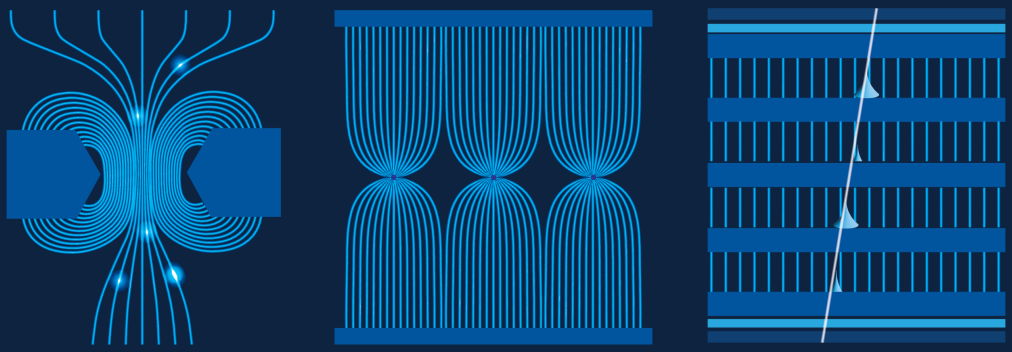
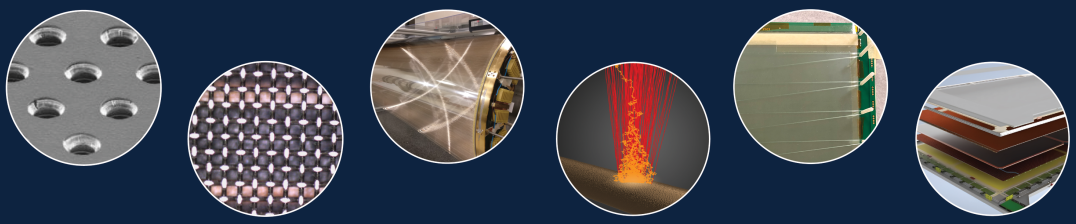
DRD1 Gaseous Detectors School

Lab Book

Simulation Excerpt





CERN, Geneva, Switzerland

November 27 - December 6, 2024

DRD1

DRD1
Gaseous Detector School

CERN

LAB 8: DETECTOR SIMULATIONS 1

Introduction

The objective of Lab 8 is to simulate the crucial processes occurring in a gaseous detector, from the creation of ionization by a passing particle up to the induction of a signal in the readout electrodes. The lab will start with an introduction to some of the primary tools commonly used in our community, such as ROOT [1] and Garfield++ [2]. Subsequently, several exercises will be discussed, focusing on replicating simulations of the electric field, gas mixtures, and electron drift under influence of an electric field and behavior during signal generation in the detector.

The activities will encompass both dedicated time for collectively working through the proposed exercises with the tutor and moments for independent code development and further expansion of the exercises.

The exercises will make use of Python and C++ code and are implemented in either C++ source code that has to be compiled by the student or Python Jupyter notebooks. We provide a virtual machine with Ubuntu 24.04LTS operating system where we have already installed ROOT and Garfield++. We will illustrate to the students how to compile and execute C++ programs for the exercises on this virtual machine and we will also illustrate how a jupyter notebook can be run on this virtual machine. These notebooks can also be executed on SWAN [3] service of CERN (for students with a CERN computing account) or on a Google COLAB workspace [4].

By the end of the laboratory, students will have gained fundamental knowledge for simulating a gaseous detector and the capability to design and simulate their own setups.

The Virtual machine with ROOT and Garfield++ pre-installed can be downloaded from the following folder on CERNBOX:

https://cernbox.cern.ch/s/2GwgQCBQ0TSMnMc

The virtual machine works with Oracle Virtual Box software (version 7.0 or higher) that can be downloaded for Windows, Linux and MacOS from the following webpage: https://www.virtualbox.org/

Simulated detector technologies

The laboratory activities focus on understanding and simulating key physics principles involved in detection. Various aspects of detector simulation will be studied

using different approaches. The primary detectors simulated will be RPC, drift tube, GEM, and Micromegas.

Experimental setup and instrumentation

The following tools will be used in this laboratory:

- Individual laptop (Each student should bring their own)
- Oracle VirtualBox 7.0 (to be downloaded by the student)
- Ubuntu 24.04 LTS with ROOT and Garfield++ pre-installed (to be downloaded and opened in Virtual box by the student)
- ROOT, an analysis framework for physics
- Garfield++, a toolkit for the detailed simulation of particle detectors based on ionization energy-loss in gasses and semiconductors.

The student will learn how to install VirtualBox and how to install an Ubuntu 24.04LTS virtual machine. The code based on C++ will be provided with a CMake file for the compilation of the code.

Work plan

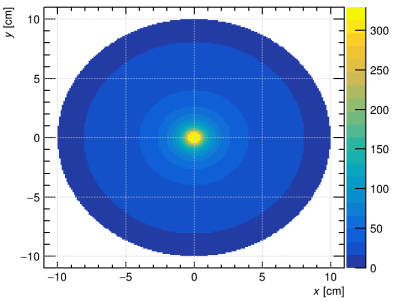
The simulation lab will benefit from various setups to simulate and measure quantities related to ionization, charge transport, avalanche, and signal induction. These setups include:

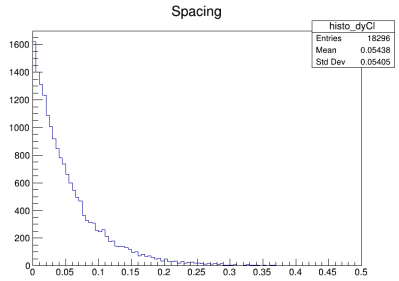
- Generate detector geometries and the associated electric field
- Visualize electron-ion creation and assess the electron-ion pair production.
- Visualize the electron drift path and evaluate the electron diffusion.
- Visualize the electric field within a GEM (Gas Electron Multiplier) hole and assess the amplification.
- Simulate a full detector and induce the signal on a readout plane.

The main steps in the Garfield++ simulation are the following:

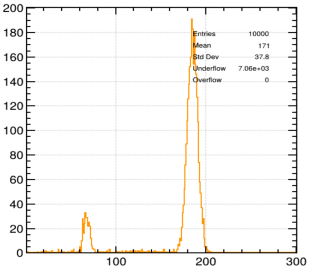
- Define the gas medium (gas mixtures, penning, ion mobility, environment parameters).
- Define the geometry and electric field(ComponentAnalyticField, ComponentAnsys123)
- Define the interacting particles and the components (tracks, clusters, electrons).
- Define the readout plane and the electronics.
- Use all the previous components for measurement as described in the following examples and exercises.

Exercise 1: Define different geometries such as: parallel plates, wire chamber and GEM: visualize the electric potential and the electric field

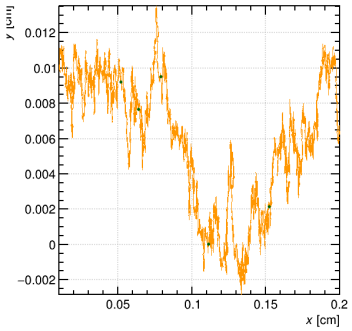




Exercise 2: Ionize the gas volume by simulating a charged particle traversing it, and evaluate the number of clusters, their spatial distribution, and the energy loss

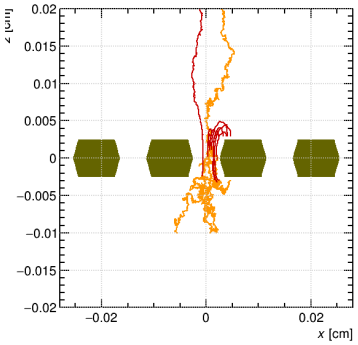


Exercise 3: Simulate an Iron-55 radioactive source starting from the photon induction in a gas volume



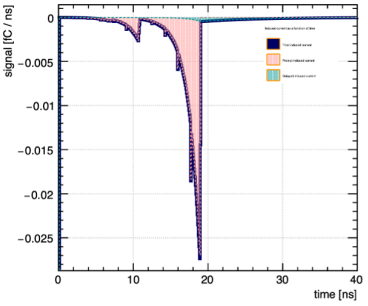
Exercise 4: Visualize and measure the electron diffusion from a primary cluster generated by a passing charged particle

Exercise 5: Visualize the impact on the electron diffusion on the given gas mixtures



Exercise 6: Reproduce an electron avalanche in a gas electron multiplier (GEM) and evaluate the detector gain

Exercise 7: Translate the moving charge into an induced signal on a readout plane



Discussion

Engage in discussions during the lab and utilize the discussed methods to propose a simulation of a gaseous detector or of a physics process occurring in a gas related to your own field of research.

References

[1] https://root.cern/

[2] https://garfieldpp.web.cern.ch/garfieldpp/ and https://github.com/winger/garfieldpp

[3] https://swan.web.cern.ch/swan/

[4] https://colab.research.google.com

LAB 9: DETECTOR SIMULATIONS 2

Introduction

In this exercise we will bring the knowledge and knowhow of the student to the next level. We will build upon what was learnt in the first detector simulation exercise (Lab 8) and focus on one specific detector technology. This is a so-called “long exercise” where we will simulate a state-of-the-art gaseous detectors to a point that the output can be compared to data. In this exercise the student will work independently (either individually or in groups of max 2 persons) and will create their own program to implement the simulation, using all available documentation online, and under guidance of the tutors.

The exercises will make use of C++ code and are implemented in either C++ source code that has to be compiled by the student. We provide a virtual machine with Ubuntu 24.04LTS operating system where we have already installed ROOT and Garfield++, as described in Lab 8.

Simulated detector technologies

The students can choose between two options such that she/he can choose a long exercise that is as close as possible to their area of research.

Option 1: Resistive Plate Chamber (RPC): Efficiency and timing resolution studies including space-charge effects.

Option 2: Resistive Micromegas: induction of signal in neighboring readout strips in the presence of a resistive layer.

Experimental setup and instrumentation

The same setup (virtual machine) of Lab 8 will be used in this one through which the student can access the exercises and Garfield++.

Work plan

Individual hand-on work of the student on the code with the assistance of tutors.
A skeleton of the code will be provided for the students that will be the starting point of the session. Following the instructions, hints, and function names indicated in the exercise sheets found below, the students will extend the code to reach the indicated objectives.

SIM2: Resistive Plate Chambers

DRD1 Gaseous Detectors School

November 2024

1. Trigger Resistive Plate Chamber

In this exercise, you'll learn how to simulate essential characteristics of a single-gap Resistive Plate Chamber (RPC) with a read-out plane electrode. For example, the detection probability of a given ionizing particle and the corresponding intrinsic time resolution. On the left of each exercise, you will find a table of useful methods from GARFIELD++ you might use to solve it. Figure 1 illustrates the trigger RPC to be simulated, whereas in Table 1 its dimensions can be found.

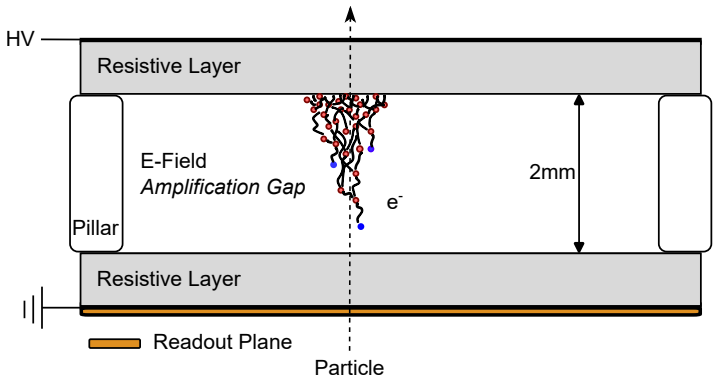


Figure 1: Trigger RPC with three stacked layers and a plane readout electrode. The primary electrons (in blue) inside the gas gap develop into an electron avalanche due to huge applied electric fields. For simplicity, the pillars to support the gap width uniformity are shown but will not be integrated into the simulation. *In your opinion, what impact do these pillars have on the detection probability?*

Table 1: Trigger RPC dimensions

Layer No	Type	Width (cm)	Relative Permittivity
1	Resistive (e.g., Bakelite)	0.2	8
2	Gas Gap	0.2	1
3	Resistive	0.2	8

a. **(Learning objective: Get to know** ComponentParallelPlate**)** Setup a 3-layer single gap RPC using ComponentParallelPlate with a voltage in the range of $8000 - 10\,000$ V. Add a plane electrode, a Box geometry containing a $10\,\mathrm{cm}^2$ RPC segment, and an initialized gas (load the gas file from the Examples/RPCSpaceCharge example). Finally, create a Sensor and add the component as well as the label of the plane-electrode to it. Retrieve the gas gap's electric field and swarm parameters (Townsend=α, Attachment=η, Drift=v). Calculate using the formula

$$\sigma_t = \frac{1.28}{(\alpha - \eta)\, v} \tag{1}$$

the expected time resolution at the specific voltage you have chosen.

Comprehension: The intrinsic time resolution is the unavoidable jitter between the time the ionizing particle ionizes the gas and the time the signal is detectable.

ComponentParallelPlate::Setup
::SetMedium
::AddPlane
::SetGeometry
::ElectricField
SolidBox
GeometrySimple::AddSolid
MediumMagboltz::LoadGasFile
::Initialise
::ElectronAttachment
::ElectronTownsend
::ElectronVelocity
Sensor::AddComponent
::AddElectrode

b. **(Learning Objective: Get to know** TrackHeed**)** Before simulating the electron avalanches, the primary electrons must be generated from the incident ionizing particle. Create an instance of TrackHeed, add the Sensor, and initialize your favorite particle (e.g. a muon with a momentum of around 100 GeV). You can simulate the primary electrons by a new track, preferably passing the gas gap head-on. Access the primary clusters via `GetClusters` and the electrons in each cluster via Cluster::`electrons`. Visualize one event by accessing the coordinates of the primary electrons.

TrackHeed::SetSensor
::SetParticle
::SetMomentum
::GetClusters
Cluster::electrons

c. **(Learning objective: Get to know** AvalancheMicroscopic & AvalancheGrid**)**
Now you know how to simulate the primary electrons. To simulate the electron avalanche, you need either an instance of AvalancheMicroscopic or AvalancheGrid. A mixed-use of both has been shown to work best with experimental results, which will be the goal of this task.

AvalancheMicroscopic::SetSensor
::SetTimeWindow
::AvalancheElectron
AvalancheGrid::SetSensor
::SetGrid
::ImportElectronsFrom
-AvalancheMicroscopic
::StartGridAvalanche
::Reset
Sensor::ExportSignal
::ClearSignal

i. Create an instance of both classes and add the Sensor to both. For the Grid method, you need to define a proper grid. For the microscopic method, you can set a time window for the electrons' integration, e.g., from 0 to a few nanoseconds.

ii. Simulate the electron avalanche first with the microscopic tracking by creating a new track from TrackHeed. Loop over the primary electrons and let them 'avalanche' using `AvalancheElectron`. After each `AvalancheElectron` call, import the electrons into the AvalancheGrid instance.

Continue the avalanche using `StartGridAvalanche`. The signal caught by the plane electrode can be exported from the Sensor instance.

iii. Sample 5 events and plot their signal evolution in a log-lin plot. What happens at the initial phase?

Comprehension: The signal is the electric displacement current, which occurs when charges move relatively to an electrode.

d. **(Learning Objective: Learn to simulate the efficiency (= detection probability), time resolution, and charge spectrum)**
Set a signal threshold (units are fC/ns = µA, A = Ampere) and sample 100 - 200 events. For each event, determine if the signal threshold is passed (efficiency = detection probability), at what time, and the charge (integral of electron signal). *Hint: Use appropriate methods from the* SENSOR*-class.*

SENSOR::ComputeThresholdCrossings
::GetThresholdCrossing
::IntegrateSignal
::GetSignal

i. Determine the detection probability based on the frequentist's method,

$$\epsilon := \frac{N_{\text{detected}}}{N_{\text{total}}} \tag{2}$$

(Optional) Which discrete statistical distribution describes this process best?

ii. Plot the time crossing distribution and determine the signal spread (standard deviation). Compare it to the theoretically derived result from a.

iii. Plot the charge distribution. (Optional) What charge threshold would you need to set to get the same detection probability as derived in d.i.?

You can redo this exercise for various voltages to get, e.g., the efficiency and time resolution as a function of the voltage.

e. **(Optional: Get to know** AVALANCHEGRIDSPACECHARGE**)** Consider replacing AVALANCHEGRID with AVALANCHEGRIDSPACECHARGE. Turn on the diffusion but turn off the space charge effect. Redo exercise d.i., i.e., determine the efficiency; what do you observe?

AVALANCHEGRIDSPACECHARGE
::EnableDiffusion
::EnableSpaceChargeEffect
::Set2dGrid
::SetSensor
::StartGridAvalanche

f. **(Optional: Get to know the space-charge effect)** Space charge effects influence the charge spectrum because the time evolution of the avalanches changes from exponential to sub-exponential (space-charge deceleration). Choose a coarse grid (around 10 µm) and simulate 100 electrons starting at the cathode. Plot the electron evolution as log-lin and explain what happens. How does the charge spectrum change if you account for this effect (qualitatively)?

AVALANCHEGRIDSPACECHARGE
::EnableSpaceChargeEffect
::ImportEllipticIntegralValues
::AvalancheElectron
::GetElectronEvolution

SIM2: Resistive Micromegas

DRD1 Gaseous Detectors School

November 2024

1 Introduction

In this session, we will simulate the response of the resistive Micromegas shown in Fig. 1. Resistive planes can be introduced to make detectors more robust by mitigating the destructive effects of discharges. Alternatively, they can improve spatial resolution by spreading the signal across the resistive layer, as exemplified by the near detector of T2K in Japan [1, 2].

The objective of this session is to construct a simulation of the signal response of the described detector for 150 GeV/c muons using Garfield++ [3]. This exercise will require integrating various concepts covered in previous simulation sessions and lectures [4, 5]. Emphasis will be placed on using time-dependent weighting potentials to calculate induced signals on electrodes in the presence of resistive materials, with W. Riegler's lecture on induced signals in particle detectors [6] serving as the primary reference.

2 Electric field of a Micromegas

For a Micromegas detector, the applied drift and amplification field is determined by the potential difference between the cathode, the woven mesh, and the anode (which, in our case, is the resistive layer). Given the geometry of the woven mesh, obtaining an analytical solution for the electric field is not feasible. Instead, we use a numerical technique called the Finite Element Method (FEM) to calculate it. Various open-source and commercially available FEM solver packages are suitable for this purpose. For this session, we will use the field map generated by COMSOL [7], which will be imported into the Garfield++ simulation. Both the geometry and its division into smaller finite elements are shown in Fig. 2. In Garfield++ the COMSOL solution is accessed through three files:

- *mesh.mphtxt*: contains the node positions and cell types that define the mesh.
- *Potential.txt*: holds the static potential values at each mesh node for the electrode biasing that creates the applied field to drift the charges.
- *dielectrics.dat*: assigns dielectric constants to each material.

Using the *ComponentCOMSOL* class, we can import this information into our simulation:

```
// Import COMSOL's potential, mesh and dielectric constant map
ComponentCOMSOL fmMesh;
fmMesh.Initialise("mesh.mphtxt", "dielectrics.dat", "Potential.txt", "mm");
fmMesh.EnableMirrorPeriodicityX();
fmMesh.EnableMirrorPeriodicityY();
fmMesh.PrintRange();
```

To assign the drift medium to the corresponding domains in the FEM model, we use the following commands:

```
// Setup of the gas
MediumMagboltz gas;
...

// Assign relative permittivity to geometry domains
const unsigned int nMaterials = fmMesh.GetNumberOfMaterials();
```

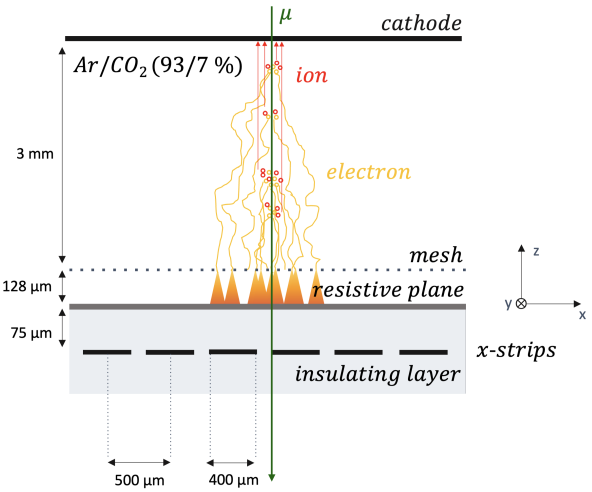


Figure 1: Cross-sectional schematic representation of a Micromegas with an infinitely expanding resistive plane and strip readout electrodes. Not drawn to scale.

```
for (unsigned int i = 0; i < nMaterials; ++i){
    const double eps = fmMesh.GetPermittivity(i);
    if(eps==1) fmMesh.SetMedium(i, &gas);
}

// Print all materials
fmMesh.PrintMaterials();

// Creating the sensor
Sensor sensor;
sensor.AddComponent(&fmMesh);
```

Here, the dots indicate parts of the code that still need to be filled in, such as the definition of the gas. With the FEM solution for the applied field imported, we will estimate the effective gain of the detector by introducing primary electrons in the drift gap.

2.1 Defining the gas mixture

TO-DO: Set the gas mixture to Ar/CO_2 93/7% at room temperature and atmospheric pressure with the appropriate Penning transfer rate [8]. How do you expect the detector's gain to vary with different values of the Penning transfer rate?

```
// For this use the functions:
// * MediumMagboltz::SetComposition
// * MediumMagboltz::SetTemperature
// * MediumMagboltz::SetPressure
// * MediumMagboltz::Initialise
MediumMagboltz gas;
...

// Set the Penning transfer efficiency.
// Use the function:
// * MediumMagboltz::EnablePenningTransfer
...
```

TO-DO: Approximate the ion mobility by using the measured values for Ar^+ in pure Ar. Import these values from the file "IonMobility_Ar+_Ar.txt" located in the folder *<GARFIELD_INSTALL>/share/Garfield/Data*.

```
// Load the ion mobilities.
// It points to the path where the table is saved.
const std::string path = std::getenv("GARFIELD_INSTALL");

// Set the ion mobility use the function:
// * MediumMagboltz::LoadIonMobility
```

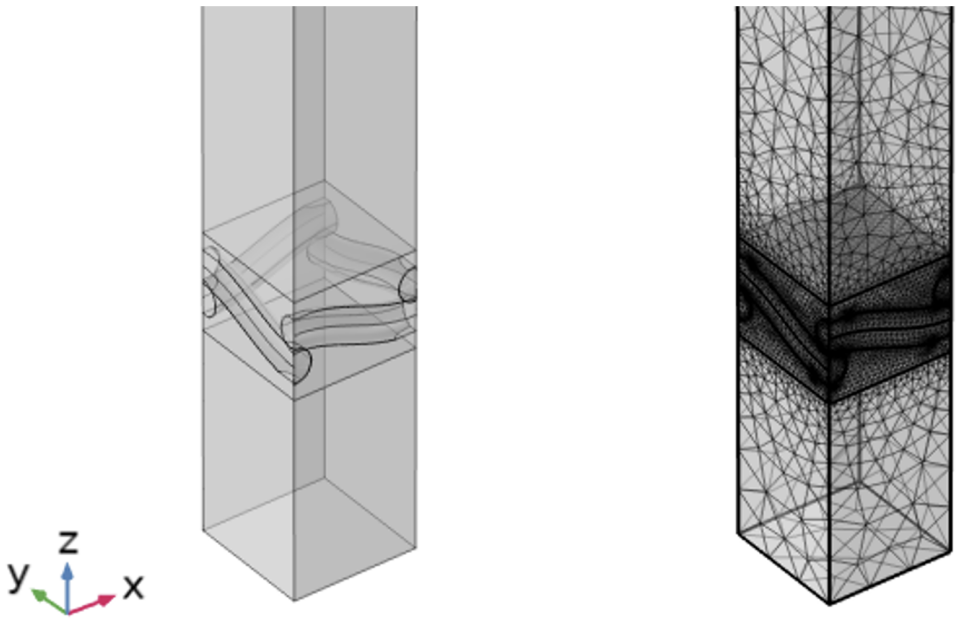


Figure 2: The Finite Element Model to calculate the applied electrical field in a Micromegas detector. **Left:** Geometry of the woven mesh suspended at $z = 128$ µm above the anode. **Right:** Partitioning the geometry into smaller mesh elements for solving the model.

```
...
```

2.2 Plotting the applied electrical field

TO-DO: Plot the amplification field between the mesh (center at z = 128 µm) and resistive layer (located at z = 0) in the xz-plane. For the plot range along the x-axis, use the mesh periodicity of 128 µm.

```
const double pitch = 128.e-4; // [cm]

ViewField fieldView;
constexpr bool plotField = true;
if (plotField) {
    // For this the function main function to be used is:
    // * ViewField::SetComponent or ViewField::SetSensor
    ...

    // Now the field of view still needs to be set. Find the function to do this.
    ...

    // Setting the canvas using the function
    // * ViewField::SetCanvas
    TCanvas* cf = new TCanvas("cf", "", 600, 600);
    ...

    // Plot the field using the command
    // * ViewField::Plot
    ...
}
```

TO-DO: Identify the region with the highest field strength.

2.3 Simulating gain for a Micromegas

To simulate the detector's gain, we need to perform a Monte Carlo simulation of the movement and amplification of charge carriers within the gas medium. For this, we will use both microscopic and

macroscopic approaches:

```
// Microscopic electron transport
AvalancheMicroscopic aval;
aval.SetSensor(&sensor);

// Macroscopic ion transport
AvalancheMC drift;
drift.SetSensor(&sensor);
drift.SetDistanceSteps(2e-5); // cm
```

TO-DO: Create N events where a single electron-ion pair is generated at rest at an initial position

$$\mathbf{x}_0 = (x_0, y_0, z_0 = 200\ \mu\text{m}),$$

with $x_0, y_0 \in [-pitch, pitch]$. Setting $N = 1$, plot the drift lines of the resulting charge carriers:

```
ViewDrift driftView;
const bool plotDrift = true;
ViewFEMesh meshView;
if (plotDrift) {
    // Use the following commands:
    // * AvalancheMicroscopic::EnablePlotting
    // * AvalancheMC::EnablePlotting
    ...
}

// Creating event
const int N = 1; // Number of events

for(int i = 1; i <= N; i++){
    // Drift electrons using:
    // * AvalancheMicroscopic::AvalancheElectron
    ...

    // Drift ions using:
    // * AvalancheMicroscopic::GetNumberOfElectronEndpoints
    // * AvalancheMicroscopic::GetElectrons
    // * AvalancheMC::DriftIon
    ...
}

if (plotDrift) {
    TCanvas* cd = new TCanvas();

    const bool plotMesh = true;
    if(plotMesh){
        // Color in mesh (having an eps_r =4)
        meshView.SetColor(4, kGray);
        // Find the other commands to overlay the plot pf the mesh on the drift lines
        ...
    }
}
```

TO-DO: For $N = 100$, give a first rough estimate of the effective gain, e.g., the average number of electrons reaching the resistive layer per event.

```
// Creating event
const int N = 100; // Number of events

for(int i = 1; i <= N; i++){
    // Drift electrons
    ...

    // For the effective some of these functions might be usefull:
    // * AvalancheMicroscopic::GetNumberOfElectronEndpoints
    // * AvalancheMicroscopic::GetElectronEndpoint
    // * AvalancheMicroscopic::GetElectrons
    // Note that you do not have to use them all!
    ...

    // Drift ions
```

```
    ...
}
```

OPTIONAL TO-DO: Make a histogram of the gain distribution.

3 Signal induction

Consider a resistive detector with K readout electrodes. Given a point charge carrier q traversing the gas medium along a trajectory $\mathbf{x}_q(t)$, it follows from the extended form of the Ramo-Shockley theorem for conductive media that the induced signal on electrode $k \in \{1, \ldots, K\}$ can be calculated using a so-called time-dependent weighting potential $\Psi_k(\mathbf{x}, t)$ of that electrode [9, 10, 11, 12]. In the quasi-static limit, we find that the induced current is given by

$$I_k(t) = -\frac{q}{V_w} \int_0^t \mathbf{H}_k \left[\mathbf{x}_q\left(t'\right), t - t'\right] \cdot \dot{\mathbf{x}}_q\left(t'\right) \, dt' \,, \tag{1}$$

where we have defined the weighting vector as

$$\mathbf{H}_k(\mathbf{x}, t) := -\nabla \frac{\partial \Psi_k(\mathbf{x}, t)\Theta(t)}{\partial t} \,. \tag{2}$$

In the quasi-static regime, when we neglect the propagation times of electric fields, the introduction of a step voltage $V_w\theta(t)$ on electrode k leads to an immediate potential $\Psi_k(x, 0)$ permeating the detector volume, from which point it is a smooth function of time. Furthermore, at $t = 0$ all conducting components exhibit insulating characteristics as they do not have sufficient time to react. Conversely, for $t \to \infty$, all resistive elements behave like perfect conductors. We can therefore say that $\Psi_k(x, t)$ is comprised of two components: the static *prompt* component $\psi_k^p(\mathbf{x}) := \Psi_k(x, 0)$, and the dynamic *delayed* component $\psi_k^d(\mathbf{x}, t)$, i.e.,

$$\Psi_k(\mathbf{x}, t) =: \psi_k^p(\mathbf{x}) + \psi_k^d(\mathbf{x}, t) \,, \tag{3}$$

where $\psi_k^d(\mathbf{x}, 0) = 0$ by definition. The first term gives the direct induction of current on the electrode due to the motion of a charge carrier within the medium, whereas the second term encompasses the reaction of the resistive materials. This we can see more clearly when we use the definition in Eq. (3), the weighting vector can be expressed as

$$\begin{aligned} \mathbf{H}_k(\mathbf{x}, t) &= -\nabla\psi_k^p(\mathbf{x})\delta(t) - \nabla \frac{\partial \psi_k^d(\mathbf{x}, t)\Theta(t)}{\partial t} \\ &=: \mathbf{E}_k^p(\mathbf{x})\delta(t) + \mathbf{H}_k^d(\mathbf{x}, t) \,, \end{aligned} \tag{4}$$

where $\mathbf{E}_k^p(\mathbf{x})$ will be the *prompt weighting field* and $\mathbf{H}_k^d(\mathbf{x}, t)$ the *delayed weighting vector*. Expressed in this way we can write the induced current as

$$I_k(t) = -\frac{q}{V_w}\mathbf{E}_k^p(\mathbf{x}_q(t)) \cdot \dot{\mathbf{x}}_q(t) - \frac{q}{V_w} \int_0^t \mathbf{H}_k^d \left[\mathbf{x}_q\left(t'\right), t - t'\right] \cdot \dot{\mathbf{x}}_q\left(t'\right) \, dt' \,, \tag{5}$$

where the first term is the relation found for the basic form of the Ramo-Shockley theorem, while the second term is the time-dependent contribution of the resistive media that this extension adds.

3.1 Using FEM solutions of time-dependent weighting potentials

At this point, we can start calculating the signals induced in three neighboring readout strips, labeled $k \in \{1, 2, 3\}$, and positioned at $x_1 = -0.5$ mm, $x_2 = 0$ mm, and $x_3 = 0.5$ mm. For simplicity, we approximate the mesh as a plane positioned at $z = 128$ µm.

TO-DO: The time-dependent weighting potential solution can be imported in the same way as the one for the woven mesh in Sec. 2. Do this for the central strip and make use of the symmetry of the system:

```
// Path to get the FEM map
std::string mapWPPath = ...;
// Load the field map for weighting potential
ComponentComsol fm;
fm.SetImportMaterial(1.); // Only importe domain that has the gas medium
// Now use the following functions to import the time-dependent weighting potential:
// * ComponentComsol::Initialise
// * ComponentComsol::EnableMirrorPeriodicityX
// * ComponentComsol::EnableMirrorPeriodicityY
...

const std::string label[3] ={"Strip1","Strip2","Strip3"};
// Load the weighting potential map called "WPotential.txt" using:
// * ComponentComsol::SetDynamicWeightingPotential
...

// Add electrode to the excisting sensor using:
// * Sensor::AddElectrode
Sensor sensor; // Already in your code...
sensor.AddComponent(&fmMesh);
...

// Include the dynamic weighting potential in the calculation:
// * Sensor::EnableDelayedSignal
...
```

TO-DO: Plot the weighting potential for different time slices. What happens in the limit $t \to \infty$?

```
// Plot the time-depentent weighting potential for a specific t using:
// * ViewField::PlotWeightingField
...
```

3.2 Induced signals in neighboring electrodes

TO-DO: For $N = 1$, calculate the induced signal on the central strip, including the delayed component. For this, you will need to set the times at which the delayed weighting vector will be evaluated during the signal calculation. Here, we will set it equal to the time slices of the imported dynamic weighting potential.

```
// Make induced signal histogram using:
// * Sensor::SetTimeWindow
const double tmin =  0.;
const double tmax = 400; // [ns]
const unsigned int nTimeBins  = 400; // Amount of time bins for the final signal
...

// Set times for evaluating the delayed weighting potential.
// We will take the time slices of the imported weighting potential maps using:
// * ComponentComsol::GetTimeInterval
// * Sensor::SetDelayedSignalTimes;
...
```

TO-DO: Plot the resulting induced current.

```
// Plot the signal using ViewSignal
...

const bool plotSignal = true;
TCanvas* cSignal1 = new TCanvas("cSignal1", "", 600, 600);
if (plotSignal) {
    // Set the canvas using:
    // * ViewSignal::SetCanvas
    ...
}

...

// Creating event
const int N = 1; // Number of events

// Generating an event and moving the charges
```

```
// Already done in Sec. 2.3
...

if (plotSignal) {
    // * ViewSignal::PlotSignal
    ...
}
```

TO-DO: Include the other two readout electrodes by coordinating the time-dependent weighting potential of the central strip. Plot their induced signals.

```
// Include the left and right readout strip using
// * ComponentComsol::CopyWeightingPotential
```

4 Muon detection

As a final step, we will use all of the above to generate the signal response of the resistive Micromegas for a muon crossing the detector perpendicular to the readout plane. For this, we use the interface with the C++ version of HEED [13].

4.1 TrackHeed

TO-DO: Generate the primary ionization pattern from a muon crossing the drift gap along the z-axis. Subsequently, calculate the resulting drift lines, amplification, and induced currents on the electrodes.

```
// Set up Heed and generate the track using the following functions:
// * TrackHeed::SetParticle
// * TrackHeed::SetMomentum
// * TrackHeed::NewTrack
// * TrackHeed::GetClusters

TrackHeed track;
track.SetSensor(&sensor);
...
```

5 Optional exercises

These are optional exercises that you can do after finishing the ones above. Up to now, we have only considered the ideal case where the induced current can be read out noiselessly directly from the electrode. However, in reality, different noise sources will contribute a fluctuating noise background, which is superimposed on the signals. As such, we will add noise to our signals, shape them with the delta response function of our readout electronics, and perform position reconstruction for the incident muon.

5.1 Adding white noise

TO-DO: Add white noise to the generated induced current with an equivalent noise charge (ENC) of your choosing.

```
// Adding white noise to the induced current throught:
// * Sensor::AddWhiteNoise
...
```

5.2 Signal shaping from front-end electronics

TO-DO: Shape the induced signal using a first-order unipolar shaper with a time constant of 50 ns and unitary gain.

```
// Using the Shaper class we can convolute the signal with the delta
// response function of the electonics with:
// * Sensor::SetTransferFunction
// * Sensor::ConvoluteSignal or Sensor::ConvoluteSignals
...
```

5.3 Position reconstruction

Given the induced charge Q_k on electrode $k \in \{1, \ldots, K\}$, a popular choice for the position reconstruction of an incident particle is the charge center-of-gravity (CoG) method:

$$x_r = \frac{\sum_k x_k Q_k}{\sum_k Q_k}\,, \tag{6}$$

where x_k is the c-coordinate of the center of the readout strip and x_r the reconstructed position.

TO-DO: Using the CoG method, reconstruct the x-position of the muon using the peak value of the shaped signal.

```
// The peak value of the signal after the shaper can be found using:
// * Sensor::GetSignal
...
```

References

[1] D. Attié et al., *Performances of a resistive micromegas module for the time projection chambers of the T2K Near Detector upgrade*, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **957** (2020) 163286.

[2] D. Attié et al., *Characterization of resistive Micromegas detectors for the upgrade of the T2K Near Detector Time Projection Chambers*, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **1025** (2022) 166109 [2106.12634].

[3] H. Schindler and R. Veenhof, *Garfield++*, *https://garfieldpp.web.cern.ch/garfieldpp* (2024) .

[4] P. Verwilligen, *Simulation and modelling 1*, *https://indi.to/wWcr2* (2024) .

[5] D. Janssens, *Simulation and modelling 2*, *https://indi.to/XZ4YP* (2024) .

[6] W. Riegler, *Signal induction*, *https://indi.to/JDLqX* (2024) .

[7] COMSOL® 6.1, *Software for multiphysics simulation*, *https://www.comsol.ch* (2024) .

[8] Özkan Şahin, T.Z. Kowalski and R. Veenhof, *High-precision gas gain and energy transfer measurements in Ar–CO_2 mixtures*, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **768** (2014) 104.

[9] E. Gatti, G. Padovini and V. Radeka, *Signal evaluation in multielectrode radiation detectors by means of a time dependent weighting vector*, *Nuclear Instruments and Methods in Physics Research* **193** (1982) 651.

[10] W. Riegler, *Induced signals in resistive plate chambers*, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **491** (2002) 258.

[11] W. Riegler, *Extended theorems for signal induction in particle detectors vci 2004*, *Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **535** (2004) 287.

[12] W. Riegler, *An application of extensions of the Ramo–Shockley theorem to signals in silicon sensors*, *Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **940** (2019) 453 [1812.07570].

[13] I. Smirnov, *Heed-C++*, *http://ismirnov.web.cern.ch/ismirnov/heed* (2024) .