Die approbierte Originalversion dieser Diplom-/Masterarbeit ist in der Hauptbibliothek der Technischen Universität Wien aufgestellt und zugänglich.

http://www.ub.tuwien.ac.at

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology.

http://www.ub.tuwien.ac.at/eng

Unterschrift Betreuer

The calculation of differential and double differential cross sections using GATE simulations

ausgeführt am Atominstitut
der Technischen Universität Wien

unter der Anleitung von
Univ.Prof. civ.ing. tekn.lic. tekn.dr. Lembit Sihver und
Univ.Ass. Dipl.-Ing. Dr.techn. Albert Hirtl

durch

Alexander Burker, Matr.-Nr. 0826360, 066 453

Steigenteschgasse 22, 1220 Wien, Österreich

16. Oktober, 2016

Unterschrift Student

Zusammenfassung

In den letzten Jahrzehnten hat sich die Strahlentherapie zu einer wichtigen Behandlungsmethode, besonders für Patienten mit tief liegenden oder strahlenresistenten Tumoren, entwickelt. Dank ihrer vorteilhaften Dosisverteilung kann mittels der Nutzung von Protonen oder leichten Ionen gesundes Gewebe besser geschont werden als mit Photonen oder Elektronen. Elektronen haben eine niedrige Eindringtiefe und Photonen übertragen ihre größte Dosis kurz nach der Oberfläche. Im Gegenzug dazu haben geladene Teilchen eine Dosisverteilung mit einem Maximum – dem Bragg peak – in einer kontrollierbaren Tiefe im Gewebe, die von der Energie des Teilchens abhängt. Außerdem ist es möglich anfällige Organe zu schützen, da geladene Teilchen vor dem Bragg peak weniger Energie deponieren.

Nutzt man anstelle von Protonen schwere Ionen, so erhöht sich die biologische Wirkung im Ziel während die Projektilstreuung deutlich abnimmt. Allerdings werden durch nukleare Reaktionen zwischen schweren Ionen und Atomen leichtere Fragmente erzeugt, die dann tiefer eindringen können als Primärionen und zu einer Dosis nach dem Bragg peak, einem *Fragmentation Tail*, führen. Die Erzeugung von sekundären Teilchen hängt vom totalen Wechselwirkungsquerschnitt ab, der wiederum von der Art des Projektils, dessen Energie und der Art des Atoms abhängt. Um die Dosis während einer Therapieplanung abschätzen zu können ist es also notwendig die differenziellen und doppelt differenziellen Wirkungsquerschnitte genau zu kennen. Leider ist zu diesem Zeitpunkt nur eine ungenügende Menge an experimentellen Messdaten verfügbar.

Bis mehr Messungen verfügbar sind, können *Monte Carlo* Simulationen genutzt werden um die Daten abzuschätzen. Es sind mehrere Software-Pakete verfügbar, die Teilchentransport und nukleare Reaktionen simulieren können, so zum Beispiel `Geant4`, `FLUKA`, `PHITS`, `MCNP` oder `SHIELD-HIT`. The Aufgabe dieser Arbeit bestand darin Wirkungsquerschnitte aus `Geant4`-Simulationen eines Strahls, der auf verschiedene Proben einwirkt, abzuschätzen. Diese Simulationen wurden für mehrere Iterationen an Projektilenergien und Probenmaterialien durchgeführt. Um die Wirkungsquerschnitte abzuschätzen wurde ein Programm entwickelt, das die Ausgabe der Simulationen durchlaufen kann um Teilchen anhand deren Art, Position und Energie zu suchen und sie zu zählen, dann aus der Zählung die Wirkungsquerschnitte zu berechnen und diese zu speichern. Die Ergebnisse dieser Arbeit wurden mit Daten aus der Literatur verglichen und hier präsentiert. Zusätzlich wurde der Einfluss wichtiger Parameter, wie des *Probenmaterials*, des *Detektionswinkels* und der *Fragmente*, auf die Übereinstimmung mit den Daten besprochen.

Abstract

During the last decades *ion beam therapy* has become an important treatment method in oncology, especially for patients with deep seated and radio-resistant tumors. One established way of treatment is conventional radiation therapy, using photons or electrons. However, due to its favorable depth-dose distribution, healthy tissue can be spared better when using protons or light ions. While electrons have a short range and X-rays deposit most of their energy shortly after the surface, charged particles have a dose distribution with a peak – the *Bragg Peak* – at a certain depth, which depends on the projectile, its kinetic energy and the target properties. Additionally, charged particles transfer little energy to the tissue on their passage through it, so they spare organs outside the targeted region.

In carbon-beam therapy, ions are used instead of protons, due to much less scattering and the higher biological impact at the terminal depth. However, when heavy ions collide with atoms, nuclear reactions can produce lighter ions that might travel further than the primary terminal depth. This contributes to a dose beyond the Bragg Peak and causes a so called fragmentation tail. Production of secondary particles is highly dependent on the total reaction cross section, which in turn, is dependent on the projectile type and energy and the target. In order to calculate the applied dose in treatment planning, knowledge of the differential and double differential cross sections is necessary. Unfortunately, there are only little experimental data available currently.

Until more data can be deduced from measurements it is possible to use *Monte Carlo* simulation software to estimate the desired cross sections. There are several software packages available, that can simulate particle transport and nuclear reactions, like `Geant4`, `FLUKA`, `PHITS`, `MCNP` or `SHIELD-HIT`. The core of this work was to produce differential and double differential cross section estimates from `Geant4` simulations of a carbon particle beam passing through a thin target. These simulations had to be repeated for several iterations of projectile energies, as well as target materials. To obtain the cross sections, an analysis program was developed, that can filter the simulation output for specific particles, based on their type, energy and location. After counting the filtered particles, the program can calculate both types of cross sections and store them on the hard disk. Results of the comparison between simulated data and data from literature will be presented. Additionally, the influence of parameters such as *target material*, the *angle of detection* and the *ejected particle type* on the agreement between data and simulation will be discussed.

Contents

1 Introduction 1
- 1.1 Motivation . . . 4
- 1.2 Expected Results . . . 5

2 Physics Background 7
- 2.1 Reactions . . . 7
- 2.2 Bragg Curve . . . 9
- 2.3 Fragmentation Tail . . . 14
- 2.4 Monte Carlo Method . . . 17
- 2.5 Cross Section Calculation . . . 19

3 Materials and Methods 21
- 3.1 Geant4 . . . 23
  - 3.1.1 Particle Transport . . . 23
  - 3.1.2 Physics Lists and Models . . . 25
  - 3.1.3 Intranuclear Cascade Model . . . 26
- 3.2 GATE . . . 30
- 3.3 HTCondor . . . 33
- 3.4 ROOT . . . 35
- 3.5 Analysis Program . . . 37

4 Results 45
- 4.1 Comparison to Data . . . 48
- 4.2 Influence of Observables . . . 50
  - 4.2.1 Influence of Different Polar Angles . . . 50
  - 4.2.2 Influence of Different Fragments . . . 51
  - 4.2.3 Influence of Different Target Materials . . . 55

5 Discussion and Conclusion 61

Bibliography 63

Appendices **67**
Appendix A MedAustron Images 67
Appendix B GATE Macro Files 69
Appendix C Analysis Program Source Files 70
Appendix D Configuration.xml 71
Appendix E Cross Section Estimates 74

List of Figures

1.1 MedAustron Accelerator Complex, from [8] 3

2.1 Depth dose distributions of x-rays, electrons and protons, from [26]. 10
2.2 Stopping power for protons in a water target, from [9]. 12
2.3 Ranges of different ions in water, taken from [1]. 13
2.4 Spread Out Bragg peak, compared to X-Ray depth dose, from [28]. 14
2.5 Bragg curve for ^{20}Ne ions in water, from [1] 15
2.6 Abrasion-ablation model for nucleus-nucleus collisions, from [9] . . . 16
2.7 On deflection function and differential cross section, from [24] . . . 19

3.1 Workflow for cross section estimate production. 22
3.2 Model map for the `INCL++` based physics lists in `Geant4`, from [25]. 28
3.3 Saxon woods density distribution of `INCL++` targets, from [39] . . . 29
3.4 Geometry of the simulated Experiments. 32
3.5 Example submit file for HTCondor. 34
3.6 Simulation Output - PhaseSpace, shown with `ROOT` *TBrowser* [32]. . 36
3.7 Schematic of the analysis program. 38
3.8 Example XML configuration file 40
3.9 Angle binning configuration for E600 comparison 41

4.1 Double differential cross sections for alpha particles produced at 2.5 ± 2.5 degrees from carbon-carbon collisions at different energies. . . 47
4.2 Differential cross sections for alpha particles produced from carbon-carbon collisions at different energies. 48
4.3 Double differential cross sections for alpha particles produced at 4 ± 1 degrees from carbon-carbon collisions at 1140 MeV. 49
4.4 Differential cross sections for alpha particles produced from carbon-carbon collisions at 1140 MeV. 50

4.5 Influence of different angles ϑ on double differential cross sections. . 51
4.6 Influence of different fragments on double differential cross sections. 53
4.7 Influence of different fragments on differential cross sections. 55
4.8 Influence of different targets on double differential cross sections. . . 57
4.9 Influence of different targets on differential cross sections. 59

A.1 Southern wall of MedAustron's building complex, from [44] 67
A.2 Artists impression of the building, from [6]. 67
A.3 Photograph of parts of the accelerator at MedAustron 68
A.4 View of the robotic table in IR 1 of MedAustron, from [44] 68
E.5 Double differential cross sections for alpha particles produced at 12.5 ± 2.5 degrees from carbon-carbon collisions. 75
E.6 Double differential cross sections for alpha particles produced at 22.5 ± 2.5 degrees from carbon-carbon collisions. 75
E.7 Double differential cross sections for protons produced at 2.5 ± 2.5 degrees from carbon-carbon collisions. 76
E.8 Differential cross sections for protons produced from carbon-carbon collisions. 76
E.9 Double differential cross sections for Li-6 produced at 2.5 ± 2.5 degrees from carbon-carbon collisions. 77
E.10 Differential cross sections for Li-6 produced from carbon-carbon collisions. 77
E.11 Double differential cross sections for alpha particles produced at 2.5 ± 2.5 degrees from carbon-hydrogen collisions. 78
E.12 Differential cross sections for alpha particles produced from carbon-hydrogen collisions. 78
E.13 Double differential cross sections for alpha particles produced at 2.5 ± 2.5 degrees from carbon-sodium collisions. 79
E.14 Differential cross sections for alpha particles produced from carbon-sodium collisions. 79

1. Introduction

A large part of oncology, called radiation therapy, deals with the use of ionizing radiation to treat cancerous tissue. It is the second most frequently used form of therapy, after surgery, and often used in combination with other available methods. The primary goal of radiation therapy is to deliver a large dose of ionizing radiation to a target volume in the tissue, which corresponds to a tumor. Additionally, the surrounding tissue must be spared as much as possible, in order to minimize risks induced by the treatment. It is possible to reduce the dose suffered by healthy tissue, but this risk can never be entirely eliminated. However, depending on the radio-sensitivity of the cancer, the tumor is often more susceptible to the treatment than healthy tissue is, which translates to a net benefit for the patients [1, 2].

There is a common distinction into two prominent forms of radiation therapy. While one method called *Brachytherapy* uses radionuclides that are injected into the patient, the other method called *External Beam Radiation Therapy* relies on external beams to deliver dose. Such a beam could be composed of photons, the use of which is commonly described as *Conventional Radiation Therapy*, or particles like nucleons, electrons or ions. Using protons requires a beam energy in the range of 20 keV to about 25 MeV and ions need energies larger than 70 MeV. An energy of up to 5.2 GeV may be necessary to treat tumors at a depth of 30 cm with carbon ions. Both conventional and ion therapy are able to ionize matter in the tissue. Photons mainly cause ionization due to the photoelectric effect, the Compton-effect and pair production. These effects lead to the production of free radicals that chemically damage the tissue due to their high reactivity. However, the biological effect depends on the amount of oxygen present in the tissue, so a much higher dose can be necessary in tissues that lack oxygen. Even so, using photons and a higher dose may not be enough to properly damage radio-resistant tumors while keeping patients safe, in which case more densely ionizing radiation is needed. In contrast to photons, ions interact mostly due to Coulomb interactions with atomic electrons and nuclear reactions with nuclei. They also produce free radicals and electrons, but their ionization density is much larger, compared to photons, which means that direct ionization has a stronger biological effect when using charged particles [1, 2].

Compared to photon radiation, charged particle based beams show a favorable depth-dose-distribution. Proton and ion beams initially deliver less dose than gamma rays. Instead, the particles slow down until they lose enough energy to be completely stopped. As they slow down, ever more interactions lead to an increasing loss of energy at a terminal depth, which produces a peak in dose, the Bragg peak. So, by using protons instead of photons it is easier to spare vital organs. The benefit of this method can be further increased, by improving the damage done to the target cells. This is possible when using ions instead of protons, because the more massive particles have a much higher ionization density at the end of their range. Due to this, the biological impact is greater. Carbon ions for example are able to ionize both strands of a DNA double helix simultaneously, so a cell's DNA is more likely to suffer double- instead of single-strand breaks, which are more difficult to repair. Another important advantage of heavy ions is that they are traveling in almost straight lines, because of little scattering, making more precise and narrowly focused beams possible [1, 2].

Currently, there are sixty-nine particle therapy centers recognized by the Particle Therapy Co-Operative Group operational worldwide [3]. The first of these facilities, which is still in service, started treating patients in Moscow, 1969, using proton beams. Today, additional 58 centers are available for proton beam therapy, as well as 10 facilities that offer carbon ion beams [3]. Including treatment centers that were previously closed, a total of 137.179 patients were treated with particle therapy by the end of 2014 [4]. At least thirty more centers are currently listed as under construction; these are planned to begin treating patients between 2016 and 2019 [3]. One of the facilities classified as under construction is called MedAustron. It is located in Wiener Neustadt, Austria, scheduled to begin treatment using protons by the end of year 2016 [5].

The MedAustron project is planned to offer proton and carbon ion therapy for patients, as well as clinical and non-clinical research. Typical non-clinical research fields are "Radiobiology", "Medical Radiation Physics", "Development and Testing of Particle Counters", "Proton Scattering" and "Computerized Proton Tomography" [5, 6]. The main field of this work is "Medical Radiation Physics" with a heavy focus on Monte Carlo simulations. Some images of the facility were added to Appendix A. At least 1200 patients per year are going to be scheduled during the operational phase of the facility [6, 7]. Treatment is planned to be carried out on weekdays during the day, while night times and weekends will be used for clinical and non-clinical research to efficiently use the facility for both treatment and scientific research [6]. A synchrotron was installed at the MedAustron facility, in order to enable the acceleration of both carbon ions and protons with only a single machine. During operation, the synchrotron will be injected with ions that will be pre-accelerated by a linear accelerator, from an energy of 8 keV/u at the ion sources to an energy of 7 MeV/u. The synchrotron will be able to further accelerate protons up to an energy of 800 MeV, however only proton energies between 60-250 MeV will be used during medical treatment. Carbon ions are going to be accelerated to energies of 120-400 MeV/u for therapy. A visual representation of

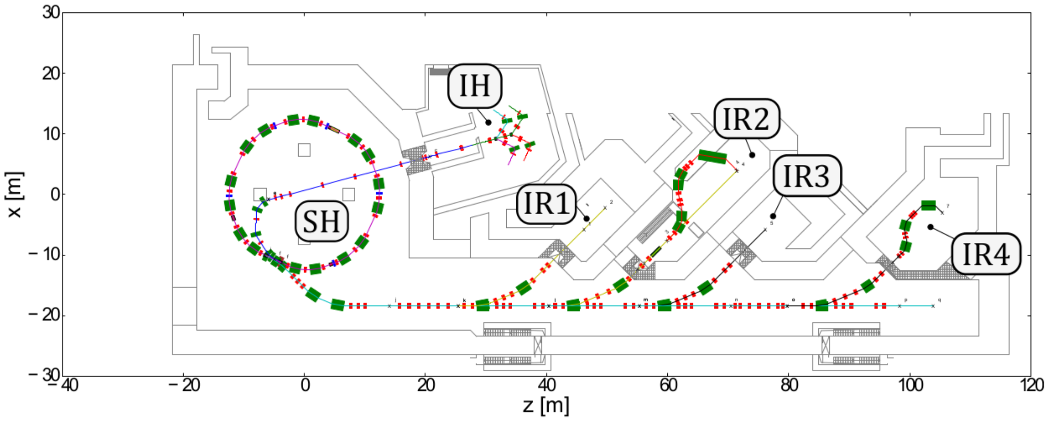


Figure 1.1.: Overview of the MedAustron accelerator complex and treatment rooms, from [8]. Additional building parts for the medical, technical and research areas were left out at the top of this image. The ion sources are located in room IH, the synchrotron is in room SH and the irradiation rooms are marked as IR1-IR4.

the accelerator complex at MedAustron is provided by figure 1.1 [8]. In it, the synchrotron is sketched as the ring on the left. Two connections can be traced from it. The linear accelerator, which supplies the synchrotron with ions from the ion sources, points to the upper mid of the image and the extraction line connecting the synchrotron to the treatment rooms leads to the lower right.

This extraction line allows the beams to be extracted from the synchrotron and delivered to three treatment rooms and an additional room reserved for nonclinical research. The first room supplied by the extraction line is the research room. It has a thicker shielding wall upstream of the beam, because it can be supplied with 800 MeV protons; an energy not going to be used in the treatment rooms. Only a horizontal beam will be possible in the research room. Then, a room with both a horizontal and a vertical beam line, is connected to the extraction line. It is going to be used for medical treatment and research, similar to the third room, which will, however, only feature a horizontal beam line. The first three rooms are going to be able to use both protons or carbon ions, but the fourth one is limited to protons due to the use of a proton gantry. It is a rotating machine with a diameter of 8 m and will be able to rotate the entire beam line for the irradiation room, by angles limited to -30° to +180° in the vertical direction. The gantry, together with the patient positioning robotic tables found in the treatment rooms, can direct the beam from a wide variety of directions towards a patient. This will enable treatment planners to better spare healthy tissue by combining several beams from different directions, thus distributing the entrance dose over more tissue [6, 7].

1.1. Motivation

For treatment planning systems it is essential to describe the biological effects that are triggered by radiation. Since the response from affected tissue, such as the cell survival rate, depends on radiation qualities, such as the particle type, the concept of Relative Biological Effectiveness (RBE) was created. This quantity is the ratio between a reference dose of x-rays and the dose of a beam in question, that produces the same biological effects, according to [1]:

$$RBE = \frac{D_{\mathrm{ref}}}{D_{\mathrm{ion}}}. \tag{1.1}$$

Thus, it is possible to determine the photon-equivalent dose D_{ref} of an applied ion beam by multiplying its RBE with the absorbed dose D_{ion}. This equivalent has high value in treatment planning, because it can be linked to cell survival rates and complications in healthy tissue. Treatment planning systems use models to obtain the RBE for every point in an area of interest. In order to estimate the physical dose that is necessary to cause a desired response, D_{ref}, an accurate description of the ion dose D_{ion} is vital. Unfortunately it depends on many parameters, amongst which are the dose rate, the type and energy of a projectile and the type of a target nucleus. To correctly estimate the impact of an applied ion beam, the software needs to model how the particles are distributed in the tissue and how much dose is absorbed in the different regions [1, 9].

This is not a trivial task, as the dose is not delivered by a single type of particles. Instead, due to nuclear fragmentation, a primary ion beam produces secondary, lighter ions as it moves through matter. These secondaries travel along with primaries, some of which have a larger range and produce a dose beyond the terminal depth of the primary beam. Additional fragments from the target nuclei also distribute dose in the tissue with a range of a few micrometers and a high linear energy transfer. In order to provide an accurate estimate of the biological effects, treatment planning systems should be able to handle fragments, since their contribution to the dose distribution is not negligible. The production of fragments depends on the total reaction cross sections, which in turn depend on the projectiles and their energy, as well as the targets [1, 9, 10, 12].

Many different cross sections need to be obtained for a complete description of the biological response, because the projectiles can gradually lose energy as they move through matter. Therefore a projectile can interact with other particles at different energies. Additionally, they can collide with a lot of different elements found in the human body and in implants. A wide variety of interactions with different parameters needs to be taken into account. The available cross section data from the literature is not enough to describe all of these reactions, at least during the production of this work. A possible solution to this is to use empirical parameterizations, such as the Sihver formula [11, 12], to estimate cross sections, however this was not the goal of this work. Monte Carlo (MC) simulations were already demonstrated to be useful in a medical context [13] and until more measurements can be carried out, MC simulations could also produce cross section estimates [14]. Several software packages, such as the platforms `Geant4` [15], `MCNP` [16], `FLUKA` [17], `PHITS` [18] or `SHIELD-HIT` [19], are able to simulate the passage of particles through matter, using Monte Carlo methods. `GATE` is another software package that adds functionality and an easy scripting language to the `Geant4` libraries [20]. Using these tools it is possible to simulate the outcomes of a wide variety of experiments, in fields like high energy physics, radiation physics, medical physics and medical imaging [15–20].

The main goal for this work was to use `Geant4` and `GATE` to simulate a particle beam passing through a thin target and to obtain cross section estimates from the results. During the simulation, particles were tracked by `GATE` and stored in a list that contained their particle names, energies and positions. In order to calculate cross sections from the simulation results, a program had to be developed for this work. It is able to run through such a list of tracks and calculate cross sections from its contents. Using this program, a large amount of cross section estimates was produced. In case of a good agreement with real data, these estimates could be used in treatment planning systems, as a supplement to cross sections from measurements.

1.2. Expected Results

At the beginning of this work an experiment of a particle beam going through a small target was going to be simulated. This simulation had to be executed for several iterations of projectile type and energy, as well as target material. The initial beam was composed of carbon ions with energies set to values between 10 and 500 MeV/nucleon. A total of eleven different incident energies were considered for the simulations. Further, sixteen target materials, which are important with regard to radio-therapy, were included. All of the combinations of projectile energies and target materials needed to be simulated, so a total of 176 simulations was planned and executed. Another series of simulations, using an incident energy of 95 MeV/u, was carried out with 5 target materials in order to be able to compare the cross section estimates to data from the literature [21]. Initially it was also envisaged to simulate different projectile types, like protons, alpha particles or other light ions which are important in particle therapy. However, it quickly became clear that also covering these iterations would need much more than the available time for the production of this work. Due to this, only carbon ions were used as projectiles during this work. The work-flow – which contains necessary steps taken to configure the simulations, run them and obtain their output – is described in chapter 3. Using this work-flow it will be possible to produce many more estimates in future projects.

To extract reaction cross sections from the simulation output, an analysis software was developed while the simulations were conducted. This program had to be able to run through the output of a simulation and calculate differential and double differential cross sections from it. After calculating, the program should store the estimates in a collection of text-files on a hard disk. For each of the 181 simulations, a large amount of estimates was going to be produced, including cross sections for 17 different fragments. The program's functionality, its configuration and the resulting output is described in section 3.5.

Selected examples from the cross section estimates are presented in this work, while the data files themselves can be found on the CD attached to this document. All of chapter 4 is dedicated to the presentation of these examples. They were chosen to generally explain the output and to judge the quality of the estimates, compared to cross section data from the literature. Also, the influence of certain key observables is discussed in chapter 4. To test the estimated cross sections, data from the E600 experiment conducted at GANIL [21] were compared to them.

The estimates were expected to contribute to an improvement in treatment planning for particle therapy. In case the agreement between estimates and data would be generally good, the produced cross sections might be used directly, or in case of a observable-dependent agreement only with observables that were found to produce accurate results. On the other hand, in the case of a bad agreement, an understanding of the disagreement between estimates and data could be used to improve available physics models or their utilization. As a next step other works, similar to this one, could carry out further iterations of simulations that cover different primary projectiles. Additionally, results from different physics models, newer versions of these models or even other simulation frameworks should be taken into account in future works, in order to improve the suggested work flow or find a more suitable system.

2. Physics Background

The purpose of this chapter is to introduce readers to the necessary theoretical background for the conducted simulations. In there, a carbon ion beam with an energy between 10 and 500 MeV/u and its interaction with a target was modeled. Such a beam is typically produced in a particle accelerator and can be used to irradiate tumors during hadron therapy, a process that is planned to begin at MedAustron using protons by the end of year 2016 [5].

In a simulation, individual particles of the beam are transported through matter, where they can interact with other particles. A brief description of the most important reactions is provided in section 2.1. Primary particles are slowing down due to a large number of interactions with other particles. This process is stochastic in nature but can be macroscopically modeled as a continuous slowing down approximation, until the particles are ultimately stopped in a target material. Section 2.2 is used to illustrate the different dose distributions of photons, electrons and heavier charged particles and to explain the stopping power for charged particles moving through matter. Then, the fragmentation of heavy ions is covered in section 2.3, along with the influence of nuclear fragmentation on the dose distribution of charged particles.

After introducing the necessary physics, this chapter also serves to present the basics of the Monte Carlo (MC) method for simulations. The use of MC simulations in particle transport software is briefly described in section 2.4. Particles were registered during the simulation, so that they could be grouped based on their locations and energies and counted. From these counts differential and double differential cross sections were calculated. This was accomplished according to a method explained in section 2.5.

2.1. Reactions

When particles interact with matter, there are several mechanisms through which they can transfer energy to other particles. The most important processes for charged high energy particles are [22, 23]:

1. inelastic collisions with atomic electrons of the medium
2. elastic scattering from nuclei

3. scattering in the Coulomb-field of a nucleus, which creates Bremsstrahlung
4. nuclear reactions
5. inelastic scattering from nuclei
6. emission of Cherenkov radiation when charged particles are faster than the speed of light in a medium

The first two mechanisms are most important for the slowing down of high energy charged particles in matter [22, 23], a process that is explained in section 2.2. While the other mechanisms are much less likely to occur, they still contribute to the overall energy loss. Nuclear reactions for example are the cause for fragmentation of projectiles or targets [23], a process described in section 2.3.

From the two predominant processes – inelastic collisions with electrons and elastic scattering from nuclei – the inelastic collisions are mainly responsible for energy loss in matter. In this case, inelastic collisions are considered as interactions between two particles, where the total kinetic energy of them is not conserved. Projectiles typically transfer energy to the atom by striking one of its electrons, thus ionizing or exciting the atom [23]. Heavy particles with an energy in the keV-MeV region lose only a small part of their energy with each interaction, but due to a large number of collisions they can lose an enormous amount overall [22, 23]. It is also possible that a large amount of energy is transferred to single electrons. In this case the recoiled electrons would be called δ-rays; these can have enough energy to cause further, secondary ionizations [23].

The second most important mechanism of energy loss after the inelastic collision with an atomic electron, is the elastic scattering off a nucleus. Here, an elastic scattering is considered an interaction where the total kinetic energy of the participating particles is conserved. This case is much less likely to occur and the transferred energy is very small if the mass of the nucleus is larger than that of the projectile. But even in cases where the masses are similar or the projectile has more mass, elastic scatterings from nuclei still do not contribute much to the overall energy loss, because these reactions do not occur as often as inelastic collisions with electrons do [23].

The chance of a collision occurring can be characterized by a cross section σ, which has the dimension of an area. Such cross sections are always associated with an interaction mechanism – like an elastic collision – and its participants, the projectile and target. Section 2.5 shows how to calculate cross sections for the production of fragments from nuclear reactions that were simulated in this work. As an example, if an atom would be modeled as a hard sphere with diameter d, the chance of another spheric atom interacting with it would be proportional to the cross section σ associated with hard spheres [24]:

$$\sigma = \pi d^2. \tag{2.1}$$

The total cross section for nucleus-nucleus interactions contains contributions from inelastic and elastic processes. To obtain the total reaction cross section for such a collision it is necessary to consider [11, 25]

$$\sigma_R = \sigma_T - \sigma_{\text{el}}, \tag{2.2}$$

using σ_R as the total reaction cross section, σ_T as the total cross section and σ_{el} as the elastic cross section. σ_R is very important for this work, since `Geant4` depends on this quantity to sample the probabilities for the occurrence of nuclear reactions. It is possible to estimate the total reaction cross sections using empirical parameterizations. `Geant4` uses four different parameterizations, one of which is a version of the Sihver formula, given by [11, 25] equation 2.3 from [11]:

$$\sigma_{R,\text{Sihver}} = \pi r_0^2 [A_p^{1/3} + A_t^{1/3} - b_0[A_p^{-1/3} + A_t^{-1/3}]]^2, \tag{2.3}$$

$$b_0 = 1.581 - 0.876(A_p^{-1/3} + A_t^{-1/3}), \tag{2.4}$$

$$r_0 = 1.36 fm. \tag{2.5}$$

Here, A_p and A_t are the mass numbers of projectile and target, respectively. The formula consists of a geometrical term $(A_p^{1/3} + A_t^{1/3})$ and its parameter b_0, called the transparency parameter, accounts for an overlap of nucleons in the nucleus. The cross section $\sigma_{R,\text{Sihver}}$ is energy-independent and suitable for energies greater than 100 MeV/u [11, 25]. It should be noted that a more recent version of this parameterization, that takes energy-dependence into account, is available [12], but this version is not being used by Geant4 and therefore not being used in this work either.

2.2. Bragg Curve

In particle therapy, patients are treated with charged particles like protons or heavy ions, instead of photons. This method uses advantages due to a favorable dose distribution and biological impact [1, 2, 9]. Figure 2.1 [26], compares the depth-dose profiles of several beam types to each other, amongst which are X-rays, electron and proton beams.

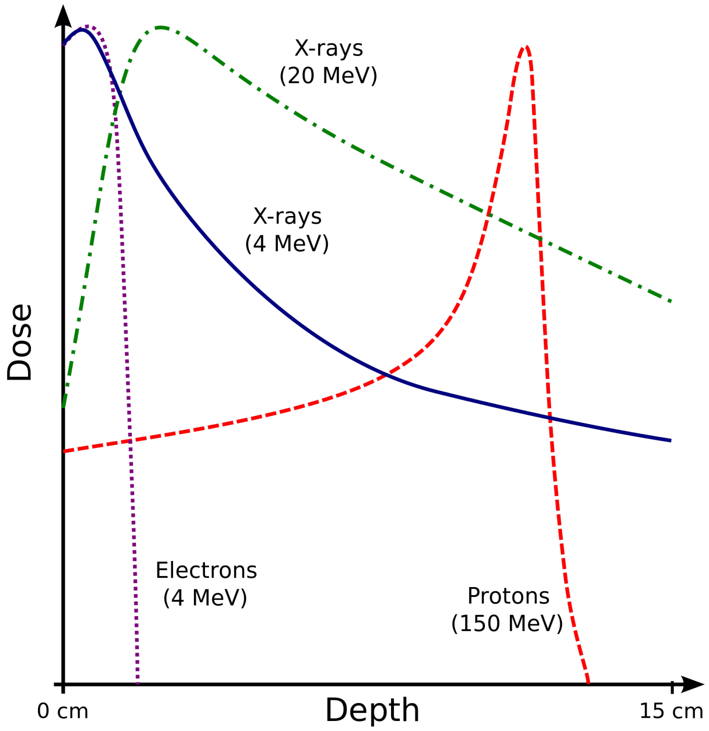


Figure 2.1.: Depth dose distributions of x-rays, electrons and protons, from [26].

As can be seen from the image, electron beams achieve a high dose at the surface that quickly falls off as the beam reaches deeper tissue. They are easily scattered by nuclei due to Coulomb interactions, primarily due to their low mass compared to atoms. In treatment they are mainly used on superficial tissues, like the skin, as a consequence of their low penetration depth. Even though electrons are charged particles, electron therapy is considered as part of an entirely different category than particle therapy [27].

Photon based treatment methods are described as conventional radiation therapy, using gamma- and X-rays between 35 keV and 25 MeV. These beams enter the body with a dose that initially rises, until shortly after the surface. Afterwards the depth-dose decreases exponentially at increasing penetration depth. The location of the largest dose can be shifted into deeper tissues, however, even for high energies a large part of the dose is delivered to shallow tissue. Photon based beams also irradiate tissues deeper than the target, if they are in the beam path, until the rays exit at the other end of the body [1, 2]. The standard method to reduce dose in the surrounding tissue while increasing the absorbed dose in the target, is to use multiple entrance channels. This can be achieved by applying several beams with a smaller intensity from different directions [2].

The disadvantages of X-rays can also be circumvented by using heavy charged particles from a particle accelerator. Particle beams like proton- or heavy ion-beams show a dose distribution with a peak, called the Bragg Curve. Initially, the particle beam deposits less energy than at the terminal depth of the beam, where a sharp peak – the Bragg peak – occurs. After the peak, the deposited energy quickly vanishes using protons or falls to a minimum with ions [1, 2, 9].

Because the position of the Bragg peak, more specifically its depth in the tissue, depends on the initial energy of the particle beam, it is possible to accurately control the deposition of dose in the tissue. In order to reach deep-seated tissues it might be necessary to achieve a range of 30cm, which corresponds to energies of 220 MeV for protons or 5.16 GeV for carbon ions. A useful description of the stopping power (average loss of energy over the penetration depth) in this energy range, can be given by the Bethe formula [1, 9]:

$$S(E) \equiv \frac{dE}{dx} = \frac{4\pi e^4 Z_t Z_p^2}{m_e v^2}\left[\ln\frac{2m_e v^2}{\langle I\rangle} - \ln(1-\beta^2) - \beta^2 - \frac{C}{Z_t} - \frac{\delta}{2}\right]. \tag{2.6}$$

Z_p and Z_t are the charges of projectile and target, respectively, m_e and e are the mass and charge of an electron and $\langle I\rangle$ is the mean ionization energy of the target. Further, C/Z_t is the shell correction term and $\delta/2$ is the density correction term. The former of the two becomes important for particle velocities close to the speed of atomic electrons, while the latter is only relevant for ultra-relativistic charged particles [1, 9]. An illustration of the stopping power is shown in figure 2.2 [9], which plots the average loss of energy per unit length as a function of the projectile's energy. The image indicates the contributions of electronic stopping and nuclear energy loss, including the Lindhard and Anderson-Ziegler models for electronic stopping, which are used in the low-energy region where the Bethe formula is no longer valid. Initially a projectiles mainly slow down due to electronic stopping, a result of interactions between projectile and atomic electrons in the target. Interactions with atomic nuclei are less likely at high energies, corresponding to an initially lower contribution of nuclear stopping power to the slowing down process, however, nuclear reactions get increasingly likely as the projectiles lose kinetic energy [9]. As particles slow down the contribution of nuclear stopping increases until it dominates the slowing down process at projectile energies below $E_p \leqslant 10$ keV/u [1].

The Bethe formula for electronic stopping can be linked to the absorbed dose in a thin slice of a target, according to [1]:

$$D = 1.6 \times 10^{-9} \times S(E) \times F \times \rho^{-1}, \tag{2.7}$$

for the absorbed dose D, the electronic stopping power $S(E)$ from equation (2.6), the particle fluence F of a beam and the mass density ρ of an absorbing material [1].

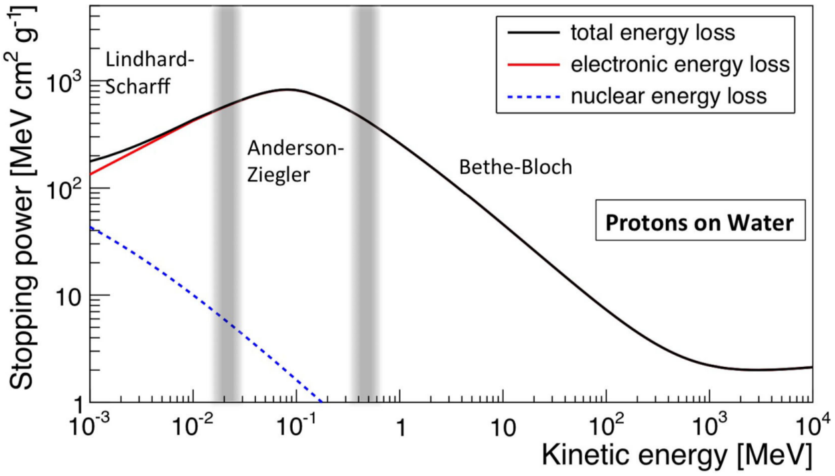


Figure 2.2.: Stopping power for protons in a water target, as a function of projectile energy, from [9].

Several things can be learned from Bethe's theory. The velocity dependency of v^{-2}, which is part of the main term, means that the stopping power is inversely proportional to higher kinetic energy. At first, the ions enter with an initial energy that was provided by a particle accelerator like a cyclotron or synchrotron. The particles suffer a large number of inelastic collisions with atomic electrons and elastic scatterings with their nuclei in the tissue, losing some of their energy bit by bit. Thus, they also slow down. Upon slowing down they are increasingly likely to lose more energy, according to equation (2.6). At some point, shortly before the depth of the Bragg peak, these interactions begin to rapidly transfer energy to the interaction partners, stopping the projectiles at the end of the beam path. According to theory, this is the reason why charged particles deposit their energy in the way Bragg described it, gradually at first and afterwards with a sharp peak at the terminal end [1, 9].

The main term of the Bethe Formula also depends on the charge of the traveling particle. At high enough energy the ion does not carry electrons with it. When they have lost enough energy to pick up electrons from the surrounding medium, their effective charge $(Z_p)_{\text{eff}}$ is reduced according to equation (2.8), from [1], which leads to a reduction in stopping power [1, 9]

$$(Z_p)_{\text{eff}} = Z_p[1 - \exp(-125\beta Z_p^{-2/3})], \tag{2.8}$$

using the charge Z_p and $\beta = v/c$ of a projectile. In order to deduce the mean range from the stopping power it is necessary to integrate equation (2.6) over energy [1]:

$$\Delta x = \int_0^{E_0} \frac{1}{S(E)} dE, \tag{2.9}$$

to obtain Δx as the *Continuous Slowing Down Approximation Range* and E_0 as the projectile's initial energy. In case of a heavy charged particle, like a carbon ion, the mean range is quite close to the total range, because they aren't scattered strongly and keep moving in the beam direction. The range of ions with a different mass but the same energy per nucleon scales with a factor of A/Z^2. This dependency can be seen in figure 2.3, which illustrates the different ranges of ions and their dependency on initial energy [1].

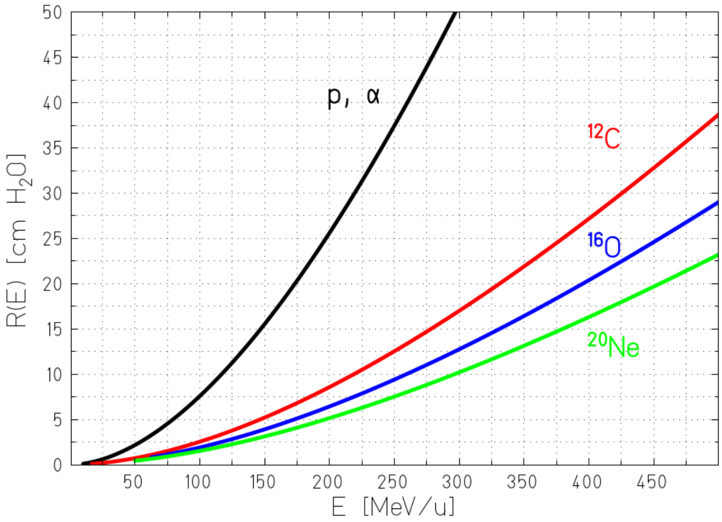


Figure 2.3.: Ranges of different ions, taken from [1]. Penetration depth in a H_2O-target is plotted as a function of incidence energy.

The energy dependence of the depth of the Bragg peak can be used to combine several beams with different beam energies into a *Spread Out Bragg Peak* (`SOBP`). By adding up the dose profiles of numerous individual beams, it is possible to create a plateau of dose that spreads over a region instead of a single peak at one depth. Such a plateau is shown in figure 2.4 [28], where a `SOBP` is compared to the dose profile of X-rays. The `SOBP` is used during treatment, because it allows physicians to create a three-dimensional dose distribution that fits to the target tissue based on CT scans [1, 2, 28].

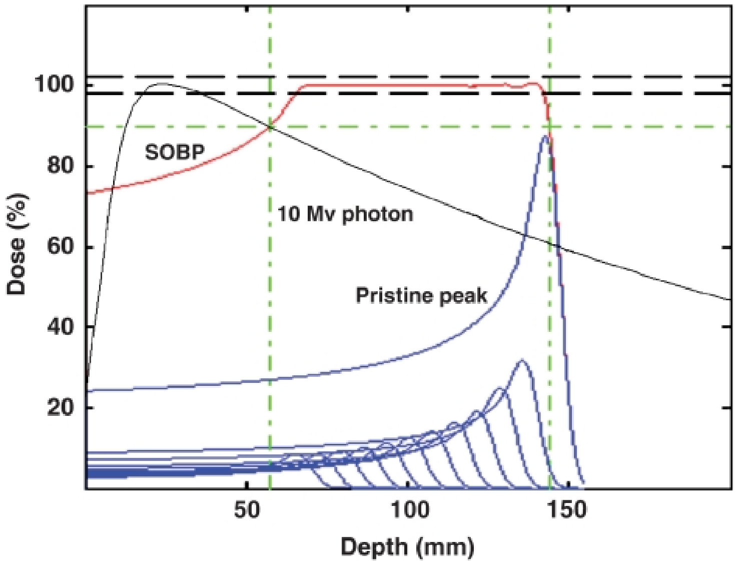


Figure 2.4.: Spread Out Bragg peak, compared to X-Ray depth dose, from [28].

When looking at the differences between `SOBP` and a dose distribution using X-Rays or gamma rays from a single beam, two advantages of heavy ions come to mind. Treatment planners are better able to geometrically fit a dose distribution to a target. Also, it is possible to spare healthy tissues more easily, at least after the terminal end of the beam. X-ray beams do not end inside a patient, since they are only attenuated. Furthermore, the tissue in front of the target can also be subjected to less dose when an equally high dose is delivered to the tumor by means of an ion beam. It is also possible to increase the dose in the target while retaining the same amount of damages that would have been accepted in other treatment methods. For example, this can provide an advantage when dealing with radio-resistant tumors. [1, 2, 28].

2.3. Fragmentation Tail

While proton beams typically terminate shortly after the Bragg peak, this is not usually the case for heavy ion beams. For example, when carbon ions collide with other nuclei, fragments are created along the beam path. This is due to nuclear reactions, which contribute less to the slowing down of projectiles than electromagnetic processes, but still play an important role. Similar to collisions governed by electromagnetic forces, nuclear reactions are able to lead to elastic and inelastic collisions. In the latter case, a proton for example could forcefully strike parts of a nucleus out of the target. In case the projectile is an ion it may also break up itself, due to a collision [1, 2, 9].

This is important to consider, because nuclear reactions significantly alter the depth dose distribution. Although the stopping process of the primaries is mainly influenced by collisions with atomic electrons, nuclear reactions do play a role, especially for larger penetration depths. Secondary particles are created along the beam path and also travel in the target. Remnants of the projectiles often travel in about the same direction as the primary beam. Due to the dependency on A/Z^2 of the range of such projectiles, certain projectile-like secondaries are able to travel further than primaries. This way, they can produce a so called fragmentation tail, a significant deposition of energy after the terminal depth of the primary beam [1, 9]. A very distinct contribution to the depth dose can be seen in figure 2.5, for a beam of primary 20-Neon ions in water [1].

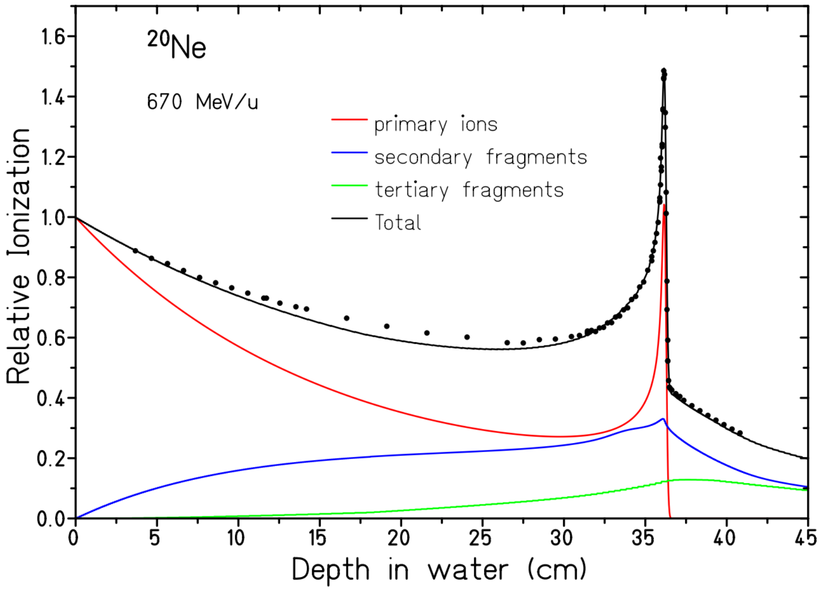


Figure 2.5.: Bragg curve for ^{20}Ne ions in water, measurement and calculated contributions of primary ions, secondary and tertiary fragments, from [1].

Another consequence of the secondary particles is that they are built up at the cost of fluence of the primary beam. Nuclear reactions adapt the dose distribution in several ways. Secondary particles increase the energy deposition in the build-up region, prior to the Bragg peak. The peak itself is reduced in size, compared to

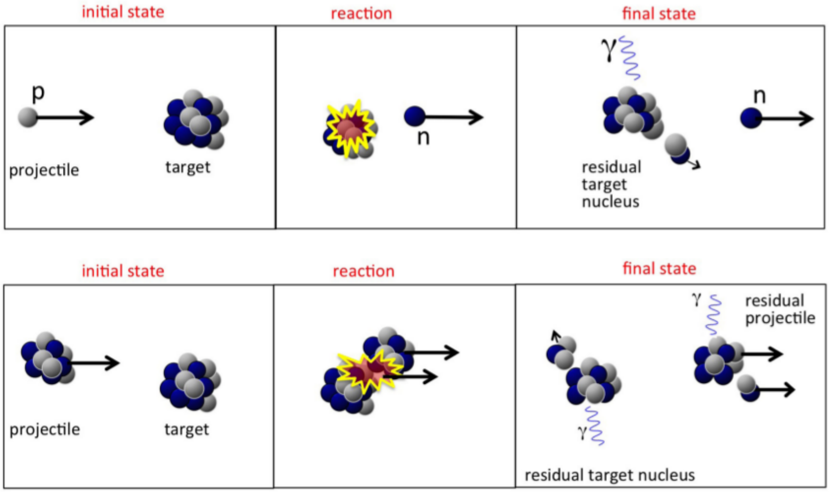


Figure 2.6.: Abrasion-ablation model for nucleon-nucleus and nucleus-nucleus collisions, from [9]. The top half illustrates a proton that hits a nucleus, immediately ejecting a neutron. In the lower image an incoming ion, instead of a proton, strikes the nucleus and splits into fragments. Both cases can lead to an excited target and/or projectile, which may further evaporate particles.

how large it would have been without nuclear reactions. For larger energies, hence Bragg peaks at greater depths in the target, these effects are more prevalent than at lower energies. As the particles are able to travel further, they can also suffer more collisions with target nuclei. Similarly, the fragmentation tail grows in size at the cost of Bragg peak height [1, 9].

The creation of nuclear fragments can be described by one of two related theories, depending on the projectile, which are shortly explained here. Really there are several more theories available, for example Quantum Molecular Dynamics [25] or the Quark Gluon String Model [25], but the mechanisms used in this work are derived from the following two models. In nucleon-nucleus reactions the projectile starts a series of two-body collisions between nucleons. The process is called *Intra-Nuclear Cascade* and is described in more detail in subsection 3.1.3. The projectile interacts with quasi-free nucleons inside the target nucleus, which are modeled by a density distribution and a nuclear potential [1, 9, 14].

Nucleus-nucleus collisions on the other hand are described by the *Abrasion-Ablation Model* and its derivatives, which are similar to nuclear cascades. In a first step called abrasion, projectile and target violently interact with each other in an overlapping region. This reaction happens within about 10^{-23}–10^{-22} s and produces lighter fragments, a quasi-projectile consisting of parts of the projectile that were unaffected and a quasi-target, often in an excited state. This step is also modeled by an *Intra-Nuclear Cascade* model, however the model must be suited for an incoming nucleus, not just a nucleon. In the ablation-step, which takes 10^{-18}–10^{-16} s, the excited remnants de-excite by evaporating additional nuclei or fragments. The two-step process is shown in figure 2.6 [9]. Often there are parts of the nuclei that are not caught in the collision. It should be noted that the remainders of the projectiles, sometimes called quasi-projectiles, continue to travel in about the same direction with much of the initial velocity. Similarly, parts of the target may not play a part in the collision, leaving a stationary quasi-target [14]. Products of the evaporations on the other hand are released isotropically in the reference frames of the excited nuclei. This means that the evaporation products of the remaining target are evaporated isotropically from a resting frame, while those of the projectile-remnant are evaporated isotropically from its moving frame, thus with a forward momentum [1, 9].

2.4. Monte Carlo Method

The Monte Carlo method has been used in several fields of physics for around 40 years. Since then, several software packages were produced for the simulation of particle transport problems, amongst which are: `MCNP` [16], `FLUKA` [17], `PHITS` [18], `SHIELD-HIT` [19] and `Geant4` [15]. These systems are able to simulate particle transport through matter along with the physical interactions that could arise, such as nuclear interactions. They have been effectively used in fields such as radiation protection and dosimetry, shielding, medical physics, detector design or accelerator target design [15–19].

The main difference between MC simulation and other numerical analysis techniques is the reliance of MC methods on the sampling of random numbers to generate solutions. Instead of solving deterministic problems, a stochastic model is set up. Then, individual values are randomly selected from probability distributions in order to statistically estimate a quantity of interest. By using a sufficiently large number of random values it is often possible to achieve an accurate enough estimate, simply by calculating the mean value of all the results [29].

An application of MC simulation that solves transport problems usually needs a large number of individual particles which are tracked as they move through space. These particles can travel short distances that are interrupted by events, like interactions with other particles or decays. In these cases both the distance traveled and the type and initial conditions of the events that interrupt normal transport are generally subject to random sampling. The obvious advantage of this approach is that the simulation of physics can be restricted to a region near the particle, at each step of its history. More detail on MC simulation for particle transport can be found in the references [25, 29] and in subsection 3.1.1 of this work.

MC simulations depend on quick access to random numbers. Although they can be produced by various hardware, the most used method is to use Pseudo-Random Number Generators (`PRNG`). These generators are arithmetic algorithms that can create a sequence of numbers which fulfill certain properties of randomness, most importantly being uniformly distributed over an interval and being uncorrelated to each other [29, 30]. It should be noted that the individual pseudo-random numbers often are not random and can be reproduced. Usually, during runtime, a `PRNG` is initialized with a seed which is a value that completely determines the sequence of pseudo-random numbers. This means that it could theoretically be possible to re-create the exact same output of a MC simulation with the knowledge of the seed that was used. However, this seed may be secret, often includes random numbers and it may also be changed during runtime [30].

It is helpful to showcase a simple example from [29] of how random numbers could be used to sample events in a particle transport setting. Suppose a projectile can be subject to one of the different events E_1,...,E_n available to it. Each of the events has a probability p_1,...,p_n of occurring, so that equation (2.10) is fulfilled [29]:

$$\sum_{i=1}^{n} p_i = 1. \tag{2.10}$$

Only one of the available events may be selected as the next event. If a random number ξ, using $0 \leqslant \xi < 1$, is chosen by the `PRNG` such that it satisfies [29]:

$$p_1 + ... + p_{i-1} \leqslant \xi < p_1 + ... + p_i, \tag{2.11}$$

then the random number ξ can be used to select the event E_i as the next event to occur in a particle's history. The probabilities of the individual events could be defined by their related cross sections $p_i = \sigma_i / \sigma_t$, using the total reaction cross section $\sigma_t = \Sigma_i^n \sigma_i$. Possible examples for such events for a neutron interacting with an atom could be capture, elastic or inelastic scattering [29].

2.5. Cross Section Calculation

This section deals with the relationship between cross sections and the total number of particles in a volume; a necessity in order to calculate the cross sections from several distributions of absolute counts in the analysis software described in section 3.5. Each of these distributions provides the number of particles from a certain type, which were found in a specific volume and with a certain energy. In case of proton differential cross sections the underlying distribution describes the total number of protons found in each volume element.

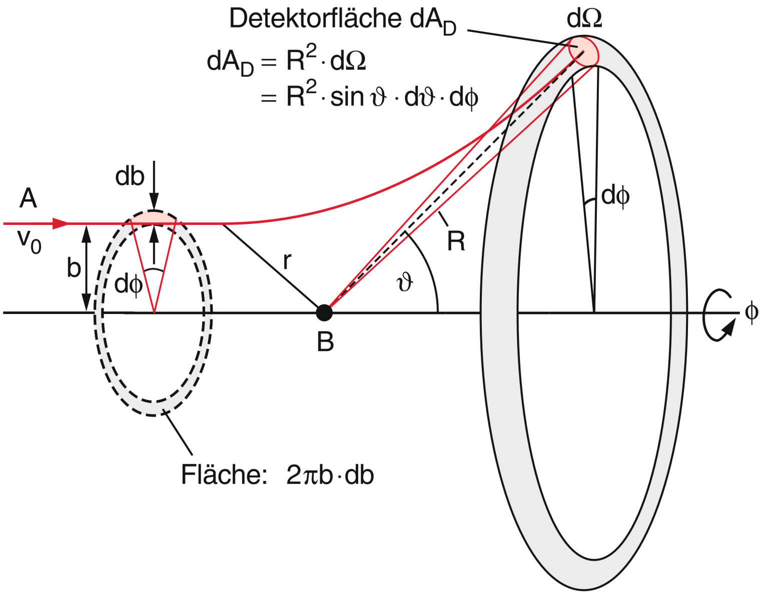

Detektorfläche dA_D
$dA_D = R^2 \cdot d\Omega$
$= R^2 \cdot \sin\vartheta \cdot d\vartheta \cdot d\phi$
$d\Omega$
db
A
v_0
b
$d\phi$
r
R
ϑ
B
$d\phi$
ϕ
Fläche: $2\pi b \cdot db$

Figure 2.7.: On the relationship between deflection function $\vartheta(b)$ and differential cross section $d\sigma/d\Omega$, from [24].

Figure 2.7 [24] illustrates how differential cross sections could be measured. Projectiles enter from a point A on the left of the image. After an interaction with a target at point B, the projectiles are scattered by a polar angle ϑ, which is the angle between incident beam and detector axis. The limits $\vartheta \pm \Delta\vartheta$ represent the minimum and maximum polar angle of the detection volume. The other angle shown in the illustration, ϕ, could be additionally used to describe a solid angle element that only uses a part of the ring around the axis [24]. In this work, however, solid angle elements $d\Omega$ were always calculated using $d\phi = 2\pi$, in order to cover the complete ring. Also, in the simulations, the impact parameter b was very close to zero.

During the simulations, a large number of particles was scattered by atoms in the target. They were recorded at the surface of a detector with their names, energies and locations. Before calculating cross sections, the particles were grouped and counted, based on the attributes mentioned before. To accomplish that, the surface of the detector – which was shaped like a sphere – was split into solid angle elements with a polar angle of $d\vartheta = \vartheta \pm \Delta\vartheta$ and an azimuthal angle of $d\phi = 2\pi$. The differential cross sections $d\sigma/d\Omega$ of a fragment were obtained using a function that depends on the number of particles that were detected in the angle elements, divided by the number of primary incident particles. Double differential cross sections $d^2\sigma/(d\Omega\ dE)$ were obtained using a similar function and a frequency distribution, which further divided the particles into energy bins $dE = E \pm \Delta E$. The particles were sorted into their according energy bin and counted based on both energy and angle of detection. The following equations were used to calculate cross sections. Equation (2.12) is used for differential cross sections, while equation (2.13) is used for double differential cross sections [31]:

$$\frac{d\sigma}{d\Omega}({}^{A}_{Z}\mathrm{X}, d\vartheta) = \frac{N_{{}^{A}_{Z}\mathrm{X}} \cdot A_T}{\rho_T \cdot d_T \cdot \mathcal{N}_A \cdot N_{{}^{12}\mathrm{C}} \cdot \Delta\Omega \cdot}, \tag{2.12}$$

$$\frac{d^2\sigma}{d\Omega dE}({}^{A}_{Z}\mathrm{X}, d\vartheta, dE) = \frac{N_{{}^{A}_{Z}\mathrm{X}} \cdot A_T}{\rho_T \cdot d_T \cdot \mathcal{N}_A \cdot N_{{}^{12}\mathrm{C}} \cdot \Delta\Omega \cdot \Delta E}, \tag{2.13}$$

$$\Delta\Omega = 2\pi \cdot \left[\cos((\vartheta - \Delta\vartheta) \cdot \frac{\pi}{180}) - \cos((\vartheta + \Delta\vartheta) \cdot \frac{\pi}{180})\right]. \tag{2.14}$$

With $N_{{}^{A}_{Z}\mathrm{X}}$ being the number of particles counted in an angle element $d\vartheta$ of the detection sphere, with an energy corresponding to the bin dE of a particular fragment ${}^{A}_{Z}\mathrm{X}$. A_T is the target mass, ρ_T the target density, d_T the thickness of the target, $\mathcal{N}_A$ the Avogadro number, $N_{{}^{12}\mathrm{C}}$ the number of incident ions, $\Delta\Omega$ is the solid angle covering the region at the polar angle element $d\vartheta = \vartheta \pm \Delta\vartheta$ and ΔE the width of the energy bins [31].

3. Materials and Methods

In this chapter the individual steps of the work-flow are covered in detail. A summary of it is presented in figure 3.1, which serves to provide an overview over the tasks and the files associated with each work step.

Before work could be started, the necessary software packages `ROOT` [32], `Geant4` [15], `GATE` [20] and `HTCondor` [33] were set up on a workstation in this order. This chapter covers the intended use of these programs, along with a description of the utilized input and the produced output. Section 3.1 deals with `Geant4` and how it links physical processes to physics models. It was not used directly; rather, `GATE` was added as an extension to `Geant4`, because it adds functionality and a simple macro language to describe Monte Carlo simulations.

Eight input files were produced to parameterize the simulations for this work. They are called macro files or simply macros and serve to completely describe a simulation task to `GATE`. Seven of the macros are static, while one of them was duplicated to create 181 permutations of similar simulations. These were used to configure the simulations for different combinations of beam incident energies and target materials. Five of the permutations were exclusively run and analyzed using parameters from a specific experiment [21], while the other 176 used more general parameters that varied more amongst the permutations. Additional details of the macros are explained after an introduction to the program `GATE`, in section 3.2.

After the macros were prepared, the simulation task for each of them was logically split up into a hundred jobs to be used with a `HTCondor` computer cluster [33]. This was done to increase the utilization of available processors, thus saving computing time. A submit file that links together the split jobs was produced for each of the simulation permutations. Section 3.3 is dedicated to the job splitting and subsequent submission to the `HTCondor` cluster.

Based on the macro files, `GATE` was able to simulate the interactions of projectiles from a particle beam with a thin target. It recorded all particles that left the area through a spherical detector and stored these records in a phase space, distributed over a hundred `ROOT` [32] output-files for a single simulation. This phase space is simply a list of all the particles that exited the simulation area and their attributes. It is explained in section 3.4, along with the `ROOT` framework.

To calculate cross sections from the phase space, an analysis program was developed while the simulations were processed by the computer cluster. This program can filter through the simulation output and group and count particles based on their attributes. After counting, the program is able to calculate double differential and differential cross sections for the selected particles and export the obtained data into text files, for further processing. Section 3.5 contains a detailed description of the program. This program is controlled by configuration `XML` files, which contain the necessary parameters for at least one analysis. Such a configuration file regulates which fragments, angle elements and energy bins are investigated for cross section calculation. An example for such a configuration is also included and explained in section 3.5. The analysis output, however, is described in chapter 4, which is dedicated to describing the output in general, presenting some of the cross section estimates and comparing them to data for an evaluation of their agreement.

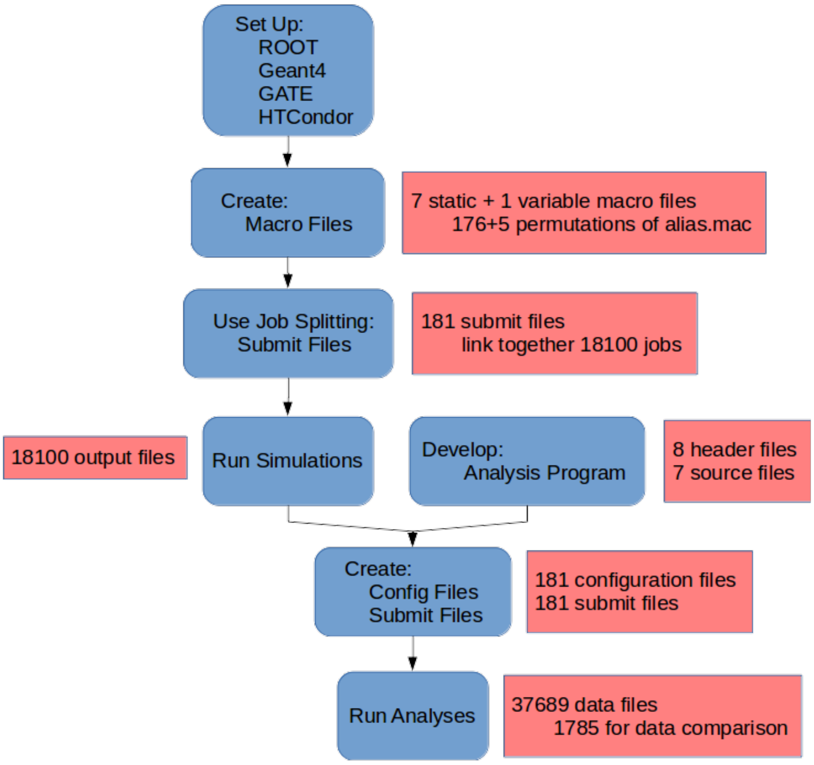


Figure 3.1.: Workflow for the production of cross section estimates from simulations.

3.1. Geant4

`Geant4` is a tool kit that is able to simulate the passage of particles through their interactions with matter. It is being developed by the `Geant4 collaboration` at CERN and supports several physics models for a wide range of materials, energies and interaction processes. There are models for electromagnetic, optic and hadronic processes, a comprehensive set of particles, materials and elements, as well as support for energies between 250 eV and the TeV energy range. The software contains functions for tracking, geometry, physics models and registering hits. Its main fields of application are high energy physics, space and cosmic radiation, medical physics and technology transfer [15, 34]. During this project `Geant4` version 10.01.p01 was used.

The main idea behind the development of `Geant4` was the production of a framework, which enables its users to construct a variety of different particle detector simulation programs, as independent of software and hardware environments as possible [15]. It enables users to program their own simulations using interfaces to components of the `Geant4` environment. Geometries are able to be defined and elements found therein can be declared as sensible to hit detection, in order to produce recordings like a detector. `Geant4` provides the physics processes and some implementations, however it is up to the users to produce their own programs to use them [15]. Using a custom made C++ program and the `Geant4` classes, users are able to reproduce their own experimental set-ups, but this can also result in a steep learning curve, depending on the backgrounds of the individual users. To help out, frequently offered seminars and training courses are available [35] in addition to the examples provided by a `Geant4` installation [15, 34]. In the context of this work another possible solution was realized to circumvent this issue, namely the use of the `GATE` toolkit.

Explanations provided in this section are based on the `Geant4` *Physics Reference Manual* [25], which contains more detailed information than this section.

3.1.1. Particle Transport

The simulations conducted during the production of this work can be summarized in a few sentences: A large number of particles emerged from a specified location, called the source. They were able to travel through space and interact with other particles found therein. Many projectiles were able to pass through the simulation area, however some suffered collisions with other particles. As they interacted with each other, particles were able to be modified, absorbed or scattered. Additional events were possible as well, like radioactive decay, but here they are not going to be explained in detail [25, 29].

In a Geant4 simulation, instead of calculating an analytical or a numerical solution for the transport equations, all of the participating particles are individually processed. Every single particle moves through the simulation region while being tracked. The individuals experience different paths and events along them, due to the use of probability distributions and random numbers. For instance, this means that not all of the particles emerge from exactly the same location. Given a large enough number of tracks, Monte Carlo simulation is able to achieve an accurate estimate of physical quantities. This is due to the law of large numbers, however it relies on the quality of the physics models that calculate the results of physical interactions [25].

At the start of a track, `Geant4` places a primary particle at a random position inside the source region and gives an energy and momentum based on the description of the beam source to this particle. This description is usually created by the user, before the simulation is started. It is possible to let `Geant4` sample the initial conditions for the particles from a distribution or set fixed values for them [25]. Once produced, the particles can start to move through space according to their momenta and equations of motion. They are able to suffer a series of events, like decays or collisions. Each of these events are likely to occur after a certain distance, the mean free path. The chance of an event happening is linked to the mean free path corresponding to the event in question, which is calculated according to [25]:

$$\lambda(E) = (\sum_i (n_i \cdot \sigma(Z_i, E)))^{-1}, \tag{3.1}$$

$$n_i = \frac{\mathcal{N}_A \cdot \rho \cdot \omega_i}{A_i}. \tag{3.2}$$

Here, $\lambda(E)$ is the mean free path of a given process E, $\sigma(Z, E)$ the total cross section per atom for this process and n_i the total number of atoms in the medium. The $\sum_i$ is used to run through all the materials composing the medium, thus combining the cross sections of its components into a macroscopic cross section. In equation (3.2), $\mathcal{N}_A$ is the Avogadro number, ρ the density of the medium, ω_i the proportion by mass of the i^{th} element building up the material and A_i its corresponding mass of a mole [25].

`Geant4` treats the trajectory of a particle as a series of steps that are interrupted by events. In order to choose the next step, `Geant4` needs to calculate the mean free paths for each of the processes the particle can be subjected to. This is done once, and in advance of the particle's trajectory. Then, `Geant4` samples how many mean free paths need to be traveled before reaching an interaction, with [25]:

$$n_\lambda = -\log(\eta), \tag{3.3}$$

where n_λ is the number of mean free paths traveled depending on a randomly sampled number from the interval [0,1], called η. The number of mean free paths needed to reach an interaction is sampled for every process available to the particle. At each step, the shortest step length Δx and its corresponding process is selected as the next interaction. Then, all of the n_λ are updated for future steps, based on the step length Δx taken and the formulae [25]:

$$\Delta x = n_\lambda \cdot \lambda(x), \tag{3.4}$$

$$n_\lambda{}' = n_\lambda - \frac{\Delta x}{\lambda(x)}. \tag{3.5}$$

During each physical event, a physics model is used to simulate the results of the interaction governing this event. Physics models can be built to support a lot of different mechanisms that are not easily comparable, but they all solve similar problems. Basically, a model is given a list of particles, along with their attributes like type and energy, that participate in an event. First, necessary initial conditions and parameters, like the impact parameter for a scattering event, are sampled from their respective probability density functions, using random numbers. Then, after sampling the parameters, the model calculates the result of the interaction based on its mechanism. This can alter the involved particles, their momenta and energies, even destroying some of them, or creating new ones. Sometimes the model produces excited particles, which have to be immediately handled by another model before the rest of the simulation carries on [15, 25]. An important example, which is used to simulate nucleon-nucleon collisions – called the *Intra-Nuclear Cascade* – is explained in the next subsection.

If the particles involved in the now finished event are not destroyed in the process, the probability distributions for the following events are adjusted as described in equation (3.5). Afterwards they can take part in further interactions [15, 25].

3.1.2. Physics Lists and Models

In `Geant4`, particle interactions are governed by physics processes. These processes are structured into major categories, like "particle decay", "electromagnetic interactions", "hadronic processes" and "gamma- and lepto-nuclear interactions". Examples for standard electromagnetic processes of photons would be Compton scattering or the photoelectric effect, however, many more can be found in the *Physics Reference Manual* [25]. The individual processes are linked to physics models that implement the aforementioned mathematically. This allows the user to select different available models for the same process, or to add his or her own model instead, allowing a high degree of customization [15].

The selection of physics models is carried out with the choice of a so called *Physics List*. These lists combine several models tailored to a specific use case. Basically, every type of particle can be part of several physics processes, all of which are stored in a list – a physics list – for that particle type. A complete physics list couples every one of the processes to a suitable physics model for it. This architecture is necessary, because not all energy ranges can be accurately described by a single algorithm for all particles and processes. Several standard physics lists are delivered with `Geant4` and are ready to use, but users can also set up their own lists in the `Geant4` *Physics Editor* [15]. During this work the standard physics list `QGSP_INCLXX` was used for the simulations. This choice was made due to the model's performance during a project preceding this work [36] and similar results in a benchmark study [14]. The main underlying model for nucleus-nucleus interactions, the *Intranuclear Cascade Model*, is explained in the next subsection.

3.1.3. Intranuclear Cascade Model

When high-energy particles move through dense matter, at some point they are going to be subjected to interactions with other particles, like electrons or nuclei. In order to simulate the progression of a large number of particles, a simulation framework has to be able to model many different interactions. Most important for this work are hadronic interactions, which occur when a high energy projectile collides with a target nucleus, generating a particle shower. This means that an incoming projectile, such as a proton or an ion, colliding with a target can create a multitude of secondary particles. Then, these secondaries are also able to interact in a way similar to the primary particle, which may lead to a large number of successor particles [9, 25].

To simulate the outcome of such a collision, several models use an approach called *Intranuclear Cascade* model. This method uses the assumption that a projectile moves through a series of two-body collisions between different targets, an assumption that is met for high energy projectiles meeting a target, because the de Broglie wavelength of the incident particle is much smaller than the average distance between nucleons in a target. It is noteworthy that this condition also limits the use of the Intranuclear Cascade (`INC`) as a modeling tool to an energy range of between 1 MeV and 20 GeV for kinetic energies. The de Broglie wavelength increases for lower energies, such that protons with a kinetic energy of 250 MeV have a de Broglie wavelength of roughly the average distance between nucleons. At this point several collisions are no longer likely to be independent of each other, although the `INC` still works surprisingly well due to several quantum effects that increase mean free paths. In a cascade event an incident ion initiates the cascade within a target nucleus and a mean-field potential assumed to be that of the target [9, 25]. This imposes another limitation, since the mean-field interactions of a projectile's constituents are often neglected. Even though this approximation is tolerable for light projectiles interacting with heavier targets, it restricts the use of heavy projectiles in `INC` models [37]. Then, as the nucleons move inside the target, they may come close enough for a collision which could scatter the nucleons or generate other particles, like π-mesons [9, 25].

The simulations carried out for this work applied the model called `INCL++`, or *Liège Intranuclear Cascade* model [37] for hadronic interactions. It was chosen due to favorable results in a benchmark study [14] and a project by the author of this work [36]. In both works, several physics models were compared to each other and to data from the literature to assess their strengths and weaknesses. The `INCL++` model was able to reproduce the data in question slightly better, however the other models offered by `Geant4` also provided good results [14].

`INCL++` is an event-generator, able to simulate the results from nucleus-nucleus interactions for light ions with incident energies between 50-3000 MeV. It is able to deal with a high energy projectile that initiates a series of binary collisions within a target nucleus and supports the emission of nucleons, pions, and light clusters [25, 37]. However, because `INCL++` does not reliably predict interactions between nuclei heavier than $A = 18$, `Geant4` applies the *Binary Cascade Model* (`BIC`), provided by `Geant4` [25] for these cases. Also, instead of `INCL++`, the *Bertini Cascade Model* (`BERT`) [25, 38] is used when K mesons are involved during an interaction. Figure 3.2 presents a schematic that illustrates which model was used, depending on particle energy and type, as well as target type.

Supported particles for the `INCL++` model are pions, nucleons and ions up to a size of A = 18 [25]. Although the `BIC` model is listed as valid only for pions and nucleons, it can still be used for large nuclei. This is due to the fact that the `BIC` model models targets as three-dimensional collections of nucleons, rather than a smooth medium, which the `INCL++` and `BERT` models do [25]. The `BERT` model supports interactions featuring gamma particles, pions, kaons, nucleons and the baryons Λ, Σ^+, Σ^-, Ξ^-, Ξ^0 and Ω^0 [25].

Due to `INCL++` only supporting energies up to 3 GeV/u, other models are used for higher energies. High energy interactions between ions are always modeled by the *Fritiof String Model* (`FTFP`), while collisions between nucleon-, pion- or kaon-projectiles and a target nucleus are modeled by `BERT` below 10 GeV/u and either by the *Quark Gluon String Model* (`QGSP`) or the `FTFP` above it. This choice depends on the physics list used, in the case of this work it would be `QGSP`. However, energies this large were not used in this work, so the string models do not need further explanation in its context. A variant of the `QGSP_INCL++` physics list offers the simulation of high precision neutron interactions at low energies, but this option also was not used. Lastly, nucleon-nucleus reactions at energies lower than 1.5 MeV/u were governed by the `Geant4` *Precompound model* [25].

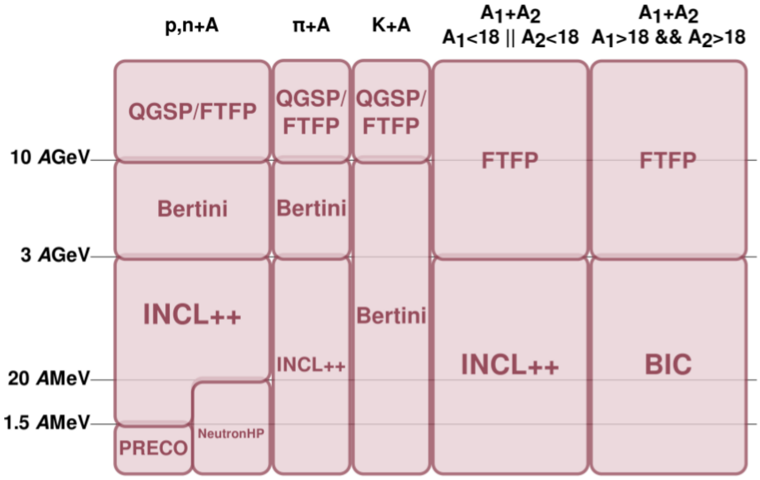


Figure 3.2.: Model map for the `INCL++` based physics lists in `Geant4`, from [25].

As previously mentioned, the `INCL++` model simulates the results of high energy collisions between nuclei. Each involved particle is tracked and all of them may only suffer binary collisions, that is interactions between two particles at once. The particles follow straight-line trajectories until two of them come close enough in space, in which case they are scattered [37, 39].

Prior to the simulation of the collisions, initial conditions of projectile and target are sampled, during an initialization step. Each cascade occurs in the spherical mean-field potential of the target, with a diffuse surface. A sketch of this surface is shown in figure 3.3, on the left. The nucleons themselves are treated as a free Fermi gas and individual particles can move inside the nucleus. Particles with a momentum between p and $p + dp$ are able to reach a maximum radial distance between $R(p)$ and $R(p + dp)$. The correlation between range and momentum is illustrated in figure 3.3 [39]. Because this range function defines the boundary of the target, each particle with momentum p is going to be reflected back into the nucleus, once it reaches a radial distance of $R(p)$. Particles with a larger momentum can reach further until they will be reflected. If a particle with an energy large enough to escape reaches it's maximum radial distance in the potential it can break free and exit the nucleus instead of being reflected back into it [39]. Target nucleons are prepared first, with an initial momentum and position. First, the momentum is chosen at random from within a Fermi sphere with a radius of p_F. Afterwards, the value of $R(p)$ is calculated based on the range-momentum correlation and the position of the nucleon is randomly chosen from a sphere with radius $R(p)$. Additional details on the mathematics of this function can be found in the original paper [39]. Then, the relative positions and momenta of the projectile nucleons are chosen after random sampling of the impact parameter. In case of an ion the nucleons of the projectile are distributed close together in space to account for their binding. Also, the sum of the four-momenta of the projectile nucleons is set to be identical with the four-momentum of the projectile [25, 37].

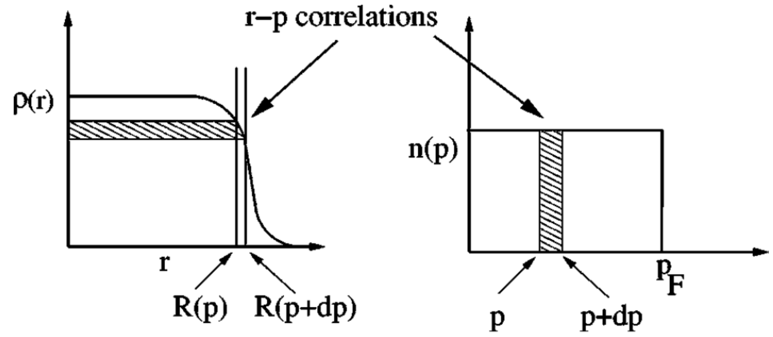


Figure 3.3.: Saxon woods density distribution and correlation between spatial and momentum distributions. Taken from [39].

After initializing projectile and target, the model assumes that all of the projectile nucleons move with a similar velocity until they suffer a collision. Projectile nucleons with a trajectory that do not intersect the calculation area are considered *geometrical spectators*, while those who do intersect are called *geometrical participants* [37]. If no participants are sampled at this point, the collision event is discarded right away and the projectile continues with its initial trajectory. In case the event is not discarded, geometrical spectators are grouped together to form a pre-fragment that is emitted from the area as new quasi-projectile [37]. All target nucleons are initially labeled *spectators*. Spectator particles are moving during the simulation, but they do not interact with other spectators. Once a spectator interacts with a participant, they are both considered participants for future collisions [37, 39].

Each particle moves on a straight trajectory, which means that any pair of trajectories can be checked to calculate the time of the next collision. This is used during simulation, in order to propagate all of the particles at once, to their respective locations during said collision. At this point the interaction of the colliding particles is carried out, scattering the two. The new trajectories are calculated and the process begins anew. The cascade usually ends when there are no participants left in the nucleus. Also, at some specific time called the stopping time t_{stop}, the cascade has to be stopped in order to give way to other physical processes. In the `INCL++` model, this time is a parameter, however it was set to a function given by [39]:

$$t_{\text{stop}} = f_{\text{stop}} \cdot t_0 \cdot \left(\frac{A_T}{208}\right)^{0.16}. \tag{3.6}$$

Here, A_T is the target mass and `Geant4` is using $f_{\text{stop}} = 1$ and $t_0 = 70$ fm/c [25, 39]. According to the original paper [39], f_{stop} and the mean-field potential are the only two free parameters of the model. Both of them are however tied to physical considerations. When a cascade is run for a long time there are two phases of variations in physical quantities such as the excitation energy. As the projectile enters the target, a phase of rapid variations begins. This ends with the emission of fast particles. Then, a phase of much slower variations takes place, which shows effects similar to the cooling of an equilibrated system. The stopping time was selected such that characteristic variations in the second phase are fully established. At this point the cascade is stopped and the simulation continues with a nuclear evaporation model [25, 39]. This is a model responsible for the de-excitation of excited nuclei; in `Geant4` the standard model for the process is called `G4ExcitationHandler` [25].

3.2. GATE

During this work `Geant4` was not used directly, rather the software package `GATE`, version 7.1, which is described in this next section, was implemented to set up and manage the simulations. This allowed for an easier and more convenient way of getting familiar with Monte Carlo simulations, without having to program too much. `GATE` is an open source Monte Carlo simulation tool, which is being developed and maintained by the `OpenGate collaboration` and can be used for a variety of simulations. It is built upon the `Geant4` libraries and allows the user to simulate an experiment using an extended version of the `Geant4` scripting language. `GATE` also adds functionality that supports the simulation of imaging,

Material	Density [g/cm^3]	State	Material	Density [g/cm^3]	State
O	1.141	liquid	C	1.644	solid
H	0.0705	liquid	N	0.808	liquid
Ca	1.54	solid	P	1.83	solid
K	0.862	solid	S	1.96	solid
Na	0.968	solid	Cl	1.563	liquid
Mg	1.738	solid	Hg	13.534	liquid
Sn	7.365	solid	Cu	8.96	solid
Ti	4.506	solid	Al	2.7	solid

Table 3.1.: Target Materials used during the simulations.

radio-therapy and dosimetry in a single environment. While `Geant4` manages low-level features such as the interaction between particles and matter, `GATE` provides additional features as an outer layer, to allow for the design and execution of `Geant4`-based experiments [20].

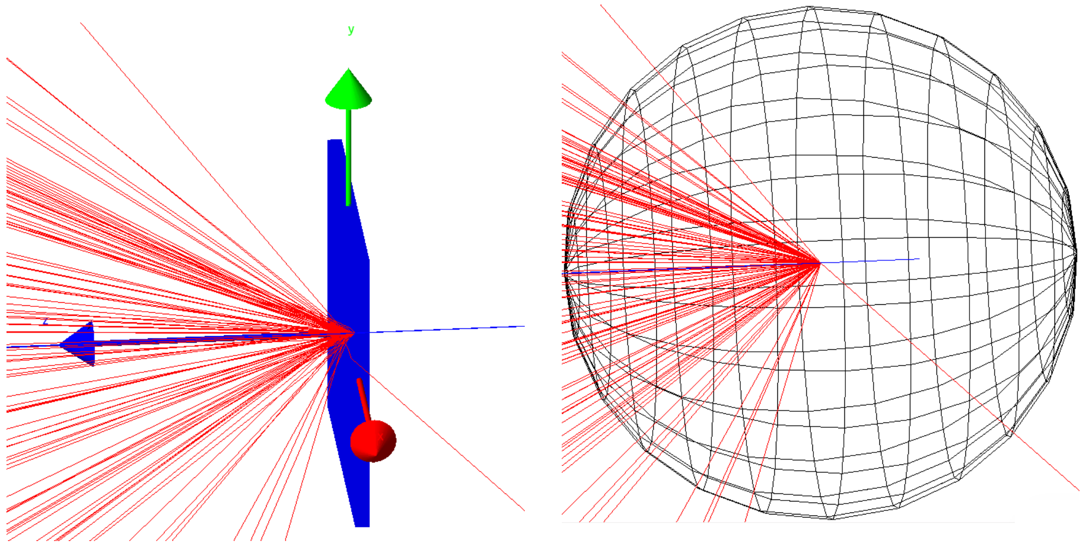
Simulations in `GATE` are controlled by so called *macro-files* which contain a description of the geometry in use and other elements, like the beam sources, targets and detectors [20]. Several distinct macro files were created in order to quickly parameterize a simulation. The parameters had to be changed for all of the iterations of incident energies and target types in this work. Because of that, each of the files only contains instructions that are specific to its purpose. For example, one file describes the experiment's geometry, while another parameterizes the beam source. Quantities, like the incident energy or the number of primary projectiles, were stored in aliases that were defined in a central file called *alias.mac*. All of the other files then had access to the parameters by calling the respective aliases needed. Due to this structure only the macro responsible for aliases had to be updated in between iterations of the simulations. The overall contents of these macro files will be explained in this section and a description of the individual files can be obtained in Appendix B.

The macro files define an object called the beam source, with the shape of a circle with a radius of 0.1 µm. It has a uniform probability distribution for the sampling of primary particles, all of which were emitted from within the source region during the simulation. This means that the primaries had the same probability of being sampled at any point within the circle. They were directed at a thin target located at a distance of 20 mm. Primary particles emitted by the source consisted of carbon ions at an energy between 10 and 500 MeV/nucleon. During each iteration all of the primary particles had exactly the same incidence energy, but the beam energies of the iterations varied (10, 50, 100, 150, ..., 450, 500 MeV/u). A number of 10^9

primaries were simulated during each iteration, as a compromise between hard disk uptake and computing time versus statistical strength.

The target's dimensions were set to 200 µm width and length, as well as a thickness of 10 µm. This was done in order to keep absorption in the target to a minimum. In each simulation permutation, the target consisted of one of the materials specified in table 3.1. These were investigated, because they were assumed to be either abundant in biologic molecules or common implant materials. The target densities of oxygen, hydrogen, nitrogen and chlorine were selected for the element's liquid states in order to guarantee a suitable amount of projectile-target-interactions. This was decided because there would have been much less interactions, had the targets been in a gaseous state.

Figure 3.4a displays a visualization of 1000 tracks in the simulation area, as they went through the sample and produced secondaries in it. The primary beam, which is displayed as a blue line, entered from the negative z-axis, which can be seen on the right, while all the non-primary particles were created along the beam path inside the target. Most of the secondaries were produced in close proximity to the beam axis; due to the short target and the high energy, only very few scatter events could have knocked a particle out of the beam path prior to a nuclear reaction.



(a) Visualization of the ion beam interacting with the target.

(b) Wireframe picture of the spherical detector, which surrounded target and beam source.

Figure 3.4.: Geometry of the simulated Experiments.

To register the outgoing particles, a large spherical detector consisting of *empty space* was placed around both source and target. This is just a custom material made of hydrogen at a density of 10^{-25}, in order to minimize possible interactions in the sphere. A visual representation of the detector sphere can be seen in figure 3.4b. It made sure that all particles which exited the simulation area were recorded, because it was selected as a phase space detector in the macros. The attributes, of each of the particles, that were recorded, were chosen in the macro file responsible for the description of this detector. Relevant data for the cross section calculation were *particle names* and *locations of detection*, as well as *their energies*. Other possible output was deliberately suppressed in order to reduce the amount of hard disk space needed for the simulation output. These data were later used to calculate cross section estimates.

A total of 176 (11 different incident energies × 16 target materials) simulations were executed for the production of estimates, each of them using the `Geant4` physics list `QGSP_INCLXX`. Another 5 simulations with an additional distinct incident energy of 95 MeV/u were carried out for data evaluation purposes. These however, were only run with oxygen, hydrogen, carbon, aluminium and titanium as target materials. A description of the output is provided in section 3.4 and the process of calculating cross sections from this data can be found in section 3.5.

3.3. HTCondor

After creating the macro files that defined the simulations, the computation task had to be split up to be used with a computer cluster. Instead of running one long process for each of the simulations, smaller jobs were distributed over a group of managed CPUs from a computer cluster. This way the calculations were finished sooner and if some jobs would have been erroneous only these would have been repeated.

`GATE` supplies a tool called `Jobsplitter` [40] with which it is able to logically divide macro files and ready them for cluster computing software. Basically, this tool creates a number of identical macros that contain a relative start- and stop-time from the interval [0, 1]. Each of the copies uses the stop-time of the previous file as its own start-time. The first macro starts at the relative time 0, while the last ends at 1. This way, instead of simulating the original task in a single process, each of the "split" macros contained a time window of a hundredth. Although any number of splits would have been possible, a separation into 100 jobs was used for this work. These smaller tasks are referred to as jobs in this work. They can be executed parallel to each other, on more than one CPU.

To handle the jobs, a HTCondor cluster [33] was used. It is a workload management system for jobs that are intensive in computing effort. HTCondor is able to control large amounts of jobs with a queuing mechanism, scheduling policies and resource monitoring and management. Users are able to submit serial and parallel jobs and HTCondor schedules where and when they are going to be executed. For the simulation task that arose during this work only the processors of a single computer were used, however HTCondor is also able to deal with a computer cluster, such as a grid, by distributing the jobs over a network [33].

Submitting a job to `HTCondor` is carried out using a script called a submit-file [41]. Such a file needs to specify several parameters, as shown here.

```
# Submit description file for Gate
executable  = Path/To/Gate
universe    = vanilla

Arguments   = Path/To/main1.mac
Output      = Path/To/main1.out
Error       = Path/To/main1.err
Log         = Path/To/main1.log
Queue


...
Arguments   = Path/To/main100.mac
Output      = Path/To/main100.out
Error       = Path/To/main100.err
Log         = Path/To/main100.log
Queue
```

Figure 3.5.: Example submit file for HTCondor. This file needs to specify parameters that can completely characterize a job.

The example in figure 3.5 demonstrates several of the contents that are mandatory. *Executable* must point to the application that carries out the job. To carry out simulations it should point to the `GATE` binaries. *Universe = vanilla* is used to keep things simple. It is possible to add further functionality like job checkpointing to programs by compiling them with `HTCondor` provided compilers [33], but this method was not used here. Before queuing the individual jobs occurs (which starts at the first *Arguments*-field in figure 3.5), it is possible to add lots of additional parameters. These may add fine-tuning options to control how the jobs are executed and depend on the system and requirements from users. A more detailed introduction to submit files will not be included in this work, but it can be found on the condor web page [41].

After the parameters for the submission are listed, the submit file must describe the job execution itself. The following parameters can be repeatedly declared, since more than one job may be queued by a submit file. For each job, the *Arguments* parameter points to the main macro file for `GATE`. *Output* is a file that is created during the execution of the job. In it, output from the standard output stream, like cout in C++, is stored. Similarly *Error* is used to store the standard output stream for errors. *Log* is another file created during execution, however, it contains logs made by `HTCondor`. The last statement after each job's parameters is *Queue* [41].

3.4. ROOT

Simulation output from `GATE` is provided in a `ROOT` file. `ROOT` itself is a framework suitable for large scale data analysis, owing to the fact that it contains a hierarchical database and tools to visualize and analyze data. Once installed, `ROOT` offers a graphical user interface and the ability to use C++ scripts due to its own interpreter. It was developed to handle massive amounts of data at CERN and is used by thousands of physicists to analyze their data. The output file is similar to a UNIX file system in that it can contain directories and objects with an unrestricted amount of levels [32].

After the first simulations finished, the `ROOT` files were able to be visualized using the *TBrowser*, in order to verify the data. It quickly became clear that the contents of it were corresponding to `GATE`'s simulation output; a phase space which contained data for all tracks that passed the spherical detector object. Every track was stored with some key attributes like particle energy, location during the detection and name. Some of the particles' names were spelled out, while others followed a common scheme. Single component particles are referred to by their names, such as proton or electron. Ions, on the other hand, or described by chemical symbol and mass number, like C12 for carbon-12. There are however three exceptions; the ions deuteron, triton and the alpha particle are referred to by their names as well. In addition to names, energies and locations, several IDs used by `GATE` were also included in the files. In the case of this work the results of every simulation were stored in 100 separate `ROOT` files, due to the job splitting.

Using the `ROOT` *TBrowser*, it is possible to view the distributions of these attributes right after the simulations [42]. The browser shown in figure 3.6 for example displays the frequency distribution, that is the number of detected particles as a

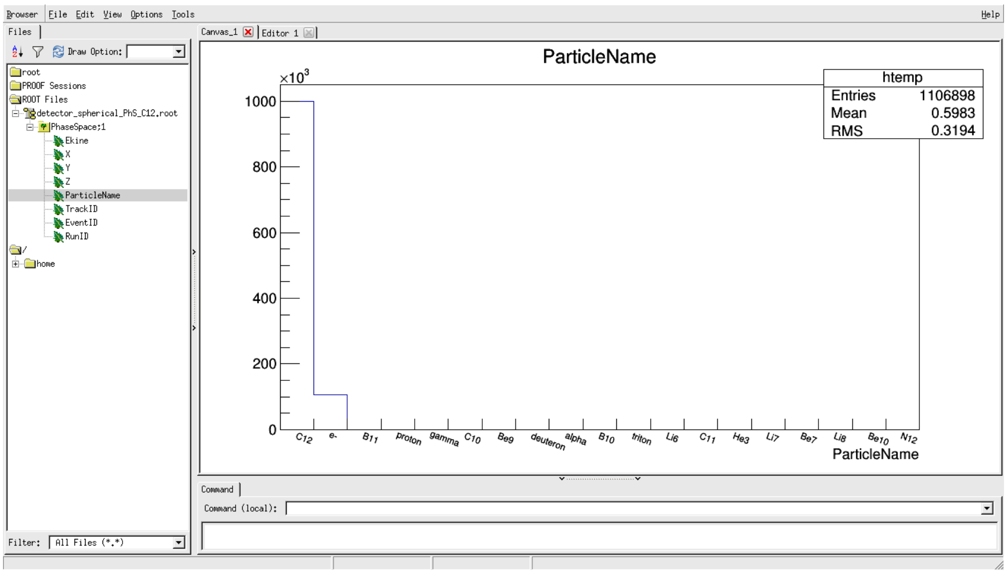


Figure 3.6.: Simulation Output - PhaseSpace. The image shows a `ROOT` *TBrowser* [32], that can be used to display the contents of `ROOT` files. Distributions of particle energies, locations and names along with several IDs can be selected. Here, the distribution of particle names is shown. The names found in the file are listed on the x-axis and the total number of particles with this name is marked on the y-axis.

function of their name, from an output file produced during this work. All particle names that were recorded from the spherical detector can be found on the x-axis of the histogram, while the heights of the bars correspond to their frequency. In the *TBrowser* it is possible to zoom in on specific regions of the graph to obtain better readings. Although the distributions of particle names, energies and locations could be visualized right after the simulations, cross sections were not possible to determine directly. It would be feasible to select particles based on energy and location and calculate cross sections for them from within `ROOT`, using a C++ script. However, this would require the user to run a script for every simulation, by hand. Also, every file contained only roughly a hundredth of the total tracks of a simulation, due to the job splitting, so the output would have needed to be combined before the cross section calculations. Considering that 181 × 100 files were produced, this would have resulted in unreasonable effort.

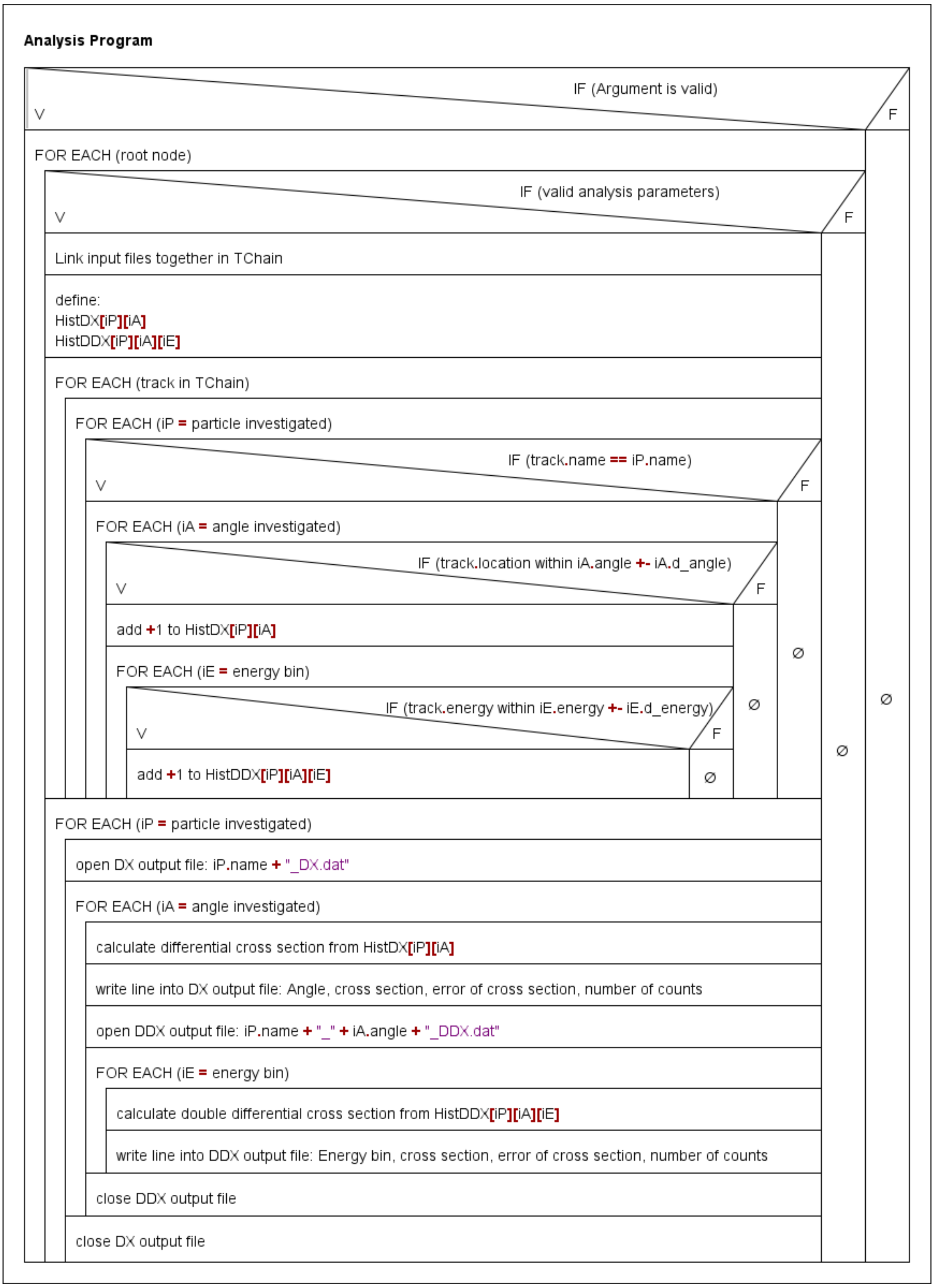
Although there are tools that can combine several `ROOT` files into a single one, their execution used up a lot of resources, mainly due to the large number of files that needed to be decompressed before and re-compressed after merging. Even after successfully merging the files it would have still been necessary to start 181 scripts by hand. So, instead of merging the files, an analysis software had to be developed as a separate program in order to solve both of these problems at once. It is described in the following section.

3.5. Analysis Program

To calculate double differential cross sections for select particles, an analysis program that filters the phase space, had to be developed. The program was built upon the `ROOT` [32] libraries. It is able to link together a variable number of `ROOT` files and to select particles from those files by their name, energy and location to count them for all of the files linked together. After the counting process the program calculates cross sections from the obtained histograms and saves the estimates to data files. A sketch of the program is provided in figure 3.7. It illustrates the whole process in a single graphic without going into too much detail about the individual functions. In addition to the description of the process featured in this section, an overview of the source files and the separate functions can be found in Appendix C.

To run the analysis program, it has to be started with an argument, namely a path which points to a configuration *XML* file. If the argument is missing, not a valid file path or it points to an inaccessible file, the program closes with a warning. Otherwise, it tries to open the configuration file, which is expected to contain parameters that describe one or more analyses. The file can have any name, however it must be an *XML* file. To parse its contents, the `pugixml` libraries were used as a *XML* parser. `pugixml` is a small library written in C++, which supplies read and write methods for *XML* files. It is distributed under MIT license, making it freely available for a lot of different uses [43].

After successfully parsing the *XML* file, the program treats all nodes at the root level as a container for the parameters of a single analysis. Each of the root nodes is opened once, by a loop. The program then verifies whether all the necessary parameters were provided in order to carry out an analysis. This includes the path to the input files and their quantity, the path to the output, the molar mass, density and thickness of the target, the number of primaries during the simulation, the energy binning parameters for double differential cross sections, the fragments being counted, the solid angle elements and whether double differential cross sections, differential cross sections or both are going to be calculated. If the parameters are erroneous or insufficient, the program will skip to the next analysis by jumping to the next root node in the first loop.



```
Analysis Program
IF (Argument is valid)
  V:
    FOR EACH (root node)
      IF (valid analysis parameters)
        V:
          Link input files together in TChain
          define:
          HistDX[iP][iA]
          HistDDX[iP][iA][iE]
          FOR EACH (track in TChain)
            FOR EACH (iP = particle investigated)
              IF (track.name == iP.name)
                V:
                  FOR EACH (iA = angle investigated)
                    IF (track.location within iA.angle +- iA.d_angle)
                      V:
                        add +1 to HistDX[iP][iA]
                        FOR EACH (iE = energy bin)
                          IF (track.energy within iE.energy +- iE.d_energy)
                            V:
                              add +1 to HistDDX[iP][iA][iE]
                            F:
                              ∅
                      F:
                        ∅
                F:
                  ∅
          FOR EACH (iP = particle investigated)
            open DX output file: iP.name + "_DX.dat"
            FOR EACH (iA = angle investigated)
              calculate differential cross section from HistDX[iP][iA]
              write line into DX output file: Angle, cross section, error of cross section, number of counts
              open DDX output file: iP.name + "_" + iA.angle + "_DDX.dat"
              FOR EACH (iE = energy bin)
                calculate double differential cross section from HistDDX[iP][iA][iE]
                write line into DDX output file: Energy bin, cross section, error of cross section, number of counts
              close DDX output file
            close DX output file
        F:
          ∅
  F:
    ∅
```

Figure 3.7.: Schematic of the analysis program. A detailed explanation can be found in the text in section 3.5.

After the program confirms the presence of all the necessary parameters, it verifies that they contain suitable values. For example, if a parameter would contain a negative value, the program would skip the current analysis. Figure 3.8 shows the structure of an example *configuration.xml*, which will be explained here. A more detailed description of the requirements for these configuration files is given in section D of the appendix. The example parameterizes a single analysis, similar to one from this work, which was used for simulation output from carbon-carbon collisions at 1200 MeV. In it, the input for the analysis consists of one hundred files named *Input1.root* to *Input100.root*, which can be found in the directory *Path/To/*, relative to the running analysis program.

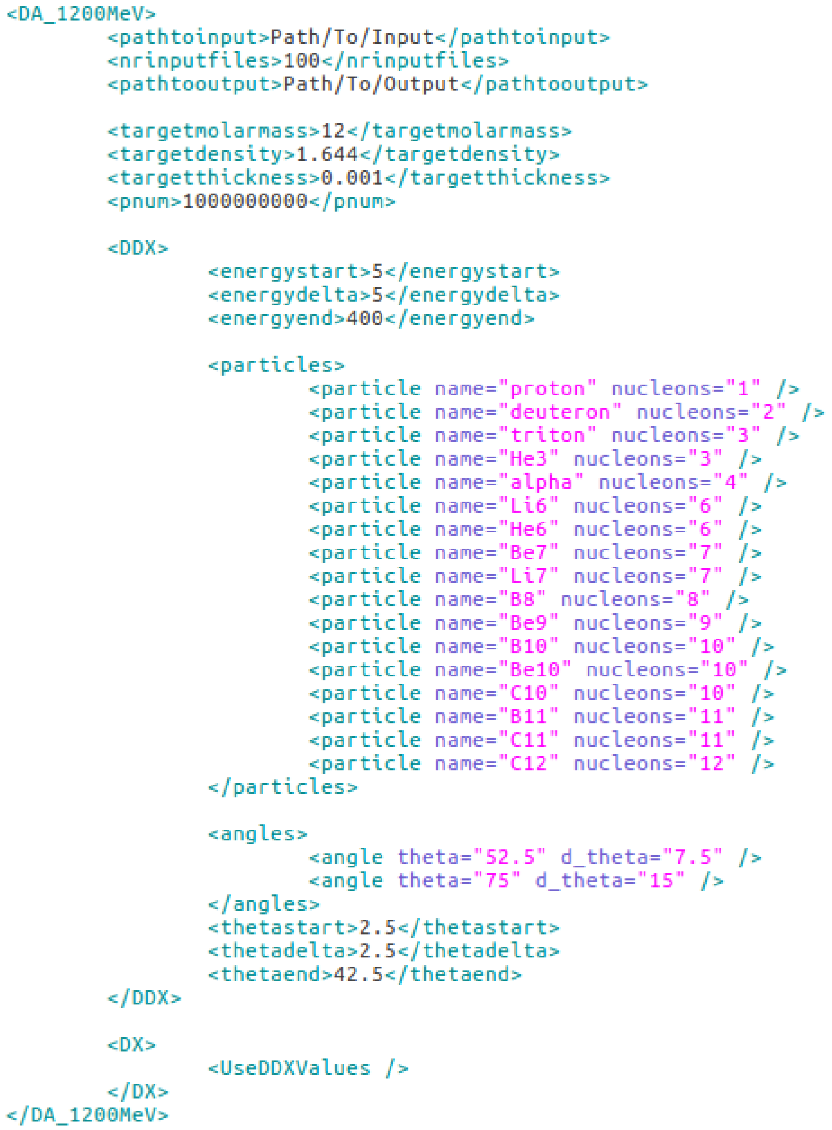
Please note that these names were specifically chosen to simplify the example here. After finishing counting and the cross section calculations, the program would put its output into the same directory, for this example, but with the file names constructed according to these two masks:

- *Path/To/Output_PARTICLENAME_ANGLE_DDX.dat*
- *Path/To/Output_PARTICLENAME_DX.dat*

Here, for each file, *PARTICLENAME* would be replaced by the respective particle name - for example *proton* - and *ANGLE* by the angle *theta* - like *2.5deg*.

The example configures an analysis for 11 different angle elements (2.5, 7.5, 12.5, 17.5, 22.5, 27.5, 32.5, 37.5 and 42.5 degrees, each with ± 2.5 degrees, 52.5 ± 7.5 degrees and 75 ± 15 degrees) and 17 particles (proton, deuteron, triton, He3, alpha, Li6, He6, Be7, Li7, B8, Be9, B10, Be10, C10, B11, C11 and C12 – note how some particles carry names, while others are described by their symbol and mass number. These names were found in the simulation output files). The program would write these into a total of 187 (17 particles × 11 angles) double differential and 17 differential cross section files. Estimates produced in this work used the same angle elements and particles shown in the example above. The number of nucleons for protons, deuterons, tritons and alpha particles is 1, 2, 3 and 4 respectively. The names for the other ions have this number included, however it is still mandatory to correctly list the quantity as an attribute of the *particle* node in the configuration file.

Additional parameters are necessary to completely configure an analysis. The molar mass, density and thickness of the target and the number of primaries were listed after the value `pathtooutput`, in the example from figure 3.8. These three values are necessary to correctly scale the cross sections during their calculations. They must be larger than zero and should be set to the values used in the `GATE` macros for the simulations, to avoid introducing mistakes.



```
<DA_1200MeV>
        <pathtoinput>Path/To/Input</pathtoinput>
        <nrinputfiles>100</nrinputfiles>
        <pathtooutput>Path/To/Output</pathtooutput>

        <targetmolarmass>12</targetmolarmass>
        <targetdensity>1.644</targetdensity>
        <targetthickness>0.001</targetthickness>
        <pnum>1000000000</pnum>

        <DDX>
                <energystart>5</energystart>
                <energydelta>5</energydelta>
                <energyend>400</energyend>

                <particles>
                        <particle name="proton" nucleons="1" />
                        <particle name="deuteron" nucleons="2" />
                        <particle name="triton" nucleons="3" />
                        <particle name="He3" nucleons="3" />
                        <particle name="alpha" nucleons="4" />
                        <particle name="Li6" nucleons="6" />
                        <particle name="He6" nucleons="6" />
                        <particle name="Be7" nucleons="7" />
                        <particle name="Li7" nucleons="7" />
                        <particle name="B8" nucleons="8" />
                        <particle name="Be9" nucleons="9" />
                        <particle name="B10" nucleons="10" />
                        <particle name="Be10" nucleons="10" />
                        <particle name="C10" nucleons="10" />
                        <particle name="B11" nucleons="11" />
                        <particle name="C11" nucleons="11" />
                        <particle name="C12" nucleons="12" />
                </particles>

                <angles>
                        <angle theta="52.5" d_theta="7.5" />
                        <angle theta="75" d_theta="15" />
                </angles>
                <thetastart>2.5</thetastart>
                <thetadelta>2.5</thetadelta>
                <thetaend>42.5</thetaend>
        </DDX>

        <DX>
                <UseDDXValues />
        </DX>
</DA_1200MeV>
```

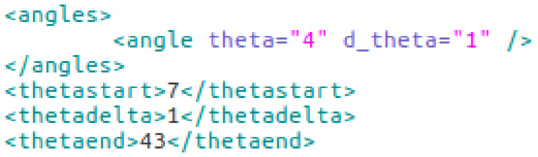
Figure 3.8.: Example XML configuration file for 1200 MeV carbon-carbon collisions.

Also, energy binning information is mandatory as a part of double differential cross section calculations. While the example above features only a single incident energy, eleven different energies were used during the simulations. Additionally, a series of five simulations, using another distinct energy of 1140 MeV, was carried out to compare the estimated cross sections to data from the literature. In order to keep the output of the analyses similar to each other, a variable energy binning was used instead of a fixed bin size and energy maximum. A total number of 40

Incidence Energy [MeV]	Energy-Start [MeV/u]	Energy-Delta (MeV/u)	Energy-End (MeV/u)
120	0.5	0.5	40
600	2.5	2.5	200
1200	5	5	400
1800	7.5	7.5	600
2400	10	10	800
3000	12.5	12.5	1000
3600	15	15	1200
4200	17.5	17.5	1400
4800	20	20	1600
5400	22.5	22.5	1800
6000	25	25	2000
1140	5	5	400

Table 3.2.: Variable bin size used for energy binning during analysis.

bins was chosen for each of the analyses, regardless of the incident energy. This resulted in different bin sizes and maximum energies, depending on the incident energy. While fragments from 1200 MeV incidents were sorted into energy bins with a size of 10 MeV/u, fragments from 4800 MeV incidents were collected in bins with a size of 40 MeV/u. A complete overview of the energy binning parameters is provided by table 3.2. The last entry – listing 1140 MeV – was used specifically to test the agreement between estimates and data from the literature, while the rest of these entries were necessary for the main goal of this work, the production of cross section estimates.



```
<angles>
        <angle theta="4" d_theta="1" />
</angles>
<thetastart>7</thetastart>
<thetadelta>1</thetadelta>
<thetaend>43</thetaend>
```

Figure 3.9.: Angle binning configuration for the comparison to data from the E600 [21] experiment.

Besides the different incidence energy used during simulations, the series used for data validation also differed in the angle binning in the analyses. It was necessary to match the analysis to the conditions of the experiment in question [21], so the angle elements differ between this series and the rest of the estimates. Figure 3.9 displays the parameters for the evaluation series, that were used for validation instead of those from the example in figure 3.8. These parameters instruct the analysis program to use angle elements with the mean angles 4, 7, 9, 11, ..., 41, 43 degrees and a spread of ± 1 for each of the elements. So, the series used for data evaluation had a total number of 20 angle elements, instead of eleven.

After verifying the parameters for the analysis, the program tries to locate the specified input `ROOT` files, which contain the simulation output, on the hard disk. Each of the input files is added to a *TChain* [42] object, which is a linked list of `ROOT` files. If a file cannot be found or accessed the program skips to the next analysis, returning an error message.

Then, several arrays are prepared so that every combination of particles and angles specified by the configuration can be counted in a frequency distribution. For double differential cross sections a histogram called `HistDDX[iP][iA][iE]` is used. It is a three dimensional array and represents the number of particles found as a function of their particle name, solid angle element and energy. The three indexes iP, iA and iE are used throughout the program; iP and iA always correspond to different particle names and angle elements, while iE is used for the energy bins.

Also, every particle has to be counted in another histogram for differential cross sections – `HistDX[iP][iA]` – which represents the number of particles as a function of the particle name and solid angle element. It uses the same indexes as the histogram for double differential cross sections, minus the energy binning. Although it would have been possible to simply add up all the counts in the different energy bins of `HistDDX[iP][iA][iE]` for a given particle iP and angle iA to obtain the counts of the angular distribution for the same indexes, this method was deliberately not used. Instead, counting for differential cross sections is separated from the counting for double differential cross sections, in order to enable users to select different settings for the two, while still being able to run the program once.

After preparing the arrays, the *TChain* is passed through, entry by entry, and filtered. These entries are the recorded tracks from the simulation and each of them is accessed only once, to reduce the time spent reading from the hard disk.

First, a filter selects entries by their *Particle Name* to quickly disregard unwanted tracks for the calculation. Every entry is compared to all the particles specified in the configuration by a loop. Then, the entry is simply disregarded for counting if the particle is not found in the configuration.

Second, a solid angle filter sorts out the tracks with valid particles according to their position. Every angle element that was provided in the configuration is tested against the position of the particle. Those that were not recorded inside a polar angle element $\vartheta(iA) \pm \Delta\vartheta(iA)$ are disregarded for the angle element with index iA. In case the particle is found to be within the element, the program increases the count stored at `HistDX[iP][iA]` by one.

At last the particle is sorted into the respective energy-bin of the DDX histogram. Similar to the solid angle filter, a loop runs over all the energy bins that were provided by the configuration. The count stored in `HistDDX[iP][iA][iE]` is increased by one, if the particle has an energy in between the limits of the bin $E(iE) \pm \Delta E$, with $E(iE)$ being the mean energy of the bin with index iE.

The steps above are repeated for every entry in the *TChain*. At some point the program finishes running through it and begins to calculate the cross sections from the counts obtained, according to equations (2.12) and (2.13) from section 2.5. An outer loop runs through the particle names from the configuration. For every name, an output file is opened. This file corresponds to differential cross section data and two header lines that describe this data are written at the beginning.

Then, a second loop runs through the available angle elements. For every angle investigated for differential cross sections a line is added to the above mentioned data file. These lines include the mean polar angle of the solid angle element, the ± spread of the element, the differential cross section $d\sigma/d\Omega$, the uncertainty of the cross section and the total number of particles that were counted in this element. By reading the value stored at `HistDX[iP][iA]`, counts can be obtained to calculate the cross sections. Afterwards, another output file is opened, this time corresponding to the double differential cross sections of particles belonging to the index iP at the angle element with the index iA. These files also start with two header lines, similar to differential cross section data files.

Now, a third loop runs through all of the specified energy bins to fill up the double differential cross section data file. Every line contains the mean energy of the energy bin, the bin's ± spread, the double differential cross section $d^2\sigma/(d\Omega\ dE)$, the uncertainty of the cross section and the total number of particles found at `HistDDX[iP][iA][iE]`.

After the calculations, the program attempts to find the next analysis, represented by another node at the root level in the XML file, and if it does, tries to repeat the whole process for it. It terminates after running through all root nodes found in the configuration *XML* file. A single analysis can run for a few hours due to the large number of tracks that need to be processed.

4. Results

A total of 37.689 data files were produced during this work. Of those, 1.785 were generated explicitly for data evaluation. The contents of these data files depend on whether they contain differential or double differential cross sections. Tables 4.1 and 4.2 contain representative cross section estimates from simulated carbon-carbon collisions at 1200 MeV.

Table 4.1 displays double differential cross sections from alpha particles detected at polar angles of 2.5 ± 2.5 degrees. Double differential cross section data files like this always specify a projectile, target, incidence energy, ejected particle and polar angle in its file name. For example, the data file corresponding to the data shown in table 4.1 is called `"C_C_1200MeV_alpha_2.5_DDC.dat"`. Additionally, each file contains a comment with the aforementioned values in the first line. All of the other lines contain 5 values each, representing the energy bin and related data. The first value of each line is the mean energy of the line's energy bin in MeV/u and the second value corresponds to the ± spread of the energy bin, which is also provided in MeV/u. On the third place comes the double differential cross section in $\frac{\mathrm{barn}}{\mathrm{sr}\times\mathrm{MeV/u}}$, followed by the fourth value, which is the uncertainty of this cross section, given in the same unit. Lastly, the total number of particles found inside the angle element described by the file (2.5 ± 2.5 degrees for table 4.1) and in the specified energy bin from the first value of the line, is stored as the fifth value.

Table 4.2, on the other hand, displays differential cross sections of alpha particles. These files contain one less value than double differential cross section files. They specify a projectile, a target, an incidence energy and the fragment particle in the file name and its first line. The file according to table 4.2, for example, is called `"C_C_1200MeV_alpha_DC.dat"`. Similar to double differential cross section files, the other lines contain 5 values, although with different dimensions. The first value contains the mean value of the polar angle element, in degrees, instead of a mean energy. Then, the ± spread of the angle is stored as the second value. This spread is also given in degrees. Cross sections are saved as the third value and their uncertainties as the fourth. These two quantities have the dimension of $\frac{\mathrm{barn}}{\mathrm{sr}}$. Similarly to double differential cross section files, the last number contains the total count of particles detected in the angle element specified by the first value.

Energy [MeV/u]	DDC [$\frac{\text{barn}}{\text{sr}\times\text{MeV/u}}$]	Uncertainty of DDC	Counts
5 ± 5	0.0056778	0.000536502	112
15 ± 5	0.0028389	0.000379364	56
25 ± 5	0.00648891	0.000573544	128
35 ± 5	0.0165264	0.000915315	326
45 ± 5	0.0366015	0.00136217	722
55 ± 5	0.0783739	0.00199327	1546
65 ± 5	0.15107	0.00276739	2980
75 ± 5	0.29048	0.00383742	5730
85 ± 5	0.450777	0.00478037	8892
95 ± 5	0.314053	0.00399009	6195
105 ± 5	0.0823788	0.00204357	1625
115 ± 5	0.0170334	0.000929248	336
125 ± 5	0.0037514	0.000436092	74
135 ± 5	0.000709725	0.000189682	14
145 ± 5	0.000405557	0.000143386	8
155 ± 5	0.000152084	8.78057e-05	3
165 ± 5	0.000152084	8.78057e-05	3

Table 4.1.: Double differential cross sections for alpha particles produced at 2.5 ± 2.5 degrees from carbon-carbon collisions at 1200 MeV.

Angle [deg]	DC [$\frac{\text{barn}}{\text{sr}}$]	Uncertainty of DC	Counts
2.5 ± 2.5	14.5747	0.0859569	28750
7.5 ± 2.5	3.94523	0.0258527	23288
12.5 ± 2.5	1.17122	0.0109388	11464
17.5 ± 2.5	0.494968	0.00603305	6731
22.5 ± 2.5	0.269441	0.00394577	4663
27.5 ± 2.5	0.171059	0.00286213	3572
32.5 ± 2.5	0.120831	0.00222998	2936
37.5 ± 2.5	0.0927348	0.00183534	2553
42.5 ± 2.5	0.0764917	0.00158229	2337
52.5 ± 7.5	0.0593978	0.000743811	6377
75 ± 15	0.0329717	0.000356665	8546

Table 4.2.: Differential cross sections for alpha particles produced from carbon-carbon collisions at 1200 MeV.

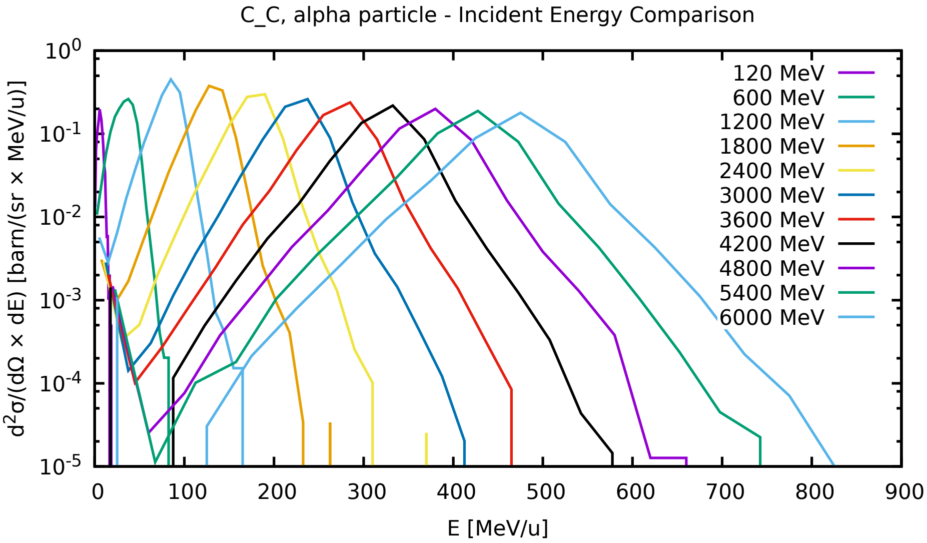


Figure 4.1.: Double differential cross sections for alpha particles produced at 2.5 ± 2.5 degrees from carbon-carbon collisions at different energies.

In addition to tables 4.1 and 4.2, further representations of the cross section estimates can be seen in figure 4.1, showing double differential cross sections and figure 4.2, which displays differential cross sections, both for a variety of incident energies. These images display estimates for carbon-carbon collisions and the production of alpha particles. Data from the analysis output was used to construct these plots, including those shown in tables 4.1 and 4.2. Figure 4.1 shows all the incident energies used in this work, except from 1140 MeV used in section 4.1, while figure 4.2 only displays six different energies to avoid clutter in the image.

Both images show a clear impact of the incidence energy on the resulting cross sections. A different incidence energy leads to another location of the peak in the distribution of fragment energies, which is often found near the specific energy of the incidence particles, at least for small angles like in figure 4.1. This peak shifts to lower energies at higher angles, an observation also described in section 4.2 and illustrated by two additional images in the appendix, figures E.5 and E.6.

Another consequence of higher incidence energies is shown for differential cross sections, in figure 4.2. The production of alpha particles from collisions strongly depends on the incidence energy, as more of the fragments are found at larger angles for smaller energies. Conversely, at higher energies more particles are ejected at smaller angles, almost in the direction of the projectiles before the collision.

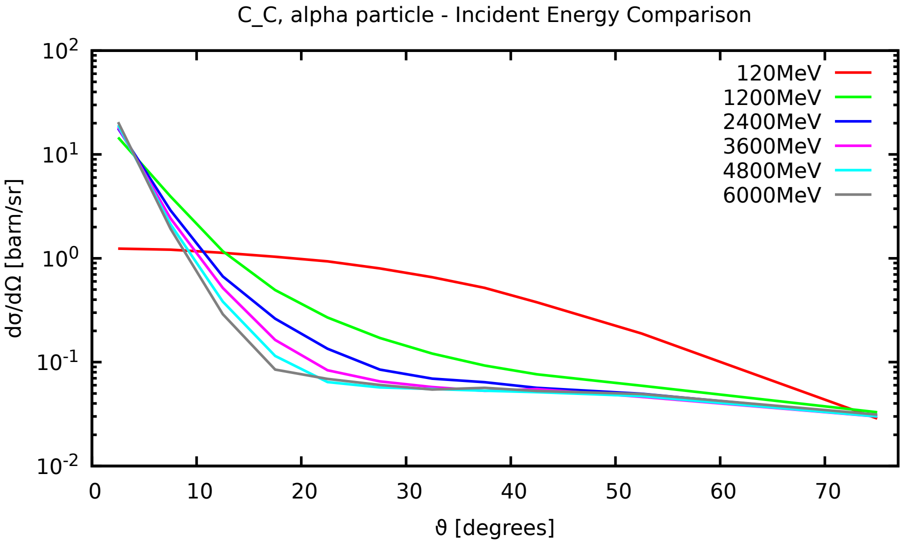


Figure 4.2.: Differential cross sections for alpha particles produced from carbon-carbon collisions at different energies.

4.1. Comparison to Data

One series of simulations, which was carried out using carbon projectiles at an energy of 1140 MeV, was done specifically to test the agreement between simulated cross sections and data from the literature. While Monte Carlo simulation can deliver a large amount of data – the production of which was the main goal of this work – there was a need to examine the quality of these estimates. In order to assess the agreement with measured cross sections and to test the influence of some observables on this agreement, data from experiments performed at GANIL [21] were used.

Figure 4.3 shows a side-by-side comparison of double differential cross sections from data versus estimates from the simulations. In figures found in this section, lines are always used for simulation results while points represent data from the literature. Simulation and analysis parameters were chosen slightly different to the other series of simulations and analyses during this work. While the workload has not changed, the analysis parameters and the incident energy of the simulations was adapted, so as to mirror the conditions from the experiments. An incidence energy of 1140 MeV was used and the smallest polar angle element was 4 ± 1 degrees for the calculation. Greater angle elements had odd mean values starting from 7 degrees and a ± 1 spread, with the largest element ending at 43 ± 1 degrees.

As can be seen in figure 4.3 the cross sections from the simulations are able to provide adequate results. The shapes of the curves from data and simulation estimates are very similar, but the curve representing the latter is shifted to lower energies. In this case, at energies larger than around 170 MeV/u, the simulation did not produce any alpha particles, so there are no more estimated cross sections above certain energies. These limits depend on several parameters though. In the lower energy region, at around 15-18 MeV/u, an abrupt minimum can be seen. This has been observed for many observables and was also discussed in a benchmark study [14]. It seems that the `INCL++` fails to produce enough fragments in the lower energy regions. Double differential cross sections often show a peak at low polar angles, located near the specific energy of the primaries. This peak corresponds to the contribution of quasi-projectiles which only glanced target nuclei, instead of striking them directly, thus keeping most of their energy [14].

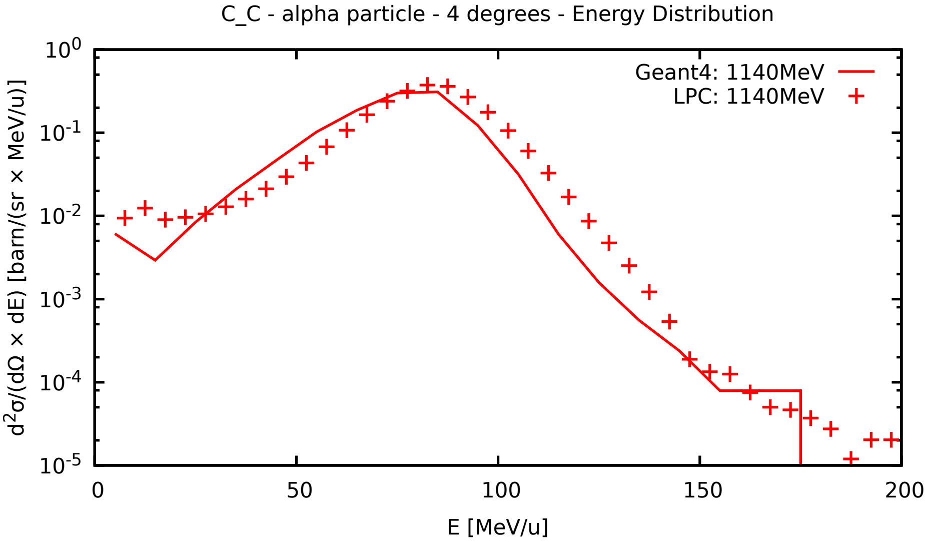


Figure 4.3.: Double differential cross sections for alpha particles produced at 4 ± 1 degrees from carbon-carbon collisions at 1140 MeV.

Similar to double differential cross sections, the same incidence parameters were used for the comparison of differential cross sections. The curves of those estimates also resemble data from the literature, but differential cross sections often tended to be in better agreement with measurement data than double differential cross sections. One example can be seen in Figure 4.4, where the two curves are very similar initially, but the simulation underestimates cross sections for larger angles.

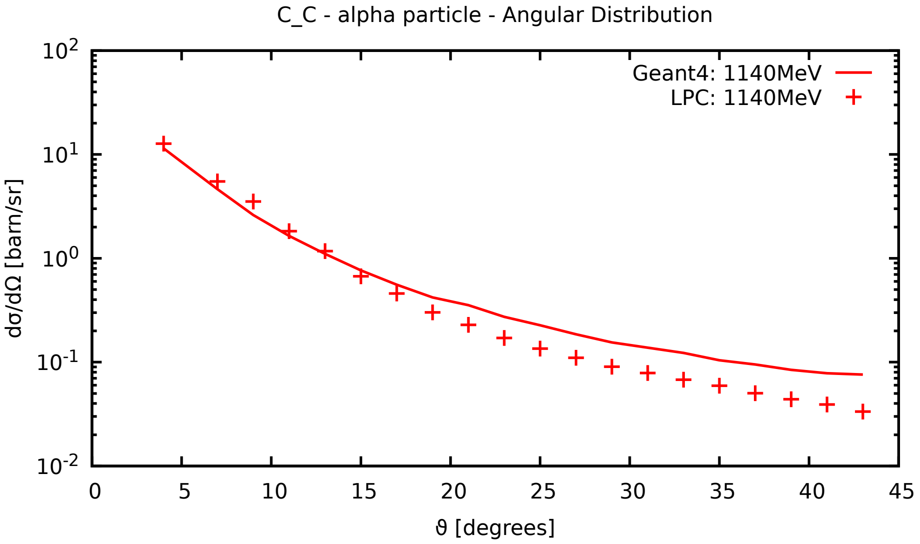


Figure 4.4.: Differential cross sections for alpha particles produced from carbon-carbon collisions at 1140 MeV.

4.2. Influence of Observables

The agreement between simulation data and measurements heavily depends on the parameters of the analysis. To explore and describe this dependency, several figures were produced for a description found in this section. The images featured herein always compare several cross section distributions, that only differ in one observable, to the data. Each of the following subsections is intended to describe one of the observables and its effect on the agreement. Standard conditions for double differential cross section comparisons were chosen to be estimates from carbon-carbon collisions at 1140 MeV, which led to the production of alpha particles at an angle of 4 ± 1 degrees. In each of the following subsections, one of these observables will be varied to expose the effect of this observable on the agreement.

4.2.1. Influence of Different Polar Angles

The first observable that was investigated in was the polar angle in differential cross sections. Figure 4.5 illustrates several plots, each of which compares double differential cross sections, taken at different angles, to the available data. As previously explained in section 4.1, lines represent simulation results, while points were used for data from the literature [21].

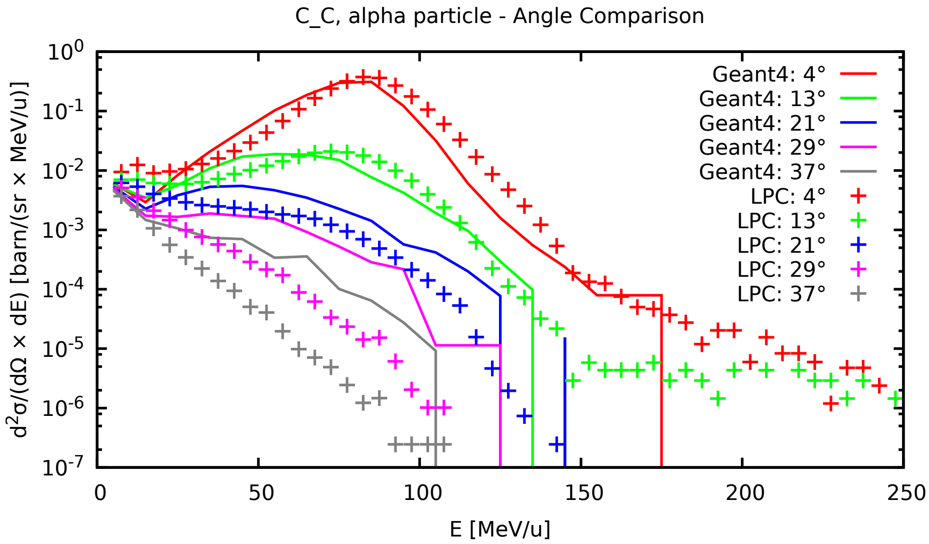


Figure 4.5.: Influence of different angles ϑ on double differential cross sections for alpha particles produced from carbon-carbon collisions at 1140 MeV.

It is obvious from figure 4.5, that the agreement between experimental data and results from simulations gets worse with an increasing detection angle, that is the polar angle θ. For smaller angles, the produced estimates show a curve that is similar to that of measured cross sections, although shifted to lower energies. Cross sections appear overestimated at energies below the peak, which occurs between 50 and 100 MeV/u, and underestimated above it. At 4 ± 1 degrees this peak is at roughly 95 MeV/u, corresponding to the specific energy of the incident projectiles. The contribution of quasi-projectiles is most prevalent at small angles and diminishes as the angle increases [14]. The simulation increasingly overestimates cross sections for all energies at larger polar angles. In figure 4.5, the largest differences can be seen for 37 ± 1 degrees, however even greater discrepancies could be possible, depending on the other observables.

4.2.2. Influence of Different Fragments

Another observable that was examined to test its effect on the agreement is the particle type of the fragments, that were registered by the detector. Figure 4.6 illustrates how the agreement of simulation and measurements for double differential cross sections changes, depending on the particles being discussed. At a polar angle of 4 ± 1 degrees, estimates and data produce similar curves, although the simulation cross sections are shifted to lower energies. Good agreement was observed for carbon-carbon collisions at 4 ± 1 degrees and for ejected alpha particles.

For smaller fragments up to alpha particles, as shown in figure 4.6a, there is no general answer. Proton cross sections are only slightly overestimated for energies between 45 to 75 MeV/u, but significantly underestimated at energies upwards of 75 MeV/u. Disagreement between simulation and measurement increases with rising energies and the largest difference appears at energies above 150 MeV/u, at which point the difference is as large as an order of magnitude.

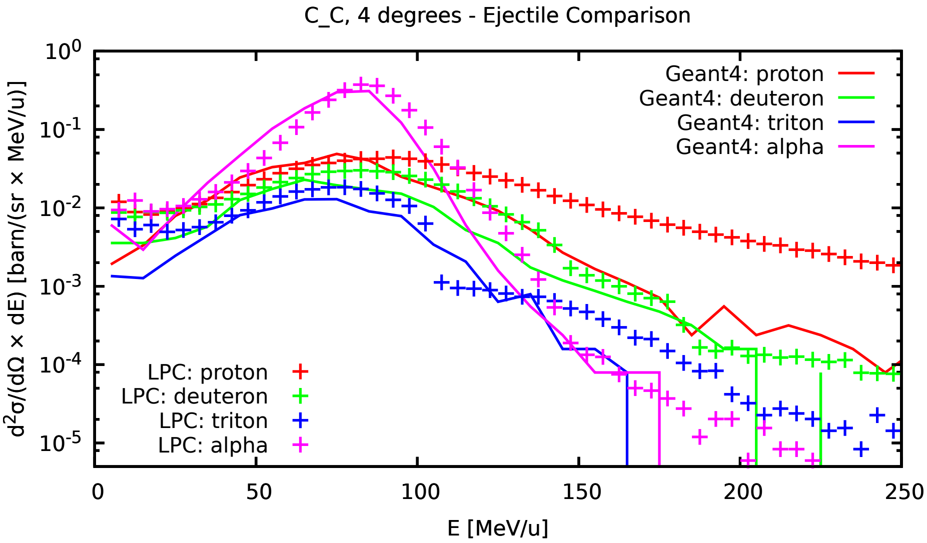
In comparison, deuteron cross sections show a much better agreement than proton cross sections. Simulated deuteron cross sections are underestimated for most of the energies, but the difference is not as large as it is for protons, the biggest being half an order of magnitude at between 115 and 130 MeV/u. The best agreement can be seen between 40 and 60 MeV/u, after which it gets worse. However, starting at around 150 MeV/u, the agreement improves again.

For tritium nuclei the situation worsens a little. Simulated cross sections underestimate real data for a wide range of energies, with a small region between 100 and 120 MeV/u, where data is overestimated a bit. The best agreement can be observed between 40 to 60 MeV/u and again from 120 to 135 MeV/u. Below 40 and above 135 MeV/u the estimates get worse for energies farther out and the largest difference approaches an order of magnitude.

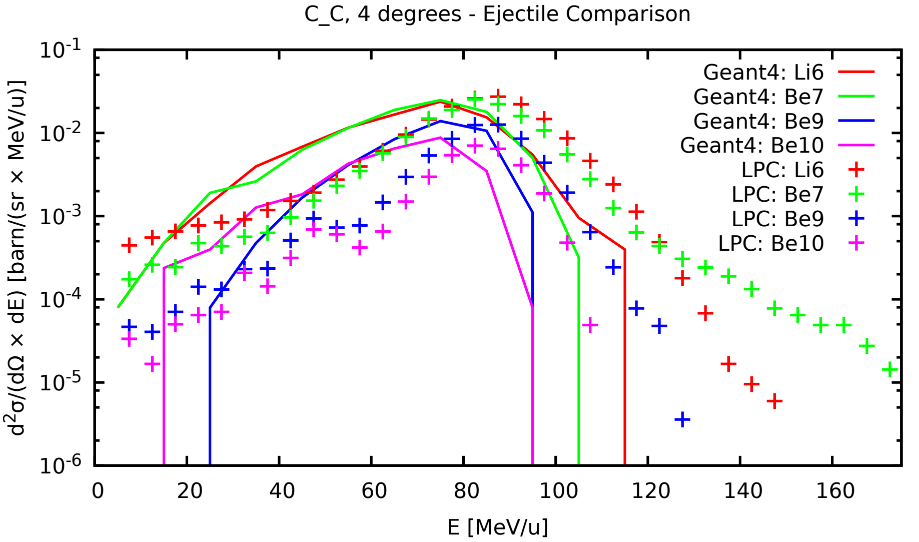
Previously it was mentioned that the best agreement can be observed for alpha particles. The curves of data and estimates are very similar for most energies; the details of their differences were discussed in section 4.1.

Figure 4.6b also contains double differential cross section data and estimates, but for selected particles larger than alpha particles. In contrast to the cases treated before, the agreements between estimates and data are much the same for these fragments. In all of the illustrated cases the curves of data and estimates are similar but the estimated cross sections appear shifted to lower energies, as is the case for alpha particles. All of the chosen particles were overestimated prior to a peak at about 75 MeV/u and underestimated shortly after this peak. The disagreement between data and estimates is similar in shape to the disagreement already described for alpha particles, but the differences are larger for bigger particles. Good agreement can be found only at certain energies.

Between 15 and 25 MeV/u for lithium-6 and beryllium-7, good agreement can be found, because the curves of estimates and data cross. The same can be observed for beryllium-9 between about 25 and 35 MeV/u. Also, for all four shown particles, another small region of energies where good agreement is available is shortly after the peak of each curve, that is between 75 and 90 MeV/u.



(a) Proton, deuteron, triton and alpha particle fragments.



(b) Lithium-6, beryllium-7, beryllium-9 and beryllium-10 fragments.

Figure 4.6.: Influence of different produced fragments on double differential cross sections at 4 ± 1 degrees from carbon-carbon collisions at 1140 MeV.

The influence of different fragments on the agreement differs between double differential and differential cross sections. Figure 4.7 illustrates the angular distribution of cross sections for different fragments. It shows that there is generally better agreement for smaller fragments. For differential cross sections there is no trend of an improving agreement up to alpha particles, as has been the case with double differential cross sections. Despite this, there is generally good agreement at small angles of detection, up to about 7 ± 1 degrees, even for larger ejected particles.

The first part of the image, figure 4.7a, compares the angular distributions of fragments up to the size of alpha particles. In general a good agreement can be observed. For protons only a slight overestimation was provided for angles up to 13 degrees. These differences are minor and the simulations produced estimates that are very similar to the data.

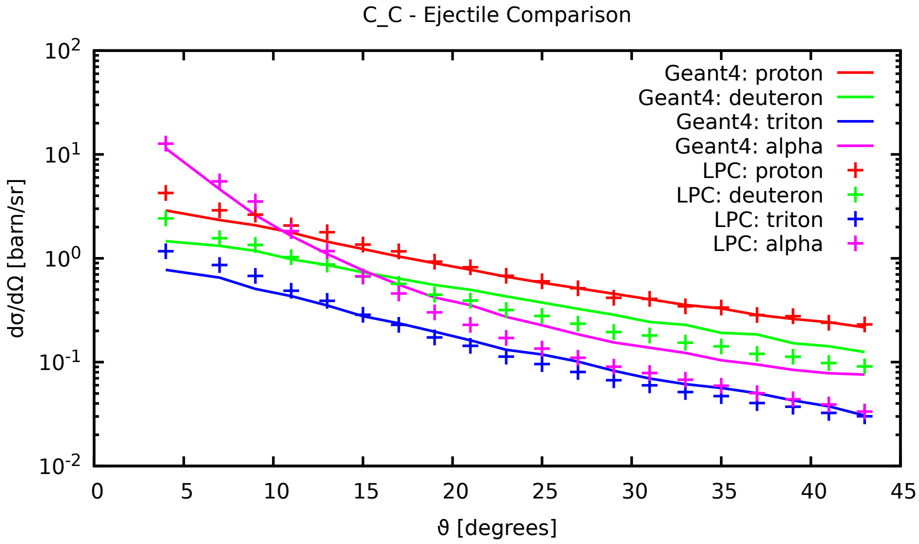
Deuteron production is also slightly overestimated at angles up to 11 degrees. Additionally, their cross sections are also underestimated at all angles larger than 17 degrees, although with an almost constant, small difference.

The situation is a bit better for tritium ions. Cross sections are overestimated at angles below 11 and underestimated between 17 to 43 degrees as well, but the difference in the latter region is smaller, compared to that of deuterons.

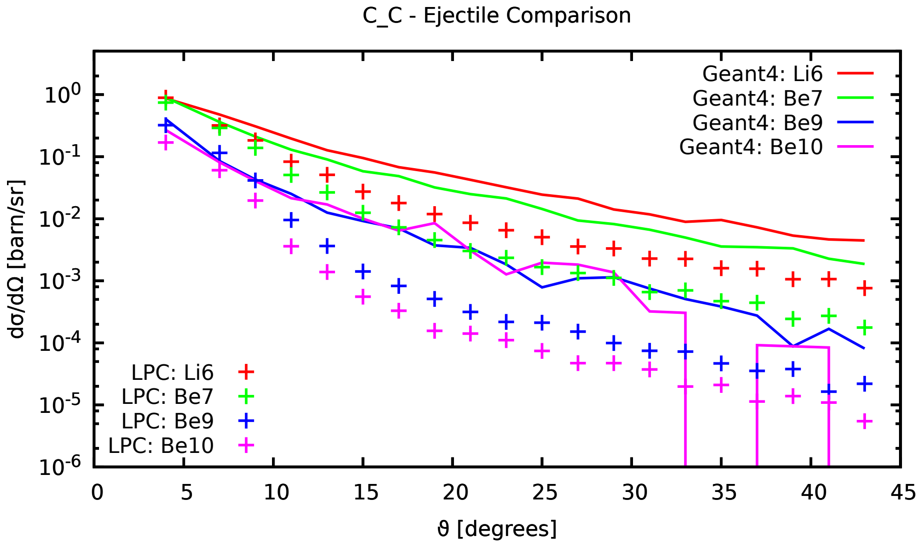
Beginning with alpha particles the estimates increasingly differ from data for larger particles. For angles smaller than 15 degrees only a minor overestimation is present in differential cross sections of alpha particles, however the estimates closely match the data. For angles larger than 15 degrees the simulation underestimates cross sections. The difference between data and estimates increases for larger angles, approaching almost half an order of magnitude at 43 degrees.

Figure 4.7b displays differential cross sections for four different fragments which are larger than alpha particles. All of them continue with a trend that began with alpha particles. Estimates are in good agreement to data only for small angles, up until 7 degrees for lithium-6, beryllium-7 and beryllium-10. For beryllium-9 this good agreement further continues up to 9 degrees.

For larger angles the estimates generally worsen, with an increasing difference. Larger fragments are overestimated much more than smaller ones, reaching a difference of an order of magnitude. The differences initially rise between 7 and 15 degrees. At this point they reach their maximum disagreement, which continues nearly constant to the largest angles.



(a) Proton, deuteron, triton and alpha particle fragments.



(b) Lithium-6, beryllium-7, beryllium-9 and beryllium-10 fragments.

Figure 4.7.: Influence of different produced fragments on differential cross sections from carbon-carbon collisions at 1140 MeV.

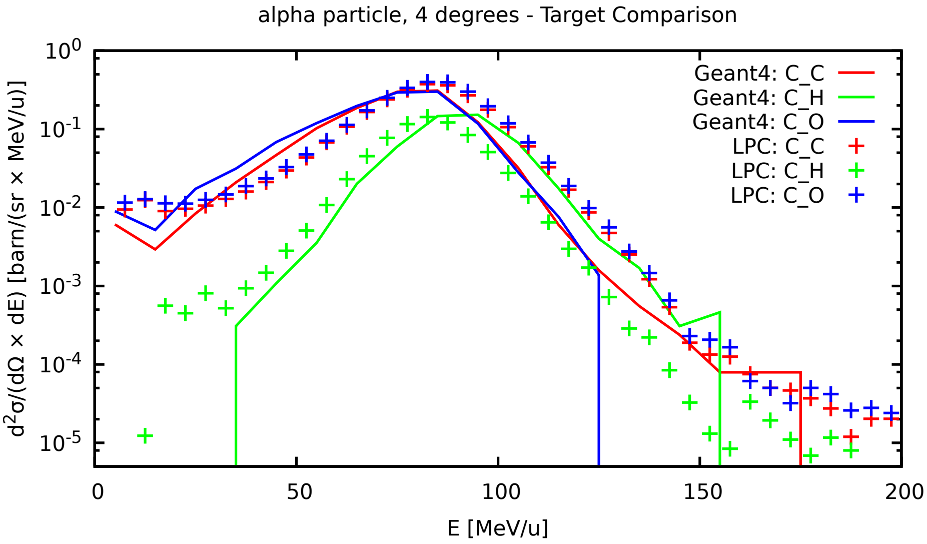
4.2.3. Influence of Different Target Materials

The last observable that was examined to test its influence on the agreement was the target type. Several selected targets are presented here: liquid hydrogen and oxygen, as well as solid carbon, aluminium and titanium. The agreement between double differential estimates and data is represented by figure 4.8. Oddly, the estimates' distributions look fairly similar between carbon and oxygen targets, as well as between aluminium and titanium. Hydrogen as a target stands out, because the agreement in this case is different to the agreements observed in the other four cases. To keep them clear and readable, two images were produced. Cross sections from carbon-carbon collisions can be seen on both of them. This was done to be able to show both the similarity of cross sections from carbon-carbon and carbon-oxygen collisions and the significant difference in the characteristics of the agreement that occurs for particles heavier than A = 18.

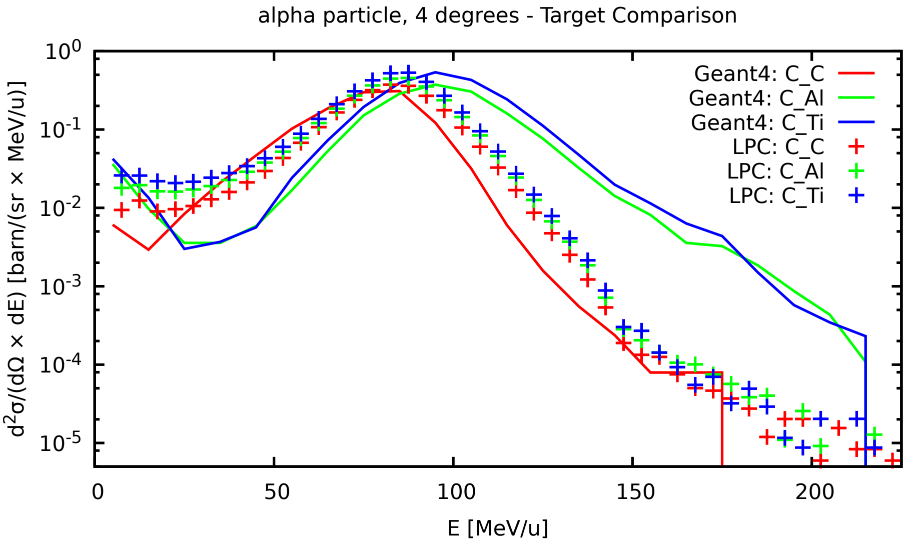
Figure 4.8a shows the estimates and data for carbon-carbon, carbon-hydrogen and carbon-oxygen collisions. For the liquid hydrogen target, the distribution is shifted to higher energies, contrary to previously shown examples. This is an interesting difference, because in most other cases the curves were shifted to lower energies instead. The quality of the agreement between data and estimates depends on the energy, but the overall shape of the curves are very similar to each other. A good agreement is found between 40 and 80 MeV/u, although with a slight but nearly constant underestimation. Below 40 MeV/u the simulation failed to provide any fragments that reached the detector sphere, so no comparison could be made in this case. Above around 80 MeV/u cross sections from the simulation overestimate the data, with a peak at about 90 MeV/u. The difference between estimate and data slightly grows for larger energies, with the largest difference occurring at 155 MeV/u with a sudden and unexpected peak. Afterwards the simulation failed to produce fragments, similar to energies below 40 MeV/u.

As previously mentioned, the situations for carbon and oxygen targets is very alike, probably because these atoms have a comparable size. Measurement data for carbon-carbon and carbon-oxygen collisions with ejected alpha particles closely resemble each other, as do the estimates. However, slightly larger cross sections were produced at energies up to 60 MeV/u for the oxygen target. Both curves contain a local minimum at 18 MeV/u, which has already been mentioned in section 4.1 in the description for figure 4.3. Above 25 and below 70 MeV/u, cross sections are overestimated for both oxygen and carbon targets. Estimates and data then peak at 80 MeV/u, after which cross sections are underestimated for all larger energies. The difference is slowly growing up to 125 MeV/u, at which point the oxygen and carbon-curves start to differ again. There are no more estimates in carbon-oxygen collisions for energies larger than 125 MeV/u. On the other hand, in the carbon-carbon case, the difference between data and estimates begins to shrink again, until there is good agreement for energies larger than 145 MeV/u.

A very different quality of estimates was produced for the aluminium and tita-



(a) Carbon-carbon, carbon-hydrogen and carbon-oxygen collisions.



(b) Carbon-carbon, carbon-aluminium and carbon-titanium collisions

Figure 4.8.: Influence of different targets on double differential cross sections for alpha particles produced at 4 ± 1 degrees from carbon-target collisions at 1140 MeV.

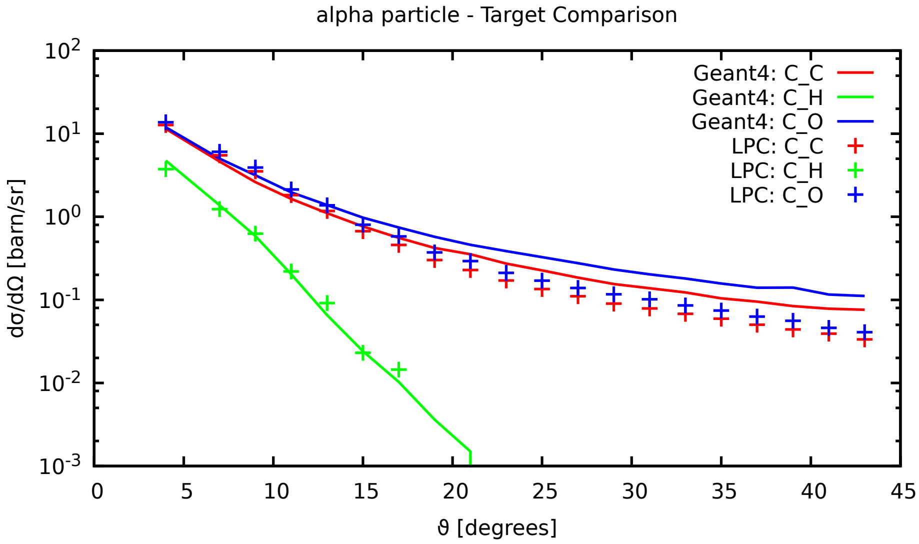
nium targets, shown in figure 4.8b. Safe from two distinct energies – at 10 and at 90 MeV/u, where the curves for estimates and data cross each other – there are larger differences than in previous cases. Cross sections are underestimated below

90 MeV/u and overestimated above. The disagreement increases for larger energies, after 90 MeV/u, and the largest difference is about 1.5 orders of magnitude. It occurs at an energy of 175 MeV/u. In the lower energy regions the largest difference is less than an order of magnitude at about 25 MeV/u. Although this local minimum looks like the same artifact already seen in the carbon-carbon collisions, it is probably caused by another mechanism. This is reasoned here, because the *Binary Cascade Model* (`BIC`) is used for nuclei with A $>$ 18 instead of `INCL++` and the previously seen minimum was produced by the `INCL++`. The authors of the benchmark study [14] argued that the `BIC` model produced worse estimates at lower energies than `INCL++`, even though both clearly underestimate the data in these regions [14]. Similar findings were observed during this work. At energies upwards of this minimum, the disagreement between data and estimates gradually lessens until the curves meet at 90 MeV/u.

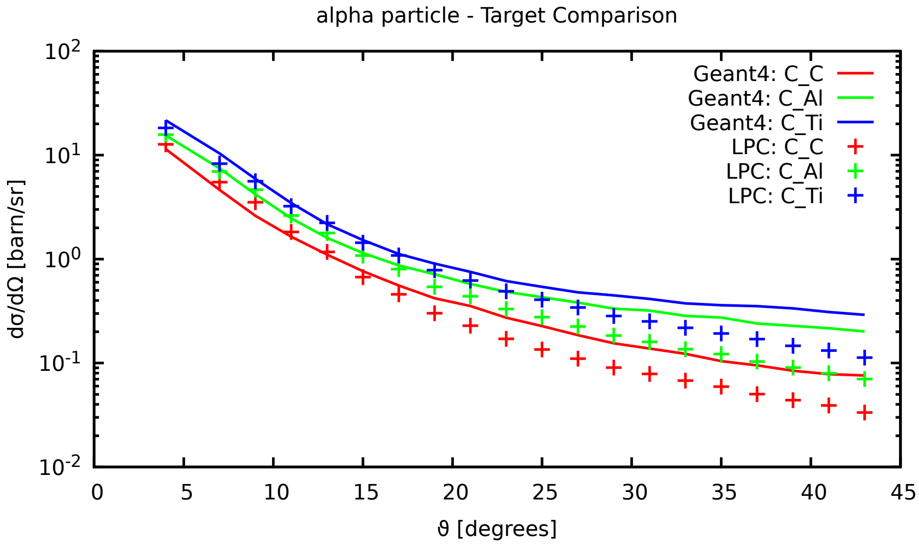
Figure 4.9 compares differential cross section estimates to data from the literature. Again, two images were produced to keep the curves of the individual targets visually apart.

With only minor differences, the agreement between data and estimates is optimal for alpha particle fragments and hydrogen as a target. This can be seen as green lines and points in figure 4.9a. In the other cases there is a trend of worsening agreement for larger targets.

Negligible differences can often be observed at small angles, regardless of the target. In the cases of solid carbon and liquid oxygen targets this good agreement is visible in figure 4.9a up to an angle of about 15 degrees, and in the case of aluminium and titanium in figure 4.9b, up to an angle of 19 degrees. Afterwards the cross sections are generally overestimated in the simulations; a difference builds up for increasing angles for all of the targets besides hydrogen. The largest difference is about half an order of magnitude for oxygen and carbon targets. For aluminium and titanium this difference starts to approach an order of magnitude instead. In each case the maximum disagreement can be found at the largest angle, 43 degrees. However, this trend is likely to continue above the investigated angles.



(a) Carbon-carbon, carbon-hydrogen and carbon-oxygen collisions.



(b) Carbon-carbon, carbon-aluminium and carbon-titanium collisions.

Figure 4.9.: Influence of different targets on differential cross sections for alpha particles produced from carbon-target collisions at 1140 MeV.

5. Discussion and Conclusion

The primary goal of this work was the simulation of the production of cross sections for high energy carbon ions impinging on several targets that are important for ion therapy. Although this goal was achieved using `GATE` simulations and a custom made analysis software, there are several things to consider when dealing with these estimates. Selected curves were presented in chapter 4, in order to highlight the strengths and weaknesses of the produced cross sections. To do that, they were compared to data from the literature. Additionally, the influence of several observables was shown in order to be able to discuss them.

Estimates from simulations generally featured curves that were very similar in shape, compared to measurement data. Although the curves were often shifted to lower energies, overestimating cross sections at low energy and underestimated them at higher ones, the differences were small in some scenarios. For small detection angles the agreement between data and estimates was better than for larger angles. The best agreement has been shown for polar angles up to about 15 degrees, but upwards of that the agreement continually worsened.

The particle type of fragments also played a major role. Beginning with double differential cross sections for protons, at first the agreement improved for larger fragments and the smallest differences were described for alpha particles. Afterwards, the tendency reversed and differences between data and estimates grew as the size of the fragments increased. The cross sections were increasingly overestimated at low energy and underestimated at higher ones.

Oddly, the material of the target did play a much smaller role. It has been shown that the double differential cross section curve was shifted to higher energies instead of lower ones, when using hydrogen as a target. The agreement was very similar for different targets, although not in the case of titanium and aluminium, where large differences were observed. It should be noted though that a comparison between those two and the smaller targets is not useful, given that they are heavier than $A = 18$ and so the Binary Cascade Model was likely used by `Geant4`.

The angular distributions caused a different impression. Overall, similar tendencies have been observed for differential cross sections, as they often had common good agreement even amongst different influences. Even though the estimates tended to generally agree with the data, they also produced significant differences for larger particles, targets and angles. However, at the smallest angles, that is up to about 9 degrees, most of the differential cross section estimates showed good agreement, regardless of the observable. Disagreement was almost absent in one case, namely for carbon-hydrogen collisions and the production of alpha particles. Besides that, the agreement often distinctively worsened for angles larger than 15 degrees. In these cases the simulations overestimated cross sections for most observables, often with an increasing difference towards the largest angles.

Obtaining cross section estimates for the production of fragments in high energy collisions is a simple task using `Geant4`, `GATE` and a counting program. Even though such a simulation and the following analysis takes many hours to finish, this is still a cheap and relatively quick method, compared to measurements. However, this approach leads to estimates that might disagree with experimental data. Further development to enhance the intra-nuclear cascade models is needed, so that future implementations can provide more accurate estimates. For example, the comparisons shown in chapter 4 showed that the `INCL++` model often produced good double differential cross section estimates at energies near the specific energy of primary particles. At lower or higher energies, large angles or for larger particles however, the model does not produce accurate predictions. Similar findings were reported in a benchmark study [14]. By improving the physics models, the estimates produced by them could then better reproduce the data. Of course there are also other simulation frameworks and physics models available.

Additionally, improving the understanding of the strengths and weaknesses of the physics models could lead to a better utilization of them. By using different models for the same process it would be possible to determine optimal observables for individual models. Then it might be possible to estimate cross sections using those models that were found to be more accurate for a set of observables. This, however, is a very complicated task, because the observable dependencies of the agreement are different amongst varying models [14].

Altogether it was shown that the estimates produced in this work varied in accuracy, depending on certain observables. Large discrepancies arise for larger particles, angles and energies, so the estimates are not suited for direct use in these cases. The most striking room for improvement is the production of fragments at large angles. Despite the discrepancies, simulating cross section production is very convenient for comparisons with other simulation frameworks, physics models and data in order to select the best approach for a given problem on a case-by-case basis. To conclude, it is important to mention that the quality of estimates produced using the discussed methods strongly depends on the mechanisms and input data that created these estimates. Because `Geant4` internally relies on empirical parameterizations of total reaction cross sections [25] and because the physics models are not without flaws, the estimates are likely to contain inaccuracies, at least to a degree.

Bibliography

[1] Dieter Schardt, Thilo Elsässer, and Daniela Schulz-Ertner. “Heavy-ion tumor therapy: Physical and radiobiological benefits”. In: *Rev. Mod. Phys.* 82 (1 Feb. 2010), pp. 383–425. DOI: `10.1103/RevModPhys.82.383`.

[2] Ugo Amaldi and Gerhard Kraft. “Radiotherapy with beams of carbon ions”. In: *Reports on Progress in Physics* 68.8 (2005), p. 1861. URL: `http://stacks.iop.org/0034-4885/68/i=8/a=R04`.

[3] *Particle Therapy Co-Operative Group*. [Online; accessed 16-August 2016]. July 2016. URL: `http://www.ptcog.ch/`.

[4] Martin Jermann. “Particle Therapy Statistics in 2014”. In: (2015), pp. 50–54. DOI: `10.14338/IJPT-15-00013`.

[5] *MedAustron Web Page*. [Online; accessed 16-August 2016]. URL: `https://www.medaustron.at`.

[6] M. Benedikt and A. Wrulich. “MedAustron – Project overview and status”. In: *The European Physical Journal Plus* 126.7 (2011), pp. 1–11. ISSN: 2190-5444. DOI: `10.1140/epjp/i2011-11069-9`.

[7] M. Benedikt and A. Fabich. “MedAustron – Austrian hadron therapy centre”. In: *IEEE Nuclear Science Symposium Conference Record* (2008). DOI: `10.1109/NSSMIC.2008.4774090`.

[8] M. Benedikt et al. *Overview of the MedAustron Design and Technology Choices*. IPAC 2010.

[9] Aafke Christine Kraan. “Range verification methods in particle therapy: underlying physics and Monte Carlo modelling”. In: *Frontiers in Oncology* 5.150 (2015). ISSN: 2234-943X. DOI: `10.3389/fonc.2015.00150`.

[10] L. Sihver et al. “Current status of the “Hybrid Kurotama model” for total reaction cross sections”. In: *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms* 334 (2014), pp. 34–39. ISSN: 0168-583X. DOI: `http://dx.doi.org/10.1016/j.nimb.2014.04.021`.

[11] L. Sihver et al. “Total reaction and partial cross section calculations in proton-nucleus ($Z_t \leq 26$) and nucleus-nucleus reactions (Z_p and $Z_t \leq 26$)”. In: *Phys. Rev. C* 47 (3 Mar. 1993), pp. 1225–1236. DOI: `10.1103/PhysRevC.47.1225`.

[12] Lembit Sihver and Davide Mancusi. "HIBRAC:a 1-D Deterministic Heavy-Ion Transport Code Optimised for Radiotherapy". In: *Radiation Measurements* (44 2009), pp. 38–46. ISSN: 1350-4487. URL: `http://publications.lib.chalmers.se/publication/111287-hibraca-1-d-deterministic-heavy-ion-transport-code-optimised-for-radiotherapy`.

[13] David Sarrut et al. "A review of the use and potential of the GATE Monte Carlo simulation code for radiation therapy and dosimetry applications". In: *Medical Physics* 41.6, 064301 (2014). DOI: `10.1118/1.4871617`.

[14] J. Dudouet et al. "Benchmarking geant4 nuclear models for hadron therapy with 95 MeV/nucleon carbon ions". In: *Phys. Rev. C* 89 (5 May 2014), p. 054616. DOI: `10.1103/PhysRevC.89.054616`.

[15] S. Agostinelli et al. "Geant4—a simulation toolkit". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 506.3 (2003), pp. 250–303. ISSN: 0168-9002. DOI: `10.1016/S0168-9002(03)01368-8`.

[16] T. Goorley et al. "Features of MCNP6". In: *Annals of Nuclear Energy* 87 (Jan. 2016), pp. 772–783. ISSN: 0306-4549. DOI: `10.1016/j.anucene.2015.02.020`.

[17] F Ballarini et al. "The FLUKA code: an overview". In: *Journal of Physics: Conference Series* 41.1 (2006), p. 151. URL: `http://stacks.iop.org/1742-6596/41/i=1/a=014`.

[18] Tatsuhiko Sato et al. "Overview of particle and heavy ion transport code system {PHITS}". In: *Annals of Nuclear Energy* 82 (2015). Joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo 2013, {SNA} + {MC} 2013. Pluri- and Trans-disciplinarity, Towards New Modeling and Numerical Simulation Paradigms, pp. 110–115. ISSN: 0306-4549. DOI: `10.1016/j.anucene.2014.08.023`.

[19] Oksana Geithner et al. "Calculation of stopping power ratios for carbon ion dosimetry". In: *Physics in Medicine and Biology* 51.9 (2006), p. 2279. URL: `http://stacks.iop.org/0031-9155/51/i=9/a=012`.

[20] S Jan et al. "GATE: a simulation toolkit for PET and SPECT". In: *Physics in Medicine and Biology* 49.19 (2004), p. 4543. URL: `http://stacks.iop.org/0031-9155/49/i=19/a=007`.

[21] J. Dudouet et al. "Double-differential fragmentation cross-section measurements of 95 MeV/nucleon ^{12}C beams on thin targets for hadron therapy". In: *Phys. Rev. C* 88 (2 Aug. 2013), p. 024606. DOI: `10.1103/PhysRevC.88.024606`.

[22] W. Demtröder. *Experimentalphysik 4*. Springer Berlin Heidelberg, 2010. DOI: `10.1007/978-3-642-01598-4`.

[23] Dr. William R. Leo. *Techniques for Nuclear and Particle Physics Experiments*. Springer Berlin Heidelberg, 1993. DOI: `10.1007/978-3-642-57920-2`.

[24] W. Demtröder. *Experimentalphysik 3*. Springer Berlin Heidelberg, 2010. DOI: `10.1007/978-3-642-03911-9`.

[25] D.H. Wright. *Physics Reference Manual*. Dec. 2014. URL: `http://geant4.web.cern.ch/geant4/support/userdocuments.shtml`.

[26] Wikipedia. *Dose Depth Curves.svg*. [Online; accessed 25-July 2016]. 2014. URL: `https://commons.wikimedia.org/wiki/File:Dose_Depth_Curves.svg`.

[27] Eric E. Klein. "Electron-Beam Therapy: Dosimetry, Planning and Techniques". In: *Perez and Brady's Principles and Practice of Radiation Oncology*. Ed. by Edward C. Halperin, Carlos A. Perez, and Luther W. Brady. Lippincott Williams & Wilkins, 2008. Chap. 7, pp. 190–217.

[28] W P Levin et al. "Proton beam therapy". In: *Br J Cancer* 93 (2005). DOI: `10.1038/sj.bjc.6602754`.

[29] L.L. Carter and E.D. Cashwell. *Particle-transport simulation with the Monte Carlo method*. Jan. 1975. DOI: `10.2172/4167844`.

[30] John Viega. "Practical Random Number Generation in Software". In: *ACSAC*. 2003.

[31] J. Dudouet et al. "Comparison of two analysis methods for nuclear reaction measurements of 12C +12C interactions at 95 MeV/u for hadron therapy". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 715 (2013), pp. 98–104. ISSN: 0168-9002. DOI: `10.1016/j.nima.2013.03.038`.

[32] Rene Brun and Fons Rademakers. "{ROOT} — An object oriented data analysis framework". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 389.1–2 (1997). New Computing Techniques in Physics Research V, pp. 81–86. ISSN: 0168-9002. DOI: `10.1016/S0168-9002(97)00048-X`.

[33] Douglas Thain, Todd Tannenbaum, and Miron Livny. "Distributed Computing in Practice: The Condor Experience: Research Articles". In: *Concurr. Comput. : Pract. Exper.* 17.2-4 (Feb. 2005), pp. 323–356. ISSN: 1532-0626. DOI: `10.1002/cpe.v17:2/4`.

[34] J. Allison et al. "Geant4 developments and applications". In: *IEEE Transactions on Nuclear Science* 53.1 (Feb. 2006), pp. 270–278. ISSN: 0018-9499. DOI: `10.1109/TNS.2006.869826`.

[35] *Geant4 – Material from previous seminars and training courses.* [Online; accessed 21-August 2016]. URL: `https://geant4.web.cern.ch/geant4/support/training.shtml`.

[36] A. Burker. "Validation of Geant4 physics models for cross section calculation". Mar. 2016.

[37] Davide Mancusi et al. "Extension of the Liège intranuclear-cascade model to reactions induced by light nuclei". In: *Phys. Rev. C* 90 (5 Nov. 2014), p. 054602. DOI: `10.1103/PhysRevC.90.054602`.

[38] D.H. Wright and M.H. Kelsey. "The Geant4 Bertini Cascade". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 804 (2015), pp. 175–188. ISSN: 0168-9002. DOI: `10.1016/j.nima.2015.09.058`.

[39] A. Boudard et al. "Intranuclear cascade model for a comprehensive description of spallation reaction data". In: *Phys. Rev. C* 66 (4 Oct. 2002), p. 044615. DOI: `10.1103/PhysRevC.66.044615`.

[40] *GATE - Users Guide V7.1.* June 2014. Chap. How to use Gate on a Cluster. URL: `http://wiki.opengatecollaboration.org/index.php/Users_Guide_V7.1`.

[41] *Examples for Submit Description Files.* [Online; accessed 10-August 2016]. URL: `https://research.cs.wisc.edu/htcondor/quick-start.html`.

[42] *ROOT - Class List.* Mar. 2016. URL: `https://root.cern.ch/doc/master/annotated.html`.

[43] *pugixml.* [Online; accessed 25-July 2016]. URL: `http://pugixml.org/`.

[44] *Courtesy of T. Schreiner, MedAustron.*

Appendices

A. MedAustron Images

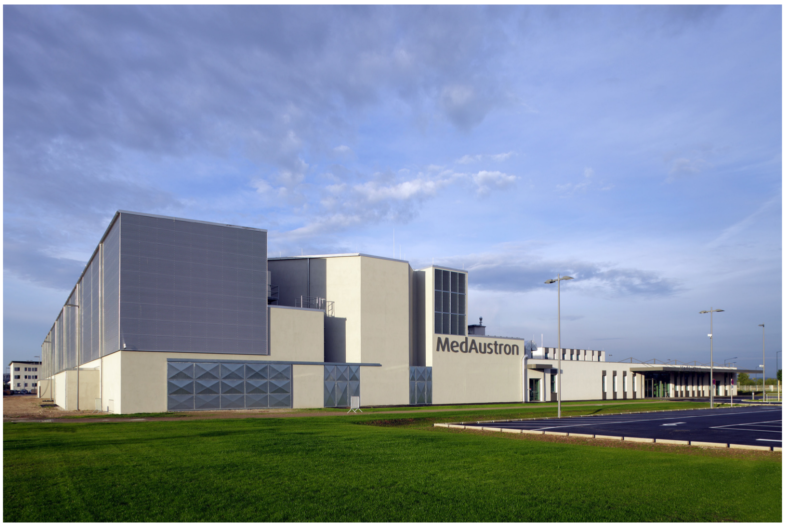


Figure A.1.: Southern wall of MedAustron's building complex. The patient entrance can be found on the far right of the image, under the projecting roof [44].

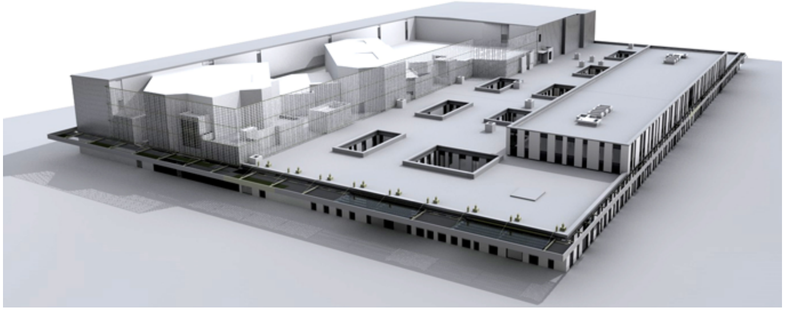


Figure A.2.: Artists impression of the building, from [6].

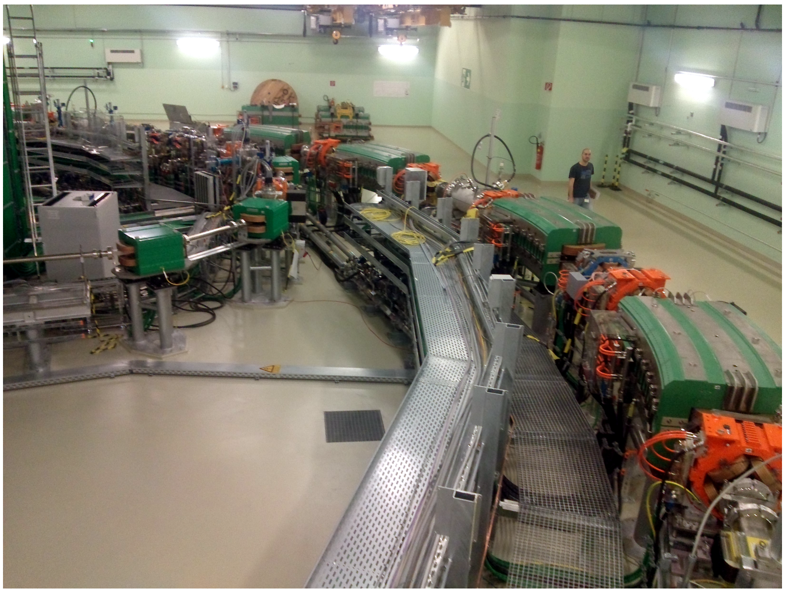


Figure A.3.: Photograph of parts of the accelerator at MedAustron, including the injection line at the left and a segment of the synchrotron ring at the right.

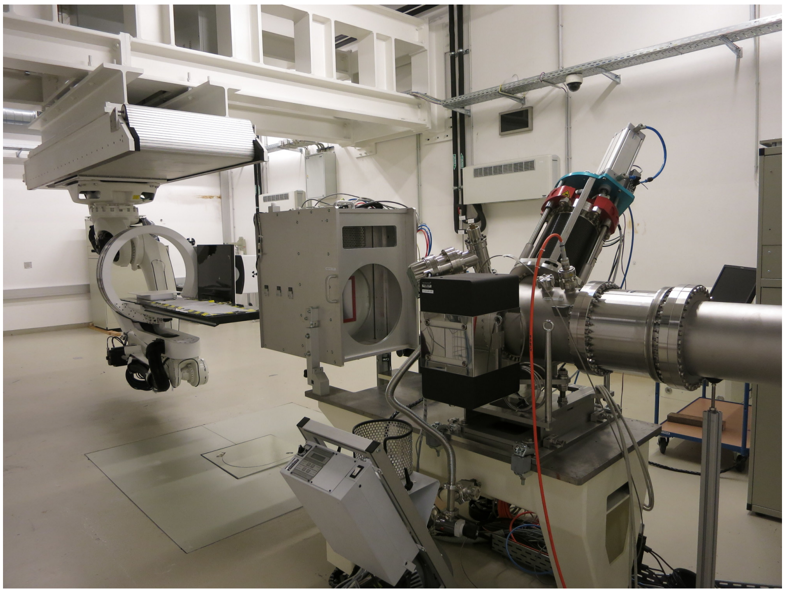


Figure A.4.: View of the robotic table in Irradiation Room 1 of MedAustron, for non-clinical research. The beam line on the right connects to the extraction line in the accelerator complex [44].

B. GATE Macro Files

In order to describe the simulations that were run during the production of this work, several macro files were produced. They contain parameters that characterize geometry, beam source, target and detector. The contents of these macros are briefly explained, the files themselves can be found on the CD attached to this work:

data/

materials.db - Although not a macro, this item is still important in order to parameterize the simulations. It contains data such as charge and mass of elements, as well as densities and composition of materials in use.

mac/

main.mac - This macro file connects the other files in the correct order. To start the simulation it needs to be passed to `GATE`.

alias.mac - Defines *alias names* for all parameters used and initializes their values. These aliases are used by all the other macros. This file is the only one that needs to be updated between different iterations of the simulations.

source.mac - Defines a region as the source to the primary beam, from which ions emerge in direction of a target.

geometry.mac - defines the target as a rectangular cuboid and shifts it into the beam path. It also defines a large sphere around both target and source, that will act as a detector.

actors_general.mac - This file implements an actor for general statistics during simulation. The actor records the number of steps, tracks, events and runs in the simulation. Its output was used for finding mistakes.

actors_spherical_detector.mac - Assigns the large sphere, defined in *geometry.mac* as a *PhaseSpaceActor* that records particles going through it, like a detector. The actor was configured to save particle names, energies and locations into an output `ROOT` file.

parameters.mac - Used to control additional `GATE` parameters, like the ionization potential of water, secondary particle production thresholds which provide a low energy cut-off and maximum step sizes.

output/ - Although this directory is initially empty, it must exist on the hard drive or the simulation fails to produce output `ROOT` files.

C. Analysis Program Source Files

The program code is distributed over fifteen files provided below. Eight of them are headers and seven are source files. Two of the headers and one of the source files belong to the `pugixml` project [43], while the others were produced for this work. Most of the headers only contain definitions for the functions with the same name.

headers/

calculateCrossSectionRing.h - Header file for the *calculateCrossSectionRing* function, which contains its interface and necessary include-terms.

checkConfiguration.h - Header file for the *checkConfiguration* function, which contains its interface and necessary include-terms.

ConfigSet.h - Header file for the *ConfigSet* class. This file contains the interface for the class and its members, as well as the Get-methods responsible to obtain the contents of the class objects. The class itself is used to represent all the parameters of a single analysis process. Necessary include-terms are also part of this file.

main.h - Main function header file, which contains necessary include-terms.

openROOTFile.h - Header file for the *openROOTFile* function, which contains its interface and necessary include-terms.

plotting.h - Header file for the *plotting* function, which contains its interface and necessary include-terms.

pugiconfig.hpp - Configuration file for `pugixml`.

pugixml.hpp - Main header for `pugixml`.

source/

calculateCrossSectionRing.cpp - Source code for the *calculateCrossSectionRing* function, which contains most of the functionality of the program. It takes a *TChain* and a *ConfigSet* object as parameters and runs through the *TChain* in order to calculate cross sections according to the parameters from the *ConfigSet*. It then writes the cross sections to hard disk files according to the configuration.

checkConfiguration.cpp - Source code for the *checkConfiguration* function, which takes a *xml-node* and a *ConfigSet* object and examines whether the nodes contained by the given xml-node provide the necessary parameters for an analysis. If so, the function fills in the parameters into the *ConfigSet* and returns true, otherwise false.

ConfigSet.cpp - Source code for the *ConfigSet* class, which contains Set-methods in order to insert content into class objects.

main.cpp - Main program source file, which links the other functions together. It verifies whether the user provides a valid *configuration.xml* file, checks whether the file parses correctly and if so, loops over the xml root-nodes. These nodes are passed to the function *checkConfiguration* and if successful, a TChain is produced with *openROOTFile* and given to *calculateCrossSectionRing* in order to calculate the cross sections.

openROOTFile.cpp - Source file for the *openROOTFile* function, which produces a *TChain* from a *ConfigSet* object and simulation output located on the hard disk.

plotting.cpp - Source file for the *plotting* function. The program will produce graphs of the cross sections and write them into `ROOT` output files, but only when specified in the *configuration.xml* file. This is optional and done in addition to the regular text based output.

pugixml.cpp - Source file for the *pugixml* functions.

D. Configuration.xml

The analysis program is executed with a parameter that points to a *"configuration XML"* file, which is responsible to parameterize the analyses that are going to be executed. Its file name does not matter to the program, however, it must be an `XML` file that can be parsed by *pugixml* [43]. In order to be able to run an analysis, the program demands the following structure for the configuration file:

the root node - The first node contains all the parameters of one analysis in its child nodes. The name of the *root* node does not matter to the analysis, but setting a useful name can be helpful when looking for mistakes. This name is not part of the output data files, but can be found in log-files, which are produced for debugging. If several analyses are going to be part of one `XML` file, the *root* nodes should have different names in order to find them in the logs. The analysis parameters are described in the following nodes, which have to be children of the *root* node. Allowed child nodes are: *pathtoinput*, *nrinputfiles*, *pathtooutput*, *targetmolarmass*, *targetdensity*, *targetthickness*, *pnum*, *DDX* and *DX*. The values for these nodes should be identical to those from the respective simulations or the cross section calculation will be scaled incorrectly.

pathtoinput - The content of this node has to point to the first `ROOT` file that is going to be analyzed, leaving out the *".root"* file extension and the numeration for a chain of files. For a file called *"input.root"* in a directory *"path/to/"* this node would read *"path/to/input"*. For numerated files – like *"input1.root"*, ..., *"input5.root"* located in the same directory – the node would still read *"path/to/input"*.

nrinputfiles - This node specifies how many `ROOT` files – for example *"in1.root", "in2.root", ..., in100.root* – are going to be used in the analysis. In the example there are 100 files, so this value would be set to 100.

pathtooutput - Optional parameter, which specifies where the analysis output is going to be saved. Previous analysis-output with the same parameter would be overwritten. *pathtoinput* is used as *pathtooutput* if no value is provided. The directory specified must exist on the hard disk and needs to be accessible or the program fails. *pathtooutput* can also be used to add a name to the data. A parameter like *"path/to/out"* would prompt the program to produce data files in the directory *"path/to/"* with file names like *"out_C_C_1140MeV_DC.dat"*

targetmolarmass - The molar mass of the target, which was used to parameterize the simulation. For carbon it would be 12.

targetdensity - Contains the density of the simulated target, in g/cm^3. The used densities for this work can be found in table 3.1.

targetthickness - The thickness of the target during simulation, in cm. For this work it was set to 0.001 cm.

pnum - The number of incident particles in the simulation. In the scope of this work it was 10^9.

DDX - This node must be included when the program has to calculate double differential cross sections. The parameters inside the *DDX* node have to specify an energy range and bin size. Also, at least one combination of a fragment particle and at least one angle theta have to be included. The analysis is skipped if no *DDX* node and no *DX* node exist as children of the *root* node, because in this case no cross sections would be calculated. Nodes that are allowed in the *DDX* node are *energystart*, *energydelta*, *energyend*, *particles*, *angles*, *thetastart*, *thetaelta* and *thetaend*.

DX - This node has to be set when the program should calculate differential cross sections. The parameters inside the *DX* node have to specify at least one combination of a fragment particle and at least one angle theta. The analysis is skipped if no *DDX* node and no *DX* node exists. The *DX* node can contain the same nodes *particles*, *angles*, *thetastart*, *thetadelta* and *thetaend*, that are available to the *DDX* node, as well as the *UseDDXValues* node.

UseDDXValues - This node can be used for convenience if the *DX*-calculations are based on the same particles and angles that were specified in the *DDX*-node. The analysis program immediately uses the same configuration for both differential and double differential cross sections, if the *UseDDXValues* node is found inside the *DX* node and the parameters inside the *DDX* node are valid.

energystart - Specifies the mean energy of the first energy bin, in MeV/u.

energydelta - Specifies the ± spread of the energy bins, in MeV/u. This value must be less than or equal to *energystart*, in order to avoid negative energies.

energyend - Specifies the mean energy of the last bin. This value has to be at least as large as $energyend \geq energystart + 2 \times energydelta$ (An analysis that counts fragments between 0 and 200 MeV, with a bin size of 10 MeV, would have the following parameters: $energystart = 5$, $energydelta = 5$ and $energyend = 200$).

particles - This node must exist. It contains *particle* nodes that specify the fragments for which cross sections are going to be calculated.

particle - Every node of this type describes a fragment for the analysis and has to specify two attributes:

1. *name* - The name of the particle for which cross sections are going to be calculated. These correspond to the names from the output of `Geant4`, like "proton" or "Be7"
2. *nucleons* - How many nucleons are part of the detected particle in question. In the case of a proton this would be 1, for carbon-12 it would be 12.

angles - This node is semi-optional. At least one angle has to be specified, but they can either be entered in form of *angle* nodes – as children of this node – or as a range of angles similar to the energy range. A combination of both methods is valid. In this case a range of angle bins can be extended by single angles with different bin sizes.

angle - Every node of this type represents an angle element for the analysis and has to contain two attributes:

1. *theta* - The mean value for the polar angle during angle selection.
2. *d_theta* - The ± spread for the polar angle selection, which must be less than or equal to *theta*.

thetastart - Similar to the energy-range, angles can be specified in a range for the analysis. This node describes the mean value of the first angle of the range. The nodes *thetadelta* and *thetaend* must contain valid entries or the range specification is ignored by the program.

thetadelta - The ± spread around the mean values of angles between *thetastart* and *thetaend*. The value of *thetadelta* must be less than or equal to *thetastart*.

thetaend - The largest angle that is part of the angle range. This value has to be at least as large as $thetadelta \geq thetastart + 2 \times thetadelta$.

E. Cross Section Estimates

Additional figures are provided here, as a representation of the vast number of produced cross section estimates. The data can be found on the attached CD, along with some prepared scripts to be able to quickly produce further plots. Figures E.5 and E.6 highlight how production cross sections decline for larger angles and incidence energies. Figures E.7 and E.8 show double differential and differential cross sections for protons, as do figures E.9 and E.10 for lithium-6, in order to show differences based on the fragments being considered. Lastly, figures E.11 and E.12 show double differential and differential cross sections for collisions with a hydrogen target, as do figures E.13 and E.14 for a sodium target.

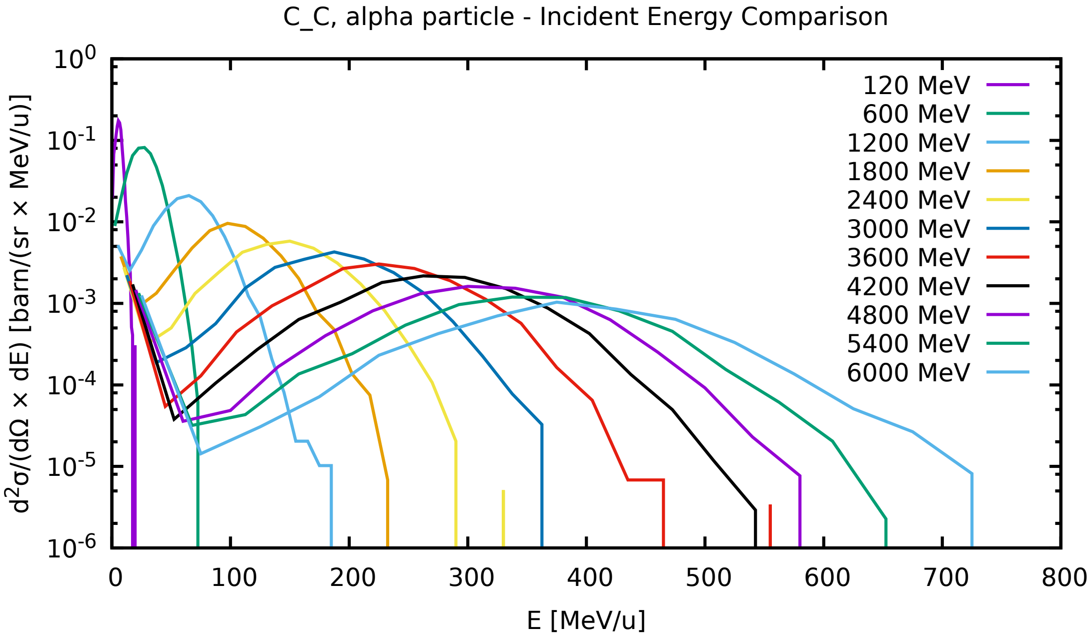


Figure E.5.: Double differential cross sections for alpha particles produced at 12.5 ± 2.5 degrees from carbon-carbon collisions.

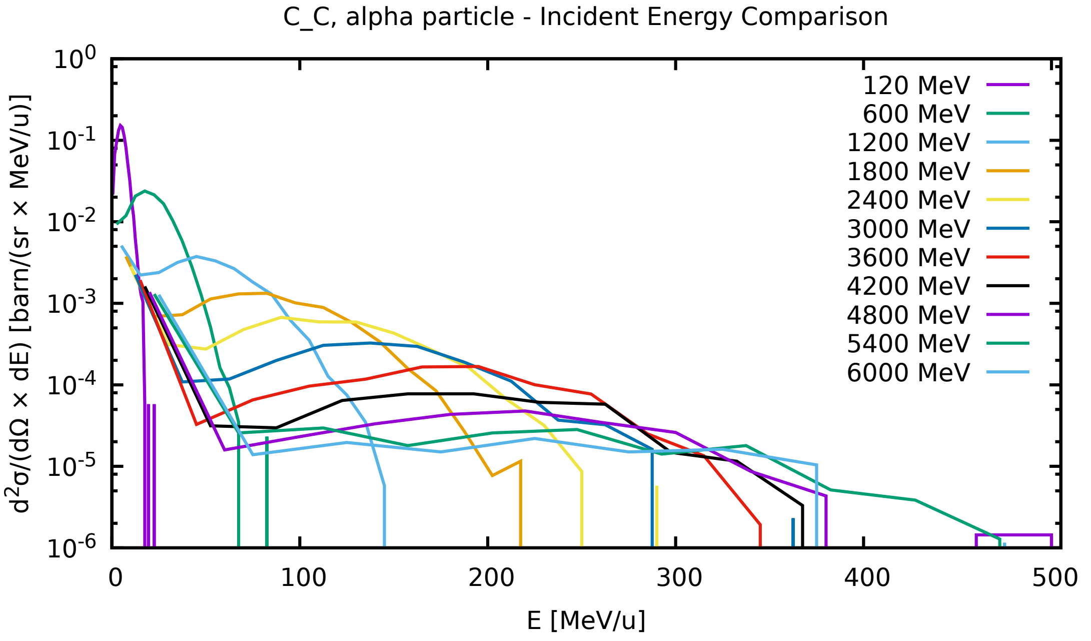


Figure E.6.: Double differential cross sections for alpha particles produced at 22.5 ± 2.5 degrees from carbon-carbon collisions.

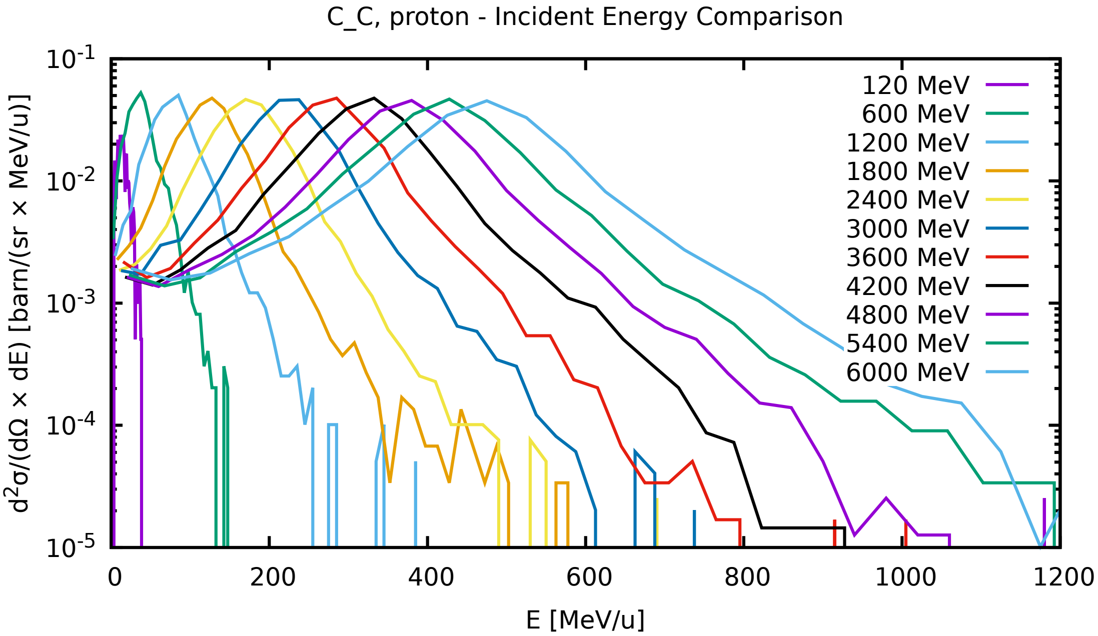


Figure E.7.: Double differential cross sections for protons produced at 2.5 ± 2.5 degrees from carbon-carbon collisions.

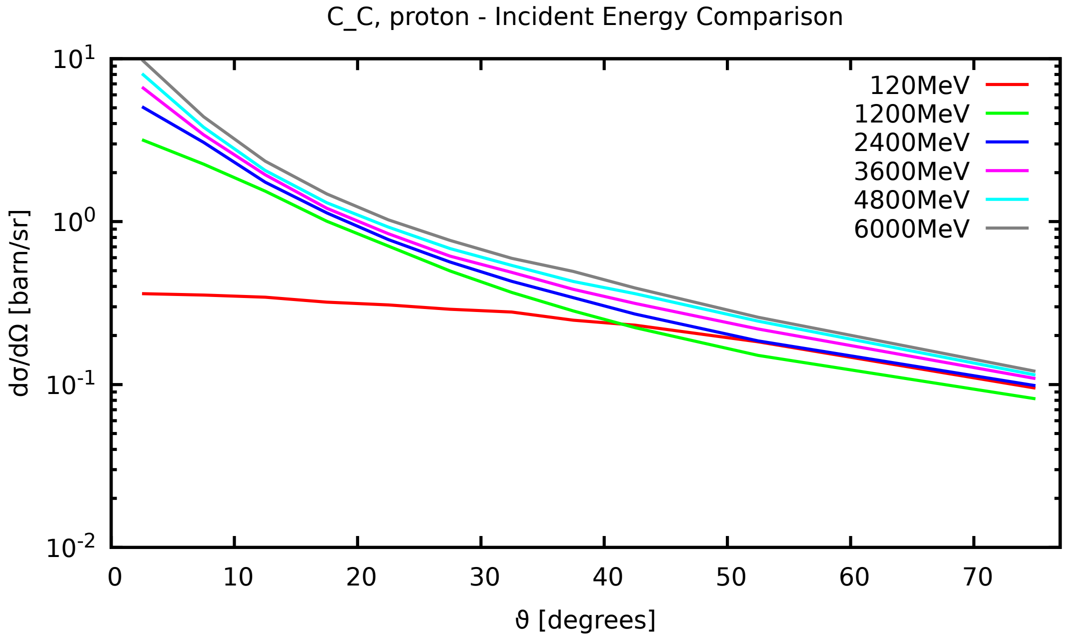


Figure E.8.: Differential cross sections for protons produced from carbon-carbon collisions.

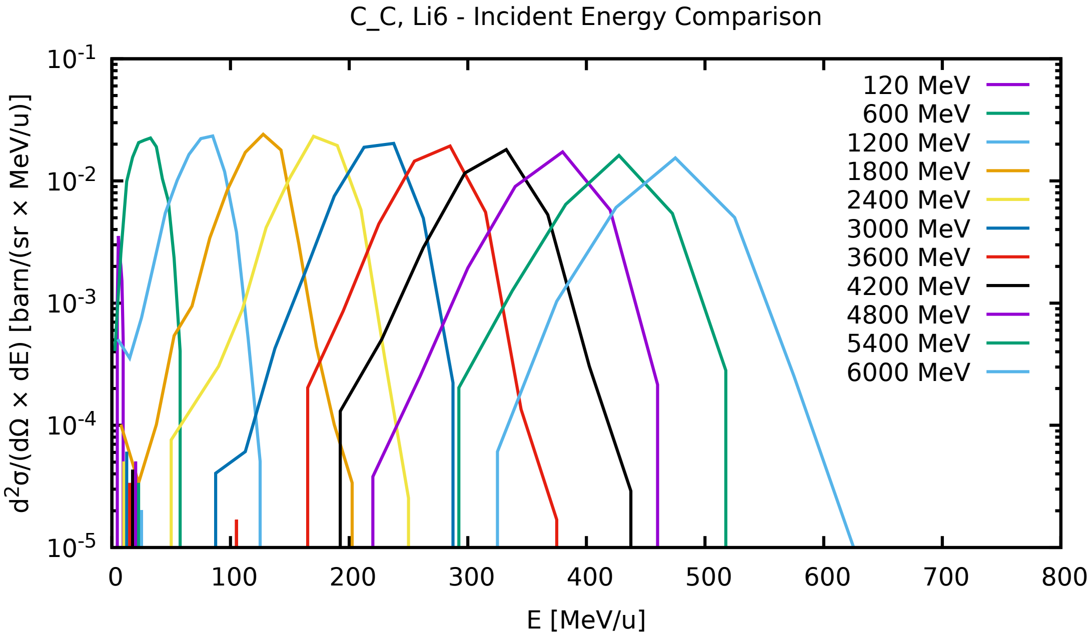


Figure E.9.: Double differential cross sections for Li-6 produced at 2.5 ± 2.5 degrees from carbon-carbon collisions.

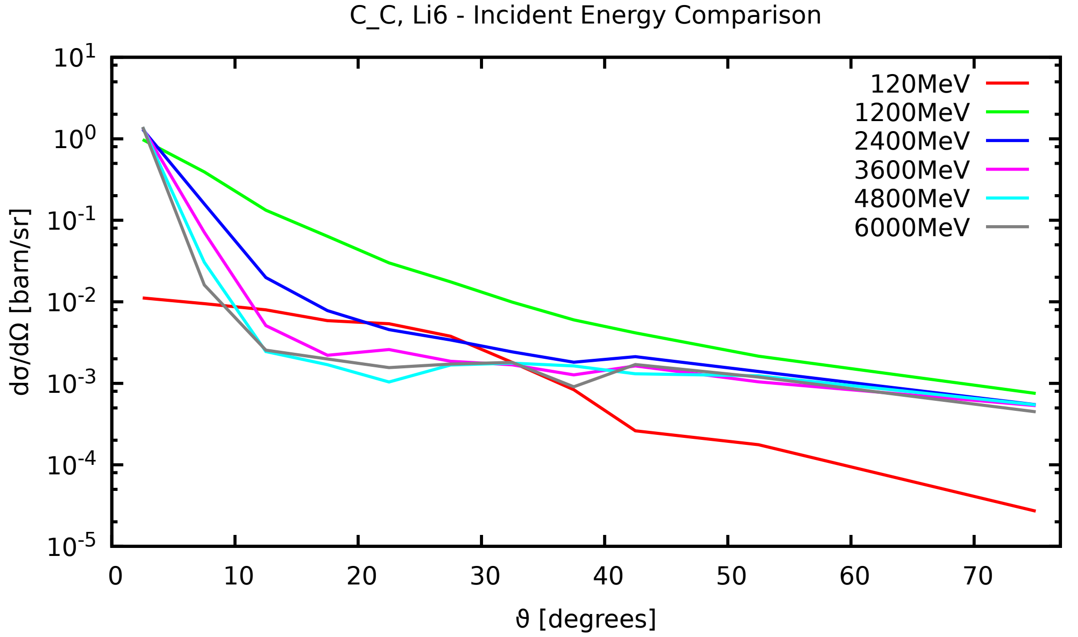


Figure E.10.: Differential cross sections for Li-6 produced from carbon-carbon collisions.

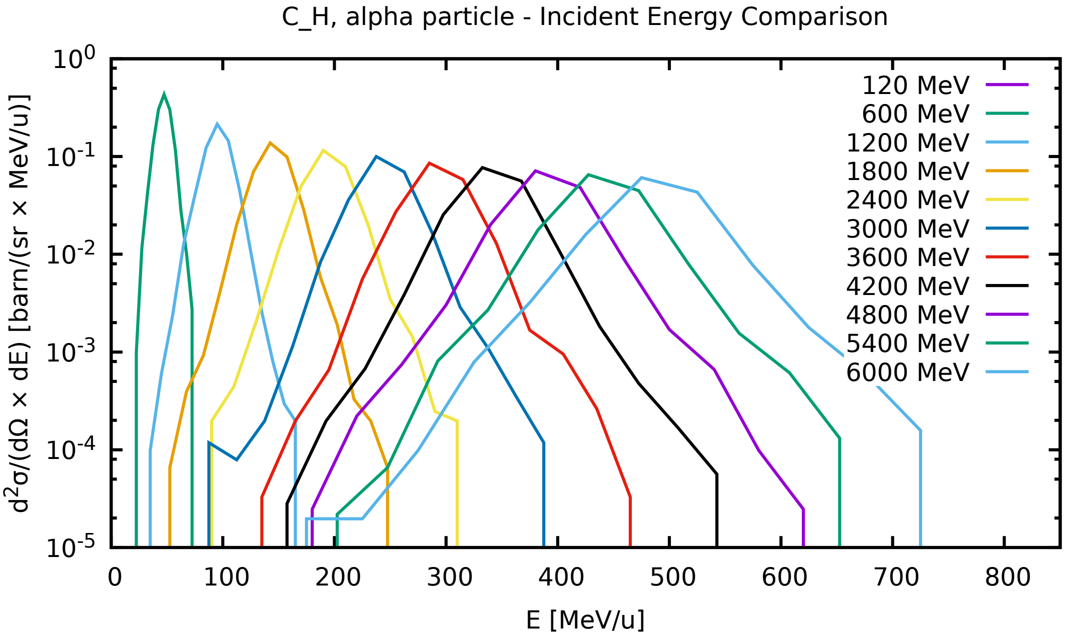


Figure E.11.: Double differential cross sections for alpha particles produced at 2.5 ± 2.5 degrees from carbon-hydrogen collisions.

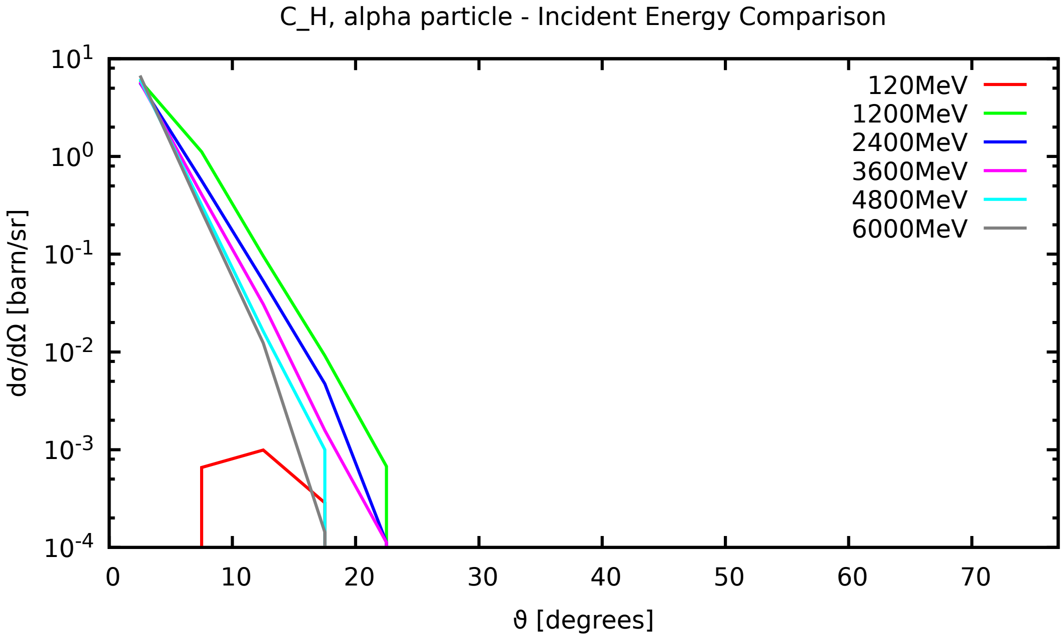


Figure E.12.: Differential cross sections for alpha particles produced from carbon-hydrogen collisions.

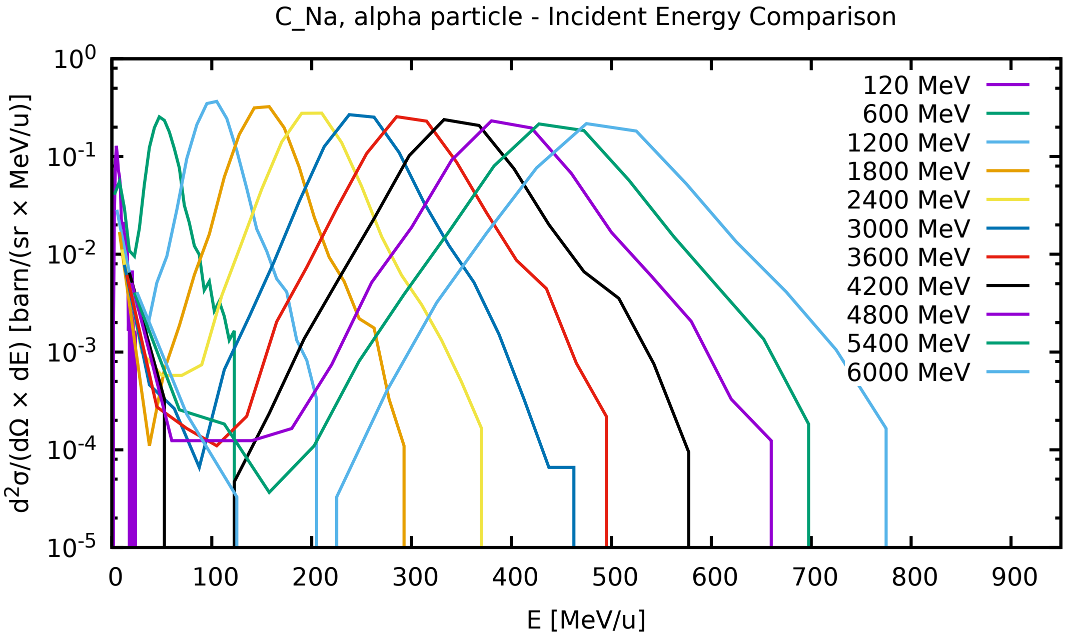


Figure E.13.: Double differential cross sections for alpha particles produced at 2.5 ± 2.5 degrees from carbon-sodium collisions.

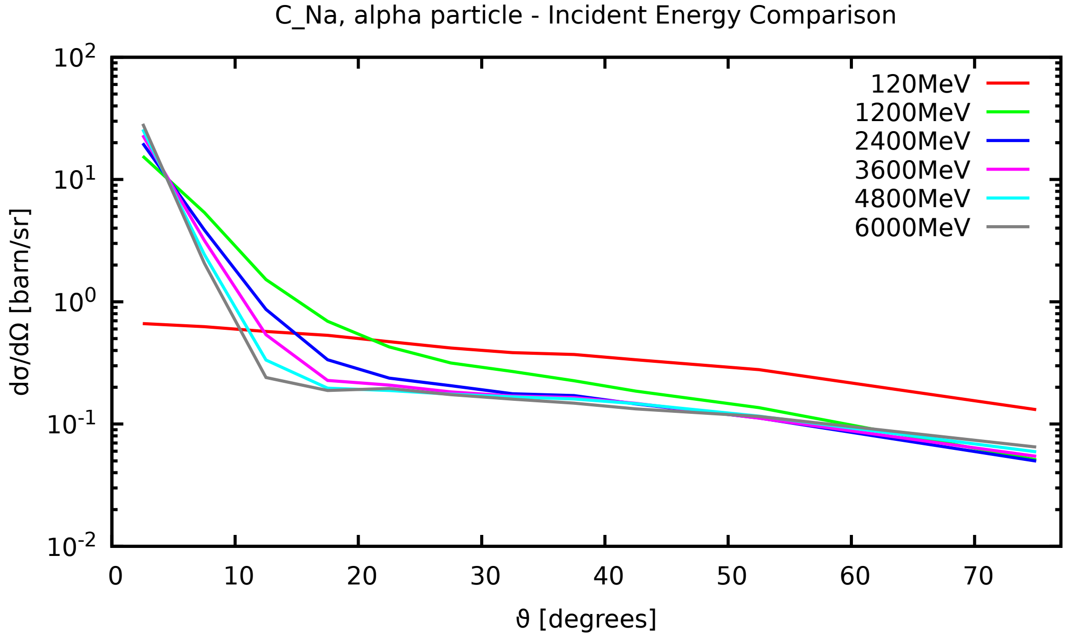


Figure E.14.: Differential cross sections for alpha particles produced from carbon-sodium collisions.