FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

BACHELOR THESIS

Martin Vavřík

Simulation and Reconstruction of Charged Particle Trajectories in an Atypic Time Projection Chamber

Institute of Particle and Nuclear Physics

Supervisor of the bachelor thesis: Mgr. Tomáš Sýkora, Ph.D.

Study programme: Physics

Prague 2025

Dedication.

Title: Simulation and Reconstruction of Charged Particle Trajectories in an Atypic Time Projection Chamber

Author: Martin Vavřík

Institute: Institute of Particle and Nuclear Physics

Supervisor: Mgr. Tomáš Sýkora, Ph.D., Institute of Particle and Nuclear Physics

Abstract: Abstract.

Keywords: key words

Contents

Introduction

Time Projection Chamber (TPC) is a type of gaseous detector that detects charged particle trajectories by measuring the position and drift time of ions created in the gas; details are provided in Section 1. The energy of such particles can be determined thanks to the curvature of their trajectory in the magnetic field.

The goal of this thesis is to develop an algorithm for the reconstruction of charged particle trajectories and energy in an atypic TPC (with orthogonal electric and magnetic fields, i.e., Orthogonal Fields TPC (OFTPC)) used in the X17 project at the Institute of Experimental and Applied Physics, Czech Technical University in Prague (IEAP CTU). Furthermore, we present the results of testing this algorithm with different samples of simulated data. In the future, we also plan to test this algorithm by measuring real particles with a known energy distribution. In order to achieve this, we use the Garfield++ toolkit [1] in combination with the ROOT framework [2]. Some of our more demanding simulations are run on MetaCentrum.

The X17 project in IEAP CTU aims to reproduce measurements of anomalous behavior in the distribution of angular correlation of pairs produced by the Internal Pair Formation (IPF) mechanism during the decay of certain excited nuclei (^{8}Be, ^{12}C, and ^{4}He) observed by the ATOMKI group in Hungary.

Add citations: MetaCentrum, X17 project, VdG, ATOMKI papers. Maybe also TPC, IPF, etc.

0.1 ATOMKI Measurements

Short summary of results of measurements in ATOMKI.

0.2 X17 Project at IEAP CTU

Short summary of our goals, maybe mention the grant.

0.2.1 Our Detector

Short description of our detector. Why we use an atypic TPC. Gas mixture used in the detector (70/30) and its effect.

0.2.2 Coordinate System

Description of the coordinate system used in this thesis (+ figures). Introduce the detector space $\mathcal{D}$, the readout space $\mathcal{R}$ and the pad space $\mathcal{P}$. TPC dimensions, first sector, pad layout.

Detector Space

The detector space $\mathcal{D}$ represents the physical space of our detector. We describe it using coordinates (x, y, z). The z-axis is the detector's axis of symmetry, with its negative direction aligned with the proton beam. The origin $(0, 0, 0)$ is located at the center of the irradiated target. The positive x-axis passes through the center of one the OFTPCs along the intersection of its two planes of symmetry. The y-axis is then chosen such that the coordinate system is right-handed.

Since the detector has sixfold symmetry, we use only one of its sectors in this work – the first sector $\mathcal{D}_1 \subset \mathcal{D}$. This sector contains one of the six OFTPCs and is defined by the condition:

$$(x, y, z) \in \mathcal{D}_1 \Leftrightarrow |y| < x \tan \frac{\pi}{6}. \tag{1}$$

Simulations in this sector can be applied to all sectors by rotating the coordinates accordingly. The volume of the OFTPC in this sector, which has the shape of a trapezoidal prism, has these boundaries:

$$x \in [x_{\min}, x_{\max}] = [6.51, 14.61] \text{ cm}, \tag{2}$$

$$z \in [z_{\min}, z_{\max}] = [-8, 8] \text{ cm}, \tag{3}$$

$$y_{\max}(x_{\min}) = -y_{\min}(x_{\min}) = 2.75 \text{ cm}, \tag{4}$$

$$y_{\max}(x_{\max}) = -y_{\min}(x_{\max}) = 7.45 \text{ cm}, \tag{5}$$

where $y_{\max}(x)$ is the maximal value of the y-coordinate for a given x. The readout is located at $z = 8$ cm; for some purposes, we also define the distance to the readout $d_r = 8 \text{ cm} - z$ as an alternative to the z-coordinate.

Readout Space

The readout space $\mathcal{R}$ represents the drift time and final position of ionization electrons as measured by an ideal continuous readout. We describe it using coordinates (x', y', t), where x' and y' correspond to the detector coordinates at the readout plane ($z = 8$ cm).

Pad Space

The pad space $\mathcal{P}$ represents the time bin and pad number of ionization electrons as measured by an ideal discrete readout.

The readout of the OFTPC will consist (is the design final?) of 128 rectangular pads arranged in a staggered pattern (add image where all the parameters are marked). Most of the pads are 0.6×0.9 cm, only pads 102 and 124 are 0.6×0.6 cm, pad 127 is 0.6×0.509 cm. The distance of neighboring pads is 0.08 cm, staggering offset is 0.3946 cm.

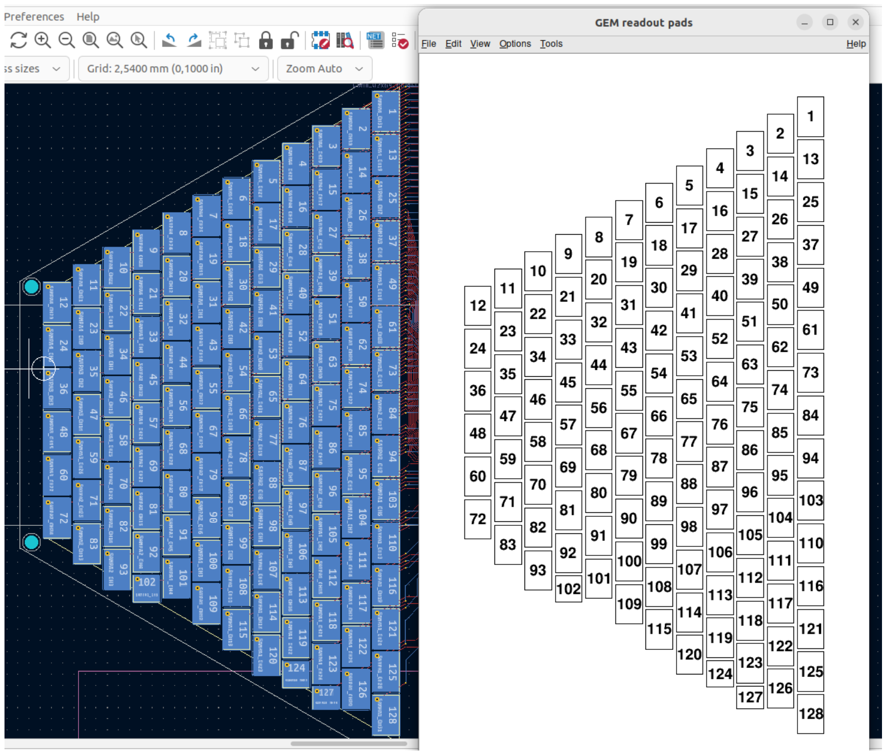


Figure 1: Pad layout of the TPC. Swap for better image.

0.2.3 Magnetic Field Simulation

Magnetic field simulations in Maxwell. Some figures. When working with the magnetic field outside the regular grid, we use trilinear interpolation.

Trilinear Interpolation

Trilinear interpolation is a generalization of linear interpolation in 3D. It can be used to interpolate a function whose values are known on a regular grid. We use this simple method for interpolating the magnetic field, and it is also used in Section 3.2.1 to interpolate the Ionization Electron Map. In both cases, we use a cubic grid.

Let us consider a cube (a cell of our regular grid) with an edge of length a containing the point $C = (x, y, z)$ where we want to interpolate a function $f\colon \mathbb{R}^3 \to X$ (it should be better explained what X is). We know the values of this function on the vertices of this cube $C_{ijk} = (x_0 + ia, y_0 + ja, z_0 + ka)$, where $i, j, k \in \{0, 1\}$. We also define the points $C_{ij} = (x, y_0 + ia, z_0 + ja)$ and $C_i = (x, y, z_0 + ia)$. Then the interpolated value $\widehat{f}(C)$ can be calculated as a composition of three linear interpolations:

$$x_d = \frac{x - x_0}{a}, \quad y_d = \frac{y - y_0}{a}, \quad z_d = \frac{z - z_0}{a}, \tag{6}$$

$$\widehat{f}(C_{ij}) = (1 - x_d) f(C_{0ij}) + x_d f(C_{1ij}), \tag{7}$$
$$\widehat{f}(C_i) = (1 - y_d) \widehat{f}(C_{0i}) \; + y_d \widehat{f}(C_{1i}), \tag{8}$$
$$\widehat{f}(C) = (1 - z_d) \widehat{f}(C_0) \; + z_d \widehat{f}(C_1). \tag{9}$$

We can also write

$$\widehat{f}(C) = \sum_{i,j,k\in\{0,1\}} t_x^i t_y^j t_z^k f(C_{ijk}), \tag{10}$$

$$t_a \overset{\text{def}}{=} \begin{pmatrix} t_a^0 \\ t_a^1 \end{pmatrix} = \begin{pmatrix} 1 - a_d \\ a_d \end{pmatrix}, \quad a \in \{x, y, z\}, \tag{11}$$

or as a polynomial:

$$\widehat{f}(C) = \sum_{a,b,c\in\{0,1\}} \sum_{i=0}^{a} \sum_{j=0}^{b} \sum_{k=0}^{c} (-1)^{(a-i)+(b-j)+(c-k)} f(C_{ijk}) x_d^a y_d^b z_d^c. \tag{12}$$

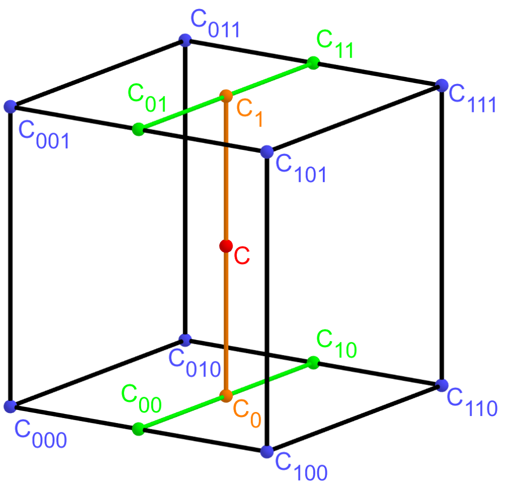


Figure 2: Visualization of trilinear interpolation as a composition of linear interpolations. Image drawn in GeoGebra and inspired by a similar image on Wikipedia (looks a bit worse) – is credit necessary?

Maybe a citation here, although I am not sure it is necessary since it could be considered common knowledge. Maybe x_0, etc. should be explicitly described.

1. Time Projection Chamber

Description of TPC, working principle, standard vs. our field layout.

2. Track Simulation

In order to develop and test the reconstruction algorithm, electron and positron tracks are simulated inside our detector with different initial parameters. Three approaches are used to simulate tracks for different purposes.

The **Microscopic Simulation** uses the Garfield++ toolkit [1]. Within this toolkit, the High Energy Electro-Dynamics (HEED) program [3] is used to simulate the primary particle and the class *AvalancheMicroscopic* to simulate the drift of secondary electrons created by ionization in the gas. This is the most precise and time-consuming simulation used; our current goal is to be able to successfully reconstruct its results and determine our best-case energy resolution.

The **Runge-Kutta Simulation** uses the 4th order Runge-Kutta numerical integration (add citation for Runge-Kutta) to simulate the trajectory of the primary particle in the electromagnetic field inside the detector. It is relatively fast since it does not simulate the secondary particles. It is used as part of our reconstruction algorithm and for testing some parts of the reconstruction.

The **Fast Simulation with Ionization Electron Map** is planned for the future. It will use the HEED program [3] to simulate the primary particle and the Ionization Electron Map (see Section 3.2) to simulate the drift of secondary electrons. It should be significantly faster than the Microscopic Simulation but offer comparable precision since it will rely on an already simulated drift map.

All of these simulations require the knowledge of the electromagnetic field inside the detector. A uniform electric field of 400 $\text{V}\cdot\text{cm}^{-1}$ is assumed. The magnetic field was simulated in Maxwell (add citation? details? own subsection with figures? more details in Section 0.2?).

Could also mention mention Monte Carlo (requires gas file generation) and Runge Kutta simulation implemented in Garfield, why we don't use them. Single track in positive x direction or initial parameter randomization. Importance of gas composition, used gas compositions.

2.1 Microscopic Simulation

The microscopic simulation, the most detailed simulation used in this work, is performed using the Garfield++ toolkit [1] for this purpose.

The electron transport properties are simulated using the program Magboltz (Add citation.). Two different gas mixtures were used: 90% Ar + 10% CO_2 and 70% Ar + 30% CO_2. The second mixture will be used in our detector. The temperature is set to 20 °C, the pressure is atmospheric.

The primary track is simulated using the program HEED [3], which is an implementation of the photo-absorption ionization model. This program provides the parameters of ionizing collisions. HEED can also be used to simulate the transport of delta electrons; we do not account for these in the current simulation but plan to include them in the future. The photons created in the atomic relaxation cascade (fluorescence reabsorption, ?) are also not simulated.

Finally, we use the microscopic tracking provided by the class *AvalancheMicroscopic* to simulate the drift of the ionization electrons. Each electron is followed from collision to collision using the equation of motion and the collision rates calculated by Magboltz.

First simulated track in the z direction. Figures.

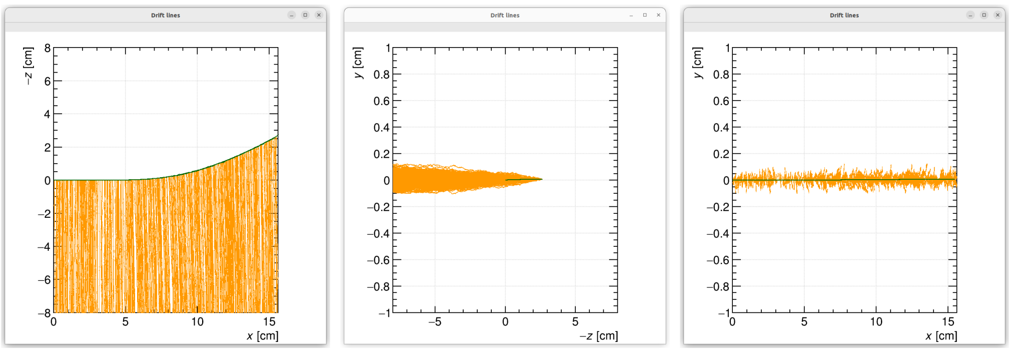


Figure 2.1: Example of a simulated electron track in 70 % argon and 30 % CO_2 atmosphere (on the left). Swap for better images, better zoom. Explain drift lines, primary particle.

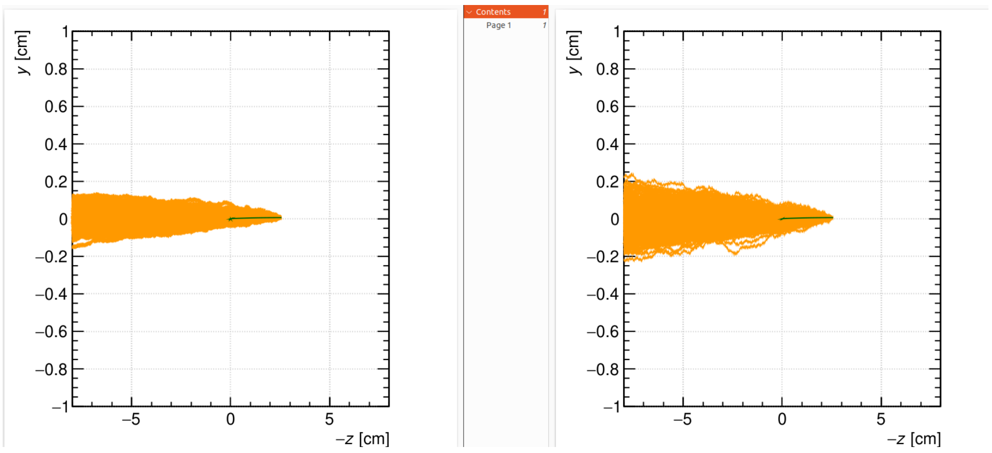


Figure 2.2: Comparison of diffusion in a simulated electron track in 70 % argon, 30 % CO_2 atmosphere and in 90 % argon, 10 % CO_2 atmosphere (on the right). Swap for better image, better zoom. Or put the same pictures for both comparisons in one subfigure, etc. Describe better.

2.2 Runge-Kutta Simulation

Trajectory simulation with 4th order Runge-Kutta. Relativistic equation that is numerically integrated by the algorithm.

2.3 Future?: Fast Simulation with the Ionization Electron Map

Primary track simulated in HEED. Readout parameters by interpolating the map. Diffusion from the map for randomization.

3. Track Reconstruction

The first stage of our reconstruction algorithm is the reconstruction of the track of the primary particle (either an electron or a positron). The results of this step are then used to determine the energy of the particle (see Section 4).

First Attempts at a track reconstruction were made using the standard approach. Here, we assume that the readout coordinates (x', y', t) are known exactly, neglecting the pads and time bins. In a standard TPC (with parallel fields), we only need to reconstruct the z coordinate from drift time using the known drift velocity.

Reconstruction using the **Ionization Electron Map** (from now on referred to as *the map*) uses a simulation of the drift of secondary (ionization) electrons within the detector volume. This simulation can then be used to interpolate the initial position of the secondary electrons. First attempts neglect the pads.

We used the map for reconstruction in two different ways. The first one uses gradient descent search along with trilinear interpolation (see Section 0.2.3) in the map. The second method uses interpolation in the irregular inverse grid with a linear polynomial.

The **Discrete Reconstruction** uses the map; instead of reconstructing the exact position of each electron, we reconstruct the center of each hit pad with the time corresponding to the midpoint of the time bin. The number of electrons in each TPC bin (consisting of the pad and the time bin) is counted and used as a charge in the energy reconstruction.

3.1 First Attempts

As the first step, we decided to try to reconstruct an electron track with a special set of initial parameters. The origin of the particle is given by the origin of our coordinate system. The initial direction is given by the positive x axis. This means the magnetic field of our detector is perpendicular to the momentum of our particle at all times, and we can reduce the problem to two-dimensional space. We use a track simulated using the microscopic simulation (see Section 2.1) with a kinetic energy of 8 MeV. The gas composition used in this simulation is 90% Ar + 10% CO_2.

In this first approach to the reconstruction of the track, we decided to use the common method used in a standard TPC. This will allow us to explore the significance of the atypical behavior in our OFTPC. Additionally, we assume the readout is continuous to further simplify the problem. In this approximation, we reconstruct the initial position of each ionization electron.

The reconstruction is then defined by the following relations between the coordinates of the detector space and the readout space (see Section 0.2.2):

$$x = x', \tag{3.1}$$

$$y = y', \tag{3.2}$$

$$z = v_d t, \tag{3.3}$$

where v_d is the drift velocity of electrons in the given gas mixture. At a phenomenological level, this velocity can be considered a function of the electric field $\boldsymbol{E}$ and the magnetic field $\boldsymbol{B}$:

$$v_d = v_d(\boldsymbol{E}, \boldsymbol{B}). \tag{3.4}$$

Taken from Garfield user manual. The Garfield++ toolkit uses this fact to accelerate their drift simulation with non-microscopic approaches. Since we assume a uniform electric field in our detector and we want to neglect the effect of our unusual magnetic field, we consider the drift velocity to be constant in this scenario. We then approximate this velocity by fitting the dependence $z(t)$ taken from the simulated ionization electrons. This is in one of the provisional figures. Also, this description is not completely accurate; in reality, we fit t1:8-y0 with a1*x+a0 and then invert this and use 8-y0 = b1*t1+b0 (old coordinates); b1=1/a1 functions as the drift velocity. Maybe also define this 8-z variable as an alternative to z in Section 0.2.2 and then use it when correcting this.

Later, in a commit after this, I plotted some residues (provisional figure), which could be useful, but for some reason they are residuals from a spline fit of the track?! Probably redo this without the spline fit; just explore the difference in individual points.

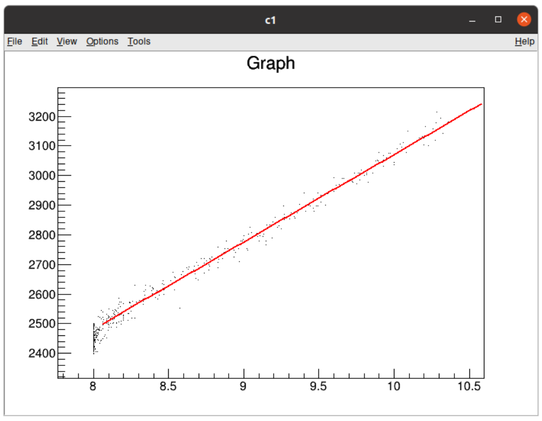


Figure 3.1: Dependence of the drift time on the z coordinate in 90 % argon and 10 % CO_2 atmosphere, fitted with a linear function. The fitted function gives us the average drift velocity in the gas and can be used for rough reconstruction in our TPC. Swap for better image with axis labels, etc. Maybe write the fitted equation.

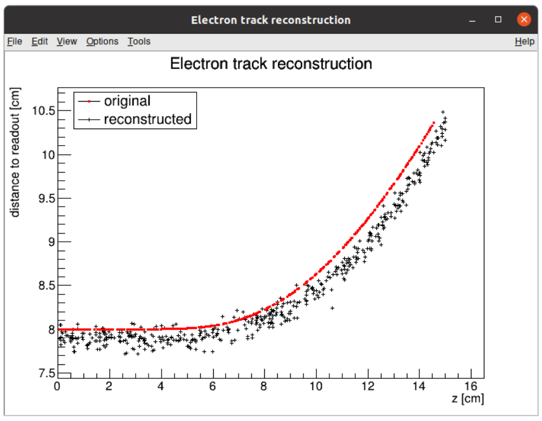


Figure 3.2: First attempt at a track reconstruction using only the drift velocity. This approach works well in a standard TPC (ideally cite some source?). 90 % argon and 10 % CO_2 atmosphere. Swap for better image, correct coordinates.

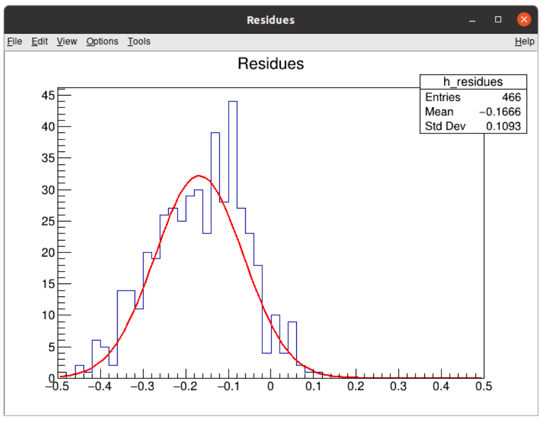


Figure 3.3: First attempt at a track reconstruction using only the drift velocity, residues. Swap for better image, correct coordinates. What's causing the shift? Explain details.

3.2 Ionization Electron Map

Inside an OFTPC, the drift of the secondary (ionization) electrons is significantly affected by its magnetic field. We need to take this into account for accurate reconstruction. In the first approximation, we assume a continuous readout (i.e., we neglect pads). We can then reconstruct the original position of each ionization electron using its readout coordinates. For this purpose, we use the ionization electron map.

The ionization electron map is a mapping from the detector space to the readout space. It tells us what readout coordinates (x', y', t) we can expect on average for an ionization electron created in the detector coordinates (x, y, z). More precisely it is a mapping to the distributions on the readout space; we can simplify this as only the means $\overline{\mathcal{M}}$ and the covariance matrices $\mathcal{M}_{\text{cov}}$, assuming Gaussian distribution.

$$\mathcal{M} : \mathcal{D} \longrightarrow \mathcal{R}, \ (x, y, z) \longmapsto (x', y', t). \tag{3.5}$$

To get an approximation of this mapping, we simulate the drift of ionization electrons generated on a regular grid inside the volume of our OFTPC (actually, we do not take the detector walls into account and simulate even outside of the OFTPC). It is also useful to simulate multiple (100 in our case) electrons originating from the same position so we can get a better approximation of the average drift and its variance. In order to get accurate results, we use the microscopic simulation of these electrons described in Section 2.1. When evaluating the map inside the grid, we use trilinear interpolation (see Section 0.2.3). From now on, we will denote this interpolated simulation with the same symbol $\mathcal{M}$.

Finally, we need to invert the map to get the original detector coordinates (x, y, z) for the given readout coordinates (x', y', t). In our case, we can reasonably assume that the mapping $\overline{\mathcal{M}}$ is one-to-one (as seen in the simulations). We implemented two methods for this purpose: the gradient descent search (Section 3.2.1) and interpolation in the inverse grid (Section 3.2.2).

If we wanted to further improve this, taking into account the whole map $\mathcal{M}$, we could make an 'inverse map' from $\mathcal{R}$ to distributions on $\mathcal{D}$. We could probably achieve this by taking the normalized probability density of an electron with initial coordinates (x, y, z) having readout coordinates (x', y', t). If we fix (x', y', t), we get an unnormalized probability density $f(x, y, z) = \mathcal{M}_{(x,y,z)}(x', y', t)$ (assuming that all initial coordinates are a priori equally likely). This could potentially improve the discrete reconstruction if we take the mean value of this probability density across the pad and time bin

$$f_{\text{pad, bin}}(x, y, z) = \frac{1}{A_{\text{pad}} \Delta t_{\text{bin}}} \int_{\text{pad, bin}} \mathcal{M}_{(x,y,z)}(x', y', t) \mathrm{d}x' \mathrm{d}y' \mathrm{d}t \tag{3.6}$$

and using it for a likelihood fit instead of using least squares. This still assumes that all initial coordinates are equally likely which is clearly not the case for a primary particle track. In the future, we could even use the fast track simulation with the map (should be possible to make around 1000 tracks per minute per core with current settings), create a big set of tracks with reasonable parameters and use these to get an approximation of the probability distribution of the detector response. Some approximations would be necessary when interpreting the data to decrease the degrees of freedom of this distribution (we would have to pick a set of parameters and assume that some of them are independent). This could give us an idea about the best achievable resolution (how significantly will the detector response differ for a given change in energy). If the difference is significant, we could try to further improve the likelihood fit.

The simulation of the map is a computationally heavy task. For this reason, we use a grid MetaCentrum (citation) to parallelize it. At first, this was done by evenly distributing the simulated electrons across the individual jobs. This was used in the first simulation with only one electron per vertex in the regular grid with the spacing of one centimeter.

Later, a better approach was implemented, accounting for the varying lengths of the drift of individual electrons. If we index the electrons in the order of increasing coordinates y, x, z (picture?), we can express the number n_l of full XY layers (i.e., electrons with the same z coordinate) of electrons with index less than or equal to i

$$n_l(i) = \left\lfloor \frac{i}{n_{xy}} \right\rfloor, \tag{3.7}$$

where n_{xy} is the number of electrons in each XY layer calculated simply by counting the electrons that satisfy boundary conditions for x and y. These conditions should be mentioned above; sector condition + maximal x value. The number of electrons remaining in the top layer is then

$$n_r(i) = i \bmod n_{xy}. \tag{3.8}$$

Finally, we can calculate the sum of the drift gaps of electrons up to index i

$$d_{\text{sum}} = (z_{\max} - z_{\min}) n_{xy} n_l - \frac{n_l(n_l - 1)}{2} n_{xy} l + n_r (z_{\max} - z_{\min} - n_l l). \tag{3.9}$$

We then use a binary search algorithm to find the maximum index i such that the value of this sum is less than the fraction $\frac{\text{job id}}{\text{max job id}}$ of the total sum. This way we obtain the minimal and the maximal index of electrons simulated in the given job. The spacing l should be probably defined above + picture of the simulating grid (1 layer). zmin zmax also

After the simulation of the map, we calculate the mean readout coordinates assuming Gaussian distribution (i.e., we use averages). We also calculate standard deviations in a later commit, should be upgraded to the covariance matrix. We never actually plotted the distributions we get when simulating the same electron multiple times, so we do not know if our assumptions are accurate (could also run some statistical test to see how well the Gaussian distribution fits).

The map is then stored in a *Field*, a custom class template, could expand on that. Maybe earlier, since the same template is used for the magnetic field.

Could insert a table here describing all 4 simulations of the map (gas composition, spacing, etc.). Simulation inside one sector (at first double angle). Extra space on the sensor. Edge cases not taken into account (TPC wall). Using qsub (not sure if important). Add plots of distortion of the coordinates. Could also do these plots in a different way (e.g., drawing all the endpoints of each ionization electron or some error ellipse plot).

Images to add (comparison of both simulations):

- 3D visualization of the map, simulation example
- z vs. t plot
- XY plane distortion for different z values; with arrows and error bars, for all z-layers with different colors
- XZ plane ($y = 0$) distortion in x (maybe not necessary?)
- XT plot ($y = 0$) showing (small) distortion in drift times

More images:

- Residuals of the continuous readout reconstruction.

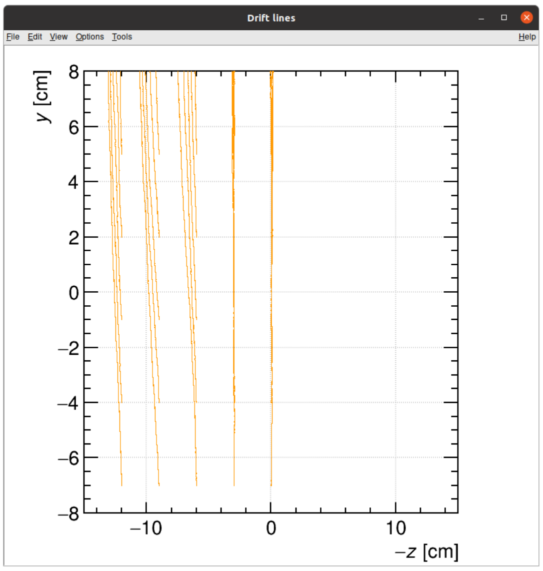


Figure 3.4: Example of map generation. Swap for better image, correct coordinates.

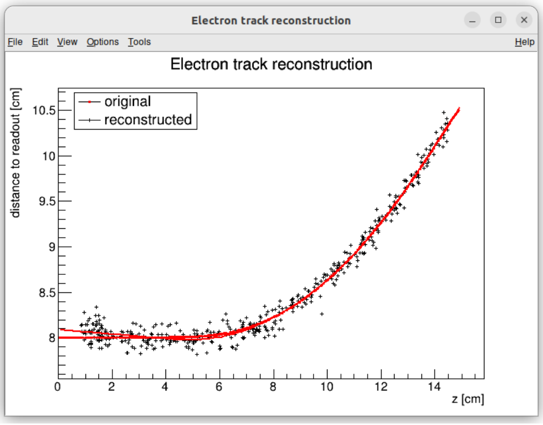


Figure 3.5: Example reconstruction with the map. Swap for better image, correct coordinates.

329 ### 3.2.1 Gradient Descent Search

The first implemented method of reconstruction uses a gradient descent search to find an approximation of the point in detector space with given mean readout coordinates. Gradient descent is an iterative minimization algorithm for multivariate functions. Let $R \in \mathcal{R}$ be a point in the readout space; we want to find a point $D = (x, y, z) \in \mathcal{D}$ in the detector space such that

$$\overline{\mathcal{M}}(D) = R = (x'_R, y'_R, t_R). \tag{3.10}$$

We define a function f_R in the readout space as a modified distance in this space:

$$f_R(x', y', t) = \sqrt{(x' - x'_R)^2 + (y' - y'_R)^2 + v_d^2(t - t_R)^2}, \tag{3.11}$$

where v_d is an approximation of the drift velocity in the TPC, obtained from the reconstruction in Section 3.1 (there will be an image with the linear fit there).

We make the initial guess (actually in the original code we just take $z = 0$):

$$D_0 = (x'_R, y'_R, v_d t). \tag{3.12}$$

Assuming we have the n-th estimate D_n, we calculate an approximation of the i-th component of the gradient of $f_R \circ \overline{\mathcal{M}}$:

$$\nabla(f_R \circ \overline{\mathcal{M}})^i \approx \frac{f_R(\overline{\mathcal{M}}(D_n + s \cdot e^i)) - f_R(\overline{\mathcal{M}}(D_n - s \cdot e^i))}{2s}, \tag{3.13}$$

where s is a number and $e^i \in \mathcal{D}$ is the i-th coordinate vector. The number s should be sufficiently small; initially, we set it as a fraction of the map's grid spacing $s = \frac{l}{10}$. During the minimization, we check that $f_R(\overline{\mathcal{M}}(D_n)) < 10s$ at all times. When using trilinear interpolation, it would be more efficient to calculate the gradient explicitly ($\pm$ same result). This could be implemented inside the *Field* template class. The next estimate can be calculated as follows:

$$D_{n+1} = D_n - \gamma \nabla(f_R \circ \overline{\mathcal{M}})(D_n), \tag{3.14}$$

where $\gamma \in \mathbb{R}^+$ is the damping coefficient. It should be set to a small enough value to ensure convergence, but large enough for sufficient converging speed. The minimization stops either when the error $f_R(\overline{\mathcal{M}}(D_n))$ drops below a specified value or when the number of iterations exceeds 1000 (in this case, a message is printed into the console). The parameters of this method could be further optimized (e.g., a better choice of γ, gradient computation); instead, we later decided to use the interpolation in the inverse grid described in the next section.

Measure reconstruction duration and compare it with the inverse grid interpolation? Also compare the result? Not sure if this has to be cited.

3.2.2 Interpolation in the Inverse Grid

Interpolating between known points in the readout space. Gaussian elimination, multivariate polynomial. Benefits compared to the gradient descent search method (one-time computation for the whole map is easy to achieve if needed).

The currently used reconstruction method is the interpolation in the inverse grid. Rather than attempting to invert the trilinearly interpolated map as in the previous section, we take advantage of the fact that the map $\overline{\mathcal{M}}$ is one-to-one. Since we have simulated values of this map on a regular grid in the detector space $\mathcal{D}$, we also know the inverse map $\overline{\mathcal{M}}^{-1}$ on the irregular inverse grid in the readout space $\mathcal{R}$. To get an approximation of the inverse map in the entire readout space, we can use interpolation.

Since the inverse grid is irregular, we cannot use trilinear interpolation. Since the simulated map is dense enough to give us a good approximation considering the size of our pads, we can use a similar approach (more complicated and computationally heavy alternative would be the natural neighbor interpolation). As shown in the equation 12 in Section 0.2.3, trilinear interpolation can be expressed as a polynomial:

$$\widehat{f}(x, y, z) = axyz + bxy + cxz + dyz + ex + fy + gz + h, \tag{3.15}$$

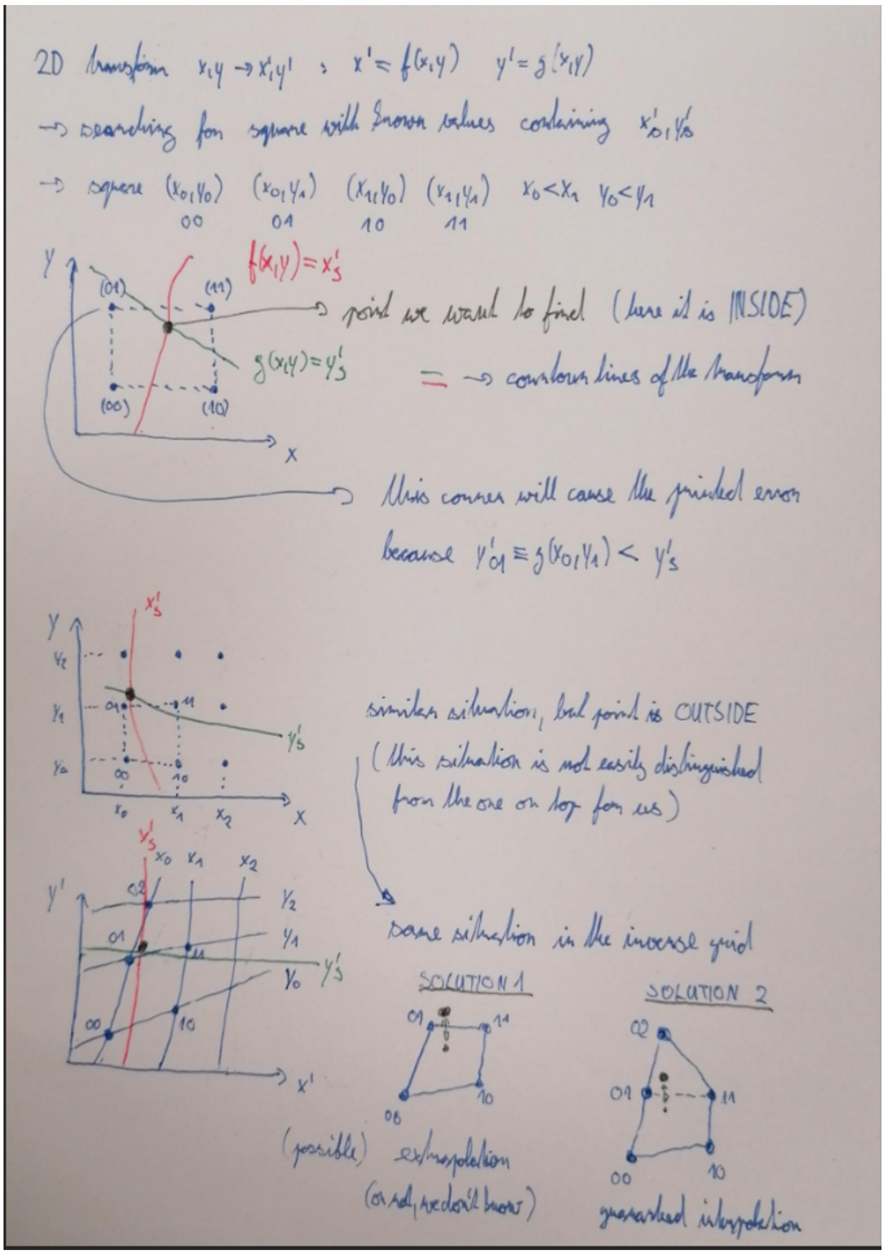
where a, b, c, d, e, f, g, h are coefficients determined uniquely by the values of the function on the vertices of the interpolation cube. We can generalize this for a function known on an irregular grid. If we know the value of a function in any eight points, we can take a system of eight linear equations

$$\begin{pmatrix} x_1y_1z_1 & x_1y_1 & x_1z_1 & y_1z_1 & x_1 & y_1 & z_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_8y_8z_8 & x_8y_8 & x_8z_8 & y_8z_8 & x_8 & y_8 & z_8 & 1 \end{pmatrix} \begin{pmatrix} a \\ \vdots \\ h \end{pmatrix} = \begin{pmatrix} f(x_1, y_1, z_1) \\ \vdots \\ f(x_8, y_8, z_8) \end{pmatrix}, \qquad (3.16)$$

which gives us a unique solution for the coefficients for most values of (x_n, y_n, z_n) and $f(x_n, y_n, z_n)$, where $n \in \{1, \ldots, 8\}$.

Using this approach introduces a small problem: finding the correct pseudocell (i.e., the image of eight vertices forming a cubic cell in the regular grid) in the inverse grid. The eight irregularly spaced vertices of this pseudocell do not define a unique solid body, so there is no straightforward way to divide $\mathcal{R}$ into pseudocells using only these vertices.

We are currently ignoring this problem and performing binary search along x, y, z (in this order). It shouldn't matter too much because the 70/30 map doesn't cause such a big distortion and was even accidentally extrapolated for all z different from the central plane. Interpolation should be generally faster than the gradient descent since we don't need to iterate. We also don't need to optimize it to improve performance, if it's too slow we can even calculate the coefficients for the entire map before reconstruction.


2D transform $x,y \rightarrow x',y'$: $x' = f(x,y)$ $y' = g(x,y)$

→ searching for square with known values containing x'_0, y'_0

→ square (x_0,y_0) (x_0,y_1) (x_1,y_0) (x_1,y_1) $x_0<x_1$ $y_0<y_1$

00 01 10 11

point we want to find (here it is INSIDE)

= → contour lines of the transform

This corner will cause the printed error because $y'_{01} \equiv g(x_0,y_1) < y'_s$

similar situation, but point is OUTSIDE (this situation is not easily distinguished from the one on top for us)

same situation in the inverse grid

(possible) extrapolation (or not, we don't know)

guaranteed interpolation

Figure 3.6: Selection of the points for interpolation. Create better images; use the explanation interpolation vs. extrapolation strange property. Solution 2 probably does not make much sense.

3.3 Discrete Reconstruction

Reconstruction with pads and time bins. Maybe testing different pads. Mapping the center of the pad (along with the midpoint of the time bin) isn't necessarily the best approach since it might not correspond to the average parameters of an electron with these readout parameters (insignificant?).

It is also possible to make this a subsection of the map, making the previous subsections parts of a new subsection 'Map Inversion'.

4. Energy Reconstruction

The second stage of our reconstruction algorithm is the reconstruction of the particle's energy using its reconstructed track (see Section 3). We can achieve this by fitting the track and extracting the needed parameters of the trajectory. We have tested three ways of reconstructing the energy. Fitting is done using the MINUIT algorithm implemented in ROOT [2]. Maybe cite some CERN article directly on MINUIT?

The **Cubic Spline Fit** is a rejected attempt at the reconstruction of energy. It uses smoothly connected piecewise cubic polynomials between uniformly spaced nodes. Energy can then be computed using the fit parameters by computing the radius of curvature in different points of the fitted curve using the known magnitude of the magnetic field perpendicular to the trajectory. This approach was rejected because tuning the fit to have a reasonably stable radius of curvature is unpractical.

The **Circle and Lines Fit** was chosen as an alternative since this corresponds to the shape of a trajectory of a charged particle crossing a finite volume with a homogeneous magnetic field. The energy of the particle can be estimated using the fitted radius and the magnitude of the perpendicular magnetic field in the middle of the TPC.

The **Runge-Kutta Fit** uses the 4th order Runge-Kutta numerical integration described in Section 2.2. Initial parameters of the track (including the particle's energy) are optimized so that the integrated trajectory fits to the reconstructed one. This fit can also be performed as a single parameter (i.e., energy) fit if we can get the initial position and orientation of the particle on the entrance to the TPC from previous detectors (Timepix 3 (Tpx3) and Multi-Wire Proportional Chamber (MWPC), see Section 0.2).

4.1 Cubic Spline Fit

The first attempt to get an early estimate of the kinetic energy of the particle uses a cubic spline fit. This approach was later rejected in favor of the circle and lines fit described in Section 4.2. We use an electron track starting in the origin of

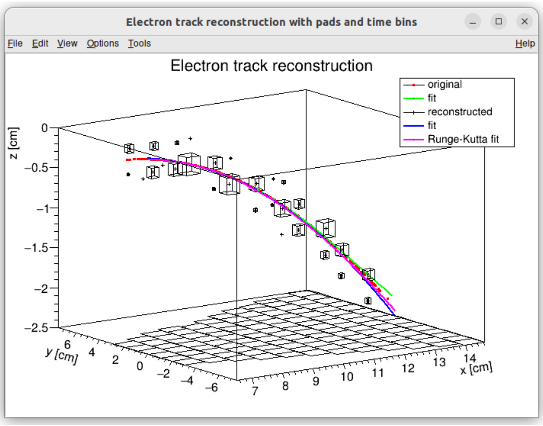


Figure 4.1: Example of a fitted reconstructed track. Swap for better image.

our coordinate system with an initial direction in the positive x axis. The track is simulated microscopically (see Section 2.1) with a kinetic energy of 8 MeV in a gas mixture 90% Ar + 10% CO_2 (the same track was used in Section 3.1).

In order to calculate the spline, we use the class *TSpline3* from ROOT. This allows us to evaluate the spline using the coordinates (x_n, z_n) of each node and the derivatives d_1, d_2 in the first and the last node. We can fit these parameters of a fixed amount of nodes to the simulated trajectory. We use the IMPROVE algorithm provided by the *TMinuit* class in ROOT. This algorithm attempts to find a better local minimum after converging.

After the fit, we want to get an energy estimate. We can calculate it at every point using the radius of curvature of the fitted spline. In ROOT, the part of the spline corresponding to a given node is defined as

$$z(x) = z_n + b\Delta x + c(\Delta x)^2 + d(\Delta x)^3, \tag{4.1}$$

where $\Delta x = x - x_n$ and b, c, d are coefficients. Using this equation, we can derive the radius of curvature:

$$r(x) = \frac{(1 + z'^2(x))^{\frac{3}{2}}}{z''(x)} = \frac{\left(1 + (b + 2c\Delta x + 3d(\Delta x)^2)^2\right)^{\frac{3}{2}}}{2c + 6d\Delta x}. \tag{4.2}$$

Based on the geometry of the detector, we can assume the magnetic field $\boldsymbol{B}(x, 0, z) = (0, B(x, z), 0)$ for a track in the XZ plane. Since the electron is relativistic, the effect of the electric field on its trajectory is negligible. The Lorentz force F_L is then always perpendicular to the momentum of the electron and is therefore equal to the centripetal force F_c:

$$\begin{aligned}
F_L &= F_c, && (4.3)\\
e\boldsymbol{v} \times \boldsymbol{B} &= \frac{\gamma m_e v^2}{r}, && (4.4)\\
ec\beta B &= \frac{E_{0e}\beta^2}{r\sqrt{1-\beta^2}}, && (4.5)\\
\sqrt{1-\beta^2} &= \frac{E_{0e}\beta}{ecBr}, && (4.6)\\
\beta^2(x) &= \frac{1}{1 + \left(\frac{E_{0e}}{ecB(x,z(x))r(x)}\right)^2} && (4.7)
\end{aligned}$$

where e is the elementary charge, c is the speed of light in vacuum, m_e is the rest mass of electron, $E_{0e} = m_e c^2$ is the corresponding energy, γ is the Lorentz factor, $\boldsymbol{v}$ is the velocity of the electron, and $\beta = \frac{v}{c}$. We can then finally get our estimate of the kinetic energy for a given point on the trajectory as follows:

$$E_{\text{kin}}(x) = \left(\frac{1}{\sqrt{1-\beta^2(x)}} - 1\right) E_{0e}. \tag{4.8}$$

We can then average these estimates at multiple points to get one final estimate. Add some figures.

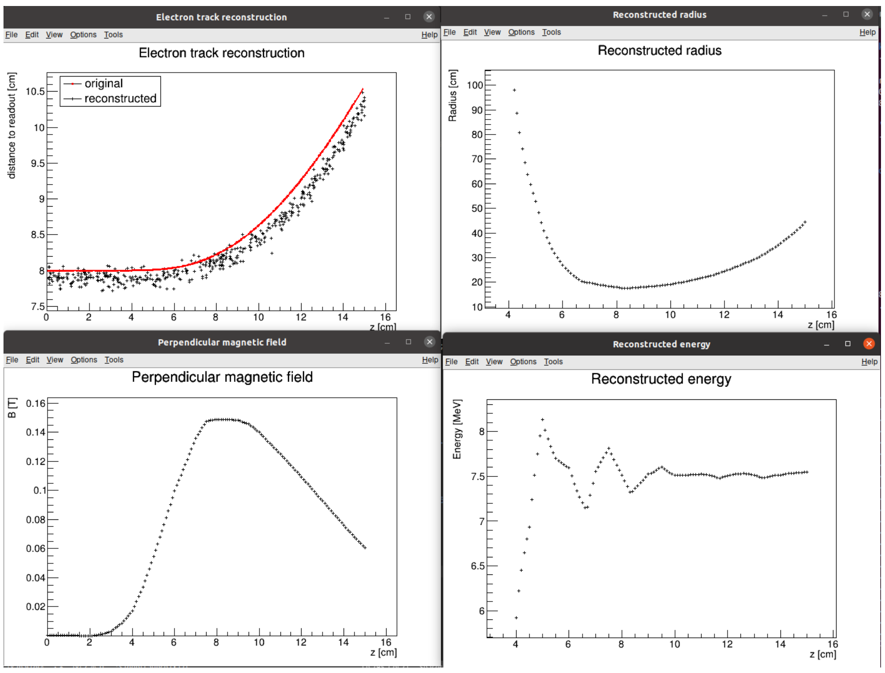


Figure 4.2: First attempt at a track reconstruction using only the drift velocity. Spline energy reconstruction attempt. Swap for better image(s) – subfigure environment., correct coordinates.

4.2 Circle and Lines Fit

A simpler alternative for the first estimation of the particle's kinetic energy is a fit of the trajectory with a circular arc with lines attached smoothly. This shape of trajectory corresponds to a movement of a charged particle through a homogeneous magnetic field perpendicular to the particle's momentum and limited to a certain volume. In general, the shape of such a trajectory in a non-perpendicularly oriented field is a spiral. In our case, this component should be negligible since the field is approximately toroidal and the particle motion is nearly perpendicular to it. At first, we tested a 2D version of this fit, then we adapted it to 3D.

Since our field is not homogeneous, it is not entirely clear what value of magnetic field should be used along with the fitted radius (using equations 4.7 and 4.8) to get the best estimate for the kinetic energy. Since we only use this method to get a first rough estimate that we later refine, an optimal solution of this problem is not required. Instead, we tested two options: taking the value of the field in the middle of the fitted circular arc and taking the average field along it. We haven't really tried to plot this for multiple tracks, but these estimates are saved somewhere and could be plotted.

4.2.1 Two-dimensional fit

In the 2D case, the fitted function used for the electron track (which bends down, so we need to use the upper part of the circle) described in Section 4.1 looks like this: Maybe describe this track that we used at the beginning somewhere earlier (section microscopic simulations → Testing track?) so that it is easier to refer to it in multiple sections. It is not part of the early GitHub commits, so maybe it won't be possible to create exact replicas of the images, but they should be at least very similar.

$$z(x) = \begin{cases} a_1x + b_1 & x < x_1 \\ z_0 + \sqrt{r^2 - (x - x_0)^2} & x_1 \leq x \leq x_2 \\ a_2x + b_2 & x > x_2 \end{cases}, \tag{4.9}$$

where $a_{1,2}$ and $b_{1,2}$ are the parameters of the lines, (x_0, z_0) is the center of the circle, r is its radius, and $(x_{1,2}, z_{1,2})$ are the coordinates of the function's nodes. That means we have 9 parameters ($z_{1,2}$ is not used in the function) along with 2 continuity conditions and 2 smoothness conditions. For the fit, we use the coordinates of the nodes and the radius of the circle, which gives us 5 independent parameters (only the radius has to be larger than half of the distance between nodes). The continuity conditions (combined with the relations for $z_{1,2}$) are as follows:

$$z_{1,2} = a_{1,2}x_{1,2} + b_{1,2} = z_0 - \sqrt{r^2 - (x_{1,2} - x_0)^2}. \tag{4.10}$$

The smoothness conditions are as follows:

$$a_{1,2} = \frac{x_0 - x_{1,2}}{\sqrt{r^2 - (x_{1,2} - x_0)^2}}. \tag{4.11}$$

Equation 4.10 gives us the values of $b_{1,2}$

$$b_{1,2} = z_{1,2} - a_{1,2}x_{1,2}. \tag{4.12}$$

For the coordinates of the center of the circle, we can use the fact that the center has to lie on the axis of its chord. In other words, there is a value of a parameter t such that, using the parametric equation of the axis

$$\begin{pmatrix} x_0 \\ z_0 \end{pmatrix} = \begin{pmatrix} \frac{x_1+x_2}{2} \\ \frac{z_1+z_2}{2} \end{pmatrix} + t \begin{pmatrix} \frac{z_2-z_1}{2} \\ \frac{x_1-x_2}{2} \end{pmatrix}. \tag{4.13}$$

At the same time, the center has to be in a distance of r from the nodes:

$$(x_1 - x_0)^2 + (z_1 - z_0)^2 = r^2, \tag{4.14}$$

$$\left(\frac{x_1 - x_2}{2} + \frac{z_1 - z_2}{2}t\right)^2 + \left(\frac{z_1 - z_2}{2} + \frac{x_2 - x_1}{2}t\right)^2 = r^2, \tag{4.15}$$

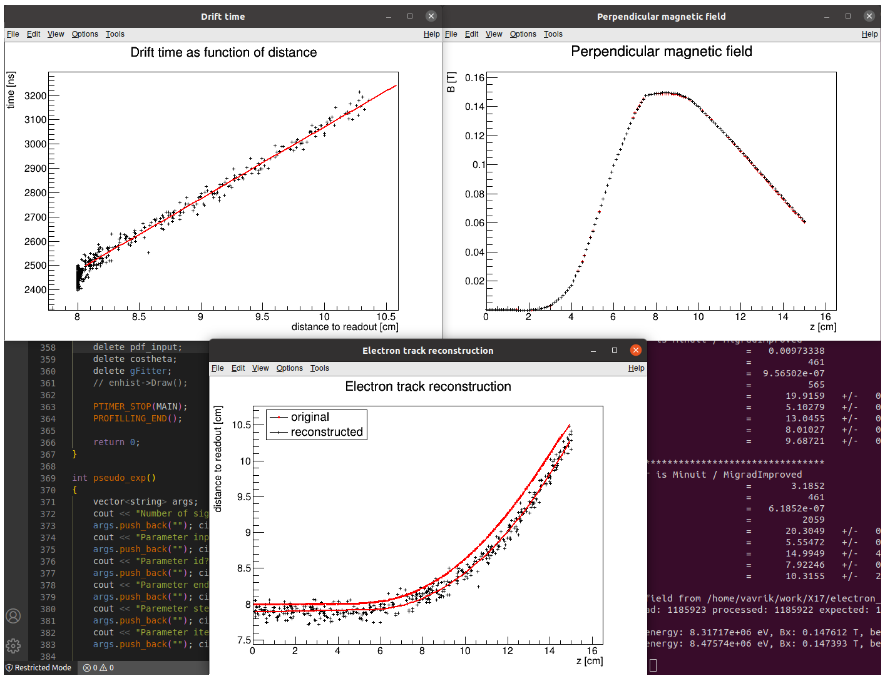
$$\left(\left(\frac{x_2 - x_1}{2}\right)^2 + \left(\frac{z_2 - z_1}{2}\right)^2\right)t^2 + \left(\frac{x_2 - x_1}{2}\right)^2 + \left(\frac{z_2 - z_1}{2}\right)^2 - r^2 = 0. \tag{4.16}$$

Since our electron track bends towards negative z and $x_2 > x_1$, we only care about the solution with $t > 0$

$$t = \sqrt{\frac{r^2}{\left(\frac{x_2-x_1}{2}\right)^2 + \left(\frac{z_2-z_1}{2}\right)^2} - 1}, \tag{4.17}$$

$$x_0 = \frac{x_1 + x_2}{2} + \frac{z_2 - z_1}{2}\sqrt{\frac{r^2}{\left(\frac{x_2-x_1}{2}\right)^2 + \left(\frac{z_2-z_1}{2}\right)^2} - 1}, \tag{4.18}$$

$$z_0 = \frac{z_1 + z_2}{2} - \frac{x_2 - x_1}{2}\sqrt{\frac{r^2}{\left(\frac{x_2-x_1}{2}\right)^2 + \left(\frac{z_2-z_1}{2}\right)^2} - 1}. \tag{4.19}$$


Drift time
File Edit View Options Tools Help
Drift time as function of distance
time [ns]
distance to readout [cm]
Perpendicular magnetic field
File Edit View Options Tools Help
Perpendicular magnetic field
B [T]
z [cm]
Electron track reconstruction
File Edit View Options Tools Help
Electron track reconstruction
distance to readout [cm]
original
reconstructed
z [cm]
Restricted Mode

```
    delete pdf_input;
    delete costheta;
    delete gFitter;
    // enhist->Draw();

    PTIMER_STOP(MAIN);
    PROFILLING_END();

    return 0;
}

int pseudo_exp()
{
    vector<string> args;
    cout << "Number of sig
    args.push_back(""); ci
    cout << "Parameter inp
    args.push_back(""); ci
    cout << "Parameter id?
    args.push_back(""); ci
    cout << "Parameter end
    args.push_back(""); ci
    cout << "Paremeter ste
    args.push_back(""); ci
    cout << "Parameter ite
    args.push_back(""); ci
```

```
                   =   0.00973338
                   =          461
                   =  9.56502e-07
                   =          565
                   =      19.9159   +/-   0
                   =      5.10279   +/-   0
                   =      13.0455   +/-   0
                   =      8.01027   +/-   0
                   =      9.68721   +/-   0

******************************
 is Minuit / MigradImproved
                   =       3.1852
                   =          461
                   =   6.1852e-07
                   =         2059
                   =      20.3049   +/-   0
                   =      5.55472   +/-   0
                   =      14.9949   +/-   4
                   =      7.92246   +/-   0
                   =      10.3155   +/-   2

field from /home/vavrik/work/X17/electron_
ad: 1185923 processed: 1185922 expected: 1

energy: 8.31717e+06 eV, Bx: 0.147612 T, be
energy: 8.47574e+06 eV, Bx: 0.147393 T, be
```

Figure 4.3: First attempt at a track reconstruction using only the drift velocity. Circle and Lines Fit in 2D. Swap for better image, correct coordinates.

The function defined in the equation 4.9 along with equations 4.11, 4.12, 4.18 and 4.19 derived using the continuity and smoothness conditions (combined with the relations for $z_{1,2}$) fully define our fitted function with parameters $r, x_{1,2}, z_{1,2}$. Some pictures of the fit on the tested track. Results of the fit. Again, the actual fit uses 8-z. Use GeoGebra schematics to generate a picture of 2D geometry.

Energy reconstruction with circle and lines fit. Trilinear interpolation of the magnetic field. Tested on a Runge-Kutta sample; future testing with microscopic simulations and map simulation. Preliminary 2D version (done) and complete 3D version. Geometry of the fit with its derivation.

4.2.2 Three-dimensional fit

Explain the geometry and least square method used for the 3D fit.

4.3 Runge-Kutta Fit

Single parameter fit with 4th order Runge-Kutta simulated track. Future testing with microscopic simulations and map simulation. Derivation of the geometry (least squares).

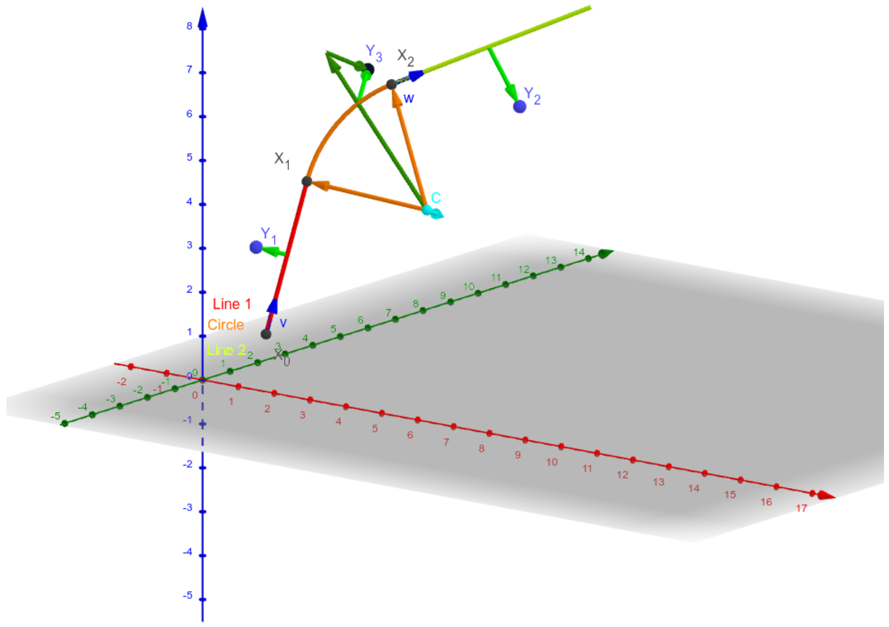


Figure 4.4: Circle and Lines Fit 3D geometry. Swap for better image.

Conclusion

Here or at the end of each section. Something about the future of this work?

Notes

General notes about the thesis:

- Check that all of the classes and other code are marked the same way in the text. I used italics somewhere, could use different font for this instead.
- Check unbreakable space in front of articles. Remove excessive article usage with proper nouns.
- Only electrons that start and end in the sector closer than 0.5 cm are used for reconstruction (newest version).

Bibliography

[1] Garfield++. `https://garfieldpp.web.cern.ch/garfieldpp/`. Accessed: 2023-05-18.

[2] Rene Brun and Fons Rademakers. Root — an object oriented data analysis framework. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 389(1–2):81–86, Apr 1997. Proceedings AIHENP'96 Workshop, Lausanne, Sep. 1996, See also https://root.cern/, Paper published in the Linux Journal, Issue 51, July 1998.

[3] I.B. Smirnov. Modeling of ionization produced by fast charged particles in gases. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 554(1):474–493, 2005.

List of Figures

List of Tables

List of Abbreviations

HEED High Energy Electro-Dynamics

IEAP CTU Institute of Experimental and Applied Physics, Czech Technical University in Prague

IPF Internal Pair Formation

MWPC Multi-Wire Proportional Chamber

OFTPC Orthogonal Fields TPC

TPC Time Projection Chamber

Tpx3 Timepix 3