PPD Summer Student Report

MIGDAL Collaboration

Tom Szwarcer

University of Oxford

1st July 2024 - 23 August 2024

Contents

1 Introduction

During this summer placement, simulations were done to better understand the visible scintillation and electron transport properties of CF_4/Ar mixtures in gas electron multipliers (GEMs). The simulated GEM specifications matched those of the detector used in the MIGDAL experiment [1], in the hopes that results obtained during this placement could be used to advance the understanding of processes occurring in the detector, with the goal of optimising its performance.

Part I
Preliminary Work

2 Goals and Motivation

A project goal was the detailed simulation of visible scintillation in a CF_4/Ar double GEM system. Before carrying this out, some preliminary work was required. Firstly, a literature search was done in order to gauge the current understanding of visible scintillation in this particular gas mixture. It was also useful to know if simulations of this type had been done before. Secondly, a brief investigation into the properties of CF_4 was done, primarily to learn if the Cython-based Pyboltz [2] could be more adaptable and useful for the needs of the project than the Fortran-based Magboltz [3]. Thirdly, a single GEM system was simulated, in order to gain familiarity with the tools required to generate the geometry and field map and simulate electron transport. Using a single GEM system at this stage had the added benefit of being able to test code more quickly, as double GEMs are much more computationally intensive to simulate than single GEMs. Within this single GEM geometry, code for tracking individual scattering processes was developed, and later applied to a double GEM geometry.

3 Literature Search

3.1 Understanding Scintillation Mechanisms in CF_4/Ar Mixtures

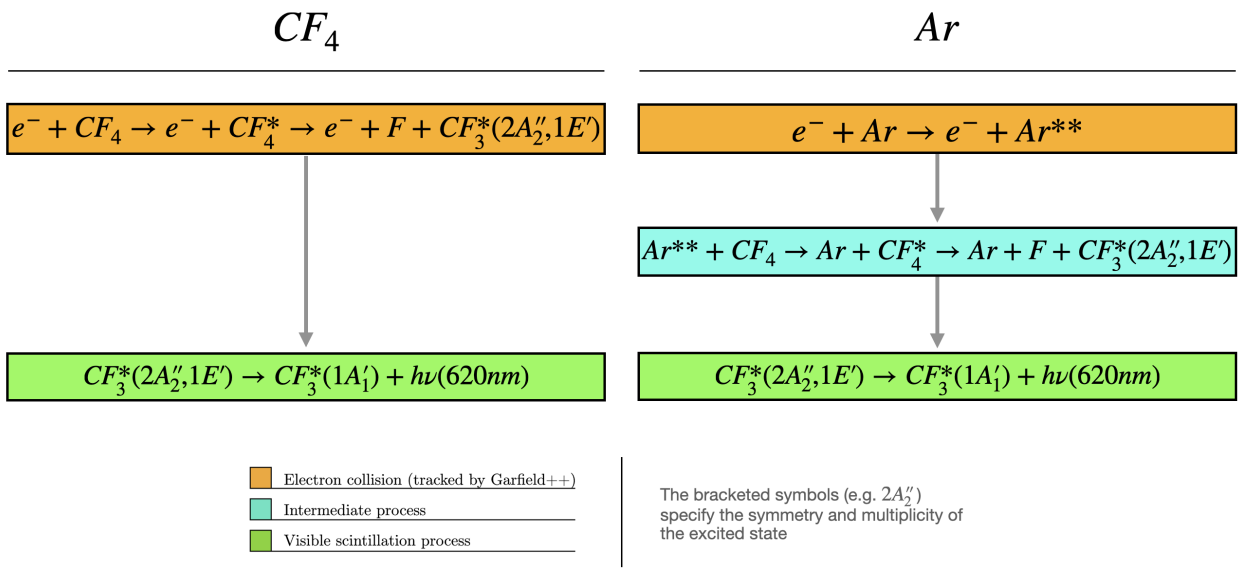


Figure 1: Overview of the processes involved in the scintillation of CF_4 and Argon.

Data collected by the MIGDAL collaboration shows the addition of Ar to CF_4 can result in higher light yields for a given amount of charge collected. We would like to simulate the scintillation of CF_4 and Ar in order to better understand this effect and the processes involved. A summary of the mechanisms [4] is given in Figure 1.

4 Properties of CF_4

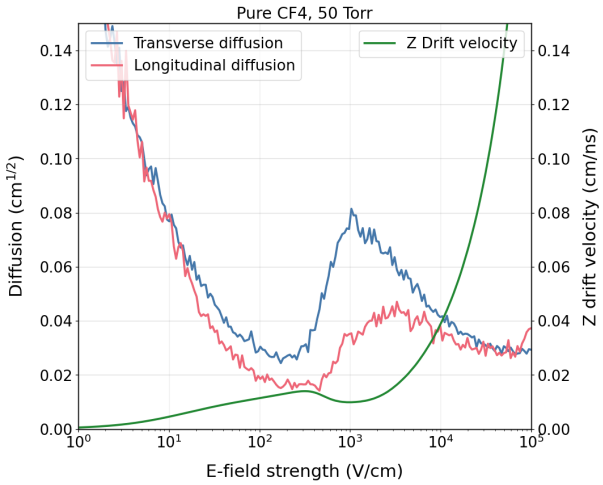
A short investigation was done into the electron transport properties of CF_4. Normally, this would have been done using Magboltz, but PyBoltz, a refactorization of Magboltz into Cython, showed promise in potentially being more adaptable to our needs. Using Pyboltz for these simulations would also tell us about its accuracy and efficiency compared to Magboltz, which could inform future projects.

4.1 PyBoltz Installation

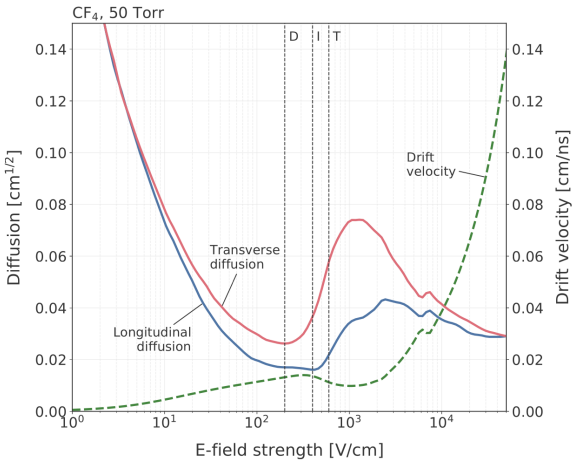
PyBoltz proved difficult to install on Python 3.9.18, the version used on our systems. After a few small modifications to the source code and writing of new setup files, a new version of PyBoltz plus installation instructions was uploaded to GitHub [5].

4.2 Data Acquisition

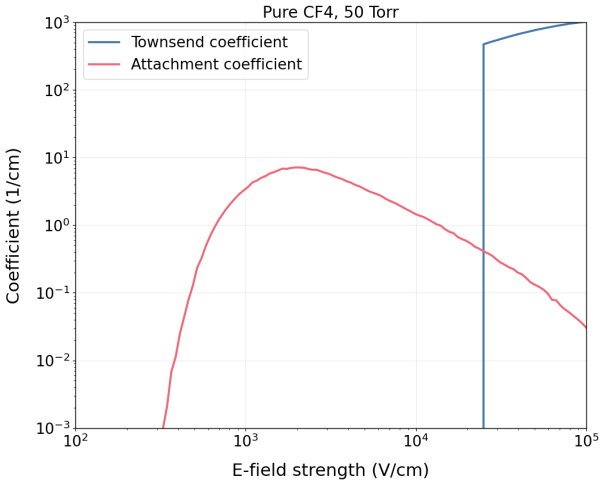
A Python script was written in order to get the relevant data out of PyBoltz, in the correct format for later analysis. The program generated 200 logarithmically spaced E-field values; the gas parameters were calculated once for each value. Certain parameters were highly susceptible to statistical fluctuation (such as diffusion coefficients), and this process meant they would not benefit from the smoothing that taking multiple measurements and finding a mean would yield. The choice to take a single measurement per E-field was made due to time constraints.



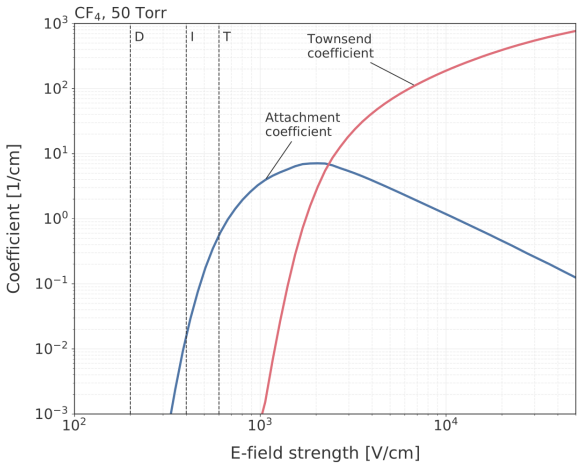
(a) PyBoltz simulation



(b) MIGDAL simulation



(c) PyBoltz simulation



(d) MIGDAL simulation

Figure 2: Comparison between PyBoltz simulations and Magboltz simulations carried out by the MIGDAL collaboration

4.3 Data Analysis

A Python notebook was created in order to analyse the data from PyBoltz. Its functionality was simply to read in the parameters generated by PyBoltz and plot them as a function of E-field.

4.4 Results

The results and comparison to MIGDAL simulations (done in Magboltz) can be seen in Figure 2. It is shown that Magboltz and PyBoltz agree quite well in diffusion, drift velocity and attachment, but there is a disagreement with the Townsend coefficient. It was unclear what the cause of this disagreement was, and this project was not taken any further in order to focus on other simulations.

5 Single GEM Geometry and Electric Field

5.1 Overview

There are a few steps involved with creating the GEM geometry and defining the E-field within it;

1. Create a 3D model of the GEM unit cell
2. Create a mesh over the model for finite element analysis
3. Define the relevant boundary conditions (BCs)
4. Solve for the E-field over the mesh
5. Test the E-field map
6. Drift electrons within the GEM geometry, in the presence of gas and E-field

5.2 GEM Unit Cell Model

We begin with creating a 3D model of the GEM unit cell. This was done with the software Gmsh [6]. A choice needed to be made regarding which unit cell configuration would be used; the simplest (half-hole) was tried first, with the number of holes increased until the simplest configuration that suited our needs was found.

5.2.1 Half-Hole

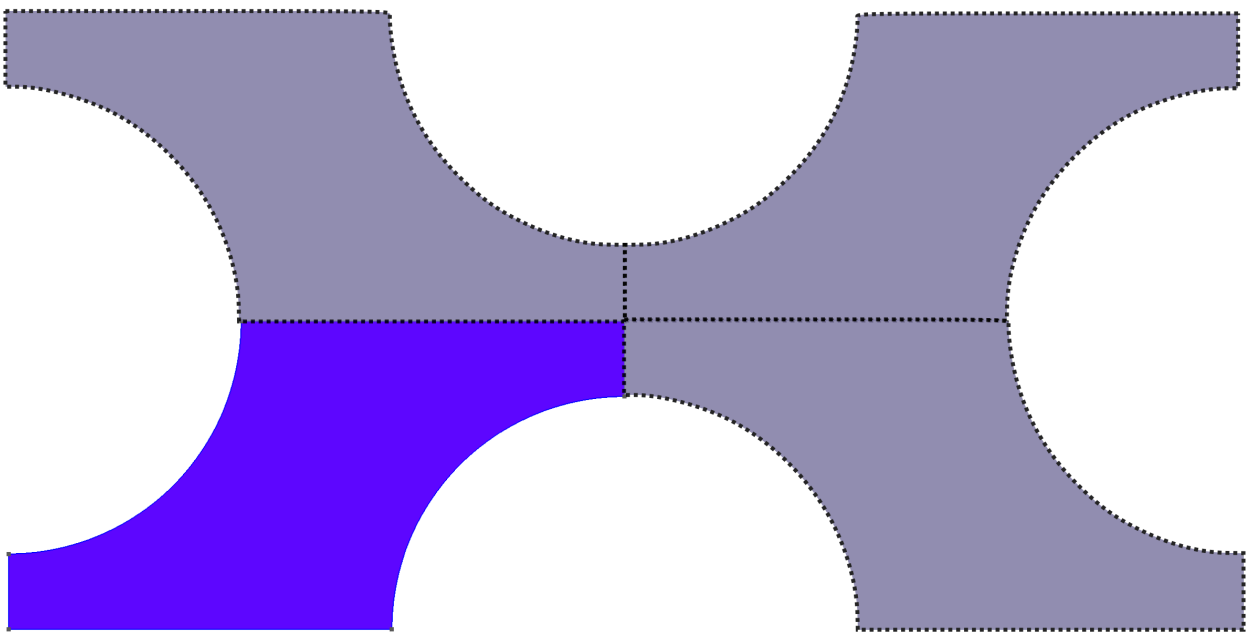
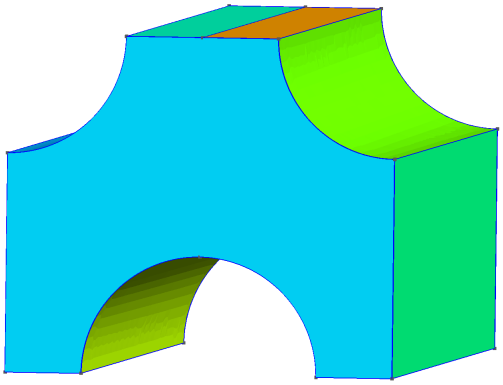


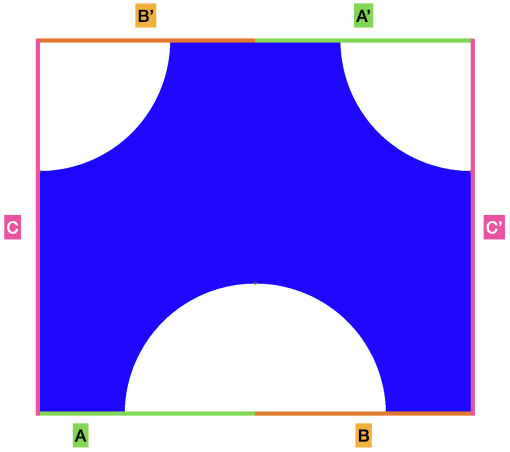
Figure 3: The half-hole unit cell (bottom left) and its tiling to reproduce the GEM geometry

The half-hole unit cell (Figure 3) was easy to design but difficult to implement boundary conditions with. The software Elmer FEM [7] was used to generate the E-field maps; in Elmer, periodic boundary conditions are defined by specifying surfaces (known as physical surfaces) subject to BCs, and defining how different physical surfaces are linked to each other periodically. The issue with this geometry arises due to the half-hole unit cell not having a complete set of the necessary surfaces that are accessible through translation. With reference to Figure 3, all sides are periodic to themselves under a reflection due to the mirror periodicity in both directions. It was unclear if such reflections of the boundary conditions were possible in Gmsh/Elmer, and it was decided that we would proceed with trying larger unit cells.

5.2.2 One-Hole



(a) The one-hole unit cell. Note the splitting of the upper surface, necessary to assign periodic BCs.



(b) The translations required to impose periodic BCs in the one-hole unit cell

Figure 4: The one-hole unit cell

The one-hole geometry is shown in Figure 4. This geometry has all the needed sides accessible through translations, as shown in Figure 4b. Here the issue is the splitting of the top and bottom sides into A/B and A'/B'. Within the OpenCASCADE system in Gmsh, it proved difficult to split the surfaces that bound a volume into multiple connected (but separate) individual surfaces. It was a particular challenge to do this without ending up with duplicate surfaces, and then to impose the correct boundary conditions. In the interests of time, the decision was made to use a larger but more symmetric unit cell.

5.2.3 Two-Hole

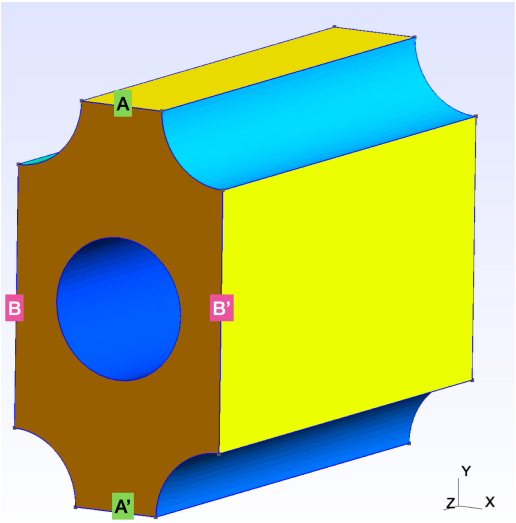


Figure 5: The two-hole unit cell.

The two-hole geometry is shown in Figure 5. In this case, two pairs of surfaces are simply related, with no surface splitting required. It was simple to implement periodic BCs in Elmer as well.

5.3 Generating the E-Field Map

5.3.1 Boundary Conditions

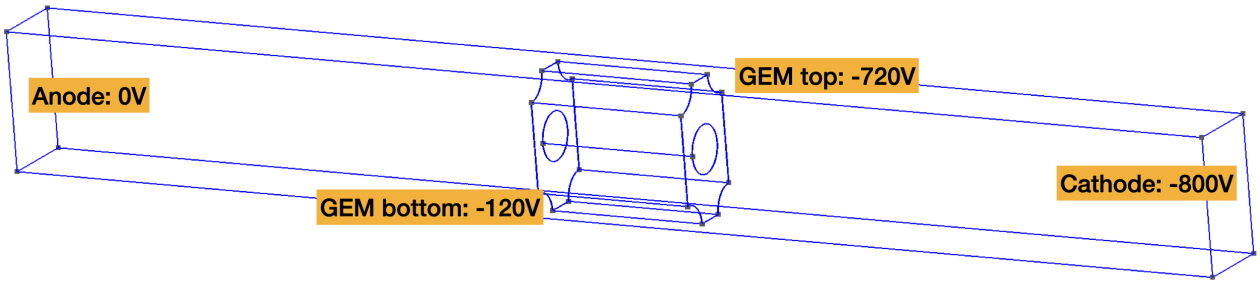


Figure 6: The voltage configuration of the single GEM geometry

For the single GEM case, all testing was done at a $\Delta V = 600$V. The voltages are indicated in Figure 6. These voltages were set in the `.sif` file, to be read by ElmerSolver. Also set was the requirement of the potential being periodic for opposite surfaces.

5.3.2 Finite Element Analysis

Once a 3D mesh was produced with Gmsh, it was converted to a format that ElmerSolver could understand by using ElmerGrid. Following this, the boundary conditions were implemented, and ElmerSolver was used to find the potential throughout the mesh.

5.4 Testing the Model & Field Map

The field map and model geometry were tested in Garfield++.

5.4.1 Field Artefacts

The first successful iteration of the model revealed artefacts present in the field map (Figure 7). These were fixed by re-compiling Garfield++.

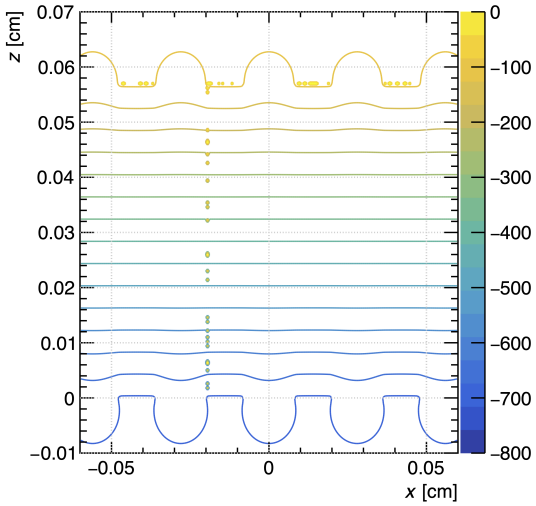
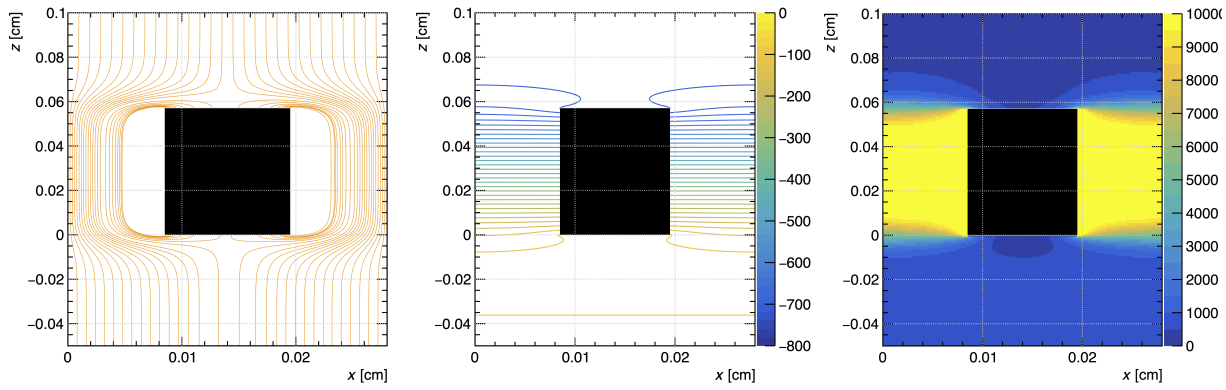


Figure 7: Equipotentials of the single GEM configuration. Note the artefacts present in the field map. Note further that in this configuration, the voltage order was reversed from that shown in Figure 6. This was corrected for all subsequent plots

5.4.2 Results

The field map was investigated, and plots of field lines, equipotentials and E-field magnitude were produced (Figure 8). In addition, test avalanches were run, an example of which can be seen in Figure 9.



(a) Field lines (b) Equipotentials (c) E-field magnitude

Figure 8: Testing carried out on the single GEM field map and geometry. The black box is the dielectric, with the holes at either side.

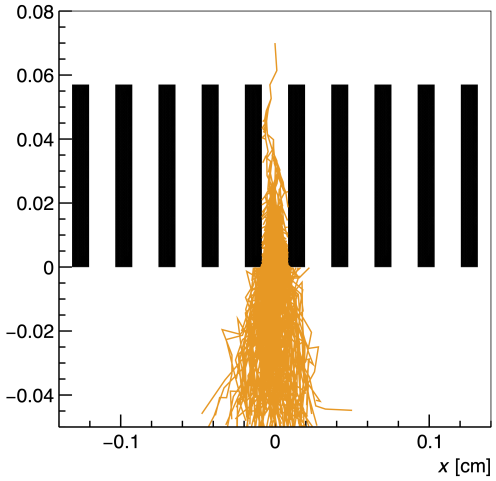


Figure 9: Test avalanche run in the single GEM configuration. Note the losses to GEM walls

6 Collision Tracking in Garfield++

It was unclear how to go from an understanding of which states were involved with the visible scintillation of CF_4/Ar to tracking the number of scintillation events that occurred for a given avalanche. It was noticed that Magboltz gives as output collision frequencies for a given process resulting from a collision, e.g. ionisation. An example of such processes for pure CF_4 is given in Figure 10.
Hence, the key would be to gain access to this information, and as Garfield++ interfaces with Magboltz, to use it during electron avalanches to track collisions resulting in scintillation-related processes.

6.1 Tracking with `GetNumberOfElectronCollisions()`

The first way this was done was with Garfield's function `GetNumberOfElectronCollisions()`, which is part of `MediumMagboltz`. This takes as input the level index of the scattering process you would like to track the collisions in - indexed consecutively from one. If you are interested in tracking a particular process, you would first need to identify its index, and then simply call the function with the index as input. In practice, the index was identified by saving a list of all the possible levels, choosing an energy from the list, then comparing the energy of all levels reported by Garfield++ to the energy of the level of interest and returning the index if there is a match.

description	energy
ELASTIC ANISOTROPIC CF4	0
ION CF3+	15.7
ION CF2+	21.47
ION CF+	29.14
ION F+	34.5
ION C+	34.77
DOUBLE ION CF3+, F+	36
DOUBLE ION CF2+, F+	40
IONS CF3++ OR CF2++	41
DOUBLE ION CF+, F+	43
DOUBLE ION C+, F+	63
ATTACHMENT	0
VIB V2 ANISOTROPIC	-0.0539
VIB V2 ANISOTROPIC	0.0539

Figure 10: The first few electron collision-induced processes in pure CF_4. Energy listed is electron energy loss.

6.2 Tracking With User Handles

Another way of tracking collisions involved user handles. A user handle is a function that is called whenever a certain event takes place; for example, upon each collision. User handles provide access to the location of the collision, energies involved, directions of the electron before and after, and crucially the level index of the scattering process. This allows us to track the number of collisions occurring in particular levels of interest, and restrict the tracking of collisions to a particular region. This is beneficial as in the MIGDAL experiment's double GEM configuration, the vast majority of the light detected by the camera originates from scintillation in the lower GEM. This could allow for increased efficiency by reducing the region collisions are tracked over. Nevertheless, with the benefit of hindsight, it seems clear that `GetNumberOfElectronCollisions()` would be more efficient than the user handle method: Garfield++ will calculate the number of collisions in a particular level with `GetNumberOfElectronCollisions()` anyway, and using a user handle would essentially count them in two different ways for no reason. Another potential slowdown of the user handle method is that for each collision, a for loop had to be run to compare the level index of the collision to the tracked level indices, and increment the relevant level's counter accordingly. While this is a trivial operation, it may lead to inefficiencies over many millions of collisions. Nevertheless, the programs that were written to track collisions in levels of interest used the user handle method. In future, inefficiencies like this will be taken more into consideration.

Part II
Simultaneous Light and Charge Simulations

7 Goals and Motivation

As discussed previously, data collected by the MIGDAL collaboration shows the addition of Ar to CF_4 can result in higher light yields for a given amount of charge collected. We would like to simulate the scintillation of CF_4 and Ar in order to better understand this effect and the processes involved.
It was decided that two important MIGDAL results should be reproduced in simulations. The first of these was a plot showing gas gain for different CF_4/Ar ratios (Figure 11), in order to confirm the simulations accurately measure charge, and the second was a plot showing simultaneous charge and light measurement in the same gas mixtures (Figure 12). The latter shows the aforementioned effect on light yield of adding Argon to CF_4. This effect is useful as it allows the GEM ΔV to be kept lower with no loss in light, preventing discharges while still allowing low energy tracks to be resolved.

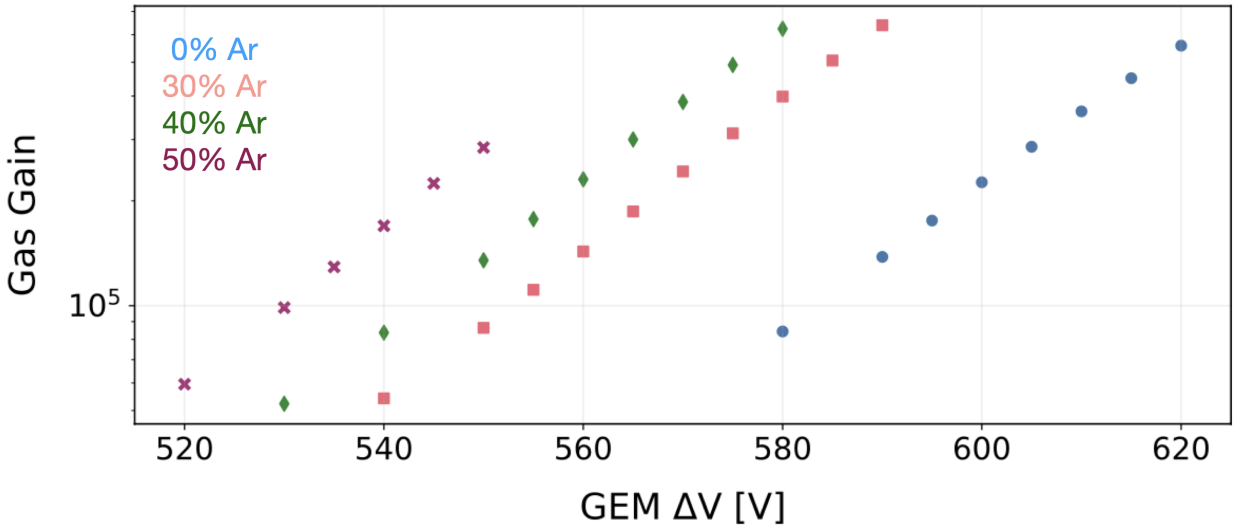


Figure 11: MIGDAL data showing gas gain for different CF_4/Ar ratios.

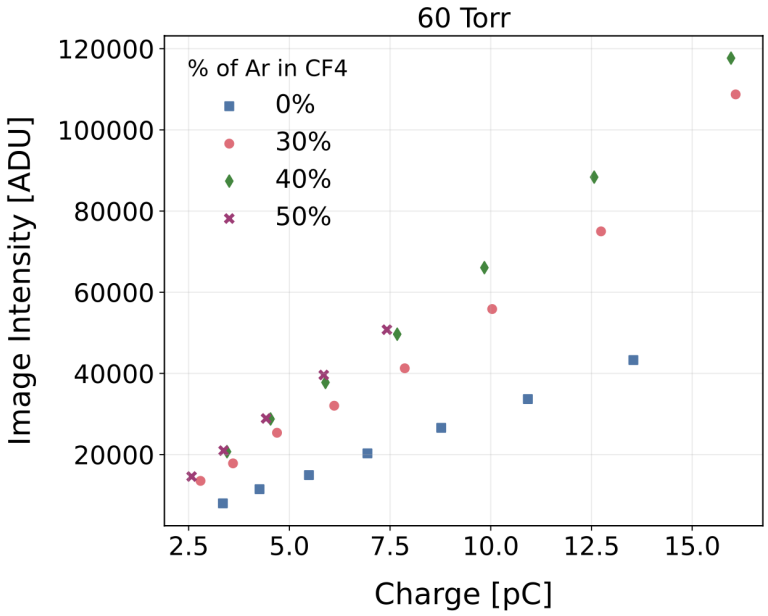


Figure 12: MIGDAL data showing simultaneous light and charge measurements for different CF_4/Ar ratios.

8 Double GEM Geometry and Electric Field

In order to simulate electron transport in a double GEM, the electric field and geometry needed to be generated. Again, this was done in Gmsh and Elmer. The two-hole unit cell was used once again, with the process being identical to the single GEM case. The specifications are shown in Figure 13.

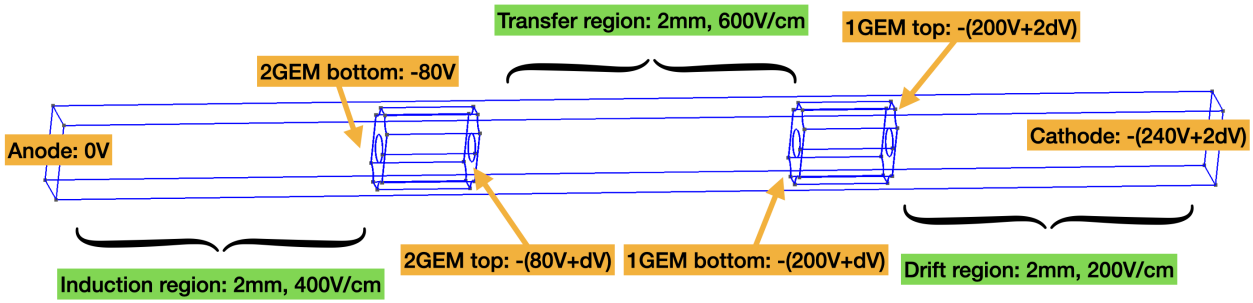
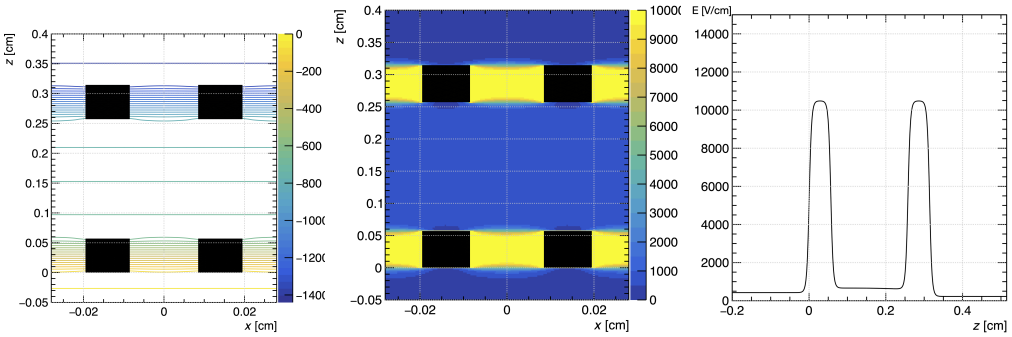


Figure 13: Specifications of the double GEM setup.

8.1 Testing the Model & Field Map

Once the geometry and field map had been generated, they were tested. This was once again done in Garfield++. Figure 14 shows the results of these tests.



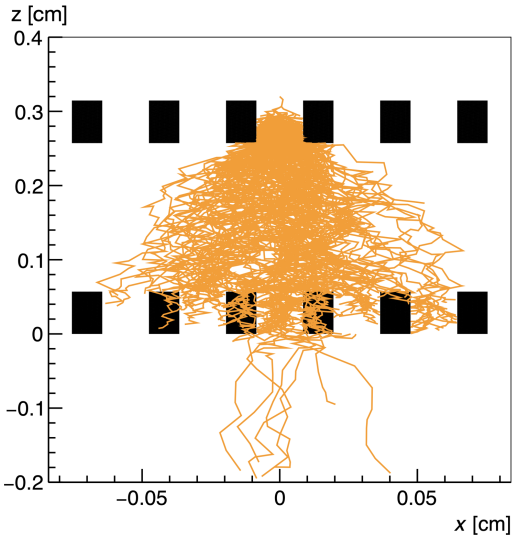
(a) Equipotentials (b) E-field magnitude (c) E-field profile

Figure 14: Testing carried out on the double GEM field map and geometry at $\Delta V = 600$V.

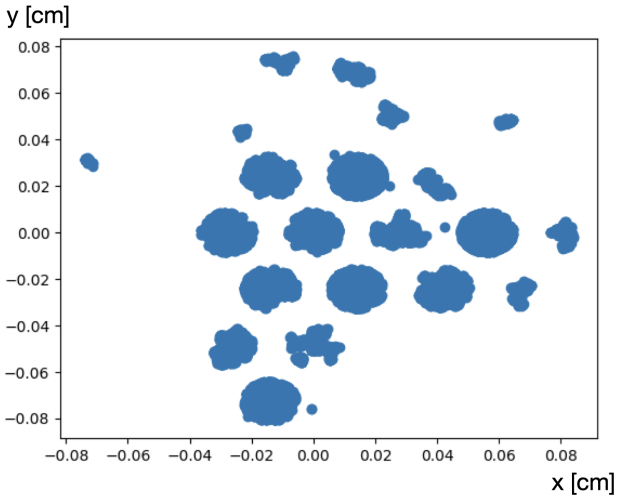
In addition to the field map tests, test avalanches were run. These were limited to a maximum size of 1000, as the double GEM system proved to be computationally intesive to simulate. No data needed to be collected at this stage, so the size limitation was justified. A sample result can be seen in Figure 15a. Here, it can be seen that drift lines appear to pass through the dielectric: this is a side effect of projecting a 3D avalanche into a 2D plane. As Figure 15b shows, all electrons pass through the holes in these simulations. The data in Figure 15b was collected with user handles: every time a collision occurred near the bottom edge of the second GEM, a user handle would record the location of said collision. These were saved to a CSV file and plotted with a Python notebook.

9 Gas Gain Simulations

In order to measure light and charge simultaneously, we first wanted to be sure that our simulations could handle charge measurements correctly. Once this was confirmed, we would then proceed to light measurements. Gas gain was defined as the number of electrons collected at the anode divided by the number of primary electrons. As each avalanche was started by a single primary electron, the gas gain is simply given by the number of electrons collected at the anode.



(a) Test avalanche with size limit 1000



(b) Electrons exiting holes (for a different avalanche), viewed from below the 2nd GEM

Figure 15: Test avalanches run on the double GEM field map and geometry.

9.1 Counting Electrons

The method for counting the number of electrons reaching the anode is as follows.

1. Collect information about all of the endpoints of electrons in the avalanche
2. Count the number of endpoints with z-coordinates within a certain small tolerance of the anode z-position.

In practice the second of these was done with the Garfield++ function `GetElectronEndpoint()`.

9.2 Data Analysis

Once the gain data was collected, it was analysed using a Python notebook (viewable on the GitHub repository [8]). Data was collected for a ΔV range of 590V-620V in pure CF_4, and 550V-590V in 70% CF_4/30% Ar. The gain distribution was plotted, and a skew-Gaussian distribution was fitted to the data. The parametrisation of the skew-Gaussian PDF (f) was given as:

$$\phi = e^{-\frac{1}{2}(\frac{n}{m}-\mu)^2} \tag{1}$$

$$\varphi = 2\left(1 + \text{erf}\left(\alpha\left(\frac{n}{m} - \mu\right)\right)\right) \tag{2}$$

$$f = a\phi\varphi \tag{3}$$

where n is the gain, and m, μ, a and α are fit parameters. The function erf is the error function. This parametrisation was used to both avoid errors reported by Python and large correlations between parameters that occurred when using more common parametrisations found in the literature. The mean and standard error were found from the fit distribution.

9.3 Results

The results are shown in Figure 16. There was a discrepancy of $\approx 1.75\times$ for pure CF_4 and $\approx 2.9\times$ for 70/30 CF_4/Ar. A discrepancy of a factor of two has previously been observed in low-pressure GEM simulations using Garfield++ ([9], [10]), likely due to the E-field being taken as constant over the mean free path of the electron. The long mean free paths and rapidly changing E-fields in a low pressure GEM call into question the validity of this approximation.

In the operation of real GEMs, there would be some contribution to gain from electrons that hit the walls, either from back-scattering or emission of secondary electrons. This was not taken into account in these simulations. To implement this, some information about the rates of these processes would be needed, from which a probability for each process could be calculated. A certain fraction of all electrons that impact the walls could be re-introduced into the drift medium, based on this probability.

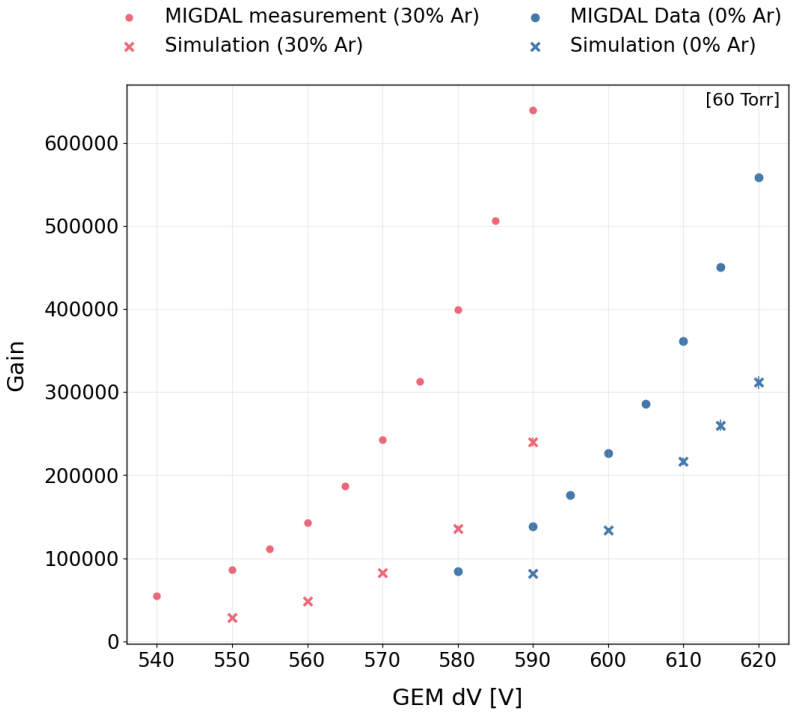


Figure 16: Comparison between simulation results and MIGDAL data for gas gain

10 Simultaneous Light and Charge Simulations

Although there was a discrepancy factor present, this could be corrected for later. Importantly, the gain curves showed the correct exponential relationship. We proceeded on to simultaneous light and charge measurements.

10.1 Converting Collisions to Scintillation

As shown previously, Garfield can return information about the number of collisions for various different scattering processes (levels) for Argon and CF_4. The levels of interest for CF_4 are the neutral dissociation levels, as CF_4^* undergoes neutral dissociation into $CF_3^*(2A_2'', 1E')$+F. In Argon, we are interested in the production of the so-called Ar** state [4].
Possible thresholds (12.5eV for CF_4 and 12.91eV for Ar) were identified from E. Seravalli's PhD thesis [11]. These states were identified in the Magboltz output, and code was written to track the number of collisions in these states over an avalanche.

10.2 Understanding Scintillation Mechanisms

As these were threshold energies, more information was needed in order to know which range of states we should track. A meeting was held with D. González-Díaz and P. Amedo to obtain this info. The conclusion that was reached was that little can be said about the states corresponding to Ar** production without further data; for CF_4, their suggestion was to track all states above 14eV, this energy being the likely threshold for the production of the $CF_3^*(2A_2'', 1E')$ state. For both cases, they recommended that we implement a fixed probability that collisions in the tracked levels will result in scintillation. This probability could be tuned to fit simulation results to the data. Without sufficient understanding of Ar** production, we proceeded with CF_4 with the aim of obtaining a reasonable value for the number of photons produced per electron.

10.3 Reducing Computation Time

Up until this point, avalanche simulations in the double GEM configuration had been taking a lot of time. A few strategies were explored to mitigate this.

10.3.1 `AvalancheMC` and `AvalancheMicroscopic`

Garfield's `AvalancheMicroscopic` class is used to microscopically simulate electrons drifting in the presence of gas and E-field. This simulation is done using a table of collision rates for each scattering process, and in

practice was quite computationally intensive in the double GEM case. This is because an extra ∼2.5mm region had to be simulated compared to the single GEM case, due to the presence of the transfer region and second GEM. One solution to this was to use the less computationally intensive `AvalancheMC` class, which simulates electron avalanches using Monte Carlo methods.
The main idea was to split the double GEM system into multiple regions. The primary electron began just above the top GEM, so the drift region did not need to be simulated with `AvalancheMC`. The system was divided as follows:

1. Top GEM: `AvalancheMicroscopic`
2. Transfer region: `AvalancheMC`
3. Bottom GEM: `AvalancheMicroscopic`
4. Induction region: `AvalancheMC`

`AvalancheMicroscopic` was used in GEMs due to its higher accuracy. The challenge in implementing this was the hand-off between electrons leaving one region and entering another. The solution was to create a small overlap between regions and record all endpoints of electrons in this overlap. Then, a new avalanche was started from each point.
`AvalancheMC` was used in the transfer and induction regions, where accuracy was less important and large amounts of multiplication would be unlikely to occur.
Once the GEM had been divided into multiple regions, code was written to hand off all the electrons that made it into the next region to the relevant avalanche class. This was done using a `struct` that stored the position, time, energy and direction of the electrons that reached the next region. The issue with this method was that `AvalancheMC` does not make a final electron energy available as output (nor does it take an initial energy as input): this presented a problem when handing off from the transfer region (`AvalancheMC`) to the bottom GEM (`AvalancheMicroscopic`) as the latter required an energy to be provided as input, but the energy of electrons reaching the bottom GEM was not reported by `AvalancheMC`. To resolve this, code was written to find the energies of electrons reaching just above the bottom GEM, using `AvalancheMicroscopic`. This code can be found in the GitHub repository under `e_finder.C` [8]. The code only calculates the energy for a single avalanche at 600V; ideally, it would be calculated over many avalanches, for a range of voltages. In the interests of time, and knowing that an individual electron's energy is subject to huge statistical fluctuations over short timescales, it was only calculated twice, with good agreement between both values.
The code for this 'hand-off' process can be found on the GitHub repository [8].

10.3.2 Parallelising the Gas File

The `AvalancheMC` class requires a gas file (generated by Magboltz/Pyboltz) to be specified. This is a file that takes a long time to generate, especially for many electric field values. Code was written (see `gastable/` in [8]) to allow gas files to be generated in parallel, saving a significant amount of time.

10.3.3 Parallelising Second-GEM Avalanches

An investigation was done into the possibility of time saving from running avalanches in the second GEM in parallel. The idea was to begin the first avalanche from a single electron above the first GEM, then record the locations, times, energies, and directions of all resulting electrons before they reach the second GEM. This information would be exported and then read in by multiple programs (one for each electron), which would then simulate an avalanche for each of these second GEM electrons in parallel. In theory, this could lead to time saves by factors of at least 100, but in practice it was severely limited by the number of cores and typical number of jobs that could be run in parallel on our cluster. Code was successfully written, but unfortunately this method resulted in a net time loss for the aforementioned reasons.

10.4 Efficiency

As we were interested in number of photons per electron produced, we needed to know the total number of electrons produced in each avalanche, including the ones lost to the walls. This number N is related to the gain G by the following formula:

$$N = G/e \tag{4}$$

where e is the efficiency of the double GEM system, i.e. the average probability of an electron making it to the anode. Code was written to obtain a simulated value for e (`efficiency.C` in [8]), and was run for $\Delta V = 590$V over many avalanches. The code was only run for a single ΔV due to time constraints. It yielded a value of $e = 0.1759 \pm 0.0007$, in agreement with other simulations of losses that had been done previously by a colleague.

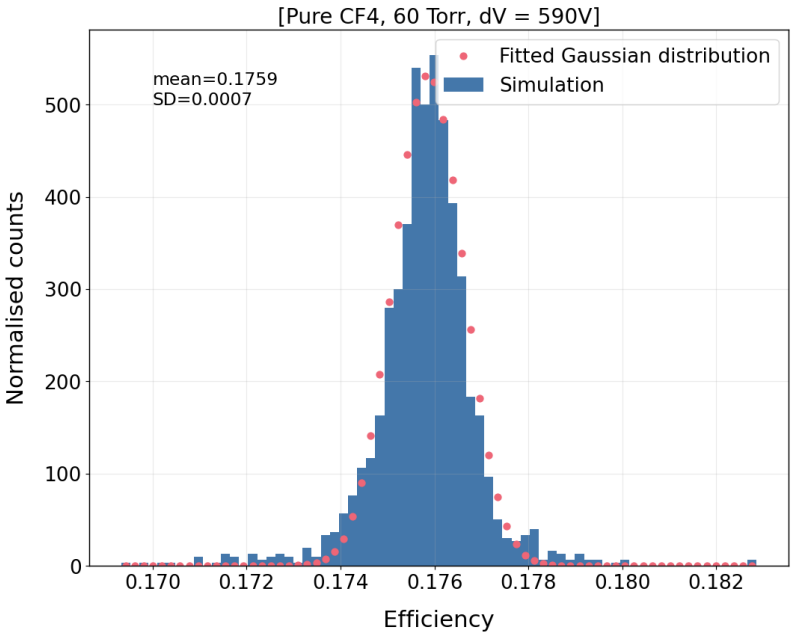
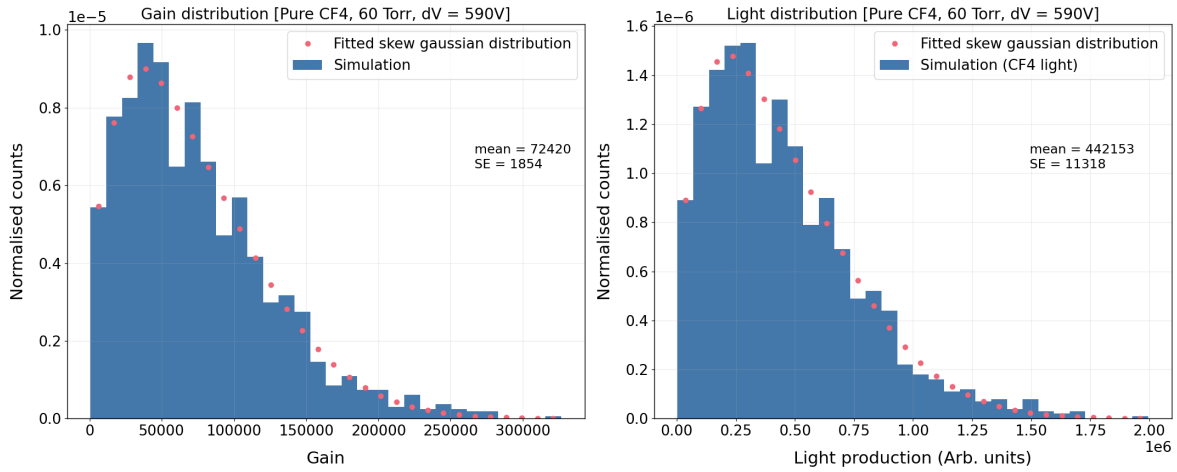


Figure 17: Plot showing efficiency distribution

10.5 Data Analysis

Data analysis for these results was done in a Python notebook, viewable on the GitHub repository [8]. Simulations were run for a ΔV range of 550V-610V in pure CF_4. For both the light and gain measurements, a skew-Gaussian fit was used, and the corresponding mean and standard error were found (Figure 18).



(a) Charge production

(b) Light production

Figure 18: Example plots for charge and light production in pure CF_4 at 590V. These values have not yet had corrective factors applied (see text)

Corrections were made to the light and charge data. For the charge, a gain correction factor of 1.75 was added, in order to account for the discrepancy between simulation and data. For the light production, the 'scintillation fraction', i.e. probability of a collision in a scintillation-related scattering processes actually resulting in scintillation was fixed at 0.2, on the recommendation of Dr González-Díaz. This value was chosen in order to match results to a rough target of 0.1 photons per electron [12].

10.6 Results

Plots of photon numbers against gain, as well as photons per electron against gain were produced (Figure 19). As the photons per electron value is expected to be pressure-dependent, a rigorous comparison to literature could not be made, but it is noted that the same downwards trend for increasing gain was visible in [13]. 'Photons per electron produced' is effectively the (corrected) number of collisions in the relevant levels divided by N (see Eq. 4), a number also subject to a corrective factor as described previously.

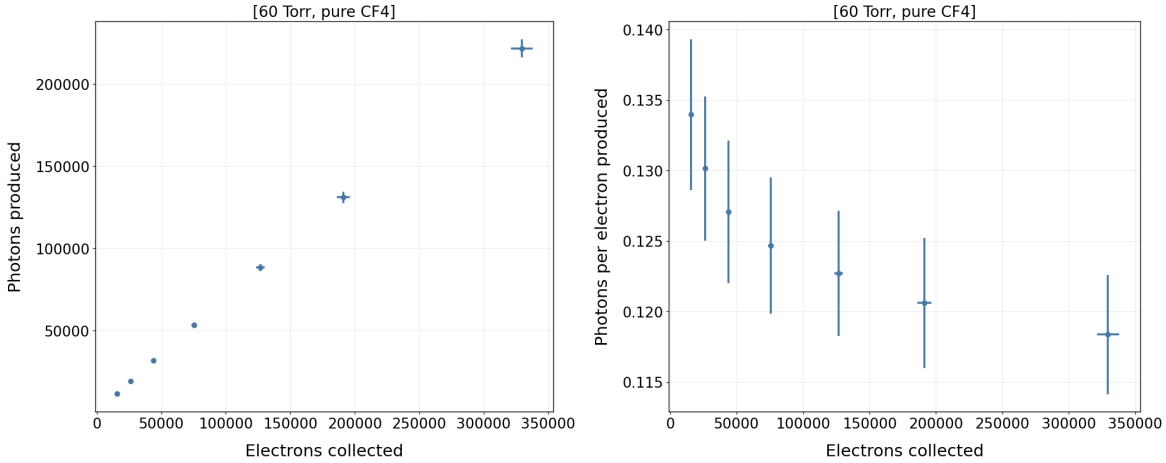


Figure 19: Charge and light simulation results

11 Conclusion

During this summer placement, detailed simulations of charge production in Argon/CF_4 mixtures, and of simultaneous charge and light production in pure CF_4 were done. Although there were few results produced, the key importance in this work is the capacity to be built upon for future research, and the documentation of the information avaliable from Garfield++/Magboltz for the purposes of simulating scintillation. A GEM simulation of CF_4/Ar visible scintillation could not be found in the literature, and it is hoped that the code written will prove useful in future work.

It is unfortunate that this project was subject to time constraints; these were brought on by the large amount of time required to gain familiarity with the software tools that were to be used, and also by inefficient decisions that were made along the way. This project has highlighted the importance of careful and thorough planning of the steps towards solving a scientific problem.

References

[1] URL: https://migdal.pp.rl.ac.uk/.

[2] B. Al Atoum et al. "Electron transport in gaseous detectors with a Python-based Monte Carlo simulation code". In: *Computer Physics Communications Volume 254* (2020).

[3] Stephen Biagi. URL: https://magboltz.web.cern.ch/magboltz/.

[4] P. Amedo et al. "Observation of strong wavelength-shifting in the argon-tetrafluoromethane system". In: *Front. Detect. Sci. Technol 1:1282854.* (2023).

[5] URL: https://github.com/tomszwarcer/PyBoltz.

[6] C. Geuzaine and J.-F. Remacle. "Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities". In: *International Journal for Numerical Methods in Engineering 79(11), pp. 1309-1331* (2009).

[7] URL: https://research.csc.fi/web/elmer/elmer.

[8] URL: https://github.com/tomszwarcer/gemsim.

[9] Djunes Janssens. *CERN summer student program rapport: A better understanding of gas gain simulations in GEM detectors.* 2019.

[10] URL: https://twiki.cern.ch/twiki/bin/view/MPGD/WG4-Simulation.

[11] Enrica Seravalli. "A Scintillating GEM Detector for 2D Dose Imaging in Hadron Therapy". PhD thesis. Delft University of Technology, 2008.

[12] F. Brunbauer. Private communication.

[13] A. Morozov et al. "Secondary scintillation in CF4: emission spectra and photon yields for MSGC and GEM". In: *JINST 7 P02008* (2012).