**Monte Carlo Small Angle Scattering, By Max Proft**

**Installation**

You need python (version 2.7 or newer) and a range of common packages. If you like, you can download python from the following website since it contains all packages that you will need for this program.
https://store.enthought.com/downloads/
This program works on Windows and Linux (and probably mac as well).
If you have Fortran installed, run `make` to create a shared object to speed up the intensity calculation by 2-10x.

**Overview**

Run newgui.py

- on Windows, open in Canopy and then click run
- on Linux or OS X run directly (if necessary, make sure it's executable and that the first line points to your Python installation)

Choose a Monte Carlo Model
Click 'Real Space' to view the points shape.
Click 'Calculate Intensity' to calculate the detector intensity.
A description of the parameters for each model can be found below.

To make a sequence (or not), click on 'Edit Sequence' (or 'No Sequence')
To decide which variable you want to run, click on "Common Variables". It is case sensitive.
If you want a linear sequence, ensure that 'linear sequence is selected.
If you want a random value from a Gaussian distribution, select 'Gaussian'.
The mean is given by the variable in "Parameters", and the standard deviation is in the same units.

To show/hide more advanced options, click on 'Advanced Options/Simple Options'

Ensure that you have no spaces at the start or end of "Subfolder" or the start of "File Name".

After an intensity is created, you may want to plot a ring around the centre from 0 to 2π. Specify the width of the ring, the radius as a proportion of the distance to the edge, and the number of points you would like to plot. If you request too many points, there may be some wedges which do not contain any points.

CSV files are created containing the information. It does not get affected by specifying upper and lower bounds, log plots, etc. Only the output images are affected.

You can add Monte Carlo function by editing density_formula.py
You can add analytic models by editing analytic_formula.py

Please send any bugs to me at proftmax+mcsas@gmail.com
Also, if you define any new shapes, send the code to me and I can put it into the program.

**How The Program Works**

Making Points – Points_For_Calculation(), in Monte_Carlo_Functions.py
A regular lattice of points is created. The dimensions of the lattice are **x_dim**, **y_dim**, and **z_dim.** The program is set up so that **x_dim = y_dim = 2*radius_1.**

The distance between neighbouring points in the x and y direction is **ave_dist** or 'Neighbouring Point Distance' (i.e. the density will be inversely proportional to the cube). Once the lattice is created, the points are made to move around randomly. This was done with a Gaussian probability distribution with standard deviation **travel**. Even when running a sequence**, ave_dist** and **travel** are forced to be the same.

In the z direction, you can scale it so that you can decrease the number of points in the z direction (useful if you have a large aspect ratio). The equivalent values in the z direction are **z_scale** * **ave_dist** and **z_scale*travel**.

The coordinates of each point is put into the Monte Carlo density function, density_formula.py. Then the points are rotated around the x, y and z axes.

Making an Intensity – Detector_Intensity(Points), in Monte_Carlo_Functions.py
Each point on the detector is converted to QSpace.

If we don't make the small angle approximation, the z component is given by:
Qz = 2***EHC***sin( sqrt(x**2 + y**2) ***QSize/pixels**/2/**EHC** )**2 (EHC = 2pi/wavelength)
Else, Qz = 0.

If we assume that it is not radially symmetric, the formula for all points R with density f, and any point on the detector Q:

$$I = \left[\sum_k f_k \cos(Q \cdot R_k)\right]^2 + \left[\sum_i f_k \sin(Q \cdot R_i)\right]^2$$

Else, if it is radially symmetric:

$$I = \left[\sum_k f_k \cos(Q \cdot R_k)\right]^2$$

In the program, I refer to this as **Intensity.**
The output of this function normalised: **Intensity/sum(intensity)**

(Equations from "The use of Monte-Carlo Simulations to Calculate Small-Angle Scattering Patterns" by Bryan C. McAlister and Brian P. Grady)

Average Intensity – Average_Intensty(), Monte_Carlo_Functions.py
The above two programs are run **num_plots** times (Number of Plots to Average). Each time one is finished, it gets added to **cumulative.** Once all plots have been run, we output
**Intensity** = **cumulative/num_plots.**

Radial Intensity – radial(), Monte_Carlo_Functions.py

Going through step by step from 0 to **num_plot_points** = **pixels**/2, the radius in pixel space is increased. We go through the intensity array and find which points lie within **delta**/2 of this radius, **delta**=1.

Points_Plot – plots real space, Plotting_Functions.py
This plots real space. Don't have too many points otherwise it will take a long time to load.

Intensity_plot, Plotting_Functions.py
This plots the 2D intensity

radial_intensity_plot, Plotting_Functions.py

Creating data - Plotting The Angle At A Fixed Radius – plotting_circle(Intensity), Monte_Carlo_Functions.py

angle_plot, Plotting_Functions.py
Plots the angle at a fixed radius

**Parameters**
x_dim
y_dim
z_dim
x_theta
y_theta
z_theta
ave_dist
z_scale
Qz
symmetric
num_plots
QSize
pixels
EHC
num_plot_points
delta
circ_delta
theta_delta
pixel_radius
scale
altitude
azimuth
maximum
minimum
log_scale
bound
save_name
ThreeD
analytic
shape
subfolder
radius_1
radius_2
rho_1
rho_2
energy_wavelength
energy_wavelength_box
degrees
s_start
s_stop
s_step
SD
Gaussian
Save_img
Title
comments
path_to_subfolder
a variable with 2 at the end – used so I can 'hide' checkboxes and radio buttons.

**TO DO LIST**

**MODELS TO MAKE:**
**Hexacone**
**A series of ellipses of Random Length**

**Comment the code further**
**Make a better description of the code (redo this document)**
**Make a set of instructions on how to define a function.**
**Clear_mem function should be implemented in a better way.**

**When is it better to average more plots/decrease neighbouring point distance?**
**When is it better to decrease neighbouring point distance/zscale**

**Gaussian of two angles (x_theta and y_theta are both varied in a sequence)**