Main code

```
% This program uses the mean free paths of water, graphite, and lead to
% calculate the percentage of neutrons  absorbed, reflected, and transmitted
through a
% given thickeness of a medium.
%
% This program is split into parts, defined as follows
% 1 - Calculating constants:
% Using given microscopic cross sections and densities, and accurate values
% for molecular masses and Avogradro's constant, find macroscopic cross
% sections and the mean free path for each material from the total
% macroscopic cross section
%%%%%%%%%%%%%%%%%
% 2 - Exponential steps
% Investigate how an exponential distribution can be made by taking the
% natural logarithm of uniformly distritubted random numbers, and find how
% an input mean free path compares to an output mean free path calculated
% from this created distribution
%%%%%%%%%%%%%%%%%
% 3 - Uniform spherical distribution
% Use random numbers to generate random points on a sphere's surface.
% Produce a plot of points  to show that there is no bunching at the poles,
% which was observed when translating two random polar coordinates into
% three random cartesian coordinates.
%%%%%%%%%%%%%%%%%
% 4 - Vaccum random walk
% Use two functions (isotropic direction, exponentially distributed step)
% which behave as simplified versions of sections
% 2 and 3 above(included after this program in the appendix) to generate
% a random walk for a particle in a vacuum
%%%%%%%%%%%%%%%%%
% 5 - Variation of transmitted neutrons with thickness
% Use a loop to vary thicknesses, a loop to calculate how many neutrons
% reflect, absorb, and transmit for a given thickness of material, and
% another loop to repeat these calculations to find how resulting values
% deviate
% This is done separately for water, graphite, and lead, and by using
% functions to avoid repeating code unnecessarily the code omits the nouns
% water, graphite, and lead, so the figures produced must be manually
% edited, and the output screen considered in the above order (w,g,l)
%%%%%%%%%%%%%%%%%

clear all; close all;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% 1 - Constants - Start

Avo = 6.0221413e+23; % Avogadro's constant

% Define various quantities for 1) water, 2) graphite, 3) lead
% water:
w_abs = 0.6652; % cross section in barns
w_sca = 103.0; % cross section in barns
w_rho = 1; % density in grams per cm^3
w_Mr = 18.01528; % molecular mass in grams per mole
w_p_absorbed = w_abs/(w_abs+w_sca);
% graphite:
g_abs = 0.0045;
g_sca = 4.74;
g_rho = 1.67;
g_Mr = 12.0107;
g_p_absorbed = g_abs/(g_abs+g_sca);
% lead:
```

```
l_abs = 0.158;
l_sca = 11.221;
l_rho = 11.35;
l_Mr = 207.2;
l_p_absorbed = l_abs/(l_abs+l_sca);

% get number densities - needed for macroscopic cross-sections
w_n_density = Avo*w_rho*(10^6)/w_Mr; % shift to grams and metres
g_n_density = Avo*g_rho*(10^6)/g_Mr;
l_n_density = Avo*l_rho*(10^6)/l_Mr;

% convert microscopic x-sections to macroscopic x-sections
% find the total, and take the inverse ( = mean free path)
w_abs_x_section = w_n_density*w_abs*(10^-28);
w_sca_x_section = w_n_density*w_sca*(10^-28);
w_total_x_section = w_abs_x_section + w_sca_x_section;
w_lambda = 100/w_total_x_section; % shift back to cgs units

g_abs_x_section = g_n_density*g_abs*(10^-28);
g_sca_x_section = g_n_density*g_sca*(10^-28);
g_total_x_section = g_abs_x_section + g_sca_x_section;
g_lambda = 100/g_total_x_section; % shift back to cgs units

l_abs_x_section = l_n_density*l_abs*(10^-28);
l_sca_x_section = l_n_density*l_sca*(10^-28);
l_total_x_section = l_abs_x_section + l_sca_x_section;
l_lambda = 100/l_total_x_section; % shift back to cgs units

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% 1: Constants - End

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% 2: Exponential steps - Start

% define how many points to generate, how many bins to put them in, and the
% mean free path of neutrons in water due to absorption only
nrand = 1000000;
nbins = 100;
mfp = 100/w_abs_x_section;
rand_i = rand(nrand,1); % generate "nrand" number of points between 0 and 1

% show the uniformity of these points
hist(rand_i,nbins); % split all of these points into 100 bins, and make a histogram
titlestr = sprintf('A histogram of uniformly generated random points.\nBins: %g.
Generated random points: %g.',nbins,nrand);
title(titlestr);xlabel('Value of random points (in 100 bins)'); ylabel('Number of
points in a bin'); pause;

rand_f = -mfp*log(rand_i); % use inverse cumulative distribution
% and mean free path to model exponential decay

figure; hist(rand_f,nbins); % split all of these points into 100 bins, and make a
histogram
titlestr = sprintf('A histogram of the previous figure''s points converted to
exponentially decaying form.\nBins: %g. Generated random points: %g. Mean free
path: %g cm.',nbins,nrand,mfp);
title(titlestr);xlabel('Position / cm'); ylabel('Number of points in a bin');
pause;
% the form is exponential: exp(-x/lambda)

% get positions of middles of bins, and bin heights
[bincount,binposition] = hist(rand_f,100);

% ignore zeros (their logs skew things)
binposition = binposition(bincount~=0);
bincount = bincount(bincount~=0);
```

```
% plot these as points - will still be exponential
plot(binposition,bincount,'b.','markersize',16);
titlestr = sprintf('A conversion from the previous histogram by extracting bin
heights and centres.\nBins with zero elements in have been neglected due to their
logs being infinite');
title(titlestr);xlabel('Position / cm'); ylabel('Number of points that were in a
bin'); pause;

% find the gradient and y-intercept of this curve
[P,S] = polyfit(binposition,log(bincount),1);
figure; plot(binposition,log(bincount),'g.','markersize',14);  hold
bestfitx = linspace(0,500,500);
bestfity = P(2)+bestfitx*P(1); % generate points on this straight line
plot(bestfitx,bestfity,'r'); % plot them
titlestr = sprintf('The logarithm of the previous graph has been taking,
linearising it.\n A line of best fit for all points has been ploted.');
title(titlestr);xlabel('Position / cm'); ylabel('Natural log of number of points
that were in a bin'); pause;

% tell the user how the output mean free path various from the input one
fprintf('Mean free path was %g, and according to this linearised plot it is
%g\n',mfp,-1/P(1));
fprintf('The increase is due to the removal of bins where zero neutrons reached -
making it seem like they moved further, on average');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% 2: Exponential steps - End

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% 3: Spherical distribution - Start

nrand = 10000; % too many points makes the plot hard to observe

u = -1 +(2*rand(nrand,1)); % randomly generate points - between -1 and 1
theta = 2*pi*rand(nrand,1); % - between 0 and 2 pi

n = 1; % prepare a loop to convert these random positions to coordinates
x=zeros*size(nrand); y=zeros*size(nrand); z=zeros*size(nrand);
% preallocate these arrays
while (n<=nrand);
x(n) = cos(theta(n))*sqrt(1-u(n)^2);
y(n) = sin(theta(n))*sqrt(1-u(n)^2);
z(n) = u(n);
n = n + 1 ;
end

plot3(x,y,z,'b.'); % plot these points to show uniformity
daspect([1 1 1]); % make the aspect ratio 1:1
title('A figure to show the uniform distribution of points on a sphere''s
surface');
xlabel('x');ylabel('y');zlabel('z');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% 3: Spherical distribution - End

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% 4: Vacuum random walk - Start

lambda = 1; % simplify the resulting figure
x = 0; y = 0; z = 0; % initial position
store_pos = [x,y,z];
plot3(x,y,z); hold;

N_steps = 100000; % too many steps makes the figure less clear
store_dx = 0; store_dy = 0; store_dz = 0; % to measure the average over N steps
store_pathlength = 1; % to note how this differs from lambda

n = 2;
while n<=N_steps
```

```
pathlength = pathlengthf(lambda);
stepvector = spp(pathlength);
dx = stepvector(1); dy = stepvector(2); dz = stepvector(3);
x = x+dx; y = y+dy; z = z+dz;
store_pos = [store_pos;x,y,z];
store_dx = [store_dx;dx]; store_dy = [store_dy;dy]; store_dz = [store_dz;dz];
store_pathlength = [store_pathlength;pathlength];
n = n + 1;
end

allx = store_pos(:,[1,1]); % take all rows from 1st, 2nd, and 3rd colums
ally = store_pos(:,[2,2]);
allz = store_pos(:,[3,3]);
plot3(allx,ally,allz,'b.-'); % plot the points and their connections in blue
plot3(0,0,0,'go','markersize',20); % show the start and end points
plot3(allx(n-1),ally(n-1),allz(n-1),'ro','markersize',20);

% see how these values compare with expected - dx/y/z should be near 0 for
% large N, and the mean pathlength should be a bit larger than lambda
fprintf('mean dx = %g\n',mean(store_dx));
fprintf('mean dy = %g\n',mean(store_dy));
fprintf('mean dz = %g\n',mean(store_dz));
fprintf('mean pathlength = %g\n',mean(store_pathlength));
titlestr = sprintf('A figure showing a random walk of a particle using random
directions and exponentially distributed steps. %g Steps taken.',N_steps);
title(titlestr);
xlabel('x position /cm'); ylabel('y position /cm'); zlabel('z position /cm');
pause;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% 4: Vacuum random walk - End

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% 5: RAT vs thickness - Start
% Note that many things, such as the number of neutrons and loops can be
% changed, and the thicknesses to do calculations over can be changed too.
% An accurate value at 10cm can be obtained separately to finding the
% characteristic attenuation length over a range of thicknesses.

materialtype = 0;
while materialtype ~=3;

% reset values
Thickness = 0; % start finding how RAT varies with thickness at this initial
thickness
Thickness_increment = 0.25; % how muuch to increase the thickness by
T_f = 50; % stop doing calculations at this thickness (for graphite and lead)
num_loops = 10; % do N neutrons for num_loops loops, to get mean and s.d.
N_neutrons = 10000;

    if materialtype == 0
       fprintf('The following figures and values are specific to water!\n');
       sca = w_sca; abs = w_abs;
       lambda = w_lambda;
       T_f = 12; % end at 12cm for water
    end

    if materialtype == 1
       fprintf('The following figures and values are specific to graphite!\n');
       sca = g_sca; abs = g_abs;
       lambda = g_lambda;
    end

      if materialtype == 2
       fprintf('The following figures and values are specific to lead!\n');
       sca = l_sca; abs = l_abs;
       lambda = l_lambda;
      end
```

```
p_absorbed = abs/(abs+sca);

% Input values into a function which finds the percentage of neutrons
% reflected, absorbed or transmitted
RAT_VaryT =
f_RAT_vs_Thickness(Thickness,Thickness_increment,T_f,lambda,N_neutrons,p_absorbed,n
um_loops);

% RAT_VaryT = {Thickness_store,R_varying_Th,A_varying_Th,T_varying_Th}
Thickness_store = RAT_VaryT{1}; % access cells with {}, as shown
R_varying_Th = RAT_VaryT{2};
A_varying_Th = RAT_VaryT{3};
T_varying_Th = RAT_VaryT{4};
% pass these to a function which creates figures and finds the
% characteristic attenuation length and its error
attenuation_length  =
f_RAT_analyse(R_varying_Th,A_varying_Th,T_varying_Th,Thickness_store,N_neutrons);
fprintf('The characteristic attenuation length is %g +- %g
cm',attenuation_length(1),attenuation_length(2));

materialtype = materialtype + 1; % move from water to graphite to lead
end
```

---

pathlengthf.m

```
% generate a random point on a graph of exponential decay to give step length
% pathlength.m

function pathlength = pathlengthf(lambda)

randompoint = rand(1,1);
pathlength = -lambda*log(randompoint);

end
```

---

spp.m

```
% spherical point picker (spp)
% generate a random point of a sphere's surface
% with a method suggested on http://mathworld.wolfram.com/SpherePointPicking.html

function stepvector = spp(pathlength)

u = -1 +(2*rand(1,1)); % randomly generate points - between -1 and 1
theta = 2*pi*rand(1,1); % - between 0 and 2 pi

x = cos(theta)*sqrt(1-u^2);
y = sin(theta)*sqrt(1-u^2);
z = u;

stepvector = pathlength*[x,y,z];

end
```

---

f_RAT_vs_Thickness

```
function RAT_VaryT =
f_RAT_vs_Thickness(Thickness,Thickness_increment,T_f,lambda,N_neutrons,p_absorbed,n
um_loops)
```

```
% initialise storage arrays
R_varying_Th = [101,101]; % percentages won't go above 100
A_varying_Th = [101,101]; % it's better to initialise with this value
T_varying_Th = [101,101]; % than to do so with 0!
Thickness_store = 0; % track thicknesses - for plotting
Thickness_flag=0;
while (Thickness <= T_f) % loop over various lengths

    RATstore = [101,101,101]; % create an array to store data about neutron
"deaths"
    % percents must be less than 100 - this initialising row will be removed

    loop_for_mean_sd = 0; % initialise a counter for the following loop
    while (loop_for_mean_sd<=num_loops-1) % run the process for N neutrons 100
times
        fprintf('Loop number %g.\n',loop_for_mean_sd+1); % check calculations are
going
        RAT = sca_or_abs_final(lambda,p_absorbed,Thickness,N_neutrons);
        RATstore = [RATstore;RAT(1),RAT(2),RAT(3)];
        loop_for_mean_sd = loop_for_mean_sd + 1;
    end

    % split this into R, A, and T
    Rstore = RATstore(2:end,1);
    Astore = RATstore(2:end,2);
    Tstore = RATstore(2:end,3);

    % put the means and standard deviations of these in storage array
    % each row has with different thickness of material
    R_varying_Th = [R_varying_Th;mean(Rstore),std(Rstore)];
    A_varying_Th = [A_varying_Th;mean(Astore),std(Astore)];
    T_varying_Th = [T_varying_Th;mean(Tstore),std(Tstore)];

    if (Thickness == 10)
        Thickness_flag=1;
        fprintf('Percentage neutrons reflected in 10cm: %g +-
%g.\n',mean(Rstore),std(Rstore));
        fprintf('Percentage neutrons absorbed in 10cm: %g +-
%g.\n',mean(Astore),std(Astore));
        fprintf('Percentage neutrons transmitted in 10cm: %g +-
%g.\n',mean(Tstore),std(Tstore));
    end
%     fprintf('Thickness = %g cm\n',Thickness);  % useful to track progress
    Thickness = Thickness + Thickness_increment; % increase thickness for next loop
    Thickness_store = [Thickness_store;Thickness]; % store thickness for graphing

    % output of function - thicknesses and RATvsT
    RAT_VaryT = {Thickness_store,R_varying_Th,A_varying_Th,T_varying_Th};
end

end
```

---

f_RAT_analyse

```
function attenuation_length  =
f_RAT_analyse(R_varying_Th,A_varying_Th,T_varying_Th,Thickness_store,N_neutrons)

% remove the initialising values
R_varying_Th=R_varying_Th(2:end, :);
A_varying_Th=A_varying_Th(2:end, :);
```

```
T_varying_Th=T_varying_Th(2:end, :);
Thickness_store = Thickness_store(1:(end-1),1);

% show how R/A/T vary with thickness
errorbar(Thickness_store,R_varying_Th(1:end,1),R_varying_Th(1:end,2),'rx'); hold;
errorbar(Thickness_store,A_varying_Th(1:end,1),A_varying_Th(1:end,2),'bx');
errorbar(Thickness_store,T_varying_Th(1:end,1),T_varying_Th(1:end,2),'gx');
legend('Percent reflected','Percent absorbed','Percent transmitted');
title('A graph to show how the number of neutron reflected, absorbed, and
transmitted varies with thickness.');
xlabel('Thickness /cm');
ylabel('Percent reflected, absorbed, or transmitted'); pause;

% find the characteristic attenuation lengths
% convert percentages to numbers of neutrons transmitted, and take the log
% dealing with natural logs, so
T_varying_Th_E_ln = (T_varying_Th(1:end,2)/T_varying_Th(1:end,1));
T_varying_Th_E_ln = T_varying_Th_E_ln(1:end,1);
T_varying_Th_Y_ln = log(T_varying_Th(1:end,1)*N_neutrons/100); %replace with
#neutrons final
errorbar(Thickness_store,T_varying_Th_Y_ln,T_varying_Th_E_ln,'bx');
title('The natural log of the number of transmitted neutrons against thickness
produced the plot shown below.');
xlabel('Thickness /cm');
ylabel('Natural log of the number of neutrons transmitted'); pause;

% isolate the linear part
% the user must define the indecies of the linear region, in place of "41"
% and "end" - this can be done by eye with the previous figure
Thickness_store = Thickness_store(41:end,1);
T_varying_Th_Y_ln = T_varying_Th_Y_ln(41:end,1);
T_varying_Th_E_ln = T_varying_Th_E_ln(41:end,1);
% plot the linear part
errorbar(Thickness_store,T_varying_Th_Y_ln,T_varying_Th_E_ln,'bx'); hold;
title('The linear part of the previous figure has been isolated, and the line best
fit drawn.');
xlabel('Thickness /cm');
ylabel('Natural log of the number of neutrons transmitted');
[P,S] = polyfit(Thickness_store,T_varying_Th_Y_ln,1);
fitted_y = P(2)+Thickness_store*P(1);
plot(Thickness_store,fitted_y); pause;

% find the error on this best fit gradient
covm = mean(T_varying_Th_E_ln)*inv(S.R)*inv(S.R)';
error_m = sqrt(covm(1,1));

error_attentuationL = P(1)*(error_m/P(1));
attenuation_length = [-1/P(1),error_attentuationL];
Chi2 = (S.normr/mean(T_varying_Th_E_ln))^2;

fprintf('The characteristic attentuation length is %g.\n',-1/P(1));
fprintf('Error in attenuation length: %g\n',error_attentuationL);
fprintf('Reduced Chi squared: %g\n)',Chi2/size(Thickness_store,1));
figure;plot(1,1,'go'); title('This material has been analysed. Press enter to
continue.'); pause;

end
```