TOF spectrometer - Multi-electrode devices

Alain MERY
CIMAP, UMR 6252, 6 Boulevard Maréchal Juin,
F-14050 Caen cedex 04, France

mery@ganil.fr

Simion user meeting, GANIL, 11/06/2015

Introduction

We will describe the specific case of devices containing several similar electrodes. To illustrate, we choose the example of a time of flight (TOF) mass spectrometer which uses similar electrodes and resistor chains to create a uniform electric field. We will present different methods to define geometry using GEM files. We will also present the use of scalable electrodes (PA+ file) which may allow a significant reduction of RAM and disk space. We consider the particular case of a TOF spectrometer consisting of an "acceleration region" of length d$_s$ where the electric field is uniform followed by a "free flight region" of length L. We suppose ions created at rest but with different initial positions along the spectrometer axis and then accelerated in the TOF spectrometer. Solving the equations of motion leads to the conclusion that it exists a focal plan at L=2.d$_s$ where the TOF of the ions is independent of the initial position along the spectrometer axis. Note that this focusing effect is a first order approximation and is only valid for ions created "close" to the first acceleration plate. Fig.1 illustrates this focusing effect.

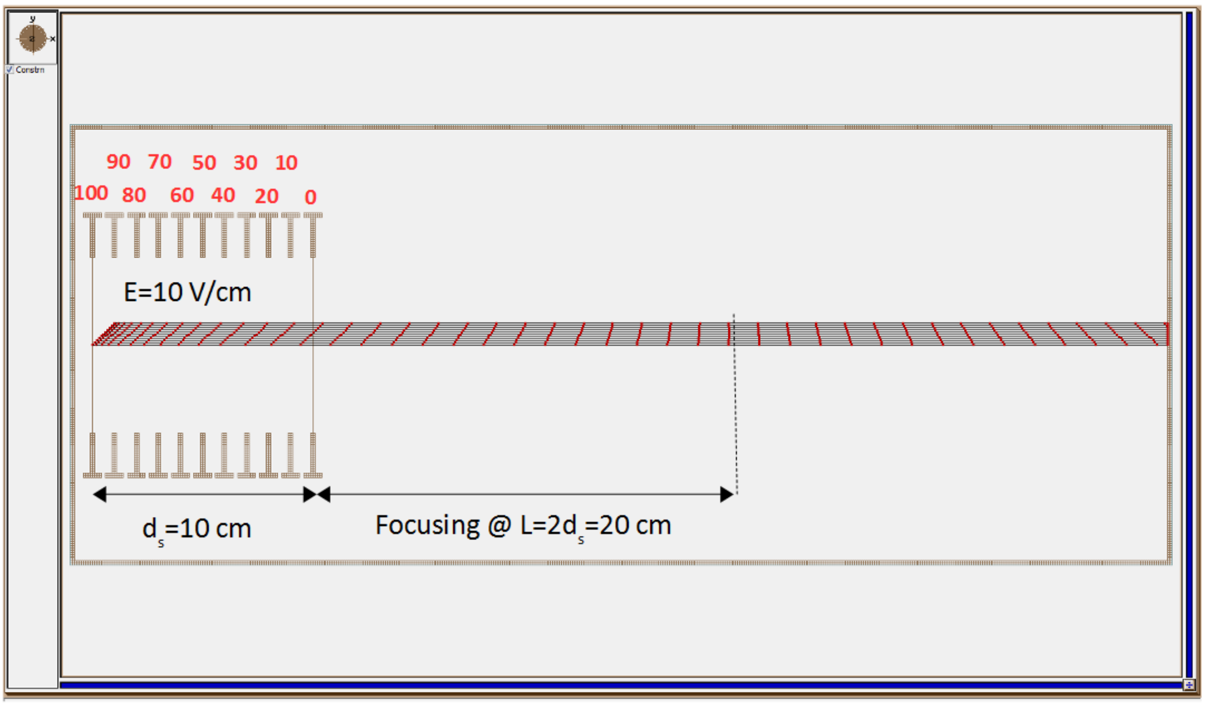

Figure 1: TOF spectrometer with uniform electric field. If a detector is placed at distance L=2.d$_s$, the TOF of ions is independent of their initial position along the spectrometer axis

Geometry file

In the case of this TOF spectrometer, one can write the GEM file in a very basic way :

```
pa_define(501,101,1,cylindrical,none,electrostatic)

; *************     grounded chamber    **********************
locate(0,0,0)   {
e(0)    {Fill{within{box(0,0,2,100)}}
        Fill{within{box(0,98,500,100)}}
        Fill{within{box(498,0,500,100)}} }      }

; ************  spectrometer electrodes  *********************
locate(10,0,0)  {
e(1)    {Fill{within{box(-1,40,1,60)}}
        Fill{within{box(0,0,0,60)}}
        Fill{within{box(-4,58,4,60)}}   }       }

locate(20,0,0)  {
e(2)    {Fill{within{box(-1,40,1,60)}}
        Fill{within{box(-4,58,4,60)}}   }       }

locate(30,0,0)  {
e(3)    {Fill{within{box(-1,40,1,60)}}
        Fill{within{box(-4,58,4,60)}}   }       }

locate(40,0,0)  {
e(4)    {Fill{within{box(-1,40,1,60)}}
        Fill{within{box(-4,58,4,60)}}   }       }

locate(50,0,0)  {
e(5)    {Fill{within{box(-1,40,1,60)}}
        Fill{within{box(-4,58,4,60)}}   }       }

locate(60,0,0)  {
e(6)    {Fill{within{box(-1,40,1,60)}}
        Fill{within{box(-4,58,4,60)}}   }       }

locate(70,0,0)  {
e(7)    {Fill{within{box(-1,40,1,60)}}
        Fill{within{box(-4,58,4,60)}}   }       }

locate(80,0,0)  {
e(8)    {Fill{within{box(-1,40,1,60)}}
        Fill{within{box(-4,58,4,60)}}   }       }

locate(90,0,0)  {
e(9)    {Fill{within{box(-1,40,1,60)}}
        Fill{within{box(-4,58,4,60)}}   }       }

locate(100,0,0) {
e(10)   {Fill{within{box(-1,40,1,60)}}
        Fill{within{box(-4,58,4,60)}}   }       }

locate(110,0,0) {
e(11)   {Fill{within{box(-1,40,1,60)}}
        Fill{within{box(0,0,0,60)}}
        Fill{within{box(-4,58,4,60)}}   }       }
```

However it can be convenient and more flexible to define parts of the device in separate GEM files and to include them from the main GEM file (i.e. the one loaded from the Simion GUI). Following is an example of a secondary GEM file (*TOF_electrode.gem*):

```
locate(0,0,0,1)
        {
        Fill{within{box(-1,40,1,60)}}
        Fill{within{box(-4,58,4,60)}}
        }
```

This GEM file is finally included inside the main GEM file (as well as *TOF_electrode_with_grid.gem*). The following code corresponds to the main GEM file (*TOF_v1.gem*):

```
pa_define(501,101,1,cylindrical,none,electrostatic)

; **********************     grounded chamber    ************************
locate(0,0,0)   {       e(0)    {Fill{within{box(0,0,2,100)}}
                                Fill{within{box(0,98,500,100)}}
                                Fill{within{box(498,0,500,100)}} }      }

; ********************  spectrometer electrodes  **********************
locate(10,0,0)  {       e(1)    {include(TOF_electrode_with_grid.gem)}  }

locate(20,0,0)  {       e(2)    {include(TOF_electrode.gem)}    }
locate(30,0,0)  {       e(3)    {include(TOF_electrode.gem)}    }
locate(40,0,0)  {       e(4)    {include(TOF_electrode.gem)}    }
locate(50,0,0)  {       e(5)    {include(TOF_electrode.gem)}    }
locate(60,0,0)  {       e(6)    {include(TOF_electrode.gem)}    }
locate(70,0,0)  {       e(7)    {include(TOF_electrode.gem)}    }
locate(80,0,0)  {       e(8)    {include(TOF_electrode.gem)}    }
locate(90,0,0)  {       e(9)    {include(TOF_electrode.gem)}    }
locate(100,0,0) {       e(10)   {include(TOF_electrode.gem)}    }

locate(110,0,0) {       e(11)   {include(TOF_electrode_with_grid.gem)}  }
```

This solution has the advantage that the size of all electrodes can be changed easily inside the corresponding *TOF_electrode.gem* file. However, if the user wants to modify the distance between electrodes then 11 values has to be modified in the main GEM file.

To give more flexibility, LUA code can be used inside GEM files. Thus, the user has the possibility to define local variables (size, position of electrodes...), to use loops for definition of similar electrodes... The following code (*TOF_v2.gem*) generates the geometry of the TOF spectrometer using LUA code.

```

# local R1 = 40                         -- definition of local variables
# local R2 = 58
# local R3 = 60
# local e1 = 2
# local e2 = 8

# local d = 10
# local N_elect = 11
# local x0 = 10

# local Lx =  x0 + d*(N_elect-1) + 401 -- array points in x
# local Ryz =  R3 + 41                  -- array points in y - z


pa_define($(Lx),$(Ryz),1, cylindrical, non-mirrored)

; **********************     grounded chamber    ***********************
# local e_gnd = 2
locate(0,0,0)
```

```
{        e(0){Fill{within{box(0, 0, $(e_gnd), $(Ryz-1))}}
          Fill{within{box(0, $(Ryz-1-e_gnd), $(Lx-1), $(Ryz-1))}}
          Fill{within{box($(Lx-1-e_gnd), 0, $(Lx-1), $(Ryz-1))}} } }


; ********************  spectrometer electrodes  **********************
#for n=1,N_elect do
locate( $(x0 + (n-1)*d) )
        {
        electrode($(n))
                {
                Fill{within{box($(-e1/2), $(R1), $(e1/2), $(R3))}}
                Fill{within{box($(-e2/2), $(R2), $(e2/2), $(R3))}}

-- Warning : "#" should be placed at the beginning of the line
#if n==1 or n==N_elect then
                Fill{within{box(0, 0, 0, $(R3))}}
#end
                }
        }
# end
```

When LUA macros are included in a GEM file, lines starting with "#" or text inside "$()" will be interpreted as LUA code. When a GEM file containing LUA macros is used, SIMION interprets the LUA code and generates a pre-processed GEM file (TOF_v2.processed.gem). Finally, SIMION automatically loads this resulting pre-processed file.

LUA workbench program

After loading the geometry in SIMION, one can apply the desired voltages on electrodes by using a LUA user program. The following example is used to apply 100 V to electrode #1, 90 V to electrode #2, 80 V to electrode #3..., 10 V to electrode #10 and 0 to electrode #11. This creates the desired uniform electric field.

```
simion.workbench_program()

local N_elect = 11
adjustable Vs=100

function segment.init_p_values()
        for n=1, N_electrodes
        do
                adj_elect[n]= Vs - Vs/(N_elect-1)*(n-1)
        end
end

function segment.fast_adjust()
        for n=1, N_electrodes
        do
                adj_elect[n]= Vs - Vs/(N_elect-1)*(n-1)
        end
end


function segment.terminate()
   sim_retain_changed_potentials = 1
end
```

Note that one can use `init_p_values` segment and the `terminate` segment (specific instruction: `sim_retain_changed_potentials = 1`)) to retain and restore potentials values at the end of Fly'm and control them in the SIMION GUI.

Scalable electrodes - PA+ file

In the case where several electrodes have distinct potentials but not completely independent (resistor chains,...), SIMION offers the possibility to group them as *scalable electrodes*. This presents the big advantage that only one solution array (.PA file) is generated for each group of electrodes. It allows to reduce significantly space on disk and RAM. In our example, only one group containing all 11 adjustable electrodes is needed to run the TOF spectrometer with uniform electric field. Since Simion 8.0.3, scalable electrodes are defined in an additional .PA+ file where each scalable electrode is defined as a matrix with coefficients corresponding to the relative scaling factors applied to each electrode of one group. For the TOF spectrometer, the following .PA+ file defines one group of electrodes with suitable coefficients to produce the uniform electric field.

```
potential_array
{
  scalable_electrodes =
        {
        [1] = {10,9,8,7,6,5,4,3,2,1,0}
        }
}
```

Here only .PA0 and .PA1 files are created instead of .PA0, .PA1, .PA2, .PA3,..., .PA11 which represents a gain of a factor of 6 compared to the classical definition of adjustable electrodes. Potentials can then be adjusted in the SIMION GUI or in the LUA workbench user program with the same syntax but then `adj_elect[1]` refers to scalable electrode as defined in the .PA+ file and not to the adjustable electrode defined in the .GEM file.

```
simion.workbench_program()

adjustable Vs=100

function segment.fast_adjust()
        adj_elect[1]= Vs
end
```

In comparison with previous scalable electrodes available in simion 7.0, the .PA+ file allow the definition of several groups which can be scaled independently. As a last example, we add a lens after the TOF spectrometer (electrodes #12 and #13 in the GEM file). The associated .PA+ file is the following:

```
potential_array
{
  scalable_electrodes =
        {
        [1] = {10,9,8,7,6,5,4,3,2,1,0, 0,0},
        [2] = { 1,1,1,1,1,1,1,1,1,1,1, 0,0},
        [3] = { 0,0,0,0,0,0,0,0,0,0,0, -1,1}
        }
}
```

Suitable potentials are applied to each group of electrodes using the workbench user program :

```
simion.workbench_program()

adjustable Vs=100
adjustable Vdc=50
adjustable Vlens=20

function segment.fast_adjust()
        adj_elect[1]= Vs
        adj_elect[2]= Vdc
        adj_elect[3]= Vlens
end
```

As a result, we have :

```
spectrometer electrodes : group #1 and #2
e(1)=150 V  (=10*Vs/10 + 1*Vdc/1 + 0*Vlens/1)
e(2)=140 V  (= 9*Vs/10 + 1*Vdc/1 + 0*Vlens/1)
e(3)=130 V  (= 8*Vs/10 + 1*Vdc/1 + 0*Vlens/1)
...
e(10)=60 V  (= 1*Vs/10 + 1*Vdc/1 + 0*Vlens/1)
e(11)=50 V  (= 0*Vs/10 + 1*Vdc/1 + 0*Vlens/1)

lens electrodes : group #3
e(12)=-20 V  (=0*Vs/10 + 0*Vdc/1 + -1*Vlens/1)
e(13)=20 V   (=0*Vs/10 + 0*Vdc/1 + 1*Vlens/1)
```

Note that for each group, the potentials are scaled with respect to the highest absolute value in the corresponding array. One has to keep this in mind when defining the coefficients in the PA+ file.