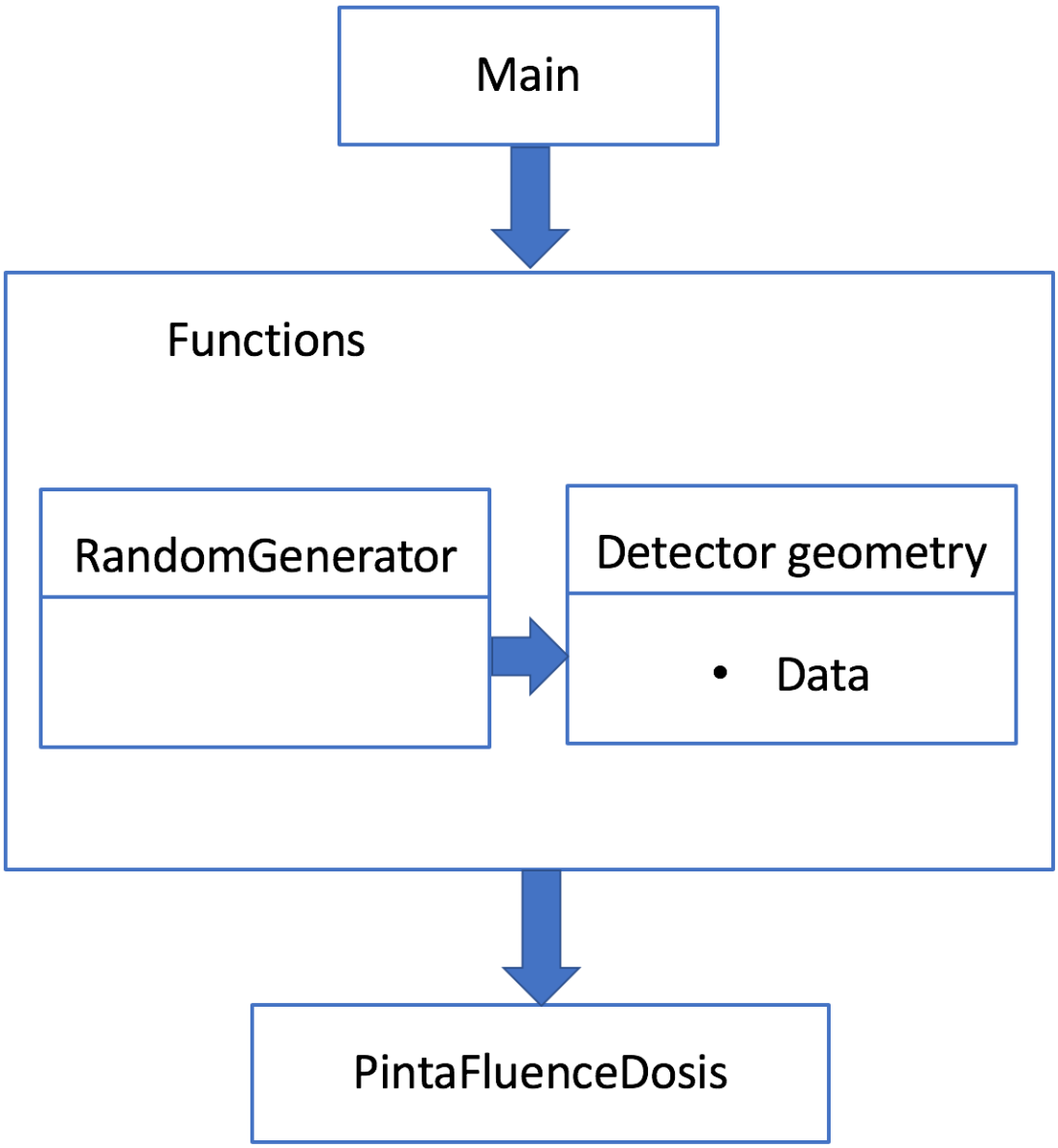
SCXM Code Guide

Daniel Fernandez Fernandez
daniel.fernandez.fernandez.94@gmail.com

The scheme of the code:



Download the code:

```
git clone https://github.com/ferdani/SCXM
cd SCXM
```

Functions

```
Created on Fri Dec 14 12:29:56 2018

@author: Daniel Fernandez Fernandez
daniel.fernandez.fernandez.94@gmail.com

DEFINE THE READ-GET DATA FUNCTION, DEFINE THE DISTRIBUTION FUNCTIONS
AND DEFINE THE DOSE, FLUENCE AND AUXILIAR FUNCTIONS.
```

Modules

scipy.interpolate
numpy
openpyxl
matplotlib.pyplot
random

Functions

BeamPhotons(n)
```
Generate the empty array of the beam pipe line with angle 0 (the angle of the beam line)
```

CoherentCrossSection(InitEnergy, theta, Z, Finterpolant)
```
The coherent cross section, by definition: is the Klein-Nishina cross section corrected with F
atomic form factor
```

CoherentPolarization3D(InitEnergy, theta, phi, Z, Finterpolant)
```
The coherent cross section in three dimensions with the polarization
```

Dose(fluence, InitEnergy, Sigma_photo, Sigma_incoh)
```
Estimated the minimum dose according with Villanueva's paper. It is a local dose in [Gy]
```

FinalEnergy(E, theta)
```
Calculate the final energy of the photon with the Compton equation
```

FinterpolantFunction(X, Z, Finterpolant)
```
Using the data from F values per each element an interpolant is build, with this interpolant it's posible
to extract the correction value
```

FluenceDosePlotterFeatureSize(InitEnergy, Fluence, DoseGy, FeatureSize, Sample)
```
The plotter for Fluence and Dose. Exist a file called PintaFluenceDosis.py to do that.
This function is not useful
```

Fluence_bkg_air_water(L, d_prime, d, a, rho_matter, rho_bkg_air, rho_bkg_water, Bkg_air_Elements, Bkg_water_Elements, Bkg_air_Atoms, Bkg_water_Atoms, MoleculeElements, MoleculeAtoms, Sigma_matter, Ageometric_matter, Effi_matter, Sigma_air_bkg, Sigma_water_bkg, Ageometric_air_bkg, Ageometric_water_bkg, Effi_air_bkg, Effi_water_bkg)
```
Estimate molar mass of molecule, then estimate sigma prime and then estimated the Fluence in a system: bkg-matter-bkg
units: cm and g. All using auxiliar functions. With water-
equivalent around the material of interest and then air around all.
It means, the realistic situation with cell (water-equivalent and material of interest) + bkg (air).
Where the parameter "a" is the size of air.
```

Fluence_bkg_water(L, d_prime, d, rho_matter, rho_bkg, BkgElements, BkgAtoms, MoleculeElements, MoleculeAtoms, Sigma_matter, Ageometric_matter, Effi_matter, Sigma_bkg, Ageometric_bkg, Effi_bkg)
```
Estimate molar mass of molecule, then estimate sigma prime and then estimated the Fluence in a system: bkg-matter-bkg
units: cm and g. All using previous auxiliar functions. With bkg only water-equivalent around the material of interest
```

IncoherentCrossSection(InitEnergy, theta, SinterpolantSmall, SinterpolantBig)
```
The incoherent cross section, by definition: is the Klein-Nishina cross section corrected with S
scattering function
```

IncoherentPolarization3D(InitEnergy, theta, phi, SinterpolantSmall, SinterpolantBig)
```
The incoherent cross section in three dimensions with the polarization
```

IntegratedCrossSection(theta, dSigmaArray)
```
Integrated any differential cross section array without phi dependence, returns Sigma total and
the error in the integration
```

IntegratedCrossSectionEnergyCorrection(theta, dSigmaArray, InitEnergy, MoleculeElements, MoleculeAtoms)
```
Integrated a differential cross section without phi dependence with de energy dependence
(see the equation in Villanueva's paper) corrected with the energy and molecular mass
```

KNpolarization(InitEnergy, theta, phi)
```
The Compton cross section with polarization, it means without phi average in Klein-Nishina
```

KleinNishinaCrossSection(InitEnergy, theta)
```
The differential Compton cross section called Klein-Nishina
```

Molar_mass(MoleculeElements, MoleculeAtoms)
```
Estimate molar mass of a molecule
```

PhotoabsorptionSigma(MoleculeElements, MoleculeAtoms, InitEnergy)
```
Estimate the photoabsorption sigma cross section for molecule with the NIST tables for each energy
```

PhotonScatteringElementNumbers(nMoleculeIncoherent, nMoleculeCoherent, ProbabilityIncoherentElement, ProbabilityCoherentElement)
```
Calculate the number of events n_inco_element and n_coh_element (per each element in the molecule).
This function use the results per molecule for PhotonScatteringMoleculeNumbers function
```

PhotonScatteringMoleculeCrossSectionsCorrected(InitEnergy, thetaRad, ArrayZ, ArraySinterpolantSmall, ArraySinterpolantBig, ArrayFinterpolant, n, MoleculeElements, MoleculeAtoms)

Estimate the cross sections per element into molecule and the total cross section, both for the incoherent process
and for the coherent process with the energy and molecular mass corrections of the integral dose integral

PhotonScatteringMoleculeNumbers(n, ProbabilityIncoherentMolecule, ProbabilityCoherentMolecule)
Calculate the number of events n that are produced in the molecule by a distribution or another of
cross sections. In principle: incoherent and coherent scattering. That is, n_inco and n_coh.
This function use the results for PhotonScatteringMoleculeProbabilities function

PhotonScatteringMoleculeProbabilities(InitEnergy, thetaRad, ArrayZ, ArraySinterpolantSmall, ArraySinterpolantBig, ArrayFinterpolant, n, MoleculeAtoms)
Estimate the cross section per element in a molecule and the total cross-section, both for the incoherent process
and for the coherent process. The probabilities associated with scatterings on each element of the molecule
are also estimated.

ReadGetData(Element, TableName)
Read the data into the Excel and load the values for S and F functions per each element (the choosen Element as variable)

SigmaTotalMatter(InitEnergy, thetaRad, ArrayZ, ArraySinterpolantSmall, ArraySinterpolantBig, ArrayFinterpolant, n, MoleculeAtoms)
Estimate the total cross section for the material of interest (axuliar function in the Dose and Fluence evaluation)

Sigma_Mass_Omega(Sigma, Ageometric, Effi, M_molec)
Estimate the generic sigma prime (axuliar function in the Dose and Fluence evaluation)

SinterpolantFunction(X, SinterpolantSmall, SinterpolantBig)
Using the data from S values per each element an interpolant is build, with this interpolant it's posible
to extract the correction value

THpolarization(theta, phi)
The Thomson cross section with polarization

ThompsonCrossSection(theta)
The differential Thomson cross section

Xe_MeanFreePath(energy, Xe_MFP_interpolant)
Use the interpolant for Xenon with the mean free path and evaluate it for an input-variable energy

Xe_MeanFreePath_interpolant()
Using the data of photoelectric absorption from the NIST an interpolant is created to build the
mean free path for Xenon

Data

c = 1.0
hbar = 6.582119514e-22
m = 0.510998928
re = 2.8179402894e-13

RandomGenerator

Created on Fri Dec 14 12:44:36 2018

@author: Daniel Fernandez Fernandez
daniel.fernandez.fernandez.94@gmail.com

MONTE CARLO SIMULATION RANDOM NUMBERS FOR A DISTRIBUTION FUNCTION

Modules

Functions
matplotlib.pyplot
sys
numpy
random

Functions

PlotterRandomNumbersDistributions(C, n, InitEnergy, thetaRad, SigmaArray, XR, YR)

The Poltter for the random numbers and the real curve distribution. That function is useful to see all the points randomly generated under the real distribution function (in color red)

RandomNumberDistribution(C, n, InitEnergy, thetaRad, Z, SinterpolantSmall, SinterpolantBig, Finterpolant)

The principal function to build random pair numbers for a distribution function. For example the Klein-Nishina or for example the incoherent cross section. That function gives a Monte Carlo simulation for a distribution function

RealDistributionFunctionNumbers(C, InitEnergy, thetaRad, Z, SinterpolantSmall, SinterpolantBig, Finterpolant)

Build the array of the choosen differential cross section (aka distribution function) to plot it (in red) then with the random pair numbers generated

DetectorGeometry

```
Created on Fri Dec 14 12:48:56 2018

@author: Daniel Fernandez Fernandez
daniel.fernandez.fernandez.94@gmail.com

DETECTOR GEOMETRY, BACKGROUNDS, INTERACTIONS ON THE DETECTOR, GEOMETRIC ACEPTANCE AND EFFICIENCIES
```

Modules

math	matplotlib.pyplot	sys
numpy	random	

Functions

AirBlock(a_air, b_air, IntroducedPhotonsAir_XR)
```
The air into the space between the water-equivalent block and the detector,
in this function the interaction of the photons into this block is evaluated
```

Corners(d, large, hole, theta1)
```
The function that corrects the corners for the right evaluation of the mean free path
```

DetectorBox(a_det, b_det, c_det, IntroducedPhotonsDetector_XR, Lambda_det)
```
This function gives the geometry of the detector and evaluated the response of the detector to photons
```

Efficiencies(UpperPhotons_XR, Upper_NoWaterInteract_XR, Upper_NoAirInteract_XR, permitted_XR, MeasuredPhoton_XR)
```
The function that evaluates the efficiencies and aceptances with the ratio between the length of the arrays
that contains the photons
```

GenerateBackgrounds(l1_air, l1_water, l_cell, l2_water, l2_air, n)
```
Probability to colision in the air line after the beam pipe, then in the water, then in the cell
and again into water and finally air
```

InteractDetectorBackgrounds(a_det, b_det, c_det, l1_air, l2_air, AirLine_1_Bkg, WaterLine_1_Bkg, WaterLine_2_Bkg, AirLine_2_Bkg, Lambda_det)
```
With the angles of each background independently we can evaluate the permitted angles and the
interaction into de detector
```

PlotterEfficiencyDistributions(n, InitEnergy, UpperPhotons_XR, Upper_NoAirInteract_XR, permitted_XR, MeasuredPhoton_XR)
```
The function that plots in histograms the photons (all the types) distributed in theta
The types are: upper, not interact in air or water, permitted by geometry and measured in the detector
```

ProbabilityInteract(Path, Lambda)
```
The law of interact probability
```

UpperPhotons(XR)
```
Extact the photons in the upper part of the beam linea. Between theta [0, pi] angles interval
```

WaterBlock(a_water, b_water, IntroducedPhotonsWater_XR)
```
The water-equivalent around the material of interest block, in this function the interaction of the photons
into thi block is evaluated
```

Data

X_cell = 0.0
X_det = 0.0
Y_cell = 0.0
Y_det = 0.0
Z_cell = 0.0
Z_det = 0.0