Technische Universität München
Fakultät für Physik

Abschlussarbeit im Bachelorstudiengang Physik

GEANT4 Simulation of a Doppler Shift Attenuation Experiment

GEANT4 Simulation eines Experimentes auf Grundlage der Doppler-Shift-Attenuation Methode

Andreas Schimpf

24. Juli 2011

Erstgutachter (Themensteller): Prof. S. Bishop
Zweitgutachter: Prof. S. Paul

Contents

Chapter 1

Introduction

Today, computers and electronic devices play an always increasing prominent role in our daily life. From shopping over communication and entertainment to actual work, everything is done with the help of a computer or a derived device. Especially engineering and science rely heavily on the raw processing power of todays supercomputers to create complex machines or to understand an even more complex universe. From the very beginning of its existence computation changed the work of scientists. Universally programmable calculation machines like those built by men like Konrad Zuse, John Presper Eckert and John William Mauchly in the 1940's provided for the first time the means to automatically perform complex algorithms quicker and more accurate than any human could. Without doubt these computers boosted scientific progress, which in return lead to more powerful computers. Since those first days, computers have made a quantum leap forward. Supercomputers like the japanese K Computer, today's fastest computer, allow us to simulate more and more complex systems with more and more detail. In physics, simulations are used to improve and design real experimental setups and can give us hints were to look for relevant experimental data. Furthermore, the results of simulations allow us to investigate processes, which for example can not be reproduced in a real experiment for some reason. So instead of running an actual experiment, data from the simulation is compared to observations made.

Many examples for this can be found in astrophysics, where the circumstances are often extreme and of huge scale: Stellar birth and the whole life cycle of different types of stars, supernovae or classical novae are simulated in order to better understand them and to test the latest theories. To make these astronomical simulations of whole events more accurate and to develop the actual physical theories behind the computing further, a profound knowledge of the properties of the nuclei and particles involved is necessary. Experiments on earth can deliver these. One of these experiments is the Doppler Shift Attenuation Experiment of the E12 Group of the physics department of the Technische Universität München (TUM) conducted at the tandem accelerator of the Maier-Leibnitz-Labor (MLL) of the TUM and the Ludwigs-Maximilians Universität (LMU). Goal of this experiment is to measure lifetimes of excited states in compound nuclei from A=20 up to A=40 which are important in the understanding of classical novae. In chapter 2 I talk about the astrophysical motivation of this experiment. In the third chapter, I introduce the concept of the Doppler Shift Attenuation Method before I speak about the experimental setup in chapter 4. Chapter 5 is about the GEANT4 Monte Carlo simulation toolkit and the work I've dine with it. This thesis ends with the conclusions in chapter six.

Chapter 2

Motivation

2.1 Classical Novae

As mentioned in the introduction, the motivation for the Doppler Shift Attenuation Method (DSAM) experiment at the MLL is to better understand classical novae. A classical nova is an cataclysmic event, where nuclear burning is ignited for a short time on the surface of a white dwarf after hydrogen mass deposition. For several days, the luminosity increases by a factor of 10^4 [1] coupled with a strong outburst of gamma rays. It is estimated, that 30 to 60 novae happen every year in the Milky Way.

From what is known today, classical novae occur in binary stellar systems, where a white dwarf has a close companion. These stars are either main sequence stars or alternatively aging stars which might even be turning into a red giant. If the bigger companion exceeds its Roche lobe, the layer outside the Roche lobe can escape the gravitational pull of the star and be accreted by the white dwarf. A so called accretion disc will be formed. A typical accretion rate would be $\dot{M} \simeq 10^{-9} M \odot / a$, [3] of mainly helium and hydrogen gas. The very slow increase in mass means that the accreting white dwarf isn't running the risk of getting near the Chandrasekhar-limit for a very long time. As the white dwarf is made of so called degenerate matter, it does not expand despite the enormous heat coming from the compression of the accreted gases on the surface of the white dwarf by the immense gravitation. The inner layers of the hydrogen shell are compressed until they too, reach degenerate conditions. Degenerate matter is so far compressed, that the Pauli exclusion principle plays an important role, after which fermions, in this case electrons, can not be in the same state. Further compression of a white dwarf would mean that the electrons of the closely packed nuclei would have to be lifted to higher energy states, which in turn would require even more energy. A degenerate pressure manifests and competes with the gravitational force up to the Chandrasekhar-limit when the degenerate matter collapses which would lead to a type Ia supernova and, in the end, to a neutron star or black whole, making accreting white dwarves possible progenitors for core collapse supernovae. If the heat and pressure on the white dwarf's surface become sufficiently high, a thermonuclear reaction takes place:

The hydrogen on the surface begins to burn via the proton-proton-chain during the accreting phase. This reaction is sensitive to the temperature ($\propto T^4$) [3] and produces more energy with increasing temperature, which again increases the energy output of the burning in return. This continuous increase in temperature under these partially degenerate conditions is called a thermonuclear runaway (TNR). At temperatures around 13 million K the CNO cycle begins and becomes the dominant reaction at about 17 million K. The CNO cycle has an even higher temperature dependency ($\propto T^{18}$). [3] This leads to a quickly increasing temperature and energy output, which exceeds the amount of energy that can be radiated away. Convection starts, mixing core material of the white dwarf with the gas envelope, bringing even more C, N and O nuclei, the catalysts needed for the CNO cycle, in the hydrogen rich envelope. At high temperatures ($T > 10^8 K$) the nuclear energy generation is limited by the time-scales of the slower and temperature intensive $\beta+$ - decays, particularly $^{13}N(\tau_{1/2} = 598s)$,$^{14}O(\tau_{1/2} = 71s)$, and $^{15}O(\tau_{1/2} = 122s)$. [3] At these temperatures most of the CNO nuclei become $\beta+$-unstable. Near the peak of the thermonuclear reaction a big fraction of the $\beta+$ unstable nuclei are able to reach the outer layers of the dwarf's accretion envelope by convection, before they actually decay. These decays lead to a respectable energy deposition in the outer layers of the accreted envelope, able to exceed 10^{13} to $10^{15}\frac{erg}{g \cdot s}$. [3] Consequence of this intensive heating is the flattening of the temperature gradient in the envelope and thus the end of convection. The energy provided by the $\beta+$-decays is sufficient to make the thermal pressure outmatch the degenerate pressure. Expansion of the envelope sets in and the material is ejected off the white dwarf, not destroying it. There are white dwarves known to go repeatedly through this process (e.g. recurrent nova RS Ophiuchi in 2006, 1985, 1967,1958, 1933, 1898)[8] and it is thought, that actually all classical novae are repetitive in a timeframe reaching from a few years to 10^5 years.

2.2 Nucleosynthesis

Nucleosynthesis in classical novae are of great interest, as these are expected to play a role in the enrichment of the interstellar medium. Studying these events helps us to understand the galactic abundances of elements such as ^{15}N, ^{17}O, ^{13}C and ^{7}Li, which are significantly overproduced in nova outbursts on an CO white dwarf.[11] Nova explosions with an ONe white dwarf produce even intermediate mass elements up to Ca, according to observations of corresponding nova shells.

Reaction rates in explosive H-burning, like it happens during a classical nova, are dominated by resonant proton-capture into excited states, which can be explained with the compound nucleus model. This model says, that some nuclear reactions between two particles, especially ones with a low energy (typically < 50MeV for bombarding particle[12]), do not directly produce two new particles, but do so via an intermediate step:

$$A + B \rightarrow C^* \tag{2.1}$$

First the initial nuclei react into a new, highly excited and unstable compound nucleus.

$$C^* \rightarrow D + E \tag{2.2}$$

After a relatively long time (from 10^{-19} to 10^{-15}s) the compound nucleus disintegrates. Often into a small particle like an alpha particle, proton or a gamma and a product nucleus. So observations of gamma ray emissions from novae could allow us to look directly at these processes. The properties of the compound nuclei such as spin, excitation energy, lifetime and partial decay widths have a big influence on the proton-γ reaction rates. This becomes clear if we look at the thermal reaction cross section for the (p-γ) - reaction:

$$\langle \sigma v \rangle = \left(\frac{8}{\pi\mu}\right)^{1/2} (kT)^{(-3/2)} \int\limits_0^\infty E\sigma(E) exp\left(-\frac{E}{kT}\right) \tag{2.3}$$

where μ is the reduced mass, T the temperature, k the Boltzmann-constant and finally σ the reaction cross section.

For medium and heavy nuclei the density of the resonances increases, so that basically all reactions can be considered resonant. And after some mathematical operations we get

$$\langle \sigma v \rangle = \left(\frac{2\pi}{\mu kT}\right)^{3/2} \hbar^2 \sum_i (\omega\gamma)_i exp\left(-\frac{E_i}{kT}\right) \tag{2.4}$$

where the sum over i is the sum over the resonances in the compound nucleus and where

$$\omega\gamma = \frac{2J_i + 1}{(2J_p + 1)(2J_X + 1)} \frac{\Gamma_p \Gamma_\gamma}{\Gamma_p + \Gamma_\gamma} \tag{2.5}$$

is the resonance strength. The J_i is the spin of the resonance state, J_p and J_X are the spins of the proton and of the target (X). Γ_p and Γ_γ are the partial widths of proton an γ decays. The resonance strength can be further transformed into

$$\omega\gamma = g(1 - B_p) B_p \frac{\hbar}{\tau} \tag{2.6}$$

with B_p as the branch ratio $B_p = \frac{\Gamma_p}{\Gamma_p+\Gamma_\gamma}$. In the end the total reaction rate is

$$r_{12} = \left(\frac{2\pi}{\mu\tau}\right)^{3/2} \hbar^2 \frac{N_1 N_2}{1+\delta_{12}} \sum_i \hbar^2 exp\left(-\frac{E_i}{\tau}\right) \tag{2.7}$$

where the N_i are the number of particles of the different species, τ is the lifetime, E is the energy and finally δ_{12} is the Kronecker-delta. A full explanation of thermonuclear reactions rates van be found in [4]. In 2.7 the dependency of the reaction rate from the inverse lifetimes of the excited states can be seen nicely. So as already was mentioned above, it is important to know the exact properties of the compound nuclei. This is why the E12 group of the TUM is conducting its doppler shift attenuation experiment at the MLL's tandem facility.

Chapter 3

Doppler Shift Attenuation Method

The Doppler Shift Attenuation Method (DSAM) is a well characterized method to determine the lifetimes of gamma emitting excited nuclei produced in a nuclear reaction in the range from 10^{-10} s down to 10^{-14} s [2].

The basic idea behind DSAM is that a projectile particle is shot on a thin target like a implanted metal foil, where a direct transfer reaction takes place, producing an excited nucleus. The excited nucleus decays at some time and emits a gamma while it is being slowed down in the target. If the stopping power of the target is high and the nucleus is already at rest when it decays, the emitted gamma is not Doppler-shifted. If the nucleus decays whilst still moving with a velocity v(t), the gamma is Doppler-shifted and has the energy

$$E_\gamma = E_0 \left(1 + \frac{v(t)}{c} cos\theta\right) \tag{3.1}$$

where E_0 is the energy of the emitted gamma with the excited nucleus being at rest, $v(t)$ is the velocity of the nucleus, c being the speed of light and θ being the angle between the momentum vectors of the decaying nucleus and of the emitted gamma.

If the stopping power of the target material is known, the lifetime of the excited nucleus can be obtained from the spectrum of the detected gammas. The reaction in the target is of the form

$$A + B \rightarrow C^* + d \tag{3.2}$$

where A, B are the initial particles (projectile and target), C^* is the excited nucleus and d a light ejectile like α, p, d, t. It is important that the stopping power of the light ejectile in the target material is small, so that the ejectile is able to travel with virtually no loss of energy through the target material and if it is the case, through the target backing made of a high Z stopper material. The energy and the exit angle of the ejectile are measured and can then be used to determine the angle and energy of the excited nucleus C^* via inverse kinematics. To know the mean energy of the measured gammas, we need a distribution of the velocity of the nucleus in the moment of its gamma-decay. If we define the attenuation coefficient F as the ratio

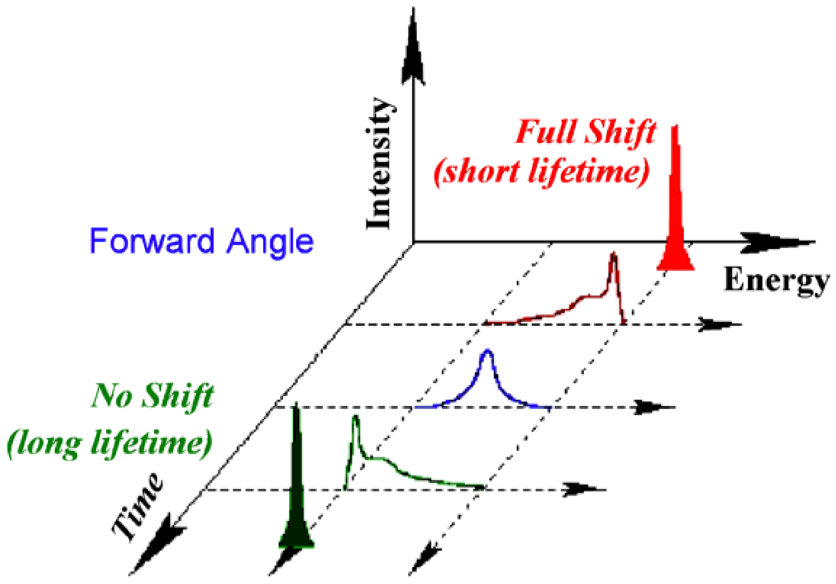

Figure 3.1: Effect of the lifetime of the excited nucleus on the gamma spectrum at the same stopping power of the target material. Changing the target material and therefore the stopping power has the same effect on the spectrum.

of the average velocity $\bar{v}$ to the initial velocity v_i one can write the energy of the measured gammas as

$$\bar{E}_\gamma = (E_\gamma)_0 \left(1 + F\frac{v_i}{c}cos\theta\right) \tag{3.3}$$

with

$$F = \frac{\bar{v}}{v_i} \tag{3.4}$$

Using $\beta_0 = \frac{v_i}{c}$, equation 3.3 can be rearranged to

$$\frac{\bar{E}_\gamma - (E_\gamma)_0}{(E_\gamma)_0} = F(\tau)\beta_0 cos\theta \tag{3.5}$$

so that we find the velocity distribution of the excited nucleus on the righthand side. In order to calculate $F(\tau)$ one first needs the stopping power of the used nuclei in the target material, which allows to calculate a velocity distribution $\beta_m(t) = \beta(t)cos\theta$

of the nuclei as a function of the time of flight (tof). This distribution can then be folded with the decay probability of the excited nucleus giving us

$$F(\tau)\beta_{m,0} = \frac{1}{\tau}\int\limits_{0}^{\infty} \beta_m(t)e^{-\frac{t}{\tau}}dt \tag{3.6}$$

where $\beta_{m,0}$ is the modeled velocity at t=0. To get the lifetime τ, the righthand side of 3.6 has to be folded with the specific response function of the detector to fit the measured gamma-energy spectra [7]. This is basically a optimization problem where the simulated energy spectrum of the detector is compared to the actual shape and iterated for a minimum χ^2.

Chapter 4

Experimental Setup

The experiment at the tandem accelerator of the Maier-Leibnitz-Labor (MLL) in Garching aims at measuring the lifetimes of different excited nuclei in the range from A=20 to A=40. As mentioned above, gamma emissions of some of these nuclei could allow us to look at direct products of nuclear reaction in classical novae. The first goal of the experiment it is then to measure the lifetime of $^{31}S^*$ using the following reaction:

$$^{32}S +^{3} He \rightarrow^{31} S^* + \alpha \tag{4.1}$$

The ^{32}S is accelerated with the MLL's 14 MeV tandem and hits the target with a beam energy of 85 MeV. The target is a $18\mu m$ thick gold foil, implanted at the Forschungszentrum Rossendorf (FZR) with ^{3}He to a depth of 0.2 μm with a implantation dose of $6 \cdot 10^{17} \frac{ions}{cm^2}$ and is positioned on a target ladder, able to hold 5 different targets. Additional backing for the target is not used in this experiment as the gold has sufficient stopping power. Diffusion of ^{3}He throughout the gold foil induced by heating by the beam make it necessary to cool the target. To keep interferences to a minimum the target ladder is kept in vacuum, inside a cylindrical steel containment. In order to prevent any remaining molecules and particles in the target chamber to condense on the targets, a so called "cool trap"is installed in the target chamber: It basically consists of a 40cm long cooled copper tube around the beam along the latter's axis. It ends almost directly in front of the target and is in thermal contact with the target ladder through copper braid. The copper tube itself is cooled with a LN2 cooling finger. Its task is to catch the remaining particles in the target chamber and make them condense mostly on its surface instead of the targets' surface and helps to obtain a higher quality vacuum and to keep the actual target clean. To ensure this, the target can be lowered inside the steel containment at the beginning of the cooling process so that any remnant particles rather adsorb on the much cooler cool trap instead of the targets. After experimental conditions have been reached, the target ladder is moved up and thermal contact with the copper tube is established. Two tantalum collimators placed inside the coper tube 10 cm apart from each other are used to align the incoming ions more along the beam axis and for beam analysis and control.

For detection two different types of detectors are used: A silicon ΔE-E particle identification telescope and one or multiple high purity germanium (HPGe)detectors. The particle identification telescope consist of two monolithic silicon detectors of different thicknesses and is placed in the beam line inside the target chamber. The thinner ΔE detector of either 50μm or 250 μm is placed closer to the target and the thicker position sensitive E detector of 1000μm are placed 5 mm apart. The distance between the target and the front surface of the position sensitive silicon detector is 3 cm. The particle identification telescope is used to detect the alpha particles coming from the target, as every charged particle has a characteristic ΔE-E ratio. The alpha particles are stopped in the second detector. From the deposited energies in both detectors, one gets the particle energy, and with the detected position the exit angle of the alpha can be calculated. The gammas of the decaying nuclei should pass the silicon detectors without effort.

To detect these, one or multiple HPGe detectors are used. These detectors are considerably larger than the telescope and therefore placed outside the steel containment, which does not largely affect the experimental results as the gamma particles will most likely travel through the possible obstacles like the silicon detectors, the target chamber and the air gap between the latter and the HPGe detector. The detectors are placed on the level of the beam at different angles around the cylindrical target chamber to be able to compare doppler shifted spectra to non-shifted spectra or to compare the gamma spectra of the forward hemisphere (meaning in the beam direction) to that of the backward hemisphere. In order to be able to select the important data and hence the right nuclear reactions, coincidence between the particle identification telescope detecting the alpha particle and the high purity germanium detector signals is used. Finally, as one picture tells more than a thousand words a CAD rendering provided to me from Clemens Herlitzius is shown below(Figure 4.1).

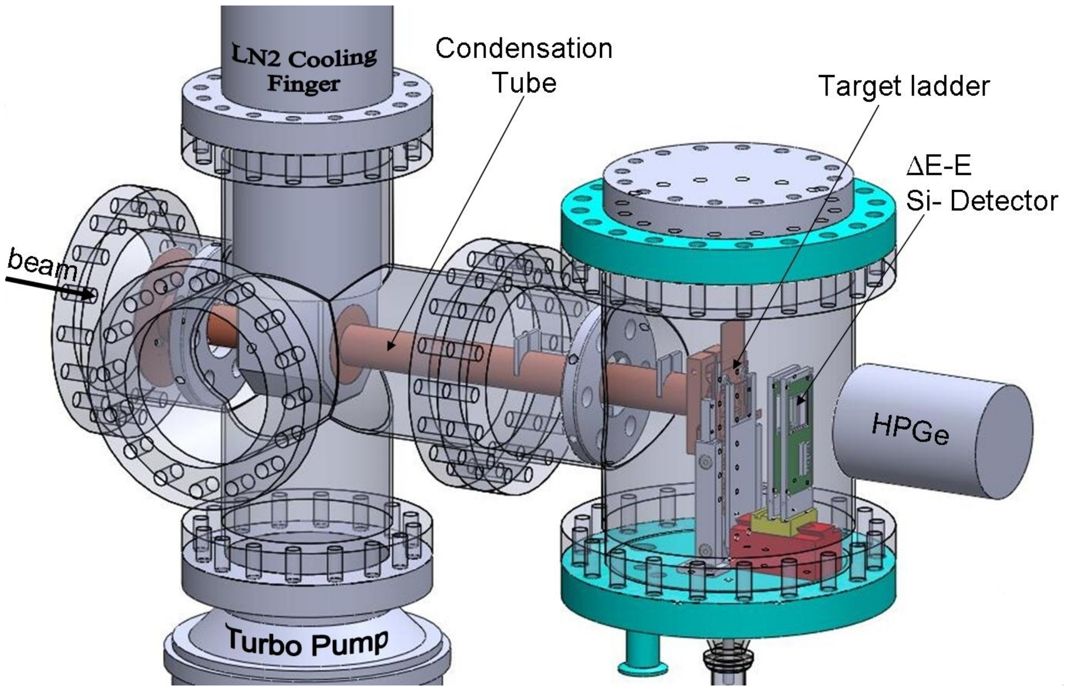

Figure 4.1: CAD rendering of the experiment

Chapter 5

GEANT4 Simulation

As already mentioned in the introduction, simulations are a commonly used tool in the development and optimizing of experimental setups. In this experiment a GEANT4 Monte Carlo simulation can be used to simulate a gamma spectrum of the HPGe detector and to simulate the data collected with the particle identification telescope in the form of a E-ΔE plot. This simulation can then be used to investigate the effects that changing geometries, projectile properties and materials would have on the experiment. Without the long and more expensive work at the real experiment, quick answers to questions like under which angle the detectors should ideally be positioned or what material is best suited to build a target foil from it if the projectile is changed, can be found. In this thesis I took an already existing GEANT4 simulation of the DSAM experiment done by Janina Fiehl for her diploma thesis in 2010 [7] and extended it with relativistic kinematics of the nuclear reaction 4.1, with a more realistic geometry by adding the target chamber to the simulation and finally by implementing the particle identification telescope into the simulation.

GEANT stands for GEometry ANd Tracking and is a software toolkit developed at CERN for simulating "the passage of particles through matter"[6] using Monte Carlo methods. Since GEANT4, the software bundle uses object oriented programming and switched from Fortran 77, that was used up to GEANT3, to C++. A wide range of applications of GEANT4 exist, among these are high energy physics like simulations of the BABAR experiment at SLAC or of the ATLAS experiment at CERN, "space and radiation"analysis done by ESA for radiation analysis of equipment like the XMM-Newton X-ray telescope or dose estimation for the ISS and medicine-orientated simulations, dealing for example with medical imaging or radiotherapy.

5.1 Monte Carlo simulations

The Monte Carlo (MC) method was developed in the 1940's at the Los Alamos Lab by John von Neumann and Stanislaw Ulam and named by von Neumann after the famous Monte Carlo casino, where Ulam's uncle used to gamble away his money. It was first used during the development of the H-bomb. The MC method uses random numbers or pseudo-random numbers to iteratively evaluate a deterministic problem and turning it basically into a statistical one. Monte Carlo simulations are used in a broad field of applications, going from statistical physics, theoretical chemistry, engineering to business and many more. In physics, MC simulations are used for example in statistical systems, in quantum field theory, lattice quantum chromo dynamics, molecular dynamics, with the finite element method, for designing detectors or for the simulation of the evolution of galaxies.

The basic idea of the MC method is best explained with a simple example: We have a square paper of 1m x1m with a very complex two dimensional geometrical figure printed on it and want to know how large the surface of this figure is. We can not calculate the value directly, so we transform the geometrical problem in a statistical one: We throw a "random dart"at the paper, which of course hits it every time but at a absolutely random spot. We now note if the dart has hit the figure or a blank spot and repeat the throwing many thousand times. The ratio of "dart hit the figure"to "total number of darts thrown"approximately gives us the fraction of the paper surface which is covered by the figure. If we then multiply this fraction with the actual size of the whole paper, which here of course is 1 m^2, we get a very good estimated value of the real surface size of the complex figure.

It is obvious that the MC method demands a high number of quality random numbers to make the results precise. It is faster to use pseudo-random number algorithms than algorithms that produce real random numbers, but it has to be assured that the pseudo-random numbers meet some criteria: First of all, the random numbers should be produced in a uniform distribution (random number is taken from [0,1]) and not be correlated with each other. As all pseudo-random generators will repeat their sequence of random numbers, the amount of random numbers used in the MC simulation has to be within one period of random numbers. Most pseudo-random number algorithms use a "seed"value which is used to initialize the algorithm. For practical reasons it is of advantage that the same seed produces the same sequence of random numbers to make debugging processes easier to handle. Further should the algorithm be fast to run, give the same results on different computers and be insensitive to seeds, meaning that the length of the sequence of random numbers and other properties do not depend on the used seeds. GEANT4 uses random number generators found in the CLHEP (Class Library for High Energy Physics) [5] library it uses. As Monte Carlo simulations are basically integrators

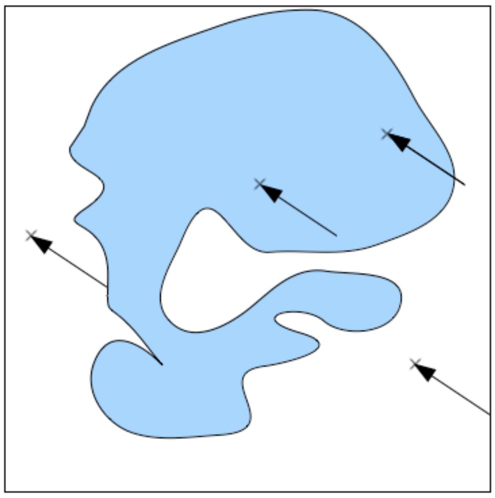

Figure 5.1: "random darts"on the paper

I will shortly show how one dimensional integrals are calculated using the MC method. Integrals of the form

$$I = \int_a^b f(x)dx \tag{5.1}$$

can be approximated by taking N random numbers x_i with i= 1...N using

$$I \approx \frac{b-a}{N} \sum_{i=1}^{N} f(x_i) \tag{5.2}$$

If N is sufficiently large, the central limit theorem, which will not be discussed here, says that the error of this approximation is $\propto \frac{1}{\sqrt{N}}$. If N $\rightarrow \infty$ the error becomes 0 and the approximation gives an exact result. Further explanations of Monte Carlo simulations in physics can be found for example in [10].

5.2 GEANT 4

GEANT4 is not a program in the usual sense. It has no user interface and can not be opened and run alone. An external C++ compiler (mostly g++ on unix systems) and a external text editor are necessary. It can not be compared to, for example, to a CAD program like CATIA or Solid Works, which too are a vastly complex programs, but where your constructions are saved in files which you can only use together with the program. GEANT4 is best described as a development tool for creating C++-written programs that simulate experimental setups. The simulation itself is basically a stand-alone program. GEANT4 comes with a huge number of specialized libraries, which contain lists of particles, physical processes, math and tons of predefined functions, which make it highly versatile and save a lot of programing time. To prevent a catastrophic scale of confusion on the simulation developer side, it gives the developer a more or less strict structure of how the program has to be written, reducing the confusion for the beginner to barley manageable. The given structure enables experienced developers around the world to understand and to work with someone else's code rather quickly and makes it relatively easy to adapt a already written program, or at least parts of it, to other experiments or even applications. To build a program with GEANT4 you first have to define your geometrical setup, including all the materials and Volumes which you have to place in a "world volume". This "world volume"is basically an empty box with a finite size and a given coordinate system, in which the whole experiment takes place. Next you have to tell GEANT what particles will be used in the simulation and which physical processes might occur. It is important to note that absolutely all physical processes and particles which might occur somewhere during the experiment, including secondary particles, have to be predefined. Further GEANT asks you to define how an event is started and to generate the primary tracks: Where does the particle enter the simulation and what are its properties? Possible particle properties are position, energy, momentum, electric charge, spin, rest mass, lifetime and so on. Finally you have to extract the information you need, for example by defining which volumes in the world volume are to be considered as detectors and what properties these detectors detect. This data can be used to produce histograms. In addition, visualization or user interfaces can be added to the simulation. An overview of the GEANT4 file organization is shown below:

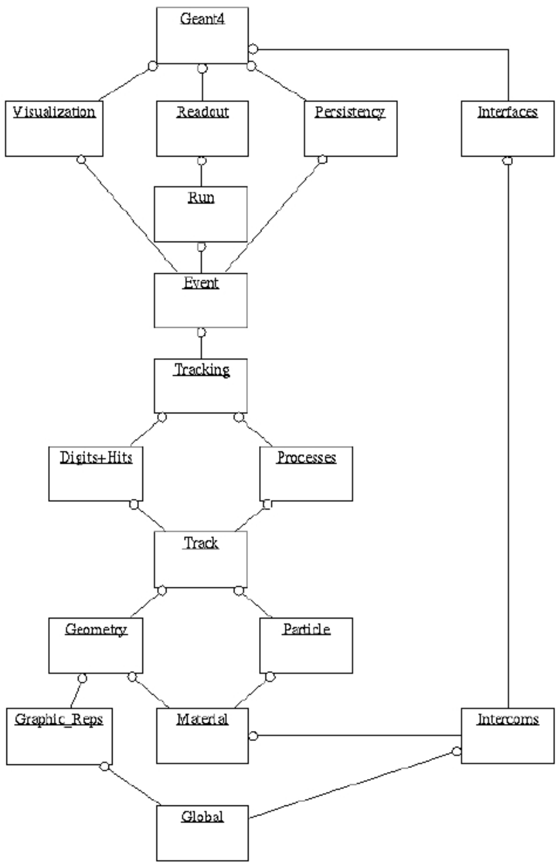

Figure 5.2: GEANT4 class categories from the GEANT4 Application Developer Manual

Figure (**??**) should be read from bottom to top. The bottom class "Global", being the most basic one, is necessary for all the following classes, which contain all the physics, geometries, visualization commands, histogramming and user interaction commands. The final GEANT4 class puts all the pieces together to build the simulation. I will provide an overview over the most important kernel classes (not made by the user):

1. **Global Usage Classes**
   These are the most basic classes, which are already predefined and have only to be included in the other files of the simulation as header files. This category contains all constants, structures, classes and types which are commonly used

in all types of simulations. It further is the interface between the GEANT4 libraries and third party libraries like those of CLHEP. To make sure that GEANT4 simulations behave the same on all computers, GEANT4 uses special typedefs, which even redefine standard C++ definitions like int (G4int), double (G4double) or basic functions like cout (G4cout). These typedefs are also embedded in the global classes and it is recommended to use them.

2. **Run**
   This is the largest unit of the simulation. With "beam on", an analogy to real life experiments, the run is started and a sequence of Events, the next smaller simulation unit, is run. A run is a complete simulation, but with the the functions for the particle sources being called only once. So only one predefined set of primary particles is generated and further processed. During a run the detector setup and the participating processes and particles are definite and can not be changed. This is due to the fact that the geometry is initialized and cross-section tables dependent on the materials and particles are calculated before a run. In the code, a run is represented as the G4Run class or a derived class.

3. **Event**
   An Event is best imagined as a stack of particle tracks and their properties. At the beginning of the Event, this stack is filled with the primary Tracks produced during initialization of the Event. These Events are then calculated one by one after their order in the stack ("first in first out"). If secondary particles are produced at some point, these are pushed back into the stack and calculated like the primary ones. If the stack is empty, the Event is over. During the Event, the particles' trajectories and their hits in detectors are registered and saved. The according class is the G4Event class or a derived class.

4. **Track**
   A Track is a snapshot of a particle and has all the current information and only the current information of the particle (i.e. position, energy, momentum, charge, spin and so on at a given moment). A Track is deleted, if the particle leaves the world volume, disappears due to some processes like decay or inelastic scattering, loses all its energy and has no further influence on the simulation or because it is simply deleted by the user. Tracks can only be manipulated by process class objects, which where called or defined at the beginning of the run. A Track is represented by the G4Track class.

5. **Step**
   This is the smallest unit of the simulation an can be considered as a "δ"-information of the Track class. This does not mean that the Track is a sum of

Steps, but Steps update the Track (a Track has only the current particle properties) to the latest state. A Step consists of a start point (PreStepPoint) and an ending point (PostStepPoint). Each of these points contains the information about its coordinates, the volume it is in and its material. A Step also holds all the changes of the Track properties. To determine which process should be applied in the simulation, Step uses all defined processes available to the particle and estimates the interaction length of every single one. The process with the shortest interaction length is then chosen to calculate the δ-changes of the Track. These changes are then applied to the Track at the PostStepPoint. The according class is the G4Step class.

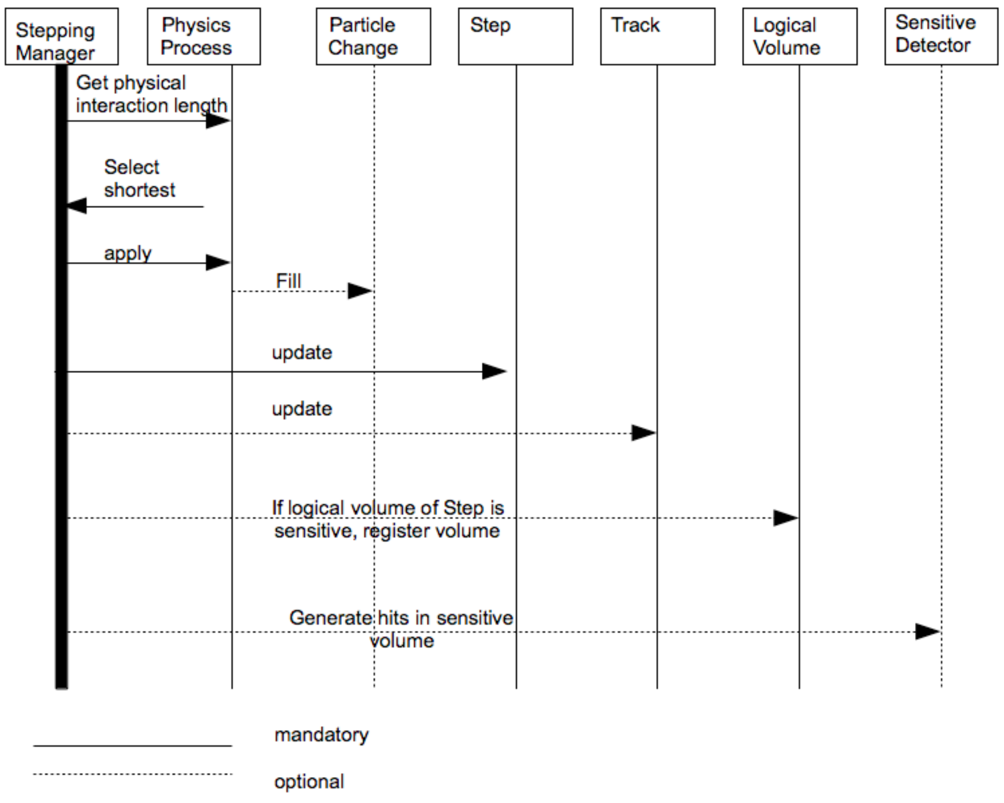

Figure 5.3: Visualization of how one step is calculated

The user usually does not directly program any of these steps in the kernel, but so called "user classes", which are used by the kernel classes to create the simulation. The mandatory and most important ones are:

1. **main()**
   GEANT4 does not provide a main, as it can not know how complex the simulation is going to be and what optional classes the user wants to apply. The main() contains all the initializations of the user and GEANT4 kernel classes and the initialization of optional classes such as visualization or user interface implementation. It also reads macro files which belong to any of the initialized classes.

2. **UserDetectorConstruction**
   The UserDetectorConstruction contains the geometry and materials of the whole experiment. It contains all the user or standard definitions of the used materials. Further, all geometry volumes, including the world volume, are defined here. To do this, there are two constructions to be used: The physical volume and the logical volume. First the logical volume has to be created by giving it the geometrical shape, the material and a name. Second, the physical volume is created by giving it the mother volume, meaning in which already existing volume it should be placed with the world volume being the default, the position and rotation in this volume, the corresponding logical volume and a name. If it is the case, a geometrical volume can be defined as a "sensitive detector". This means that hits of particles interacting with the volume by a physical process are stored with every Step. Together with different other user classes, this is then used to simulate a detector.

3. **UserPhysicsList**
   The name already suggests what is done in this class: Here all particles and physical processes that might occur during the simulation have to be defined by the user or by calling predefined definitions in GEANT's libraries. In GEANT4 even the simple particle transportation is defined as a physical process. This may seem odd, but makes sense if one considers, that only processes are able to alter the Track class objects. Each particle has a list of processes which, like can be seen in figure 5.3 are all invoked before a Step, in order to get suggested physical interaction lengths of each process to decide which process should be used in this Step.

4. **UserPrimaryGeneratorAction** The primary generator action is used to define the used particle source of the experiment. In most cases it is sufficient to use the so called particleGun function which requires the developer or, with an implemented user interface, the user, to give it the kind of particle that is "shot", the position of the particle source and the particle energy and a direction vector of the particle. It is here that I implemented the relativistic kinematics of the nuclear reaction and handled them over to the particleGun function.

For a more complete introduction to GEANT4 I recommend the "Geant4 User's Guide for Application Developpers"which can be downloaded at [6] and the novice examples which come with GEANT4 and which are well explained at the GEANT4 webpage of the FNAL (Fermi National Accelerator Laboratory)[9].

5.3 Geometry of the Simulation

Other than might be expected, the simulation does not include the nuclear transfer reaction, but only the products coming from it, hence the decaying $^{31}S^*$ and the alpha particle. The reason for that, is that GEANT does not provide any classes or functions for such reactions, and that it is not really necessary, because the simulation at this stage only aims at modeling the detector output for the γ and the alpha particles. Simulating the proper transfer reaction inside the ^{3}He-implanted gold foil would make the simulation very much more complicated and slow it down considerably. So far, the simulation contains the steel containment made from the correct composition of the applied stainless steel, one high purity germanium detector at 0°, the particle identification telescope and the gold foil without implantation. As particle source the relatively simple particleGun function is used. To simulate that two different particles are produced in the nuclear reaction, I just used two particleGun functions, one for the excited sulfur ion and one for the alpha particle, that are placed both at the same spot 5μm before the target gold foil. The reason for placing the "spot"of the reaction before the target, instead of putting it inside the gold foil in a depth corresponding to the ^{3}He implanted layer, is, that GEANT4 had problems with the particle source being placed inside the foil. Again, this is also negligible considering that the important part is that the excited nucleus is stopped inside the target foil. A OpenGL rendering of the simulation geometry can be found below.

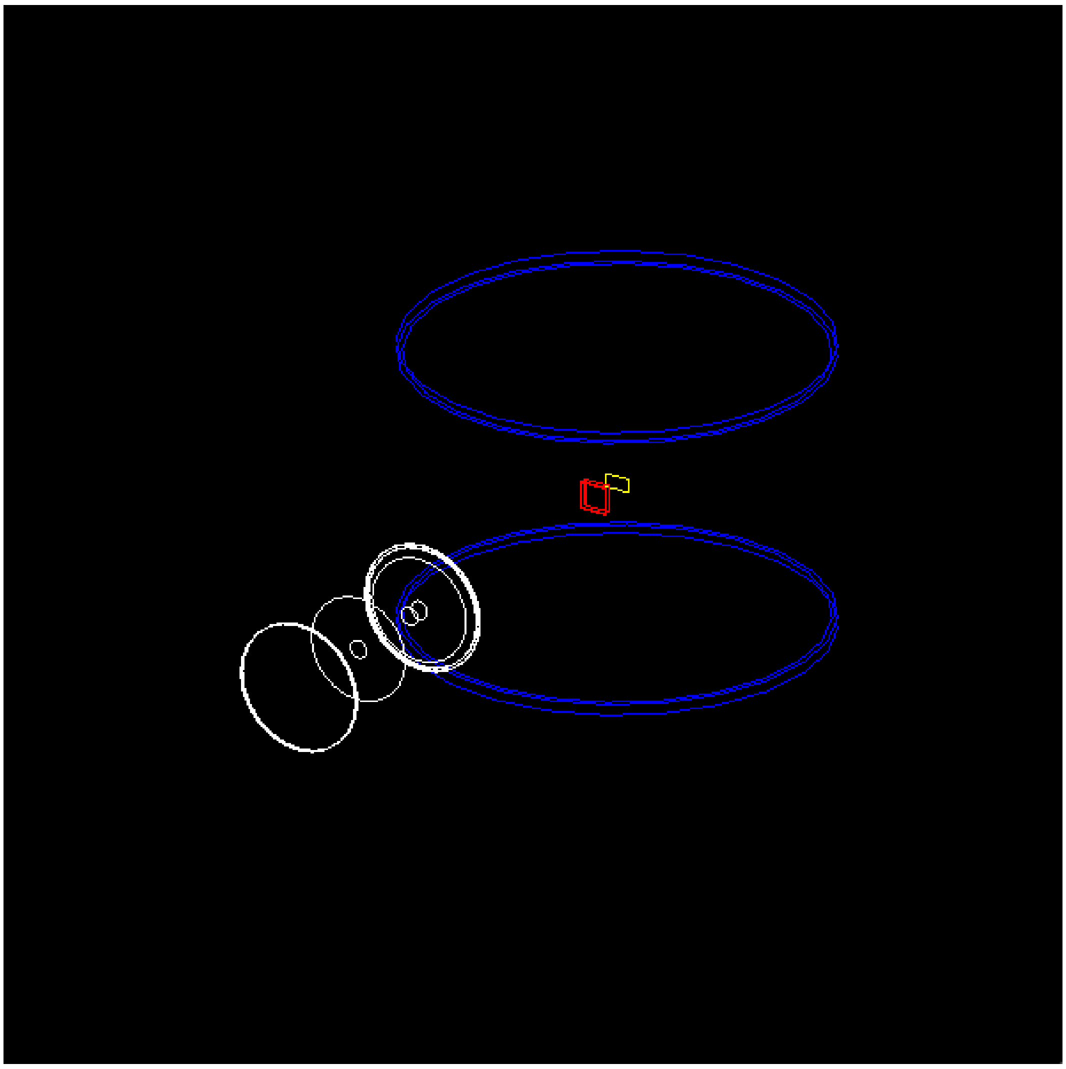

Figure 5.4: Visualization of the simulation. For better distinction, colours are used: yellow for gold, red for silicon, blue for stainless steel and white for the HPGe detector. The particle source is invisible and not represented in the visualization. The beam would come from the upper right, along the axis through gold foil, silicon detectors and the HPGe detector.

5.4 Relativistic Kinematics

The original simulation had no alpha particles in it, only the decaying nucleus, thus only one particle source was used. A simple isotropic statistical distribution of the particle direction vector and a fixed particle energy simulated the kinematics of the reaction. The primary aim was to include both types of particles produced in the nuclear reaction with properties calculated with relativistic kinematics of a fixed target experiment. The particleGun function needs to know, apart from the particle type, the particle's kinetic energy and its momentum direction vector.

For the calculations it was assumed that the ^{32}S (rest mass: 29.78179 GeV/c^2) is moving with 85 MeV of kinetic energy hitting a resting ^{3}He (rest mass: 2.809413 GeV/c^2) particle. For simplification reasons, the problem was transformed from the lab frame to the center of mass frame, where the particle energies and the momenta of the reaction products were calculated with energy-momentum 4-vectors. Still in the CM frame, an isotropic angular distribution was used to model the different exit angles of the reaction products, giving the ^{31}S*(rest mass: 28.85727 GeV/c^2 + 1.25 MeV excitation energy) and the ^{4}He(rest mass: 3.728401GeV/c^2) opposite moving directions. The resulting energy-momentum 4-vectors were then boosted back to the lab frame. In a final step the particles' kinetic energies and momentum direction vectors were extracted and handed over to the corresponding particleGun.

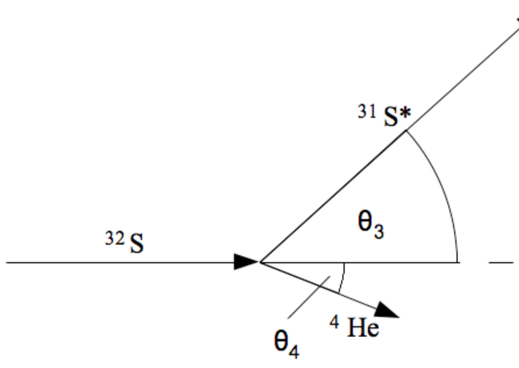

Figure 5.5: The reaction in the lab frame with arrows representing momentum vectors

The energy-momentum relation is the basis to the following calculations. It is

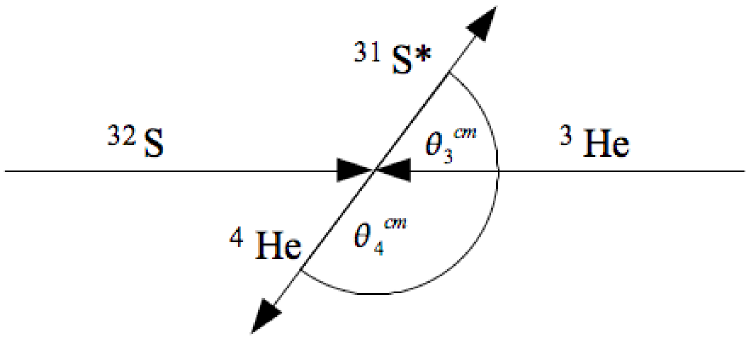

Figure 5.6: The reaction in the center of mass frame frame with arrows representing momentum vectors

valid in the center of mass frame and the lab frame.

$$E^2 = m^2c^4 + p^2c^2 \tag{5.3}$$

where E is the total energy of the particle, m its rest mass, $p = |\boldsymbol{p}|$ the momentum and c the speed of light. The definition of a energy-momentum 4-vector is:

$$\hat{p} := \begin{pmatrix} E \\ \boldsymbol{p}c \end{pmatrix}$$

From here on a hat like in $\hat{p}$ stands for a 4-vector, whereas a bold letter like $\boldsymbol{p}$ stands for a 3-vector. Equation 5.3 can be rearranged to calculate the momentum value and becomes

$$p = \frac{1}{c}\sqrt{E^2 - m^2c^4} \tag{5.4}$$

which gives us the momentum of the incoming ^{32}S. With the beam going in z-direction, the two lab frame 4-vectors are:

$$\hat{p}^L_{32} := \begin{pmatrix} E_{32} = T_{32} + m_{32} \\ 0 \\ 0 \\ p_z \end{pmatrix}, \quad \hat{p}^L_{He} := \begin{pmatrix} E_{He} = m_{He} \\ \vec{0} \end{pmatrix} \tag{5.5}$$

T stands for particle kinetic energy, E_X for total energy (index 32 for ^{32}S and index He for ^{3}He), m_X for the particles' rest masses and p_z for the momentum in z-direction. A transformation in the CM frame is recommendable, because here the momenta of the particles are the same, simplifying the problem. To transform to the center of mass frame, I started with the CM-Energy:

$$E_{cm} = \sqrt{(E_{32} + E_{He})^2 - c^2(\boldsymbol{p}_{32}^L + \boldsymbol{p}_{He}^L)^2} \tag{5.6}$$

The conservation of momentum in the CM frame says that $\boldsymbol{p}_{31}^{cm} = \boldsymbol{p}_{\alpha}^{cm}$. Together with equation 5.3 this gives us

$$E_{31}^{cm} - E_{\alpha}^{cm} = \frac{m_{31}^2 c^4 - m_{\alpha}^2 c^4}{E_{cm}} \tag{5.7}$$

with $E_{31}^{cm} = T_{31} + m_{31} + E_{exc}$ as the total energy of the excited nucleus, E_α as the total energy of the alpha particle, m_X as the corresponding rest masses and c as speed of light. Together with the simple relation

$$E_{cm} = E_{31} + E_{\alpha} \tag{5.8}$$

we obtain the particle energies after the collision in the center of mass frame:

$$E_{31}^{cm} = \frac{E_{cm}}{2} + \frac{m_{31}^2 c^4 - m_{\alpha}^2 c^4}{2E_{cm}} \tag{5.9}$$

and

$$E_{\alpha}^{cm} = \frac{E_{cm}}{2} - \frac{m_{31}^2 c^4 - m_{\alpha}^2 c^4}{2E_{cm}} \tag{5.10}$$

At this point int the simulation code, the angular distribution is called and creates a random vector pointing at a point on the unit sphere. I will address this issue later on.The direction vector is then multiplied with the momentum in the CM frame, again calculated with equation 5.3. At this point, the calculation in the CM frame is complete: The total particle energies and the momentum direction vectors are done. To transform the results from the center of mass frame to the lab frame, the former ones have to be Lorentz-boosted. The first step in doing that, is to determine the speed of the CM frame:

$$\boldsymbol{\beta}_{cm} = c\frac{\boldsymbol{p}_{32} + \boldsymbol{p}_{He}}{E_{32} + E_{He}} = \frac{\boldsymbol{p}_{32} c}{E_{32} + m_{He} c^2} \tag{5.11}$$

which gives us

$$\gamma = \frac{1}{\sqrt{1 - \beta_{cm}^2}} \tag{5.12}$$

With these two quantities we can transform the particle energies and the momentum vectors back to the lab frame:

$$E_{31}^{L} = \gamma(E_{31}^{cm} + \beta_{cm} c \boldsymbol{p}_{31}^{cm}) \tag{5.13}$$

$$E_{\alpha}^{L} = \gamma(E_{\alpha}^{cm} + \beta_{cm} c \boldsymbol{p}_{\alpha}^{cm}) \tag{5.14}$$

and for the momentum direction vector components:

$$\begin{gathered} p_{31,z}^{L} = \gamma p_{31}^{cm} \left(cos\theta_3^{cm} \frac{\beta_{cm} E_{31^{cm}}}{c p_{31}^{cm}} \right) \\ p_{31,xy}^{L} = p_{31,xy}^{cm} = p_{31}^{cm} sin\theta_3^{cm} \end{gathered} \tag{5.15}$$

$$\begin{gathered} p_{\alpha,z}^{L} = \gamma p_{\alpha}^{cm} \left(cos\theta_4^{cm} \frac{\beta_{cm} E_{\alpha^{cm}}}{c p_{\alpha}^{cm}} \right) \\ p_{\alpha,xy}^{L} = p_{\alpha,xy}^{cm} = p_{\alpha}^{cm} sin\theta_4^{cm} \end{gathered} \tag{5.16}$$

The angles θ_3^{cm} and θ_4^{cm} are as is depicted in figure 5.6. The momentum direction vectors are ready for use with the particleGun. With equations 5.13 and 5.14 and

$$T = E - mc^2 \tag{5.17}$$

the kinetic energies of the products is calculated. Now all quantities needed for running particleGun are complete. They are handed over to the functions and the simulation is ready to run. If the simulation is run now, we can see the particle trajectories in the visualization. The code doing these calculations can be found in the appendix. The results in Table 5.1 were calculated by GEANT in an example run. The results are the same as calculated by hand with an calculator.

p_z^{32}	2251.69 MeV/c
E_{cm}	32.5985 GeV
E_{31}^{cm}	28.8598 MeV
E_{α}^{cm}	3.73868 MeV
momentum in CM frame	277.051 MeV/c
β_{cm}	(0, 0, 0.0689093)
$\hat{p}_{31}^{L}$	(28.8598 GeV, 208.302 MeV/c, 7.50535 MeV/c, -182.515 MeV/c)
$\hat{p}_{\alpha}^{L}$	(3.73868 GeV,- 208.302 MeV/c, -7.50535 MeV/c, 182.515 MeV/c)
T_{31}^{L}	57.4883 MeV
T_{α}^{L}	31.7947 MeV

Table 5.1: Results of an example run

5.5 Implementation of the particle identification telescope

Modeling the particle identification telescope was the second goal of this thesis. This was done by splitting the the telescope into independent sensitive silicon detectors. To simulate a detector, four user classes are needed: The DetectorConstruction class for geometry, the sensitiveDetectorHit class to define the structure of a hit, the sensitiveDetector class to link the user classes of the GEANT4 Kernel and finally the EventAction to calculate the "measured"values and hand them over to the histoManager. The following list shows what was done to implement the thin ΔE silicon detector. The same was done for the thick silicon detector.

1. IS410DetectorConstruction class
   Like was explained in section 5.2, this is where the geometry is built and volumes are defined and registered with the GEANT kernel as detectors. The geometric properties are explained in section 5.3. The detector geometry is implemented in a way, that its position can quickly be altered by directly giving the geometric definition an angle and the distance from the thicker E-detector to the back of the target.

2. IS410DeltaEHitSD class
   In this class, the structure of a construct called a "Hit"is defined. For this silicon detector, a Hit saves deposited energy, the track ID of the particle to be able to calculate deposited energy of a certain particle and the particle position. A Hit is a snapshot of the physical interactions of a particle passing through the sensitive detector. Commands for the visualization of hits are included here, too.

3. IS410DeltaESD class
   In this class, there are two major functions: Initialize and ProcessHits. In the

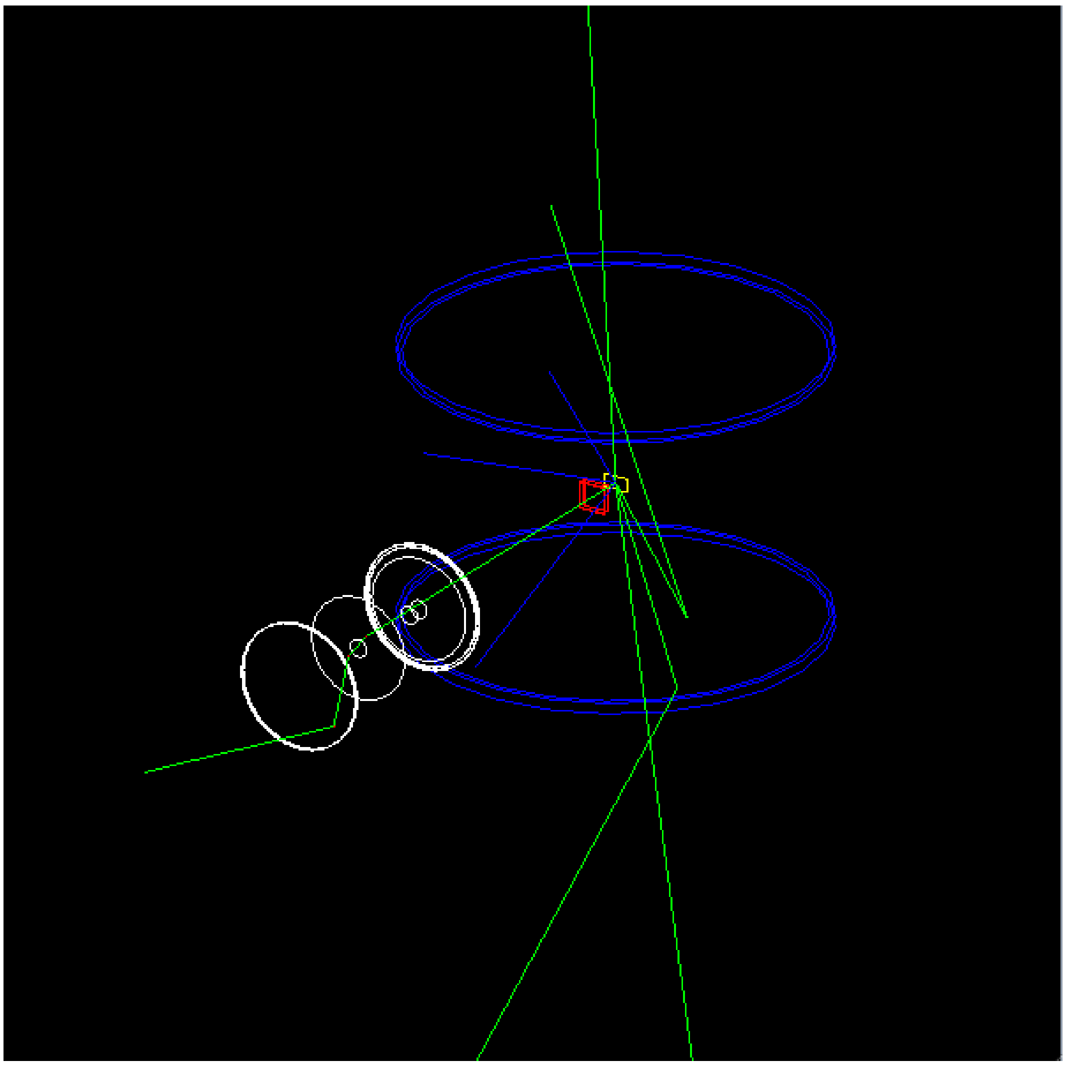

Figure 5.7: Visualization of the simulation with trajectories. The blue lines which do not belong to the geometry represent the alpha particles, the green ones are the gamma trajectories. The ^{31}S* trajectories can not be seen, as they are stopped inside the target. One sees nicely that the alpha particles are stopped by the target chamber and one gamma inside the HPGe detector. A total of 5 runs is shown here.

Initialize function, a so called HitsCollection is initialized and linked to the sensitive detector volume, is given a name and linked to the G4 kernel via an Event. Every detector has its own HitsCollection for every Event. In the second function, ProcessHits, a new Hit is created with data from a Step. In the

Hit the deposited energy, PostStepPoint position and the Track ID of the Step are saved. This Hit is then pushed into the HitsCollection of the ΔE detector.

4. IS410EventAction class
   This class has also two important methods: beginOfEventAction() and endOfEventAction(). The first method is invoked before an event and is used here to give the HitCollections of the different detectors an ID. The second method is invoked at the very end of event processing, and is therefore often used for first analysis. In this case, one object of every detector HitCollection was created and linked to the corresponding detector. Further than that, the total deposited energy of one event is calculated by summarizing over the energies of all hits in the detector. To make the modeling of the detectors more realistic, the deposited energy values of the single hits can be smeared, for example with a gaussian function. The obtained values are than handed over to the histoManager, used to create a root file with all the "measured"quantities.

From these root files, histograms and plots like the following can be created.

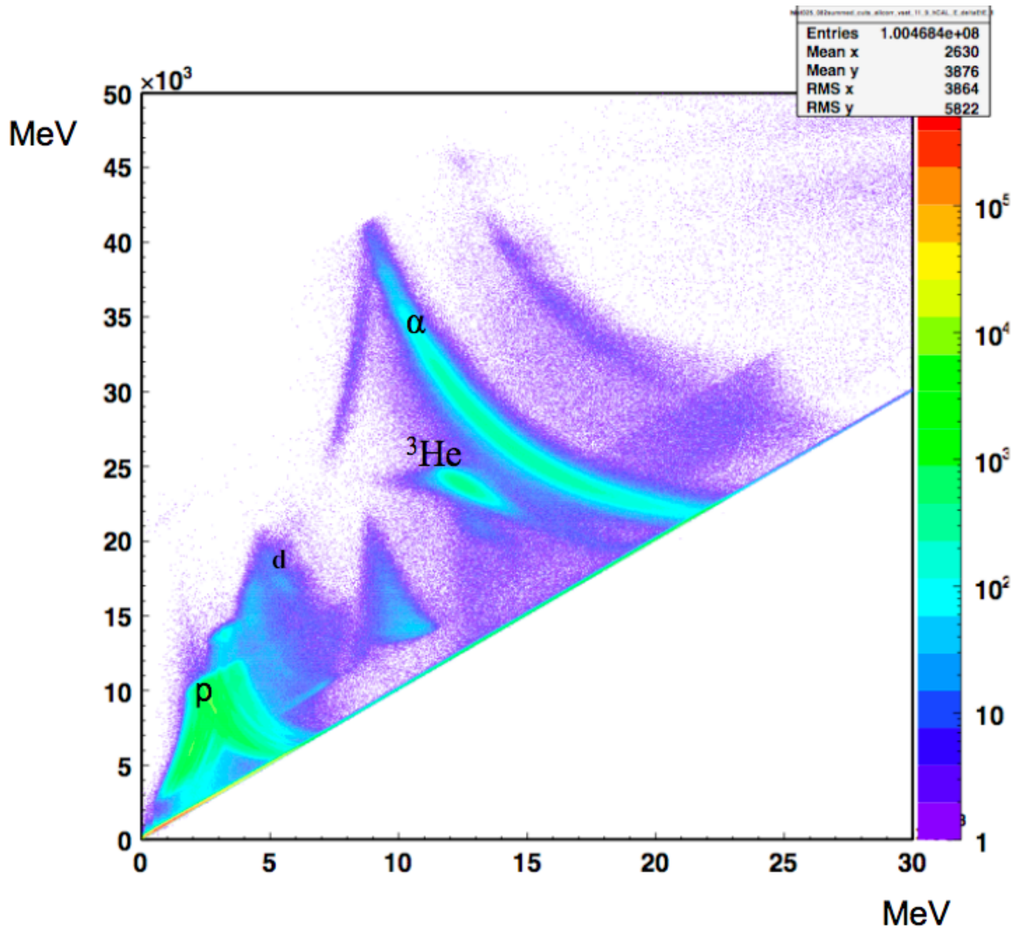

Figure 5.8: This is original data from the real experiment. The total energy is plotted against the energy deposition in the first silicon detector. One can see clearly the different spots and lines of the different particle types, with the thin green line representing α particles on the upper right. This plot was generously made available by Clemens Herlitzius.

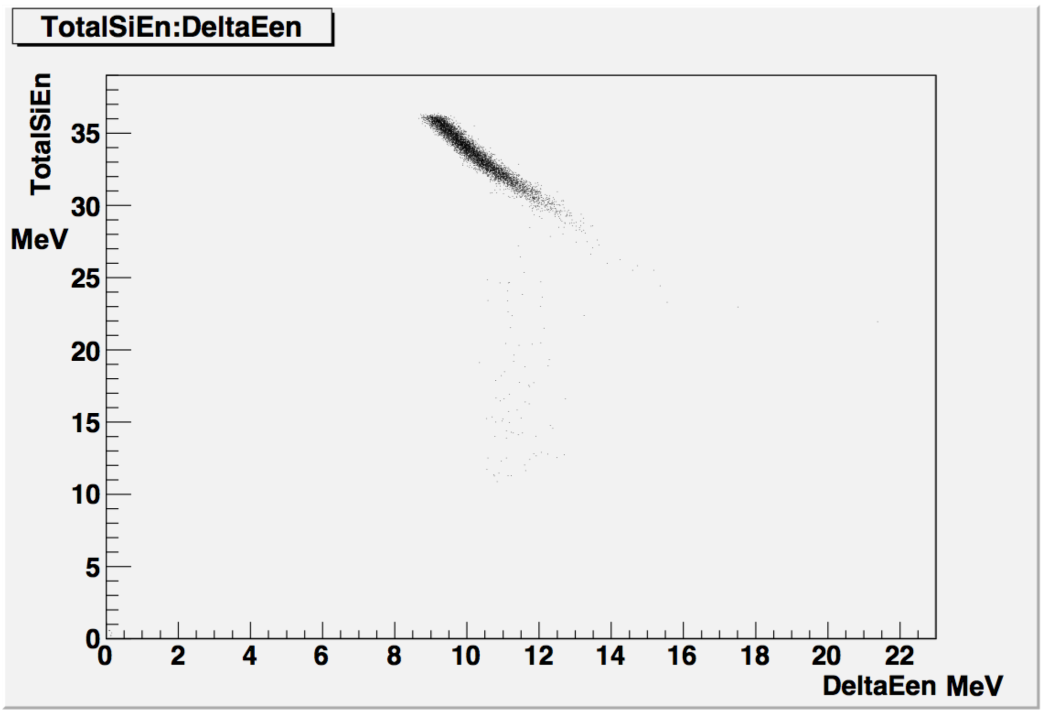

Figure 5.9: This is a result of the ΔE-E telescope simulation, including the target chamber. Again, the total energy is plotted against the energy deposited in the first detector. The shape and position of the α-particle line is good compared to the original data above. A closer look reveals that the plot seams to shows a similar diagonal threshold like the real data. For this plot only runs hitting all three detectors were used, 10000 in total.

5.6 Random isometric vectors on the unit sphere

It is important that the randomly created vector directions are not biased in any way. Otherwise the simulation itself would deliver biased results. What is needed, is a truly isotropic distribution. It is most practical to address this issue with polar coordinates on the unit sphere:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} sin\theta cos\phi \\ sin\theta sin\phi \\ cos\theta \end{pmatrix} \tag{5.18}$$

Isotropy means, that if we choose a random value for ϕ, this does not have any effect on the value of θ, hence no favored directions exist. The first idea that springs to mind is to give ϕ random numbers in the range of $[-\pi, \pi]$ and θ random numbers in the range of [0,π]. Unfortunately, this does not deliver the wanted results. The distribution is isotropic in the azimuthal ϕ distribution, but not for θ. With this simple solution the created vectors point mainly towards the poles of the uniform sphere because the area element is $d\Omega = sin\theta d\theta d\phi = -d\phi d(cos\theta)$. There are several solutions to this problem.

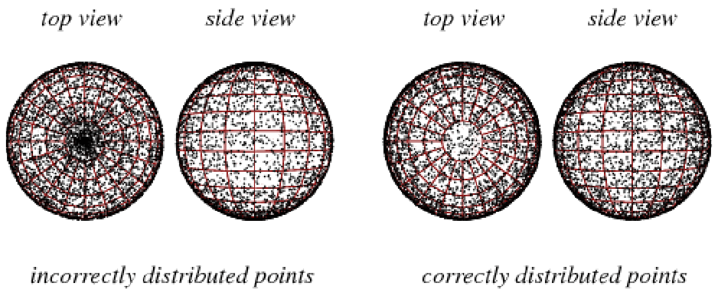

Figure 5.10: Creating random direction vectors is basically the same problem as sphere point picking illustrated here at [13]

If we take the area of a sphere $4\pi r^2$, the probability density function for uniform density over a sphere is $\frac{1}{4\pi r^2}$. This makes the cumulative density function

$$\iiint \frac{1}{4\pi r^2}\rho^2 sin\theta d\rho d\theta d\phi \tag{5.19}$$

This can be transformed by factoring the integrand in order to generate independent densities p(X)

$$\iiint \left[\frac{1}{2\pi}\right]\left[\frac{sin\theta}{2}\right]\left[\frac{\rho^2}{r^2}\right] d\rho d\theta d\phi \tag{5.20}$$

which leaves us with

$$\begin{aligned} p(\phi) &= \frac{1}{2\pi} \\ p(\theta) &= \frac{sin\theta}{2} \\ p(\rho) &= \frac{\rho^2}{r^2} \end{aligned} \tag{5.21}$$

Obviously $p(\rho) = 1$ as we integrate over the surface, which means that $\rho = r$. Now we only need to randomly generate θ and ϕ from their cumulative densities. With u

and v being independent uniform random numbers from the range [0,1], the angles are:

$$\begin{aligned} \phi &= 2\pi u \\ \theta &= arccos(1-2v) \end{aligned} \tag{5.22}$$

A second method is the one found by Marsaglia[13]. In his method, two independent random values u and v in the range from (-1,1) are chosen, but all values with $u^2 + v^2 \geq 1$ are rejected. The remaining points are used to calculate the cartesian coordinates of isotropically distributed points in three dimensions,

$$\begin{aligned} x &= 2u\sqrt{1-u^2-v^2} \\ y &= 2v\sqrt{1-u^2-v^2} \\ z &= 1-2(u^2+v^2) \end{aligned} \tag{5.23}$$

In this simulation a mixture of both alternatives was used: Like in the first method, the value for the azimuthal angle ϕ is randomly chosen from [-π,π]. To calculate a truly isotropic value for θ, two independent random numbers are chosen like in the second method. The first random number u, is pulled from the range [0,1] for values of $(sin\theta)$ and the second one v, is taken from the range $[0,\pi]$ for values for θ . Now all points for whom $u > sin(v)$ are rejected, basically comparing a uniform distribution of $(sin\theta)$ to the sin() value obtained by using a random θ. With this technique, the shape of the sin function was taken into account, delivering a distribution with no bunching at the poles. The allowed values of v are used as θ. Together with the random ϕ we have a isotropic distribution of the angles, which can then be used to calculate a vector with 5.18.

5.7 Gamma spectra

As we have seen in the section about the particle identification telescope, the results are reasonable and comparable to the real data. Exact analysis of the data will not be done in this thesis, but a simulated gamma spectrum is shown here, as well as real experiment data.

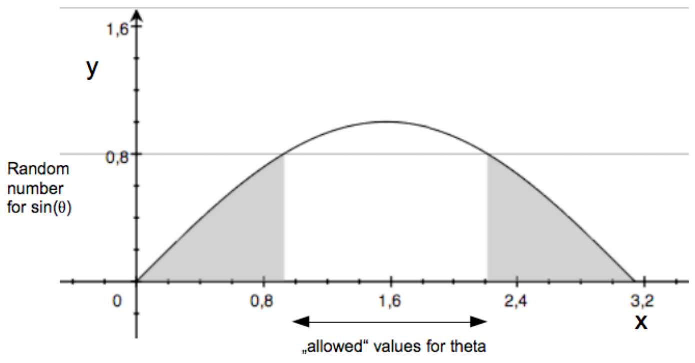

Figure 5.11: On the y axis a random value for sinθ is pulled. This defines the threshold for the allowed θ values on the x axis.

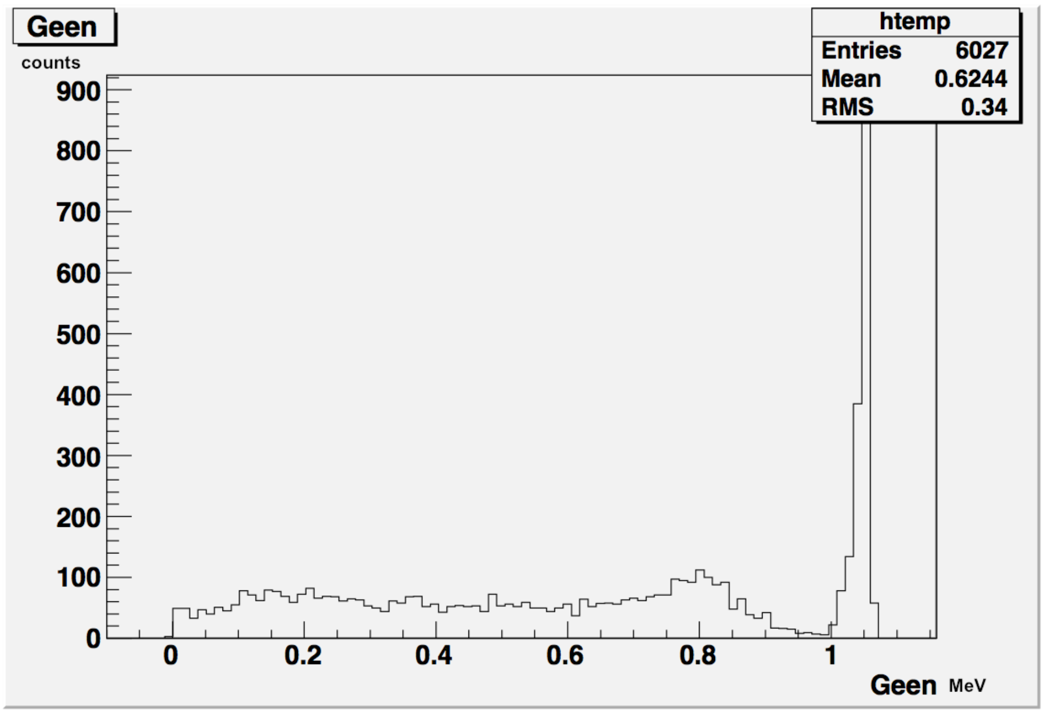

Figure 5.12: This is a simulated gamma spectrum with the target chamber included. The value for the peak position is too low compared to the real data. For debugging reasons, the code only produces γ-particles with 1 MeV maximum energy. This allows it too directly see the maximum doppler shift of β_{cm} = 0.069. At values of 0.85 MeV one sees the Compton edge typical for this type of detectors.

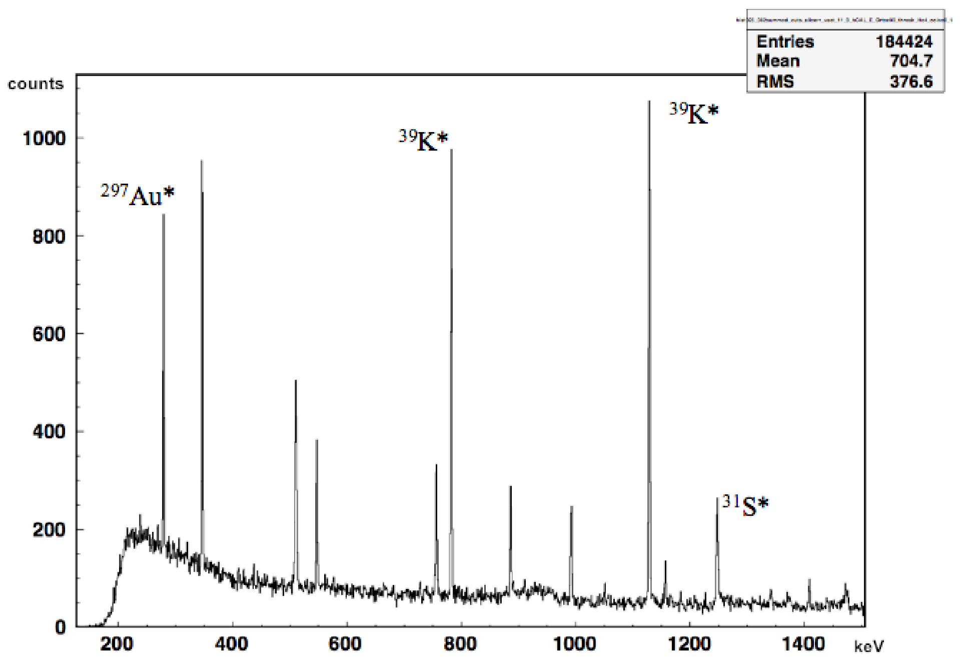

Figure 5.13: This is a measured gamma spectrum from the experiment. It is clear that this spectrum is more complicated than the simulated one, because the latter one does not simulate the reaction itself. Here we can see many peaks instead of only one. The ^{39}K* peaks come from the reaction. One can not effectively compare the two spectra. It was only included to give a complete picture. This spectrum is courtesy of Clemens Herlitzius.

Chapter 6

Conclusion

With this thesis I familiarized myself with the concepts of project oriented programming, the programming language C++ and the Monte Carlo simulation toolkit GEANT4, the DSAM method and the real experiment conducted at the MLL. The greatest challenge of this project was by far the GEANT4 software itself. Trying to understand the basic structure, with only knowing some C and bad documentation of the software, one was quickly lost in the jungle of functions and libraries. But with the help of the novice examples and the "Geant's Application Developer Manual"I learned step by step to use this toolkit and learned a lot about its versatility and possibilities. After the steep learning curve of the basic structure, I took an already existing code of the experiment in order to extend it. The first step, was to implement the new kinematics, putting more "physics"in the simulation. This first step went very well and comparisons between calculations done by hand and the values from the code corresponded nicely. The second step was to prepare the existing code for the implementation of several detectors instead of one, which was basically rearranging, renaming and cleaning of the code of no longer necessary artifacts. The final step of this thesis was to include the particle identification telescope. It was here that I ran into some difficulties, which costed me a lot of time but finally were solved with the help of Stefanie Klupp, whom I want to thank here. The simulation itself is still not completed and leaves room for further extensions and runtime optimization. As the experimental setup might be altered, the simulation must also be adapted. To get more information out of the simulation further detectors, for example another pair of silicon detectors or HPGe detectors, could be included and the functionalities of the installed ones extended. The simulated ΔE-E detector could be used for example, to double-check the inverse kinematics calculations, with which the the energies and exit angles of the reaction products are obtained from the detected position of the alpha-particle in the position sensitive silicon detector. In the simulation these properties are of course perfectly known. To enhance the performance of the code, it would be nice to "shoot"the gammas from the decaying nuclei only in the forward hemisphere, to increase the ratio of "hits in detector"/"particles shot".

Appendix A

More simulation results

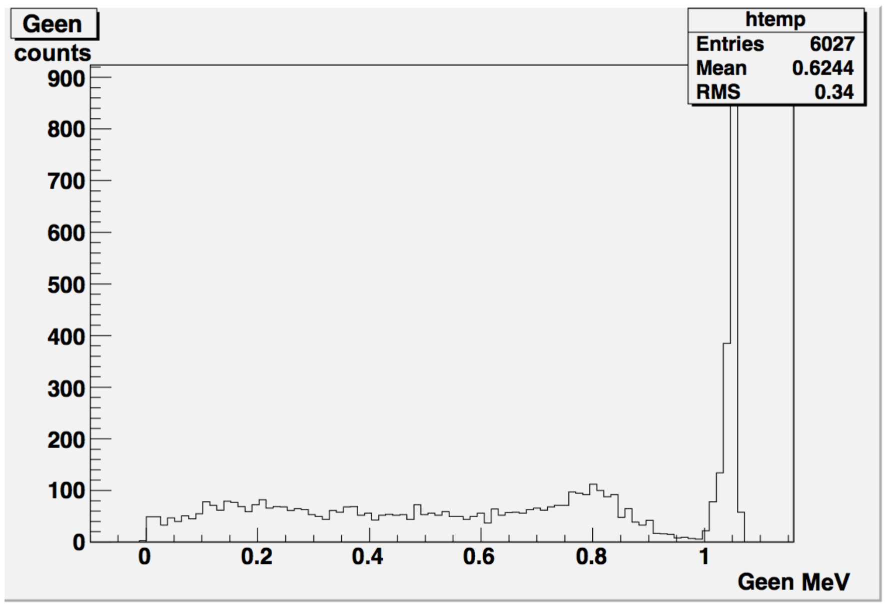

Figure A.1: This is a simulated gamma spectrum with the target chamber included. Coincidence between all three detectors.

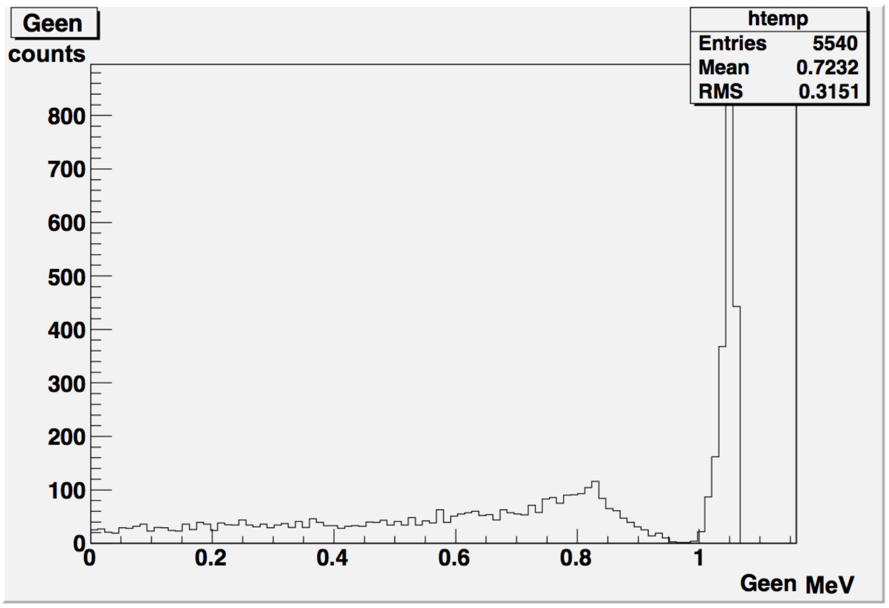

Figure A.2: This is a simulated gamma spectrum without the target chamber included. One sees that more hits were registered from the same amount of runs. Coincidence between all three detectors.

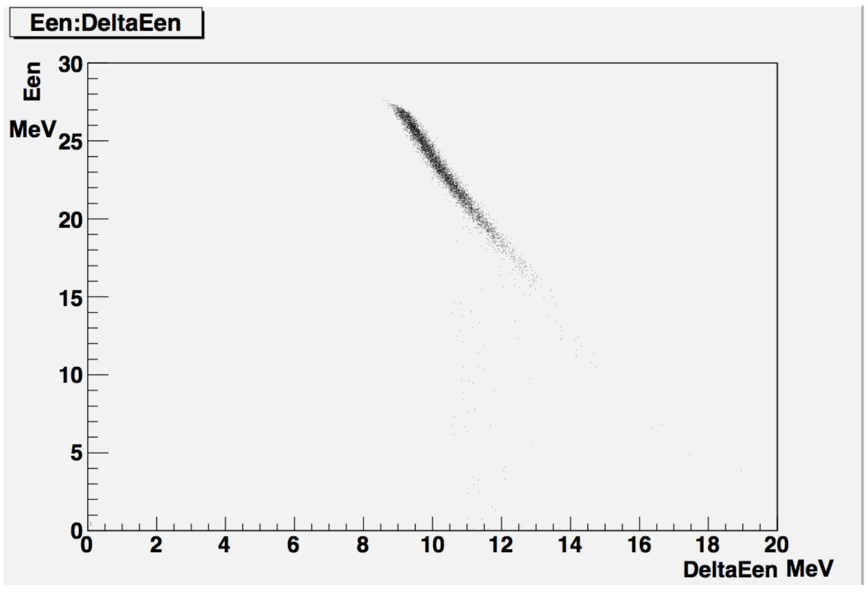

Figure A.3: This is a simulated ΔE-E plot without the target chamber included. Coincidence between all three detectors.

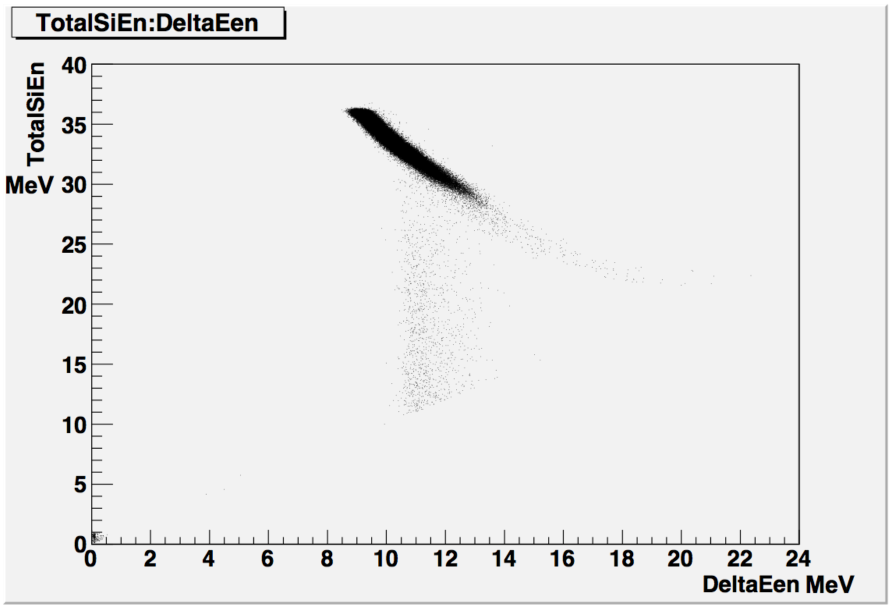

Figure A.4: This is a simulated ΔE-E plot without the target chamber included. All hits in the telescope are shown here. One sees the curved shape of the alpha particle line, similar to the one seen in the real data and the diagonal threshold.

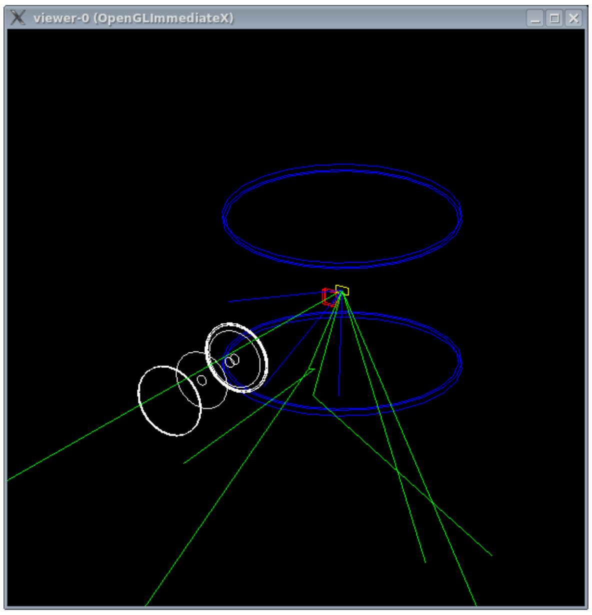

Figure A.5: Another visualization of the geometry and the trajectories of 5 runs

Appendix B

Relativistic kinematics code

```
//Experimental parameters

G4double dbeamenergy = 85*MeV;

//particle rest masses
G4double dm1= 29.78179*GeV; //sulfur 32
G4double dm2= 2.809413 * GeV; //He3
G4double dm3= 28.85727*GeV +1.249*MeV; //sulfur 31
G4double dm4= 3.728401*GeV; //He4/alpha

//energies of particles
G4double dECM;
G4double dECM3;
G4double dECM4;
G4double dELAB1 = dbeamenergy+dm1;
G4double dELAB2 = dm2;

//momentum vectors
G4double dzmom= sqrt((dbeamenergy+dm1)*(dbeamenergy+dm1)-pow(dm1,2));

if(check == true)
std::cout<<"...oo00@TEST1:AUSGABE:DZMOM@00oo..." << dzmom << std::endl;

G4ThreeVector pproj (0.,0.,dzmom); //projectile momentum vector
G4ThreeVector ptag; //fixed target momentum vector

//LAB 4-vectors
G4LorentzVector LLABproj (pproj,dELAB1);
G4LorentzVector LLABtar(ptag,dELAB2);
```

```
//Center-of-mass total Energy

dECM=sqrt(pow(dELAB1+dELAB2,2)-(pproj+ptag)*(pproj+ptag)); //ECM in MeV

if(check == true)
std::cout<<"...oo00@TEST2:AUSGABE:ECM@00oo..." << dECM << std::endl;


//CM-frame energies after collision
dECM3 = dECM/2 + ((pow(dm3,2)-pow(dm4,2))/(2*dECM));
dECM4 = dECM/2 - ((pow(dm3,2)-pow(dm4,2))/(2*dECM));

if(check == true)
std::cout<<"...oo00@TEST3:AUSGABE:ECM3/4@00oo..." << dECM3 <<"\t"<< dECM4


//angular distribution
const G4double THETA=M_PI; //values from 0 to M_PI
const G4double SIN_THETA=1.; // values from 0 to 1

G4double theta_gamma;
G4double sin_theta;

//random direction
G4double phi_gamma = RandFlat::shoot(-M_PI,M_PI);

do{
theta_gamma = RandFlat::shoot(0.,THETA);
sin_theta = RandFlat::shoot(0.,SIN_THETA);
}while(sin_theta > sin(theta_gamma));  //CONSIDERS COSINE SHAPE

//momentum direction CMS
G4ThreeVector momentumdir1  (0.,0.,1.);
G4ThreeVector momentumdir2 ;

momentumdir1.rotateY(theta_gamma);
momentumdir1.rotateZ(phi_gamma);

// set vector in opposite direction
momentumdir2.setX(-momentumdir1.x());
momentumdir2.setY(-momentumdir1.y());
```

```
momentumdir2.setZ(-momentumdir1.z());

if(check == true)
std::cout<<"...oo00@TEST4:AUSGABE:UNSCALED@00oo..." << momentumdir1<< "\t" <<
<< std::endl;


//momentum in CMS after collsion d(ouble)p(momentum)cm(frame)a(fter)c(ollision

G4double dpcmac = sqrt( dECM4*dECM4 - dm4*dm4) ;

if(check == true)
std::cout<<"...oo00@TEST5:AUSGABE:dpcmac@00oo..." << dpcmac <<  std::endl;

//scaling of vectors with CMS after collision momentum
momentumdir1 *= dpcmac;
momentumdir2 *= dpcmac;

G4LorentzVector LCM3 (momentumdir1,dECM3);
G4LorentzVector LCM4 (momentumdir2,dECM4);

if(check == true)
{
std::cout<<"...oo00@TEST6:AUSGABE:SCALED@00oo..." << momentumdir1<< "\t" << mo
std::cout<<"...oo00@TEST7:AUSGABE:LORENTZ@00oo..." << LCM3<< "\t" << LCM4 << s
}

// boosting
G4LorentzVector LLAB3;
G4LorentzVector LLAB4;


G4ThreeVector boostvec;
G4double dboost;

boostvec = LLABtar.findBoostToCM(LLABproj);

dboost = boostvec.getZ();

if(check == true)
std::cout<<"...oo00@TEST8:AUSGABE:BOOSTVEC@00oo..." << boostvec<< std::endl;
```

```
LLAB3= LCM3.boostZ(-dboost); //CM to LS needs negative beta
LLAB4= LCM4.boostZ(-dboost);

if(check == true)
std::cout<<"...oo00@TEST9:AUSGABE:LORENTZBOOSTED@00oo..." << LLAB3<< "\t"

// calculating LS kinetic energy
G4double dLSionT;
G4double dLSalphaT;


G4double d3LSenergy = LLAB3.e();
G4double d4LSenergy = LLAB4.e();

dLSionT = d3LSenergy - dm3;
dLSalphaT = d4LSenergy - dm4;
std::cout<<"...oo00@TEST9:AUSGABE:LSKIN@00oo..." << dLSionT<< "\t" << dLSa
//testing block
if(check == false)
{
G4double dtest1 = LLAB3.e();
G4double dtest2 = LLAB4.e();;

std::cout<<"...oo00@TEST10:AUSGABE:ENERGIES in MeV @00oo..." << dtest1/MeV

//G4double Ltest = test1*test2; //scalar product

//G4double Ltest2 = test1.angle(test2);  // angle between vectors

//std::cout<<"...oo00@TEST11:AUSGABE:LORENTZTEST@00oo..." << Ltest <<"\t"
}
```

Appendix C

Bibliography

[1] H.J. Wendker A. Weigert and L. Wisotzki. *Astronomie und Astrophysik*. 5. Wiley-VCH Verlag, 2009.

[2] A.E. Blaugrund. Notes on doppler-shift lifetime measurements. *Nuclear Physics 88*, pages 501–512, 1966.

[3] M.F. Bode and Aneurin Evans. *Classical Novae*. Cambridge University Press, 2 edition, 2008.

[4] Donald D. Clayton. *Principles of Stellar Evolution and Nucleosynthesis*, volume 2. The University of Chicago Press Chicago and London, 1983.

[5] CLHEP Collaboration. "clhep collaboration website", 7/2011.

[6] GEANT4 Collaboration. Geant4 main page, 2011.

[7] Janina Fiehl. Calibration of an hpge detector and geant4 simulation for a doppler shift attenuation experiment. Master's thesis, Technische Universität München, 2010.

[8] Wikimedia Group. Wikipedia article "classical nova".

[9] GEANT4 group at FNAL. Geant tutorial, 7/2011.

[10] Jan Tobochnik Harvey Goul and Wolfgang Christian. *An introduction to Computer Simulation Methods*. Number 0-8053-7758-1. Addison-Wesley, 3 edition, 2006.

[11] Prod. Jordi José and Prof. Alain Coc. Nucleosynthesis in classical nova explosions: Modeling and nuclear uncertainties. *AAPPS Bulletin*, 2006.

[12] Britannica online. Britannica online article "compound nucleus model", 7/2011.

[13] Eric W. Weisstein. "sphere point picking". from mathworld - a wolfram web source.