

```

#include <iostream>
#include <random>
#include <chrono>
#include "math.h"
#include <iomanip>      // std::setprecision

double PDF(double E, int state, double w);
double Gauss(double x, double w);

bool ok;
double MRP; //MRTOF mass resolving power
double s_alpha; //Alpha-tof energy resolution at sigma in keV
double E_x; //Excitation energy in keV
int MaxCounts = 400*1024;

int main(int argc, char* argv[]){
if(argc == 3){
    MRP = atof(argv[1])*1e3;
    s_alpha = atof(argv[2]);
    E_x = 250;
    ok = true;
}
else if(argc == 4){
    MRP = atof(argv[1])*1e3;
    s_alpha = atof(argv[2]);
    E_x = atof(argv[3]);
    ok = true;
}
else{
    ok = false;
    std::cout << "Usage: SO <MPR x1000> <s_alpha in keV> [E_x]\n";
}
if(ok){
    std::cout << "Events Confidence STDEV E_x\n";
    for(int Events = 7; Events< 500; Events+=1){ //Scan over the number of correlated events
        double Energy[MaxCounts];
        double Mass[MaxCounts];
        double M1[MaxCounts/Events], M2[MaxCounts/Events], M1_ave, M2_ave;
        double Diff[MaxCounts/Events], Diff_ave;
        double stdevDiff;
        double calcE_x = 0;
        int winners = 0;
        int total = 0;
        std::random_device rd; //Will be used to obtain a seed for the random number engine
        std::mt19937 gen(rd()); //Standard mersenne twister engine seeded with rd()
        std::uniform_real_distribution<> dis(0.0, 1.0);
        for (int n = 0; n < MaxCounts; ++n) { //Assign a mass and alpha-decay energy to each count
            // Use dis to transform the random unsigned int generated by gen into a double in [1, 2). Each call to dis(gen) generates a new random double
            double U1 = dis(gen);
            double U2 = dis(gen);
            double E = sqrt(-2.0*log(U1))*cos(2*3.1415*U2)*s_alpha; //Box-Muller
            bool isIsomer = false;
            double WhichAlpha = dis(gen);
            if(WhichAlpha < 0.02) E+=8856;
            else if(WhichAlpha < 0.37) E+=8965;
            else if(WhichAlpha < 0.61) E+=9066;
            else if(WhichAlpha < 1) {
                isIsomer = true; // arbitrary name -- result is agnostic of which state is assigned to the isomer
                E+=9155;
            }
            U1 = dis(gen);
            U2 = dis(gen);
            double moq = sqrt(-2.0*log(U1))*cos(2*3.1415*U2)*(257*931494/(MRP*2.355)); //Box-Muller
            if(isIsomer) moq += E_x; // isomer energy
            // std::cout << n << " " << isIsomer << " " << std::setprecision(4) << E << " " << std::setprecision(9) << moq << '\n';
            Energy[n]=E;
            Mass[n]=moq;
        }
        for(int i=0;i<MaxCounts;i+=Events){
            /*
            For each ensemble of "Events" counts
            calculate the mass using the PDF of each state's alphas
            determine state order from the masses
            record whether or not the correct answer was found
            do this MaxCounts/Events times
            */
            int weightPower = 1; //Can compare result of using weight or weight^2
            double sumWeight = 0;
            double sum = 0;
            double stateMass[2];
            //Calculate PDF-weighted mass of ensemble for one state's alpha's
            for(int j=0;j<Events;j++){ if(i+j<MaxCounts){
                double weight = pow(PDF(Energy[i+j], 0, 2*s_alpha), weightPower);
                sumWeight += weight;
                sum += Mass[i+j]*weight;
            }
            stateMass[0] = sum/sumWeight;

            //Calculate PDF-weighted mass of ensemble for the other state's alpha's
            sumWeight = 0;
            sum = 0;
            for(int j=0;j<Events;j++){ if(i+j<MaxCounts){
                double weight = pow(PDF(Energy[i+j], 1, 2*s_alpha), weightPower);
                sumWeight += weight;
                sum += Mass[i+j]*weight;
            }
            stateMass[1] = sum/sumWeight;

            if(total < (int)(MaxCounts/Events)){
                M1[total] = stateMass[0];
                M2[total] = stateMass[1];
            }
            total++;
            if(stateMass[1] - stateMass[0] > 0) winners++; //Did we get the right answer?
            calcE_x += stateMass[1] - stateMass[0];
        }
        std::cout << Events << " " << 100*double(winners)/double(total) << " " << 100*sqrt(double(winners))/double(total) << " " << calcE_x/(double(total)) << "\n";
    }
}
return 0;

```

```
    }  
    else return -1;  
}  
  
double PDF(double E, int state, double w){  
    double val = 0;  
    if(1==state) val = 0.39*Gauss(E-9155, w);  
    if(0==state) val = 0.02*Gauss(E-8856, w)+0.35*Gauss(E-8965, w)+0.24*Gauss(E-9066, w);  
  
    return val;  
}  
  
double Gauss(double x, double w){  
    return exp(-2*x*x/(w*w));  
}
```