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
<?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">
```

```
      <materialref ref="Copper"/>  <!-- Tambien existen StainlessSteel y TitanPure -->
      <solidref ref="vesselSolid"/>
      </volume>

      <volume name="waterVolume">
      <materialref ref="Water"/>
      <solidref ref="waterSolid"/>
      </volume>

      <!-- }}} -->

<!-- Volumenes fisicos -->
<volume name="World">
<materialref ref="Vacuum" />
<solidref ref="worldSolid" />

<physvol name="gas">
<volumeref ref="gasVolume" />
<position name="gasPosition" unit="mm" x="0" y="0" z="0" />
</physvol>

<physvol name="vessel">
<volumeref ref="vesselVolume" />
<position name="vesselPosition" unit="mm" x="0" y="0" z="0" />
</physvol>

<physvol name="water">
<volumeref ref="waterVolume" />
<position name="waterPosition" unit="mm" x="0" y="0" z="0" />
</physvol>
</volume>

</structure>
```

4. Materiales y elementos

```
<!--  ##VERSION REST materials 1.4##  -->
<!--  densities in mg/cm3 @ 293.15K  -->
<materials>
<!--  {{{ Elements definition  -->
      <element name="Hydrogen" formula="H" Z="1">
      <atom value="1.00794"/>
      </element>
      <element name="Helium" formula="N" Z="2">
      <atom value="4.0026"/>
      </element>
      <element name="Carbon" formula="C" Z="6">
      <atom value="12.0107"/>
      </element>
      <element name="Nitrogen" formula="N" Z="7">
      <atom value="14.0067"/>
      </element>
```

```
        <element name="Fluor" formula="F" Z="9">
        <atom value="18.9984"/>
        </element>
        <element name="Silicon" formula="Si" Z="14" state="solid">
        <atom value="28.0855"/>
        </element>
        <element name="Chromium" formula="Cr" Z="24" state="solid">
        <atom value="51.996"/>
        </element>
        <element name="Manganese" formula="Mn" Z="25">
        <atom value="54.938045"/>
        </element>
        <element name="Iron" formula="Fe" Z="26" state="solid">
        <atom value="55.845"/>
        </element>
        <element name="Nickel" formula="Ni" Z="28" state="solid">
        <atom value="58.6934"/>
        </element>
        <element name="Aluminium" formula="Al" Z="13" state="solid">
        <atom value="26.982"/>
        </element>
        <element name="Xenon" formula="Xe" Z="54" state="gas">
        <atom value="131.293"/>
        </element>
        <element name="Argon" formula="Ar" Z="18" state="gas">
        <atom value="39.948"/>
        </element>
        <element name="Neon" formula="Ne" Z="10" state="gas">
        <atom value="20.1797"/>
        </element>
        <element name="Zinc" formula="Zn" Z="30">
        <atom value="65.4094"/>
        </element>
        <element name="Magnesium" formula="Mg" Z="12">
        <atom value="24.3050"/>
        </element>
        <element name="Sulfur" formula="S" Z="16">
        <atom value="32.0655"/>
        </element>
        <element name="Chlorine" formula="Cl" Z="17">
        <atom value="35.4532"/>
        </element>
        <element name="Titanium" formula="Ti" Z="22">
        <atom value="47.867"/>
        </element>
        <!-- }}} -->
  <!-- {{{ Quenchers -->
        <material name="TMA" state="gas">
        <D unit="mg/cm3" value="quencherDensity"/>
        <T unit="K" value="gasTemperature"/>
        <P unit="bar" value="gasPressure"/>
        <composite n="3" ref="Carbon"/>
        <composite n="9" ref="Hydrogen"/>
        <composite n="1" ref="Nitrogen"/>
        </material>
        <material name="isobutane" state="gas">
        <D unit="mg/cm3" value="quencherDensity"/>
        <T unit="K" value="gasTemperature"/>
        <P unit="bar" value="gasPressure"/>
        <composite n="4" ref="Carbon"/>
        <composite n="10" ref="Hydrogen"/>
        </material>
        <material name="methane" state="gas">
        <D unit="mg/cm3" value="quencherDensity"/>
        <T unit="K" value="gasTemperature"/>
        <P unit="bar" value="gasPressure"/>
        <composite n="1" ref="Carbon"/>
        <composite n="4" ref="Hydrogen"/>
        </material>
        <!-- }}} -->
  <!-- {{{ Gas mixtures -->
        <material name="PureHelium" state="gas">
        <T unit="K" value="gasTemperature"/>
        <P unit="bar" value="gasPressure"/>
        <MEE unit="eV" value="188"/>
        <D unit="mg/cm3" value="targetGasDensity"/>
        <fraction n="1" ref="Helium"/>
        </material>
        <material name="PureXenon" state="gas">
        <T unit="K" value="gasTemperature"/>
        <P unit="bar" value="gasPressure"/>
        <MEE unit="eV" value="482"/>
        <D unit="mg/cm3" value="targetGasDensity"/>
        <fraction n="1" ref="Xenon"/>
        </material>
        <material name="Xenon_ISO" state="gas">
        <D unit="mg/cm3" value="targetGasDensity+quencherDensity"/>
        <P unit="bar" value="gasPressure"/>
        <T unit="K" value="gasTemperature"/>
        <fraction n="quencherFraction" ref="isobutane"/>
        <fraction n="1-quencherFraction" ref="Xenon"/>
        </material>
        <material name="Xenon_MTH" state="gas">
        <D unit="mg/cm3" value="targetGasDensity+quencherDensity"/>
        <P unit="bar" value="gasPressure"/>
        <T unit="K" value="gasTemperature"/>
        <fraction n="quencherFraction" ref="methane"/>
        <fraction n="1-quencherFraction" ref="Xenon"/>
        </material>
        <material name="PureArgon" state="gas">
        <T unit="K" value="gasTemperature"/>
        <P unit="bar" value="gasPressure"/>
        <MEE unit="eV" value="188"/>
        <D unit="mg/cm3" value="targetGasDensity"/>
        <fraction n="1" ref="Argon"/>
        </material>
        <material name="Ar_ISO" state="gas">
        <D unit="mg/cm3" value="targetGasDensity+quencherDensity"/>
        <P unit="bar" value="gasPressure"/>
        <T unit="K" value="gasTemperature"/>
        <fraction n="quencherFraction" ref="isobutane"/>
        <fraction n="1-quencherFraction" ref="Argon"/>
        </material>
        <material name="Neon_ISO" state="gas">
        <D unit="mg/cm3" value="targetGasDensity+quencherDensity"/>
        <P unit="bar" value="gasPressure"/>
        <T unit="K" value="gasTemperature"/>
        <fraction n="quencherFraction" ref="isobutane"/>
        <fraction n="1-quencherFraction" ref="Neon"/>
        </material>
        <material name="Helium_ISO" state="gas">
        <D unit="mg/cm3" value="targetGasDensity+quencherDensity"/>
        <P unit="bar" value="gasPressure"/>
        <T unit="K" value="gasTemperature"/>
        <fraction n="quencherFraction" ref="isobutane"/>
        <fraction n="1-quencherFraction" ref="Helium"/>
        </material>
        <!--  }}}  -->
  <!--  {{{ Copper  -->
        <isotope N="63" Z="29" name="Cu63">
        <atom unit="g/mole" value="62.9296"/>
        </isotope>
        <isotope N="65" Z="29" name="Cu65">
        <atom unit="g/mole" value="64.9278"/>
        </isotope>
        <element name="Cu">
        <fraction n="0.6917" ref="Cu63"/>
        <fraction n="0.3083" ref="Cu65"/>
        </element>
        <material name="Copper" state="solid">
        <MEE unit="eV" value="322"/>
        <D unit="g/cm3" value="8.96"/>
        <fraction n="1" ref="Cu"/>
        </material>
        <!--  }}}  -->
  <!--  {{{ Stainlesssteel  -->
        <material name="Stainlesssteel" state="solid">
        <MEE unit="eV" value="280.927567486627"/>
        <D unit="g/cm3" value="8.02"/>
        <fraction n="0.02" ref="Manganese"/>
        <fraction n="0.01" ref="Silicon"/>
        <fraction n="0.19" ref="Chromium"/>
        <fraction n="0.1" ref="Nickel"/>
        <fraction n="0.68" ref="Iron"/>
        </material>
        <!--  }}}  -->
  <material name="TitanPure" state="solid">
  <D unit="g/cm3" value="4.6"/>
  <fraction n="0.99475" ref="Titanium"/>
  <fraction n="0.0003" ref="Nitrogen"/>
  <fraction n="0.0010" ref="Carbon"/>
  <fraction n="0.00015" ref="Hydrogen"/>
  <fraction n="0.0020" ref="Iron"/>
  <fraction n="0.0018" ref="Oxygen"/>
  </material>
```

```xml

    <!-- {{{ Air -->
          <isotope N="14" Z="7" name="N14">
          <atom unit="g/mole" value="14.0031"/>
          </isotope>
          <isotope N="15" Z="7" name="N15">
          <atom unit="g/mole" value="15.0001"/>
          </isotope>
          <element name="Nitrogen">
          <fraction n="0.99632" ref="N14"/>
          <fraction n="0.00368" ref="N15"/>
          </element>
          <isotope N="16" Z="8" name="O16">
          <atom unit="g/mole" value="15.9949"/>
          </isotope>
          <isotope N="17" Z="8" name="O17">
          <atom unit="g/mole" value="16.9991"/>
          </isotope>
          <isotope N="18" Z="8" name="O18">
          <atom unit="g/mole" value="17.9992"/>
          </isotope>
          <element name="Oxygen">
          <fraction n="0.99757" ref="O16"/>
          <fraction n="0.00038" ref="O17"/>
          <fraction n="0.00205" ref="O18"/>
          </element>
          <material name="Air" state="gas">
          <T unit="K" value="293.13"/>
          <MEE unit="eV" value="85.7030667332999"/>
          <D unit="g/cm3" value="0.00129"/>
          <fraction n="0.7" ref="Nitrogen"/>
          <fraction n="0.3" ref="Oxygen"/>
          </material>
          <!-- }}} -->
    <!-- {{{ Vacuum -->
          <material name="Vacuum" state="gas">
          <T unit="K" value="293.13"/>
          <MEE unit="eV" value="85.7030667332999"/>
          <D unit="g/cm3" value="1.e-20"/>
          <fraction n="0.7" ref="Nitrogen"/>
          <fraction n="0.3" ref="Oxygen"/>
          </material>
          <!-- }}} -->
    <!-- {{{ Water -->
          <element Z="1" formula="H" name="Hydrogen">
          <atom value="2"/>
          </element>
          <element Z="8" formula="O" name="Oxygen">
          <atom value="16"/>
          </element>
          <material name="Water" formula="H2O">
          <D value="1.0"/>
          <composite n="2" ref="Hydrogen"/>
          <composite n="1" ref="Oxygen"/>
          </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]");


```

```
    Double_t Cdaux,Vdaux,Max,Min;
    Double_t rmin=500;                       //en mm, el del detector
    Double_t rmax=1500;              //en mm, el de la esfera , considerando el grosor despreciable
    Double_t voltage=750;                   //en V
    Double_t pressure=10.;                   //bar
    int p=rmax-rmin+1;                      //+1 para almacenar rmax
    Double_t field;
    Double_t Cd[p],Vd[p],E[p],Td[p],R[p];

    // Calculamos las carateristicas para cada gas

    TRestDetectorGas *gas1=new TRestDetectorGas ("server","Xenon-TMA 0.5Pct 0.1-10E3Vcm");
    gas1->SetPressure(pressure);
    for (int i=0;i<=rmax-rmin;i++)                              //resolucion cada mm (delta_r=1)
    {
      R[i]=rmin+i;                                 //posicion en mm
       field=(voltage/(R[i]*R[i]))*rmin*rmax*(1/(rmax-rmin));      //en V/mm
       GasProperties(field ,Cdaux,Vdaux,gas1);                      //en V/mm
      Vd[i]=Vdaux;                                              //en mm/us
      Cd[i]=Cdaux;                                          //en cm^(1/2)
      Td[0]=1/(Vd[0]*1000); if (i>0) {Td[i]=Td[i-1]+(1/(Vd[i]*1000));}    //0,1 y 1000 para ms
    }
    cout << "Cd promedio:"<<Promedio(Cd,p)<<"          Tmax:"<<Td[p-1]<<endl;


    auto Vdgr1 = new TGraph(p,R,Vd);
    Vdgr1->SetTitle("Xenon-TMA 0.5Pct");
    Vdgr1->SetMarkerStyle(kFullDotSmall);
    Vdgr1->SetMarkerColor(1);
    Vdgr1->SetLineColor(1);
    Vdgr1->SetLineWidth(2);
    auto Cdgr1 = new TGraph(p,R,Cd);
    Cdgr1->SetTitle("Xenon-TMA 0.5Pct");
    Cdgr1->SetMarkerStyle(kFullDotSmall);
    Cdgr1->SetMarkerColor(1);
    Cdgr1->SetLineColor(1);
    Cdgr1->SetLineWidth(2);
    auto Tdgr1 = new TGraph(p,R,Td);
    Tdgr1->SetTitle("Xenon-TMA 0.5Pct");
    Tdgr1->SetMarkerStyle(kFullDotSmall);
    Tdgr1->SetMarkerColor(1);
    Tdgr1->SetLineColor(1);
    Tdgr1->SetLineWidth(2);


    TRestDetectorGas *gas2=new TRestDetectorGas ("server","Xenon-TMA 2Pct 0.1-10E3Vcm");
    gas2->SetPressure(pressure);
    for (int i=0;i<=rmax-rmin;i++){
      R[i]=rmin+i;
      field=(voltage/(R[i]*R[i]))*rmin*rmax*(1/(rmax-rmin));  GasProperties(field, Cdaux,Vdaux,gas2);
      Vd[i]=Vdaux;
      Cd[i]=Cdaux;
      Td[0]=1/(Vd[0]*1000);
      if (i>0)
        {Td[i]=Td[i-1]+(1/(Vd[i]*1000));}
      }
    cout << "Cd promedio:"<<Promedio(Cd,p)<<"          Tmax:"<<Td[p-1]<<endl;

    auto Vdgr2 = new TGraph(p,R,Vd);
    Vdgr2->SetTitle("Xenon-TMA 2Pct");
    Vdgr2->SetMarkerStyle(kFullDotSmall);
    Vdgr2->SetMarkerColor(2);
    Vdgr2->SetLineColor(2);
    Vdgr2->SetLineWidth(2);
    auto Cdgr2 = new TGraph(p,R,Cd);
    Cdgr2->SetTitle("Xenon-TMA 2Pct");
    Cdgr2->SetMarkerStyle(kFullDotSmall);
    Cdgr2->SetMarkerColor(2);
    Cdgr2->SetLineColor(2);
    Cdgr2->SetLineWidth(2);
    auto Tdgr2 = new TGraph(p,R,Td);
    Tdgr2->SetTitle("Xenon-TMA 2Pct");
    Tdgr2->SetMarkerStyle(kFullDotSmall);
    Tdgr2->SetMarkerColor(2);
    Tdgr2->SetLineColor(2);
    Tdgr2->SetLineWidth(2);

    //#####################################

    TRestDetectorGas *gas3=new TRestDetectorGas ("server","Xenon-Methane 0.5Pct 0.1-10E3Vcm");
    gas3->SetPressure(pressure);
    for (int i=0;i<=rmax-rmin;i++){
      R[i]=rmin+i;
      field=(voltage/(R[i]*R[i]))*rmin*rmax*(1/(rmax-rmin));  GasProperties(field, Cdaux,Vdaux,gas2);
      Vd[i]=Vdaux;
      Cd[i]=Cdaux;
      Td[0]=1/(Vd[0]*1000);
      if (i>0)
      {Td[i]=Td[i-1]+(1/(Vd[i]*1000));}
    }
    cout << "Cd promedio:"<<Promedio(Cd,p)<<"          Tmax:"<<Td[p-1]<<endl;

    auto Vdgr3 = new TGraph(p,R,Vd);
    Vdgr3->SetTitle("Xenon-Methane 0.5Pct");
    Vdgr3->SetMarkerStyle(kFullDotSmall);
    Vdgr3->SetMarkerColor(3);
    Vdgr3->SetLineColor(3);
    Vdgr3->SetLineWidth(2);
    auto Cdgr3 = new TGraph(p,R,Cd);
    Cdgr3->SetTitle("Xenon-Methane 0.5Pct");
    Cdgr3->SetMarkerStyle(kFullDotSmall);
    Cdgr3->SetMarkerColor(3);
    Cdgr3->SetLineColor(3);
    Cdgr3->SetLineWidth(2);
    auto Tdgr3 = new TGraph(p,R,Td);
    Tdgr3->SetTitle("Xenon-Methane 0.5Pct");
    Tdgr3->SetMarkerStyle(kFullDotSmall);
    Tdgr3->SetMarkerColor(3);
    Tdgr3->SetLineColor(3);
    Tdgr3->SetLineWidth(2);

    TRestDetectorGas *gas4=new TRestDetectorGas("server","Xenon-Methane 2Pct
        0.1-10E3Vcm");
    gas4->SetPressure(pressure);
    for (int i=0;i<=rmax-rmin;i++){
      R[i]=rmin+i;
      field=(voltage/(R[i]*R[i]))*rmin*rmax*(1/(rmax-rmin));   GasProperties(field,
          Cdaux,Vdaux,gas2);
      Vd[i]=Vdaux;
      Cd[i]=Cdaux;
      Td[0]=1/(Vd[0]*1000);
      if (i>0)
      {Td[i]=Td[i-1]+(1/(Vd[i]*1000));}
    }
    cout << "Cd promedio:"<<Promedio(Cd,p)<<"            Tmax:"<<Td[p-1]<<endl;


    auto Vdgr4 = new TGraph(p,R,Vd);
    Vdgr4->SetTitle("Xenon-Methane 2Pct");
    Vdgr4->SetMarkerStyle(kFullDotSmall);
    Vdgr4->SetMarkerColor(4);
    Vdgr4->SetLineColor(4);
    Vdgr4->SetLineWidth(2);
    auto Cdgr4 = new TGraph(p,R,Cd);
    Cdgr4->SetTitle("Xenon-Methane 2Pct");
    Cdgr4->SetMarkerStyle(kFullDotSmall);
    Cdgr4->SetMarkerColor(4);
    Cdgr4->SetLineColor(4);
    Cdgr4->SetLineWidth(2);
    auto Tdgr4 = new TGraph(p,R,Td);
    Tdgr4->SetTitle("Xenon-Methane 2Pct");
    Tdgr4->SetMarkerStyle(kFullDotSmall);
    Tdgr4->SetMarkerColor(4);
    Cdgr4->SetLineColor(4);
    Cdgr4->SetLineWidth(2);

    //#####################################

    TRestDetectorGas *gas5=new TRestDetectorGas ("server","Xenon-Isobutane 0.5Pct
        0.1-10E3Vcm");
    gas5->SetPressure(pressure);
    for (int i=0;i<=rmax-rmin;i++){
      R[i]=rmin+i;
      field=(voltage/(R[i]*R[i]))*rmin*rmax*(1/(rmax-rmin));   GasProperties(field,
          Cdaux,Vdaux,gas2);
      Vd[i]=Vdaux;
      Cd[i]=Cdaux;
      Td[0]=1/(Vd[0]*1000);
      if (i>0)
      {Td[i]=Td[i-1]+(1/(Vd[i]*1000));}
    }
    cout << "Cd promedio:"<<Promedio(Cd,p)<<"          Tmax:"<<Td[p-1]<<endl;

    auto Vdgr5 = new TGraph(p,R,Vd);
    Vdgr5->SetTitle("Xenon-Isobutane 0.5Pct");
    Vdgr5->SetMarkerStyle(kFullDotSmall);
    Vdgr5->SetMarkerColor(5);
    Vdgr5->SetLineColor(5);
    Vdgr5->SetLineWidth(2);
    auto Cdgr5 = new TGraph(p,R,Cd);
    Cdgr5->SetTitle("Xenon-Isobutane 0.5Pct");
    Cdgr5->SetMarkerStyle(kFullDotSmall);
    Cdgr5->SetMarkerColor(5);
    Cdgr5->SetLineColor(5);
    Cdgr5->SetLineWidth(2);
    auto Tdgr5 = new TGraph(p,R,Td);
    Tdgr5->SetTitle("Xenon-Isobutane 0.5Pct");
    Tdgr5->SetMarkerStyle(kFullDotSmall);
    Tdgr5->SetMarkerColor(5);
    Tdgr5->SetLineColor(5);
    Tdgr5->SetLineWidth(2);

    TRestDetectorGas *gas6=new TRestDetectorGas ("server","Xenon-Isobutane 2Pct 0.1-10E6Vcm");
    gas6->SetPressure(pressure);
    for (int i=0;i<=rmax-rmin;i++){
      R[i]=rmin+i;
      field=(voltage/(R[i]*R[i]))*rmin*rmax*(1/(rmax-rmin));   GasProperties(field, Cdaux,Vdaux,gas2);
      Vd[i]=Vdaux;
      Cd[i]=Cdaux;
      Td[0]=1/(Vd[0]*1000);
      if (i>0)
      {Td[i]=Td[i-1]+(1/(Vd[i]*1000));}
    }
    cout << "Cd promedio:"<<Promedio(Cd,p)<<"          Tmax:"<<Td[p-1]<<endl;

    auto Vdgr6 = new TGraph(p,R,Vd);
    Vdgr6->SetTitle("Xenon-Isobutane 2Pct");
    Vdgr6->SetMarkerStyle(kFullDotSmall);
    Vdgr6->SetMarkerColor(6); Vdgr6->SetLineColor(6);
    Vdgr6->SetLineWidth(2);
    auto Cdgr6 = new TGraph(p,R,Cd);
    Cdgr6->SetTitle("Xenon-Isobutane 2Pct");
    Cdgr6->SetMarkerStyle(kFullDotSmall);
    Cdgr6->SetMarkerColor(6);
    Cdgr6->SetLineColor(6);
    Cdgr6->SetLineWidth(2);
    auto Tdgr6 = new TGraph(p,R,Td);
    Tdgr6->SetTitle("Xenon-Isobutane 2Pct");
    Tdgr6->SetMarkerStyle(kFullDotSmall);
    Tdgr6->SetMarkerColor(6);
    Tdgr6->SetLineColor(6);
    Tdgr6->SetLineWidth(2);

    //######################################

    TRestDetectorGas *gas7=new TRestDetectorGas ("server","Helium-Isobutane 0.5Pct 0.1-10E3Vcm");
    gas7->SetPressure(pressure);
    for (int i=0;i<=rmax-rmin;i++){
      R[i]=rmin+i;
       field=(voltage/(R[i]*R[i]))*rmin*rmax*(1/(rmax-rmin));  GasProperties(field,Cdaux,Vdaux,gas2);
      Vd[i]=Vdaux;
      Cd[i]=Cdaux;
      Td[0]=1/(Vd[0]*1000);
       if (i>0)
       {Td[i]=Td[i-1]+(1/(Vd[i]*1000));}
    }
    cout << "Cd promedio:"<<Promedio(Cd,p)<<"            Tmax:"<<Td[p-1]<<endl;

    auto Vdgr7 = new TGraph(p,R,Vd);
    Vdgr7->SetTitle("Helium-Isobutane 0.5Pct");
    Vdgr7->SetMarkerStyle(kFullDotSmall);
    Vdgr7->SetMarkerColor(7);
    Vdgr7->SetLineColor(7);
    Vdgr7->SetLineWidth(2);
    auto Cdgr7 = new TGraph(p,R,Cd);
    Cdgr7->SetTitle("Helium-Isobutane 0.5Pct");
    Cdgr7->SetMarkerStyle(kFullDotSmall);
    Cdgr7->SetMarkerColor(7);
    Cdgr7->SetLineColor(7);
    Cdgr7->SetLineWidth(2);
    auto Tdgr7 = new TGraph(p,R,Td);
    Tdgr7->SetTitle("Helium-Isobutane 0.5Pct");
    Tdgr7->SetMarkerStyle(kFullDotSmall);
    Tdgr7->SetMarkerColor(7);
    Tdgr7->SetLineColor(7);
    Tdgr7->SetLineWidth(2);

    TRestDetectorGas *gas8=new TRestDetectorGas ("server","Helium-Isobutane 3Pct 0.1-10E3Vcm");
    gas8->SetPressure(pressure);
    for (int i=0;i<=rmax-rmin;i++){
      R[i]=rmin+i;
       field=(voltage/(R[i]*R[i]))*rmin*rmax*(1/(rmax-rmin));  GasProperties(field,Cdaux,Vdaux,gas2);
      Vd[i]=Vdaux;
      Cd[i]=Cdaux;
      Td[0]=1/(Vd[0]*1000);
       if (i>0)
       {Td[i]=Td[i-1]+(1/(Vd[i]*1000));}
    }
    cout << "Cd promedio:"<<Promedio(Cd,p)<<"            Tmax:"<<Td[p-1]<<endl;

    auto Vdgr8 = new TGraph(p,R,Vd);
    Vdgr8->SetTitle("Helium-Isobutane 3Pct");
    Vdgr8->SetMarkerStyle(kFullDotSmall);
    Vdgr8->SetMarkerColor(8);
    Vdgr8->SetLineColor(8);
     Vdgr8->SetLineWidth(2);
    auto Cdgr8 = new TGraph(p,R,Cd);
    Cdgr8->SetTitle("Helium-Isobutane 3Pct");
    Cdgr8->SetMarkerStyle(kFullDotSmall);
    Cdgr8->SetMarkerColor(8);
    Cdgr8->SetLineColor(8);
    Cdgr8->SetLineWidth(2);
    auto Tdgr8 = new TGraph(p,R,Td);
    Tdgr8->SetTitle("Helium-Isobutane 3Pct");
    Tdgr8->SetMarkerStyle(kFullDotSmall);
    Tdgr8->SetMarkerColor(8);
    Tdgr8->SetLineColor(8);
    Tdgr8->SetLineWidth(2);

    //######################################

    TRestDetectorGas *gas9=new TRestDetectorGas ("server","Neon-Isobutane 0.5Pct 0.1-10E3Vcm");
    gas9->SetPressure(pressure);
    for (int i=0;i<=rmax-rmin;i++){
      R[i]=rmin+i;
       field=(voltage/(R[i]*R[i]))*rmin*rmax*(1/(rmax-rmin));  GasProperties(field,Cdaux,Vdaux,gas2);
      Vd[i]=Vdaux;
      Cd[i]=Cdaux;
      Td[0]=1/(Vd[0]*1000);
       if (i>0)
      {Td[i]=Td[i-1]+(1/(Vd[i]*1000));}
    }
    cout << "Cd promedio:"<<Promedio(Cd,p)<<"            Tmax:"<<Td[p-1]<<endl;

    auto Vdgr9 = new TGraph(p,R,Vd);
    Vdgr9->SetTitle("Neon-Isobutane 0.5Pct");
    Vdgr9->SetMarkerStyle(kFullDotSmall);
    Vdgr9->SetMarkerColor(1);
    Vdgr9->SetLineColor(1);
     Vdgr9->SetLineWidth(2);
    auto Cdgr9 = new TGraph(p,R,Cd);
    Cdgr9->SetTitle("Neon-Isobutane 0.5Pct");
    Cdgr9->SetMarkerStyle(kFullDotSmall);
    Cdgr9->SetMarkerColor(1);
    Cdgr9->SetLineColor(1);
    Cdgr9->SetLineWidth(2);
    auto Tdgr9 = new TGraph(p,R,Td);
    Tdgr9->SetTitle("Neon-Isobutane 0.5Pct");
    Tdgr9->SetMarkerStyle(kFullDotSmall);
    Tdgr9->SetMarkerColor(1);
    Tdgr9->SetLineColor(1);
    Tdgr9->SetLineWidth(2);

```

```
  TRestDetectorGas *gas10=new TRestDetectorGas ("server","Neon-Isobutane 3Pct 0.1-10E3Vcm");
  gas10->SetPressure(pressure);
  for (int i=0;i<=rmax-rmin;i++){
    R[i]=rmin+i;
     field=(voltage/(R[i]*R[i]))*rmin*rmax*(1/(rmax-rmin));  GasProperties(field,Cdaux,Vdaux,gas2);
    Vd[i]=Vdaux;
    Cd[i]=Cdaux;
    Td[0]=1/(Vd[0]*1000);
     if (i>0)
     {Td[i]=Td[i-1]+(1/(Vd[i]*1000));}
  }
  cout << "Cd promedio:"<<Promedio(Cd,p)<<"          Tmax:"<<Td[p-1]<<endl;

  auto Vdgr10 = new TGraph(p,R,Vd);
  Vdgr10->SetTitle("Neon-Isobutane 3Pct");
  Vdgr10->SetMarkerStyle(kFullDotSmall);
  Vdgr10->SetMarkerColor(2);
  Vdgr10->SetLineColor(2);
  Vdgr10->SetLineWidth(2);
  auto Cdgr10 = new TGraph(p,R,Cd);
  Cdgr10->SetTitle("Neon-Isobutane 3Pct");
  Cdgr10->SetMarkerStyle(kFullDotSmall);
  Cdgr10->SetMarkerColor(2);
  Cdgr10->SetLineColor(2);
  Cdgr10->SetLineWidth(2);
  auto Tdgr10 = new TGraph(p,R,Td);
  Tdgr10->SetTitle("Neon-Isobutane 3Pct");
  Tdgr10->SetMarkerStyle(kFullDotSmall);
  Tdgr10->SetMarkerColor(2);
  Tdgr10->SetLineColor(2);
  Tdgr10->SetLineWidth(2);

  //###################################

  TRestDetectorGas *gas11=new TRestDetectorGas ("server","Argon-Isobutane 0.5Pct 0.1-10E3Vcm");
  gas11->SetPressure(pressure);
  for (int i=0;i<=rmax-rmin;i++){
    R[i]=rmin+i;
     field=(voltage/(R[i]*R[i]))*rmin*rmax*(1/(rmax-rmin));  GasProperties(field,Cdaux,Vdaux,gas2);
    Vd[i]=Vdaux;
    Cd[i]=Cdaux;
    Td[0]=1/(Vd[0]*1000);
     if (i>0)
     {Td[i]=Td[i-1]+(1/(Vd[i]*1000));}
  }
  cout << "Cd promedio:"<<Promedio(Cd,p)<<"          Tmax:"<<Td[p-1]<<endl;

  auto Vdgr11 = new TGraph(p,R,Vd);
  Vdgr11->SetTitle("Argon-Isobutane 0.5Pct");
  Vdgr11->SetMarkerStyle(kFullDotSmall);
  Vdgr11->SetMarkerColor(11);
  Vdgr11->SetLineColor(11);
  Vdgr11->SetLineWidth(2);
  auto Cdgr11 = new TGraph(p,R,Cd);
  Cdgr11->SetTitle("Argon-Isobutane 0.5Pct");
  Cdgr11->SetMarkerStyle(kFullDotSmall);
  Cdgr11->SetMarkerColor(11);
  Cdgr11->SetLineColor(11);
  Cdgr11->SetLineWidth(2);
  auto Tdgr11 = new TGraph(p,R,Td);
  Tdgr11->SetTitle("Argon-Isobutane 0.5Pct");
  Tdgr11->SetMarkerStyle(kFullDotSmall);
  Tdgr11->SetMarkerColor(11);
  Tdgr11->SetLineColor(11);
  Tdgr11->SetLineWidth(2);

  TRestDetectorGas *gas12=new TRestDetectorGas ("server","Argon-Isobutane 3Pct 0.1-10E3Vcm");
  gas12->SetPressure(pressure);
  for (int i=0;i<=rmax-rmin;i++){
   R[i]=rmin+i;
   field=(voltage/(R[i]*R[i]))*rmin*rmax*(1/(rmax-rmin));  GasProperties(field, Cdaux,Vdaux,gas2);
   Vd[i]=Vdaux;
   Cd[i]=Cdaux;
   Td[0]=1/(Vd[0]*1000);
   if (i>0)
   {Td[i]=Td[i-1]+(1/(Vd[i]*1000));}
  }
  cout << "Cd promedio:"<<Promedio(Cd,p)<<"            Tmax:"<<Td[p-1]<<endl;

  auto Vdgr12 = new TGraph(p,R,Vd);
  Vdgr12->SetTitle("Argon-Isobutane 3Pct");
  Vdgr12->SetMarkerStyle(kFullDotSmall);
  Vdgr12->SetMarkerColor(12);
  Vdgr12->SetLineColor(12);
  Vdgr12->SetLineWidth(2);
  auto Cdgr12 = new TGraph(p,R,Cd);
  Cdgr12->SetTitle("Argon-Isobutane 3Pct");
  Cdgr12->SetMarkerStyle(kFullDotSmall);
  Cdgr12->SetMarkerColor(12);
  Cdgr12->SetLineColor(12);
  Cdgr12->SetLineWidth(2);
  auto Tdgr12 = new TGraph(p,R,Td);
  Tdgr12->SetTitle("Argon-Isobutane 3Pct");
  Tdgr12->SetMarkerStyle(kFullDotSmall);
  Tdgr12->SetMarkerColor(12);
  Tdgr12->SetLineColor(12);
  Tdgr12->SetLineWidth(2);

  // Mostramos la velocidad de deriva, coeficiente de difusion o tiempo de deriva

  Vdmg->Add(Vdgr3); Vdmg->Add(Vdgr4);
  Vdmg->Add(Vdgr7); Vdmg->Add(Vdgr8);
  Vdmg->Add(Vdgr9);  Vdmg->Add(Vdgr10);
```

```
    Vdmg->Add(Vdgr11); Vdmg->Add(Vdgr12);  Vdmg->Draw("ACP");

    Cdmg->Add(Cdgr3); Cdmg->Add(Cdgr4);
    Cdmg->Add(Cdgr7); Cdmg->Add(Cdgr8);
    Cdmg->Add(Cdgr9);  Cdmg->Add(Cdgr10);
    Cdmg->Add(Cdgr11); Cdmg->Add(Cdgr12);  Cdmg->Draw("ACP");

    Tdmg->Add(Tdgr3); Tdmg->Add(Tdgr4);
    Tdmg->Add(Tdgr7); Tdmg->Add(Tdgr8);
    Tdmg->Add(Tdgr9);  Tdmg->Add(Tdgr10);
    Tdmg->Add(Tdgr11); Tdmg->Add(Tdgr12);  Tdmg->Draw("APE");

    auto legendCd = new TLegend(0.1,0.6,0.48,0.9);   //<-x1  ^| y1   x2->  (x1,y1,x2,y2)

    Cdgr3->SetMarkerStyle(kFullDotLarge);
    Cdgr4->SetMarkerStyle(kFullDotLarge);
    Cdgr7->SetMarkerStyle(kFullDotLarge);
    Cdgr8->SetMarkerStyle(kFullDotLarge);
    Cdgr9->SetMarkerStyle(kFullDotLarge);
    Cdgr10->SetMarkerStyle(kFullDotLarge);
    Cdgr11->SetMarkerStyle(kFullDotLarge);
    Cdgr12->SetMarkerStyle(kFullDotLarge);

    legendCd->AddEntry(Cdgr3,"","p");  legendCd->AddEntry(Cdgr4,"","p");
    legendCd->AddEntry(Cdgr7,"","p");  legendCd->AddEntry(Cdgr8,"","p");
    legendCd->AddEntry(Cdgr9,"","p");  legendCd->AddEntry(Cdgr10,"","p");
    legendCd->AddEntry(Cdgr11,"","p"); legendCd->AddEntry(Cdgr12,"","p");
    legendCd->Draw();

    return 0;
  }
```

7. Estudio de los eventos

```

#include "TRestGeant4Event.h"
#include "TRestGeant4Metadata.h"
#include "TRestTask.h"
#include <TH3D.h>
#include <TMath.h>
#include <chrono>
using namespace std;
using namespace std::chrono;

Double_t rmin=500, rmax=1500;
Double_t resolution=0.001;              //en mm

Double_t Search (Double_t target, Int_t n, const std::vector<Double_t> &A){
  Int_t L = 0;
  Int_t R = n-1;
  Int_t mid, ans= -1;

```

```
    while( L<=R ) {
      mid= L+((R-L)/2);
      if(A[mid] >= target){
        ans = mid;
        R=mid-1;}
      else L=mid+1;
    }
    return ans;
  }
  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){
    TString fName=File;
    cout<<"\n Archivo: "<<fName<<"\n"<<endl;
    TRestRun* run = new TRestRun();
    Double_t pi=TMath::Pi();
    Double_t rho=0,theta=0,phi=0;
    Double_t Xhit=0,Yhit=0,Zhit=0,Ehit=0;

    string fname = fName.Data();
    if (!TRestTools::fileExists(fname)) {
      cout << "WARNING. Input file does not exist" << endl;
    } else  run->OpenInputFile(fName);

    //run->PrintMetadata();

    ////////// Escogemos eventos aleatorios de la base de datos

    TRestGeant4Metadata *g4=(TRestGeant4Metadata*)run->GetMetadataClass("TRestGeant4Metadata");
    Double_t Generated=g4->GetNumberOfEvents();      //lanzados en simulacion
    Double_t Detected=run->GetEntries();             //los almacenados
    Double_t GeneratedReal=EventNumber;                     //los recibidos en nuestro tiempo deriva
    Double_t DetectedRealOriginal=GeneratedReal*Detected/Generated ;    //los que deberiamos haber detectado
    Double_t DetectedReal=0;

    TRandom3 *r3 = new TRandom3();  r3->SetSeed(0);
    DetectedReal=r3->Poisson(DetectedRealOriginal);

    cout << "\n######Eventos lanzados(sim): "<<Generated<<"    Eventos finales(sim): "<<Detected<<endl;
    cout << "\n######Eventos lanzados: "<<GeneratedReal<<"    Eventos finales: "<<DetectedRealOriginal<<"  Tras Poisson: "<<DetectedReal<<endl;

    TRandom* aleatorio= new TRandom();
    aleatorio->SetSeed(0);
    UInt_t var;
    Bool_t repe=false;
    std::vector<Int_t> Selected;

    if(DetectedReal <= Detected)
    {for (int v=0; v<DetectedReal;v++)
      //cout << v <<endl;
      {
        do
        {var=aleatorio->Integer(run->GetEntries());
          repe=false;
          for (int i=0;i<Selected.size();i++)
          {if (var == Selected[i])
            repe=true;
          }
        }
        while (repe != false);
        Selected.push_back(var);
      }  //cout<<"Eventos escogidos"<<endl;
    } else if (Detected > DetectedReal )
    {cout<<"No hay eventos suficientes en: "<<fName<<endl;

    } else   {cout<<"No hay eventos. Eventos:"<<run->GetEntries()<<" Archivo: "<<fName<<endl;}


    /////////////  Guardar y convertir hits a Td,theta,phi

    TRestHits* TallHits = new TRestHits();
    TRestHits* XallHits = new TRestHits();
    TRestGeant4Event* ev = new TRestGeant4Event();
    run->SetInputEvent(ev);

    Double_t aux=0.;
    Int_t index=0;

    for (int i = 0; i < Selected.size(); i++)
    {
      run->GetEntry(Selected[i]);
      //ev->PrintEvent();
      for (int n = 0; n < ev->GetNumberOfTracks(); n++)
      {
        TRestGeant4Hits* hits = ev->GetTrack(n)->GetHits();

        for (int m = 0; m < hits->GetNumberOfHits(); m++)
        {
          if((hits->GetEnergy(m))>0)
          {
            Xhit=hits->GetX(m);  Yhit=hits->GetY(m);  Zhit=hits->GetZ(m);  Ehit=hits->GetEnergy(m);
            TVector3 v(Xhit,Yhit,Zhit);
            rho=v.Mag();

            if (rho >= rmin && rho<=rmax)
            {
              theta=v.Theta(); phi=v.Phi();
              index=(rho-rmin)/resolution;
              aux=Td[index];
              TallHits->AddHit(aux,theta,phi,Ehit);  // asi es tiempo de deriva
            }
          }
        }
      }
    }
    cout << "Hits:"<<TallHits->GetNumberOfHits()<<endl;
    cout <<"HITS CARGADOS"<<endl;

    ///////////////////  Clusterizado

    TRestMesh *Tmesh = new TRestMesh(TVector3(2*Tmax,2*Tmax,2*Tmax), TVector3(0,0,0),sizeR,sizeT,sizeP);
    Tmesh->SetSpherical(true);
    Tmesh->SetNodesFromSphericalHits(TallHits);
    cout << "TMesh generada"<<endl;

    cout<<"Numero de nodos:"<<Tmesh->GetNumberOfNodes()<<endl;
    Int_t TnTracksFound = Tmesh->GetNumberOfGroups();  //numero de grupos
    cout <<"Tracks:"<<TnTracksFound<<endl;

    ////////// Calculating the energy of each track

    cout<<"Clusterizando energia"<<endl;
    Int_t i=0;
    Double_t RhoLimit=1300.;
    Double_t TdLimit=0.;  //Valor de rho que fiducializa y su equivalente en tiempo de deriva

    index=(RhoLimit-rmin)/resolution;
    TdLimit=Td[index];
    TVector3 v(0,0,0);

    fprintf(Data,"%d\n",TnTracksFound);
    for (int n = 0; n < TnTracksFound; n++){
      v=Tmesh->GetGroupPosition(n);        //posicion del track promedio
      Double_t td=v.Mag();                 //Td promedio del track
      index = Search(td,Td.size(),Td);     //indice del valor
      rho=Rd[index];                       //Rho promedio
      h1r2->Fill(rho/1000);
      *Existr2=1;
      Double_t Energy= Tmesh->GetGroupEnergy(n);
      h1->Fill(Energy);
      fprintf(Data,"%f \t %f \n",Energy,rho/1000);
      if (Energy <= LimTh1 ){
        i++;
        *Exist=1;    //para anadir de la leyenda
        //cout << "Track " << n << ", E: " << Energy <<" , Td: "<<td<< ", Rho: "<< rho <<", Rho^2: "<<pow((rho/1000),2)<< endl;
      }
      if( rho < RhoLimit && Energy <= LimTh1){  //FIDUCIALIZACION
        h1reduce->Fill(Energy);
        *Existred=1;
      }

    }
    //////////  Normalizacion
    h1->Scale(BinsTh1);
    h1reduce->Scale(BinsTh1);
    cout<<"Tracks con E<"<<LimTh1<<" : "<<i<<endl;
    cout<<"Energia clusterizada en TMesh"<<endl;


    delete ev;
    delete run;
    return 0;
}


Int_t SphereSegmentation(){

    const Int_t EventTypes=22;        //numero de tipos de particulas generadas

    //gamma, n mu, vessel, gas
    //Xe-0 to 10-5-Cu-1 (cambiar el nombre de la carpera)
    Double_t EventNumber[EventTypes]=
    //{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-4,1.84e-4,1.62e-3} ;
    {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} ;

    // Xe-1-5-Cu-10 y Xe-10-5-Cu-10
    //{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};
    //{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}          ;

    //Xe-1-10-Cu-100
    //{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}   ;
    //{326726,163363,40841,20475,40625,   4,14,21,  5,7,26,18,88,136,8,58,  37,37,60,25,22,189} ;

    //Xe-1-10-St-115
    //{4812,2406,601,302,598,         21,1,2,12,1,3,  6.25e-1,6.33e-1,1,4.19e-1,3.66e-1,3}          ;
    //{326726,163363,40841,20475,40625,          1396,68,124,836,74,186,  42,43,69,28,25,218}      ;

    //Xe-1-10-Titan-90
    //{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}  ;
    //{326726,163363,40841,20475,40625,     576,2.56e-5,    33,34,54,22,19.5,170}          ;

    //He-0-5-Cu-1
    //{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    };
    //{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  } ;

    //Ne-0-5-Cu-1
    //{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     } ;
    //{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};

    //Xe-2500-5-Cu-10
    //{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}                               ;
    //{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};



    Int_t Color[EventTypes]=
    { 803,793,632,600,430                 //cosmic
      ,416,419,828                    //vessel cobre
      ,594,875,616,810,892,619,871,886          //activar cobre
      //,892,594,875,616,810,619            //activar steel
      //,594,828                      //activar titanio
      ,623,920,396,392,832,831              //activar xenon
    };


    Char_t Title[EventTypes][40]={
      "#gamma (1<E<10) MeV","#gamma (10<E<100) MeV","#gamma (0.1<E<100) GeV","Neutrones", "Muones",
      "{}^{238}U","{}^{232}Th","{}^{40}K",
      "{}^{46}Sc","{}^{48}V","{}^{54}Mn","{}^{56}Co","{}^{57}Co","{}^{58}Co","{}^{59}Fe","{}^{60}Co",
      //"{}^{7}Be","{}^{46}Sc","{}^{48}V","{}^{54}Mn","{}^{56}Co","{}^{58}Co",
      //"{}^{46}Sc","{}^{40}K",
      "{}^{3}H","{}^{7}Be","{}^{125}Sb","{}^{121}Te","{}^{123}Te","{}^{127}Xe"
    };


    TCanvas *c1 = new TCanvas("c1","c1",700,500);
    TCanvas *c2 = new TCanvas("c2","c2",700,500);
    TCanvas *c3 = new TCanvas("c3","c3",700,500);
    Int_t sizeR=450;
    Int_t sizeT=150;
    Int_t sizeP=100;
    Double_t pressure=10.;
    Double_t LimTh1=200;  // en KeV
    Double_t BinsTh1=0.5;  // numero de bins por cada kev
    string File="Xe-10-5-Cu-1";
    string FileData=File+"data.md";
    string Filepdf=File+".pdf";
    string Filepdfr2=File+"r2.pdf";
    string Filepdfreduce=File+"reduce.pdf";
    gROOT->SetBatch(kTRUE);
```

```

  char NameFile[100], NameFile2[100], NameFile3[100];
  strcpy(NameFile,Filepdf.c_str());
  strcpy(NameFile2,Filepdfr2.c_str());
  strcpy(NameFile3,Filepdfreduce.c_str());

  char NameData[100];
  strcpy(NameData,FileData.c_str());
  FILE *Data;
  Data=fopen(NameData,"w");
  //// Conversion entre R y Td
  Double_t Vdaux,field;
  Double_t voltage=750;
  std::vector<Double_t> Vd;
  std::vector<Double_t> Td;
  std::vector<Double_t> Rd;

  ///////////  Obtener lista de td y su posicion asociada
  TRestDetectorGas *gas=new TRestDetectorGas ("server", "Xenon-Methane 0.5Pct 0.1-10E3Vcm");
  gas->SetPressure(pressure);
  for (double i=0;i<=rmax-rmin+resolution;i=i+resolution){
    Rd.push_back(rmin+i);
     field=(voltage/((rmin+i)*(rmin+i)))*rmin*rmax*(1/(rmax-rmin));
    gas->SetElectricField(field);
    Vdaux=gas->GetDriftVelocity();
    Vd.push_back(Vdaux);
    if (i==0) {Td.push_back(resolution/(Vd[0]*1000));    }
    if (i>0)  {Td.push_back((Td.back())+(resolution/(Vdaux*1000))); }
  }
  Double_t Tmax=Td.back();
  cout<<"Tmax (ms):"<<Tmax<<endl;
  cout <<"Conversion de R a t correcta"<<endl;

  ///////////////////////// Plot 1D

  THStack *hs= new THStack("hs","; #it{E} [keV]; Eventos keV^{-1} s^{-1}");
  THStack *hsr2= new THStack("hsr2","; #it{r} [m]; Eventos s^{-1}");
  THStack *hsreduce= new THStack("hsreduce","; #it{E} [keV]; Eventos keV^{-1} s^{-1}")

  Int_t Exist, Existred, Existr2;
  std::vector<Int_t> Exist2;
  std::vector<Int_t> Exist2red;
  std::vector<Int_t> Exist2r2;
  for (int i=5;i<EventTypes;i++){
    string aux=std::to_string(i);
    string FileRoot=File+"/Runwater"+aux+".root";
    TH1* h1 = new TH1D("h1",";#it{E} [keV]; Eventos keV^{-1} s^{-1}",BinsTh1*LimTh1,0,LimTh1);
    TH1* h1r2  = new TH1D("h1r2","#it{r} [m]; Eventos s^{-1}",(rmax-rmin)/10 , rmin/1000,rmax/1000);
    TH1* h1reduce = new TH1D("h1reduce",";#it{E} [keV]; Eventos keV^{-1} s^{-1}",BinsTh1*LimTh1,0,LimTh1);
    h1r2->SetStats(0);
```

```
          h1r2->SetMarkerColor(Color[i]);
          h1r2->SetFillColor(Color[i]);
          h1->SetStats(0);
          h1->SetMarkerColor(Color[i]);
          h1->SetFillColor(Color[i]);
          h1reduce->SetStats(0);
          h1reduce->SetMarkerColor(Color[i]);
          h1reduce->SetFillColor(Color[i]);
          Exist=0;   Existred=0;   Existr2=0;
          Single(FileRoot,(EventNumber[i]/13),h1,sizeR,sizeT,sizeP,Td,Rd,Tmax,LimTh1,BinsTh1,&Exist,h1r2,h1reduce,&Existred,&Existr2,Data);

          c1->cd();   h1->Draw("HIST BAR");   hs->Add(h1);
          c2->cd();   h1r2->Draw("HIST BAR");     hsr2->Add(h1r2);
          c3->cd();   h1reduce->Draw("HIST BAR");   hsreduce->Add(h1reduce);
          if (Exist != 0 ){  Exist2.push_back(1);}   else Exist2.push_back(0);
          if (Existred != 0 ){  Exist2red.push_back(1);}   else Exist2red.push_back(0);
          if (Existr2 != 0 ){  Exist2r2.push_back(1);}   else Exist2r2.push_back(0);

        }

        fclose(Data);
        //////////////  Leyenda

        Exist=0;
        Existred=0;
        Existr2=0;
        for (int i=0; i < Exist2.size(); i++)
        Exist+=Exist2[i];           //exist pasa a ser el numero de eventos no nulos total
        for (int i=0; i< Exist2red.size(); i++)
        Existred+=Exist2red[i];
        for (int i=0; i < Exist2r2.size(); i++)
        Existr2+=Exist2r2[i];

        Double_t aux=0.35+(0.025*(EventTypes-Exist));
        Double_t auxred=0.35+(0.025*(EventTypes-Existred));
        Double_t auxr2=0.35+(0.025*(EventTypes-Existr2));
        TLegend *legend1 = new TLegend(0.80,aux,0.98,0.9);
        TLegend *legend1red = new TLegend(0.80,auxred,0.98,0.9);
        TLegend *legend1r2 = new TLegend(0.80,auxr2,0.98,0.9);

        legend1->SetTextSizePixels(23);
        legend1red->SetTextSizePixels(23);
        legend1r2->SetTextSizePixels(23);

        for (int i=0;i <EventTypes;i++){
          if (Exist2[i]==1. ){
            TH1* h1aux = new TH1D("h1aux",";  ;  ",1,0,2);
            h1aux->SetMarkerColor(Color[i]);
            h1aux->SetFillColor(Color[i]);
            legend1->AddEntry(h1aux,Title[i],"f");
          }
          if (Exist2red[i]==1. ){
            TH1* h1auxred = new TH1D("h1auxred",";  ;  ",1,0,2);
            h1auxred->SetMarkerColor(Color[i]);
            h1auxred->SetFillColor(Color[i]);
            legend1red->AddEntry(h1auxred,Title[i],"f");
          }
          if (Exist2r2[i]==1. ){
            TH1* h1auxr2 = new TH1D("h1auxr2","; ; ",1,0,2);
            h1auxr2->SetMarkerColor(Color[i]);
            h1auxr2->SetFillColor(Color[i]);
            legend1r2->AddEntry(h1auxr2,Title[i],"f");
          }
        }
        /////////////////// Senal de neutrinos

        Double_t pi=TMath::Pi();
        Double_t J=31*pow(pi,6)/252 , T ,Landa;
        T=3.500;    Landa=0.89;   //nu_e, el mas dificil de ver al ser el promedio mas bajo de energia
        Double_t U = (5e52)* 624151;
        Double_t LANDA=U/T;
        TF1 *distr = new TF1("distr","[0]*pow((x),4)/ ( [1]*pow([2],5) * (1+exp((x)/[2])) )",0,80);
        distr->SetParameters(LANDA,J,T);

        T=5.000;  Landa=0.63;   LANDA=U/T;     //anti nu_e
        TF1 *distrAnti = new TF1("distr","[0]*pow((x),4)/ ( [1]*pow([2],5) * (1+exp((x)/[2])) )",0,80);
        distrAnti->SetParameters(LANDA,J,T);

        T=8.000;  Landa=0.39;  LANDA=U/T;    //otros
        TF1 *distrOtros = new TF1("distr","[0]*pow((x),4)/ ( [1]*pow([2],5) * (1+exp((x)/[2])) )",0,80);
        distrOtros->SetParameters(LANDA,J,T);

        //distr->GetXaxis()->SetTitle("E_{#nu} [MeV]"); distr->GetYaxis()->SetTitle("#frac{dN}{dE_{#nu}}");
        //distr->SetTitle("");   distr->SetLineColor(kBlack);
        //TAxis *distraxis=distr->GetYaxis();
        //distraxis->SetNdivisions(503);
        //distraxis->SetMaxDigits(2);
        //distr->GetYaxis()->SetTitleOffset(1.1);
        //distr->Draw();
        //distrAnti->SetLineColor(kRed); distrAnti->Draw("SAME");
        //distrOtros->SetLineColor(kBlue);   distrOtros->Draw("SAME");
        //TLegend *distrleg =new TLegend(0.72,0.7,0.9,0.9);
        //distrleg->AddEntry(distr,"#nu_{e}","l");              distrleg->AddEntry((TObject*)0, "", "");
        //distrleg->AddEntry(distrAnti,"#bar{#nu}_{e}","l");     distrleg->AddEntry((TObject*)0, "", "");
        //distrleg->AddEntry(distrOtros,"Otros sabores","l");
        //distrleg->Draw();

        Double_t angle=0.;
        TRandom3 *angleaux= new TRandom3();
        angleaux->SetSeed(0);
        angle= (angleaux->Rndm())*pi/2;
```

```
Int_t Z=10;  Int_t N=10;  Int_t A=N+Z;    Double_t mn=(N*939.565560)+(Z*938.272013); // en MeV
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);

Z=2; N=2; A=N+Z;    mn=(N*939.565560)+(Z*938.272013);
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);

Z=54;  N=77;  A=N+Z;  mn=(N*939.565560)+(Z*938.272013);
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);
recoilXe->SetParameters(mn,angle);

//recoilHe->SetTitle(""); recoilHe->SetLineColor(kRed);  recoilHe->Draw();
//recoilHe->GetXaxis()->SetTitle("E_{#nu} [MeV]");  recoilHe->GetYaxis()->SetTitle("E_{recoil} [MeV]");
//recoilNe->SetLineColor(kBlack);  recoilNe->Draw("SAME");
//recoilXe->SetLineColor(kBlue);  recoilXe->Draw("SAME");
//TLegend *recleg=new TLegend(0.1,0.7,0.25,0.9);
//recleg->AddEntry(recoilHe,"Helio","l");      recleg->AddEntry((TObject*)0, "", "");
//recleg->AddEntry(recoilNe,"Neon","l");      recleg->AddEntry((TObject*)0, "", "");
//recleg->AddEntry(recoilXe,"Xenon","l");
//recleg->Draw();

TRandom3 *r3 = new TRandom3();
r3->SetSeed(0);
TH1D *h1 = new TH1D("h1",";#it{E} [KeV]; Eventos",BinsTh1*LimTh1,0,LimTh1);
Double_t NuEvents=153*pow((N/22),2)*(pressure/10)*pow((rmax/(1000*4)),3)*(300/293.15)*(Time/10000); //Recordar cambiar Time
if ( NuEvents<1.0)
NuEvents=r3->Poisson(NuEvents);
cout<<"Eventos de neutrinos: "<<NuEvents<<endl;

for (int i = 0;i < NuEvents; i++){
  Double_t aux = distr->GetRandom(); //devuelve E en MeV
  Double_t AuxRecoil=(recoilXe->Eval(aux))*1000; //en KeV
  h1->Fill(AuxRecoil);
  cout<<"E generada [MeV]: "<<aux<<"\nE recoil [keV]: "<<AuxRecoil<<endl;
}
h1->SetStats(0);
h1->SetMarkerColor(1);
h1->SetFillColor(1);
h1->Draw("HIST BAR");
hs->Add(h1);

//////////////////////  Graficar

c1->cd();
TH1D *last=(TH1D*)hs->GetStack()->Last();  hs->SetMaximum((1.25)*(last->GetBinContent(last->GetMaximumBin())));  hs->Draw("HIST BAR");
```

```
  gPad->RedrawAxis("Y");
  legend1->Draw();

  c2->cd();
  TH1D *lastr2=(TH1D*)hsr2->GetStack()->Last();  hsr2->SetMaximum((1.25)*(lastr2->GetBinContent(lastr2->GetMaximumBin())));  hsr2->Draw("HIST BAR");
  gPad->RedrawAxis("Y");
  legend1r2->Draw();

  c3->cd();
  TH1D *lastreduce=(TH1D*)hsreduce->GetStack()->Last();  hsreduce->SetMaximum((1.25)*(last->GetBinContent(last->GetMaximumBin())));  hsreduce->Draw("HIST BAR");
  gPad->RedrawAxis("Y");
  legend1red->Draw();

  //last->Draw("HIST BAR");

  c1->cd();  c1->SaveAs(NameFile);  c1->Close();
  c2->cd();  c2->SaveAs(NameFile2);  c2->Close();
  c3->cd();  c3->SaveAs(NameFile3);  c3->Close();

  /////////  Optimizacion tamano de la mesh de clusterizado

  BinsTh1=1;
  string FileRoot="Xe-0-5-Cu-1/CalibrationMesh.root";
  string FilePdf="Calibration.pdf";
  TH1* h1r2  = new TH1D("h1r2","#it{r^{2}} [m^{2}]; Eventos s^{-1}",(rmax-rmin)/10 ,pow((rmin/1000),2),pow((rmax/1000),2));
  TH1* h1reduce = new TH1D("h1reduce",";#it{E} [keV]; Eventos keV^{-1} s^{-1}",BinsTh1*LimTh1,0,LimTh1);
  Double_t aux=0, aux2=0, aux3=0, aux4=0, aux5=0, aux6=0;
  Double_t Efi5, Efi10, Efi15, Efi20, Efi25, EfiReal;
  TMultiGraph *mg= new TMultiGraph("mg","");
  mg->SetTitle(";Numero de nodos; #varepsilon_{#it{cluster}}");
  TGraph *g2= new TGraph();  g2->SetMarkerStyle(21); g2->SetMarkerColor(kRed); g2->SetLineColor(kRed);
  TGraph *g3= new TGraph();  g3->SetMarkerStyle(22); g3->SetMarkerColor(kBlue); g3->SetLineColor(kBlue);
  TGraph *g4= new TGraph();  g4->SetMarkerStyle(23); g4->SetMarkerColor(kGreen); g4->SetLineColor(kGreen);

  FILE * values;
  values= fopen("values.md","w");
  fprintf( values,"Tracks SizeR  SizeT  SizeP \t Efi5 \t Efi10 \t Efi15 \t Efi20 \t Efi25 \n");
  Int_t Exist;
  sizeR=450;
  sizeT=150;
  sizeP=100;
  //for ( sizeP=50;  sizeP<=1500 ; sizeP+=50){
    //for ( sizeT=50;  sizeT<=1500 ; sizeT+=50){
      for ( sizeR=10; sizeR<=2000 ; sizeR+=10){
        TH1* h1 = new TH1D("h1",";#it{E} [keV]; Eventos",BinsTh1*LimTh1,0,LimTh1);
```

```
          Single(FileRoot,10000,h1,sizeR,sizeT,sizeP,Td,Rd,Tmax,LimTh1,BinsTh1,&Exist,h1r2,h1reduce);
          //h1->Draw("HIST BAR");
          aux=0;          //numero de tracks conseguidos
          Int_t Bin5 = h1->Fill(5);
          Int_t Bin10 = h1->Fill(10);
          Int_t Bin15 = h1->Fill(15);
          Int_t Bin20 = h1->Fill(20);
          Int_t Bin25 = h1->Fill(25);


          aux2=(h1->GetBinContent(Bin5))-1;
          aux3=(h1->GetBinContent(Bin10))-1;
          aux4=(h1->GetBinContent(Bin15))-1;
          aux5=(h1->GetBinContent(Bin20))-1;
          aux6=(h1->GetBinContent(Bin25))-1;
          aux=0;

          for (int i=0; i<=LimTh1;i++){
            Int_t Bin = h1->Fill(i);
            aux=aux+(h1->GetBinContent(Bin))-1;
          }

          Efi5=aux2/10000;
          Efi10=aux3/10000;
          Efi15=aux4/10000;
          Efi20=aux5/10000;
          Efi25=aux6/10000;


          if (Efi5>=0.9 && NTracks>=8000)
          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);

          g2->AddPoint(sizeR,Efi5);
          g3->AddPoint(sizeR,Efi10);
          g4->AddPoint(sizeR,Efi15);

          h1->Reset();
        }
        //}}
    fclose (values);

    TCanvas *gcanvas= new TCanvas("gcanvas","gcanvas",700,500);
    mg->Add(g2,"AC");   mg->Add(g3,"AC");   mg->Add(g4,"AC");
    mg->Draw("AC");
    TLegend *leg= new TLegend(0.7,0.7,0.9,0.83);
    leg->AddEntry(g2," 5 keV","l");
    leg->AddEntry(g3," 10 keV","l");
    leg->AddEntry(g4," 15 keV","l");
    leg->Draw();

    char NameFile[100];
    strcpy(NameFile,FilePdf.c_str());
    gcanvas->SaveAs(NameFile);
    gcanvas->Close();


    return 0;
}
```

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.