University of Padova

Department of Physics and Astronomy "Galileo Galilei"

Master Thesis in Physics

Study of the ^{26}Al destruction processes at energies of astrophysical interest

Supervisor
Prof. Marco Mazzocco
University of Padova

Co-supervisor
Dr. Marco La Cognata

Master Candidate
Francesco Andreis

Student ID
2090006

Academic Year
2024-2025

AI MIEI GENITORI

Abstract

The radioactive isotope of aluminum ^{26}Al was the first γ-ray emitter observed in our galaxy, playing a crucial role in understanding various nucleosynthesis processes in the Universe, especially within the framework of Multi-Messenger Astronomy [2]. Due to its half-life ($7.17 \cdot 10^5$ years), which is long on human timescales but extremely short in interstellar processes, its presence can be directly observed through the radiation produced by its decay. The presence of ^{26}Al reveals valuable information about the evolution of nuclei within the galaxy and supernova explosions. Moreover, meteorites have been found in our solar system with clear remnants of ^{26}Al decay, indicating that there was an injection of radioactive aluminum before the formation of "our" solar system [4].

Several space-born measurements and observations have been made regarding the presence of ^{26}Al in our galaxy, which has a total mass estimated at $2.0 - 3.6$ solar masses [3]. It has been shown that the production of this nucleus occurs predominantly in very massive stars. Specifically, stellar models suggest that the nucleosynthesis of ^{26}Al occurs during three distinct phases in the evolution of such stars: the hydrogen-burning phase in Wolf-Rayet stars, the convective carbon-shell burning phase, and the explosive neon/carbon burning phase before and during the core collapse in a supernova explosion [1] [7]. To explain the presence of ^{26}Al in the solar system, it has been hypothesized that this nucleus could have been produced within Asymptotic Giant Branch (AGB) stars, evolved, luminous, and cool stars characterized by a carbon and oxygen core [5]. The most widely accepted hypothesis today is that ^{26}Al was injected by a cosmic wind from a nearby AGB star or supernova into the protosolar nebula, playing a crucial role in the fusion, differentiation, and crust formation of planetary bodies in the early evolutionary stages of the solar system [34].

Two reactions of great interest to the nucleosynthesis of ^{26}Al are the neutron destruction channels $^{26}\text{Al}(n,\alpha)^{23}\text{Na}$ and $^{26}\text{Al}(n,p)^{26}\text{Mg}$. These reactions involve ^{26}Al in both the ground state and the isomeric state at 228 keV [6], which has a relatively short half-life ($T_{\frac{1}{2}} = 6.35$ s) and can be considered a different nuclear species in most cases of astrophysical interest. However, the reaction rates are still poorly known, and despite some studies showing a predominance of the $^{26}\text{Al}^{gs}(n,p)^{26}\text{Mg}$ reaction, a variation of a factor 10 in the reaction rates would lead to critical changes in the astrophysical field.

Since experimental data on nuclear reactions involving ^{26}Al are scarce and cover only part of the energy range of astrophysical interest (where temperatures are in the range of $1.1 - 2.3$ GK), a more in-depth analysis of the cross-sections of ^{26}Al$(n,\alpha)^{23}$Na and ^{26}Al$(n,p)^{26}$Mg is necessary. Research on neutron-induced reactions on unstable nuclei is extremely challenging, even when the unstable nucleus has a long half-life and can be used to produce a physical target. This difficulty is mainly due to the unavailability of ^{26}Al in nature (especially in pure form) and the challenges of producing a neutron beam, often resulting in experiments with non-negligible systematic errors and low statistics. A valid alternative to simplify the experiment in such cases is the so-called "Trojan Horse Method" (THM). In this work the THM is applied to the ^{2}H$(^{26}$Al$^{gs},\alpha^{23}$Na$)p$ and ^{2}H$(^{26}$Al$^{gs},p^{26}$Mg$)p$ QF reactions.

Contents

1 Introduction

Aluminum-26 (^{26}Al), a radioactive isotope with a half-life of approximately $7.2 \cdot 10^5$ years, serves as a critical tracer for ongoing nucleosynthesis within our Galaxy. The γ-rays emitted from the decay of ^{26}Al with an energy $E = 1809$ keV offer a unique “snapshot” of nuclear processes, as the Galaxy is relatively transparent to this form of radiation. High-resolution spectral measurements of ^{26}Al emission have confirmed that the regions emitting ^{26}Al co-rotate with the Galaxy, supporting a widespread origin rather than local sources [3]. The current mass of ^{26}Al in the Galaxy has been estimated at approximately 2.8 ± 0.8 solar masses, derived from the γ-ray emission detected across the Galactic plane.

Satellite observations across the galaxy (COMPTEL and INTEGRAL) have consistently shown that ^{26}Al is predominantly concentrated along the Galactic plane, suggesting that massive stars, which are distributed throughout the Galaxy, play a significant role in its production. In addition, meteorite studies have provided evidence of ^{26}Al enrichment in the early solar system, potentially due to nearby supernovae or other stellar events. However, the broad distribution of ^{26}Al across the Galaxy implies that it is primarily produced in star-forming regions rather than from isolated stellar events.

These findings have significant implications for our understanding of the star formation rate in the Galaxy, particularly for massive stars, which are the primary

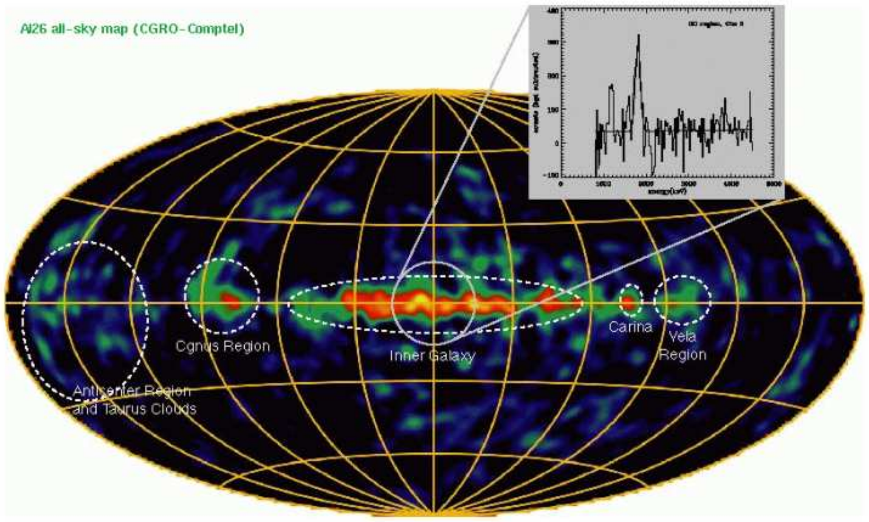

Al26 all-sky map (CGRO-Comptel)
Cgnus Region
Inner Galaxy
Carina
Vela Region
Anticenter Region and Taurus Clouds

Figure 1.1: Abundances of ^{26}Al in the all-sky map: as it is easily seen, ^{26}Al is more concentrated in the Galactic plane.

sources of ^{26}Al. The data suggest that the current rate of core-collapse supernovae events that contribute to the synthesis of ^{26}Al is approximately 1.9 ± 1.1 events per century [2]. This, in turn, provides an estimate for the overall star formation rate in the Galaxy, aligning with other methods used to study star formation in spiral galaxies similar to our own. From abundances of ^{26}Al, it is also possible to derive the neutron stars (NS) formation rate and, consequently, the expected NS merger and gravitational waves (GWs). Today, the neutron star formation rates derived from abundances of ^{26}Al and from observation of pulsars ($_{PSR} = 2.8 \pm 0.5$ events per century, to which we have to add RRATs, XDINs and Magnetars to reach the value of $10.8^{+7.0}_{-5.0}$) are different, so there is a necessity to better understand processes involving the formation and the destruction of ^{26}Al [2] [3].

The production and distribution of ^{26}Al in the Galaxy not only offer insights into stellar nucleosynthesis but also serve as a key tool for probing the large-scale dynamics and structure of the Milky Way. Understanding the origin and behavior of ^{26}Al can help refine models of Galactic evolution and the processes that govern the lifecycle of stars.

1.1 Level scheme of ^{26}Al

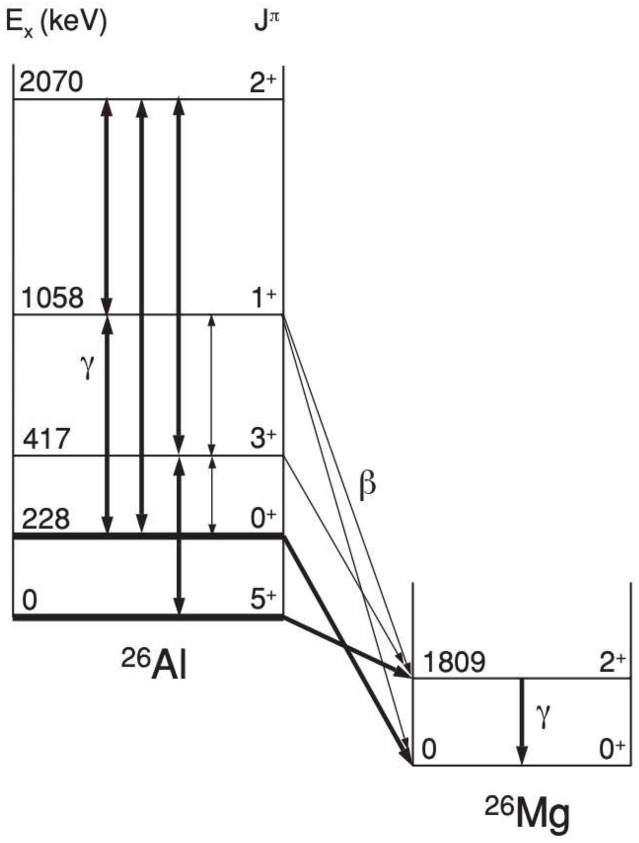


Figure 1.2: Level scheme of ^{26}Al. Energies and J$^\pi$-values are reported. The vertical arrows represent γ-ray transitions. In ^{26}Al, the thick vertical lines denote experimentally measured transitions, while the decay rates for the thin vertical lines have been estimated using shell model calculations.

Aluminum-26 has two main states: the ground state (^{26}Al^g) and an isomeric state (^{26}Al^m) with an excitation energy of 228 keV, as seen in Figure 1.2. The ground state decays via beta emission to magnesium-26 (^{26}Mg). The first excited state, then, emits the characteristic γ-ray at 1809 keV mentioned above, which has been observed in the Galactic plane.

The isomeric state, on the other hand, has a much shorter half-life of 6.35 seconds and decays directly to the ground state of ^{26}Mg without γ emission. Due to the significant difference in spin and parity between the ground and isomeric states, direct

γ-ray transitions between these two states are highly suppressed. However, at high temperatures, these states can achieve thermal equilibrium through intermediate excited states of ^{26}Al [1].

Thermal equilibration between ^{26}Al^g and ^{26}Al^m depends on the temperature of the stellar environment. At temperatures above approximately 0.45 GK, the two states are in thermal equilibrium, meaning their populations are governed by the Boltzmann distribution. Below 0.15 GK, the states are decoupled, and each decays independently according to its own half-life. In the intermediate temperature range, between 0.15 GK and 0.45 GK, the equilibration process is more complex, and the effective decay rate of ^{26}Al can differ significantly from the rates assumed for individual states [3].

Previous studies provided a basic framework for understanding the thermal equilibration of ^{26}Al, but more recent work has revealed that this process is more nuanced. Modern approaches utilize shell model calculations to estimate the γ-ray transition rates that facilitate equilibration between the ground and isomeric states. These findings indicate that accurate modeling of ^{26}Al synthesis and decay requires careful consideration of these thermal equilibration processes, particularly in environments with temperatures within the critical range.

1.2 Existing data

A missing piece of information is the experimental determination of nuclear reaction rates which influence the amount of ^{26}Al produced in different stellar sites. Major uncertainties to determine ^{26}Al abundances during convective C burning and explosive Ne/C burning in stars before and during core collapse are the neutron-induced destruction reactions (n, α) and (n, p).

In their sensitivity study, Iliadis et al. [6] conclude that present uncertainties of the ^{26}Al(n, p) reaction at stellar temperatures above 1 GK have the largest impact on estimates of overall ^{26}Al production in massive stars. Among the candidates for the pollution of the early solar system with ^{26}Al are asymptotic giant branch (AGB) stars. The contribution of low-mass (< 4 M$_\odot$) AGB stars to this process depends sensitively on how much ^{26}Al is destroyed by the neutron-induced (n, p) and (n, α)

reactions. This requires an accurate knowledge of these reaction rates for stellar temperatures around 0.3 GK.

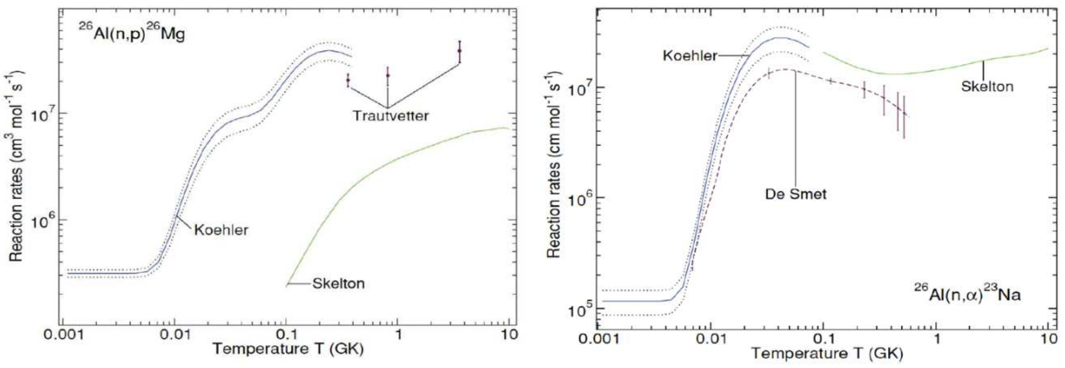


Figure 1.3: Left panel: the experimental reaction rates for the ^{26}Al$^{gs}(n,p)^{26}$Mg channel from [8] (blue line), from [9] (solid circles) and from [10] (green line). This latter result refers to the p_0 channel only, leaving ^{26}Mg in the ground state. Right panel: The experimental reaction rates for the ^{26}Al$^{gs}(n,\alpha)^{23}$Na channel from [8] (blue line), from [11] (purple line with error bars) and from [10] (green line). This latter result refers to the α_0 channel only, leaving ^{23}Na in the ground state.

The data on the ^{26}Al$^{gs}(n,p)^{26}$Mg and of the ^{26}Al$^{gs}(n,\alpha)^{23}$Na cross sections are quite scarce. All the available data were measured at quite low energies, below about 100 keV and often information on ^{26}Mg and ^{23}Na excited states contribution to the total cross sections are only partial (or inclusive). Finally, the few data sets often significantly disagree, though the situation is quite better at thermal neutron energy, which lays way outside the astrophysical region of interest. Figure 1.3 shows a summary of the available data for the experimental ^{26}Al$^{gs}(n,p)^{26}$M and of the ^{26}Al$^{gs}(n,\alpha)^{23}$Na reaction rates (the contribution of the ^{26}Al^{m} to the total cross sections and reaction rates is presently very uncertain, even two orders of magnitude in some temperature ranges) [5].

In more details, there is a significant disagreement among the different experimental data sets available in literature, as evidenced by the results from Koehler et al. [8] and De Smet et al. [11] for the (n,α) channel. Additionally, the cross sections reported by Koehler et al. [8] include either the p_1 or the α_0 channels only, which correspond to the population of the first excited state of ^{26}Mg and the ground state of ^{23}Na. Furthermore, the data from Trautvetter et al. [9] seem to conflict with the results from Koehler et al. [8]. Computational models, as could be seen in [5], also exhibit considerable variation depending on the parameters used.

This analysis of the existing data clearly highlights the need for new, more precise measurements of the cross sections for $^{26}\mathrm{Al}^{gs}(n,p)^{26}\mathrm{Mg}$ and $^{26}\mathrm{Al}^{gs}(n,\alpha)^{23}\mathrm{Na}$, as the current data are either questionable or, in the case of the high-precision data from De Smet et al. [11], limited to a single channel (the $(n,\alpha_0+\alpha_1)$ channel) and only partially cover the energy range of astrophysical relevance.

2
Astrophysical nuclear reactions

In this chapter the main features of nuclear reactions are briefly presented (see Section 2.1), focusing in particular on the peculiarities of the astrophysical ones (see Section 2.2) and those induced by neutrons (see Section 2.3).

2.1 Basic features of nuclear reations

A general nuclear reaction is expressed in the form:

$$A + x \rightarrow B + y \tag{2.1}$$

or, equivalently, in the condensed form $A(x, y)B$. In this notation, A is the target, x is the projectile, y is the ejectile and B is the residual recoil nucleus. Every nuclear reaction must conserve the energy, so it is useful to define the reaction Q-value:

$$Q = (m_x + m_A - m_y - m_B) \cdot c^2 = E_{k,y} + E_{k,B} - E_{k,x} - E_{k,A} \tag{2.2}$$

where m are the masses of the nuclei and E_k are the kinetic energies. $Q > 0$ means that the reaction is exothermic, so it releases energy, while $Q < 0$ stands for an endothermic reaction that needs energy in order to occur.

Thermonuclear reactions are important for the nucleosynthesis and the energy production in stars. A useful parameter to express the probability for a nuclear reaction to take place is the cross section σ, which is a quantitative measure defined as follows:

$$\sigma = \frac{\text{\# interactions per time}}{(\text{\# incident particles per area per time})(\text{\# target nuclei within the beam})}$$

$$\sigma = \frac{N_R/t}{[N_b/(tA)] \cdot N_T} \tag{2.3}$$

where σ is measured in barns $[1b = 10^{-24} cm^2]$. In reality, it is more useful to use the so-called differential cross section, which gives the amount of particles produced in the reaction as a function of the solid angle in which the detector is placed. Assuming an ideal detector able to detect every particle produced, it is possible to define the differential cross section as follows:

$$\frac{d\sigma}{d\Omega} = \frac{N_y(\theta)}{N_x \rho_A \Delta\Omega} \tag{2.4}$$

where $N_y(\theta)$ represents the number of y-type nuclei emitted at a specific polar angle θ, N_x denotes the number of incident particles of type x, ρ_A is the density of A-type nuclei present in the target, and $\Delta\Omega$ is the solid angle subtended by the detector.

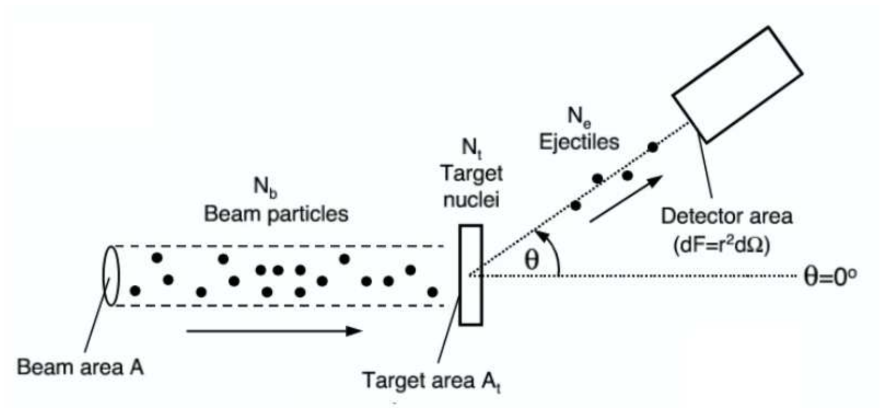


Figure 2.1: Schematic representation of the cross section of a reaction with a fixed target.

Both cross section and differential cross section help to determine another important quantity in the astrophysical context, which is instead the reaction rate, which is the number of reactions per time t and unit of volume V that occur in the stellar plasma. The reaction rate is defined as:

$$R_{By} = N_A N_x \sigma v \tag{2.5}$$

where N stands for number density of the interacting particles and v is the velocity of the projectile in a fixed target experiment or the relative velocity between the incident nuclei in a stellar plasma. Equation 2.5 can be easily obtained from Equation 2.3 through little manipulation. However, differently from an experiment, the particles in a star do not move with a fixated kinetic energy, but, since the stellar plasma under quiescent burning conditions can be described as a non-degenerate and non-relativistic gas of particles, follow a Maxwell-Boltzmann distribution [12]:

$$P(v)dv = \left(\frac{m}{2\pi kT}\right)^{3/2} e^{\frac{-mv^2}{2kT}} 4\pi v^2 dv \tag{2.6}$$

It is possible to substitute the values of v and $\sigma(v)$ in Equation 2.5 using the Maxwell-Boltzmann equation in 2.6:

$$R_{By} = (1+\delta_{Ax})^{-1} N_A N_x < \sigma v > \tag{2.7}$$

where δ_{Ax} is a factor: $\delta_{Ax} = 0$ if A≠x and $\delta_{Ax} = 1$ if A=x. It is also possible to write:

$$< \sigma v > = \int_0^\infty P(v) v \sigma(v) dv \tag{2.8}$$

Once the rate is determined, it becomes possible to obtain information on the time evolution of the abundances of the nuclei involved in a specific nuclear reaction. For a generic reaction $A(x,y)B$, the change of abundance of nuclei A caused by the bombardment of nuclei x is expressed as:

$$\left(\frac{dN_A}{dt}\right)_x = - < \sigma v > N_x N_A \tag{2.9}$$

The average lifetime of nuclei A depends, therefore, on the densities N_A and N_x and the reaction rate per particle pair. This last quantity depends on the temperature of the astrophysical environment. Moreover, the cross section depends on the energy, which in turn reflects the dependence of the abundance variation on the mechanism through which the reaction proceeds.

2.1.1 Centrifugal barrier

The strong nuclear force is an attractive force that binds nucleons together. The nuclear strong force has unique characteristics: it acts only at distances on the order of a femtometer (1 Fermi), comparable to the size of a nucleus, it is attractive at medium distances and repulsive at very short distances, preventing nucleons from collapsing into each other. In order to model the behavior of a neutral nucleon (i.e. a neutron) under this force, we use a central potential $V(r)$, which is only a function of the modulus of the radial distance r, and the time-independent Schroedinger equation for a single particle:

$$\left[-\frac{\hbar}{2m} + V(r)\right]\psi(\vec{r}) = E\psi(\vec{r}) \tag{2.10}$$

where $\hbar$ is the reduced Planck constant, m and E are respectively the mass and the total energy of the particle and $\psi(\vec{r})$ is the wave function of the particle. In order to isolate the radial part, we separate the wave function into a radial part and an angular part:

$$\psi(\vec{r}) = R(r)Y_{lm}(\theta,\phi) \tag{2.11}$$

where $R(r)$ is the radial wave function and $Y_{lm}(\theta,\phi)$ are the spherical harmonics that describe the angular dependence.

Substituting into the Schroedinger equation and focusing on the radial part, we obtain the radial Schroedinger equation:

$$-\frac{\hbar}{2m}\frac{d^2u(r)}{dr^2} + \left(V(r) + \frac{l(l+1)\hbar^2}{2mr^2}\right)u(r) = Eu(r) \tag{2.12}$$

where $u(r) = rR(r)$ and the term $\frac{l(l+1)\hbar^2}{2mr^2}$ represents the centrifugal potential due to the nucleon angular momentum. This term creates an effective potential that pushes the neutron outward if it has non-zero angular momentum.

It is possible to calculate the transmission probability P_l of such a barrier for a free neutral particle, such as a neutron: $P_l \propto E^{\frac{1}{2}+l}$ and hence $\sigma \propto E^{l-\frac{1}{2}}$ [35]. As a consequence we have:

- s-wave neutron capture usually dominates at low energies (except if hindered by selection rules);

- higher l neutron capture only plays role at higher energies (or if $l = 0$ capture suppressed).

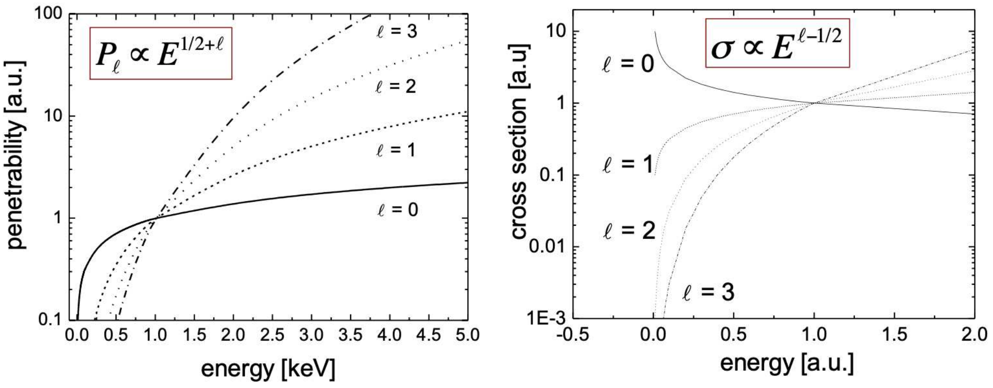


Figure 2.2: l dependence of penetrability through centrifugal barrier (left) and dependence of neutron capture cross section (right). It is clearly shown that at low energies lower l values dominate the reaction rate and that the cross section decreases strongly with decreasing energy.

2.1.2 Coulomb barrier

In the specific case of a nuclear reaction between two charged particles, it is possible to define another barrier that comes into play, which is the Coulomb barrier generated by the repulsion of the charges of the nuclei. In order to simplify the discussion, let us consider two nuclei A and x with charge Z_A and Z_x, respectively. If these nuclei are at distances larger than their nuclear dimensions $r_n = R_A + R_x$, they experience a Coulomb repulsion force and therefore the Coulomb potential energy is equal to:

$$E_C(r) = \frac{1}{4\pi\epsilon_0}\frac{Z_A Z_x e^2}{r} \tag{2.13}$$

At large distances ($r >> r_n$) the interaction between the two nuclei is essentially governed only by the electromagnetic force, while at distances comparable or smaller than the nuclear dimensions the interaction occurs only by means of the strong nuclear force. The net result is seen in Fig. 2.3.

In a classical picture, in order for the nuclear reaction to take place, the distance between the colliding nuclei has to be smaller than r_n, so the incident energy has to overcome the threshold value of:

$$E_{th} = 1.44\frac{Z_A Z_x}{r_n}\text{MeV} \tag{2.14}$$

Considering both stellar and primordial nucleosynthesis processes, the energy E of the incident nucleus can vary from 10^{-1} to 10^2 keV. This value is typically much lower than the Coulomb barrier between the two interacting nuclei. Therefore, from a classical point of view, the reaction should be impossible because crossing the Coulomb barrier is not permitted.
However, from a quantum point of view there exists a probability, small but finite, that such a crossing can occur also in the conditions where the relative energy between the interacting nuclei is smaller than the Coulomb potential energy. This phenomenon of penetration of the Coulomb barrier, known as "tunnel effect", is of fundamental importance for stellar processes. The same reasoning can also be applied to the centrifugal barrier, as long as $l > 0$.

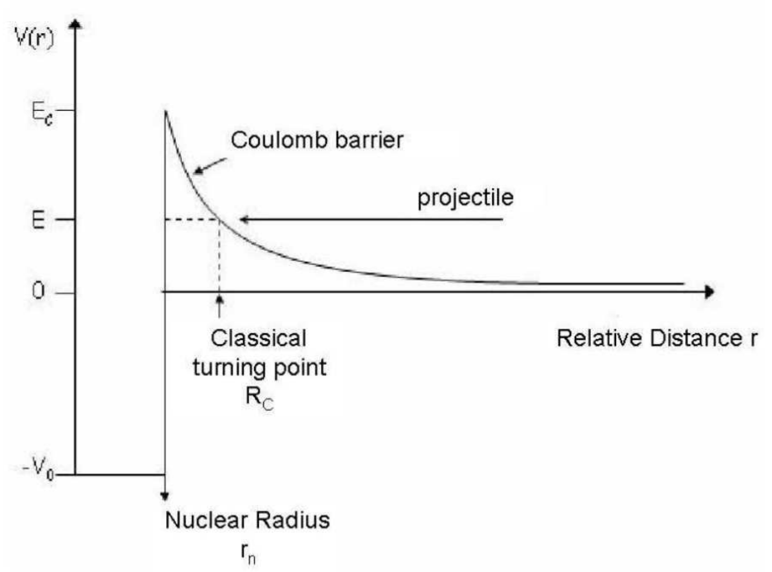


Figure 2.3: Total potential between two interacting nuclei as function of their relative distance r. The sharp shape of the Coulomb barrier around the nuclear radius r_n is only indicative.

2.1.3 Electron screening

The discussion so far has taken into account only bare nuclei, where the forces in play are only the obvious Coulomb repulsion plus, as seen in the previous subsection, the centrifugal barrier. In reality, for nuclear reactions studied in the laboratory, the target and the projectile are respectively neutral atoms or molecules and ions. This implies the presence of an electronic cloud around the interacting nuclei and therefore of an electronic shielding potential. This potential is called electron screening [14]. The presence of a negative electronic potential reduces the height of the Coulomb barrier between the two interacting nuclei, making it zero outside the atomic radius R_a. Since the presence of the electron shielding reduces the height of the Coulomb barrier, the probability of interaction between the two nuclei increases and so does the reaction cross section.

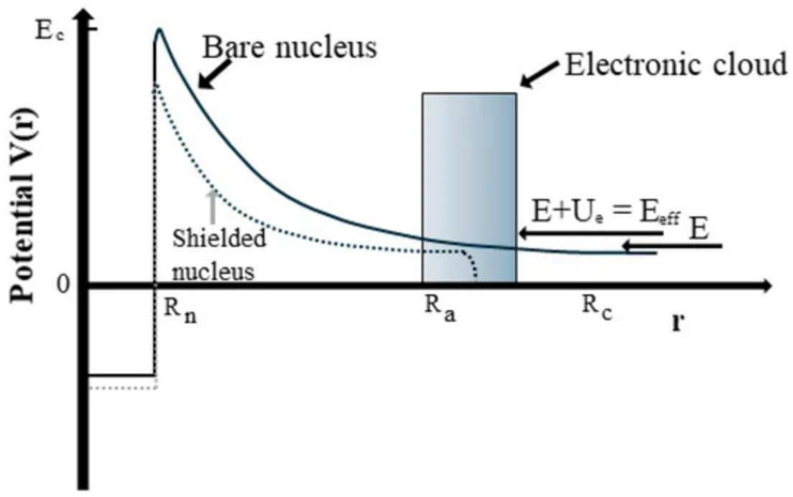


Figure 2.4: Behavior of the potential between charged particles: the presence of the electron cloud reduces the Coulomb barrier between the interacting nuclei. The electron screening effects cause an enhancement of the cross section.

2.2 Features of astrophysical reactions

Nuclear reactions play a fundamental role in astrophysical environments, being crucial for energy production, stellar evolution and nucleosynthesis. These reactions are typically referred to as thermonuclear reactions because a star contracts, converting gravitational energy into thermal energy until the temperature and density reach levels high enough to initiate these processes. In the stellar environment, under thermodynamic equilibrium, the velocities and energies of interacting particles follow the Maxwell-Boltzmann distribution. The temperatures, which vary depending on the star's mass and evolutionary stage, generally range between 10^6 and 10^9 K. By using the relationship between energy and temperature given by $E = k_B T$, where k_B is the Boltzmann constant, the energies at which thermonuclear reactions occur in stars can be estimated.

In this section, we will focus on the different types of astrophysical reactions and the ways to determine the cross sections of each of them. The particular case of reactions induced by neutrons will be treated in the following Section 2.3.

2.2.1 Non-resonant reactions

Non-resonant reactions are direct transitions from the initial state to the final state without the formation of an intermediate excited state. In this case, the probability of tunneling through Coulomb barrier for charged particle reactions at energies $E << E_C$, assuming full ion charges and zero orbital angular momentum, can be written as:

$$P_l \propto e^{-2\pi\eta} = e^{-\frac{b}{\sqrt{E}}} \tag{2.15}$$

where η is called Sommerfeld parameter and for a generic interaction between two charged nuclei, it can be written as:

$$\eta = \sqrt{\frac{\mu}{2E}} \frac{Z_A Z_x e^2}{\hbar} \tag{2.16}$$

and determines an exponential drop in the abundance curve, as can be seen in Fig. 2.5. The factor $2\pi\eta$ is also called Gamow factor and can be expressed as $2\pi\eta = 31.29 Z_A Z_x (\mu/E)^{1/2}$ with μ in amu and E in keV.
For non resonant reactions of the type $a + X \to b + Y$, the cross section can be written as:

$$\sigma \propto \pi\lambda^2_{DB} \cdot P_l(E) \cdot | < b + y|H|A + x > |^2 \tag{2.17}$$

where $\pi\lambda^2_{DB}$ is a geometrical factor that includes the particle's de Broglie wavelenght, $P_l(E)$ is the penetrability probability which depends on projectile angular momentum l and energy E, while $| < b + Y|H|a + X > |$ is the interaction matrix

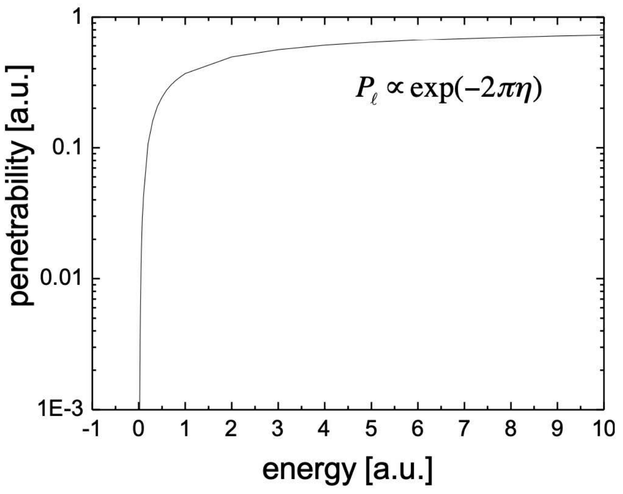


Figure 2.5: Penetrability, the probability of tunneling, as a function of energy: it is easy to see the exponential drop of such probability near 0.

element. An easier way to express this formula is given by the following relation [12]:

$$\sigma(E) = \frac{1}{E} e^{-2\pi\eta} S(E) \tag{2.18}$$

in which the first term has non-nuclear origin, but only strong energy dependence, whereas the second contains has nuclear origin and consequently a weak energy dependence. Equation 2.18 defines the so-called astrophysical S(E)-factor. If angular momentum is non zero, the centrifugal barrier has also to be taken into account.

Substituting Eq. 2.18 in Eq. 2.8, it is possible to obtain the following expression:

$$< \sigma v > \propto \int S(E) e^{-\frac{E}{k_B T} - \frac{b}{\sqrt{E}}} \tag{2.19}$$

which is a function that has its maximum at an energy E_G, that is called Gamow energy peak, that depends on the type of reaction and the temperature. Since for non-resonant reactions the astrophysical factor S(E) varies little with energy (as shown in Fig. 2.6) [15], the behavior of the Eq. 2.19 depends only the exponential term in the integral function.

The Gamow energy peak has a proper width and is the energy range in which the reaction of interest has the maximum probability of happening, so it identifies the

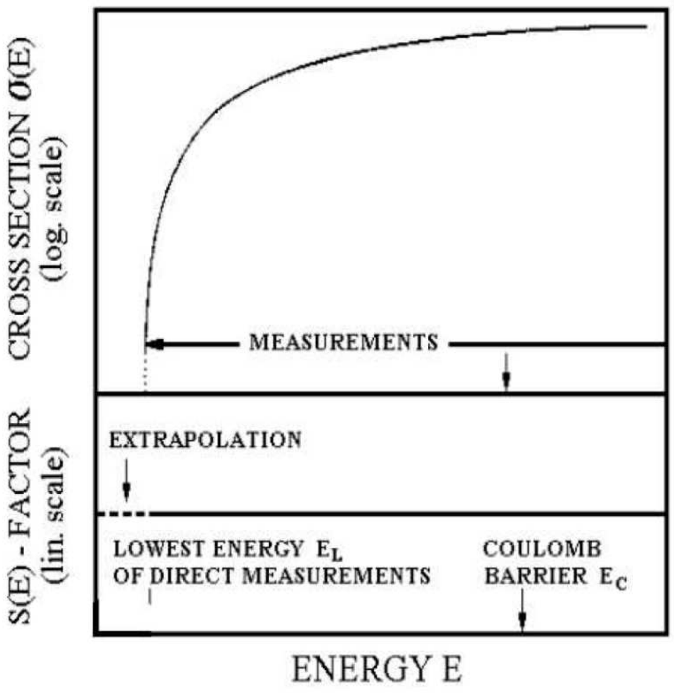

CROSS SECTION σ(E) (log. scale)
S(E) - FACTOR (lin. scale)
MEASUREMENTS
EXTRAPOLATION
LOWEST ENERGY E_L OF DIRECT MEASUREMENTS
COULOMB BARRIER E_C
ENERGY E

Figure 2.6: Development of the cross-section and astrophysical S-factor as energy varies. Figure taken from [12].

energy window of interest, as can be seen in Fig. 2.7.

$$E_G = \left(\frac{bk_BT}{2}\right)^{3/2} = 0.122(Z_A^2 Z_x^2 A)^{1/3} T_9^{2/3} \text{ MeV} \tag{2.20}$$

$$\Delta E_G = \frac{4}{\sqrt{3}}\sqrt{E_G k_B T} = 0.237(Z_A^2 Z_x^2 A)^{1/6} T_9^{5/6} \text{ MeV} \tag{2.21}$$

where A is the reduced mass. Thermonuclear reactions within a star's core primarily occur within an energy range centered around the Gamow energy peak. The width of this range depends on the nuclear species involved. This energy region is significantly lower than the Coulomb barrier (in quiescent burning, $E_G/E_C \sim 0.01 - 0.1$), making it extremely challenging to directly measure reaction cross sections at astrophysically relevant energies. Consequently, there are considerable uncertainties in determining $\sigma(E)$. In order to address this issue, the typical approach involves measuring S(E) over a broad energy range down to the lowest achievable labora-

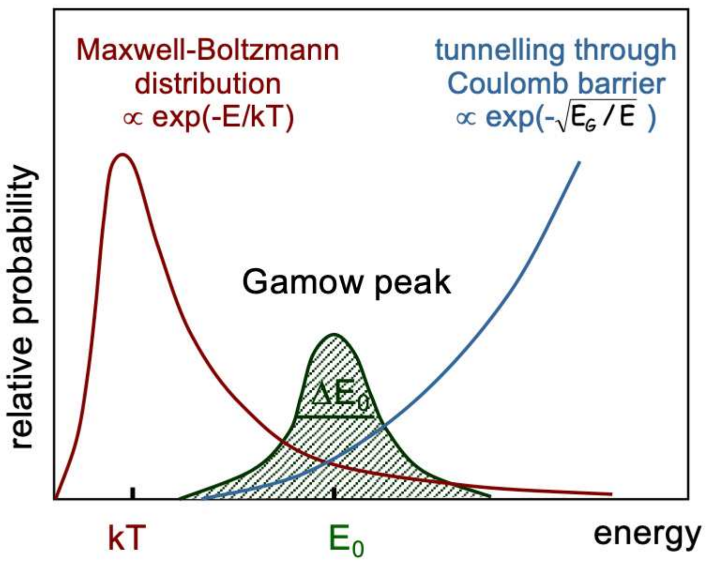


Figure 2.7: Representation of the Gamow peak as a convolution of the Maxwell-Boltzmann distribution and the probability of tunneling through the Coulomb barrier.

tory energies, and then extrapolating the results to astrophysical energies using the theory of nuclear reactions.

2.2.2 Resonant reactions

Resonant nuclear reactions occur when the energy of the incoming particles aligns closely with the excitation energy of a specific state in the intermediate compound nucleus. This situation can be described by the formula:

$$A + x \to C^* \to B + y \tag{2.22}$$

where C^* is the compound nucleus at an excited state. For a reaction to be resonant, the following condition must be satisfied [13]:

$$E_{CM} + Q = E_r \tag{2.23}$$

where E_{CM} represents the center-of-mass energy of the reacting particles A and x, Q is the reaction threshold energy, and E_r is the resonance energy corresponding to the excited state of C^*.

The probability of such reactions is described by the Breit-Wigner formula, which provides the resonant cross section σ_{BW} as a function of the energies involved [12]:

$$\sigma_{BW} = \pi \lambda\!\!\!^{-2} \frac{2J+1}{(2J_1+1)(2J_T+1)} \frac{\Gamma_1 \Gamma_2}{(E-E_r)^2 + (\Gamma/2)^2} \tag{2.24}$$

where:

- the first term is the geometrical fraction term proportional to $1/E$;
- the second term is a spin factor in which J is the spin of the compound nucleus, J_1 is the spin of the projectile and J_T the one of the target;
- the third term is strongly energy dependent: Γ_1 is the partial width for decay as "entrance channel configuration" (i.e. the probability of compound nucleus formation via entrance channel), Γ_2 is the partial width for decay as "exit channel configuration" (i.e. the probability of compound nucleus decay via exit channel), E_r is the resonance energy and Γ is the total width of the compound's excited state.

It is important to underline that partial widths are not constant but energy dependent. In fact they can be written as:

$$\Gamma_i = \frac{2\hbar}{R} P_l(E) \theta_l \tag{2.25}$$

where $P_l(E)$ gives strong energy dependence and θ_l is the so called "reduced width" that contains nuclear physics info.

It is possible to distinguish two simplifying cases of resonances:

- narrow (isolated) resonances;
- broad resonances.

When only narrow, isolated resonances (so that $\Gamma << E_r$) are near astrophysically relevant energies, they dominate the reaction rates, making it easier to approximate certain parameters, such as the Maxwell-Boltzmann distribution and the partial widths, as constant over the resonance region. The reaction rate can be expressed through the following formula:

$$< \sigma v >_{12} = \left(\frac{2\pi}{\mu_{12} k_B T} \right)^{3/2} \hbar^2 \left(\omega\gamma\right)_R \cdot e^{-\frac{E_r}{k_B T}} \tag{2.26}$$

$$\omega\gamma = \frac{2J+1}{(2J_1+1)(2J_T+1)}\frac{\Gamma_1\Gamma_2}{\Gamma} \tag{2.27}$$

where the rate is entirely determined by the "resonance strength" $\omega\gamma$ and the energy of the resonance E_r. The exponential dependence on energy means that the rate is strongly dominated by low-energy resonances ($E_r \to k_B T$) and that small uncertainties in E_r (even a few keV) imply large uncertainties in the reaction rate, as can be seen from Fig. 2.8.

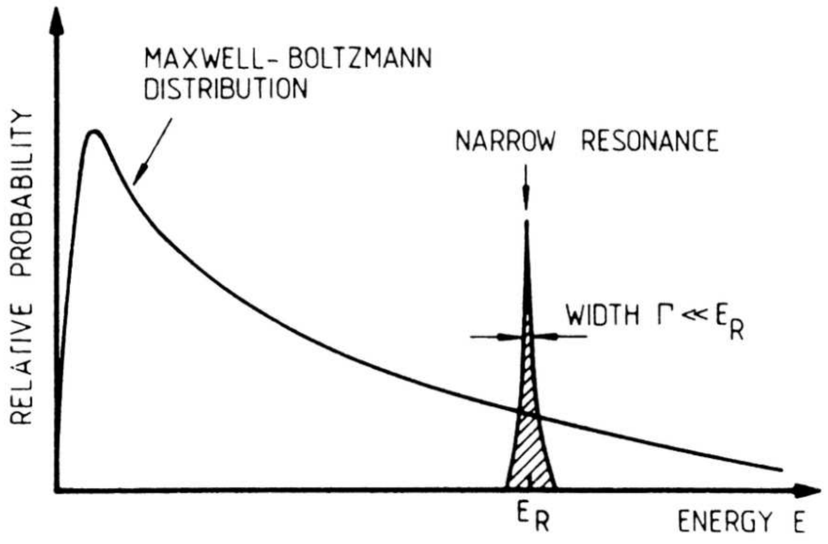


Figure 2.8: Isolated narrow resonance near the energy range of astrophysical interest. The Maxwell-Boltzman distribution is assumed constant over resonance region. The Partial widths are also constant $\Gamma_i(E) < \Gamma_i(E_r)$. Figure taken from [12].

In the case of broad resonances ($\Gamma \sim E_r$) the peak is broader than the relevant energy window for the given temperature. In this case the cross section can be determined using the Breit-Wigner formula seen in Eq. 2.24 with the energy dependence of partial and total widths, differently from the case of narrow resonances. There is also a possibility of overlapping broad resonances that give interference effects and resonances outside the energy range can also contribute through their wings.

2.2.3 Sub-threshold resonances

In nuclear astrophysics, sub-threshold resonances are a crucial phenomenon where the energy E_r associated with an excited state C^* of the compound nucleus is below the threshold energy E_{th} required to form C^* directly. Specifically, a sub-threshold resonance occurs if $E_r < Q$, where Q is the reaction's Q-value.
Although the resonance energy lies below the threshold, its influence extends above the threshold because of the energy width (Γ) of the state. This extension can provide a significant contribution to the nuclear reaction cross-section and, consequently, to the astrophysical S-factor. The physical basis for this lies in the tails of the resonance, which extend into the region of astrophysical interest, as depicted in Fig. 2.9. In this case, the contribution of such resonances is evaluated using the Breit-Wigner formula for cross-section calculations.

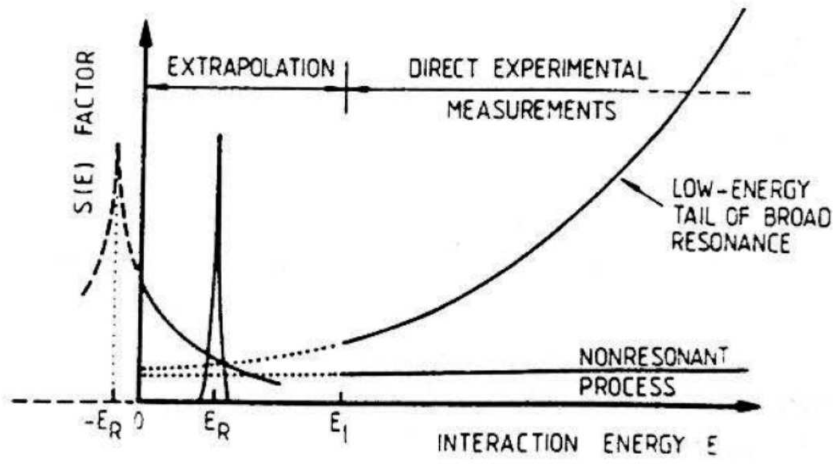


Figure 2.9: Example of sub-threshold resonance and its possible effects on the cross section in the energy range of astrophysical interest. Figure taken from [12].

2.3 Reactions induced by neutrons

Non-resonant neutron-induced reactions typically involve a two-particle exit channel, described schematically as $A(n,x)B$. As in the case of charged particles, these processes can be understood as two-step interactions, where an excited intermediate compound nucleus C^* is first formed and then decays into the observed products B

and x. Mathematically, the cross-section for such reactions can be expressed in a semi-classical form:

$$\sigma_{A+n} \propto \lambda_{DB}^2 | < B + x | H_2 | C >< C | H_1 | A + n > |^2 \tag{2.28}$$

where λ_{DB} is the de Broglie wavelength of the incident neutron, H_1 and H_2 are the interaction Hamiltonians for the entrance and exit channels, respectively.
The probability of forming the compound nucleus via neutron capture is encoded in the partial width $\Gamma_n(E_n)$ defined as follows:

$$\Gamma_n(E_n) \propto v_n P_{l_n}(E_n) \tag{2.29}$$

where v_n is the neutron velocity, $P_{l_n}(E_n)$ is the penetration probability through the centrifugal barrier, and l_n is the orbital angular momentum of the neutron.
In the low-energy regime (typically $E_n < 500$ keV) neutron-induced reactions are dominated by s-wave interactions ($l_n = 0$), as higher partial waves are suppressed due to the centrifugal barrier. For s-wave neutrons, the penetration probability $P_{l_n}(E_n)$ approaches unity, and the cross-section becomes inversely proportional to the neutron velocity:

$$\sigma(E_n) \propto \frac{1}{v_n} \tag{2.30}$$

This results in the reaction rate per particle pair is nearly constant at astrophysical energies, where thermal neutrons dominate ($E_n \sim k_B T$). The low-energy behavior of neutron cross-sections simplifies the determination of reaction rates in astrophysical environments. For non-resonant neutron-induced reactions, the constancy of σv_n ensures that reaction rates are less sensitive to temperature variations compared to reactions involving charged particles. However, as neutron energy increases, contributions from higher partial waves ($l_n > 0$) begin to play a role, and the cross-section acquires a mild velocity dependence.

As neutrons are particles without electric charge, they are not subject to the electromagnetic fields generated by the protons in the nuclei involved in the reaction, so they are more penetrating in the matter. Being subject only to the strong nuclear force generated by the other nucleons, neutrons can travel within matter even for a few centimeters without being detected. Neutrons interact with atomic nuclei giving origin to [16]:

- emission of secondary radiation (n, α), (n, γ), (n, p) or $(n, \text{fission})$, in particular for slow neutrons ($E_n < 100$ keV);
- Change in energy and direction of the neutrons, especially in the case of elastic scattering for fast neutrons ($E_n > 100$ keV).

2.3.1 Features of neutron induced reactions

Producing neutron beams for experimental studies poses significant challenges, primarily due to the unique characteristics of neutrons. Unlike charged particles, neutrons are unaffected by electromagnetic fields, making them difficult to transport or accelerate. Furthermore, using pure neutron targets is very difficult due to their short mean life.
In realizing neutron beams, the beam energy can only be manipulated indirectly, primarily through moderation, a process that slows neutrons down using materials like water or graphite.

In most neutron production methods, the emitted neutrons exhibit a Maxwellian energy distribution. Consequently, specialized techniques are required to determine the energy of each individual neutron involved in the reaction of interest. For two-body reactions in the final state, the conservation of momentum and energy can be utilized to select specific neutron energies by measuring their emission angles.
Another widely employed technique is the Time-of-Flight (ToF) method. This approach involves measuring the time a neutron takes to traverse a known distance, leveraging the relationship between kinetic energy and velocity, $E = \frac{1}{2}mv^2$. To achieve high precision in time measurements, flight paths typically extend over several tens of meters.

The experimental methods discussed in the previous paragraphs provide a basis for understanding the dynamics of reactions induced by neutrons, yet their application is often hindered by significant obstacles:

- as already said, neutron beams typically exhibit a Maxwellian energy distribution, making it necessary to use of advanced tagging systems to isolate neutrons of the desired energy, minimizing the influence of background neutrons and enabling precise reaction studies;
- in the case of mono-energetic neutron beams generated via binary reactions, achieving sufficient statistical data often requires long periods of time, given the relatively low flux of neutrons at specific energies;
- the ToF technique resolution is related on the length of the flight path: experimental setups using this technique often require lengths of tens of meters, which introduce additional logistical and instrumental complexities;
- neutrons can activate surrounding materials in the experimental environment, leading to the emission of secondary radiation.

As we will see in Chapter 3, the THM offers an innovative solution to some of these challenges by employing a virtual neutron source embedded within a Trojan Horse nucleus, such as deuterium. This approach provides several advantages:

- the method enables the study of neutron-induced reactions directly at astrophysical energies, bypassing the need for neutron beams and avoiding complications related to barrier penetration and electron screening effects;
- only a single beam energy is required to explore a wide range of center-of-mass energies: this eliminates the need for energy-tagged neutron beams, thereby streamlining the experimental design;
- while not yielding absolute cross-section values, THM allows for comparison with direct measurements via normalization procedures, ensuring consistency and reliability of the extracted data.

3
Trojan Horse Method (THM)

In order to overcome the challenges outlined in the previous chapter, several alternative methods have been developed in recent years: rather than studying the reaction of interest directly, these approaches rely on different techniques to derive the cross-section of astrophysical reactions indirectly. These techniques, collectively referred to as indirect methods, are each grounded in a specific reaction mechanism and are connected to the cross-section of astrophysical interest through a corresponding theoretical framework.

Among these techniques there is the so called Trojan Horse Method (THM). This approach utilizes a transfer reaction involving an unbound system to measure the cross-section of a two-body process. It is particularly effective for studying charged particle reactions at astrophysical energies, allowing the extraction of the astrophysical two-body cross-section from the quasi-free (QF) contribution of an appropriate three-body reaction. For charged particle-induced reactions, the THM [18] [19] is a robust indirect technique: the three-body reaction is induced at energies above the Coulomb barrier in the entrance channel, allowing the THM to bypass complications such as Coulomb barrier penetration and electron screening effects [17].
In fact, while direct measurement of the S-factor at ultra-low energies is notoriously challenging and necessitates the extrapolation of experimental data obtained at

much higher energies, the THM provides a direct way to extract the S-factor in the Gamow energy region without extrapolation. However, as the method does not yield absolute cross-section values, a normalization procedure based on direct measurements is required, making THM a complementary tool for investigating nuclear reactions in astrophysics.

In recent years, the THM has been extended to the study of neutron-induced reactions. This development leverages deuterium as a virtual neutron source, simplifying experimental setups for studying such reactions and enabling precise investigations of neutron-related nuclear processes.

3.1 Quasi-free breakup mechanism

In the study of the interaction between two nuclear systems, it is possible to distinguish two extreme cases: direct and compound nucleus reaction. In direct reactions, the interacting nuclei exchange nucleons without forming an intermediate state, resulting in a rapid interaction time ($\sim 10^{-22}$ s). On the other hand, compound reactions involve the formation of an intermediate excited nucleus, redistributing kinetic energy into internal excitation before decaying into final states.

The THM is based on the theory of direct nuclear reactions and in particular of QF breakup mechanisms [19] [20], that can be carried out using the Impulse Approximation (IA) [18] [19] [20] [21]. The QF mechanism is a specific subset of direct reactions where one of the reaction participants acts as a spectator, minimally interacting with the primary participants of the reaction of interest.
For the sake of simplicity, let us consider the reaction $A + a \to c + C + s$, where a (the target nucleus, in this case) can be described with high probability by the wave function of a cluster configuration. Let these clusters be x and s (in short, $a = x \oplus s$). At this point, the Impulse Approximation (IA) can be employed, which relies on the following three assumptions:

- the incident particle A never interacts simultaneously with both clusters of a;
- the interaction between the projectile A and x occurs as if x were a free particle, meaning that the presence of s does not influence the interaction;

- the binding energy of the clusters in a is negligible compared to the interaction energy between A and x.

Under these conditions, s serves as a spectator, retaining its original momentum distribution within a and not actively participating in the reaction. This framework enables the study of the two-body reaction $A(x, c)C$ through the three-body QF reaction $A(a, cC)s$. The reaction is sketched in Fig. 3.1.

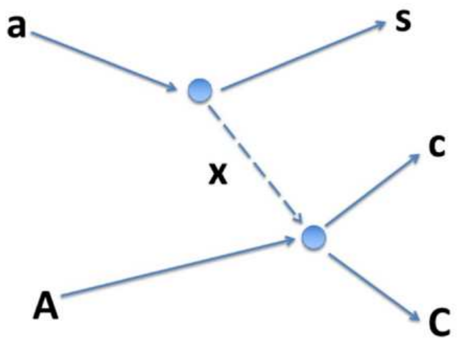


Figure 3.1: Diagram of the functioning of a generic nuclear reaction studied with the THM.

The nucleus a (called "Trojan Horse nucleus") is chosen because of:

- its large amplitude in the $a = x \oplus s$ cluster configuration;
- its relative low binding energy;
- its known $x - s$ momentum distribution $|\Phi(\vec{p_s})|$ in a.

3.2 Kinematical conditions

The QF mechanism imposes specific kinematic constraints that simplify reaction analysis:

- the momentum distribution of the spectator s post-breakup mirrors its original distribution in a, with a peak centered around zero momentum for s-wave interactions for example;

- the reaction products (c and C) emerge at characteristic angles (θ_c and θ_C, respectively) corresponding to QF kinematics. These "QF angles" are essential for selecting events dominated by the QF mechanism.

By detecting particles c and C at these angles, it is possible to isolate the QF contribution to the reaction, minimizing contamination from other mechanisms. One of this other mechanisms is the so called sequential mechanism: in some cases, the same final-state particles (s, c, C) can arise from sequential decay processes involving intermediate compound states. For instance, the formation and subsequent decay of a nucleus X_1 can mimic the products of a QF reaction, as seen in Fig. 3.2. These sequential mechanisms introduce a background signal that must be identified and subtracted during data analysis. The QF mechanism's unique kinematic conditions aid in discriminating it from these sequential processes.

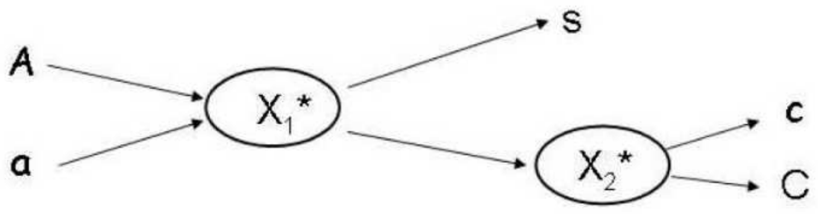


Figure 3.2: Formation and decay of intermediate states during the $A - a$ interaction: these kind of sequential mechanisms lead to the same particles in the exit channel, causing background for the event selection.

3.3 Features of the THM

The idea of the extension of the QF mechanisms to reactions of astrophysical interest was proposed by Baur in [18]. However, in that idea, it was foreseen that the Fermi velocity of the particle x in a, which is the velocity associated with the relative motion of one of the particles in the Trojan Horse nucleus, could partly compensate the energy of the incident nucleus a (see Fig. 3.3), which implies a relative energy E_{Ax} comparable to the energies at which thermonuclear reactions are triggered in the astrophysical environment. However, Baur's idea is very difficult to achieve experimentally. In order to overcome these problems, a measurement technique was devised using the low binding energy E_B of a nucleus to reach the astrophysical

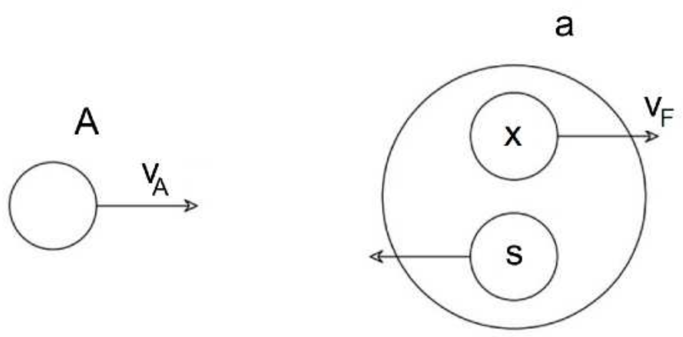


Figure 3.3: Sketch of a general reaction with a cluster particle: v_F is the Fermi velocity, so the relative velocity of x inside the cluster.

energies and exploiting the knowledge of QF mechanisms, developed in Catania in the 70s and 90s by researchers from Catania and Zagreb [19] [20]. In detail, it is possible to select the accessible energy region according to the relation:

$$E_{CM} = E_{Ax} - E_{B_{xs}} \tag{3.1}$$

where E_{CM} is the centre of mass energy of the 2-body astrophysically relevant reaction, E_{ax} is the center of mass energy for the reaction of the lower pole in Fig. 3.1 and $E_{B_{xs}}$ represents the binding energy of the system $x - s$. From the law of conservation of energy and in the hypothesis of post-collision prescription suggested by [23], the value of E_{CM} is:

$$E_{CM} = E_{Cc} - Q_{2\text{ bodies}} \tag{3.2}$$

where $Q_{2\text{ bodies}}$ is the Q-value for the two-body reaction $A + x \rightarrow C + c$.

3.4 THM formulation in Plane Wave Impulse Approximation (PWIA)

The quasi-free (QF) break-up mechanism, central to the Trojan Horse Method (THM), can be rigorously described using various theoretical formalisms. Among these, the Distorted Wave Impulse Approximation (DWIA) (Chant and Roos, 1977), the Modified Plane Wave Born Approximation (MPWBA) (Typel and Wolter, 2000), and the Plane Wave Impulse Approximation (PWIA) (Fallica et al., 1978) are most frequently employed. Each approach offers varying degrees of complexity and precision.

The DWIA is the most sophisticated formalism as it accounts for distortions in the momentum distribution of interacting particles. These distortions arise from phenomena such as Coulomb interactions between the nuclear fragments, which are significant at low energies. While the DWIA provides nuanced corrections, studies have shown that both the DWIA and the PWIA yield similar energy dependencies for the deduced cross-section when the recoil momenta of the spectator particle k_s are less than 100 MeV/c. This simplifies the problem significantly, as in such cases the PWIA becomes a viable and computationally efficient alternative [24].

The PWIA operates on three primary assumptions, collectively referred to as the impulse approximation (IA) already seen in Sec. 3.1 [24]. Under these assumptions, the triple-differential cross-section for the reaction $A+a \to C+c+s$ can be expressed in the PWIA as:

$$\frac{d^3\sigma}{d\Omega_c d\Omega_C dE_c} \propto (KF)\,|\Phi(\vec{p_s})|^2 \left(\frac{d\sigma_{Ax}}{d\Omega}\right)_{CM}^{HOES} \tag{3.3}$$

where:

- KF is a kinematic factor accounting for phase space population and depending on the observable variables. In particular:

$$KF = \frac{k_c k_C^2 E_s E_{CM}^2}{k_A E_x k_c E_s + E_c[k_c - k_A cos(\theta_c) + k_C cos(\theta_C - \theta_c)]} \tag{3.4}$$

  where k_i and E_i are, respectively, the wave numbers and the energy of the i$-$th particle involved in the reaction, while θ_i indicates the respective angles;

- $|\Phi(\vec{p_s})|^2$ represents the distribution of impulses, that is the Fourier transform of the wave function $\phi(\vec{r})$ of the relative motion of the clusters in a:

$$|\Phi(\vec{p_s})| = (2\pi)^{-2/3} \int_{-\infty}^{+\infty} \phi(\vec{r}) e^{-i\vec{k_s}} \vec{r} d\vec{r} \tag{3.5}$$

- $\left(\frac{d\sigma_{Ax}}{d\Omega}\right)_{CM}^{HOES}$ is the Half-Off-Energy-Shell (HOES) two body cross section for the reaction $A(x,c)C$ induced at energy E_{CM} in the centre of mass. The cross section is referred to as "half off-energy shell" because the participant particle x in the reaction is virtual [25]. This arises from the binding energy of the Trojan Horse nucleus, which disrupts the standard relationship between energy and momentum given by the mass-shell equation, $E_x = \frac{p_x^2}{2m}$. Under quasi-free (QF) conditions, the relative energy between a and x is instead determined by the relation:

$$E_{ax} = \frac{p_{ax}^2}{2m_{ax} - \epsilon_{ax}} \tag{3.6}$$

  where ϵ_{ax} represents the binding energy of the Trojan Horse nucleus. However, in the exit channel, this relationship is restored because the emitted c and C particles are real. In the context of the THM, the cross section derived for the virtual two-body reaction reflects only the nuclear component, as contributions from the Coulomb and centrifugal barriers are absent [19]. This allows a direct focus on the nuclear processes driving the reaction. The half-off-energy-shell cross section is expressed in arbitrary units. Consequently, it must be normalized using cross-section values obtained from direct measurements at suitable energies to ensure consistency and enable meaningful interpretation.

Reversing Eq. 3.3, we can obtain the HOES cross-section:

$$\left(\frac{d\sigma_{Ax}}{d\Omega}\right)_{CM}^{HOES} \propto \frac{\frac{d^3\sigma}{d\Omega_c d\Omega_C dE_c}}{(KF)\,|\Phi(\vec{p_s})|^2} \tag{3.7}$$

To transition from the HOES cross-section to the direct two-body cross-section, corrections for the Coulomb field effect are introduced using the penetration factor P_l [Cherubini et al., 1996]:

$$\frac{d\sigma_{Ax}}{d\Omega} \propto \sum W_l P_l \left(\frac{d\sigma_{Ax}}{d\Omega}\right)_{CM}^{HOES} \tag{3.8}$$

where P_l represent the transmission coefficient for the l^{th} partial wave and W_l is the weight coefficient through the barriers.

3.5 THM in ^{26}Al$^{gs}(n,\alpha)^{23}$Na and ^{26}Al$^{gs}(n,p)^{26}$Mg

This thesis focuses on the experimental measurement of the cross sections for the reactions $^{26}\text{Al}^{gs}(n,\alpha)^{23}\text{Na}$ and $^{26}\text{Al}^{gs}(n,p)^{26}\text{Mg}$ using the Trojan Horse Method (THM) applied to the quasi-free reaction $^{2}\text{H}+^{26}\text{Al}$. As outlined earlier, the aim of the experiment was to analyze the three-body decays $^{2}\text{H}(^{26}\text{Al}^{gs},\alpha^{23}\text{Na})p$ and $^{2}\text{H}(^{26}\text{Al}^{gs},p^{26}\text{Mg})p$, specifically considering only events consistent with the quasi-free mechanism required to satisfy the THM conditions for studying the corresponding two-body decays.

The deuteron was selected as the Trojan Horse nucleus because of its very low proton-neutron binding energy ($E_B = 2.2$ MeV) and the well-characterized momentum distribution of its clusters, determined through independent experiments [26]. Furthermore, it has been verified that the angular momentum coupling of the deuteron is predominantly in the s-wave state (96%) meaning that the cluster momentum distribution peaks around 0 MeV/c [27]. In particular its analytical form is given by the Hulthén function as expressed by the following equation:

$$\phi(\vec{r}) \propto \frac{e^{-ar} - e^{-br}}{e} \Longrightarrow F[\psi] \Longrightarrow \Phi(\vec{p}) \propto \left[\frac{1}{a^2 + p_s^2} - \frac{1}{b^2 + p_s^2}\right] \tag{3.9}$$

where $a = 0.2317$ fm^1 and $b = 1.202$ fm^1 are fixed [26]. The shape of the $p - n$ relative motion momentum distribution is sketched, in arbitrary units, in Fig. 3.4, while the reactions are illustrated schematically in Fig. 3.5.

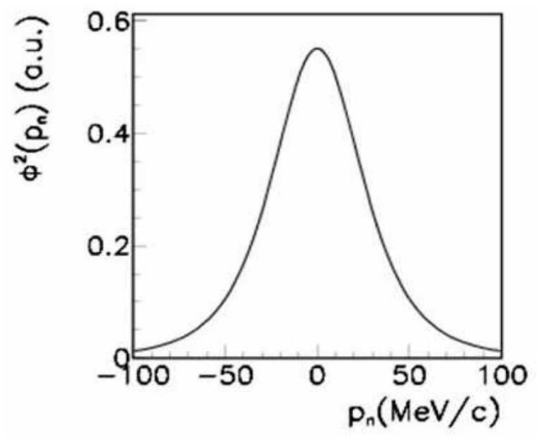


Figure 3.4: Momentum distribution for the $p - n$ relative motion inside the deuteron. The shape of such function is described by the Hulthén function.

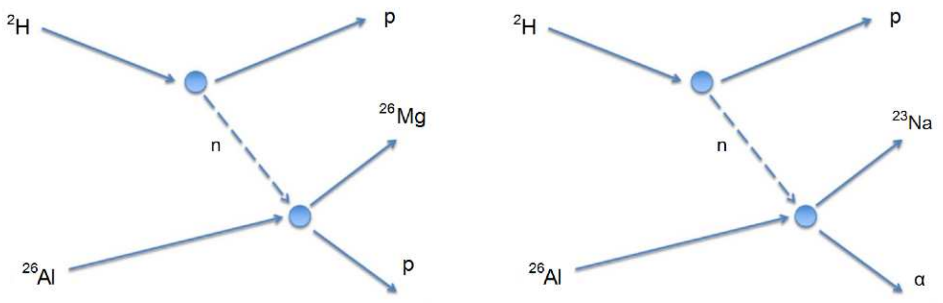


Figure 3.5: Sketch of the reactions analysed in this work.

4
Experiment and data analysis

4.1 Radioactive Ion Beam (RIB)

Experiments with radioactive ion beams (RIB) are generally characterized by low statistics, and during beam production, numerous technical and physical challenges must be addressed. The experiment under consideration, utilizing ^{26}Al, which is a metastable nucleus with a very long half-life, will not face issues related to short half-lives. However, this does not mean that other equally significant problems do not need to be taken into account during the experiment design phase.

In particular, it is essential to ensure that the chosen reaction mechanism provides the highest possible cross section for the process of interest. This goal is achieved by optimizing the combination of projectile and target, intensity of the primary beam and the power dissipated in the target. Additionally, beam production must be selective to avoid contamination by unwanted nuclear species.

4.1.1 The ISAC facility at TRIUMF

The experiment was conducted at the TRIUMF laboratory, located in Vancouver, Canada, where the ISAC (Isotope Separator and ACcelerator) facility employs the Isotope Separation On-Line (ISOL) technique for the production of radioactive ion beams. An accelerator, called a driver, delivers high energy light projectiles, the primary beam, toward a thick target. The light projectiles, protons or light ions, interact with the target nuclei producing neutral radioactive isotopes. These neutral atoms are then transported into a source where they are ionized and extracted at source potential. The radioactive ions are magnetically separated and post accelerated to reach the final energy requested [28] [29]. The ISOL method produces high quality emittances but the complicated and relatively slow process reduces the possibility of extracting isotopes with few ms half-lives. However, in this case, as mentioned above, the half-life of ^{26}Al is long, so this is not an issue.

In the TRIUMF facility, the primary beam is produced through a H^- cyclotron, which is the largest cyclotron in the world and has operated for almost 35 years. It accelerates H^- ions up to an intensity of 250 μA to a maximum energy of 500 MeV. The H^- are then stripped and protons are extracted in three different beam lines at different energies, the maximum energy being 500 MeV. One of these beam lines is dedicated for the ISAC radioactive beam production. In this case the beam is extracted at 500 MeV and up to 100 μA [28].

The ISAC facility has two independent target stations. This allows service on one target station while producing and delivering radioactive beams with the other. Each target station is composed of five modules. The entrance module houses the diagnostic and protection monitors for the proton beam. The target module contains the target and the source: this module is routinely removed to change both target and source. Three target modules are available (Nb, SiC, or Ta, depending on the experimental requirements). The beam dump module is located downstream of the target module. The last two are the extraction modules housing the optics elements. They are oriented perpendicular to the proton beam direction.
Downstream of the targets there is a common preseparator. The target modules and preseparator are inside a concrete shielded area. The preseparator reduces the radioactivity transported outside the shielded area in the downstream beam line. After the preseparator the RIBs are selected using the mass separator. This device is installed on a biased platform to increase the resolution [29].

The first stage of acceleration uses a radio frequency quadrupole (RFQ) acting as an injector. The RFQ boosts the energy from 2 keV/u to 150 keV/u. It can

accelerate mass to charge ratio of $3 \leq A/Q \leq 30$. The eight meter long resonant structure is composed of nineteen split rings supporting the electrodes. Part of the beam transmitted but not accelerated is stopped into a fixed collimator downstream of the RFQ. After the RFQ the charge state of the ions is increased by stripping the ions through a thin carbon foil (4 μg/cm^2) [29].
After that, the Drift Tube Linac (DTL) is a variable energy machine accelerating the beam in the entire range of energies 150 keV/u $\leq E \leq$ 1.8 MeV/u. The DTL is also used as an injector for the ISAC-II superconducting (SC) linac. The SC linac is at present composed of five cryomodules. Each cryomodule houses four superconducting cavities and one superconducting solenoid, allowing the production of RIBs with energies up to 5-11 MeV/u [28].

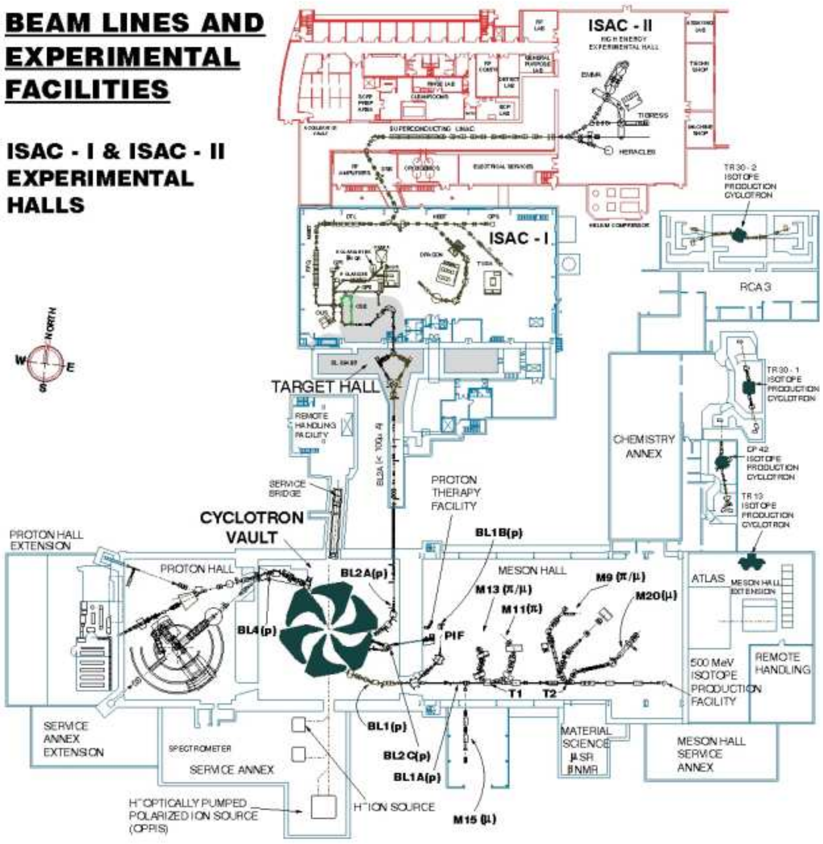


Figure 4.1: Overview of the TRIUMF site. Figure taken from [29].

4.1.2 ^{26}Al beam features

To investigate the (n, p) and (n, α) reactions of interest, it has been used a radioactive ^{26}Al^{gs} beam, accelerated to 3.5 MeV/u and directed towards the TUDA reaction chamber. This beam impinges on a thin CD$_2$ target with a thickness of 116 μg/cm^2. In this setup, the deuteron undergoes breakup, generating a neutron (participant) and a proton (spectator) [25]. Only the reaction products originating from the excited ^{27}Al*, specifically above the proton and alpha separation thresholds, will be detected. However, there are a number of considerations that need to be done:

- Potential beam contamination from the ^{26}Al isomeric state does not pose a significant challenge as angular distributions will be measured. This allows differentiation between the contributions from the 0+ excited state and those from other states due to their distinct angular patterns.
- In order to achieve measurements at angles as small as 1°, the maximum beam intensity is constrained to $2 \cdot 10^6$ ^{26}Al^{gs} per second.
- The CD$_2$ target thickness balances the required angular and energy resolution with the expected yield. At the proposed beam energy of 3.5 MeV/u, energy loss within the target is negligible (1.4%), ensuring high fidelity in the Q-value reconstruction [25].

4.2 Experimental setup

Since the intensity of the Radioactive Ion Beam is of the order of 10^6 particles per second (pps), designing a high-performance detector requires the following key features:

- identification of charge and mass for all reaction products, with the highest possible energy resolution;
- wide solid angle coverage to compensate for the low RIB intensities and enable coincident detection of particles emitted simultaneously at large relative angles;
- high segmentation to improve angular resolution for the detected particles.

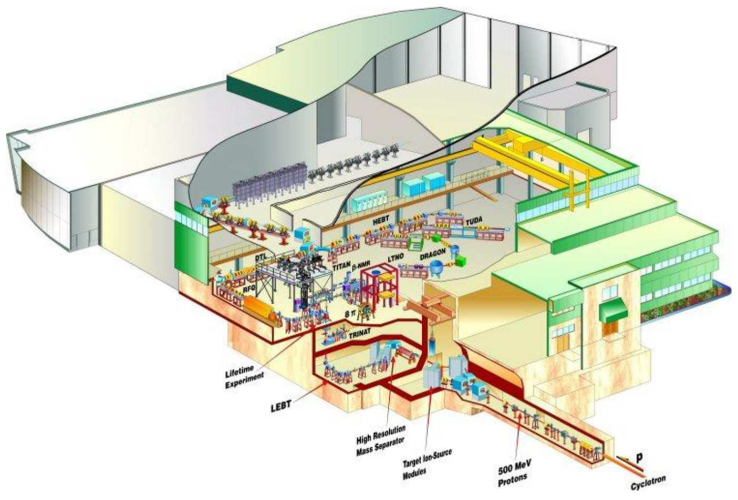


Figure 4.2: ISAC-I facility in TRIUMF. Figure taken from [28].

The energy deposition of a particle traversing a thin material layer depends significantly on the layer material and is strongly influenced by the particle charge and mass. This makes the first requirement achievable through the use of at least two-stage telescopes for particle detection. To meet the second and third requirements, Double-Sided Silicon Strip Detectors (DSSSDs) can be arranged in a suitable configuration around the target.

4.2.1 Target

A significant issue that can arise, and for which there is no simple remedy, is the case where the incoming ^{26}Al beam is not well collimated. Since the experiment is conducted in inverse kinematics, the heaviest reaction product is emitted at very small forward angles. This is highly favorable for detection efficiency under optimal conditions but can become problematic if the beam deviates by a few degrees from its nominal direction as it strikes the target.

Using a CD_2 target with a surface density of 116 μg/cm^2, simulations indicate that the expected count rate for the innermost strip of the detectors in our setup is approximately 150 counts per second for a beam intensity of $2 \cdot 10^6$ ^{26}Al atoms per second. Although such a rate is well within the tolerance of the silicon telescopes

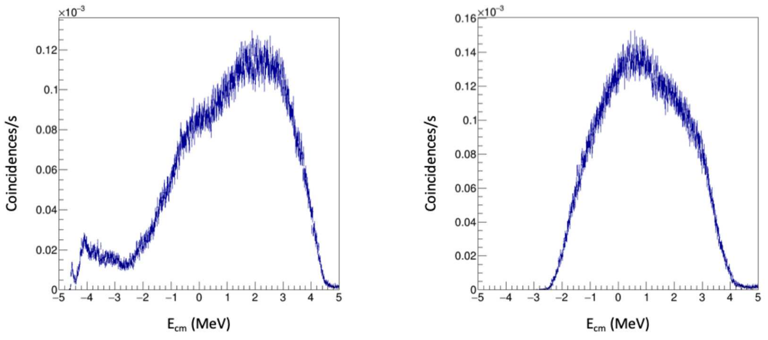


Figure 4.3: Simulated coincidence rate for the ^{2}H(^{26}Al$^{gs}, \alpha^{23}$Na)p and the ^{2}H(^{26}Al$^{gs}, p^{26}$Mg)p reactions. The yield is expressed as a function of the ^{26}Al$^{gs} - n$ relative energy for the two channels. Integration of the spectra leads to a count rate mentioned above. Quasi-free process is enforced by introducing a 40 MeV/c momentum cut on the proton spectator.

used (see Subsec. 4.2.2), it is worth noting that the distance between the beam and the considered strip is only 12 mm. If the beam were not properly collimated, it might end up directly hitting the detector. This could result in malfunctions, damage, or, at best, the generation of a high number of spurious coincidences.

In order to prevent such a scenario, a collimation system has been installed at the beam production point. This system includes three anti-scattering diaphragms with diameters of 4.7 mm, 2.5 mm and 2 mm in the target holder. When the beam in the experiment was changed, there was a check that it was correctly collimated by counting the events of the elastic scattering due to the collision or the grazing of the nuclei of the beam against the metal of the target holder. If no counts were seen, it was a signal that the beam was going through the anti-scattering diaphragms perfectly. However, among the factors that could cause beam deflection is the target itself, which might not be optimally positioned or perfectly homogeneous.

4.2.2 Detectors

In the experiment under consideration, the Trojan Horse Method (THM) is employed to derive the cross sections of these two-body reactions by measuring the cross sections of surrogate three-body reactions. The heavy fragment of the reaction (^{26}Mg or ^{23}Na) is detected in coincidence with the corresponding light fragment (p or α) or the remaining low-energy proton, which acts as a spectator under THM

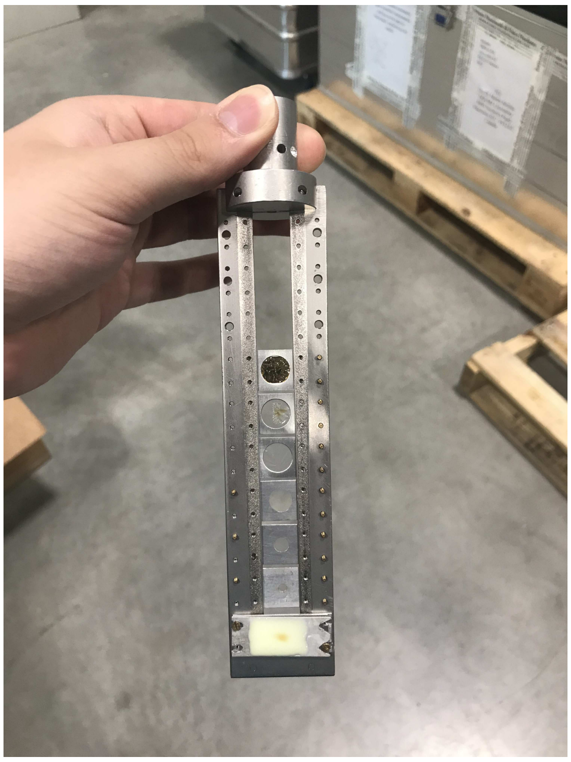


Figure 4.4: The target holder used in the experiment: as it can be seen, there are different types of target used throughout the experimental run. From above: a thin foil of pure ^{197}Au (90 μg/cm^2), two targets of CD$_2$ with a thickness of 116 μg/cm^2, three "collimators" (anti-scattering diaphragms) of respectively $4.7 - 2.5 - 2$ mm of diameter. The last step is a ZnS layer in which we can see the beam spot in dark yellow.

conditions. The other undetected particle is reconstructed through energy and momentum conservation.

The experiment is conducted in inverse kinematics, where a beam of ^{26}Al, as seen in Sec. 4.1, is directed onto a CD$_2$ target to initiate the two reactions. In this setup, the conservation of momentum ensures that the heavy fragment is emitted at very small forward angles, while the light fragments can have a broad range of emission angles, particularly in the case of protons. For this reason, the NEFASTA (NEar FAr Silicon Telescope Array) detector array, consisting of 8 telescopes, has been rearranged into two separate groups. Four telescopes are positioned close to the target at a distance of approximately 49.5 mm (referred to as Group A), while the other four are placed at small angles at a distance of around 69.5 cm from the target (referred to as Group B). The position of the detectors is visible in Fig. 4.5.

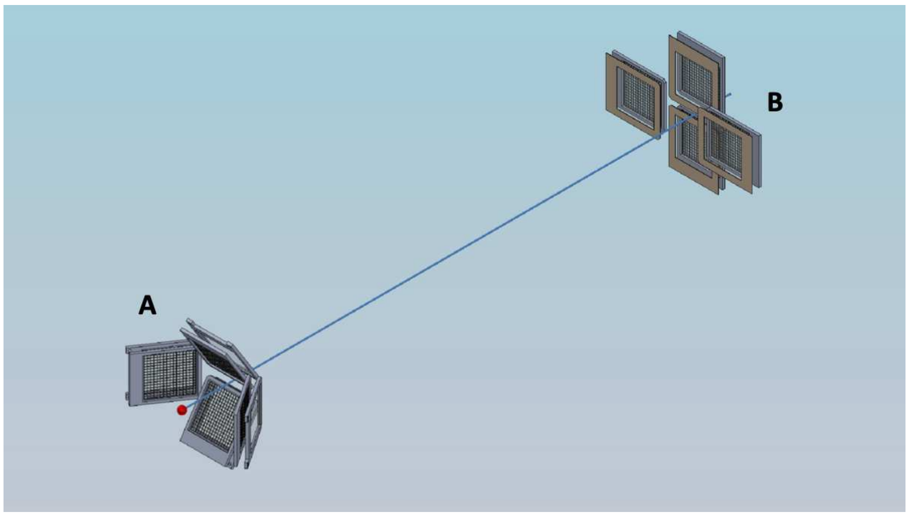


Figure 4.5: Layout of the detectors used in the experiment. The eight telescopes are arranged into two groups. Group A is placed close to the target, about 49.5 mm away. Group B is mounted at about 69.5 cm from the target. For sake of clarity, the position of the target is highlighted with a red sphere and the beam direction with a blue line.

Each telescope in Group A consists of a Double-Sided Silicon Strip Detector (DSSSD) with a thickness of 1000 μm, featuring 32×32 strips over an active area of 51.2×51.2 mm^2, providing a resolution of 1024 pixels. This is followed by a PAD detector, 1500 μm thick, for detecting the protons from the (n, p) reaction channel of interest (which, as will be seen, typically have energies below 20.7 MeV). The angular coverage of Group A detectors varies based on their arrangement: those in the vertical plane (AU and AD) cover the angles $\theta = 9.4° - 55.6°$, while those in the horizontal plane (AR and AL) cover an angular range of $\theta = 32.9° - 79.1°$.

The Group B detectors are positioned farther from the target to achieve good angular resolution for the small angles at which the heavy fragments are emitted. Like Group A, these are arranged in pairs on the vertical (BU and BD) and horizontal (BL and BR) planes. Each Group B telescope consists of a Single-Sided Silicon Strip Detector (SSSSD) with 16 strips and a nominal thickness of 20 μm, followed by a 16×16 strips DSSSD with a thickness of 1000 μm for detecting the heavy fragments. The 20 μm detector introduces an energy threshold of 44 MeV for ^{23}Na and of 48.8 MeV for ^{26}Mg. The BU and BD telescopes cover an angular range of $\theta = 1° - 5.1°$, while the BL and BR telescopes cover the angles $\theta = 2.9° - 7.1°$. The B2 detectors partially overlap with B1, but only in the inactive regions of the B1 detectors. The angles covered by the detectors can be seen in Fig. 4.6.

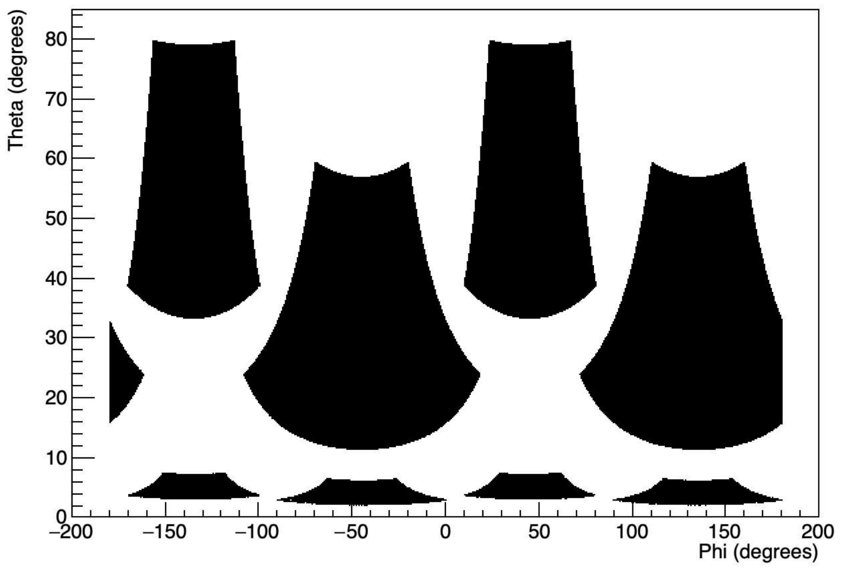


Figure 4.6: Positioning of the detectors as a function of the polar and azimuthal angles. We can see clearly that the A telescopes, being near the target, cover the region at higher θ, while the B telescopes cover small angles in the forward region.

Simulations using the GROOT (GEANT4 and ROOT Object-Oriented Toolkit) simulation system [30], incorporating reaction kinematics, target thickness, beam intensity, and cross-section data, yield the following estimates:

- for the (n, p) channel, a geometric efficiency of 37% and a quasi-free coincidence rate of ~ 3.3 events per minute are expected;
- for the (n, α) channel, a geometric efficiency of 39% and a coincidence rate of ~ 3.4 events per minute are anticipated.

Double Sided Silicon Strip Detectors

The DSSSD (Double-Sided Silicon Strip Detector) belongs to the broad category of semiconductor radiation detectors. A semiconductor detector, as illustrated in Fig. 4.8, is essentially a reverse-biased $p - n$ junction. In such a configuration, the depletion layer surrounding the junction—devoid of intrinsic charge carriers—serves as the active detection volume. When radiation interacts with the detector, it excites electrons from the valence band to the conduction band, resulting in

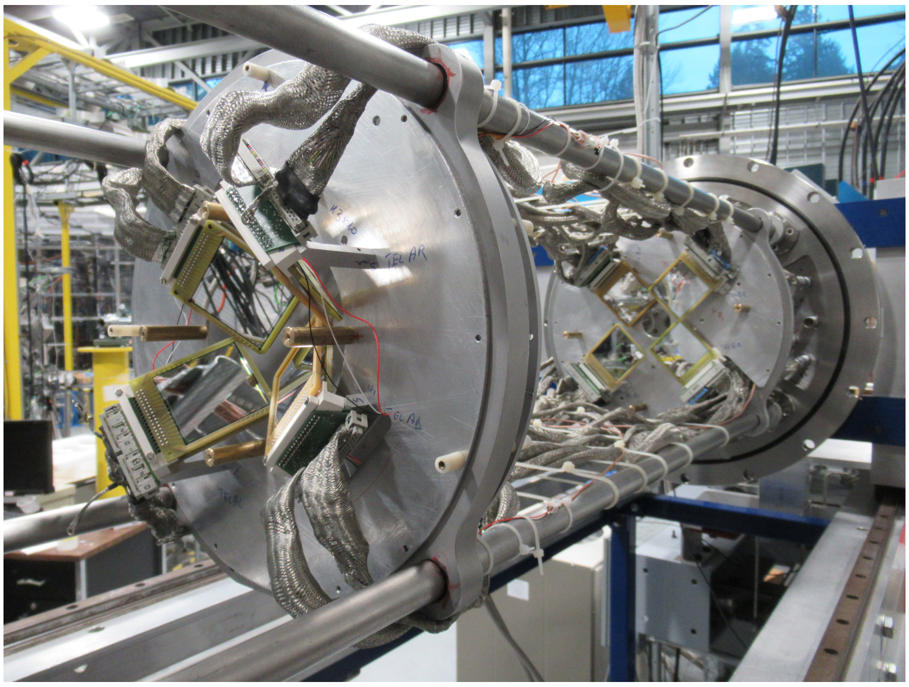


Figure 4.7: Photo of the experimental apparatus in the TUDA reaction chamber with the detectors mounted.

the formation of electron-hole pairs. These charge carriers are subsequently driven toward the electrodes by the electric field present within the junction. As they move, they induce a current on the electrodes, which can be quantified at any moment using Ramo's theorem, ultimately producing the detectable signal.

The DSSSDs employed in this setup feature 16 or 32 orthogonal strips on each side depending on the belonging to group A or B, providing fine granularity. The strips on the front side are oriented perpendicularly to those on the back side. As we can easily see in Fig. 4.8, the strips on the front are along the radial direction. Each strip is 51.2 mm long, with a width of 3.1 mm (B detectors) or 1.5 mm (A detectors), with an inter-strip spacing of 40 μm. This arrangement creates a pixel structure with an angular resolution of approximately 1.4° for detectors A and 0.3° for detectors B.

PAD

The PAD detector consists of a DSSSD described above, with the difference that all the strips are short-circuited at the entrance of the preamplification stage, so that

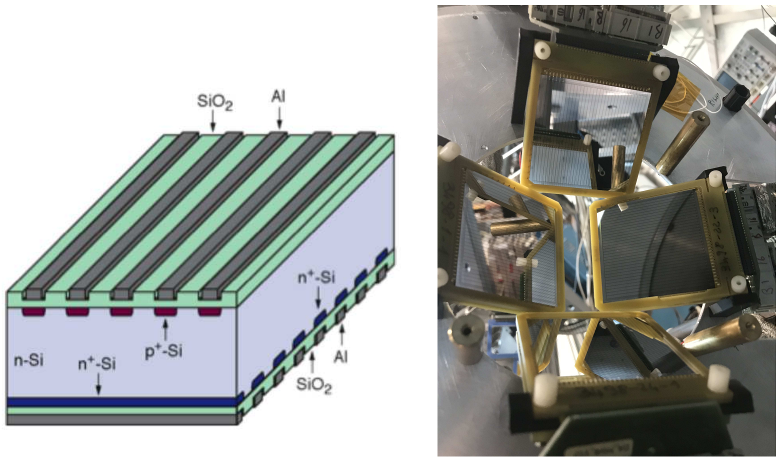


Figure 4.8: Conceptual drawing of a semiconductor detector (left panel) and a photo of the DSSSDs mounted in the experiment (right panel).

only one signal in output from the detector is read.

4.2.3 Electronic chain

In this paragraph the electronics readout employed for the treatment of the signals collected by the different detectors are briefly presented.

Preamplifiers

It is used a set of 16-channel custom low-noise charge sensitive preamplifiers "mesytec MPR-16", which are specially designed for single or double sided multistrip silicon detectors. The detector front and back sides are connected to the electronic boards with cables. The cable ends with a finger that plugs directly into the connector on the preamplifier board. The outputs of the preamplifiers are then connected to the electronic modules outside for further processing.

MEGAMPs

The 16 differential output signals from the preamplifier boards of the DSSSD and SSSSD modules and PADs are processed by a specialized amplifier module known as MEGAMP. This module extracts key information such as energy, timing, and pulse shape analysis. The MEGAMP is a single NIM-standard module housing 16 channels, each divided into sections dedicated to energy and timing processing.

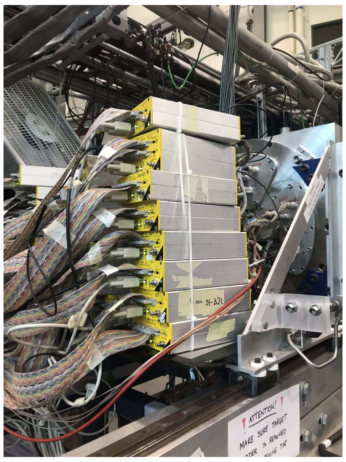


Figure 4.9: Photo of the preamplifiers "mesytec MPR-16" used in the experiment.

The energy section features a spectroscopy amplifier designed to accept differential input signals. Each amplifier incorporates not only a spectroscopy amplifier but also a timing filter and a constant fraction discriminator (CFD). Each MEGAMP produces an OR signal from all the strips of the detector it serves. These signals are then combined by taking the OR of the OR signals (using FIFO modules) for all the front strips of the A detectors and similarly for the B detectors. The outputs from these combined signals are then fed into a coincidence module, where it is applied an AND operation with a coincidence window set to 200 ns. This AND signal serves as the "free" trigger for the data acquisition system. We also record this free trigger on a scaler to measure the total number of valid events detected.

The free trigger is then sent to an I/O register, which generates a signal with a duration equivalent to the conversion and readout time of the ADCs. This signal is used to generate the gate for the ADCs and is also counted. By comparing the total number of gates with the total number of free triggers, we can determine the number of missed events, or the system's dead time. If another free trigger arrives during the active conversion and readout period, it will not generate a gate, and thus the ADCs will not record the event, highlighting the dead time in our system. Fig. 4.10 illustrates the MEGAMP module and the block diagram for a single channel.

ADC

The ADC digitizes the linear analog signals coming out from the MEGAMP modules.

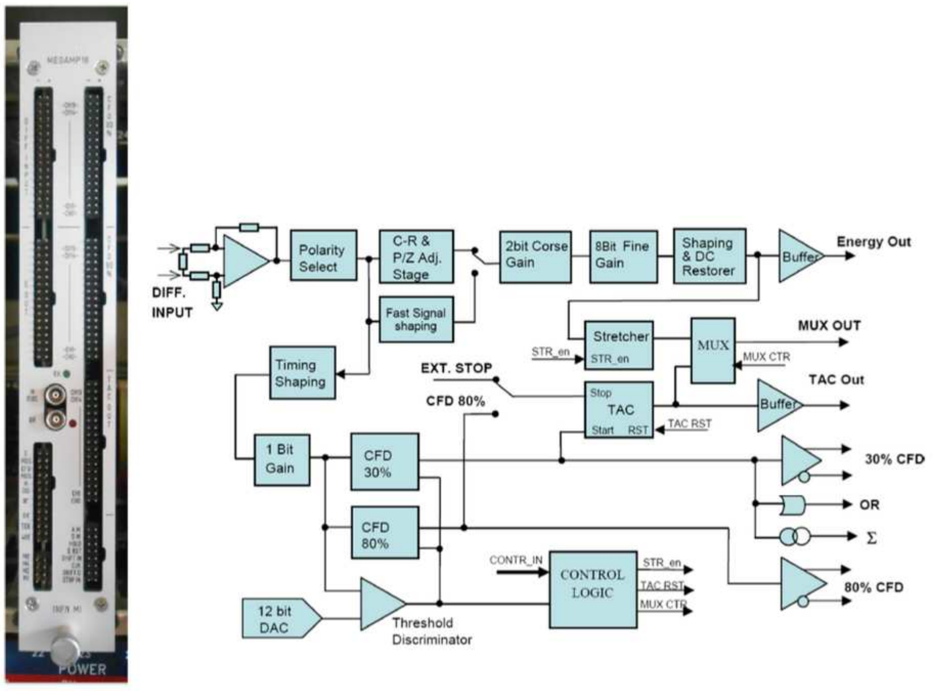


Figure 4.10: MEGAMP (left) and block diagram of a single channel of the MEGAMP (right).

The ADC card consists of 8 analog differential signal receivers and 8 12-bit ADC-chip converters that sample the input signal with a 50 MHz frequency. With the arrival of an external trigger, the control logic starts the conversion sequence. After a programmable delay time the hold signals are sent to the MEGAMP modules in order to capture the maximum of a Gaussian peak. At this point, the ADC card generates the logic signal to bring out in sequence the amplitudes captured by the hold circuit and acquires them by means of an ADC-chip with a sampling rate of 50 MHz.

4.2.4 Vacuum system

In order to do the experiment, the TUDA reaction chamber has to be put under vacuum. The vacuum is realized through different vacuum pumps. The first pumping stage (primary pump) is performed by a rotary vane pump: the working principle of a rotary pump consist of a cylindrically-bored housing with a suction inlet on one side and a discharge outlet on the other side. A rotor (smaller in diameter than the cylinder) is driven at a rotating speed around 1000 rpm about an axis that is placed above the center line of the cylinder to provide minimum clearance between the rotor and cylinder at the top and maximum clearance at the bottom. At the outlet there is an exhaust valve that opens up only when the compressed gas reaches a pressure above the environment atmospheric pressure. Suitable vacuum oils are used to guarantee lubrication of the moving vanes and the sealing between the volumes [31].

The second pumping stage (secondary pump) is a turbo-molecular pump (TMP). The TMP is a bladed turbine that compresses gases by momentum transfer from the rapidly rotating blades of the rotor disks to the gas molecules. The rotation speed of the rotor is typically comprised between 20000 and 70000 rounds per minute. The rotor impulse is transmitted to the particles by the superposition of the thermal velocity of colliding particles with the velocity component of the moving rotor surface. The casual motion of the particles is changed to a directed motion, yielding the pumping process [31].

The third and last pumping stage is the cryogenic pump for reaching the high vacuum region. It removes gas molecules by cooling surfaces to extremely low temperatures (below 20 K), causing gases to condense or adsorb. Condensable gases freeze onto the surfaces, while non-condensable gases like hydrogen and helium are trapped by adsorption onto cold, porous materials.

4.3 Data analysis

As observed so far, it is clear that the correct functioning of the experimental apparatus is crucial for accurately determining the cross-section of the nuclear reactions under investigation. For this reason, it is essential to perform calibration tests before the experiment to ensure that the entire setup operates flawlessly. In particular, the procedures, the ROOT C++ macros developed to analyze the data and the results, giving a clear overview of the logical and computational processes that were necessary to reach the final result will be explained in this section extensively.

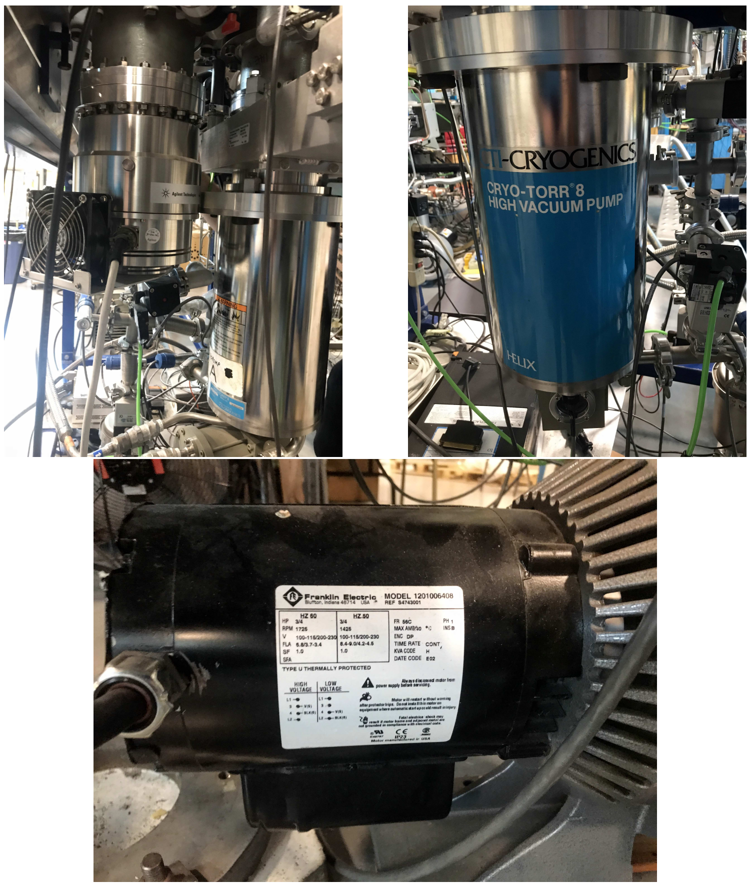


Figure 4.11: Photo of the vacuum pumps used in the laboratory: from top to bottom, left to right, there are the TMP, the cryogenic pump and the rotary vane pump.

4.3.1 Energy calibration

Energy calibration is a fundamental step in the accurate interpretation of data collected. The procedure ensures that the energy spectra obtained from detectors are precisely aligned with known reference points, allowing for the reliable extraction of kinematic properties and reaction observables.

In the context of the experiment, calibration was performed independently for each horizontal and vertical strip of the telescopes (both DSSSDs and SSSSDs, while PADs are not calibrated). The methodology involved for DSSSDs using multiple well-characterized sources and reactions as calibration benchmarks, including:

- a four-peak alpha source emitting particles with well-defined energies for detectors A: 3.182 MeV (^{148}Gd), 4.620 MeV and 4.687 MeV (^{230}Th), 5.805 MeV

(^{244}Cm);

- a different three-peak alpha source for detectors B: 5.157 MeV (^{239}Pu), 5.486 MeV (^{241}Am) and 5.805 MeV (^{244}Cm) (see Fig. 4.12);
- elastic scattering processes induced by ^{12}C at 1.4712 MeV/u, 2.45 MeV/u and 3.4966 MeV/u on a target of ^{197}Au with a thickness of 90 μg/cm^2: these reactions provided high-energy calibration points essential for extending the linearity of the calibration curve.

It has to be specified that all the calibration runs for the DSSSDs need to be done without the SSSSDs mounted, in order to have a more precise measurement on the real positions of the peaks. In total, an amount of 6 (detectors B) or 7 (detectors A) calibration points were obtained.

The SSSSD detectors, being only 20 μm thick, require dedicated calibration runs. Ideally, the calibration particle should stop within the SSSSD itself to achieve maximum precision. For this reason, the calibration of the ΔE detectors was performed using 3 points: one from the alpha peak of the (^{148}Gd) source at 3.182 MeV, another from the elastic scattering of ^{12}C on a ^{197}Au target at 1.4712 MeV/u, and the last from the scattering of the ^{26}Al beam on the same ^{197}Au target at an energy of 38.38 MeV (1.474 MeV/u).

For each strip, the data points were fitted using a linear relationship between the channel number E_{ch} and real energy E(MeV):

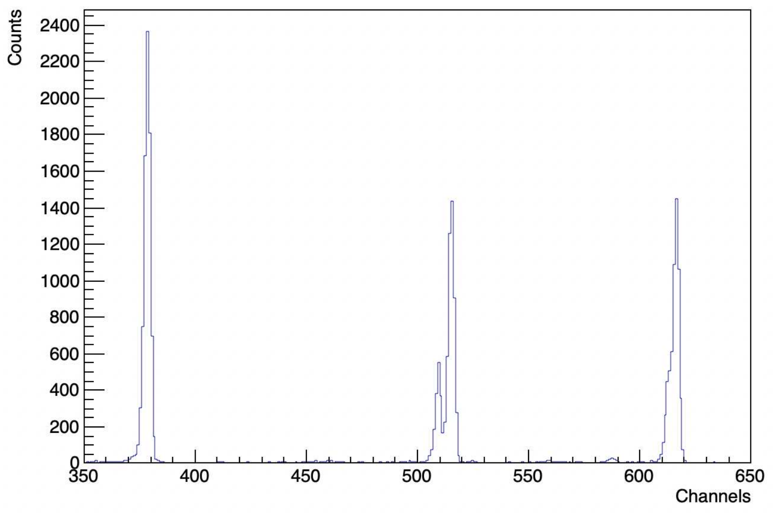
$$E(\text{MeV}) = m \cdot E_{\text{ch}} + q \tag{4.1}$$

where m and q are the slope and intercept derived from the linear fit. The Full Width at Half Maximum (FWHM) energy resolution was then calculated using the relation:

$$\text{FWHM} = 2\sqrt{2ln(2)}m\sigma_x \tag{4.2}$$

where σ_x represents the width of the peak in channel units. A typical resolution of 31 keV for 5.157 MeV α particles was achieved for the DSSSD modules.

The expected energies for the calibration points obtained from elastic scattering measurements were estimated using a macro for the simulation of such processes



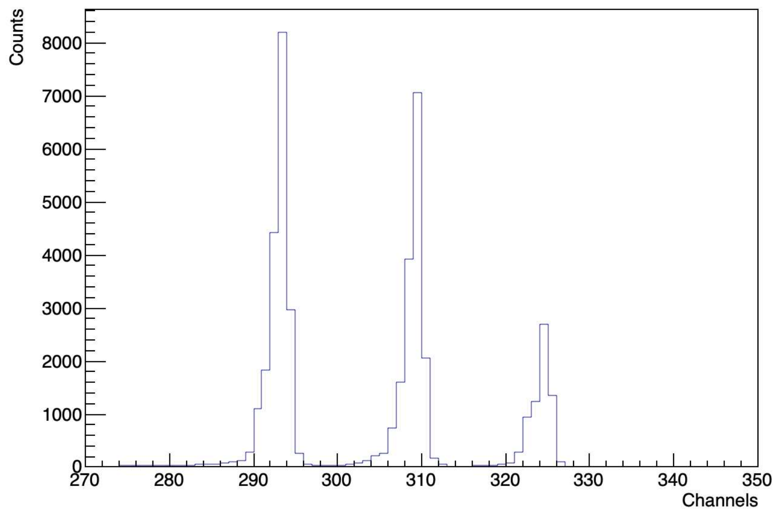


Figure 4.12: Typical spectra of the alpha sources used for the calibration in one strip of the detector A (above) and the detector B (below). All peaks are extremely distinguishable.

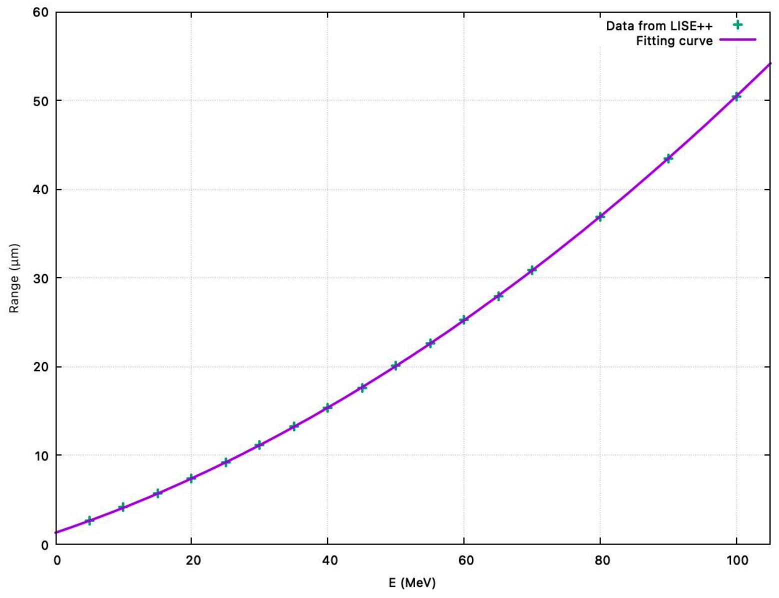


Figure 4.13: Range as a function of energy for the nucleus of ^{12}C in a range of ^{197}Au.

(for the explanation see in appendix "Macros"). In order to have reliable results it was necessary to subtract the energy lost by the beam as it passed through half the target thickness, assuming the reaction occurred at the mid-target position. In order to do so, some energy loss calculations using the LISE++ software [36] were done. The same procedure had to be implemented to account for the energy losses of the outgoing particles after the reaction: nuclei lose energy in the other half of the target and in the dead layer of aluminum in front of every detector. The latter correction was also applied to the α particles. In order to do so, we determined the parameters of the curve governing the energy loss of different nuclei in the target element by approximating it with a parabola ($R = aE^2 + bE + c$), where E is the energy of the particle and R is the range of such particle in a specific material. As shown in Fig. 4.13, this approximation is satisfactory within the energy range of interest. Once the range was found, the thickness of half the target was added. Subsequently, to retrieve the original energy accounting for the energy loss, the formula was inverted using the expression:

$$E = \frac{-b + \sqrt{b^2 - 4a(c - R)}}{2a} \tag{4.3}$$

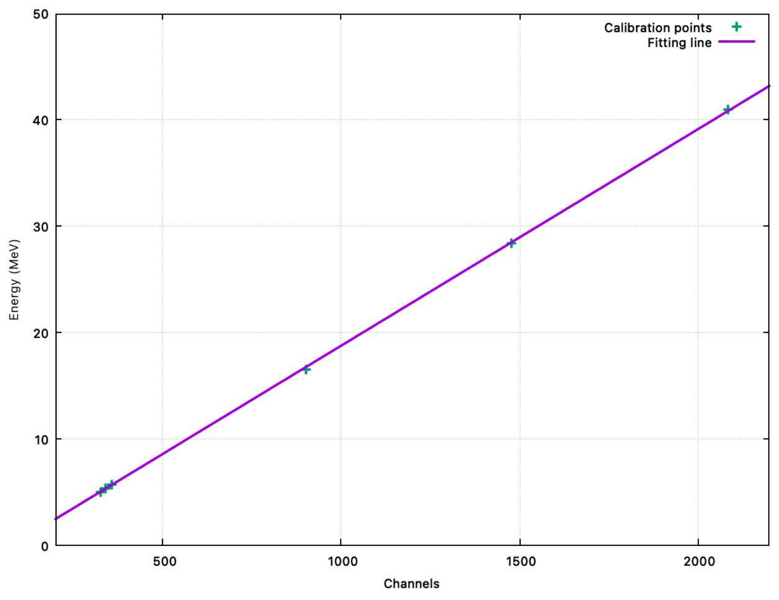


Figure 4.14: Typical calibration plot for a generic strip in the experiment. In particular, this is the calibration for the strip BLB[15].

The resulting effective particle energy was calculated as:

$$E_{\text{eff}} = E_{\text{det}} + \Delta E_{\text{tar}} + \Delta E_{\text{dead}} \tag{4.4}$$

where E_{det} is the energy revealed in the detector, ΔE_{tar} represents the energy lost in half the target thickness and ΔE_{dead} is the energy lost in the dead layer of aluminum.

To validate the accuracy of the calibration, the kinematic loci for the ^{12}C+^{197}Au reaction were analyzed. Specifically, E-θ_{lab} 2D-correlation plots were generated for all the telescopes. These plots prominently displayed an intense line corresponding to elastic scattering events, which was confronted with the one of the simulation, as seen in Fig. 4.15.
The calibration data is instead presented in Fig. 4.14.

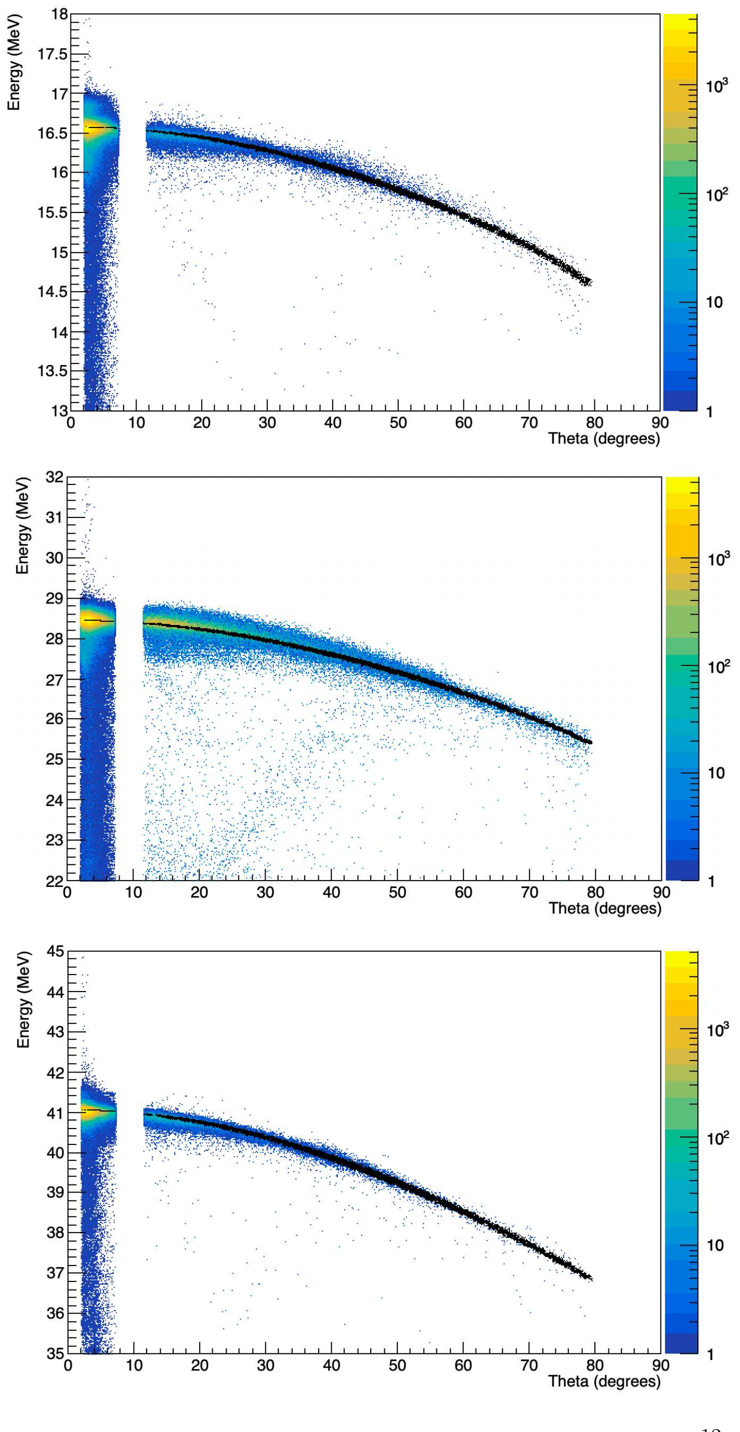


Figure 4.15: Spectra of E-θ_{lab} for the runs of elastic scattering for the reaction ^{12}C+^{197}Au. The colored data are from the runs of calibration for both the detectors A and B, while the black data is from the simulation. The energy losses are accounted for in the simulations. From top to bottom, the energy of the primary beam is: 1.4712 MeV/u, 2.45 MeV/u and 3.4966 MeV/u.

4.3.2 $\Delta E - E$ identification method

The $\Delta E - E$ identification technique used in the experiment is a pivotal method for particle identification, leveraging the principles of energy loss in matter as described by the Bethe-Bloch formula. This approach employs a telescope configuration, consisting in this case of two silicon detectors: one 20 μm thick SSSSD and a DSSSD. It is possible to do the same thing also with the telescopes in the A group using respectively the DSSSD and the PAD detectors. The SSSSD measures the energy lost (ΔE) by particles traversing it, while the DSSSD captures the residual energy (E_{res}), enabling the determination of both the charge (Z) and possibly, mass (in case of good energy resolution) (A) of the incoming particles [33].

In practice, the $\Delta E - E$ correlation plots are generated by plotting the energy lost in the SSSSD against the energy deposited in the DSSSD. This creates a distinctive set of hyperbolic curves, with each curve corresponding to a unique isotopic species. The location of a particle within these plots is governed by the stopping power $\frac{dE}{dx}$, which depends on its atomic and mass numbers as well as its velocity. In the non-relativistic approximation, we have [33]:

$$\frac{dE}{dx} = C_1 \frac{MZ^2}{E} ln \left(C_2 \frac{M}{E} \right) \tag{4.5}$$

where C_1 and C_2 are appropriate constants. For a finite thickness we have:

$$\Delta E = \Delta x \frac{dE}{dx} \propto \frac{MZ^2}{E} \tag{4.6}$$

A useful representation is provided by the Fig. 4.16. In this figure, an additional phenomenon known as the “Punching-Through Energy” can also be observed. This occurs when the detector’s thickness is insufficient to completely stop the incoming particle, leading to a progressive decrease in the energy deposited in both the first and second detectors. This phenomenon is primarily due to the fact that the stopping power of the material, neglecting relativistic effects, decreases as the kinetic energy of the charged particle increases. This produces the characteristic banana-shaped pattern on the plot.

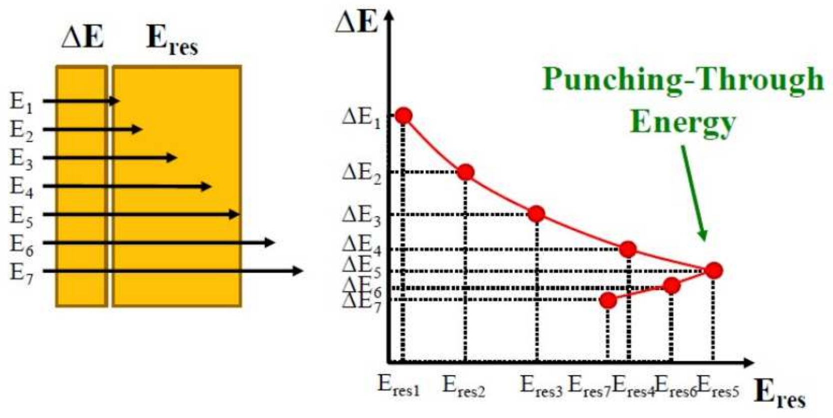


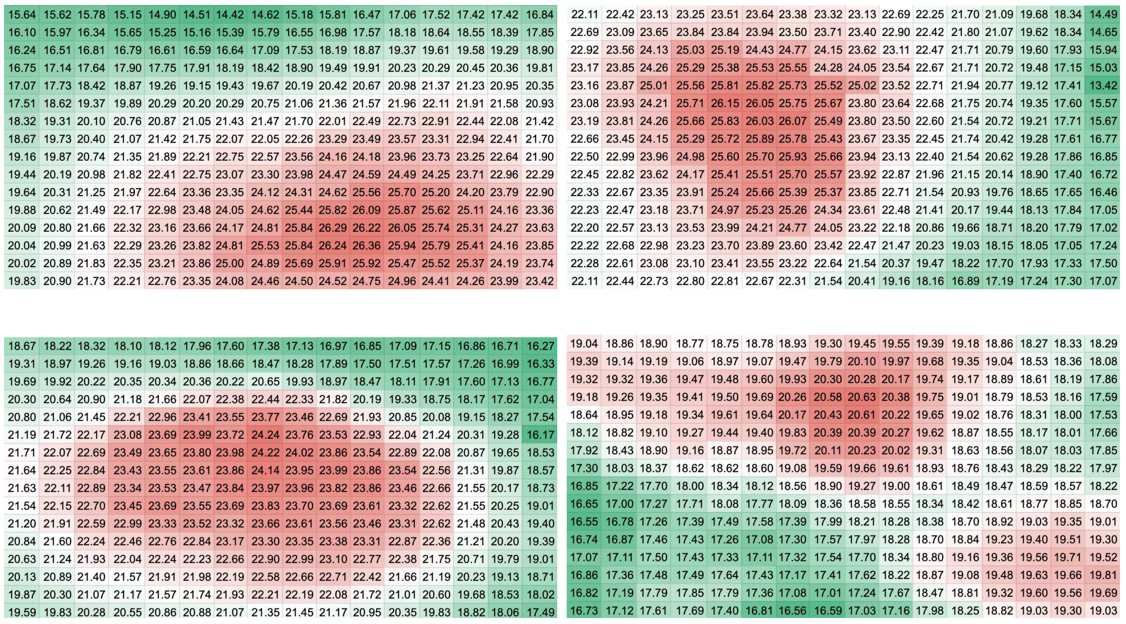
Figure 4.16: Schematic representation of the operating principle of a telescope and the visualization of the punching-trough energy concept.

Using this type of analysis, it is possible to clearly distinguish the reaction products and the presence of specific isotopes because the curves do not intersect. Reaction elements with lower atomic numbers (A) are located in the bottom-left portion of the graph. However, in cases of low energy resolution, experimental data points may overlap along the curves, making it effectively impossible to determine which curve they belong to.

In our experiment, the $\Delta E - E$ technique primarily serves to determine the atomic number (Z) of the heavy fragment produced. The main challenge lies in the fact that for the 20 μm-thick silicon detectors, the thickness is not uniform. Manufacturing defects are relatively common and can add or subtract several micrometers from the nominal thickness. In order to address this issue, it is necessary to perform a mapping of the SSSSD detectors used (see Fig. 4.17), allowing the energy recorded by the detectors to be corrected with a geometric factor of the form:

$$\Delta E_{\mathrm{corr}} = \Delta E_{\mathrm{det}} \cdot \left(\frac{20}{\Delta x_{\mathrm{px}}} \right) \tag{4.7}$$

where ΔE_{det} is the value of the energy read by the SSSSD, Δx_{px} is the thickness of the SSSSD corresponding to the DSSSD pixel behind and ΔE_{corr} is the corrected value of the energy. This correction, as can easily be seen, considers only the geometry of the detector, assuming that the particle's stopping power remains constant. While this assumption is not generally true, in the specific case of this experiment it is sufficient to enable the element separation.



15.64	15.62	15.78	15.15	14.90	14.51	14.42	14.62	15.18	15.81	16.47	17.06	17.52	17.42	17.42	16.84
16.10	15.97	16.34	15.65	15.25	15.16	15.39	15.79	16.55	16.98	17.57	18.18	18.64	18.55	18.39	17.85
16.24	16.51	16.81	16.79	16.61	16.59	16.64	17.09	17.53	18.19	18.87	19.37	19.61	19.58	19.29	18.90
16.75	17.14	17.64	17.90	17.75	17.91	18.19	18.42	18.90	19.49	19.91	20.23	20.29	20.45	20.36	19.81
17.07	17.73	18.42	18.87	19.26	19.15	19.43	19.67	20.19	20.42	20.67	20.98	21.37	21.23	20.95	20.35
17.51	18.62	19.37	19.89	20.29	20.20	20.29	20.75	21.06	21.36	21.57	21.96	22.11	21.91	21.58	20.93
18.32	19.31	20.10	20.76	20.87	21.05	21.43	21.47	21.70	22.01	22.49	22.73	22.91	22.44	22.08	21.42
18.67	19.73	20.40	21.07	21.42	21.75	22.07	22.05	22.26	23.29	23.49	23.57	23.31	22.94	22.41	21.70
19.16	19.87	20.74	21.35	21.89	22.21	22.75	22.57	23.56	24.16	24.18	23.96	23.73	23.25	22.64	21.90
19.44	20.19	20.98	21.82	22.41	22.75	23.07	23.30	23.98	24.47	24.59	24.49	24.25	23.71	22.96	22.29
19.64	20.31	21.25	21.97	22.64	23.36	23.35	24.12	24.31	24.62	25.56	25.70	25.20	24.20	23.79	22.90
19.88	20.62	21.49	22.17	22.98	23.48	24.05	24.62	25.44	25.82	26.09	25.87	25.62	25.11	24.16	23.36
20.09	20.80	21.66	22.32	23.16	23.66	24.17	24.81	25.84	26.29	26.22	26.05	25.74	25.31	24.27	23.63
20.04	20.99	21.63	22.29	23.26	23.82	24.81	25.53	25.84	26.24	26.36	25.94	25.79	25.41	24.16	23.85
20.02	20.89	21.83	22.35	23.21	23.86	25.00	24.89	25.69	25.91	25.92	25.47	25.52	25.37	24.19	23.74
19.83	20.90	21.73	22.21	22.76	23.35	24.08	24.46	24.50	24.52	24.75	24.96	24.41	24.26	23.99	23.42

22.11	22.42	23.13	23.25	23.51	23.64	23.38	23.32	23.13	22.69	22.25	21.70	21.09	19.68	18.34	14.49
22.69	23.09	23.65	23.84	23.84	23.94	23.50	23.71	23.40	22.90	22.42	21.80	21.07	19.62	18.34	14.65
22.92	23.56	24.13	25.03	25.19	24.43	24.77	24.15	23.62	23.11	22.47	21.71	20.79	19.60	17.93	15.94
23.17	23.85	24.26	25.29	25.38	25.53	25.55	24.28	24.05	23.54	22.67	21.71	20.72	19.48	17.15	15.03
23.16	23.87	25.01	25.56	25.81	25.82	25.73	25.52	25.02	23.52	22.71	21.94	20.77	19.12	17.41	13.42
23.08	23.93	24.21	25.71	26.15	26.05	25.75	25.67	23.80	23.64	22.68	21.75	20.74	19.35	17.60	15.57
23.19	23.81	24.26	25.66	25.83	26.03	26.07	25.49	23.80	23.50	22.60	21.54	20.72	19.21	17.71	15.67
22.66	23.45	24.15	25.29	25.72	25.89	25.78	25.43	23.67	23.35	22.45	21.74	20.42	19.28	17.61	16.77
22.50	22.99	23.96	24.98	25.60	25.70	25.93	25.66	23.94	23.13	22.40	21.54	20.62	19.28	17.86	16.85
22.45	22.82	23.62	24.17	25.41	25.51	25.70	25.57	23.92	22.87	21.96	21.15	20.14	18.90	17.40	16.72
22.33	22.67	23.35	23.91	25.24	25.66	25.39	25.37	23.85	22.71	21.54	20.93	19.76	18.65	17.65	16.46
22.23	22.47	23.18	23.71	24.97	25.23	25.26	24.34	23.61	22.48	21.41	20.17	19.44	18.13	17.84	17.05
22.20	22.57	23.13	23.53	23.99	24.21	24.77	24.05	23.22	22.18	20.86	19.66	18.71	18.20	17.79	17.02
22.22	22.68	22.98	23.23	23.70	23.89	23.60	23.42	22.47	21.47	20.23	19.03	18.15	18.05	17.05	17.24
22.28	22.61	23.08	23.10	23.41	23.55	23.22	22.64	21.54	20.37	19.47	18.22	17.70	17.93	17.33	17.50
22.11	22.44	22.73	22.80	22.81	22.67	22.31	21.54	20.41	19.16	18.16	16.89	17.19	17.24	17.30	17.07

18.67	18.22	18.32	18.10	18.12	17.96	17.60	17.38	17.13	16.97	16.85	17.09	17.15	16.86	16.71	16.27
19.31	18.97	19.26	19.16	19.03	18.86	18.66	18.47	18.28	17.89	17.50	17.51	17.57	17.26	16.99	16.33
19.69	19.92	20.22	20.35	20.34	20.36	20.22	20.65	19.93	18.97	18.47	18.11	17.91	17.60	17.13	16.77
20.30	20.64	20.90	21.18	21.66	22.07	22.38	22.44	22.33	21.82	20.19	19.33	18.75	18.17	17.62	17.04
20.80	21.06	21.45	22.21	22.96	23.41	23.55	23.77	23.46	22.69	21.93	20.85	20.08	19.15	18.27	17.54
21.19	21.72	22.17	23.08	23.69	23.99	23.72	24.24	23.76	23.53	22.93	22.04	21.24	20.31	19.28	16.17
21.71	22.07	22.69	23.49	23.65	23.80	23.98	24.22	24.02	23.86	23.54	22.89	22.08	20.87	19.65	18.53
21.64	22.25	22.84	23.43	23.55	23.61	23.86	24.14	23.95	23.99	23.86	23.54	22.56	21.31	19.87	18.57
21.63	22.11	22.89	23.34	23.53	23.47	23.84	23.97	23.96	23.82	23.86	23.46	22.66	21.55	20.17	18.73
21.54	22.15	22.70	23.45	23.69	23.55	23.69	23.83	23.70	23.69	23.61	23.32	22.62	21.55	20.25	19.01
21.20	21.91	22.59	22.99	23.33	23.52	23.32	23.66	23.61	23.56	23.46	23.31	22.62	21.48	20.43	19.40
20.84	21.60	22.24	22.46	22.76	22.84	23.17	23.30	23.35	23.38	23.31	22.87	22.36	21.21	20.20	19.39
20.63	21.24	21.93	22.04	22.24	22.23	22.66	22.90	22.99	23.10	22.77	22.38	21.75	20.71	19.79	19.01
20.13	20.89	21.40	21.57	21.91	21.98	22.19	22.58	22.66	22.71	22.42	21.66	21.19	20.23	19.13	18.71
19.87	20.30	21.07	21.17	21.57	21.74	21.93	22.21	22.19	22.08	21.72	21.01	20.60	19.68	18.53	18.02
19.59	19.83	20.28	20.55	20.86	20.88	21.07	21.35	21.45	21.17	20.95	20.35	19.83	18.82	18.06	17.49

19.04	18.86	18.90	18.77	18.75	18.78	18.93	19.30	19.45	19.55	19.39	19.18	18.86	18.27	18.33	18.29
19.39	19.14	19.19	19.06	18.97	19.07	19.47	19.79	20.10	19.97	19.68	19.35	19.04	18.53	18.36	18.08
19.32	19.32	19.36	19.47	19.48	19.60	19.93	20.30	20.28	20.17	19.74	19.17	18.89	18.61	18.19	17.86
19.18	19.26	19.35	19.41	19.50	19.69	20.26	20.58	20.63	20.38	19.75	19.01	18.79	18.53	18.16	17.59
18.64	18.95	19.18	19.34	19.61	19.64	20.17	20.43	20.61	20.22	19.65	19.02	18.76	18.31	18.00	17.53
18.12	18.82	19.10	19.27	19.44	19.40	19.83	20.39	20.39	20.27	19.62	18.87	18.55	18.17	18.01	17.66
17.92	18.43	18.90	19.16	18.87	18.95	19.72	20.11	20.23	20.02	19.31	18.63	18.56	18.07	18.03	17.85
17.30	18.03	18.37	18.62	18.62	18.60	19.08	19.59	19.66	19.61	18.93	18.76	18.43	18.29	18.22	17.97
16.85	17.22	17.70	18.00	18.34	18.12	18.56	18.90	19.27	19.00	18.61	18.49	18.47	18.59	18.57	18.22
16.65	17.00	17.27	17.71	18.08	17.77	18.09	18.36	18.58	18.55	18.34	18.42	18.61	18.77	18.85	18.70
16.55	16.78	17.26	17.39	17.49	17.58	17.39	17.99	18.21	18.28	18.38	18.70	18.92	19.03	19.35	19.01
16.74	16.87	17.46	17.43	17.26	17.08	17.30	17.57	17.97	18.28	18.70	18.84	19.23	19.40	19.51	19.30
17.07	17.11	17.50	17.43	17.33	17.11	17.32	17.54	17.70	18.34	18.80	19.16	19.36	19.56	19.71	19.52
16.86	17.36	17.48	17.49	17.64	17.43	17.17	17.41	17.62	18.22	18.87	19.08	19.48	19.63	19.66	19.81
16.82	17.19	17.79	17.85	17.79	17.36	17.08	17.01	17.24	17.67	18.47	18.81	19.32	19.60	19.56	19.69
16.73	17.12	17.61	17.69	17.40	16.81	16.56	16.59	17.03	17.16	17.98	18.25	18.82	19.03	19.30	19.03

Figure 4.17: Mapping of the thickness of the SSSSD detectors used in the experiment. From top to bottom, left to right: DEBU, DEBD, DEBR and DEBL. The numbers are the thicknesses in μm and the colors differentiate thicker pixels (in red) from thinner ones (in green).

In order to verify the accuracy of the thicknesses and test the validity of the assumption that the stopping power is constant, an elastic scattering run on ^{197}Au was conducted using a beam composed of ^{26}Al and ^{26}Mg. The kinematic region covered by the B telescopes is at such small angles that no appreciable energy variation as a function of the polar angle is expected according to the Rutherford cross-section. For brevity, only the data from the DEBU and the DEBL detector, which exhibit respectively the poorest and the best performance among the detectors considered, are reported here. For each pixel, the two peaks generated by the different elements were fitted with Gaussian curves, and their centroids were recorded (see Fig. 4.18). As shown, there is agreement between the behavior of the thicknesses and the positions of the peaks.

If the stopping power were constant, the ratio between the peak positions and the thicknesses for each pixel would be nearly identical across the detector. However, Fig. 4.19 shows the percentage deviations of this ratio from the mean. Differences

26.43	26.81	26.87	26.14	25.24	24.67	24.62	25.30	26.07	26.97	28.14	28.98	30.07	30.27	30.01	29.20
27.23	27.41	28.03	27.63	26.64	26.31	26.81	27.63	28.57	29.46	30.58	31.57	32.31	32.55	32.08	31.23
28.00	28.42	29.07	29.37	29.07	29.00	29.49	30.32	31.25	32.19	33.48	34.46	34.82	34.64	34.10	33.08
28.77	29.67	30.70	31.52	31.64	31.73	32.31	33.06	33.89	34.73	35.76	36.19	36.30	36.70	36.38	35.24
29.32	30.76	32.38	33.62	34.19	34.32	34.71	35.48	36.44	37.14	37.28	37.88	38.53	38.35	37.61	36.24
30.23	32.05	33.82	35.31	36.20	36.54	36.83	37.58	38.49	38.61	39.19	40.13	40.35	39.71	38.79	37.21
31.40	33.35	35.09	36.62	37.66	38.31	38.89	39.45	39.89	40.63	41.74	42.14	42.08	41.14	39.91	38.22
32.41	34.28	35.95	37.54	38.70	39.65	40.78	40.96	42.03	43.97	44.00	43.49	43.11	42.18	40.71	39.08
33.24	35.06	36.79	38.54	39.78	40.92	41.96	42.30	44.50	46.17	45.76	45.26	44.61	43.16	41.37	39.73
33.92	35.57	37.26	39.11	40.66	42.20	43.07	44.01	45.60	46.48	46.95	46.95	46.18	44.74	42.67	40.70
34.73	36.14	37.62	39.48	41.42	43.18	44.44	46.16	47.01	47.64	48.20	48.00	47.05	45.85	44.27	42.17
34.90	36.54	38.08	39.96	42.09	43.86	45.44	47.39	48.73	49.44	49.36	48.70	47.91	46.76	45.15	43.28
34.68	36.69	38.41	40.30	42.41	44.14	45.96	47.74	48.92	49.65	49.54	48.78	48.02	46.99	45.46	43.69
34.39	36.50	38.36	40.31	42.26	43.99	46.13	47.49	48.51	49.03	48.79	48.06	47.53	46.90	45.36	43.85
34.28	36.48	38.33	39.93	41.36	42.84	45.18	46.79	47.48	47.47	47.17	46.75	46.41	45.76	44.76	43.30
34.11	36.22	38.02	39.42	39.95	41.09	43.57	45.01	45.26	45.38	45.29	45.38	45.02	43.92	42.96	41.87

30.17	30.62	30.66	29.84	28.82	28.12	28.06	28.87	29.80	30.75	32.14	33.08	34.35	34.57	34.27	33.39
31.10	31.30	31.99	31.56	30.45	30.03	30.56	31.69	32.78	33.60	34.90	36.09	36.97	37.22	36.70	35.73
31.99	32.47	33.17	33.55	33.23	33.11	33.63	34.83	35.80	36.80	38.28	39.44	39.85	39.62	39.00	37.87
32.86	33.91	35.09	36.06	36.21	36.28	36.94	37.92	38.78	39.73	40.91	41.44	41.61	42.04	41.64	40.35
33.49	35.19	37.02	38.46	39.14	39.26	39.70	40.65	41.74	42.54	42.68	43.39	44.15	43.94	43.05	41.54
34.54	36.69	38.68	40.41	41.47	41.86	42.14	43.07	44.08	44.23	44.91	45.98	46.27	45.50	44.43	42.65
35.91	38.16	40.15	41.96	43.18	43.93	44.54	45.22	45.65	46.56	47.83	48.34	48.26	47.18	45.74	43.79
37.09	39.27	41.16	43.03	44.38	45.48	46.67	46.96	48.08	50.47	50.47	49.90	49.47	48.38	46.68	44.75
38.04	40.17	42.15	44.15	45.63	46.90	48.11	48.47	51.12	53.05	52.54	51.96	51.22	49.52	47.44	45.56
38.93	40.74	42.66	44.83	46.65	48.44	49.31	50.61	52.27	53.40	53.87	53.91	53.07	51.38	48.93	46.73
39.74	41.40	43.11	45.28	47.45	49.54	51.01	53.09	53.77	54.72	55.30	55.12	54.09	52.69	50.87	48.43
39.96	41.93	43.67	45.86	48.28	50.37	52.12	54.46	55.83	56.70	56.62	55.88	55.01	53.73	51.87	49.84
39.75	41.99	44.02	46.25	48.71	50.58	52.66	54.82	56.20	56.98	56.93	55.96	55.17	53.97	52.20	50.22
39.42	41.92	44.02	46.35	48.60	50.49	52.82	54.33	55.49	56.34	56.11	55.14	54.70	53.85	52.13	50.28
39.34	41.34	43.71	45.87	47.49	49.18	51.67	53.67	54.09	54.59	54.28	53.80	53.29	52.55	51.40	49.69
39.06	41.23	43.67	45.24	46.05	47.36	49.90	51.36	51.72	52.01	52.09	52.02	51.47	50.46	49.42	47.70

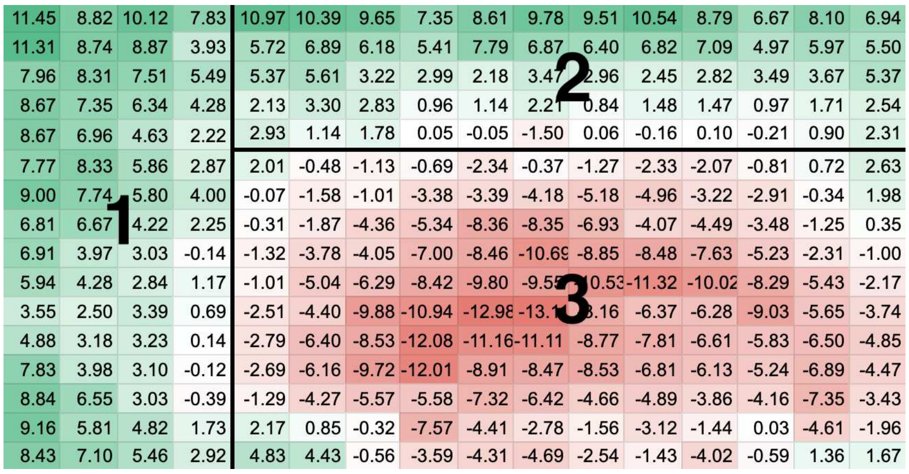
33.82	33.98	33.68	33.39	33.13	33.01	33.41	34.11	34.42	34.78	34.81	34.73	34.43	33.55	32.88	32.68
34.27	34.37	34.32	34.18	33.97	34.00	34.60	35.14	35.45	35.63	35.48	35.14	34.69	33.87	33.15	32.76
34.23	34.34	34.52	34.58	34.56	34.83	35.63	36.18	36.33	36.28	35.86	35.14	34.54	33.68	32.64	31.60
33.93	34.26	34.55	34.78	34.97	35.35	36.12	36.67	36.83	36.70	35.98	34.94	34.37	33.56	32.48	31.24
33.17	33.91	34.46	34.79	35.03	35.31	36.08	36.66	36.93	36.87	35.95	34.88	34.40	33.51	32.36	31.30
32.21	33.23	34.10	34.54	34.68	34.86	35.76	36.60	37.00	36.79	35.72	34.67	34.05	33.07	32.27	31.41
31.19	32.32	33.48	34.00	34.09	34.23	35.12	36.22	36.58	36.37	35.19	34.34	33.98	32.80	32.34	31.66
30.22	31.29	32.34	33.03	33.38	33.30	33.96	35.01	35.61	35.46	34.27	33.83	33.82	33.24	32.72	31.92
29.28	30.24	31.03	31.87	32.58	32.32	32.58	33.43	34.19	34.21	33.36	33.62	33.85	33.72	33.32	32.54
28.81	29.65	30.39	31.13	31.69	31.61	31.68	32.25	32.96	33.05	32.90	33.54	34.32	34.33	34.02	33.22
29.06	29.48	30.28	30.74	30.79	30.76	30.66	31.24	32.05	32.54	32.97	33.82	34.90	34.97	34.67	33.90
29.85	29.80	30.54	30.78	30.41	30.20	30.15	30.78	31.55	32.50	33.49	34.46	35.41	35.67	35.14	34.56
29.72	30.08	30.64	30.80	30.78	30.47	30.33	30.71	31.19	32.51	33.81	34.91	35.71	36.06	35.73	35.12
29.22	30.09	30.88	31.14	31.28	30.73	30.10	30.01	30.75	31.92	33.39	34.74	35.58	36.06	35.74	35.32
28.77	29.76	30.75	31.50	31.05	29.96	29.26	29.05	29.76	30.61	32.41	34.02	34.99	35.46	35.17	34.90
28.34	29.43	30.07	30.34	29.72	28.78	29.06	28.35	29.25	29.95	31.21	32.76	33.78	34.34	34.19	33.45

38.76	38.97	38.56	38.23	37.92	37.82	38.28	39.08	39.46	39.89	39.90	39.79	39.43	38.40	37.68	37.47
39.26	39.41	39.29	39.16	38.92	38.95	39.68	40.28	40.64	40.87	40.69	40.21	39.69	38.76	38.02	37.55
39.22	39.38	39.52	39.65	39.61	39.92	40.87	41.46	41.66	41.62	41.09	40.24	39.57	38.56	37.41	36.21
38.89	39.31	39.54	39.87	40.09	40.52	41.42	42.05	42.27	42.10	41.22	40.00	39.37	38.47	37.22	35.81
37.99	38.86	39.47	39.88	40.17	40.46	41.39	42.03	42.37	42.27	41.22	39.93	39.41	38.41	37.11	35.83
36.93	38.10	39.06	39.59	39.73	39.95	41.03	41.96	42.44	42.21	40.95	39.70	39.02	37.99	36.97	36.00
35.72	37.03	38.32	39.00	39.08	39.23	40.29	41.52	41.96	41.73	40.33	39.32	38.95	37.68	37.07	36.26
34.59	35.81	37.02	37.87	38.24	38.19	38.91	40.16	40.84	40.66	39.28	38.77	38.75	38.13	37.51	36.59
33.50	34.57	35.53	36.51	37.32	37.06	37.34	38.33	39.20	39.22	38.20	38.48	38.83	38.67	38.18	37.30
33.05	33.95	34.80	35.62	36.29	36.20	36.31	36.99	37.77	37.00	37.66	38.40	39.23	39.38	39.04	38.10
33.27	33.79	34.58	35.14	35.29	35.23	35.08	35.77	36.72	37.33	37.78	38.73	40.02	40.10	39.80	38.88
34.14	34.17	34.97	35.27	34.87	34.60	34.56	35.27	36.20	37.25	38.34	39.38	40.62	40.84	40.37	39.64
33.98	34.50	35.08	35.25	35.25	34.91	34.79	35.16	35.79	37.26	38.78	40.03	40.83	41.36	41.04	40.29
33.40	34.48	35.34	35.70	35.87	35.18	34.54	34.39	35.26	36.61	38.23	39.76	40.77	41.23	41.05	40.51
33.05	34.07	35.23	36.10	35.57	34.33	33.47	33.28	34.09	35.16	37.13	38.98	39.95	40.61	40.36	40.04
32.62	33.72	34.48	34.69	34.05	32.96	32.36	32.44	33.52	34.39	35.79	37.60	38.62	39.09	39.14	38.35

Figure 4.18: Position of the the centroids of the peaks for the pixels of detector DEBU (above) and DEBL (below) because of the elastic scattering of ^{26}Al and ^{26}Mg on ^{197}Au. We can see that there is agreement between

of up to $\pm10\%$ are observed, a clear indication that the stopping power is not constant but varies with the energy of the particle. Specifically, as will be discussed in Subsec. 4.3.4, it is necessary to divide the DEBU detector into three regions to clearly separate the elements with different Z, as indicated by the black lines in the figure. In Fig. 4.20, the same analysis is shown for DEBL.

As mentioned earlier, the ΔE identification technique was also used for the A telescopes. In this case, no correction was necessary, as variations of a few micrometers in a 1000 μm thickness are much less significant (a maximum impact of approximately 0.6%, compared to 30% in the previous case). However, the downside of using thicker ΔE layers is the introduction of an energy threshold for the particles detected by the A telescopes. For protons, for instance, the minimum detectable energy is $E_{\text{min}} \sim 12.25$ MeV.



11.45	8.82	10.12	7.83	10.97	10.39	9.65	7.35	8.61	9.78	9.51	10.54	8.79	6.67	8.10	6.94
11.31	8.74	8.87	3.93	5.72	6.89	6.18	5.41	7.79	6.87	6.40	6.82	7.09	4.97	5.97	5.50
7.96	8.31	7.51	5.49	5.37	5.61	3.22	2.99	2.18	3.47	2.96	2.45	2.82	3.49	3.67	5.37
8.67	7.35	6.34	4.28	2.13	3.30	2.83	0.96	1.14	2.21	0.84	1.48	1.47	0.97	1.71	2.54
8.67	6.96	4.63	2.22	2.93	1.14	1.78	0.05	-0.05	-1.50	0.06	-0.16	0.10	-0.21	0.90	2.31
7.77	8.33	5.86	2.87	2.01	-0.48	-1.13	-0.69	-2.34	-0.37	-1.27	-2.33	-2.07	-0.81	0.72	2.63
9.00	7.74	5.80	4.00	-0.07	-1.58	-1.01	-3.38	-3.39	-4.18	-5.18	-4.96	-3.22	-2.91	-0.34	1.98
6.81	6.67	4.22	2.25	-0.31	-1.87	-4.36	-5.34	-8.36	-8.35	-6.93	-4.07	-4.49	-3.48	-1.25	0.35
6.91	3.97	3.03	-0.14	-1.32	-3.78	-4.05	-7.00	-8.46	-10.69	-8.85	-8.48	-7.63	-5.23	-2.31	-1.00
5.94	4.28	2.84	1.17	-1.01	-5.04	-6.29	-8.42	-9.80	-9.55	-10.53	-11.32	-10.02	-8.29	-5.43	-2.17
3.55	2.50	3.39	0.69	-2.51	-4.40	-9.88	-10.94	-12.98	-13.1[illegible]	[illegible].16	-6.37	-6.28	-9.03	-5.65	-3.74
4.88	3.18	3.23	0.14	-2.79	-6.40	-8.53	-12.08	-11.16	-11.11	-8.77	-7.81	-6.61	-5.83	-6.50	-4.85
7.83	3.98	3.10	-0.12	-2.69	-6.16	-9.72	-12.01	-8.91	-8.47	-8.53	-6.81	-6.13	-5.24	-6.89	-4.47
8.84	6.55	3.03	-0.39	-1.29	-4.27	-5.57	-5.58	-7.32	-6.42	-4.66	-4.89	-3.86	-4.16	-7.35	-3.43
9.16	5.81	4.82	1.73	2.17	0.85	-0.32	-7.57	-4.41	-2.78	-1.56	-3.12	-1.44	0.03	-4.61	-1.96
8.43	7.10	5.46	2.92	4.83	4.43	-0.56	-3.59	-4.31	-4.69	-2.54	-1.43	-4.02	-0.59	1.36	1.67

Figure 4.19: Percentage difference from the mean of the ratios between peak positions and thicknesses. It can be observed that DEBU has been divided into three areas to facilitate data analysis.

4.3.3 Elastic scattering on ^{197}Au

After the calibration and the calculation of the corrections on the SSSSD detectors, the functionality of the entire experimental apparatus was verified with a run of elastic scattering of the ^{26}Al beam at 3.5 MeV/u on a ^{197}Au target with a thickness of 90 μg/cm^2. This run provided the opportunity to work with the beam intended for the final experiment. Specifically, the focus was on understanding whether the energy calibrations were correct and on verifying the functionality of the $\Delta E - E$ identification technique for elastic scattering. In order to simplify the data analysis process, it was decided to handle the B telescopes separately in case any of them did not function correctly. Below are the 2D $E - \theta$ plot in Fig. 4.21 for a generic telescope B and the $\Delta E - E$ matrices for each detector B in Fig. 4.22, corrected with the appropriate geometric factor.

As can be clearly seen from the plotted matrices, there is evidently another contribution to the beam besides ^{26}Al. This appears in Fig. 4.21 as a line at higher energies and in Fig. 4.22 as an additional point. Specifically, from the latter figure, it is evident that the contribution of this additional nucleus to the beam is even greater than that of ^{26}Al. The fact that this contribution appears at higher energies, despite the beam being mono-energetic, is a clear indication that the energy reconstruction performed by the data analysis software overestimates the actual energy.

0.61	-1.97	0.08	0.35	1.59	2.48	1.79	1.54	1.24	0.33	-1.24	-2.78	-4.30	-5.40	-1.17	-0.47
1.54	-1.36	-0.57	-1.09	-0.85	-0.05	0.53	0.73	1.85	-0.17	-2.05	-3.32	-3.90	-4.51	-2.28	-2.91
1.04	0.50	-0.06	0.58	0.89	0.58	-0.54	-0.01	-0.87	-1.64	-3.42	-5.04	-4.61	-2.71	-1.21	1.32
1.36	0.35	-0.25	-0.87	-1.08	-1.23	-0.02	0.11	-0.26	-1.83	-3.94	-5.59	-4.72	-2.89	-0.60	0.69
0.27	-0.70	-1.39	-1.62	-0.40	-1.57	-0.65	-1.16	-0.91	-4.11	-4.71	-5.11	-5.11	-4.73	-1.50	-0.28
0.52	1.67	-0.26	-1.02	-0.08	-1.37	-2.09	-1.19	-3.20	-3.27	-3.84	-5.48	-5.30	-3.69	-0.94	0.44
4.18	2.88	1.12	0.77	-2.44	-2.32	0.10	-1.80	-2.54	-3.41	-3.93	-6.04	-4.81	-3.28	-1.09	0.91
3.58	4.68	2.16	0.87	-0.96	-0.78	0.30	-0.46	-2.86	-2.54	-2.81	-2.09	-5.27	-3.42	-1.31	0.59
4.51	2.62	2.96	1.18	0.64	-0.17	2.72	1.36	0.82	-1.78	-1.00	-3.61	-5.03	-3.12	-1.22	-0.31
5.19	3.84	2.27	2.45	2.99	0.33	3.08	2.54	0.85	0.04	-1.18	-3.79	-6.21	-4.60	-2.27	0.64
2.70	2.51	2.41	1.48	2.16	3.26	1.94	4.58	2.24	0.23	-1.15	-2.59	-6.21	-5.49	-0.97	-0.08
-0.04	1.64	3.34	1.69	2.04	1.39	3.94	3.10	2.67	0.42	-0.85	-4.64	-5.87	-5.58	-1.83	-0.84
4.11	2.47	3.17	1.58	0.68	0.13	3.13	3.20	2.06	1.00	-1.61	-3.90	-6.21	-6.15	-2.99	-1.67
4.91	4.89	1.60	0.22	0.91	1.93	2.95	5.86	3.73	3.11	1.28	-3.78	-4.43	-5.40	-3.52	-0.04
7.24	5.17	5.41	1.80	3.68	5.69	6.89	7.46	5.64	5.05	2.76	-2.58	-2.90	-2.66	-1.54	1.04
8.88	6.35	7.51	6.77	7.49	7.09	2.72	7.35	6.50	3.78	4.72	-1.25	-1.25	-2.14	1.10	2.49

Figure 4.20: Same thing for the detector DEBL: in this case there is no need to subdivide the detector, because the variations are less consistent with respect to the DEBU detector.

Since energy loss scales with Z^2, as shown in Eq. 4.6, it can be easily deduced that the contaminant is a nucleus with a lower Z but the same A as ^{26}Al. The $\Delta E - E$ matrices provide further evidence of this: since the beamline selects a fixed magnetic rigidity (mv/q), it can therefore be stated that the contaminant is ^{26}Mg.

A question that might arise is whether such beam contamination could pose any issues for the measurement of the cross section intended in the experiment. As already discussed in Subsec. 4.1.2, in the case of the metastable excited state of $^{26}\mathrm{Al}^m$, contamination does not present a significant challenge because a coincidence is imposed between two particles whose energies fall within the expected ranges for QF processes. Assuming that the light particle is detected by the A telescopes and the heavy fragment by the B telescopes, Fig. 4.23 shows these ranges for both reactions of interest. Subsequently, the third particle is reconstructed using energy and momentum conservation, and an additional condition is imposed on the Q-value of the reaction, which must align with the expected value. Therefore, the contaminants certainly pose a challenge in the data analysis, but they do not compromise the validity of the experiment.

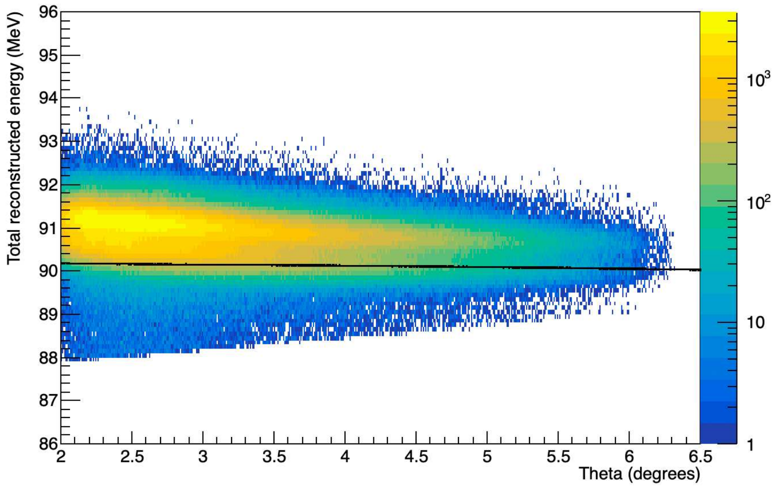


Figure 4.21: Spectrum of E-θ_{lab} for the elastic scattering of ^{26}Al on ^{197}Au. The black line are the data from the simulation. As we can see, there is clearly another contribution to the particle beam.

4.3.4 Scattering on CD_2

After verifying the proper functioning of the experimental setup, the experimental runs were carried out: the ^{26}Al beam, at an energy of 3.5 MeV/u, was directed onto the CD_2 target, that had a thickness of 116 μg/cm^2. However, it must be noted that this beam, in addition to being contaminated, was interrupted shortly after the experiment began, effectively making it impossible to complete the experiment satisfactorily and with adequate statistics for the three-body QF processes.

Regarding the data analysis, the first step was to examine the $\Delta E - E$ matrices of the B telescopes to identify the heavy fragment (see Figs. 4.24 and 4.25). In particular, it became evident that the BU telescope needed to be divided into three sections to allow for better selection. As shown in the images, the curve associated with $Z = 12$ was selected, although all nuclear species of interest (Al, Mg and Na) are clearly visible.

It is worth noting that, at this point of the analysis, we cannot be sure that we are going to select only ^{26}Mg nuclei. However, this will not be a problem because, in the data analysis, it is possible to impose much more stringent conditions: only events in coincidence between the two types of telescopes are considered and, in addition, it is possible to impose a condition on the overall Q-value.

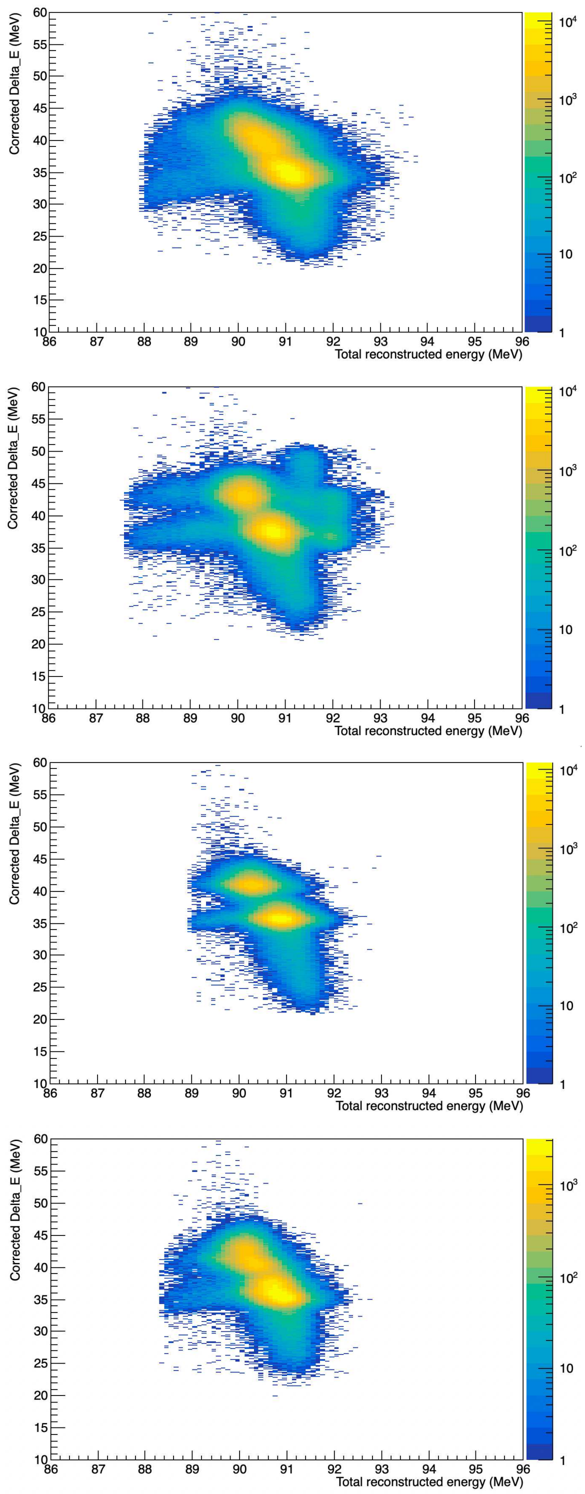


Figure 4.22: $\Delta E - E$ matrices for the elastic scattering of ^{26}Al on ^{197}Au at 3.5 MeV/u. From top to bottom we have: telescope BU, BD, BL and BR. We can see in every plot the two spots of the elastic scattering for the two contributions of the beam, namely ^{26}Al (above) and ^{26}Mg (below).

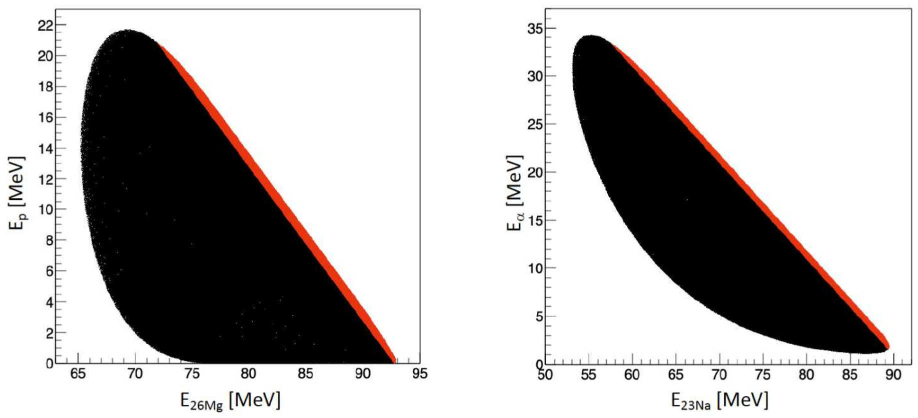


Figure 4.23: Left panel: energy ranges of protons and ^{26}Mg nuclei from the ^{2}H(^{26}Al^{gs}, p^{26}Mg)p reaction at 3.5 MeV/u and for the 1°-7.1° (^{26}Mg) – 9.4°-79.1° (p) angular intervals. The red band is the quasi-free kinematics. Right panel: same as left panel, for the ^{2}H(^{26}Al^{gs}, α^{23}Na)p reaction.

Once we are sure to have isotopes of Mg in the B telescopes, it is possible to use the DSSSD-PAD pair as a $\Delta E - E$ system to also identify the light fragment. However, as previously mentioned in Subsec. 4.3.2, this imposes an energy threshold on the selected protons, with $E_{\mathrm{min}} \sim 12.25$ MeV. Moreover, since the PADs are not calibrated, the energy information for the protons comes solely from the DSSSDs. Using LISE++, energy losses in the silicon thicknesses of the two detectors were simulated, and a fifth-degree polynomial was used to approximate the relation between the energy deposited in the ΔE detector (DSSSD) and the initial energy of the proton (see Fig. 4.26). We were careful to operate within the energy range where this approximation is valid. The cuts on the protons are shown in Fig. 4.27.

Once the selection is made, there is certainty of finding a magnesium isotope in coincidence with a proton. If the beam were composed of pure ^{26}Al, there would be a limited set of reactions falling into this coincidence scenario. However, since the beam is contaminated with ^{26}Mg, the $E - \theta$ spectrum for both the B telescopes and the A telescopes appears rather cluttered, with multiple types of possible reactions, as shown in Figs. 4.28 and 4.29. As an example, in this figure, which displays a typical spectrum with the applied graphical cuts, the results of the simulation for the two-body reaction ^{26}Mg$^{gs}(d,p)^{27}$Mg^{gs} are shown in red. All this processes need to be studied further, in order to isolate all the processes of interest in the experiment.

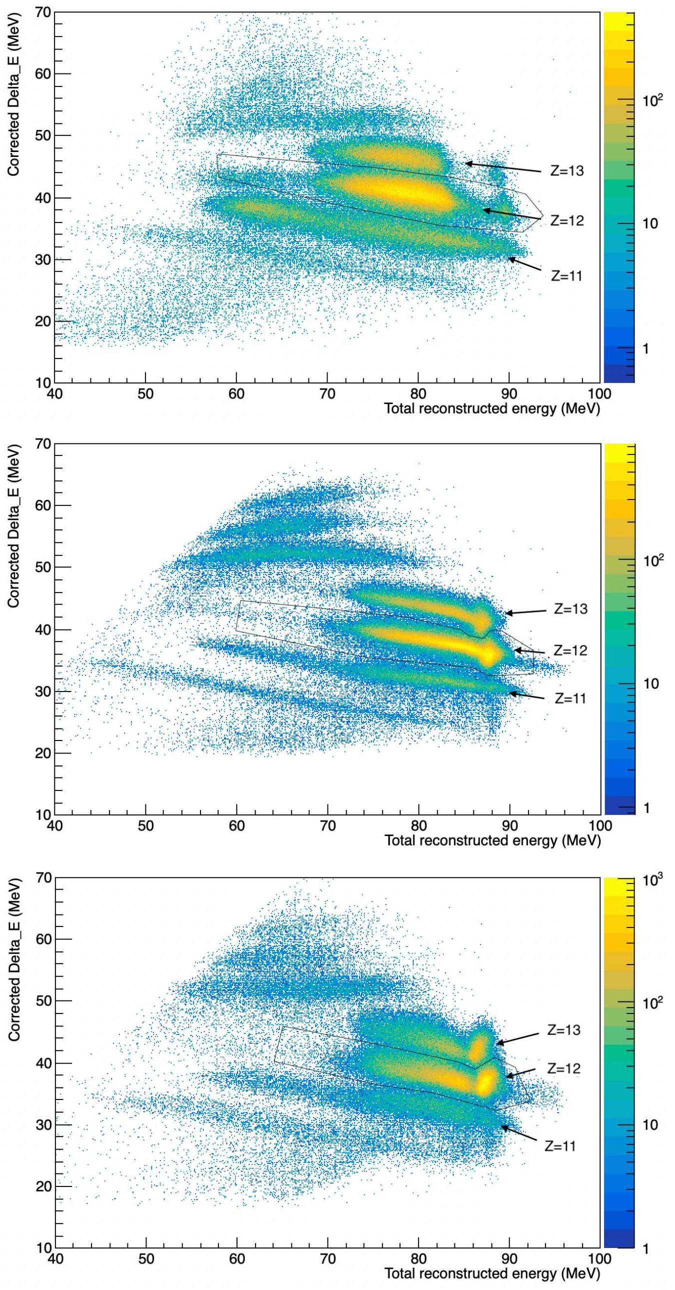


Figure 4.24: Matrices $\Delta E - E$ for the reaction of a beam of both ^{26}Al and ^{26}Mg on a target of CD$_2$. The curves of the various nuclear species are indicated, as long as the graphical cut used for the Mg. From top to bottom: data from telescope BD, BL and BR

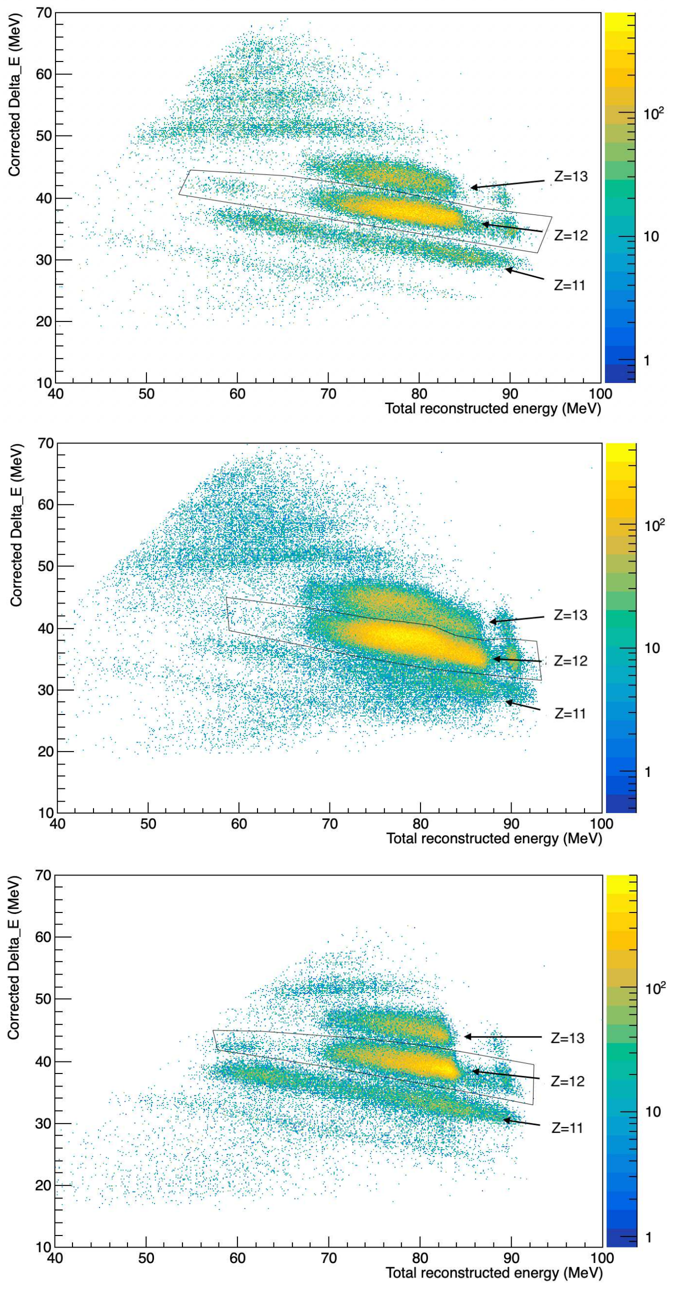


Figure 4.25: Matrices $\Delta E - E$ for the reaction of a beam of both ^{26}Al and ^{26}Mg on a target of CD$_2$ for the detector BU. From top to bottom: BU1, BU2 and BU3.

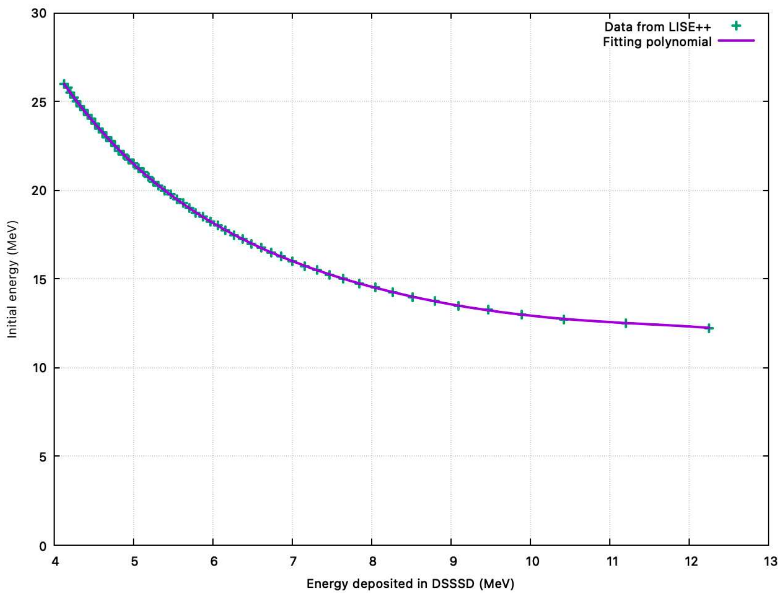


Figure 4.26: Energy deposited in the DSSSD as a function of the total energy of the proton. The fit function is a fifth-degree polynomial.

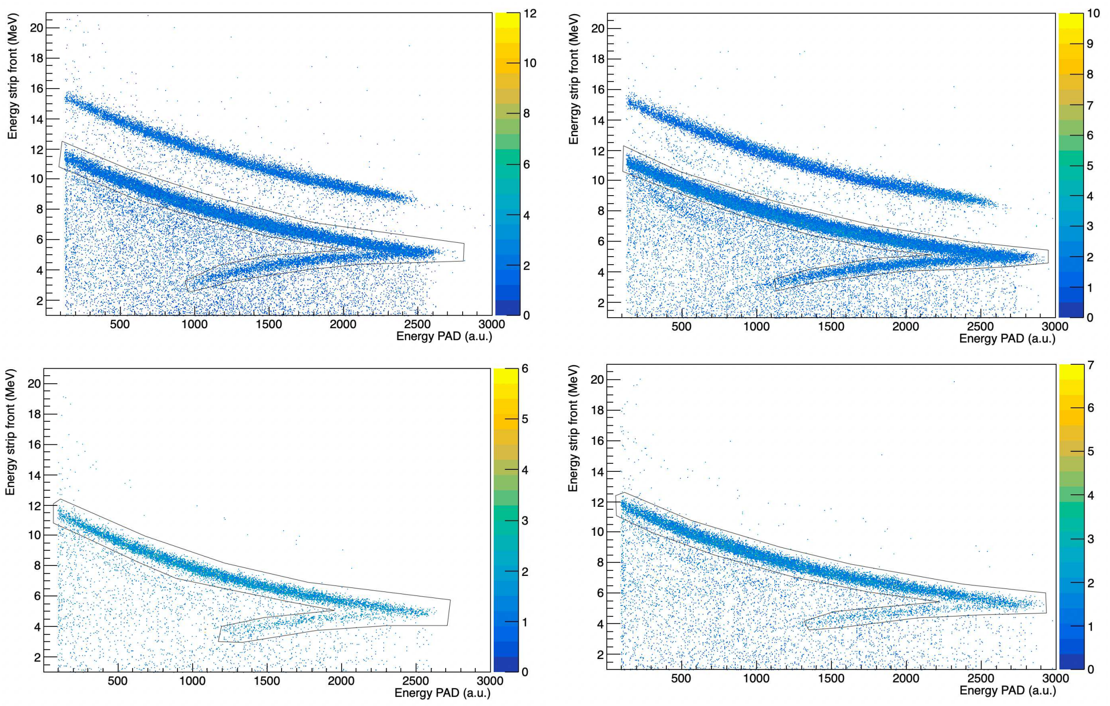


Figure 4.27: Matrices $\Delta E - E$ using the DSSSD as ΔE stage and the PAD as residual energy stage. We can see the graphical cuts on the protons. From top to bottom, left to right we have: telescope AU, AD, AL, AR.

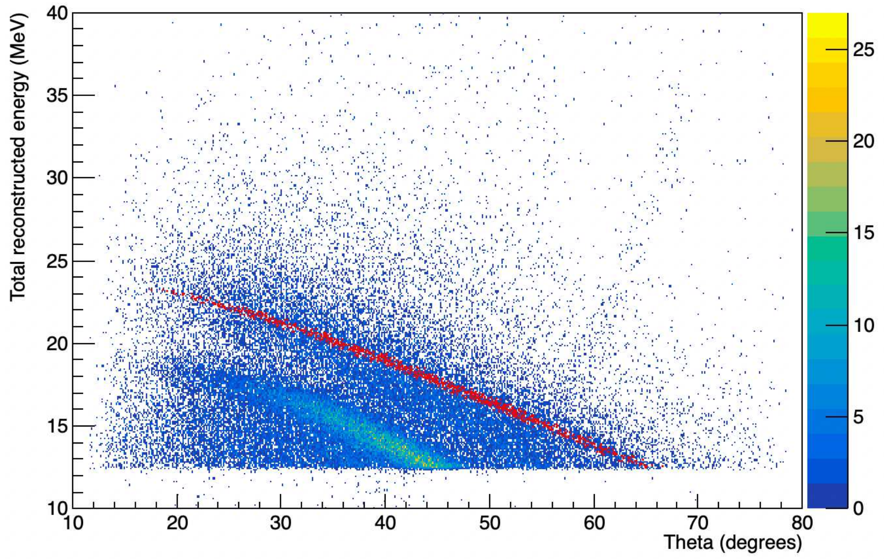


Figure 4.28: Energy as a function of the polar angle for protons detected in the A telescopes: it is clearly visible some local correlation due to two-body processes. The line in red is the result of the simulation for the reaction ^{26}Mg$^{gs}(d,p)^{27}$Mg^{gs}.

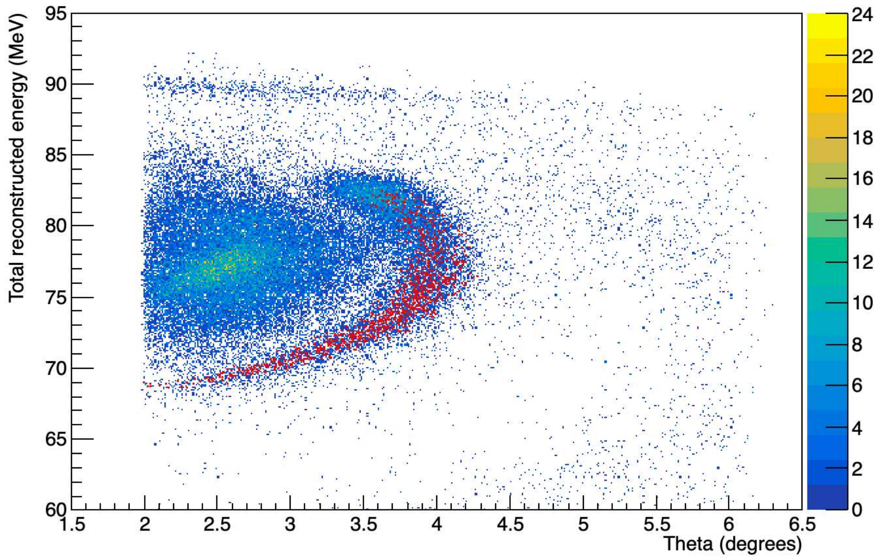


Figure 4.29: Energy as a function of the polar angle for the Mg nuclei detected in the B telescopes: we can see some lines of correlation due to two-body processes. The line in red is the result of the simulation for the reaction ^{26}Mg$^{gs}(d,p)^{27}$Mg^{gs}.

5 Conclusions

The goal of this thesis was to pave the way for the proper analysis of the experimental data from a second run of the experiment conducted in July 2024. For this reason, the thesis work focused primarily on ensuring that the experimental setup operated correctly, developing programs and macros to facilitate data analysis, and creating simulations not only for the calibration runs but also for the measurement runs.

Particular attention was given to calibrations using elastic scattering and to the performance of the SSSSD detectors, which, due to their design, are particularly challenging to handle during data analysis. This is especially true when working with nuclei with $A = 26$, as in this experiment, since they lose a significant amount of energy in thin layers of silicon. Additionally, the detectors were tested for their performance with the actual experimental data, with a preliminary selection of the reaction carried out. However, further analyses will need to be conducted in the future, possibly using data from the new experiment.

Obtaining reliable results in terms of both energy and angular measurements with a complex experimental setup like NEFASTA is extremely challenging. This setup involves large-area detectors placed very close to the target and others at angles near the beam axis. Accurately reproducing nuclear reactions through Monte Carlo simulations and continuously comparing them with experimental data requires a multi-parametric approach that considers various uncertainties, not only those inherent to the detectors but also those associated with the beam, as discussed. All these preliminary tests are crucial for identifying three-body QF processes, as these impose very stringent kinematic conditions. For this reason, highly accurate angular and energy calibrations are essential. In the case of a contaminated beam, background signals must be carefully eliminated by identifying all peripheral reactions. Nonetheless, the results obtained in this thesis are encouraging and suggest that the applied methodology is on the right track.

References

[1] C. Lederer-Woods *et al.*, **Phys. Rev. C 104**, L022803 (2021).

[2] E.F. Keane and M. Kramer, **Mon. Notices Royal Astron. Soc. 391**, 2009 (2008).

[3] R. Diehl *et al.*, **Nature 439**, 45 (2006).

[4] J. Baker *et al.*, **Nature 436**, 1127 (2005).

[5] B. M. Oginni *et al.*, **Phys. Rev. C 83**, 025802 (2011).

[6] C. Iliadis *et al.*, **Astrophys. J. Suppl. Series 193**, 16 (2011).

[7] Lederer-Woods C. *et al.*, **Phys. Rev. C 104**, L032803 (2021).

[8] P. E. Koehler *et al.*, **Phys. Rev. C 56**, 1138 (1997).

[9] H. P. Trautvetter *et al.*, **Z. Phys. A 323**, 1 (1986).

[10] R. T. Skelton, R. W. Kavanagh, and D. G. Sargood, **Phys. Rev. C 35**, 45 (1987).

[11] L. De Smet *et al.*, **Phys. Rev. C 76**, 045804 (2007).

[12] C.E. Rolfs and W.S. Rodney., **Cauldrons in the cosmos : Nuclear astrophysics**, 561, University of Chicago Press, 1988.

[13] G. R. Satchler., **Introduction to Nuclear reactions**, 316, Macmillan Press, 1980.

[14] R.G. Pizzone *et al.*, **Nuc. Phys. A 834**, 2010.

[15] H.J. Assenbaum *et al.*, **Z.Phys. 327**, 01289572 (1987).

[16] K.S. Krane, **Introductory nuclear physics**, 845, Wiley, New York, 1988.

[17] M. La Cognata *et al.* R.E. Tribble, C.A. Bertulani., **Rep. Prog. Phys. 77**, 106901 (2014).

[18] G. Baur *et al.*, **Nuc. Phys. A 458**, 90290 (1986).

[19] C. Spitaleri *et al.*, **Phys. Rev. C 60**, 0558002 (1999).

[20] A. Tumino *et al.* C. Spitaleri, L. Lamia., **Phys. Rev. C 69**, 055806 (2004).

[21] G.F. Chew *et al.*, **Phys. Rev. 85**, 85636 (1952).

[22] A.K. Jain *et al.*, **Nuc. Phys. A 142**, 905336 (1970).

[23] P.G. Roos *et al.*, **Nuc. Phys. A 257**, 906357 (1976).

[24] N.S. Chant *et al.*, **Phys. Rev. C 15**, 1977.

[25] L. Lamia *et al.*, **Astrophys. J. 879**, 23 (2019).

[26] M. Zadro *et al.*, **Phys. Rev. C 40**, 181, 101103 (1989).

[27] C. A. Bertulani *et al.* L. Lamia, C. Spitaleri., **Ap. J 850**, 175, 103847 (2017).

[28] https://www.triumf.ca/research-program/research-facilities/isac-facilities.

[29] M. Marchetto, **ISAC-II operation and future plans**.

[30] D. Lattuada *et al.*, **EPJ Web of Conferences 165**, 01034 (2017).

[31] L. Bruschi *et al.*, **Advanced Physics Lab Notes**.

[32] D. Pierroutsakou *et al.*, **N.I.M sec A 834**, 2016.

[33] S. Carboni *et al.*, **Nuc. Phys. A 664**, 251–263 (2012).

[34] K. D. McKeegan and A. M. Davies, **Treatise on Geochemistry Vol. 1: Meteorites, Comets and Planets**, 2nd edition (Elsevier, Oxford, 2007).

[35] C. Iliadis, **Nuclear Physics of Stars**, (2007)

[36] O.B. Tarasov, D. Bazin, **Nucl. Instr. and Meth. in Phys. Res. B 266**, 4657–4664 (2008)

Macros

Macro for the fitting of the spectra

```
/*
Macro to fit histograms and parameters writing
void FitSpectrum(string inputfile, string telescope, string coordinate,

*/

// Libraries required for program execution
#include "TThread.h"
#include <Riostream.h>
#include <math.h>
#include "TROOT.h"
#include "TFile.h"
#include "TTree.h"
#include "TCanvas.h"
#include "TColor.h"
#include "TH1.h"
#include "TH2.h"
#include "TCutG.h"
#include "TF1.h"
#include "TStyle.h"

using namespace std;

TCanvas *c1 = new TCanvas("c1","spectrum",10,10,600,600);

void FitSpectrumA(string inputfile = "", string detector = "", Int_t st
{
        string name = "";
        name = Form("%s_%d", detector.c_str(), strip);

        TFile *f=new TFile(inputfile.c_str());
        TH1F *h1=(TH1F*)f->Get(name.c_str());

        c1->cd();
        h1->Draw();
        //c1->SetLogy();
        h1->SetTitle(name.c_str());
        h1->SetName(name.c_str());

        c1->Update();

        // ricerca picchi
        TSpectrum *s = new TSpectrum(4);
        Int_t nPeaks;
        Double_t *xPeaks;
        Int_t sigma = 1;                        // threshold on sigma o
    Double_t minratio = 0.00001;          // minimum ratio between peak a
        nPeaks = s->Search(h1,sigma,"",minratio);
        xPeaks = s->GetPositionX();
        for (int p = 0; p < nPeaks; p++) {
                cout << "Peak␣#" << p << "␣@␣channel␣" << xPeaks[p] <<
        }

        // peak sorting
    int a, i, j;
        double array_size = 4;
        double aray[4] = { xPeaks[0], xPeaks[1], xPeaks[2], xPeaks[3]};
    for (i = 0; i < 4; i++)
            {
                for (j = i + 1; j < 4; j++)
                {
                    if (aray[i] > aray[j])
                    {
                        a =  aray[i];
                        aray[i] = aray[j];
                        aray[j] = a;
                    }
                }
            }
cout << "Ordine:␣" << aray[0] << "␣" << aray[1] << "␣" << aray[2] <
    const int index[5]={1,2,3,4};
    xPeaks[0]=aray[0];
    xPeaks[1]=aray[1];
    xPeaks[2]=aray[2];
    xPeaks[3]=aray[3];



    // 4 Gauss
    TF1 *total = new TF1("total","[0]*exp(-0.5*(((x-[1])/[2])**2))␣
TF1 *final = new TF1("final","[9]*exp(-0.5*(((x-[10])/[11])**2))",

    // range Gauss factors
Double_t n1[2] = {10., 20000.}, //{10., 2000.},
                    n2[2] = {10., 20000.}, //{100000., 500000.},
                    n3[2] = {10., 20000.},
         n4[2] = {10., 20000.}; //{10., 200.},

    // range Gauss centroids
    Double_t m1[2] = {xPeaks[0] - 3., xPeaks[0] + 3.},
                m2[2] = {xPeaks[1] - 3., xPeaks[1] + 3.},
                    m3[2] = {xPeaks[2] - 3., xPeaks[2] + 3.},
         m4[2] = {xPeaks[3] - 3., xPeaks[3] + 3.};

    // range widths
    Double_t w1[2] = {0.1,3.},
                w2[2] = {0.1,3.},
                    w3[2] = {0.1,3.},
        w4[2] = {0.1,3.};

    // initial values
    Double_t par[] = {400., xPeaks[0], 1.5,
                                     90.,  xPeaks[1], 1.5, //3000
                                     250., xPeaks[2], 1.2};
Double_t parf[] = {240., xPeaks[3], 1.3};

    //total->SetParNames("fac1","m1","s1","fac2","m2","s2","fac3","
    total->SetParameters(par);  // set initial parameters
final->SetParameters(parf);

    // set range factors
    total->SetParLimits(0,n1[0],n1[1]);
    total->SetParLimits(3,n2[0],n2[1]);
    total->SetParLimits(6,n3[0],n3[1]);
    final->SetParLimits(9,n4[0],n4[1]);

    // set range centroids
    total->SetParLimits(1,m1[0],m1[1]);
    total->SetParLimits(4,m2[0],m2[1]);
    total->SetParLimits(7,m3[0],m3[1]);
    final->SetParLimits(10,m4[0],m4[1]);

    // set range widths
    total->SetParLimits(2,w1[0],w1[1]);
    total->SetParLimits(5,w2[0],w2[1]);
    total->SetParLimits(8,w3[0],w3[1]);
    final->SetParLimits(11,w4[0],w4[1]);

    h1->Fit(total,"R");
    h1->Fit(final,"R");
        h1->GetXaxis()->SetRangeUser(xPeaks[0]-30, xPeaks[3]+10);
// set range fit

        // 1st gauss
        Double_t factor1 = total->GetParameter(0);
        Double_t mean1 = total->GetParameter(1);
        Double_t ermean1 = total->GetParError(1);
        Double_t sigma1 = total->GetParameter(2);
        Double_t ersigma1 = total->GetParError(2);

        // 2nd gauss
        Double_t factor2 = total->GetParameter(3);
        Double_t mean2 = total->GetParameter(4);
        Double_t ermean2 = total->GetParError(4);
        Double_t sigma2 = total->GetParameter(5);
        Double_t ersigma2 = total->GetParError(5);

        // 3rd gauss
        Double_t factor3 = total->GetParameter(6);
        Double_t mean3 = total->GetParameter(7);
        Double_t ermean3 = total->GetParError(7);
        Double_t sigma3 = total->GetParameter(8);
        Double_t ersigma3 = total->GetParError(8);

        // 4th gauss
        Double_t factor4 = final->GetParameter(9);
        Double_t mean4 = final->GetParameter(10);
        Double_t ermean4 = final->GetParError(10);
        Double_t sigma4 = final->GetParameter(11);
        Double_t ersigma4 = final->GetParError(11);

        TF1 *g1 = new TF1("g1","gaus", xPeaks[0]-30, xPeaks[2]+10);
        g1->SetParameters(factor1,mean1,sigma1);
        TF1 *g2 = new TF1("g2","gaus", xPeaks[0]-30, xPeaks[2]+10);
        g2->SetParameters(factor2 ,mean2,sigma2);

        TF1 *g3 = new TF1("g3","gaus", xPeaks[0]-30, xPeaks[2]+10);
        g3->SetParameters(factor3 ,mean3,sigma3);

        TF1 *g4 = new TF1("g4","gaus", xPeaks[3]-10, xPeaks[3]+10);
        g4->SetParameters(factor4 ,mean4,sigma4);

        // plot fit and each gauss
        g1->SetLineColor(2);
        g2->SetLineColor(3);
        g3->SetLineColor(4);
        g4->SetLineColor(5);

        g1->Draw("LSAME");
        g2->Draw("LSAME");
        g3->Draw("LSAME");
        g4->Draw("LSAME");

        cout<<endl;
        cout << "factor1␣" << factor1 << "␣mean1␣" << mean1 << "␣+-␣" <
        cout << "factor2␣" << factor2 << "␣mean2␣" << mean2 << "␣+-␣" <
        cout << "factor3␣" << factor3 << "␣mean3␣" << mean3 << "␣+-␣" <
        cout << "factor4␣" << factor4 << "␣mean4␣" << mean4 << "␣+-␣" <

        string fileoutput= "Fit_CalibrazioneA.txt";
        ofstream fout(fileoutput,ios::app);
        fout << name.c_str() << "\t" << mean1 << "\t" << sigma1 << "\t"
<< "\t" << mean3 << "\t" << sigma3 << "\t" << mean4 << "\t" << sigma4 <
        fout.close();

        // save histogram & fits as image
```

```
          //string filename ="";
          //filename=Form("Fit_Calibrazione/Fit_Calibrazione_%s.png",name
          //c1 -> Print(filename.c_str());


}
```

Macro for the calibration

```
#define Calibrazione_cxx
#include "Calibrazione.h"
#include <TH2.h>
#include <TStyle.h>
#include <TCanvas.h>

using namespace std;

void Calibrazione::Loop()
{
//   In a ROOT session, you can do:
//      root> .L Calibrazione.C
//      root> Calibrazione t
//      root> t.GetEntry(12); // /opt/homebrew/Cellar/root/6.30.06/shar
//      root> t.Show();       // Show values of entry 12
//      root> t.Show(16);     // Read and show values of entry 16
//      root> t.Loop();       // Loop on all entries
//

//     This is the loop skeleton where:
//    jentry is the global entry number in the chain
//    ientry is the entry number in the current Tree
//  Note that the argument to GetEntry must be:
//    jentry for TChain::GetEntry
//    ientry for TTree::GetEntry and TBranch::GetEntry
```

```
//
//       To read only selected branches, Insert statements like:
// METHOD1:
//    fChain->SetBranchStatus("*",0);  // disable all branches
//    fChain->SetBranchStatus("branchname",1);  // activate branchname
// METHOD2: replace line
//    fChain->GetEntry(jentry);       //read all branches
//by  b_branchname->GetEntry(ientry); //read only this branch
   if (fChain == 0) return;

    cout << "Inizio␣programma" << endl;

    // from deg to rad
    double dtr=TMath::Pi()/180.;

  Double_t          EAUF[32]={0.};
  Double_t          EAUB[32]={0.};
  Double_t          EARF[32]={0.};
  Double_t          EARB[32]={0.};
  Double_t          EADF[32]={0.};
  Double_t          EADB[32]={0.};
  Double_t          EALF[32]={0.};
  Double_t          EALB[32]={0.};
  Double_t          EBUF[16]={0.};
  Double_t          EBUB[16]={0.};
  Double_t          EBRF[16]={0.};
  Double_t          EBRB[16]={0.};
  Double_t          EBDF[16]={0.};
  Double_t          EBDB[16]={0.};
  Double_t          EBLF[16]={0.};
  Double_t          EBLB[16]={0.};
  Double_t          EDEBU[16]={0.};
  Double_t          EDEBR[16]={0.};
  Double_t          EDEBD[16]={0.};
Double_t           EDEBL[16]={0.};
  Double_t           EPADAU;
  Double_t           EPADAR;
  Double_t           EPADAD;
  Double_t           EPADAL;

  cout << "Fatto␣variabili" << endl;

  TFile *fout = new TFile("Cal_AlAu.root", "RECREATE");

  TTree *cal = new TTree("cal", "cal");

  cal->Branch("AUF[32]",  AUF,   "AUF[32]/I");
  cal->Branch("AUB[32]",  AUB,   "AUB[32]/I");
  cal->Branch("ARF[32]",  ARF,   "ARF[32]/I");
  cal->Branch("ARB[32]",  ARB,   "ARB[32]/I");
  cal->Branch("ADF[32]",  ADF,   "ADF[32]/I");
  cal->Branch("ADB[32]",  ADB,   "ADB[32]/I");
  cal->Branch("ALF[32]",  ALF,   "ALF[32]/I");
  cal->Branch("ALB[32]",  ALB,   "ALB[32]/I");
  cal->Branch("BUF[16]",  BUF,   "BUF[16]/I");
  cal->Branch("BUB[16]",  BUB,   "BUB[16]/I");
  cal->Branch("BRF[16]",  BRF,   "BRF[16]/I");
  cal->Branch("BRB[16]",  BRB,   "BRB[16]/I");
  cal->Branch("BDF[16]",  BDF,   "BDF[16]/I");
  cal->Branch("BDB[16]",  BDB,   "BDB[16]/I");
  cal->Branch("BLF[16]",  BLF,   "BLF[16]/I");
  cal->Branch("BLB[16]",  BLB,   "BLB[16]/I");
  cal->Branch("DEBU[16]",  DEBU,   "DEBU[16]/I");
  cal->Branch("DEBR[16]",  DEBR,   "DEBR[16]/I");
  cal->Branch("DEBD[16]",  DEBD,   "DEBD[16]/I");
  cal->Branch("DEBL[16]",  DEBL,   "DEBL[16]/I");
  cal->Branch("PADAU",  &PADAU,   "PADAU/I");
  cal->Branch("PADAR",  &PADAR,   "PADAR/I");
```

```
cal->Branch("PADAD",   &PADAD,   "PADAD/I");
cal->Branch("PADAL",   &PADAL,   "PADAL/I");
cal->Branch("EAUF[32]",   EAUF,   "EAUF[32]/D");
cal->Branch("EAUB[32]",   EAUB,   "EAUB[32]/D");
cal->Branch("EARF[32]",   EARF,   "EARF[32]/D");
cal->Branch("EARB[32]",   EARB,   "EARB[32]/D");
cal->Branch("EADF[32]",   EADF,   "EADF[32]/D");
cal->Branch("EADB[32]",   EADB,   "EADB[32]/D");
cal->Branch("EALF[32]",   EALF,   "EALF[32]/D");
cal->Branch("EALB[32]",   EALB,   "EALB[32]/D");
cal->Branch("EBUF[16]",   EBUF,   "EBUF[16]/D");
cal->Branch("EBUB[16]",   EBUB,   "EBUB[16]/D");
cal->Branch("EBRF[16]",   EBRF,   "EBRF[16]/D");
cal->Branch("EBRB[16]",   EBRB,   "EBRB[16]/D");
cal->Branch("EBDF[16]",   EBDF,   "EBDF[16]/D");
cal->Branch("EBDB[16]",   EBDB,   "EBDB[16]/D");
cal->Branch("EBLF[16]",   EBLF,   "EBLF[16]/D");
cal->Branch("EBLB[16]",   EBLB,   "EBLB[16]/D");
cal->Branch("EDEBU[16]",   EDEBU,   "EDEBU[16]/D");
cal->Branch("EDEBR[16]",   EDEBR,   "EDEBR[16]/D");
cal->Branch("EDEBD[16]",   EDEBD,   "EDEBD[16]/D");
cal->Branch("EDEBL[16]",   EDEBL,   "EDEBL[16]/D");
cal->Branch("EPADAU",   &EPADAU,   "EPADAU/D");
cal->Branch("EPADAR",   &EPADAR,   "EPADAR/D");
cal->Branch("EPADAD",   &EPADAD,   "EPADAD/D");
cal->Branch("EPADAL",   &EPADAL,   "EPADAL/D");

Double_t          aAUF[32]={0.0115274,
     0.010981,
     0.0115176,
     0.0110471,
     0.0115869,
     0.0114873,
     0.0113984,
     0.0114811,
     0.0109884,
     0.0113954,
     0.0109492,
     0.0115039,
     0.0112403,
     0.0113335,
     0.0113684,
     0.0111654,
     0.012326,
     0.0132192,
     0.0126489,
     0.0127198,
     0.0124802,
     0.0127598,
     0.012279,
     0.0127125,
     0.0128091,
     0.012579,
     0.0127847,
     0.0122413,
     0.0123339,
     0.0126696,
     0.0128399,
     0.0120028};
Double_t           aAUB[32]={0.012483,
     0.0125703,
     0.0126812,
     0.0121618,
     0.0124592,
     0.0126301,
     0.0125831,
     0.0126722,
     0.0125171,
     0.0127109,
     0.0130884,
     0.0124671,
     0.0128213,
     0.012551,
     0.012245,
     0.0120239,
     0.0124727,
     0.0127139,
     0.0129414,
     0.0129225,
     0.0125501,
     0.0129135,
     0.0127153,
     0.0126626,
     0.0130492,
     0.0127551,
     0.0130573,
     0.0126905,
     0.0126579,
     0.0128304,
     0.0133837,
     0.0130483};
Double_t        aARF[32]={0.0121854,
     0.0117948,
     0.0120147,
     0.012159,
     0.0121852,
     0.0125556,
     0.0120987,
     0.0116765,
     0.012081,
     0.0119018,
     0.0121347,
     0.0118932,
     0.0118621,
     0.0120771,
     0.0119265,
     0.0119152,
     0.0112438,
     0.0112214,
     0.0112077,
     0.01135,
     0.0109951,
     0.0113081,
     0.0110785,
     0.0112389,
     0.0110379,
     0.0109549,
     0.0109805,
     0.0112803,
     0.0110328,
     0.0109357,
     0.01086,
     0.01108};
Double_t          aARB[32]={0.0143021,
     0.0141841,
     0.0146543,
     0.0145763,
     0.0145187,
     0.0148639,
     0.0140622,
     0.0139907,
     0.0142217,
     0.01407,
     0.0143716,
     0.0148498,
     0.0142066,
     0.0144416,
     0.0145849,
     0.0141547,
     0.0141888,
     0.0136669,
     0.014338,
     0.0141449,
     0.0141978,
     0.0143124,
     0.0140109,
     0.0143293,
     0.0139833,
     0.0137897,
     0.0137386,
     0.0140827,
     0.0135951,
     0.0141145,
     0.0140793,
     0.0146489};
Double_t             aADF[32]={0.0122618,
     0.0123808,
     0.0120334,
     0.012398,
     0.0116183,
     0.0122043,
     0.0116211,
     0.012315,
     0.0119648,
     0.0127805,
     0.0117553,
     0.0120235,
     0.0117547,
     0.0120622,
     0.0121014,
     0.015401,
     0.0126033,
     0.0128621,
     0.0127505,
     0.0125882,
     0.0129944,
     0.0123672,
     0.0130706,
     0.0127841,
     0.0127117,
     0.0127387,
     0.0130614,
     0.0126201,
     0.0125674,
     0.0130913,
     0.0126774,
     0.0126251};
Double_t          aADB[32]={0.0143232,
     0.0144758,
     0.0139906,
     0.0139668,
     0.0140302,
     0.0143175,
     0.0146242,
     0.0143568,
     0.0140432,
     0.0141056,
     0.0143376,
     0.0146495,
     0.0144044,
     0.0138756,
     0.0145469,
     0.0145033,
     0.0136692,
     0.0141038,
     0.0147115,
     0.0140591,
     0.0144198,
     0.0139518,
     0.0139253,
     0.0143486,
     0.0141972,
     0.0144822,
     0.0144457,
     0.0141341,
     0.0141903,
     0.0144822,
     0.0142458,
     0.0141376};
Double_t            aALF[32]={0.0128129,
     0.013199,
     0.012678,
     0.0131958,
     0.0128781,
     0.0124941,
     0.0129559,
     0.0130342,
     0.0128458,
     0.0127154,
     0.0128187,
     0.0131682,
     0.0129088,
     0.0134389,
     0.0123475,
     0.0126942,
     0.0115599,
     0.0119896,
     0.0117028,
     0.0121914,
     0.0114099,
     0.012065,
     0.0116655,
     0.0117746,
     0.0118878,
     0.0117245,
     0.0116766,
     0.0115827,
     0.0115431,
     0.01155,
     0.0119428,
     0.011728};
Double_t              aALB[32]={0.0129628,
     0.0132428,
     0.013117,
     0.0129393,
     0.0128522,
     0.0127804,
     0.0129603,
     0.0130896,
     0.0125684,
     0.012762,
     0.0131746,
     0.0127763,
     0.0127157,
     0.0126398,
     0.0131284,
     0.0129232,
     0.0135509,
     0.0134115,
     0.017929,
     0.0137692,
     0.0135291,
     0.0137417,
     0.013824,
     0.0136173,
     0.0134608,
     0.0139798,
     0.0137209,
     0.0132925,
     0.013546,
     0.0137666,
     0.0140399,
     0.013348};
Double_t              aBUF[16]={0.0207348,
     0.0207259,
     0.0214091,
     0.0207975,
     0.0209048,
     0.0206204,
     0.0213685,
     0.0206959,
     0.0205417,
     0.0214025,
     0.0203927,
     0.0208821,
     0.0204867,
     0.0203911,
     0.0195435,
     0.0206569};
Double_t              aBUB[16]={0.0207315,
     0.0205954,
     0.0213147,
     0.0210858,
     0.0209326,
     0.0206849,
     0.0214561,
     0.0205664,
     0.0211276,
     0.0209919,
     0.0209447,
     0.0209236,
     0.0204937,
     0.0201183,
     0.0210434,
     0.0207127};
Double_t              aBRF[16]={0.0180338,
     0.0168083,
     0.0158976,
     0.016727,
     0.0302121,
     0.0158803,
     0.0163144,
     0.0167399,
     0.0221444,
     0.0163715,
     0.0157923,
     0.0160132,
     0.0165953,
     0.0163604,
     0.0161051,
     0.0163557};
Double_t              aBRB[16]={0.0163823,
     0.0166862,
     0.0164173,
     0.0169143,
     0.0168192,
     0.0165582,
     0.0164878,
     0.0164264,
     0.0161023,
     0.0162877,
     0.015802,
     0.0167168,
     0.0161106,
     0.0163709,
     0.0166225,
     0.016886};
Double_t            aBDF[16]={0.0175414,
     0.016988,
     0.0161564,
     0.0169526,
     0.0167839,
     0.0173363,
     0.0172459,
     0.0169173,
     0.0170003,
     0.0169162,
     0.0165039,
     0.0172065,
     0.0171215,
     0.0173378,
     0.0165221,
     0.0168373};
Double_t            aBDB[16]={0.0161119,
     0.0159483,
     0.016122,
     0.0158342,
     0.016177,
     0.0159989,
     0.0153391,
     0.0156443,
     0.0156167,
     0.0158555,
     0.0153191,
     0.0155332,
     0.0154921,
     0.0156717,
     0.015953,
     0.0158682};
Double_t             aBLF[16]={0.0216909,
     0.0206012,
     0.0219189,
     0.0212882,
     0.0217689,
     0.0205925,
     0.0210124,
     0.0210214,
     0.0210223,
     0.020534,
     0.0220538,
     0.0200918,
     0.0221048,
     0.0207436,
     0.021707,
     0.0207367};
Double_t             aBLB[16]={0.0213882,
     0.019989,
     0.0205982,
     0.0201601,
     0.0211843,
     0.0207537,
     0.0209914,
     0.0207769,
     0.0201556,
     0.0204513,
     0.0199352,
     0.0202692,
     0.0211464,
     0.0206616,
     0.021019,
     0.0206924};
//Double_t            aDEBU[16]={1., 1., 1., 1., 1., 1., 1., 1., 1., 1.
Double_t            aDEBU[16]={0.01751020052,
     0.0190507999,
     0.01789313417,
     0.01659914854,
     0.01538575846,
     0.01660411921,
     0.01956950389,
     0.01669213751,
     0.01663784197,
     0.01657853788,
     0.01648466951,
     0.01649532966,
     0.01970052542,
     0.0163458964,
     0.023259473,
     0.01737951971};
Double_t            aDEBR[16]={0.01699227173,
     0.01922859558,
     0.01929161019,
     0.01757429172,
     0.01718919668,
     0.01727900643,
     0.01726871934,
     0.01840365098,
     0.01863257057,
     0.01770576133,
     0.0176681309,
     0.01854505404,
     0.01871413643,
     0.01889156492,
     0.01775451713,
     0.01890181712};
Double_t          aDEBD[16]={0.01746301737,
     0.01646662074,
     0.01733790984,
     0.01729456138,
     0.0176089114,
     0.01800022339,
     0.01720842632,
     0.01809458661,
     0.01677852232,
     0.01658194538,
     0.0196410265,
     0.01861791946,
     0.0189869229,
     0.01660555217,
     0.01882006653,
     0.01731957766};
Double_t          aDEBL[16]={0.01784068733,
     0.01745241371,
     0.01751401757,
     0.01868882781,
     0.01725309553,
     0.01779389193,
     0.01741889506,
     0.01717549907,
     0.01829926828,
     0.0172825225,
     0.01826052424,
     0.01809581721,
     0.01752469161,
     0.01887935747,
     0.01762822402,
     0.01728213339};
```

```
Double_t          aPADAU=1.;
Double_t          aPADAR=1.;
Double_t          aPADAD=1.;
Double_t          aPADAL=1.;

Double_t          bAUF[32]={-1.30359,
      -1.59693,
      -1.41077,
      -1.44621,
      -1.42383,
      -1.40914,
      -1.40793,
      -1.31909,
      -1.34432,
      -1.12736,
      -1.43376,
      -1.02854,
      -1.53252,
      -1.15989,
      -1.82108,
      -1.65981,
      -1.50449,
      -2.00652,
      -1.9729,
      -1.37907,
      -1.51099,
      -1.90727,
      -1.68551,
      -1.54489,
      -1.44401,
      -1.81054,
      -1.62426,
      -1.25518,
      -1.67636,
        −1.48319,
        −1.53111,
        −1.20798};
Double_t              bAUB[32]={−0.838676,
        −1.22288,
        −0.793775,
        −1.12357,
        −1.30028,
        −1.08329,
        −1.03725,
        −1.15169,
        −1.04293,
        −0.93311,
        −1.25067,
        −1.0489,
        −0.924085,
        −0.802963,
        −1.21425,
        −0.972056,
        −1.2353,
        −1.364,
        −1.35631,
        −1.24405,
        −1.40619,
        −1.22807,
        −1.19983,
        −1.30325,
        −1.45861,
        −1.13244,
        −1.41911,
        −1.31678,
        −1.07606,
        −1.04434,
        −1.57985,
        -1.49561};
  Double_t              bARF[32]={-0.720277,
        -0.791525,
        -0.749622,
        -0.924137,
        -0.715488,
        -0.526329,
        -0.582755,
        -1.12052,
        -0.909766,
        -0.691523,
        -0.916585,
        -0.819426,
        -0.606293,
        -0.654577,
        -1.35158,
        -0.750895,
        -0.626762,
        -0.811408,
        -0.554127,
        -0.614592,
        -0.919797,
        -0.532556,
        -0.848954,
        -0.659268,
        -0.806736,
        -0.898733,
        -0.607672,
        -0.669598,
        -0.856331,
        -0.809864,
        -0.398237,
        -0.669066};
  Double_t              bARB[32]={-1.10027,
        -0.79936,
        -0.947672,
        -0.881303,
        -0.825697,
        -0.945892,
        -1.06807,
        -1.02863,
        -0.863877,
        -0.821463,
        -0.918465,
        -0.875341,
        -0.897115,
        -0.948606,
        -1.0476,
        -0.950321,
        -1.20031,
        -0.925015,
        -1.12142,
        -1.08375,
        -1.0864,
        -0.975564,
        -1.07049,
        -1.20791,
        -1.09994,
        -1.06745,
        -1.10944,
        -1.21496,
        -1.01455,
        -0.875292,
        -1.01204,
        -1.04516};
Double_t             bADF[32]={-0.904248,
        -1.16954,
        -1.14411,
        −0.856516,
        −0.725056,
        −0.796668,
        −1.12417,
        −1.39763,
        −1.33274,
        −1.53993,
        −1.0795,
        −1.22519,
        −1.12939,
        −1.44413,
        −1.25687,
        −1.84734,
        −1.11645,
        −1.37743,
        −1.06373,
        −1.42091,
        −1.56566,
        −1.18363,
        −1.27746,
        −1.13916,
        −1.24576,
        −1.26311,
        −1.30389,
        −1.6326,
        −1.06389,
        −1.07564,
        −1.27887,
        −1.56769};
Double_t            bADB[32]={−1.48962,
        −1.55606,
        −1.28522,
        −1.32968,
        −1.27494,
        −1.64484,
        −1.45574,
        −1.23969,
        −1.60001,
        −1.66853,
        −1.56943,
        −1.47959,
        −1.69897,
        −1.60066,
        −1.57009,
        −1.66279,
        −1.33894,
        −1.40142,
        −1.79776,
        −1.43549,
        −1.6904,
        −1.66736,
        −2.26254,
        −1.38679,
        −1.37727,
        −1.35673,
        −1.85385,
        −1.34244,
        −1.59002,
        −1.35967,
        −1.54606,
        −1.73892};
Double_t                bALF[32]={−1.47994,
        −1.59375,
        −1.58978,
        −1.29873,
        −1.57659,
        −1.26655,
        −1.49491,
          -1.39585,
          -1.2812,
          -1.35483,
          -1.68872,
          -1.44458,
          -1.67196,
          -1.31886,
          -1.12266,
          -1.53567,
          -1.59273,
          -1.63423,
          -1.41356,
          -1.16381,
          -1.33169,
          -1.49389,
          -1.45463,
          -1.48567,
          -1.20203,
          -1.10772,
          -1.17791,
          -1.34138,
          -1.48689,
          -1.55389,
          -1.95645,
          -1.27391};
Double_t              bALB[32]={-1.4291,
          -1.19263,
          -1.33693,
          -1.16765,
          -1.03758,
          -1.17575,
          -1.17763,
          -1.27785,
          -1.3804,
          -1.37239,
          -1.31601,
          -1.30989,
          -1.51021,
          -1.38279,
          -1.51847,
          -1.39165,
          -1.49777,
          -1.20593,
          -1.78065,
          -1.69509,
          -1.83649,
          -1.44959,
          -1.38811,
          -1.21419,
          -1.54547,
          -1.33227,
          -1.69902,
          -1.3877,
          -1.81288,
          -1.44479,
          -1.54879,
          -1.43615};
  Double_t            bBUF[16]={-1.30429,
          -1.46246,
          -1.60049,
          -1.49051,
          -1.47451,
          -2.13685,
          -1.6758,
          -1.57106,
          -2.00073,
          -1.67226,
          -1.84048,
          -1.87778,
          -1.98452,
          -2.12846,
          -1.87743,
          -1.66726};
  Double_t              bBUB[16]={-2.07949,
          -1.98395,
          -2.03066,
          -2.05891,
          -1.92563,
          -1.61902,
          -2.1923,
          -1.90906,
          -1.77777,
          -1.77435,
          -1.83193,
          -1.6567,
          -2.18525,
          -1.60198,
          -1.92518,
          -1.64115};
  Double_t              bBRF[16]={-1.95319,
          -1.68728,
          -1.4093,
          -1.70275,
          -5.99409,
          -2.27466,
          -1.88011,
          -2.19754,
          -2.80473,
          -1.81546,
          -1.68358,
          -1.75323,
          -1.76367,
          −2.03861,
          −2.14991,
          −1.80261};
   Double_t              bBRB[16]={−1.73652,
          −1.90824,
          −2.33374,
          −1.96733,
          −2.07375,
          −2.15533,
          −2.10696,
          −2.11741,
          −2.03448,
          −2.13297,
          −2.47,
          −1.99295,
          −2.24959,
          −1.81819,
          −2.7752,
          −2.16838};
   Double_t              bBDF[16]={−1.61637,
          −2.19954,
          −1.6699,
          −1.97947,
          −1.72261,
          −2.51974,
          −2.30281,
          −2.14864,
          −2.5036,
          −2.00098,
          −2.30356,
          −2.35025,
          −2.11687,
          −2.24067,
          −1.77997,
          −1.79537};
  Double_t                bBDB[16]={−2.35945,
          −1.70732,
          −2.49918,
          −1.97216,
          −1.99517,
          −2.09743,
          −2.15462,
          −2.39472,
          −2.26315,
          −1.95269,
          −1.78978,
          −1.83823,
          −1.94514,
          −2.18325,
          −2.32696,
          −2.16091};
  Double_t                bBLF[16]={−2.12543,
          −1.73569,
          −1.87858,
          −2.09517,
          −2.31802,
          −2.40165,
          −2.03515,
          −1.73018,
          −1.73962,
          −2.14935,
          −2.45234,
          −1.86502,
          −2.58643,
          −1.65797,
          −2.4212,
          −1.98785};
  Double_t                bBLB[16]={−1.75979,
          -1.90321,
          -2.53265,
          -2.20847,
          -2.08042,
          -2.61968,
          -2.45218,
          -1.8253,
          -1.92403,
          -2.31164,
          -2.19872,
          -2.67163,
          -1.9096,
          -1.93257,
          -2.03873,
          -2.12293};
    //Double_t          bDEBU[16]={0., 0., 0., 0., 0., 0., 0., 0., 0., 0.
    Double_t          bDEBU[16]={-1.449947138,
          -2.000943388,
          -1.795050962,
          -1.31813676,
          -1.033197743,
          -1.353081357,
          -2.15277497,
          -1.398647499,
          -1.385450445,
          -1.347855117,
          -1.418091956,
          -1.22654064,
          -2.283914303,
          -1.318588822,
          -3.102511561,
          -1.581642463};
    Double_t          bDEBR[16]={-1.508061071,
          -2.034099206,
          -2.092950715,
          -1.633552228,
          -1.517754568,
          -1.572935562,
          -1.611890981,
          -1.852299083,
          -1.914374895,
          -1.694243879,
          -1.712817137,
          -1.930514707,
          -1.827906426,
          -2.012278626,
          -0.912944674,
          -1.926738174};
    Double_t            bDEBD[16]={-0.7584953484,
          -1.413820644,
          -1.523735915,
          -1.560061599,
          -1.732828135,
          -1.80928959,
          -1.605686325,
          -1.665246865,
          -1.393490578,
          -1.406962529,
          -2.179975403,
          -2.00423207,
          -1.947677587,
          -1.350406616,
          -2.071049979,
          -1.538241979};
    Double_t            bDEBL[16]={-1.551725008,
          -1.590421409,
          -1.531080931,
          -1.796785549,
          -1.396170258,
          -1.6379469,
          -1.57830175,
          -1.569351492,
          -1.783820478,
          -1.440541279,
          -1.657066612,
          -1.741278948,
          -1.586254432,
          -1.896492448,
          -1.555020316,
          -1.586018008};
     Double_t          bPADAU=0.;
     Double_t          bPADAR=0.;
     Double_t          bPADAD=0.;
     Double_t          bPADAL=0.;

     cout << "Fatto␣output" << endl;

    Long64_t nentries = fChain->GetEntriesFast();

    Long64_t nbytes = 0, nb = 0;
    for (Long64_t jentry=0; jentry<nentries;jentry++) {
       Long64_t ientry = LoadTree(jentry);
       if (ientry < 0) break;
       nb = fChain->GetEntry(jentry);    nbytes += nb;
       // if (Cut(ientry) < 0) continue;

        if((AUF[1]>100 && AUF[1]<5000) && (AUF[2]>100 && AUF[2]<5000) &&


        for(int i=0; i<32; i++){
             EAUF[i]=aAUF[i]*AUF[i]+bAUF[i];
             EAUB[i]=aAUB[i]*AUB[i]+bAUB[i];
        EARF[i]=aARF[i]*ARF[i]+bARF[i];
        EARB[i]=aARB[i]*ARB[i]+bARB[i];
        EADF[i]=aADF[i]*ADF[i]+bADF[i];
        EADB[i]=aADB[i]*ADB[i]+bADB[i];
        EALF[i]=aALF[i]*ALF[i]+bALF[i];
        EALB[i]=aALB[i]*ALB[i]+bALB[i];
    }

    for(int i=0; i<16; i++){
        EBUF[i]=aBUF[i]*BUF[i]+bBUF[i];
        EBUB[i]=aBUB[i]*BUB[i]+bBUB[i];
        EBRF[i]=aBRF[i]*BRF[i]+bBRF[i];
        EBRB[i]=aBRB[i]*BRB[i]+bBRB[i];
        EBDF[i]=aBDF[i]*BDF[i]+bBDF[i];
        EBDB[i]=aBDB[i]*BDB[i]+bBDB[i];
        EBLF[i]=aBLF[i]*BLF[i]+bBLF[i];
        EBLB[i]=aBLB[i]*BLB[i]+bBLB[i];
    }

    for(int i=0; i<16; i++){
        EDEBU[i]=aDEBU[i]*DEBU[i]+bDEBU[i];
        EDEBR[i]=aDEBR[i]*DEBR[i]+bDEBR[i];
        EDEBD[i]=aDEBD[i]*DEBD[i]+bDEBD[i];
        EDEBL[i]=aDEBL[i]*DEBL[i]+bDEBL[i];
    }

    EPADAU=aPADAU*PADAU+bPADAU;
    EPADAR=aPADAR*PADAR+bPADAR;
    EPADAD=aPADAD*PADAD+bPADAD;
    EPADAL=aPADAL*PADAL+bPADAL;

    if(jentry%100000==0)
        cout << "Fatto␣calibrazione␣x100000" << endl;
```

```
            cal->Fill();

        }
        cal->Write();

        fout->Write();

        fout->Close();
}
```

Macro for the analysis of the experimental runs

```
#define Analisi_cxx
#include "Analisi.h"
#include <TH2.h>
#include <TStyle.h>
#include <TCanvas.h>
#include <TGeoManager.h>
#include <TGeometry.h>
#include <TNode.h>
#include <TGeoVolume.h>
#include <TGeoMatrix.h>
#include <TGeoMedium.h>
#include <TGeoMaterial.h>
#include <TGeoNode.h>
#include <TMaterial.h>
#include <TMixture.h>
#include <TShape.h>
#include <TString.h>
#include <Riostream.h>
#include <TFile.h>
#include <TMath.h>
#include <TLorentzVector.h>
#include <TSystem.h>
```

```
#include <stdlib.h>
#include <TROOT.h>
#include <TGenPhaseSpace.h>
#include <TStyle.h>
#include "TGeoPatternFinder.h"
#include "TSystem.h"
#include <TGeoNavigator.h>
#include <iostream>
#include <cstdlib>
#include <fstream>
#include <cmath>
#include "TApplication.h"
#include "TF1.h"
#include "TGraph.h"
#include "TGraphErrors.h"
#include "TH1.h"
#include "TLegend.h"
#include "TLegendEntry.h"
#include "TColor.h"
#include <TRandom3.h>
#include <time.h>
#include "TTree.h"
#include <random>

void Analisi::Loop()
{
//   In a ROOT session, you can do:
//      root> .L Analisi.C
//      root> Analisi t
//      root> t.GetEntry(12); // Fill t data members with entry number
//      root> t.Show();       // Show values of entry 12
//      root> t.Show(16);     // Read and show values of entry 16
//      root> t.Loop();       // Loop on all entries
//
```

```
//       This is the loop skeleton where:
//     jentry is the global entry number in the chain
//     ientry is the entry number in the current Tree
//   Note that the argument to GetEntry must be:
//     jentry for TChain::GetEntry
//     ientry for TTree::GetEntry and TBranch::GetEntry
//
//        To read only selected branches, Insert statements like:
// METHOD1:
//     fChain->SetBranchStatus("*",0);  // disable all branches
//     fChain->SetBranchStatus("branchname",1);  // activate branchname
// METHOD2: replace line
//     fChain->GetEntry(jentry);       //read all branches
//by  b_branchname->GetEntry(ientry); //read only this branch
   if (fChain == 0) return;

   Long64_t nentries = fChain->GetEntriesFast();

    TApplication theApp("App", NULL, NULL);

    gStyle->SetPalette(1);

    gSystem->Load("libPhysics.so");

    gSystem->Load("libGeom.so");

    TGeoManager *sc = new TGeoManager("scattcham","Scattering␣Chamber")

    TGeoMaterial *matVacuum = new TGeoMaterial("Vacuum",0,0,0);
    TGeoMaterial *matSi = new TGeoMaterial("Si",28.086,14,2.321);

    TGeoMedium *Vacuum = new TGeoMedium("Vacuum",1,matVacuum);
    TGeoMedium *Si = new TGeoMedium("Root␣Material",2,matSi);
```

```
TGeoVolume *top = sc->MakeSphere("TOP",Vacuum,0.,100.,0.,180.,0.,36
sc->SetTopVolume(top);

TGeoVolume *psd = sc->MakeBox("PSD",Si,2.5,0.05,2.5);

// from deg to rad
    double dtr=TMath::Pi()/180.;

// close geometry



// this is detector AR
    double thetaAR=56*dtr;
    double   phiAR=45*dtr;
    double  distAR=6.00;
    double     pxAR=distAR*sin(thetaAR)*cos(phiAR);
    double     pyAR=distAR*sin(thetaAR)*sin(phiAR);
    double     pzAR=distAR*cos(thetaAR);
    TVector3 vAR(pxAR,pyAR,pzAR); //position of det. AR wrt chamber
    TGeoRotation *rotAR = new TGeoRotation("rotAR",-45,34,0);
    TGeoCombiTrans *posAR = new TGeoCombiTrans(pxAR,pyAR,pzAR,rotAR

// this is detector AU
    double thetaAU=34*dtr;
    double   phiAU=(45+90)*dtr;
    double  distAU=6.02;
    double     pxAU=distAU*sin(thetaAU)*cos(phiAU);
    double     pyAU=distAU*sin(thetaAU)*sin(phiAU);
    double     pzAU=distAU*cos(thetaAU);
    TVector3 vAU(pxAU,pyAU,pzAU); //position of det. AU wrt chamber
    TGeoRotation *rotAU = new TGeoRotation("rotAU",45,56,0);
    TGeoCombiTrans *posAU = new TGeoCombiTrans(pxAU,pyAU,pzAU,rotAU
//this is detector AL
     double thetaAL=56*dtr;
     double   phiAL=(45+180)*dtr;
     double  distAL=6;
     double     pxAL=distAL*sin(thetaAL)*cos(phiAL);
     double     pyAL=distAL*sin(thetaAL)*sin(phiAL);
     double     pzAL=distAL*cos(thetaAL);
     TVector3 vAL(pxAL,pyAL,pzAL); //position of det. AL wrt chamber
     TGeoRotation *rotAL = new TGeoRotation("rotAL",-45,-34,0);
     TGeoCombiTrans *posAL = new TGeoCombiTrans(pxAL,pyAL,pzAL,rotAL

// this is detector AD
     double thetaAD=34*dtr;
     double   phiAD=(45+270)*dtr;
     double  distAD=6.02;
     double     pxAD=distAD*sin(thetaAD)*cos(phiAD);
     double     pyAD=distAD*sin(thetaAD)*sin(phiAD);
     double     pzAD=distAD*cos(thetaAD);
     TVector3 vAD(pxAD,pyAD,pzAD); //position of det. AD wrt chamber
     TGeoRotation *rotAD = new TGeoRotation("rotAD",45,-56,0);
     TGeoCombiTrans *posAD = new TGeoCombiTrans(pxAD,pyAD,pzAD,rotAD

// far geometry

// this is detector BR
     double thetaBR=5*dtr;
     double   phiBR=45*dtr;
     double  distBR=69.14;
     double     pxBR=distBR*sin(thetaBR)*cos(phiBR);
     double     pyBR=distBR*sin(thetaBR)*sin(phiBR);
     double     pzBR=distBR*cos(thetaBR);
     TVector3 vBR(pxBR,pyBR,pzBR); //position of det. B1 wrt chamber
     TGeoRotation *rotBR = new TGeoRotation("rotBR",-45,-90,0);
     TGeoCombiTrans *posBR = new TGeoCombiTrans(pxBR,pyBR,pzBR,rotBR
// this is detector BU
     double thetaBU=4*dtr;
     double   phiBU=(45+90)*dtr;
     double  distBU=71.06;
     double    pxBU=distBU*sin(thetaBU)*cos(phiBU);
     double    pyBU=distBU*sin(thetaBU)*sin(phiBU);
     double    pzBU=distBU*cos(thetaBU);
     TVector3 vBU(pxBU,pyBU,pzBU); //position of det. BU wrt chamber
     TGeoRotation *rotBU = new TGeoRotation("rotBU",45,-90,0);
     TGeoCombiTrans *posBU = new TGeoCombiTrans(pxBU,pyBU,pzBU,rotBU

// this is detector BL
     double thetaBL=5*dtr;
     double   phiBL=(45+180)*dtr;
     double  distBL=69.14;
     double    pxBL=distBL*sin(thetaBL)*cos(phiBL);
     double    pyBL=distBL*sin(thetaBL)*sin(phiBL);
     double    pzBL=distBL*cos(thetaBL);
     TVector3 vBL(pxBL,pyBL,pzBL); //position of det. B3 wrt chamber
     TGeoRotation *rotBL = new TGeoRotation("rotBL",-45,90,0);
     TGeoCombiTrans *posBL = new TGeoCombiTrans(pxBL,pyBL,pzBL,rotBL

// this is detector BD
     double thetaBD=4*dtr;
     double   phiBD=(45+270)*dtr;
     double  distBD=71.06;
     double    pxBD=distBD*sin(thetaBD)*cos(phiBD);
     double    pyBD=distBD*sin(thetaBD)*sin(phiBD);
     double    pzBD=distBD*cos(thetaBD);
     TVector3 vBD(pxBD,pyBD,pzBD); //position of det. BD wrt chamber
     TGeoRotation *rotBD = new TGeoRotation("rotBD",45,90,0);
     TGeoCombiTrans *posBD = new TGeoCombiTrans(pxBD,pyBD,pzBD,rotBD
// detector placing in space

top->AddNode(psd,1,posAR);
top->AddNode(psd,2,posAU);
top->AddNode(psd,3,posAL);
top->AddNode(psd,3,posAD);

top->AddNode(psd,5,posBR);
top->AddNode(psd,6,posBU);
top->AddNode(psd,7,posBL);
top->AddNode(psd,8,posBD);

gGeoManager->CloseGeometry();
top->SetLineColor(kRed);
gGeoManager->SetTopVisible();


// get the nodeid of each detector

gGeoManager->SetCurrentPoint(pxAR,pyAR,pzAR);
gGeoManager->FindNode();
TGeoNode *nodeAR = gGeoManager->GetCurrentNode();
Int_t idAR = gGeoManager->GetCurrentNodeId();
cout << "AR␣" << idAR << endl;

gGeoManager->SetCurrentPoint(pxAU,pyAU,pzAU);
gGeoManager->FindNode();
TGeoNode *nodeAU = gGeoManager->GetCurrentNode();
Int_t idAU = gGeoManager->GetCurrentNodeId();
cout << "AU␣" << idAU << endl;

gGeoManager->SetCurrentPoint(pxAL,pyAL,pzAL);
gGeoManager->FindNode();
TGeoNode *nodeAL = gGeoManager->GetCurrentNode();
```

```
Int_t idAL = gGeoManager->GetCurrentNodeId();
cout << "AL␣" << idAL << endl;

gGeoManager->SetCurrentPoint(pxAD,pyAD,pzAD);
gGeoManager->FindNode();
TGeoNode *nodeAD = gGeoManager->GetCurrentNode();
Int_t idAD = gGeoManager->GetCurrentNodeId();
cout << "AD␣" << idAD << endl;

gGeoManager->SetCurrentPoint(pxBR,pyBR,pzBR);
gGeoManager->FindNode();
TGeoNode *nodeBR = gGeoManager->GetCurrentNode();
Int_t idBR = gGeoManager->GetCurrentNodeId();
cout << "BR␣" << idBR << endl;

gGeoManager->SetCurrentPoint(pxBU,pyBU,pzBU);
gGeoManager->FindNode();
TGeoNode *nodeBU = gGeoManager->GetCurrentNode();
Int_t idBU = gGeoManager->GetCurrentNodeId();
cout << "BU␣" << idBU << endl;

gGeoManager->SetCurrentPoint(pxBL,pyBL,pzBL);
gGeoManager->FindNode();
TGeoNode *nodeBL = gGeoManager->GetCurrentNode();
Int_t idBL = gGeoManager->GetCurrentNodeId();
cout << "BL␣" << idBL << endl;

gGeoManager->SetCurrentPoint(pxBD,pyBD,pzBD);
gGeoManager->FindNode();
TGeoNode *nodeBD = gGeoManager->GetCurrentNode();
Int_t idBD = gGeoManager->GetCurrentNodeId();
cout << "BD␣" << idBD << endl;

//first number row, second column
```

```
const double th_DEBU[16][16]={
    {16.8355, 17.4152, 17.4207, 17.5226, 17.0615, 16.4657, 15.8059,
    {17.8539, 18.3921, 18.5543, 18.6432, 18.184, 17.5717, 16.9762,
    {18.8957, 19.2933, 19.5802, 19.609, 19.3661, 18.8681, 18.1912,
    {19.8096, 20.3598, 20.4522, 20.2866, 20.2274, 19.9146, 19.4917,
    {20.3465, 20.9535, 21.2307, 21.3698, 20.9806, 20.6734, 20.4166,
    {20.9323, 21.5843, 21.9137, 22.1127, 21.962, 21.573, 21.3558, 2
    {21.4189, 22.0779, 22.4413, 22.9147, 22.7344, 22.4886, 22.009,
    {21.7016, 22.4091, 22.9388, 23.3139, 23.5723, 23.4886, 23.2946,
    {21.9029, 22.64, 23.2509, 23.7255, 23.9626, 24.1791, 24.1602, 2
    {22.2914, 22.9631, 23.7111, 24.2519, 24.4882, 24.5904, 24.4666,
    {22.9004, 23.7912, 24.2023, 25.2043, 25.6985, 25.5602, 24.6164,
    {23.3642, 24.1574, 25.1092, 25.6177, 25.8727, 26.0889, 25.8156,
    {23.6308, 24.2732, 25.3137, 25.7428, 26.0542, 26.2195, 26.2864,
    {23.8492, 24.1572, 25.4114, 25.7922, 25.9352, 26.3613, 26.2402,
    {23.7429, 24.1911, 25.3686, 25.5215, 25.4714, 25.9228, 25.9117,
    {23.4244, 23.9922, 24.2641, 24.4101, 24.9568, 24.7542, 24.515,
};

const double th_DEBD[16][16]={
    {14.4941, 18.3351, 19.6828, 21.0938, 21.7032, 22.2515, 22.6902,
    {14.6477, 18.3396, 19.6221, 21.0726, 21.8033, 22.4206, 22.9041,
    {15.9383, 17.9312, 19.6007, 20.7885, 21.7093, 22.4745, 23.1102,
    {15.0283, 17.1452, 19.4815, 20.7232, 21.7086, 22.6729, 23.5403,
    {13.4174, 17.4117, 19.1239, 20.7651, 21.9353, 22.7082, 23.5158,
    {15.5718, 17.5992, 19.3462, 20.7369, 21.751, 22.676, 23.6406, 2
    {15.6704, 17.706, 19.212, 20.7206, 21.5426, 22.5959, 23.5042, 2
    {16.7699, 17.6095, 19.2773, 20.4236, 21.7354, 22.4518, 23.3482,
    {16.8524, 17.8622, 19.2762, 20.615, 21.5382, 22.4047, 23.1349,
    {16.7224, 17.402, 18.9013, 20.1395, 21.1542, 21.9601, 22.8653,
    {16.4633, 17.6481, 18.6517, 19.76, 20.9292, 21.5441, 22.7127, 2
    {17.0506, 17.8376, 18.1266, 19.4419, 20.1678, 21.4061, 22.4775,
    {17.0179, 17.7934, 18.1989, 18.7063, 19.6557, 20.8636, 22.1805,
    {17.244, 17.0534, 18.0528, 18.1494, 19.0309, 20.2261, 21.4724,
    {17.5033, 17.3318, 17.9342, 17.6954, 18.2199, 19.4704, 20.3737,
    {17.067, 17.3049, 17.2381, 17.1949, 16.8947, 18.1626, 19.1598,
};

const double th_DEBL[16][16]={
    {18.2872, 18.326, 18.2684, 18.8629, 19.1834, 19.392, 19.5484, 1
    {18.084, 18.3608, 18.5317, 19.0441, 19.3511, 19.6805, 19.9723,
    {17.8571, 18.1875, 18.6135, 18.8866, 19.1686, 19.7395, 20.1687,
    {17.5936, 18.1625, 18.5276, 18.7853, 19.008, 19.7492, 20.3789,
    {17.5313, 18.0035, 18.3132, 18.7618, 19.0212, 19.6513, 20.2157,
    {17.6635, 18.0054, 18.1731, 18.5524, 18.8691, 19.6166, 20.2678,
    {17.8525, 18.0302, 18.0702, 18.5636, 18.6317, 19.3139, 20.018,
    {17.9663, 18.22, 18.2942, 18.4287, 18.7581, 18.9287, 19.6142, 1
    {18.2226, 18.5667, 18.5903, 18.4697, 18.4882, 18.6088, 19.0003,
    {18.7023, 18.8463, 18.7733, 18.6061, 18.4227, 18.3368, 18.5463,
    {19.01, 19.3469, 19.0297, 18.9183, 18.703, 18.3756, 18.2771, 18
    {19.2978, 19.5147, 19.4044, 19.2333, 18.8427, 18.6972, 18.2759,
    {19.5169, 19.7136, 19.555, 19.3565, 19.1646, 18.7989, 18.3427,
    {19.8079, 19.6612, 19.6341, 19.4776, 19.0845, 18.8684, 18.2242,
    {19.6921, 19.5627, 19.5975, 19.3174, 18.8124, 18.4662, 17.6742,
    {19.0341, 19.2973, 19.0336, 18.8173, 18.2515, 17.9841, 17.1646,
};

const double th_DEBR[16][16]={
    {16.2715, 16.7138, 16.8561, 17.1471, 17.089, 16.8454, 16.9712,
    {16.3279, 16.9896, 17.2626, 17.5676, 17.5123, 17.4959, 17.8851,
    {16.7678, 17.1346, 17.6024, 17.9091, 18.1093, 18.4656, 18.9724,
    {17.0407, 17.6229, 18.1702, 18.7534, 19.3304, 20.19, 21.8212, 2
    {17.536, 18.2733, 19.1518, 20.0839, 20.8547, 21.9255, 22.6881,
    {16.1721, 19.2803, 20.3134, 21.2355, 22.0373, 22.9279, 23.5256,
    {18.5311, 19.6534, 20.8703, 22.0761, 22.8935, 23.5401, 23.8636,
    {18.566, 19.8732, 21.3085, 22.5585, 23.5421, 23.8558, 23.9873,
    {18.7272, 20.1654, 21.5497, 22.6641, 23.4586, 23.8576, 23.8247,
    {19.0103, 20.2545, 21.5462, 22.6161, 23.3167, 23.6131, 23.6851,
        {19.402 ,  20.4257 ,  21.4847 ,  22.6224 ,  23.3105 ,  23.4646 ,  23.5561 ,
        {19.3947 ,  20.199 ,  21.2128 ,  22.3631 ,  22.867 ,  23.3079 ,  23.377 ,  23
        {19.0131 ,  19.7925 ,  20.7143 ,  21.7502 ,  22.3813 ,  22.7748 ,  23.1017 ,
        {18.7143 ,  19.1258 ,  20.2253 ,  21.1852 ,  21.6638 ,  22.4222 ,  22.7063 ,
        {18.0196 ,  18.5344 ,  19.6848 ,  20.5953 ,  21.0135 ,  21.7185 ,  22.0799 ,
        {17.4869 ,  18.0613 ,  18.8173 ,  19.8276 ,  20.3525 ,  20.949 ,  21.1661 ,
    };


  const  double  corr_DEBL[16][16]={
    {−0.72 ,   −0.884 ,   −0.5824 ,   −0.2934 ,   −0.0325 ,   0.0887 ,
−0.3084 ,   −1.0102 ,   −1.3223 ,   −1.6811 ,   −1.707 ,   −1.6284 ,
−1.3349 ,   −0.4509 ,   0.2189 ,   0.4166} ,
    {−1.1654 ,   −1.2729 ,   −1.2169 ,   −1.0785 ,   −0.8692 ,   −0.895 ,
−1.5036 ,   −2.0401 ,   −2.3492 ,   −2.5349 ,   −2.3848 ,   −2.0368 ,
−1.5892 ,   −0.7693 ,   −0.0467 ,   0.339} ,
    {−1.1325 ,   −1.2397 ,   −1.4175 ,   −1.4845 ,   −1.4553 ,   −1.727 ,
−2.5307 ,   −3.0802 ,   −3.2329 ,   −3.1821 ,   −2.7612 ,   −2.035 ,
−1.437 ,   −0.5841 ,   0.4606 ,   1.5049} ,
    {−0.8274 ,   −1.1648 ,   −1.446 ,   −1.6752 ,   −1.8709 ,   −2.2452 ,
−3.0203 ,   −3.569 ,   −3.7285 ,   −3.5985 ,   −2.8824 ,   −1.8449 ,
−1.2727 ,   −0.4618 ,   0.6155 ,   1.8597} ,
    {−0.0716 ,   −0.8131 ,   −1.3571 ,   −1.6903 ,   −1.9297 ,   −2.2088 ,
−2.9788 ,   −3.5601 ,   −3.825 ,   −3.767 ,   −2.8542 ,   −1.778 ,   −1.302 ,
−0.4097 ,   0.7385 ,   1.8} ,
    {0.8933 ,   −0.1277 ,   −1.0032 ,   −1.4382 ,   −1.5766 ,   −1.7552 ,
−2.6643 ,   −3.496 ,   −3.8975 ,   −3.69 ,   −2.6208 ,   −1.5683 ,   −0.9536 ,
0.0345 ,   0.8349 ,   1.6911} ,
    {1.9071 ,   0.7772 ,   −0.3768 ,   −0.8999 ,   −0.9948 ,   −1.1276 ,
−2.0247 ,   −3.1167 ,   −3.4809 ,   −3.2652 ,   −2.087 ,   −1.2368 ,
−0.8835 ,   0.2959 ,   0.7646 ,   1.4394} ,
    {2.8775 ,   1.808 ,   0.7602 ,   0.0662 ,   −0.2762 ,   −0.2047 ,
−0.861 ,   −1.9066 ,   −2.5124 ,   −2.3617 ,   −1.1739 ,   −0.7293 ,
−0.7212 ,   −0.1358 ,   0.3843 ,   1.1808} ,
    {3.8204, 2.8635, 2.0707, 1.2255, 0.5244, 0.7774,
0.5209, −0.3304, −1.0884, −1.1063, −0.2567, −0.524, −0.752,
−0.6188, −0.2224, 0.5601},
    {4.2877, 3.4503, 2.712, 1.9704, 1.408, 1.4871, 1.4173,
0.8471, 0.144, 0.0482, 0.1971, −0.4376, −1.2208, −1.2271,
−0.9225, −0.1182},
    {4.0378, 3.6189, 4.9384, 2.3553, 2.3063, 2.3402,
2.4432, 1.8558, 1.0489, 0.5615, 0.133, −0.7234, −1.7976,
−1.8655, −1.5737, −0.8017},
    {3.2491, 3.3017, 2.5635, 2.3185, 2.6916, 2.8978,
2.9493, 2.3187, 1.5472, 0.5999, −0.3869, −1.363, −2.3125,
−2.5723, −2.0423, −1.4616},
    {3.3821, 3.0223, 2.4606, 2.3033, 2.3196, 2.6255,
2.7728, 2.391, 1.9141, 0.5863, −0.7128, −1.8087, −2.6053,
−2.9611, −2.6291, −2.0151},
    {3.8753, 3.006, 2.216, 1.9576, 1.8153, 2.3698, 2.9988,
3.086, 2.3478, 1.1815, −0.2921, −1.6394, −2.4823, −2.9585,
−2.638, −2.217},
    {4.3285, 3.342, 2.3545, 1.5965, 2.0478, 3.1393, 3.84,
4.0501, 3.3384, 2.4869, 0.693, −0.9193, −1.8947, −2.3551,
−2.0729, −1.7962},
    {4.7579, 3.671, 3.0264, 2.7591, 3.3841, 4.3248, 4.0365,
4.7451, 3.8451, 3.1528, 1.8911, 0.3383, −0.6777, −1.2355,
−1.0854, −0.3547},
  };


  const double corr_DEBU[16][16]={
    {12.0685, 11.6946, 11.6272, 12.3607, 13.2566, 13.8348,
13.8797, 13.1972, 12.4256, 11.5298, 10.3593, 9.5169,
8.4284, 8.2337, 8.4924, 9.2955},
    {11.2728, 11.0867, 10.4678, 10.8739, 11.8567, 12.1878,
11.6924, 10.8685, 9.9292, 9.0396, 7.9229, 6.9347, 6.1872,
5.9483, 6.4159, 7.2725},
    {10.5, 10.0809, 9.4275, 9.1344, 9.4305, 9.5042, 9.0089,
8.1757, 7.2522, 6.3117, 5.0184, 4.0352, 3.677, 3.8586,
4.401, 5.4243},
    {9.7302, 8.831, 7.796, 6.9775, 6.8616, 6.7724, 6.1939,
5.4416, 4.6105, 3.7667, 2.7399, 2.3071, 2.1994, 1.801,
2.118, 3.2638},
    {9.1808, 7.7423, 6.1211, 4.8781, 4.312, 4.1755, 3.7862,
3.0172, 2.0596, 1.3595, 1.2165, 0.6163, −0.0313, 0.1533,
0.8869, 2.2619},
    {8.2719, 6.4538, 4.6811, 3.1852, 2.3014, 1.9605,
1.6702, 0.9207, 0.015, −0.1079, −0.6941, −1.6339, −1.8516,
−1.2131, −0.2854, 1.2858},
    {7.0965, 5.1544, 3.4116, 1.8793, 0.8401, 0.1884,
−0.3864, −0.9531, −1.3863, −2.1254, −3.2366, −3.6434,
−3.5789, −2.6405, −1.4063, 0.2815},
    {6.0869, 4.2214, 2.5492, 0.9569, −0.2023, −1.1488,
−2.2794, −2.4623, −3.528, −5.4701, −5.5032, −4.9857,
−4.6066, −3.682, −2.2095, −0.5763},
    {5.2616, 3.4424, 1.7068, −0.0448, −1.281, −2.4156,
−3.4642, −3.7958, −5.9984, −7.6703, −7.2616, −6.7626,
−6.1145, −4.6619, −2.8684, −1.2342},
    {4.5779, 2.9315, 1.2414, −0.6069, −2.1581, −3.6953,
−4.5726, −5.5051, −7.1037, −7.9779, −8.4523, −8.4518,
−7.683, −6.2435, −4.1738, −2.2002},
    {3.7688, 2.3602, 0.8756, −0.9751, −2.916, −4.678,
−5.9381, −7.6551, −8.5139, −9.1386, −9.7002, −9.4998,
−8.5532, −7.3497, −5.7659, −3.6707},
    {3.5995, 1.9612, 0.4197, −1.4645, −3.5921, −5.3591,
−6.9381, −8.8855, −10.2329, −10.9414, −10.8555, −10.1966,
−9.4093, −8.2642, −6.6517, −4.7846},
    {3.8201, 1.8094, 0.0864, −1.7952, −3.9106, −5.6378,
−7.458, −9.2364, −10.4189, −11.1506, −11.0396, −10.2794,
−9.5207, −8.4939, −6.9644, −5.1888},
    {4.1128, 2.0003, 0.1356, −1.8068, −3.7587, −5.4932,
−7.6334,  −8.9907,  −10.0143,  −10.525,  −10.287,  −9.5591,
−9.0271,  −8.4025,  −6.8585,  −5.3456},
    {4.2244,  2.0233,  0.1684,  −1.4278,  −2.8647,  −4.3376,
−6.6836,  −8.2914,  −8.977,  −8.9668,  −8.6714,  −8.2479,
−7.9113,  −7.2604,  −6.2573,  −4.8},
    {4.3892,  2.2822,  0.481,  −0.9217,  −1.4536,  −2.5867,
−5.0737,  −6.5098,  −6.7585,  −6.8774,  −6.7872,  −6.8819,
−6.5204,  −5.4192,  −4.4571,  −3.3678},
  };


  const double corr_DEBD[16][16]={
    {1.7707,  0.0911,  −1.4534,  −2.3925,  −3.1232,  −3.725,
−3.1009,  −2.4963,  −1.853,  −0.8099,  0.2999,  1.4525,  3.0428,
5.564,  8.5535,  11.0052},
    {0.3265,  −1.4346,  −3.1101,  −4.3422,  −4.8732,  −5.1707,
−4.441,  −3.5279,  −2.3669,  −1.374,  −0.2714,  1.1577,  3.0067,
5.5562,  8.4072,  10.8152},
    {−0.9425,  −2.6298,  −4.5124,  −6.0304,  −6.6202,  −6.7934,
−6.0526,  −4.9261,  −3.4448,  −2.1346,  −0.9077,  0.8402,
3.0719,  5.7878,  8.7086,  11.4598},
    {−1.5858,  −3.363,  −5.3376,  −7.1075,  −7.7369,  −8.0594,
−7.3773,  −6.2136,  −4.6122,  −2.9646,  −1.2765,  0.8243,
3.3162,  6.2869,  9.4984,  12.3676},
    {−1.6556,  −3.4804,  −5.5639,  −7.6217,  −8.5637,  −9.0054,
−8.1966,  −6.837,  −5.2384,  −3.3436,  −1.2066,  0.836,  3.4049,
6.6562,  10.0323,  12.9924},
    {−1.472,  −3.2933,  −5.3734,  −7.707,  −9.1487,  −9.5968,
−8.6448,  −7.1305,  −5.3635,  −3.3506,  −1.0443,  0.9276,
3.4912,  6.7498,  10.001,  12.6193},
    {−1.0251,  −2.8682,  −5.0448,  −7.4297,  −9.1812,  −9.932,
−8.8876,  −7.0731,  −5.2526,  −3.1935,  −1.0119,  1.1306,
3.6048,  6.6304,  9.6506,  11.7805},
    {−0.3122,  −2.0512,  −4.49,  −6.8208,  −8.6924,  −9.5885,
-8.6474, -7.1748, -5.1279, -3.0576, -0.9047, 1.2855,
3.6634, 6.5583, 9.4219, 11.1998},
    {-0.1898, -1.6395, -4.0622, -6.4776, -7.9733, -8.9103,
-8.6441, -7.7894, -5.3006, -2.5645, -0.5, 1.4137, 3.5782,
6.5838, 9.5487, 10.8057},
    {-0.4058, -1.1529, -3.2668, -5.7482, -7.3827, -8.5748,
-8.2404, -7.3061, -5.0866, -1.9855, 0.477, 2.3815, 4.2853,
6.9831, 9.7537, 10.6889},
    {-0.3825, -0.8159, -2.3471, -4.6941, -6.8725, -8.4535,
-7.8209, -6.8325, -4.5999, -1.491, 1.33, 3.4061, 5.4484,
7.7636, 9.772, 10.4401},
    {0.1719, -0.5275, -2.0724, -4.052, -6.0173, -7.61,
-6.8928, -5.7739, -3.8701, -1.0915, 1.9343, 4.412, 6.5743,
8.4687, 9.9597, 10.0859},
    {0.3597, -0.6314, -2.1509, -3.735, -5.2806, -6.2962,
-5.611, -4.5373, -2.5693, 0.1868, 2.9974, 5.7225, 7.8483,
8.9391, 9.848, 9.8676},
    {0.4839, -0.6961, -2.1515, -3.3807, -4.3067, -4.8603,
-4.0037, -2.541, -0.5676, 2.0373, 4.7529, 7.1274, 8.8929,
9.4273, 9.8265, 9.4547},
    {0.6332, -0.7049, -1.7716, -2.5125, -3.0903, -3.2598,
-1.8823, -0.262, 2.0724, 4.6513, 7.1382, 9.025, 9.9166,
10.2275, 10.2158, 9.5413},
    {1.2706, 0.2957, -0.7614, -1.7102, -1.3591, -0.7797,
0.4416, 2.1282, 4.5397, 7.0931, 9.2858, 10.7025, 11.2008,
11.3966, 10.9127, 10.2384},
  };


    Double_t valuef=0;
    Double_t valueb=0;
    Int_t j,jj;
    Int_t nevent=0;
    Double_t EAr=0,EA=0,TA=0,PA=0,TAd=0,PAd=0;  // energia ricostr., en
    Double_t EBr=0,EB=0,TB=0,PB=0,TBd=0,PBd=0;  // energia ricostr., en
    Double_t Q=0,ES=0,TS=0,PS=0,TSd=0,PSd=0,e12=0,e13=0,e23=0,ecm=0,tcm
// qvalore, energia, theta, phy spettatore e in degrees

    Int_t xhitA=0,yhitA=0,xhitB=0,yhitB=0;      // strip colpita
    Double_t posxA=0,posyA=0,posxB=0,posyB=0;   // posizione randomizza
    Double_t p1=0,p2=0;
    Double_t DEA1=0,EA1=0,DEA2=0;
    Double_t DEB1=0,EB1=0,DEB2=0;

    Double_t CDEB=0.;
    Double_t CDEBU[16]={0.};
    Double_t CDEBD[16]={0.};
    Double_t CDEBL[16]={0.};
    Double_t CDEBR[16]={0.};

    Double_t      u=931.49410242;
 // unita' di massa (rif. 12C) in MeV/c

    Double_t      mp=25.97602/*25.982592929*/; Double_t
zp=13;
 // massa proiettile

    Double_t      mt=2.01355/*2.01410177785*/; Double_t
zt=1;
 // massa target

    Double_t      m2=25.97776/*25.982592929*/;
 // massa prima part. uscente 26Mg

    Double_t      m1=1.00728/*1.00782503207*/;
 // massa seconda part. uscente p1
```

```
    Double_t        m3=/*1.00728*/1.00782503207;
// massa terza part. uscente p2

    Double_t        mx=/*1.00866*/1.00866491574;
// massa part. trasferita n

    Double_t        mNa=/*25.97602*/22.98374;
// massa 23Na

    Double_t        mHe=/*25.97602*/4.00151;
// massa 4He

    Double_t        mAu=196.92379;
// massa 197Au

    Double_t        rm=mp/mt;

// Q della reazione
    Double_t        q3=u*(mp+mt-m1-m2-m3);
    cout << "q3:␣" << q3 << endl;

    Double_t        qn=u*(mp+mt-m1-mNa-m3);
    cout << "qn:␣" << qn << endl;

//(Momentum, Energy units are Gev/C, GeV)
    Double_t masses[3] = { u*m1, u*m2, u*m3 } ;

    Double_t deg=180./TMath::Pi();
    Double_t radius=100.;

    Double_t step=0.3125; //strip width in cm
    Double_t stepA=0.3125/2; //strip width in cm for 32 strip detectors

    // thickness half-target (in um)
// Double_t thickness=0.023302781; // for 0018, 0019
// Double_t thickness=0.023043861; // for 0020
// Double_t      thickness=1.4524146/2; // for CD2
Double_t      thickness=0.046605562/2; // for 0008, 0009

// energy loss
Double_t E1=0;
Double_t E2=0;
Double_t E3=0;
Double_t Ereal=0;
Double_t Eapp=0;
Double_t R1=0;
Double_t R2=0;
Double_t R3=0;
Double_t aMgCD=0.00296283;
Double_t bMgCD=0.566994;
Double_t cMgCD=4.85825;
Double_t apCD=10.8609;
Double_t bpCD=67.4126;
Double_t cpCD=-159.25;
Double_t aMgAl=0.000972343;
Double_t bMgAl=0.269383;
Double_t cMgAl=2.98099;
Double_t apAl=3.8405;
Double_t bpAl=32.9213;
Double_t cpAl=-72.6179;
Double_t aAlAu=0.000148004;
Double_t bAlAu=0.0956982;
Double_t cAlAu=1.99091;
Double_t aAlAl=0.000929501;
Double_t bAlAl=0.232333;
Double_t cAlAl=3.08366;
Int_t mA=0;
Int_t mB=0;
```

```
    Int_t mA1B1=0;
    Int_t mA2B1=0;
    Int_t mA1B2=0;
    Int_t mA2B2=0;
    Int_t mA0B0=0;
    Int_t mA0B1=0;
    Int_t mA1B0=0;
    Int_t mAD=0;
    Int_t mAU=0;
    Int_t mAR=0;
    Int_t mAL=0;
    Int_t mBD=0;
    Int_t mBU=0;
    Int_t mBR=0;
    Int_t mBL=0;
    Double_t aA=-0.000868214;
    Double_t bA=0.0402821;
    Double_t cA= -0.758212;
    Double_t dA=7.39883;
    Double_t eA=-38.7164;
    Double_t fA=102.426;

    //4-momenta
    TLorentzVector pp1; //p1 4-momentum in GeV/c
    TLorentzVector pp2; //26Mg 4-momentum in GeV/c
    TLorentzVector pp3; //prot.2 4-momentum in GeV/c

    //vector momenta (parte spaziale del 4-momentum)
    TVector3 ppp1;
    TVector3 ppp2;
    TVector3 ppp3;

    Int_t nnevent=0;
/*
```

```
long sec;
time( &sec );
TRandom3* ran = new TRandom3((unsigned)sec);
*/
TRandom3* r3 = new TRandom3();



  TFile *fout = new TFile("An_CD2_BD.root", "RECREATE");

  TTree *h1 = new TTree("h1", "h1");
  h1->Branch("EAUF[32]", EAUF, "EAUF[32]/D");
  h1->Branch("EAUB[32]", EAUB, "EAUB[32]/D");
  h1->Branch("EARF[32]", EARF, "EARF[32]/D");
  h1->Branch("EARB[32]", EARB, "EARB[32]/D");
  h1->Branch("EADF[32]", EADF, "EADF[32]/D");
  h1->Branch("EADB[32]", EADB, "EADB[32]/D");
  h1->Branch("EALF[32]", EALF, "EALF[32]/D");
  h1->Branch("EALB[32]", EALB, "EALB[32]/D");
  h1->Branch("EBUF[16]", EBUF, "EBUF[16]/D");
  h1->Branch("EBUB[16]", EBUB, "EBUB[16]/D");
  h1->Branch("EBRF[16]", EBRF, "EBRF[16]/D");
  h1->Branch("EBRB[16]", EBRB, "EBRB[16]/D");
  h1->Branch("EBDF[16]", EBDF, "EBDF[16]/D");
  h1->Branch("EBDB[16]", EBDB, "EBDB[16]/D");
  h1->Branch("EBLF[16]", EBLF, "EBLF[16]/D");
  h1->Branch("EBLB[16]", EBLB, "EBLB[16]/D");
  h1->Branch("EDEBU[16]", EDEBU, "EDEBU[16]/D");
  h1->Branch("EDEBR[16]", EDEBR, "EDEBR[16]/D");
  h1->Branch("EDEBD[16]", EDEBD, "EDEBD[16]/D");
  h1->Branch("EDEBL[16]", EDEBL, "EDEBL[16]/D");
  h1->Branch("CDEB", &CDEB, "CDEB/D");
  h1->Branch("CDEBU", CDEBU, "CDEBU[16]/D");
  h1->Branch("CDEBR", CDEBR, "CDEBR[16]/D");
```

```
h1->Branch("CDEBD",   CDEBD,   "CDEBD[16]/D");
h1->Branch("CDEBL",   CDEBL,   "CDEBL[16]/D");
h1->Branch("EPADAU",  &EPADAU,  "EPADAU/D");
h1->Branch("EPADAR",  &EPADAR,  "EPADAR/D");
h1->Branch("EPADAD",  &EPADAD,  "EPADAD/D");
h1->Branch("EPADAL",  &EPADAL,  "EPADAL/D");

h1->Branch("nevent", &nevent, "nevent/I");
h1->Branch("EA",  &EA,  "EA/D");
h1->Branch("EAr", &EAr, "EAr/D"); //"r" for reconstructed
h1->Branch("TAd", &TAd, "TAd/D"); //"d" for degrees
h1->Branch("PAd", &PAd, "PAd/D");
h1->Branch("EB",  &EB,  "EB/D");
h1->Branch("EBr", &EBr, "EBr/D");
h1->Branch("TBd", &TBd, "TBd/D");
h1->Branch("PBd", &PBd, "PBd/D");
h1->Branch("ES",  &ES,  "ES/D");
h1->Branch("TSd", &TSd, "TSd/D");
h1->Branch("PSd", &PSd, "PSd/D");
h1->Branch("Q", &Q, "Q/D");
h1->Branch("e12", &e12, "e12/D");
h1->Branch("e13", &e13, "e13/D");
h1->Branch("e23", &e23, "e23/D");
h1->Branch("ecm", &ecm, "ecm/D");
h1->Branch("tcm", &tcm, "tcm/D");
h1->Branch("ps", &ps, "ps/D"); // in MeV/c
h1->Branch("romx", &romx, "romx/D");
h1->Branch("romy", &romy, "romy/D");
h1->Branch("mA", &mA, "mA/I");
h1->Branch("mB", &mB, "mB/I");

//interesting histograms
TH1F *he=new TH1F("he","histogram calibrated",8000,1,80);
TH2F *hptp=new TH2F("hptp","th-ph positions",3600,-180,180,900,0,90
  // theta and energy of the detected particles
  TH2F *h2_e_t_a  = new TH2F("h2_e_t_a","en-th positions", 900,0,90,


Long64_t nbytes = 0, nb = 0;
for (Long64_t jentry=0; jentry<nentries; jentry++) {
   Long64_t ientry = LoadTree(jentry);
   if (ientry < 0) break;
   nb = fChain->GetEntry(jentry);   nbytes += nb;
   // if (Cut(ientry) < 0) continue;


  // Double_t rp = ran->Rndm();


   // energia e impulso del proiettile
      Double_t      ep=(90.92919-0.1914); //elastico su oro
      // Double_t      ep=(90.92919-(1.4824/2)); //CD2
      Double_t      pp=sqrt(2.*u*mp*ep);

      TLorentzVector target(0.0, 0.0, 0.0, u*mt);
      TLorentzVector beam(0.0, 0.0, pp, u*mp+ep);
      TLorentzVector W = beam + target;

  mA=0;
  mB=0;

     for (j=0; j<32; j++){
         valuef = EAUF[j];
         if(valuef>2 && valuef<120){
             for (jj=0; jj<32; jj++){
                 valueb = EAUB[jj];
                 if(valuef>valueb-0.5 && valuef<valueb+0.5){
```

```
mAU++;
Eapp = (valuef+valueb)/2;
Ereal = aA*Eapp*Eapp*Eapp*Eapp*Eapp+bA*Eapp*Eapp*
// EAr = (valuef+valueb)/2;
xhitA = j;
yhitA = jj;
posxA=(xhitA-16)*stepA+stepA/2+r3->Uniform(-stepA
//negative position closer to beam axis
posyA=(yhitA-16)*stepA+stepA/2+r3->Uniform(-stepA
//negative position bottom of the chamber

 Double_t avdAUr[3]={posxA,0.,posyA};
 Double_t avdAU[3]={0.,0.,0.};

 //we go back to the real position in the space
 rotAU->LocalToMaster(avdAUr,avdAU);

 //this is the vector from det.A1 center to appar
 TVector3 vdAUa(avdAU[0],avdAU[1],avdAU[2]);

 //apparent trajectory of 12C
 TVector3 vbinAU=vAU+vdAUa;
 TVector3 ubinAU=vbinAU.Unit();
 TA=vbinAU.Theta();
 PA=vbinAU.Phi();
 TAd=(vbinAU.Theta())*deg;
 PAd=(vbinAU.Phi())*deg;

 //energy losses
 R1=apAl*Ereal*Ereal+bpAl*Ereal+cpAl+0.8;
 E1=(-bpAl+sqrt(bpAl*bpAl-4*apAl*(cpAl-R1)))/(2*a

 R2=(apCD*E1*E1+bpCD*E1+cpCD)+(thickness/cos(TA))
 E2=(-bpCD+sqrt(bpCD*bpCD-4*apCD*(cpCD-R2)))/(2*a
          EAr=E2; //energy after reaction
          // Double_t EA=EAr/1000.;

        /*
        R1=aAlAl*Eapp*Eapp+bAlAl*Eapp+cAlAl+0.8;
        E1=(-bAlAl+sqrt(bAlAl*bAlAl-4*aAlAl*(cAlAl-R1)))/(

        R2=(aAlAu*E1*E1+bAlAu*E1+cAlAu)+(thickness/cos(TA)
        E2=(-bAlAu+sqrt(bAlAu*bAlAu-4*aAlAu*(cAlAu-R2)))/(
        EAr=E2; //energy after reaction
        */
          p1=sqrt(2*m1*u*EAr); //alpha momentum in GeV/c

          TVector3 vp1=p1*ubinAU; //alpha momentum (vector

          pp1.SetPx(vp1(0));
          pp1.SetPy(vp1(1));
          pp1.SetPz(vp1(2));
          pp1.SetE(u*m1+EAr);

          mA++;

          //filling histograms
          he->Fill(EAr);

          hptp->Fill(PA+deg, TA*deg);

          h2_e_t_a->Fill(TA*deg, EAr);
          h1->Fill();
      }
    }
  }
}
```

```
for (j=0; j<32; j++){
     valuef = EADF[j];
     if(valuef>2 && valuef<120){
          for (jj=0; jj<32; jj++){
               valueb = EADB[jj];
               if(valuef>valueb-0.5 && valuef<valueb+0.5){
                    mAD++;
                    Eapp = (valuef+valueb)/2;
                    Ereal = aA*Eapp*Eapp*Eapp*Eapp*Eapp+bA*Eapp*Eapp
                    // EAr = (valuef+valueb)/2;
                    xhitA = j;
                    yhitA = jj;
                    posxA=(xhitA-16)*stepA+stepA/2+r3->Uniform(-stepA
                    //negative position closer to beam axis
                    posyA=(yhitA-16)*stepA+stepA/2+r3->Uniform(-stepA
                    //negative position bottom of the chamber

                    Double_t avdADr[3]={posxA,0.,posyA};
                    Double_t avdAD[3]={0.,0.,0.};

                    //we go back to the real position in the space
                    rotAD->LocalToMaster(avdADr,avdAD);

                    //this is the vector from det.A1 center to appar
                    TVector3 vdADa(avdAD[0],avdAD[1],avdAD[2]);

                    //apparent trajectory of 12C
                    TVector3 vbinAD=vAD+vdADa;
                    TVector3 ubinAD=vbinAD.Unit();
                    TA=vbinAD.Theta();
                    PA=vbinAD.Phi();
                    TAd=(vbinAD.Theta())*deg;
                    PAd=(vbinAD.Phi())*deg;
```

```
   //energy losses
   R1=apAl*Ereal*Ereal+bpAl*Ereal+cpAl+0.8;
   E1=(-bpAl+sqrt(bpAl*bpAl-4*apAl*(cpAl-R1)))/(2*a

   R2=(apCD*E1*E1+bpCD*E1+cpCD)+(thickness/cos(TA))
   E2=(-bpCD+sqrt(bpCD*bpCD-4*apCD*(cpCD-R2)))/(2*a
   EAr=E2; //energy after reaction
   // Double_t EA=EAr/1000.;

/*
R1=aAlAl*Eapp*Eapp+bAlAl*Eapp+cAlAl+0.8;
E1=(-bAlAl+sqrt(bAlAl*bAlAl-4*aAlAl*(cAlAl-R1)))/(

R2=(aAlAu*E1*E1+bAlAu*E1+cAlAu)+(thickness/cos(TA)
E2=(-bAlAu+sqrt(bAlAu*bAlAu-4*aAlAu*(cAlAu-R2)))/(
EAr=E2; //energy after reaction
*/
   p1=sqrt(2*m1*u*EAr); //alpha momentum in GeV/c

   TVector3 vp1=p1*ubinAD; //alpha momentum (vector

   pp1.SetPx(vp1(0));
   pp1.SetPy(vp1(1));
   pp1.SetPz(vp1(2));
   pp1.SetE(u*m1+EAr);

   mA++;

   //filling histograms
he->Fill(EAr);

hptp->Fill(PA+deg, TA*deg);

h2_e_t_a->Fill(TA*deg, EAr);
```

```
                    h1->Fill();

                  }
              }
          }
      }

      for (j=0; j<32; j++){
          valuef = EARF[j];
          if(valuef>2 && valuef<120){
              for (jj=0; jj<32; jj++){
                  valueb = EARB[jj];
                  if(valuef>valueb-0.5 && valuef<valueb+0.5){
                      mAR++;
                      Eapp = (valuef+valueb)/2;
                      Ereal = aA*Eapp*Eapp*Eapp*Eapp*Eapp+bA*Eapp*Eapp
                      // EAr = (valuef+valueb)/2;
                      xhitA = j;
                      yhitA = jj;
                      posxA=(xhitA-16)*stepA+stepA/2+r3->Uniform(-step
                      //negative position closer to beam axis
                      posyA=(yhitA-16)*stepA+stepA/2+r3->Uniform(-step
                      //negative position bottom of the chamber

                      Double_t avdARr[3]={posxA,0.,posyA};
                      Double_t avdAR[3]={0.,0.,0.};

                      //we go back to the real position in the space
                      rotAR->LocalToMaster(avdARr,avdAR);

                      //this is the vector from det.A1 center to appar
                      TVector3 vdARa(avdAR[0],avdAR[1],avdAR[2]);

                      //apparent trajectory of 12C
```

```
   TVector3 vbinAR=vAR+vdARa;
   TVector3 ubinAR=vbinAR.Unit();
   TA=vbinAR.Theta();
   PA=vbinAR.Phi();
   TAd=(vbinAR.Theta())*deg;
   PAd=(vbinAR.Phi())*deg;

   //energy losses
   R1=apAl*Ereal*Ereal+bpAl*Ereal+cpAl+0.8;
   E1=(-bpAl+sqrt(bpAl*bpAl-4*apAl*(cpAl-R1)))/(2*a

   R2=(apCD*E1*E1+bpCD*E1+cpCD)+(thickness/cos(TA))
   E2=(-bpCD+sqrt(bpCD*bpCD-4*apCD*(cpCD-R2)))/(2*a
   EAr=E2; //energy after reaction
   // Double_t EA=EAr/1000.;

/*
R1=aAlAl*Eapp*Eapp+bAlAl*Eapp+cAlAl+0.8;
E1=(-bAlAl+sqrt(bAlAl*bAlAl-4*aAlAl*(cAlAl-R1)))/(

R2=(aAlAu*E1*E1+bAlAu*E1+cAlAu)+(thickness/cos(TA)
E2=(-bAlAu+sqrt(bAlAu*bAlAu-4*aAlAu*(cAlAu-R2)))/(
EAr=E2; //energy after reaction
*/

   p1=sqrt(2*m1*u*EAr); //alpha momentum in GeV/c

   TVector3 vp1=p1*ubinAR; //alpha momentum (vector

   pp1.SetPx(vp1(0));
   pp1.SetPy(vp1(1));
   pp1.SetPz(vp1(2));
   pp1.SetE(u*m1+EAr);
```

```
                mA++;

                //filling histograms
               he->Fill(EAr);

               hptp->Fill(PA+deg, TA*deg);

               h2_e_t_a->Fill(TA*deg, EAr);
               h1->Fill();

             }
          }
       }
   }

   for (j=0; j<32; j++){
       valuef = EALF[j];
       if(valuef>2 && valuef<120){
           for (jj=0; jj<32; jj++){
               valueb = EALB[jj];
               if(valuef>valueb-0.5 && valuef<valueb+0.5){
                   mAL++;
                   Eapp = (valuef+valueb)/2;
                   Ereal = aA*Eapp*Eapp*Eapp*Eapp*Eapp+bA*Eapp*Eapp
                   // EAr = (valuef+valueb)/2;
                   xhitA = j;
                   yhitA = jj;
                   posxA=(xhitA-16)*stepA+stepA/2+r3->Uniform(-step
                   //negative position closer to beam axis
                   posyA=(yhitA-16)*stepA+stepA/2+r3->Uniform(-step
                   //negative position bottom of the chamber

                   Double_t avdALr[3]={posxA,0.,posyA};
                   Double_t avdAL[3]={0.,0.,0.};
```

```
  //we go back to the real position in the space
  rotAL->LocalToMaster(avdALr,avdAL);

  //this is the vector from det.A1 center to appar
  TVector3 vdALa(avdAL[0],avdAL[1],avdAL[2]);

  //apparent trajectory of 12C
  TVector3 vbinAL=vAL+vdALa;
  TVector3 ubinAL=vbinAL.Unit();
  TA=vbinAL.Theta();
  PA=vbinAL.Phi();
  TAd=(vbinAL.Theta())*deg;
  PAd=(vbinAL.Phi())*deg;

  //energy losses
  R1=apAl*Ereal*Ereal+bpAl*Ereal+cpAl+0.8;
  E1=(-bpAl+sqrt(bpAl*bpAl-4*apAl*(cpAl-R1)))/(2*a

  R2=(apCD*E1*E1+bpCD*E1+cpCD)+(thickness/cos(TA))
  E2=(-bpCD+sqrt(bpCD*bpCD-4*apCD*(cpCD-R2)))/(2*a
  EAr=E2; //energy after reaction
  // Double_t EA=EAr/1000.;

/*
R1=aAlAl*Eapp*Eapp+bAlAl*Eapp+cAlAl+0.8;
E1=(-bAlAl+sqrt(bAlAl*bAlAl-4*aAlAl*(cAlAl-R1)))/(

R2=(aAlAu*E1*E1+bAlAu*E1+cAlAu)+(thickness/cos(TA)
E2=(-bAlAu+sqrt(bAlAu*bAlAu-4*aAlAu*(cAlAu-R2)))/(
EAr=E2; //energy after reaction
*/

  p1=sqrt(2*m1*u*EAr); //alpha momentum in GeV/c
```

```
                        TVector3 vp1=p1*ubinAL; //alpha momentum (vector

                        pp1.SetPx(vp1(0));
                        pp1.SetPy(vp1(1));
                        pp1.SetPz(vp1(2));
                        pp1.SetE(u*m1+EAr);

                        mA++;

                        //filling histograms
                      he->Fill(EA);

                      hptp->Fill(PA+deg, TA*deg);

                      h2_e_t_a->Fill(TA*deg, EAr);
                      h1->Fill();

                    }
                }
            }
        }


/*
        for (j=4; j<16; j++){
            valuef = EBUF[j];
            if(valuef>2 && valuef<120){
                for (jj=5; jj<16; jj++){
                    valueb = EBUB[jj];
                    if(valuef>valueb-0.5 && valuef<valueb+0.5 && mB==0){
                        mBU++;
                        Eapp = (valuef+valueb)/2;
                        //EBr = (valuef+valueb)/2;
```

```
xhitB = j;
yhitB = jj;
posxB=(xhitB-8)*step+step/2+r3->Uniform(-step/2,
//negative position closer to beam axis
posyB=(yhitB-8)*step+step/2+r3->Uniform(-step/2,
//negative position bottom of the chamber

Double_t avdBUr[3]={posxB,0.,posyB};
Double_t avdBU[3]={0.,0.,0.};

//we go back to the real position in the space
rotBU->LocalToMaster(avdBUr,avdBU);

//this is the vector from det.A1 center to appar
TVector3 vdBUa(avdBU[0],avdBU[1],avdBU[2]);

//apparent trajectory of 12C
TVector3 vbinBU=vBU+vdBUa;
TVector3 ubinBU=vbinBU.Unit();
TB=vbinBU.Theta();
PB=vbinBU.Phi();
TBd=(vbinBU.Theta())*deg;
PBd=(vbinBU.Phi())*deg;

Double_t EDE=0.;
Int_t jjj=-1;

if(j!=0 && EDEBU[15-j-1]>20){
  EDE=EDEBU[15-j-1];
  jjj=j-1;
}

if(j!=15 && EDEBU[15-j+1]>20){
  EDE=EDEBU[15-j+1];
```

```
    jjj=j+1;
  }

  if(EDEBU[15-j]>20){
    EDE=EDEBU[15-j];
    jjj=j;
  }

  //energy losses
  R1=aMgAl*Eapp*Eapp+bMgAl*Eapp+cMgAl+1.6;
  E1=((-bMgAl+sqrt(bMgAl*bMgAl-4.*aMgAl*(cMgAl-R1)

  R2=aMgAl*E1*E1+bMgAl*E1+cMgAl+0.8;
  E2=(-bMgAl+sqrt(bMgAl*bMgAl-4.*aMgAl*(cMgAl-R2))

  R3=(aMgCD*E2*E2+bMgCD*E2+cMgCD)+(thickness/cos(T
  E3=(-bMgCD+sqrt(bMgCD*bMgCD-4.*aMgCD*(cMgCD-R3))
  EBr=E3; //energy after reaction
  // Double_t EB=EBr/1000.;

R1=aAlAl*Eapp*Eapp+bAlAl*Eapp+cAlAl+1.6;
E1=((-bAlAl+sqrt(bAlAl*bAlAl-4.*aAlAl*(cAlAl-R1)))

R2=aAlAl*E1*E1+bAlAl*E1+cAlAl+0.8;
E2=(-bAlAl+sqrt(bAlAl*bAlAl-4.*aAlAl*(cAlAl-R2)))/

R3=(aAlAu*E2*E2+bAlAu*E2+cAlAu)+(thickness/cos(TB)
E3=(-bAlAu+sqrt(bAlAu*bAlAu-4.*aAlAu*(cAlAu-R3)))/
EBr=E3; //energy after reaction

  EB=EDE+Eapp;
  for(int p=0; p<16; p++){
    CDEBU[p]=0.;
  }
```

```
                    CDEB=0.;

                    //double corr=20./th_DEBU[j][jj];
                    CDEBU[j]=EDE;
                    CDEB=EDE*(20./th_DEBU[jj][15-j]);

                      p2=sqrt(2*m2*u*EBr); //alpha momentum in GeV/c

                      TVector3 vp2=p2*ubinBU; //alpha momentum (vector

                      pp2.SetPx(vp2(0));
                      pp2.SetPy(vp2(1));
                      pp2.SetPz(vp2(2));
                      pp2.SetE(u*m2+EBr);

                      mB++;

                      //filling histograms
                      he->Fill(EBr);

                      hptp->Fill(PB*deg, TB*deg);

                      h2_e_t_a->Fill(TB*deg, EBr);
                      h1->Fill();

                  }
              }
          }
      }
*/
    for (j=0; j<16; j++){
        valuef = EBDF[j];
        if(valuef>2 && valuef<120){
            for (jj=0; jj<16; jj++){
```

```
valueb = EBDB[jj];
if(valuef>valueb-0.5 && valuef<valueb+0.5 && mB==0){
  mBD++;
  Eapp = (valuef+valueb)/2;
  // EBr = (valuef+valueb)/2;
  xhitB = j;
  yhitB = jj;
  posxB=(xhitB-8)*step+step/2+r3->Uniform(-step/2,
  //negative position closer to beam axis
  posyB=(yhitB-8)*step+step/2+r3->Uniform(-step/2,
  //negative position bottom of the chamber

  Double_t avdBDr[3]={posxB,0.,posyB};
  Double_t avdBD[3]={0.,0.,0.};

  //we go back to the real position in the space
  rotBD->LocalToMaster(avdBDr,avdBD);

  //this is the vector from det.A1 center to appar
  TVector3 vdBDa(avdBD[0],avdBD[1],avdBD[2]);

  //apparent trajectory of 12C
  TVector3 vbinBD=vBD+vdBDa;
  TVector3 ubinBD=vbinBD.Unit();
  TB=vbinBD.Theta();
  PB=vbinBD.Phi();
  TBd=(vbinBD.Theta())*deg;
  PBd=(vbinBD.Phi())*deg;

  Double_t EDE=0.;
  Int_t jjj=-1;

  if(j!=0 && EDEBD[15-j-1]>20){
    EDE=EDEBD[15-j-1];
    jjj=j-1;
  }

  if(j!=15 && EDEBD[15-j+1]>20){
    EDE=EDEBD[15-j+1];
    jjj=j+1;
  }

  if(EDEBD[15-j]>20){
    EDE=EDEBD[15-j];
    jjj=j;
  }


  //energy losses
  R1=aMgAl*Eapp*Eapp+bMgAl*Eapp+cMgAl+1.6;
  E1=((-bMgAl+sqrt(bMgAl*bMgAl-4*aMgAl*(cMgAl-R1))

  R2=aMgAl*E1*E1+bMgAl*E1+cMgAl+0.8;
  E2=(-bMgAl+sqrt(bMgAl*bMgAl-4*aMgAl*(cMgAl-R2)))

  R3=(aMgCD*E2*E2+bMgCD*E2+cMgCD)+(thickness/cos(T
  E3=(-bMgCD+sqrt(bMgCD*bMgCD-4*aMgCD*(cMgCD-R3)))
  EBr=E3; //energy after reaction
  // Double_t EB=EBr/1000.;
/*
R1=aAlAl*Eapp*Eapp+bAlAl*Eapp+cAlAl+1.6;
E1=((-bAlAl+sqrt(bAlAl*bAlAl-4.*aAlAl*(cAlAl-R1)))

R2=aAlAl*E1*E1+bAlAl*E1+cAlAl+0.8;
E2=(-bAlAl+sqrt(bAlAl*bAlAl-4.*aAlAl*(cAlAl-R2)))/

R3=(aAlAu*E2*E2+bAlAu*E2+cAlAu)+(thickness/cos(TB)
E3=(-bAlAu+sqrt(bAlAu*bAlAu-4.*aAlAu*(cAlAu-R3)))/
```

```
                    EBr=E3; //energy after reaction
                    */
                     EB=EDE+Eapp;
                     for(int p=0; p<16; p++){
                       CDEBD[p]=0.;
                     }
                     CDEB=0.;

                    CDEBD[j]=EDE;
                    CDEB=EDE*(20.0/th_DEBD[jj][15-j]);

                      p2=sqrt(2*m2*u*EBr); //alpha momentum in GeV/c

                      TVector3 vp2=p2*ubinBD; //alpha momentum (vector

                      pp2.SetPx(vp2(0));
                      pp2.SetPy(vp2(1));
                      pp2.SetPz(vp2(2));
                      pp2.SetE(u*m2+EBr);

                     mB++;

                      //filling histograms
                    he->Fill(EBr);

                    hptp->Fill(PB*deg, TB*deg);

                    h2_e_t_a->Fill(TB*deg, EBr);
                    h1->Fill();

                  }
              }
          }
      }
```

```
/*
          for (j=0; j<16; j++){
                valuef = EBRF[j];
                if(valuef>2 && valuef<120){
                      for (jj=0; jj<16; jj++){
                            valueb = EBRB[jj];
                            if(valuef>valueb-0.5 && valuef<valueb+0.5 && mB==0){
                                  mBR++;
                                  Eapp = (valuef+valueb)/2;
                                  //EBr = (valuef+valueb)/2;
                                  xhitB = j;
                                  yhitB = jj;
                                  posxB=(xhitB-8)*step+step/2+r3->Uniform(-step/2,
                                  //negative position closer to beam axis
                                  posyB=(yhitB-8)*step+step/2+r3->Uniform(-step/2,
                                  //negative position bottom of the chamber

                                  Double_t avdBRr[3]={posxB,0.,posyB};
                                  Double_t avdBR[3]={0.,0.,0.};

                                  //we go back to the real position in the space
                                  rotBR->LocalToMaster(avdBRr,avdBR);

                                  //this is the vector from det.A1 center to appar
                                  TVector3 vdBRa(avdBR[0],avdBR[1],avdBR[2]);

                                  //apparent trajectory of 12C
                                  TVector3 vbinBR=vBR+vdBRa;
                                  TVector3 ubinBR=vbinBR.Unit();
                                  TB=vbinBR.Theta();
                                  PB=vbinBR.Phi();
                                  TBd=(vbinBR.Theta())*deg;
                                  PBd=(vbinBR.Phi())*deg;
```

```
    Double_t EDE=0;
    Int_t  jjj=-1;

    if(j!=0 && EDEBR[15-j-1]>20){
      EDE=EDEBR[15-j-1];
       jjj=j-1;
    }

    if(j!=15 && EDEBR[15-j+1]>20){
      EDE=EDEBR[15-j+1];
       jjj=j+1;
    }

    if(EDEBR[15-j]>20){
      EDE=EDEBR[15-j];
       jjj=j;
    }

    //energy losses
    R1=aMgAl*Eapp*Eapp+bMgAl*Eapp+cMgAl+1.6;
    E1=((-bMgAl+sqrt(bMgAl*bMgAl-4*aMgAl*(cMgAl-R1))

    R2=aMgAl*E1*E1+bMgAl*E1+cMgAl+0.8;
    E2=(-bMgAl+sqrt(bMgAl*bMgAl-4*aMgAl*(cMgAl-R2)))

    R3=(aMgCD*E2*E2+bMgCD*E2+cMgCD)+(thickness/cos(T
    E3=(-bMgCD+sqrt(bMgCD*bMgCD-4*aMgCD*(cMgCD-R3)))
    EBr=E3; //energy after reaction
    // Double_t EB=EBr/1000.;

  R1=aAlAl*Eapp*Eapp+bAlAl*Eapp+cAlAl+1.6;
  E1=((-bAlAl+sqrt(bAlAl*bAlAl-4*aAlAl*(cAlAl-R1)))/

  R2=aAlAl*E1*E1+bAlAl*E1+cAlAl+0.8;
```

```
E2=(-bAlAl+sqrt(bAlAl*bAlAl-4*aAlAl*(cAlAl-R2)))/(

R3=(aAlAu*E2*E2+bAlAu*E2+cAlAu)+(thickness/cos(TB)
E3=(-bAlAu+sqrt(bAlAu*bAlAu-4*aAlAu*(cAlAu-R3)))/(
EBr=E3; //energy after reaction


  EB=EDE+Eapp;
  for(int p=0; p<16; p++){
    CDEBR[p]=0;
  }
CDEB=0.;


  CDEBR[j]=EDE;
  CDEB=EDE*(20.0/th_DEBR[jj][15-j]);


  p2=sqrt(2*m2*u*EBr); //alpha momentum in GeV/c


  TVector3 vp2=p2*ubinBR; //alpha momentum (vector


  pp2.SetPx(vp2(0));
  pp2.SetPy(vp2(1));
  pp2.SetPz(vp2(2));
  pp2.SetE(u*m2+EBr);


  mB++;


  //filling histograms
he->Fill(EBr);


hptp->Fill(PB*deg, TB*deg);


h2_e_t_a->Fill(TB*deg, EBr);
h1->Fill();
```

```
            }
        }
    }
}

for (j=0; j<16; j++){
    valuef = EBLF[j];
    if(valuef>2 && valuef<120){
        for (jj=0; jj<16; jj++){
            valueb = EBLB[jj];
            if(valuef>valueb-0.5 && valuef<valueb+0.5 && mB==0){
                mBL++;
                Eapp = (valuef+valueb)/2;
                //EBr = (valuef+valueb)/2;
                xhitB = j;
                yhitB = jj;
                posxB=(xhitB-8)*step+step/2+r3->Uniform(-step/2,
                //negative position closer to beam axis
                posyB=(yhitB-8)*step+step/2+r3->Uniform(-step/2,
                //negative position bottom of the chamber

                Double_t avdBLr[3]={posxB,0.,posyB};
                Double_t avdBL[3]={0.,0.,0.};

                //we go back to the real position in the space
                rotBL->LocalToMaster(avdBLr,avdBL);

                //this is the vector from det.A1 center to appar
                TVector3 vdBLa(avdBL[0],avdBL[1],avdBL[2]);

                //apparent trajectory of 12C
                TVector3 vbinBL=vBL+vdBLa;
                TVector3 ubinBL=vbinBL.Unit();
                TB=vbinBL.Theta();
```

```
PB=vbinBL.Phi();
TBd=(vbinBL.Theta())*deg;
PBd=(vbinBL.Phi())*deg;

Double_t EDE=0;
Int_t jjj=-1;

if(j!=0 && EDEBL[15-j-1]>20){
  EDE=EDEBL[15-j-1];
   jjj=j-1;
}

if(j!=15 && EDEBL[15-j+1]>20){
  EDE=EDEBL[15-j+1];
   jjj=j+1;
}

if(EDEBL[15-j]>20){
  EDE=EDEBL[15-j];
   jjj=j;
}

//energy losses
R1=aMgAl*Eapp*Eapp+bMgAl*Eapp+cMgAl+1.6;
E1=((-bMgAl+sqrt(bMgAl*bMgAl-4*aMgAl*(cMgAl-R1))

R2=aMgAl*E1*E1+bMgAl*E1+cMgAl+0.8;
E2=(-bMgAl+sqrt(bMgAl*bMgAl-4*aMgAl*(cMgAl-R2)))

R3=(aMgCD*E2*E2+bMgCD*E2+cMgCD)+(thickness/cos(T
E3=(-bMgCD+sqrt(bMgCD*bMgCD-4*aMgCD*(cMgCD-R3)))
EBr=E3; //energy after reaction
// Double_t EB=EBr/1000.;
```

```
R1=aAlAl*Eapp*Eapp+bAlAl*Eapp+cAlAl+1.6;
E1=((-bAlAl+sqrt(bAlAl*bAlAl-4*aAlAl*(cAlAl-R1)))/

R2=aAlAl*E1*E1+bAlAl*E1+cAlAl+0.8;
E2=(-bAlAl+sqrt(bAlAl*bAlAl-4*aAlAl*(cAlAl-R2)))/(

R3=(aAlAu*E2*E2+bAlAu*E2+cAlAu)+(thickness/cos(TB)
E3=(-bAlAu+sqrt(bAlAu*bAlAu-4*aAlAu*(cAlAu-R3)))/(
EBr=E3; //energy after reaction

  EB=EDE+Eapp;
  for(int p=0; p<16; p++){
    CDEBL[p]=0;
  }
CDEB=0.;

  CDEBL[j]=EDE;
  CDEB=EDE*(20.0/th_DEBL[jj][15-j]);

  p2=sqrt(2*m2*u*EBr); //alpha momentum in GeV/c

  TVector3 vp2=p2*ubinBL; //alpha momentum (vector

  pp2.SetPx(vp2(0));
  pp2.SetPy(vp2(1));
  pp2.SetPz(vp2(2));
  pp2.SetE(u*m2+EBr);

  mB++;

  //filling histograms
he->Fill(EBr);

hptp->Fill(PB*deg, TB*deg);
```

```
                    h2_e_t_a->Fill(TB*deg, EBr);
                    h1->Fill();

                  }
               }
            }
         }
*/
         nevent++;


      if(mA==1 && mB==1){
         //calculation of the relative energies (including resolution)
         TLorentzVector p12=pp1+pp2;

         pp3=beam+target-pp1-pp2;
         ppp3=pp3.Vect();
         ES=(ppp3.Mag2())/(2*m3*u);
         TSd=(ppp3.Theta())*deg;
         PSd=(ppp3.Phi())*deg;

           Q=EAr+EBr+ES-ep;

           romx=(ppp3.Mag2())/(2*u);
           romy=ep-EAr-EBr;

           ppp1=pp1.Vect();
           ppp2=pp2.Vect();

           //calculation of theta_cm (including resolution)
           TVector3 v0b = pow(u*mp,-1)*(beam.Vect());
           TVector3 v1b = pow(u*m1,-1)*ppp1; //p1 speed
           TVector3 v2b = pow(u*m2,-1)*ppp2; //26Mg speed
```

```
    TVector3 v3b = pow(u*m3,-1)*ppp3; //p2 speed


    // calculated from p1 (p2 spectator)
    TVector3 k30b=-(m3/mx)*v3b-v0b;
    TVector3 k12b= v1b-v2b;
    tcm= (k30b.Angle(k12b))*deg;

    // if p2 is spectator, then:
    Double_t scp2;
    if(cos(ppp3.Phi())>0)
      scp2=1;
    else if (cos(ppp3.Phi())<0)
      scp2=-1.;
    else
      scp2=0.;
    ps=(ppp3.Mag())*scp2; //MeV/c


  mA1B1++;

}



   if(nevent%1000000==0)
       cout << "Fatti␣1M␣eventi" << endl;
if(mA==2 && mB==1){
  mA2B1++;
}
if(mA==1 && mB==2){
  mA1B2++;
}
if(mA==2 && mB==2){
  mA2B2++;
```

```
    }
    if(mA==0 && mB==0){
      mA0B0++;
    }
    if(mA==0 && mB==1){
      mA0B1++;
    }
    if(mA==1 && mB==0){
      mA0B1++;
    }

  }

cout << "Eventi␣mA=1␣e␣mB=1:␣" << mA1B1 << endl;
cout << "Eventi␣mA=2␣e␣mB=1:␣" << mA2B1 << endl;
cout << "Eventi␣mA=1␣e␣mB=2:␣" << mA1B2 << endl;
cout << "Eventi␣mA=2␣e␣mB=2:␣" << mA2B2 << endl;
cout << "Eventi␣mA=0␣e␣mB=1:␣" << mA0B1 << endl;
cout << "Eventi␣mA=1␣e␣mB=0:␣" << mA1B0 << endl;
cout << "Eventi␣mA=0␣e␣mB=0:␣" << mA0B0 << endl;
cout << "Somma:␣" << mA1B1+mA1B0+mA1B2+mA0B1+mA2B1+mA2B2+mA0B0 << end

cout << "Eventi␣in␣AU:␣" << mAU << endl;
cout << "Eventi␣in␣AD:␣" << mAD << endl;
cout << "Eventi␣in␣AL:␣" << mAL << endl;
cout << "Eventi␣in␣AR:␣" << mAR << endl;
cout << "Eventi␣in␣BU:␣" << mBU << endl;
cout << "Eventi␣in␣BD:␣" << mBD << endl;
cout << "Eventi␣in␣BR:␣" << mBR << endl;
cout << "Eventi␣in␣BL:␣" << mBL << endl;


  cout << "number␣of␣events␣=␣" << nevent <<endl;
```

```
    h1->Write();

    fout->Write();

    fout->Close();
}
```

Macro for the simulation of the experimental runs

```
#include <TGeoManager.h>
#include <TGeometry.h>
#include <TNode.h>
#include <TGeoVolume.h>
#include <TGeoMatrix.h>
#include <TGeoMedium.h>
#include <TGeoMaterial.h>
#include <TGeoNode.h>
#include <TMaterial.h>
#include <TMixture.h>
#include <TShape.h>
#include <TString.h>
#include <Riostream.h>
#include <TFile.h>
#include <TCanvas.h>
#include <TH2F.h>
#include <TMath.h>
#include <TLorentzVector.h>
#include <TSystem.h>
#include <stdlib.h>
#include <TROOT.h>
#include <TGenPhaseSpace.h>
#include <TStyle.h>
#include "TGeoPatternFinder.h"
#include "TSystem.h"
```

```
#include <TGeoNavigator.h>
#include <iostream>
#include <cstdlib>
#include <fstream>
#include <cmath>
#include "TApplication.h"
#include "TF1.h"
#include "TGraph.h"
#include "TGraphErrors.h"
#include "TH1.h"
#include "TLegend.h"
#include "TLegendEntry.h"
#include "TColor.h"
#include <TRandom3.h>
#include <time.h>
#include "TTree.h"
#include <random>

using namespace std;

double func(double x){
double am3=1.007825;
double xm3=am3*931.;
double gpsn=sqrt(2.25*59.8)*pow(sqrt(2.25)+sqrt(59.8),3);
double gpsd=pow(2.25+2.*x*x/(2.*xm3),2)*pow(59.8+2.*x*x/(2.*xm3),2);
double gps=gpsn/gpsd;
double distp3=1.98286*gps;
return(distp3);
}

double reso(double x){
double xr[6]={-0.5,0.1, 0.2, 0.5, 1.0, 1.5};
double wd=0.02;
double res0=0.25*wd*wd/(pow(x-xr[0],2)+0.25*wd*wd);
double res1=0.25*wd*wd/(pow(x-xr[1],2)+0.25*wd*wd);
double res2=0.25*wd*wd/(pow(x-xr[2],2)+0.25*wd*wd);
double res3=0.25*wd*wd/(pow(x-xr[3],2)+0.25*wd*wd);
double res4=0.25*wd*wd/(pow(x-xr[4],2)+0.25*wd*wd);
double res5=0.25*wd*wd/(pow(x-xr[5],2)+0.25*wd*wd);
double sumres=1.0*res0+1.0*res1+1.0*res2+1.0*res3+1.0*res4+1.0*res5;
return(sumres);
}

void geometry_and_physics_MgD() {

TApplication theApp("App", NULL, NULL);

gStyle->SetPalette(1);

gSystem->Load("libPhysics.so");

gSystem->Load("libGeom.so");

TGeoManager *sc = new TGeoManager("scattcham","Scattering␣Chamber");

TGeoMaterial *matVacuum = new TGeoMaterial("Vacuum",0,0,0);
TGeoMaterial *matSi = new TGeoMaterial("Si",28.086,14,2.321);

TGeoMedium *Vacuum = new TGeoMedium("Vacuum",1,matVacuum);
TGeoMedium *Si = new TGeoMedium("Root␣Material",2,matSi);

TGeoVolume *top = sc->MakeSphere("TOP",Vacuum,0.,100.,0.,180.,0.,360.);
sc->SetTopVolume(top);

TGeoVolume *psd = sc->MakeBox("PSD",Si,2.5,0.05,2.5);

// from deg to rad
    double dtr=TMath::Pi()/180.;
```

```
// close geometry

// this is detector A1
    double theta1=56*dtr;
    double   phi1=45*dtr;
    double  dist1=6.00;
    double     px1=dist1*sin(theta1)*cos(phi1);
    double     py1=dist1*sin(theta1)*sin(phi1);
    double     pz1=dist1*cos(theta1);
    TVector3 vA1(px1,py1,pz1); //position of det. A1 wrt chamber center
    TGeoRotation *rot1 = new TGeoRotation("rot1",-45,34,0);
TGeoCombiTrans *pos1 = new TGeoCombiTrans(px1,py1,pz1,rot1);

// this is detector A2
    double theta2=34*dtr;
    double   phi2=(45+90)*dtr;
    double  dist2=6.02;
    double     px2=dist2*sin(theta2)*cos(phi2);
    double     py2=dist2*sin(theta2)*sin(phi2);
    double     pz2=dist2*cos(theta2);
    TVector3 vA2(px2,py2,pz2); //position of det. A2 wrt chamber center
    TGeoRotation *rot2 = new TGeoRotation("rot2",45,56,0);
TGeoCombiTrans *pos2 = new TGeoCombiTrans(px2,py2,pz2,rot2);

//this is detector A3
    double theta3=56*dtr;
    double   phi3=(45+180)*dtr;
    double  dist3=6;
    double     px3=dist3*sin(theta3)*cos(phi3);
    double     py3=dist3*sin(theta3)*sin(phi3);
    double     pz3=dist3*cos(theta3);
    TVector3 vA3(px3,py3,pz3); //position of det. A3 wrt chamber center
    TGeoRotation *rot3 = new TGeoRotation("rot3",-45,-34,0);
```

```
TGeoCombiTrans *pos3 = new TGeoCombiTrans(px3,py3,pz3,rot3);

// this is detector A4
    double theta4=34*dtr;
    double   phi4=(45+270)*dtr;
    double  dist4=6.02;
    double     px4=dist4*sin(theta4)*cos(phi4);
    double     py4=dist4*sin(theta4)*sin(phi4);
    double     pz4=dist4*cos(theta4);
    TVector3 vA4(px4,py4,pz4); //position of det. A4 wrt chamber center
    TGeoRotation *rot4 = new TGeoRotation("rot4",45,-56,0);
TGeoCombiTrans *pos4 = new TGeoCombiTrans(px4,py4,pz4,rot4);

// far geometry

// this is detector B1
    double theta5=5*dtr;
    double   phi5=45*dtr;
    double  dist5=69.14;
    double     px5=dist5*sin(theta5)*cos(phi5);
    double     py5=dist5*sin(theta5)*sin(phi5);
    double     pz5=dist5*cos(theta5);
    TVector3 vB1(px5,py5,pz5); //position of det. B1 wrt chamber center
    TGeoRotation *rot5 = new TGeoRotation("rot5",-45,90,0);
TGeoCombiTrans *pos5 = new TGeoCombiTrans(px5,py5,pz5,rot5);

// this is detector B2
    double theta6=4*dtr;
    double   phi6=(45+90)*dtr;
    double  dist6=71.06;
    double     px6=dist6*sin(theta6)*cos(phi6);
    double     py6=dist6*sin(theta6)*sin(phi6);
    double     pz6=dist6*cos(theta6);
    TVector3 vB2(px6,py6,pz6); //position of det. B2 wrt chamber center
    TGeoRotation *rot6 = new TGeoRotation("rot6",45,90,0);
TGeoCombiTrans *pos6 = new TGeoCombiTrans(px6,py6,pz6,rot6);

// this is detector B3
    double theta7=5*dtr;
    double   phi7=(45+180)*dtr;
    double  dist7=69.14;
    double     px7=dist7*sin(theta7)*cos(phi7);
    double     py7=dist7*sin(theta7)*sin(phi7);
    double     pz7=dist7*cos(theta7);
    TVector3 vB3(px7,py7,pz7); //position of det. B3 wrt chamber center
    TGeoRotation *rot7 = new TGeoRotation("rot7",-45,-90,0);
TGeoCombiTrans *pos7 = new TGeoCombiTrans(px7,py7,pz7,rot7);

// this is detector B4
    double theta8=4*dtr;
    double   phi8=(45+270)*dtr;
    double  dist8=71.06;
    double     px8=dist8*sin(theta8)*cos(phi8);
    double     py8=dist8*sin(theta8)*sin(phi8);
    double     pz8=dist8*cos(theta8);
    TVector3 vB4(px8,py8,pz8); //position of det. B4 wrt chamber center
    TGeoRotation *rot8 = new TGeoRotation("rot8",45,-90,0);
TGeoCombiTrans *pos8 = new TGeoCombiTrans(px8,py8,pz8,rot8);

// detector placing in space

top->AddNode(psd,1,pos1);
top->AddNode(psd,2,pos2);
top->AddNode(psd,3,pos3);
top->AddNode(psd,3,pos4);

top->AddNode(psd,5,pos5);
top->AddNode(psd,6,pos6);
```

```
top->AddNode(psd,7,pos7);
top->AddNode(psd,8,pos8);

gGeoManager->CloseGeometry();
top->SetLineColor(kRed);
gGeoManager->SetTopVisible();


    // get the nodeid of each detector

    gGeoManager->SetCurrentPoint(px1,py1,pz1);
    gGeoManager->FindNode();
    TGeoNode *nodeA1 = gGeoManager->GetCurrentNode();
    Int_t idA1 = gGeoManager->GetCurrentNodeId();
    cout << "A1␣" << idA1 << endl;

    gGeoManager->SetCurrentPoint(px2,py2,pz2);
    gGeoManager->FindNode();
    TGeoNode *nodeA2 = gGeoManager->GetCurrentNode();
    Int_t idA2 = gGeoManager->GetCurrentNodeId();
    cout << "A2␣" << idA2 << endl;

    gGeoManager->SetCurrentPoint(px3,py3,pz3);
    gGeoManager->FindNode();
    TGeoNode *nodeA3 = gGeoManager->GetCurrentNode();
    Int_t idA3 = gGeoManager->GetCurrentNodeId();
    cout << "A3␣" << idA3 << endl;

    gGeoManager->SetCurrentPoint(px4,py4,pz4);
    gGeoManager->FindNode();
    TGeoNode *nodeA4 = gGeoManager->GetCurrentNode();
    Int_t idA4 = gGeoManager->GetCurrentNodeId();
    cout << "A4␣" << idA4 << endl;
    gGeoManager->SetCurrentPoint(px5,py5,pz5);
    gGeoManager->FindNode();
    TGeoNode *nodeB1 = gGeoManager->GetCurrentNode();
    Int_t idB1 = gGeoManager->GetCurrentNodeId();
    cout << "B1␣" << idB1 << endl;

    gGeoManager->SetCurrentPoint(px6,py6,pz6);
    gGeoManager->FindNode();
    TGeoNode *nodeB2 = gGeoManager->GetCurrentNode();
    Int_t idB2 = gGeoManager->GetCurrentNodeId();
    cout << "B2␣" << idB2 << endl;

    gGeoManager->SetCurrentPoint(px7,py7,pz7);
    gGeoManager->FindNode();
    TGeoNode *nodeB3 = gGeoManager->GetCurrentNode();
    Int_t idB3 = gGeoManager->GetCurrentNodeId();
    cout << "B3␣" << idB3 << endl;

    gGeoManager->SetCurrentPoint(px8,py8,pz8);
    gGeoManager->FindNode();
    TGeoNode *nodeB4 = gGeoManager->GetCurrentNode();
    Int_t idB4 = gGeoManager->GetCurrentNodeId();
    cout << "B4␣" << idB4 << endl;




TCanvas *c0 = new TCanvas("c0","c0",0,0,800,800);
top->Draw();
c0->SaveAs("geometry_all_A32.pdf");

    long sec;
    time( &sec );
    TRandom3* ran = new TRandom3((unsigned)sec);
    Double_t         u=931.49410242/1000.;
// unita' di massa (rif. 12C) in GeV/c

// 2H(26Al,p26Mg)p gs to gs

    Double_t         mp=25.97602;
// massa proiettile

    Double_t         mt=2.01355;
// massa target

  Double_t         m1=1.00728;
// massa prima part. uscente p1

  Double_t         m2=26.97776;
// massa seconda part. uscente 27Mg

    Double_t         m3=1.00782503207;
// massa terza part. uscente p2

    Double_t         mx=1.00866491574;
// massa part. trasferita n


// Q della reazione 26Mg+p-->26Mg+p gs to gs
    Double_t         q2=u*1000.*(mp+mt-m1-m2);

    // Double_t         q3=u*1000.*(mp+mt-m1-m2-m3);
    // cout << "q3: " << q3 << endl;
//(Momentum, Energy units are Gev/C, GeV)
    Double_t masses[2] = { u*m1, u*m2} ;
  Double_t deg=180./TMath::Pi();
  Double_t radius=100.;
  Int_t geve=0;
  Int_t geve1=0;
  Int_t geve2=0;

  Double_t step=0.3125; //strip width in cm
  Double_t stepA=0.3125/2; //strip width in cm for 32 strip detectors


  // theta and phi in degrees
  Double_t Tp1;
  Double_t Pp1;
  Double_t Tmg;
  Double_t Pmg;

  // detected energy including det. res. 0.5%
  Double_t Ep1;
  Double_t Emg;

  //statistical weight
  Double_t WW;

Int_t m=0;

  // this tells us which particle is undetected, so energies and angl
  Int_t undetected;

  //relative energies including resolution effects
  Double_t e12b; //p1 - 26Mg

  //cm energies
  Double_t ecm1b; // calculated from p1 (p2 spectator?)
  Double_t ecm2b; // calculated from p2 (p1 spectator?)
```

```
//theta cm
Double_t tcm1b; // calculated from p1 (p2 spectator?)
Double_t tcm2b; // calculated from p2 (p1 spectator?)

//ps
Double_t ps1b; // p2 spectator?
Double_t ps2b; // p1 spectator?

// parameters with infinite resolution
Double_t e12;
Double_t e13;
Double_t e23;
Double_t ecm1;
Double_t ecm2;
Double_t ps1;
Double_t ps2;
Double_t Q=-1;
Double_t Q1=-1;
Double_t romx=-20;
Double_t romy=-20;
Double_t epfake;
Double_t Ep1exp=-1;
Double_t Ep2exp=-1;
Double_t Emgexp=-1;
Double_t Tp1exp=-360;
Double_t Tp2exp=-360;
Double_t Tmgexp=-360;
Double_t Pp1exp=-360;
Double_t Pp2exp=-360;
Double_t Pmgexp=-360;
```

```
TFile *fout = new TFile("sim_3d_prova_27Mg_pchannel_A32.root", "REC

TTree *sim = new TTree("sim", "sim");

sim->Branch("undetected", &undetected, "undetected/I");
sim->Branch("WW", &WW, "WW/D");

sim->Branch("Ep1", &Ep1, "Ep1/D"); //prot.1 energy in MeV
sim->Branch("Tp1", &Tp1, "Tp1/D"); //prot.1 theta  in deg
sim->Branch("Pp1", &Pp1, "Pp1/D"); //prot.1 phi    in deg

sim->Branch("Emg", &Emg, "Emg/D"); //26Mg   energy in MeV
sim->Branch("Tmg", &Tmg, "Tmg/D"); //26Mg   theta  in deg
sim->Branch("Pmg", &Pmg, "Pmg/D"); //26Mg   phi    in deg

// energies including resolution
sim->Branch("e12b", &e12b, "e12b/D");
sim->Branch("ecm1b", &ecm1b, "ecm1b/D");
sim->Branch("ecm2b", &ecm2b, "ecm2b/D");

// theta cm including resolution
sim->Branch("tcm1b", &tcm1b, "tcm1b/D");
sim->Branch("tcm2b", &tcm2b, "tcm2b/D");

// ps including resolution
sim->Branch("ps1b", &ps1b, "ps1b/D");
sim->Branch("ps2b", &ps2b, "ps2b/D");

sim->Branch("Q", &Q, "Q/D");
sim->Branch("romx", &romx, "romx/D");
sim->Branch("romy", &romy, "romy/D");
sim->Branch("epfake", &epfake, "epfake/D");
sim->Branch("Q1", &Q1, "Q1/D");
sim->Branch("Ep1exp", &Ep1exp, "Ep1exp/D"); //prot.1 energy in MeV
```

```
    sim->Branch("Tp1exp", &Tp1exp, "Tp1exp/D"); //prot.1 theta
in deg
    sim->Branch("Pp1exp", &Pp1exp, "Pp1exp/D"); //prot.1 phi
in deg

    sim->Branch("Emgexp", &Emgexp, "Emgexp/D"); //26Mg   energy in MeV
    sim->Branch("Tmgexp", &Tmgexp, "Tmgexp/D"); //26Mg   theta
in deg
    sim->Branch("Pmgexp", &Pmgexp, "Pmgexp/D"); //26Mg   phi
in deg



    TFile *fhisto=new TFile("histo_sim_3d_prova_27Mg_pchannel_A32.root"
    // energies of the detected particles
    TH2F *h2_e_p1_mg = new TH2F("h2_e_p1_mg","h2_e_p1_mg", 300,0,25,
300,60,95);
    TH2F *h2_e_p2_mg = new TH2F("h2_e_p2_mg","h2_e_p2_mg", 300,0,25,
300,60,95);
    TH2F *h2_e_p1_p2 = new TH2F("h2_e_p1_p2","h2_e_p1_p2", 300,0,25,
300,0  ,25);
    // theta of the detected particles
    TH2F *h2_t_p1_mg = new TH2F("h2_t_p1_mg","h2_t_p1_mg", 360,0,90,
360,1,7  );
    TH2F *h2_t_p2_mg = new TH2F("h2_t_p2_mg","h2_t_p2_mg", 360,0,90,
360,1,7  );
    TH2F *h2_t_p1_p2 = new TH2F("h2_t_p1_p2","h2_t_p1_p2", 360,0,90,
360,0,90);
    // theta and phi of the detected particles
    TH2F *h2_t_p_p1 = new TH2F("h2_t_p_p1","h2_t_p_p1", 360,-180,180, 3
    TH2F *h2_t_p_mg = new TH2F("h2_t_p_mg","h2_t_p_mg", 360,-180,180, 3
    TH2F *h2_t_p_p2 = new TH2F("h2_t_p_p2","h2_t_p_p2", 360,-180,180, 3

    //ecm theta_cm calculated from p1 (p2 spectator?)
```

```
TH2F *h2_e_t_cm_1 = new TH2F("h2_e_t_cm_1","h2_e_t_cm_1", 90,0,180,
//ecm theta_cm calculated from p2 (p1 spectator?)
TH2F *h2_e_t_cm_2 = new TH2F("h2_e_t_cm_2","h2_e_t_cm_2", 90,0,180,

//ecm ps calculated from p1 (p2 spectator?)
TH2F *h2_ecm_ps_1 = new TH2F("h2_ecm_ps_1","h2_ecm_ps_1", 100,-200,
//ecm ps calculated from p2 (p1 spectator?)
TH2F *h2_ecm_ps_2 = new TH2F("h2_ecm_ps_2","h2_ecm_ps_2", 100,-200,

//1D ecm spectra: from p1 (p2 spectator?)
TH1F *h1_ecm_1 = new TH1F("h1_ecm_1","h1_ecm_1", 120,-1,2);
//1D ecm spectra: from p2 (p1 spectator?)
TH1F *h1_ecm_2 = new TH1F("h1_ecm_2","h1_ecm_2", 120,-1,2);

//control spectra calculated with infinite resolution
//ecm ps calculated from p1 (p2 spectator?)
TH2F *h2_ecm_ps_1_i = new TH2F("h2_ecm_ps_1_i","h2_ecm_ps_1_i", 100
//ecm ps calculated from p2 (p1 spectator?)
TH2F *h2_ecm_ps_2_i = new TH2F("h2_ecm_ps_2_i","h2_ecm_ps_2_i", 100
//1D ecm spectra: from p1 (p2 spectator?)
TH1F *h1_ecm_1_i = new TH1F("h1_ecm_1_i","h1_ecm_1_i", 120,-1,2);
//1D ecm spectra: from p2 (p1 spectator?)
TH1F *h1_ecm_2_i = new TH1F("h1_ecm_2_i","h1_ecm_2_i", 120,-1,2);


//multiple hit counter (two particles in the same detector) -> reje
Int_t nmulth=0;

//number of coincidences
Int_t ncoinc=0;

TGenPhaseSpace event;

fout->cd();
```

```
Double_t ninc=5e06; //pps

cout << "inizio" << endl;

for (int nfired=1; nfired<=ninc; nfired++) {

    Double_t rp = ran->Rndm();

    // energia e impulso del proiettile
    epfake=((90.92919-(0.7394*2))+rp*(0.7394*2))/1000.;


    Double_t      epf=(90.92919-0.7394)/1000.; //apparent beam energ
    Double_t ep=epf; // added on 27 September 2024
    Double_t      pp=sqrt(2.*u*mp*ep);
    Double_t        ppf=sqrt(2.*u*mp*epf);
    TLorentzVector beamf(0.0, 0.0, ppf, u*mp+epf);

    TLorentzVector target(0.0, 0.0, 0.0, u*mt);
    TLorentzVector beam(0.0, 0.0, pp, u*mp+ep);
    TLorentzVector W = beam + target;

    event.SetDecay(W, 2, masses);

    Double_t weight = event.Generate();

   WW=weight;

    TLorentzVector *p1 = event.GetDecay(0); //protone 1
    TLorentzVector *p2 = event.GetDecay(1); //26Mg

    TVector3 pp1=p1->Vect();
    TVector3 pp2=p2->Vect();
```

```
TVector3 dir1=pp1.Unit();
TVector3 dir2=pp2.Unit();


// dimensione del beam spot: diametro 0.2 cm
Double_t rx = ran->Rndm();
Double_t ry = ran->Rndm();

Double_t postx=-0.1+rx*0.2;
Double_t posty=-0.1+ry*0.2;
if(postx*postx+posty*posty>=0.01)continue;


// particles tracking

Double_t xx1=dir1(0);
Double_t yy1=dir1(1);
Double_t zz1=dir1(2);
gGeoManager->SetCurrentPoint(postx,posty,0);
gGeoManager->SetCurrentDirection(xx1,yy1,zz1);
TGeoNode *current1 = gGeoManager->GetCurrentNode();
gGeoManager->FindNode();
gGeoManager->FindNextBoundary(radius);
Double_t snext1 = gGeoManager->GetStep();
TGeoNode *newNode1 = gGeoManager->Step();
Bool_t istate1 = gGeoManager->IsStepEntering();
Int_t idnode1 = gGeoManager->GetCurrentNodeId();

TGeoNode *current11 = gGeoManager->GetCurrentNode();
const Double_t *cpoint1 = gGeoManager->GetCurrentPoint();
TVector3 vhp1(cpoint1[0],cpoint1[1],cpoint1[2]); //hit position
```

```
Double_t xx2=dir2(0);
Double_t yy2=dir2(1);
Double_t zz2=dir2(2);
gGeoManager->SetCurrentPoint(postx,posty,0);
gGeoManager->SetCurrentDirection(xx2,yy2,zz2);
TGeoNode *current2 = gGeoManager->GetCurrentNode();
gGeoManager->FindNode();
gGeoManager->FindNextBoundary(radius);
Double_t snext2 = gGeoManager->GetStep();
TGeoNode *newNode2 = gGeoManager->Step();
Bool_t istate2 = gGeoManager->IsStepEntering();
Int_t idnode2 = gGeoManager->GetCurrentNodeId();

TGeoNode *current22 = gGeoManager->GetCurrentNode();
const Double_t *cpoint2 = gGeoManager->GetCurrentPoint();
TVector3 vhp2(cpoint2[0],cpoint2[1],cpoint2[2]); //hit position


//flagging undetected particles
Int_t lost1flag=0;
Int_t lost2flag=0;




//apparent direction of the particles from detection pixel
TVector3 vbinp1;
TVector3 vbinmg;

//4-momenta
TLorentzVector pb1; //p1 4-momentum in GeV/c
TLorentzVector pb2; //26Mg 4-momentum in GeV/c

//vector momenta (parte spaziale del 4-momentum)
TVector3 ppb1;
```

```
TVector3 ppb2;


// setting the coincidence level >=2: two particles hitting the

if(istate1 + istate2 >1) {

    ncoinc++;

    if(idnode1==idnode2)
    {
        nmulth++;
        continue;
    }

    //searching for proton 1 apparent trajectory

    if      (idnode1==idA1) {
        //cout << "p1 in A1" << endl;
        TVector3 vdA1=vhp1-vA1;
        //vector from det. A1 center to hit point

        Double_t vvdA1[3]={vdA1(0),vdA1(1),vdA1(2)};
        Double_t vvdA1r[3]={0.,0.,0.};

        //cout << vdA1(0) << " " << vdA1(1) << " " << vdA1(2) <

        //this is the transformation to planar configuration
        rot1->MasterToLocal(vvdA1,vvdA1r);
        //for A1, B1 +largeZ, B32 -largeZ
        //          F1 -largeX, F32 +largeX

        TVector3 vdA1r(vvdA1r[0],vvdA1r[1],vvdA1r[2]);
```

```
//cout << vdA1r(0) << " " << vdA1r(1) << " " << vdA1r(2

Double_t detA1f=vvdA1r[0];
Double_t detA1b=vvdA1r[2];

Int_t ii,jj;

//search for the pixel f->front, b->back
for (ii=0; ii<32; ii++){
    if(detA1f>=-2.5+ii*stepA && detA1f<-2.5+(ii+1)*step
}
for (jj=0; jj<32; jj++){
    if(detA1b<2.5-jj*stepA && detA1b>=2.5-(jj+1)*stepA)
}
Double_t rxxs = ran->Rndm();
Double_t ryys = ran->Rndm();
Double_t cfA1=-2.5+ii*stepA+stepA*rxxs;
Double_t cbA1=2.5-jj*stepA-stepA*ryys;

//binned (apparent->a) position on detector A1
Double_t avdA1r[3]={cfA1,vvdA1r[1],cbA1};
Double_t avdA1[3]={0.,0.,0.};

//we go back to the real position in the space
rot1->LocalToMaster(avdA1r,avdA1);

//this is the vector from det.A1 center to apparent hit
TVector3 vdA1a(avdA1[0],avdA1[1],avdA1[2]);

//cout << vdA1a(0) << " " << vdA1a(1) << " " << vdA1a(2

//apparent trajectory of proton 1
TVector3 vbinA1=vA1+vdA1a;
Tp1=(vbinA1.Theta())*deg;
```

```
        Pp1=(vbinA1.Phi())*deg;
        vbinp1=vbinA1;

    }
    else if (idnode1==idA2) {
        //cout << "p1 in A2" << endl;
        TVector3 vdA2=vhp1-vA2;
        //vector from det. A2 center to hit point

        Double_t vvdA2[3]={vdA2(0),vdA2(1),vdA2(2)};
        Double_t vvdA2r[3]={0.,0.,0.};

        //cout << vdA2(0) << " " << vdA2(1) << " " << vdA2(2) <

        //this is the transformation to planar configuration
        rot2->MasterToLocal(vvdA2,vvdA2r);
        //for A2, B1 +largeZ, B32 -largeZ
        //          F1 -largeX, F32 +largeX

        TVector3 vdA2r(vvdA2r[0],vvdA2r[1],vvdA2r[2]);

        //cout << vdA2r(0) << " " << vdA2r(1) << " " << vdA2r(2

        Double_t detA2f=vvdA2r[0];
        Double_t detA2b=vvdA2r[2];

        Int_t ii,jj;

        //search for the pixel f->front, b->back
        for (ii=0; ii<32; ii++){
            if(detA2f>=-2.5+ii*stepA && detA2f<-2.5+(ii+1)*step
        }
        for (jj=0; jj<32; jj++){
            if(detA2b<2.5-jj*stepA && detA2b>=2.5-(jj+1)*stepA)
```

```
     }
     Double_t rxxs = ran->Rndm();
     Double_t ryys = ran->Rndm();
     Double_t cfA2=-2.5+ii*stepA+stepA*rxxs;
     Double_t cbA2=2.5-jj*stepA-stepA*ryys;

     //binned (apparent->a) position on detector A2
     Double_t avdA2r[3]={cfA2,vvdA2r[1],cbA2};
     Double_t avdA2[3]={0.,0.,0.};

     //we go back to the real position in the space
     rot2->LocalToMaster(avdA2r,avdA2);

     //this is the vector from det.A2 center to apparent hit
     TVector3 vdA2a(avdA2[0],avdA2[1],avdA2[2]);

     //cout << vdA2a(0) << " " << vdA2a(1) << " " << vdA2a(2

     //apparent trajectory of proton 1
     TVector3 vbinA2=vA2+vdA2a;
     Tp1=(vbinA2.Theta())*deg;
     Pp1=(vbinA2.Phi())*deg;
     vbinp1=vbinA2;

}
else if (idnode1==idA3) {
     //cout << "p1 in A3" << endl;
     TVector3 vdA3=vhp1-vA3;
     //vector from det. A3 center to hit point

     Double_t vvdA3[3]={vdA3(0),vdA3(1),vdA3(2)};
     Double_t vvdA3r[3]={0.,0.,0.};

     //cout << vdA3(0) << " " << vdA3(1) << " " << vdA3(2) <
```

```
//this is the transformation to planar configuration
rot3->MasterToLocal(vvdA3,vvdA3r);
//for A3, B1 +largeZ, B32 -largeZ
//        F1 +largeX, F32 -largeX

TVector3 vdA3r(vvdA3r[0],vvdA3r[1],vvdA3r[2]);

//cout << vdA3r(0) << " " << vdA3r(1) << " " << vdA3r(2

Double_t detA3f=vvdA3r[0];
Double_t detA3b=vvdA3r[2];

Int_t ii,jj;

//search for the pixel f->front, b->back
for (ii=0; ii<32; ii++){
    if(detA3f<2.5-ii*stepA && detA3f>=2.5-(ii+1)*stepA)
}
for (jj=0; jj<32; jj++){
    if(detA3b<2.5-jj*stepA && detA3b>=2.5-(jj+1)*stepA)
}
Double_t rxxs = ran->Rndm();
Double_t ryys = ran->Rndm();
Double_t cfA3=2.5-ii*stepA-stepA*rxxs;
Double_t cbA3=2.5-jj*stepA-stepA*ryys;

//binned (apparent->a) position on detector A3
Double_t avdA3r[3]={cfA3,vvdA3r[1],cbA3};
Double_t avdA3[3]={0.,0.,0.};

//we go back to the real position in the space
rot3->LocalToMaster(avdA3r,avdA3);
```

```
        //this is the vector from det.A3 center to apparent hit
        TVector3 vdA3a(avdA3[0],avdA3[1],avdA3[2]);

        //cout << vdA3a(0) << " " << vdA3a(1) << " " << vdA3a(2

        //apparent trajectory of proton 1
        TVector3 vbinA3=vA3+vdA3a;
        Tp1=(vbinA3.Theta())*deg;
        Pp1=(vbinA3.Phi())*deg;
        vbinp1=vbinA3;

    }
    else if (idnode1==idA4) {
        //cout << "p1 in A4" << endl;
        TVector3 vdA4=vhp1-vA4;
        //vector from det. A4 center to hit point

        Double_t vvdA4[3]={vdA4(0),vdA4(1),vdA4(2)};
        Double_t vvdA4r[3]={0.,0.,0.};

        //cout << vdA4(0) << " " << vdA4(1) << " " << vdA4(2) <

        //this is the transformation to planar configuration
        rot4->MasterToLocal(vvdA4,vvdA4r);
        //for A4, B1 +largeZ, B32 -largeZ
        //        F1 +largeX, F32 -largeX

        TVector3 vdA4r(vvdA4r[0],vvdA4r[1],vvdA4r[2]);

        //cout << vdA4r(0) << " " << vdA4r(1) << " " << vdA4r(2

        Double_t detA4f=vvdA4r[0];
        Double_t detA4b=vvdA4r[2];
        Int_t ii,jj;

        //search for the pixel f->front, b->back
        for (ii=0; ii<32; ii++){
            if(detA4f<2.5-ii*stepA && detA4f>=2.5-(ii+1)*stepA)
        }
        for (jj=0; jj<32; jj++){
            if(detA4b<2.5-jj*stepA && detA4b>=2.5-(jj+1)*stepA)
        }
        Double_t rxxs = ran->Rndm();
        Double_t ryys = ran->Rndm();
        Double_t cfA4=2.5-ii*stepA-stepA*rxxs;
        Double_t cbA4=2.5-jj*stepA-stepA*ryys;

        //binned (apparent->a) position on detector A4
        Double_t avdA4r[3]={cfA4,vvdA4r[1],cbA4};
        Double_t avdA4[3]={0.,0.,0.};

        //we go back to the real position in the space
        rot4->LocalToMaster(avdA4r,avdA4);

        //this is the vector from det.A4 center to apparent hit
        TVector3 vdA4a(avdA4[0],avdA4[1],avdA4[2]);

        //cout << vdA4a(0) << " " << vdA4a(1) << " " << vdA4a(2

        //apparent trajectory of proton 1
        TVector3 vbinA4=vA4+vdA4a;
        Tp1=(vbinA4.Theta())*deg;
        Pp1=(vbinA4.Phi())*deg;
        vbinp1=vbinA4;

    }
    else if (idnode1==idB1) {
```

```
//cout << "p1 in B1" << endl;
TVector3 vdB1=vhp1-vB1;
//vector from det. B1 center to hit point

Double_t vvdB1[3]={vdB1(0),vdB1(1),vdB1(2)};
Double_t vvdB1r[3]={0.,0.,0.};

//cout << vdB1(0) << " " << vdB1(1) << " " << vdB1(2) <

//this is the transformation to planar configuration
rot5->MasterToLocal(vvdB1,vvdB1r);
//for B1, B1 +largeZ, B16 -largeZ
//         F1 -largeX, F16 +largeX

TVector3 vdB1r(vvdB1r[0],vvdB1r[1],vvdB1r[2]);

//cout << vdB1r(0) << " " << vdB1r(1) << " " << vdB1r(2

Double_t detB1f=vvdB1r[0];
Double_t detB1b=vvdB1r[2];

Int_t ii,jj;

//search for the pixel f->front, b->back
for (ii=0; ii<16; ii++){
    if(detB1f>=-2.5+ii*step && detB1f<-2.5+(ii+1)*step)
}
for (jj=0; jj<16; jj++){
    if(detB1b<2.5-jj*step && detB1b>=2.5-(jj+1)*step) b
}
Double_t rxxs = ran->Rndm();
Double_t ryys = ran->Rndm();
Double_t cfB1=-2.5+ii*step+step*rxxs;
Double_t cbB1=2.5-jj*step-step*ryys;
```

```
    //binned (apparent->a) position on detector B1
    Double_t avdB1r[3]={cfB1,vvdB1r[1],cbB1};
    Double_t avdB1[3]={0.,0.,0.};

    //we go back to the real position in the space
    rot5->LocalToMaster(avdB1r,avdB1);

    //this is the vector from det.B1 center to apparent hit
    TVector3 vdB1a(avdB1[0],avdB1[1],avdB1[2]);

    //cout << vdB1a(0) << " " << vdB1a(1) << " " << vdB1a(2

    //apparent trajectory of proton 1
    TVector3 vbinB1=vB1+vdB1a;
    Tp1=(vbinB1.Theta())*deg;
    Pp1=(vbinB1.Phi())*deg;
    vbinp1=vbinB1;

}
else if (idnode1==idB2) {
    //cout << "p1 in B2" << endl;
    TVector3 vdB2=vhp1-vB2;
    //vector from det. B2 center to hit point

    Double_t vvdB2[3]={vdB2(0),vdB2(1),vdB2(2)};
    Double_t vvdB2r[3]={0.,0.,0.};

    //cout << vdB2(0) << " " << vdB2(1) << " " << vdB2(2) <

    //this is the transformation to planar configuration
    rot6->MasterToLocal(vvdB2,vvdB2r);
    //for B2, B1 +largeZ, B16 -largeZ
    //          F1 -largeX, F16 +largeX
```

```
TVector3 vdB2r(vvdB2r[0],vvdB2r[1],vvdB2r[2]);

//cout << vdB2r(0) << " " << vdB2r(1) << " " << vdB2r(2

Double_t detB2f=vvdB2r[0];
Double_t detB2b=vvdB2r[2];

Int_t ii,jj;

//search for the pixel f->front, b->back
for (ii=0; ii<16; ii++){
    if(detB2f>=-2.5+ii*step && detB2f<-2.5+(ii+1)*step)
}
for (jj=0; jj<16; jj++){
    if(detB2b<2.5-jj*step && detB2b>=2.5-(jj+1)*step) b
}
Double_t rxxs = ran->Rndm();
Double_t ryys = ran->Rndm();
Double_t cfB2=-2.5+ii*step+step*rxxs;
Double_t cbB2=2.5-jj*step-step*ryys;

//binned (apparent->a) position on detector B2
Double_t avdB2r[3]={cfB2,vvdB2r[1],cbB2};
Double_t avdB2[3]={0.,0.,0.};

//we go back to the real position in the space
rot6->LocalToMaster(avdB2r,avdB2);

//this is the vector from det.B2 center to apparent hit
TVector3 vdB2a(avdB2[0],avdB2[1],avdB2[2]);

//cout << vdB2a(0) << " " << vdB2a(1) << " " << vdB2a(2
    //apparent trajectory of proton 1
    TVector3 vbinB2=vB2+vdB2a;
    Tp1=(vbinB2.Theta())*deg;
    Pp1=(vbinB2.Phi())*deg;
    vbinp1=vbinB2;

}
else if (idnode1==idB3) {
    //cout << "p1 in B3" << endl;
    TVector3 vdB3=vhp1-vB3;
    //vector from det. B3 center to hit point

    Double_t vvdB3[3]={vdB3(0),vdB3(1),vdB3(2)};
    Double_t vvdB3r[3]={0.,0.,0.};

    //cout << vdB3(0) << " " << vdB3(1) << " " << vdB3(2) <

    //this is the transformation to planar configuration
    rot7->MasterToLocal(vvdB3,vvdB3r);
    //for B3, B1 +largeZ, B16 -largeZ
    //         F1 +largeX, F16 -largeX

    TVector3 vdB3r(vvdB3r[0],vvdB3r[1],vvdB3r[2]);

    //cout << vdB3r(0) << " " << vdB3r(1) << " " << vdB3r(2

    Double_t detB3f=vvdB3r[0];
    Double_t detB3b=vvdB3r[2];

    Int_t ii,jj;

    //search for the pixel f->front, b->back
    for (ii=0; ii<16; ii++){
        if(detB3f<2.5-ii*step && detB3f>=2.5-(ii+1)*step) b
    }
    for (jj=0; jj<16; jj++){
        if(detB3b<2.5-jj*step && detB3b>=2.5-(jj+1)*step) b
    }
    Double_t rxxs = ran->Rndm();
    Double_t ryys = ran->Rndm();
    Double_t cfB3=2.5-ii*step-step*rxxs;
    Double_t cbB3=2.5-jj*step-step*ryys;

    //binned (apparent->a) position on detector B3
    Double_t avdB3r[3]={cfB3,vvdB3r[1],cbB3};
    Double_t avdB3[3]={0.,0.,0.};

    //we go back to the real position in the space
    rot7->LocalToMaster(avdB3r,avdB3);

    //this is the vector from det.B3 center to apparent hit
    TVector3 vdB3a(avdB3[0],avdB3[1],avdB3[2]);

    //cout << vdB3a(0) << " " << vdB3a(1) << " " << vdB3a(2

    //apparent trajectory of proton 1
    TVector3 vbinB3=vB3+vdB3a;
    Tp1=(vbinB3.Theta())*deg;
    Pp1=(vbinB3.Phi())*deg;
    vbinp1=vbinB3;

}
else if (idnode1==idB4) {
    //cout << "p1 in B4" << endl;
    TVector3 vdB4=vhp1-vB4;
    //vector from det. B4 center to hit point

    Double_t vvdB4[3]={vdB4(0),vdB4(1),vdB4(2)};
```

```
Double_t vvdB4r[3]={0.,0.,0.};

//cout << vdB4(0) << " " << vdB4(1) << " " << vdB4(2) <

//this is the transformation to planar configuration
rot8->MasterToLocal(vvdB4,vvdB4r);
//for B4, B1 +largeZ, B16 -largeZ
//        F1 +largeX, F16 -largeX

TVector3 vdB4r(vvdB4r[0],vvdB4r[1],vvdB4r[2]);

//cout << vdB4r(0) << " " << vdB4r(1) << " " << vdB4r(2

Double_t detB4f=vvdB4r[0];
Double_t detB4b=vvdB4r[2];

Int_t ii,jj;

//search for the pixel f->front, b->back
for (ii=0; ii<16; ii++){
    if(detB4f<2.5-ii*step && detB4f>=2.5-(ii+1)*step) b
}
for (jj=0; jj<16; jj++){
    if(detB4b<2.5-jj*step && detB4b>=2.5-(jj+1)*step) b
}
Double_t rxxs = ran->Rndm();
Double_t ryys = ran->Rndm();
Double_t cfB4=2.5-ii*step-step*rxxs;
Double_t cbB4=2.5-jj*step-step*ryys;

//binned (apparent->a) position on detector B4
Double_t avdB4r[3]={cfB4,vvdB4r[1],cbB4};
Double_t avdB4[3]={0.,0.,0.};
```

```
          //we go back to the real position in the space
          rot8->LocalToMaster(avdB4r,avdB4);

          //this is the vector from det.B4 center to apparent hit
          TVector3 vdB4a(avdB4[0],avdB4[1],avdB4[2]);

          //cout << vdB4a(0) << " " << vdB4a(1) << " " << vdB4a(2

          //apparent trajectory of proton 1
          TVector3 vbinB4=vB4+vdB4a;
          Tp1=(vbinB4.Theta())*deg;
          Pp1=(vbinB4.Phi())*deg;
          vbinp1=vbinB4;

     }
     else{
          //cout << "p1 lost in space" << endl;

          lost1flag=1;
     }

     //searching for 26Mg apparent trajectory

     if        (idnode2==idA1){
          //cout << "26Mg in A1" << endl;
          TVector3 vdA1x=vhp2-vA1;
          //vector from det. A1 center to hit point

          Double_t vvdA1x[3]={vdA1x(0),vdA1x(1),vdA1x(2)};
          Double_t vvdA1xr[3]={0.,0.,0.};

          //cout << vdA1x(0) << " " << vdA1x(1) << " " << vdA1x(2

          //this is the transformation to planar configuration
```

```
rot1->MasterToLocal(vvdA1x,vvdA1xr);
//for A1, B1 +largeZ, B32 -largeZ
//          F1 -largeX, F32 +largeX

TVector3 vdA1xr(vvdA1xr[0],vvdA1xr[1],vvdA1xr[2]);

//cout << vdA1xr(0) << " " << vdA1xr(1) << " " << vdA1x

Double_t detA1xf=vvdA1xr[0];
Double_t detA1xb=vvdA1xr[2];

Int_t ii,jj;

//search for the pixel f->front, b->back
for (ii=0; ii<32; ii++){
    if(detA1xf>=-2.5+ii*stepA && detA1xf<-2.5+(ii+1)*st
}
for (jj=0; jj<32; jj++){
    if(detA1xb<2.5-jj*stepA && detA1xb>=2.5-(jj+1)*step
}
Double_t rxxs = ran->Rndm();
Double_t ryys = ran->Rndm();
Double_t cfA1x=-2.5+ii*stepA+stepA*rxxs;
Double_t cbA1x=2.5-jj*stepA-stepA*ryys;

//binned (apparent->a) position on detector A1
Double_t avdA1xr[3]={cfA1x,vvdA1xr[1],cbA1x};
Double_t avdA1x[3]={0.,0.,0.};

//we go back to the real position in the space
rot1->LocalToMaster(avdA1xr,avdA1x);

//this is the vector from det.A1 center to apparent hit
TVector3 vdA1xa(avdA1x[0],avdA1x[1],avdA1x[2]);
```

```
        //cout << vdA1xa(0) << " " << vdA1xa(1) << " " << vdA1x

        //apparent trajectory of 26Mg
        TVector3 vbinA1x=vA1+vdA1xa;
        Tmg=(vbinA1x.Theta())*deg;
        Pmg=(vbinA1x.Phi())*deg;
        vbinmg=vbinA1x;

    }
    else if (idnode2==idA2) {
        //cout << "26Mg in A2" << endl;
        TVector3 vdA2x=vhp2-vA2;
        //vector from det. A2 center to hit point

        Double_t vvdA2x[3]={vdA2x(0),vdA2x(1),vdA2x(2)};
        Double_t vvdA2xr[3]={0.,0.,0.};

        //cout << vdA2x(0) << " " << vdA2x(1) << " " << vdA2x(2

        //this is the transformation to planar configuration
        rot2->MasterToLocal(vvdA2x,vvdA2xr);
        //for A2, B1 +largeZ, B32 -largeZ
        //         F1 -largeX, F32 +largeX

        TVector3 vdA2xr(vvdA2xr[0],vvdA2xr[1],vvdA2xr[2]);

        //cout << vdA2xr(0) << " " << vdA2xr(1) << " " << vdA2x

        Double_t detA2xf=vvdA2xr[0];
        Double_t detA2xb=vvdA2xr[2];

        Int_t ii,jj;
        //search for the pixel f->front, b->back
        for (ii=0; ii<32; ii++){
            if(detA2xf>=-2.5+ii*stepA && detA2xf<-2.5+(ii+1)*st
        }
        for (jj=0; jj<32; jj++){
            if(detA2xb<2.5-jj*stepA && detA2xb>=2.5-(jj+1)*step
        }
        Double_t rxxs = ran->Rndm();
        Double_t ryys = ran->Rndm();
        Double_t cfA2x=-2.5+ii*stepA+stepA*rxxs;
        Double_t cbA2x=2.5-jj*stepA-stepA*ryys;

        //binned (apparent->a) position on detector A2
        Double_t avdA2xr[3]={cfA2x,vvdA2xr[1],cbA2x};
        Double_t avdA2x[3]={0.,0.,0.};

        //we go back to the real position in the space
        rot2->LocalToMaster(avdA2xr,avdA2x);

        //this is the vector from det.A2 center to apparent hit
        TVector3 vdA2xa(avdA2x[0],avdA2x[1],avdA2x[2]);

        //cout << vdA2xa(0) << " " << vdA2xa(1) << " " << vdA2x

        //apparent trajectory of 26Mg
        TVector3 vbinA2x=vA2+vdA2xa;
        Tmg=(vbinA2x.Theta())*deg;
        Pmg=(vbinA2x.Phi())*deg;
        vbinmg=vbinA2x;

    }
    else if (idnode2==idA3) {
        //cout << "26Mg in A3" << endl;
        TVector3 vdA3x=vhp2-vA3;
```

```
//vector from det. A3 center to hit point

Double_t vvdA3x[3]={vdA3x(0),vdA3x(1),vdA3x(2)};
Double_t vvdA3xr[3]={0.,0.,0.};

//cout << vdA3x(0) << " " << vdA3x(1) << " " << vdA3x(2

//this is the transformation to planar configuration
rot3->MasterToLocal(vvdA3x,vvdA3xr);
//for A3, B1 +largeZ, B32 -largeZ
//         F1 +largeX, F32 -largeX

TVector3 vdA3xr(vvdA3xr[0],vvdA3xr[1],vvdA3xr[2]);

//cout << vdA3xr(0) << " " << vdA3xr(1) << " " << vdA3x

Double_t detA3xf=vvdA3xr[0];
Double_t detA3xb=vvdA3xr[2];

Int_t ii,jj;

//search for the pixel f->front, b->back
for (ii=0; ii<32; ii++){
    if(detA3xf<2.5-ii*stepA && detA3xf>=2.5-(ii+1)*step
}
for (jj=0; jj<32; jj++){
    if(detA3xb<2.5-jj*stepA && detA3xb>=2.5-(jj+1)*step
}
Double_t rxxs = ran->Rndm();
Double_t ryys = ran->Rndm();
Double_t cfA3x=2.5-ii*stepA-stepA*rxxs;
Double_t cbA3x=2.5-jj*stepA-stepA*ryys;

//binned (apparent->a) position on detector A3
```

```
        Double_t avdA3xr[3]={cfA3x,vvdA3xr[1],cbA3x};
        Double_t avdA3x[3]={0.,0.,0.};

        //we go back to the real position in the space
        rot3->LocalToMaster(avdA3xr,avdA3x);

        //this is the vector from det.A3 center to apparent hit
        TVector3 vdA3xa(avdA3x[0],avdA3x[1],avdA3x[2]);

        //cout << vdA3xa(0) << " " << vdA3xa(1) << " " << vdA3x

        //apparent trajectory of 26Mg
        TVector3 vbinA3x=vA3+vdA3xa;
        Tmg=(vbinA3x.Theta())*deg;
        Pmg=(vbinA3x.Phi())*deg;
        vbinmg=vbinA3x;

    }
    else if (idnode2==idA4) {
        //cout << "26Mg in A4" << endl;
        TVector3 vdA4x=vhp2-vA4;
        //vector from det. A4 center to hit point

        Double_t vvdA4x[3]={vdA4x(0),vdA4x(1),vdA4x(2)};
        Double_t vvdA4xr[3]={0.,0.,0.};

        //cout << vdA4x(0) << " " << vdA4x(1) << " " << vdA4x(2

        //this is the transformation to planar configuration
        rot4->MasterToLocal(vvdA4x,vvdA4xr);
        //for A4, B1 +largeZ, B32 -largeZ
        //          F1 +largeX, F32 -largeX

        TVector3 vdA4xr(vvdA4xr[0],vvdA4xr[1],vvdA4xr[2]);
```

```
//cout << vdA4xr(0) << " " << vdA4xr(1) << " " << vdA4x

Double_t detA4xf=vvdA4xr[0];
Double_t detA4xb=vvdA4xr[2];

Int_t ii,jj;

//search for the pixel f->front, b->back
for (ii=0; ii<32; ii++){
    if(detA4xf<2.5-ii*stepA && detA4xf>=2.5-(ii+1)*step
}
for (jj=0; jj<32; jj++){
    if(detA4xb<2.5-jj*stepA && detA4xb>=2.5-(jj+1)*step
}
Double_t rxxs = ran->Rndm();
Double_t ryys = ran->Rndm();
Double_t cfA4x=2.5-ii*stepA-stepA*rxxs;
Double_t cbA4x=2.5-jj*stepA-stepA*ryys;

//binned (apparent->a) position on detector A4
Double_t avdA4xr[3]={cfA4x,vvdA4xr[1],cbA4x};
Double_t avdA4x[3]={0.,0.,0.};

//we go back to the real position in the space
rot4->LocalToMaster(avdA4xr,avdA4x);

//this is the vector from det.A4 center to apparent hit
TVector3 vdA4xa(avdA4x[0],avdA4x[1],avdA4x[2]);

//cout << vdA4xa(0) << " " << vdA4xa(1) << " " << vdA4x

//apparent trajectory of 26Mg
TVector3 vbinA4x=vA4+vdA4xa;
```

```
    Tmg=(vbinA4x.Theta())*deg;
    Pmg=(vbinA4x.Phi())*deg;
    vbinmg=vbinA4x;

}
else if (idnode2==idB1) {
    //cout << "26Mg in B1" << endl;
    TVector3 vdB1x=vhp2-vB1;
    //vector from det. B1 center to hit point

    Double_t vvdB1x[3]={vdB1x(0),vdB1x(1),vdB1x(2)};
    Double_t vvdB1xr[3]={0.,0.,0.};

    //cout << vdB1x(0) << " " << vdB1x(1) << " " << vdB1x(2

    //this is the transformation to planar configuration
    rot5->MasterToLocal(vvdB1x,vvdB1xr);
    //for B1, B1 +largeZ, B16 -largeZ
    //         F1 -largeX, F16 +largeX

    TVector3 vdB1xr(vvdB1xr[0],vvdB1xr[1],vvdB1xr[2]);

    //cout << vdB1xr(0) << " " << vdB1xr(1) << " " << vdB1x

    Double_t detB1xf=vvdB1xr[0];
    Double_t detB1xb=vvdB1xr[2];

    Int_t ii,jj;

    //search for the pixel f->front, b->back
    for (ii=0; ii<16; ii++){
        if(detB1xf>=-2.5+ii*step && detB1xf<-2.5+(ii+1)*ste
    }
    for (jj=0; jj<16; jj++){
        if(detB1xb<2.5-jj*step && detB1xb>=2.5-(jj+1)*step)
    }
    Double_t rxxs = ran->Rndm();
    Double_t ryys = ran->Rndm();
    Double_t cfB1x=-2.5+ii*step+step*rxxs;
    Double_t cbB1x=2.5-jj*step-step*ryys;

    //binned (apparent->a) position on detector B1
    Double_t avdB1xr[3]={cfB1x,vvdB1xr[1],cbB1x};
    Double_t avdB1x[3]={0.,0.,0.};

    //we go back to the real position in the space
    rot5->LocalToMaster(avdB1xr,avdB1x);

    //this is the vector from det.B1 center to apparent hit
    TVector3 vdB1xa(avdB1x[0],avdB1x[1],avdB1x[2]);

    //cout << vdB1xa(0) << " " << vdB1xa(1) << " " << vdB1x

    //apparent trajectory of 26Mg
    TVector3 vbinB1x=vB1+vdB1xa;
    Tmg=(vbinB1x.Theta())*deg;
    Pmg=(vbinB1x.Phi())*deg;
    vbinmg=vbinB1x;

}
else if (idnode2==idB2) {
    //cout << "26Mg in B2" << endl;
    TVector3 vdB2x=vhp2-vB2;
    //vector from det. B2 center to hit point

    Double_t vvdB2x[3]={vdB2x(0),vdB2x(1),vdB2x(2)};
    Double_t vvdB2xr[3]={0.,0.,0.};
```

```
//cout << vdB2x(0) << " " << vdB2x(1) << " " << vdB2x(2

//this is the transformation to planar configuration
rot6->MasterToLocal(vvdB2x,vvdB2xr);
//for B2, B1 +largeZ, B16 -largeZ
//         F1 -largeX, F16 +largeX

TVector3 vdB2xr(vvdB2xr[0],vvdB2xr[1],vvdB2xr[2]);

//cout << vdB2xr(0) << " " << vdB2xr(1) << " " << vdB2x

Double_t detB2xf=vvdB2xr[0];
Double_t detB2xb=vvdB2xr[2];

Int_t ii,jj;

//search for the pixel f->front, b->back
for (ii=0; ii<16; ii++){
    if(detB2xf>=-2.5+ii*step && detB2xf<-2.5+(ii+1)*ste
}
for (jj=0; jj<16; jj++){
    if(detB2xb<2.5-jj*step && detB2xb>=2.5-(jj+1)*step)
}
Double_t rxxs = ran->Rndm();
Double_t ryys = ran->Rndm();
Double_t cfB2x=-2.5+ii*step+step*rxxs;
Double_t cbB2x=2.5-jj*step-step*ryys;

//binned (apparent->a) position on detector B2
Double_t avdB2xr[3]={cfB2x,vvdB2xr[1],cbB2x};
Double_t avdB2x[3]={0.,0.,0.};

//we go back to the real position in the space
rot6->LocalToMaster(avdB2xr,avdB2x);
```

```
    //this is the vector from det.B2 center to apparent hit
    TVector3 vdB2xa(avdB2x[0],avdB2x[1],avdB2x[2]);

    //cout << vdB2xa(0) << " " << vdB2xa(1) << " " << vdB2x

    //apparent trajectory of 26Mg
    TVector3 vbinB2x=vB2+vdB2xa;
    Tmg=(vbinB2x.Theta())*deg;
    Pmg=(vbinB2x.Phi())*deg;
    vbinmg=vbinB2x;

}
else if (idnode2==idB3) {
    //cout << "26Mg in B3" << endl;
    TVector3 vdB3x=vhp2-vB3;
    //vector from det. B3 center to hit point

    Double_t vvdB3x[3]={vdB3x(0),vdB3x(1),vdB3x(2)};
    Double_t vvdB3xr[3]={0.,0.,0.};

    //cout << vdB3x(0) << " " << vdB3x(1) << " " << vdB3x(2

    //this is the transformation to planar configuration
    rot7->MasterToLocal(vvdB3x,vvdB3xr);
    //for B3, B1 +largeZ, B16 -largeZ
    //         F1 +largeX, F16 -largeX

    TVector3 vdB3xr(vvdB3xr[0],vvdB3xr[1],vvdB3xr[2]);

    //cout << vdB3xr(0) << " " << vdB3xr(1) << " " << vdB3x

    Double_t detB3xf=vvdB3xr[0];
    Double_t detB3xb=vvdB3xr[2];
    Int_t ii,jj;

    //search for the pixel f->front, b->back
    for (ii=0; ii<16; ii++){
        if(detB3xf<2.5-ii*step && detB3xf>=2.5-(ii+1)*step)
    }
    for (jj=0; jj<16; jj++){
        if(detB3xb<2.5-jj*step && detB3xb>=2.5-(jj+1)*step)
    }
    Double_t rxxs = ran->Rndm();
    Double_t ryys = ran->Rndm();
    Double_t cfB3x=2.5-ii*step-step*rxxs;
    Double_t cbB3x=2.5-jj*step-step*ryys;

    //binned (apparent->a) position on detector B3
    Double_t avdB3xr[3]={cfB3x,vvdB3xr[1],cbB3x};
    Double_t avdB3x[3]={0.,0.,0.};

    //we go back to the real position in the space
    rot7->LocalToMaster(avdB3xr,avdB3x);

    //this is the vector from det.B3 center to apparent hit
    TVector3 vdB3xa(avdB3x[0],avdB3x[1],avdB3x[2]);

    //cout << vdB3xa(0) << " " << vdB3xa(1) << " " << vdB3x

    //apparent trajectory of 26Mg
    TVector3 vbinB3x=vB3+vdB3xa;
    Tmg=(vbinB3x.Theta())*deg;
    Pmg=(vbinB3x.Phi())*deg;
    vbinmg=vbinB3x;

}
```

```
else if (idnode2==idB4) {
    //cout << "26Mg in B4" << endl;
    TVector3 vdB4x=vhp2-vB4;
    //vector from det. B4 center to hit point

    Double_t vvdB4x[3]={vdB4x(0),vdB4x(1),vdB4x(2)};
    Double_t vvdB4xr[3]={0.,0.,0.};

    //cout << vdB4x(0) << " " << vdB4x(1) << " " << vdB4x(2

    //this is the transformation to planar configuration
    rot8->MasterToLocal(vvdB4x,vvdB4xr);
    //for B4, B1 +largeZ, B16 -largeZ
    //         F1 +largeX, F16 -largeX

    TVector3 vdB4xr(vvdB4xr[0],vvdB4xr[1],vvdB4xr[2]);

    //cout << vdB4xr(0) << " " << vdB4xr(1) << " " << vdB4x

    Double_t detB4xf=vvdB4xr[0];
    Double_t detB4xb=vvdB4xr[2];

    Int_t ii,jj;

    //search for the pixel f->front, b->back
    for (ii=0; ii<16; ii++){
        if(detB4xf<2.5-ii*step && detB4xf>=2.5-(ii+1)*step)
    }
    for (jj=0; jj<16; jj++){
        if(detB4xb<2.5-jj*step && detB4xb>=2.5-(jj+1)*step)
    }
    Double_t rxxs = ran->Rndm();
    Double_t ryys = ran->Rndm();
    Double_t cfB4x=2.5-ii*step-step*rxxs;
```

```
          Double_t cbB4x=2.5-jj*step-step*ryys;

          //binned (apparent->a) position on detector B4
          Double_t avdB4xr[3]={cfB4x,vvdB4xr[1],cbB4x};
          Double_t avdB4x[3]={0.,0.,0.};

          //we go back to the real position in the space
          rot8->LocalToMaster(avdB4xr,avdB4x);

          //this is the vector from det.B4 center to apparent hit
          TVector3 vdB4xa(avdB4x[0],avdB4x[1],avdB4x[2]);

          //cout << vdB4xa(0) << " " << vdB4xa(1) << " " << vdB4x

          //apparent trajectory of 26Mg
          TVector3 vbinB4x=vB4+vdB4xa;
          Tmg=(vbinB4x.Theta())*deg;
          Pmg=(vbinB4x.Phi())*deg;
          vbinmg=vbinB4x;

     }
     else{
          //cout << "26Mg lost in space" << endl;

          lost2flag=1;
     }




     //cout << endl;

     //associating the energy accounting for det. res.

     std::default_random_engine generator;
```

```
if(lost1flag==0)
{

    Double_t e1=(p1->Energy()-u*m1); //p1 energy in GeV
    Double_t mp1=pp1.Mag(); //p1 momentum in GeV/c
    Double_t rnnum1 = ran->Rndm();
    generator.seed(rnnum1);
    std::normal_distribution<double> distribution1(/*mean=*
    Double_t mp1s = distribution1(generator);
    Double_t dmp1=(mp1s-mp1)/mp1; //variazione percentuale
    TVector3 dirb1=vbinp1.Unit(); //apparent direction
    TVector3 vpb1=mp1s*dirb1; //p1 momentum (vector) in GeV

    pb1.SetPx(vpb1(0));
    pb1.SetPy(vpb1(1));
    pb1.SetPz(vpb1(2));
    pb1.SetE(u*m1+e1*(1+2*dmp1));
    ppb1=pb1.Vect();
    Ep1=e1*(1+2*dmp1)*1000.; //detected energy of p1

    fhisto->cd();
    h2_t_p_p1->Fill(Pp1,Tp1,weight);

}

if(lost2flag==0)
{

    Double_t e2=(p2->Energy()-u*m2); //26Mg energy in GeV
    Double_t mp2=pp2.Mag(); //26Mg momentum in GeV/c
    Double_t rnnum2 = ran->Rndm();
    generator.seed(rnnum2);
    std::normal_distribution<double> distribution2(/*mean=*
        Double_t mp2s = distribution2(generator);
        Double_t dmp2=(mp2s-mp2)/mp2; //variazione percentuale
        TVector3 dirb2=vbinmg.Unit(); //apparent direction
        TVector3 vpb2=mp2s*dirb2; //26Mg momentum (vector) in G

        pb2.SetPx(vpb2(0));
        pb2.SetPy(vpb2(1));
        pb2.SetPz(vpb2(2));
        pb2.SetE(u*m2+e2*(1+2*dmp2));
        ppb2=pb2.Vect();
        Emg=e2*(1+2*dmp2)*1000.; //detected energy of 26Mg

        fhisto->cd();
        h2_t_p_mg->Fill(Pmg,Tmg,weight);

    }

    //if particles are lost in space, I need to calculate angle

    if(lost1flag==1)
    {
        pb1=beamf+target-pb2; //prot.1 4-momentum in GeV/c
        Ep1=(pb1.Energy()-u*m1)*1000.;
        Tp1=(pb1.Theta())*deg;
        Pp1=(pb1.Phi())*deg;
        ppb1=pb1.Vect();



    }

    if(lost2flag==1)
    {
        pb2=beamf+target-pb1; //prot.1 4-momentum in GeV/c
        Emg=(pb2.Energy()-u*m2)*1000.;
        Tmg=(pb2.Theta())*deg;
        Pmg=(pb2.Phi())*deg;
        ppb2=pb2.Vect();


    }

    // this variable tells which particle is lost: 1->p1, 2->mg

    if(lost1flag==1)
    {
        undetected=1;
    }
    else if(lost2flag==1)
    {
        undetected=2;
    }
    else
    {
        undetected=0;
    }


/*
if(lost1flag==0 && Ep1>12.25 && lost2flag==0 && Tp1>10 && Tmg
    m++;
    Ep1exp=Ep1;
    Tp1exp=Tp1;
    Pp1exp=Pp1;
    Emgexp=Emg;
    Tmgexp=Tmg;
    Pmgexp=Pmg;
}
    */
```

```
            fout->cd();
            sim->Fill();


         if(nfired%50000==0) cout << "Fatti␣50000␣eventi" << endl;

      }

   }
 cout << "m:␣" << m << endl;

   cout << "good␣events␣=␣" << geve << "␣␣␣␣␣multihit␣=␣" << nmulth <<

   fout->cd();

   sim->Write();

   fout->Write();

   fhisto->cd();

   TCanvas *c1 = new TCanvas("c1","c1",100,0,800,1000);
   c1->Divide(1,3);
   c1->cd(1);
   h2_t_p_p1->Draw("colz");
   c1->cd(2);
   h2_t_p_mg->Draw("colz");
   c1->cd(3);
   h2_t_p_p2->Draw("colz");

   TCanvas *c2 = new TCanvas("c2","c2",200,0,800,800);
   c2->Divide(2,2);
   c2->cd(1);
   h2_e_p2_mg->Draw("colz");
```

```
c2->cd(2);
h2_t_p2_mg->Draw("colz");

TCanvas *c3 = new TCanvas("c3","c3",300,0,800,800);
c3->Divide(2,2);
c3->cd(1);
h2_e_p1_p2->Draw("colz");
c3->cd(2);
h2_t_p1_p2->Draw("colz");

TCanvas *c4 = new TCanvas("c4","c4",400,0,800,800);
c4->Divide(2,2);
c4->cd(1);
h2_e_p1_mg->Draw("colz");
c4->cd(2);
h2_t_p1_mg->Draw("colz");

TCanvas *c5 = new TCanvas("c5","c5",500,0,800,800);
c5->Divide(2,2);
c5->cd(1);
h2_e_t_cm_1->Draw("colz");
c5->cd(2);
h2_e_t_cm_2->Draw("colz");
c5->cd(3);
h2_ecm_ps_1->Draw("colz");
c5->cd(4);
h2_ecm_ps_2->Draw("colz");

TCanvas *c6 = new TCanvas("c6","c6",600,0,800,800);
c6->Divide(2,2);
c6->cd(1);
h2_ecm_ps_1_i->Draw("colz");
c6->cd(2);
h2_ecm_ps_2_i->Draw("colz");
```

```
TCanvas *c7 = new TCanvas("c7","c7",700,0,800,800);
c7->Divide(2,2);
c7->cd(1);
h1_ecm_1->Draw("colz");
c7->cd(2);
h1_ecm_2->Draw("colz");
c7->cd(3);
h1_ecm_1_i->Draw("colz");
c7->cd(4);
h1_ecm_2_i->Draw("colz");



h2_t_p_p1->Write();
h2_t_p_mg->Write();
h2_t_p_p2->Write();
h2_e_p2_mg->Write();
h2_t_p2_mg->Write();
h2_e_p1_p2->Write();
h2_t_p1_p2->Write();
h2_e_p1_mg->Write();
h2_t_p1_mg->Write();
h2_e_t_cm_1->Write();
h2_e_t_cm_2->Write();
h2_ecm_ps_1->Write();
h2_ecm_ps_2->Write();
h2_ecm_ps_1_i->Write();
h2_ecm_ps_2_i->Write();
h1_ecm_1->Write();
h1_ecm_2->Write();
h1_ecm_1_i->Write();
h1_ecm_2_i->Write();
```

```
//  theApp.Run();

}
```