**Phys488 Exercises week 5**

Luke Daniel Jones, Lorna Baker, Zac Humphreys

Department of Physics, the University of Liverpool,

29/02/2017

**Abstract**

For this week’s task we were tasked with finding the average energy loss per distance of a muon through a material and the angles that the muon emerges from the material.

**Task 1**

For this task we were tasked with creating 2 classes called EnergyLoss and MCS.

The first part that will be made is the EnergyLoss class. This task is designed so that for the instantiation the parameters that are typed in are the only variables needed to work out the energy loss. The parameters are the atomic mass and atomic number of the material, and the density of the material. To work out the energy loss the following 2 equations have to be used which are listed equation 1 and 2. Where K is equal to 0.07075 MeV/cm, is equal to 0.511 MeV, I is equal to 0.0000135\*Z MeV, z is equal to 1, and M is equal to 106 MeV.

Using these 2 equations and the parameters which will be given in the instantiation of the class, the only parts that need to be worked out are beta and gamma which are both relativistic variables. The equation listed as equation 3 works out the energy of the muon which is needed for equation 4 which works out beta. While the equation listed as equation 5 uses beta to work out gamma. Where m is the mass of the muon while P is the momentum of the muon.

A class file is made which is very similar to the Histogram.java used in the previous weeks. The class is called EnergyLoss and the contents first states all the variables used and then has a constructor for the class which is where the parameters go into. Inside the constructor contain code specifying the new values of the parameters for that constructor. After this a class method is made called getEnergyLoss(double m) which when the momentum of the muon is inserted it returns the energy loss calculated using the parameters. This class method has all the equations needed to work out the energy loss inside it. The code for the class may be found in the back of the report in the appendix under figure fjfjfjfjfujfujf.

The second part of task 1 now is creating the class MCS which is used to find the angle at which the particles emerges from the material which is called sigma. This class will be done in the same way as the previous task but with different equation and variables. The parameters for the materials in the constructor are the same as before but this time also includes the thickness of the material. The constructor again is used to state the new values for the parameters for the materials specified. There are 3 equations used for sigma are below where equation 6 is to work out the radiation length, equation 7 is to work out theta\_0, and equation 8 is to work out sigma.

There are 3 class methods in the class MCS, the first being a class method which is called X\_0() and contains equation 6 and returns. The next class method is called theta\_0(double particle\_momentum) and includes the equations 4 and 7 and uses the previous class method for, and is used to return. The last class method is called findsigma(double particle\_momentum) and includes equation 8, 4, and the class method for, this method is used to return sigma. The code for this class can be found at the back of the report in the appendix under figure gffdgfagfgfsa.

Now that the 2 classes EnergyLoss and MCS have been made we created a small program which was used to find the energy loss per distance and sigma for various values of momentum for iron. This code first used the instantiation with the parameters of iron with a thickness of 10 cm. This was done for both MCS and EnergyLoss. The code then found both the values using getEnergyLoss() and findSigma() and did this for 6 different momentums. The output for the code can be found in the appendix under figure sfsfeffw.

**Task 2**

This part of the tasks involved the investigation of tracking a muon travelling through an iron sheet, whereby the simulation of such a particle is monitored by counting the ‘hits’ on two detectors placed at 10cm and 20cm beyond the back of the iron. The code ‘TrackMuon’ was downloaded from VITAL and put into BlueJ, whereby several adjustments were made in order to carry out the assigned tasks.

First it was necessary to ensure an understanding of how the program itself operated. By input of the necessary code, the instance classes for Energy Loss and MCS angle (previously created in Task 1) were added to the Trackmuon code, in order to calculate both these aspects through each step of the detector. The energy loss was taken into account by taking away the energy loss at each step by multiplying by the distance it had taken, which in this case is d. While the MSC angle was taken into account by changing theta to theta plus a Gaussian with a mean of 0 and a width of the MSC angle theta\_T. It should be noted that the constructor for the energy loss was put in normally while the constructor for the MSC angle had to be put within the loop because the thickness changed at each step so the thickness should be the variable step.

From the assumption that the muon starts at position co-ordinates (x, y) = (0, 0), i.e. just inside the iron, the user is prompted to type in the initial variables that will be used within the program, including: starting energy, step size, thickness of the iron and the number of muons to track. The positions of the aforementioned counters that detect the muons are also defined in the code.

Muons are tracked through the iron in small steps (whose size is specified by the user), and the energy loss and MCS angle is calculated at each of these steps. As the particle is followed, the (x, y) position co-ordinate pairs are stored in a 2D array (‘trackOfMuon’). This array was developed via use of a ‘for’ loop that laid out the initial double values that were to be measured, inclusive of the start energy, the initial starting position (for both x and y co-ordinates), the number of steps and the angle at which the muon starts out (parallel to the x-axis). A ‘while’ loop within this then stores each step and the scattering angle, as well as also incorporating a ‘nextGaussian’ instruction to return randomly generated values of theta around the mean of the distribution.

The use of an ‘if’ loop was also used to cause a break in the ‘for’ loop should either the energy of the muon fall below the rest mass. The maximum value of steps that the muon was to be tracked for was set to 200. The muons will be tracked at each step and will be followed until the use of a second ‘if’ loop causes another ‘break’ should the number of steps taken by the muon go above this set value nmax. Thus, overall the muon is tracked until either of the two specified ‘if’ loops are enabled, or until the muon leaves the back of the iron sheet. Should the tracking of the muon fall within the allowed parameters and pass through the back of the sheet, the co-ordinate values are then stored in the double array for each point through the detector.

Upon completion of this array, the useful information is then passed to the class method ‘lookAtThisMuon’ in order to be prepared for external analysis. In order to export data from the program, the writeToDisk function was also included to enable data to be exported as .csv file for analysis in Excel.

First it was required that the co-ordinates of 10 sample muon tracks (the (x, y) pairs) were written to a file on disk and plotted as a scatter chart in Excel and can be seen in Figure 1.

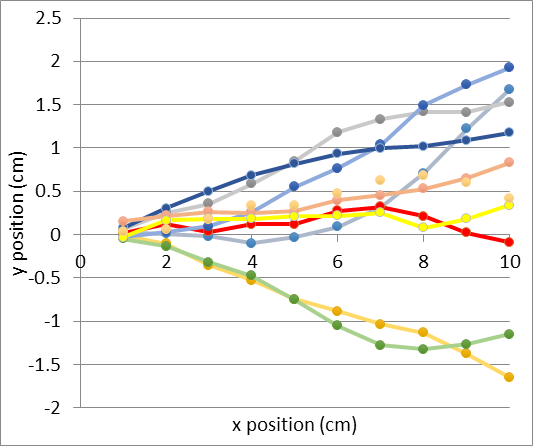


Figure 1 - Scatter graph depicting the x and y co-ordinate pairs of 10 sample muon tracks travelling through the first 10cm of the iron sheet.

From this plot is may be observed that each of the muons follow a similar path through the first 2-3cm of the iron. After a few centimetres however, it may be seen that due to the addition of further MSC angles through each step, there is a greater effect on the y-position of each muon - thus their paths begin to spread further from each other and change track as they travel along the x-axis.

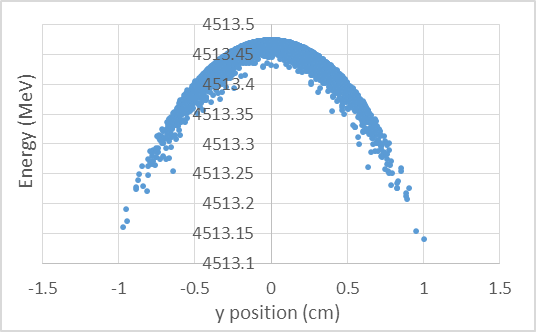


Figure 3 - Graph showing the energy of a muon at it leaves the iron and also the y position that the muon left at. This is for a starting energy of 5,000 MeV.

Further code was also required in order to generate histogram plots of the exit energy and y-positions of the muons as they exit the back of the iron sheet. This was done by first adding 2 array, ylastE and muonenergyXY. The first array ylastE was set so for each loop it will add in the y position as it leaves the iron, while the muonenergyXY does the same for the energy. Both of these are found the same was as it is found in lookAtThisMuon. The next step was to be able to plot it, to do this a new WriteToDisk function was made, this one being called WriteToDiskXY. This new function had the inputs of both arrays, nsteps, and the filename instead of just the filename. The inside of the function was the same expect the only output files were the 2 arrays and position which were in the loop.

The output energy and y position were calculated with a step value of 1 cm and a thickness of 1 cm with 10,000 muons. While the energy was first at 10,000 MeV, then at 5,000 MeV, and then at 500 MeV. This was done for 3 different energies to see the difference between them so comparisons can be made. The first graph which was made using the 10,000 MeV energy can be seen below under figure 2, while figure 3 represents the 5,000 MeV energy graph, and figure 4 represents the 500 MeV energy graph. The 500 MeV graph has 97 muons which have stopped inside the iron, while the rest made it out of the iron.

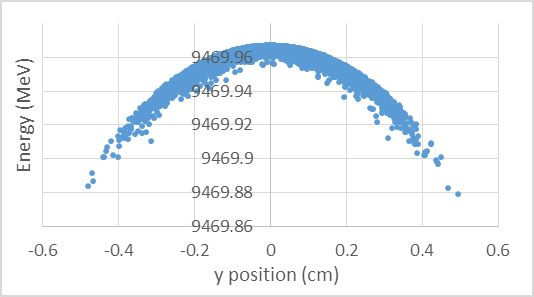


Figure 2 - Graph showing the energy of a muon at it leaves the iron and also the y position that the muon left at. This is for a starting energy of 10,000 MeV.

It can be seen from looking at figure 2, 3, and 4 how different starting energies affect the energy that it leaves the iron at. As the starting energy decreased the range of the y position increased also, this is because there is much more collisions when the muons are going slower. It can also be seen how as the starting energy decreases, the energy gap between highest and the lowest energy muon leaving the iron increases. This energy gap is greatest as the energy is very low and can be seen to be of around 140 MeV. It can also be seen that at the very low starting energy the distribution becomes a lot more wide and scattered, this is due to the amount of random collisions occurring. rfafdgagaggf

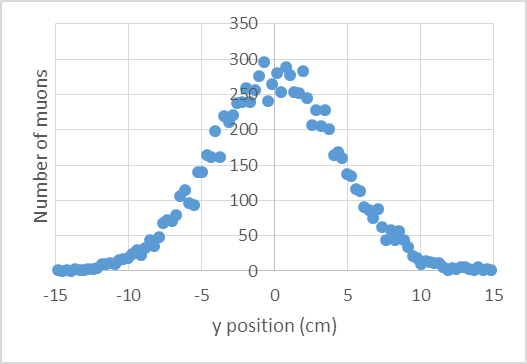


Figure 7 - Graph showing the number of muons vs the y positions for detector 3

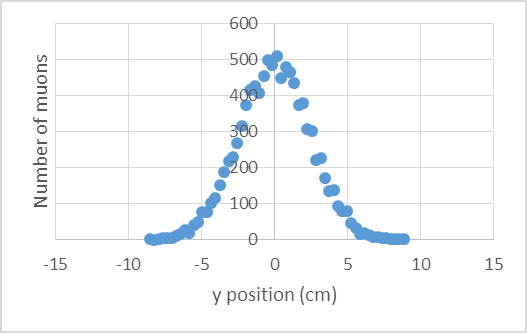


Figure 5 - Graph showing the number of muons vs the y positions for detector 1

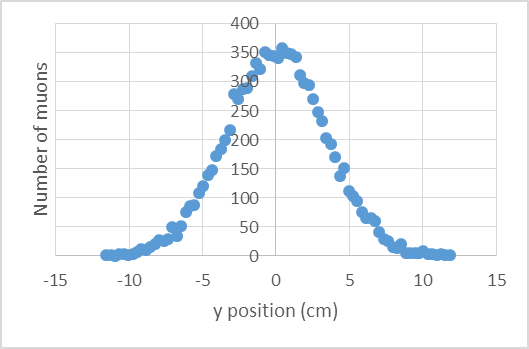


Figure 6 - Graph showing the number of muons vs the y positions for detector 2

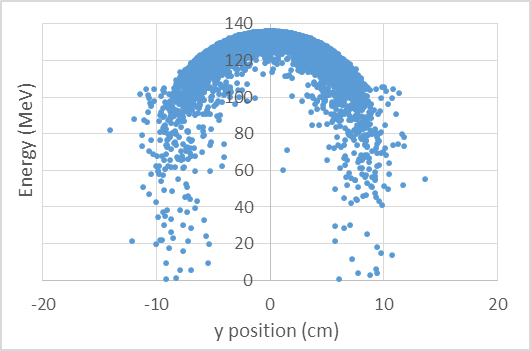


Figure 4 - Graph showing the energy of a muon at it leaves the iron and also the y position that the muon left at. This is for a starting energy of 500 MeV.

The next task asks to add in another detector this time 30 cm away from the iron and to plot histograms for each detector. This was done by adding in another contractor this time called detector3. Then another xc3 to determine the distance the detector is away, then adding another yhitOnC3, then another trackOfMuon, then filling the histogram, and finally writing the histogram to disk. All these steps are very similar to what is done for detector 2 compared to detector 1. Once the third detector was added the program was run with an energy of 1,000 MeV, steps of 1 cm, thickness of 30 cm, and 10,000 muons. The bin high and bin low of all three detector was set to ± 15. Figure 5, 6, and 7 show each detector where figure 5 is for detector 1, figure 6 for detector 2, and figure 7 for detector 3.

It can be seen from looking at the 3 graphs that the further away the detectors the more spread out the muons are, this is what is expected because of the angle of the muons emerging from the iron. This is because when the muons emerge with an angle the further away they go the further from the midpoint they will go.

import java.io.\*;

class MCS

{

// Stating Global variables needed for class

//Constants

static private final double Rydberg\_constant = 13.6; //MeV

static private final double c = 3e8; //ms^{-1}

//Material properties

private double material\_density; //gcm^{-3}

private int material\_Z;

private double material\_A;

private double material\_thickness; //cm

//Particle variables: Taken as constants

private double particle\_charge = 1; //Coulombs

private double particle\_mass = 106; //MeV

//Constructor

public MCS(int in\_Z, double in\_A, double in\_density, double in\_thickness)

{

material\_density = in\_density;

material\_Z = in\_Z;

material\_A = in\_A;

material\_thickness = in\_thickness;

}

//Calculates the material radiation length

public double X\_0()

{

double X\_0 = ((716.4\*material\_A)/(material\_density\*material\_Z\*(material\_Z+1)\*Math.log(287/Math.sqrt(material\_Z))));

return X\_0;

}

//Calculates the theta\_0 angle of scattering

public double theta\_0(double particle\_momentum)

{

double x\_ratio = material\_thickness/X\_0();

//Calculates the velocity as speed of light fractions

double particle\_beta = (particle\_momentum/Math.sqrt((Math.pow(particle\_momentum,2)+Math.pow(particle\_mass,2))));

double theta\_0 = ((Rydberg\_constant\*particle\_charge\*Math.sqrt(x\_ratio))\*(1+0.038\*Math.log(x\_ratio))/(particle\_beta\*particle\_momentum));

return theta\_0;

}

//Calculates the theta\_t angle of scattering

public double findSigma(double particle\_momentum)

{

double theta\_t = (Math.sqrt(2)\*theta\_0(particle\_momentum));

return theta\_t;

}

}

Figure fdzffzdfzdbfbzd - figure showing the code for the MSC function

import java.io.\*;

class EnergyLoss

{

//Universal constant

private final double K = 0.307075; // MeV cm^2

private final double electronmass = 0.511; // MeV

private final double c = 3E8; // m/s

//Material variables

int AtomicNumber; // atomic Z

double AtomicMass; // atomic A

double density; // of material

//Particle variables

double charge = 1; //Coulombs

double mass = 106; // MeV

public EnergyLoss(int Z, double A, double p)

{

AtomicNumber = Z;

AtomicMass = A;

density = p;

}

public double getEnergyLoss(double m)

{

double momentum = m; //MeV

double energy = Math.pow((Math.pow(momentum,2)+Math.pow(mass,2)),0.5); //energy of muon //MeV

double Beta2 = Math.pow((momentum/energy),2); //Velocity as fraction of speed of light squared

double Gamma2 = Math.pow((1/(Math.pow((1-Beta2),0.5))),2); //Lorentz Factor squared

double I = 0.0000135\*AtomicNumber; //Mean excitation energy //MeV

double Wmax = (2\*electronmass\*Beta2\*Gamma2)/(1+((2\*Math.sqrt(Gamma2)\*electronmass)/mass)+Math.pow((electronmass/mass),2));

double energyloss = K\*Math.pow(charge,2)\*density\*(AtomicNumber/AtomicMass)\*(1/Beta2)\*(0.5\*Math.log((2\*electronmass\*Beta2\*Gamma2\*Wmax)/Math.pow(I,2))-Beta2);

return energyloss; // energy lost per distance (MeV/cm

}

}

Figure dssdgdgdgdssgd - figure showing the code for the energy loss function

dE/dx = 81.51653534295967 MeV/cm for p = 30.0 MeV

dE/dx = 11.585269701779907 MeV/cm for p = 300.0 MeV

dE/dx = 15.256713226864003 MeV/cm for p = 3000.0 MeV

dE/dx = 19.813461482173267 MeV/cm for p = 30000.0 MeV

dE/dx = 17.719572021723977 MeV/cm for p = 10000 MeV

dE/dx = 21.96216778207107 MeV/cm for p = 100000 MeV

Sigma = 1.7172427473173746 rads for p = 30.0 MeV

Sigma = 0.049597683799808064 rads for p = 300.0 MeV

Sigma = 0.004679356019119925 rads for p = 3000.0 MeV

Sigma = 4.6764669845279614E-4 rads for p = 30000.0 MeV

Sigma = 0.028682230416898373 rads for p = 500 MeV

Sigma = 0.014107909901408252 rads for p = 1000 MeV

Figure fdzffzdfzdbfbzd - figure showing the output of the code that is used to test the energy loss and MSC functions