MICRO-ROUGHNESS MODEL: IMPLEMENTATION IN PENTRACK

PRAMODH SENARATH YAPA
Winter-Summer 2014 Co-Op Student
UltraCold Neutron Group
TRIUMF

INTRODUCTION

This report hopes to outline the progress made in implementing the Micro-Roughness Model (MRM) in PENTrack, the Monte Carlo simulation being developed for the UCN facility at TRIUMF, as well as show its shortcomings in the current state.

The baseline functions for implementing the model in PENTrack were taken from the latest version of GEANT4UCN (G4UCN) that was modified by Peter Gumplinger, to incorporate the MRM. This modification was based on code that was originally developed for use in GEANT4 UCN simulations by Stefan Heule, outlined in his PhD dissertation (Heule, 2008).

ACKNOWLEDGEMENTS

The task of implementing the MicroRoughness Model at times resembled the Hydra of Lerna, and tackling it would not have been possible without many in the UCN group. Special thanks to my supervisor, Rüdiger Picker, who patiently guided my efforts over my entire 8 months; Peter Gumplinger, for providing the template for the MicroRoughness model, answering all my questions about its inner working and being exceptionally gracious when I came to his door with the issues discovered in the implementation; Andy Miller for his advice on how the model should be implemented. My gratitude is also extended to the students during the summer term who listened to my many troubles; Erika Lloyd, Jan Becker, Katharina Martin, Morgan Cassels, and Katerina Katsika who wasn't a student, but had enough youthful enthusiasm to qualify many times over.

THE MICROROUGHNESS MODEL, AT A GLANCE

The MicroRoughness model is a neutron reflection and transmission model for rough surfaces, developed by Dr. Albert Steyerl. It is 'a perturbation method based on distorted Green's functions' (Heule, 2008), but it's formalism will not be described here. (For a complete description, see (Steyerl, 1972) or a less complete one, Chapter 4 of (Heule, 2008))

In the context of PENTrack, the MicroRoughness (MR) model is called when a neutron strikes a material boundary, and its behavioral changes (velocity/potential energy etc) need to be calculated.

The MR model characterizes the roughness of the surface **on the nanometer scale** and produces probability distributions based on 2 roughness parameters (**b** & **w**), the incident energy of the neutron and its incident angle and the Fermi pseudopotential of the material. It models the surface using a non-repeating pattern of Gaussian forms.

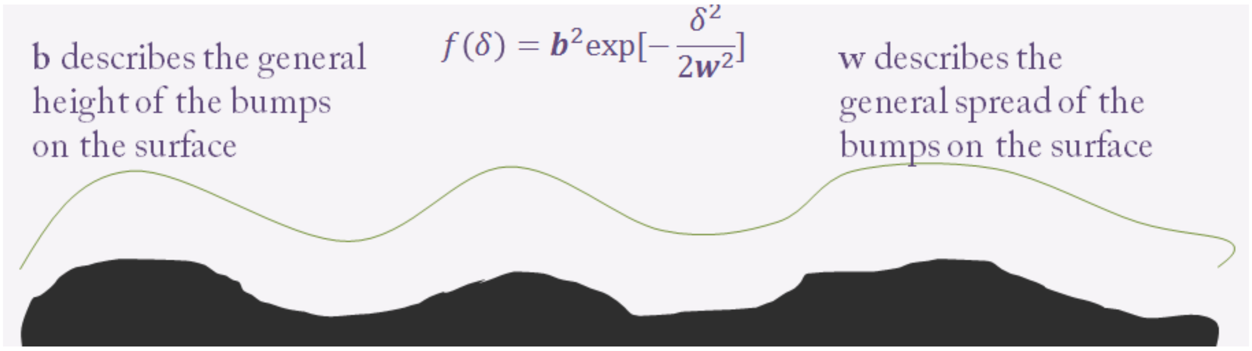

b describes the general height of the bumps on the surface
$f(\delta) = \boldsymbol{b}^2 \exp[-\frac{\delta^2}{2\boldsymbol{w}^2}]$
w describes the general spread of the bumps on the surface

FIGURE 1: A TOY REPRESENTATION OF A ROUGH SURFACE

It produces probability distributions for the solid angle into which the neutron could be transmitted/reflected, and then uses Monte Carlo methods to determine its trajectory. As these probability distributions are computationally intensive to produce, they can be saved as lookup tables, which are produced prior to simulation of the neutrons, and then referred to as needed. However, this approach has pitfalls as is described in a parallel report (Yapa, 2014).

GENERAL OVERVIEW

Due to issues discovered in the G4UCN implementation of the MRM, two parallel versions of PENTrack were developed. These issues will not be discussed in depth within this report, as they are outlined in its own report (**Implementation issues in GEANT4UCN MicroRoughness model**). In a nutshell, the issues encountered were that the probabilities calculated and saved in the lookup tables (which will be described later in this report) were incorrect in some situations. In light of this, in one version the implementation was done closely resembling the G4UCN implementation, while in the other, the lookup tables were eschewed. For large simulations, it is expected that the latter version will consume more computing resources and time.

One file was added to PENTrack in the course of this implementation,

MRHelperPointerless.cpp/hh (version 1)/ MicroRoughnessHelper.cpp/hh(version 2)

This file, for each version, contains the function used in the calculation of the MR probabilities. The files are named differently as some of the functions had to be modified to account for how the MRM implementation was carried out, however the values that the functions return are the same.

The files that were modified in the course of the implementation are,

globals.cpp* and *globals.h

- 6 new constants were added in *globals.h*. These were necessary to perform the probability calculations and produce the lookup tables.
- A new function *MicroRoughnessRotateVector()* was added.

geometry.cpp

The bulk of the functions from the *G4UCNMaterialProperties.cc/hh* were ported over to these files, into the struct called *material*. These functions initialize and compute the lookup tables, parse the tables for required values and check the validity of the MR transmission and reflection conditions. These were modified slightly between the two versions of PENTrack developed.

geometry.h

The variables required for MR calculations were declared in the *material* struct, as well as pointers to the lookup tables. An int called *interaction* was added to distinguish between the type of model to be used for a neutron-material-boundary interaction, ie whether it is specular, diffuse or MR. This variable is loaded from the *geometry.in* file. The function prototypes from the *geometry.cpp* were also added.

geometry.in

In the *[MATERIALS]* section, 3 new columns were added to define the material properties,

1. *RMS Roughness [nm]*: indicates the root mean square roughness value, b, for the MRM calculations.
2. *Correlation Length [nm]*: indicates the correlation length, w, for the MRM calculations.
3. *Interaction*: indicates the type of model to be used for a neutron interaction with the material, as explained above.

neutron.h

The OnHit function was overhauled to include the MR model, and the code was made modular; this is to say that individual functions were created for each possible way the neutron's behavior could be modified (ex. Specular Reflection, MicroRoughness Transmission, Diffuse Reflection, etc). The variable, *interaction*, described in the previous section is now used to determine which function should be called to modify the neutron's behavior.

main.c and Makefile

These file were changed to reflect that a new file, mentioned above, was added to PENTrack.

LOOKUP TABLES

The two different versions of PENTrack were developed because of the lookup tables potentially introducing error into the calculations. To understand this, first what lookup tables are and their need must be understood.

In the MR model, each pair of outgoing theta and phi angles, (θ_o *and* ϕ_o) have an associated probability for transmission and reflection, which can be calculated using the formula given by the model. The probability density for this will have one maximum value, which serves a useful purpose in the Monte Carlo method. The maximum value is indicated by the green circle in the below plot.

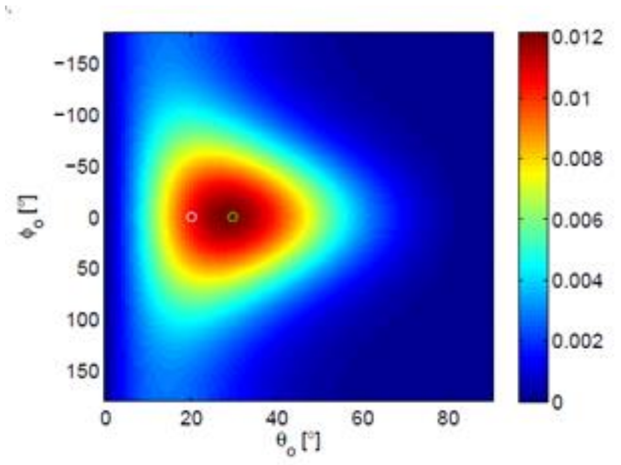


FIGURE 2: THE PROBABILITY DENSITY FOR THETA AND PHI

A plot like the above can be integrated over to give the total probability for the MR model to be used for that particular incident neutron. This value is also important in determining how the neutron's behavior is modified.

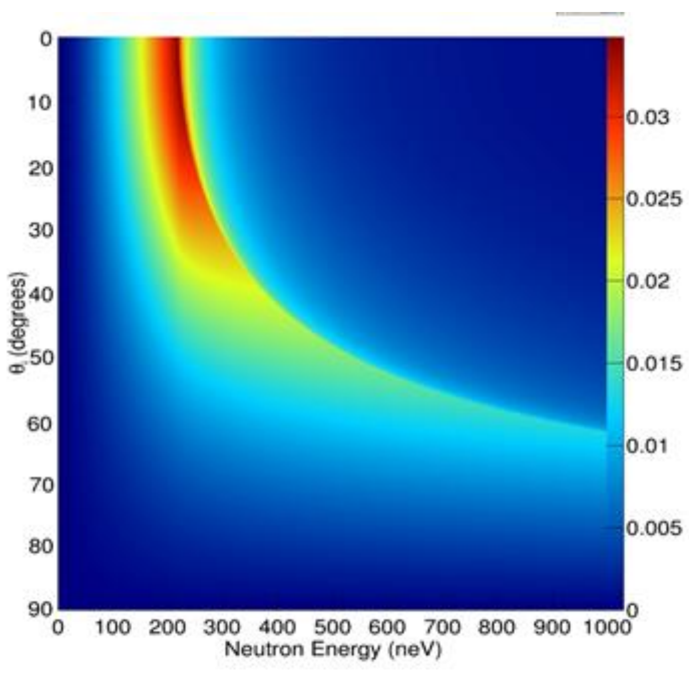


FIGURE 3: THE INTEGRATED PROBABILITY PLOTTED FOR NEUTRON ENERGY AND INCIDENT ANGLE

The max probability and integrated probability can be calculated as required when running a simulation, but for large simulations, this would prove costly in terms of resources and time. To get around this, lookup tables were introduced, which compute these values in a given range and increments, and then save them to be used when simulating the neutrons. These tables are initialized and filled before starting the actual simulation, and then when the probabilities are required, the tables are parsed for the closest value (as the tables are by definition discretized, the probabilities will only be approximate values, with the error being dependent on the increments (the granularity) of the table).

For each material, 4 tables are required in total, as shown by the figure below.

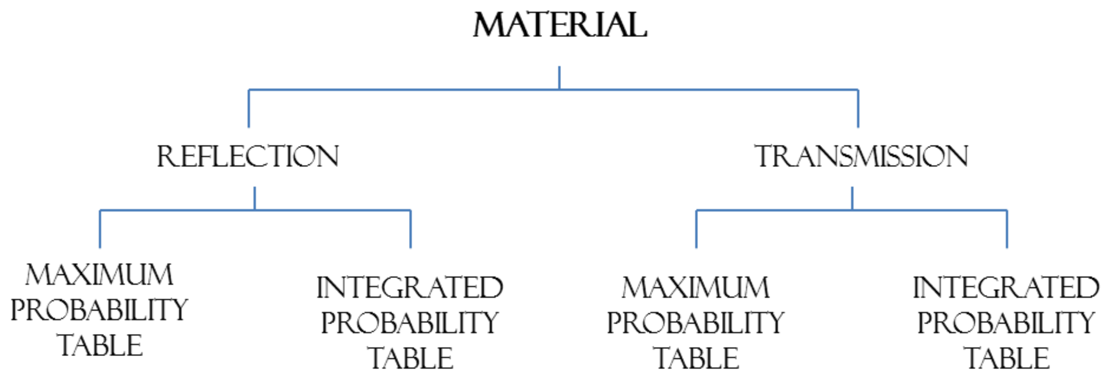

MATERIAL
REFLECTION
TRANSMISSION
MAXIMUM PROBABILITY TABLE
INTEGRATED PROBABILITY TABLE
MAXIMUM PROBABILITY TABLE
INTEGRATED PROBABILITY TABLE

FIGURE 4: TYPES OF LOOKUP TABLES

However, issues in the how the tables are calculated were discovered, which could possibly introduce errors into the simulation. To determine how these issues would affect the simulations, a version of PENTrack had to be developed that did not use lookup tables, but calculated the probabilities on the fly.

THE FINER DETAILS

This section will elucidate the points made in the general overview. As there is a significant overlap between the two versions of PENTrack, the functionalities described should be assumed to be in both versions unless otherwise specified. When specified, PENTrackNoMRTables will refer to the version developed without using lookup tables, and PENTrackMRTables will refer to the version with lookup tables, which mirrors the G4UCN implementation.

MRHelperPointerless.cpp/hh (version 1)/ MicroRoughnessHelper.cpp/hh(version 2)

This was the only file that was added to PENTrack, which contains the functions used for calculating the MR probabilities. The functions are contained within a class called MicroRoughnessHelper. These functions are,

Function name	Description
MicroRoughnessHelper	The constructor for the class, which is empty.
~MicroRoughnessHelper	The destructor for the class, which deletes any object of MicroRoughnessHelper that was created
GetInstance	Creates a MicroRoughnessHelper object named fpInstance, if it didn't exist already, and returns its pointer.
S2	The ratio of the amplitude of the diffusely reflected wave to the amplitude of the incident wave within same medium. (TRANSLATION: measure of the intensity of the reflected wave)
SS2	The ratio of the amplitude of the diffusely reflected wave to the amplitude of the incident wave from medium 2 to medium 1. (TRANSLATION: measure of the intensity of the transmitted wave)
Fmu	The Fourier transform of the correlation function in vacuum (TRANSLATION: It is the Fourier transform of the function that defines the vertical component of the roughness of a surface.)
FmuS	The Fourier transform of the correlation function in a medium (TRANSLATION: It is the Fourier transform of the function that defines the vertical component of the roughness of a surface.)
IntIplus	The integrated probability for non-specular reflection (TRANSLATION: The total probability for the incident wave reflecting off the surface in a non-specular fashion, added up over all outgoing angles). This uses the above functions S2 and Fmu.
ProbIplus	The probability for the incident wave to be reflected at a given pair of azimuthal and polar angles. This uses the above functions S2 and Fmu.
IntIminus	The integrated probability for non-specular transmission (TRANSLATION: The total probability for the incident wave to penetrate into the material, added up over all in-going(?) angles). This uses the above functions SS2 and FmuS.
ProbIminus	The probability for the incident wave to be transmitted at a given pair of azimuthal and polar angles within the material. This uses the above functions SS2 and FmuS.

globals.cpp **and** ***globals.h***

The new constants that were added to *globals.h* are,

Constants (all static const)	**Description**
long double neutron_mass_c2 = 939.565378L	mass of the neutron in MeV/c^2
long double hbarc_squared = 3.893796626548067E-20L	reduced Planck's constant multiplied by c and squared
long double eV = 1e-06L	Conversion factor from MeV to eV
long double twopi = pi*2.L	Two pi
int NEdim = 1001	the number of energy increments for the MicroRoughness tables
int Nthetadim = 91	the number of theta increments for the MicroRoughness tables

A function called *MicroRoughnessRotateVector(....)* was added to *globals.cpp (*and the function prototype was added to *globals.h*), which requires a bit of explanation. This function takes in a vector to be rotated, a normal vector around which the rotation is performed, a theta angle of rotation and a phi angle of rotation.

Inputs to function	**Description**
long double v[3]	This is the vector to be rotated
const long double n[3]	The normal vector
long double theta	Theta angle in radians
long double phi	Phi angle in radians

This rotation is necessary as the MicroRoughness model changes the trajectory of the neutron with respect to its incident trajectory. That is, the outgoing angle phi angle is measured with the incident phi angle being defined as zero. The theta angle is measured from the normal; however it is convenient to imagine the MR model as also changing the theta angle with the incident theta angle being defined as zero.

This is done using an extended form of the Rodrigues formula (Rodrigues' Rotation Formula).

The Rodrigues formula describes a method to rotate a vector around an axis of rotation defined by a unit vector. The rotation is performed anti-clockwise, or according to the right hand rule, where the thumb is the unit vector and the curl of the fingers point in the direction of the rotation. In the below picture, the vector **v** is rotated around the z axis by angle ϕ, to make a new vector $\mathbf{v}_{\mathrm{rot}}$.

In mathematical form, $\mathbf{v}_{\mathrm{rot}}$ is given by,

$$\mathrm{v}_{\mathrm{rot}} = \mathrm{v}\cos\varphi + (\mathrm{n} \times \mathrm{v})\sin\varphi + \mathrm{n}(\mathrm{n} \, . \, \mathrm{v})(1 - \cos\varphi)$$

where **n** is the unit vector of rotation. The full derivation can be found in the referenced Wikipedia page.

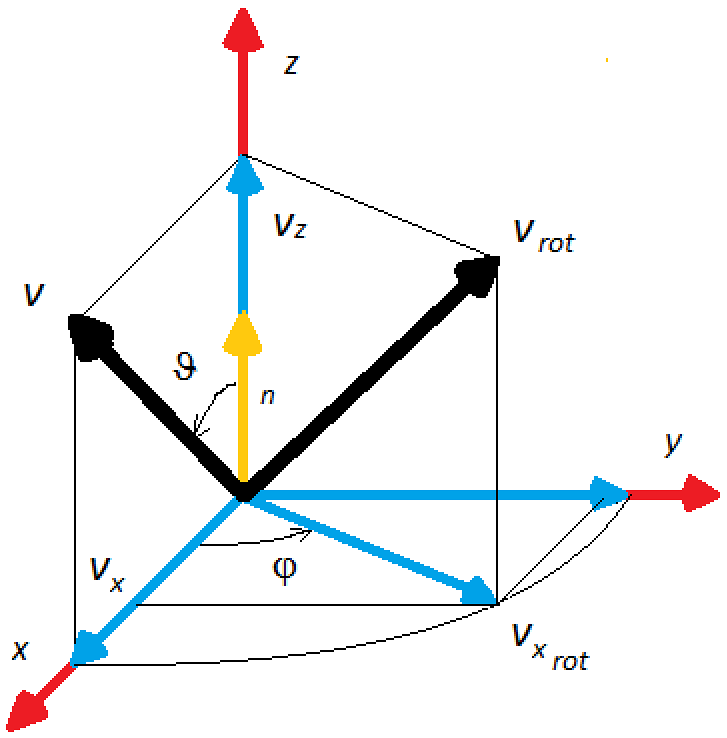


FIGURE 5: RODRIGUES ROTATION OF VECTOR V

However, the rotation required for the MicroRoughnessRotationVector() needs to be done in the θ direction as well as the ϕ direction. This was done by using the Rodrigues Formula again, but around a unit vector perpendicular to both the previous axis of rotation and $\mathbf{v}_{rot}$.

This new unit vector, **w**, is constructed by taking,

$$w = \frac{n \times v_{rot}}{|n \times v_{rot}|}$$

The denominator, $|n \times v_{rot}|$, is just the magnitude of $n \times v_{rot}$, thus making **w** a unit vector.

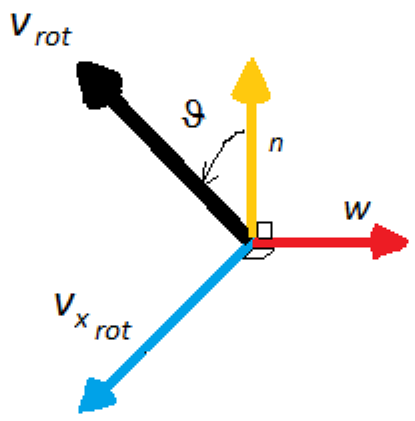


FIGURE 6: UNIT VECTOR FOR THETA ROTATION

Using the right hand rule, I can be seen that the rotation of $\mathbf{v}_{rot}$ around **w,** will produce rotation away from the original unit vector **n**. This rotation is done in the exact same way as outlined in the phi rotation step. Explicitly, this looks like,

$$v_{phi-theta\ rot} = v_{rot}\cos\theta + (w \times v_{rot})\sin\theta + w(w \,.\, v_{rot})(1-\cos\theta)$$

Where $v_{phi-theta\ rot}$ is the final vector of v, which has been rotated in both the phi and theta direction.

IMPORTANT NOTE: It is important to notice that this rotation of θ is the change of θ of the vector v_{rot}. It does not set the vector to the specified θ value, but merely shifts it by θ according the right hand rule.

geometry.h

All the functionality from *G4UCNMaterialPropertiesTable.cc* and *G4UCNMaterialPropertiesTable.hh* were ported over to PENTrack by merging it with *geometry.h* and *geometry.cpp*. The variables added to the *geometry.h* file are given below. Only the variables and functions with the tag, [NoMRT], were added to the PENTrackNoMRTables version.

Type	Variable name	Description
int	Interaction [NoMRT]	The switch for deciding which model of reflection/transmission to use. (0 for Specular reflection/transmission, 1 for Diffuse-specular reflection/transmission, 2 for MicroRoughness-specular reflection/transmission)
double	RMSRough [NoMRT]	The root mean square roughness parameter of the MR model, otherwise called **b**
double	CorrelLength [NoMRT]	The correlation length parameter of the MR model, otherwise called **w**
double*	theMicroRoughnessTable	Pointer to the table of integral probabilities for neutron reflection
double*	maxMicroRoughnessTable	Pointer to the table of maximum probabilities of

		direction of neutron reflection
double*	theMicroRoughnessTransTable	Pointer to the table of integral probabilities for neutron transmission
double*	maxMicroRoughnessTransTable	Pointer to the table of maximum probabilities of direction of neutron transmission
double	theta_i_min	The minimum incident theta angle for computation of the above tables
double	theta_i_max	The maximum theta incident angle for computation of the above tables
double	Emin	The minimum neutron energy for computation of the above tables
double	Emax	The maximum neutron energy for computation of the above tables
int	no_theta_i	The number of divisions of incident theta for which the above tables are computed
int	noE	The number of divisions of neutron energy for which the above tables are computed
double	theta_i_step	The step size between values of incident theta for which the tables are computed (this is calculated by (theta_i_max – theta_i_min)/no_theta_i)
double	E_step	The step size between values of neutron energy for which the tables are computed (this is calculated by (Emax – Emin)/noE)
int	AngNoTheta [NoMRT]	The number of divisions of outgoing theta angles used to calculate the integral probability
int	AngNoPhi [NoMRT]	The number of divisions of outgoing phi angles used to calculate the integral probability
double	AngCut [NoMRT]	As described by Heule, ": for angles which are closer to the specular direction than a certain value (0.01°), the probability is set to 0 in order to avoid a hang-up at the generation of the polar angle due to a very sharp angular distribution" The angular cut sets this 'certain value'

The function prototypes for the functions described in the next section were also added to the *material* struct.

geometry.cpp

The following functions were added to the *material* struct,

Type	Variable name	Description
	material()[NoMRT]	Constructor for the material class
virtual	~material()	Destructor for the material classr
double*	GetMicroRoughnessTable()	Function to return a pointer to the integral probability table

		for reflection
double*	GetMicroRoughnessTransTable()	Function to return a pointer to the integral probability table for transmission
void	LoadMicroRoughnessTables(double*, double*, double*, double*)	Function to load the MicroRoughnessTables into memory if they're already computed
void	InitMicroRoughnessTables()	Function to initialize the tables
void	ComputeMicroRoughnessTables()	Functions that fills the tables with their computed values
double	GetMRIntProbability (double, double)	Parses the MR Reflection table to find the integral probability for the input energy and incident angle
double	GetMRMaxProbability (double, double)	Parses the MR Max Reflection table to find the maximum probability for the input energy and incident angle
double	GetMRProbability (double, double, double, double, double)	Calculates and returns the probability (not integral) for the neutron reflection, given the energy of the neutron, the Fermi potential it encounter, the incident angle, the outgoing theta angle and the outgoing phi angle.
double	GetMRIntTransProbability (double, double)	Parses the MR Transmission table to find the integral probability for the input energy and incident angle
double	GetMRMaxTransProbability (double, double)	Parses the MR Max Transmission table to find the maximum probability for the input energy and incident angle
double	GetMRTransProbability (double, double, double, double, double)	Calculates and returns the probability (not integral) for the neutron transmission, given the energy of the neutron, the Fermi potential it encounter, the incident angle, the outgoing theta angle and the outgoing phi angle.
bool	ConditionsValid (double E, double VFermi, double theta_i) [NoMRT]	Checks if the conditions required for MR reflection are met. These conditions are outlined in Heule's thesis, equations 4.17-4.20
bool	TransConditionsValid (double E,	Checks if the conditions

	double VFermi, double theta_i) [NoMRT]	required for MR transmission are met. These conditions are outlined in Heule's thesis, equations 4.17-4.20

The TGeometry constructor was modified to load the values for the new variables from the geometry.in file:

ss >> mat.FermiReal >> mat.FermiImag >> mat.DiffProb >> mat.SpinflipProb >> mat.RMSRough >> mat.CorrelLength >> mat.Interaction;

And the following lines were added to the PENTrackNoMRTables version, to compute the lookup tables as each material is loaded, if they are specified as undergoing MR Interactions (Interaction == 2),

```
if (mat.Interaction == 2) {

                    cout << "ABOUT TO CALCULATE MR TABLES!!! for: " << mat.name <<
'\n';

                    mat.ComputeMicroRoughnessTables();

                    cout << "CALCULATED MR TABLES!!!
WOOOOOOOOOOOOOOOOOOOO!" << '\n';

        }
```

geometry.in

The general overview covers the gist of it, but it is paraphrased here for completeness:

In the *[MATERIALS]* section, 3 new columns were added to define the material properties,

1. *RMS Roughness [nm]*: indicates the root mean square roughness value, b, for the MRM calculations.
2. *Correlation Length [nm]*: indicates the correlation length, w, for the MRM calculations.
3. *Interaction*: indicates the type of model to be used for a neutron interaction with the material, with '0' indicating specular reflection/transmission, '1' indicating diffuse-specular reflection/transmission, and '2' indicating MicroRoughness-specular reflection/transmission,

neutron.h

The bulk of the changes made to include the MR model were made here, in the OnHit function, or for use within it. None of the other pre-existing functions were modified in this file, but many new functions were created.

The OnHit function

The OnHit function determines what changes are made to the neutron's velocity when it hits a material.

Inputs to function	Description

long double x1	Starting time of the neutron
long double y1[6]	Starting position and speed of the neutron (y1[0] to y1[2] contain x-y-z position, and y1[3] to y1[5] contain x-y-z speed
long double &x2	Ending time of the neutron
long double y2[6]	Ending position and speed of the neutron (y2[0] to y2[2] contain x-y-z position, and y2[3] to y2[5] contain x-y-z speed
int &polarisation	The polarization of the neutron
const long double normal[3]	The unit vector defining the normal to the surface that the neutron hits
solid *leaving	The solid that the neutron is leaving
solid *entering	The solid that the neutron is entering
bool &trajectoryaltered	Boolean indicating if the neutron's trajectory was changed
bool &traversed	Boolean indicating if the neutron crossed the material boundary

The logic of this function proceeds as follows,

If the normal energy of the neutron is larger than the potential step encountered, transmission is attempted. The transmission probability is calculated, and if the random number thrown is less than this probability, transmission is done according to chosen model (Specular/Diffuse-Specular/MicroRoughness-Specular).

If transmission does not occur, absorption is attempted. The reflection probability is calculated, and if the randomly thrown number is **more** than this probability, absorption is done.

If absorption does not occur, reflection is performed according to the chosen model.

Added Functions

1) long double TransmissionProb	
Inputs to function	**Description**
long double Enormal	The energy of the neutron normal to the material surface
long double Estep	The potential step the neutron encounters
solid *entering	The properties of the material the neutron is entering
int I	A number giving the type of model used to model the neutron behaviour

This function calculates the transmission probability for the neutron. If the int I is equal to 2, then the MicroRoughness enhancement is performed on the probability.

2) long double ReflectionProb

Inputs to function	Description
long double Enormal	The energy of the neutron normal to the material surface
long double Estep	The potential step the neutron encounters
solid *entering	The properties of the material the neutron is entering
int I	A number giving the type of model used to model the neutron behaviour

This function calculates the reflection probability for the neutron. If the int I is equal to 2, then the MicroRoughness enhancement is performed on the probability.

3) void DoSnellTransmission	
Inputs to function	**Description**
long double y1[6]	Starting position and speed of the neutron (y1[0] to y1[2] contain x-y-z position, and y1[3] to y1[5] contain x-y-z speed
long double y2[6]	Ending position and speed of the neutron (y2[0] to y2[2] contain x-y-z position, and y2[3] to y2[5] contain x-y-z speed
const long double normal[3]	The unit vector defining the normal to the surface that the neutron hits
long double k1	Wavenumber in first solid
long double k2	Wavenumber in second solid

Performs Snell transmission. The method used to do this is unchanged from the implementation.

4) void DoAbsorption	
Inputs to function	**Description**
long double x1	Starting time of the neutron
long double y1[6]	Starting position and speed of the neutron (y1[0] to y1[2] contain x-y-z position, and y1[3] to y1[5] contain x-y-z speed
long double &x2	Ending time of the neutron
long double y2[6]	Ending position and speed of the neutron (y2[0] to y2[2] contain x-y-z position, and y2[3] to y2[5] contain x-y-z speed
solid *entering	The properties of the material the neutron is entering

Performs absorption. The method used to do this is unchanged from the implementation.

5) void DoSpecularReflection	
Inputs to function	**Description**
const long double normal[3]	The unit vector defining the normal to the surface that the neutron hits
long double x1	Starting time of the neutron
long double y1[6]	Starting position and speed of the neutron (y1[0] to y1[2] contain x-y-z position, and y1[3] to y1[5] contain x-y-z speed

long double &x2	Ending time of the neutron
long double y2[6]	Ending position and speed of the neutron (y2[0] to y2[2] contain x-y-z position, and y2[3] to y2[5] contain x-y-z speed

Performs specular reflection. The method used to do this is unchanged from the implementation.

6) void DoDiffuseReflection	
Inputs to function	**Description**
long double x1	Starting time of the neutron
long double y1[6]	Starting position and speed of the neutron (y1[0] to y1[2] contain x-y-z position, and y1[3] to y1[5] contain x-y-z speed
long double &x2	Ending time of the neutron
long double y2[6]	Ending position and speed of the neutron (y2[0] to y2[2] contain x-y-z position, and y2[3] to y2[5] contain x-y-z speed
const long double normal[3]	The unit vector defining the normal to the surface that the neutron hits

Performs diffuse reflection. The method used to do this is unchanged from the implementation.

7) void DoDiffuseTransmission	
Inputs to function	**Description**
long double x1	Starting time of the neutron
long double y1[6]	Starting position and speed of the neutron (y1[0] to y1[2] contain x-y-z position, and y1[3] to y1[5] contain x-y-z speed
long double &x2	Ending time of the neutron
long double y2[6]	Ending position and speed of the neutron (y2[0] to y2[2] contain x-y-z position, and y2[3] to y2[5] contain x-y-z speed
const long double normal[3]	The unit vector defining the normal to the surface that the neutron hits

Performs diffuse transmission. The method used to do this is unchanged from the implementation.

8) void DoMicroRoughnessReflection	
Inputs to function	**Description**
long double Estep	The potential step that the neutron encounters, ie the difference between the real Fermi potential of the entering material and the leaving material
long double x1	Starting time of the neutron
long double y1[6]	Starting position and speed of the neutron (y1[0] to y1[2] contain x-y-z position, and y1[3] to y1[5] contain x-y-z speed
long double &x2	Ending time of the neutron

long double y2[6]	Ending position and speed of the neutron (y2[0] to y2[2] contain x-y-z position, and y2[3] to y2[5] contain x-y-z speed
const long double normal[3]	The unit vector defining the normal to the surface that the neutron hits
solid *entering	The properties of the material the neutron is entering

Performs MicroRoughness reflection. This is done by using the accept reject method: two random values for phi and theta are thrown, and a random probability is also thrown, between 0 and the ma probability for the distribution (in the NoMRTables case, the max probability is not known so the probability is thrown between 0 and 0.5, which is an overestimate). The actual MR probability for this theta-phi pair is calculated, and if the random probability is less than this value, the neutron is reflected in this direction. The MicroRoughnessRotateVector() in globals.cpp is used for this purpose.

9) void DoMicroRoughnessTransmission	
Inputs to function	**Description**
long double Estep	The potential step that the neutron encounters, ie the difference between the real Fermi potential of the entering material and the leaving material
long double x1	Starting time of the neutron
long double y1[6]	Starting position and speed of the neutron (y1[0] to y2[2] contain x-y-z position, and y1[3] to y2[5] contain x-y-z veolocity
long double &x2	Ending time of the neutron
long double y2[6]	Ending position and velocity of the neutron (y1[0] to y2[2] contain x-y-z position, and y1[3] to y2[5] contain x-y-z speed
const long double normal[3]	The unit vector defining the normal to the surface that the neutron hits
solid *entering	The solid that the neutron is entering

Performs MicroRoughnessTransmission, in the same manner described above. The only difference is that the velocity is scaled according to the material it is transmitted into.

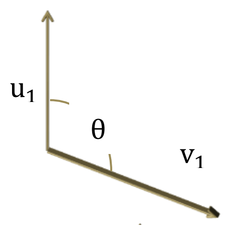
10) long double IncidentTheta	
Inputs to function	**Description**
long double y1[6]	The initial position and speed of the neutron
const long double normal[3]	The normal to the surface

This function changes the neutron's trajectory according to the MR model for transmission. According to the accept-reject method in 2 dimensions, it randomly picks a test values for $-\pi \leq \varphi \leq \pi$, and $0 \leq \theta \leq \pi/2$ according to a uniform distribution. Then, *PENTrackMRTables*: The maximum transmission probability for the neutron incidence angle and energy (calculated using the pre-defined kinetic energy function) is retrieved from the maximum transmission porbability lookup table. A test probability is randomly picked according to a uniform distribution between 0 and 1.5 $\times$the retrieved max probability. The actual probability for the randomly picked φ and θ pair is calculated using the GetMRTransProbability() function, and compared to the test probability. If the test probability is less than the actual probability, then the neutron's velocity trajectory is changed to the φ and θ pair, using the MicroRoughnessRotateVector() function

PENTrackNoMRTables: A probability is randomly picked according to a uniform distribution between 0 and 0.5. The 0.5 value is arbitrary and is possibly too large, but as we cannot know the maximum of the probability distribution, a large value is taken.

The angle of the neutron is not something that is not directly saved/computed as an object property, therefore the incident neutron angle has to be calculated from available information. To calculate this value, the normal vector and the incident velocity vector of the neutron were used. The angle, θ, between 2 arbitrary vectors, $\vec{u}$ and $\vec{v}$, in 3-D space can be written as,

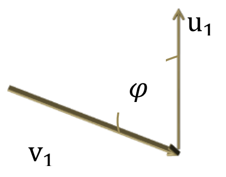
$$\theta = \cos^{-1}\left[\frac{\mathrm{u.v}}{\|\mathrm{u}\|.\|\mathrm{v}\|}\right] = \cos^{-1}\left[\frac{(\mathrm{u}_1.\mathrm{v}_1)+(\mathrm{u}_2.\mathrm{v}_2)+(\mathrm{u}_3.\mathrm{v}_3)}{\sqrt{\mathrm{u}_1{}^2+\mathrm{u}_2{}^2+\mathrm{u}_3{}^2}.\sqrt{\mathrm{v}_1{}^2+\mathrm{v}^2+\mathrm{v}_3{}^2}}\right]$$



Since, in this case, the incident angle would be given by, φ

*W*e can define the incident angle by,

$$\varphi = 180° - \theta$$



The function returns this angleφ.

FUTURE WORK

1) Testing. Lots and lots of testing of the functions and methods used. An ideal geometry for testing the model needs to be designed; the double plate model used by Heule may be a logical starting point to test this. This setup is described in Chapter 4 of the thesis.
2) Check DoDiffuseTransmission function. This was added as an aside, using the logic from the other defined functions, but it should be checked accuracy.

BIBLIOGRAPHY

Heule, S. (2008). *Production, Characterization and Reflectivity Measurements of Diamond-like Carbon and other Ultracold Neutron Guide Materials.* Zurich: University of Zurich.

Rodrigues' Rotation Formula. (n.d.). Retrieved from Wikipedia: http://en.wikipedia.org/wiki/Rodrigues'_rotation_formula

Steyerl, A. (1972). Effect of Surface Roughness on the Total Reflexion and Transmission of Slow Neutrons.

Yapa, P. (2014). *Implementation issues in GEANT4UCN Microroughness model.*