cs_rg2[0], cs_rg2[1])
120         fcs2.SetLineColor(2)
121
122         for i, pn in gauss_params:
123             fcs2.SetParName(i, pn)
124             fcs2.SetParameter(i, cs_par2[i])
125
126         hcs.Fit(fcs2, 'QR+')
127         cs_peak2 = (fcs2.GetParameter(2), fcs2.GetParError(2))
128         print_fit_result(fcs2, show_fstr=False); print
129         mcs.canvas.Update()
130
131         # Erzeuge Legenden
132         lcs1 = self.lcs1 = create_fit_legend(
133             fcs1, 'f(x)', '477keV Comptonkante',
134             lpos = (0.59, 0.65, 0.89, 0.88))
```

```
135         lcs1.Draw()
136
137         lcs2 = self.lcs2 = create_fit_legend(
138             fcs2, 'f(x)', '661keV Photopeak',
139             lpos = (0.59, 0.43, 0.89, 0.66))
140         lcs2.Draw()
141
142         # Eichfit
143         peaks = [na_peak1, na_peak2, na_peak3, cs_peak1, cs_peak2]
144         ch = array('d', [p[0] for p in peaks])
145         sch = array('d', [max(1, p[1]) for p in peaks])
146         E = array('d', [ Ena_anihi/keV, Ena_compt/keV, Ena/keV,
147                          Ecs_compt/keV, Ecs/keV ])
148         g = self.graph = TGraphErrors(len(ch), ch, E, sch)
149
150         fe = self.fe = TF1('fe_'+name, '[0]*x + [1]')
151         for i, pn in enumerate(['a', 'b']):
152             fe.SetParName(i, pn)
153             fe.SetParameter(i, 1)
154
155         g.Fit(fe, 'Q0+')
156         print_fit_result(fe)
157
158         self.a = (fe.GetParameter(0), fe.GetParError(0))
159         self.b = (fe.GetParameter(1), fe.GetParError(1))
160
161
162 # Fuehre Eichung durch
163 print 'Eichung 1:'
164 e1 = Eichung(
165     'Eichung1',
166     'na_nai.asc',
167     (0, 2, 350, 20), (330, 450),
168     (0, 0, 0,  700), (590, 740),
169     (0, 0, 830, 30), (790, 950),
170     'cs_nai.asc',
171     (0, 0, 0,  320), (260, 370),
172     (0, 5, 440, 20), (420, 600) )
173
174 print '\nEichung 2:'
175 e2 = Eichung(
176     'Eichung2',
177     'na_naix.asc',
178     (0, 2, 350, 20), (330, 450),
179     (0, 0, 0,  700), (590, 740),
180     (0, 0, 830, 30), (790, 950),
181     'nai_int.asc',
182     (0, 0, 0,  320), (260, 370),
183     (0, 5, 440, 20), (420, 600) )
184
185 f1, g1, f2, g2 = e1.fe, e1.graph, e2.fe, e2.graph
186
```

```
187
188 print '\nErgebnisse:'
189 print 'a1 = %.3f +- %.3f, b1 = %.3f +- %.3f' % (e1.a + e1.b)
190 print 'a2 = %.3f +- %.3f, b2 = %.3f +- %.3f' % (e2.a + e2.b)
191
192 a = gew_mittel([e1.a, e2.a])
193 b = gew_mittel([e1.b, e2.b])
194 print 'a  = %.3f +- %.3f, b  = %.3f +- %.3f' % (a + b)
195
196
197 # Zeichne die Eichgeraden
198 c = TCanvas('eeich_nai', 'Energieeichung, NaI')
199 c.SetGrid()
200
201 h = TH1F('h', ';Kanalnummer;Energie [keV]', 1, 300, 900)
202 h.SetStats(0); h.SetMinimum(400); h.SetMaximum(1400)
203 h.GetYaxis().SetTitleOffset(1.2)
204 h.Draw()
205
206 f1.SetRange(300, 900)
207 f1.SetLineColor(2); f1.SetLineStyle(2); f1.SetLineWidth(2)
208 f1.Draw('same')
209
210 f2.SetRange(300, 900)
211 f2.SetLineColor(4); f2.SetLineStyle(4); f2.SetLineWidth(2)
212 f2.Draw('same')
213
214 for gi in [g1, g2]:
215     gi.SetMarkerColor(2); gi.SetMarkerStyle(3); gi.SetMarkerSize(.8)
216     gi.Draw('P')
217
218 f = TF1('f', '[0]*x + [1]', 300, 900)
219 f.SetParameter(0, a[0]); f.SetParameter(1, b[0])
220 f.SetLineWidth(2)
221 f.Draw('same')
222
223 lg = TLegend(.57, .15, .88, .36)
224 lg.SetFillColor(0)
225 lg.SetHeader('')
226 lg.AddEntry(f1, 'Erste Eichung', 'l')
227 lg.AddEntry(f2, 'Zweite Eichung', 'l')
228 lg.AddEntry(f, 'Verwendete Eichung', 'l')
229 lg.AddEntry(f, '', '')
230 lg.Draw()
231
232 # Speichere Daten
233 d = shelve.open('results.out')
234 d['eeich_nai.a'] = (a[0]*keV, a[1]*keV)
235 d['eeich_nai.b'] = (b[0]*keV, b[1]*keV)
236 d.close()
```

B.3. Routinen zur Kanal-Energie-Umrechnung (eeich.py)

```
#!/usr/bin/python
# -*- coding: iso-8859-1 -*-

import shelve
from konst import Q, keV

# Lade Energieeichungen
d = shelve.open('results.out')
(nai_a, nai_sa), (nai_b, nai_sb) = d['eeich_nai.a'], d['eeich_nai.b']
(pla_a, pla_sa), (pla_b, pla_sb) = d['eeich_pla.a'], d['eeich_pla.b']
d.close()

def nai_E(ch):
    E = nai_a * ch[0] + nai_b
    sE = ((ch[0]*nai_sa)**2 + (nai_a*ch[1])**2 + nai_sb**2)**0.5
    return E, sE

def pla_E(ch):
    E = pla_a * ch[0] + pla_b
    sE = ((ch[0]*pla_sa)**2 + (pla_a*ch[1])**2 + pla_sb**2)**0.5
    return E, sE
```

B.4. Bestimmung der Rückstreupeaks (rueckstreu.py)

```
#!/usr/bin/python
# -*- coding: iso-8859-1 -*-

import shelve
from eeich import nai_E
from mca_messung import Messung
from konst import Q, keV, me, c, Ena, Ecs
from fit_tools import create_fit_legend, print_fit_result
from ROOT import gROOT, TH1F, TCanvas, TF1, TGraphErrors, TMath, TLegend

gROOT.SetStyle('Plain')

# Lade Untergrund-Messung
mug = Messung('ug_nai.asc')

# Lade Na-Messung und entferne Untergrund
mna = Messung('na_nai.asc')
mna.entferne_untergrund(mug)
mna.konvertiere_in_rate()
hna = mna.histo; hna.SetMinimum(0)
mna.draw(draw_infos=False)

# Gaussfit
fstr = '[0] + [1]*TMath::Gaus(x,[2],[3]) + [4]*x'
```

```
25 fna = TF1('fna', fstr, 70, 220)
26 fna.SetLineColor(2)
27
28 params = [
29     (0, 'y_{0}',    1),
30     (1, 'A',        1),
31     (2, '#mu',    134),
32     (3, '#sigma',  25),
33     (4, 'a',        0) ]
34
35 for i, pn, pv in params:
36     fna.SetParName(i, pn)
37     fna.SetParameter(i, pv)
38
39 print 'Na-Fit:'
40 hna.Fit(fna, 'QR')
41 print_fit_result(fna, show_fstr = False)
42 lna = create_fit_legend(
43     fna, fit_name = 'R#ddot{u}ckstreupeak',
44     lpos = (0.589, 0.655, 0.889, 0.886))
45 lna.Draw()
46
47 # Berechne Energie des Na-Rueckstreupeaks
48 ch_na = (fna.GetParameter(2), fna.GetParError(2))
49 E_na = nai_E(ch_na)
50
51 # Lade Cs-Messung und entferne Untergrund
52 mcs = Messung('cs_nai.asc')
53 mcs.entferne_untergrund(mug)
54 mcs.konvertiere_in_rate()
55 hcs = mcs.histo; hcs.SetMinimum(0)
56 mcs.draw(draw_infos=False)
57
58 # Gaussfit
59 fstr = '[0] + [1]*TMath::Gaus(x,[2],[3]) + [4]*x'
60 fcs = TF1('fcs', fstr, 80, 220)
61 fcs.SetLineColor(2)
62
63 params = [
64     (0, 'y_{0}',    2),
65     (1, 'A',        1),
66     (2, '#mu',    143),
67     (3, '#sigma',  20),
68     (4, 'a',        0) ]
69
70 for i, pn, pv in params:
71     fcs.SetParName(i, pn)
72     fcs.SetParameter(i, pv)
73
74 print '\nCs-Fit:'
75 hcs.Fit(fcs, 'QR')
76 print_fit_result(fcs, show_fstr = False)
```

```
77  lcs = create_fit_legend(
78      fcs, fit_name = 'R#ddot{u}ckstreupeak',
79      lpos = (0.589, 0.655, 0.889, 0.886))
80  lcs.Draw()
81
82  # Berechne Energie des Cs-Rueckstreupeaks
83  ch_cs = (fcs.GetParameter(2), fcs.GetParError(2))
84  E_cs = nai_E(ch_cs)
85
86  # Ergebnisse
87  print '\nRueckstreupeak, Na:'
88  print 'Kanal:   %.3f +- %.3f' % ch_na
89  print 'Energie: %.3f +- %.3f keV' % (E_na[0]/keV, E_na[1]/keV)
90
91  print '\nRueckstreupeak, Cs:'
92  print 'Kanal:   %.3f +- %.3f' % ch_cs
93  print 'Energie: %.3f +- %.3f keV' % (E_cs[0]/keV, E_cs[1]/keV)
94
95  # Speichere Ergebnisse
96  d = shelve.open('results.out')
97  d['rueckstreu.ch_na'] = ch_na
98  d['rueckstreu.E_na'] = E_na
99  d['rueckstreu.ch_cs'] = ch_cs
100 d['rueckstreu.E_cs'] = E_cs
101 d.close()
```

B.5. Streuenergien der Photonen (gstreu.py)

```
1   #!/usr/bin/python
2   # -*- coding: iso-8859-1 -*-
3
4   import shelve
5   from eeich import nai_E
6   from mca_messung import Messung
7   from konst import Q, keV
8   from fit_tools import print_fit_result, create_fit_legend
9   from ROOT import gROOT, TGaxis, TH1F, TCanvas, TF1, TMath
10
11  gROOT.SetStyle('Plain')
12  TGaxis().SetMaxDigits(4)
13
14  # Fitfunktion und Parameternamen
15  fstr = '[0]+[1]*TMath::Gaus(x,[2],[3])'
16  pnames = [ (0, 'y_{0}'), (1, 'A'), (2, '#mu'), (3, '#sigma') ]
17
18  # Anfangswerte und Fitbereiche fuer verschiedene Winkel
19  mess = [
20  #   (ang, (y0, A, mu, sigma), (r1, r2))
21      ( 30, (0, 100, 400, 30), (340, 470)),
22      ( 60, (0, 100, 280, 30), (250, 360)),
```

```
23      ( 90, (0, 100, 200, 30), (185, 300)),
24      (120, (0, 100, 170, 30), (150, 250)) ]
25
26  # Fuehre Gausfits durch
27  m = []
28  for angle, pvals, r in mess:
29      print '\n%d° Messung:' % angle
30
31      mi = Messung('gs_%d.asc' % angle)
32      mi.angle = angle
33      mi.histo.SetAxisRange(0,800)
34      mi.draw(); m += [mi]
35
36      f = mi.f = TF1('f_%d' % angle, fstr, r[0], r[1])
37      f.SetLineColor(2)
38
39      for (i, pni), pvi in zip(pnames, pvals):
40          f.SetParName(i, pni)
41          f.SetParameter(i, pvi)
42
43      mi.histo.Fit(f, 'QR')
44      print_fit_result(f)
45      mi.lg = create_fit_legend(f, lpos=(.53,.58,.88,.88))
46      mi.lg.Draw()
47
48      mi.ch = (f.GetParameter(2), f.GetParError(2))
49      mi.E = nai_E(mi.ch)
50
51      mi.canvas.Update()
52
53
54  print '\nErgebnisse:'
55  for mi in m:
56      ch, sch = mi.ch
57      E, sE = mi.E[0]/keV, mi.E[1]/keV
58      print '%3d: ch = %.3f +- %.3f (%.2f%%), E = %.3f +- %.3f keV (%.2f%%)' % (
59          mi.angle, ch, sch, sch/ch*100, E, sE, sE/E*100)
60
61  # Speichere Ergebnisse
62  ch, E = {}, {}
63  for mi in m:
64      ch[mi.angle] = mi.ch
65      E[mi.angle] = mi.E
66
67  d = shelve.open('results.out')
68  d['gstreu.ch'] = ch
69  d['gstreu.E'] = E
70  d.close()
```

B.6. Streuenergien der Elektronen (estreu.py)

```
#!/usr/bin/python
# -*- coding: iso-8859-1 -*-

import shelve
from eeich import pla_E
from mca_messung import Messung
from konst import Q, keV
from fit_tools import print_fit_result, create_fit_legend
from ROOT import gROOT, TGaxis, TH1F, TCanvas, TF1, TMath

gROOT.SetStyle('Plain')
TGaxis().SetMaxDigits(4)

# Fitfunktion und Parameternamen
fstr = '[0]+[1]*TMath::Gaus(x,[2],[3])'
pnames = [ (0, 'y_{0}'), (1, 'A'), (2, '#mu'), (3, '#sigma') ]

# Anfangswerte und Fitbereiche fuer verschiedene Winkel
mess = [
#   (ang, (y0, A,  mu, sigma), (r1, r2))
    ( 30, (1, 100, 100, 50), ( 30, 220)),
    ( 60, (1, 200, 300, 70), (125, 450)),
    ( 90, (1, 200, 400, 70), (250, 800)),
    (120, (1, 400, 440, 80), (350, 900)) ]

# Fuehre Gausfits durch
m = []
for angle, pvals, r in mess:
    print '\n%d° Messung:' % angle

    mi = Messung('es_%d.asc' % angle)
    mi.angle = angle
    mi.draw(); m += [mi]

    f = mi.f = TF1('f_%d' % angle, fstr, r[0], r[1])
    f.SetLineColor(2)

    for (i, pni), pvi in zip(pnames, pvals):
        f.SetParName(i, pni)
        f.SetParameter(i, pvi)

    mi.histo.Fit(f, 'QR')
    print_fit_result(f)
    mi.lg = create_fit_legend(f, lpos=(.53,.58,.88,.88))
    mi.lg.Draw()

    mi.ch = (f.GetParameter(2), f.GetParError(2))
    mi.E = pla_E(mi.ch)

```

```
50      mi.canvas.Update()
51
52  print '\nErgebnisse:'
53  for mi in m:
54      ch, sch = mi.ch
55      E, sE = mi.E[0]/keV, mi.E[1]/keV
56      print '%3d: ch = %.3f +- %.3f (%.2f%%), E = %.3f +- %.3f keV (%.2f%%)' % (
57          mi.angle, ch, sch, sch/ch*100, E, sE, sE/E*100)
58
59  # Speichere Ergebnisse
60  ch, E = {}, {}
61  for mi in m:
62      ch[mi.angle] = mi.ch
63      E[mi.angle] = mi.E
64
65  d = shelve.open('results.out')
66  d['estreu.ch'] = ch
67  d['estreu.E'] = E
68  d.close()
```

B.7. Vergleich der Streuenergien mit Theorie (streu_energie.py)

```
1   #!/usr/bin/python
2   # -*- coding: iso-8859-1 -*-
3
4   import shelve
5   from array import array
6   from math import pi, sin, cos
7   from konst import Q, c, me, keV, Ecs
8   from ROOT import gROOT, TF1, TCanvas, TH1F, TLegend, TLatex, TGraphErrors
9
10  gROOT.SetStyle('Plain')
11
12  # Lade gemessene Streuenergien
13  d = shelve.open('results.out')
14  Eg = d['gstreu.E'].items(); Eg.sort()
15  Ee = d['estreu.E'].items(); Ee.sort()
16  d.close()
17
18  # Berechne Summe der gemessenen Energien
19  Ege = []
20  for (agi, (Egi,sEgi)), (aei, (Eei,sEei)) in zip(Eg, Ee):
21      assert agi == aei
22      Ei = Egi + Eei
23      sEi = ( sEgi**2 + sEei**2 )**0.5
24      Ege += [(agi, (Ei, sEi))]
25
26  # Fehler des Winkel
27  sa = 2.0
28
```

```
29 # Koeffizient zur Vereinfachung der Formel
30 alpha = Ecs / (me * c**2)
31
32 # Berechne theoretische Werte (beruecksichtige Winkelfehler)
33 Eg_th = []
34 for a, (E, sE) in Eg:
35     a_rad, sa_rad = a*pi/180, sa*pi/180
36     E_th = Ecs / (1 + alpha*(1-cos(a*pi/180)))
37     sE_th = abs(alpha*Ecs*sin(a_rad)/(1+alpha*(1-cos(a_rad)))**2)*sa_rad
38     Eg_th += [(E_th, sE_th)]
39
40 Ee_th = []
41 for a, (E, sE) in Ee:
42     a_rad, sa_rad = a*pi/180, sa*pi/180
43     E_th = Ecs / (1 + 1./(alpha*(1-cos(a_rad))))
44     sE_th = abs(alpha*Ecs*sin(a_rad)/(1+alpha*(1-cos(a_rad)))**2)*sa_rad
45     Ee_th += [(E_th, sE_th)]
46
47 # Erzeuge Graphen mit den Messwerten der jeweiligen Messung
48 gg = TGraphErrors(
49     len(Eg),
50     array('d', [z[0] for z in Eg]),
51     array('d', [z[1][0]/keV for z in Eg]),
52     array('d', [sa]*len(Eg)),
53     array('d', [z[1][1]/keV for z in Eg]))
54
55 ge = TGraphErrors(
56     len(Ee),
57     array('d', [z[0] for z in Ee]),
58     array('d', [z[1][0]/keV for z in Ee]),
59     array('d', [sa]*len(Ee)),
60     array('d', [z[1][1]/keV for z in Ee]))
61
62 gge = TGraphErrors(
63     len(Ege),
64     array('d', [z[0] for z in Ege]),
65     array('d', [z[1][0]/keV for z in Ege]),
66     array('d', [sa]*len(Ege)),
67     array('d', [z[1][1]/keV for z in Ege]))
68
69
70 # Erstelle Funktionen fuer den Theoretischen Verlauf
71 params = [ (0, 'E_{#gamma}', Ecs/keV),
72            (1, '#alpha',     alpha) ]
73
74 fg_str = '[0] / (1 + [1]*(1 - cos(x/180*pi)))'
75 fe_str = '[0] / (1 + 1/([1]*(1 - cos(x/180*pi))))'
76
77 f  = TF1('f', '%s + %s' % (fg_str, fe_str), 0, 130)
78 fg = TF1('fg', fg_str, 0, 130)
79 fe = TF1('fe', fe_str, 0, 130)
80
```

```
81  for fj in [fg, fe, f]:
82      for i, pn, pv in params:
83          fj.SetParName(i, pn)
84          fj.SetParameter(i, pv)
85
86  # Erzeuge Zeichenflaeche
87  canvas = TCanvas('c_streu_energie', 'Streu-Energien')
88  h = TH1F('h', ';Streuwinkel #vartheta [#circ];Energie [keV]', 1, 0, 130)
89  h.SetMinimum(0)
90  h.SetMaximum(Ecs/keV+20)
91  h.SetStats(0)
92  h.Draw()
93
94  # Zeichne theoretische Verlaeufe
95  fg.SetLineColor(2); fg.SetLineStyle(2)
96  fe.SetLineColor(4); fe.SetLineStyle(4)
97  f.SetLineColor(8); f.SetLineStyle(1)
98  for fi in [fg, fe, f]:
99      fi.SetLineWidth(2)
100     fi.Draw('same')
101
102 # Zeichne Graphen mit Messdaten
103 for g,mcolor,mstyle,msize in [(gg,2,21,.4), (ge,4,21,.4), (gge,8,21,.4)]:
104     g.SetMarkerColor(mcolor)
105     g.SetMarkerStyle(mstyle)
106     g.SetMarkerSize(msize)
107     g.Draw('Psame')
108
109 # Zeichne Legende
110 l = TLegend(0.51, 0.13, 0.85, 0.32)
111 l.SetFillColor(0)
112 l.AddEntry(gge, 'Summe der gemessenen Energien', 'p')
113 l.AddEntry(gg, 'Gemessene Photonenenergien', 'p')
114 l.AddEntry(ge, 'Gemessene Elektronenenergien', 'p')
115 l.AddEntry(f, 'Theoretische Summe der Energien', 'l')
116 l.AddEntry(fg, 'Theoretische Photonenenergie', 'l')
117 l.AddEntry(fe, 'Theoretische Elektronenergie', 'l')
118 l.Draw()
119
120 # Theoretische Werte und Messwerte ausgeben
121 print 'Elektronen:'
122 for (a, (E, sE)), (E_th, sE_th) in zip(Ee, Ee_th):
123     print '%3d°: mess = %7.3f +- %.3f, th = %7.3f +- %6.3f, diff = %.2f' % (
124         a, E/keV, sE/keV, E_th/keV, sE_th/keV, fe.Eval(a)-E/keV)
125
126 print '\nPhotonen:'
127 for (a, (E, sE)), (E_th, sE_th) in zip(Eg, Eg_th):
128     print '%3d°: mess = %7.3f +- %.3f, th = %7.3f +- %6.3f, diff = %.2f' % (
129         a, E/keV, sE/keV, E_th/keV, sE_th/keV, fg.Eval(a)-E/keV)
130
131 print '\nSumme:'
132 for a, (E, sE) in Ege:
```

```
133     print '%3d°: mess = %7.3f +- %.3f, th = %7.3f, diff = %.2f' % (
134         a, E/keV, sE/keV, f.Eval(a), f.Eval(a)-E/keV)
135
136 print_tables = False
137 if print_tables:
138     from texgen import table_streu_energie
139     print table_streu_energie(Eg, Eg_th, Ee, Ee_th, sa)
```

B.8. Differentieller Wirkungsquerschnitt (diff_cross_sect.py)

```
1  #!/usr/bin/python
2  # -*- coding: iso-8859-1 -*-
3
4  from math import exp, sqrt, pi, cos, tan
5  from konst import Q, Ecs, me, c, re
6  from fit_tools import create_fit_legend, print_fit_result
7  from mca_messung import Messung
8  from array import array
9  from ROOT import gROOT, TCanvas, TGraphErrors, TF1, TLegend, TH1F
10
11 gROOT.SetStyle("Plain")
12
13
14 # Erzeuge Zeichenflaeche
15 canvas = TCanvas('c_diff_cross_sect', 'Differentieller Wirkungsquerschnitt')
16 h = TH1F('h', ';Streuwinkel #vartheta [#circ];diff. Wirkungsquerschnitt (d#sigma/d#Omega)_{#vart
17 h.SetMinimum(0)
18 h.SetMaximum(1e-29)
19 h.SetStats(0)
20 h.Draw()
21
22 # Zeichne theoretischen Verlauf nach Klein-Nishina
23 fstr = '0.5*[0]^2 * ( (1+[1]*(1-cos(x*pi/180)))^(-3) * (-[1]*(cos(x*pi/180))^3'
24 fstr += '+ ([1]^2+[1]+1)*(1+(cos(x*pi/180))^2) - [1]*(2*[1]+1)*cos(x*pi/180)))'
25 f = TF1('f', fstr, 0, 130)
26 params = [ (0, 'r_{0}', re.inUnitsOf('m')/Q('1m')),
27            (1, '#alpha', Ecs/(me*c**2)) ]
28 for i, pn, pv in params:
29     f.SetParName(i, pn); f.SetParameter(i, pv)
30 f.SetLineColor(4)
31 f.Draw('same')
32
33 # Absorptionskoeffizienten des Plastik-Szintis
34 mu = {
35     0: Q(0.089, '1/cm'),  # 662keV
36    30: Q(0.093, '1/cm'),  # 564keV
37    60: Q(0.108, '1/cm'),  # 402keV
38    90: Q(0.121, '1/cm'),  # 288keV
39   120: Q(0.132, '1/cm'),  # 225keV
40     }
```

```
41
42 # Dicke des Plastik-Szintis
43 d = Q('1 cm')
44
45 # Effizienzen des NaI-Szintis fuer verschiedene Energien (Streuwinkel)
46 nai_eff = {
47     0: 0.427,  # 662keV
48    30: 0.465,  # 564keV
49    60: 0.564,  # 402keV
50    90: 0.691,  # 288keV
51   120: 0.797   # 225keV
52     }
53 snai_eff = 0.02
54
55 # Abstand von Plastik- und NaI-Szinti
56 r, sr = Q(13.4, 'cm'), Q(1, 'cm')
57
58 # Fläche des NaI-Szintis
59 A = Q(pi * (3.8/2)**2, 'cm**2')
60
61 # Bestrahlter Raumwinkel
62 dW = A/r**2
63 sdW = 2*dW*sr/r
64
65 # Dichte der Streuzentren (Elektronen)
66 rho = Q(3.4e23, 'cm**-3')
67
68 # Rate am NaI-Szinti bei 0°
69 m_int = Messung('nai_int.asc')
70 R_int = Q(m_int.count/m_int.time, '1/s')
71 sR_int = Q(sqrt(m_int.count)/m_int.time, '1/s')
72
73 # Untergrundrate des NaI-Szintis
74 m_ug = Messung('ug_nai.asc')
75 R_ug = Q(m_ug.count/m_ug.time, '1/s')
76 sR_ug = Q(sqrt(m_ug.count)/m_ug.time, '1/s')
77
78 # entferne Untergrund
79 R_int2 = R_int - R_ug
80 sR_int2 = ( sR_int**2 + sR_ug**2 )**0.5
81
82 # korrigierte Rate des NaI-Szintis
83 R0 = R_int2 * exp(mu[0]*d) / nai_eff[0]
84 sR0 = R0 * ( (sR_int2/R_int2)**2 + (snai_eff/nai_eff[0])**2 )**0.5
85
86 # untersuchte Winkel
87 theta = [ 30., 60., 90., 120. ]
88 stheta = 2.
89
90 # der durch den Plastik-Szinti zurueckgelegte Weg
91 x = [d / cos(ti*pi/180./2.) for ti in theta]
92 sx = [0.5*xi * tan(ti*pi/180./2.) * stheta*pi/180. for xi,ti in zip(x,theta)]
```

```
93
94  # im NaI-Szinti gemessene Raten bei verschiedenen Winkeln
95  m_mes = [Messung('gs_%d.asc' % ti) for ti in theta]
96  R_mes = [Q(mi.count/mi.time, '1/s') for mi in m_mes]
97  sR_mes = [Q(sqrt(mi.count)/mi.time, '1/s') for mi in m_mes]
98
99  # Untergund bei Koinzidenz im NaI-Szinti
100 m_ugco = Messung('ugco_nai.asc')
101 R_ugco = Q(m_ugco.count/m_ugco.time, '1/s')
102 sR_ugco = Q(sqrt(m_ugco.count)/m_ugco.time, '1/s')
103
104 # Beruecksichtigung zufaelliger Koinzidenzen
105 m_zuf = Messung('zuf_co.asc')
106 R_zuf = Q(m_zuf.count/m_zuf.time, '1/s')
107 sR_zuf = Q(sqrt(m_zuf.count)/m_zuf.time, '1/s')
108
109 # Korrigiere gemessene Raten
110 #R_mes2 = [Ri-R_zuf for Ri in R_mes]
111 #sR_mes2 = [(sRi**2 + sR_zuf**2)**0.5 for sRi in sR_mes]
112 R_mes2 = [Ri - R_zuf - R_ugco for Ri in R_mes]
113 sR_mes2 = [(sRi**2 + sR_zuf**2 + sR_ugco**2)**0.5 for sRi in sR_mes]
114
115 # Beruecksichtige Effizienz und Absorptionskoeffizienten
116 R_streu = [Ri/nai_eff[ti] * exp(mu[ti]*xi/2.)
117            for ti,Ri,xi in zip(theta, R_mes2, x)]
118 sR_streu = [
119     R_streu[i] * (
120     (sR_mes2[i] / R_mes2[i])**2 +
121     (snai_eff / nai_eff[ti])**2 +
122     (0.5 * mu[ti] * sx[i])**2 )**0.5
123     for i, ti in enumerate(theta)]
124
125 # Berechne experimentellen differentiellen Wirkungsquerschnitt
126 dcs = [(Ri/R0 /(rho*xi*dW)).inUnitsOf('m**2') for Ri,xi in zip(R_streu,x)]
127 sdcs = [
128     dcsi * (
129     (sR_streu[i]/R_streu[i])**2 + (sR0/R0)**2 +
130     (sx[i]/x[i])**2 + (sdW/dW)**2 )**0.5
131     for i, dcsi in enumerate(dcs)]
132
133 # Zeichne experimentelle Werte
134 y = array('d', [dcsi / Q('1 m**2') for dcsi in dcs])
135 sy = array('d', [sdcsi / Q('1 m**2') for sdcsi in sdcs])
136 graph = TGraphErrors(len(y), array('d', theta), y,
137                      array('d', [stheta]*len(theta)), sy)
138 graph.SetMarkerStyle(3); graph.SetMarkerColor(2)
139 graph.Draw('Psame')
140
141 # Zeichne Legende
142 l = TLegend(0.38, 0.73, 0.88, 0.87)
143 l.SetFillColor(0)
144 l.AddEntry(graph, 'Experimentell bestimmte Werte', 'p')
```

```
145 l.AddEntry(f, 'Theoretisch berechnet mit Klein-Nishina', 'l')
146 l.Draw()
147
148 # Gebe Ergebnisse aus
149 for i, ti in enumerate(theta):
150     print '%3d: %.7g +- %.7g m^2, th = %.7g' % (
151         ti, dcs[i]/Q('1m**2'), sdcs[i]/Q('1m**2'), f.Eval(ti))
```

B.9. Routinen für MCA-Messungen (mca_messung.py)

```
1  #!/usr/bin/python
2  # -*- coding: iso-8859-1 -*-
3
4  '''
5  Klassen zur Handhabung der MCA Messungen
6  '''
7
8  from math import sqrt
9  from array import array
10 from ROOT import TH1F, TCanvas, TLegend, TPaveText, TF1, TGraphErrors
11
12 class Messung:
13     '''
14     Klasse zum Erstellen von Histogrammen aus den MCA Messdaten;
15     mit Methoden fuer Gaussfits und Doppelgaussfits.
16     '''
17     def __init__(self, name):
18         '''
19         Konstruktor.
20         name: Dateiname der Messung
21         '''
22         self.name = name
23
24         i, header = 1, True
25         h = TH1F(name, name, 1024, 0, 1023)
26         h.SetTitle(';MCA Kanal;Ereignisse')
27         #h.SetTitleOffset(1.3, 'y')
28         h.SetStats(False)
29
30         for line in open('messdaten/' + name, 'r'):
31             if not header:
32                 tokens = line.split(',')
33                 h.SetBinContent(i, int(tokens[1]))
34                 i += 1
35             elif line.strip()[0:3] == 'Chn':
36                 header = False
37             else:
38                 tokens = line.split()
39                 if len(tokens) > 1 and tokens[-2] == 'Time:':
40                     self.time = int(tokens[-1])
```

```
41
42          h.SetEntries(h.GetSum())
43          self.histo = h
44          self.bins = h.GetNbinsX()
45          self.count = h.GetSum()
46
47      def draw(self, draw_grid = False, draw_infos = True):
48          '''
49          Zeichnet das Histogramm.
50          draw_grid:  Zeige Raster an
51          draw_infos: Zeichne Informationen zur Messung
52          '''
53          self.canvas = c = TCanvas('c' + self.name, self.name)
54          if draw_grid: c.SetGrid()
55
56          self.histo.Draw()
57
58          if draw_infos:
59              self.infos = pt = TPaveText(0.58, 0.73, 0.88, 0.87, 'NDC')
60              pt.SetFillColor(0)
61              pt.SetTextAlign(12)
62              pt.AddText('Messreihe: '+self.name)
63              pt.AddText('Messdauer: %d sec' % self.time)
64              pt.AddText('Ereignisse: %d' % self.histo.GetEntries())
65              pt.Draw()
66
67          c.Update()
68
69      def entferne_untergrund(self, m):
70          assert self.bins == m.bins
71          hs, hm = self.histo, m.histo
72
73          for i in range(1,self.bins+1):
74              # Lese Bin der Messung und des Untergrunds aus
75              s = hs.GetBinContent(i)
76              ss = hs.GetBinError(i)
77
78              # Rechne den Untergrund auf die Zeit der Hauptmessung um
79              u = hm.GetBinContent(i) * float(self.time) / float(m.time)
80              su = hm.GetBinError(i) * float(self.time) / float(m.time)
81
82              # Ziehe die Untergrundereignisse ab und berücksichtige die Fehler
83              v = s - u
84              sv = sqrt(ss**2 + su**2)
85
86              # Schreibe das Ergebnis in das aktuelle Bin
87              hs.SetBinContent(i, v)
88              hs.SetBinError(i, sv)
89
90          # Korrigiere die Zahl der Gesamtereignisse
91          self.count = self.histo.GetSum()
92          self.histo.SetEntries(self.count)
```

```
93
94
95      def konvertiere_in_rate(self):
96          h = self.histo
97          for i in range(1,self.bins+1):
98              h.SetBinContent(i, h.GetBinContent(i) / float(self.time))
99              h.SetBinError(i, h.GetBinError(i) / float(self.time))
100         self.time = 1
101         self.count = h.GetSum()
102         h.GetYaxis().SetTitle('Z#ddot{a}hlrate [1/s]')
```

B.10. Hilfsroutinen für Fits (`fit_tools.py`)

```
1   #!/usr/bin/python
2   # -*- coding: iso-8859-1 -*-
3
4   '''
5   Fit-Hilfsroutinen
6   '''
7
8   from ROOT import TLegend, TF1
9
10
11  def create_fit_legend(f, fcn_name = 'f(x)', fit_name = 'Fit',
12                        show_fstr = True, fstr_nl = True, fstr = [],
13                        lpos = (0.59, 0.65, 0.89, 0.88)):
14      '''
15      Erzeugt eine Legende fuer die uebergebene Fitfunktion.
16      f:         Die Fitfunktion
17      fcn_name:  Bezeichnung der Fitfunktion
18      fit_name:  Bezeichnung des Fits
19      show_fstr: Fitfunktion mit ausgeben
20      fstr_nle:  Fitfunktion in neuer Zeile ausgeben
21      fstr:      Liste zum Ueberschreiben der Funktionsformeldarstellung
22      lpos:      Die Position der Legende
23      '''
24      lg = TLegend(lpos[0], lpos[1], lpos[2], lpos[3])
25      lg.SetFillColor(0)
26      lg.SetHeader('')
27
28      par_count = f.GetNpar()
29      if show_fstr and not fstr:
30          ft = f.GetTitle()
31          for i in range(par_count):
32              ft = ft.replace('[%d]' % i, f.GetParName(i))
33          if fstr_nl:
34              lg.AddEntry(f, '%s: %s' % (fit_name, fcn_name), 'l')
35              lg.AddEntry(f, '%s = %s' % (fcn_name, ft), '')
36          else:
37              lg.AddEntry(f, '%s: %s = %s' % (fit_name, fcn_name, ft), 'l')
```

```
38      else:
39          lg.AddEntry(f, '%s: %s' % (fit_name, fcn_name), 'l')
40
41      if fstr:
42          for fstri in fstr:
43              s = fstri
44              for i in range(par_count):
45                  s = s.replace('[%d]' % i, f.GetParName(i))
46              lg.AddEntry(f, s, '')
47
48      ndf = f.GetNDF()
49      if ndf > 0:
50          chisq = f.GetChisquare()
51          lg.AddEntry(f, '#chi^{2}/ndf = %.2f/%d = %.2f' % (
52              chisq, ndf, chisq/ndf), '')
53
54      for i in range(par_count):
55          pn, pv, pe = f.GetParName(i), f.GetParameter(i), f.GetParError(i)
56          lg.AddEntry(f, '%s = %.4g #pm %.4g' % (pn, pv, pe), '')
57
58      lg.AddEntry(f, '', '')
59
60      return lg
61
62
63  def print_fit_result(f, fcn_name = 'f(x)', show_fstr = True):
64      '''
65      Gibt ein huebscheres Fit-Ergebnis aus.
66      f:         Die Fitfunktion
67      fcn_name:  Bezeichnung der Fitfunktion
68      show_fstr: Fitfunktion mit ausgeben
69      '''
70      par_count = f.GetNpar()
71
72      pnames = []
73      max_pname_len = 0
74      for i in range(par_count):
75          pn = f.GetParName(i)
76          for c in '#_{}':
77              pn = pn.replace(c, '')
78          pnames += [pn]
79          max_pname_len = max(max_pname_len, len(pn))
80
81      if show_fstr:
82          ft = f.GetTitle()
83          for i in range(par_count):
84              ft = ft.replace('[%d]' % i, pnames[i])
85          print '%s = %s' % (fcn_name, ft)
86
87      ndf = f.GetNDF()
88      if ndf > 0:
89          chisq = f.GetChisquare()
```

```
90          print 'chisq/ndf = %.2f/%d = %.4f' % (chisq, ndf, chisq/ndf)
91
92      ostr = '%' + '%d' % (max_pname_len+1) + 's = %10g +- %g'
93      for i in range(par_count):
94          pn, pv, pe = pnames[i], f.GetParameter(i), f.GetParError(i)
95          print ostr % (pn, pv, pe)
96
97
98  def get_fit_result(f):
99      '''
100     Liefert ein Dictionary mit den Fitergebnissen.
101     f: Die Fitfunktion
102     '''
103     par_count = f.GetNpar()
104
105     pnames = []
106     for i in range(par_count):
107         pn = f.GetParName(i)
108         for c in '#_{}':
109             pn = pn.replace(c, '')
110         pnames += [pn]
111
112     d = {}
113
114     d['ndf'] = f.GetNDF()
115     d['chisq'] = f.GetChisquare()
116     if d['ndf'] > 0:
117         d['rchisq'] = d['chisq'] / d['ndf']
118
119     for i in range(par_count):
120         d[pnames[i]] = (f.GetParameter(i), f.GetParError(i))
121
122     return d
```

B.11. Physikalische Konstanten und angegebene Werte (`konst.py`)

```
1   #!/usr/bin/python
2   # -*- coding: iso-8859-1 -*-
3
4   from math import cos, pi
5   from Scientific.Physics.PhysicalQuantities import PhysicalQuantity as Q
6
7   # Physikalische Konstanten
8   c  = Q('1 c')    # Lichtgeschwindigkeit
9   me = Q('1 me')   # Ruhemasse des Elektrons
10  re = Q(2.817940325e-15, 'm')  # Klassischer Elektronenradius
11
12  # Einheiten
13  keV = Q('1 keV')  # keV-Einheit
14
```

```
15 # Uebergangs-Energien der verwendeten Quellen (www.nndc.bnl.gov)
16 Ena = 1274.53 * keV                    # Photopeak-Energie
17 Ena_compt = Ena/(1. + .5*me*c**2/Ena)  # Compton-Kante
18 Ena_anihi = (me*c**2).inUnitsOf('keV') # Photopeak, Anihilation
19 Ena_anihi_compt = Ena_anihi/(1. + .5*me*c**2/Ena_anihi) # Compton-Kante
20 Ecs = 661.657 * keV                    # Photopeak-Energie
21 Ecs_compt = Ecs/(1. + .5*me*c**2/Ecs)  # Compton-Kante
```

B.12. Weitere Hilfsroutinen (`tools.py`)

```
1  #!/usr/bin/python
2  # -*- coding: iso-8859-1 -*-
3
4  def gew_mittel(xsx):
5      '''gew_mittel(list(float,float)) -> (float, float)
6      xsx  : Liste aus Tupeln der Messwerte mit jeweiligen Fehlern
7      ->   Tupel (gx, sgx) aus gewichtetem Mittel und dessen Fehler'''
8      suma = 0. * xsx[0][0] / xsx[0][1]**2
9      sumb = 0. * 1. / xsx[0][1]**2
10     for xi,sxi in xsx:
11         suma += xi / sxi**2
12         sumb += 1. / sxi**2
13     return (suma/sumb, sumb**(-0.5))
14
15 def arith_mittel(x):
16     '''arith_mittel(list(float)) -> float
17     x  : Liste aus Messwerte
18     ->   arithmetisches Mittel der Messwerte'''
19     sumx = 0. * x[0]
20     for xi in x:
21         sumx += xi
22     return sumx / len(x)
```