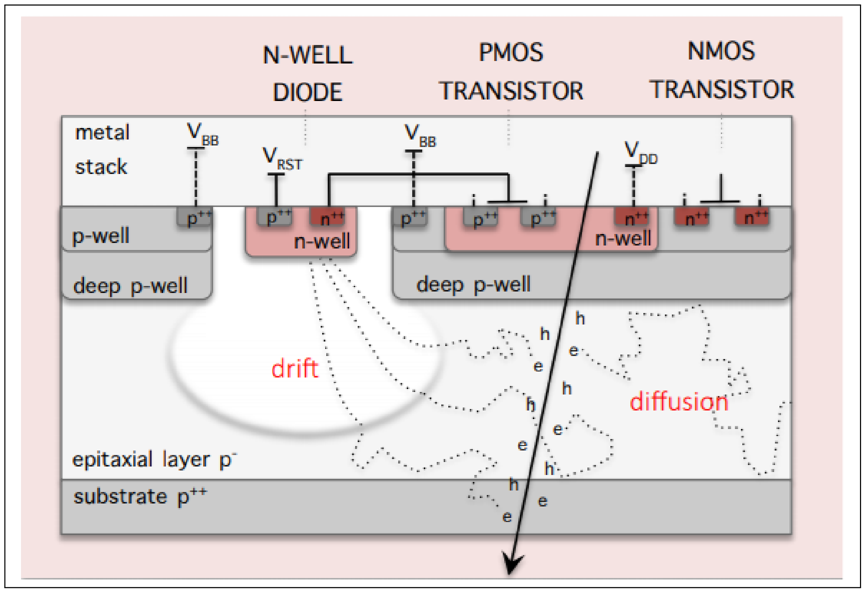
Utrecht University

Department of Physics

Simulations of the ALPIDE CMOS Sensors in a Forward Calorimeter

Bachelor Thesis

Nikita Mischenko



Supervisors:

Prof. Dr. T. Peitzmann
Institute for Subatomic Physics (SAP)

Dr. ing. N. van der Kolk
Institute for Subatomic Physics (SAP)

June 11, 2019

Abstract

The newly proposed FoCal detector at the ALICE experiment is meant to enhance the understanding of the low momentum fraction x regions ($\mathcal{O}(10^{-5})$) of the Parton Distribution Functions. The FoCal prototype, the mTower, is being developed and tested. In this thesis, virtual models of the prototype mTower geometry and the ALPIDE CMOS chips were recreated. Two separate methods, Allpix^2 and SimpDiff, were used to simulate electrons of 50, 100, and 150 GeV passing through the detector geometry. These simulations included effects of charge sharing, clustering, and digitisation. Results were obtained in the form of hit maps, event sizes, and cluster sizes, to be compared to test beam data. Results showed that the SimpDiff method generated more hits per event ($\sim 50\%$) and slightly bigger cluster sizes ($\sim 5\%$); hit maps were comparable with Allpix^2. The relative RMS (mean/RMS) of event sizes for both methods differs at most by 0.04, implying that the shapes of the event sizes implementations were very similar. It was found that SimpDiff had 26.4% less run time than Allpix^2, signalling that SimpDiff is a more practical method for simple physics simulations, whereas Allpix^2 has the capability of performing simulations with more advanced physics phenomena such as capacitive charge transfer and TCAD mesh implementation. While the method of charge sharing in the ALPIDE chips is known, the parameters of the distributions are not; as such, both methods should be compared to test beam data in order to decide which is better suited for any given situation.

Contents

1 Introduction

Throughout history, physicists have tried to describe the regular and peculiar phenomena that occur in daily life. While it started out with describing phenomena that can be seen with the naked eye, touched with the tips of the fingers, or observed with other bare human senses, physicists delved deeper into the unknown with the advancement of technology. With time came telescopes, electricity, microscopes, computers; an unbelievable collection of machinery that made it possible to explore regions far beyond the mere human experiences. From the grand stars in the sky down to the smallest constituents of our world, there was a lot to be explored to give a better understanding of Nature.

As our machinery became more refined, it became possible to probe the smaller regimes of the universe. CERN, the European Organisation for Nuclear Research, was founded in 1954. Their main function was to provide the scientific community with particle accelerators necessary to explore the confines of these tiny regimes of our world. As time went on, these accelerators have contributed immensely to our understanding of how our universe was formed, and what the most fundamental ingredients are for its existence; most notably, the discovery of the W, Z, and H bosons, which filled the blanks in the Standard Model of particle physics.

The particle accelerators house different detectors which conduct experiments aimed at improving the understanding of a specific branch in particle physics, such as heavy-ion collisions and dark matter. Particle physicists create simulations of these detectors in order to compare real data to theoretical models. The physics in the simulations are implemented in such a way that they mirror the real world; as such, the simulation is a mere reflection of the actual experiments. What makes simulations so useful and necessary in particle physics is that simulations can be readily reconfigured in terms of geometry, material composition, incident particles and much more; this allows experimentalists to learn how certain changes in parameters will affect the outcome of an experiment, without wasting the time, money, and materials that would be necessary to build countless test setups.

This thesis provides a description and results of simulations performed for the mTower, a prototype for the newly proposed FoCal detector; a digital sampling calorimeter that is meant to be installed as a component in the forward region of CERN's ALICE detector. The FoCal is supposed to give physicists new insights into the behaviours of quarks and gluons at certain physical boundaries; for example, what the universe looked like mere moments after its creation. The detector aims to improve the understanding on certain scales to get a better look at these fundamental building blocks of the universe.

The new FoCal design incorporates the usage of ALPIDE CMOS pixel sensors, which provide digital data readout for any electromagnetic particle interactions. These new sensors are meant to improve data accuracy and readout speed over the old MIMOSA CMOS pixel sensors.

For this thesis, Monte-Carlo models of the mTower prototype have been constructed in the Allpix^2 framework, which have been designed by CERN in order to perform full physics simulations of silicon sensors, and assess expected performance for new detector proposals; the other model was built by hand using GEANT4 output data, in a method dubbed SimpDiff. The goal of this thesis is to use both aforementioned methods to generate simulation data for the mTower prototype, which can later be compared to actual test data after the detector construction is finished. More specifically, this thesis focuses on the difference between the two methods, and compares the efficiency, quality of results, and user-friendliness of using one over the other. As such, the simulations have been created in such a way that the results of both methods are comparable.

1.1 ALICE

1.1.1 Detector

ALICE [1] is one of the four large detectors at the Large Hadron Collider (LHC) ring near Geneva, Switzerland. The ALICE experiment is situated under St. Genis-Pouilly in France. ALICE's main function is the study of particle behaviour at extreme energy densities and temperatures, which are mainly achieved by colliding Pb-ions (lead); because of the large amount of nucleons in Pb-ions, these collisions lead to a high multiplicity of product particles, which the detector is specifically designed to handle. ALICE also deals with dedicated pp (proton-proton) and pA (proton-nucleus) collisions, in order to vary energy density as a reference [2]. Refer to figure 1.1 below for an idea of ALICE's components.

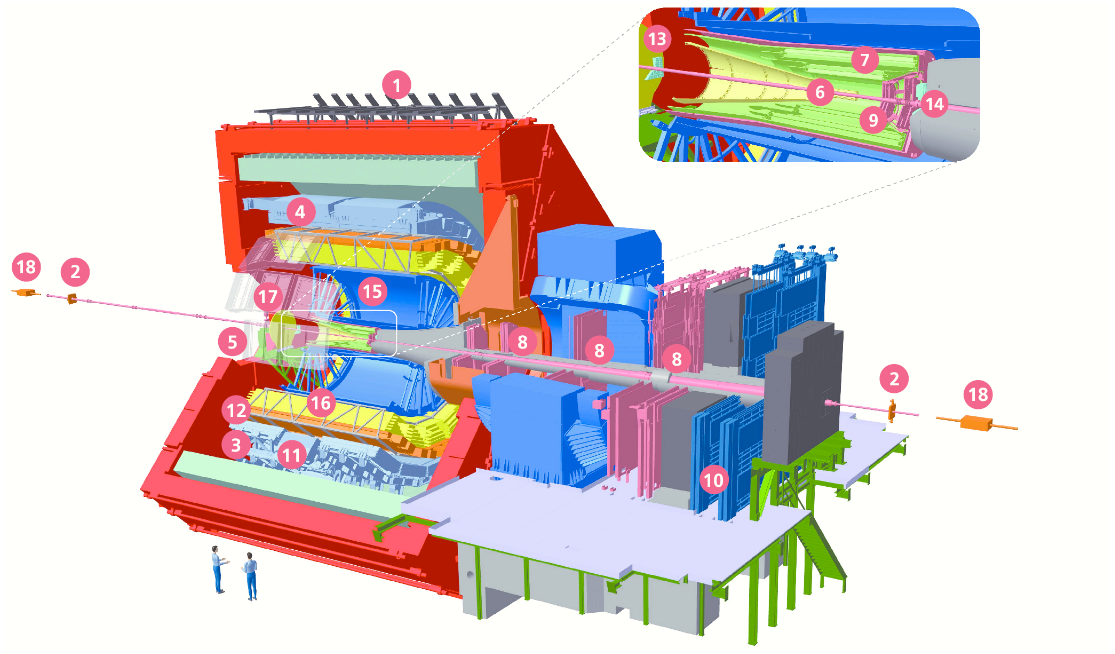


Figure 1.1: A schematic overview of all the components present in ALICE. This thesis mainly concerns the front (left) part of the detector, where the FoCal is proposed to be situated [3].

1.1.2 Quark-Gluon Plasma

Colliding the ions at such high energies leads to temperatures so extreme ($\sim 10^{12}$ K) that quarks and gluons (the constituents of hadrons) are no longer confined by the strong force, and can move freely; this new state of matter has been dubbed the *quark-gluon plasma* (QGP). The QGP can be described by quantum chromodynamics (QCD) [4]; the theory of the strong force in the Standard Model of Particle Physics. The QGP is believed to have existed (naturally) between 10^{-12} and 10^{-6} seconds after the Big Bang, in a period that was called the *Quark Epoch*; a time where the four fundamental forces (strong, weak, electromagnetic and gravitational) assumed their present forms, after which the universe cooled down enough to enable baryogenesis to occur.[5] The QGP is of significant interest to particle physicists, since it may provide a better understanding of features of the strong interaction studied by QCD, such as the equations of state or its evolution. The ALICE experiment employs a number of methods to study the produced plasma in more detail.

As mentioned above, quarks and gluons are confined inside hadrons; this is due to the nature of the strong force, which increases in strength as the distance between quarks increases. Quarks and gluons carry a property called colour charge. There are three different colours and corresponding anti-colours. In order to be physically viable states, hadrons need to be colour neutral: either through a combination of a quark and anti-quark pair (leptons), or a combination of 3 quarks which combine all three colours (baryons). In high-energy collisions, a large amount of particles are produced that are can be unstable in terms of colour charge. Most of these particles are contained within the QGP; however, some high-energy partons may escape the confines of the QGP. These particles start to create objects around them in order to form colourless objects once again. The collection of these high-energy particles is called a *jet*.

Jet quenching is a phenomenon where jets emerging from the collision interact with the QGP, which leads to a marked reduction of their energy. The ALICE experiment enables the study of the decay particles of hadrons formed in these jets; namely measurements of these decay particles at high transverse momentum p_T[1].

The detector also allows for the study of the decay electrons of hadrons that form in the jets, which show a reduction in yield of energy deposition in Pb-Pb collisions compared to pp collisions, which suggests that the quarks (heavy flavours in particular) interact with the QGP.

The larger samples and frequencies of Pb-Pb collisions in Run 3[2] will improve measurements and will allow to study more properties of the quark-gluon plasma, such as its hydrodynamics and production of strange-antistrange quark pairs.

1.1.3 Colour-Glass Condensate

Quarks and gluons (partons) contribute their individual share of momenta to the hadron; these relative contributions are described by the Parton Distribution Function (PDF). This distribution is a function of the *momentum fraction* of the partons, x, as well as the *momentum transfer* involved in the interaction with the hadron, Q^2. The momentum fraction x is given by:

$$x = \frac{p_{\text{constituent}}}{p_{\text{hadron}}} \tag{1}$$

These distributions evolve according to the Dokshitzer–Gribov–Lipatov–Altarelli–Parisi (DGLAP) equations, which predict that at high Q^2 and low x, the gluon contribution increases continually. Since this is not a physically feasible state, the likelihood of gluons merging increases, which will in turn stabilise the amount of gluons in the hadron, leading to *gluon saturation*.

The Colour-Glass Condensate (CGC) is a model for describing this saturated gluon state; it provides a depiction of the initial state conditions of heavy-ion collisions. Low-x parton states can be described by a classical colour field, which are generated by the random static sources of the high-x partons. High-x partons appear to be static due to extreme time dilation; these phenomena, along with high gluon density, give rise to the name of the Colour-Glass Condensate.

As mentioned before, the CGC becomes interesting in the low-x regime; the effects of gluon saturation start to occur in processes that reside below a certain momentum threshold, which is called the momentum *saturation scale*, Q_S, and can be approximated by:

$$Q_S^2 \propto x^{-\lambda} \cdot A^{\frac{1}{3}}, \tag{2}$$

[1]A quantity that characterises momenta in collider experiments. Measured perpendicular to the beam axis, which is usually chosen as the z-axis. $p_T = \sqrt{p_x^2 + p_y^2}$[6]. This quantity is related to the rapidity, which is defined in equation 4.

[2]An upgrade scheduled for ALICE that will improve data readout and processing capabilities. Set to take place in 2020.

where λ is a dimensionless parameter, and A is the mass of the nucleus. Saturation effects are stronger for large nuclei and for small x.

The gluon density is expected to be smaller in the saturation regions than originally predicted by the linear QCD evolution. Processes at small x are the most sensitive to the saturation scale; x can be obtained from a process' final-state kinematics in agreement with:

$$x \approx \frac{2p_T}{\sqrt{s}} e^{-y}, \tag{3}$$

where y is the *rapidity* of the produced particle. Accurate examinations of small x regimes will have to be done at high beam energy to satisfy the high centre-of-mass energy, and high rapidity to satisfy the exponential relation.

1.1.4 (Pseudo)rapidity

In the relativistic regime, it is necessary to define rapidity in order to determine in which direction the particle of the high-energy collision is moving. The rapidity allows for the comparison of relativistic velocities in different reference frames. The rapidity is given by:

$$y = \frac{1}{2}\ln\left(\frac{E + p_z c}{E - p_z c}\right) \quad \Rightarrow \quad y = \tanh\left(\frac{p_z c}{E}\right) \tag{4}$$

This equations shows that when a particle's track is close to the beam axis, the particle's energy is approximately equal to the momentum; as a result, the rapidity goes to infinity, whereas if the particle moves transverse to the beam axis, the rapidity is zero (as the momentum in z-direction goes to 0, y approaches $\ln(1) = 0$).

It is, however, possible and even convenient to define a *pseudorapidity* η; for all intents and purposes of heavy ion physics, the pseudorapidity is a very reasonable approximation of regular rapidity. The pseudorapidity is also useful when some information about a particle is unknown, as the rapidity requires the full 4-momentum of a particle. The pseudorapidity is given by (derivation given in A.1):

$$\eta = -\ln\left(\tan\left(\frac{\theta}{2}\right)\right), \tag{5}$$

which leads to the conclusion that large rapidities imply small θ, which is the angle relative to the beam direction (the z-axis in most definitions; parallel to the stacking direction of the detector). Measurements at large rapidities, and thus small angles, are usually called *forward measurements*.

1.2 Thesis content

This thesis starts off with a chapter on calorimetry, which explains how electromagnetic cascades form and propagate, and which physical processes need to be taken into account when designing an electromagnetic calorimeter like FoCal. This chapter is followed by a chapter on the hardware and software, which outlines the components of the mTower prototype, its geometry, and readout capabilities, as well as explaining the inner workings of the SimpDiff and Allpix^2 methods. The successive chapter describes the methodology for generation of results of both methods in detail. After the methodology, results are presented for both simulation proposals; this chapter describes the obtained results and denotes the elapsed time and efficiency of both methods. A discussion of results is presented directly afterwards. The thesis ends with a conclusion on which method is better to use for the simulations based on what the user is looking for; an appendix is provided where derivations and extra simulation results are presented.

2 Calorimetry

This section mainly follows R. Wigmans [7], and a presentation by B. Surrow [8]; other sources are mentioned appropriately.

Calorimeters are devices that detect particles by capturing their final-state energy through total absorption, and transform said energy into a measurable quantity, e.g. light or electric charge. As such, a calorimeter has two tasks: it needs to make the particle lose energy through interaction with the calorimeter, and it needs to have a way to measure the energy. These two components of a calorimeter are called the *absorber* and *sensor*, respectively.

While there are more than one type of calorimeter, the type studied in this thesis is an electromagnetic one; this is to reflect one of the two calorimeters that are proposed for the FoCal: FoCal-E (electromagnetic) and FoCal-H (hadronic). Electromagnetic calorimeters are designed to detect and measure energies of electrons, positrons, and photons. The next sections describe how the calorimeter and the physical processes behind the interactions of these particles work in more detail.

2.1 Electromagnetic Showers

When an electron, positron, or photon enter a dense material, they start to lose energy through interactions with that material. In this process, lower energy electrons, positrons, and photons are produced in a so-called electromagnetic *shower* or *cascade*.

The charged particles (electrons and positrons) interact in four main ways:

1. *Bremsstrahlung* for high energy ($\geq$ 10 MeV) particles, in which the particles emit energy in the form of EM radiation at the expense of its kinetic energy,
2. *ionisation*, which produces energetic electrons after an incoming electron collides with a nucleus, releasing an electron from an electron shell,
3. *Cherenkov radiation*, which occurs when charged particles travel faster than the speed of light in a medium,
4. *Delta electrons*, which are produced when a charged particle transfers enough energy to an electron of an atom to free it from its shell; if this electron has enough energy to ionise other atoms, it is called a *delta ray*.

A representation of the energy loss of the electrons and positrons can be seen in Figure 2.1.

The photons, on the other hand, interact in three main ways:

1. *Pair production*, which is a process in which a high energy photon (larger than twice the rest mass of an electron) is converted into an electron and a positron; this process happens in the vicinity of an electric field. In a calorimeter, the atomic nucleus is responsible for providing this electric field,
2. *Compton scattering*, in which a photon scatters off an electron in an atom, freeing the electron from its shell,

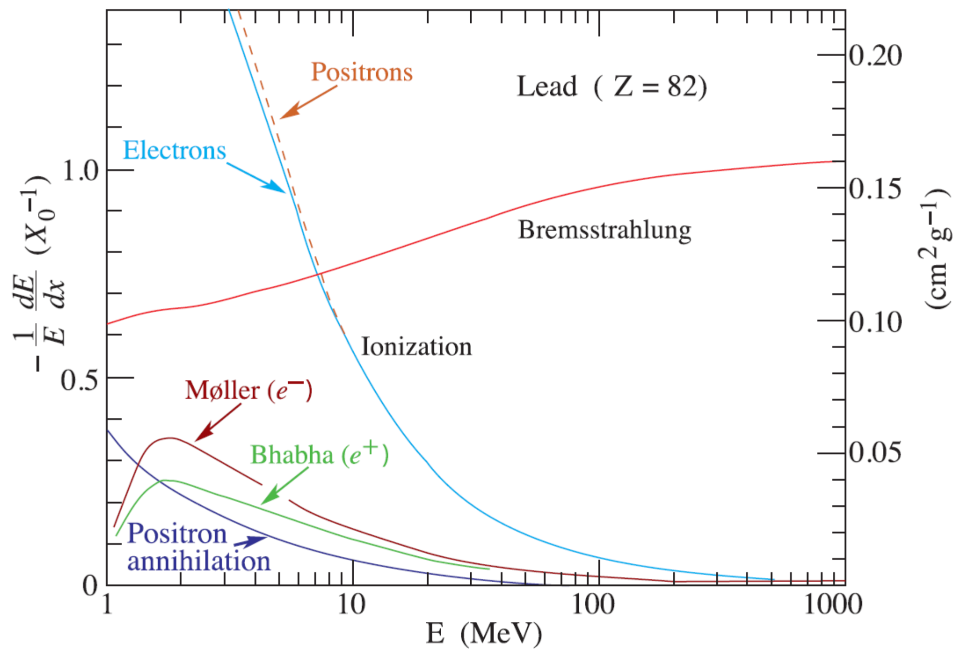


Figure 2.1: Fractional energy loss per radiation length in Pb as a function of the energy of electrons [9] (fig. 33.11, p. 447).

3. *the photoelectric effect*, where a photon is fully absorbed by an atom, and an electron is emitted from a shell.

Photon pair production probabilities for elements of various mass can be seen in figure 2.2. At low energies, the photoelectric effect dominates. Clearly, higher energy photons have a higher probability of producing an electron-positron pair for any interaction.

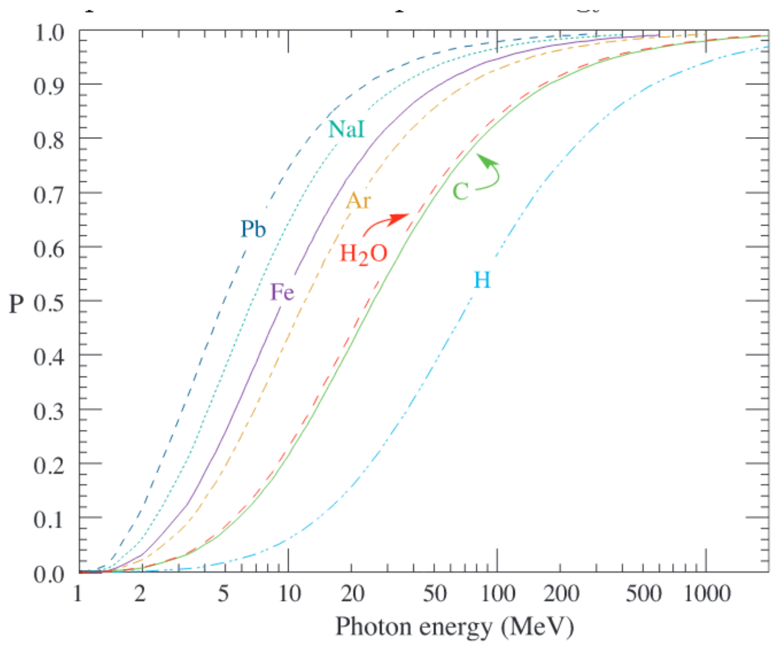


Figure 2.2: Probability P that any given photon interaction will result in a conversion to an e^+e^- pair as a function of the photon energy [9] (fig. 33.15, p. 449).

All of the processes mentioned above are listed in descending order of the particles' energies or their influences on energy loss of a particle. As such, as high energy particles enter the calorimeter, they start losing energy first through pair production and bremsstrahlung (at energies ≥ 100 MeV) gradually losing enough energy to be fully absorbed by the absorber material.

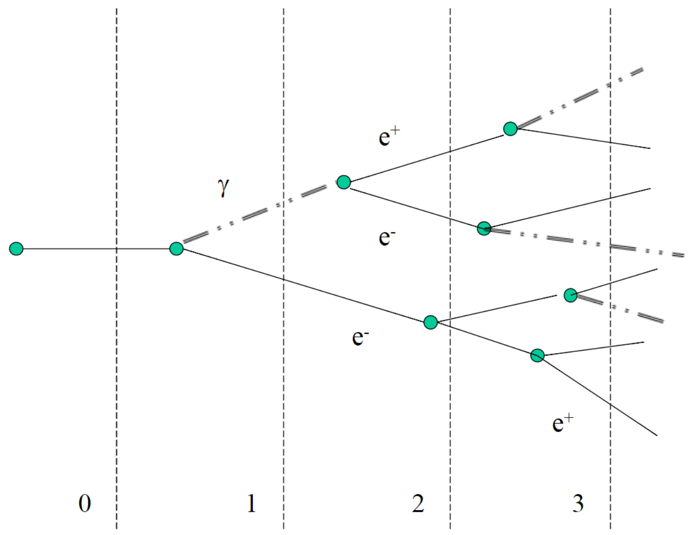


Figure 2.3: A schematic depiction of an electromagnetic cascade [10]. The numbers on the bottom indicate the number of radiation lengths traversed.

The cascade progression can be divided into three sections: the *initial* shower, the shower *maximum*, and the shower *tail*. A typical cascade starts with an electron or photon in the range of a few GeV. The primary loss of energy at this stage is caused by Bremsstrahlung. Only a few photons have energies high enough (> 5 MeV) to avoid being absorbed through Compton scattering and the photoelectric effect; these photons go on to lose their energy through pair production. If these pairs (e^+ and e^-) have a high enough energy, these will once again lose energy through Bremsstrahlung and pair production, consequently. This section of the cascade is called the *core*.

In order to derive some information on the electromagnetic showers, a simple model is generally used. In this model, each particle has exactly one interaction per radiation length (equation 6); in that interaction, an electron loses half of its energy to produce a photon, and a photon produces an electron-positron pair which each carry half of the photon energy. This model is illustrated in figure 2.3.

The average energy of the particles decreases as they traverse the material. The particles reach a point where pair production no longer occurs; this cascade section is called the *maximum*. After the shower maximum, particles in the cascade lose their energy through other processes than pair production. Ionisation and the photoelectric effect become dominant, which diminish the particle showers until they have no more energy to produce any secondary particles; all the remaining energy is exhausted without generating any new particles.

Finally, the energy and the number of particles start to decrease, which leads to a halt in the deposition of energy in the medium. The cascade section where the energy deposition becomes insignificant, is called the *halo*.

Of course, actual electromagnetic cascades are far removed from the ideal model where the secondary particles propagate in a confined angle-space, interact exactly once per radiation length, and have uniform energy distributions among them. A more accurate representation of an electromagnetic cascade can be seen in figure 2.4.

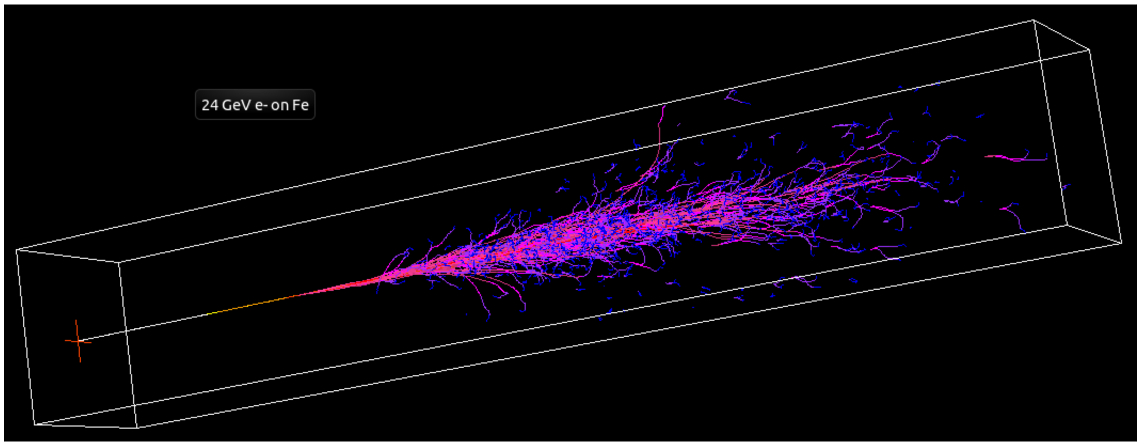


Figure 2.4: A more accurate depiction of an electromagnetic cascades [11]. Picture shows a simulation of a 24 GeV electron in iron.

The longitudinal evolution of an electromagnetic shower is characterised by the *radiation length*, which is a material characteristic related to electromagnetically interacting particles' loss of energy. The radiation length is defined as the mean distance over which a high-energy electron loses all but $\frac{1}{e}$ of its energy by bremsstrahlung. The parametrisation of the material dependence of the radiation length is given by [12]:

$$X_0 = \frac{716(\mathrm{g}\cdot\mathrm{cm}^{-2})A}{Z(Z+1)\ln(287/\sqrt{Z})} \approx 180A/Z^2(\mathrm{g}\cdot\mathrm{cm}^{-2}) \tag{6}$$

We can also approximate the radiation length in a compound or mixture by:

$$1/X_0 = \sum V_j/X_j, \tag{7}$$

where V_j and X_j are the fractional volume and radiation length for the jth element.

The critical energy E_c is defined as the energy at which the losses from ionisation are equal to the losses from bremsstrahlung (EGS4 definition), or the energy at which the ionisation loss per radiation length equals the electron's own energy (Rossi definition) [13]. The definitions of the critical energy can be seen in Figure 2.5.

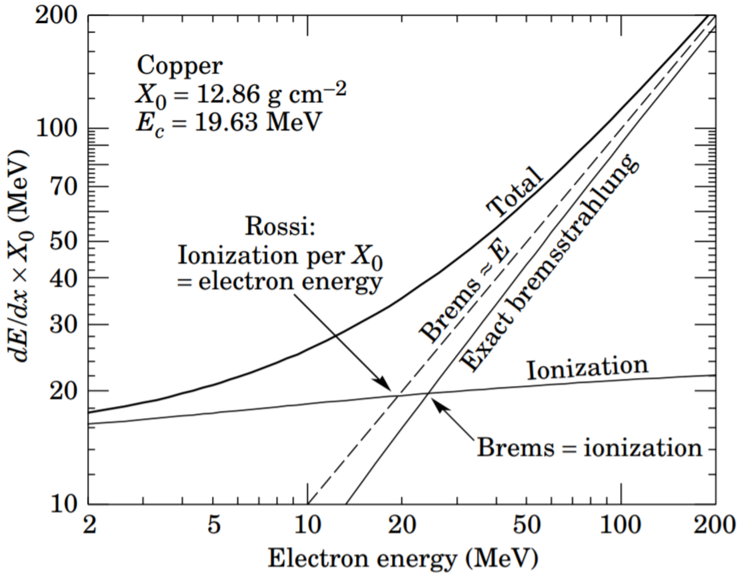


Figure 2.5: Energy loss $\frac{\mathrm{d}E}{\mathrm{d}x}$ for electrons in copper as a function of the electron energy [9] (fig. 33.13, p. 448).

The critical energy for electrons in solids is given by:

$$E_c = \frac{610\mathrm{MeV}}{Z + 1.24}, \tag{8}$$

where Z is the atomic number of a material. Note that this is an empirical fit of E_c based on Rossi's definition of the critical energy, which has the form $a/(Z + b)$. The fits are different for different states of matter because of density effects [13]. The electron critical energy as obtained from Rossi's definition can be seen in Figure 2.6 below.

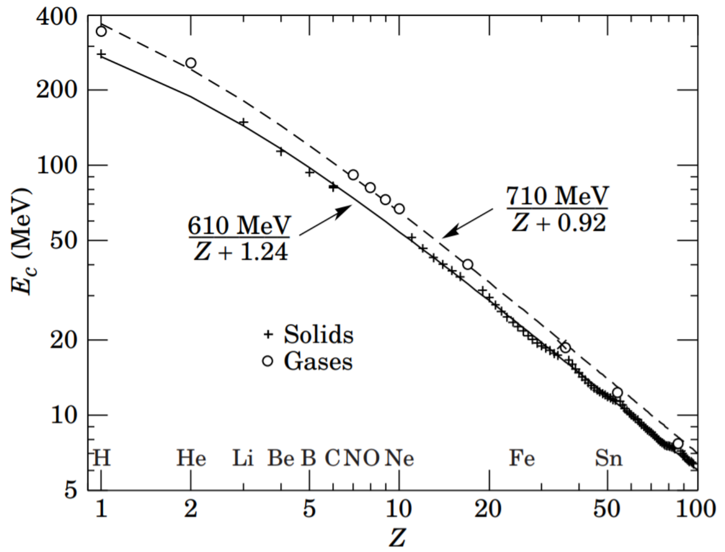


Figure 2.6: Electron critical energy for chemical elements using Rossi's defition. Fits are shown for solids and liquids (solid line) and gases (dashed line) [9] (fig. 33.14, p. 448).

In order to describe cascade behaviour, it is convenient to introduce the scale variables

$$t = x/X_0, \qquad y = E_0/E_c, \tag{9}$$

in order to ensure the distance a cascade traverses is scaled with a material's radiation length, and its energy is scaled with the critical energy. The mean longitudinal evolution of the energy deposition in such a cascade can be described by the empirical equation [9]:

$$\frac{\mathrm{d}E}{\mathrm{d}t} = E_0 b \frac{(bt)^{a-1} e^{-bt}}{\Gamma(\alpha)}, \tag{10}$$

where a and b are dimensionless fit parameters, and $\Gamma(\alpha)$ is the well-defined gamma function. From the previous equation, the cascade maximum depth is given by $t_{max} = a - 1/b$; it is found that there is an energy-dependent expression of the shower maximum:

$$t_{max} \approx 1.0 \times [\ln y + C_j], \qquad j = e, \gamma, \tag{11}$$

where C_e = -0.5 for cascades induced by electrons, and C_γ = +0.5 for cascades induced by photons. One can then use equation 10 by finding $(a-1)/b$ from equation 11, and then assuming that $b \approx 0.5$ [9]. One can empirically find that the depth that contains 95% of the cascade's energy is given by:

$$t_{95\%} \approx \frac{a-1}{b} + 0.08Z + 9.6 \approx t_{max} + 0.08Z + 9.6 \tag{12}$$

The transverse evolution of an electromagnetic shower is characterised by a material's *Molière radius*, which can be described in terms of the material's radiation length and the critical energy. It is defined as the radius of a cylinder with axis coinciding with the shower axis, containing on average 90% of the total energy deposition of the shower (~95% of the cascade energy is contained in a cylinder of radius $2R_M$) [14]. The Molière radius is given by:

$$R_M = \frac{21.2\text{MeV}}{E_c} X_0, \tag{13}$$

The measurement of energy in an electromagnetic cascade is based on the principle that the total deposited energy in the absorber/detector material is proportional to the initial energy of the incoming particle, E_0. As such, the total track length of a cascade T_0, which is defined as the sum of all particle tracks in a cascade, is given by [15]:

$$T_0(\text{g/cm}^2) \propto X_0 \frac{E_0}{E_c} = X_0 \cdot y, \tag{14}$$

which leads to the conclusion that measuring the tracks produced by the cascade provides information about the particle's initial energy E_0; energy measurement accuracy of a calorimeter is therefore a necessary quantity to define in order to gauge its efficiency.

The full energy resolution of a calorimeter is given by:

$$\frac{\sigma_E}{E} = \frac{a}{\sqrt{E}} \oplus \frac{b}{E} \oplus c, \tag{15}$$

which can be divided into three terms: the stochastic term involving a, the noise term involving b, and the constant term c. The $\oplus$ indicates the quadratic sum of the terms. Clearly, each term in equation 15 depends on the energy in a different manner; as such, different energy ranges can give significantly different results across experiments. For an ideal calorimeter of infinite size that does not suffer from noise or other instrumental interference, T_0 is proportional to the total number of tracks in the evolution of a cascade; as such, the energy resolution in such a calorimeter is given by:

$$\sigma(E) \propto \sqrt{T_0} \quad \Rightarrow \quad \frac{\sigma(E)}{E} \propto \frac{1}{\sqrt{T_0}} \propto \frac{1}{\sqrt{E_0}}, \tag{16}$$

The main concern of calorimeter design is the average shower development. The shower development (direction, depth, number of interactions) is a purely stochastic process, which leads to certain variations at equal depths, in turn resulting in measurement uncertainty. As detectors cannot be infinitely large, some information about interactions in showers will be lost; if a cascade starts earlier, more of the cascade can be contained (figure 2.4, bottom). Since the average number of interactions increases with the logarithm of the energy, while statistical fluctuations decrease with energy, variations in energy also form a large contribution to the measurement uncertainty.

2.2 Current ALICE calorimeters

Currently, there are two electromagnetic calorimeters installed at ALICE: the PHOS (PHOton Spectrometer), and the ALICE Electromagnetic Calorimeter (EMCal).

PHOS is a lead-tungstate crystal ($PbWO_4$) detector designed to measure photons that are directly emergent from a collision. PHOS contains 17,280 detection channels, measuring $2.2 \times 2.2 \times 18\text{cm}^3$ volume of $PbWO_4$ crystal, which have characteristics $R_M = 20$ mm, and $X_0 = 8.9$ mm [16]. EMCal is a lead-scintillator sampling calorimeter, which aims to improve measurements related to jet quenching. EMCal's geometry and inner workings are quite extensive, and are detailed in its technical design report [17]. Its characteristics are $R_M = 32$ mm, and $X_0 = 12.3$ mm. The positions of these calorimeters in ALICE are indicated in figure 1.1.

3 Hardware and software

3.1 The FoCal detector

The proposed FoCal detector is designed to study forward physics of incident particles, and will be installed outside of ALICE's magnet at a distance of 7 m. The detector's purpose is to measure incident particles of high pseudorapidity ($3.2 < \eta < 5.3$) in order to test the existence of the colour-glass condensate. The idea is to discover more about parton kinematics with the help of photons, as they do not suffer from final state interactions; the FoCal is designed to measure photons in the forward region.

Measurements done with the FoCal detector will significantly improve the understanding of parton distribution functions in the very low-x (parton momentum fraction) regions for nuclei, by providing unique experimental constraints on these systems. As it stands, the uncertainties in the low-x ($\mathcal{O}(10^{-5})$) regions are orders of magnitude larger than in medium-x ($\mathcal{O}(10^{-4} - 10^{-2})$) regions; the new detector aims to improve the accuracy of low-x measurements by performing forward measurements. The smallest accessible x in the LHC is given by equation 3; the FoCal detector will provide measurements at high rapidity and relatively low Q ($\mathcal{O}(10^0 - 10^1)$) [18].

The separation distance between the π^0 decay photons is extremely small ($\mathcal{O}(10^{-3})$ m); as such, the FoCal has been designed with high-granularity on a sub-millimeter scale in order to distinguish direct photons from photons that are a result of interactions, and separate the decay photons and their products accurately.

The FoCal consists of an electromagnetic calorimeter and an hadronic calorimeter, called FoCal-E and FoCal-H, respectively. This research focuses solely on a simulation of a prototype of the electromagnetic component, called the mTower.

FoCal-E is a sampling calorimeter; it has sensitive sampling layers made of silicon, and absorber layers made of tungsten. The FoCal will be constructed with low-granularity layers (LGLs) and high-granularity layers (HGLs). The high-granularity layers are especially capable of separating close-by photons and provide a very comprehensive analysis of the cascade shape, because of their detector cell size. Tungsten is chosen as the absorber material because of its rather small radiation length (3.5 mm), Molière radius (9.3 mm), and low cost compared to other materials with these properties [19]. Having a material as dense as tungsten also allow the calorimeter to be quite short in length; these characteristics make tungsten an excellent choice for a compact detector, since the diameter of the detector need ideally be only about 4 R_M to contain ~95% of the cascade energy.

A schematic overview of the FoCal detector design is given in figure 3.1.

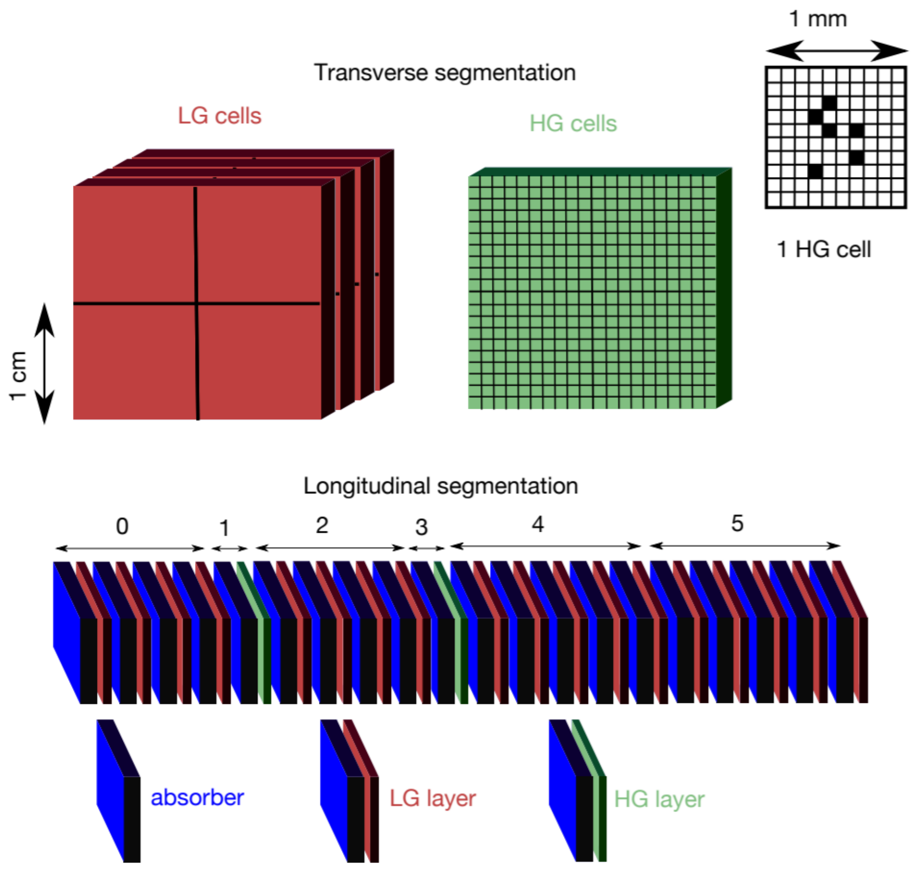


Figure 3.1: A schematic depiction of the proposed FoCal design [20]. *Top*: illustration of the difference between the LGLs and HGLs. *Bottom:* layout of the full calorimeter design. The absorbers will be approximately one radiation length thick, with HGLs and LGLs sandwiched between them.

3.1.1 ALPIDE CMOS sensors

The sensor layers of the mTower HGLs are made of silicon chips. The new ALPIDE CMOS chips [21][22] are monolithic active pixel sensors (MAPS). These detectors aim to collect charge by means of drift or diffusion onto a collection electrode. This is done by maximising the thickness of the sensitive layer, and applying reverse bias to uniformly deplete the sensitive region of the pixel. When compared to analog sensors, the biggest difference with MAPS are the CMOS structures employed in the construction of the pixel circuitry; this is because the readout electronics are on the same wafer as the sensor. This allows them to process all the analog data on the wafer itself, making the chips smaller and more efficient. These CMOS structures have an epitaxial layer of high resistivity material; these chips can simultaneously collect generated charge from the high resistivity layer, and amplify the collected charge. A depiction of the inner structure of the sensors can be seen in figure 3.2.

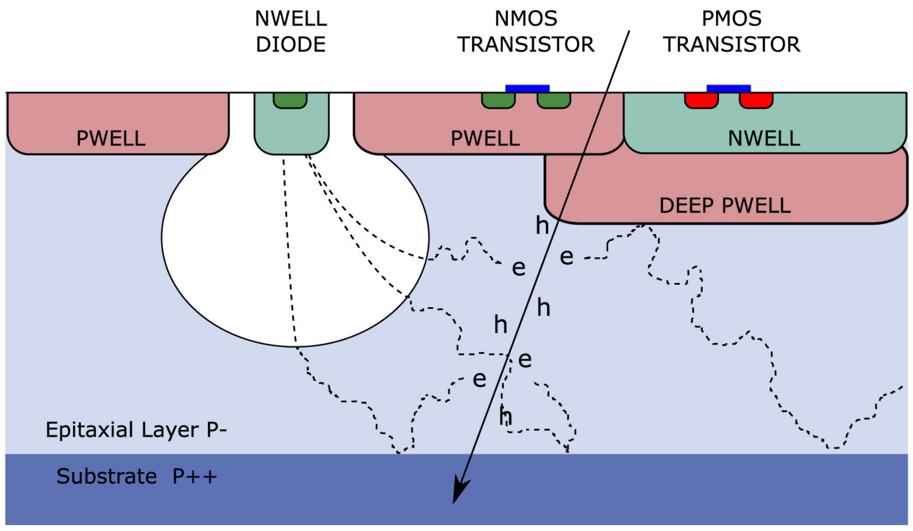


Figure 3.2: An illustration of the internal ALPIDE CMOS structures, with different layers and regions indicated [21].

The ALPIDE sensors have a pixel matrix of 512 × 1024 pixels of 29.24 × 26.88 μm^2 surrounded by a peripheral circuit region which converts deposited energy into a binary-based *digital* signal (hit/no-hit). This circuit region allows the chip to interface with other electronic systems; with serial and parallel data ports that can be used with FPGA's[3], and a bi-directional control line which allows for command-relay and reading/writing registers. A schematic view of the ALPIDE chips can be seen in figure 3.3.

The proposed FoCal-E design and its motivation are given in [23]; this thesis does not delve into the specifics of the final detector design, as it is of little relevance. The prototype for the FoCal-E detector which is studied in this thesis is called *mTower*.

Each sensor layer of the mTower prototype consists of two chips placed alongside one another, which makes the combined pixel grid have dimensions 1024 × 1024. Any time a pixel is activated ('hit'), the pixels will send out a digital signal to their readout module; the combined data of all the pixels is then used to measure the incoming particle energy.

Each chip has dimensions 30 × 15 mm^2 × 100 μm, where the z-axis is chosen to coincide with the direction of the layer stack (the direction of the incident particles). A fraction of the total chip area is inactive because of the periphery necessary for data readout; this accounts for 1.208 mm. Placing all pixels adjacent to one another leads to an inactive area that spans 30 μm in the x-direction on either side of the connection.

The separate 1024 × 512 pixel grids are connected to form a 30 × 30 mm^2 chip layer. In practice, this connection is not perfect, and there is an average gap between the two sensors of 100 μm. The ALPIDE chips are attached to the tungsten absorbers with the use of adhesive. The chips are superceded by the readout electronics and the flex.

[3]Field-programmable gate array: an integrated circuit that contains an array of programmable logic blocks, which can be configured to perform complex combinational functions or function as logical gates.

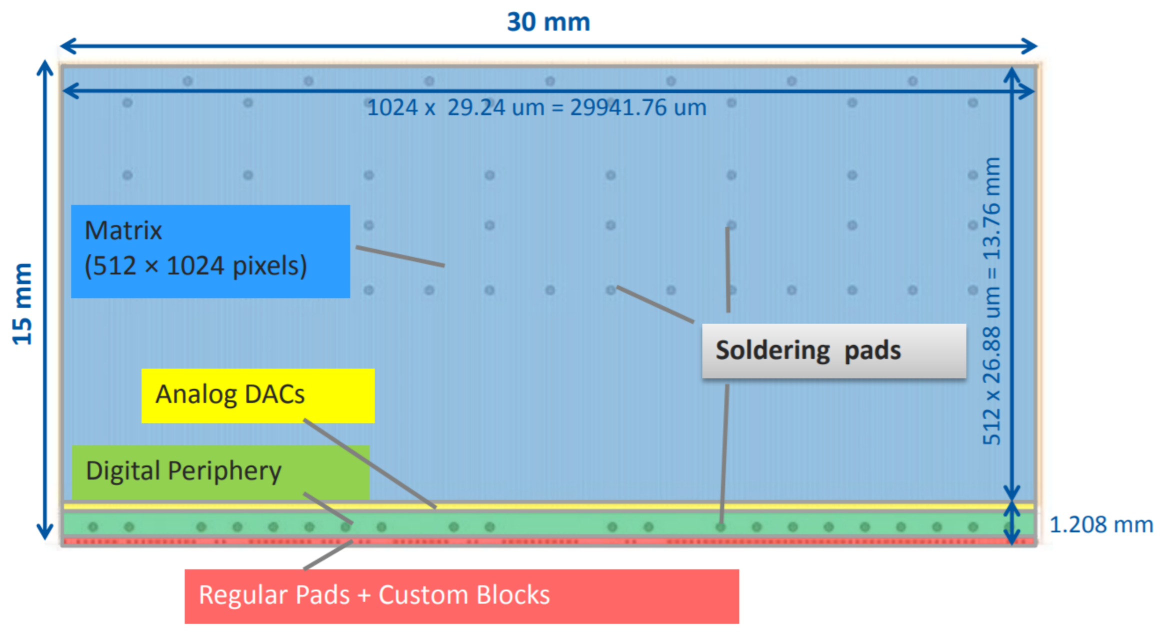


Figure 3.3: A schematic depiction of the ALPIDE CMOS chip, which shows active area and dimensions.[24]

3.1.2 Charge sharing

Whenever a particle interacts with pixelated chips inside the detector, the processes which the electronics experience are not quite straightforward. Charge is collected on this pixel diode, which can then communicate with the readout electronics about receiving charge. Each ALPIDE CMOS pixel sensor contains several regions which interact with charge carriers in a different way. The particle may penetrate the sensor from the 'top' (as indicated in figure 3.2), releasing charge carriers (free electrons/holes in the material); the charge carriers will then travel through the epitaxial layer with a certain resistivity. This generates free charge, an electron-hole pair, at the point of impact.

As one can imagine, the size of the pixels makes it so that not every particle impacts a pixel exactly on the collection diode. Whenever a particle impacts within the epitaxial layer, its charge does not stay in one place; in fact, it starts to diffuse away from the impact zone. This process of diffusion is purely stochastic; as such, measurements based on charge diffusion have a certain statistical uncertainty.

A few assumptions can be made to simplify the model of diffusion in a pixel; namely, that the charge diffuses away from the point of impact *isotropically*, and that the charges *reflect on the substrate*, with a reflection angle equal to the incident angle. The former allows one to simulate the stochastic nature of this process by the means of a Gaussian distribution, which gives the probability of the charge travelling a distance r away from the impact position (and consequently, the probability of the charge reaching the collection diode). This assumption makes it possible to easily implement the effects of charge diffusion in a simulation for the ALPIDE CMOS chips.

Because of the random nature of the diffusion process, having a large amount of charge deposited in one pixel, or the impact zone being very close to the edge of two pixels, can easily lead to what is known as charge *sharing*. When charge is shared, not only the diode on the pixel which was initially hit gives a signal, but other diodes in adjacent pixels can send out a signal too. This creates a *clustering* effect, in which multiple pixels are clustered together when impacted by any particle. The size of these clusters could depend on the total deposited energy in the pixel, its threshold, and its incident angle.

There is a way to influence the trajectory of the charge created at the impact point: charge *drift*. Charge drift is a process in which an external electric field is applied to the pixel to make the charge move in a certain desired direction. This makes it possible to move the free charge closer to the diode as a way of ensuring more charge is collected. While the effects of drift can be implemented in the ALPIDE chips, the mTower prototype will not make use of this feature.

3.1.3 Detector geometry

The tungsten absorber layers of the mTower each have dimensions of 40 × 35 × 3 mm^3. The different layers in the calorimeter are separated by tungsten spacers that ensure adjacent layers do not rest directly on top of each other; these spacers have dimensions 4 × 35 × 0.5 mm^3. The ALPIDE chips are placed between these spacers. Each layer of tungsten is constructed to have the thickness of approximately one radiation length. Because of the dimensions of the prototype, measurements will have to be done at perpendicular incidence.

We can estimate the radiation length of the mTower; by referring to equation 7, we can calculate the radiation length as [20]:

$$\begin{aligned}\frac{1}{X_0} &= \frac{V_W}{X_W} + \frac{V_{Si}}{X_{Si}} + \frac{V_{PCB}}{X_{PCB}} + \frac{V_{adhesive}}{X_{adhesive}} + \frac{V_{air}}{X_{air}} \\ &= \frac{0.846}{3.5\text{mm}} + \frac{0.031}{93.6\text{mm}} + \frac{0.041}{167.6\text{mm}} + \frac{0.038}{400\text{mm}} + \frac{0.044}{303900\text{mm}} \\ &\approx \frac{1}{4.0\text{mm}}\end{aligned} \tag{17}$$

In order to calculate the size of the detector necessary to encapsulate the full particle shower, it is assumed that the critical energy of the mTower is equal to that of tungsten, which is 8.107 MeV according to equation 8. According to equation 13, combined with equation 17, the Molière radius is then given by:

$$R_M = \frac{21.2\text{MeV}}{8.1074\text{MeV}} 4.0\text{mm} \approx 10.5\text{mm}. \tag{18}$$

The thickness necessary for the full detector can be calculated using equation 12; for incident particles in the 100 MeV range, we find that $t_{95\%} \approx 18.1 + C_j$, and in the 10 GeV range we find $t_{95\%} \approx 23.4 + C_j$, where $j = e, \gamma$, $C_e = -0.5$, $C_\gamma = +0.5$. Based on these calculations, the tungsten absorber layers should have a total thickness of about $24X_0$ in order to contain 95% of the cascade energy in the high-energy ranges; this will automatically contain all cascade energy in the lower ranges.

3.2 GEANT4 toolkit

This section on the GEANT4 simulation toolkit by and large follows the GEANT4 Physics Reference Manual [25].

GEANT4 is a C++-based object-oriented toolkit made to simulate the interactions of particles with matter. GEANT4 allows the user to specify detector geometry, physics processes that dictate how the particles behave, and adjust parameters concerning the generation of incident particles. GEANT4's simulations of interactions are based on Monte-Carlo methods.

3.2.1 Monte-Carlo methods

Monte-Carlo methods are a class of algorithms that employ repeated random sampling in order to obtain numerical results. These methods are used in physics mostly for generating draws from probability distributions, under the assumptions that the systems they are used for are probabilistic in nature, and can be described by probability density functions. By generating random samples that are based on the properties of the system's probability density function, the algorithm can deduct or solve processes or problems of said system.

Monte-Carlo methods are particularly useful in problems that have many coupled degrees of freedom and are stochastic in nature, like in particle physics. Any interactions of particles in a medium are analytically intractable; as such, methods have to be introduced in order to solve problems related to the evolution of these particles and their products numerically.

3.2.2 GEANT4 processes

The main principle GEANT4 simulates is particle transport. The toolkit does this through a combination of its stepping manager class (which simulates time steps), its physics processes class and the transportation class (which identifies the upcoming volumes that need to be simulated).

As mentioned, GEANT4 simulates particle transport through matter step by step. *True step lengths* for interactions are randomly sampled, and are based on an interaction's *mean free path* length (λ), or derived from other step limitations that can be specified by different GEANT4 settings. The toolkit will take the smallest limit and define it as the true step length.

The mean free path length is calculated with the use of the *cross section* of a particular physics process, and the *number density* of particles in a given volume. In a compound material, the number density of atoms of the i^{th} volume element is given by:

$$n_i = \frac{\mathcal{N} \rho w_i}{A_i}, \tag{19}$$

where $\mathcal{N}$ is Avogadro's number, ρ is the density of the medium, w_i is the proportion of mass of the i^{th} element, and A_i is the molar mass of the i^{th} element. The mean free path length (interaction length) of a physics process can be expressed in terms of this number density and the total cross section:

$$\frac{1}{\lambda(E)} = \left(\sum_i [n_i \cdot \sigma(Z_i, E)] \right), \tag{20}$$

where $\sigma(Z,E)$ is the total cross section per atom of the physics process, and the summation runs over all of the components of the material. From this equation, it is clear that GEANT4 models the interaction length of a certain process as the probability of a particle with a cross section σ to collide with an atom of the material. This implies that the average distance a particle travels without any process occuring is equal to the average distance that particle travels without colliding with an atom.

The interaction length is a powerful measure of how far the particle travels in the simulation. Unfortunately, this interaction length cannot be used to directly sample the probability of an interaction in a heterogeneous detector, as the processes of interaction of a certain particle depend on the medium. The number of interaction lengths a particle travels is given by:

$$n_\lambda = \int_{x_1}^{x_2} \frac{dx}{\lambda(x)}, \tag{21}$$

Now, if n_r is a random variable that represents the number of mean free paths from a random point x_i to the point of interaction x_{int}, the variable n_r has the cumulative distribution function (CDF):

$$P(n_r < n_\lambda) = 1 - e^{-n\lambda} \tag{22}$$

With this in mind, n_λ is updated after each step length Δx according to the formula:

$$n'_\lambda = n_\lambda - \frac{\Delta x}{\lambda(x)} \tag{23}$$

Ideally, the true step length is minimised in time until $n'_\lambda \to n_\lambda$, and the step originating from $s(x) = n_\lambda \cdot \lambda(x)$ is the shortest, which causes this specific process to trigger for the upcoming steps; after this, GEANT4 updates n_λ for other interactions and repeats the process described above.

The transportation class simulates the evolution of particles whenever they enter a new material; this class determines the boundaries of the volumes and calculates the true step length which would cause a particle to cross over into the next volume. Whenever a geometrical boundary is reached, the transportation class identifies the physics processes of the next volume and determines the particle's motion. Since FoCal-E is an electromagnetic detector, the main application of GEANT4's transportation class will be the description of motion of electrons, positrons, and photons in an electromagnetic field, for which the toolkit solves the equations of motion using Runge Kutta methods[4].

Similar to the particle transport, most of the simulations of physics processes are based on their cross sections. GEANT4 offers an extensive list of packages that simulate physics processes in different energy ranges. There are several hadronic models, which include the quark-gluon string model (QGS), the Fritiof model (FTF), and the Bertini intra-nuclear cascade model (BERT). CERN's documentation on GEANT4 recommends using the Fritiof and quark-gluon string models in combination with the Bertini cascade model to simulate high energy physics calorimetry, as the studies of testbeam data show that a combination of string- and cascade models improves agreement with LHC testbeam data in longitudinal and lateral evolution, as well as resolution [26]. As these are the best suitable lists, this thesis will exploit these in the construction of simulations in GEANT4.

3.3 Allpix^2

Allpix^2 is a generic pixel detector simulation framework designed to allow integration of new detectors in an easy matter, as well as providing a framework that allows precise measurements of energy deposition propagation models and facilitate the usage of very precise electric fields. Allpix^2's integration of these phenomena is mostly based on the existing Allpix framework, which in turn was based on GEANT4.

The framework is written in (modern) C++, and allows the user to log and specify information, configurations, and geometry of the pixel detectors, as well as an intuitive way of implementing individual physics modules to test these detectors. A useful property of the framework is that it allows for single detector chips to be tested, as well as nesting them in so-called telescopes, which allows for more accurate data measurements as the telescope consists of chips for which data is known. Alternatively, the telescope can be made out of identical layers as the device under test (DUT), which effectively simulates an entire tracker or calorimeter.

Allpix^2 is shipped with a fairly extensive list of detector and physics modules, which enables fluent configuration of a test setup of single or multiple chips inside a testbeam. The framework

[4]A family of iterative methods to approximate solutions of ordinary differential equations.

uses a modular approach to construct the full simulation of an event as it passes through the detector; a depiction of the different modules can be seen in figure 3.4; this approach lets the user quickly plug together simple simulations based on the modules they require for their simulation.

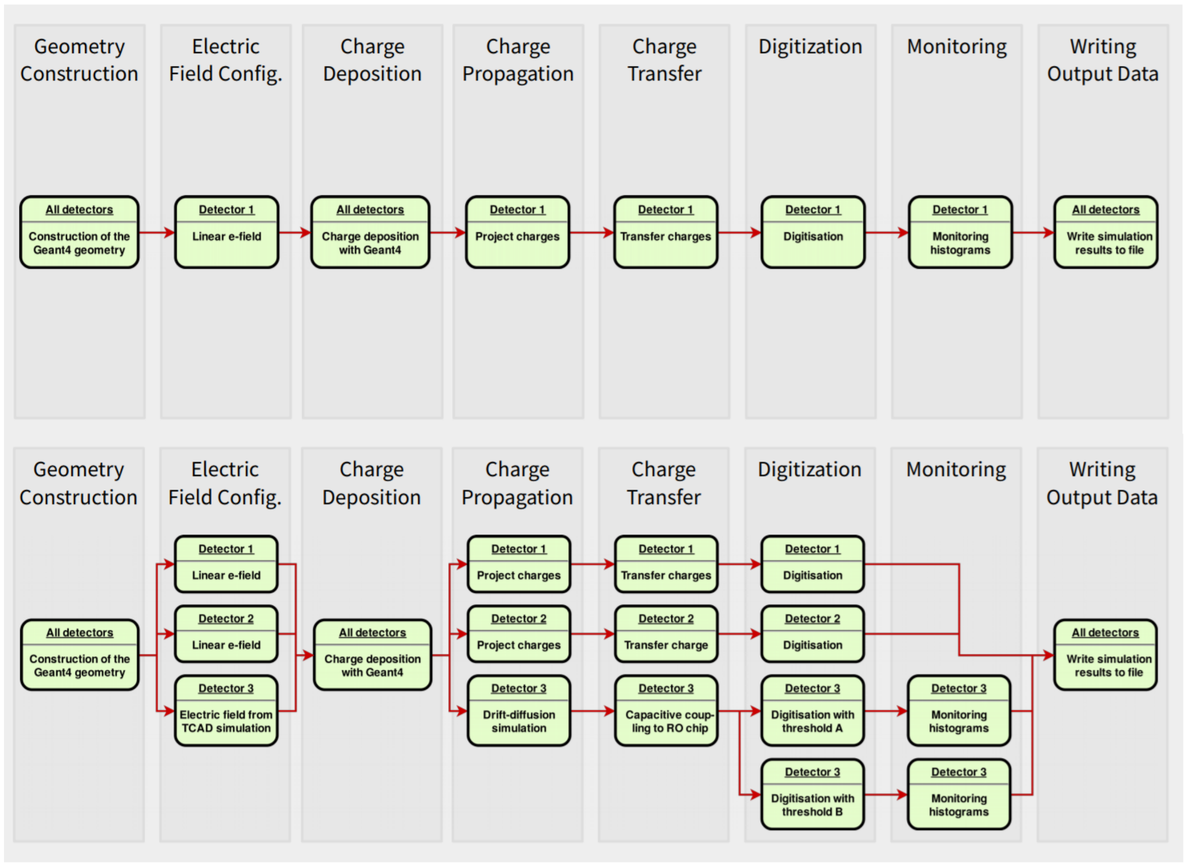


Figure 3.4: *Top*: an example, simple overview of modules used in an Allpix^2 configuration. *Bottom*: a more complicated set of modules. These modules are placed in linear order in the configuration file.

The framework allows the user to specify the type of electric field to simulate; this makes it possible to study non-linear electric fields such as those that can be configured for charge drift in ALPIDE chips.

The framework has module dependence on GEANT4, which contains a model for charge deposition. On top of that, charge propagation can be simulated by means of either projection[5] or drift diffusion[6]. Allpix^2 also has a module dependence on Eigen3, which is a framework written in C++ that allows for complex calculus operations.

Allpix^2 is capable of simulating a full detector setup, not just the DUT. It also contains Monte-Carlo Truth[7] information, which allows for precise description of entry and exit points of primary and secondary particles.

While Allpix^2 does not add a remarkably huge amount of functionality in terms of physics processes, it refines a lot of what GEANT4 has to offer. Starting from the digitisation of analog data, to enabling the user to accurately describe deposition and propagation for each separate charge carrier, and even determining thresholds and uncertainties from collected data. The framework

[5]Project charges on side of the chip implant, adding diffusion; only to be used for linear fields.
[6]Drift along field lines determined by Runge-Kutta-Fehlberg methods.
[7]A way of connecting simulations using Monte-Carlo methods and theory/reality [27].

produces more viable and interesting output than GEANT4, and is more versatile in terms of visualisation and parameter changes. The main point of interest is of course to compare the results that using GEANT4 directly provides with the ones that Allpix^2 does, in the modules that produce similar output.

4 Methodology

The following section discusses the methods used to obtain results for this thesis, including general outlines for the incorporated Allpix^2 modules and detector models. For the sake of clarity, the method which was developed for this thesis in the bare GEANT4 framework was dubbed 'SimpDiff' (simple diffusion model); therefore, every reference to data obtained directly from GEANT4 with this addition will henceforth be referred to as SimpDiff.

4.1 Allpix^2 simulations

The first simulations were performed in the Allpix^2 framework, which allows for very precise simulations of electric fields and microscopic properties of the ALPIDE chip, including TCAD simulation data[8] for the electric fields and the depletion/implant zones specific to each chip.

As this is a fairly new framework that has not been used to simulate the ALPIDE chips yet, this thesis (briefly) provides the knowledge necessary to understand why and how simulations were performed.

4.1.1 Modules

The modules used in the simulations of the ALPIDE chip (and consequently the mTower stack) are listed below:

- GeometryBuilderGeant4,
- DepositionGeant4,
- ElectricFieldReader,
- GenericPropagation,
- SimpleTransfer,
- DefaultDigitizer,
- DetectorHistogrammer.

GeometryBuilderGeant4 This module is used to create a 'canvas' world for the detectors to exist in, based on the GEANT4 framework. It generates a blank 3D world frame with configurable margin and enables the usage of various materials to construct the detectors within this frame. The parameters of this module allow the user to define the detector geometry in the world frame.

[8]Technology Computer-Aided Design: the use of computer simulations to develop and optimise semiconductor processing technologies and devices.

DepositionGeant4 A module which deposits charge carriers (electrons or electron holes/-pairs) in the active volume of all specified detectors. It functions as a wrapper around the logic present in GEANT4 and depends on the global geometry specified by aforementioned Geometry-Builder module.

DepositionGeant4 allows for customisation of physics lists, particle types, source and beam characteristics, and energy deposition/charge carrier creation. The module's method for determining when processes take place and what to do with the event information is based on the familiar and predefined GEANT4 step length.

ElectricFieldReader This module allows the specification of the electric field present within the sensors; they can be constant, linear (specifying depletion and bias voltage), or more complex. The complex electric fields can be read out from a .init-file. The framework is capable of building these files from provided TCAD meshes. These meshes are mapped onto each pixel individually, meaning it allows for a very precise simulation of a sensor's internal field. *ElectricFieldReader* also enables the specification of depletion region, which is very useful for simulations of the ALPIDE sensors, as the depletion depth is generally known where the depletion voltage might not be.

GenericPropagation A module which simulates the propagation of electrons and/or electron holes through all sensitive volumes of the detector. This module deposits the particles in the sensor volume and propagates the charges through a simulation which is a combination of charge drift and charge diffusion. The charge drift is calculated using the charge carrier velocity and the magnetic field; these are derived from the charge carrier mobility parametrisation done by C. Jacoboni et al.[28] and the properties of the Lorentz drift, respectively. The mobility is given by:

$$\mu(\vec{x}) = \frac{v_m}{E_c}\frac{1}{(1+(E(\vec{x})/E_c)^{\beta})^{1/\beta}} = \mu_0\frac{1}{(1+(E(\vec{x})/E_c)^{\beta})^{1/\beta}}, \tag{24}$$

where v_m is the charge carrier velocity, E_c is the critical energy specific to the material, $E(\vec{x})$ is the energy of the current particle based on its position, and $\beta = 1/k_bT$, where k_b is the Boltzmann constant, and T is the temperature. The three parameters v_m, E_c, and β are defined distinctly for electrons and holes in [28].

A fourth-order[9] Runge-Kutta-Fehlberg method [29] has been employed to integrate particle propagation with the help of magnetic and electric fields. After each Runge-Kutta step[10], charge diffusion is calculated by drawing an offset from a two-dimensional Gaussian distribution which is calculated from the Einstein relation:

$$\sigma = \sqrt{\frac{2k_bT}{e}\mu t}, \tag{25}$$

where e is the carrier charge, μ is the mobility, and t is the Runge-Kutta time step. The propagation continues until the (set of) charges reach the surface of a sensitive detector volume.

The direction of propagation depends on the electric and magnetic fields present in the simulation. Through configuration of linear electric fields as they will be present in the ALPIDE chips, the module will ensure that the charge carriers are transported to the implant side and not to the substrate, depending on which charge carrier type is selected.

[9]Implies that the local truncation error is on the order of $\mathcal{O}(h^5)$, where h is the step size.

[10]Iteration of the approximation procedure.

SimpleTransfer This module combines individual propagated charges to form a set of charges on the sensor pixels, allowing them to be processed by the detector electronics. It maps the incident charge to the nearest pixel, while ignoring pixels whose distance to the implants is too large.

The module also allows for specification of diode characteristics for each detector, enabling one to pick charge carriers from specific regions within the sensors when applying TCAD meshes.

DefaultDigitizer A module which translates collected charges into a digitised signal proportional to input charge. It is capable of simulating Gaussian noise from the readout electronics, as well as allowing a threshold to be set. On top of that, it is possible to simulate the response of an ADC[11] with customisable resolution.

For every pixel charge, the module follows specific steps in its process of digitisation. These steps are listed below:

- Gaussian noise is added to the input charge,
- Input charge is preamplified (multiplied by a predefined gain factor), and then smeared with a Gaussian distribution for each event to simulate the stochastic nature behind charge sharing as it is described in section 3.1.2,
- Threshold is applied, which discards any pixel charges that fall below it and only accounts the ones that surpass it. Again, the threshold is smeared with a Gaussian distribution for each event,
- ADC inaccuracy is simulated using Gaussian smearing, which allows for ADC noise simulations.

While it is possible to simulate noise in the Allpix^2 framework using this module, it was not used in the generation of results for this thesis.

DetectorHistogrammer This module provides a large collection of output data, which allows for quick inspection of the simulation data. The module clusters together input hits in every sensor volume. It does this by taking the input data from the PixelHit object[12] which is inherently kept track of, and loops over the hits. If a PixelHit is free-standing, a new cluster is created. Hits are added to a cluster if they are adjacent to the cluster. Clusters are merged if case of multiple adjacent or overlapping clusters.

The framework decides whether a hit should be added to a cluster on the basis of distance from the first incidental particle; when a cluster is created, that pixel is added to a list of 'used' pixels, and the neighbouring pixels are checked for hits by means of distance by pixel index (meaning the module scans a 3×3 pixel area around the first pixel). Subsequently, it checks whether the neighbouring pixels are either contained in the list of used pixels, or are touching the cluster (≤ 1 pixel away, including diagonals), and adds these pixels to the cluster. This ensures that all pixels have their neighbours checked.

DetectorHistogrammer uses the Monte-Carlo truth position given by the MCParticle object as a reference for the track on which the particles travel in order to determine where the particles interact with the detector materials; this object goes hand in hand with PixelHits, which allow it to determine the precise position of impact.

The module creates the following histograms, for every simulation, that are of interest for the purposes of this thesis:

[11] Analog-to-digital converter.

[12] The digitised pixel hits after processing in the detectors front-end electronics.

- A hitmap of all pixels on the pixel grid,
- Distribution of the number of hits per event,
- Distributions of cluster sizes.

The module allows the user to set a granularity for the pixel grid. Effectively, this module serves as a mini-analysis for any detector setup, providing a lot of basic data, some of which is also provided by SimpDiff.

4.1.2 Geometry

A model for the detector geometry has been built within the Allpix^2 framework. For this task, the predefined materials and characteristics were used, which resulted in small inaccuracies

The configuration of one layer of the mTower stack is as follows: the sensor is preceded by the flex and the chip cable, which are specified in the world geometry as being made of aluminum and kapton. The sensor, which is defined to be a monolithic one, is followed by a layer of tungsten absorber. There are two tungsten spacers that separate each layer, which are placed on the tungsten absorber alongside the sensor, chip cable and flex.

The dimensions specified for the chip and absorber are the same as those specified in section 3.1.1. The inactive regions of the pixel sensors were specified as excess regions in the configuration, so their energy depositions are not taken into account when particles pass through them; they do, however, have a seperate PixelHit object associated with them. The internal structure of the electronics was ignored in the construction of this geometry.

Since the Allpix^2 framework does not have any adhesive materials incorporated into its world-building, the thickness of the glue between the separate components has been added to the kapton thickness, as the density of the adhesive more closely resembles that of kapton than that of aluminum. This approximation of the composition leads to a total layer thickness of about 3.5 mm. These layers can be stacked directly behind one another, enabling the simulations of various amounts of layers.

4.1.3 Particle beam

The particles originate from a point source. The source of the particle beam was placed 17 meters in front of the detector, along the z-axis. The particles have a Gaussian spread that varies the incident angle to a hit radius of 6 mm from the origin on the first detector plane, which corresponds to an angle variation of about 0.02 degrees in a cone around the z-axis. With these fairly small deviations, most of the showering is contained within the geometry of the calorimeter. The simulation has been performed for electrons with energies of 50, 100, and 150 GeV.

The particle beam is visualised by means of straight lines of various colours. Every line colour represents a different particle set: the default colour for positively charged particles is blue, red for negatively charged particles, and green for neutral particles. Particles that are a product of interaction with the mTower material can unfortunately not be distinguished by their type, apart from their charge; that is, Delta electrons will have the same colour as regular electrons, and π^0's will have the same colour as photons. This is a sacrifice made for visualisation, though the output data provides us with sufficient information to distinguish the particles.

In the visualisation module, the particles that are inside of the mTower materials cannot be seen explicitly; the visualisation is therefore best suited as an indicator of how the particles enter the stack, and how they leave. An example of an electron test beam fired at a single mTower layer can be seen in figure 4.1.

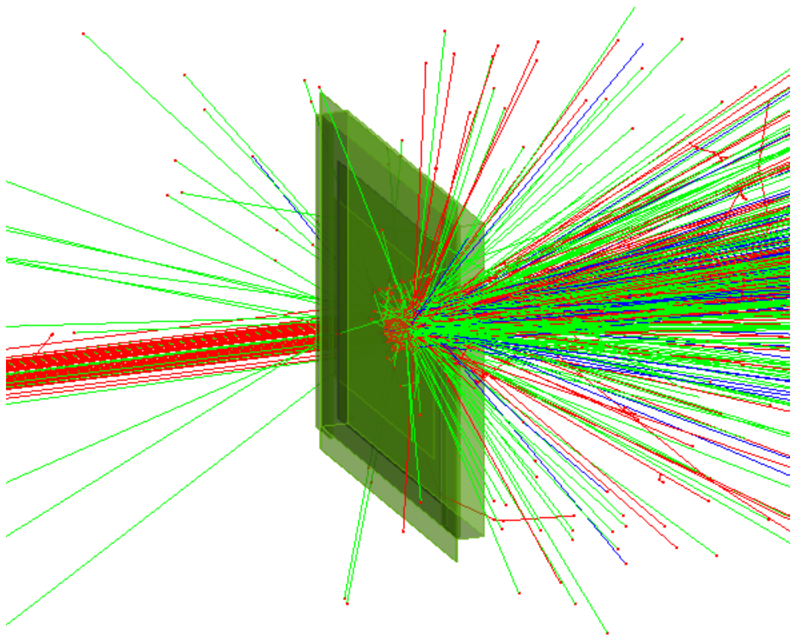


Figure 4.1: A test beam of 100 GeV electrons fired at a single layer of the mTower stack in the Allpix^2 framework, with a 1 mm Gaussian spread of incident angles. This picture is from a simulation of 100 events.

4.1.4 Data analysis

Allpix^2 produces output in the form of a ROOT[13]-file, which has a folder hierarchy that separates the output from all the individual specified modules. For some modules, additional output data can be generated, which can produce more distinct results (adding information, changing scales, exploring several variables, etc.).

Output from *DepositionGeant4* gives a histogram of deposited charges per event, in terms of deposited charge (in ke) and number of events. *GenericPropagation* produces histograms of charge carriers' drift time and their group sizes. *SimpleTransfer* gives a histogram that shows the charge carrier arrival time. *DefaultDigitizer* provides histograms of pixel charges, along with electronics and ADC noise, and a threshold; these remain mostly unused, but function as input for the *DetectorHistogrammer* module, as do the remaining modules.

Histograms obtained from *DetectorHistogrammer* are provided in section 5.1.

4.1.5 Simulations

Several simulations have been run in Allpix^2 to obtain the results for this thesis. The number of layers and incident particle energies were varied. The number of simulated layers is 1, 4, and 8. Simulations of one layer were also done with tungsten absorbers placed in front of the sensors, of 20 and 28 mm; this was done to be able to compare existing experimental data with the simulation

[13]A modular scientific software toolkit developed by CERN to aid in data processing and analysis. ROOT's documentation can be found in [30].

data made in this thesis. The simulation generates one particle from the source per event. A threshold of $1100e$ was set for each pixel.

4.2 SimpDiff simulations

The second part of simulations for this thesis have been done in the SimpDiff method, which is based upon GEANT4 geometry configured by J. Baas [31]. His thesis involved creating all of the GEANT4 information, including the stepping action, the run action, and the geometry. This code was rewritten for this thesis to accommodate a pixel grid overlay and including charge diffusion for GEANT4 data.

This section outlines the general setup of the simulation environment, by providing information on the detector construction, simulation steps, and some details about how data is analysed and interpreted.

4.2.1 Geometry

The detector geometry was constructed within the SimpDiff method. Some predefined materials were used; the adhesive material was added as a compound of carbon, hydrogen and oxygen atoms. The tungsten, kapton, and aluminum are predefined in the method's configuration.

The geometry used in the SimpDiff method also follows the dimensions and specifics mentioned for the chip and absorber specified in section 3.1.1. The main difference between the two methods is the addition of the adhesive material that fixes the components together. The adhesive was added as a compound of carbon, hydrogen, and oxygen atoms. Once again, the internal structure of the electronics (the chip cable and readout peripherals on the edges of the chips) was ignored in the construction as it is expected to exert only a small effect.

The SimpDiff geometry has a variable layer number which allows the user to stack mTower layers on top of each other; in this way, simulations of a single layer can easily be extended to the desired amount of layers.

4.2.2 Particle beam

Similar to section 4.1, GEANT4's general particle source was used to simulate the particle beam for this thesis' simulations. The particle source was placed 17 meters in front of the detector, along the beam axis (z-axis). An example of how GEANT4 visualises can be seen in figure 4.2.

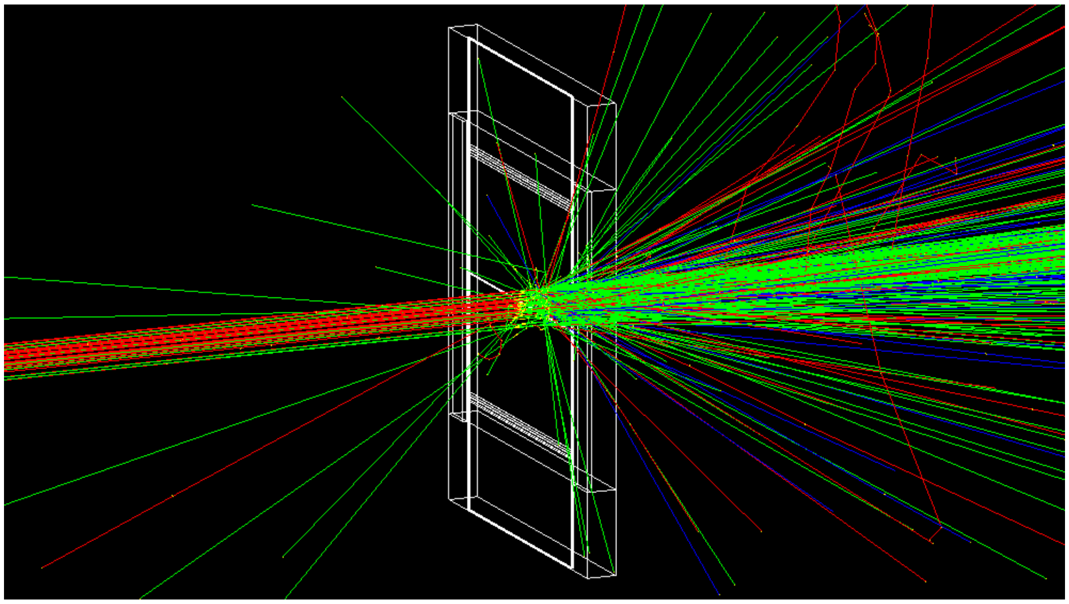


Figure 4.2: A test beam of 100 GeV electrons fired at a single layer of the mTower stack in the GEANT4 framework, with a 1 mm Gaussian spread of incident angles. This picture is from a simulation of 100 events.

4.2.3 Data analysis

The SimpDiff simulations produce a ROOT-file with a tree hierarchy. The branches of this tree (which are of interest) contain the event number, position, event size, and cluster size for each timestep in the simulation. A script written in C++ was used to analyse the simulation data in 'post-production'; a pixel grid was laid over the layers, since the simulation data provides the absolute positions in x, y and z instead of the pixel numbers that are of interest. This analysis provided the pixel- and layer numbers, and accumulated the deposited energy in every pixel on the grid. This is an analog approach to the deposited energy, as opposed to the digital (on/off) measure for pixel hits in the actual detector. Additional code was added to the C++-script to incorporate the process of charge diffusion in the SimpDiff simulations by using analog energy deposition to determine whether charge sharing occurs and translating the charge into digital data; isotropic diffusion was implemented by means of a two-dimensional Gaussian whose mean is the absolute impact position and whose spread and height depend on the total accumulated energy on that position. The Gaussian distribution is generated once per event per pixel, after which a threshold is applied; the remaining number of pixels underneath the Gaussian are noted as 'hit'. In reality, when two of these Gaussians overlap, their tails could have a combined energy that is higher than the threshold for the pixels that would not be triggered by the singular Gaussians, and could be added to the cluster as well. This has not been implemented in this method, as the Gaussians are drawn on a hit-by-hit basis. A clustering algorithm was used to determine the cluster sizes and number of hits per event. All the above data can be compared to the data provided by the Allpix^2 framework, whose methodology is described in section 4.1.

4.2.4 Simulations

Numerous simulations have been run in SimpDiff to obtain the results for this thesis. In these simulations, the number of layers and incident particle energies were varied. The number of layers simulated were 1, 4, and 8. The energies used in the simulations were 50, 100, and 150

GeV. Simulations were performed with 10000 events, to correspond to the simulations produced performed in the Allpix^2 framework.

5 Results and discussion

This section shows results for the proposed methodology of section 4; the results are therefore presented alongside one another, containing both the Allpix^2 simulation results as well as the SimpDiff simulation results.

Results were generated with Allpix^2 with the idea to compare simulation run times, efficiency, and user-friendliness to that of SimpDiff, and allow one to see which method is better suited for the mTower simulation data generation. This section shows results for the simulations done with 1, 4, and 8 layers.

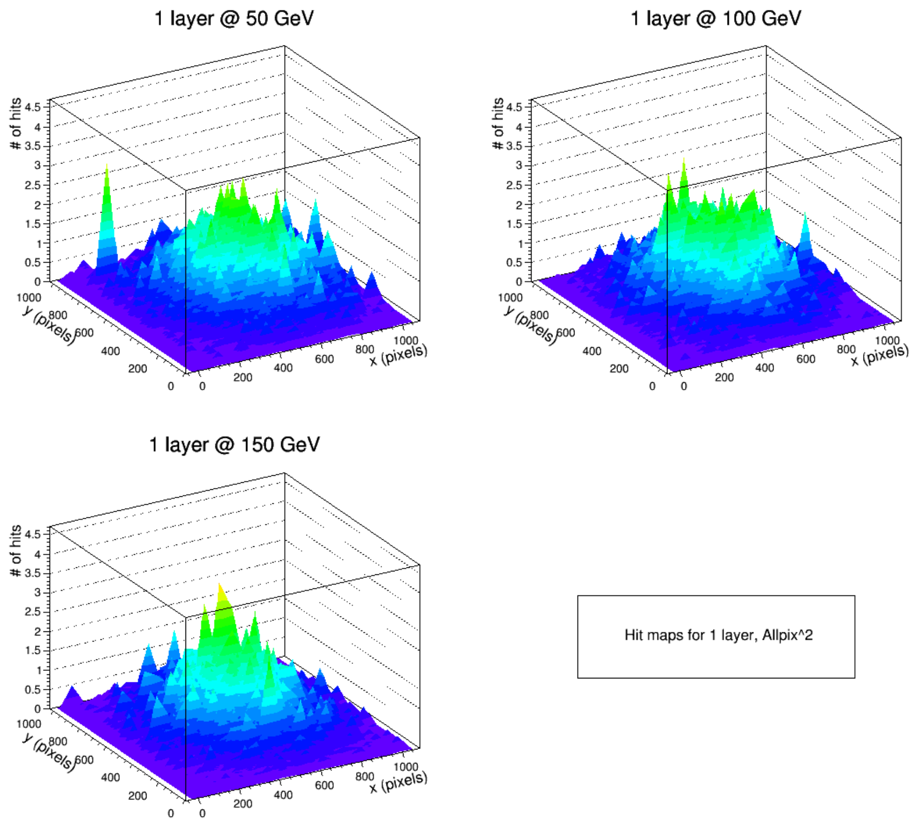
5.1 Simulation results

The Allpix^2 and SimpDiff simulations cover 3 different output results:

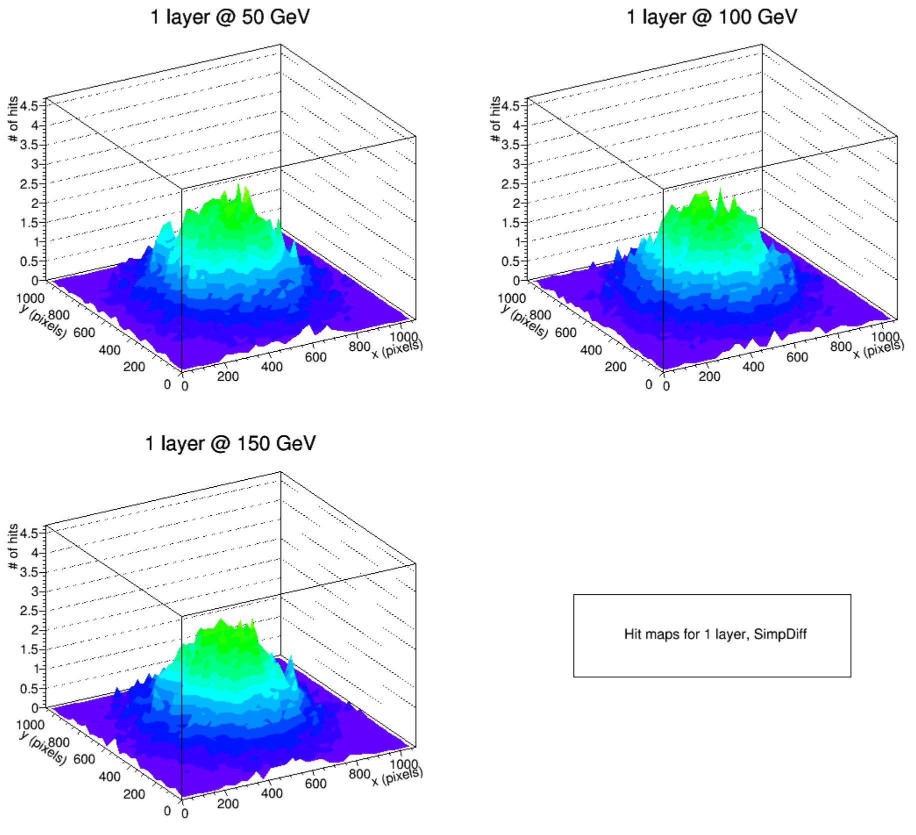
- Hit maps, which show the density of impact positions on the pixel grid,
- Event sizes, which show the distribution of the number of hits per generated event,
- Cluster sizes, which show the distribution of the cluster size as charge propagates on the pixel grid.

Histograms Results for Allpix^2 were obtained from the ROOT-file generated by Allpix^2; scripts were written in ROOT to restructure the data and make it more clear and concise. Results for SimpDiff were obtained through an analysis file written in C++/ROOT, which in turn received data from a GEANT4 output file. For simulation specifics, please refer to section 4. Data was extracted only for the device under test (DUT); in all configurations, this was the *last* layer. Each event in the simulation corresponds to one particle being emitted from the particle source.

The histograms for both simulation results are presented alongside one another. Explanations for the histograms are given after all three layer configurations have been shown.

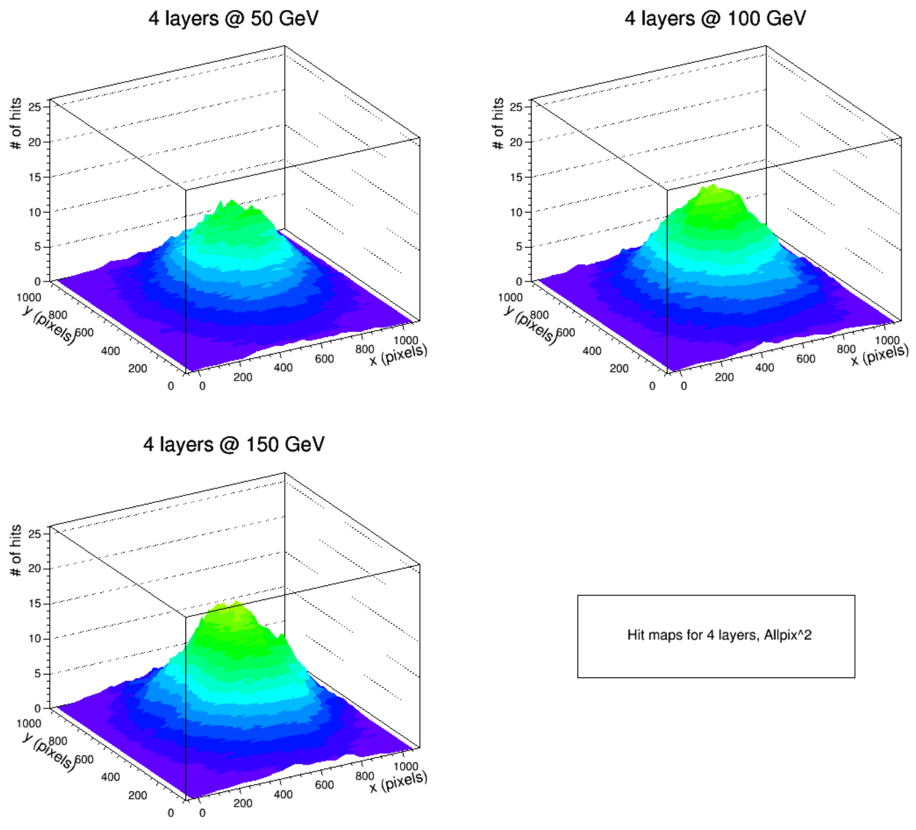


(a)

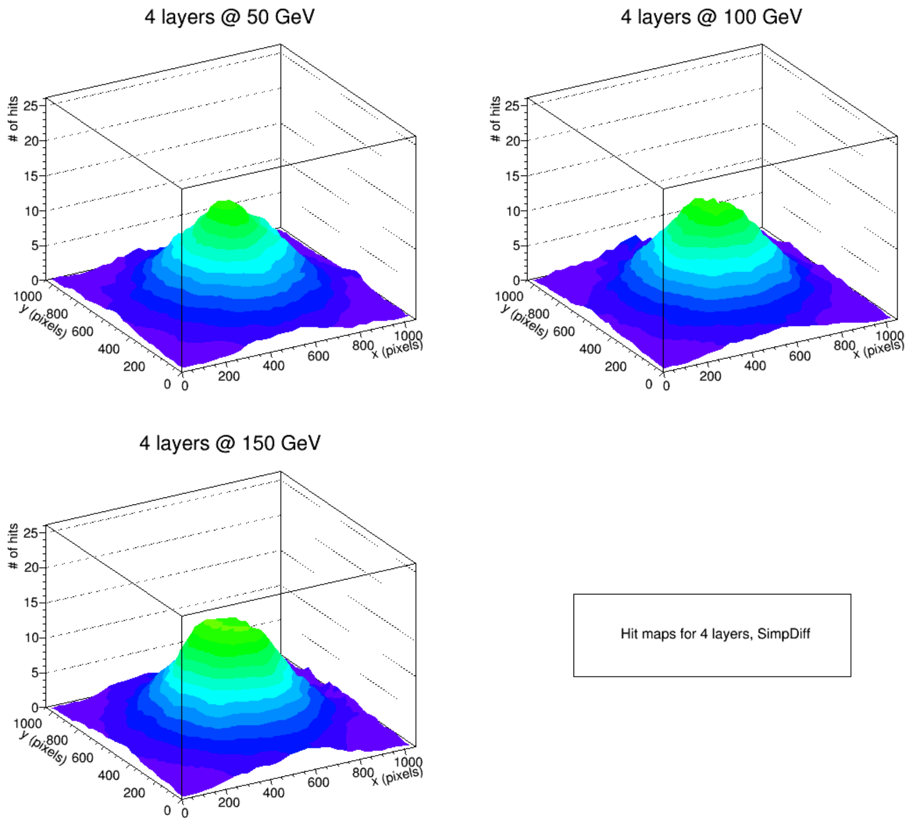


(b)

Figure 5.1: Hit maps for 1 layer of the mTower stack for 50, 100, and 150 GeV electrons. *a*: Allpix^2. *b*: SimpDiff.

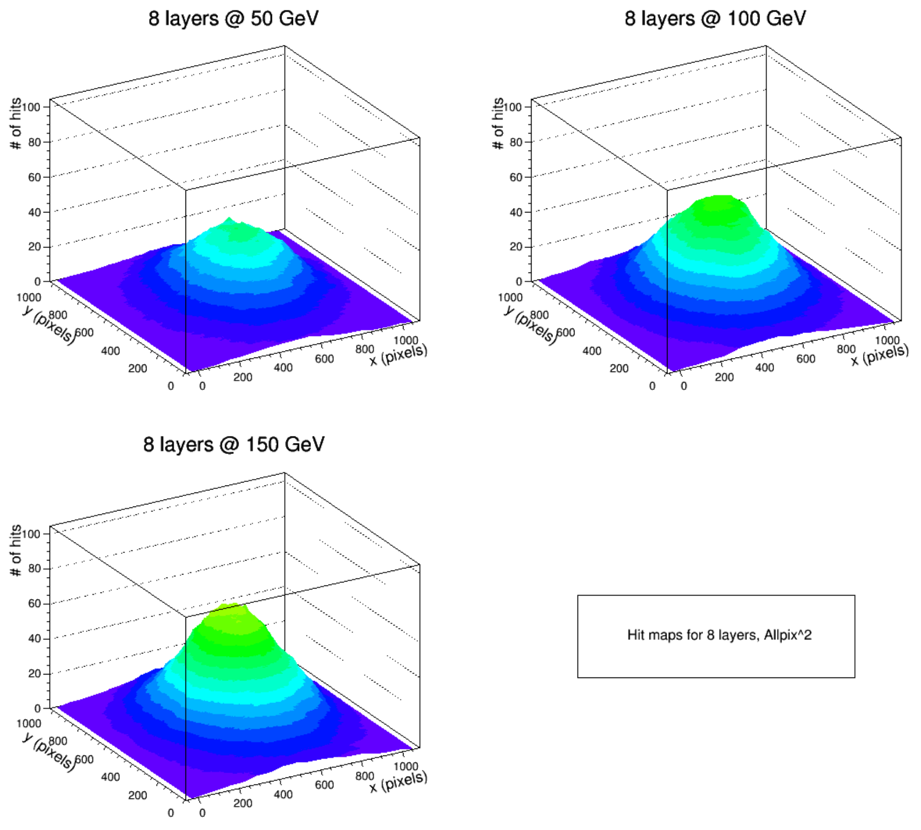


(a)

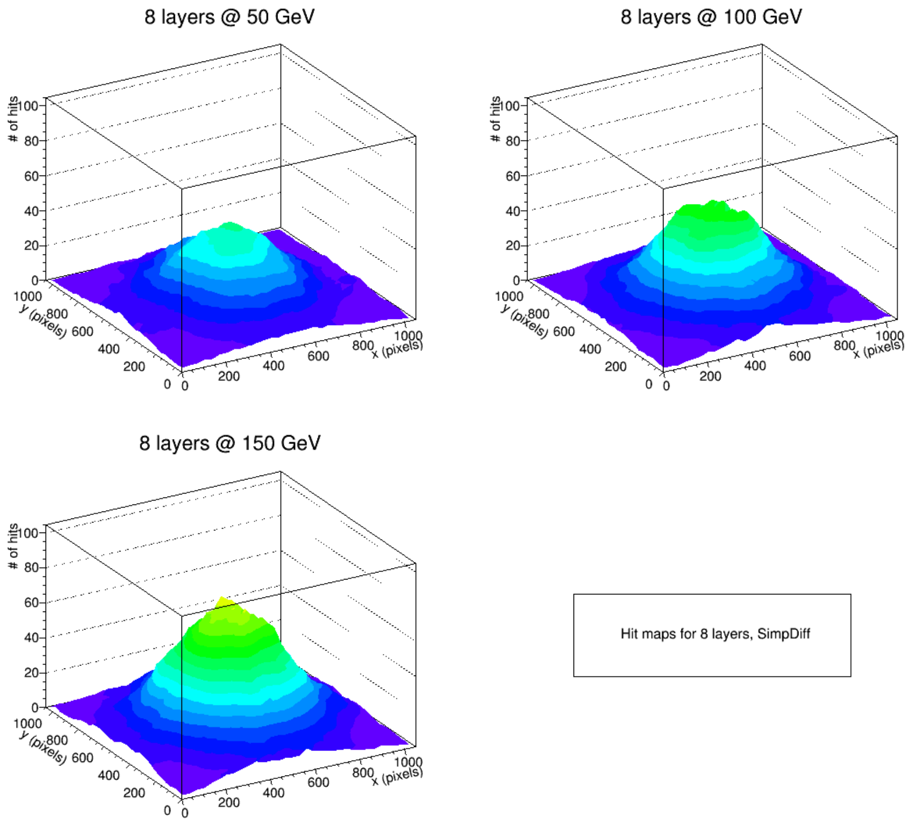


(b)

Figure 5.2: Hit maps for the 4th layer of the mTower stack for 50, 100, and 150 GeV electrons. *a*: Allpix^2. *b*: SimpDiff.



(a)



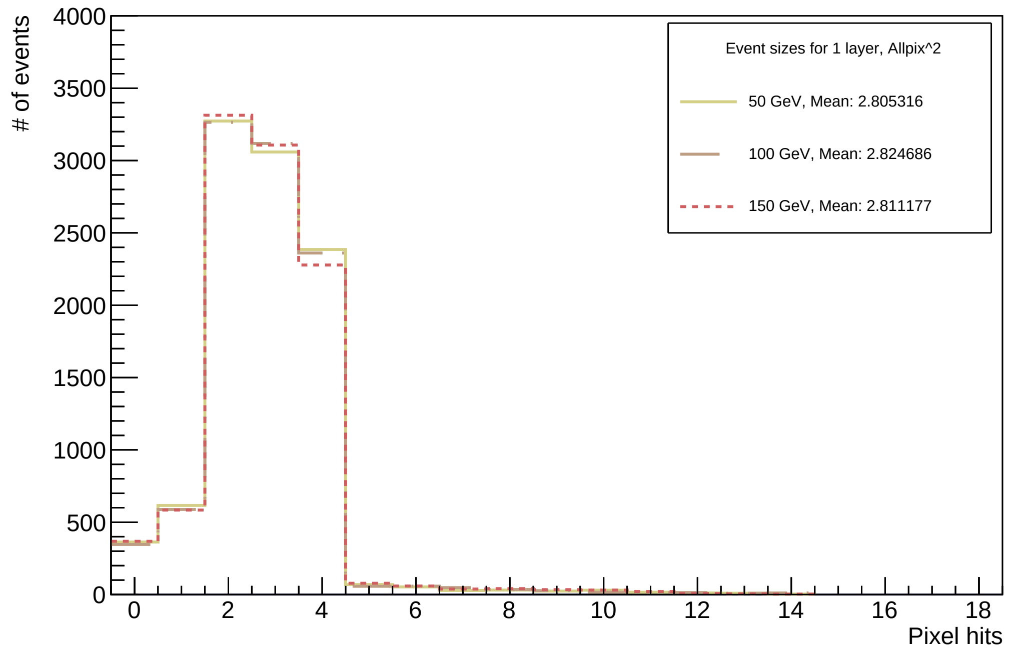
(b)

Figure 5.3: Hit maps for 8th layer of the mTower stack for 50, 100, and 150 GeV electrons. *a*: Allpix^2. *b*: SimpDiff.

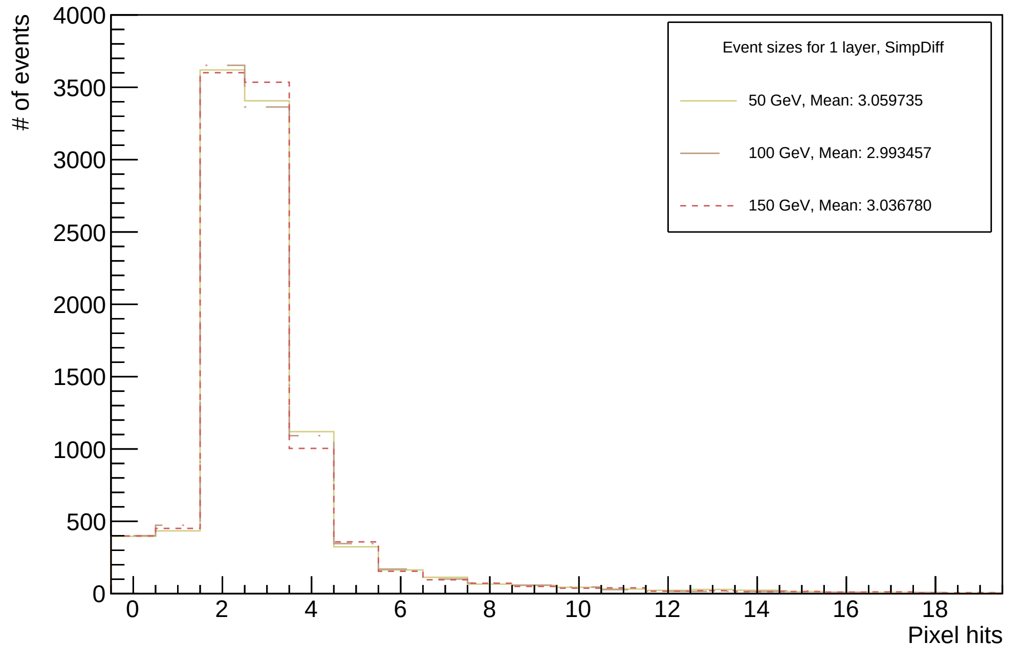
Figures 5.1a, 5.1b, 5.2a, 5.2b, 5.3a, and 5.3b show the distribution of hits as they pass through the sensitive volume of the detector for 1, 4, and 8 layers, respectively.

These figures show a clear Gaussian distribution of impact points across the sensitive volume. The histograms for all configurations have been rebinned in order to make them more readable. Interesting to note is the large spike in the 50 GeV histogram in figure 5.1a near coordinates (100, 1000); no explanation was found for this 'hot spot' in the sensitive volume. Integrating the number of pixel hits in the 1 layer simulations gives $\sim 3 \times 10^4$ hits; as would be expected when compared to the cluster sizes for 1 layer in 5.7a and 5.7b.

There is no sharp peak in the centre of the hit maps of one layer as the Gaussian spread from the particle source is quite large, covering about half the chip width. As such, the FWHM for this distribution is much wider than it would be for no particle spread. The simulation data can be compared to test beam data only to a certain degree, because the beam spread and particle source distance are an approximation that was made based on test beam data and has not yet been fully confirmed.

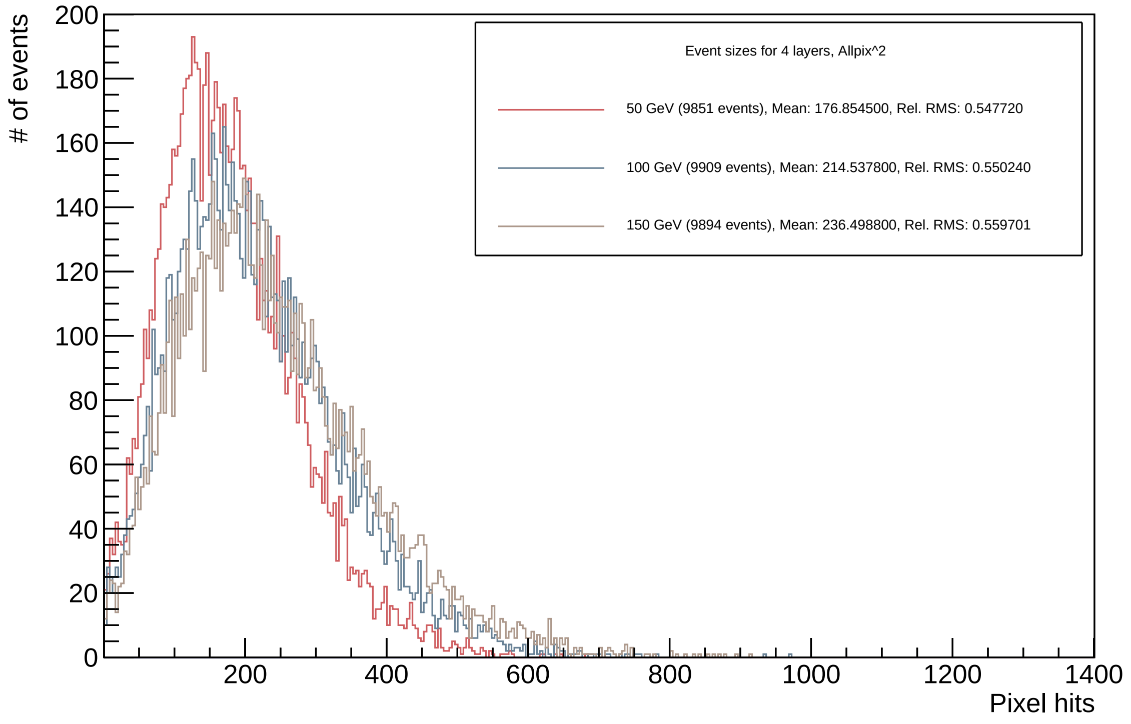


(a)

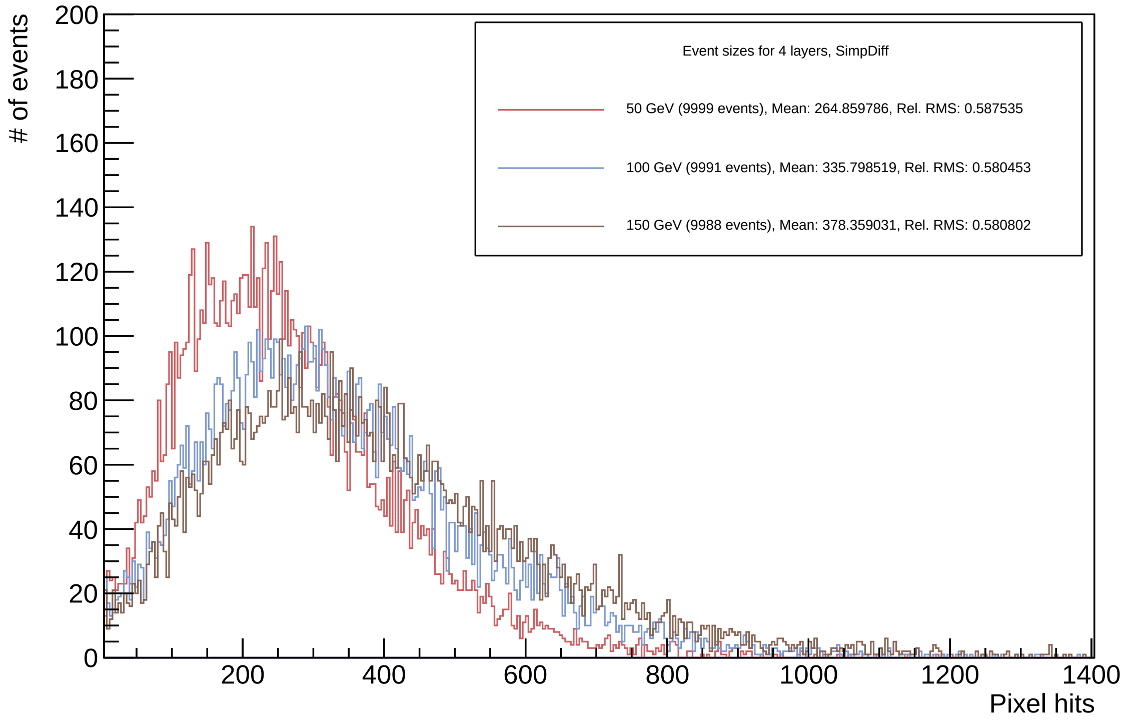


(b)

Figure 5.4: Number of pixel hits per event for 1 layer of the mTower stack for 50, 100, and 150 GeV electrons. *a*: Allpix^2. *b*: SimpDiff.

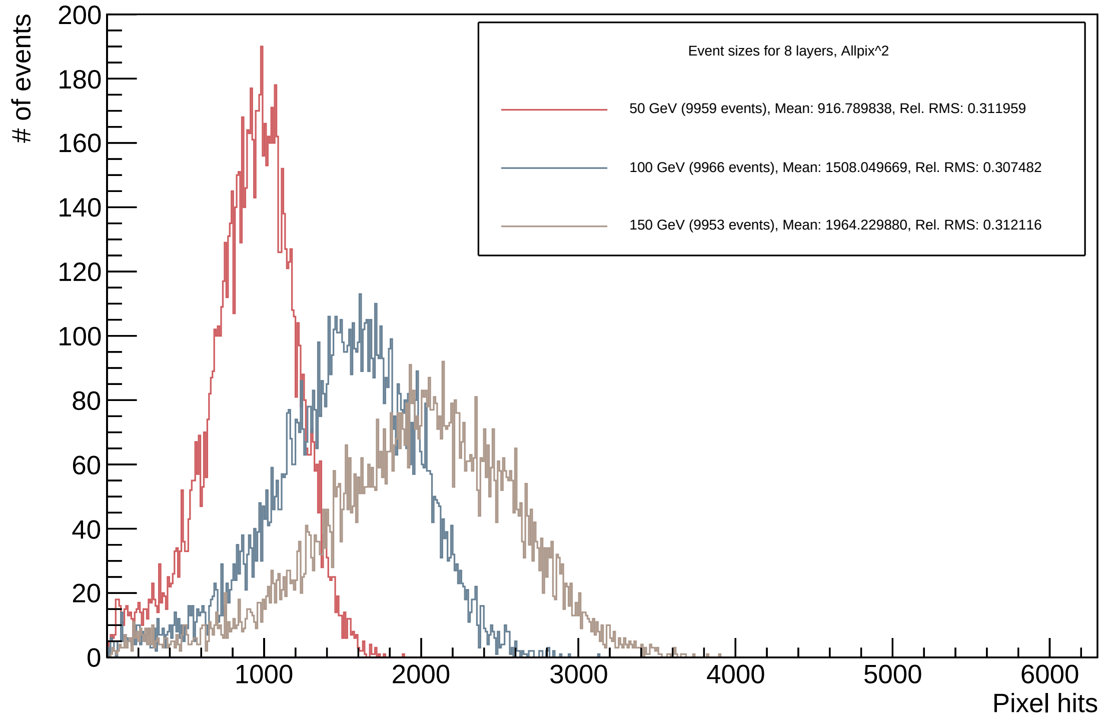


(a)

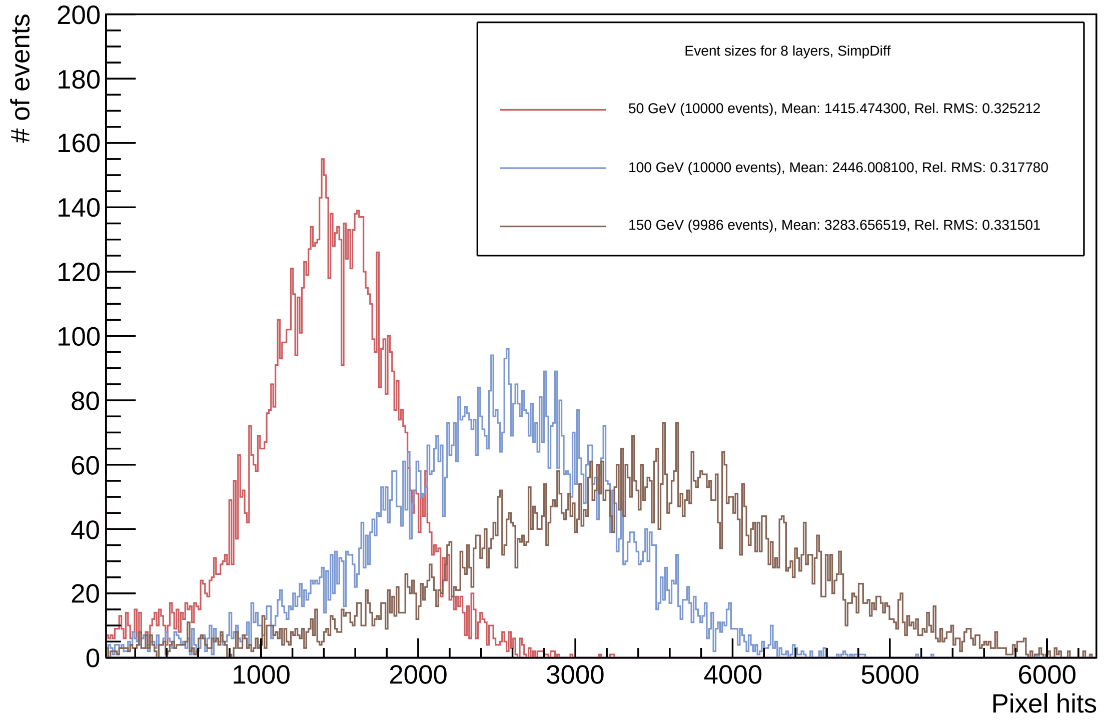


(b)

Figure 5.5: Number of pixel hits per event for the 4th layer of the mTower stack for 50, 100, and 150 GeV electrons. *a*: Allpix^2. *b*: SimpDiff.



(a)



(b)

Figure 5.6: Number of pixel hits per event for the 8th layer of the mTower stack for 50, 100, and 150 GeV electrons. *a*: Allpix^2. *b*: SimpDiff.

Figures 5.4a, 5.4b, 5.5a, 5.5b, 5.6a, and 5.6b show the distributions of the number of hits per event for 1, 4, and 8 layers, respectively. These distributions indicate the number of pixels hit per simulation event; that is, the number of pixels hit by incoming particles that were above the threshold of $1100e$. The numbers beside the particle energies in the legend indicate the integral under the distribution, the mean value, and the relative RMS (mean/RMS) values, respectively.

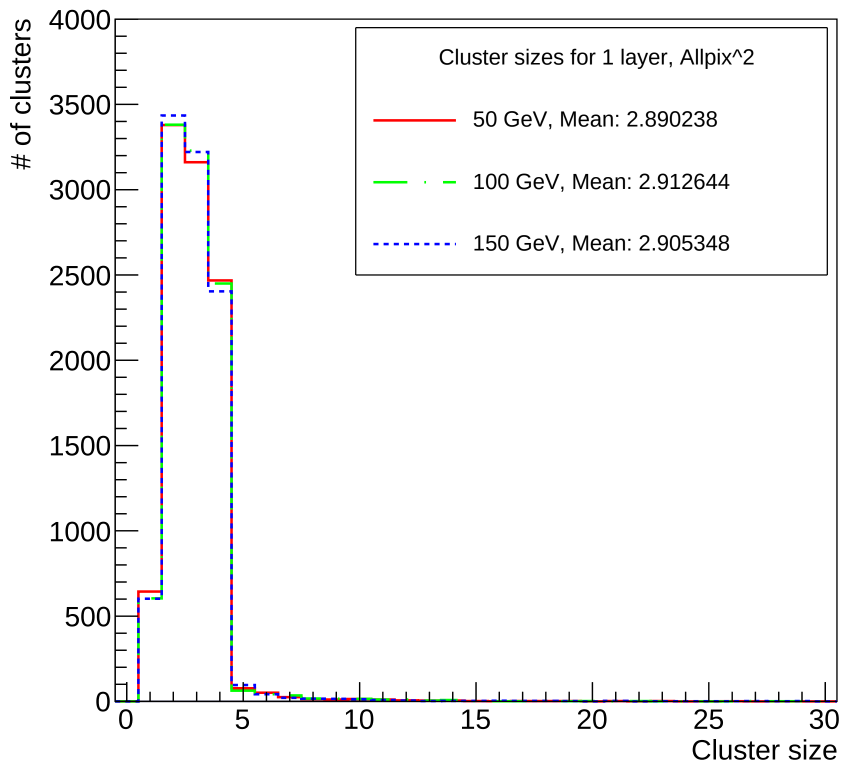
The event sizes for one layer are roughly the same for all three incident particle energies. The one layer configuration has no absorber material in front of it, meaning that the incoming particles have no opportunity to start a cascade. The higher energy particles produce slightly longer tails, though this is hard to tell from figures 5.4a and 5.4b because of bin sizes.

For 4 layers, the charged particles pass through ~ 3 radiation lengths X_0. For all three energies, this is still part of the shower core, meaning that the distributions of event sizes are quite similar. The 150 GeV particles produce slightly longer tails because they shower at a later point, which leads to a slightly flatter distribution, as more particles are produced which undergo cascade processes.

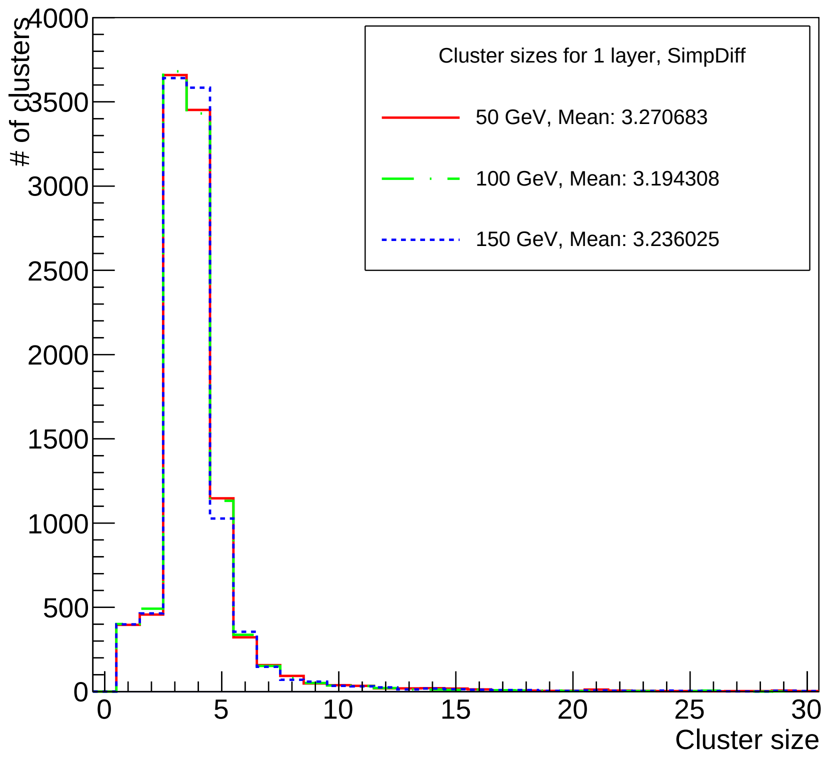
For 8 layers, the three different energies clearly display a difference in event size distributions. As the particles pass through $\sim 7\ X_0$, the 50 GeV electrons will have already reached (close to) the shower maximum, where pair production no longer occurs; this leads to a short tail. The higher energy particles are still undergoing pair production at this point, leading to longer tails and wider peaks.

Figures 5.5a, 5.5b, 5.6a, and 5.6b show the number of events as the integral under each of the histograms. Strangely, the histograms for Allpix^2 show a lower number of events than the SimpDiff histograms, even though the full range of data was plotted. No explanation was found for this.

It should be noted that these histograms have been rebinned as a slight sacrifice to accuracy to make the data more readable. The raw data shows the same distributions, though with larger fluctuations.

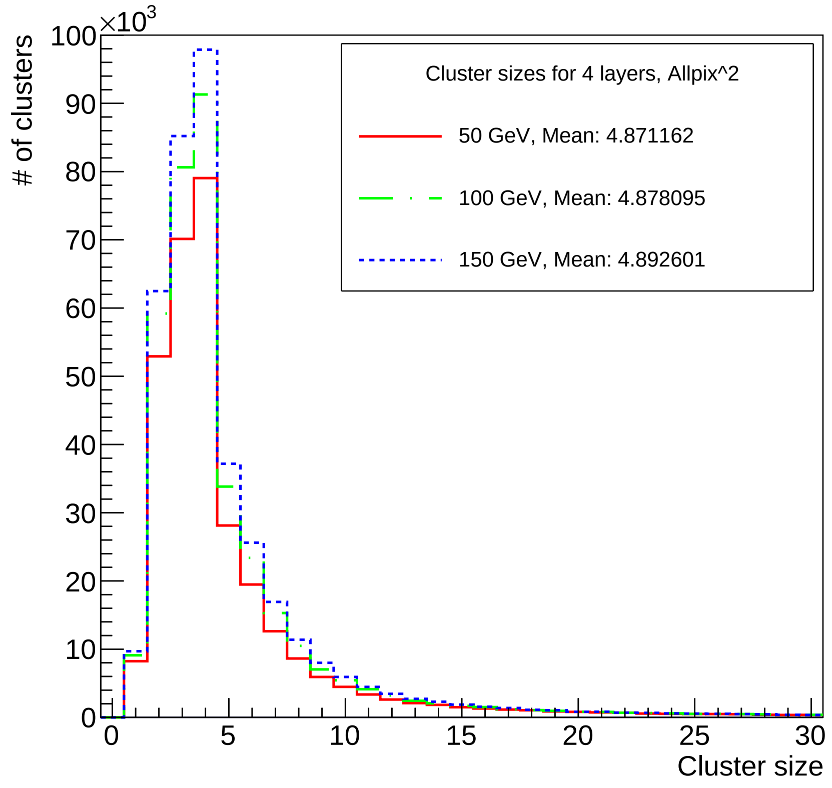


(a)

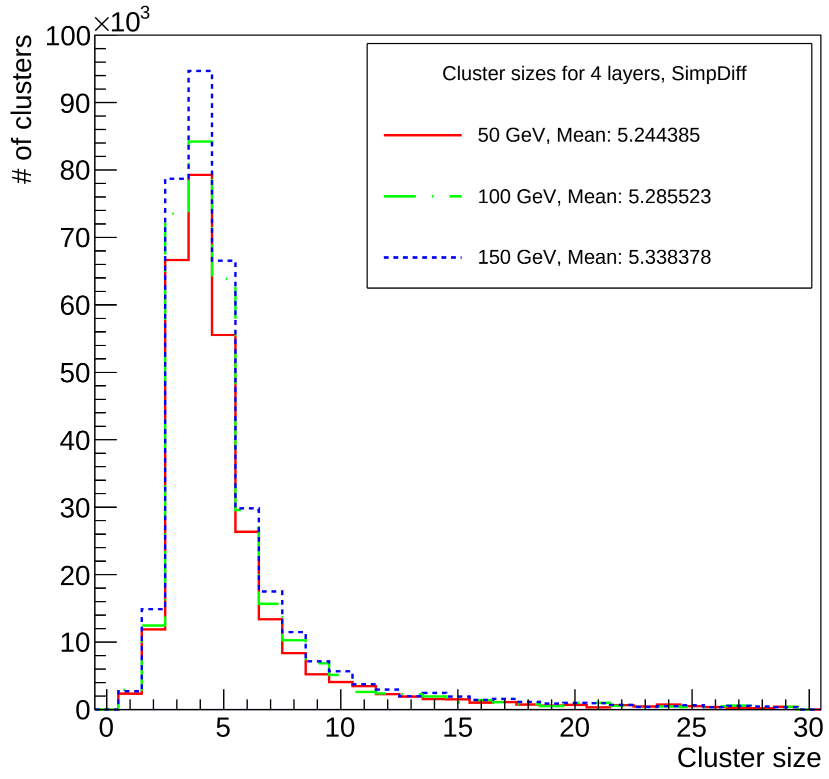


(b)

Figure 5.7: Cluster sizes for 1 layer of the mTower stack for 50, 100, and 150 GeV electrons. *a*: Allpix^2. *b*: SimpDiff.

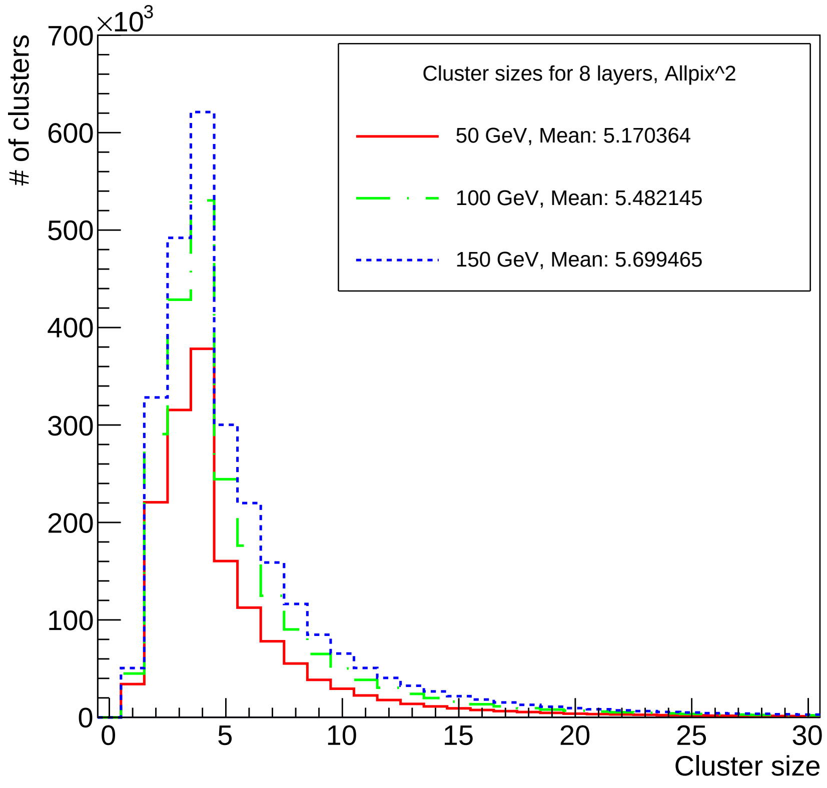


(a)

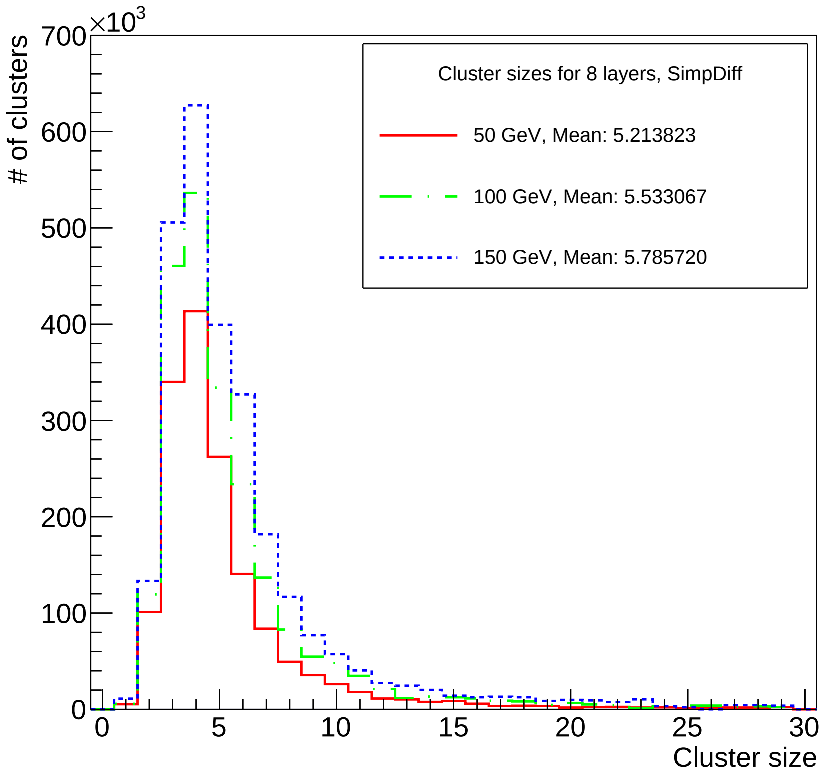


(b)

Figure 5.8: Cluster sizes for the 4th layer of the mTower stack for 50, 100, and 150 GeV electrons. *a*: Allpix^2. *b*: SimpDiff.



(a)



(b)

Figure 5.9: Cluster sizes for the 8th layer of the mTower stack for 50, 100, and 150 GeV electrons. *a*: Allpix^2. *b*: SimpDiff.

Figures 5.7a, 5.7b, 5.8a, 5.8b, 5.9b and 5.9b show the distributions of cluster sizes on the pixel grid for 1, 4, and 8 layers, respectively. These distributions indicate the average size (in pixels) of the clusters as they are produced through charge sharing on the pixel diodes. A cluster is defined as a set of contiguous hits and the size is defined as the number of hits that belong to that cluster. The numbers beside the particle energies in the legend indicate the mean value of the distribution.

Figures 5.7a and 5.7b are clearly very similar to figures 5.4a and 5.4b, as the quantities represented in these histograms are correlated. For 1 layer, it is obvious that the cluster size is equal to the number of hits. This has two reasons: as the particles undergo no cascade processes, any hit that triggers multiple pixels is automatically registered as a cluster, and the quantities are therefore nearly identical (only difference being that there can be an event with no hits, whereas a cluster is only defined for ≥ 1 pixel hits); no noise was simulated for either method, so no clustering was done because of interference. The tails for the 1 layer configuration for Allpix^2 are notably short; this likely has to do with how Allpix^2 determines the impact positions of pixels. Whenever a pixel is hit, it has a predefined position on the pixel at which the impact is placed, after which a Gaussian is drawn over this position. If this position were to be in the centre, having a Gaussian that covers a circle with radius r is much more likely to hit 5 pixels instead of 4; however, if this position is near the corners of the pixel, these small circles will have a much higher probability to only hit 4 pixels. This would explain the sharp drop after a 4 pixel cluster size.

For 4 and 8 layers, the cluster sizes are also similar to the number of hits per event, albeit normalised. For higher energy particles, there are more clusters of the same size, since more particles are produced by cascade processes. Interesting to note is that the cluster size most probable value (MPV) slightly increases when absorbers are introduced into the detector configuration; this can be explained by a higher probability of clusters and/or hits overlapping due to the higher number of particles present.

Simulation run times As mentioned at the beginning of this section, simulation run times were kept track of in order to compare the efficiency (in terms of time and resources necessary) of the Allpix^2 framework with that of the SimpDiff method. All simulations were run on the Quark server cluster hosted by the Subatomic Physics department at Utrecht University. The cluster has 48 available cores, 6 of which were used for the simulations. The absolute speed of the cores is unknown, but the only relevant information is the relative time and resources employed in comparison to the other method.

All 15 Allpix^2 simulations were run with the same settings and active modules. Runs were done with 10000 events. The additional modules applied on top of the GEANT4 core were *ElectricFieldReader*, *GenericPropagation*, *SimpleTransfer*, *DefaultDigitizer*, and *DetectorHistogrammer*. The only module that was set to produce output was *DetectorHistogrammer*. The 15 configurations include 5 different detector layouts; 1 layer, 4 layers, 8 layers, and 1 layer with a 20 mm as well as a 28 mm tungsten absorber. All 5 layouts were tested with 3 different incident particle energies of 50, 100, and 150 GeV. Simulation run times for Allpix^2 can be seen in table 1.

All 9 SimpDiff simulations were run with the same macros and number of events. Table 2 below illustrates the elapsed time for the simulations. These times consist of two separate actions. The first one is generating the data; the second one is the analysis of the raw data, which involves overlaying a pixel grid, a clustering algorithm, and position tracker. The times in the table have been added together. For each simulation, about 30% of the total time used was in SimpDiff's data generation; the remaining time was needed to generate the analysed data file. Simulation run times for SimpDiff can be seen in table 2.

Allpix^2 simulation run times			
Energy Configuration	50 GeV	100 GeV	150 GeV
1 layer	7.1 minutes	7.2 minutes	7.2 minutes
4 layers	41.4 minutes	56.3 minutes	1 hour 8.2 minutes
1 l + 20 mm W	4 hours 2.1 minutes	5 hours 12.3 minutes	6 hours 29.6 minutes
1 l + 28 mm W	6 hours 45.7 minutes	8 hours 23.5 minutes	10 hours 7.0 minutes
8 layers	9 hours 46.3 minutes	11 hours 23.8 minutes	13 hours 2.1 minutes

Table 1: Simulation run times for each individual Allpix^2 configuration.

SimpDiff simulation run times			
Energy Configuration	50 GeV	100 GeV	150 GeV
1 layer	4.3 minutes	4.3 minutes	4.4 minutes
4 layers	28.3 minutes	33.4 minutes	37.9 minutes
8 layers	6 hours 42.3 minutes	8 hours 51.8 minutes	10 hours 41.5 minutes

Table 2: Simulation run times for each individual SimpDiff configuration.

The total elapsed time for the collection of the results in section 5.1 is **37 hours and 19.6 minutes** for Allpix^2, and **28 hours and 8 minutes** for SimpDiff. From tables 1 and 2 it is clear that the elapsed time does not scale linearly with the amount of layers of absorber material present in from of the DUTs.

Additional notes The C++ analysis file contains about 300 lines of code which were written with the help of two C++ classes made by Dr. N. van der Kolk. Designing this analysis file took a good portion of two weeks, down to managing memory leaks and segmentation faults for the incredibly large data sets. The discussion will take this into account when examining the viability of one simulation method over the other.

5.2 Comparison

Data quality The hit maps for both methods look extremely similar; aside from statistical fluctuations that arise naturally from a (relatively) small sample size of 10000 events, the figures represent a similar Gaussian beam spread across all layer configurations. All hit maps have been rebinned to 32 bins and rescaled accordingly.

The mean, RMS, and relative RMS values for the event sizes are given in table 3; the left values in each cell corresponds to values for Allpix^2, the right values correspond to SimpDiff. For 1 layer, only the mean is shown.

Mean, RMS, and relative RMS for both methods			
Energy / Layer	50 GeV	100 GeV	150 GeV
1 layer:			
Mean	2.81 / 3.06	2.82 / 2.99	2.81 / 3.04
4th layer:			
Mean	176.85 / 264.86	214.54 / 335.80	236.50 / 378.36
RMS	322.89 / 450.80	389.90 / 578.51	422.54 / 651.44
Rel. RMS	0.55 / 0.59	0.55 / 0.58	0.56 / 0.58
8th layer:			
Mean	916.79 / 1415.47	1508.05 / 2446.01	1964.23 / 3283.66
RMS	2938.81 / 4352.46	4904.51 / 7697.17	6293.27 / 9905.42
Rel. RMS	0.31 / 0.33	0.31 / 0.32	0.31 / 0.33

Table 3: The mean, RMS, and relative RMS values of the event sizes for both methods. Left values in each cell correspond to Allpix^2, right values to SimpDiff.

The histograms which contain the number of hits per event look similar for 1 layer; although there seem to be less events in the SimpDiff simulations which contain only one hit, though this is compensated in the higher number of events which had two or three pixel hits. Interesting to note is that in all plots for the event sizes, SimpDiff seems to produce longer tails and flatter peaks than Allpix^2; this leads to the conclusion that the variance (width) of the Gaussian charge sharing was determined in different ways in both methods. Particularly, the 2D Gaussian produced by an energy deposition on the a pixel has a slightly sharper peak and shorter tail in Allpix^2 compared to SimpDiff. This leads to more pixel hits in the SimpDiff simulations, though with less events corresponding to these pixel hits. Another explanation could be (also mentioned in section 5.1) that Allpix^2 uses a predefined 'hit position' on a pixel; when a pixel is hit, it always registers the charge deposited at a certain point in the pixel, whereas SimpDiff gives an absolute position, which could be anywhere within a pixel. This leads to a decreased likelihood of finding a cluster size of > 4 within a radius r around the impact position if it is in a corner, whereas a centered impact position would lead to a cluster size of 5 or more with the same probability. Having a larger cluster size also increases the probability of clusters overlapping, even in a 1 layer configuration.

From the values in table 3, it can be concluded that the histograms for both methods follow similar behaviour; namely, the relative RMS is very similar. This means that though the mean is shifted from one method to the other, the shape of both distributions follows the same trend, implying that these methods use a similar approach and can be compared.

The mean values for the cluster sizes can be seen in table 4. Once again, the left values in each cell correspond to Allpix^2, the right ones to SimpDiff.

Mean, RMS, and relative RMS for both methods			
Energy Layer	50 GeV	100 GeV	150 GeV
1 layer: Mean	2.89 / 3.27	2.91 / 3.19	2.91 / 3.24
4th layer: Mean	4.87 / 5.24	4.88 / 5.29	4.89 / 5.34
8th layer: Mean	5.17 / 5.21	5.48 / 5.53	5.70 / 5.79

Table 4: The mean values of the cluster sizes for both methods. Left values in each cell correspond to Allpix^2, right values to SimpDiff.

As mentioned in section 5.1, the cluster sizes and event sizes for 1 layer are correlated. If we compare the mean values of the 1 layer configuration in table 3 and 4, we see that the values for Allpix^2 differ about 0.1 from the event size to the cluster size, while the values for SimpDiff differ about 0.2. Though these histograms are correlated, their values need not be identical; in fact, the cluster size is only counted for events where the number of hits was ≥ 1. As such, the mean value in the case of event sizes is lower, because the events with 0 hits are also taken into account. Another thing to note is that the cluster sizes for Allpix^2 are slightly smaller than those for SimpDiff, in every configuration; this supports the flatter Gaussian curves in the case of SimpDiff.

The cluster sizes for both simulations appear to be very similar for both methods in all layer configurations. Interestingly enough, the SimpDiff simulations of 1 layer appear to follow a Landau distribution moreso than the Allpix^2 simulations. Figures 5.7a and 5.7b have clear differences. Allpix^2 generated many more clusters of size 2, and dropped sharply after cluster size of 4. This can be explained by the impact position definition used in Allpix^2 (as was detailed in the paragraph above), which would lead to many clusters of 2 being produced even with very small energy depositions. The impact position definition also explains why clusters of size ≥ 5 were found in the SimpDiff simulations. Figures 5.8b and 5.9b show an ever-so-slightly lower MPV than figures 5.8a and 5.9a; this is likely due to the stochastic nature of the simulations. In the end, the number of clusters is approximately equal. The latter plots appear to follow theory very closely, as the clusters start overlapping at higher numbers of layers, which drives the Landau distribution to have a longer tail. The same principle of the sharp drop in figure 5.7a compared to 5.7b can be explained by the difference in impact position definition between both methods.

In order to examine the behaviour of the event sizes at different energies, the two quantities were plotted against each other. The points in the plot represent the mean values found in table 3; the plot can be seen in figure 5.10. From this plot, it is clear that although the number of hits does not scale linearly with the energy (as is expected when only viewing one layer of the configuration), all of the configurations follow the same trend in the graph; the event sizes increase logarithmically with increasing energies. The reason for this can be seen in equation 11, which states that $t_{max} \propto \ln(E_0/E_c)$; a higher initial energy will increase the maximum amount of radiation lengths the particles can pass through logarithmically. Since only the last layer is inspected, it is clear that higher incident particle energy leads to a higher number of particles reaching the last sensitive layer.

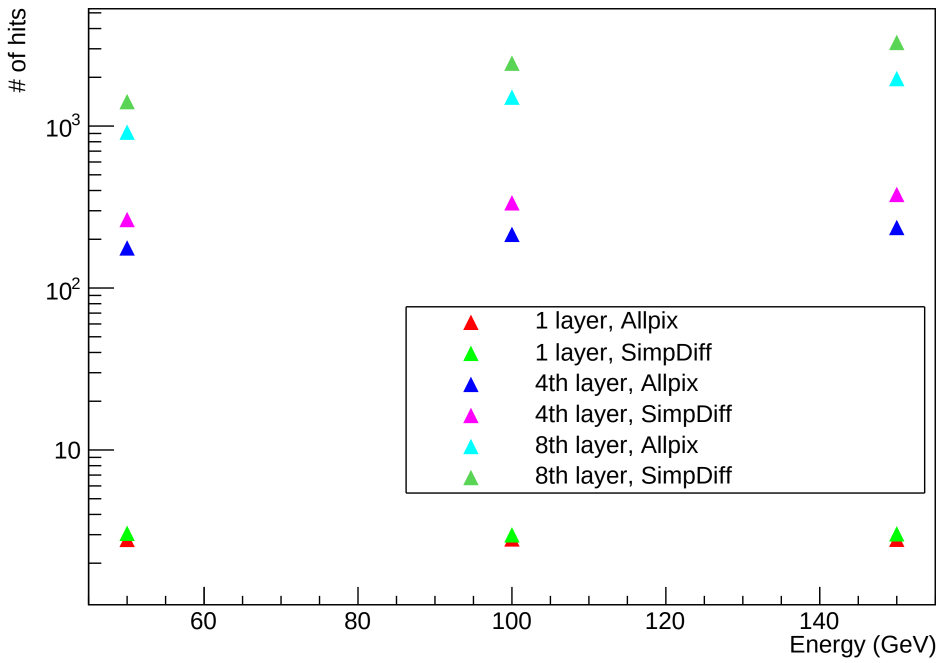


Figure 5.10: The mean number of pixel hits as a function of the energy for 50, 100, and 150 GeV electrons. Shows both methods.

Efficiency The efficiency of both methods can be compared in terms of simulation run times and additional time required to write analysis code. As mentioned in section 5.1, the simulation run times were **37 hours 19.6 minutes** for Allpix^2, and **28 hours and 8 minutes** for SimpDiff; this signals an improvement in elapsed time of **24.6%** for the SimpDiff procedure over that of Allpix^2.

Extending the SimpDiff script takes significantly less time than starting from scratch, and can easily be done; as such, the SimpDiff procedure ends up saving a tremendous amount of time in cases of 12 to 24 layer simulations.

On the other hand, Allpix^2 offers a lot more functionality in terms of results that can be produced; e.g. TCAD mesh implementation, advanced propagation and charge transfer algorithms, and the ability to keep track of the Monte-Carlo information on each particle. So, depending on what data is necessary, Allpix^2 offers viable options.

User-friendliness A lot can be said for GEANT4 in terms of UI and customisability; for a legacy simulation system, it has a lot of great interactive elements that can make the simulation process a lot less straining on the user. However, in order to properly perform any sort of full physics simulation, GEANT4 requires the user to build the world by hand, defining all the materials, positions, interactions, etc. Allpix^2 does not suffer from this incredibly tedious process of manually constructing the world piece by piece; the framework allows the user to specify a configuration file in which parameters can be set, such as material, relative or absolute position, rotation, active sensor area, and diode implant sizes. Another thing to note is that in order to change the geometry in GEANT4, the user is required to rebuild the entire executable using CMake, forcing the user to create distinct build directories and paths if they want to compare simulations side by side; this is not the case in Allpix^2, where one can update a configuration file and be on their way. This results in a much smoother experience when building and testing simulations.

A final thing to note is that Allpix^2's modules generate histograms which can provide a lot of information about the simulation at hand at a quick glance; for the same convenience in SimpDiff, the user has to write their own scripts to extract the same information.

6 Conclusion

In this thesis, physics simulations for the newly proposed FoCal prototype with ALPIDE CMOS sensors, the mTower, were developed in order to compare the test beam data to simulation data. Simulations allows for quick parameter adjustments such as detector geometry, material composition, and incident beams. The simulations have been developed in two separate methods: SimpDiff and Allpix^2. Allpix^2 functions as a wrapper around GEANT4 and has a much broader application scope.

Separate analysis files were written for the SimpDiff simulations. These included a clustering algorithm, an absolute position to pixel position conversion, and Gaussian charge diffusion. These analysis files include configurations for 1, 4, and 8 layers; this can be expanded with a little effort to any other amount of layers.

The simulations provided similar results; the hit maps show a clear 2D Gaussian distribution; the event sizes' means were $\sim$ 50% higher for SimpDiff, but the relative RMS (mean/RMS) differs only by 0.04 at a maximum; the cluster sizes' means are slightly higher for SimpDiff ($\sim$ 5%), which is in agreement with a larger event size; the event sizes as a function of the energy follow a logarithmic trend in both methods. All of the above implies that the implementation of charge diffusion and the clustering algorithm in SimpDiff analysis is in agreement with that of Allpix^2; the main difference being that Allpix^2 has a predefined impact position when a pixel is triggered from which the charge sharing occurs, where in SimpDiff, the absolute impact position is used to determine where the diffusion process should start. After running the simulations in both methods, it was found that for 1, 4, and 8 layers of the FoCal prototype, the Allpix^2 simulations took **37 hours and 19.6 minutes**, whereas SimpDiff simulations took **28 hours and 8 minutes**; this signals a time-improvement of **26.4%** of SimpDiff over Allpix^2.

Allpix^2 provides the user with the ability to implement advanced physics phenomena such as capacitive charge transfer, applying TCAD meshes, and automatic signal digitisation; something the user would have to implement on their own if they were to use SimpDiff.

For simple physics simulations where data such as cluster- and event sizes are of interest, SimpDiff is the recommended method, especially when using the code that was used for this thesis; SimpDiff has significantly less server load and is unquestionably faster for the results that were produced in this thesis. If one has little to no knowledge of C++ and the ROOT framework, it might be in their best interest to use the Allpix^2 method even for simple physics simulations, as expanding or morphing the code to fit their needs might be beyond their scope of knowledge. For advanced physics simulations with very specific parameter settings, one should definitely consider using Allpix^2, if simulation times or server load are not relevant.

Since it is unknown how exactly the charge sharing process in the ALPIDE chips occurs, it is necessary to compare both SimpDiff and Allpix^2 to test beam data and see which agrees more with reality. If test beam results appear to correspond better to one method over the other, that method should be used for simulations; either of the methods could also be fine-tuned to fit the test beam data better.

Future work could examine the effects of different charge sharing implementations across more layer configurations, or perhaps devise the SimpDiff method in such a way that tungsten absorbers are added to the detector geometry. The next logical step in improving either simulation is the implementation of noise, which would make the simulations more realistic. One could also implement more complex phenomena into the SimpDiff method to see if the efficiency remains the same compared to Allpix^2, or alter the current diffusion method to correspond better to test data.

References

[1] CERN, *ALICE: A CERN Experiment*, URL https://home.cern/science/experiments/alice.

[2] The ALICE Collaboration, Journal of Instrumentation **3**, S08002 (2008).

[3] CERN, *Schematic view of the ALICE experiment and sub detectors*, URL http://cds.cern.ch/record/2263642.

[4] CERN, *Time lead collisions at LHC*, URL https://home.cern/news/news/accelerators/time-lead-collisions-lhc.

[5] CERN, *The evolution of the early universe*, URL https://home.cern/science/physics/early-universe.

[6] Institute for Particle Physics Phenomenology, *Quark and lepton kinematics*, URL https://www.ippp.dur.ac.uk/~krauss/Lectures/QuarksLeptons/Basics/Kin_2.html.

[7] R. Wigmans, *Calorimetry in High Energy Physics, CERN academic training*, CERN (1989), URL http://cds.cern.ch/record/319296/files/AT00000342.pdf.

[8] B. Surrow, *Calorimetry presentation, MIT*, URL http://physics.bu.edu/neppsr/2004/Talks/Calorimetry-Surrow.pdf.

[9] M. Tanabashi et al. (Particle Data Group), Phys. Rev. **D98**, 030001 (2018).

[10] D. Brooks, *Schematic depiction of an electromagnetic cascade*, URL https://slideplayer.com/slide/9912596/.

[11] Max Planck Institut für Physik, *An accurate EM cascade*, URL https://www.mpp.mpg.de/~menke/elss/pic3.shtml.

[12] CERN, *Calculation of radiation length in materials*, URL https://cds.cern.ch/record/1279627/files/PH-EP-Tech-Note-2010-013.pdf.

[13] W. Frass, *Passage of Particles Through Matter*, Oxford (2009).

[14] H. Abramowicz et al. (FCAL), Eur. Phys. J. **C78**, 135 (2018), 1705.03885.

[15] C. W. Fabjan, *Calorimeters in Particle Physics* (American Cancer Society, 2009), pp. 649–676, ISBN 9783527600434.

[16] The ALICE Collaboration, *Technical Design Report of the Photon Spectrometer*, CERN (1999).

[17] A. Fantoni, Journal of Physics: Conference Series **293**, 012043 (2011).

[18] T. Peitzmann, *In-house FoCal physics presentation*, UU/Nikhef (2019).

[19] Particle Data Group, *Tungsten properties*, URL `http://pdg.lbl.gov/2018/AtomicNuclearProperties/HTML/tungsten_W.html`.

[20] C. Zhang, *Measurements with a High-Granularity Digital Electromagnetic Calorimeter* (2016).

[21] M. Mager, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **824**, 434 (2016).

[22] M. Aquilina, *Bringing Cutting-Edge Monolithic Active Pixel Sensor Technology to Schools: the ALPIDE Arduino Shield*, CERN (CERN-STUDENTS-Note-2017-188).

[23] G.-J. Nooren, *FoCal Motivation*, UU/Nikhef (2011).

[24] G. Aglieri, *The ALPIDE Pixel Sensor Chip for the Upgrade of the ALICE Inner Tracking System*, URL `https://indico.cern.ch/event/391665/contributions/1827407/attachments/1229908/1803453/20160218-VCI2016-Aglieri.pdf`.

[25] GEANT4 Collaboration, *GEANT4 Physics Reference Manual*, CERN (2018), URL `http://geant4-userdoc.web.cern.ch/geant4-userdoc/UsersGuides/PhysicsReferenceManual/fo/PhysicsReferenceManual.pdf`.

[26] GEANT4 Collaboration, *GEANT4 Use Cases - Reference List*, URL `https://geant4.web.cern.ch/node/302`.

[27] A. DellAcqua, *Monte Carlo truth in the ATLAS detector simulation programs*, CERN (2015).

[28] C. Jacoboni, C. Canali, G. Ottaviani, and A. A. Quaranta, Solid-State Electronics **20**, 77 (1977), ISSN 0038-1101, URL `http://www.sciencedirect.com/science/article/pColliderii/0038110177900545`.

[29] E. Fehlberg, Computing (1969).

[30] CERN, *ROOT User's Guide*, URL `https://root.cern.ch/guides/users-guide`.

[31] J. Baas, *Geant4 simulation of the calorimetric application of the ALPIDE CMOS sensor* (2019).

Acknowledgements

I would firstly like to thank my daily supervisor, Naomi van der Kolk. She has helped me tremendously with my questions and problems, and reassured me that I was going to do just fine. She has helped me with contacts, writing code, and general scientific knowledge; I learned a great deal about writing a thesis from her.

I would also like to thank Marco van Leeuwen, who has helped me with technical issues on quite a number of occasions. Without you, the quark cluster would have probably been the equivalent of a dumpster fire.

Lastly, I would like to thank the people around me, most notably a good friend Remco, who also wrote his thesis at SAP, and my girlfriend Margot; you were the people that were here for me when I needed to wind down.

Though there have been stressful times, I have very much enjoyed working in a physics group, where we worked on a project as a team. Everyone shares their ideas and healthy criticisms, which I found to be a necessary factor in doing research. I wish Naomi, Thomas, Gert-Jan, and Hiroki the best in their further research with the FoCal and whatever else time may bring.

A Appendix

A.1 Derivations

This section contains a derivation of the pseudorapidity quantity defined in the theory, starting from the rapidity.

$$\begin{aligned} y &= \frac{1}{2}\ln\frac{E+p_zc}{E-p_zc} \\ &= \frac{1}{2}\ln\frac{\sqrt{p^2c^2+m^2c^4}+p_zc}{\sqrt{p^2c^2+m^2c^4}-p_zc} \\ &= \frac{1}{2}\ln\frac{pc\sqrt{1+m^2c^4/p^2c^2}+p_zc}{pc\sqrt{1+m^2c^4/p^2c^2}-p_zc} \end{aligned} \tag{26}$$

Expanding as $(1+x)^n \approx 1+nx$ gives:

$$\begin{aligned} y &\simeq \frac{1}{2}\ln\frac{pc(1+m^2c^4/2p^2c^2)+p_zc}{pc(1+m^2c^4/2p^2c^2)-p_zc} \\ &\simeq \frac{1}{2}\ln\frac{pc+m^2c^4/2pc)+p_zc+\dots}{pc+m^2c^4/2pc)-p_zc+\dots} \\ &\simeq \frac{1}{2}\ln\frac{1+p_z/p+m^2c^4/2p^2c^2+\dots}{1-p_z/p+m^2c^4/2p^2c^2+\dots} \end{aligned} \tag{27}$$

Following basic trigonometry, one can see that $p_z/p = \cos\theta$:

$$1+\frac{p_z}{p} = 1+\cos\theta = 1+(\cos^2(\theta/2)-\sin^2(\theta/2)) = 2\cos^2(\theta/2) \tag{28}$$

$$1-\frac{p_z}{p} = 1-\cos\theta = 1-(\cos^2(\theta/2)-\sin^2(\theta/2)) = 2\sin^2(\theta/2) \tag{29}$$

$$\begin{aligned} y &\simeq \frac{1}{2}\ln\frac{\cos^2(\theta/2)}{\sin^2(\theta/2)} \\ &\simeq -\ln\tan(\theta/2) \end{aligned} \tag{30}$$

$$\Downarrow$$

$$\eta = -\ln\tan(\theta/2) \tag{31}$$

A.2 Additional histograms

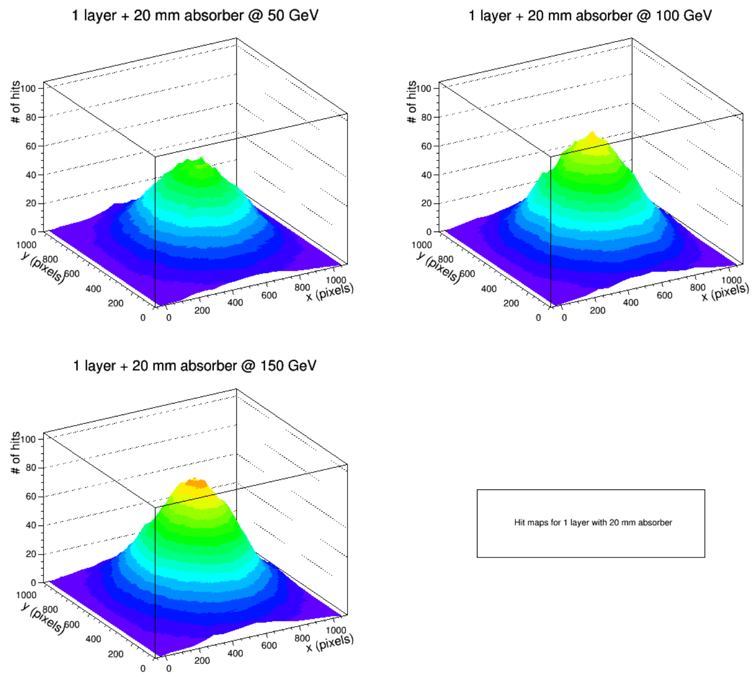


Figure A.1: Hit map for 1 layer of the mTower stack preceded by a 20 mm tungsten absorber in Allpix^2 for 50, 100, and 150 GeV electrons.

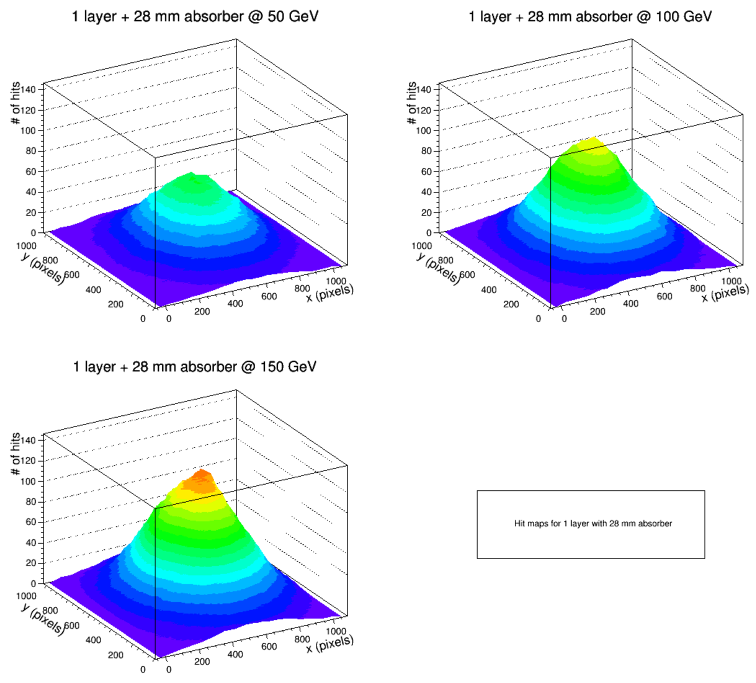


Figure A.2: Hit map for 1 layer of the mTower stack preceded by a 28 mm tungsten absorber in Allpix^2 for 50, 100, and 150 GeV electrons.

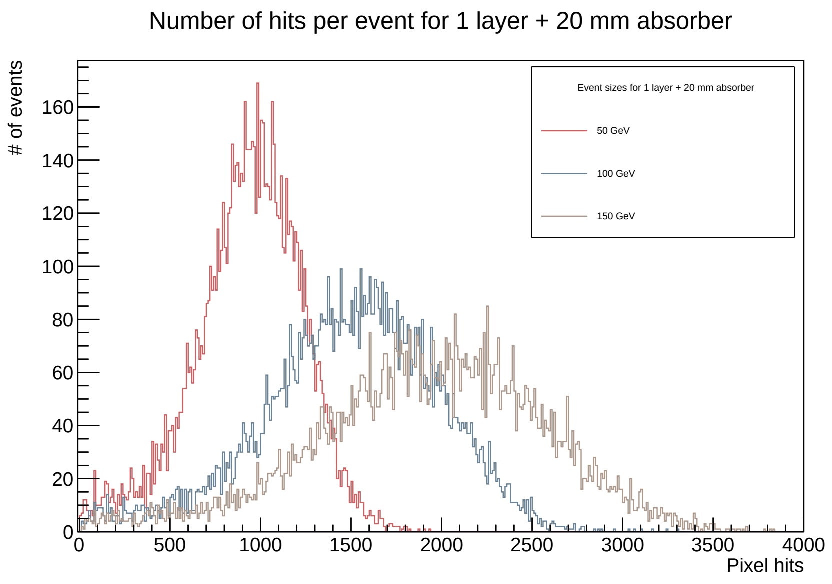


Figure A.3: Number of hits per event for 1 layer of the FoCal stack preceded by a 20 mm tungsten absorber in Allpix^2 for 50, 100, and 150 GeV electrons.

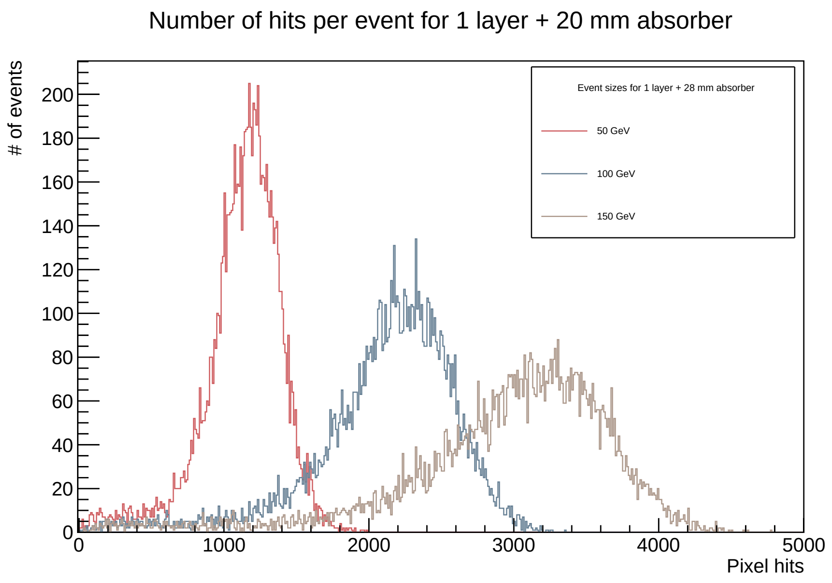


Figure A.4: Number of hits per event for 1 layer of the mTower stack preceded by a 28 mm tungsten absorber in Allpix^2 for 50, 100, and 150 GeV electrons.

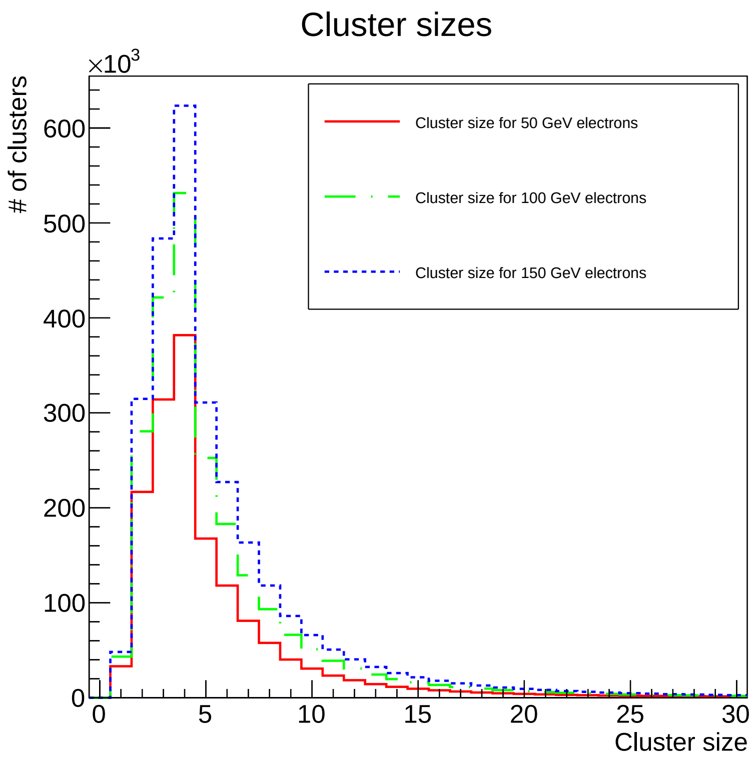


Figure A.5: Cluster size for 1 layer of the mTower stack preceded by a 20 mm tungsten absorber in Allpix^2 for 50, 100, and 150 GeV electrons.

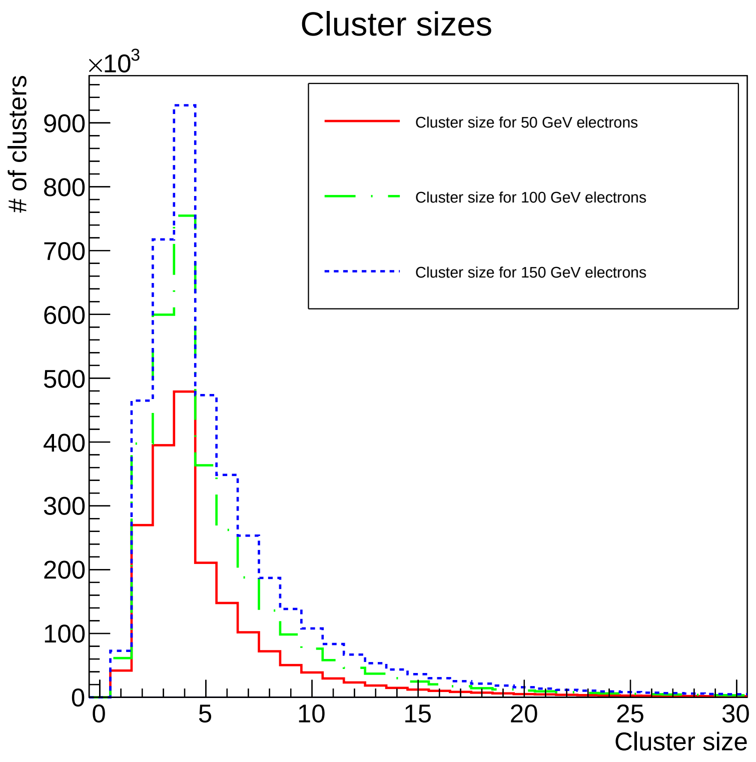


Figure A.6: Cluster size for 1 layer of the mTower stack preceded by a 28 mm tungsten absorber in Allpix^2 for 50, 100, and 150 GeV electrons.

B Code clarification

This section explains the workings of the code used in the SimpDiff method. The SimpDiff method uses output data from the GEANT4 B4a example modified by J. Baas as input for analysis. This modified setup outputs the data from GEANT4 as a ROOT TTree, which is always named **B4Grid.root**; for convenience, this can be edited in the source files of his code. In the analysis performed for this thesis, the ROOT-files were renamed to B4Grid_(1)layer_(2)GeV.root, where (1) corresponds to the number of layers in the configuration, and (2) corresponds to the beam energy used in the simulation. To get an idea of how the geometry was defined in this modified example, one can refer to [31].

The SimpDiff method consists of a total of 5 different files:

- mTowerMCEvent.cxx,
- mTowerMCEvent.h,
- mTowerMCHit.cxx,
- mTowerMCHit.h,
- G4tomTowerEvent.C.

The first 4 files are class definitions used in structuring the analysed data. The mTowerMCEvent and mTowerMCHit classes contain constructors and destructors. Their header files include get and set functions for the positions and cluster sizes; the mTowerMCHit.h file also contains a set function for the 'coordinates' of a hit, which include the lane, row, column, energy deposition, and absolute position.

The G4tomTowerEvent.C file contains the analysis script used in the data. This script takes two input variables; the number of layers, and the beam energy used in simulations. These have predefined values of 1, 4, and 8, and 50, 100, and 150, respectively. If-statements for these different variables have been written into the script manually, and can be extended by copying and modifying the code to correspond to different layer numbers and beam energies.

The positions that were read out from GEANT4 in this file have been manually calculated, as the mean z position is defined differently for every layer configuration in GEANT4. In order to determine which conversion factors are necessary for absolute z position to layer number, one can take a look at the output tree that is produced by GEANT4.

The G4tomTowerEvent script and mTowerMCHit header file contain an array called *used*; because of C++ structure, this array has a predefined maximum length in the header file. At the time of writing this thesis, this array can contain a maximum of 50 million entries, which correspond to the entries obtained from GEANT4 output. These entries represent each interaction with the sensitive volume in the detector. For 8 layers, the number of interactions was approximately 13 million; for higher layer numbers, this maximum array length should be extended to properly accommodate the number of entries.

The G4tomTowerEvent script generates an output tree that registers information about every single hit per event. The script checks every single entry in the GEANT4 output tree, and compares it to other hits that occured in the same event with the use of two for-loops; if two hits happen to correspond to the same pixel hit, the deposited energy of both hits is added together to be used in the total energy calculation. The script only checks hits that are in the last layer of the configuration. All hits are stored in the aforementioned *used* array. The script also applies a Gaussian at the impact position whose mean is the absolute impact position, and whose variance depends on the total energy deposited in that pixel. After applying a threshold, the script checks which pixels are still covered by the Gaussian; these hits are then registered as hit. At the end of the second for-loop, the hits are added to the output tree; their coordinates are set in terms of absolute and pixel position, energy deposition, pixel number, and layer number.

During the process of checking triggered pixels, the script has a *cluster counter*. This cluster counter is incremented by one every time a pixel that is adjacent to any pixels hit is hit as well; this includes the hit obtained from the first for-loop, as well as all the hits from the second for-loop and the Gaussian smearing process. This counter is reset back to 1 at the beginning of the first for-loop, ensuring that all clusters have size ≥ 1.

At the end of the script, data is written to the output tree for every single event. The output tree contains (as relevant) the number of hits, the x and y pixel positions, and the cluster sizes. These files are named B4Events_(1)layer_(2)GeV.root, where (1) once again corresponds to the number of layers in the configuration, and (2) corresponds to the beam energy used in the simulation.

C Additional software information

For any bachelor student that might continue this research, a few pointers are given in this appendix. It should be duly noted that everything concerning this thesis was performed on a Unix system.

The applications that were used in this thesis were GEANT4, Emacs, ROOT, and Allpix^2. Emacs is a GNU text editor that was used to write the C++ code; ROOT is the CERN data analysis framework, as mentioned in section 4.1.4.

All of these applications have been installed on the Quark server cluster. However, the cluster does not offer any visualisation for either GEANT4 or Allpix^2. As such, it might be useful for one to install both of the frameworks individually on a local Unix machine. In order to install any application written in C++, the Unix machine requires a compiler; most Linux distros (Ubuntu, CentOS, Mint) come with ‘gxx’ pre-installed. It is also recommended to install CMake, as this is used by both GEANT4 and Allpix^2 to create executables; info can be found on their website (clickable). To review the prerequisites for installing GEANT4, one can read the GEANT4 installation manual. To see the prerequisites for Allpix^2, one can read the Allpix^2 manual. In order to install Allpix^2 it is necessary to install GEANT4 (and other software) beforehand, as Allpix^2 functions as a wrapper around legacy software.

In order to enter the Quark cluster, one must ssh into server cluster by entering (only when connected from eduroam):

```
ssh -Y -X quark.science.uu.nl
```

In order to use the applications on the cluster, different commands must be entered in the terminal to load into certain environments. For the GEANT4 environment, it is recommended one loads the following (in order):

```
module load python/2.7

export PATH=/cm/local/apps/environment-modules/3.2.10/Modules/3.2.10/bin/:$PATH

export ALIBUILD_WORK_DIR=/data1/software/alisoft

alienv enter VO_ALICE@CMake::latest,VO_ALICE@GEANT4::latest, VO_ALICE@ROOT::
    ↪ latest
```

If the Allpix^2 environment is desired, one can replace the last line in the previous listing with:

```
alienv enter --shellrc AliPhysics/latest-master-root6,AliRoot-OCDB/latest-release

export PATH=/data1/software/allpix-squared/inst/bin:$PATH

export LD_LIBRARY_PATH=/data1/software/allpix-squared/inst/lib:$LD_LIBRARY_PATH
```

After these environments have been loaded, one can freely use the applications and all their underlying dependencies. It should be noted that Emacs can be used on the cluster without loading any environment.

The definition of GEANT4 geometry, stepping action, analysis, and data generation are located in a folder called 'V7-Focal-Detector'; this folder contains the modified GEANT4 B4a example. In the source files, one can edit parameters to fit their simulations better. Every time the source code for a GEANT4 example is edited, it must be rebuilt using CMake. A good instruction on how to build GEANT4 examples can be found here.

All data for Allpix^2 is contained in the appropriate 'allpix-squared' folder. This folder contains all data necessary for simulation construction. All the original files left over from installation were left in. A folder called 'ALPIDE_conf' was added to the allpix-squared/examples folder; this folder contains all the geometry definitions the chip and detector, as well as the different simulation setups with module parameters preset.