Anexo

Estudios de sensibilidad a SN utilizando un detector TPC esférico

Índice

1. Cálculo del espesor y presión para un STPC de diferentes materiales

Para obtener el valor de grosor necesario capaz de soportar las presiones requeridas necesitamos conocer la relación entre el espesor y la presión máxima permitida, teniendo en cuenta que emplearemos el 66 % de este valor (P_{real}). La expresión que relaciona ambos parámetros es la siguiente

$$P_{\max} = \frac{4\, f\, z\, e_a}{D_m} \tag{1}$$

donde P_{max} es la presión máxima permitida, f es el coeficiente de estrés nominal, z es el coeficiente de uniones, e_a la mitad del espesor de nuestra esfera y D_m el diámetro máximo de la esfera. El valor de estrés nominal es una característica de cada material que se puede encontrar en las referencias [1], [2], [3] y se encuentran visibles en la tabla 1. En el caso de una geometría esférica debido a que su construcción no requiere uniones extra como en otras geometrías, el coeficiente z tiene un valor de 1, el máximo posible.

Material	Estrés nominal (MPa)	Espesor (cm)	P_{max} (bar)	P_{real} (bar)
Cobre	23	5	14.8	1
				10
	120	10	150	100
Acero	136	10	170	115
Titanio	113	10	141.3	90

Tabla 1: Espesores del detector escogidos según el material y presión empleada.

2. Parámetros de la simulación

```
<?xml version="1.0" encoding="utf-8" standalone="no" ?>

<!DOCTYPE gdml [
<!ENTITY geometry SYSTEM "geometry.gdml">
<!ENTITY materials SYSTEM "materials.xml">
]>

<gdml xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:
    noNamespaceSchemaLocation="http://service-spi.web.cern.ch/service-spi/app/
    releases/GDML/schema/gdml.xsd">


<!--  Todos los valores estan en mm, bar, K, mg/cm3 -->

<define>
<constant name="world_size" value="20000" />

<constant name="radioIn" value="500" />
<constant name="radioOut" value="1500" />
<constant name="vesselThick" value="50" />
<constant name="DistVesselToWater" value="5000" />

```

```
<!-- Las densidades deberan ser adapatads a la presion escogida -->
<variable name="gasPressure" value="1.0" />
<variable name="gasTemperature" value="293.15" />
<variable name="targetGasDensity" value="5.492" />
<variable name="quencherDensity" value="0.659" />
<variable name="quencherFraction" value="0.05" />

</define>

&materials;

&geometry;

<setup name="Default" version="1.0">
<world ref="World"/>
</setup>

</gdml>
```

3. Geometría del detector

```
<box name="worldSolid" x="world_size" y="world_size" z="world_size" lunit="mm" />
<sphere name="readout" rmin="0" rmax="radioIn" deltaphi="360" deltatheta="180" aunit="deg" lunit="mm" />
<sphere name="gasSolid" rmin="radioIn" rmax="radioOut" deltaphi="360" deltatheta="180" aunit="deg" lunit="mm" />
<sphere name="vesselSolid" rmin="radioOut" rmax="radioOut+vesselThick" deltaphi="360" deltatheta="180" aunit="deg" lunit="mm" />
<sphere name="detectorSolid" rmin="0" rmax="radioOut+vesselThick" deltaphi="360" deltatheta="180" aunit="deg" lunit="mm" />
<box name = "waterSolidtotal" x="2.0*radioOut+2.0*vesselThick+2.0*DistVesselToWater" y="2.0*radioOut+2.0*vesselThick+2.0*DistVesselToWater" z="2.0*radioOut+2.0*vesselThick+2.0*DistVesselToWater" lunit="mm" />

<subtraction name="waterSolid">
<first ref="waterSolidtotal"/>
<second ref="detectorSolid"/>
</subtraction>

</solids>

<structure>

<!--{{{ Definimos los volumenes(material and solid) -->
      <volume name="gasVolume">
      <materialref ref="Xenon_MTH"/>  <!--Tambien existen Neon_ISO y Helium_ISO -->
      <solidref ref="gasSolid"/>
      </volume>

      <volume name="vesselVolume">
```

```
24          <materialref ref="Copper"/>  <!-- Tambien existen StainlessSteel y
                TitanPure -->
25          <solidref ref="vesselSolid"/>
26          </volume>
27
28          <volume name="waterVolume">
29          <materialref ref="Water"/>
30          <solidref ref="waterSolid"/>
31          </volume>
32
33          <!-- }}}  -->
34
35    <!-- Volumenes fisicos -->
36    <volume name="World">
37    <materialref ref="Vacuum" />
38    <solidref ref="worldSolid" />
39
40    <physvol name="gas">
41    <volumeref ref="gasVolume" />
42    <position name="gasPosition" unit="mm" x="0" y="0" z="0" />
43    </physvol>
44
45    <physvol name="vessel">
46    <volumeref ref="vesselVolume" />
47    <position name="vesselPosition" unit="mm" x="0" y="0" z="0" />
48    </physvol>
49
50    <physvol name="water">
51    <volumeref ref="waterVolume" />
52    <position name="waterPosition" unit="mm" x="0" y="0" z="0" />
53    </physvol>
54    </volume>
55
56    </structure>
```

4. Materiales y elementos

```
1     <!--  ##VERSION REST materials 1.4##  -->
2     <!--  densities in mg/cm3 @ 293.15K  -->
3     <materials>
4     <!--  {{{ Elements definition  -->
5           <element name="Hydrogen" formula="H" Z="1">
6           <atom value="1.00794"/>
7           </element>
8           <element name="Helium" formula="N" Z="2">
9           <atom value="4.0026"/>
10          </element>
11          <element name="Carbon" formula="C" Z="6">
12          <atom value="12.0107"/>
13          </element>
14          <element name="Nitrogen" formula="N" Z="7">
15          <atom value="14.0067"/>
16          </element>
```

```
17          <element name="Fluor" formula="F" Z="9">
18          <atom value="18.9984"/>
19          </element>
20          <element name="Silicon" formula="Si" Z="14" state="solid">
21          <atom value="28.0855"/>
22          </element>
23          <element name="Chromium" formula="Cr" Z="24" state="solid">
24          <atom value="51.996"/>
25          </element>
26          <element name="Manganese" formula="Mn" Z="25">
27          <atom value="54.938045"/>
28          </element>
29          <element name="Iron" formula="Fe" Z="26" state="solid">
30          <atom value="55.845"/>
31          </element>
32          <element name="Nickel" formula="Ni" Z="28" state="solid">
33          <atom value="58.6934"/>
34          </element>
35          <element name="Aluminium" formula="Al" Z="13" state="solid">
36          <atom value="26.982"/>
37          </element>
38          <element name="Xenon" formula="Xe" Z="54" state="gas">
39          <atom value="131.293"/>
40          </element>
41          <element name="Argon" formula="Ar" Z="18" state="gas">
42          <atom value="39.948"/>
43          </element>
44          <element name="Neon" formula="Ne" Z="10" state="gas">
45          <atom value="20.1797"/>
46          </element>
47          <element name="Zinc" formula="Zn" Z="30">
48          <atom value="65.4094"/>
49          </element>
50          <element name="Magnesium" formula="Mg" Z="12">
51          <atom value="24.3050"/>
52          </element>
53          <element name="Sulfur" formula="S" Z="16">
54          <atom value="32.0655"/>
55          </element>
56          <element name="Chlorine" formula="Cl" Z="17">
57          <atom value="35.4532"/>
58          </element>
59          <element name="Titanium" formula="Ti" Z="22">
60          <atom value="47.867"/>
61          </element>
62          <!--  }}}  -->
63      <!--  {{{ Quenchers  -->
64          <material name="TMA" state="gas">
65          <D unit="mg/cm3" value="quencherDensity"/>
66          <T unit="K" value="gasTemperature"/>
67          <P unit="bar" value="gasPressure"/>
68          <composite n="3" ref="Carbon"/>
69          <composite n="9" ref="Hydrogen"/>
70          <composite n="1" ref="Nitrogen"/>
71          </material>
```

```
72          <material name="isobutane" state="gas">
73          <D unit="mg/cm3" value="quencherDensity"/>
74          <T unit="K" value="gasTemperature"/>
75          <P unit="bar" value="gasPressure"/>
76          <composite n="4" ref="Carbon"/>
77          <composite n="10" ref="Hydrogen"/>
78          </material>
79          <material name="methane" state="gas">
80          <D unit="mg/cm3" value="quencherDensity"/>
81          <T unit="K" value="gasTemperature"/>
82          <P unit="bar" value="gasPressure"/>
83          <composite n="1" ref="Carbon"/>
84          <composite n="4" ref="Hydrogen"/>
85          </material>
86          <!--  }}}  -->
87    <!--  {{{ Gas mixtures  -->
88          <material name="PureHelium" state="gas">
89          <T unit="K" value="gasTemperature"/>
90          <P unit="bar" value="gasPressure"/>
91          <MEE unit="eV" value="188"/>
92          <D unit="mg/cm3" value="targetGasDensity"/>
93          <fraction n="1" ref="Helium"/>
94          </material>
95          <material name="PureXenon" state="gas">
96          <T unit="K" value="gasTemperature"/>
97          <P unit="bar" value="gasPressure"/>
98          <MEE unit="eV" value="482"/>
99          <D unit="mg/cm3" value="targetGasDensity"/>
100         <fraction n="1" ref="Xenon"/>
101         </material>
102         <material name="Xenon_ISO" state="gas">
103         <D unit="mg/cm3" value="targetGasDensity+quencherDensity"/>
104         <P unit="bar" value="gasPressure"/>
105         <T unit="K" value="gasTemperature"/>
106         <fraction n="quencherFraction" ref="isobutane"/>
107         <fraction n="1-quencherFraction" ref="Xenon"/>
108         </material>
109         <material name="Xenon_MTH" state="gas">
110         <D unit="mg/cm3" value="targetGasDensity+quencherDensity"/>
111         <P unit="bar" value="gasPressure"/>
112         <T unit="K" value="gasTemperature"/>
113         <fraction n="quencherFraction" ref="methane"/>
114         <fraction n="1-quencherFraction" ref="Xenon"/>
115         </material>
116         <material name="PureArgon" state="gas">
117         <T unit="K" value="gasTemperature"/>
118         <P unit="bar" value="gasPressure"/>
119         <MEE unit="eV" value="188"/>
120         <D unit="mg/cm3" value="targetGasDensity"/>
121         <fraction n="1" ref="Argon"/>
122         </material>
123         <material name="Ar_ISO" state="gas">
124         <D unit="mg/cm3" value="targetGasDensity+quencherDensity"/>
125         <P unit="bar" value="gasPressure"/>
126         <T unit="K" value="gasTemperature"/>
```

```
127         <fraction n="quencherFraction" ref="isobutane"/>
128         <fraction n="1-quencherFraction" ref="Argon"/>
129         </material>
130         <material name="Neon_ISO" state="gas">
131         <D unit="mg/cm3" value="targetGasDensity+quencherDensity"/>
132         <P unit="bar" value="gasPressure"/>
133         <T unit="K" value="gasTemperature"/>
134         <fraction n="quencherFraction" ref="isobutane"/>
135         <fraction n="1-quencherFraction" ref="Neon"/>
136         </material>
137         <material name="Helium_ISO" state="gas">
138         <D unit="mg/cm3" value="targetGasDensity+quencherDensity"/>
139         <P unit="bar" value="gasPressure"/>
140         <T unit="K" value="gasTemperature"/>
141         <fraction n="quencherFraction" ref="isobutane"/>
142         <fraction n="1-quencherFraction" ref="Helium"/>
143         </material>
144         <!--  }}}  -->
145   <!--  {{{ Copper  -->
146         <isotope N="63" Z="29" name="Cu63">
147         <atom unit="g/mole" value="62.9296"/>
148         </isotope>
149         <isotope N="65" Z="29" name="Cu65">
150         <atom unit="g/mole" value="64.9278"/>
151         </isotope>
152         <element name="Cu">
153         <fraction n="0.6917" ref="Cu63"/>
154         <fraction n="0.3083" ref="Cu65"/>
155         </element>
156         <material name="Copper" state="solid">
157         <MEE unit="eV" value="322"/>
158         <D unit="g/cm3" value="8.96"/>
159         <fraction n="1" ref="Cu"/>
160         </material>
161         <!--  }}}  -->
162   <!--  {{{ Stainlesssteel  -->
163         <material name="Stainlesssteel" state="solid">
164         <MEE unit="eV" value="280.927567486627"/>
165         <D unit="g/cm3" value="8.02"/>
166         <fraction n="0.02" ref="Manganese"/>
167         <fraction n="0.01" ref="Silicon"/>
168         <fraction n="0.19" ref="Chromium"/>
169         <fraction n="0.1" ref="Nickel"/>
170         <fraction n="0.68" ref="Iron"/>
171         </material>
172         <!--  }}}  -->
173   <material name="TitanPure" state="solid">
174   <D unit="g/cm3" value="4.6"/>
175   <fraction n="0.99475" ref="Titanium"/>
176   <fraction n="0.0003" ref="Nitrogen"/>
177   <fraction n="0.0010" ref="Carbon"/>
178   <fraction n="0.00015" ref="Hydrogen"/>
179   <fraction n="0.0020" ref="Iron"/>
180   <fraction n="0.0018" ref="Oxygen"/>
181   </material>
```

```
182
183   <!--  {{{ Air  -->
184         <isotope N="14" Z="7" name="N14">
185         <atom unit="g/mole" value="14.0031"/>
186         </isotope>
187         <isotope N="15" Z="7" name="N15">
188         <atom unit="g/mole" value="15.0001"/>
189         </isotope>
190         <element name="Nitrogen">
191         <fraction n="0.99632" ref="N14"/>
192         <fraction n="0.00368" ref="N15"/>
193         </element>
194         <isotope N="16" Z="8" name="O16">
195         <atom unit="g/mole" value="15.9949"/>
196         </isotope>
197         <isotope N="17" Z="8" name="O17">
198         <atom unit="g/mole" value="16.9991"/>
199         </isotope>
200         <isotope N="18" Z="8" name="O18">
201         <atom unit="g/mole" value="17.9992"/>
202         </isotope>
203         <element name="Oxygen">
204         <fraction n="0.99757" ref="O16"/>
205         <fraction n="0.00038" ref="O17"/>
206         <fraction n="0.00205" ref="O18"/>
207         </element>
208         <material name="Air" state="gas">
209         <T unit="K" value="293.13"/>
210         <MEE unit="eV" value="85.7030667332999"/>
211         <D unit="g/cm3" value="0.00129"/>
212         <fraction n="0.7" ref="Nitrogen"/>
213         <fraction n="0.3" ref="Oxygen"/>
214         </material>
215         <!--  }}}  -->
216   <!--  {{{ Vacuum  -->
217         <material name="Vacuum" state="gas">
218         <T unit="K" value="293.13"/>
219         <MEE unit="eV" value="85.7030667332999"/>
220         <D unit="g/cm3" value="1.e-20"/>
221         <fraction n="0.7" ref="Nitrogen"/>
222         <fraction n="0.3" ref="Oxygen"/>
223         </material>
224         <!--  }}}  -->
225   <!--  {{{ Water  -->
226         <element Z="1" formula="H" name="Hydrogen">
227         <atom value="2"/>
228         </element>
229         <element Z="8" formula="O" name="Oxygen">
230         <atom value="16"/>
231         </element>
232         <material name="Water" formula="H2O">
233         <D value="1.0"/>
234         <composite n="2" ref="Hydrogen"/>
235         <composite n="1" ref="Oxygen"/>
236         </material>
```

```
        <!--   }}}   -->


  </materials>
```

5. Simulación de un fondo de detección

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>

<restG4>

<TRestRun name="Background" title="Test">
<parameter name="experiment" value="TFM"/>
<parameter name="readOnly" value="false" />
<parameter name="runNumber" value="auto"/> <!-- Will be set by restG4ToSlurm/
    ToCondor.py -->
<parameter name="runDescription" value=""/>
<parameter name="user" value="${USER}"/>
<parameter name="verboseLevel" value="1"/>
<parameter name="overwrite" value="off" />
<parameter name="outputFileName" value="Runwater0.root" />
</TRestRun>


<TRestGeant4Metadata name="restG4 run" title="Gammawater">
<parameter name="gdml_file" value="Geometry/setupwater.gdml"/>
<parameter name="subEventTimeDelay" value="100us" />
<parameter name="Nevents" value="1000000" />


<!-- Externo -->

<generator type="virtualWall" position="(0,0,6000)mm" lenX="5000" lenY="5000" >
<source particle="gamma" fullChain="on">
<energyDist type="TH1D" file="${REST_PATH}/data/distributions/CosmicGammas.root"
    spctName="hLEgamma" />
<angularDist type="TH1D" file="${REST_PATH}/data/distributions/CosmicAngles.root"
    spctName="Theta2" />
</source>
</generator>


<storage sensitiveVolume="gas">
<parameter name="energyRange" value="(0,1000)" units="GeV" />
<activeVolume name="gas" chance="1" maxStepSize="0.2mm"/>
</storage>

</TRestGeant4Metadata>

<TRestGeant4PhysicsLists name="default" file="Common/physicslist.xml" />

</restG4>
```

6. Análisis del gas blanco

```
Int_t GasProperties (Double_t field, Double_t &Cd, Double_t &Vd,
    TRestDetectorGas *gas) // bar and V/mm
{
  gas->SetElectricField(field);
  Vd=gas->GetDriftVelocity();
  Cd=gas->GetLongitudinalDiffusion();
  return 0;
}
Double_t Maximo (Double_t Vec[], int N)
{  double aux=-10000000.0;
   int i;
   for (i=0; i<N ;i++){
     if (Vec[i]>aux)
     aux=Vec[i];}
  return aux;
 }
Double_t Minimo (Double_t Vec[], int N)
{ double aux=1000000.0;
  int i;
  for (i=0; i<N ;i++){
    if (Vec[i]<aux)
    aux=Vec[i];}
return aux;
}

Double_t Promedio(Double_t Vec[], int N)  //regla trapecio compuesto
{   Double_t aux=0.0;  // N numero de subintervalos y tamano del array
  for(int i=1; i<=N-2; i++){
    aux=aux+Vec[i];}
  aux=(2*aux)+Vec[0]+Vec[N-1];
  aux=aux/(2*N);
  return aux;
}


//######################################## MAIN

Int_t PrintGasPropertiesR ()
{
  auto Vdc3 = new TCanvas();
  auto Vdmg = new TMultiGraph();
  Vdmg->SetTitle("Velocidad de deriva; #it{r} [mm] ; #it{V_{d}} [mm/#mus]");

  auto Cdc3 = new TCanvas();
  auto Cdmg = new TMultiGraph();
  Cdmg->SetTitle("Coeficientes de difusi#acute{o}n; #it{r} [mm] ; #it{C_{d}} [cm
      ^{1/2}]");

  auto Tdc3 = new TCanvas();
  auto Tdmg = new TMultiGraph();
  Tdmg->SetTitle("Tiempo de deriva; #it{r} [mm] ; #it{T_{d}} [ms]");


```

```
52      Double_t Cdaux,Vdaux,Max,Min;
53      Double_t rmin=500;                          //en mm, el del detector
54      Double_t rmax=1500;                 //en mm, el de la esfera, considerando el
            grosor despreciable
55      Double_t voltage=750;                       //en V
56      Double_t pressure=10.;                       //bar
57      int p=rmax-rmin+1;                          //+1 para almacenar rmax
58      Double_t field;
59      Double_t Cd[p],Vd[p],E[p],Td[p],R[p];
60
61      // Calculamos las carateristicas para cada gas
62
63      TRestDetectorGas *gas1=new TRestDetectorGas ("server","Xenon-TMA 0.5Pct 0.1-10
            E3Vcm");
64      gas1->SetPressure(pressure);
65      for (int i=0;i<=rmax-rmin;i++)                                  //resolucion
            cada mm (delta_r=1)
66      {
67        R[i]=rmin+i;                                      //posicion en mm
68        field=(voltage/(R[i]*R[i]))*rmin*rmax*(1/(rmax-rmin));     //en V/mm
69        GasProperties(field,Cdaux,Vdaux,gas1);                     //en V/mm
70        Vd[i]=Vdaux;                                             //en mm/us
71        Cd[i]=Cdaux;                                          //en cm^(1/2)
72        Td[0]=1/(Vd[0]*1000); if (i>0)  {Td[i]=Td[i-1]+(1/(Vd[i]*1000));}    //0,1  y
              1000 para ms
73      }
74      cout << "Cd promedio:"<<Promedio(Cd,p)<<"          Tmax:"<<Td[p-1]<<endl;
75
76
77      auto Vdgr1 = new TGraph(p,R,Vd);
78      Vdgr1->SetTitle("Xenon-TMA 0.5Pct");
79      Vdgr1->SetMarkerStyle(kFullDotSmall);
80      Vdgr1->SetMarkerColor(1);
81      Vdgr1->SetLineColor(1);
82      Vdgr1->SetLineWidth(2);
83      auto Cdgr1 = new TGraph(p,R,Cd);
84      Cdgr1->SetTitle("Xenon-TMA 0.5Pct");
85      Cdgr1->SetMarkerStyle(kFullDotSmall);
86      Cdgr1->SetMarkerColor(1);
87      Cdgr1->SetLineColor(1);
88      Cdgr1->SetLineWidth(2);
89      auto Tdgr1 = new TGraph(p,R,Td);
90      Tdgr1->SetTitle("Xenon-TMA 0.5Pct");
91      Tdgr1->SetMarkerStyle(kFullDotSmall);
92      Tdgr1->SetMarkerColor(1);
93      Tdgr1->SetLineColor(1);
94      Tdgr1->SetLineWidth(2);
95
96
97      TRestDetectorGas *gas2=new TRestDetectorGas ("server","Xenon-TMA 2Pct 0.1-10
            E3Vcm");
98      gas2->SetPressure(pressure);
99      for (int i=0;i<=rmax-rmin;i++){
100       R[i]=rmin+i;
```

```
101       field=(voltage/(R[i]*R[i]))*rmin*rmax*(1/(rmax-rmin));  GasProperties(field,
              Cdaux,Vdaux,gas2);
102       Vd[i]=Vdaux;
103       Cd[i]=Cdaux;
104       Td[0]=1/(Vd[0]*1000);
105       if (i>0)
106         {Td[i]=Td[i-1]+(1/(Vd[i]*1000));}
107       }
108     cout << "Cd promedio:"<<Promedio(Cd,p)<<"          Tmax:"<<Td[p-1]<<endl;
109
110     auto Vdgr2 = new TGraph(p,R,Vd);
111     Vdgr2->SetTitle("Xenon-TMA 2Pct");
112     Vdgr2->SetMarkerStyle(kFullDotSmall);
113     Vdgr2->SetMarkerColor(2);
114     Vdgr2->SetLineColor(2);
115     Vdgr2->SetLineWidth(2);
116     auto Cdgr2 = new TGraph(p,R,Cd);
117     Cdgr2->SetTitle("Xenon-TMA 2Pct");
118     Cdgr2->SetMarkerStyle(kFullDotSmall);
119     Cdgr2->SetMarkerColor(2);
120     Cdgr2->SetLineColor(2);
121     Cdgr2->SetLineWidth(2);
122     auto Tdgr2 = new TGraph(p,R,Td);
123     Tdgr2->SetTitle("Xenon-TMA 2Pct");
124     Tdgr2->SetMarkerStyle(kFullDotSmall);
125     Tdgr2->SetMarkerColor(2);
126     Tdgr2->SetLineColor(2);
127     Tdgr2->SetLineWidth(2);
128
129     //#####################################
130
131     TRestDetectorGas *gas3=new TRestDetectorGas ("server","Xenon-Methane 0.5Pct
            0.1-10E3Vcm");
132     gas3->SetPressure(pressure);
133     for (int i=0;i<=rmax-rmin;i++){
134       R[i]=rmin+i;
135       field=(voltage/(R[i]*R[i]))*rmin*rmax*(1/(rmax-rmin));  GasProperties(field,
              Cdaux,Vdaux,gas2);
136       Vd[i]=Vdaux;
137       Cd[i]=Cdaux;
138       Td[0]=1/(Vd[0]*1000);
139       if (i>0)
140       {Td[i]=Td[i-1]+(1/(Vd[i]*1000));}
141     }
142     cout << "Cd promedio:"<<Promedio(Cd,p)<<"          Tmax:"<<Td[p-1]<<endl;
143
144     auto Vdgr3 = new TGraph(p,R,Vd);
145     Vdgr3->SetTitle("Xenon-Methane 0.5Pct");
146     Vdgr3->SetMarkerStyle(kFullDotSmall);
147     Vdgr3->SetMarkerColor(3);
148     Vdgr3->SetLineColor(3);
149     Vdgr3->SetLineWidth(2);
150     auto Cdgr3 = new TGraph(p,R,Cd);
151     Cdgr3->SetTitle("Xenon-Methane 0.5Pct");
152     Cdgr3->SetMarkerStyle(kFullDotSmall);
```

```
153     Cdgr3->SetMarkerColor(3);
154     Cdgr3->SetLineColor(3);
155     Cdgr3->SetLineWidth(2);
156     auto Tdgr3 = new TGraph(p,R,Td);
157     Tdgr3->SetTitle("Xenon-Methane 0.5Pct");
158     Tdgr3->SetMarkerStyle(kFullDotSmall);
159     Tdgr3->SetMarkerColor(3);
160     Tdgr3->SetLineColor(3);
161     Tdgr3->SetLineWidth(2);
162
163     TRestDetectorGas *gas4=new TRestDetectorGas("server","Xenon-Methane 2Pct
            0.1-10E3Vcm");
164     gas4->SetPressure(pressure);
165     for (int i=0;i<=rmax-rmin;i++){
166       R[i]=rmin+i;
167       field=(voltage/(R[i]*R[i]))*rmin*rmax*(1/(rmax-rmin));  GasProperties(field,
              Cdaux,Vdaux,gas2);
168       Vd[i]=Vdaux;
169       Cd[i]=Cdaux;
170       Td[0]=1/(Vd[0]*1000);
171       if (i>0)
172       {Td[i]=Td[i-1]+(1/(Vd[i]*1000));}
173     }
174     cout << "Cd promedio:"<<Promedio(Cd,p)<<"          Tmax:"<<Td[p-1]<<endl;
175
176
177     auto Vdgr4 = new TGraph(p,R,Vd);
178     Vdgr4->SetTitle("Xenon-Methane 2Pct");
179     Vdgr4->SetMarkerStyle(kFullDotSmall);
180     Vdgr4->SetMarkerColor(4);
181     Vdgr4->SetLineColor(4);
182     Vdgr4->SetLineWidth(2);
183     auto Cdgr4 = new TGraph(p,R,Cd);
184     Cdgr4->SetTitle("Xenon-Methane 2Pct");
185     Cdgr4->SetMarkerStyle(kFullDotSmall);
186     Cdgr4->SetMarkerColor(4);
187     Cdgr4->SetLineColor(4);
188     Cdgr4->SetLineWidth(2);
189     auto Tdgr4 = new TGraph(p,R,Td);
190     Tdgr4->SetTitle("Xenon-Methane 2Pct");
191     Tdgr4->SetMarkerStyle(kFullDotSmall);
192     Tdgr4->SetMarkerColor(4);
193     Cdgr4->SetLineColor(4);
194     Cdgr4->SetLineWidth(2);
195
196     //#########################################
197
198     TRestDetectorGas *gas5=new TRestDetectorGas("server","Xenon-Isobutane 0.5Pct
            0.1-10E3Vcm");
199     gas5->SetPressure(pressure);
200     for (int i=0;i<=rmax-rmin;i++){
201       R[i]=rmin+i;
202       field=(voltage/(R[i]*R[i]))*rmin*rmax*(1/(rmax-rmin));  GasProperties(field,
              Cdaux,Vdaux,gas2);
203       Vd[i]=Vdaux;
```

```
204         Cd[i]=Cdaux;
205         Td[0]=1/(Vd[0]*1000);
206         if (i>0)
207         {Td[i]=Td[i-1]+(1/(Vd[i]*1000));}
208       }
209       cout << "Cd promedio:"<<Promedio(Cd,p)<<"          Tmax:"<<Td[p-1]<<endl;
210
211       auto Vdgr5 = new TGraph(p,R,Vd);
212       Vdgr5->SetTitle("Xenon-Isobutane 0.5Pct");
213       Vdgr5->SetMarkerStyle(kFullDotSmall);
214       Vdgr5->SetMarkerColor(5);
215       Vdgr5->SetLineColor(5);
216       Vdgr5->SetLineWidth(2);
217       auto Cdgr5 = new TGraph(p,R,Cd);
218       Cdgr5->SetTitle("Xenon-Isobutane 0.5Pct");
219       Cdgr5->SetMarkerStyle(kFullDotSmall);
220       Cdgr5->SetMarkerColor(5);
221       Cdgr5->SetLineColor(5);
222       Cdgr5->SetLineWidth(2);
223       auto Tdgr5 = new TGraph(p,R,Td);
224       Tdgr5->SetTitle("Xenon-Isobutane 0.5Pct");
225       Tdgr5->SetMarkerStyle(kFullDotSmall);
226       Tdgr5->SetMarkerColor(5);
227       Tdgr5->SetLineColor(5);
228       Tdgr5->SetLineWidth(2);
229
230       TRestDetectorGas *gas6=new TRestDetectorGas ("server","Xenon-Isobutane 2Pct
              0.1-10E6Vcm");
231       gas6->SetPressure(pressure);
232       for (int i=0;i<=rmax-rmin;i++){
233         R[i]=rmin+i;
234         field=(voltage/(R[i]*R[i]))*rmin*rmax*(1/(rmax-rmin));  GasProperties(field,
              Cdaux,Vdaux,gas2);
235         Vd[i]=Vdaux;
236         Cd[i]=Cdaux;
237         Td[0]=1/(Vd[0]*1000);
238         if (i>0)
239         {Td[i]=Td[i-1]+(1/(Vd[i]*1000));}
240       }
241       cout << "Cd promedio:"<<Promedio(Cd,p)<<"          Tmax:"<<Td[p-1]<<endl;
242
243       auto Vdgr6 = new TGraph(p,R,Vd);
244       Vdgr6->SetTitle("Xenon-Isobutane 2Pct");
245       Vdgr6->SetMarkerStyle(kFullDotSmall);
246       Vdgr6->SetMarkerColor(6); Vdgr6->SetLineColor(6);
247       Vdgr6->SetLineWidth(2);
248       auto Cdgr6 = new TGraph(p,R,Cd);
249       Cdgr6->SetTitle("Xenon-Isobutane 2Pct");
250       Cdgr6->SetMarkerStyle(kFullDotSmall);
251       Cdgr6->SetMarkerColor(6);
252       Cdgr6->SetLineColor(6);
253       Cdgr6->SetLineWidth(2);
254       auto Tdgr6 = new TGraph(p,R,Td);
255       Tdgr6->SetTitle("Xenon-Isobutane 2Pct");
256       Tdgr6->SetMarkerStyle(kFullDotSmall);
```

```
257     Tdgr6->SetMarkerColor(6);
258     Tdgr6->SetLineColor(6);
259     Tdgr6->SetLineWidth(2);
260
261     //#####################################
262
263     TRestDetectorGas *gas7=new TRestDetectorGas ("server","Helium-Isobutane 0.5Pct
            0.1-10E3Vcm");
264     gas7->SetPressure(pressure);
265     for (int i=0;i<=rmax-rmin;i++){
266       R[i]=rmin+i;
267       field=(voltage/(R[i]*R[i]))*rmin*rmax*(1/(rmax-rmin));  GasProperties(field,
              Cdaux,Vdaux,gas2);
268       Vd[i]=Vdaux;
269       Cd[i]=Cdaux;
270       Td[0]=1/(Vd[0]*1000);
271       if (i>0)
272       {Td[i]=Td[i-1]+(1/(Vd[i]*1000));}
273     }
274     cout << "Cd promedio:"<<Promedio(Cd,p)<<"          Tmax:"<<Td[p-1]<<endl;
275
276     auto Vdgr7 = new TGraph(p,R,Vd);
277     Vdgr7->SetTitle("Helium-Isobutane 0.5Pct");
278     Vdgr7->SetMarkerStyle(kFullDotSmall);
279     Vdgr7->SetMarkerColor(7);
280     Vdgr7->SetLineColor(7);
281     Vdgr7->SetLineWidth(2);
282     auto Cdgr7 = new TGraph(p,R,Cd);
283     Cdgr7->SetTitle("Helium-Isobutane 0.5Pct");
284     Cdgr7->SetMarkerStyle(kFullDotSmall);
285     Cdgr7->SetMarkerColor(7);
286     Cdgr7->SetLineColor(7);
287     Cdgr7->SetLineWidth(2);
288     auto Tdgr7 = new TGraph(p,R,Td);
289     Tdgr7->SetTitle("Helium-Isobutane 0.5Pct");
290     Tdgr7->SetMarkerStyle(kFullDotSmall);
291     Tdgr7->SetMarkerColor(7);
292     Tdgr7->SetLineColor(7);
293     Tdgr7->SetLineWidth(2);
294
295     TRestDetectorGas *gas8=new TRestDetectorGas ("server","Helium-Isobutane 3Pct
            0.1-10E3Vcm");
296     gas8->SetPressure(pressure);
297     for (int i=0;i<=rmax-rmin;i++){
298       R[i]=rmin+i;
299       field=(voltage/(R[i]*R[i]))*rmin*rmax*(1/(rmax-rmin));  GasProperties(field,
              Cdaux,Vdaux,gas2);
300       Vd[i]=Vdaux;
301       Cd[i]=Cdaux;
302       Td[0]=1/(Vd[0]*1000);
303       if (i>0)
304       {Td[i]=Td[i-1]+(1/(Vd[i]*1000));}
305     }
306     cout << "Cd promedio:"<<Promedio(Cd,p)<<"          Tmax:"<<Td[p-1]<<endl;
307
```

```
308     auto Vdgr8 = new TGraph(p,R,Vd);
309     Vdgr8->SetTitle("Helium-Isobutane 3Pct");
310     Vdgr8->SetMarkerStyle(kFullDotSmall);
311     Vdgr8->SetMarkerColor(8);
312     Vdgr8->SetLineColor(8);
313      Vdgr8->SetLineWidth(2);
314     auto Cdgr8 = new TGraph(p,R,Cd);
315     Cdgr8->SetTitle("Helium-Isobutane 3Pct");
316     Cdgr8->SetMarkerStyle(kFullDotSmall);
317     Cdgr8->SetMarkerColor(8);
318     Cdgr8->SetLineColor(8);
319     Cdgr8->SetLineWidth(2);
320     auto Tdgr8 = new TGraph(p,R,Td);
321     Tdgr8->SetTitle("Helium-Isobutane 3Pct");
322     Tdgr8->SetMarkerStyle(kFullDotSmall);
323     Tdgr8->SetMarkerColor(8);
324     Tdgr8->SetLineColor(8);
325     Tdgr8->SetLineWidth(2);
326
327     //#####################################
328
329     TRestDetectorGas *gas9=new TRestDetectorGas ("server","Neon-Isobutane 0.5Pct
            0.1-10E3Vcm");
330     gas9->SetPressure(pressure);
331     for (int i=0;i<=rmax-rmin;i++){
332       R[i]=rmin+i;
333       field=(voltage/(R[i]*R[i]))*rmin*rmax*(1/(rmax-rmin));  GasProperties(field,
              Cdaux,Vdaux,gas2);
334       Vd[i]=Vdaux;
335       Cd[i]=Cdaux;
336       Td[0]=1/(Vd[0]*1000);
337       if (i>0)
338       {Td[i]=Td[i-1]+(1/(Vd[i]*1000));}
339     }
340     cout << "Cd promedio:"<<Promedio(Cd,p)<<"          Tmax:"<<Td[p-1]<<endl;
341
342     auto Vdgr9 = new TGraph(p,R,Vd);
343     Vdgr9->SetTitle("Neon-Isobutane 0.5Pct");
344     Vdgr9->SetMarkerStyle(kFullDotSmall);
345     Vdgr9->SetMarkerColor(1);
346     Vdgr9->SetLineColor(1);
347      Vdgr9->SetLineWidth(2);
348     auto Cdgr9 = new TGraph(p,R,Cd);
349     Cdgr9->SetTitle("Neon-Isobutane 0.5Pct");
350     Cdgr9->SetMarkerStyle(kFullDotSmall);
351     Cdgr9->SetMarkerColor(1);
352     Cdgr9->SetLineColor(1);
353     Cdgr9->SetLineWidth(2);
354     auto Tdgr9 = new TGraph(p,R,Td);
355     Tdgr9->SetTitle("Neon-Isobutane 0.5Pct");
356     Tdgr9->SetMarkerStyle(kFullDotSmall);
357     Tdgr9->SetMarkerColor(1);
358     Tdgr9->SetLineColor(1);
359     Tdgr9->SetLineWidth(2);
360
```

```
361     TRestDetectorGas *gas10=new TRestDetectorGas ("server","Neon-Isobutane 3Pct
            0.1-10E3Vcm");
362     gas10->SetPressure(pressure);
363     for (int i=0;i<=rmax-rmin;i++){
364       R[i]=rmin+i;
365       field=(voltage/(R[i]*R[i]))*rmin*rmax*(1/(rmax-rmin));  GasProperties(field,
              Cdaux,Vdaux,gas2);
366       Vd[i]=Vdaux;
367       Cd[i]=Cdaux;
368       Td[0]=1/(Vd[0]*1000);
369       if (i>0)
370       {Td[i]=Td[i-1]+(1/(Vd[i]*1000));}
371     }
372     cout << "Cd promedio:"<<Promedio(Cd,p)<<"          Tmax:"<<Td[p-1]<<endl;
373
374     auto Vdgr10 = new TGraph(p,R,Vd);
375     Vdgr10->SetTitle("Neon-Isobutane 3Pct");
376     Vdgr10->SetMarkerStyle(kFullDotSmall);
377     Vdgr10->SetMarkerColor(2);
378     Vdgr10->SetLineColor(2);
379     Vdgr10->SetLineWidth(2);
380     auto Cdgr10 = new TGraph(p,R,Cd);
381     Cdgr10->SetTitle("Neon-Isobutane 3Pct");
382     Cdgr10->SetMarkerStyle(kFullDotSmall);
383     Cdgr10->SetMarkerColor(2);
384     Cdgr10->SetLineColor(2);
385     Cdgr10->SetLineWidth(2);
386     auto Tdgr10 = new TGraph(p,R,Td);
387     Tdgr10->SetTitle("Neon-Isobutane 3Pct");
388     Tdgr10->SetMarkerStyle(kFullDotSmall);
389     Tdgr10->SetMarkerColor(2);
390     Tdgr10->SetLineColor(2);
391     Tdgr10->SetLineWidth(2);
392
393     //#####################################
394
395     TRestDetectorGas *gas11=new TRestDetectorGas ("server","Argon-Isobutane 0.5Pct
            0.1-10E3Vcm");
396     gas11->SetPressure(pressure);
397     for (int i=0;i<=rmax-rmin;i++){
398       R[i]=rmin+i;
399       field=(voltage/(R[i]*R[i]))*rmin*rmax*(1/(rmax-rmin));  GasProperties(field,
              Cdaux,Vdaux,gas2);
400       Vd[i]=Vdaux;
401       Cd[i]=Cdaux;
402       Td[0]=1/(Vd[0]*1000);
403       if (i>0)
404       {Td[i]=Td[i-1]+(1/(Vd[i]*1000));}
405     }
406     cout << "Cd promedio:"<<Promedio(Cd,p)<<"          Tmax:"<<Td[p-1]<<endl;
407
408     auto Vdgr11 = new TGraph(p,R,Vd);
409     Vdgr11->SetTitle("Argon-Isobutane 0.5Pct");
410     Vdgr11->SetMarkerStyle(kFullDotSmall);
411     Vdgr11->SetMarkerColor(11);
```

```
412     Vdgr11->SetLineColor(11);
413     Vdgr11->SetLineWidth(2);
414     auto Cdgr11 = new TGraph(p,R,Cd);
415     Cdgr11->SetTitle("Argon-Isobutane 0.5Pct");
416     Cdgr11->SetMarkerStyle(kFullDotSmall);
417     Cdgr11->SetMarkerColor(11);
418     Cdgr11->SetLineColor(11);
419     Cdgr11->SetLineWidth(2);
420     auto Tdgr11 = new TGraph(p,R,Td);
421     Tdgr11->SetTitle("Argon-Isobutane 0.5Pct");
422     Tdgr11->SetMarkerStyle(kFullDotSmall);
423     Tdgr11->SetMarkerColor(11);
424     Tdgr11->SetLineColor(11);
425     Tdgr11->SetLineWidth(2);
426
427     TRestDetectorGas *gas12=new TRestDetectorGas ("server","Argon-Isobutane 3Pct
            0.1-10E3Vcm");
428     gas12->SetPressure(pressure);
429     for (int i=0;i<=rmax-rmin;i++){
430       R[i]=rmin+i;
431       field=(voltage/(R[i]*R[i]))*rmin*rmax*(1/(rmax-rmin));  GasProperties(field,
              Cdaux,Vdaux,gas2);
432       Vd[i]=Vdaux;
433       Cd[i]=Cdaux;
434       Td[0]=1/(Vd[0]*1000);
435       if (i>0)
436       {Td[i]=Td[i-1]+(1/(Vd[i]*1000));}
437     }
438     cout << "Cd promedio:"<<Promedio(Cd,p)<<"          Tmax:"<<Td[p-1]<<endl;
439
440     auto Vdgr12 = new TGraph(p,R,Vd);
441     Vdgr12->SetTitle("Argon-Isobutane 3Pct");
442     Vdgr12->SetMarkerStyle(kFullDotSmall);
443     Vdgr12->SetMarkerColor(12);
444     Vdgr12->SetLineColor(12);
445     Vdgr12->SetLineWidth(2);
446     auto Cdgr12 = new TGraph(p,R,Cd);
447     Cdgr12->SetTitle("Argon-Isobutane 3Pct");
448     Cdgr12->SetMarkerStyle(kFullDotSmall);
449     Cdgr12->SetMarkerColor(12);
450     Cdgr12->SetLineColor(12);
451     Cdgr12->SetLineWidth(2);
452     auto Tdgr12 = new TGraph(p,R,Td);
453     Tdgr12->SetTitle("Argon-Isobutane 3Pct");
454     Tdgr12->SetMarkerStyle(kFullDotSmall);
455     Tdgr12->SetMarkerColor(12);
456     Tdgr12->SetLineColor(12);
457     Tdgr12->SetLineWidth(2);
458
459     // Mostramos la velocidad de deriva, coeficiente de difusion o tiempo de
            deriva
460
461     Vdmg->Add(Vdgr3); Vdmg->Add(Vdgr4);
462     Vdmg->Add(Vdgr7); Vdmg->Add(Vdgr8);
463     Vdmg->Add(Vdgr9);  Vdmg->Add(Vdgr10);
```

```
464     Vdmg->Add(Vdgr11); Vdmg->Add(Vdgr12);   Vdmg->Draw("ACP");
465
466     Cdmg->Add(Cdgr3); Cdmg->Add(Cdgr4);
467     Cdmg->Add(Cdgr7); Cdmg->Add(Cdgr8);
468     Cdmg->Add(Cdgr9);   Cdmg->Add(Cdgr10);
469     Cdmg->Add(Cdgr11); Cdmg->Add(Cdgr12);   Cdmg->Draw("ACP");
470
471     Tdmg->Add(Tdgr3); Tdmg->Add(Tdgr4);
472     Tdmg->Add(Tdgr7); Tdmg->Add(Tdgr8);
473     Tdmg->Add(Tdgr9);   Tdmg->Add(Tdgr10);
474     Tdmg->Add(Tdgr11); Tdmg->Add(Tdgr12);   Tdmg->Draw("APE");
475
476     auto legendCd = new TLegend(0.1,0.6,0.48,0.9);    //<-x1   ^| y1    x2->   (x1,y1,
            x2,y2)
477
478     Cdgr3->SetMarkerStyle(kFullDotLarge);
479     Cdgr4->SetMarkerStyle(kFullDotLarge);
480     Cdgr7->SetMarkerStyle(kFullDotLarge);
481     Cdgr8->SetMarkerStyle(kFullDotLarge);
482     Cdgr9->SetMarkerStyle(kFullDotLarge);
483     Cdgr10->SetMarkerStyle(kFullDotLarge);
484     Cdgr11->SetMarkerStyle(kFullDotLarge);
485     Cdgr12->SetMarkerStyle(kFullDotLarge);
486
487     legendCd->AddEntry(Cdgr3,"","p");   legendCd->AddEntry(Cdgr4,"","p");
488     legendCd->AddEntry(Cdgr7,"","p");   legendCd->AddEntry(Cdgr8,"","p");
489     legendCd->AddEntry(Cdgr9,"","p");   legendCd->AddEntry(Cdgr10,"","p");
490     legendCd->AddEntry(Cdgr11,"","p");  legendCd->AddEntry(Cdgr12,"","p");
491     legendCd->Draw();
492
493     return 0;
494   }
```

7. Estudio de los eventos

```
1
2   #include "TRestGeant4Event.h"
3   #include "TRestGeant4Metadata.h"
4   #include "TRestTask.h"
5   #include <TH3D.h>
6   #include <TMath.h>
7   #include <chrono>
8   using namespace std;
9   using namespace std::chrono;
10
11  Double_t rmin=500, rmax=1500;
12  Double_t resolution=0.001;                  //en mm
13
14  Double_t Search (Double_t target, Int_t n, const std::vector<Double_t> &A){
15    Int_t L = 0;
16    Int_t R = n-1;
17    Int_t mid, ans= -1;
18
```

```
19      while( L<=R ) {
20        mid= L+((R-L)/2);
21        if(A[mid] >= target){
22          ans = mid;
23          R=mid-1;}
24        else L=mid+1;
25      }
26      return ans;
27    }
28    Int_t Single(std::string File,Double_t EventNumber, TH1 *h1, Int_t sizeR, Int_t
        sizeT, Int_t sizeP, const std::vector<Double_t> &Td, const std::vector<
        Double_t> &Rd, Double_t Tmax, Double_t LimTh1, Double_t BinsTh1, Int_t *
        Exist, TH1 *h1r2, TH1 *h1reduce, Int_t *Existred, Int_t *Existr2, FILE *Data
        ){
29      TString fName=File;
30      cout<<"\n Archivo: "<<fName<<"\n"<<endl;
31      TRestRun* run = new TRestRun();
32      Double_t pi=TMath::Pi();
33      Double_t rho=0,theta=0,phi=0;
34      Double_t Xhit=0,Yhit=0,Zhit=0,Ehit=0;
35
36      string fname = fName.Data();
37      if (!TRestTools::fileExists(fname)) {
38        cout << "WARNING. Input file does not exist" << endl;
39      } else  run->OpenInputFile(fName);
40
41      //run->PrintMetadata();
42
43      ////////// Escogemos eventos aleatorios de la base de datos
44
45      TRestGeant4Metadata *g4=(TRestGeant4Metadata*)run->GetMetadataClass("
          TRestGeant4Metadata");
46      Double_t Generated=g4->GetNumberOfEvents();       //lanzados en simulacion
47      Double_t Detected=run->GetEntries();              //los almacenados
48      Double_t GeneratedReal=EventNumber;                     //los recibidos en
          nuestro tiempo deriva
49      Double_t DetectedRealOriginal=GeneratedReal*Detected/Generated ;    //los que
          deberiamos haber detectado
50      Double_t DetectedReal=0;
51
52      TRandom3 *r3 = new TRandom3();  r3->SetSeed(0);
53      DetectedReal=r3->Poisson(DetectedRealOriginal);
54
55      cout << "\n#######Eventos lanzados(sim): "<<Generated<<"    Eventos finales(sim
          ): "<<Detected<<endl;
56      cout << "\n#######Eventos lanzados: "<<GeneratedReal<<"    Eventos finales: "<<
          DetectedRealOriginal<<"  Tras Poisson: "<<DetectedReal<<endl;
57
58      TRandom* aleatorio= new TRandom();
59      aleatorio->SetSeed(0);
60      UInt_t var;
61      Bool_t repe=false;
62      std::vector<Int_t> Selected;
63
64      if(DetectedReal <= Detected)
```

```
65      {for (int v=0; v<DetectedReal;v++)
66        //cout << v <<endl;
67        {
68          do
69          {var=aleatorio->Integer(run->GetEntries());
70            repe=false;
71            for (int i=0;i<Selected.size();i++)
72            {if (var == Selected[i])
73              repe=true;
74            }
75          }
76          while (repe != false);
77          Selected.push_back(var);
78        }  //cout<<"Eventos escogidos"<<endl;
79      } else if (Detected > DetectedReal )
80      {cout<<"No hay eventos suficientes en: "<<fName<<endl;
81
82      } else    {cout<<"No hay eventos. Eventos:"<<run->GetEntries()<<" Archivo: "<<
          fName<<endl;}
83
84
85      //////////////  Guardar y convertir hits a Td,theta,phi
86
87      TRestHits* TallHits = new TRestHits();
88      TRestHits* XallHits = new TRestHits();
89      TRestGeant4Event* ev = new TRestGeant4Event();
90      run->SetInputEvent(ev);
91
92      Double_t aux=0.;
93      Int_t index=0;
94
95      for (int i = 0; i < Selected.size(); i++)
96      {
97        run->GetEntry(Selected[i]);
98        //ev->PrintEvent();
99        for (int n = 0; n < ev->GetNumberOfTracks(); n++)
100       {
101         TRestGeant4Hits* hits = ev->GetTrack(n)->GetHits();
102
103         for (int m = 0; m < hits->GetNumberOfHits(); m++)
104         {
105           if((hits->GetEnergy(m))>0)
106           {
107             Xhit=hits->GetX(m);   Yhit=hits->GetY(m);   Zhit=hits->GetZ(m);   Ehit=
                  hits->GetEnergy(m);
108             TVector3 v(Xhit,Yhit,Zhit);
109             rho=v.Mag();
110
111             if (rho >= rmin && rho<=rmax)
112             {
113               theta=v.Theta(); phi=v.Phi();
114               index=(rho-rmin)/resolution;
115               aux=Td[index];
116               TallHits->AddHit(aux,theta,phi,Ehit);  // asi es tiempo de deriva
117             }
```

```
118             }
119           }
120         }
121       }
122       cout << "Hits:"<<TallHits->GetNumberOfHits()<<endl;
123       cout <<"HITS CARGADOS"<<endl;
124 
125       ////////////////////  Clusterizado
126 
127       TRestMesh *Tmesh = new TRestMesh(TVector3(2*Tmax,2*Tmax,2*Tmax), TVector3
              (0,0,0),sizeR,sizeT,sizeP);
128       Tmesh->SetSpherical(true);
129       Tmesh->SetNodesFromSphericalHits(TallHits);
130       cout << "TMesh generada"<<endl;
131 
132       cout<<"Numero de nodos:"<<Tmesh->GetNumberOfNodes()<<endl;
133       Int_t TnTracksFound = Tmesh->GetNumberOfGroups();  //numero de grupos
134       cout <<"Tracks:"<<TnTracksFound<<endl;
135 
136       /////////// Calculating the energy of each track
137 
138       cout<<"Clusterizando energia"<<endl;
139       Int_t i=0;
140       Double_t RhoLimit=1300.;
141       Double_t TdLimit=0.;  //Valor de rho que fiducializa y su equivalente en
              tiempo de deriva
142 
143       index=(RhoLimit-rmin)/resolution;
144       TdLimit=Td[index];
145       TVector3 v(0,0,0);
146 
147       fprintf(Data,"%d\n",TnTracksFound);
148       for (int n = 0; n < TnTracksFound; n++){
149         v=Tmesh->GetGroupPosition(n);        //posicion del track promedio
150         Double_t td=v.Mag();                 //Td promedio del track
151         index = Search(td,Td.size(),Td);     //indice del valor
152         rho=Rd[index];                       //Rho promedio
153         h1r2->Fill(rho/1000);
154         *Existr2=1;
155         Double_t Energy= Tmesh->GetGroupEnergy(n);
156         h1->Fill(Energy);
157         fprintf(Data,"%f \t %f \n",Energy,rho/1000);
158         if (Energy <= LimTh1 ){
159           i++;
160           *Exist=1;     //para anadir de la leyenda
161           //cout << "Track " << n << ", E: " << Energy <<" , Td: "<<td<< ", Rho: "<<
                  rho <<", Rho^2: "<<pow((rho/1000),2)<< endl;
162         }
163         if( rho < RhoLimit && Energy <= LimTh1){  //FIDUCIALIZACION
164           h1reduce->Fill(Energy);
165           *Existred=1;
166         }
167 
168       }
169       ///////////  Normalizacion
```

```
170     h1->Scale(BinsTh1);
171     h1reduce->Scale(BinsTh1);
172     cout<<"Tracks con E<"<<LimTh1<<" : "<<i<<endl;
173     cout<<"Energia clusterizada en TMesh"<<endl;
174
175
176     delete ev;
177     delete run;
178     return 0;
179   }
180
181
182   Int_t SphereSegmentation(){
183
184     const Int_t EventTypes=22;          //numero de tipos de particulas generadas
185
186     //gamma, n mu, vessel, gas
187     //Xe-0 to 10-5-Cu-1 (cambiar el nombre de la carpera)
188     Double_t EventNumber[EventTypes]=
189     //{279,139,35,17,35,  1.74e-3,5.96e-3,8.86e-3  ,3.92e-3,5.72e-3,2.24e-2,1.56e
            -2,7.53e-2,1.16e-1,6.89e-3,4.94e-2   ,3.15e-4,3.19e-4,5.08e-4,2.11e-4,1.84
            e-4,1.62e-3} ;
190     {326726,163363,40841,20475,40625,  2.04,6.98,10.4,  5,7,26,18,88,136,8,58,
            3.69e-1,3.74e-1,5.96e-1,2.48e-1,2.16e-1,2} ;
191
192     // Xe-1-5-Cu-10 y Xe-10-5-Cu-10
193     //{1349,674,169,85,168,  8.43e-3,2.88e-2,4.29e-2,   1.89e-2,2.76e-2,1.08e
            -1,7.56e-2,3.64e-1,5.6e-1,3.33e-2,2.39e-1,  1.52e-2,1.54e-2,2.46e-2,1.02e
            -2,8.92e-3,7.81e-2};
194     //{326726,163363,250941,20475,40625,   2.04,6.98,10.4,  5,7,26,18,88,136,8,58,
              4,4,6,2,2,19}         ;
195
196     //Xe-1-10-Cu-100
197     //{4685,2342,586,294,582,  6.05e-2,2.07e-1,3.08e-1,  6.58e-2,9.6e-2,3.76e
            -2,2.63e-1,1,2,1.16e-1,8.3e-1,  5.29e-1,5.36e-1,8.54e-1,3.55e-1,3.1e-1,3}
                                    ;
198     //{326726,163363,40841,20475,40625,   4,14,21,  5,7,26,18,88,136,8,58,
            37,37,60,25,22,189} ;
199
200     //Xe-1-10-St-115
201     //{4812,2406,601,302,598,       21,1,2,12,1,3,  6.25e-1,6.33e-1,1,4.19e-1,3.66
            e-1,3}         ;
202     //{326726,163363,40841,20475,40625,          1396,68,124,836,74,186,
            42,43,69,28,25,218}     ;
203
204     //Xe-1-10-Titan-90
205     //{4578,2289,572,287,569,       8,3.59e-7,   4.66e-1,4.71e-1,7.51e-1,3.12e
            -1,2.73e-1,2}  ;
206     //{326726,163363,40841,20475,40625,    576,2.56e-5,   33,34,54,22,19.5,170}
                ;
207
208     //He-0-5-Cu-1
209     //{40,20,5,3,5,   2.52e-4,8.59e-4,1.28e-3,   5.65e-4,8.25e-4,3.23e-3,2.26e
            -3,1.09e-2,1.67e-2,9.95e-4,7.13e-3   };
```

```
210     //{326726,163363,40841,20475,40625,  2.04,6.98,10.4,
            4.59,6.7,26.2,18.3,88.3,136,8.08,57.9  } ;
211
212     //Ne-0-5-Cu-1
213     //{8,4,1,0.53,1,    5.25e-5,1.79e-4,2.67e-4,    1.18e-4,1.72e-4,6.73e-4,4.71e
            -4,2.27e-3,3.49e-3,2.08e-4,1.49e-3    } ;
214     //{326726,163363,40841,20475,40625,  2.04,6.98,10.4,
            4.59,6.7,26.2,18.3,88.3,136,8.08,57.9};
215
216     //Xe-2500-5-Cu-10
217     //{1349,674,169,85,6.71e-3,  8.43e-3,2.88e-2,4.29e-2,   3.52e-5,4.83e-16,7.13e
            -3,8.7e-5,2.11e-2,3.69e-4,3.61e-7,1.17e-2,  3.78e-4,1.04e-6,1.76e-3,2.14e
            -4,8.39e-5,1.41e-9}                                    ;
218     //{326726,163363,40841,20475,2,   2,7,10,  8.52e-3,1.17e-13,2,2.11e-2,5,8.94e
            -2,8.76e-5,2.83,  9.16e-2,2.52e-4,4.26e-1,5.19e-2,2.03e-2,3.43e-7};
219
220
221
222     Int_t Color[EventTypes]=
223     { 803,793,632,600,430              //cosmic
224       ,416,419,828                   //vessel cobre
225       ,594,875,616,810,892,619,871,886      //activar cobre
226       //,892,594,875,616,810,619          //activar steel
227       //,594,828                   //activar titanio
228       ,623,920,396,392,832,831          //activar xenon
229     };
230
231
232     Char_t Title[EventTypes][40]={
233       "#gamma (1<E<10) MeV","#gamma (10<E<100) MeV","#gamma (0.1<E<100) GeV","
              Neutrones", "Muones",
234       "{}^{238}U","{}^{232}Th","{}^{40}K",
235       "{}^{46}Sc","{}^{48}V","{}^{54}Mn","{}^{56}Co","{}^{57}Co","{}^{58}Co","
              {}^{59}Fe","{}^{60}Co",
236       //"{}^{7}Be","{}^{46}Sc","{}^{48}V","{}^{54}Mn","{}^{56}Co","{}^{58}Co",
237       //"{}^{46}Sc","{}^{40}K",
238       "{}^{3}H","{}^{7}Be","{}^{125}Sb","{}^{121}Te","{}^{123}Te","{}^{127}Xe"
239     };
240
241
242     TCanvas *c1 = new TCanvas("c1","c1",700,500);
243     TCanvas *c2 = new TCanvas("c2","c2",700,500);
244     TCanvas *c3 = new TCanvas("c3","c3",700,500);
245     Int_t sizeR=450;
246     Int_t sizeT=150;
247     Int_t sizeP=100;
248     Double_t pressure=10.;
249     Double_t LimTh1=200;  // en KeV
250     Double_t BinsTh1=0.5;  // numero de bins por cada kev
251     string File="Xe-10-5-Cu-1";
252     string FileData=File+"data.md";
253     string Filepdf=File+".pdf";
254     string Filepdfr2=File+"r2.pdf";
255     string Filepdfreduce=File+"reduce.pdf";
256     gROOT->SetBatch(kTRUE);
```

```
257
258     char NameFile[100], NameFile2[100], NameFile3[100];
259     strcpy(NameFile,Filepdf.c_str());
260     strcpy(NameFile2,Filepdfr2.c_str());
261     strcpy(NameFile3,Filepdfreduce.c_str());
262
263     char NameData[100];
264     strcpy(NameData,FileData.c_str());
265     FILE *Data;
266     Data=fopen(NameData,"w");
267     //// Conversion entre R y Td
268     Double_t Vdaux,field;
269     Double_t voltage=750;
270     std::vector<Double_t> Vd;
271     std::vector<Double_t> Td;
272     std::vector<Double_t> Rd;
273
274     ////////////  Obtener lista de td y su posicion asociada
275     TRestDetectorGas *gas=new TRestDetectorGas ("server", "Xenon-Methane 0.5Pct
            0.1-10E3Vcm");
276     gas->SetPressure(pressure);
277     for (double i=0;i<=rmax-rmin+resolution;i=i+resolution){
278       Rd.push_back(rmin+i);
279       field=(voltage/((rmin+i)*(rmin+i)))*rmin*rmax*(1/(rmax-rmin));
280       gas->SetElectricField(field);
281       Vdaux=gas->GetDriftVelocity();
282       Vd.push_back(Vdaux);
283       if (i==0) {Td.push_back(resolution/(Vd[0]*1000));    }
284       if (i>0)  {Td.push_back((Td.back())+(resolution/(Vdaux*1000))); }
285     }
286     Double_t Tmax=Td.back();
287     cout<<"Tmax (ms):"<<Tmax<<endl;
288     cout <<"Conversion de R a t correcta"<<endl;
289
290     ////////////////////////// Plot 1D
291
292     THStack *hs= new THStack("hs","; #it{E} [keV]; Eventos keV^{-1} s^{-1}");
293     THStack *hsr2= new THStack("hsr2","; #it{r} [m]; Eventos s^{-1}");
294     THStack *hsreduce= new THStack("hsreduce","; #it{E} [keV]; Eventos keV^{-1} s
            ^{-1}")
295
296     Int_t Exist, Existred, Existr2;
297     std::vector<Int_t> Exist2;
298     std::vector<Int_t> Exist2red;
299     std::vector<Int_t> Exist2r2;
300     for (int i=5;i<EventTypes;i++){
301       string aux=std::to_string(i);
302       string FileRoot=File+"/Runwater"+aux+".root";
303       TH1* h1 = new TH1D("h1",";#it{E} [keV]; Eventos keV^{-1} s^{-1}",BinsTh1*
              LimTh1,0,LimTh1);
304       TH1* h1r2  = new TH1D("h1r2","#it{r} [m]; Eventos s^{-1}",(rmax-rmin)/10 ,
              rmin/1000,rmax/1000);
305       TH1* h1reduce = new TH1D("h1reduce",";#it{E} [keV]; Eventos keV^{-1} s^{-1}"
              ,BinsTh1*LimTh1,0,LimTh1);
306       h1r2->SetStats(0);
```

```
307         h1r2->SetMarkerColor(Color[i]);
308         h1r2->SetFillColor(Color[i]);
309         h1->SetStats(0);
310         h1->SetMarkerColor(Color[i]);
311         h1->SetFillColor(Color[i]);
312         h1reduce->SetStats(0);
313         h1reduce->SetMarkerColor(Color[i]);
314         h1reduce->SetFillColor(Color[i]);
315         Exist=0;  Existred=0;  Existr2=0;
316         Single(FileRoot,(EventNumber[i]/13),h1,sizeR,sizeT,sizeP,Td,Rd,Tmax,LimTh1,
                BinsTh1,&Exist,h1r2,h1reduce,&Existred,&Existr2,Data);
317
318         c1->cd();  h1->Draw("HIST BAR");  hs->Add(h1);
319         c2->cd();  h1r2->Draw("HIST BAR");  hsr2->Add(h1r2);
320         c3->cd();  h1reduce->Draw("HIST BAR");  hsreduce->Add(h1reduce);
321         if (Exist != 0 ){  Exist2.push_back(1);}  else Exist2.push_back(0);
322         if (Existred != 0 ){  Exist2red.push_back(1);}  else Exist2red.push_back(0);
323         if (Existr2 != 0 ){  Exist2r2.push_back(1);}  else Exist2r2.push_back(0);
324
325     }
326
327     fclose(Data);
328     ///////////////  Leyenda
329
330     Exist=0;
331     Existred=0;
332     Existr2=0;
333     for (int i=0; i < Exist2.size(); i++)
334     Exist+=Exist2[i];           //exist pasa a ser el numero de eventos no nulos
            total
335     for (int i=0; i< Exist2red.size(); i++)
336     Existred+=Exist2red[i];
337     for (int i=0; i < Exist2r2.size(); i++)
338     Existr2+=Exist2r2[i];
339
340     Double_t aux=0.35+(0.025*(EventTypes-Exist));
341     Double_t auxred=0.35+(0.025*(EventTypes-Existred));
342     Double_t auxr2=0.35+(0.025*(EventTypes-Existr2));
343     TLegend *legend1 = new TLegend(0.80,aux,0.98,0.9);
344     TLegend *legend1red = new TLegend(0.80,auxred,0.98,0.9);
345     TLegend *legend1r2 = new TLegend(0.80,auxr2,0.98,0.9);
346
347     legend1->SetTextSizePixels(23);
348     legend1red->SetTextSizePixels(23);
349     legend1r2->SetTextSizePixels(23);
350
351     for (int i=0;i <EventTypes;i++){
352       if (Exist2[i]==1. ){
353         TH1* h1aux = new TH1D("h1aux","; ; ",1,0,2);
354         h1aux->SetMarkerColor(Color[i]);
355         h1aux->SetFillColor(Color[i]);
356         legend1->AddEntry(h1aux,Title[i],"f");
357       }
358       if (Exist2red[i]==1. ){
359         TH1* h1auxred = new TH1D("h1auxred","; ; ",1,0,2);
```

```
360         h1auxred->SetMarkerColor(Color[i]);
361         h1auxred->SetFillColor(Color[i]);
362         legend1red->AddEntry(h1auxred,Title[i],"f");
363       }
364       if (Exist2r2[i]==1. ){
365         TH1* h1auxr2 = new TH1D("h1auxr2","; ; ",1,0,2);
366         h1auxr2->SetMarkerColor(Color[i]);
367         h1auxr2->SetFillColor(Color[i]);
368         legend1r2->AddEntry(h1auxr2,Title[i],"f");
369       }
370     }
371     ////////////////////// Senal de neutrinos
372
373     Double_t pi=TMath::Pi();
374     Double_t J=31*pow(pi,6)/252 , T ,Landa;
375     T=3.500;     Landa=0.89;    //nu_e, el mas dificil de ver al ser el promedio mas
          bajo de energia
376     Double_t U = (5e52)* 624151;
377     Double_t LANDA=U/T;
378     TF1 *distr = new TF1("distr","[0]*pow((x),4)/ ( [1]*pow([2],5) * (1+exp((x)
          /[2])) )",0,80);
379     distr->SetParameters(LANDA,J,T);
380
381     T=5.000;   Landa=0.63;    LANDA=U/T;      //anti nu_e
382     TF1 *distrAnti = new TF1("distr","[0]*pow((x),4)/ ( [1]*pow([2],5) * (1+exp((x
          )/[2])) )",0,80);
383     distrAnti->SetParameters(LANDA,J,T);
384
385     T=8.000;   Landa=0.39;   LANDA=U/T;     //otros
386     TF1 *distrOtros = new TF1("distr","[0]*pow((x),4)/ ( [1]*pow([2],5) * (1+exp((
          x)/[2])) )",0,80);
387     distrOtros->SetParameters(LANDA,J,T);
388
389     //distr->GetXaxis()->SetTitle("E_{#nu} [MeV]"); distr->GetYaxis()->SetTitle("#
          frac{dN}{dE_{#nu}}");
390     //distr->SetTitle("");   distr->SetLineColor(kBlack);
391     //TAxis *distraxis=distr->GetYaxis();
392     //distraxis->SetNdivisions(503);
393     //distraxis->SetMaxDigits(2);
394     //distr->GetYaxis()->SetTitleOffset(1.1);
395     //distr->Draw();
396     //distrAnti->SetLineColor(kRed); distrAnti->Draw("SAME");
397     //distrOtros->SetLineColor(kBlue);   distrOtros->Draw("SAME");
398     //TLegend *distrleg =new TLegend(0.72,0.7,0.9,0.9);
399     //distrleg->AddEntry(distr,"#nu_{e}","l");                distrleg->AddEntry((
          TObject*)0, "", "");
400     //distrleg->AddEntry(distrAnti,"#bar{#nu}_{e}","l");     distrleg->AddEntry((
          TObject*)0, "", "");
401     //distrleg->AddEntry(distrOtros,"Otros sabores","l");
402     //distrleg->Draw();
403
404     Double_t angle=0.;
405     TRandom3 *angleaux= new TRandom3();
406     angleaux->SetSeed(0);
407     angle= (angleaux->Rndm())*pi/2;
```

```
408     Int_t Z=10;  Int_t N=10;  Int_t A=N+Z;     Double_t mn=(N*939.565560)+(Z
            *938.272013); // en MeV
409     TF1 *recoilNe=new TF1("re1","(2*[0]*pow((x*TMath::Cos([1])),2))/(pow(([0]+x)
            ,2)-pow((x*TMath::Cos([1])),2))",0,80);  recoilNe->SetParameters(mn,angle)
            ;
410
411     Z=2; N=2; A=N+Z;     mn=(N*939.565560)+(Z*938.272013);
412     TF1 *recoilHe=new TF1("re2","(2*[0]*pow((x*TMath::Cos([1])),2))/(pow(([0]+x)
            ,2)-pow((x*TMath::Cos([1])),2))",0,80);  recoilHe->SetParameters(mn,angle)
            ;
413
414     Z=54;  N=77;  A=N+Z;  mn=(N*939.565560)+(Z*938.272013);
415     TF1 *recoilXe=new TF1("re3","(2*[0]*pow((x*TMath::Cos([1])),2))/(pow(([0]+x)
            ,2)-pow((x*TMath::Cos([1])),2))",0,80);
416     recoilXe->SetParameters(mn,angle);
417
418     //recoilHe->SetTitle(""); recoilHe->SetLineColor(kRed);  recoilHe->Draw();
419     //recoilHe->GetXaxis()->SetTitle("E_{#nu} [MeV]");  recoilHe->GetYaxis()->
            SetTitle("E_{recoil} [MeV]");
420     //recoilNe->SetLineColor(kBlack);  recoilNe->Draw("SAME");
421     //recoilXe->SetLineColor(kBlue);  recoilXe->Draw("SAME");
422     //TLegend *recleg=new TLegend(0.1,0.7,0.25,0.9);
423     //recleg->AddEntry(recoilHe,"Helio","l");        recleg->AddEntry((TObject*)0,
            "", "");
424     //recleg->AddEntry(recoilNe,"Neon","l");       recleg->AddEntry((TObject*)0,
            "", "");
425     //recleg->AddEntry(recoilXe,"Xenon","l");
426     //recleg->Draw();
427
428     TRandom3 *r3 = new TRandom3();
429     r3->SetSeed(0);
430     TH1D *h1 = new TH1D("h1",";#it{E} [KeV]; Eventos",BinsTh1*LimTh1,0,LimTh1);
431     Double_t NuEvents=153*pow((N/22),2)*(pressure/10)*pow((rmax/(1000*4)),3)
            *(300/293.15)*(Time/10000); //Recordar cambiar Time
432     if ( NuEvents<1.0)
433     NuEvents=r3->Poisson(NuEvents);
434     cout<<"Eventos de neutrinos: "<<NuEvents<<endl;
435
436     for (int i = 0;i < NuEvents; i++){
437       Double_t aux = distr->GetRandom(); //devuelve E en MeV
438       Double_t AuxRecoil=(recoilXe->Eval(aux))*1000; //en KeV
439       h1->Fill(AuxRecoil);
440       cout<<"E generada [MeV]: "<<aux<<"\nE recoil [keV]: "<<AuxRecoil<<endl;
441     }
442     h1->SetStats(0);
443     h1->SetMarkerColor(1);
444     h1->SetFillColor(1);
445     h1->Draw("HIST BAR");
446     hs->Add(h1);
447
448     ////////////////////  Graficar
449
450     c1->cd();
451     TH1D *last=(TH1D*)hs->GetStack()->Last();  hs->SetMaximum((1.25)*(last->
            GetBinContent(last->GetMaximumBin())));  hs->Draw("HIST BAR");
```

```
452     gPad->RedrawAxis("Y");
453     legend1->Draw();
454 
455     c2->cd();
456     TH1D *lastr2=(TH1D*)hsr2->GetStack()->Last();   hsr2->SetMaximum((1.25)*(lastr2
            ->GetBinContent(lastr2->GetMaximumBin())));   hsr2->Draw("HIST BAR");
457     gPad->RedrawAxis("Y");
458     legend1r2->Draw();
459 
460     c3->cd();
461     TH1D *lastreduce=(TH1D*)hsreduce->GetStack()->Last();   hsreduce->SetMaximum
            ((1.25)*(last->GetBinContent(last->GetMaximumBin())));   hsreduce->Draw("
            HIST BAR");
462     gPad->RedrawAxis("Y");
463     legend1red->Draw();
464 
465     //last->Draw("HIST BAR");
466 
467     c1->cd();   c1->SaveAs(NameFile);   c1->Close();
468     c2->cd();   c2->SaveAs(NameFile2);   c2->Close();
469     c3->cd();   c3->SaveAs(NameFile3);   c3->Close();
470 
471     /////////   Optimizacion tamano de la mesh de clusterizado
472 
473     BinsTh1=1;
474     string FileRoot="Xe-0-5-Cu-1/CalibrationMesh.root";
475     string FilePdf="Calibration.pdf";
476     TH1* h1r2  = new TH1D("h1r2","#it{r^{2}} [m^{2}]; Eventos s^{-1}",(rmax-rmin)
            /10 ,pow((rmin/1000),2),pow((rmax/1000),2));
477     TH1* h1reduce = new TH1D("h1reduce",";#it{E} [keV]; Eventos keV^{-1} s^{-1}",
            BinsTh1*LimTh1,0,LimTh1);
478     Double_t aux=0, aux2=0, aux3=0, aux4=0, aux5=0, aux6=0;
479     Double_t Efi5, Efi10, Efi15, Efi20, Efi25, EfiReal;
480     TMultiGraph *mg= new TMultiGraph("mg","");
481     mg->SetTitle(";Numero de nodos; #varepsilon_{#it{cluster}}");
482     TGraph *g2= new TGraph();   g2->SetMarkerStyle(21); g2->SetMarkerColor(kRed);
            g2->SetLineColor(kRed);
483     TGraph *g3= new TGraph();   g3->SetMarkerStyle(22); g3->SetMarkerColor(kBlue);
            g3->SetLineColor(kBlue);
484     TGraph *g4= new TGraph();   g4->SetMarkerStyle(23); g4->SetMarkerColor(kGreen);
             g4->SetLineColor(kGreen);
485 
486     FILE * values;
487     values= fopen("values.md","w");
488     fprintf( values,"Tracks SizeR  SizeT  SizeP \t Efi5 \t Efi10 \t Efi15 \t Efi20
            \t Efi25 \n");
489     Int_t Exist;
490     sizeR=450;
491     sizeT=150;
492     sizeP=100;
493     //for ( sizeP=50;  sizeP<=1500 ; sizeP+=50){
494       //for ( sizeT=50;  sizeT<=1500 ; sizeT+=50){
495         for ( sizeR=10; sizeR<=2000 ; sizeR+=10){
496           TH1* h1 = new TH1D("h1",";#it{E} [keV]; Eventos",BinsTh1*LimTh1,0,LimTh1
                );
```

```
497             Single(FileRoot,10000,h1,sizeR,sizeT,sizeP,Td,Rd,Tmax,LimTh1,BinsTh1,&
                    Exist,h1r2,h1reduce);
498             //h1->Draw("HIST BAR");
499             aux=0;          //numero de tracks conseguidos
500             Int_t Bin5 = h1->Fill(5);
501             Int_t Bin10 = h1->Fill(10);
502             Int_t Bin15 = h1->Fill(15);
503             Int_t Bin20 = h1->Fill(20);
504             Int_t Bin25 = h1->Fill(25);
505
506
507             aux2=(h1->GetBinContent(Bin5))-1;
508             aux3=(h1->GetBinContent(Bin10))-1;
509             aux4=(h1->GetBinContent(Bin15))-1;
510             aux5=(h1->GetBinContent(Bin20))-1;
511             aux6=(h1->GetBinContent(Bin25))-1;
512             aux=0;
513
514             for (int i=0; i<=LimTh1;i++){
515               Int_t Bin = h1->Fill(i);
516               aux=aux+(h1->GetBinContent(Bin))-1;
517             }
518
519             Efi5=aux2/10000;
520             Efi10=aux3/10000;
521             Efi15=aux4/10000;
522             Efi20=aux5/10000;
523             Efi25=aux6/10000;
524
525
526             if (Efi5>=0.9 && NTracks>=8000)
527             fprintf( values,"%d \t %d \t %d \t %f \t %f \t %f \t %f \t %f \t %f \n",
                    sizeR,sizeT,sizeP,EfiReal,Efi5,Efi10,Efi15,Efi20,Efi25);
528
529             g2->AddPoint(sizeR,Efi5);
530             g3->AddPoint(sizeR,Efi10);
531             g4->AddPoint(sizeR,Efi15);
532
533             h1->Reset();
534           }
535           //}}
536       fclose (values);
537
538       TCanvas *gcanvas= new TCanvas("gcanvas","gcanvas",700,500);
539       mg->Add(g2,"AC");  mg->Add(g3,"AC");  mg->Add(g4,"AC");
540       mg->Draw("AC");
541       TLegend *leg= new TLegend(0.7,0.7,0.9,0.83);
542       leg->AddEntry(g2," 5 keV","l");
543       leg->AddEntry(g3," 10 keV","l");
544       leg->AddEntry(g4," 15 keV","l");
545       leg->Draw();
546
547       char NameFile[100];
548       strcpy(NameFile,FilePdf.c_str());
549       gcanvas->SaveAs(NameFile);
```

```
550     gcanvas->Close();
551
552
553     return 0;
554   }
```

Bibliografía

[1] Adrian P. Mouritz. 9 - titanium alloys for aerospace structures and engines. In *Introduction to Aerospace Materials*, pages 202–223. Woodhead Publishing, 2012.

[2] Luvata. Oxygen-free electronic grade copper cu-ofe-luvata alloy ofe-ok, 2018.

[3] UNE Normalizacion Española. Une-en 13445-3:2014. unfired pressure vessels - part 3:design, 2014.