pyFC: a TRIM-based fission chamber pulse shape simulator

CTH-NT-318 NOVEMBER 2015

pyFC: a TRIM-based fission chamber pulse shape simulator Documentation

Zsolt Elter

NUCLEAR ENGINEERING
DEPARTMENT OF APPLIED PHYSICS
CHALMERS UNIVERSITY OF TECHNOLOGY
S - 412 96 GÖTEBORG, SWEDEN

ISSN 1653-4662

pyFC: a TRIM-based fission chamber pulse shape simulator
Documentation

Zs. Elter

Abstract

This report presents the code system pyFC (*py*thon-based simulation of *F*ission *C*hambers) which simulates the pulse creation in fission chambers. The current pulses in a fission chamber are generated due to the ionization of the filling gas by the heavy ion emitted from the neutron induced fission in the fissile deposit. In pyFC the path of the heavy ion and the spatial distribution of the charges emerging from the ionization process are simulated with the TRIM code, and the parameters of the charge collection between the electrodes are computed with the BOLSIG software. The coupling of the codes is done in Python.

The report presents the physical and geometrical considerations implemented in pyFC and the verification of the code through comparison with the results of Chester, a CEA code for simulation of fission chambers.

Contents

1 Introduction, motivation

The simulation of the pulse shape generation is not a new topic. It was already achieved with the Chester code suite [1] developed at CEA, which is based on the Garfield software (the suite was named only after the publication, but in this document the code suite will be referred as Chester). Garfield is an open source suite [2], which has been developed at CERN to simulate drift chambers and has been extended to deal with other types of gas detectors. Those detectors are routinely used in particle physics and share many features in common with fission chambers: in both cases energetic particles ionize gas within which reigns an electric field, ensuring the separation and eventually the collection of the electron/ion pairs, yielding an electric current.

Some issues were identified with the clustering procedure of Garfield, which resulted artificial heavy ion trajectories as shown in Fig. 1 (the trajectories are rather jagged while in reality the heavy ion has a rather smooth path as described in Sec. 7). As a consequence of this behaviour, a large fraction of the paths starting from a given electrode returns almost immediately to the same electrode. It was verified that after eliminating these returning paths, the mean pulse shapes provided by Chester became realistic. Nevertheless, calculating these artificial paths and then eliminating them is not a practical solution; rather, their calculation should be avoided. For instance, in case of a detailed sensitivity and uncertainty analysis, a large amount of computations has to be performed. This means the waste of a large amount of computational time, which is not affordable.

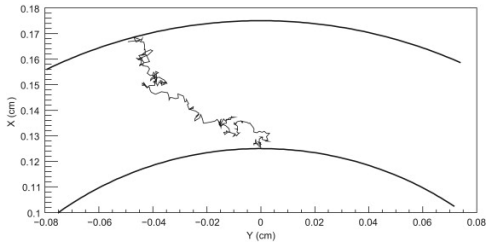

Figure 1: A sample simulated path in a CF4 chamber with Chester

To alleviate this problem, a new code system was developed, which does not use the Garfield software.

The code pyFC (*py*thon-based simulation of *F*ission *C*hambers) employs TRIM [3] as a heavy ion transport simulation engine. TRIM is the transport code of SRIM, which was already shown to be suitable for fission chamber simulations [4]. TRIM computes the heavy ion paths and the ionization in the filling gas. The parameters of the charge collection between the electrodes are determined with the BOLSIG software [5]. The electron transport is solved within pyFC based on the transport parameters.

The documentation first gives a short description of the processes leading to current creation in the fission chamber (i.e. the processes to be simulated).

After a general description of the code system and a summary of the underlying assumptions are given. Then the geometrical and physical considerations implemented in pyFC are discussed. The verification of the implementation is made by comparing the pyFC results to Chester outputs (as it was mentioned earlier, although certain issues related to the heavy ion paths were identified, the results of Chester were previously verified, therefore the comparison gives adequate insights about the implementations in pyFC). Finally the report presents a user guide to explain how to set up the code system, how to prepare input files and run calculations.

2 Physical processes in fission chambers

Fission chambers are nuclear detectors that are widely used to deliver online neutron flux measurements. This type of detector is an ionization chamber containing fissile material in order to detect neutrons. The most common design consists of one or more electrode pairs, at least one electrode is coated with a fissile layer from a few micrograms to a few grams. The spacing between each anode and cathode goes from tens of microns to few millimeters. The chamber itself is filled with an argon-based gas pressurized at a few bar. The processes leading to a current pulse after a neutron entering the chamber are the following:

(a) When a neutron reaches the fissile coating, it is likely to induce a fission event which generates (usually) two heavily charged ions, the fission products emitted in two nearly opposite directions.

(b) The heavy ion which is emitted out of the fissile layer ionizes the filling gas along its trajectory (therefore creates electron/ion pairs).

(c) A DC-voltage of a few hundred volts is applied between the electrodes, therefore the electrons and positive ions drift across the filling gas in opposite direction towards the anode and cathode respectively

(d) During the drift both the electrons and the gas ions induce a current pulse (named in this document as electronic and ionic pulse) in the electrodes.

3 Code suite

pyFC is a code suite implemented in Python language. Its may goal is to simulate the processes described in the previous section. It is built around five modules:

- pyFC.py: samples the fission products, their energies and their emission angles; performs the necessary rotations of the heavy ion track to simulate the correct fission chamber geometry (see in Sec. 4); calculates the charge generation and the pulse generation in the chamber (see in Sec. 5 and Sec. 6)

- trim.py: creates and runs the TRIM inputs; extracts the outputs
- bolsig.py: creates and runs the BOLSIG inputs; extracts the outputs
- input.py: contains the user defined description of the simulation job and the fission chamber (see in Sec. 8)
- main.py: couples all the above mentioned modules

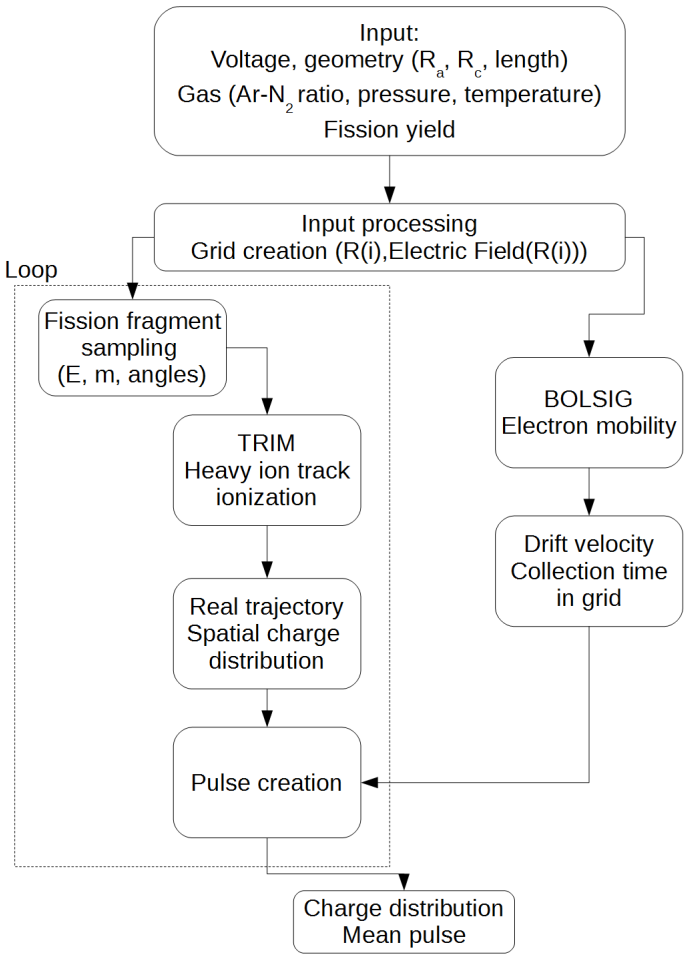

Figure 2: The structure of the code

The basic structure is illustrated in Fig. 2. The code first reads its input file (detailed in Sec. 8). Then it creates the electric field in the chamber, and

calls BOLSIG to compute the electron mobility between the electrodes (these are needed to simulate the processes (c) and (d) mentioned in Sec. 2; for more details see Sec. 6). Subsequently the fission fragments entering the filling gas are sampled for each fission event (process (a) in Sec. 2), and then the code system calls TRIM to determine the path of the fragments, the amount of electrons created and their locations in the chamber (process (b) in Sec. 2). Finally pyFC computes the induced current by the moving charges (process (d) in Sec. 2).

The underlying approximations are discussed in the related sections, but here a brief summary of them is given:

- No recombination events and no avalanches are considered (i.e. the chamber is in the saturation regime)
- Only one of the fission fragments emerging from the fission event ionizes the gas
- The space charge effects are neglected (each fission product entering the inter-electrode space ionizes the gas independently)
- The filling gas is homogeneous
- Only the current induced by the electrons is considered (since the mobility of the ions created by the fission fragment is much lower)
- The self-absorption of the fission fragments within the coating is neglected
- The mean energy to create an electron/ion pair W is assumed to be 26.4 eV for pure argon and 26.9 eV for argon-nitrogen mixtures (for details, see [6])

4 Geometrical considerations, TRIM results

The fission chambers considered in pyFC consist of two coaxial electrodes from which one electrode is fissile coated. More complex chambers (multi-electrode chambers, multi-coating chambers) can be investigated with superposition, i.e. combining the results of two or more runs, each corresponding to one fissile coating, by assuming that the inter-electrode spaces are mutually independent and by neglecting the space charge effects.

Hence the geometrical description is quite simple. One has to define the radius of the electrodes, the length of the chamber, and the sensitive lengths. In the following the inner and outer electrode radii are denoted as R_I and R_E, respectively. As a default, the anode is considered to be the inner and the cathode the outer electrode, the related radii being denoted as R_a and R_c, respectively. Therefore in this document $R_I = R_a$ and $R_E = R_c$ for simplicity. Nevertheless, pyFC can handle cases when the cathode is the inner and the anode is the outer electrode (the electric field may be negative). To avoid

confusion, the notation of this document applies the inner and outer radii while describing the geometrical considerations, whereas it uses the anode and cathode radii when describing the electric field.

The fission chamber length and the sensitive length are denoted as l_{fc} and l_{sens}, respectively.

4.1 The TRIM geometry and its limitations

TRIM defines the target material as a 3-dimensional bulk cuboid. The user defined target depth lies on the x axis, while the geometry is infinite in the y and z directions, as shown in Fig. 3. (Note that although the figure shows fairly straight trajectories, this is not necessarily always the case: in denser filling gas the heavy ions can experience serious lateral straggling, i.e. the spread of the trajectories.) The user has the option to set the incident angle of the heavy ion, and the location of the source. The results are given in tables as illustrated in Fig. 4. The energy and the electronic stopping can be printed periodically when the ion loses a certain amount of energy at locations (x_k^T, y_k^T, z_k^T), where $k = 1, ...S$ and the number of locations S varies since the heavy ion is tracked by TRIM until it either looses all its energy or it reaches a defined target depth. The superscript T indicates that the positions are TRIM coordinates. These locations build up the trajectory of a given heavy ion. The energy decrease of the heavy ion includes both the nuclear and the electronic stopping.

In fission chambers the target material (filling gas) has a cylindrical shape (see in Fig. 5), therefore some geometrical considerations are needed to transform the TRIM geometry. The issue comes from the fact that depending on the incident angle, the heavy ion sees a different target depth (e.g. the length of the $\mathrm{I_{hi}E_{hi}}$ vector in Fig. 5). To overcome the differences between the TRIM geometry and the fission chamber geometry, in the pyFC system TRIM is always called with the source at $(0, 0, 0)$, and the heavy ions are always entering the target perpendicularly to the border of the target (zero incident angle). Hence only the depth of the gas layer has to be changed in each TRIM run. And subsequently the trajectories defined by the locations (x_k^T, y_k^T, z_k^T) have to be transformed into the fission chamber geometry (x_k^p, y_k^p, z_k^p), where the superscript p indicates pyFC coordinates..

In pyFC the emission source is at the $(R, 0, z_{rand})$, where R is the radius of the inner or outer electrode (depending on the problem) and z_{rand} is the random position along the z-axis ($z_{rand} \in [\frac{-l_{sens}}{2}, \frac{l_{sens}}{2}]$, where l_{sens} is the sensitive length of the fission chamber). As this shows, azimuthal symmetry is assumed.

In TRIM the heavy ion travels along the x-axis until it reaches the depth seen by the heavy ion in the fission chamber. Hence, to reconcile the geometry of the fission chamber with the geometry of TRIM, a translation and a rotation of the TRIM results are performed. The translation consists of moving the TRIM source $(0, 0, 0)$ to the pyFC source location $(R, 0, z_{rand}))$, whereas the rotation is a "virtual" one with the incident angles, since the coordinate system is not rotated (more details on this later).

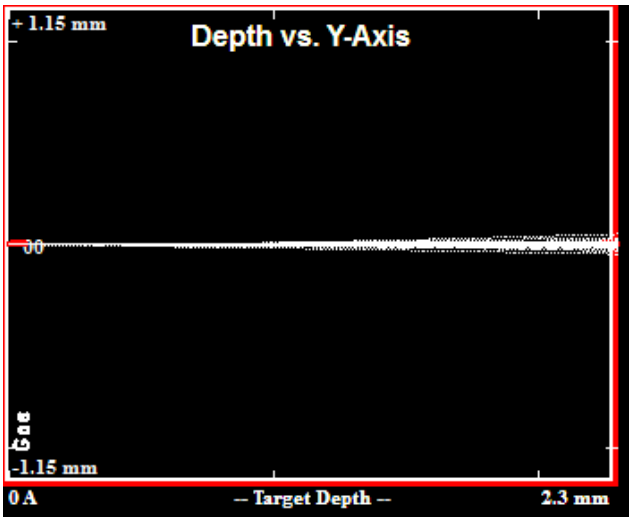

Figure 3: TRIM geometry example

Ion Number	Energy (keV)	Depth (X) (Angstrom)	Y (Angstrom)	Z (Angstrom)	Electronic Stop.(eV/A)	Energy Lost to Last Recoil(eV)
0000001	1.2000E+05	0.0000E+00	0.0000E+00	0.0000E+00	1.6707E+00	0.0000E+00
0000001	1.1992E+05	4.0585E+04	-3.3151E-03	-6.0443E-04	1.6704E+00	3.8940E+00
0000001	1.1989E+05	7.4998E+04	4.4304E-02	-3.4053E+00	1.6703E+00	1.5989E+01
0000001	1.1979E+05	1.3436E+05	7.7398E+00	-1.5163E-01	1.6701E+00	2.3724E+01
0000001	1.1966E+05	1.9370E+05	1.6287E+01	1.7553E+01	1.6697E+00	1.1284E+01

Figure 4: TRIM output example

The first task is therefore to determine the target depth seen by the heavy ion. This way the code can optimize the computational time of TRIM (since it guarantees that sufficient target depth is set in TRIM for each pulse event: long enough to let the ion either reach the opposite electrode or loose its energy, but not much longer than needed to save computational time). First, the emission angles in the coordinate system of the fission chamber (see Figs 6 - 9) are sampled, and with these, the maximum depth traveled by the ion is estimated (due to the lateral straggling of the ion this will be just an estimation). The problem can be separated into the depth estimation in the x-y plane and in the x-z plane.

4.2 The target depth in the x-y plane

In this case the target depth is different for the cases when the inner or the outer electrode is coated.

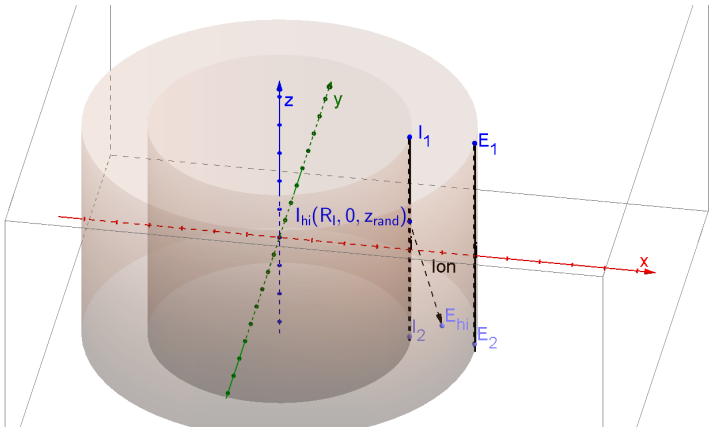

Figure 5: Fission chamber geometry

4.2.1 Inner electrode coated

The source of the heavy ions is the point $I(R_I, 0, z_{rand})$. The heavy ion is emitted along the vector $\mathbf{d_x}$ defined by an emission angle α. The task is to determine the length d_x from the known quantities R_I, R_E and α (see Fig. 6).

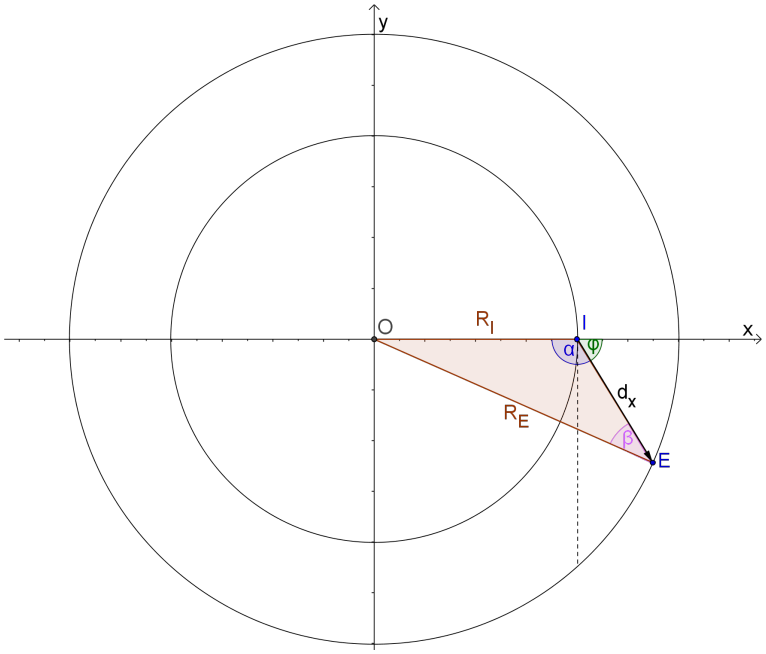

Figure 6: Target depth in the x-y plane: inner electrode

Applying the sine law twice consecutively (with the notation of Fig. 6):

$$\frac{R_E}{\sin\alpha} = \frac{R_I}{\sin\beta} \tag{1}$$

allows one to determine the angle β as

$$\beta = \arcsin\left(\frac{R_I}{R_E}\sin\alpha\right)$$

from which with yet another application of the sine rule relation one obtains

$$d_x = \frac{R_E}{\sin\alpha}\cdot\sin(\pi-\alpha-\beta) \tag{2}$$

4.2.2 Outer electrode coated

The case related to the coated outer electrodes is more complex. As one can see in Figs 7 and 8, the problem has to be separated into two different cases. One can define the angle φ_t belonging to the tangential drawn from the source point $E_1(R_E, 0, z_{rand})$ towards the inner electrode. The two cases when $\varphi > \varphi_t$ and $\varphi < \varphi_t$, where φ is the emission angle, need to be treated separately.

If the emission angle φ is greater than $\varphi_t (= \arcsin\frac{R_I}{R_E})$, then the target depth d_x is a chord of the outer electrode and can be determined by applying the cosine law (with the notations of Fig. 7):

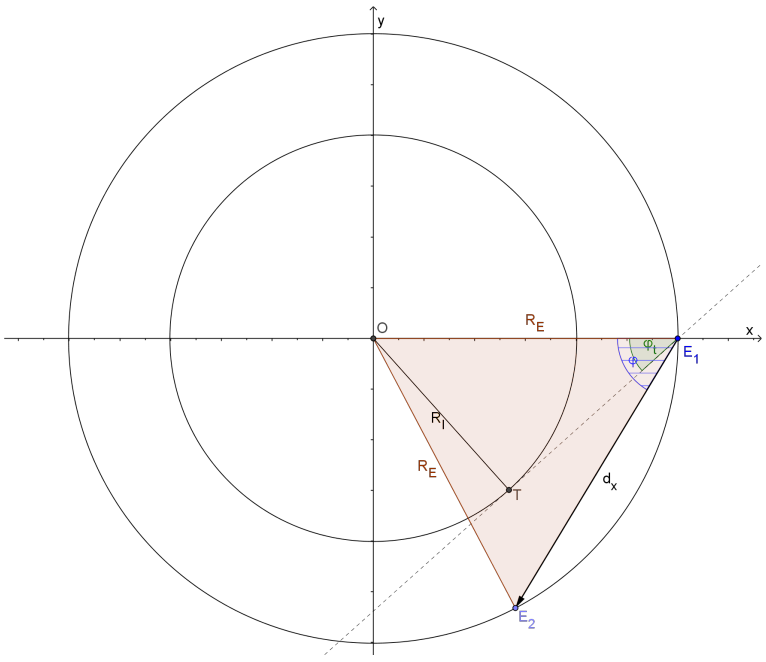

Figure 7: Target depth in the x-y plane: outer electrode 1.

$$d_x = \sqrt{2R_E^2 - 2R_E^2 \cos(\pi - 2\varphi)} \tag{3}$$

If the emission angle is smaller than φ_t then by applying the law of sines one gets (with the notation of Fig. 8):

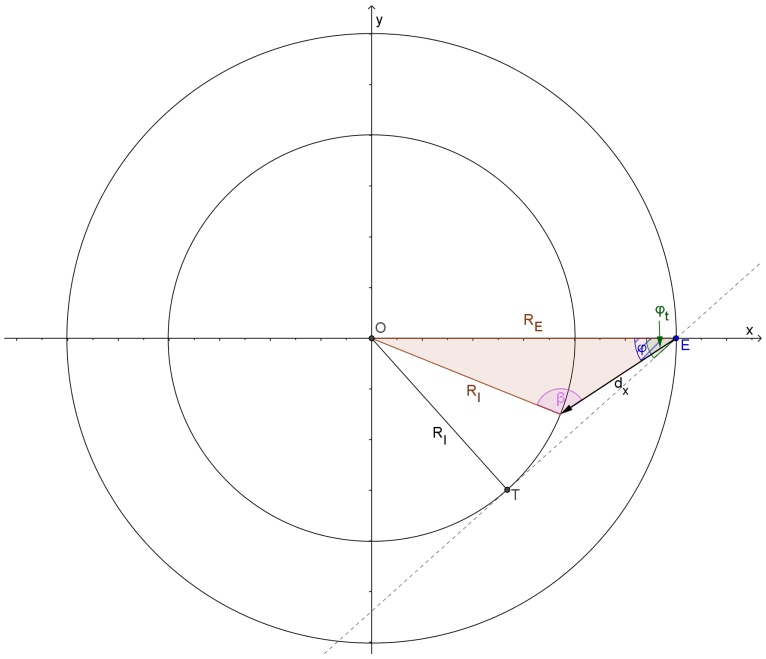

Figure 8: Target depth in the x-y plane: outer electrode 2.

$$\frac{R_I}{\sin\varphi} = \frac{R_E}{\sin\beta} \tag{4}$$

where

$$\beta = \arcsin\left(\frac{R_E}{R_i}\sin\varphi\right)$$

and therefore

$$d_x = \frac{R_I}{\sin\varphi} \cdot \sin(\pi - \varphi - \beta) \tag{5}$$

4.3 The target depth in the *x*-*z* plane

Axially there is no major difference between the coated inner or coated outer electrode cases. The target depth contribution of the axial direction is (with the notation of Fig. 9:

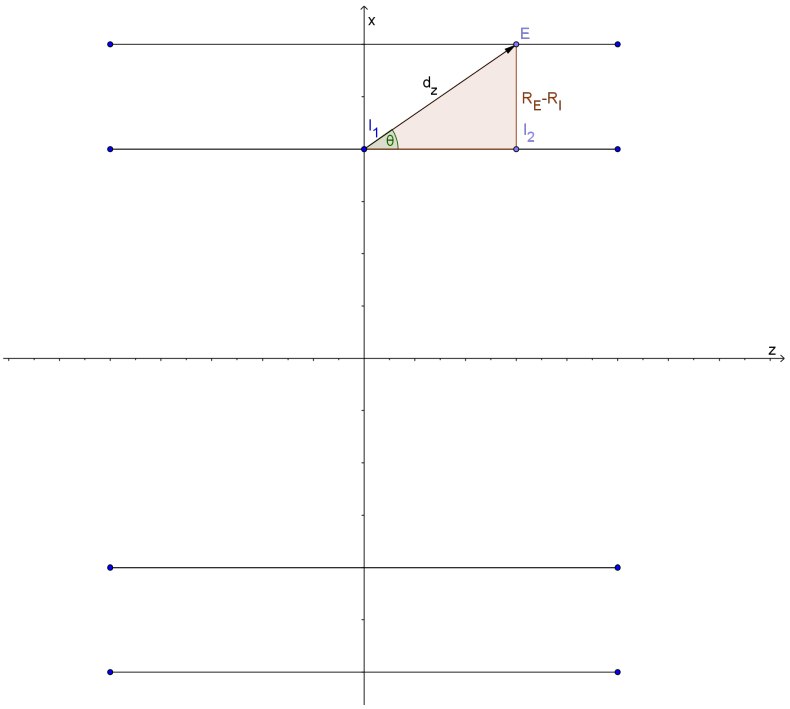

Figure 9: Target depth in the x-z plane

$$d_z = \frac{R_E - R_I}{\sin\theta} \tag{6}$$

4.4 The final target depth estimation

The final depth which is set up in TRIM becomes:

$$d = \sqrt{d_x^2 + d_z^2} \cdot 1.2 \tag{7}$$

where the 1.2 multiplier is introduced because the heavy ion trajectory is not a straight line due to the lateral straggling. In this way the code overestimates the depth. Note that in fission chambers with high pressure filling gas the heavy ions may not reach the opposite electrode, but this is not a problem since with Eq. (7) the target depth is still overestimated.

4.5 Rotation, angle sampling, translation

The sine of the emission angle in the x-y-plane (φ) and that of the angle in the x-z-plane (θ) was uniformly sampled between $[-1, 1]$.

To create the pyFC coordinates from the TRIM results, first one has to perform a rotation about the z-axis:

$$R_z(\varphi) = \begin{bmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

and afterwards about the y-axis:

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{9}$$

and the final transformation includes the rotations, the translation in x direction with the radius of the source electrode and the translation in z direction with the random axial coordinate:

$$\begin{bmatrix} x_k^p \\ y_k^p \\ z_k^p \end{bmatrix} = R_y R_z \begin{bmatrix} x_k^T \\ y_k^T \\ z_k^T \end{bmatrix} + \begin{bmatrix} R \\ 0 \\ z_{rand} \end{bmatrix} =$$

$$\begin{bmatrix} (z_k^T \sin\theta + x_k^T \cos\varphi\cos\theta - y_k^T \cos\theta\sin\varphi) + R \\ y_k^T \cos\varphi + x_k^T \sin\varphi \\ (z_k^T \cos\theta - x_k^T \cos\varphi\sin\theta + y_k^T \sin\varphi\sin\theta) + z_{rand} \end{bmatrix} \tag{10}$$

(Note: for the outer electrode case the x_k^T coordinate has to be multiplied with -1.)

4.6 The final trajectory

Since the target depth was overestimated, the rotated trajectories may end outside of the chamber. Thus after the rotation and the translations the locations are kept only if the following conditions are fulfilled:

$$(x_k^p)^2 + (y_k^p)^2 \geq R_I^2 \quad \cap \quad (x_k^p)^2 + (y_k^p)^2 \leq R_E^2 \quad \cap \quad z_k^p \leq \frac{l_{fc}}{2} \quad \cap \quad z_k^p \geq \frac{-l_{fc}}{2}$$

where $k = 1, ...S'$ and the updated number of locations S' is less than or equal to the number of locations in the TRIM output S. The equality holds for cases when the heavy ion is absorbed before reaching the opposite electrode (i.e. in cases when the filling gas pressure is high).

5 Spatial charge distribution

Next the charge creation along the trajectory is determined within the pyFC module. Each heavy ion ionizes the gas separately. Only the number of electrons is determined because due to the lower mobility of the positive ions, they can be considered as stationary during the time it takes to collect the electrons.

TRIM outputs the electronic stopping loss ES_k at each location (x_k, y_k, z_k) in $ev/\mathring{A}$ as it is shown in Fig. 4. After determining the distance s_k between

consecutive trajectory locations, the ionization loss IL_k and the number of generated electrons N_e, k can be computed at each location:

$$s_k = \sqrt{(x_k - x_{k-1})^2 + (y_k - y_{k-1})^2 + (z_k - z_{k-1})^2} \tag{11}$$

where $k = 1, ...S'$ and (x_0, y_0, z_0) is the source location.

$$IL_k = ES_k \cdot s_k \tag{12}$$

$$N_{e,k} = \frac{IL_k}{W} \tag{13}$$

where W is the mean energy to create an electron/ion pair. It is considered as 26.4 eV for pure argon and 26.9 eV for argon-nitrogen mixtures (details in [6]). Thus the charge at each location along the trajectory is given as

$$q_k = N_{e,k} \cdot q_E \tag{14}$$

where q_E is the electron charge.

TRIM also provides the kinetic energy of the heavy ion, hence the determination of the time of the charge generation (when the heavy ion reached the given location) is also possible. The user has an opportunity to choose whether the charge creation is instantaneous or not. The impact of this is detailed later.

6 Electron transport, induced current

After that pyFC determined the amount of electrons created in the TRIM computations, it calculates the electron transport between the electrodes. The BOLSIG software is employed to compute the electron mobility in the filling gas.

pyFC is capable to work with two approximations. The first assumes that the electron mobility is independent from the radial position of the charges. In this case an analytic solution of the induced current is applied. The other option assumes that the mobility depends on the radial position. Then the computation of the induced current is solved numerically.

In both cases the induced current is assumed to occur according to the Shockley-Ramo theorem [7]. The following subsections provide more details on both options.

6.1 BOLSIG version and electron cross sections

In pyFC the 07/2015 version of BOLSIG is used. This version provides a console application, therefore it is possible to run it via scripts. The cross sections are extracted from the MAGBOLTZ code, version 8.9 March 2010 (downloaded from the LXcat, an open-access website) [8].

The drift velocity computed by BOLSIG using the above mentioned cross sections was compared with the experimental results of [9]. Fig. 10 shows that the computed results show good agreement with the measured data. According

to Ref. [9], the error of the measurements is estimated to be less then 1%, hence the error bars shown in the figure correspond to this value.

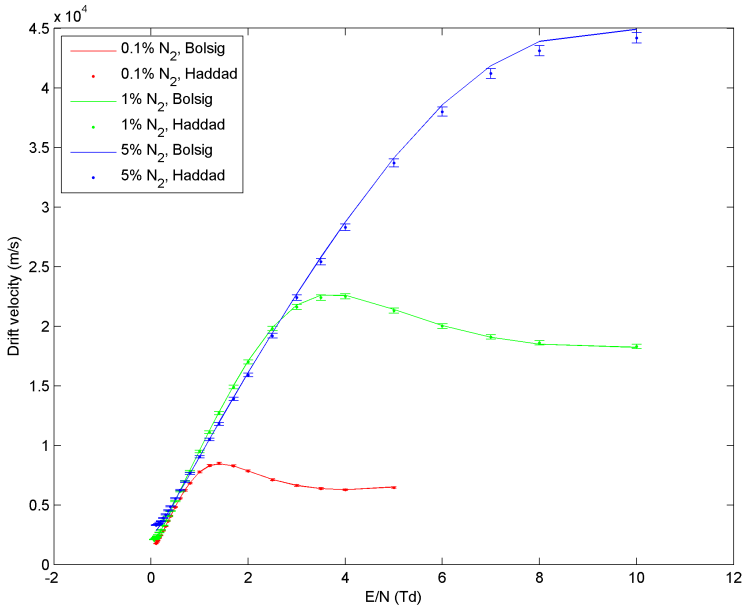

Figure 10: BOLSIG results with Biagi-v8.9 libraries vs Haddad experimental results

6.2 The induced current, time sampling

The induced current current by a charge q moving between electrodes will be

$$i = \frac{1}{U} q E v_d \tag{15}$$

according to the Shockley-Ramo theorem, where U, E and v_d are the voltage, the electric field, and the electron drift velocity between the electrodes.

The induced current will be a time dependent function. Therefore a time-resolved sampling of the induced current is necessary. The user can set the sampling time dt. Then the code first computes the maximum collection time $T_{coll,max}$ in the chamber (the time it takes for an electron to reach the anode from the cathode; see more details later) and then creates a time array with elements $T_j = j \cdot dt$ where $j = 0, 1, ...M$ (and $M = \lfloor T_{coll,max}/dt \rceil + 1$).

6.3 Constant electron mobility

In the analytic solution the electric field is constant and is calculated with a planar approximation:

$$E = \frac{U}{R_c - R_a} \tag{16}$$

The BOLSIG code is called only once during the simulation. The bolsig.py module creates the input, runs the code and reads the output. It receives the filling gas parameters and the constant reduced electric field, and it provides the electron mobility. Thereafter the induced current is calculated in a cylindrical geometry [10]. The collection time $T_-(r_k)$ of electrons from a location $r_k \in [R_a, R_c]$ (here the anode is considered as the inner electrode, and $r_k = \sqrt{x_k^2 + y_k^2}$) is given as

$$T_-(r_k) = \frac{\log(R_c/R_a)}{2\mu_- U}(r_k^2 - R_a^2) \tag{17}$$

which formula is used also to determine the maximum collection time ($T_{max,coll} = T_-(R_c)$). Then, by introducing a temporary variable

$$\tau_- = \frac{r_k^2 \log(R_c/R_a)}{2\mu_- U}, \tag{18}$$

the induced current by a charge q_k at location r_k is given as

$$i_-(t) = \frac{q_k}{2\log(R_c - R_a)\tau_-}\big(1 - \frac{t}{\tau_-}\big)^{-1} \tag{19}$$

hence with the time sampling

$$I_j = i_-(T_j) = \frac{q_k}{2\log(R_c - R_a)\tau_-}\big(1 - \frac{T_j}{\tau_-}\big)^{-1} \quad j = 0, 1, ..., M \tag{20}$$

The induced current by each charge along the heavy ion trajectory is computed and summed to determine the pulse created by a single heavy ion path.

It is shown in Sec. 7 that the analytic solution does not always give a correct solution, but it is sensitive to small electrode radius changes, therefore it is useful when performing uncertainty analysis. (Note: "correct solution" means here that, as it will be seen, the numerical solution of the maximum collection time is inherently closer to the real value since the mobility is not assumed to be constant).

6.4 Numerical solution

The homogeneous approximation of the electric field is not valid in every fission chamber (especially not in large chambers as it is shown in Sec. 7). Therefore a numerical solution was also implemented.

pyFC can generate a radial grid as illustrated in Fig. 11. The user defines the number N of grid parcels. The distance between the neighbouring parcel borders is

$$dr = \frac{R_E - R_I}{N} \tag{21}$$

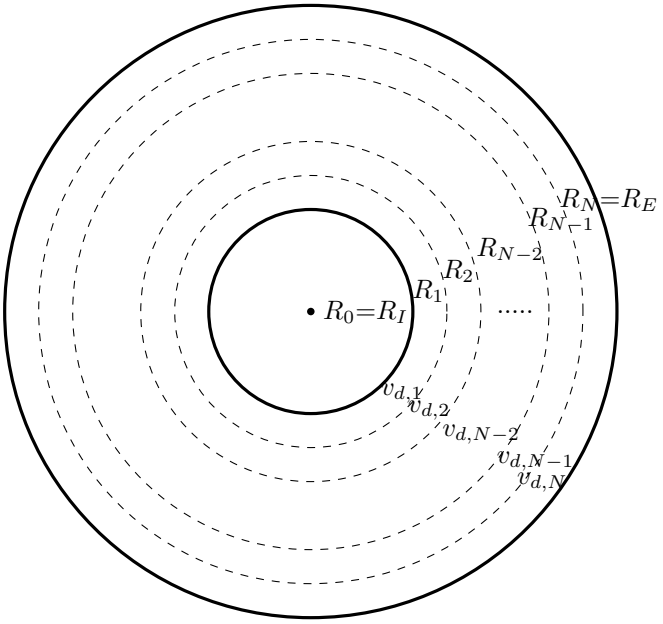

Figure 11: Grid scheme

with $R_0 = R_I$, $R_N = R_E$.

The electric field is calculated with cylindrical approximation in each parcel:

$$E_i = \frac{U}{R_i \log(R_c/R_a)} \quad i = 1, 2...N \tag{22}$$

BOLSIG computes the electron mobility $\mu_{e,i}$ in each parcel. The electron drift velocity in the parcel is

$$v_{d,i} = \mu_{e,i} E_i \quad i = 1, 2...N \tag{23}$$

The collection time between the boundaries of the given parcel is

$$T_{c,i} = \frac{dr}{v_{d,i}} \quad i = 1, 2...N \tag{24}$$

The maximum collection time in the chamber is approximated as the sum of the collection times in the parcels ($\sum_i T_{c,i}$).

According to the Shockley-Ramo theorem, the induced current by a charge q_k is then given as

$$I_j = i_-(T_j) = \frac{1}{U} q_k E(T_j) v_d(T_j) = \frac{1}{U} q_k E(R_i(T_j)) v_d(R_i(T_j)). \tag{25}$$

pyFC first identifies in which parcel the given charge q_k is located. Then it calculates the collection time between the initial location r_k and the closest

Table 1: Parameters of the simulated fission chambers

Name	CFUZ	CFUR	CF4	CF8
R_a (mm)	0.35	1	1.25	0.5
R_c (mm)	0.65	1.25	1.75	3.15
l_{sens} (mm)	12	14	8	33.5
l_{fc} (mm)	47	14	12	33.5
Bias voltage (V)	150	250	300	300
Gas	Argon+4% N2	Argon+4% N2	Argon	Argon
Pressure (bars)	5	5	12	9

parcel boundary towards the anode. Afterwards pyFC tracks the charge travelling towards the anode through the parcels and evaluates the induced current (hence it determines the position of the given charge $R_i(T_j)$ at a given moment T_j).

The induced current by each charge along the heavy ion trajectory is computed and summed to determine the pulse created by a single heavy ion path.

It has to be remarked that in case the spatial grid is very fine, while the time sampling is rather scarce, it may take shorter time to the electrons to cross a parcel than the related collection time in the sampling unit in that parcel. This would cause a numerical error. Therefore the user has to be prudent while setting the number of parcels and the sampling time.

(Note: it is seen that, as shown in Eq. (22), the electric field is logarithmic, therefore a logarithmic grid may be rather suitable. Nevertheless, the determination of the parcel which hosts the position r_k is faster for a linear grid.)

7 Verification, comparison with Chester

The comparison between pyFC and Chester results was performed with four fission chambers, published in [1]. The characteristics of the investigated fission chambers are summarized in Table 1. The fissile deposit is located on the anode for all the fission chambers except the CF8.

As it was mentioned earlier, although certain issues related to the heavy ion paths were identified, the results of Chester were verified previously, therefore the comparison gives adequate insights about the implementations in pyFC, in particular regarding the treatment of the electron transport.

In the Chester calculations, the trajectories with a cluster number higher than 20 were kept to discriminate the trajectories turning back towards the source [1]. pyFC calculations were performed with both the analytic and the numerical solution, and in both cases a finite heavy ion speed was considered (i.e. the charge creation was not instantaneous). The mean pulses are summarized in Fig. 12 and the charge distributions are shown in Fig. 13. The charge distribution in pyFC is independent from the electron transport solution, therefore the same distribution is valid for both the analytic and the numerical case.

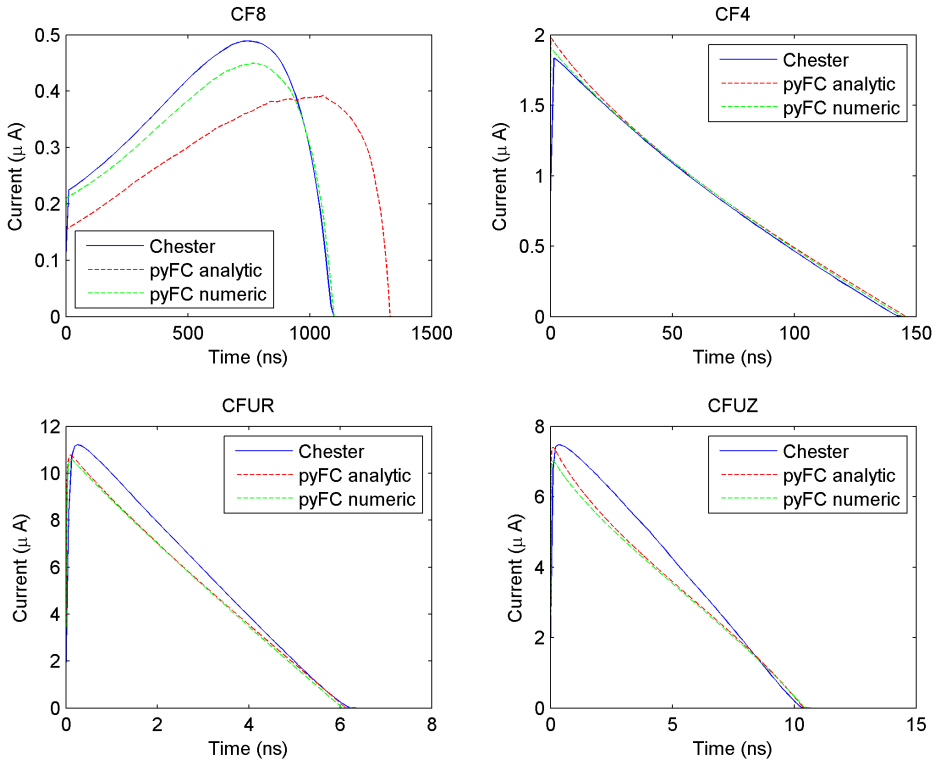

Figure 12: Mean pulses generated by Chester and pyFC

It can be concluded that the pyFC results are in good agreement with the Chester results. The analytic solution tends to fail in geometries where the electric field changes significantly inside the chamber (such as the CF8 in which the anode radius is relatively small compared to the cathode radius). The pyFC current amplitudes and the time integrals of the pulses are a bit lower than the results by Chester. The reason is that Chester overestimates the created charge due to the artificially jagged trajectories as illustrated in Fig. 1. This can be seen on the charge distribution Fig. 13 as well: the charge distribution computed by pyFC is narrower and its median is shifted towards lower charges.

One can notice that the peak of the pulses computed by pyFC arrives earlier than the peak of the Chester pulses. The reason is again the meandering heavy ion path: in Garfield it takes a longer time for the heavy ion to reach the opposite electrode than in reality, hence it overestimates the length of the rising part of the pulse. (Note: it was also shown that for chambers with coated anode the short rising part is a consequence of the heavy ion speed. If the charge creation is considered to be instantaneous, then the rising part disappears.)

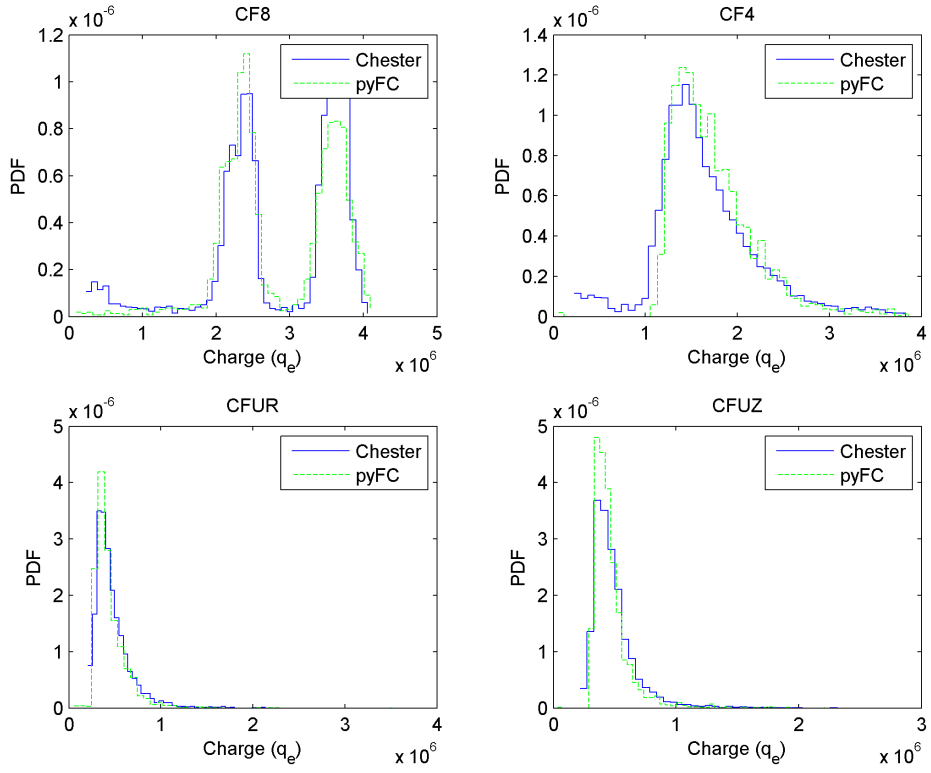

Figure 13: Charge distribution by Chester and pyFC

The heavy ion path length and the travelled distance was determined to verify that the Garfield artifact causes the above mentioned differences. The path length L is considered as the length of the heavy ion trajectory, while the travelled distance D is the distance between the beginning and the end of a trajectory. An illustration is given in Fig. 14 where the length of the red and blue lines L_1 and L_2 is the path length, while the length of the green line D is the travelled distance. The travelled distance is the same for both L_1 and L_2, but the jagged trajectory results in a much longer path length.

The results are summarized in Fig. 15. The pyFC calculated path length distribution is slightly wider than the travelled distance distribution due to the lateral straggling (illustrated in Fig. 14 by L_1 compared to D). One can also see that the travelled distance calculated by Chester and pyFC are in good agreement, but the path length is 2-3 times longer in Chester. As a consequence, it takes longer time to the heavy ion to travel through the chamber (therefore the peak of the pulses computed by Chester in Fig. 12 appears at a later time than that of the pulses determined by pyFC); and also, more charges are created

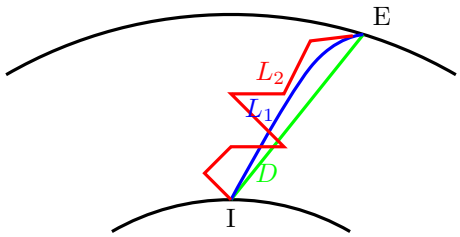

Figure 14: Path length and travelled distance

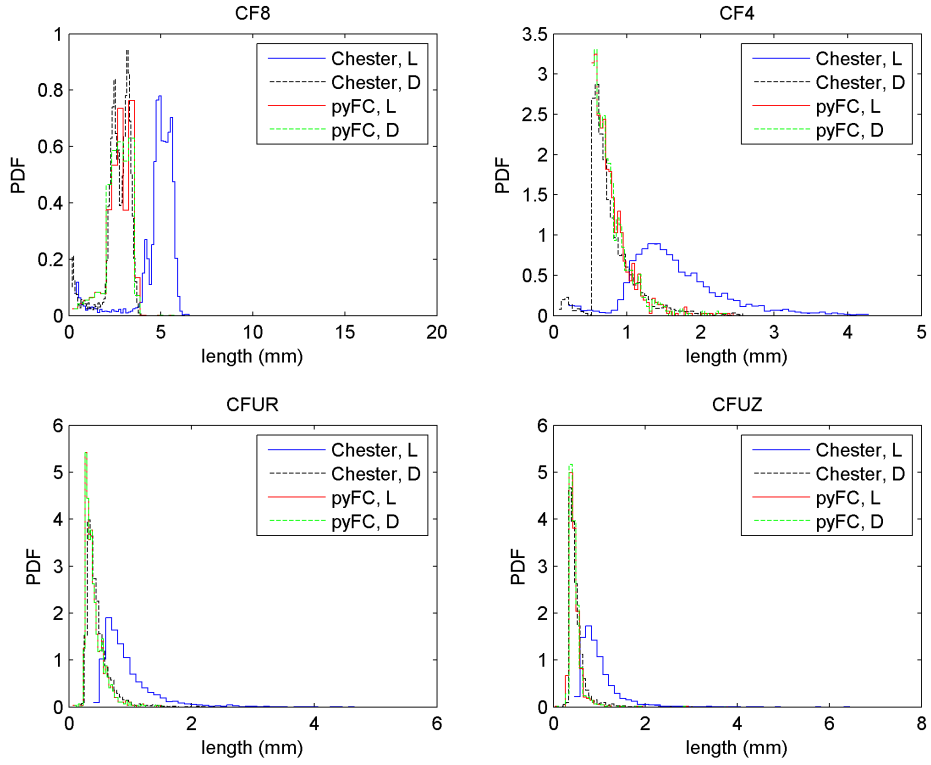

Figure 15: Path length and travelled distance distribution by Chester and pyFC

along the trajectory of the heavy ion in Chester.

8 Input form, tutorial

The current version 1.0 of the pyFC code can be run with *main.py*. The user has to define the fission chamber characteristics, and add some additional information about the calculation in the file *input.py*. An example is given below.

```
#general path information
srimPath="C:\\srim"
pyFCpath="C:\\chalmersphd\\pyFC\\pyFC_ver1"
fyFile='thermal_u235_yield.dat'    #here some conditions
    can be added to find out the file from U235

#fission chamber description
#cfht esp2
Ra=21.5                          #mm
Rc=23.                           #mm
source="cathodic"
lsens=231./2                     #mm
lfc=231./2                       #mm
U=2000                           #V
GasTemperature=300               #K
GasPressure=4.0                  #bar
ArRatW=1.000                     #w%
NRatW=0.000                      #w%
#calculation description (number of pulses, time
    resolution, name, path for the results, required
    results)
Nsim=1000
NGrid=15
TimeStep=0.1                      #ns
tagResu='CFHT_esp2_cat'
resuPath= 'CFHT_esp2_cat_test'
needCharge=False
needPosition=False
needPulse=True
needPulseHI=True
```

First, the user can set up the path for the SRIM code (*srimPath)*, for the pyFC (*pyFCpath)*, and the used fission yield library (*fyFile*).

Thereafter, the description of the fission chamber is given (the geometrical sizes in *mm*: anode radius *Ra*, cathode radius *Rc*, sensitive length *lsens*, fission chamber length *lfc*; polarization voltage in volts *U*, gas temperature *GasTemperature* in *K*, gas pressure *GasPressure* in bars, and finally the argon and nitrogen weight fractions *ArRatW,NRatW*.

Finally some additional parameters have to be defined: the number of pulses *Nsim* defines the length of the calculation, the number of parcels *NGrid* defines how fine the radial grid is (if this parameter is set to *0* the code uses the ana-

lytic approximation), the time resolution *TimeStep* gives the resolution of the created pulses, and the variables *tagResu* and *resuPath* describe the name of the output files and the folder where they will be placed. The variables *needCharge*, *needPosition*, *needPulse* and *needPulseHI* indicate to the program what information should be printed: the charges created along the heavy ion path, the position of the heavy ion reactions or the created pulses. The *needPulseHI* case takes into account the heavy ion speed, therefore the charge creation is not instantaneous.

To run the code the user has to install the SRIM-TRIM code according to the instructions on the SRIM hompage `srim.org`. After the installation is done, the TRIMAUTO file has to be updated: the first number in the file has to be changed to 1 (note that running TRIM as a normal program will change back this number to 0).

Thereafter the user has to download and install Python3 from `www.python.org/`. (The code pyFC is implemented in Python 3 and was tested with the 3.4 version. Since Python2 contains differences in the syntax it is not possible to run pyFC with that version.)

Finally, the content of the file pyFC.zip has to be extracted (the file can be obtained on request from the author of the present document). This file contains the BOLSIG software and all the necessary files of pyFC (the source files of the modules, the reference inputs for TRIM and BOLSIG, the fission yield for ^{235}U, and the electron cross section file. Different electron cross sections may be downloaded from `http://fr.lxcat.net/home/`; in this case the user has to edit the BOLSIG input, an example input and the main.py file). The user can edit the input.py file, and run the main.py source from the command prompt.

The outputs are created in the requested path in ASCII file format.

9 Conclusions

pyFC overcomes the issue related to the artificially jagged trajectories produced by Chester. As a consequence, it reduces significantly the computational time of the mean pulse shape, since no events have to be discriminated against.

The code shows good agreement with the Chester results. The discrepancies between the two codes are explained with the artificially jagged heavy ion trajectories generated by Garfield.

pyFC is very robust, stable and simple to handle, hence it is suitable to be used as a tool for an extensive uncertainty and sensitivity analysis in the future.

References

[1] P. Filliatre, C. Jammes, B. Geslot, R. Veenhof, A Monte Carlo simulation of the fission chambers neutron-induced pulse shape using the GARFIELD suite, Nuclear Instruments and Methods in Physics Research Section A:

Accelerators, Spectrometers, Detectors and Associated Equipment 678 (0) (2012) 139–147.

[2] R. Veenhof, Garfield, recent developments, Nuclear Instruments and Methods in Physics Research Section A, 419 (1998), p. 726-730 (GARFIELD, A Drift Chamber Simulation Program, CERN, 1994, CERN Program Library, Entry W5050 http://cern.ch/garfield)

[3] J. F. Ziegler, J. P. Biersack, M. D. Ziegler, The Stopping and Range of Ions in Matter, SRIM.org, (2008)

[4] P. Filliatre, C. Jammes, B. Geslot, Stopping power of fission fragments of ^{252}Cf in argon: A comparison between experiments and simulation with the SRIM code, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 618 (1) (2010) 294–297.

[5] G. J. M. Hagelaar, L. C. Pitchford, Solving the Boltzmann equation to obtain electron transport coefficients and rate coefficients for fluid models, Plasma Sources Sci. Technol. 14, 722-733 (2005)

[6] International Commission on Radiation Units and Measurements, Average Energy Required to Produce an Ion Pair, ICRU Report 31, (1979)

[7] W. Shockley, Currents to Conductors Induced by a Moving Point Charge, Journal of Applied Physics 9, 635–636 (1938)

[8] S.F. Biagi, Nuclear Instruments and Methods A 421(1999) 234

[9] G. N. Haddad, Drift Velocity of Electrons in Nitrogen-Argon Mixtures, Aust. J. Phys., 36, 297–303 (1983)

[10] C. Jammes, P. Filliatre, Technical Report published by CEA, SPEX/LD-CI/08/013