UNIVERSITY OF IOANNINA
SCHOOL OF PHYSICAL SCIENCES
DEPARTMENT OF PHYSICS

Coupling the GEF code with GEANT4 Monte Carlo simulations towards the interpretation of the fission data from the n_TOF facility at CERN

M.Sc. Thesis

Loli Eleftheria

Academic supervisor
Patronis Nikolaos, Associate Professor, University of Ioannina

Ioannina, June 2020

ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΦΥΣΙΚΗΣ

Συνδυασμός του κώδικα GEF με προσομοιώσεις Monte-Carlo, μέσω του λογισμικού GEANT4, με στόχο την ερμηνεία των δεδομένων σχάσης του πειράματος n_TOF στο CERN

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ
Λώλη Ελευθερία

Ακαδημαϊκός επιβλέπων
Πατρώνης Νικόλαος, Αναπληρωτής καθηγητής, Πανεπιστήμιο Ιωαννίνων

Ιωάννινα, Ιούνιος 2020

Acknowledgments

Foremost, I would like to thank Assoc. Prof. Nikolas Patronis of the Physics Department of the University of Ioannina, who supervised the present thesis and gave me the opportunity to work in the field of Nuclear Physics. I am grateful for his support and motivation throughout this project. Without his guidance, encouragement and constructive feedback I would not be writing this thesis.

Moreover, I owe special thanks to my dear friend and Ph.D. candidate of the Physics Department of the University of Ioannina, Zinovia Eleme for her invaluable help. As the context of this project is related to the experiment she carried out in the framework of her Ph.D., she was the most experienced person in case I had a question or needed a bit of advice. I am deeply indebted to her for her support.

Finally, thanks to my friends and colleagues Efstathia Georgali Ph.D. candidate, Maria-Elisso Stamati and Maria Peoviti M.Sc. students of the Physics Department of the University of Ioannina for the wonderful moments we shared.

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω τον Αναπλ. Καθ. Νικόλαο Πατρώνη του τμήματος Φυσικής του Πανεπιστημίου Ιωαννίνων, που επέβλεψε αυτήν την εργασία και μου προσέφερε την ευκαιρία να δουλέψω στο κομμάτι της Πυρηνικής Φυσικής. Είμαι υπόχρεη για την υποστήριξη και την ενθάρρυνσή του κατά τη διάρκεια αυτού του ερευνητικού έργου. Χωρίς την καθοδήγηση και τα εποικοδομητικά του σχόλια, δεν θα έγραφα αυτή την εργασία.

Επιπλέον, οφείλω ένα ευχαριστώ στη φίλη μου και υποψήφια διδάκτορα του τμήματος Φυσικής του Πανεπιστημίου Ιωαννίνων, Ζηνοβία Ελεμέ για την πολύτιμη βοήθειά της. Καθώς το περιεχόμενο αυτής της εργασίας σχετίζεται με το πείραμα που πραγματοποίησε στα πλαίσια του διδακτορικού της, ήταν το πιο έμπειρο άτομο στην περίπτωση που είχα μια απορία ή χρειαζόμουν μια συμβουλή. Της είμαι βαθιά υπόχρεη για την υποστήριξη που μου προσέφερε.

Ολοκληρώνοντας, ευχαριστώ θερμά τους φίλους και συναδέλφους μου, Ευσταθία Γεωργαλή, υποψήφια διδάκτορα, Μαρία-Ελισσώ Σταμάτη και Μαρία Πεοβίτη, μεταπτυχιακές φοιτήτριες του τμήματος Φυσικής του Πανεπιστημίου Ιωαννίνων για τις υπέροχες στιγμές που μοιραστήκαμε.

Abstract

Towards the interpretation of the fission reaction data from Micromegas detectors, the pulse height spectrum should be fully characterized. In this thesis, extended simulations of Micromegas detectors were performed. To ensure the accuracy of the fission process in the Monte Carlo simulations, the fission process and fission fragments yield were calculated by means of the GEF code and the rest of the particle transportation was accomplished through the GEANT4 toolkit. The results of standalone GEANT4 simulations were compared with the ones from the combination of the GEF code with GEANT4. Additionally, the energy deposition of light and heavy fission products and the effect of the surface homogeneity and chemical composition of the targets on the energy deposition was examined. The simulation results with respect to the energy deposition of the fission products within the active gas volume of the detector were combined with the appropriate response function, in an attempt to reproduce the experimentally observed pulse height spectrum, with emphasis in the low energy region. This work is an important step for the accurate estimation of the needed correction factors that correspond to the adopted software thresholds set in general and in any n_TOF fission data analysis process.

Περίληψη

Ο πλήρης χαρακτηρισμός του φάσματος ύψους παλμών του ανιχνευτή Micromegas αποτελεί βασική προϋπόθεση για την ακριβή ερμηνεία των αντίστοιχων δεδομένων σχάσης. Σε αυτή την διατριβή πραγματοποιήθηκαν εκτενείς προσομοιώσεις τέτοιων ανιχνευτών. Τα θραύσματα σχάσης υπολογίστηκαν με τον κώδικα GEF και οι υπόλοιπες αλληλεπιδράσεις τους στον ενεργό όγκο του ανιχνευτή με το λογισμικό πακέτο GEANT4. Με αυτό τον τρόπο διασφαλίσθηκε η ακρίβεια των υπολογισμών παραγωγής των θραυσμάτων σχάσης στις προσομοιώσεις Monte-Carlo. Τα αποτελέσματα αυτόνομων προσομοιώσεων από το GEANT4 αξιολογήθηκαν συγκρίνοντάς τα με τα αντίστοιχα από τον συνδυασμό του κώδικα GEF με to GEANT4. Επιπλέον, διερευνήθηκε η εναπόθεση ενέργειας των ελαφριών και βαριών προϊόντων σχάσης και η επίδραση της ομοιογένειας της επιφάνειας και της χημικής σύστασης του στόχου στο ενεργειακό φάσμα. Ακολούθως, η καταγραφή της εναπόθεσης ενέργειας των προϊόντων σχάσης στον εναέριο όγκο του ανιχνευτή συνδυάστηκε με την κατάλληλη συνάρτηση απόκρισης (response function) προκειμένου να αναπαραχθεί όσο γίνεται καλύτερα το πειραματικό φάσμα ύψους παλμών -με έμφαση στην περιοχή χαμηλών ενεργειών. Η συγκεκριμένη έρευνα αποτελεί ένα σημαντικό βήμα για τον υπολογισμό των διορθωτικών παραγόντων σχετικά με την εισαγωγή κατωφλίων (software thresholds) όπως αυτά εφαρμόζονται σε κάθε διαδικασία ανάλυσης δεδομένων από τα πειράματα σχάσης της δραστηριότητας n_TOF στο CERN.

Contents

Introduction

The accurate knowledge of fission data, such as neutron-induced fission cross sections of various minor actinides, is important for the design of advanced nuclear systems as well as for the development of theoretical models of the fission process. At the n_TOF/CERN facility, fission related experiments are performed with Micromegas detectors, taking advantage of the wide energy range (from thermal to tens of MeV) and the high instantaneous flux of the neutron beam. For the interpretation of the fission reaction data obtained, the pulse height spectrum of the detector should be fully characterized.

Monte Carlo simulations can be used for a better understanding of the fission process and detector behavior. The present work adopts a coupling of two such methods. More specifically, the GEF code is used for the description of the fission process and the properties of the fission fragments and the GEANT4 for the rest of the particle transportation. Thus, the accuracy of the fission process in the Monte Carlo simulations is ensured.

During the simulation, the energy deposition of the detected particles inside the active gas volume of the detector is recorded. Thus, by varying some of the parameters that describe the experimental set up of the detector and the primary particles, the effect of those changes on the obtained energy spectrum can be examined.

The simulation results can be used within any data analysis framework to apply a response function that describes the behavior of the detector components, in an attempt to reproduce the experimentally obtained pulse height spectrum, with emphasis in the low energy region. This work can be later used for the accurate estimation of the needed correction factors that correspond to the adopted software thresholds set in general and in any n_TOF fission data analysis process.

This thesis is structured as follows. In Chapter 1 the necessary background is briefly described. That involves the fission process and the experimental set up in a fission experiment, i.e. the n_TOF facility and Micromegas detector. The simulation environment of GEF and GEANT4 codes is introduced. Additionally, the physics case of $^{241}Am(n,f)$ which is studied in the context of this thesis is presented. Chapter 2 includes the theoretical calculations with the GEF code and Chapter 3 consists of the Monte Carlo simulations. The detector geometry and primary particles are defined and the results of a GEANT4 standalone simulation are compared with the ones from the combination of GEF code with GEANT4. Moreover, the energy deposition of heavy and light fission products is studied and the effects of the target surface homogeneity and chemical composition on the energy deposition of the fission products are examined. Finally, the Micromegas detector response function is the subject of Chapter 4. An attempt to reproduce the experimentally observed pulse height spectrum with emphasis in the low energy region is demonstrated for different incident neutron energies.

Chapter 1

Background

1.1 Nuclear fission

Nuclear fission is a process in which a highly-deformed heavy nucleus is divided into two fragments of comparable masses with the release of energy. A concise description of the fission process is given below.

The Liquid Drop Model

Bohr and Wheeler (1939) [1] were the first to provide a theoretical description of the fission mechanism, based on the Liquid Drop Model (LDM). In the context of the LDM, the nucleus is described as an electrically charged liquid drop. It is represented by a spherical drop of incompressible nuclear liquid in which the nucleons interact with a limited number of their nearest neighbors. The analogy of the nuclear behavior to that of a charged liquid drop is reflected by the semi-empirical mass formula:

$$\begin{aligned} E &= E_v + E_s + E_c + E_a + \delta(A,Z) \\ &= a_v A - a_s A^{2/3} - a_c \frac{Z^2}{A^{1/3}} - a_a \frac{(N-Z)^2}{A} + \delta(A,Z) \end{aligned} \tag{1.1}$$

Since the nuclear radius is proportional to $A^{1/3}$, E_v expresses that the binding energy increases with increasing number of nucleons *(volume term)*, E_s that the nucleons at the surface interact with fewer other nucleons and are less bound *(surface term)*, E_c describes the repulsive electrostatic force between the protons *(Coulomb)*, E_a the tendency of nuclei to be symmetric, i.e. $(Z = N)$ for light nuclei, for heavy this term is reduced *(asymmetry term)* and

$$\delta(A,Z) = \begin{cases} -\delta_0 & Z, N \text{ even} \\ 0 & A \text{ odd} \\ +\delta_0 & Z, N\ odd \end{cases} \tag{1.2}$$

that the energy of a nucleus is lower when there is an equal number of protons (neutrons) with spin up and spin down *(pairing term)*.

The volume of the nucleus is considered to remain constant, thus if a spherical nuclear drop gets deformed only the surface E_s and Coulomb E_c terms will be affected. Bohr and Wheeler [1] described small deformations by expanding the radius r in Legendre series:

$$r(\theta) = R\left[1 + \sum_{n=1} a_n P_n(cos\theta)\right] \tag{1.3}$$

where R is the radius of the spherical nuclei. The coefficients a_1 and a_2 correspond to the quadrupole and octupole deformation, hence they describe an ellipsoid.

They proved that the sum of E_s and E_c terms E_{s+c} can be written as:

$$\begin{aligned} E_{s+c} &= E_s^0\left(1+\frac{2}{5}a_2^2\right)+E_c^0\left(1-\frac{1}{5}a_2^2\right) \\ &= E_{s+c}^0+\frac{2}{5}E_s^0\left(1-\frac{E_c^0}{2E_s^0}\right)a_2^2 \end{aligned} \tag{1.4}$$

where E_s^0, E_c^0, E_{s+c}^0 are the energies of a sphere. The term

$$\chi \equiv \frac{E_c^0}{2E_s^0} \cong \frac{1}{50}\frac{Z^2}{A} \tag{1.5}$$

Is denoted as the *fissility parameter* χ. It is characteristic of the nucleus and defines whether the binding energy of a distorted sphere will increase, decrease or remain constant.

If $\chi > 1$, then the *deformation energy* $\Delta E_{s+c} = E_{s+c} - E_{s+c}^0$ is positive ($\Delta E_{s+c} > 0$) and fission occurs spontaneously. If $\chi < 1$, the deformation energy will increase and then decrease leading to a single-humped potential barrier. For zero deformation, a local minimum in the deformation potential appears, hence the LDM predicts a spherical shape for nuclei in their ground state, which is not generally true.

The double-humped fission barrier

A single-humped barrier provides a macroscopic treatment of the process. A weakness of this model is that it ignores the nuclear structure. It further suggests a strong dependence of the barrier on Z^2/A in contradiction to experimental data and a spherical nuclei shape in the ground state [2].

Strutinsky [3] proposed a "macroscopic-microscopic" approach to take shell effects into account with the addition of a shell correction term δU to the binding energy of LDM (eq. 1.1) E_{LMD} . It was expanded to

$$E = E_{LMD} + \delta U \tag{1.6}$$

where

$$\delta U = U - \overline{U} \tag{1.7}$$

is the difference between the total energy of the nucleus calculated with a realistic shell model with distinct single-particle states and one with a uniform distribution of states. The addition of this term results in a double-humped fission barrier as displayed in Figure 1. States in the first well are classified as 'Class-I' and in the second one as 'Class-II'.

The theory of a double-humped potential provided an explanation for experimental observations, among which a non-spherical equilibrium shape, fission barrier heights, spontaneous fission isomers and cluster resonances. A detailed description is given by Bjornholm and Lynn [2].

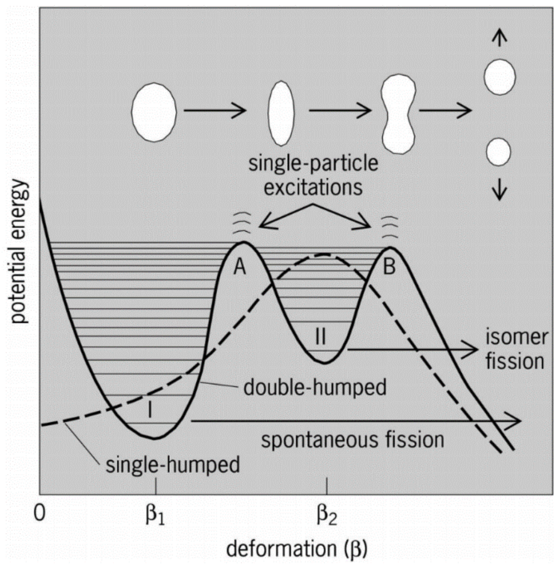


Figure 1.1: Schematic plots of a single-humped fission barrier of LDM and a double-humped introduced by shell corrections.[4].

Characteristics of fission

The fission process is the deformation of a nucleus from a ground-state shape to an elongated configuration that is divided into two *fission fragments* along with the emittance of neutrons and energy.

If the fragments have equal mass, the fission is called mass-symmetric; otherwise it is referred to as mass-asymmetric. The presence of two independent deformation paths has been proposed to interpret the preference for asymmetric mass-division in low energy fission. The asymmetric mode which has a lower energy threshold and the symmetric mode with a higher one. A detailed description is given in [5].

At the instance of fission, the fragments shred their energy excess through the emission of *prompt neutrons* $(\sim 2{,}3)$ (within $10^{-16}s$) and *prompt* γ *rays* (within $10^{-14}s$) resulting in the *fission products*. *Delayed neutrons* are often emitted after the β decay of the products. At a fission event, the large amount of energy $(\sim 200\ MeV)$ appears mainly as fission fragments kinetic energy $(\sim 80\%)$, while the rest appears as *prompt* γ *rays* $(\sim 8\ MeV)$, β decays $(\sim 19\ MeV)$ and γ decays $(\sim 7\ MeV)$ from the fragments (Figure 1.2)[6].

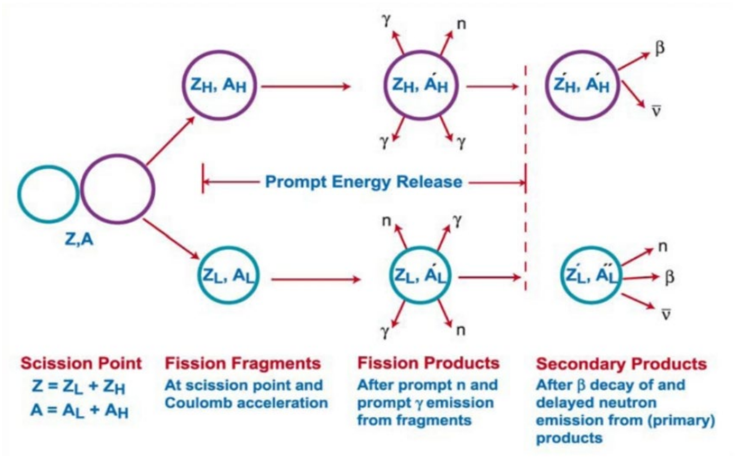

Z,A
ZH, AH
ZL, AL
γ
n
β
ν̄
Prompt Energy Release
Scission Point
Z = ZL + ZH
A = AL + AH
Fission Fragments
At scission point and Coulomb acceleration
Fission Products
After prompt n and prompt γ emission from fragments
Secondary Products
After β decay of and delayed neutron emission from (primary) products

Figure 1.2: Schematic of post scission in fission [7]

Neutron-induced fission cross-section

In the context of the present thesis, simulations of the reactions $^{241}Am(n,f)$ and $^{235}U(n,f)$ were performed. Following the theoretical description of the fission process, the characteristics of the cross-sections of these reactions can be explained. The $^{238}U(n,f)$ reaction is studied as well.

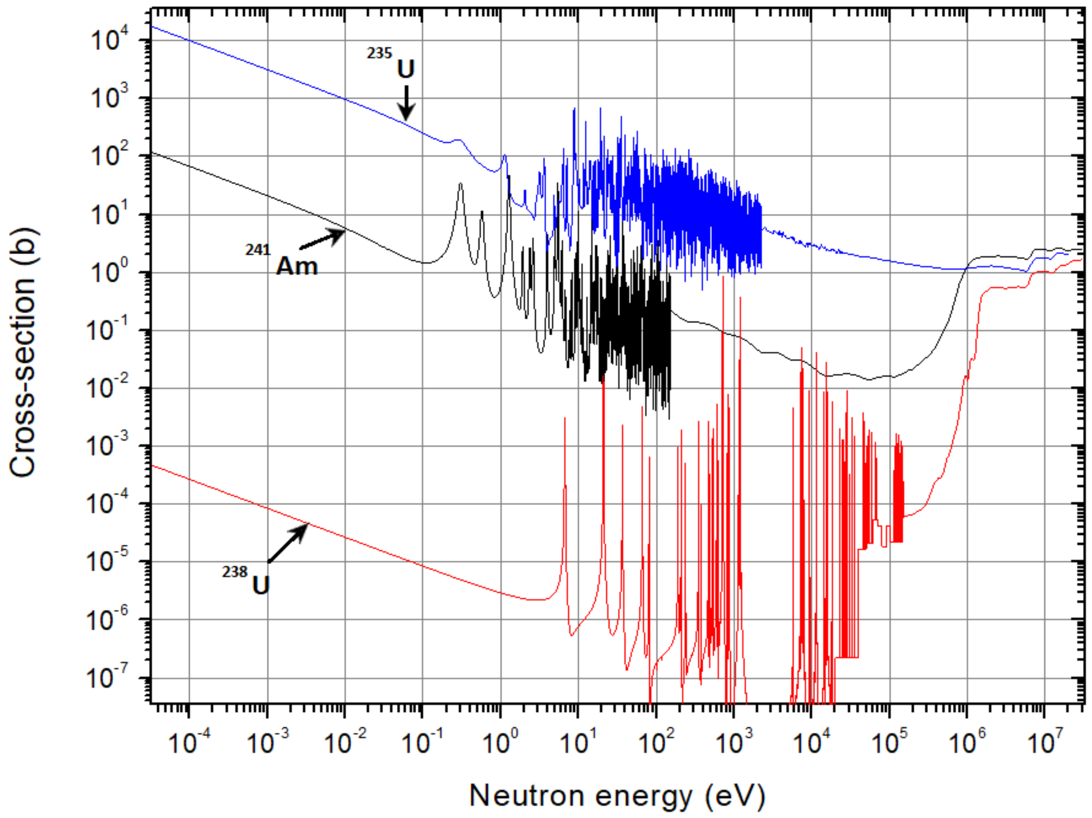


Figure 1.3: The cross-sections of the reactions $^{241}Am(n,f)$ $^{235}U(n,f)$ $^{238}U(n,f)$

For low incident neutron energies (thermal up to 1 eV), the probability of a reaction is proportional to the time the neutron spends inside the nucleus. Therefore, the cross-section decreases with increasing velocity and energy ($^1/_u$ law). The difference in the magnitude of the cross-sections is a result of the pairing effect. The fission barrier is approximately the same for the nuclei $\sim 6-7\ MeV$. The *neutron separation energy* (the energy increase of the nucleus after the absorption of a neutron) depends on the configuration of the compound nucleus. If it is even-N, the excitation energy is higher than the barrier even with thermal neutrons. On the contrary, if it is an odd-N incident neutron energy greater than $\sim 1\ MeV$ is required. *Resonances* appear in the 1eV-1keV region, due to excited states of the compound nucleus. In the MeV region, where the neutron carries enough energy so that the nucleus can overcome the fission barrier (f*ission threshold*), the cross-sections are similar for all nuclei. The steps at higher energies correspond to *multi-chance fission*. The nucleus fissions after the emission of pre-equilibrium neutrons, (n,xnf) channel.

1.2 The n_TOF facility at CERN

The n_TOF facility at CERN is a time-of-flight facility based on a spallation neutron source [8,9]. It offers two beam lines: A horizontal one with high resolution and moderate flux with a flight path of $185\ m$ and a vertical one with high flux and moderate resolution with a $20\ m$ flight path, which lead to Experimental Area 1 (EAR1) and Experimental Area 2 (EAR2), respectively. Neutrons are produced via spallation when $20\ GeV/c$ proton bunches from CERN's Proton Synchrotron (PS) impinge on a $40\ cm$ in length and $60\ cm$ in diameter lead spallation target. The proton bunch intensity varies from $\sim 3 \times 10^{12}$(parasitic bunches) up to $\sim 8 \times 10^{12}$(dedicated punches) with a nominal of $\sim 7 \times 10^{12}$ protons. The small repetition rate which does not exceed $0.8\ Hz$ (1.2 s between consecutive bunches) and the small width of the proton bunch (7 ns rms in dedicated and 20 ns in parasitic mode) allows for well-separated neutron bunches delivered in both experimental areas. The high instantaneous flux, good energy resolution and background suppression allow the measurement of low cross-sections, highly radioactive samples and isotopes available in small masses.

The spallation target is surrounded by a $1\ cm$ thick circulating layer of cooling water and an additional $4\ cm$ thick circulating layer of borated water in the horizontal direction for moderation reasons. The neutron spectrum at n_TOF covers a wide neutron energy range from the thermal region up to several tens of MeV [11]. A graphical representation of the facility that includes both experimental areas is shown in Figure 1.4.

The measurement of the reactions studied in the present thesis was performed at EAR2, thus a concise description will be given for this experimental hall [11]. A dipole magnet is in place to deflect the charged particles traveling with the neutrons inside the vacuum tube, thus preventing them from reaching the experimental hall. Two collimators are installed in the beam line: the first has an inner diameter of $20\ cm$. The second one shapes the neutron beam and two different configurations are available. For capture cross-section measurement, where a narrow well-defined beam is required, the inner diameter is $2.2\ cm$ and for fission measurements which employ very thin and wider targets, is $6\ cm$. The higher flux and shorter times of flights involved in EAR2 offer a stronger suppression of sample-induced background.

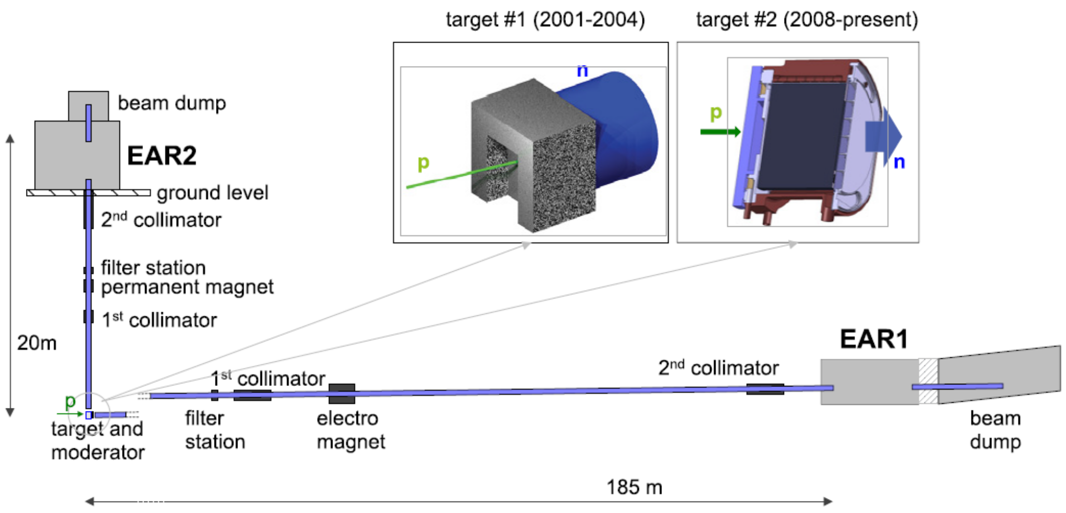


Figure 1.4: Graphical representation of the n_TOF facility with its two neutron beam lines ending in the experimental areas EAR1 and EAR2. Both spallation targets are also shown. Target #1 was used during Phase 1 of the n_TOF experiment, while target #2 was used during the Phases 2 and 3 (2008-2018). [11]

1.3 The Micromegas detector

The *Micromegas* detector (*MICRO Mesh Gaseous Structure*) [12, 13, 14] belongs to the Micro-pattern gaseous detectors (*MPGDs*) [15, 16]. MPGSs with their excellent spatial and time resolution, high granularity and rate capabilities have been adopted by many experiments handling high count rates.

For neutron measurements it is important to minimize the amount of material present in the beam to reduce the background. Thus, the *microbulk* [17] design was developed which is used in n_TOF experiments.

The Micromegas is an asymmetric stage parallel plate detector; its basic operation principle is illustrated in Figure 1.5. The space between the cathode and the anode is separated by a micromesh into two regions: the drift or conversion region and the amplification region. The micromesh is a thin $5\ \mu m$ copper conductive layer with $35\ \mu m$ diameter holes and a pitch of $50\ \mu m$.

The primary particle ionizes the gas in the drift region ($\sim 7\ mm$) creating electron-ion pairs. An electric field of ($\sim 5\ k/V$) is applied which causes the electrons to drift to the mesh, but is not strong enough to allow the creation of avalanches. In the amplification region ($\sim 50\ \mu m$), the electric field is stronger ($\sim 50\ k/V$), thus avalanches are formed improving the signal-to-noise ratio of the detector. The charge is finally collected in the anode and the induced signal is readout.

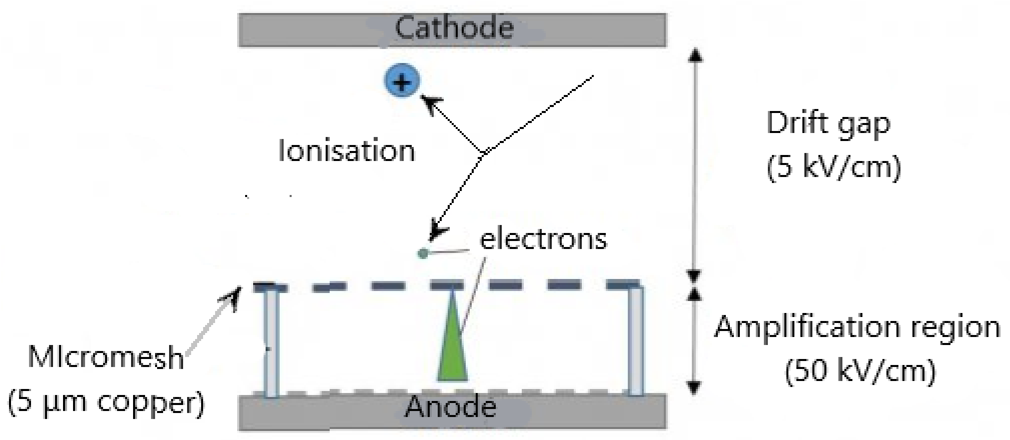


Figure 1.5: Basic operation principle of Micromegas detector. A particle emitted from a source ionizes the atoms of the gas creating positive ions and electrons. As the electrons travel towards the anode the pass through the mesh and are amplified before reaching the readout strips.

The electric field in both the drift and amplification gap is homogeneous. It only exhibits a funnel-like shape close to the micromesh holes as shown in Figure 1.6. The field lines are compressed to the openings into a small diameter, depending on the electric field ratio between the two gaps. This ratio is tuned to ensure electron transparency and quick collection of ions created in the amplification region to the micromesh.

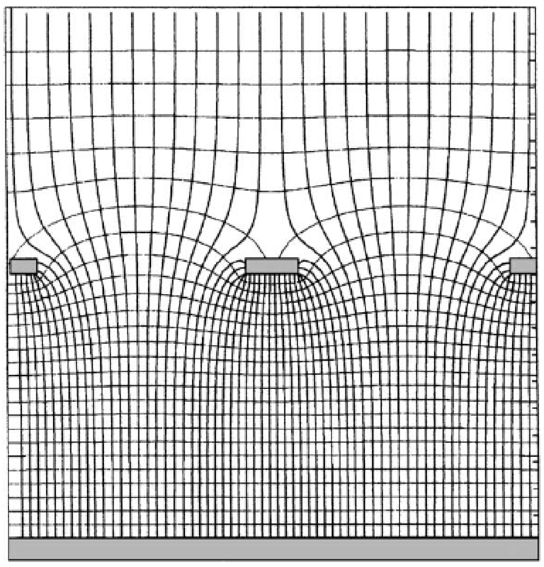


Figure 1.6: Electric field of Micromegas. The electrons in the drift region are carried towards the center of the holes and the positive ions created in the amplification region are captured at the mesh [15].

The detector operates with an $Ar: CF_4: isoC_4H_{10}$ $(88:10:2)$ gas mixture at atmospheric pressure. The use of CF_4 mixture increases the electron drift velocity and decreases the diffusion coefficient thus improving the energy resolution.

In an n_Tof experiment, a stainless steel chamber is constructed to hold the samples and Micromegas detectors. The chamber is placed along the beam line and its axis is aligned to the neutron beam. The entrance and exit windows are made of Kapton and air-tight connectors are installed to apply the high-voltages and read the output signals.

1.4 The GEF code

The GEneral Fission (GEF) code is a semi-empirical model for the description of fission observables [18, 19]. It combines general laws of physics and mathematics with empirical information and provides results for a wide range of fissioning nuclei from polonium to seaborgium up to excitation energies of 100 MeV including multi-chance fission. Because the model is based on a firm theoretical frame and preserves the link between different fission observables, GEF also gives reliable results in cases no experimental information is available, contrary to most of the existing fission models. It is a Monte Carlo code and for each fission event it calculates all the properties of the two fission fragments at scission: mass, charge, excitation energy and angular momentum, as well as the fission fragments kinetic energies. In addition, GEF treats the deexcitation of the fission fragments and provides the prompt-neutron and prompt-gamma multiplicities associated with each fragment, as well as the prompt-neutron and prompt-gamma kinetic energies and angles. All this information can be delivered by the code on an event-by-event basis and can serve as an event generator for simulation purposes. A graphical representation of GEF is shown in Figure 1.7.

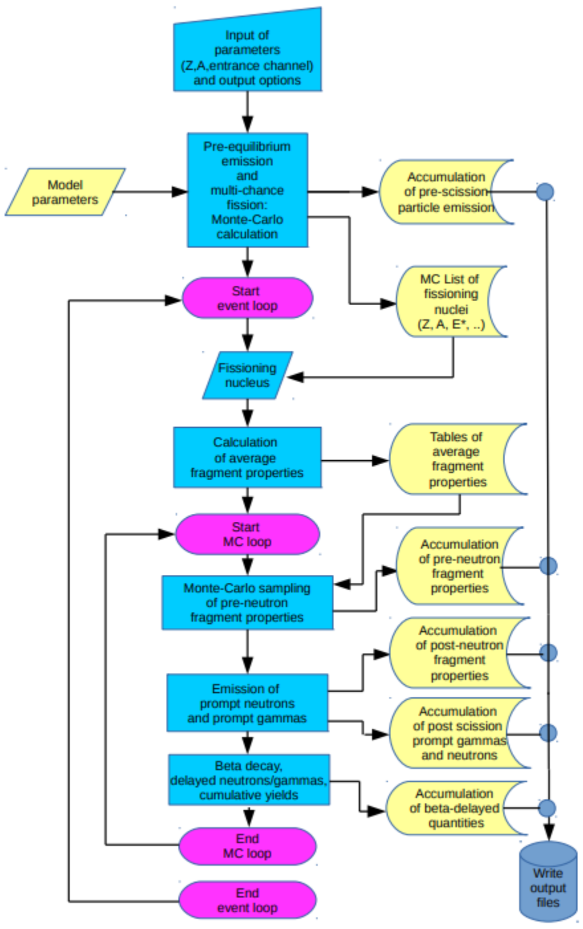

Input of parameters (Z,A,entrance channel) and output options
Model parameters
Pre-equilibrium emission and multi-chance fission: Monte-Carlo calculation
Accumulation of pre-scission particle emission
MC List of fissioning nuclei (Z, A, E*, ..)
Start event loop
Fissioning nucleus
Calculation of average fragment properties
Tables of average fragment properties
Start MC loop
Monte-Carlo sampling of pre-neutron fragment properties
Accumulation of pre-neutron fragment properties
Accumulation of post-neutron fragment properties
Emission of prompt neutrons and prompt gammas
Accumulation of post scission prompt gammas and neutrons
Beta decay, delayed neutrons/gammas, cumulative yields
Accumulation of beta-delayed quantities
End MC loop
End event loop
Write output files

Figure 1.7: A graphical representation of GEF model [20].

Physics behind GEF

GEF combines general laws of quantum and statistical mechanics with specific experimental information. A complete description of the code can be found in [18]. The main ideas used to derive the shape of the fission-fragments yields and the partition of the intrinsic excitation energy between the fragments are discussed below.

Fission-fragment yields

The fission fragment yields are determined by the potential energy landscape between the fission barrier and scission as a function of the mass-asymmetry degree of freedom. The microscopic-macroscopic approach is adopted, according to which, the shape of the potential energy on the way to scission is given by the combination of the macroscopic potential, as given by the liquid-drop model, and shell effects. The two-center shell model is used to study the single-particle structure in a di-nuclear potential with a necked-in shape. Mosel and Schmitt [21] revealed that the single-particle structure in the area of the outer fission barrier already very much resembles the sequence of single-particle levels in the two separated fragments after scission. As a result, the microscopic properties of the fissioning system are essentially determined by the shells of the fragments, and only the macroscopic properties are specific to the fissioning system [22]. This “separability principle” is used in the GEF code, which relies on empirical information to determine the stiffness of the macroscopic potential and the position and strength of the fragment shells. The latter are valid for all fissioning systems, which explains why the GEF code is able to give results for a very large number of fissioning nuclei with one set of parameters. The magnitudes of the shell effects that form the fission valleys and the stiffness parameters in mass-asymmetric distortions are determined by a global fit of measured mass distributions.

Regarding the shell effects, asymmetric fission is related to the influence of a deformed ($\beta \approx 0.6$) fragment shell at $N = 88$, the spherical shells at $N = 82$ and $Z = 50$ [23] and with an empirical origin the shell at $Z = 54$. The transition from symmetric to asymmetric fission for heavy nuclei can be explained by the interplay between the macroscopic potential and shell effects, as seen in Figure 1.8. The strength of the shell and its position is fixed. However, the stiffness and the position of the minimum of the macroscopic potential increase with the mass of the fissioning nucleus. As a result, the minimum of the total potential moves from symmetric to asymmetric splits.

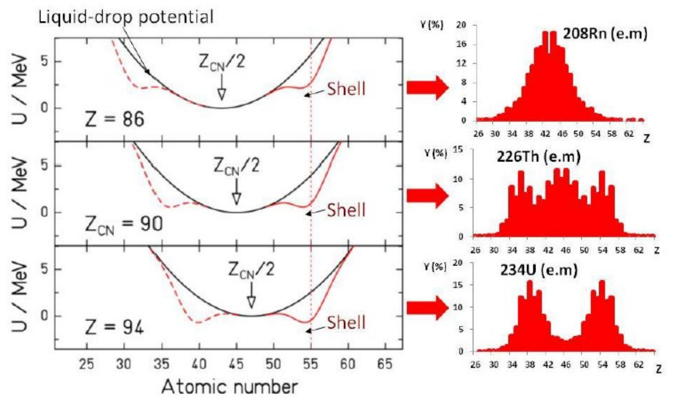


Figure 1.8: (Left) Total potential energy (red lines) and macroscopic potential lines (black lines) as a function of the fragment charge for fissioning nuclei around Th. The minimum of the macroscopic potential located at symmetry is indicated and a shell at $Z = 55$ is assumed. (Right) Experimental charge distributions obtained in electromagnetic-induced fission. [19].

Partition of excitation energy between the fragments

The manifestation of fragment shells on the fission path, as mentioned above, suggests that the fragments acquire their individual characteristics in the area of the fission barrier. Therefore, at this position, the fissioning nucleus consists of two well-defined nuclei in contact through the neck. Before scission, the available intrinsic excitation energy E^*_{intr} (given by the sum of the excitation energy above the barrier and the dissipated energy after the saddle) is divided between the fragments. In GEF, the excitation-energy partition is determined according to statistical mechanics. It is assumed that the system formed by the two nuclei in contact reaches a statistical equilibrium where all the configurations that are energetically possible have the same probability to be populated. The nuclear temperature by the influence of pairing correlation is assumed constant. Thus, the fissioning nucleus represents a system of two coupled microscopic thermostats with a limited total energy. The most probable configurations are when the available excitation energy concentrates in the heavy fragment [24].

1.5 The GEANT4 simulation toolkit

GEANT4 is a Monte Carlo simulation toolkit, modeling the interaction of particles with matter [25-27]. It is based on object-oriented technology and is implemented in C++ programming language. It used from High Energy Physics, to medical physics and space science.

GEANT4 offers comprehensive detector and physics modelling capabilities. It is possible to model the experimental set-up in terms of geometry and materials and to define the particles involved and their physics interactions. The user can track particles in matter but also in the presence of electromagnetic fields and inspect the response of the detector. Furthermore, interfaces are provided which enable users to interact with their simulation application and store their results in analysis objects (histograms, trees, etc.). Visualization drivers and graphical user interfaces are included in the toolkit.

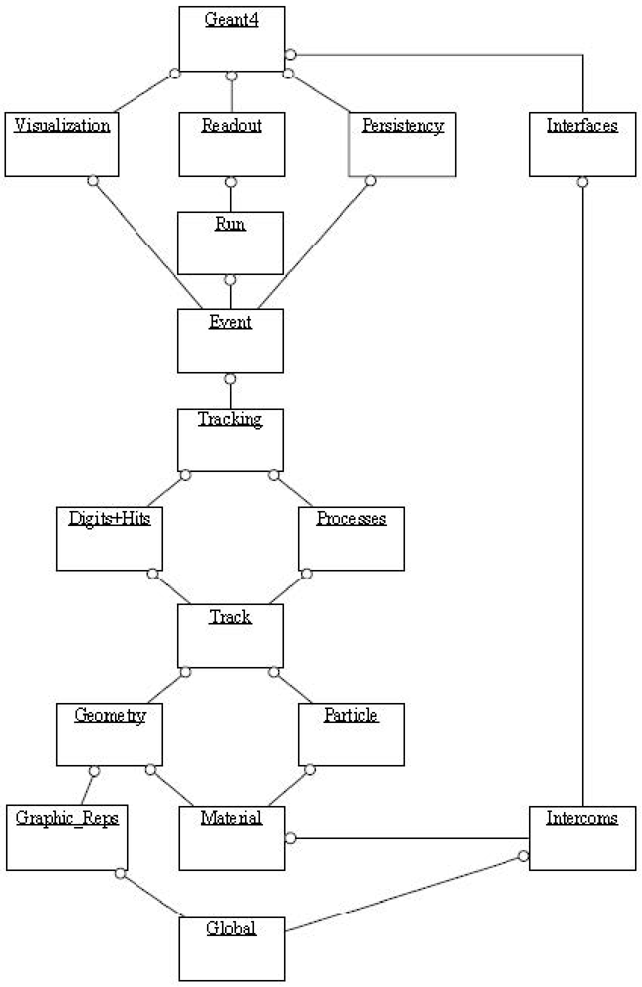


Figure 1.9: The top-level category diagram of the GEANY4 toolkit. The open circle on the joining lines represents a using relationship; the category at the circle end uses the adjoined category [26].

The top-level categories and how each category depends on the others are illustrated in Figure 1.9. The categories (packages) at the bottom of the diagram are used by higher categories and provide the foundation of the toolkit. These include the package *global* (covering the system of units, constants, numerics and random number handling), *materials, particles, graphical representations, geometry* (including the volumes for detector description and the particle navigation in the geometry model) and *intercoms*, which provide both a means of interacting with GEANT4 through the user interface and a way of communicating between models that should not otherwise depend on one another.

Above these are packages required to describe the tracking of particles and the physical processes they undergo. The *track* package contains classes for tracks and steps, used by processes, which contain implementations of models of physical interactions. Furthermore, *transportation* handles the transport of particles in the geometry model. All these processes may be invoked by the *tracking* package, which manages their contribution to the evolution of a track's state and provides information from hits in the sensitive volumes (i.e. energy deposition, generation of secondary particles, etc.) and digitization (model of the detector response).

Over these the *event* package manages events and *run* manages collections of events that share a common experimental configuration. A readout package allows handling the description of the electronics and the *readout* associated with the experimental setup.

1.6 The physics case of $^{241}Am(n,f)$

In the context of the present thesis, simulations of Micromegas detectors used for the study of $^{241}Am(n,f)$ reaction were performed. Therefore, a concise description of the importance of this reaction as well as the experimental setup and data acquisition of a ^{241}Am cross-section measurement at n_TOF are presented.

The nuclear power production through ADS (Accelerator Driven Systems) [28] as well as Generation-IV [29] fast neutron reactors is considered as one of the possible solutions that could address the issue of climate change – given the nearly zero CO_2 emission rate of nuclear energy compared to other methods of energy production [30]. The same systems also provide the ability of incarnation/transmutation of the existing actinides in high-level nuclear waste from spent PWR UOx fuel [31] which is an important aspect, as well. The safe design and operation of such nuclear power systems require high-accuracy cross-section data for a variety of neutron-induced reactions from thermal energies to several tens of MeV. One of the most important fissionable isotopes to be considered as a potential candidate for incineration/transmutation is $^{241}Am\ (T_{1/2} = 433\ y)$ which represents about 1.8% of the actinide mass in spent fuel. Also taking into account the production rate of ^{241}Am within spent fuels coming from the β^- decay of $^{241}Pu\ (T_{1/2} = 14.3\ y)$ its importance becomes even higher. Thus, accurate information of the fission reaction rate of ^{241}Am for an extended energy region is needed and this reaction is included in the Nuclear Energy Agency (NEA) "High Priority Request" (HPRL) [32] since target accuracies are not yet met by the existing data. The importance of the measurement is also highlighted in the OECD/NEA WPEC Subgroup 26 Final Report [33] that summarizes the needs and target accuracies for nuclear data.

In view of all the above, the $^{241}Am(n,f)$ reaction has been studied at the first experimental area (EAR1) and in the Experimental Area 2 (EAR2) of the n_TOF facility at CERN. The adopted experimental setup and data acquisition of a $^{241}Am(n,f)$ measurement at EAR2 are discussed below. More information can be found in [34].

Samples

Six samples of ^{241}Am (98% purity) were used with a total mass of $0.78\,mg$ ($\sim 4.6\,\mu g/cm^2$ per sample) and activity of $\sim 0.1\,GBq$. Additionally, for the determination of the neutron flux two ^{235}U ($0.26\,mg, 0.30\,mg$) samples with $\sim 9.9\,\mu g/cm^2$ per sample and two ^{235}U ($2.07\,mg, 2.21\,mg$) samples with $\sim 75.7\,\mu g/cm^2$ per sample, were used as reference foils. The sample material was in all cases electroplated in a surface $6\,cm$ on top of a thick $0.025\,mm$ aluminum backing.

Detectors

For the measurement, an array of ten in total Micromegas detectors was used, six for the americium and four for the uranium reference samples (Figure 1.10). The sample-detector modules were housed in a cylindrical aluminum fission chamber filled with a gas mixture of Ar:CF_4:isoC_4H_{10} (88:10:2) at atmospheric pressure. During the measurement, the gas flow and pressure were monitored and controlled with a dedicated flow regulation system. This way, the gas pressure was kept constant ensuring stable gain conditions during the data-taking period.

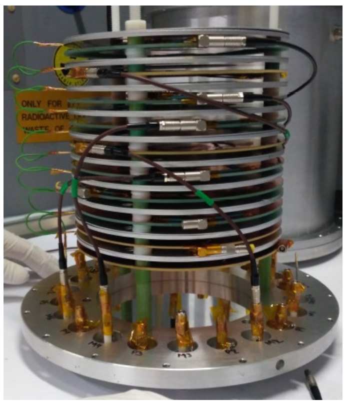


Figure 1.10: The stack of detectors and samples [18].

Data acquisition

The detector signals were stored as waveforms and digitized afterwards. The raw data for each detector were recorded using a *digital acquisition system* [35] in flash Analog-to-Digital Converters (ADC's) with a $16\,ms$ time acquisition window.

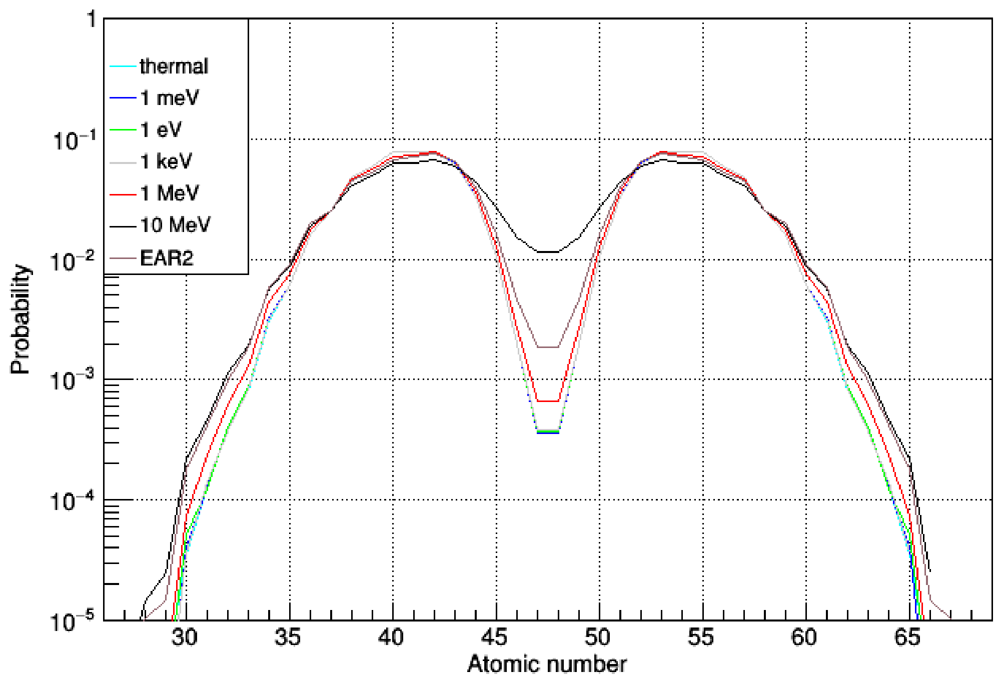
Chapter 2

Theoretical calculations using GEF

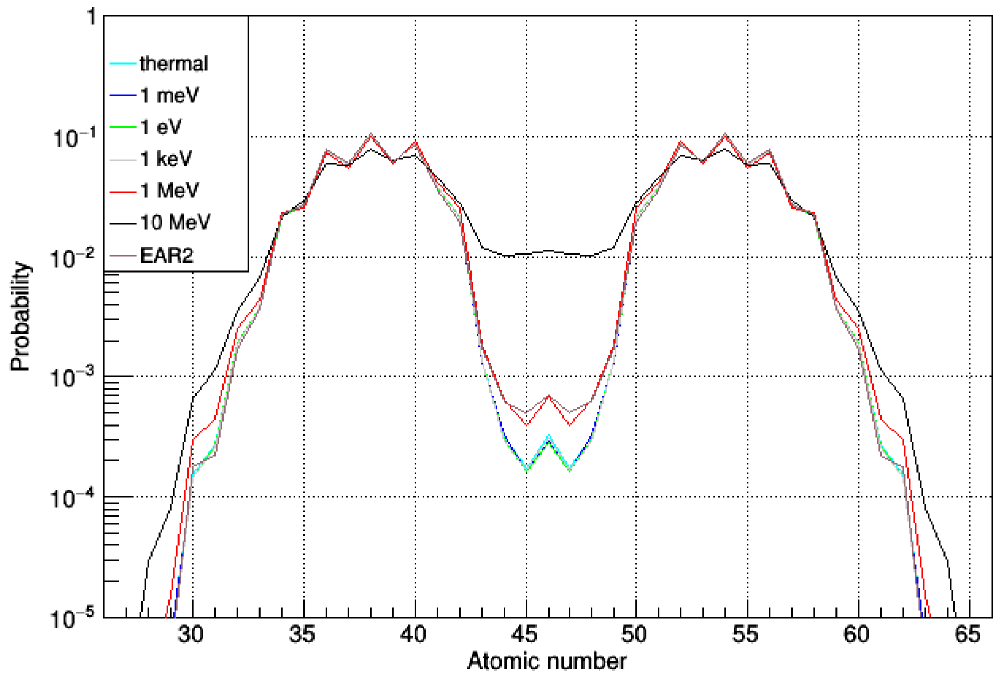
In the Monte Carlo simulations the fission process and the fragment yields were calculated with the GEF code and the rest of the particle transportation was handled by the GEANT4 toolkit. In this chapter the theoretical GEF calculations are described.

2.1 Dependence on the incident neutron energy

The dependence of the fragment yields on the incident neutron energy was examined. The mass and element distributions were obtained for neutron beams of thermal, 1 meV, 1 eV, 1 keV, 1 MeV, 10 MeV energy as well as the EAR2 flux up to 15 MeV. In Figures 2.1 and 2.2 the mass and element probability distributions are shown for the fissile nuclei ^{242}Am and ^{236}U, respectively. For low incident energies, fragments of different mass are produced and the fission is asymmetric, while it becomes more symmetric with increasing energy. This observation is in agreement with the theory of the presence of two independent deformation paths that was introduced in Chapter 1.

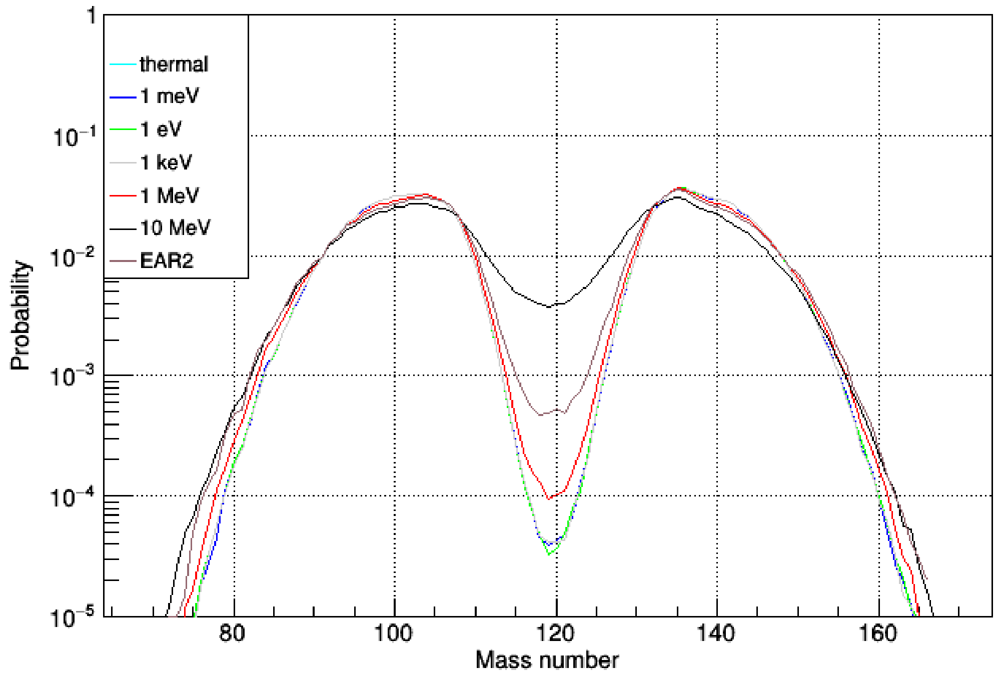


(a)

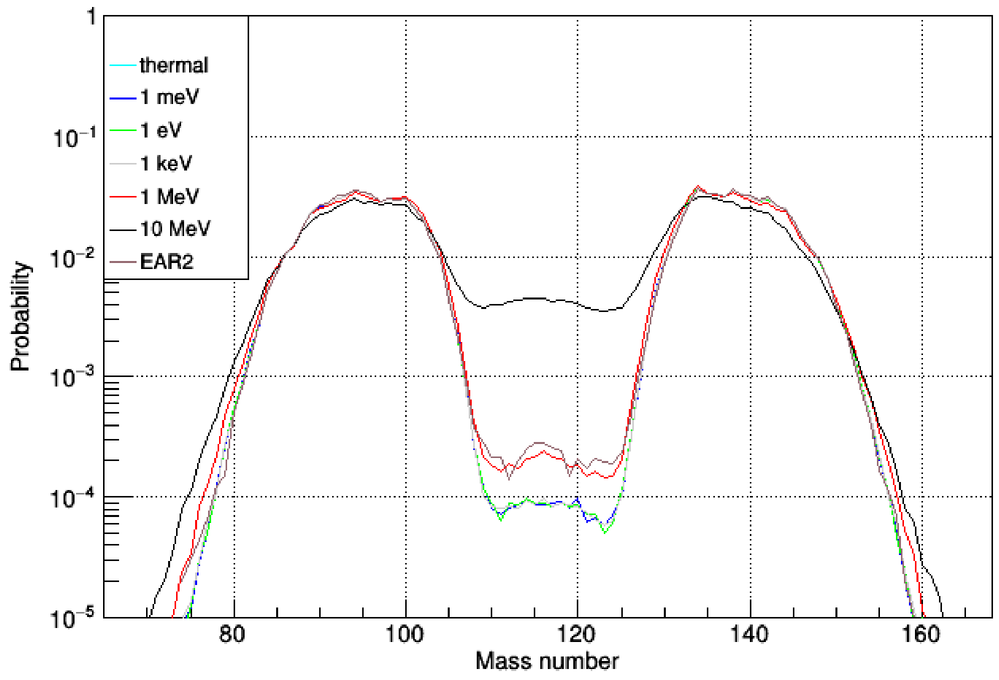


(b)

Figure 2.1: Element distribution of (a) ^{242}Am and (b) ^{236}U fission products for different incident neutron energies as obtained with the GEF code. For low energies the fission is asymmetric and it becomes more symmetric with increasing energy.



(a)



(b)

Figure 2.2: Mass distribution of (a) $^{242}\,Am$ and (b) $^{236}\,U$ fission products for different incident neutron energies as obtained with the GEF code. For low energies the fission is asymmetric and it becomes more symmetric with increasing energy.

2.2 Distributions of fission products

The distribution of fission product properties such as the atomic and mass number (Z, A respectively), as well as the kinetic energy (K) post neutron evaporation can be obtained with the GEF code. The element and mass distributions of $^{242}\,Am$ and $^{236}\,U$ fission fragments induced by a thermal neutron beam are displayed in Figures 2.3 and 2.4 respectively. The mass of the heavy fragments is observed to be independent of the compound nucleus and close to 140. On the contrary, for the light fragments, there is dispersion and the heavier the nucleus, the heavier would be the light fragments. This can be explained by shell structure effects and more specifically the presence of the doubly magic $^{132}\,Sn\ (Z = 50, N = 82)$ that dominates the mass split. As seen in Figure 2.3 there is a preference for the fissioning system to form a nucleus with $Z \cong 50$ and the remaining protons are given to the light fragment. In Figure 2.5 the mass number of fission fragments against their kinetic energy, obtained by GEF, is shown.

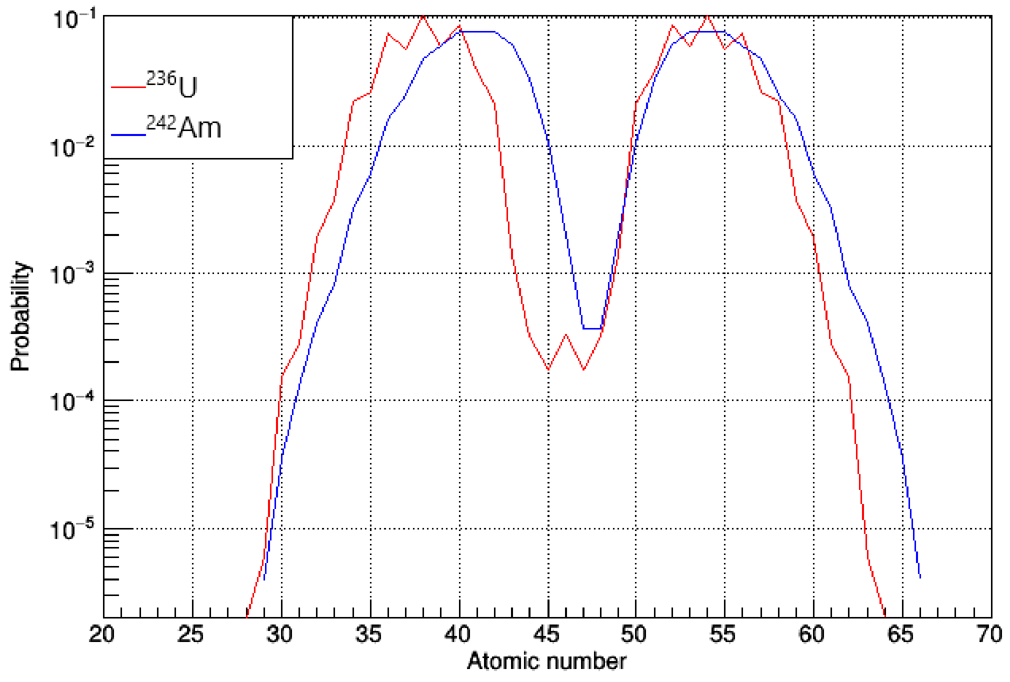


Figure 2.3 Atomic number distributions of ^{242}Am and ^{236}U fission products for thermal incident neutrons calculated with the GEF code. It is observed that the heavy fragments have the same atomic number independently of the compound nucleus, whereas the light fragments grow heavier for heavier compound nuclei.

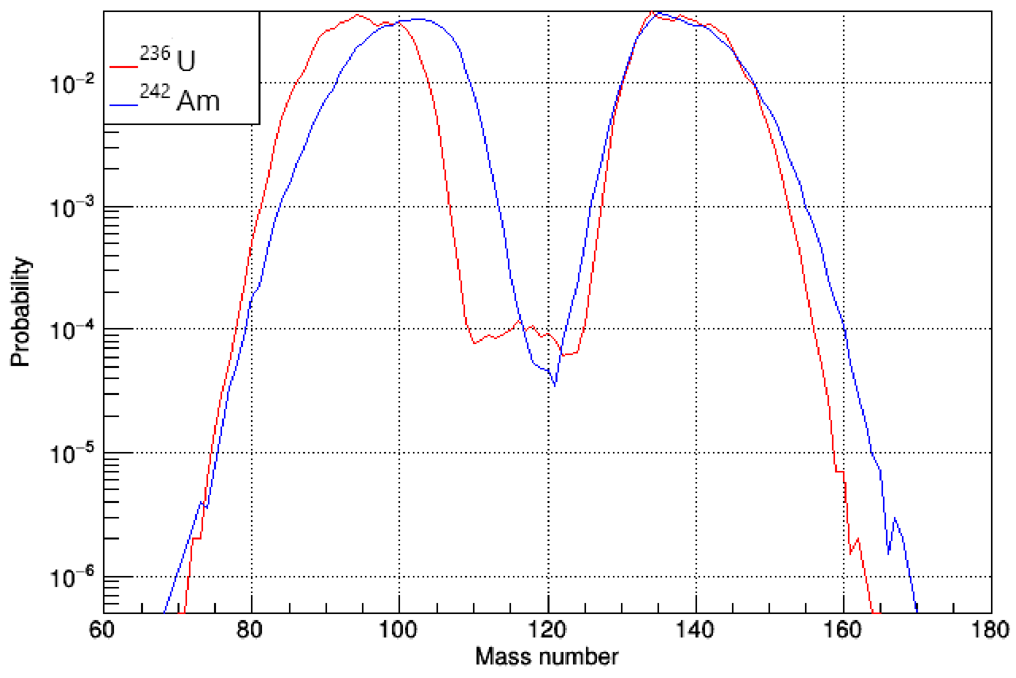
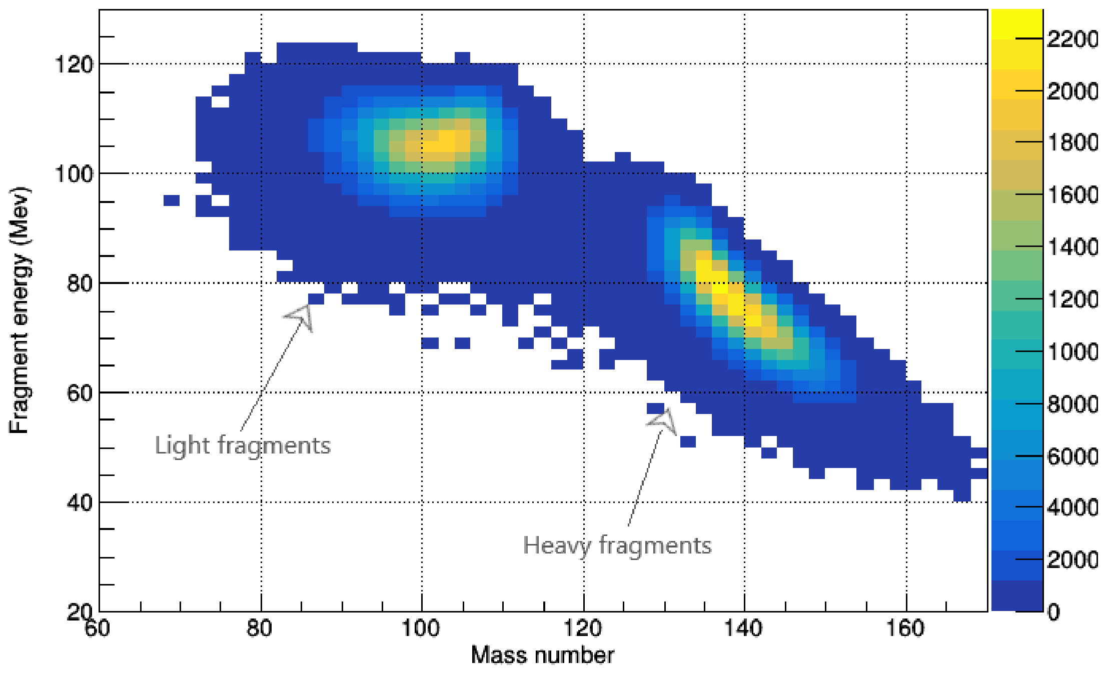
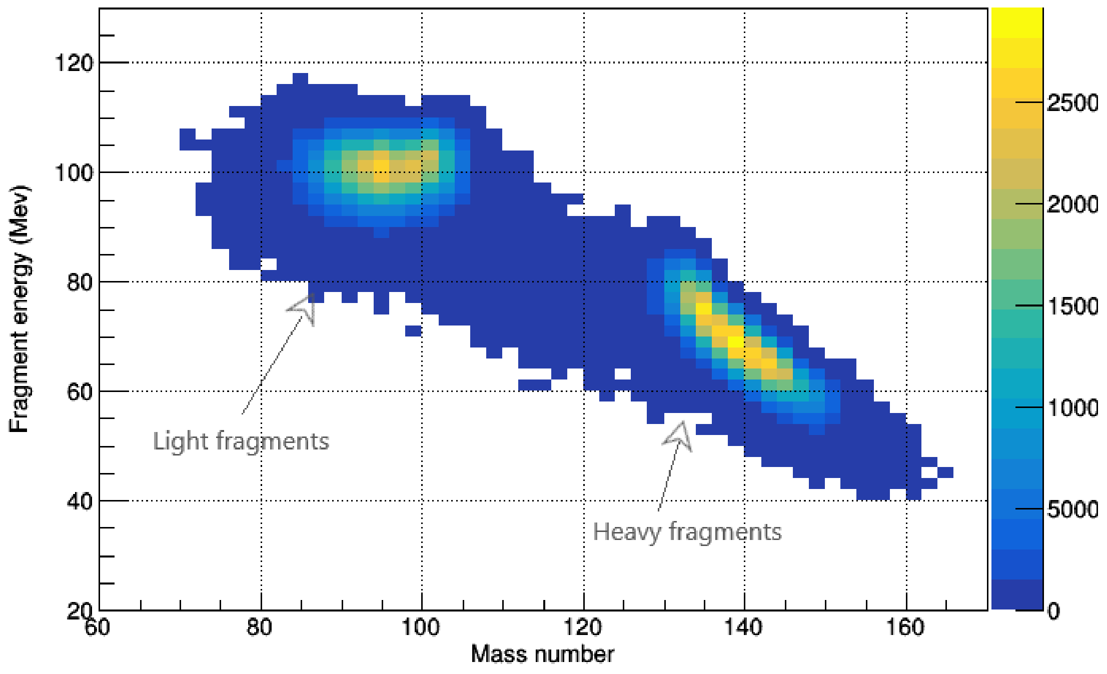


Figure 2.4: Mass distributions of ^{242}Am and ^{236}U fission products for thermal incident neutrons, as calculated with the GEF code. It is observed that heavy fragments have the same atomic number independently the compound nucleus, whereas the heavier the nucleus is, the heavier are the light fragments are.



(a)



(b)

Figure 2.5: Mass number of fission products against their kinetic energy, as obtained from GEF, for (a) ^{242}Am and (b) ^{236}U fission products for thermal incident neutrons calculated with the GEF code.

Chapter 3

Monte Carlo simulations

Following the theoretical calculations with the GEF code and the study of the fission product properties, Monte Carlo simulations of Micromegas detectors used for the study of $^{241}Am(n,f)$ were performed, by combining the GEF code with the GEANT4 toolkit. More specifically, the fission product properties were calculated by means of GEF and were used as primaries to GEANT4, where the transportation of these ions was accomplished. The purpose of these simulations was to better understand the behavior of the detector and examine the influence of various parameters on the obtained energy spectrum.

3.1 Detector geometry and primary particle source

The geometry consisted of a Micromegas detector and a sample deposit, either ^{241}Am or ^{235}U, housed in a chamber filled with gas, as illustrated in Figure 3.1 and Figure 3.2. More specifically, a sample of ^{241}Am with mass $148.9\,\mu g$ ($5.27\ \mu g/cm^2$) or ^{235}U with mass $297.7\,\mu g$ ($10.53\ \mu g/cm^2$) with a $6\,cm$ diameter was placed on top of a $0.025\,mm$ thick aluminum backing. The thickness of each sample was calculated to match the surface density and total mass of the ones used in an ^{241}Am fission cross-section measurement [34]. The sample-backing configuration was housed in the center of a cylindrical aluminum chamber with dimensions $30\,cm \times 30\,cm\ (diameter \times length)$ filled with a gas mixture of Ar:CF$_4$:isoC$_4$H$_{10}$ (88:10:2) at atmospheric pressure. Kapton windows $15\,cm$ in diameter were placed at the entrance and exit of the chamber. Since the region of importance for the energy deposition is the drift region, the amplification region was not included in the simulations. Thus, the active gas volume was simulated to be $7\,mm$ in length and a $9.5\,cm$ in diameter.

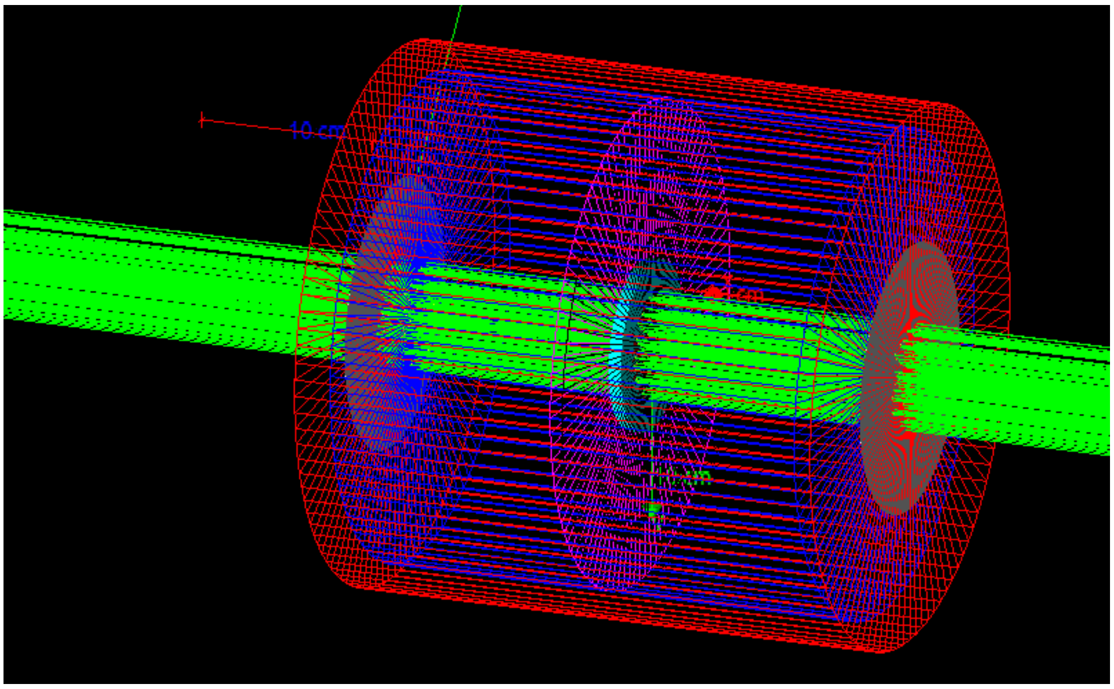


Figure 3.1: The geometry of the Micromegas detector and the sample inside the fission chamber. An incident neutron beam is also included in the visualization.

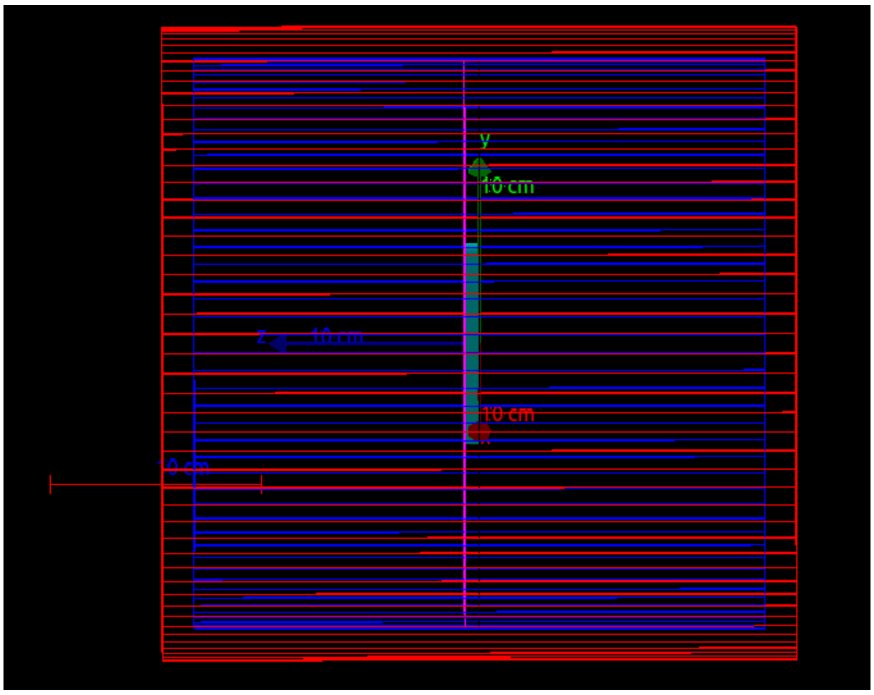
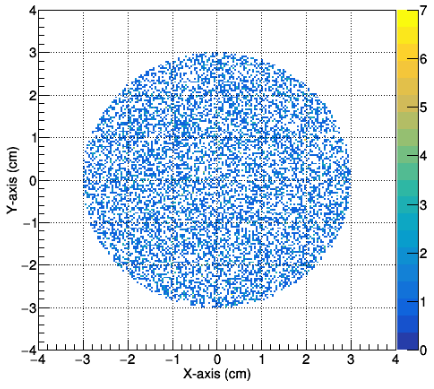


Figure 3.2: The geometry of the Micromegas detector and the sample inside the fission chamber used in the simulations.

For the definition of the primary particles, a dedicated procedure was adopted. For each fissile nuclei ^{242}Am or ^{236}U and incident neutron energy, the properties of the fission fragments after neutron emission (Z, A) and their total kinetic energy (K) were calculated by GEF. Then the heavy or light fragment was randomly chosen. It was given starting coordinates and a momentum direction that ensured uniform distribution inside the sample and uniform 2π emission towards the gas volume, as seen in Figure 3.3 and Figure 3.4. During the simulation, the properties of the selected fragments $(Z, A, K,)$ and whether heavy or light, as well as the coordinates (x, y, z) and momentum (p_x, p_y, p_z) at their generation point, were stored for further analysis.



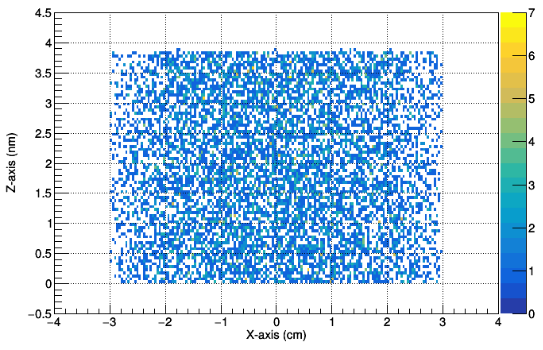


Figure 3.3: Initial position of the fission fragments inside the sample volume. These are used as primary particles at GEANT4 following a uniform distribution.

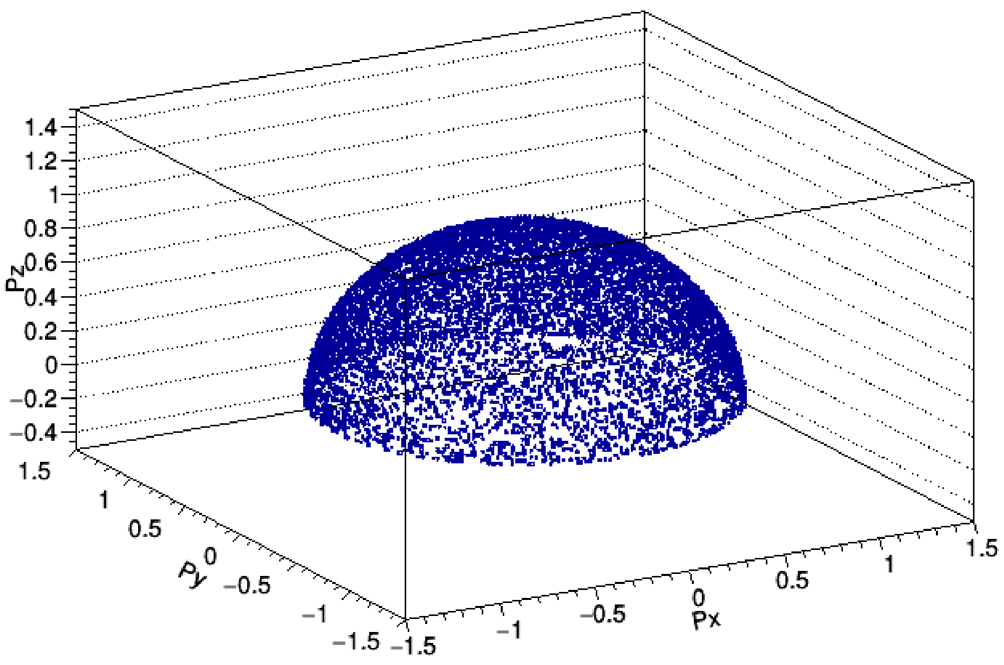


Figure 3.4: Initial momentum of the fission fragments used as primary particles at GEANT4 following a 2π uniform momentum towards the gas volume.

3.2 A GEF/GEANT4 and a GEANT4 standalone simulation comparison

Before the study of the Micromegas detector behavior, it is important to ensure the accuracy of the simulation results. That way, any conclusion achieved can be trusted. Therefore, results from the combination of the GEF code with the GEANT4 toolkit are compared with the ones obtained from a GEANT4 standalone simulation. More specifically, the reaction $^{235}U(n,f)$ with a thermal incident neutron beam is simulated using both methods. The detector geometry and the sample are kept the same with only the primary particle source varied. With GEANT4 standalone a thermal neutron beam is simulated, while with the GEF/GEANT4 combination, the reaction is calculated by GEF and the fragments produced are used as primary particles in GEANT4.

The distributions describing the energy deposition and identity (Z, A) of the detected particles are compared in Figure 3.5 and Figure 3.6, respectively. A good agreement between the two methods is observed. This agreement with the well established GEANT4 toolkit allows for the simulations to be performed using GEF for the fission product properties. As a result, computation speed is increased, since the probability of a fission induced reaction is small in GEANT4 when neutrons are used as primary particles. Finally, it should be noted that a similar comparison of an $^{241}Am(n,f)$ reaction was not possible as the GEANT4 standalone simulation could not produce sufficient statistics with reasonable CPU time. The reason behind this could be the change of physics libraries used to describe the fission process of transuranic elements, indicating another advantage of the GEF code, which provides fission observables for a range of isotopes.

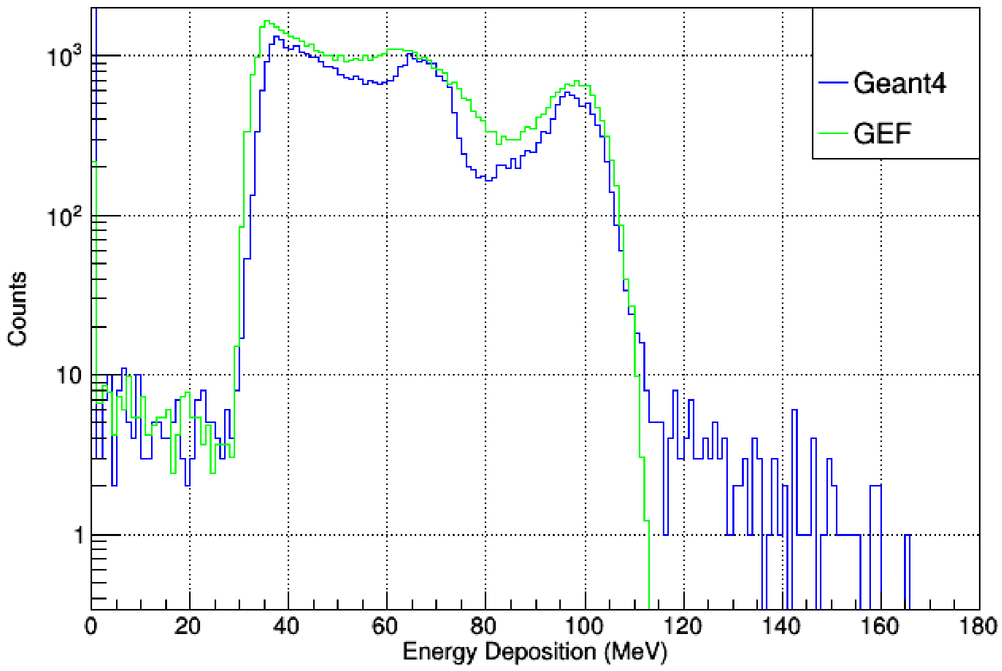
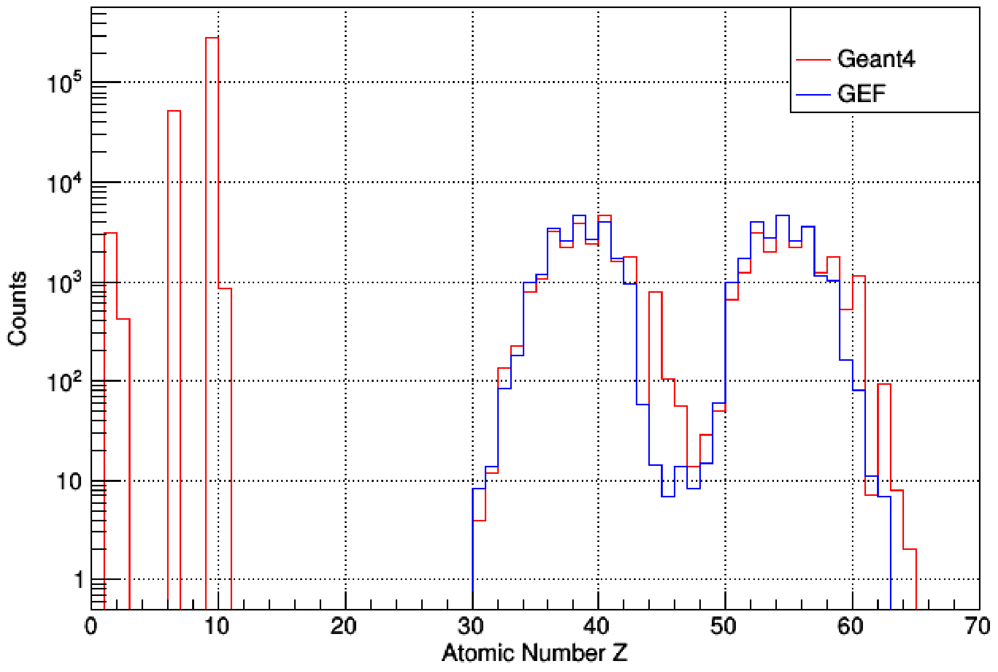
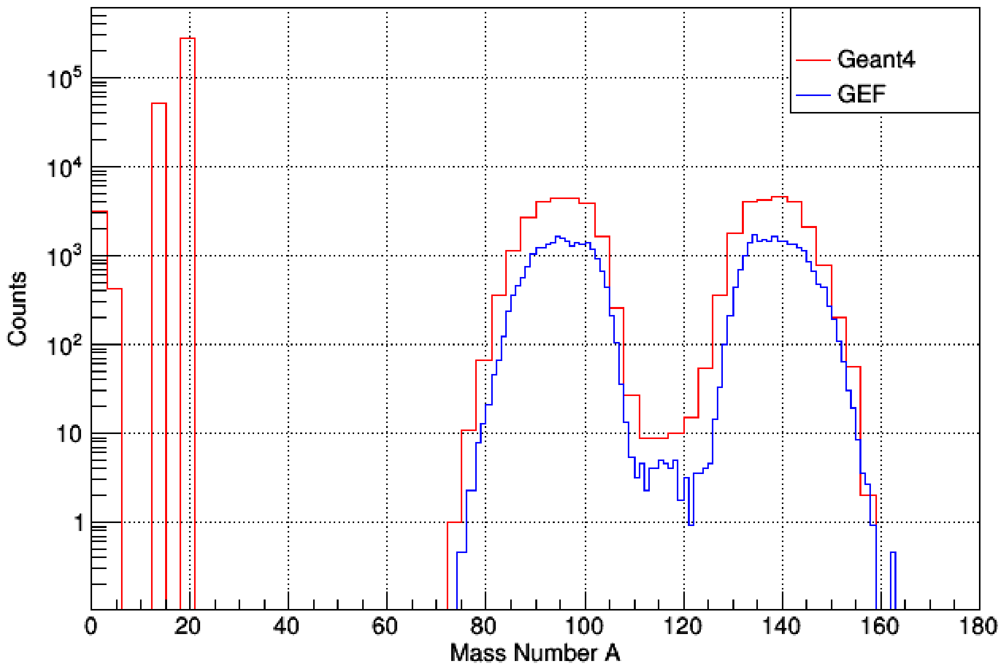


Figure 3.5: Energy deposition of the detected particles.



(a)



(b)

Figure 3.6: (a) Atomic number and (b) mass distribution of the detected particles

3.3 Energy deposition of light and heavy fission products

During the simulations, the properties of the fission product used as the primary particle were stored, including whether it was the light or heavy fragment of the compound nucleus. This information was vital to better understand how differently heavy and light fragments interact with the gas.

The energy deposition of heavy and light fragments from an $^{241}Am(n,f)$ reaction induced by thermal neutrons can be seen in Figure 3.7. For comparison, the energy deposition of all the fragments is plotted as well. It is observed, that the peak seen in higher energies is due to the energy deposition of the light fragments. Additionally, the percent of the deposited energy of the fragments compared to their initial kinetic energy is shown in Figure 3.8. The light fragments lose less energy than the heavy ones.

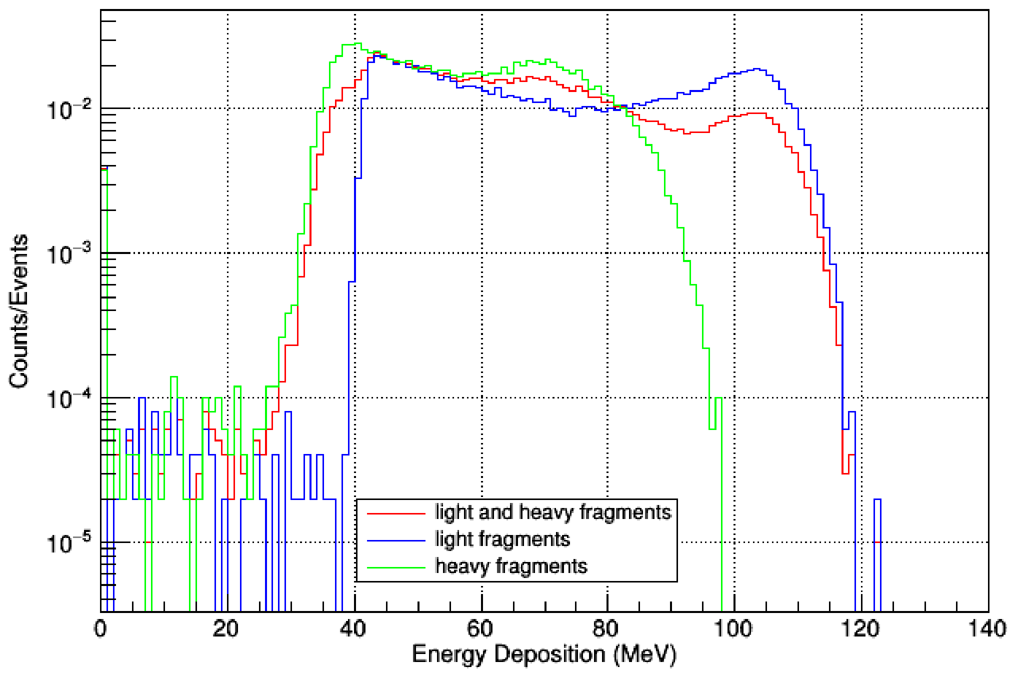


Figure 3.7: Energy deposition of the heavy and light fission fragments (green and blue respectively) from the $^{241}Am(n,f)$ reaction induced by thermal neutrons. The energy deposition of all the fragments (red) is also plotted. The distributions were normalized to the same number of detected particles.

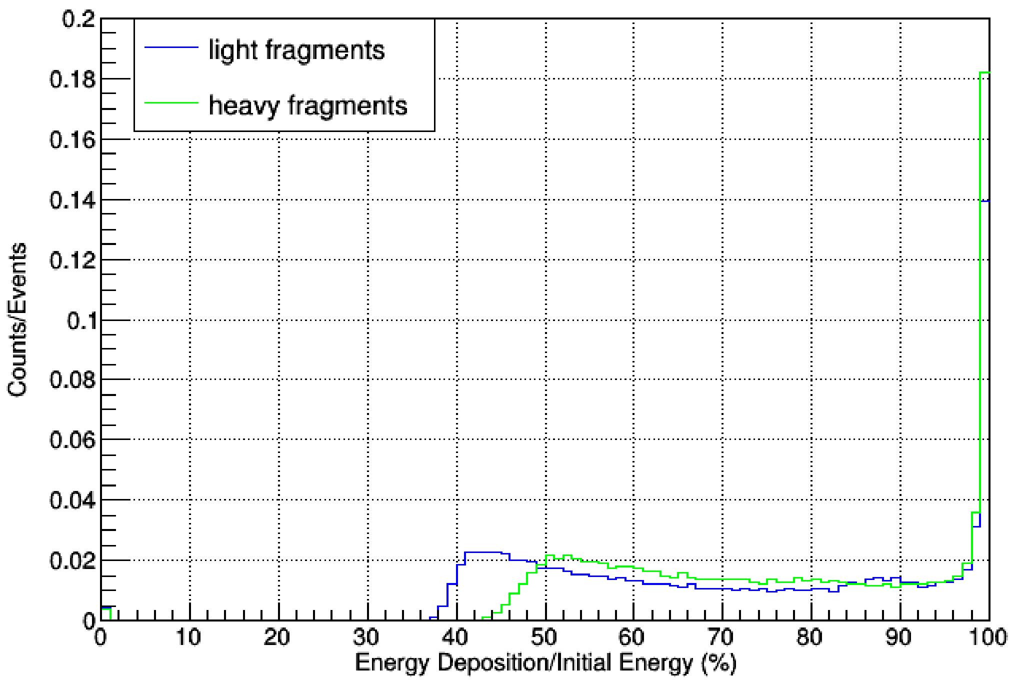


Figure 3.8: Percent of the deposited energy compared to the initial kinetic energy for the heavy and light fission fragments from the $^{241}Am(n,f)$ reaction induced by thermal neutrons. The distributions were normalized to the same number of detected particles.

3.3 Effect of the surface homogeneity of the target

With simulations, the exact reproduction of experimental conditions is impossible. For instance, the surface of the samples used at a measurement of $^{241}Am(n,f)$ cross-section [34] was discovered to be non-homogeneous. The two most extreme target configurations (Figure 3.9) were simulated to determine whether a deviation from a homogeneous surface of a sample affected the energy deposition of the fragments and should be taken into account.

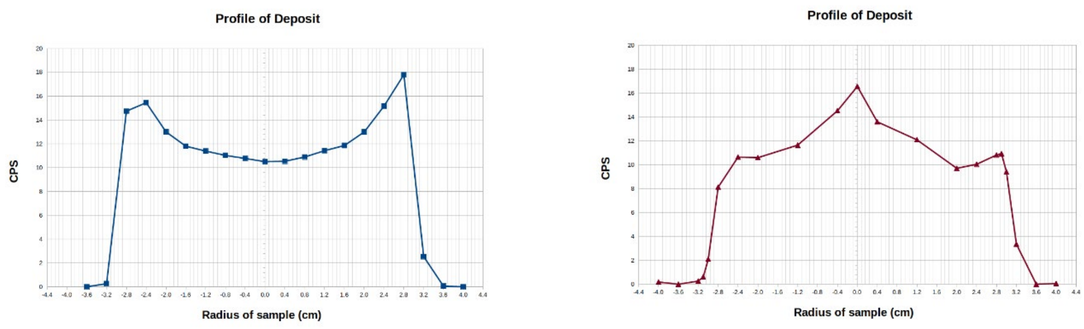
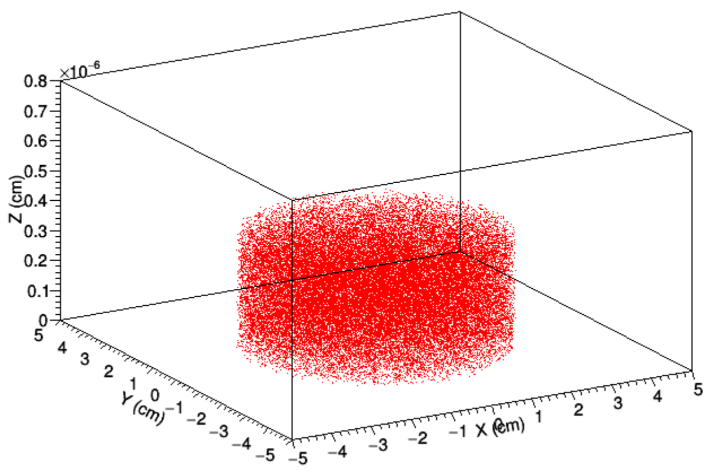


Figure 3.9: Sample configurations from the $^{241}Am(n,f)$ measurement at n_TOF. (Courtesy of Eleme Zinovia) [34]

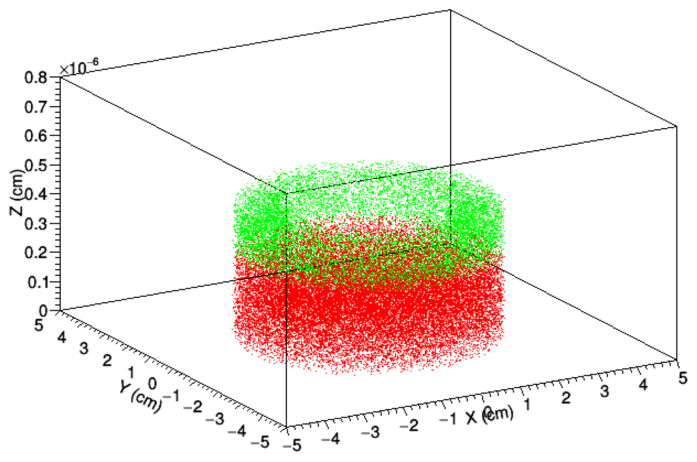
Each simulated sample configuration consisted of two parts: Part 1 and Part 2. More specifically, the first simulated sample configuration (Sample B) consisted of a cylindrical volume with the addition of a ring and the second one (Sample C) was a cylindrical volume with another cylinder on top. For comparison the default geometry of a homogeneous surface of the target (Sample A) was used, as well. In all geometries, the total mass of $148.9\ \mu g$ and $13.6\ gr/cm^3$ density were the same. The exact dimensions of the samples can be seen in Table 3.1 and Figure 3.10. It should be noted that in Sample B and Sample C only the volume describing the second part (Part 2) was different. As a result, a possible dependence on the shape of that volume could be detected.

Sample	**Part 1**				**Part 2**			
	Shape	**R$_{min}$ (cm)**	**R$_{max}$ (cm)**	**Height (nm)**	**Shape**	**R$_{min}$ (cm)**	**R$_{max}$ (cm)**	**Height (nm)**
Sample A	Cylinder	0	3.00	3.85	-	-	-	-
Sample B	Cylinder	0	3.00	2.85	Ring	2.00	3.00	1.80
Sample C	Cylinder	0	3.00	2.85	Cylinder	0	1.60	3.52

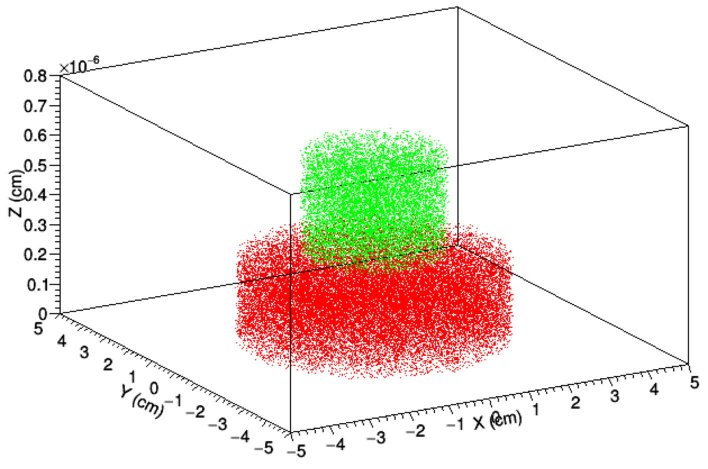
Table 3.1: Dimensions of the simulated sample configurations used to describe a non-homogeneous target configuration. For comparison a homogeneous sample is included, as well. Each configuration consisted of a homogeneous part (Part 1) and a non-homogeneous one (Part 2).



(a)



(b)



(c)

Figure 3.10: Sample configurations used in the simulation: (a) Sample A: a cylinder, (b) Sample B: a cylinder with a ring on top, (c) Sample C: a cylinder with another cylinder. Each configuration consisted of two parts: Part 1 (red volume) and Part 2 (green volume).

The simulated energy deposition of fission fragments from all the sample configurations can be found in Figure 3.7. It should be noted that in the case of Sample C, in which a cylinder is added, slightly less energy deposition is observed. A possible explanation could be that the height of the added volume is increased compared to the other sample geometries. Therefore, if a particle was generated at the bottom of that cylinder and propagated in the forward direction, it would cover more distance, thus losing energy inside the sample before being detected. However, these changes in energy deposition are only very slight and, overall, no significant effect of the target shape is observed on the spectrum. Therefore, we can proceed with the analysis without having to exactly reproduce the target configuration.

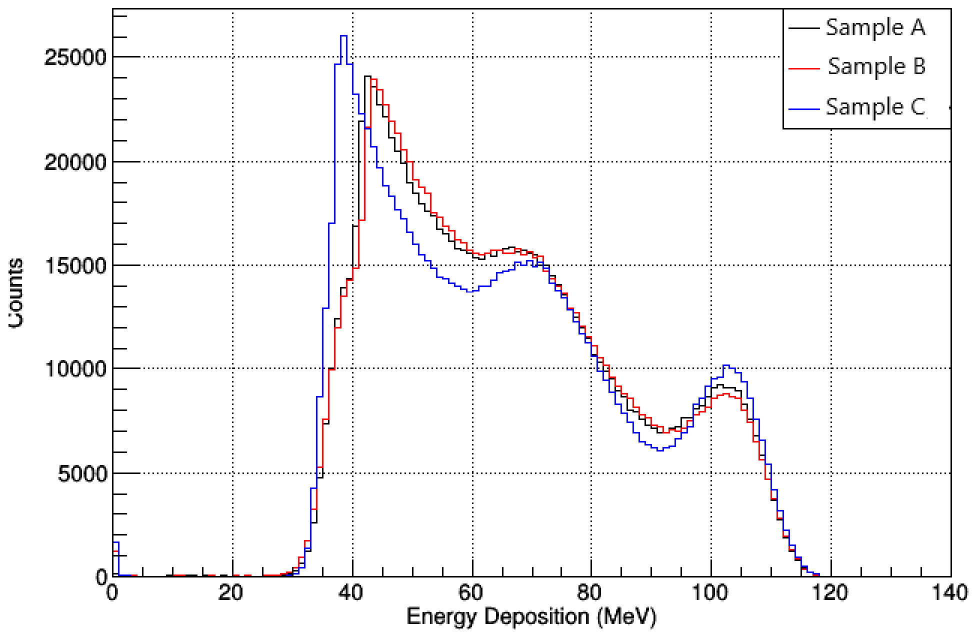


Figure 3.7: Simulated energy deposition of the fission fragments from an $^{241}Am(n,f)$ reaction induced by thermal neutrons. Two non-homogeneous surfaces of a sample were simulated: Sample B (red line) and Sample C (blue line). For comparison a homogeneous surface (Sample A, black line) was also examined.

3.4 Effect of the chemical composition of the target

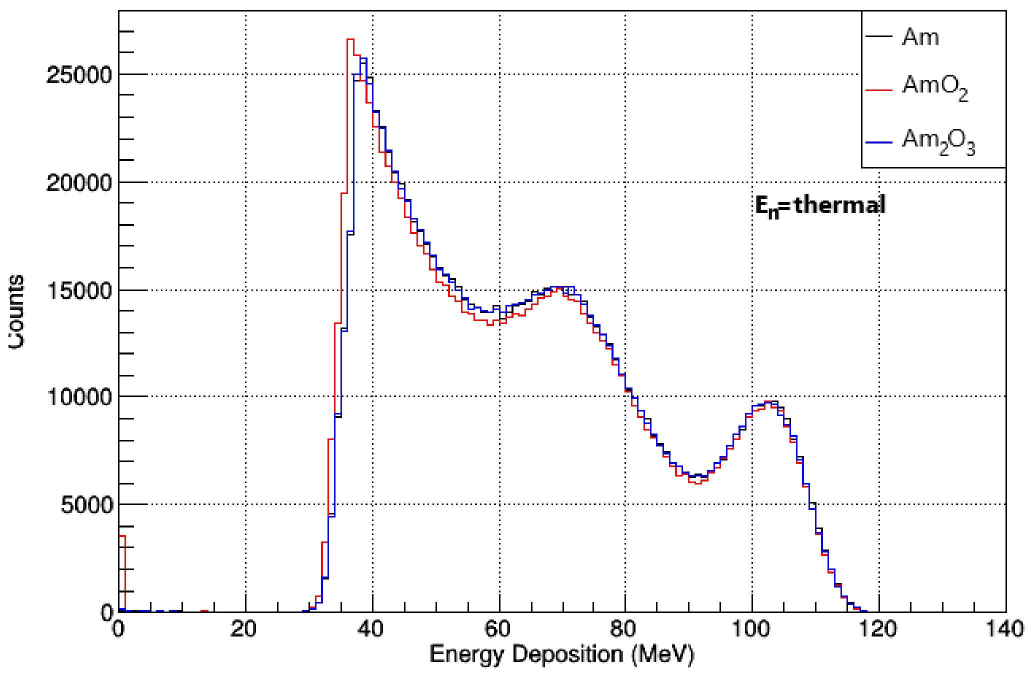
Another effect that was studied was the dependence of the fragments energy deposition on the target chemical composition. Humidity can cause an increase in the oxygen concentration of the samples, thus leading to the production of oxides. To investigate whether the presence of these oxides can affect the obtained results, simulations were performed using various target chemical compositions. More specifically, for the $^{241}Am(n,f)$ and $^{235}U(n,f)$ reactions, the samples of each simulation consisted of a different oxide. In Table 3.2 and Table 3.3, a list with the main characteristics of the samples is provided [36, 37]. The incident neutron energy was chosen to be in the thermal region but also as high as 10 MeV so as to examine a possible dependence on the incident energy. For comparison, the isotopic samples as defined from the preparation procedure at an n_TOF experiment were also simulated. The energy deposition of the fragments can be seen in Figure 3.8 and Figure 3.9 for the reaction $^{241}Am(n,f)$ and $^{235}U(n,f)$ respectively. The different types of oxides in the sample did not significantly affect the energy deposition of the fragments. Therefore, the exact knowledge of the chemical composition is not required.

Sample	**Mass** $(\boldsymbol{\mu g})$	**Density** $(\boldsymbol{g/cm^3})$**[36]**	**Thickness** $(\boldsymbol{nm})$	**Composition** (%)	
Am	148.9	13.67	3.85	99.9838	^{241}Am
				0.0162	^{242}Am
				0.0002	^{243}Am
AmO_2	168.7	11.68	5.11	88.2812	Am
				11.7188	O
Am_2O_3	163.8	11.77	4.92	90.9457	Am
				9.0543	O

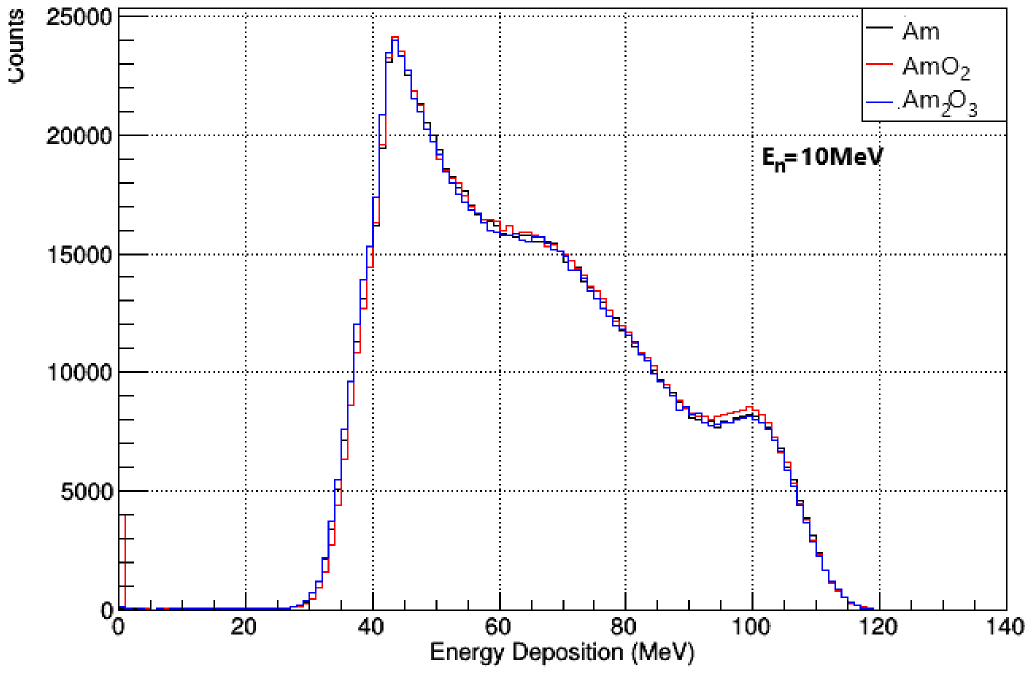
Table 3.2: Main characteristics of the Am samples used in the simulations.

Sample	**Mass** $(\boldsymbol{\mu g})$	**Density** $(\boldsymbol{g/cm^3})$**[36]**	**Thickness** $(\boldsymbol{nm})$	**Composition** (%)	
U	297.9	18.97	5.55	0.035973	^{234}U
				99.9336	^{235}U
				0.009629	^{236}U
				0.02073	^{238}U
UO_2	338.5	10.95	10.93	88.0174	U
				11.9826	O

Table 3.3: Main characteristics of the U samples used in the simulations.

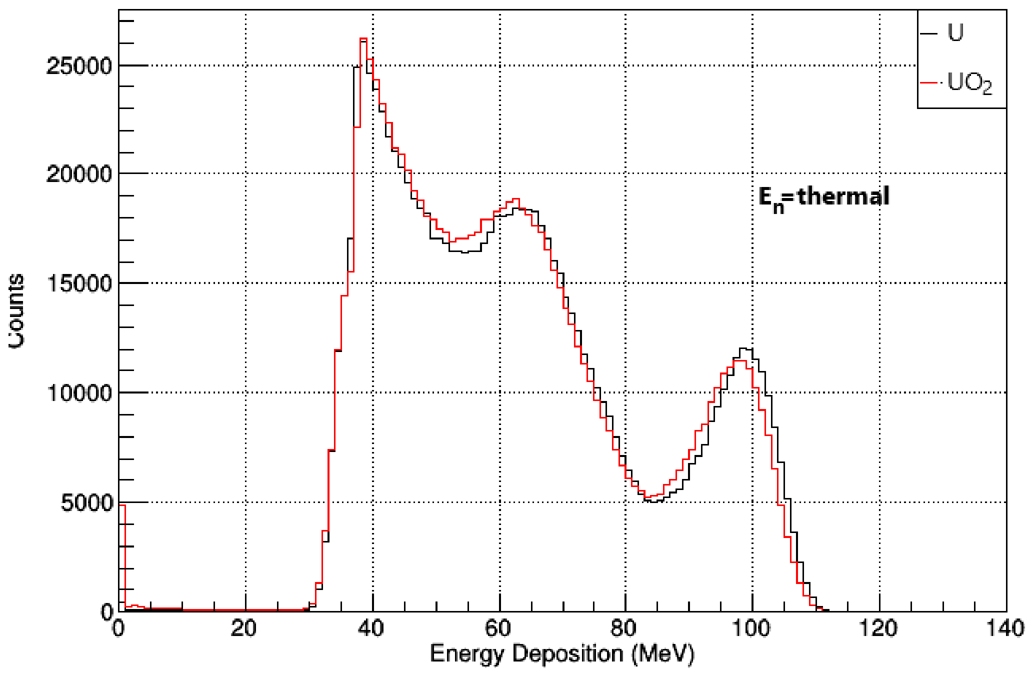


(a)

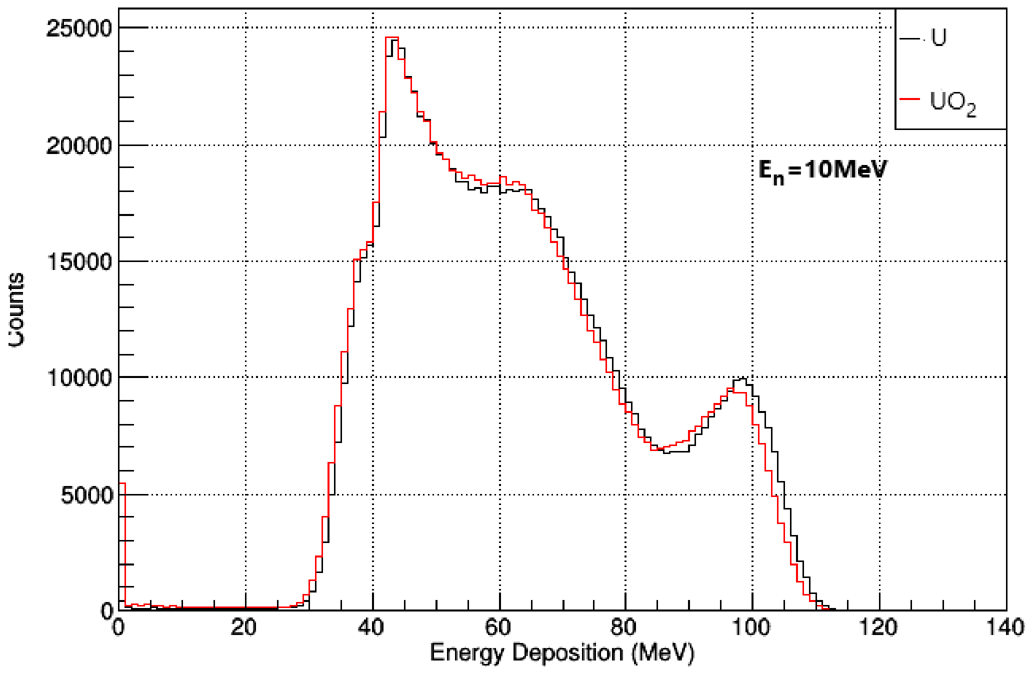


(b)

Figure 3.8: Simulated energy deposition of fission fragments from the $^{241}Am(n,f)$ reaction induced by (a) thermal and (b) 10 MeV neutrons for different target chemical composition.



(a)



(b)

Figure 3.9: Simulated energy deposition of fission fragments from the $^{235}U(n,f)$ reaction induced by (a) thermal and (b) 10 MeV neutrons for different target chemical composition.

Chapter 4

Micromegas detector response function

The simulations performed so far, provided an insight concerning the energy deposition of the fission fragments in the gas. In this chapter, the simulation results are combined with the appropriate response function, in an attempt to reproduce the experimentally observed pulse height spectrum, with emphasis in the low energy region.

The exact response function of the Micromegas detector is yet unknown. Therefore, a qualitative approach was adopted. More specifically, a user-defined function was applied to the simulation results *(output)* with respect to the energy deposition of the fission fragments inside the active gas volume of the detector. The purpose of this function was to reproduce the behavior of the detection set-up and convert the *energy deposition E* into the expected *amplitude in ADC channel*. A description of this approach is discussed below.

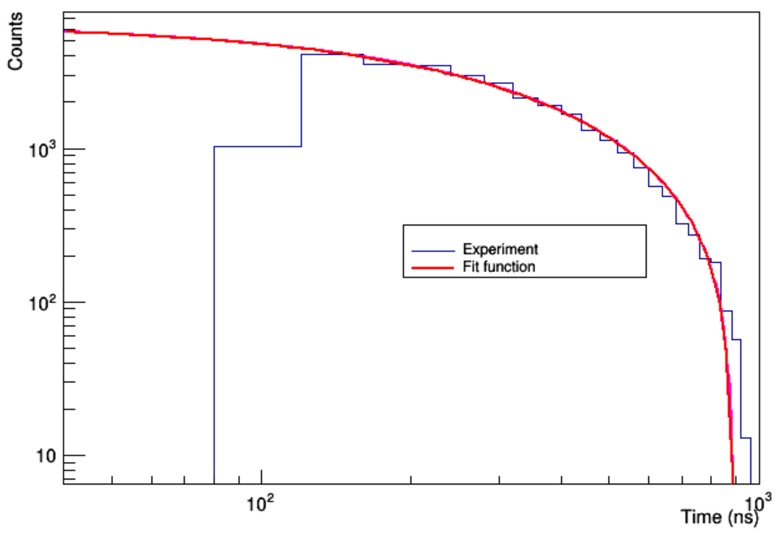
A detector cannot record the exact value of the deposited energy. On the contrary, a distribution is obtained. As a first approximation, a Gaussian one was used with the deposited energy *E* as the distribution's mean value and an energy-depended function as the *deviation* $\sigma(E)$ *(resolution function).* Additionally, voltage is proportional to the energy. Thus, there is a linear dependence between them. As a result, the function is of the form:

$$ADC\ channel = A + B \times Gauss(E, \sigma(E)) \tag{4.1}$$

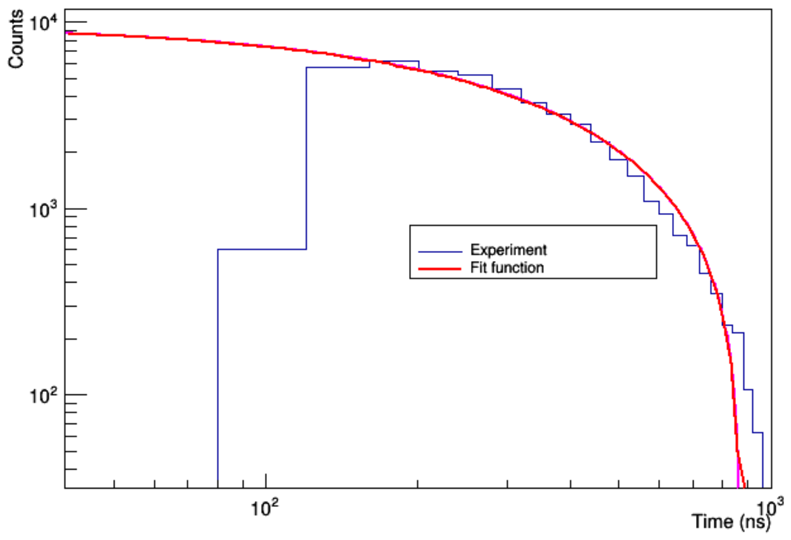
The values of the parameters A, B and the function $\sigma(E)$ were determined by the user.

An important consideration is that, in a detection set-up, the recorded pulses may correspond to the *pile-up* of more than one event. Thus, the recorded amplitude distribution is distorted compared with the true event spectrum. Pile-up occurs, when two pulses are so close together that they are treated as a single pulsed by the analysis system. The superposition of pulses will lead to a combined pulse with an apparent amplitude equal to the sum of the two individual amplitudes. Lesser degrees of overlap will give a combined pulse with an amplitude somewhat less than the sum. [38].

The analysis procedure we opted for in this work is described below. The simulation output events were read in pairs and the time difference between the two events of a pair was estimated. To achieve this, histograms with the recorded time differences between two events for Micromegas detectors from an ^{241}Am fission cross-section measurement [34] were used. More specifically, a fitting and extrapolation of the histogram matching the simulated sample and energy region of the neutron beam, as seen in Figure 4.1, provided a function with the possible time differences. A random number following this function determined the time difference between the events. If that time was higher than a *threshold* set by the user, the events were considered as separate and their energy was converted to the expected ADC channel separately. On the contrary, if it was lower, they were considered as a single event with its energy the sum of the energies that the original events had.



(a)



(b)

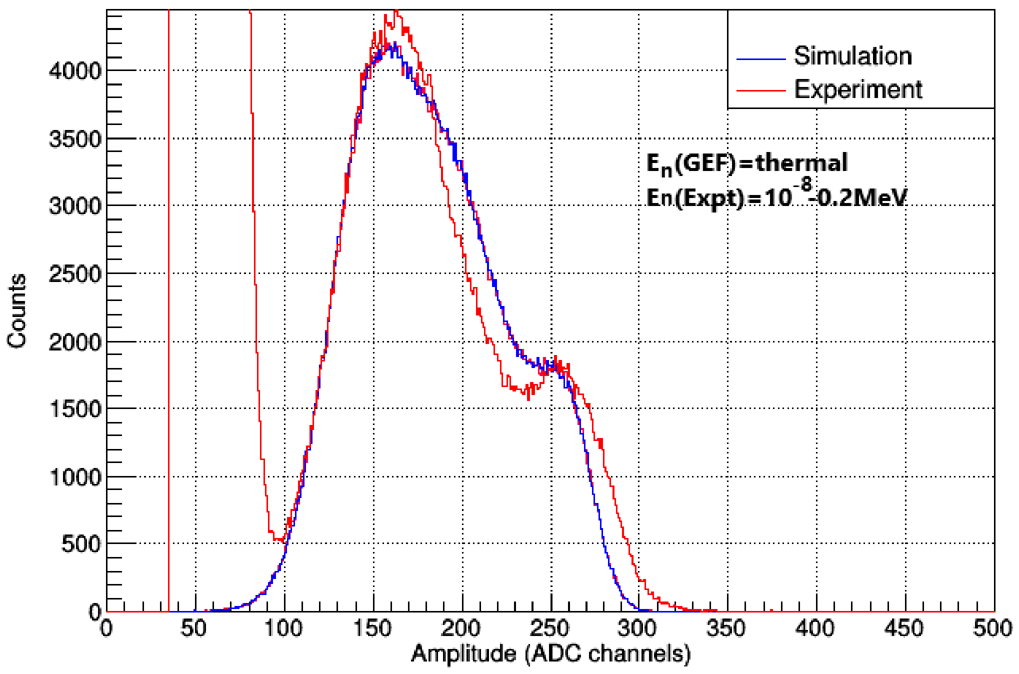
Figure 4.1: The function describing the time differences between two events for a Micromegas detector with: (a) ^{241}Am sample and (b) ^{235}U used in the simulations.

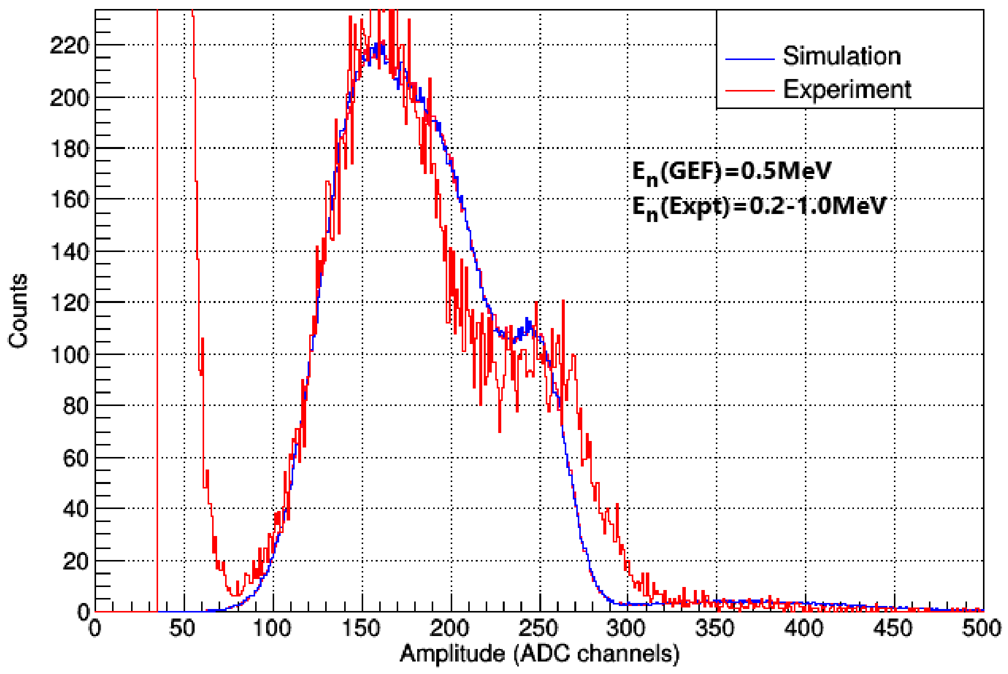
Chapter 5

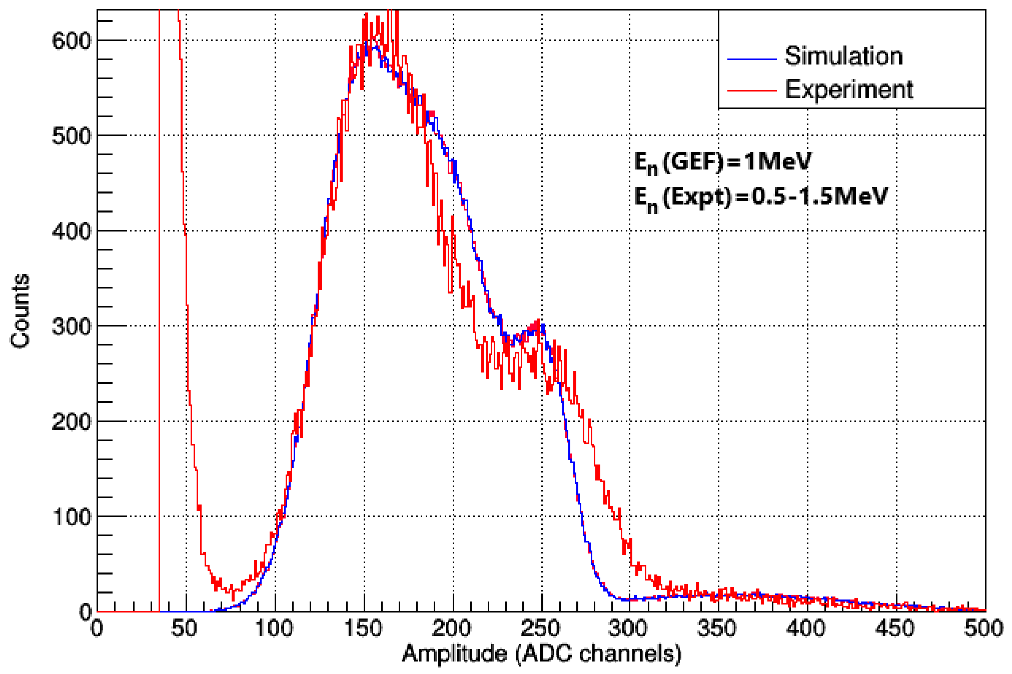
Results

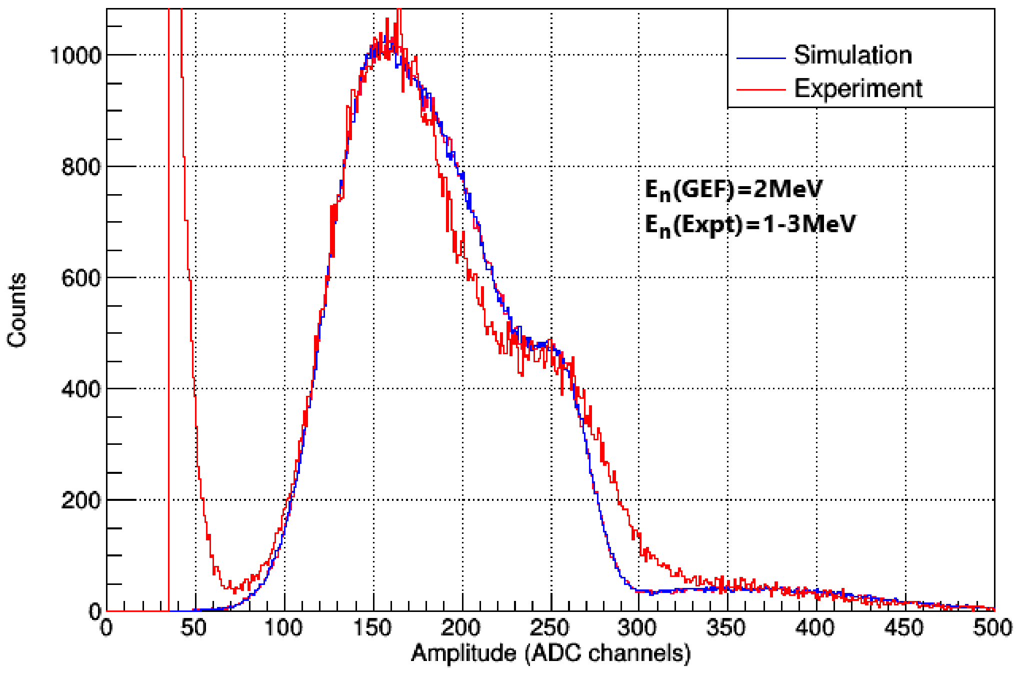
The procedure described in the previous chapter was applied for ^{241}Am and ^{235}U samples and the obtained results were compared with experimental amplitude spectra from the same $^{241}Am(n,f)$ measurements at n_TOF [34]. During the analysis the values of the user-defined threshold as well as the parameters A, B and the function σ in eq. 4.1 were varied until a satisfying reproduction of the experimental amplitude spectra was achieved. The emphasis was given on the low energy region. Moreover, to take into consideration a possible dependence of the detector response function on the incident neutron energy, different energy regions of the neutron beam were examined. As seen in Figures 5.1 and Figure 5.2 the basic characteristics of the experimental spectrum, i.e. the low amplitude region that corresponds to small energy deposition, the position of the peaks and the tail in higher amplitude ADC channels due to pile-up, are well reproduced. However, in the simulation spectra, an overestimation followed by an underestimation between the channels $200 - 250$ and $250 - 300$, respectively, is observed. A possible explanation might be that, in the analysis so far, pile-up was considered as a first approach, a combined pulse with an amplitude equal to the sum of the two original pulses. Different degrees of overlapping, however, can lead to an amplitude smaller than the sum, so this first approach is not a perfect approximation. Therefore, a more detailed description of the pile-up process that includes different degrees of overlapping might sufficiently overcome this inconsistency.

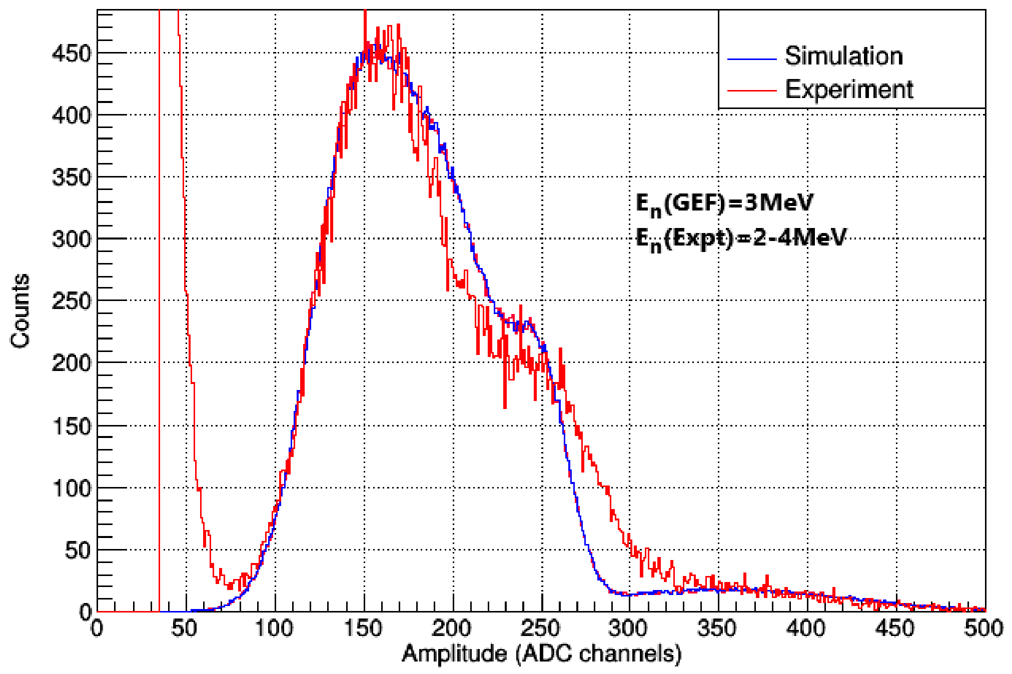
It should be noted, the work performed provided a satisfying reproduction of the experimental amplitude spectrum in the low energy region. This is important for the accurate estimation of the needed correction factors that correspond to the adopted software thresholds set in general and in any n_TOF fission data analysis process. For instance, the amplitude cut correction factor can be calculated, by estimating the expected fission counts that lied below the applied amplitude threshold. Another correction factor that can be estimated through this work is the pile-up correction factor, seeing how the original events from the simulation are stored. If the tail in the high amplitude region results from pile-up, the number of events that have been summed can then be estimated leading to the estimation of the correction factor.


Simulation
Experiment
E_n(GEF)=thermal
En(Expt)=10^{-8}-0.2MeV
Counts
Amplitude (ADC channels)


Simulation
Experiment
E_n(GEF)=0.5MeV
E_n(Expt)=0.2-1.0MeV
Counts
Amplitude (ADC channels)


Simulation
Experiment
E_n(GEF)=1MeV
E_n(Expt)=0.5-1.5MeV
Counts
Amplitude (ADC channels)


Simulation
Experiment
E_n(GEF)=2MeV
E_n(Expt)=1-3MeV
Counts
Amplitude (ADC channels)



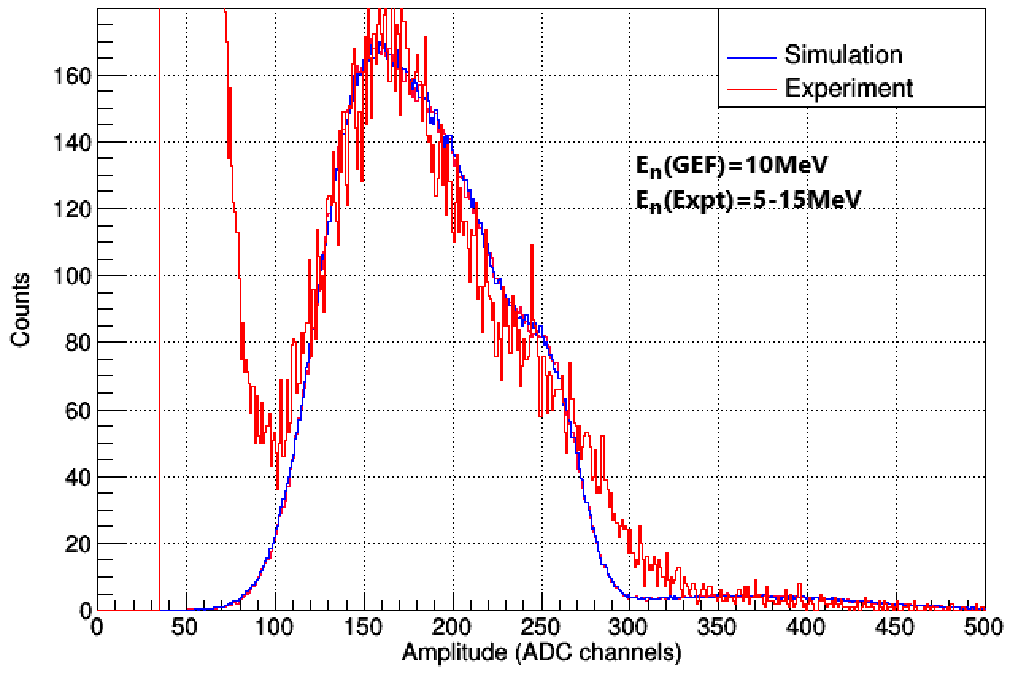
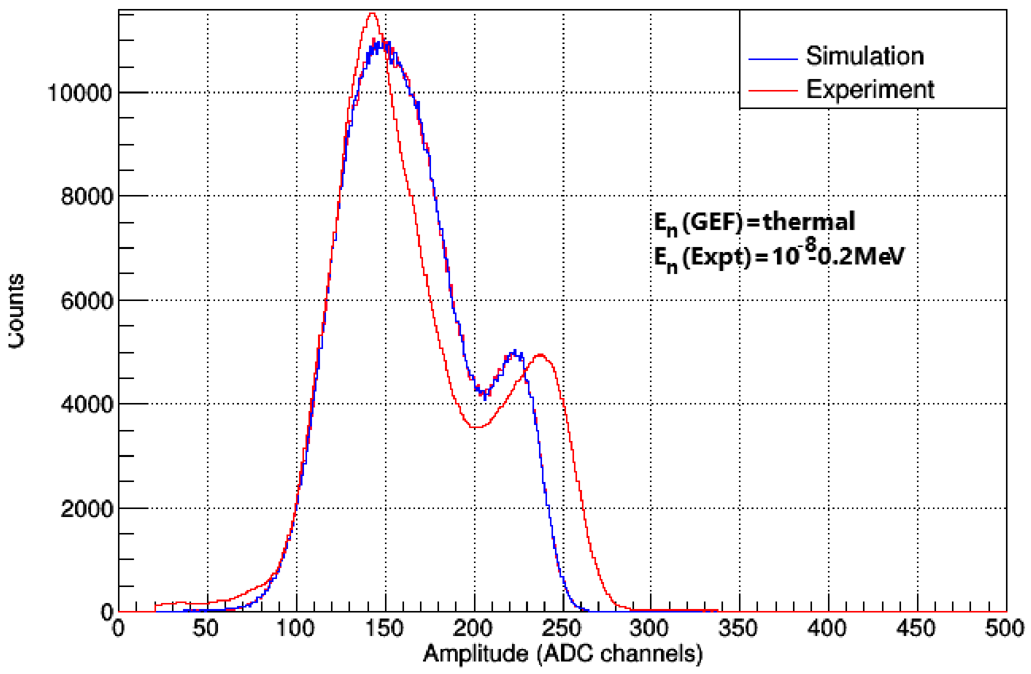
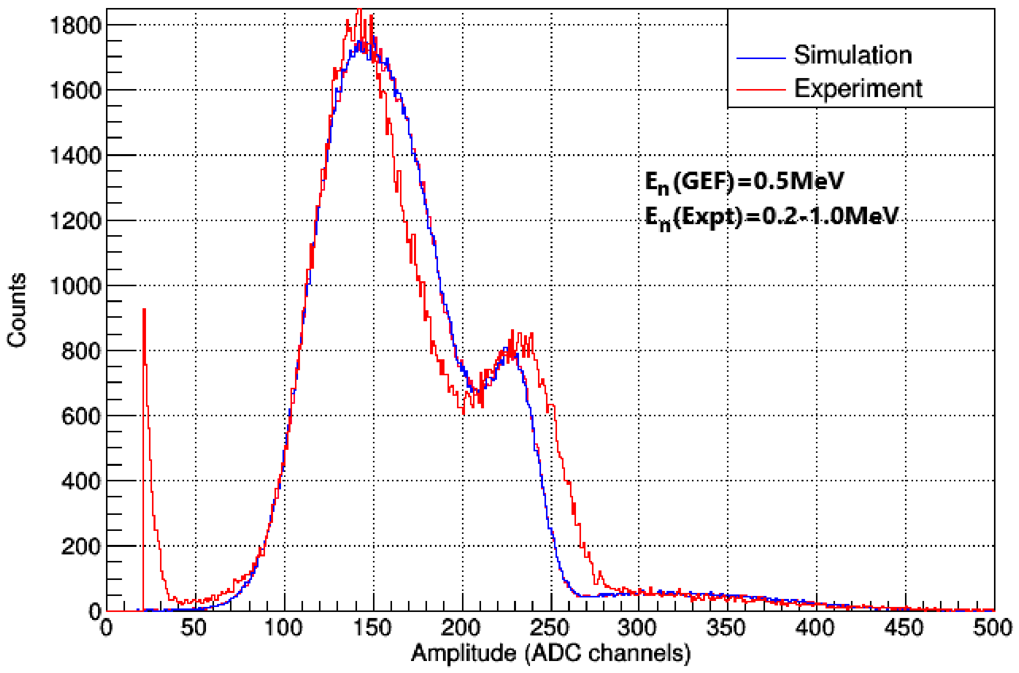


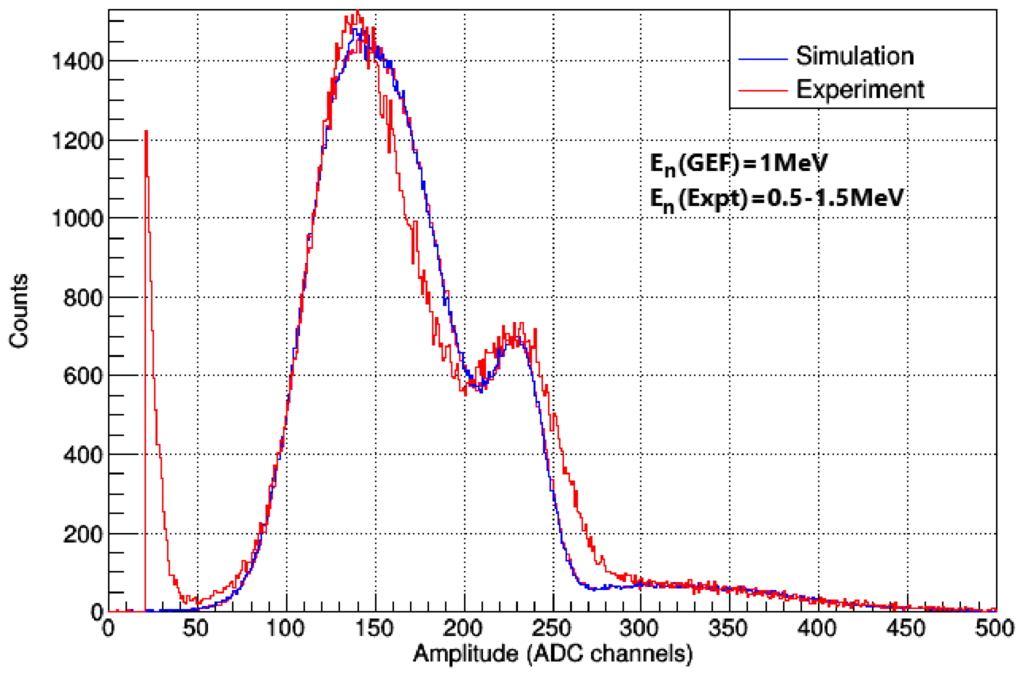
Figure 5.1: Comparison between experimental and simulated ^{241}Am amplitude spectra for various incident neutron energies. The energy used for GEF, as well as the experimental energy ranges, are indicated.



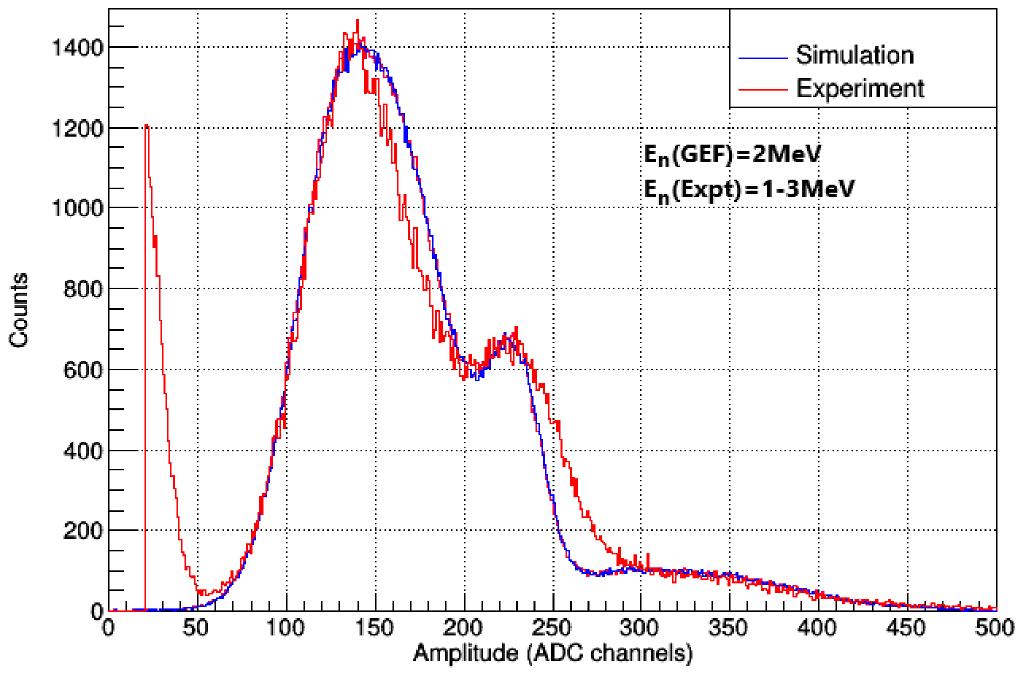
Simulation
Experiment
E_n(GEF)=thermal
E_n(Expt)=10^{-8}-0.2MeV
Counts
Amplitude (ADC channels)



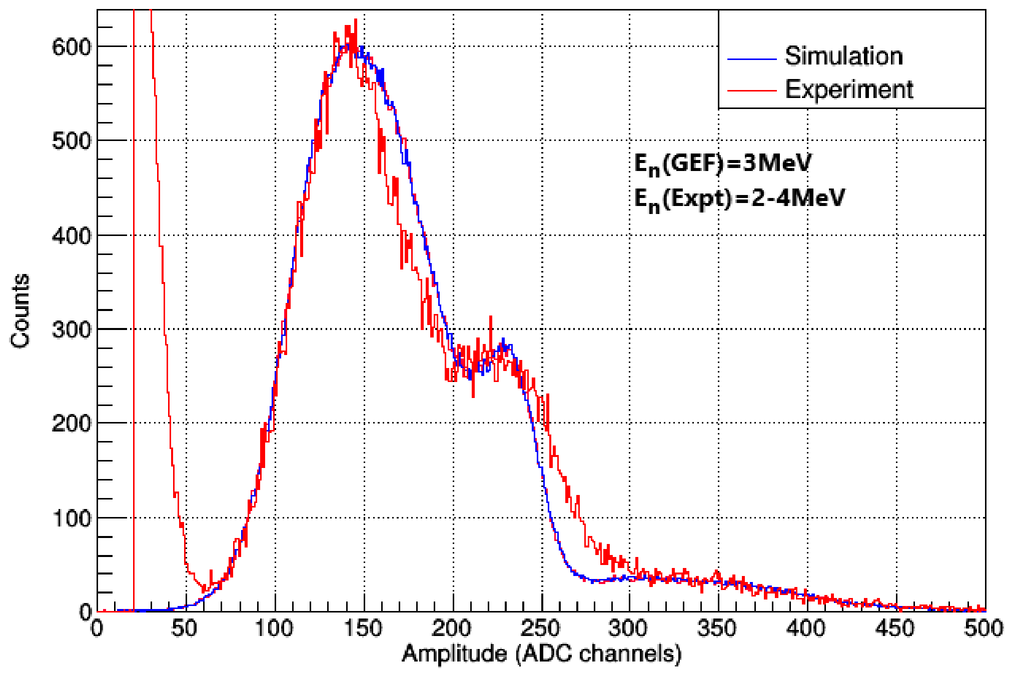
Simulation
Experiment
E_n(GEF)=0.5MeV
E_n(Expt)=0.2-1.0MeV
Counts
Amplitude (ADC channels)



Counts
Amplitude (ADC channels)
Simulation
Experiment
E_n(GEF)=1MeV
E_n(Expt)=0.5-1.5MeV



Counts
Amplitude (ADC channels)
Simulation
Experiment
E_n(GEF)=2MeV
E_n(Expt)=1-3MeV



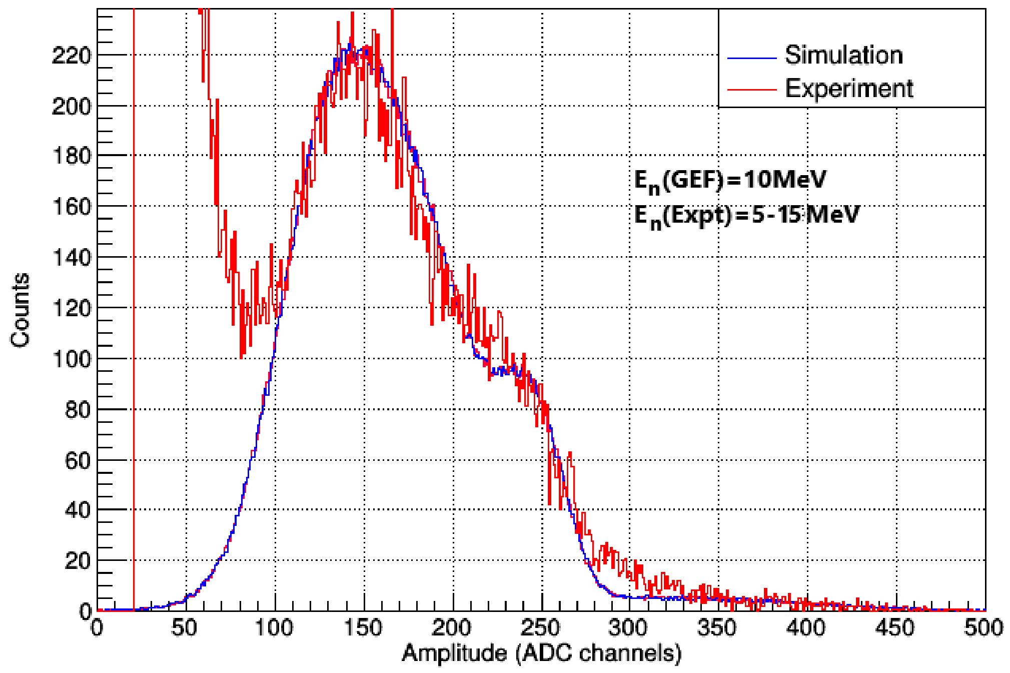


Figure 5.2: Comparison between experimental and simulated ^{235}U amplitude spectra for various incident neutron energies. The energy used for GEF, as well as the experimental energy ranges, are indicated.

Conclusions

In this thesis, extended simulations of Micromegas detectors were performed. The fission process and fission fragments yield were calculated by means of the GEF code and the rest of the particle transportation was accomplished through the GEANT4 toolkit. The results from the combination of the GEF code with GEANT4 were in agreement with the ones of a standalone GEANT4 simulation. Additionally, the energy deposition of light and heavy fission products was studied and no strong dependence of the target homogeneity and chemical composition was observed on the fission products energy deposition. Finally, the simulation results with respect to the energy deposition of fission products within the active gas volume of the detector were combined with the appropriate response function, taking pile-up events into consideration. A satisfying reproduction of the low amplitude region of the experimentally observed spectrum that corresponds to small energy deposition was achieved. Further improvement of the analysis with a more detailed description of the pile-up process that includes different degrees of overlapping may overcome any inconsistency.

Appendix A

ROOT analysis script

This routine reads the GEF output and prepares a file with the fission products properties. This file is read by the GEANT4 application used as a primary particle source. The programs were written in C++ and can be used within ROOT's interpreter.

```
//-----------------------------------------------------------------------------------------------------------------------
{
  gROOT->Reset();

  #include "Riostream.h"
  #include "TMath.h"

  ifstream in;
  in.open("GEFAm.dat");
  ofstream out;
  out.open("GEFAm_out.dat");

  Float_t    Z1=0., Z2=0., A1pre=0. ,A2pre=0., A1post=0., A2post=0., I1pre=0., I2pre=0.,
I1gs=0., I2gs=0., Eexc1=0., Eexc2=0., n1=0., n2=0., TKEpre=0., TKEpost=0.;
  Float_t E1post=0., E2post=0.;

  Int_t counter1=0;

while(in>>Z1>>Z2>>A1pre>>A2pre>>A1post>>A2post>>I1pre>>I2pre>>I1gs>>I2gs>>Eexc1>>
Eexc2>>n1>>n2>>TKEpre>>TKEpost){
    counter1++;
  }

  cout<<"Fission Events in the input file="<<counter1<<endl;

  in.close();
  in.open("GEFAm.dat");


  Int_t counter2=0;
  while(
in>>Z1>>Z2>>A1pre>>A2pre>>A1post>>A2post>>I1pre>>I2pre>>I1gs>>I2gs>>Eexc1>>Eexc2
>>n1>>n2>>TKEpre>>TKEpost){

      E1post = (A2post*TKEpost)/(A1post+A2post);
      E2post = (A1post*TKEpost)/(A1post+A2post);

out<<Z1<<"\t\t"<<A1post<<"\t\t"<<E1post<<"\t\t"<<Z2<<"\t\t"<<A2post<<"\t\t"<<E2post<
<endl;
      counter2++;
      }
  cout<<"Fission Events in the output file="<<counter2<<endl;
  in.close();
  out.close();
  }

//--------------------------------------------------------------------------------------------------------------------
```

^{241}Am

```
//--------------------------------------------------------------------------------------------------------------------
#include "Riostream.h"
#include "TMath.h"
#include "TRandom3.h"
#include "TFile.h"
#include <time.h>

void GEFPrims_Am()
{

   gROOT->Reset();
   TRandom3 *gg = new TRandom3(time(NULL));

   ifstream in;
   in.open("GEFAm_out.dat");
   ofstream out;
   out.open("GEFAm_prim.dat");

   // primaries parameters
   Double_t Z1=0.,A1=0.,E1=0.,Z2=0.,A2=0.,E2=0.;
   Double_t Z=0.,A=0.,Ekin=0.,px0=0.,py0=0.,pz0=0.,x0=0.,y0=0.,z0=0.;
   Int_t ff=0;//output light=0, heavy=1
   Int_t totcounter=0,lcounter=0,hcounter=0; //light, heavy ff counter
   Int_t c0=0,c1=0;

   while(in>>Z1>>A1>>E1>>Z2>>A2>>E2){
    Int_t randNum = rand()%2;
    cout<<randNum<<endl;
    if(randNum){
      A = A2;
      Ekin = E2;
      ff=1; // heavy
      c1++;
      cout<<"1:"<<randNum<<" "<<ff<<endl;
    }
    else{
      Z = Z1;
      A = A1;
      Ekin = E1;
      ff=0;
      c0++;
      cout<<"0:"<<randNum<<" "<<ff<<endl;
    }

    if(ff){
      hcounter++;
    }
    else{
      lcounter++;
    }
    totcounter++;

    Double_t pi = TMath::Pi();
    Double_t phi= 2.*pi* (gg->Uniform());
    Double_t cosTheta = 1. - (gg->Uniform());
    Double_t sinTheta = sqrt(1. - cosTheta * cosTheta);
    px0= sinTheta * cos(phi);
    py0= sinTheta * sin(phi);
    pz0 =cosTheta;

    Double_t Amtc = 0.0000003852422, Altc = 0.0025;
    Double_t Radius = 3.0; //
    Double_t dx0 = 6., dy0 = 6., dz0 = Amtc;
    while(true){
      x0=((gg->Uniform())*dx0)-3.0;
      y0=((gg->Uniform())*dy0)-3.0;
      z0=((gg->Uniform())*dz0)+(Altc/2.);
      if(((x0*x0)+(y0*y0)) <= (Radius*Radius)){
            break;
      }
    }
out<<Z<<"\t"<<A<<"\t"<<Ekin<<"\t"<<px0<<"\t"<<py0<<"\t"<<pz0<<"\t"<<x0<<"\t"<<y0<<"
\t"<<z0<<"\t"<<ff<<endl;
  }
  cout<<"total="<<totcounter<<" light="<<lcounter<<" heavy="<<hcounter<<" c0="<<c0<<"
c1="<<c1<<endl;
  in.close();
  out.close();
}
/------------------------------------------------------------------------------------------------------------------------
```

^{235}U

```
//------------------------------------------------------------------------------------------------------------------------
#include "Riostream.h"
#include "TMath.h"
#include "TRandom3.h"
#include "TFile.h"
#include <time.h>

void GEFPrims_U()
{
  gROOT->Reset();
  TRandom3 *gg = new TRandom3(time(NULL));

  ifstream in;
  in.open("GEFUthermal_out.dat");
  ofstream out;
```

```
out.open("GEFUthermal_prim.dat");

Double_t Z1=0.,A1=0.,E1=0.,Z2=0.,A2=0.,E2=0.;
Double_t Z=0.,A=0.,Ekin=0.,px0=0.,py0=0.,pz0=0.,x0=0.,y0=0.,z0=0.;
Int_t ff=0;
Int_t totcounter=0,lcounter=0,hcounter=0;
Int_t c0=0,c1=0;

while(in>>Z1>>A1>>E1>>Z2>>A2>>E2){
  Int_t randNum = rand()%2;
  cout<<randNum<<endl;
  if(randNum){
    Z = Z2;
    A = A2;
    Ekin = E2;
    ff=1; // heavy
    c1++;
    cout<<"1:"<<randNum<<" "<<ff<<endl;
  }
  else{
    Z = Z1;
    A = A1;
    Ekin = E1;
    ff=0;
    c0++;
    cout<<"0:"<<randNum<<" "<<ff<<endl;
  }

  if(ff){
    hcounter++;
        }
  else{
    lcounter++;
  }

  totcounter++;

  Double_t pi = TMath::Pi();
  Double_t phi= 2.*pi* (gg->Uniform());
  Double_t cosTheta = 1. - (gg->Uniform());
  Double_t sinTheta = sqrt(1. - cosTheta * cosTheta);
  px0= sinTheta * cos(phi);
  py0= sinTheta * sin(phi);
  pz0 =cosTheta;

   Double_t Amtc = 0.0000005550334, Altc = 0.0025;
  Double_t Radius = 3.0;
  Double_t dx0 = 6., dy0 = 6., dz0 = Amtc;
  while(true){
    x0=((gg->Uniform())*dx0)-3.0;
    y0=((gg->Uniform())*dy0)-3.0;
```

```
      z0=((gg->Uniform())*dz0)+(Altc/2.);
      if(((x0*x0)+(y0*y0)) <= (Radius*Radius)){
             break;
      }
     }

out<<Z<<"\t"<<A<<"\t"<<Ekin<<"\t"<<px0<<"\t"<<py0<<"\t"<<pz0<<"\t"<<x0<<"\t"<<y0<<"
\t"<<z0<<"\t"<<ff<<endl;
  }
  cout<<"total="<<totcounter<<" light="<<lcounter<<" heavy="<<hcounter<<" c0="<<c0<<"
c1="<<c1<<endl;
  in.close();
  out.close();
}
```

Appendix B

GEANT4 simulation code

The GEANT4 simulation code.

```
//----------------------------------------------------------------------------------------------------------------------
                                        Main.cc (241Am)
//----------------------------------------------------------------------------------------------------------------------
#include "G4RunManager.hh"
#include "G4VisExecutive.hh"
#include "G4UImanager.hh"
#include "G4UIterminal.hh"
#include "globals.hh"
#include "G4ios.hh"
#include "XriDetectorConstruction.hh"
#include "QBBC.hh"
#include "XriPrimaryGeneratorAction.hh"
#include "XriRunAction.hh"
#include "XriEventAction.hh"
#include "XriSteppingAction.hh"
#include <stdlib.h> //library ths c++
#include "G4ios.hh"
#include <stdio.h>
#include <iostream>
#include <fstream>
#include <sstream>
#include "Randomize.hh"
#include "time.h"
#include "vars.h"

using namespace std;

ofstream outfilem;

// declare global variables
G4double Depos, evtNo;
const int GEF_SIZE = 1000000;
G4double eventspicked[GEF_SIZE];
G4double heavylight[GEF_SIZE];
G4double ffZ[GEF_SIZE];
G4double ffA[GEF_SIZE];

struct sevent totalevents[GEF_SIZE];

int main(int argc, char** argv)
{
  G4double ffid = 0.;
  G4double Z = 0.;
  G4double A = 0.;
  G4double Ekin=0.;
  G4double px0 = 0.;
  G4double py0 = 0.;
  G4double pz0 = 0.;
  G4double x0 = 0.;
  G4double y0 = 0.;
  G4double z0 = 0.;
  G4double ff = 0.;

  ifstream in;
  in.open("GEFAm_prim.dat");

  for(int i=0; i<GEF_SIZE; i++){
    totalevents[i].ffid = i+1;
in>>totalevents[i].Z>>totalevents[i].A>>totalevents[i].Ekin>>totalevents[i].px0>>totalevents[
i].py0>>totalevents[i].pz0>>totalevents[i].x0>>totalevents[i].y0>>totalevents[i].z0>>totaleve
nts[i].ff;
  }

  in.close();

  outfilem.close();

  for(int i=0; i<GEF_SIZE; i++){
    eventspicked[i]=0.;  //history of picked events is reset before each run
   }

  // check if eventspicked is reset before each execution of programme
  G4double mres =0.;
  for(int i=0; i<GEF_SIZE; i++){
    if(eventspicked[i]){ //an event is not reset
      mres ++;
    }
  }
  if(mres){
    G4cout<<"main.cc event is not reset!!!"<<endl;
   }
  else{
    G4cout<<"main.cc events all reset!!!"<<endl;
    }

  //choose the Random engine
  CLHEP::HepRandom::setTheEngine(new CLHEP::RanecuEngine());
  //set random seed with system time
  G4long seed = time(NULL);
  CLHEP::HepRandom::setTheSeed(seed);

  // Construct the default run manager
  G4RunManager* runManager = new G4RunManager;
```

```
// set mandatory initialization classes
XriDetectorConstruction *XriDet = new XriDetectorConstruction;
runManager->SetUserInitialization(XriDet);
// Physics list
G4VModularPhysicsList* physicsList = new QBBC;
physicsList->SetVerboseLevel(1);
runManager->SetUserInitialization(physicsList);

  #ifdef G4VIS_USE
G4VisManager *visManager = new G4VisExecutive;
visManager->Initialize();
#endif

// set mandatory user action class
runManager->SetUserAction(new XriPrimaryGeneratorAction);

// set optional user action class
runManager->SetUserAction(new RunActionXri);

XriEventAction *eventAction = new XriEventAction;
runManager->SetUserAction(eventAction);

runManager->SetUserAction(new XriSteppingAction(XriDet)); //,eventAction));


// Initialize G4 kernel
runManager->Initialize();

// get the pointer to the UI manager and set verbosities
G4UImanager* XUI = G4UImanager::GetUIpointer();
if (argc==1)
{
  G4UIsession *XriSession = new G4UIterminal;

  XUI->ApplyCommand("/run/verbose 0");
  XUI->ApplyCommand("/event/verbose 0");
  XUI->ApplyCommand("/tracking/verbose 0");
  XriSession->SessionStart();
  delete XriSession;
}
else
{
  G4String command = "/control/execute ";
  G4String fileName = argv[1];  // giati 1 ki oxi 0 ?
  XUI->ApplyCommand(command+fileName);
}

// job termination

#ifdef G4VIS_USE
   delete visManager;
  #endif

  delete runManager;

  return 0;
}


//-----------------------------------------------------------------------------------------------------------------
```

Main.cc (^{235}U)

```
//-----------------------------------------------------------------------------------------------------------------
#include "G4RunManager.hh"
#include "G4VisExecutive.hh"
#include "G4UImanager.hh"
#include "G4UIterminal.hh"
#include "globals.hh"
#include "G4ios.hh"
#include "XriDetectorConstruction.hh"
#include "QGSP_BIC_HP.hh"
#include "XriPrimaryGeneratorAction.hh"
#include "XriRunAction.hh"
#include "XriEventAction.hh"
#include "XriSteppingAction.hh"
#include <stdlib.h> //library ths c++
#include "G4ios.hh"
#include <stdio.h>
#include <iostream>
#include <fstream>
#include <sstream>
#include "Randomize.hh"
#include "time.h"
#include "vars.h"

using namespace std;

ofstream outfilem;

// declare global variables
G4double Depos, evtNo;
const int GEF_SIZE = 1000000;
G4double eventspicked[GEF_SI
Main.cc (235 U)
#include "G4RunManager.hh"
#include "G4VisExecutive.hh"
#include "G4UImanager.hh" ZE];
G4double heavylight[GEF_SIZE];
G4double ffZ[GEF_SIZE];
G4double ffA[GEF_SIZE];

struct sevent totalevents[GEF_SIZE];
```

```
int main(int argc, char** argv)
{
 G4double ffid = 0.;
 G4double Z = 0.;
 G4double A = 0.;
 G4double Ekin=0.;
 G4double px0 = 0.;
 G4double py0 = 0.;
 G4double pz0 = 0.;
 G4double x0 = 0.;
 G4double y0 = 0.;
 G4double z0 = 0.;
 G4double ff = 0.;

 // read GEF output
 ifstream in;
 in.open("GEFU_prim.dat");

 for(int i=0; i<GEF_SIZE; i++){
   totalevents[i].ffid = i+1;

in>>totalevents[i].Z>>totalevents[i].A>>totalevents[i].Ekin>>totalevents[i].px0>>totalevents[
i].py0>>totalevents[i].pz0>>totalevents[i].x0>>totalevents[i].y0>>totalevents[i].z0>>totaleve
nts[i].ff;
 }

 in.close();

 // fill array with picked events aka history
 for(int i=0; i<GEF_SIZE; i++){
   eventspicked[i]=0.;
  }

 // check if eventspicked is reset before each execution of programme
 G4double mres =0.;
 for(int i=0; i<GEF_SIZE; i++){
   if(eventspicked[i]){ //an event is not reset
     mres ++;
   }
 }
 if(mres){
   G4cout<<"main.cc event is not reset!!!"<<endl;
  }
 else{
   G4cout<<"main.cc events all reset!!!"<<endl;
 }
}

 //choose the Random engine
 CLHEP::HepRandom::setTheEngine(new CLHEP::RanecuEngine());
 //set random seed with system time
```

```
G4long seed = time(NULL);
CLHEP::HepRandom::setTheSeed(seed);

// Construct the default run manager
G4RunManager* runManager = new G4RunManager;

// set mandatory initialization classes
XriDetectorConstruction *XriDet = new XriDetectorConstruction;
runManager->SetUserInitialization(XriDet);
// Physics list
G4VModularPhysicsList* physicsList = new QGSP_BIC_HP;
physicsList->SetVerboseLevel(1);
runManager->SetUserInitialization(physicsList);

  #ifdef G4VIS_USE
G4VisManager *visManager = new G4VisExecutive;
visManager->Initialize();
#endif

// set mandatory user action class
runManager->SetUserAction(new XriPrimaryGeneratorAction);

// set optional user action class
runManager->SetUserAction(new RunActionXri);

XriEventAction *eventAction = new XriEventAction;
runManager->SetUserAction(eventAction);

runManager->SetUserAction(new XriSteppingAction(XriDet));

// Initialize G4 kernel
runManager->Initialize();

// get the pointer to the UI manager and set verbosities
G4UImanager* XUI = G4UImanager::GetUIpointer();
if (argc==1)
{
  G4UIsession *XriSession = new G4UIterminal;
  XUI->ApplyCommand("/run/verbose 0");
  XUI->ApplyCommand("/event/verbose 0");
  XUI->ApplyCommand("/tracking/verbose 0");
  XriSession->SessionStart();
  delete XriSession;
}
else
{
  G4String command = "/control/execute ";
  G4String fileName = argv[1];
  XUI->ApplyCommand(command+fileName);
}
```

```
  // job termination

  #ifdef G4VIS_USE
    delete visManager;
  #endif

  delete runManager;

  return 0;
}
```

//--

vars.h

//--

```
/*---------------------------------------------------------------------------
VARS.H
---------------------------------------------------------------------------
---------------------------------------------------------------------------*/
#ifndef vars_h
#define vars_h 1

#include "globals.hh"

struct sevent{
  G4double ffid;
  G4double Z;
  G4double A;
  G4double Ekin;
  G4double px0;
  G4double py0;
  G4double pz0;
  G4double x0;
  G4double y0;
  G4double z0;
  G4double ff;
};

extern struct sevent totalevents[];

extern G4double eventspicked[];

extern G4double heavylight[];

extern G4double ffZ[];

extern G4double ffA[];

extern const int GEF_SIZE;

#endif
```

XriDetectorConstruction.cc

```
#include "XriDetectorConstruction.hh"
#include "G4SDManager.hh"
#include "G4Element.hh"
#include "G4Material.hh"
#include "G4Box.hh"
#include "G4Tubs.hh"
#include "G4LogicalVolume.hh"
#include "G4ThreeVector.hh"
#include "G4PVPlacement.hh"
#include "G4UnitsTable.hh"
#include "globals.hh"
#include "G4SystemOfUnits.hh"
#include "G4PhysicalConstants.hh"


#include "G4VisAttributes.hh"
#include "G4Colour.hh"

XriDetectorConstruction::XriDetectorConstruction()
{;}

XriDetectorConstruction::~XriDetectorConstruction()
{;}

G4VPhysicalVolume* XriDetectorConstruction::Construct()
{

G4UnitDefinition::BuildUnitsTable();

//========================== elements ===========================//

  G4double a;
  G4double z;
  // G4int iz, in;
  G4double density;
  G4String name, symbol;
  G4int ncomponents;
  G4double fractionmass;
  G4int natoms;
//  G4double temperature, pressure;

  a = 39.948*g/mole;
  G4Element* elAr  = new G4Element(name="Argon", symbol=" F", z= 9., a);

  a = 18.998*g/mole;
  G4Element* elF  = new G4Element(name="Fluorine", symbol=" F", z= 9., a);

  a = 12.011*g/mole;
```

```
 G4Element* elC  = new G4Element(name="Carbon",symbol=" C" , z= 6., a);

 a = 1.008*g/mole;
 G4Element* elH  = new G4Element(name="Hydrogen",  symbol=" H" , z= 1., a);

 a = 15.999*g/mole;
 G4Element* elO  = new G4Element(name="Oxygen",  symbol=" O" , z= 8., a);

 a = 14.007*g/mole;
 G4Element* elN  = new G4Element(name="Nitrogen",symbol=" N" , z= 7., a);

G4double iz, in;
  G4Isotope* isoU235 = new G4Isotope(name="U235", iz=92, in=235,  a = 235.043*g/mole);

 G4Element* elenrichedU = new G4Element("enrichedU", "U" ,ncomponents=1);
 elenrichedU->AddIsotope(isoU235, fractionmass=100.*perCent);


 /*  G4Isotope*  Am241  =  new  G4Isotope(name="241Am",  iz=95,  in=241,    a  =
241.056*g/mole);
  */

//======================materials=====================================//

// ---------- air of Micro_Megas

 density = 1.661*mg/cm3;
 G4Material *ArGas = new G4Material(name="ArGas",density,ncomponents=1);
 ArGas->AddElement(elAr, fractionmass=100.0*perCent);

 density = 3.6602189*mg/cm3;
 G4Material* CF4Gas = new G4Material(name="CF4Gas",density,ncomponents=2);
 CF4Gas->AddElement(elC, natoms=1);
 CF4Gas->AddElement(elF, natoms=4);

 density = 2.489*mg/cm3;
 G4Material* C4H10Gas = new G4Material(name="C4H10Gas",density,ncomponents=2);
 C4H10Gas->AddElement(elC, natoms=4);
 C4H10Gas->AddElement(elH, natoms=10);

 density = 1.87748189*mg/cm3;
 G4Material* MegasGas = new G4Material(name="MegasGas", density,ncomponents=3);
 MegasGas->AddMaterial(ArGas,fractionmass=77.92*perCent);
 MegasGas->AddMaterial(CF4Gas,fractionmass=19.51*perCent);
 MegasGas->AddMaterial(C4H10Gas,fractionmass=2.58*perCent);



 // ------- defining Al backing

 a = 26.981539*g/mole;
```

```
  density = 2.70*g/cm3;
  G4Material* Al = new G4Material(name="Al", z=13., a, density);

  //----------defining Air
  density = 1.29*mg/cm3;
  G4Material *Air = new G4Material(name="Air ",density,ncomponents=2);
  Air->AddElement(elO, fractionmass=30.0*perCent);
  Air->AddElement(elN, fractionmass=70.0*perCent);

  //-----------defining source 235U

  density = 18.97*g/cm3;
  G4Material* U235 = new G4Material(name="U235", density,ncomponents=1);
  U235->AddElement(elenrichedU,fractionmass=100.0*perCent);

   //----------defining Kapton

  density = 1.43*g/cm3;
  G4Material *Kapton = new G4Material(name="Kapton",density,ncomponents=4);
  Kapton->AddElement(elH, fractionmass=2.73*perCent);
  Kapton->AddElement(elC, fractionmass=72.13*perCent);
  Kapton->AddElement(elN, fractionmass=7.65*perCent);
  Kapton->AddElement(elO, fractionmass=17.49*perCent);

  //-------------defining Copper

  a = 63.546*g/mole;
  density = 8.94*g/cm3;
  G4Material* Copper = new G4Material(name="Copper", z=29., a, density);



  G4cout << "\n\n ####-----------------------------------------------------#### \n";
  G4cout << "\n\t\t#### List of isotopes used #### \n";
//  G4cout << *(G4Isotope::GetIsotopeTable());
  G4cout << "\n\n\n\n\t\t #### List of elements used #### \n";
  G4cout << *(G4Element::GetElementTable());
  G4cout << "\n\n\n\n\t\t #### List of materials used #### \n";
  G4cout << *(G4Material::GetMaterialTable());
  G4cout << "\n\n ####-----------------------------------------------------#### \n";


  //================================ volumes ===========================//
//----------------------------- beam line along z axis

  G4double startFi = 0.0*deg;
  G4double endFi = 360.0*deg;

//----------------------------- world volume

  G4double World_hx = 100./2.*cm;
```

```
G4double World_hy = 100./2.*cm;
G4double World_hz = 100./2.*cm;

G4Box *World_box
  = new G4Box("World_box",World_hx,World_hy,World_hz);

G4LogicalVolume *World_log
  = new G4LogicalVolume(World_box,Air,"World_log",0,0,0);

G4VPhysicalVolume *World_phys
  = new G4PVPlacement(0,G4ThreeVector(),World_log,"World",0,false,0);

//--------------- Al chamber

G4double CAlthickness = 1.5*cm;
G4double CAlOutR = 15.0*cm;
G4double CAlInR = 0.0*cm;
G4double CAlHalf = 30./2.*cm;

G4Tubs *CAl_tube
  = new G4Tubs("CAl_tube",CAlInR,CAlOutR,CAlHalf,
                                    startFi,endFi);

G4LogicalVolume *CAl_log
  = new G4LogicalVolume(CAl_tube,Al,"CAl_log",0,0,0);

G4double Pos_x = 0.0*cm;
G4double Pos_y = 0.0*cm;
G4double Pos_z = 0.0*cm;
G4VPhysicalVolume *CAlTube_phys
  = new G4PVPlacement(0,
        G4ThreeVector(Pos_x,Pos_y,Pos_z),
        CAl_log,"CAlTube",World_log,false,0);

 //--------------Gas volume

G4double GasInR = 0.0*cm;
G4double GasOutR = CAlOutR - CAlthickness;
G4double GasHalf = CAlHalf - CAlthickness;
G4Tubs *Gas_tube
  = new G4Tubs("Gas_tube", GasInR, GasOutR, GasHalf, startFi, endFi);

G4LogicalVolume *Gas_log
  = new G4LogicalVolume(Gas_tube, MegasGas, "Gas_log", 0,0,0);

Pos_x = 0.0*cm;
Pos_y = 0.0*cm;
Pos_z = 0.0*cm;
G4VPhysicalVolume *Gas_phys
  = new G4PVPlacement(0,
                                G4ThreeVector(Pos_x, Pos_y, Pos_z),
                                  Gas_log, "GasTube", CAl_log, false,0);

//--------------KUGas volume

  G4double KUGasInR = 0.0*cm;
  G4double KUGasOutR = 15/2.*cm;
G4double KUGasHalf = CAlthickness/2.;
  G4Tubs *KUGas_tube
   = new G4Tubs("KUGas_tube",KUGasInR, KUGasOutR, KUGasHalf, startFi, endFi);

  G4LogicalVolume *KUGas_log
   = new G4LogicalVolume(KUGas_tube, MegasGas, "KUGas_log", 0,0,0);

  Pos_x = 0.0*cm;
  Pos_y = 0.0*cm;
  Pos_z = CAlHalf - CAlthickness/2.;
  G4VPhysicalVolume *KUGas_phys
   = new G4PVPlacement(0,
                                  G4ThreeVector(Pos_x, Pos_y, Pos_z),
                                  KUGas_log, "KUGasTube", CAl_log, false,0);
  //--------------- Kapton up

  G4double KapUOutR = KUGasOutR;
  G4double KapUInR = 0.0*cm;
  G4double KapUHalf = 25./2.*um;

  G4Tubs *KapU_tube
   = new G4Tubs("KapU_tube",KapUInR,KapUOutR,KapUHalf,
                                    startFi,endFi);

  G4LogicalVolume *KapU_log
   = new G4LogicalVolume(KapU_tube,Kapton,"KapU_log",0,0,0);

  Pos_x = 0.0*cm;
  Pos_y = 0.0*cm;
  Pos_z = KUGasHalf- KapUHalf;
  G4VPhysicalVolume *KapUTube_phys
   = new G4PVPlacement(0,
          G4ThreeVector(Pos_x,Pos_y,Pos_z),
          KapU_log,"KapUTube",KUGas_log,false,0);

  //--------------KDGas volume

  G4double KDGasInR = 0.0*cm;
  G4double KDGasOutR = 15/2.*cm;
G4double KDGasHalf = CAlthickness/2.;
  G4Tubs *KDGas_tube
   = new G4Tubs("KDGas_tube",KDGasInR, KDGasOutR, KDGasHalf, startFi, endFi);

  G4LogicalVolume *KDGas_log
   = new G4LogicalVolume(KDGas_tube, MegasGas, "KDGas_log", 0,0,0);
```

```
Pos_x = 0.0*cm;
Pos_y = 0.0*cm;
Pos_z = -(CAlHalf - CAlthickness/2.);
G4VPhysicalVolume *KDGas_phys
  = new G4PVPlacement(0,
                                G4ThreeVector(Pos_x, Pos_y, Pos_z),
                                KDGas_log, "KDGasTube", CAl_log, false,0);
//--------------- Kapton down

G4double KapDOutR = KDGasOutR;
G4double KapDInR = 0.0*cm;
G4double KapDHalf = 25/2.*um;

G4Tubs *KapD_tube
  = new G4Tubs("KapD_tube",KapDInR,KapDOutR,KapDHalf,
                                   startFi,endFi);

G4LogicalVolume *KapD_log
  = new G4LogicalVolume(KapD_tube,Kapton,"KapD_log",0,0,0);

Pos_x = 0.0*cm;
Pos_y = 0.0*cm;
Pos_z = -(KDGasHalf- KapDHalf);
G4VPhysicalVolume *KapDTube_phys
  = new G4PVPlacement(0,
         G4ThreeVector(Pos_x,Pos_y,Pos_z),
         KapD_log,"KapDTube",KDGas_log,false,0);


 //--------------Al backing volume
G4double AlBackthickness = 25.0*um;
G4double AlBackInR = 0.0*cm;
G4double AlBackOutR = GasOutR;
G4double AlBackHalf = AlBackthickness/2.;
G4Tubs *AlBack_tube
  = new G4Tubs("AlBack_tube", AlBackInR, AlBackOutR, AlBackHalf, startFi, endFi);

G4LogicalVolume *AlBack_log
  = new G4LogicalVolume(AlBack_tube, Al, "AlBack_log", 0,0,0);

Pos_x = 0.0*cm;
Pos_y = 0.0*cm;
Pos_z = 0.0*cm;
G4VPhysicalVolume *AlBack_phys
  = new G4PVPlacement(0,
                                G4ThreeVector(Pos_x, Pos_y, Pos_z),
                                AlBack_log, "AlBackTube", Gas_log, false,0);


//--------------Megas volume
```

```
 G4double MegasInR = 0.0*cm;
 G4double MegasOutR = 9.5/2.*cm;
 G4double MegasHalf = 7./2.*mm;
 G4Tubs *Megas_tube
   = new G4Tubs("Megas_tube", MegasInR, MegasOutR, MegasHalf, startFi, endFi);

 G4LogicalVolume *Megas_log
   = new G4LogicalVolume(Megas_tube, MegasGas, "Megas_log", 0,0,0);

 Pos_x = 0.0*cm;
 Pos_y = 0.0*cm;
 Pos_z = +(AlBackthickness/2. + MegasHalf);
 G4VPhysicalVolume *Megas_phys
   = new G4PVPlacement(0,
                       G4ThreeVector(Pos_x, Pos_y, Pos_z),
                       Megas_log, "MegasTube", Gas_log, false,0);

 //------------------ Copper(Micromesh)
 G4double Copthickness = 5.*um;
 G4double CopInR = 0.0*cm;
G4double CopOutR = GasOutR;
G4double CopHalf = Copthickness/2.;
 G4Tubs *Cop_tube
   = new G4Tubs("Cop_tube", CopInR, CopOutR, CopHalf,
                         startFi,endFi);

 G4LogicalVolume *Cop_log
   = new G4LogicalVolume(Cop_tube,Copper,"Cop_log",0,0,0);

 Pos_x = 0.0*cm;
 Pos_y = 0.0*cm;
 Pos_z = +(AlBackthickness/2. + MegasHalf*2. + CopHalf);
 G4VPhysicalVolume *Cop_phys
   = new G4PVPlacement(0,
         G4ThreeVector(Pos_x,Pos_y,Pos_z),
         Cop_log,"CopTube",Gas_log,false,0);

 //-------------235U volume
 G4double U235thickness = 5.550334*nm;
 G4double U235InR = 0.0*mm;
 G4double U235OutR = 6./2.*cm;
 G4double U235Half = U235thickness/2.;
 G4Tubs *U235_tube
   = new G4Tubs("U235_tube", U235InR, U235OutR, U235Half, startFi, endFi);

 G4LogicalVolume *U235_log
   = new G4LogicalVolume(U235_tube,U235, "U235_log", 0,0,0);

 Pos_x = 0.0*cm;
 Pos_y = 0.0*cm;
 Pos_z = -MegasHalf + U235thickness/2.;
 G4VPhysicalVolume *U235_phys
   = new G4PVPlacement(0,
                               G4ThreeVector(Pos_x, Pos_y, Pos_z),
                               U235_log, "U235Tube", Megas_log, false,0);

//========================= Visualization attributes ====================//

 World_log->SetVisAttributes (G4VisAttributes::Invisible);

 G4VisAttributes *CAlTubeAttr = new G4VisAttributes(G4Colour(1.,0.,0.));
 //red
G4VisAttributes *GasTubeAttr = new G4VisAttributes(G4Colour(0.,1.,0.));    //green
G4VisAttributes *KUGasTubeAttr = new G4VisAttributes(G4Colour(0.,0.,1.));        //blue
G4VisAttributes *KapUTubeAttr = new G4VisAttributes(G4Colour(0.8,0.8,0.8));    //grey
G4VisAttributes *KDGasTubeAttr = new G4VisAttributes(G4Colour(0.,0.4,0.3));   //tale
G4VisAttributes *KapDTubeAttr = new G4VisAttributes(G4Colour(0.3,0.3,0.3));  //light gray
G4VisAttributes *AlBackTubeAttr = new G4VisAttributes(G4Colour(0.0,0.0,0.0));  //black
G4VisAttributes *MegasTubeAttr = new G4VisAttributes(G4Colour(0.0,1.0,1.0));  //cyan
G4VisAttributes *CopTubeAttr = new G4VisAttributes(G4Colour(1.0,0.0,1.0));  //magenta
G4VisAttributes *U235TubeAttr = new G4VisAttributes(G4Colour(1.0,1.0,1.0));  //yellow


CAlTubeAttr->SetVisibility(true);
CAlTubeAttr->SetForceWireframe(true);
CAl_log->SetVisAttributes(CAlTubeAttr);

GasTubeAttr->SetVisibility(true);
GasTubeAttr->SetForceWireframe(true);
Gas_log->SetVisAttributes(GasTubeAttr);

KUGasTubeAttr->SetVisibility(true);
 KUGasTubeAttr->SetForceWireframe(true);
 KUGas_log->SetVisAttributes(KUGasTubeAttr);

 KapUTubeAttr->SetVisibility(true);
 KapUTubeAttr->SetForceWireframe(true);
 KapU_log->SetVisAttributes(KapUTubeAttr);

 KDGasTubeAttr->SetVisibility(true);
 KDGasTubeAttr->SetForceWireframe(true);
 KDGas_log->SetVisAttributes(KDGasTubeAttr);

 KapDTubeAttr->SetVisibility(true);
 KapDTubeAttr->SetForceWireframe(true);
 KapD_log->SetVisAttributes(KapDTubeAttr);

 AlBackTubeAttr->SetVisibility(true);
 AlBackTubeAttr->SetForceWireframe(true);
 AlBack_log->SetVisAttributes(AlBackTubeAttr);
```

```
  MegasTubeAttr->SetVisibility(true);
  MegasTubeAttr->SetForceSolid(true);
  Megas_log->SetVisAttributes(MegasTubeAttr);

  CopTubeAttr->SetVisibility(true);
  CopTubeAttr->SetForceWireframe(true);
  Cop_log->SetVisAttributes(CopTubeAttr);

  U235TubeAttr->SetVisibility(true);
  U235TubeAttr->SetForceWireframe(true);
  U235_log->SetVisAttributes(U235TubeAttr);


  return World_phys;
}

//------------------------------------------------------------------------------------------------------------------
```

For a ^{235}U sample the following parts in the code should be replaced.

```
G4Element* elenrichedU = new G4Element("enrichedU", "U" ,ncomponents=1);
  elenrichedU->AddIsotope(isoU235, fractionmass=100.*perCent);

density = 18.97*g/cm3;
  G4Material* U235 = new G4Material(name="U235", density,ncomponents=1);
  U235→AddElement(elenrichedU,fractionmass=100.0*perCent);

 G4double U235thickness = 5.550334*nm;
  G4double U235InR = 0.0*mm;
  G4double U235OutR = 6./2.*cm;
  G4double U235Half = U235thickness/2.;
  G4Tubs *U235_tube
    = new G4Tubs("U235_tube", U235InR, U235OutR, U235Half, startFi, endFi);

  G4LogicalVolume *U235_log
    = new G4LogicalVolume(U235_tube,U235, "U235_log", 0,0,0);

  Pos_x = 0.0*cm;
  Pos_y = 0.0*cm;
  Pos_z = -MegasHalf + U235thickness/2.;
  G4VPhysicalVolume *U235_phys
    = new G4PVPlacement(0,
                        G4ThreeVector(Pos_x, Pos_y, Pos_z),
                        U235_log, "U235Tube", Megas_log, false,0);
//------------------------------------------------------------------------------------------------------------------
```

XriDetectorConstructor.hh

```
//------------------------------------------------------------------------------------------------------------------
#ifndef XriDetectorConstruction_H
#define XriDetectorConstruction_H 1

#include "globals.hh"
```

```
class G4VPhysicalVolume;

#include "G4VUserDetectorConstruction.hh"

class XriDetectorConstruction : public G4VUserDetectorConstruction
{
  public:
    XriDetectorConstruction();
    ~XriDetectorConstruction();

  public:
     G4VPhysicalVolume* Construct();

};

#endif
//-----------------------------------------------------------------------------------------------------------------------
```

XriEventAction.cc

```
//-----------------------------------------------------------------------------------------------------------------------
#include "XriEventAction.hh"
#include "G4Event.hh"
#include "G4EventManager.hh"
#include "G4TrajectoryContainer.hh"
#include "G4Trajectory.hh"
#include "G4VVisManager.hh"
#include "G4UImanager.hh"
#include "G4ios.hh"
#include <stdio.h>
#include "G4SystemOfUnits.hh"
#include "G4PhysicalConstants.hh"
#include "vars.h"

using namespace std;

ofstream outfileEvnt;

XriEventAction::XriEventAction()
{}

XriEventAction::~XriEventAction()
{}

void XriEventAction::BeginOfEventAction(const G4Event* evt)
{
// resetting energy accumulator...
   Depos = 0.0*MeV;
}

void XriEventAction::EndOfEventAction(const G4Event* evt)
{
  if ((evt->GetEventID()+1) % 100000 == 0)
```

```
    G4cout << ">>> Event " << evt->GetEventID()+1 << G4endl;

evtNo = evt->GetEventID();
FILE *opf1= fopen("out.dat", "a");

 if (Depos>0)
    {
      outfileEvnt.open ("outdetailed.dat", ios::app);

      // output detailed information about event
      outfileEvnt<<evt->GetEventID()<<"\t"<<Depos/MeV<<"\t"<<heavylight[evt-
>GetEventID()]<<"\t"<<ffZ[evt->GetEventID()]<<"\t"<<ffA[evt->GetEventID()]<<G4endl;

   }
   fclose(opf1);
   outfileEvnt.close();
}

//--------------------------------------------------------------------------------------------------------------------
```

XriEventAction.hh

```
//--------------------------------------------------------------------------------------------------------------------
#ifndef XriEventAction_h
#define XriEventAction_h 1

#include "G4UserEventAction.hh"
#include "globals.hh"

extern G4double Depos;  // 19-1-2006
extern G4double evtNo;

class G4Event;

class XriEventAction : public G4UserEventAction
{
  public:
    XriEventAction();
    ~XriEventAction();

  public:
    void BeginOfEventAction(const G4Event*);
    void EndOfEventAction(const G4Event*);

  private:

};

#endif
//--------------------------------------------------------------------------------------------------------------------
```

XriPrimaryGeneratorAction.cc

```
//--------------------------------------------------------------------------------------------------------------------
#include "XriPrimaryGeneratorAction.hh"
```

```
#include "G4Event.hh"
#include "G4ChargedGeantino.hh"
#include "G4IonTable.hh"
#include "G4ParticleGun.hh"
#include "G4ParticleTable.hh"
#include "G4ParticleDefinition.hh"
#include "G4GenericMessenger.hh"
#include "G4SystemOfUnits.hh"
#include "G4PhysicalConstants.hh"
#include "G4GenericMessenger.hh"
#include "G4UImanager.hh"
#include "globals.hh"
#include "Randomize.hh"
#include <stdio.h>
#include <cmath>
#include <iostream>
#include <fstream>
#include <string>
#include "vars.h"

using namespace std;

ofstream outfile1;

XriPrimaryGeneratorAction::XriPrimaryGeneratorAction()
//rndmFlag("on")
  : G4VUserPrimaryGeneratorAction(),
   particleGun(0)
{
  G4int n_particle = 1;
  particleGun = new G4ParticleGun(n_particle);

  // default particle kinematic
  G4ParticleTable* particleTable = G4ParticleTable::GetParticleTable();
  G4ParticleDefinition* particle
                  = particleTable->FindParticle("chargedgeantino");
  particleGun->SetParticleDefinition(particle);
  particleGun->SetParticlePosition(G4ThreeVector(0.,0.,0.));
  particleGun->SetParticleEnergy(1*eV);
  particleGun->SetParticleMomentumDirection(G4ThreeVector(1.,0.,0.));
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

XriPrimaryGeneratorAction::~XriPrimaryGeneratorAction()
{
  delete particleGun;
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
```

```
void XriPrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent)
{
  G4int countevpicked = 0.;
  for(int i=0; i<GEF_SIZE; i++){
    countevpicked += eventspicked[i];
  }
  if((countevpicked-1) == GEF_SIZE){
     G4cout<<"All events already chosen!!"<<endl;
  }
  G4cout<<(countevpicked - 1)<<" is the eventID of chosen primary"<<endl;

   //store if ff is heavy or light
  heavylight[(countevpicked-1)]=(totalevents[rndi].ff); //[0] eventID =0

  //eventID=0, countevpicked=1
  //store id of primary ff
  ffZ[(countevpicked-1)]=(totalevents[rndi].Z);
  ffA[(countevpicked-1)]=(totalevents[rndi].A);

  G4cout<<" heavy or light= "<<heavylight[(countevpicked-1)]<<" "<<totalevents[rndi].ff<<"
Z="<<ffZ[(countevpicked-1)]<<" A="<<ffA[(countevpicked-1)]<<endl;


  // output chosen event

outfile1<<totalevents[rndi].ffid<<"\t"<<totalevents[rndi].Z<<"\t"<<totalevents[rndi].A<<"\t"
<<totalevents[rndi].Ekin<<"\t"<<totalevents[rndi].px0<<"\t"<<totalevents[rndi].py0<<"\t"<<
totalevents[rndi].pz0<<"\t"<<totalevents[rndi].x0<<"\t"<<totalevents[rndi].y0<<"\t"<<totale
vents[rndi].z0<<"\t"<<totalevents[rndi].ff<<"\t"<<endl;

G4double ionCharge   = 0.*eplus;
G4double excitEnergy = 0.*keV;

  // particle id
  G4ParticleDefinition* particle = particleGun->GetParticleDefinition();
  if (particle == G4ChargedGeantino::ChargedGeantino()) {
    G4ParticleDefinition*          ion          =          G4IonTable::GetIonTable()-
>GetIon(totalevents[rndi].Z,totalevents[rndi].A,excitEnergy);
    particleGun->SetParticleDefinition(ion);
    particleGun->SetParticleCharge(ionCharge);
    }

  particleGun->SetParticleEnergy(totalevents[rndi].Ekin*MeV);
  particleGun-
>SetParticleMomentumDirection(G4ThreeVector(totalevents[rndi].px0,totalevents[rndi].py0
,totalevents[rndi].pz0));
  particleGun-
>SetParticlePosition(G4ThreeVector(totalevents[rndi].x0*cm,totalevents[rndi].y0*cm,totale
vents[rndi].z0*cm));

  //create vertex
  particleGun->GeneratePrimaryVertex(anEvent);  //to generate an event

  }
//-----------------------------------------------------------------------------------------------------------------------
```

XriPrimaryGeneratorAction.hh

```
//-----------------------------------------------------------------------------------------------------------------------
#ifndef XriPrimaryGeneratorAction_h
#define XriPrimaryGeneratorAction_h 1

#include "G4VUserPrimaryGeneratorAction.hh"
#include "globals.hh"

class G4ParticleGun;
class G4Event;

class XriPrimaryGeneratorAction : public G4VUserPrimaryGeneratorAction
{
  public:
    XriPrimaryGeneratorAction();
    ~XriPrimaryGeneratorAction();

 public:
    void GeneratePrimaries(G4Event *anEvent);

  private:
    G4ParticleGun* particleGun;
};

#endif
//-----------------------------------------------------------------------------------------------------------------------
```

XriRunAction.cc

```
//-----------------------------------------------------------------------------------------------------------------------
#include "XriRunAction.hh"
#include "G4Run.hh"
#include "G4UImanager.hh"
#include "G4VVisManager.hh"
#include "G4ios.hh"
#include <stdio.h>
#include <iostream>
#include <fstream>
#include "vars.h"

using namespace std;

RunActionXri::RunActionXri()
{
  runIDcounter = 0;
}

RunActionXri::~RunActionXri()
{}
```

```
void RunActionXri::BeginOfRunAction(const G4Run* aRun)
{
 ((G4Run *)(aRun))->SetRunID(runIDcounter++);

 G4cout << "### Run " << aRun->GetRunID() << " start." << G4endl;

 //reset eventspicked
 for(int i=0; i<GEF_SIZE; i++){
   eventspicked[i]=0.;  //history of picked events is reset before each run
  }
 // check if eventspicked is reset before each execution of programme
 G4double rres =0.;
 for(int i=0; i<GEF_SIZE; i++){
   if(eventspicked[i]){ //an event is not reset
     rres ++;}
 }

 if(rres){
   G4cout<<"run.cc event is not reset!!!"<<endl;
  }
 else{
   G4cout<<"run.cc events all reset!!!"<<endl;
 }

   srand(time(0));
   FILE *opf1 = fopen("out.dat","w");
   fclose(opf1);

   ofstream stepout;
   stepout.open("stepping.dat");

   G4UImanager* UI = G4UImanager::GetUIpointer();
   UI->ApplyCommand("/tracking/storeTrajectory 1");

   G4VVisManager* pVVisManager = G4VVisManager::GetConcreteInstance();
   if(pVVisManager)
     {
          UI->ApplyCommand("/vis~/clear/view");
          UI->ApplyCommand("/vis~/draw/current");
     }
}

void RunActionXri::EndOfRunAction(const G4Run* aRun)
{
 G4cout << "@#@#@# Run " << aRun->GetRunID() << " ended." << G4endl;
 G4VVisManager* pVVisManager = G4VVisManager::GetConcreteInstance();

 if(pVVisManager)
 {
   G4UImanager::GetUIpointer()->ApplyCommand("/vis~/show/view");
  }
}
//-------------------------------------------------------------------------------------------------------------------
```

XriRunAction.hh

```
//-------------------------------------------------------------------------------------------------------------------
#ifndef RunActionXri_h
#define RunActionXri_h 1

#include "G4UserRunAction.hh"
#include "globals.hh"

class G4Run;

class RunActionXri : public G4UserRunAction
{
  public:
    RunActionXri();
    virtual ~RunActionXri();

  public:
    virtual void BeginOfRunAction(const G4Run* aRun);
  virtual void EndOfRunAction(const G4Run* aRun);

  private:
    G4int runIDcounter;
};

#endif
//-------------------------------------------------------------------------------------------------------------------
```

XriSteppingAction.cc

```
//-------------------------------------------------------------------------------------------------------------------
#include <stdio.h>
#include<cstring>
#include "XriSteppingAction.hh"
#include "XriDetectorConstruction.hh"
#include "G4SteppingManager.hh"
#include "G4Track.hh"
#include "G4SystemOfUnits.hh"
#include "G4PhysicalConstants.hh"
#include <iostream>
#include <fstream>
#include "vars.h"

using namespace std;

XriSteppingAction::XriSteppingAction(XriDetectorConstruction*
myDC):myDetector(myDC)//, eventAction(myEA)
{ }
ofstream stepout("stepping.dat", ios::out | ios::app);
void XriSteppingAction::UserSteppingAction(const G4Step* aStep)
{
 const       G4VPhysicalVolume*       currentVolume1       =       aStep->GetPreStepPoint()->
GetPhysicalVolume();

 // collect the energy deposited in the absorbers
 G4Track *aTrack = aStep->GetTrack();

        if (currentVolume1->GetName() == "MegasTube")
{

  if(aStep→GetTotalEnergyDeposit()>0){
      Depos += aStep->GetTotalEnergyDeposit();
  I   f(aStep->GetTrack()->GetGlobalTime()<(1E-07)){
      G4cout<<"Too          small          time!!!          <1E-07"<<"\t"<<aStep->GetTrack()-
>GetGlobalTime()/s<<G4endl;}
  }
 }
}
```

```
//-------------------------------------------------------------------------------------------------------------------
```

XriSteppingAction.hh

```
//-------------------------------------------------------------------------------------------------------------------
#ifndef XriSteppingAction_h
#define XriSteppingAction_h 1

#include "G4UserSteppingAction.hh"
#include "globals.hh"

extern G4double Depos;   // 19-1-2006
extern G4double evtNo;

class XriDetectorConstruction;
class G4Track;

class XriSteppingAction : public G4UserSteppingAction
{
  public:
    XriSteppingAction(XriDetectorConstruction* myDC);
    virtual ~XriSteppingAction(){};

    virtual void UserSteppingAction(const G4Step*);

  private:
    XriDetectorConstruction* myDetector;
};

#endif
//-------------------------------------------------------------------------------------------------------------------
```

References

[1] N. Bohr and J. A. Wheeler, The Mechanism of Nuclear Fission, Phys. Rev. 56, (1939) pp. 426–450, DOI: 10.1103/PhysRev.56.426

[2] S. Bjørnholm and J. E. Lynn, The double-humped fission barrier, Rev. Mod. Phys. 52, (1980) pp. 725–931, DOI: 10.1103/RevModPhys.52.7254

[3] V. Strutinsky, Shell effects in nuclear masses and deformation energies, Nucl. Phys. A 95(2), (1967) pp. 420 – 442, DOI: 10.1016/0375-9474(67)90510-6

[4] J. R. Huizenga, Nuclear fission revisited, Science, 168:1405–1413, 1979

[5] "Introductory Nuclear Physics", K. S. Krane, John Wiley & Sons.

[6]Y.Nagame and H. Nakahara, Two-mode fission- experimental verification and characterization of two fission-modes, Radiochim. Acta 100, 605–614 (2012),DOI 10.1524/ract.2012.1968

[7] D. Madland, Nuclear Physics A, 772, 113-137, 2006

[8] F. Gunsing, et al., The measurement programme at the neutron time-of-flight facility n-TOF at CERN, Eur. Phys. J., Web of Conferences 146. DOI:10.1051/epjconf/201714611002.

[9] F. Gunsing, et al., Nuclear data activities at the n TOF facility at CERN, Eur. Phys. J. Plus 131 (10) (2016) 371. DOI:10.1140/epjp/i2016-16371-4.

[10] M. Sabate-Gilarte, et al., High-accuracy determination of the neutron flux in the new experimental area n TOF-EAR2 at CERN, Eur. Phys. J. A 53 (10). DOI:10.1140/epja/i2017-12392-4.

[11] C. Weiss, et al., The new vertical neutron beam line at the CERN n TOF facility design and outlook on the performance, Nucl. Instrum. Meth. A 799 (2015) 90 – 98. DOI:10.1016/j.nima.2015.07.027.

[12] Y. Giomataris et al., MICROMEGAS: a high-granularity position-sensitive gaseous detector for high particle-flux environments, Nucl. Instrum. Meth. A 376(1), (1996) pp. 29 – 35, DOI: 10.1016/0168-9002(96)00175-1

[13] Y. Giomataris, Development and prospects of the new gaseous detector “Micromegas”, Nucl. Instrum. Meth. A 419(2–3), (1998) pp. 239 – 250, DOI: 10.1016/S0168-9002(98)00865-1

[14] I. Giomataris, MICROMEGAS: results and prospects, ICFA Instrum. Bul. 19, www.slac.stanford.edu/pubs/icfa/fall99/paper1/paper1a.html //cf4

[15] F. Sauli, Micro-pa□ern gas detectors, Nucl. Instrum. Meth. A 477(1–3), (2002) pp. 1 – 7, DOI: 10.1016/S0168-9002(01)01903-9, 5th Int. Conf. on Position-Sensitive Detectors

[16] L. Shekhtman, Micro-pa�ern gaseous detectors, Nucl. Instrum. Meth. A 494(1–3), (2002) pp. 128 – 141, DOI: 10.1016/S0168-9002(02)01456-0, Proceedings of the 8th International Conference on Instrumentation for Colliding Beam Physics

[17] S Andriamonje et. al.Development and performance of Microbulk Micromegas detectors S Andriamonje et al 2010 JINST 5 P02001

[18] K.-H. Schmidt et al., Nucl. Data Sheets 131 (2016) 107

[19] B. Jurado and K. -H. Schmidt, Status of the general description of fission observables by the GEF code, CENBG, CNRS/IN2P3, Chemin du Solarium, B. P. 120, 33175 Gradignan, France

[20] http://www.khs-erzhausen.de/GEF.html

[21] U. Mosel, H. W. Schmitt, Potential energy surfaces for heavy nuclei in the two-center model, Nucl. Phys. A 165, 73 (1971)

[22] K.-H. Schmidt, A. Kelic, M. V. Ricciardi, Experimental evidence Europh. Lett. 83, 32001 (2008)

[23] B. D. Wilkins, Scission-point model of nuclear fission based on deformed-shell effects, E. P. Steinberg, R. R. Chasman, Phys. Rev. C **14,** 1832 (1976)

[24] K.-H. Schmidt, B. Jurado, Entropy Driven Excitation Energy Sorting in Superfluid Fission Dynamics, Phys. Rev. Lett. 104, 212501 (2010)

[25] Geant4 Collaboration (S. Agostinelli et al.), Nucl. Instrum. Meth. Phys. Res. A 506, 250-303 (2003)

[26] Geant4 Collaboration (J. Allison et al.), IEEE Trans. Nucl. Sci., 53, 270-278 (2006)

[27] Guatelli, S, Cutajar, D, Rosenfeld, A B, Oborn, B, and Centre for Medical Radiation Physics, Introduction to the Geant4 Simulation toolkit. United States: N. p., 2011. Web DOI: 10.1063/1.3576174.

[28] A. Stanculescu, Annals of Nuclear Energy 62, 607-612, (2013)

[29] Generation-IV International Forum, www.gen-4.org/

[30] https://www.iaea.org/sites/default/files/16/11/npparisagreement.Pdf

[31] A.J.M. Plompen, Nucl. Data Sheets 118, 78-84 (2014)

[32] E. Dupont et al., these proceedings, NEA Nuclear Data High Priority Request List, www.nea.fr/html/dbdata/hprl

[33] M. Salvatores et al., OECD/NEA WPEC, Subgroup 26 Final Report, www.oecdnea.org/science/wpec/volume26/volume26.pdf

[34] Eleme, & et al. (2019). Cross section measurement of 241Am(n,f) reaction at the Experimental Area 2 of the n_TOF facility at CERN: First Results. HNPS Proceedings, 27, 189-194.

[35] U. Abbondanno et al., Nucl. Instrum. Meth. A 538,692-702 (2005)

[36] R. Kirk, D. Othmer, Kirk-Othmer Encyclopedia of Chemical Technology, Vol 1, 4th edition John Wiley and Sons, 2004

[37] https://www.atsdr.cdc.gov/ToxProfiles/tp156.pdf

[38] G. Knoll, Radiation Detection and Measurement, Wiley, 4th ed. (2010)