The next task, task 2.4 was to estimate the minimum muon energy needed for 50% of the muons to get through 1m of iron without being stopped. To do this the code used for the rest of task 2 was used and edited so that everything that doesn’t concern exitE.fill was deleted. Then the base of the code was put into a for loop which went up to 10,000. The part where the code asked you for the initial values was also replaced so it just stated the values. The values I chose were step size of 10cm, thickness of 100cm, amount of muons of 1,000, and a start energy of 0.2 multiplied by the number of the loop. This causes the code to go up in energy over and over again but we need it to stop when the amount of muons getting through is 50%. To do this I added an if statement which looked at the exitE.fill and when the underflows is equal to the sum of the bins then the program will first print the energy, then export exitE to excel, and most importantly break the loop. When the sum of the bins equal the underflows is when it is exactly 50%. Because of the deleting of unnecessary code this new code takes about 30 seconds to get an answer and got a minimum energy of 466.2 MeV. This is with an uncertainty of about 0.2 MeV. This value was then put into the original code and was correct as the amount of muons getting through was equal to 50%. This uncertainty can be changed very easily by reducing the interval of the start energy and by decreasing the step size.

import java.io.\*;

import java.util.Random;

class Track

{

static BufferedReader keyboard = new BufferedReader (new InputStreamReader(System.in)) ;

static PrintWriter screen = new PrintWriter( System.out, true);

static Random randGen = new Random();

static double startEnergy;

static double stepSize;

static double ironThickness;

static double numberOfMuons;

static final double muonmass = 106.;

static double [][] trackOfMuon;

static int nsteps;

private static void lookAtThisMuon(int nsteps, double [][] track, double finalE)

{

double xlast,ylast;

xlast = track [nsteps-1][0];

ylast = track [nsteps-1][1];

screen.println(" last (x,y) of track = ( " + xlast + " , " + ylast + ")" );

screen.println(" energy of muon as it leaves the material = " + finalE +" MeV");

return;

}

public static void main (String [] args) throws IOException

{

randGen.setSeed(7894694); //sets seed and obtains input data from the user

screen.println("Type in starting energy in MeV ");

double startEnergy = new Double(keyboard.readLine() ).doubleValue();

screen.println("Type in a step size in cm ");

double stepSize = new Double(keyboard.readLine() ).doubleValue();

screen.println("Type in the thickness of iron, cm ");

double ironThickness = new Double(keyboard.readLine()).doubleValue();

screen.println("Type in the number of muons to track ");

int numberOfMuons = new Integer(keyboard.readLine() ).intValue();

EnergyLoss ironEloss = new EnergyLoss(26,55.845,7.87);//sets up code to make use of

final double xc1= ironThickness + 10; // x - coord of first counter after the iron

final double xc2 =ironThickness + 20;

final double xc3 =ironThickness + 30;

final double counterYcoordResolution = 0.1; // sigma of y coord resolution in cm.

final int nmax = 200; // maximum allowed number of steps before we stop following a muon

double [] muonenergyXY = new double [numberOfMuons];

double [] ylastE = new double [numberOfMuons];

Histogram exitE = new Histogram(50, 0, startEnergy); //muon exit from iron

Histogram exitY = new Histogram(50, -1,1);

Histogram detector1 = new Histogram(100,-15,15);

Histogram detector2 = new Histogram(100,-15,15);

Histogram detector3 = new Histogram(100,-15,15);

for (int n =0; n < numberOfMuons; n++)

{

trackOfMuon = new double[nmax+2][2];

double muonEnergy = startEnergy;

double x = 0; // Set the initial starting position of muon

double y = 0; //

nsteps = 0; //counts the number of steps

double theta = 0; // muon starts out parallel to x-axis

screen.println("Start tracking muon " + n + " , energy = " + muonEnergy);

Figure dssdgdgdgdssgd - figure showing the code for task 2.1, 2.2, and 2.3

while (x < ironThickness && nsteps < nmax) {

double step = Math.min(stepSize, ironThickness-x);

MCS ironMS = new MCS(26,55.845,7.87); //MSC for each step

double muonMomentum = Math.sqrt(muonEnergy\*muonEnergy - muonmass\*muonmass);

theta = theta + randGen.nextGaussian()\*ironMS.getThetaT(muonMomentum, step); //new theta after MSC angle

double d = Math.abs(step/Math.cos(theta));

muonEnergy = muonEnergy - ironEloss.getEnergyLoss(muonMomentum)\*Math.abs(d); //new energy after energyloss and abs of d means always positive as d was negative sometimes

if(muonEnergy < muonmass) {

screen.print("Energy of muon below rest mass - got stuck!");

break; // This causes the 'for' loop to terminate.

}

double xnew = x + step ;

double ynew = y + d\*Math.sin(theta);

trackOfMuon[nsteps][0] = xnew;

trackOfMuon[nsteps][1] = ynew;

x = xnew;

y = ynew;

nsteps++;

}

if (nsteps == nmax) {

screen.println("Too many steps for muon " + n + ", abandon it");

} else {

exitE.fill(muonEnergy);

}

double yhitOnC1 = (xc1 - x)\*Math.tan(theta) + y;

yhitOnC1 += randGen.nextGaussian() \* counterYcoordResolution;

double yhitOnC2 = (xc2 - x)\*Math.tan(theta) + y;

yhitOnC2 += randGen.nextGaussian() \* counterYcoordResolution;

double yhitOnC3 = (xc3 - x)\*Math.tan(theta) +y;

yhitOnC3 += randGen.nextGaussian() \* counterYcoordResolution;

trackOfMuon[nsteps+1][0] = xc1;

trackOfMuon[nsteps+1][1] = yhitOnC1;

trackOfMuon[nsteps+2][0] = xc2;

trackOfMuon[nsteps+2][1] = yhitOnC2;

trackOfMuon[nsteps+3][0] = xc3;

trackOfMuon[nsteps+3][0] = yhitOnC3;

detector1.fill(yhitOnC1);

detector2.fill(yhitOnC2);

detector3.fill(yhitOnC3);

lookAtThisMuon(nsteps, trackOfMuon, muonEnergy);

exitY.fill(trackOfMuon [nsteps-1][1]); //fills exity with all the y positions like for each detector

ylastE [n] = trackOfMuon [nsteps-1][1]; //used to fill the final energy for each y position

muonenergyXY[n] = muonEnergy; //used to fill the final y positions for each energy

}

exitE.writeToDisk("muon\_exitE.csv");

exitY.writeToDisk("muon\_exitY.csv");

exitY.writeToDiskXY(muonenergyXY,ylastE,nsteps,"muon\_ENERGY.csv"); //uses new writetodisk thing in histogram used for task 2.2

detector1.writeToDisk("muonDetector1.csv");

detector2.writeToDisk("muonDetector2.csv");

detector3.writeToDisk("muonDetector3.csv");

}

}

Figure dssdgdgdgdssgd - figure showing the code for task 2.1, 2.2, and 2.3

import java.io.\*;

import java.util.Random;

// 2D tracking of a high energy muon through iron with no magnetic field

class task24

{

static BufferedReader keyboard = new BufferedReader (new InputStreamReader(System.in)) ;

static PrintWriter screen = new PrintWriter( System.out, true);

static Random randGen = new Random(); //instantiate the class random

static double startEnergy;

static double stepSize;

static double ironThickness;

static double numberOfMuons;

static final double muonmass = 106.;

static double [][] trackOfMuon;

static int nsteps;

public static void main (String [] args) throws IOException

{

outerloop:

for (int i = 1500; i < 10000; i++) {

randGen.setSeed(7894694); //sets seed and obtains input data from the user

double startEnergy = 0.2\*i;

double stepSize = 10;

double ironThickness = 100;

int numberOfMuons = 1000;

EnergyLoss ironEloss = new EnergyLoss(26,55.845,7.87);

final double xc1= ironThickness + 10; // x - coord of first counter after the iron

final double xc2 =ironThickness + 20;

final double xc3 =ironThickness + 30;

final double counterYcoordResolution = 0.1; // sigma of y coord resolution in cm.

//counts the number of steps

final int nmax = 200; // maximum allowed number of steps before we stop following a muon

double [] muonenergyXY = new double [numberOfMuons];

double [] ylastE = new double [numberOfMuons];

Histogram exitE = new Histogram(50, 0, startEnergy); //muon exit from iron

Histogram exitY = new Histogram(50, -1,1);

Histogram detector1 = new Histogram(100,-15,15);

Histogram detector2 = new Histogram(100,-15,15);

Histogram detector3 = new Histogram(100,-15,15);

for (int n =0; n < numberOfMuons; n++)

{

trackOfMuon = new double[nmax+2][2];

double muonEnergy = startEnergy;

double x = 0; // Set the initial starting position of muon

double y = 0; //

nsteps = 0; //counts the number of steps

double theta = 0; // muon starts out parallel to x-axis

//screen.println("Start tracking muon " + n + " , energy = " + muonEnergy);

while (x < ironThickness && nsteps < nmax) { // Note the 2 conditions here to avoid infinite loop

// Step is the direction in the x-direction

double step = Math.min(stepSize, ironThickness-x);

Figure dssdgdgdgdssgd - figure showing the code for task 2.4

MCS ironMS = new MCS(26,55.845,7.87); //MSC for each step

double muonMomentum = Math.sqrt(muonEnergy\*muonEnergy - muonmass\*muonmass);

theta = theta + randGen.nextGaussian()\*ironMS.getThetaT(muonMomentum, step); //new theta after MSC angle

double d = Math.abs(step/Math.cos(theta));

muonEnergy = muonEnergy - ironEloss.getEnergyLoss(muonMomentum)\*Math.abs(d); //new energy after energyloss and abs of d means always positive as d was negative sometimes

if(muonEnergy < muonmass) {

break; // This causes the 'for' loop to terminate.

}

double xnew = x + step ;

double ynew = y + d\*Math.sin(theta);

trackOfMuon[nsteps][0] = xnew;

trackOfMuon[nsteps][1] = ynew;

x = xnew;

y = ynew;

nsteps++;

}

if (nsteps == nmax) {

screen.println("Too many steps for muon " + n + ", abandon it");

} else {

exitE.fill(muonEnergy);

}

double yhitOnC1 = (xc1 - x)\*Math.tan(theta) + y;

yhitOnC1 += randGen.nextGaussian() \* counterYcoordResolution;

double yhitOnC2 = (xc2 - x)\*Math.tan(theta) + y;

yhitOnC2 += randGen.nextGaussian() \* counterYcoordResolution;

double yhitOnC3 = (xc3 - x)\*Math.tan(theta) +y;

yhitOnC3 += randGen.nextGaussian() \* counterYcoordResolution;

// Add these coords into the array

trackOfMuon[nsteps+1][0] = xc1;

trackOfMuon[nsteps+1][1] = yhitOnC1;

trackOfMuon[nsteps+2][0] = xc2;

trackOfMuon[nsteps+2][1] = yhitOnC2;

trackOfMuon[nsteps+3][0] = xc3;

trackOfMuon[nsteps+3][0] = yhitOnC3;

detector1.fill(yhitOnC1);

detector2.fill(yhitOnC2);

detector3.fill(yhitOnC3);

exitY.fill(trackOfMuon [nsteps-1][1]); //fills exity with all the y positions like for each detector

ylastE [n] = trackOfMuon [nsteps-1][1]; //used to fill the final energy for each y position

muonenergyXY[n] = muonEnergy; //used to fill the final y positions for each energy

// Now generate the next muon

}

int underflow = exitE.getunder();

int sumbins = exitE.getIntegral();

if (underflow == sumbins){

screen.println("The start energy is " + startEnergy);

screen.println("undeflows = " + underflow);

screen.println("sum of bins = " + sumbins);

exitE.writeToDisk("muon\_exitE.csv");

break outerloop;

}

screen.println(startEnergy);

}

}

}

Figure dssdgdgdgdssgd - figure showing the code for task 2.4