Analysis of neutron penetration through shielding using Monte Carlo techniques

Meirin Evans

9214122

School of Physics and Astronomy

The University of Manchester

Second Year Computing Report

May 2016

Abstract

Through simulation, the neutron transmission, reflection and absorption properties of water, lead and graphite were compared. For 10 cm slabs it was found that transmission, reflection and absorption values respectively were; 0.33 ± 0.05 %, 80.02 ± 0.08 % and 19.66 ± 0.37 % for water; 28.37 ± 0.46 %, 62.80 ± 0.42 % and 8.83 ± 0.29 % for lead and 30.77 ± 0.45 %, 68.49 ± 0.47 % and 0.74 ± 0.08 % for graphite. 10 cm slabs of different materials in contact were also simulated using the Woodcock method.

1. Introduction

Vast numbers of neutrons are produced in nuclear reactors. Fission reactions, such as those currently done with uranium-235, produce neutrons in a chain reaction [1]. To prevent a runaway reaction (leading to a bomb) the number of neutrons in the reactor needs controlling. The number of neutrons leaving the reactor also needs controlling, as too many neutrons leaving the reactor could endanger someone working nearby [2]. To do this, neutron reflectors (materials which tend to scatter neutrons backwards) and absorbers are used as shielding to nuclear reactors. Examples of such materials are water, lead and graphite. Monte Carlo [3] is used in this project since neutron transport is inherently random. A distribution of results is produced, then averages are taken.

2. Theory

Thermal neutrons are in thermal equilibrium with the ambient (room) temperature, *T*, and have kinetic energy, *E*, given by Equation 1,

$$E = kT \quad (1)$$

where *k* is Boltzmann's constant. Any neutrons that undergo a sufficient number of scatters eventually thermalise. Different materials present a different cross section to thermal neutrons for scattering and absorption. Given a cross section, σ, a mean free path, λ, can be calculated through Equation 2,

$$\lambda = \frac{mu}{\rho\sigma} \quad (2)$$

where *m* is the material's molecular weight, *u* is the atomic mass unit and ρ is the material density. The total mean free path, λ_T, is given by Equation 3,

$$\frac{1}{\lambda_T} = \frac{1}{\lambda_S} + \frac{1}{\lambda_A} \quad (3)$$

where λ_S is the scattering mean free path and λ_A is the absorption mean free path. The total mean free path represents the 1/e distance travelled by a neutron before absorption or scattering. On average, the ratio of scatters to absorptions is given by the ratio of the scattering to absorption cross sections.

It is possible to model neutrons travelling through a slab of two different materials using the Woodcock method [4]. This works by taking steps according to the material with the lower mean free path and introducing fictitious steps when in the other material. Fictitious steps do nothing (neutrons do not scatter or absorb).

3. Analytical method

In a **MATLAB** [5] script 10000 neutrons were simulated to be incident upon 10 cm thick semi-infinite (in the other directions) slabs of water, lead and graphite separately. Each neutron was tracked until it was absorbed, or had left the slab. The number of neutrons with finishing positions before the start of the slab, after the end of the slab and inside the slab were counted. These corresponded to reflected, transmitted and absorbed neutrons respectively. 50 runs were completed to calculate the mean and standard deviation for each material. The data used are shown in Table 1.

Material	Water	Lead	Graphite
Molecular weight *m* (atomic mass units)	18	207	12
Absorption cross section σ_A (barn)	0.6652	0.158	0.0045
Scattering cross section σ_S (barn)	103.0	11.221	4.74
Density (g cm^{-3})	1.00	11.35	1.67

Table 1. Data for molecular weight, absorption cross section, scattering cross section and density for each of water, lead and graphite. A barn is 10^{-28} m^2.

Runs were repeated for slabs of different thickness to observe how reflection, transmission and absorption varied with thickness for each material. From this an attenuation length was calculated by taking the negative inverse of the gradient of a log graph. Thus a slab thickness needed to stop a given number of neutrons was calculated.

To test the Woodcock method each possible combination (order dependent) of water, lead and graphite 10 cm slabs was simulated.

4. Results

4.1. Absorption only

Firstly, neutrons were simulated for absorption only. The results are shown in Figure 1, which shows that each material displays exponential behaviour but with different mean free paths. The errors on the heights of each bin are given by the square root of the number of neutrons in that bin from Poisson statistics.

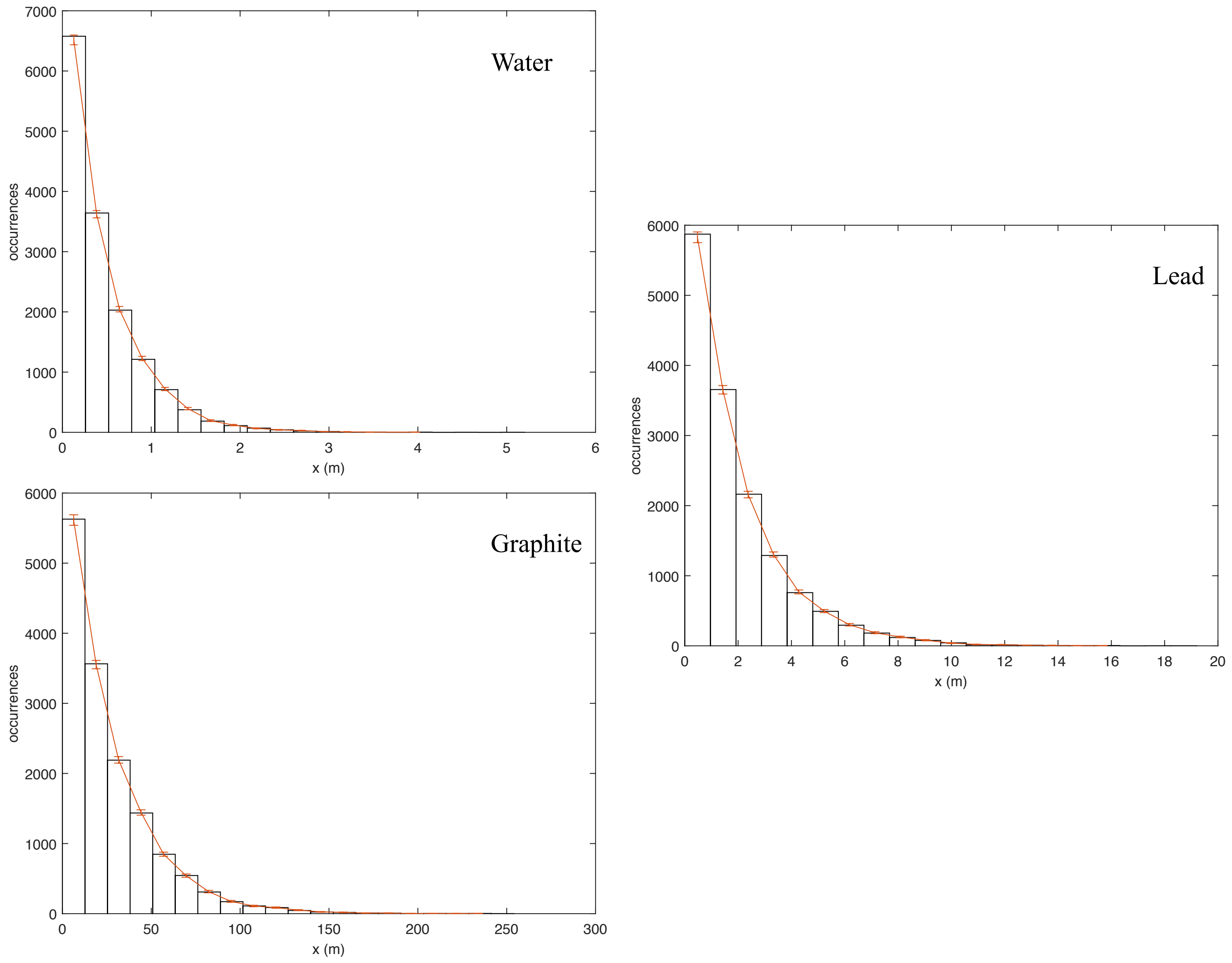

Fig 1. Histograms for number of neutrons (on *y* axis) travelling to a certain distance *x* in m (on *x* axis) for absorption only. Top left is for water, centre right for lead, bottom left for graphite.

4.2. Start neutrons in middle of slab

As an extra check to see whether scattering was implemented correctly, neutrons were started in the middle of a slab and allowed to perform random walks. The results are shown in Figure 2. All materials show nearly equal numbers of neutrons transmitted out of both slab sides, with exact numbers shown in Table 2. Water shows lots of absorption, lead a little less and graphite not very much. Within errors, the number of neutrons transmitted out of both slab sides are equal and thus the plots look symmetrical. The expected neutron density at a particular point would follow an inverse square law with distance from the source due to the spreading of neutrons, modulated by an exponential distribution due to absorption, but this would be difficult to measure.

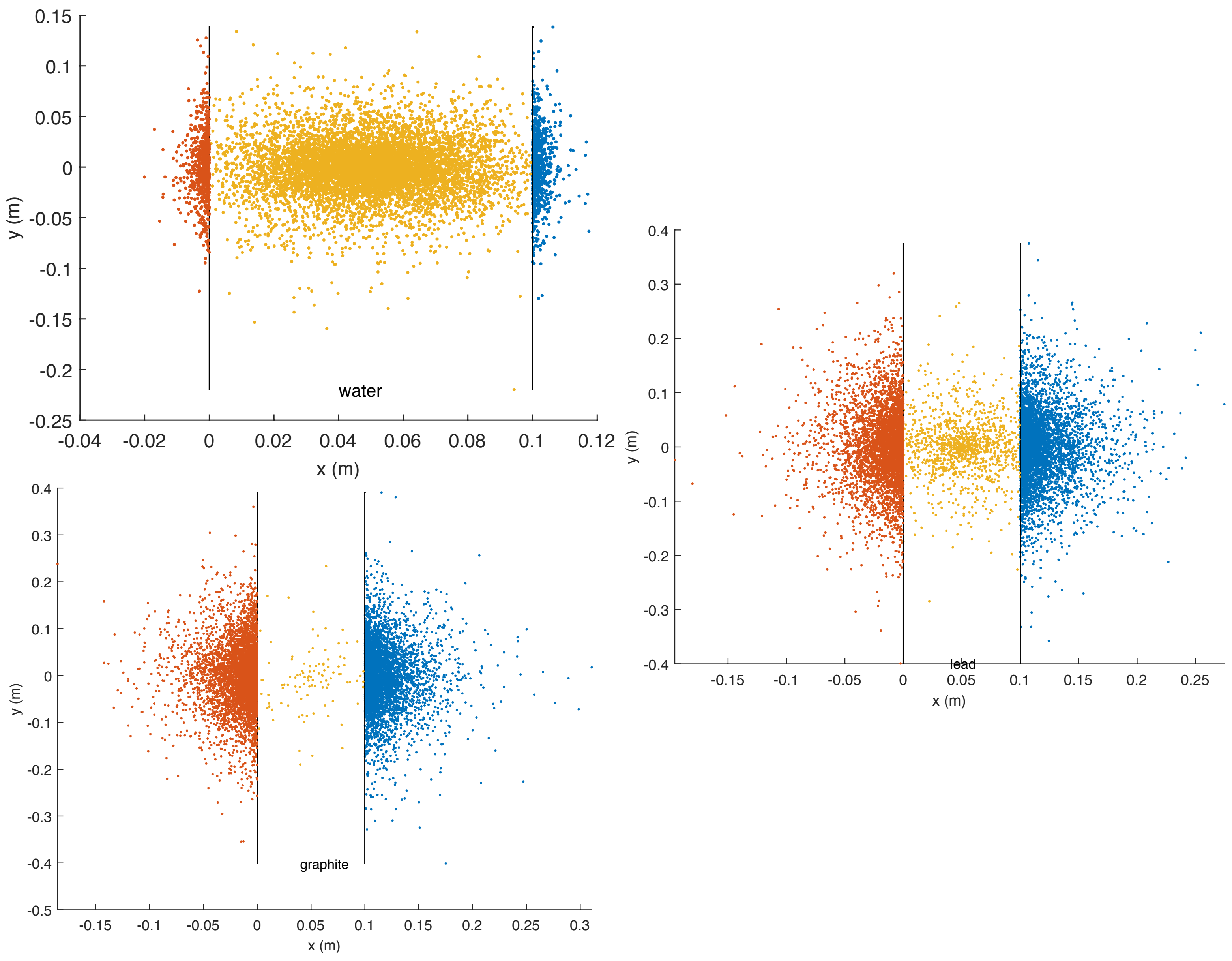

Fig 2. Plots of neutron history ends for source in the middle of a 10 cm slab for water (top left), lead (centre right) and graphite (bottom left). The dots for absorbed neutrons indicate where they were absorbed but the dots outside the slab indicate the end of a step which caused the neutron to leave the slab. In the model where what surrounds the slab has an infinite mean free path, once a neutron leaves the slab it will carry on to infinity so the 'end of a step' does not mean much. Though these looks like 2D plots the particles have a third component in *z*.

Material	Water	Lead	Graphite
Neutrons transmitted through $x = 0.1$ m edge	814 ± 28	4383 ± 52	4955 ± 52
Neutrons transmitted through $x = 0$ m edge	818 ± 29	4390 ± 52	4944 ± 52

Table 2. Table showing number of neutrons transmitted out of both slab sides for the source in the middle of the slab at $x = 0.05$ m for water, lead and graphite. Errors are quoted to the nearest whole neutron.

4.3. Start neutrons on slab side

In another experiment, neutrons were started at one slab side and allowed to diffuse. Results are illustrated in Figure 3. Lead and graphite look similar to their counterparts in Figure 2, whilst water shows little transmission and the asymmetry is clearer.

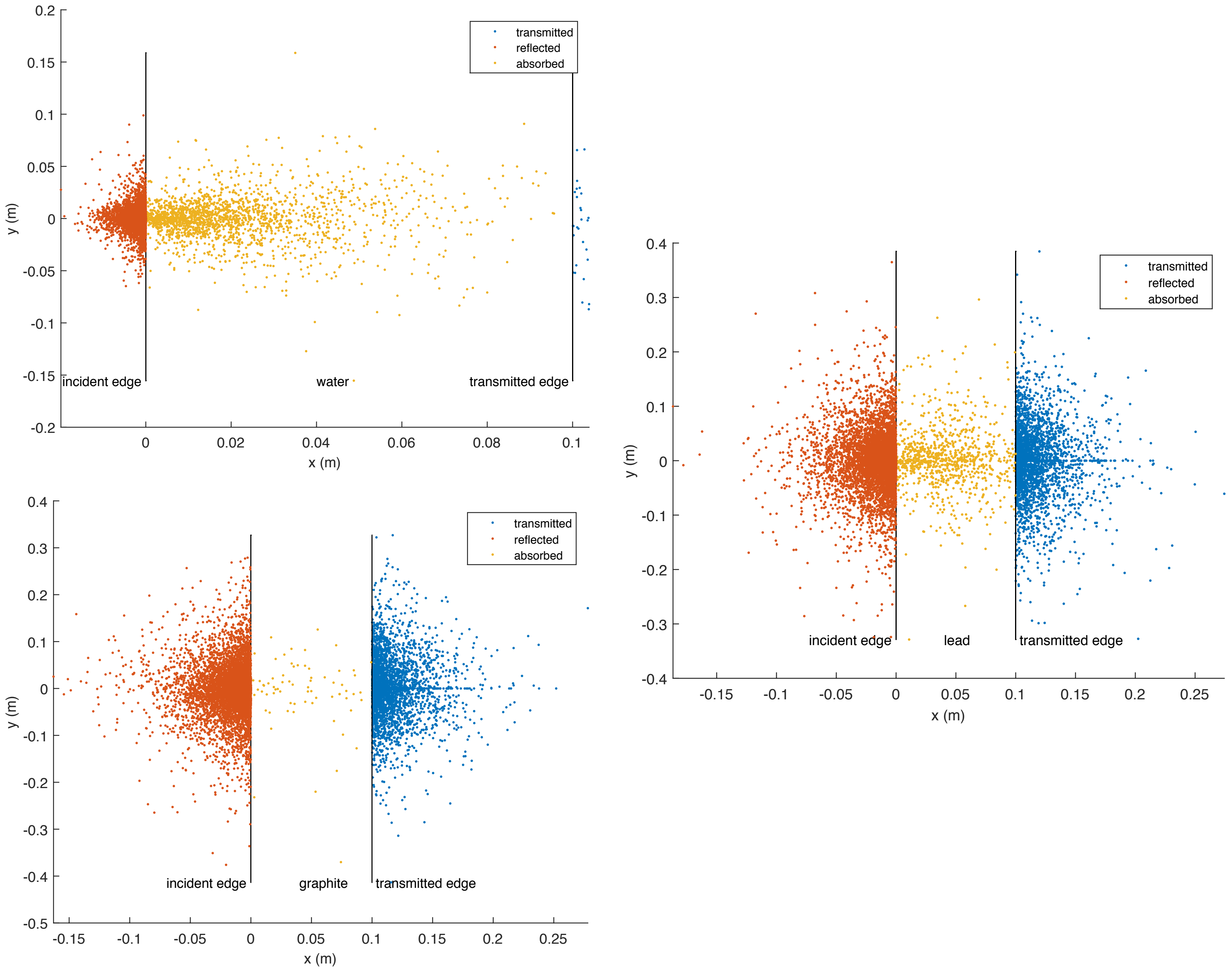

Fig 3. Like Figure 2, but plots of neutron history ends for source incident upon a 10 cm slab side (rather than source in middle of a slab) for water (top left), lead (centre right) and graphite (bottom left). The dots for absorbed neutrons indicate where they were absorbed but the dots outside the slab indicate the end of a step which caused the neutron to leave the slab.

Table 3 shows transmission, reflection and absorptions percentages for 10 cm slabs of water, lead and graphite when neutrons are started at one slab side. Here, transmission and reflection are different within errors.

Material	Water	Lead	Graphite
Transmission (%)	0.33 ± 0.05	28.37 ± 0.46	30.77 ± 0.45
Reflection (%)	80.02 ± 0.38	62.80 ± 0.42	68.49 ± 0.47
Absorption (%)	19.66 ± 0.37	8.83 ± 0.29	0.74 ± 0.08

Table 3. Data for transmission, reflection and absorption percentages for 10 cm slabs of water, lead and graphite when neutrons were started at slab side. Errors are quoted to 2 decimal places (to be able to convert to number of neutrons by multiplying by 10000).

Some particle histories were tracked at each step to plot a random walk, as shown in Figure 4. Figure 4 illustrates how neutrons in water have a short mean free path (zigzag trajectory) and are mostly reflected or absorbed. Lead has a longer mean free path but still mostly reflects and absorbs. Graphite, however, transmits some neutrons.

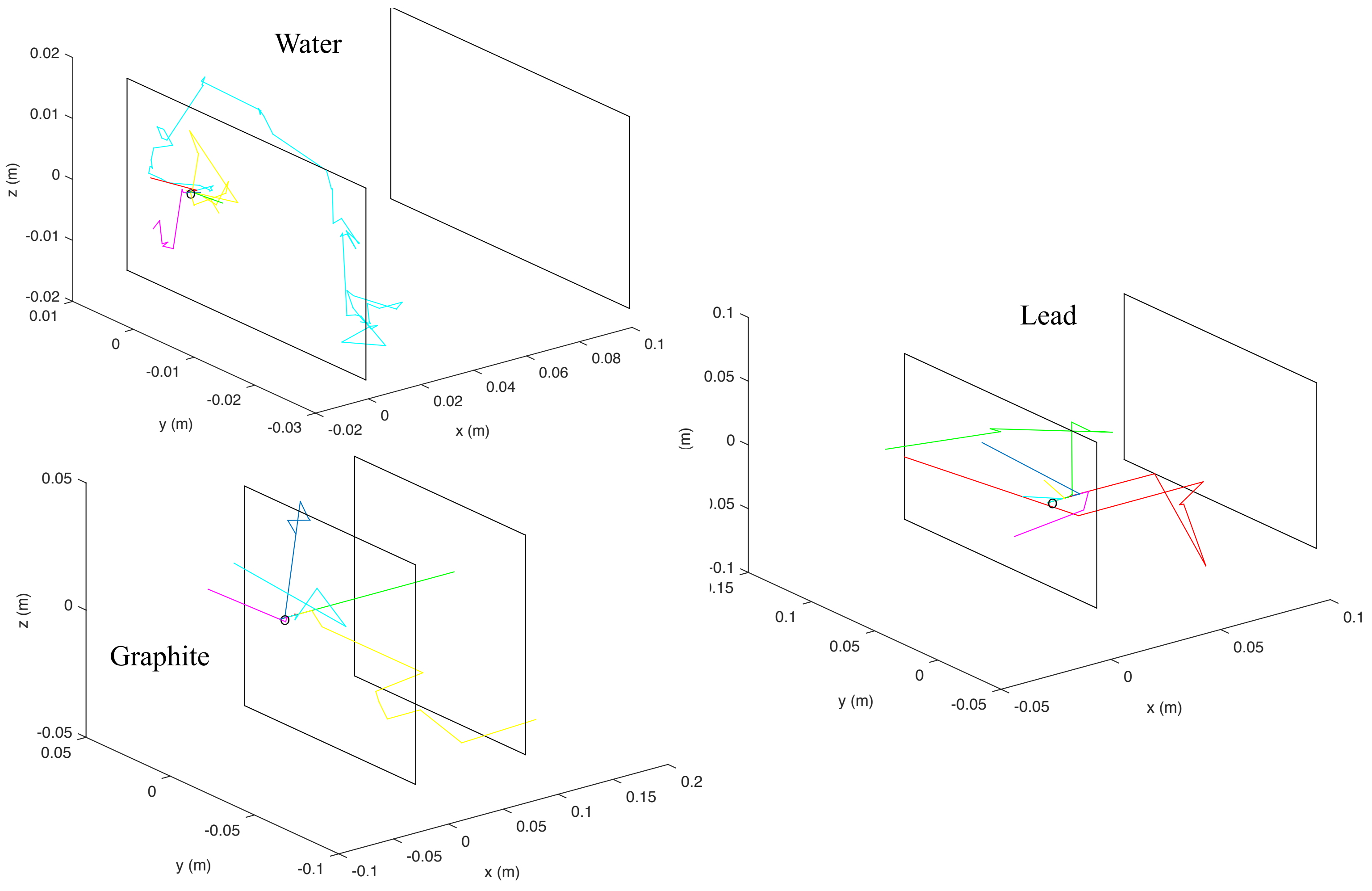

Fig 4. Random walk 3D visualisations for water (top left), lead (centre right) and graphite (bottom left). Each different colour represents a different neutron. The planes represent both sides of the 10 cm thick slabs. The origin is labelled as O and each axis is in m.

From varying slab thicknesses the attenuation lengths and stopping distances for 10000 neutrons were calculated and are given in Table 4. The graphs used for determining the attenuation lengths are shown in Figure 5, which shows that transmission fraction does not exactly decrease exponentially with slab thickness.

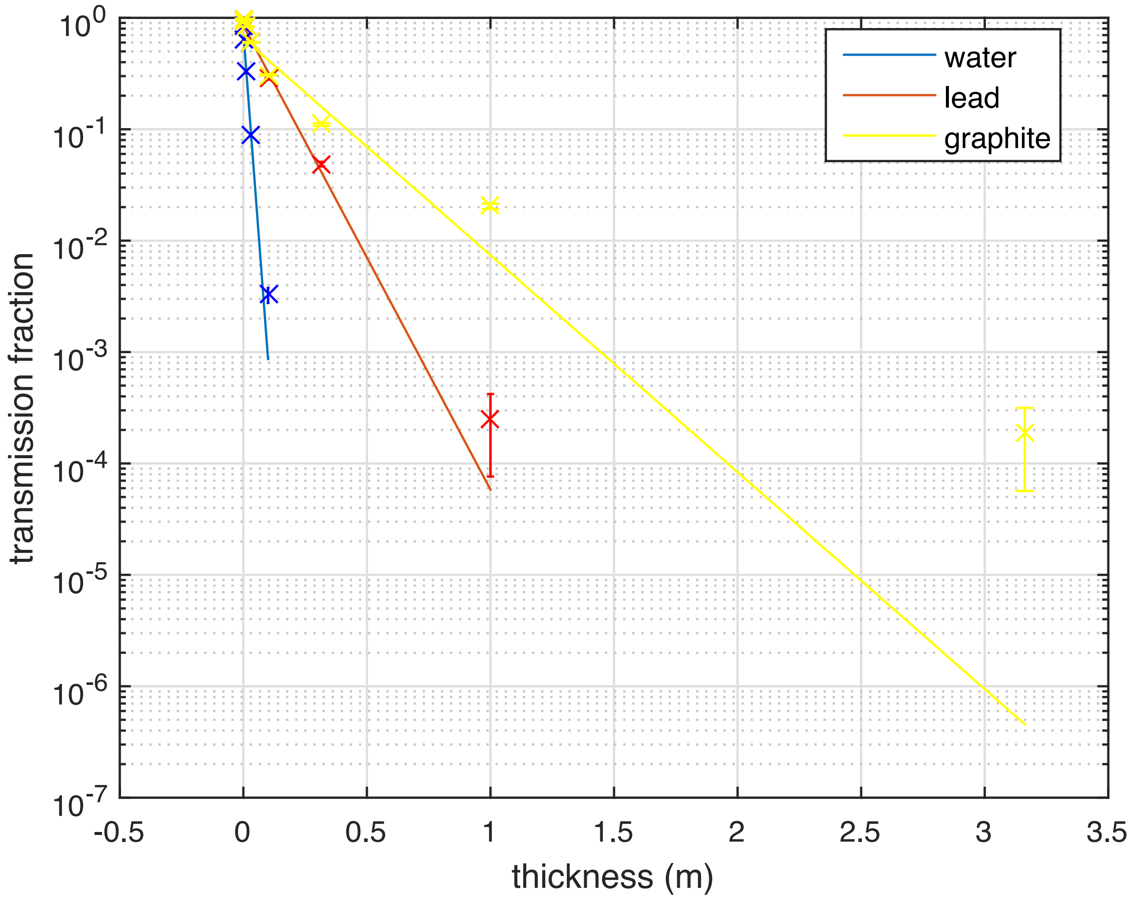

Fig 5. Plots of transmission fraction against slab thickness in m for water (blue), lead (red) and graphite (yellow). The graph shows that the relation between transmission fraction and slab thickness is not entirely exponential.

Material	Water	Lead	Graphite
Attenuation length (m)	0.0149 ± 0.0002	0.105 ± 0.002	0.227 ± 0.007
Stopping distance (m)	0.147 ± 0.002	1.04 ± 0.02	2.24 ± 0.07

Table 4. Table showing calculated attenuation lengths in m and stopping distances in m for 10000 neutrons for water, lead and graphite.

4.4. Woodcock method

Like in Figure 3, neutrons were started on one slab side and diffused through slabs of different materials in contact to produce the neutron history ends shown in Figure 6. Graphite placed before lead shows no absorption yet lead placed before graphite shows plenty of absorption, especially in lead. Water shows lots of absorption wherever placed and therefore little transmission is observed when water is present. Graphite and lead show little absorption when placed after water but much more when placed before.

To check whether the results from Figure 6 were reasonable they were compared to the transmission from 10cm slabs of material on their own, accounting for reflections between the materials. The comparisons are shown in Table 5. Where the error is larger than the value it suggests there is a fair chance of no transmission.

Materials	Graphite then lead	Graphite then water	Lead then graphite	Lead then water	Water then graphite	Water then lead	Graphite & lead	Graphite & water	Lead & water
Transmission fraction (%)	18.30 ± 0.42	0.32 ± 0.06	14.42 ± 0.33	0.33 ± 0.06	0.20 ± 0.05	0.17 ± 0.04	15.32 ± 1.54	0.22 ± 1.21	0.19 ± 1.19

Table 5. Table showing transmission percentages for different setups. "Then" means a 10 cm slab followed by another 10 cm slab using the Woodcock method whilst "&" means the combined result of a single slab of those two materials.

Similarly to Figure 4, some neutron histories for the Woodcock method are shown in Figure 7. Figure 7 shows interesting boundary effects in most plots where a particle is reflected back and forth between the different materials. It also clearly displays fictitious steps where water is the second material.

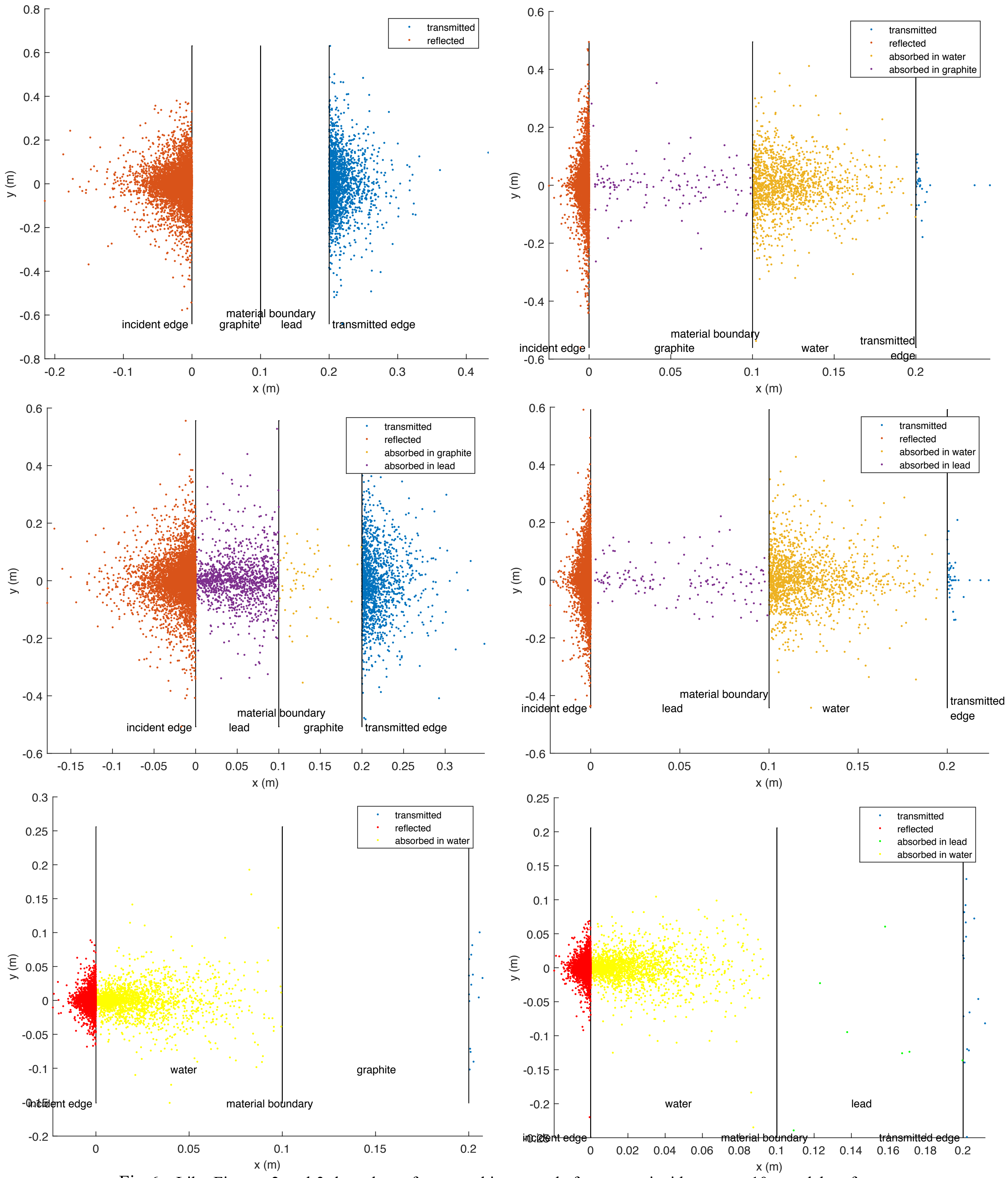

Fig 6. Like Figures 2 and 3, but plots of neutron history ends for source incident upon 10 cm slabs of different materials in contact. Top left is graphite placed before lead, top right graphite placed before water, centre left lead placed before graphite, centre right lead placed before water, bottom left water placed before graphite and bottom right water placed before lead. The dots for absorbed neutrons indicate where they were absorbed but the dots outside the slab indicate the end of a step which caused the neutron to leave the slab.

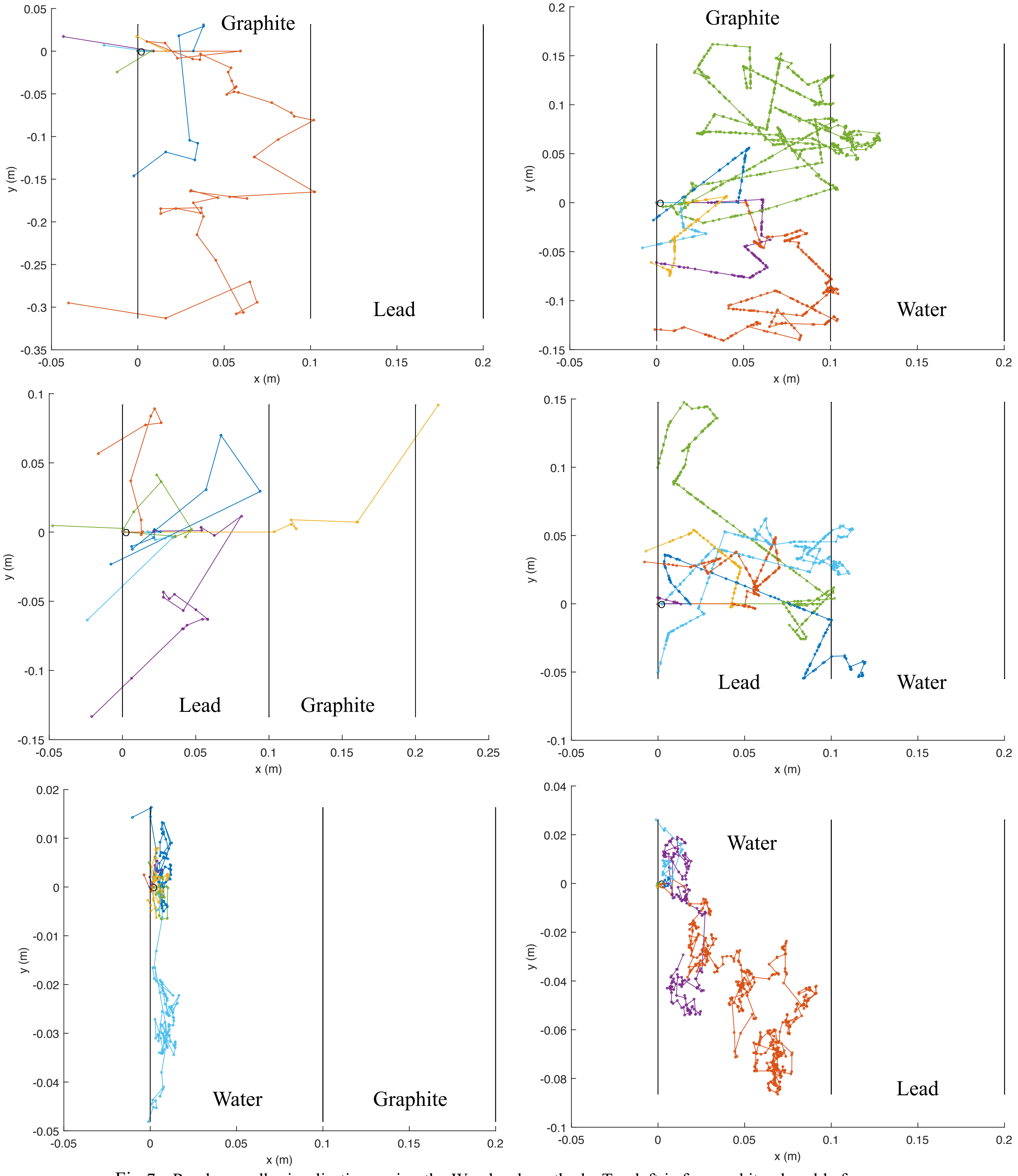

Fig 7. Random walk visualisations using the Woodcock method. Top left is for graphite placed before lead, top right graphite placed before water, centre left lead placed before graphite, centre right lead placed before water, bottom left water placed before graphite and bottom right water placed before lead. Each particle starts at the origin, indicated by O. Each dot shows the end of a step, therefore straight lines show fictitious steps. The lines from left to right indicate the incident edge at $x = 0$ m, material boundary at $x = 0.1$ m and the transmitted edge at $x = 0.2$ m.

5. Discussion

Graphite has smaller cross sections for both neutron absorption and scattering than lead, but lead has a higher mass than carbon, which means that their total mean free paths are similar. This is why little difference is seen in neutron distribution from Figures 2 and 3 for lead and graphite. From Figures 2 and 3 the conclusion to be drawn is that neutron starting position has more effect when the mean free path is shorter. Only 6 particles were plotted in Figures 4 and 7 to avoid overcrowding the plots. 10000 neutrons were simulated to be able to easily convert from percentage to number of particles. For absorption only, attenuation length decreases exponentially with thickness (as illustrated in Figure 1), but when scattering is included this relation no longer holds since scattering direction is random. It is difficult to model the exact distribution present, therefore an exponential is used as a reasonable estimate.

Figure 6 shows that the order of materials determines the final transmission, absorption and reflection. When water is placed first, a large fraction of neutrons are absorbed, therefore fewer pass into the next material. This is why less absorption takes place in lead and graphite when placed after water compared to when placed before. A clear difference is seen between lead placed before graphite and graphite placed before lead because in the first case graphite reflects a significant fraction of neutrons back into lead, therefore they spend more time in lead and more are absorbed.

6. Summary

Water, lead and graphite are all currently in use in different parts of nuclear reactors. Water absorbs well and therefore reduces the number of neutrons passing onto the next fuel rod, preventing a runaway chain reaction. Thus, water is used as a moderator in nuclear reactors. Graphite and lead reflect well, therefore they are used at the edges of a reactor to divert neutrons backwards. Since lead has a shorter attenuation length than graphite, less of it is needed to reduce transmission out of the reactor, but graphite may be cheaper. Combinations of these materials may also be employed to fulfill these needs.

References

[1] Maekawa, H., Seki, Y., Hiraoka, T., Moriyama, M,. "Uranium-238 to Uranium-235 Fission-Ratio Distribution in Spherical Lithium-Metal Assemblies With and Without a Graphite Reflector", *Nuclear Science and Engineering*, Volume 57, Issue 4, page 335, 1975.

[2] Broerse, J.J., "Survival of Cultured Human Cells after Irradiation with Fast Neutrons of Different Energies in Hypoxic and Oxygenated Conditions", *International Journal of Radiation Biology and Related Studies in Physics, Chemistry and Medicine,* Volume 13, Issue 6, page 559, 1968.

[3] Metropolis, N. & Ulam, S., “The Monte Carlo Method”, *Journal of the American Statistical Association*, Volume 44, Issue 247, pages 335-341, 1949.

[4] Woodcock, E., Murphy, T., Hemmings, P., Longworth, T., “Techniques used in the GEM code for Monte Carlo neutronics calculations in reactors and other systems of complex geometry”, In *Proc. Conference on the Application of Computing Methods to Reactor Problems,* pages 557-559, 1965.

[5] MATLAB, Version 5, The Math Works Inc, Natick, Mass 01760.

The number of words in this document is 1506.

This document was last saved on 15/5/2016 at 14:20.

Appendix

```
% project3
% Monte Carlo techniques - penetration of neutrons through shielding
% Meirin Evans, May 16
% ------------------------------------------------------------------------------------------


% ------------------------------------------------------------------------------------------
% clear workspace & figure windows
clear;
close all;
% ------------------------------------------------------------------------------------------


% ------------------------------------------------------------------------------------------
% random number visualisations
Numrandnums = 15000;                                       % # of random #s
xyzrand = rand(Numrandnums,1);                             % generate N random #s
xyzssp = randssp(Numrandnums,1);                      % generate N pseudorandom #s
randnum = struct('x', [], 'y', [], 'z', []);               % rand struct
ssp = struct('x', [], 'y', [], 'z', []);                   % randssp struct
sphere = struct('x', [], 'y', [], 'z', []);                % unitVector struct
for k = 1: Numrandnums/3                                   % loop k over N
   randnum.x(k) = xyzrand(3*k - 2);                        % xrand is 1st #
   randnum.y(k) = xyzrand(3*k - 1);                        % yrand is 2nd #
   randnum.z(k) = xyzrand(3*k);                            % zrand is next # etc
   ssp.x(k) = xyzssp(3*k - 2);                             % xssp is 1st #
   ssp.y(k) = xyzssp(3*k - 1);                             % yssp is 2nd #
   ssp.z(k) = xyzssp(3*k);                                 % zssp is next # etc
   [sphere.x(k), sphere.y(k), sphere.z(k)] = unitVector(); % call unitVector
end                                                        % end k loop

% plot uniform random numbers
figure;                                                    % open fig 1
[height, centre] = hist(xyz, 20);                          % hist data
histogram(xyz);                                            % plot random #s in 20 bins
ylabel('occurrences’, ‘FontSize’, 12);                     % x axis label
xlabel('random number', ‘FontSize’, 12);                   % y axis label
set(gca, 'FontSize', 12);                                  % axis font size
hold on;                                                   % plot on same fig
errorbar(centre, height, height*0.05*(1-0.05));            % heights error
plot([0 1], [N Numrandnums/20 N Numrandnums/20]);          % average line
hold off;                                                  % turn hold off
print('uniform #s', '-depsc');                             % save fig 1
```

```
% plot 3D random numbers
figure;                                          % open fig 2
plot3(rand.x ,rand.y ,rand.z , 'ro');            % plot random #s
xlabel('x', ‘FontSize’, 12);                     % x axis label
ylabel('y', ‘FontSize’, 12);                     % y axis label
zlabel('z', ‘FontSize’, 12);                     % z axis label
set(gca, 'FontSize', 12);                        % axis font size
print('rand', '-depsc');                         % save fig 2

% plot 3D randssp numbers
figure;                                          % open fig 3
plot3(ssp.x, ssp.y, ssp.z, 'go');                % plot pseudorandom #s
xlabel('x', ‘FontSize’, 12);                     % x axis label
ylabel('y', ‘FontSize’, 12);                     % y axis label
zlabel('z', ‘FontSize’, 12);                     % z axis label
set(gca, 'FontSize', 12);                        % axis font size
print('randssp', '-depsc');                      % save fig 3

% plot points over sphere
figure;                                          % open fig 4
plot3(sphere.x, sphere.y, sphere.z, 'o');        % plot unit vectors
xlabel('x', ‘FontSize’, 12);                     % x axis label
ylabel('y', ‘FontSize’, 12);                     % y axis label
zlabel('z', ‘FontSize’, 12);                     % z axis label
set(gca, 'FontSize', 12);                        % axis font size
print('unitVector', '-depsc');                   % save fig 4
% ---------------------------------------------------------------------------------------------


% ---------------------------------------------------------------------------------------------
% isotropic steps
atten = struct('wat', 18*1.661e-27/(1000*0.6652e-28), 'lead', 207*1.661e-27/
(11.35*1000*0.158e-28), 'graph', 12*1.661e-27/(1.67*1000*0.0045e-28));
                                                 % atten struct
dist = struct('wat', [], 'lead', [], 'graph', []);  % exponential struct
for i = 1:N                                      % loop i over N
   dist.wat(i) = -atten.wat*log(rand());         % isotropic water steps
   dist.lead(i) = -atten.lead*log(rand());       % isotropic lead steps
   dist.graph(i) = -atten.graph*log(rand());     % isotropic graphite steps
end                                              % end i loop

% water
[ninbinwat, centrewat] = hist(dist.wat, 20);     % water hist
centrewat = centrewat(ninbinwat>1);              % keep centrewat if >1 in bin
ninbinwat = ninbinwat(ninbinwat>1);              % keep ninbinwat if >1 in bin
```

```
polywat = polyfitweighted(centrewat, log(ninbinwat), 1, sqrt(ninbinwat));
                                                    % water polyfit
[ninbinlead, centrelead] = hist(dist.lead, 20);     % lead hist
centrelead = centrelead(ninbinlead>1);              % keep centrelead if >1 in bin
ninbinlead = ninbinlead(ninbinlead>1);              % keep ninbinlead if >1 in bin
polylead = polyfitweighted(centrelead, log(ninbinlead), 1, sqrt(ninbinlead));
                                                    % lead polyfit
[ninbingraph, centregraph] = hist(dist.graph, 20);        % graphite hist
centregraph = centregraph(ninbingraph>1);           % keep centregraph if >1 in bin
ninbingraph = ninbingraph(ninbingraph>1);           % keep ninbingraph if >1 in bin
polygraph = polyfitweighted(centregraph, log(ninbingraph), 1, sqrt(ninbingraph));
                                                    % graphite polyfit
lamda = struct('wat', -1/polywat(1), 'lead', -1/polylead(1), 'graph', -1/polygraph(1));
                                                    % calculated attenuation length
errwatatten = abs(atten.wat - lamda.wat);                 % water MFP error
figure;                                                   % open fig 5
histogram(dist.wat, 20, 'FaceColor', ‘none');             % plot water hist
xlabel('x (m)', ‘FontSize’, 12);                          % x axis label
ylabel('occurrences', ‘FontSize’, 12);                    % y axis label
set(gca, 'FontSize', 12);                                 % axis font size
hold on;                                                  % plot on same fig
errorbar(centrewat, ninbinwat, sqrt(ninbinwat));          % errorbars on hist
hold off;                                                 % turn hold off
print('water hist', '-depsc');                            % save fig 5

figure;                                                         % open fig 6
semilogy(centrewat, exp(polyval(pwat, centrewat)));             % plot log water hist
xlabel('x (m)', ‘FontSize’, 12);                                % x axis label
ylabel('occurrences', ‘FontSize’, 12);                          % y axis label
set(gca, 'FontSize', 12);                                       % axis font size
hold on;                                                        % plot on same fig
errorbar(centrewat, ninbinwat, sqrt(ninbinwat), 'bx'); grid     % plot water line
hold off;                                                       % turn hold off
print('water log hist', '-depsc');                              % save fig 6

% lead
errleadatten = abs(atten.lead - lamda.lead);              % lead MFP error
figure;                                                   % open fig 7
histogram(g.lead, 20, 'FaceColor', ‘none');               % plot lead hist
xlabel('x (m)', ‘FontSize’, 12);                          % x axis label
ylabel('occurrences', ‘FontSize’, 12);                    % y axis label
set(gca, 'FontSize', 12);                                 % axis font size
hold on;                                                  % plot on same fig
errorbar(centrelead, ninbinlead, sqrt(ninbinlead));       % plot lead line
hold off;                                                 % turn hold off
print('lead hist', '-depsc');                             % save fig 7
```

```
figure;                                                   % open fig 8
semilogy(centrelead, exp(polyval(plead, centrelead)));    % plot log lead hist
xlabel('x (m)', ‘FontSize’, 12);                          % x axis label
ylabel('occurrences', ‘FontSize’, 12);                    % y axis label
set(gca, 'FontSize', 12);                                 % axis font size
hold on;                                                  % plot on same fig
errorbar(centrelead, ninbinlead, sqrt(ninbinlead), 'bx'); % plot lead line
hold off;                                                 % turn hold off
print('lead log hist', '-depsc');                         % save fig8

% graphite
errgraphatten = abs(atten.graph - lamda.graph);      % graphite MFP error
figure;                                              % open fig 9
hist(g.graph, 20, 'FaceColor', ‘none');              % plot graphite hist
xlabel('x (m)', ‘FontSize’, 12);                     % x axis label
ylabel('occurrences', ‘FontSize’, 12);                    % y axis label
set(gca, 'FontSize', 12);                            % axis font size
hold on;                                             % plot on same fig
errorbar(centregraph, ninbingraph, sqrt(ninbingraph));    % plot graphite line
hold off;                                            % turn hold off
print('graphite hist', '-dpng');                     % save fig 9

figure;                                                   % open fig 10
semilogy(centregraph, exp(polyval(pgraph, centregraph))); % plot log graphite hist
xlabel('x (m)', ‘FontSize’, 12);                          % x axis label
ylabel('occurrences', ‘FontSize’, 12);                    % y axis label
set(gca, 'FontSize', 12);                                 % axis font size
hold on;                                                  % plot on same fig
errorbar(centregraph, ninbingraph, sqrt(ninbingraph), 'bx'); grid
                                                          % plot graphite line
hold off;                                                 % turn hold off
print('graphite log hist', '-depsc');                     % save fig 10
% -----------------------------------------------------------------------------------------------


% -----------------------------------------------------------------------------------------------
% MFP & cross section calculations
scat = struct('wat', 18*1.661e-27/(1000*103e-28), 'lead', 207*1.661e-27/
                    (11.35*1000*11.221e-28), 'graph', 12*1.661e-27/(1.67*1000*4.74e-28));
                                                          % scatter struct
sigma_a = struct('wat', 1/atten.wat, 'lead', 1/atten.lead, 'graph', 1/atten.graph);
                                                          % absorb x section
sigma_s = struct('wat', 1/scat.wat, 'lead', 1/scat.lead, 'graph', 1/scat.graph);
                                                          % scatter x section
```

```
sigma_t = struct('wat', sigma_s.wat + sigma_a.wat, 'lead', sigma_s.lead + sig-
ma_a.lead, 'graph', sigma_s.graph + sigma_a.graph);              % total x section
meanfree = struct('wat', 1/sigma_t.wat, 'lead', 1/sigma_t.lead, 'graph', 1/sigma_t.
                              graph);                            % mean free path
% ----------------------------------------------------------------------------------------


% ----------------------------------------------------------------------------------------
% random walks
particles = 10000;                       % # particles
numwidths = 9;                           % # different thicknesses
runs = 50;                               % # runs
T = logspace(-3, 1, numwidths);          % different thicknesses

trans = struct('wat', [], 'lead', [], 'graph', []);           % transmit struct
transerr = struct('wat', [], 'lead', [], 'graph', []);        % transmit error struct
reflect = struct('wat', [], 'lead', [], 'graph', []);         % reflect struct
reflecterr = struct('wat', [], 'lead', [], 'graph', []);      % reflect error struct
absorb = struct('wat', [], 'lead', [], 'graph', []);          % absorb struct
absorberr = struct('wat', [], 'lead', [], 'graph', []);       % absorb error struct

a = struct('wat', 2*ones(1,6), 'lead', 2*ones(1,6), 'graph', 2*ones(1,6), 'leadgraph',
         2*ones(1,6), ’leadwat’, 2*ones(1,6), 'graphlead', 2*ones(1,6), 'graphwat',
         2*ones(1,6), 'watgraph', 2*ones(1,6), 'watlead', 2*ones(1,6));     % counter
pathwat = struct('x', zeros(1,6), 'y', zeros(2,6), 'z', zeros(2,6));    % water paths
pathlead = struct('x', zeros(1,6), 'y', zeros(2,6), 'z', zeros(2,6));   % lead paths
pathgraph = struct('x', zeros(1,6), 'y', zeros(2,6), 'z', zeros(2,6)); % graphite paths

tallyoutpos = struct('wat', zeros(1, runs), 'lead', zeros(1, runs), 'graph', zeros(1,
                              runs));                % tallyout x>0 for mid
tallyoutneg = struct('wat', zeros(1, runs), 'lead', zeros(1, runs), 'graph', zeros(1,
                              runs));                % tallyout x<0 for mid
for runmid = 1:runs                                  % loop runs
   wat = struct('x', 0.05*ones(1,particles), 'y', zeros(1,particles), 'z',
                zeros(1,particles));                     % water particles
   lead = struct('x', 0.05*ones(1,particles), 'y', zeros(1,particles), 'z',
                zeros(1,particles));                     % lead particles
   graph = struct('x', 0.05*ones(1,particles), 'y', zeros(1,particles), 'z',
                zeros(1,particles));                     % graphite particles
   for l = 1:particles                                   % loop particles
      is_absorb = struct('wat', 0, 'lead', 0, 'graph', 0);      % absorb flag
      while ((is_absorb.wat==0) && (wat.x(l)>0) && (wat.x(l)<0.1))
                                                          % particle still in slab
         [xunit, yunit, zunit] = unitVector();            % call unitVector
         [dxwat, dywat, dzwat] = step(xunit, yunit, zunit, meanfree.wat);
                                                          % water steps
```

```
    wat.x(l) = wat.x(l) + dxwat;                 % water x move
    wat.y(l) = wat.y(l) + dywat;                 % water y move
    wat.z(l) = wat.z(l) + dzwat;                 % water z move

    if ((rand()<sigma_a.wat/sigma_t.wat) && (wat.x(l)>0) && (wat.x(l)<0.1))
                                                 % water x section ratio
       is_absorb.wat = 1;                        % set absorb flag to 1
    end                                          % end absorb if
end                                          % particle left slab or absorbed
while ((is_absorb.lead==0) && (lead.x(l)>0) && (lead.x(l)<0.1))
                                                 % still in slab not absorbed
    [xunit, yunit, zunit] = unitVector();        % call unitVector
    [dxlead, dylead, dzlead] = step(xunit, yunit, zunit, meanfree.lead);
                                                 % lead steps
    lead.x(l) = lead.x(l) + dxlead;              % lead x move
    lead.y(l) = lead.y(l) + dylead;              % lead y move
    lead.z(l) = lead.z(l) + dzlead;              % lead z move
    if ((rand()<sigma_a.lead/sigma_t.lead) && (lead.x(l)>0) && (lead.x(l)<0.1))
                                                 % lead x section ratio
       is_absorb.lead = 1;                       % set absorb flag to 1
    end                                          % end absorb lead if
end                                          % particles left slab or absorbed
while ((is_absorb.graph==0) && (graph.x(l)>0) && (graph.x(l)<0.1))
                                                 % still in slab
    [xunit, yunit, zunit] = unitVector();        % call unitVector
    [dxgraph, dygraph, dzgraph] = step(xunit, yunit, zunit, meanfree.graph);
                                                 % graphite steps
    graph.x(l) = graph.x(l) + dxgraph;           % graphite x move
    graph.y(l) = graph.y(l) + dygraph;           % graphite y move
    graph.z(l) = graph.z(l) + dzgraph;           % graphite z move
    if ((rand()<sigma_a.graph/sigma_t.graph) && (graph.x(l)>0) &&
                (graph.x(l)<0.1))            % graphite cross section ratio
       is_absorb.graph = 1;                  % set absorb flag to 1
    end                                      % end absorb if
end                                          % particle left slab or absorb
if (wat.x(l)>0.1)                            % transmitted x>0 water mid
    tallyoutpos.wat(runmid) = tallyoutpos.wat(runmid) + 1;
                                             % add 1 to tallyoutposwat
end                                          % end water transmit x>0 if
if (lead.x(l)>0.1)                           % lead transmitted x>0 mid
    tallyoutpos.lead(runmid) = tallyoutpos.lead(runmid) + 1;
                                             % add 1 to tallyoutposlead
end                                          % end lead transmitted x>0 if
if (graph.x(l)>0.1)                          % transmitted x>0 graphite mid
    tallyoutpos.graph(runmid) = tallyoutpos.graph(runmid) + 1;
                                             % add 1 to tallyoutposgraph
```

```
        end                                         % end graphite transmitted x>0 if
        if (wat.x(l)<0)                             % transmitted x<0 water mid
           tallyoutneg.wat(runmid) = tallyoutneg.wat(runmid) + 1;
                                                    % add 1 to tallyoutnegwat
        end                                         % end water transmit x<0 if
        if (lead.x(l)<0)                            % lead transmitted x<0 mid
           tallyoutneg.lead(runmid) = tallyoutneg.lead(runmid) + 1;
                                                    % add 1 to tallyoutneglead
        end                                         % end lead transmitted x<0 if
        if (graph.x(l)<0)                           % transmitted x<0 graphite mid
           tallyoutneg.graph(runmid) = tallyoutneg.graph(runmid) + 1;
                                                    % add 1 to tallyoutneggraph
        end                                         % end transmitted graphite x<0 if
     end                                            % end particle loop
  end                                               % end runs loop
tallyoutposavg = struct('wat', {mean(tallyoutpos.wat)}, 'lead', {mean(tallyout
         pos.lead)}, 'graph', {mean(tallyoutpos.graph)});          % mean tallyout x>0
tallyoutnegavg = struct('wat', {mean(tallyoutneg.wat)}, 'lead', {mean(tallyout
         neg.lead)}, 'graph', {mean(tallyoutneg.graph)});          % mean tallyout x<0
errtallyoutpos = struct('wat', {std(tallyoutpos.wat)}, 'lead', {std(tallyoutpos.lead)},
         'graph', {std(tallyoutpos.graph)});                       % error tallyout x>0
errtallyoutneg = struct('wat', {std(tallyoutneg.wat)}, 'lead', {std(tallyoutneg.lead)},
         'graph', {std(tallyoutneg.graph)});                       % error tallyout x<0
% -------------------------------------------------------------------------------------


% -------------------------------------------------------------------------------------
% source in slab centre
% water
minymidwat = min(wat.y);                            % min y water val for mid start
maxymidwat = max(wat.y);                            % max y water val for mid start
minzmidwat = min(wat.z);                            % min z water val for mid start
maxzmidwat = max(wat.z);                            % max z water val for mid start
figure;                                             % open fig 11
xlim([min(wat.x) max(wat.x)]);                      % x axis limits
wattranspos = struct('x', wat.x(wat.x>0.1), 'y', wat.y(wat.x>0.1), 'z',
                              wat.z(wat.x>0.1));             % left water slab at x>0
wattransneg = struct('x', wat.x(wat.x<0), 'y', wat.y(wat.x<0), 'z', wat.z(wat.x<0));
                                                             % left water slab at x<0
watabsorbmid = struct('x', wat.x(wat.x>0 & wat.x<0.1), 'y', wat.y(wat.x>0 &
         wat.x<0.1), 'z', wat.z(wat.x>0 & wat.x<0.1));       % absorbed in water slab
plot3(wattranspos.x, wattranspos.y, wattranspos.z, '.');% plot transmitted x>0 water
hold on;                                                     % plot on same fig
plot3(wattransneg.x, wattransneg.y, wattransneg.z, '.');% plot transmitted x<0 water
plot3(watabsorbmid.x, watabsorbmid.y, watabsorbmid.y, '.');% plot absorbed water
xlabel('x (m)', 'FontSize', 12);                             % x axis label
```

```
ylabel('y (m)', 'FontSize', 12);                                   % y axis label
zlabel('z (m)', 'FontSize', 12);                                   % z axis label
set(gca, 'FontSize', 12);                                          % axis font size
[yywatmid, zzwatmid] = meshgrid(minymidwat:(maxymidwat-minywat):maxymid-
wat, minzmidwat:(maxzmidwat-minzmidwat):maxzmidwat);   % water mid grid
surf(zeros(size(yywatmid)), yywatmid, zzwatmid);                   % x=0 edge
surf(0.1*ones(size(yywatmid)), yywatmid, zzwatmid);                % other edge
text(0.04, minymidwat, minzmidwat, 'lead', 'FontSize', 11);
                                                   % label where water present
hold off;                                                          % turn hold off
print('water mid particles', '-depsc');                            % save fig 11

% lead
minymidlead = min(lead.y);                         % min lead y val for mid start
maxymidlead = max(lead.y);                         % max lead y val for mid start
minzmidlead = min(lead.z);                         % min lead z val for mid start
maxzmidlead = max(lead.z);                         % max lead z val for mid start
figure;                                            % open fig 12
xlim([min(lead.x) max(lead.x)]);                   % x axis limits
leadtranspos = struct('x', lead.x(lead.x>0.1), 'y', lead.y(lead.x>0.1), 'z',
                         lead.z(lead.x>0.1));            % left lead slab at x>0
leadtransneg = struct('x', lead.x(lead.x<0), 'y', lead.y(lead.x<0), 'z', lead.z(lead.x<0));
                                                         % left lead stab at x<0
leadabsorbmid = struct('x', lead.x(lead.x>0 & lead.x<0.1), 'y', lead.y(lead.x>0 &
lead.x<0.1), 'z', lead.z(lead.x>0 & lead.x<0.1));        % absorbed in lead slab
hold on;                                                      % plot on same fig
plot3(leadtranspos.x, leadtranspos.y, leadtranspos.z, '.');
                                                         % plot transmitted x>0 lead
plot3(leadtransneg.x, leadtransneg.y, leadtransneg.z, '.');
                                                         % plot transmitted x<0 lead
plot3(leadabsorbmid.x, leadabsorbmid.y, leadabsorbmid.z, '.');
                                                         % plot absorbed lead
xlabel('x (m)', 'FontSize', 12);                         % x axis label
ylabel('y (m)', 'FontSize', 12);                         % y axis label
zlabel('z (m)', 'FontSize', 12);                         % z axis label
set(gca, 'FontSize', 12);                                % axis font size
[yyleadmid, zzleadmid] = meshgrid(minymidlead:(maxymidlead-
          minymidlead):maxymidlead, minzmidlead:(maxzmidlead-
               minzmidlead):maxzmidlead);                % lead mid grid
surf(zeros(size(yyleadmid)), yyleadmid, zzleadmid);      % x=0 edge
surf(0.1*ones(size(yyleadmid)), yyleadmid, zzleadmid);   % other edge
text(0.04, minymidlead, minzmidlead, 'lead', 'FontSize', 11);
                                                   % label where lead present
hold off;                                                % turn hold off
print('lead mid particles', '-depsc');                   % save fig 12
```

```
% graphite
minymidgraph = min(graph.y);                    % min graphite y val
maxymidgraph = max(graph.y);                    % max graphite y val
minzmidgraph = min(graph.z);                    % min graphite z val
maxzmidgraph = max(graph.z);                    % max graphite z val
figure;                                         % open fig 13
xlim([min(graph.x) max(graph.x)]);              % x axis limits
graphtranspos = struct('x', graph.x(graph.x>0.1), 'y', graph.y(graph.x>0.1), 'z',
                    graph.z(graph.x>0.1));      % transmitted x>0 graphite
graphtransneg = struct('x', graph.x(graph.x<0), 'y', graph.y(graph.x<0), 'z',
                    graph.z(graph.x<0));        % transmitted x<0 graphite
graphabsorbmid = struct('x', graph.x(graph.x>0 & graph.x<0.1), 'y',
graph.y(graph.x>0 & graph.x<0.1), 'z', graph.z(graph.x>0 & graph.x<0.1));
                                                % absorbed lead
hold on;                                        % plot on same fig
plot3(graphtranspos.x, graphtranspos.y, graphtranspos.z, '.');
                                                % plot transmitted x>0 graphite
plot3(graphtransneg.x, graphtransneg.y, graphtransneg.z, '.');
                                                % plot transmitted x<0 graphite
plot3(graphabsorbmid.x, graphabsorbmid.y, graphabsorbmid.z, '.');
                                                % plot absorbed graphite
xlabel('x (m)', 'FontSize', 12);                % x axis label
ylabel('y (m)', 'FontSize', 12);                % y axis label
zlabel('z (m)', 'FontSize', 12);                % z axis label
set(gca, 'FontSize', 12);                       % axis font size
[yygraphmid, zzgraphmid] = meshgrid(minymidgraph:(maxymidgraph-minymid
               graph):maxymidgraph, minzmidgraph:(maxzmidgraph-minzmid
                    graph):maxzmidgraph);       % graphite grid
surf(zeros(size(yygraphmid)), yygraphmid, zzgraphmid);      % x=0 edge
surf(0.1*ones(size(yygraphmid)), yygraphmid, zzgraphmid);   % other edge
text(0.04, minymidgraph, minzmidgraph, 'graphite', 'FontSize', 11);
                                                % label where graphite present
hold off;                                       % turn hold off
print('graphite mid particles', '-depsc');      % save fig 13

% source incident on slab edge
for width = 1:numwidths                                 % loop over widths
   tallyout = struct('wat', zeros(1, runs), 'lead', zeros(1, runs), 'graph', zeros(1,
                    runs));                             % tallyout struct
   tallyin = struct('wat', zeros(1, runs), 'lead', zeros(1, runs), 'graph', zeros(1, runs));
                                                        % tallyin struct
   numreflect = struct('wat', [], 'lead', [], 'graph', []);    % reflect struct
   for run = 1:runs                                     % loop runs
      wat = struct('x', [], 'y', zeros(1, particles), 'z', zeros(1, particles));
                                                        % water particles
```

```
    lead = struct('x', [], 'y', zeros(1, particles), 'z', zeros(1, particles));
                                                      % lead particles
    graph = struct('x', [], 'y', zeros(1, particles), 'z', zeros(1, particles));
                                                      % graphite particles
    for n = 1:particles                               % loop particles
      is_absorb = struct('wat', 0, 'lead', 0, 'graph', 0);  % start absorb flag at 0
      xstep1wat = steps(1, 0, 0, meanfree.wat);       % water step 1
      wat.x(n) = abs(xstep1wat);                      % water x step `
      xstep1lead = steps(1, 0, 0, meanfree.lead);     % lead step 1
      lead.x(n) = abs(xstep1lead);                    % lead x step 1
      xstep1graph = steps(1, 0, 0, meanfree.graph);   % graphite step 1
      graph.x(n) = abs(xstep1graph);                  % graphite x step 1
      if ((T(width)==0.1 && (run==1))                 % only for 1st run of T=0.1
        if ((n>0) && (n<7))                           % water particle 1-6
          pathwat.x(2,n) = abs(xstep1wat);            % water particle 1-6 step 1
          pathlead.x(2,n) = abs(xstep1lead);          % lead particle 1-6 step 1
          pathgraph.x(2,n) = abs(xstep1graph); % graphite particle 1-6 step 1
        end                                           % end step 1 if
      end                                             % end run 1 & T=0.1 if
      while ((is_absorb.wat==0) && (wat.x(n)>0) && (wat.x(n)<T(width)))
                                                      % particle still in slab
        [xunit, yunit, zunit] = unitVector();         % call unitVector
        [dxwat, dywat, dzwat] = step(xunit, yunit, zunit, meanfree.wat);
                                                      % water steps
        if ((T(width)==0.1) && (run==1))              % run 1 & T=0.1
          if ((n>0) && (n<7))                         % water particle 1-6
            pathwat.x(a.wat(n)+1,n) = pathwat.x(a.wat(n),n) + dxwat;
                                                      % water particle 1-6 x path
            pathwat.y(a.wat(n)+1,n) = pathwat.y(a.wat(n),n) + dywat;
                                                      % water particle 1-6 y path
            pathwat.z(a.wat(n)+1,n) = pathwat.z(a.wat(n),n) + dzwat;
                                                      % water particle 1-6 z path
            a.wat(n) = a.wat(n)+1;          % increment water particle 1-6 count
          end                               % end water particles path if
        end                                 % end run & T=0.1 if
        wat.x(n) = wat.x(n) + dxwat;        % water x move
        wat.y(n) = wat.y(n) + dywat;        % water y move
        wat.z(n) = wat.z(n) + dzwat;        % water z move
        if ((rand()<sigma_a.wat/sigma_t.wat) && (wat.x(n)>0) &&
(wat.x(n)<T(width)))                                % water x section ratio
          is_absorb.wat = 1;                          % set absorb flag to 1
          tallyin.wat(run) = tallyin.wat(run) + 1;    % add 1 to tallyinwater
        end                                           % end absorb if
      end                                           % particle left slab or absorbed
      while ((is_absorb.lead==0 && (lead.x(n)>0) && (lead.x(o)<T(width)))
                                                    % still in slab not absorbed
        [xunit, yunit, zunit] = unitVector();       % call unitVector
        [dxlead, dylead, dzlead] = step(xunit, yunit, zunit, meanfree.lead);
                                                    % lead steps
        if ((T(width)==0.1) && (run==1))            % run 1 & T=0.1
          if ((n>0) && (n<7))                       % lead particle 1-6
            pathlead.x(a.lead(n)+1,n) = pathlead.x(a.lead(n),n) + dxlead;
                                                    % lead particle 1-6 x path
            pathlead.y(a.lead(n)+1,n) = pathlead.y(a.lead(n),n) + dylead;
                                                    % lead particle 1-6 y path
            pathlead.z(a.lead(n)+1,n) = pathlead.z(a.lead(n),n) + dzlead;
                                                    % lead particle 1-6 z path
            a.lead(n) = a.lead(n)+1;        % increment lead particle 1-6 count
          end                               % end lead particles path if
        end                                 % end run 1 & T=0.1 if
        lead.x(n) = lead.x(n) + dxlead;     % lead x move
        lead.y(n) = lead.y(n) + dylead;     % lead y move
        lead.z(n) = lead.z(n) + dzlead;     % lead z move
        if ((rand()<sigma_a.lead/sigma_t.lead) && (lead.x(n)>0) && (lead.x(n)<
                          T(width)))        % lead x section ratio
          is_absorb.lead = 1;               % set absorb flag to 1
          tallyin.lead(run) = tallyin.lead(run) + 1;        % add 1 to tallyinlead
        end                                 % end absorb lead if
      end                                   % particle left slab or absorbed
      while ((is_absorb.graph==0) && (graph.x(n)>0) && (graph.x(n)<T(width)))
                                                    % still in slab
        [xunit, yunit, zunit] = unitVector();       % call unitVector
        [dxgraph, dygraph, dzgraph] = step(xunit, yunit, zunit, meanfree.graph);
                                            % graphite steps
        if ((T(width)==0.1) && (run==1))            % run 1 & T=0.1
          if ((n>0) && (n<7))               % graphite particle 1-6
            pathgraph.x(a.graph(n)+1,n) = pathgraph.x(a.graph(n),n) + dxgraph;
                                            % graphite particle 1-6 x path
            pathgraph.y(a.graph(n)+1,n) = pathgraph.y(a.graph(n),n) + dygraph;
                                            % graphite particle 1-6 y path
            pathgraph.z(a.graph(n)+1,n) = pathgraph.z(a.graph(n),n) + dzgraph;
                                            % graphite particle 1-6 z path
            a.graph(n) = a.graph(n)+1;      % increment graphite particle 1-6 count
          end                               % end graphite particles path if
        end                                 % end run 1 & T=0.1 if
        graph.x(n) = graph.x(n) + dxgraph;          % graphite x move
        graph.y(n) = graph.y(n) + dygraph;          % graphite y move
        graph.z(n) = graph.z(n) + dzgraph;          % graphite z move
        if ((rand()<sigma_a.graph/sigma_t.graph) && (graph.x(n)>0) &&
graph.x(n)<T(width)))                       % graphite x section ratio
          is_absorb.graph = 1;                      % set absorb flag to 1
          tallyin.graph(run) = tallyin.graph(run) + 1;  % add 1 to tallyingraph
```

```
            end                                         % end absorb if
          end                                           % particle left slab or absorb
          if (wat.x(n)>T(width))                        % transmitted
            tallyout.wat(run) = tallyout.wat(run) + 1;        % add 1 to tallyoutwat
          end                                                 % end transmit if
          if (lead.x(n)>T(width))                             % lead transmitted
            tallyout.lead(run) = tallyout.lead(run) + 1;      % add 1 to tallyoutlead
          end                                                 % end lead transmitted if
          if (graph.x(n)>T(width))                            % transmitted
            tallyout.graph(run) = tallyout.graph(run) + 1;    % add 1 to tallyoutgraph
          end                                                 % end graphite transmit if
        end                                                   % end particle loop
        numreflect.wat(run) = particles - tallyin.wat(run) - tallyout.wat(run);
                                                              % water reflected
        numreflect.lead(run) = particles - tallyout.lead(run) - tallyin.lead(run);
                                                              % lead reflected
        numreflect.graph(run) = particles - tallyout.graph(run) - tallyin.graph(run);
                                                              % graphite reflected
      end                                                     % end water runs loop
      tallyinavg = struct('wat', {mean(tallyin.wat)}, 'lead', {mean(tallyin.lead)}, 'graph',
                                  {mean(tallyin.graph)});     % mean tallyin struct
      tallyoutavg = struct('wat', {mean(tallyout.wat)}, 'lead', {mean(tallyout.lead)},
                           'graph', {mean(tallyout.graph)});  % mean tallyout struct
      reflectavg = struct('wat', {mean(nreflect.wat)}, 'lead', {mean(nreflect.lead)},
                           'graph', {mean(nreflect.graph)});  % mean reflect struct
      errtallyin = struct('wat', {std(tallyin.wat)}, 'lead', {std(tallyin.lead)}, 'graph',
                                  {std(tallyin.graph)});      % error tallyin struct
      errtallyout = struct('wat', {std(tallyout.wat)}, 'lead', {std(tallyout.lead)}, 'graph',
                                  {std(tallyout.graph)});     % error tallyout struct
      errreflect = struct('wat', {std(nreflect.wat)}, 'lead', {std(nreflect.lead)}, 'graph',
                                  {std(nreflect.graph)});     % error reflect struct
      trans.wat(width) = tallyoutavg.wat/particles; % fraction transmitted water
      transerr.wat(width) = errtallyout.wat/particles;
                                                    % error fraction transmitted water
      trans.lead(width) = tallyoutavg.lead/particles; % fraction transmitted lead

      transerr.lead(width) = errtallyout.lead/particles;
                                                    % error fraction transmitted lead
      trans.graph(width) = tallyoutavg.graph/particles;
                                                    % fraction transmitted graphite
      transerr.graph(width) = errtallyout.graph/particles;
                                                    % error fraction transmitted graphite
      reflect.wat(width) = reflectavg.wat/particles;    % fraction reflected water
      reflecterr.wat(width) = errreflect.wat/particles;
                                                    % error fraction reflected water
      reflect.lead(width) = reflectavg.lead/particles;  % fraction reflected lead
```

```
reflecterr.lead(width) = errreflect.lead/particles;% error fraction reflected lead
reflect.graph(width) = reflectavg.graph/particles; % fraction reflected graphite

reflecterr.graph(width) = errreflect.graph/particles;
                                        % error fraction reflected graphite
absorb.wat(width) = tallyinavg.wat/particles;   % fraction absorbed water
absorberr.wat(width) = errtallyin.wat/particles;
                                        % error fraction absorbed water
absorb.lead(width) = tallyinavg.lead/particles; % fraction absorbed lead
absorberr.lead(width) = errtallyin.lead/particles;
                                        % error fraction absorbed lead
absorb.graph(width) = tallyinavg.graph/particles;
                                        % fraction absorbed graphite
absorberr.graph(width) = errtallyin.graph/particles;
                                        % error fraction absorbed graphite
% errors vary approximately by sqrt(#particles transmitted/absorbed/etc)
% error slightly less than this for large #
% can be slightly more for small #
if T(width) == 0.1;                     % if T=0.1
  minywat = min(wat.y);                 % min y water val
  maxywat = max(wat.y);                 % max y water val
  minzwat = min(wat.z);                 % min z water val
  maxzwat = max(wat.z);                 % max z water val
  minylead = min(lead.y);               % min lead y val
  maxylead = max(lead.y);               % max lead y val
  minzlead = min(lead.z);               % min lead z val
  maxzlead = max(lead.z);               % max lead z val
  minygraph = min(graph.y);             % min graphite y val
  maxygraph = max(graph.y);             % max graphite y val
  minzgraph = min(graph.z);             % min graphite z val
  maxzgraph = max(graph.z);             % max graphite z val
  wattrans = struct('x', wat.x(wat.x>0.1), 'y', wat.y(wat.x>0.1), 'z',
                        wat.z(wat.x>0.1));   % transmitted water struct
  watreflect = struct('x', wat.x(wat.x<0), 'y', wat.y(wat.x<0), 'z', wat.z(wat.x<0));
                                        % reflected water struct
  watabsorb = struct('x', wat.x(wat.x>0 & wat.x<0.1), 'y', wat.y(wat.x>0 &
    wat.x<0.1), 'z', wat.z(wat.x>0 & wat.x<0.1));     % absorbed water struct
  leadtrans = struct('x', lead.x(lead.x>0.1), 'y', lead.y(lead.x>0.1), 'z',
                        lead.z(lead.x>0.1));  % transmitted lead struct
  leadreflect = struct('x', lead.x(lead.x<0), 'y', lead.y(lead.x<0), 'z',
                        lead.z(lead.x<0));    % reflected lead struct
  leadabsorb = struct('x', lead.x(lead.x>0 & lead.x<0.1), 'y', lead.y(lead.x>0 &
    lead.x<0.1), 'z', lead.z(lead.x>0 & lead.x<0.1));   % absorbed lead struct
  graphtrans = struct('x', graph.x(graph.x>0.1), 'y', graph.y(graph.x>0.1), 'z',
                   graph.z(graph.x>0.1));     % transmitted graphite struct
```

```
        graphreflect = struct('x', graph.x(graph.x<0), 'y', graph.y(graph.x<0), 'z',
                                    graph.z(graph.x<0));          % reflected graphite struct
        graphabsorb = struct('x', graph.x(graph.x>0 & graph.x<0.1), 'y',
           graph.y(graph.x>0 & graph.x<0.1), 'z', graph.z(graph.x>0 & graph.x<0.1));
                                                                   % absorbed graphite struct
    end                                                            % end T=0.1 if
end                                                                % end different thicknesses
% ---------------------------------------------------------------------------------------------


% ---------------------------------------------------------------------------------------------
% plot particles
% water
figure;                                                            % open fig 14
hold on;                                                           % plot on same fig
plot3(wattrans.x, wattrans.y, wattrans.z, '.');                    % plot transmitted water
plot3(watreflect.x, watreflect.y, watreflect.z, '.');              % plot reflected water
plot3(watabsorb.x, watabsorb.y, watabsorb.z, '.');                 % plot absorbed water
xlabel('x (m)', ‘FontSize’, 12);                           % x axis label
ylabel('y (m)', ‘FontSize’, 12);                           % y axis label
zlabel('z (m)', ‘FontSize’, 12);                           % z axis label
set(gca, 'FontSize', 12);                                  % axis font size
xlim([min(watreflect.x) max([wattrans(:).x;watabsorb.x(:)])]);  % x axis limits
[yywat, zzwat] = meshgrid(minywat:(maxywat-minywat):maxywat, minzwat:
                                    (maxzwat-minzwat):maxzwat);     % water grid
surf(zeros(size(yywat)), yywat, zzwat);                    % x=0 edge
surf(0.1*ones(size(yywat)), yywat, zzwat);                 % other edge
text(0, minywat, minzwat, 'incident edge ', 'HorizontalAlignment', ‘Right’, 'Font
          Size', 11);                                      % label x=0 edge
text(0.1, minywat, minzwat, 'transmitted edge ', 'HorizontalAlignment', 'Right',
          'FontSize', 11);                                 % label other edge
text(0.04, minywat, minzwat, 'water', 'FontSize', 11); % label where water present
if isempty(wattrans.x);                                    % if no transmitted
      l = legend('reflected', 'absorbed');                 % label reflected & absorbed
else                                                       % some transmitted
      l = legend('transmitted', 'reflected', 'absorbed');          % label all plots
end                                                        % end empty if
set(l, ‘FontSize', 10);                                    % legend font size
hold off;                                                  % turn hold off
print('water particles', '-depsc');                        % save fig 14

% lead
figure;                                                            % open fig 15
hold on;                                                           % plot on same fig
plot3(leadtrans.x, leadtrans.y, leadtrans.z, '.');                 % plot transmitted lead
plot3(leadreflect.x, leadreflect.y, leadreflect.z, '.');           % plot reflected lead
```

```
plot3(leadabsorb.x, leadabsorb.y, leadabsorb.z, '.');       % plot absorbed lead
xlabel('x (m)', ‘FontSize’, 12);                             % x axis label
ylabel('y (m)', ‘FontSize’, 12);                             % y axis label
zlabel('z (m)', ‘FontSize’, 12);                             % z axis label
set(gca, 'FontSize', 12);                                    % axis font size
xlim([min(leadreflect.x) max(leadtrans.x)]);                 % x axis limits
[yylead, zzlead] = meshgrid(minylead:(maxylead-minylead):maxylead, minzlead:
                        (maxzlead-minzlead):maxzlead);   % lead grid
surf(zeros(size(yylead)), yylead, zzlead);                   % x=0 edge
surf(0.1*ones(size(yylead)), yylead, zzlead);                % other edge
text(0, minylead, minzlead, 'incident edge ', 'HorizontalAlignment', 'Right', 'Font
      Size', 11);                                            % label x=0 edge
text(0.1, minylead, minzlead, ' transmitted edge', 'FontSize', 11); % label other edge
l = legend('transmitted', 'reflected', 'absorbed');          % label plots
set(l, ‘FontSize', 10);                                      % legend font size
text(0.04, minylead, minzlead, 'lead', 'FontSize', 11);      % label where lead present
hold off;                                                    % turn hold off
print('lead particles', '-depsc');                           % save fig 15

% graphite
figure;                                                      % open fig 16
hold on;                                                     % plot on same fig
plot3(graphtrans.x, graphtrans.y, graphtrans.z, '.');        % plot transmitted graphite
plot3(graphreflect.x, graphreflect.y, graphreflect.z, '.');  % plot reflected graphite
plot3(graphabsorb.x, graphabsorb.y, graphabsorb.z, ‘.'); % plot absorbed graphite
xlabel('x (m)', ‘FontSize’, 12);                             % x axis label
ylabel('y (m)', ‘FontSize’, 12);                             % y axis label
zlabel('z (m)', ‘FontSize’, 12);                             % z axis label
set(gca, 'FontSize', 12);                                    % axis font size
xlim([min(graphreflect.x) max(graphtrans.x)]);               % x axis limits
[yygraphite, zzgraphite] = meshgrid(minygraph:(maxygraph-minygraph):maxy-
graph, minzgraph:(maxzgraph-minzgraph):maxzgraph);
                                                             % graphite grid
surf(zeros(size(yygraphite)), yygraphite, zzgraphite);       % x=0 edge
surf(0.1*ones(size(yygraphite)), yygraphite, zzgraphite);          % other edge
text(0, minygraph, minzgraph, 'incident edge ', 'HorizontalAlignment', 'Right',
      'FontSize', 11);                                       % label x=0 edge
text(0.1, minygraph, minzgraph, ' transmitted edge', 'FontSize', 11);
                                                     % label other edge
if isempty(graphabsorb.x);                           % none absorbed
    l = legend('transmitted', 'reflected');          % label transmitted & reflected
  else                                               % some absorbed
    l = legend('transmitted', 'reflected', 'absorbed');     % label all plots
set(l, ‘FontSize', 10);                              % legend font size
text(0.04, minygraph, minzgraph, 'graphite', 'FontSize', 11);
                                                     % label where graphite present
```

```
hold off;                                    % turn hold off
print('graphite particles', '-depsc');       % save fig 16
% ---------------------------------------------------------------------------------------------------


% ---------------------------------------------------------------------------------------------------
% 2nd set of runs for comparison
for width2 = 1:numwidth                      % loop over thicknesses
  tallyout2 = struct('wat', zeros(1, runs), 'lead', zeros(1, runs), 'graph', zeros(1,
runs));                                            % 2nd tallyout struct
  for run2 = 1:runs                                % loop runs
    wat = struct('x', [], 'y', zeros(1, particles), 'z', zeros(1, particles));
                                                   % water particles
    lead = struct('x', [], 'y', zeros(1, particles), 'z', zeros(1, particles));
                                                   % lead particles
    graph = struct('x', [], 'y', zeros(1, particles), 'z', zeros(1, particles));
                                                   % graphite particles
    for m = 1:particles                            % loop particles
      is_absorb = struct('wat', 0, 'lead', 0, 'graph', 0);  % set absorb flag to 0
      xstep1wat = step(1, 0, 0, meanfree.wat);     % water step 1
      wat.x(m) = abs(xstep1wat);                   % water x step 1
      xstep1lead = step(1, 0, 0, meanfree.lead);   % lead step 1
      lead.x(m) = abs(xstep1lead);                 % lead x step 1
      xstep1graph = step(1, 0, 0, meanfree.graph); % graphite step 1
      graph.x(m) = abs(xstep1graph);               % graphite x step 1
      while ((is_absorb.wat==0) && (wat.x(m)>0) && (wat.x(m)<T(width2)))
                                             % particle still in slab
        [xunit, yunit, zunit] = unitVector();  % call unitVector
        [dxwat, dywat, dzwat] = step(xunit, yunit, zunit, meanfree.wat);
                                             % water steps
        wat.x(m) = wat.x(m) + dxwat;         % water x move
        wat.y(m) = wat.y(m) + dywat;         % water y move
        wat.z(m) = wat.z(m) + dzwat;         % water z move
        if ((rand()<sigma_a.wat/sigma_t.wat) && (wat.x(m)>0) &&
            (wat.x(m)<T(width2);             % water x section ratio
          is_absorb.wat = 1;                 % set absorb flag to 1
        end                                  % end absorb if
      end                                    % particle left slab or absorbed
      while ((is_absorb.lead==0) && (lead.x(m)>0) && (lead.x(m)<T(width2)))
                                             % still in slab not absorbed
        [xunit, yunit, zunit] = unitVector();  % call unitVector
        [dxlead, dylead, dzlead] = step(xunit, yunit, zunit, meanfree.lead);
                                             % lead steps
        lead.x(m) = lead.x(m) + dxlead;      % lead x move
        lead.y(m) = lead.y(m) + dylead;      % lead y move
        lead.z(m) = lead.z(m) + dzlead;      % lead z move
```

```
            if ((rand()<sigma_a.lead/sigma_t.lead) && (lead.x(m)>0) && (lead.x(m)<
                      T(width2);              % lead x section ratio
              is_absorb.lead = 1;             % set absorb flag to 1
            end                               % end absorb lead if
          end                                 % particles left slab or absorbed
          while ((is_absorb.graph==0) && (graph.x(m)>0) &&
                (graph.x(m)<T(width2)))             % still in slab
            [xunit, yunit, zunit] = unitVector();   % call unitVector
            [dxgraph, dygraph, dzgraph] = step(xunit, yunit, zunit, meanfree.graph);
                                                    % graphite steps
            graph.x(m) = graph.x(m) + dxgraph;      % graphite x move
            graph.y(m) = graph.y(m) + dygraph;      % graphite y move
            graph.z(m) = graph.z(m) + dzgraph;      % graphite z move
            if ((rand()<sigma_a.graph/sigma_t.graph) && (graph.x(m)>0) &&
                (graph.x(m)<T(width2);        % graphite x section ratio
              is_absorb.graph = 1;            % set absorb flag to 1
            end                               % end absorb if
          end                                 % particle left slab or absorb
          if (wat.x(m)>T(width2))             % transmitted
            tallyout2.wat(run2) = tallyout2.wat(run2) + 1;  % add 1 to tallyoutwat
          end                                 % end transmit if
          if (lead.x(m)>T(width2))            % lead transmitted
            tallyout2.lead(run2) = tallyout2.lead(run2) + 1; % add 1 to tallyoutlead
          end                                 % end lead transmitted if
          if (graph.x(m)>T(width2))           % transmitted
            tallyout2.graph(run2) = tallyout2.graph(run2) + 1;
                                              % add 1 to tallyoutgraph
          end                                 % end transmitted if
        end                                   % end particle loop
      end                                     % end runs
      tallyoutavg2 = struct('wat', {mean(tallyout2.wat)}, 'lead', {mean(tallyout2.lead)},
                'graph', {mean(tallyout2.graph)});          % mean tallyout2 struct
      errtallyout2 = struct('wat', {std(tallyout2.wat)}, 'lead', {std(tallyout2.lead)},
                      'graph', {std(tallyout2.graph)});     % error tallyout2 struct
      trans2.wat(width2) = tallyoutavg2.wat/particles; % fraction transmitted water
      transerr2.wat(width2) = errtallyout2.wat/particles;
                                              % error fraction transmitted water
      trans2.lead(width2) = tallyoutavg2.lead/particles;
                                              % fraction transmitted lead
      transerr2.lead(width2) = errtallyout2.lead/particles;
                                              % error fraction transmitted lead
      trans2.graph(width2) = tallyoutavg2.graph/particles;
                                              % fraction transmitted graphite
      transerr2.graph(width2) = errtallyout2.graph/particles;
                                              % error fraction transmitted graphite
    end                                       % end thickness loop
```

```
% -------------------------------------------------------------------------------------------------------
% Woodcock random walks
pathleadgraph = struct('x', zeros(1,6), 'y', zeros(2,6), 'z', zeros(2,6));
                                                                   % leadgraph paths
pathleadwat = struct('x', zeros(1,6), 'y', zeros(2,6), 'z', zeros(2,6));
                                                                   % leadwat paths
pathgraphlead = struct('x', zeros(1,6), 'y', zeros(2,6), 'z', zeros(2,6));
                                                                   % graphlead paths
pathgraphwat = struct('x', zeros(1,6), 'y', zeros(2,6), 'z', zeros(2,6));
                                                                   % graphwat paths
pathwatgraph = struct('x', zeros(1,6), 'y', zeros(2,6), 'z', zeros(2,6));
                                                                   % watgraph paths
pathwatlead = struct('x', zeros(1,6), 'y', zeros(2,6), 'z', zeros(2,6));
                                                                   % watlead paths
tallyoutWood = struct('leadgraph', zeros(1, runs), 'leadwat', zeros(1, runs),
             'graphlead', zeros(1, runs), 'graphwat', zeros(1, runs), 'watgraph', ze
             ros(1,runs), 'watlead', zeros(1, runs));      % Woodcock tallyout struct
for runWood = 1:runs                                       % Woodcock runs
  leadgraph = struct('x', [], 'y', zeros(1, particles), 'z', zeros(1, particles));
                                                           % lead graphite particles
  leadwat = struct('x', [], 'y', zeros(1, particles), 'z', zeros(1, particles));
                                                           % lead water particles
  graphlead = struct('x', [], 'y', zeros(1, particles), 'z', zeros(1, particles));
                                                           % graphite lead particles
  graphwat = struct('x', [], 'y', zeros(1, particles), 'z', zeros(1, particles));
                                                           % graphite water particles
  watgraph = struct('x', [], 'y', zeros(1, particles), 'z', zeros(1, particles));
                                                           % water graphite particles
  watlead = struct('x', [], 'y', zeros(1, particles), 'z', zeros(1, particles));
                                                           % water lead particles
 for q = 1:particles                                       % particles loop
    is_absorbedWood = struct('leadgraph', 0, 'leadwat', 0, 'graphlead', 0, 'graph
             wat', 0, 'watgraph', 0, 'watlead', 0);        % set absorb flag to 0
    xstep1lead = steps(1, 0, 0, meanfree.lead);            % lead graphite step 1
    leadgraph.x(q) = abs(xstep1lead);                      % lead graphite x step 1
    xstep1leadwat = step(1, 0, 0, meanfree.lead);          % lead water step 1
    leadwat.x(q) = abs(xstep1leadwat);                     % lead water x step 1
    xstep1graph = step(1, 0, 0, meanfree.graph);           % graphite lead step 1
    graphlead.x(q) = abs(xstep1graph);                     % graphite lead x step 1
    xstep1graphwat = step(1, 0, 0, meanfree.graph);        % graphite water step 1
    graphwat.x(q) = abs(xstep1graphwat);                   % graphite water x step 1
    xstep1wat = steps(1, 0, 0, meanfree.wat);              % water graphite step 1
    watgraph.x(q) = abs(xstep1wat);                        % water graphite x step 1
    xstep1watlead = steps(1, 0, 0, meanfree.wat);          % water lead step 1
    watlead.x(q) = abs(xstep1watlead);                     % water lead x step 1
    if (runWood==1)                                        % run 1
```

```
        if ((q>0 && (q<7))                              % particle 1-6
          pathleadgraph.x(2,q) = abs(xstep1lead); % lead graphite particle 1-6 step 1
          pathleadwat.x(2,q) = abs(xstep1leadwat); % lead water particle 1-6 step 1
          pathgraphlead.x(2,q) = abs(xstep1graph);
                                              % graphite lead particle 1-6 step 1
          pathgraphwat.x(2,q) = abs(xstep1graphwat);
                                              % graphite water particle 1-6 step 1
          pathwatgraph.x(2,q) = abs(xstep1wat);
                                              % water graphite partocle 1-6 step 1
          pathwatlead.x(2,q) = abs(xstep1watlead); % water lead particle 1-6 step 1
        end                                         % end particles if
      end                                           % end run 1 if
      while ((is_absorbedWood.leadgraph==0) && (leadgraph.x(q)>0) &&
                    (leadgraph.x(q)<0.2))           % still in lead graphite slab
          [xunit, yunit, zunit] = unitVector();     % call unitVector
          [dxgraph, dygraph, dzgraph] = step(xunit, yunit, zunit, meanfree.graph);
                                                    % graphite steps
          if (runWood==1)                           % run 1
            if ((q>0 && (q<7))                              % lead graphite particle 1-6
              pathleadgraph.x(a.leadgraph(q)+1,q) =
pathleadgraph.x(a.leadgraph(q),q) + dxgraph;   % lead graphite particle 1-6 x path
              pathleadgraph.y(a.leadgraph(q)+1,q) =
pathleadgraph.y(a.leadgraph(q),q) + dygraph;   % lead graphite particle 1-6 y path
              pathleadgraph.z(a.leadgraph(q)+1,q) =
pathleadgraph.z(a.leadgraph(q),q) + dzgraph;   % lead graphite particle 1-6 z path
              a.leadgraph(q) = a.leadgraph(q)+1;
                                          % increment lead graphite particle 1-6 count
            end                                 % end lead graphite particles path if
          end                                   % end lead graphite run 1 if

          leadgraph.x(q) = leadgraph.x(q) + dxgraph;
                                              % lead graphite particles x path
          leadgraph.y(q) = leadgraph.y(q) + dygraph;
                                              % lead graphite particles y path
          leadgraph.z(q) = leadgraph.z(q) + dzgraph;
                                              % lead graphite particles z path
          while ((rand()>meanfree.graph/meanfree.lead) && (leadgraph.x(q)>0) &&
                    (leadgraph.x(q)<0.1))           % fictitious lead graphite step
            [dxgraph, dygraph, dzgraph] = step(xunit, yunit, zunit, meanfree.graph);
                                                    % graphite steps
            if (runWood==1))                        % lead graphite run 1
              if ((q>0 && (q<7))                    % lead graphite particle 1-6
                pathleadgraph.x(a.leadgraph(q)+1,q) = pathleadgraph.x(a.lead
                    graph(q),q) + dxgraph;          % lead graphite particle 1-6 x path
                pathleadgraph.y(a.leadgraph(q)+1,q) = pathleadgraph.y(a.lead
                    graph(q),q) + dygraph;          % lead graphite particle 1-6 y path
                pathleadgraph.z(a.leadgraph(q)+1,q) = pathleadgraph.z(a.lead
                    graph(q),q) + dzgraph;        % lead graphite particle 1-6 z path
                a.leadgraph(q) = a.leadgraph(q)+1;
                                            % increment lead graphite particle 1-6 count
             end                            % end lead graphite particles path if
          end                                     % end lead graphite run 1 if
          leadgraph.x(q) = leadgraph.x(q) + dxgraph;
                                                  % lead graphite x path
          leadgraph.y(q) = leadgraph.y(q) + dygraph;
                                                  % lead graphite y path
          leadgraph.z(q) = leadgraph.z(q) + dzgraph;
                                                  % lead graphite z path
       end                                        % end fictitious lead graphite loop
       if ((rand()<sigma_a.lead/sigma_t.lead) && (leadgraph.x(q)>0) &&
                    (leadgraph.x(q)<0.1))         % graphite x section ratio
          is_absorbedWood.leadgraph = 1;          % set absorb flag to 1
       elseif ((rand()<sigma_a.graph/sigma_t.graph) && (leadgraph.x(q)>0.1)
                    &&(leadgraph.x(q)<0.2))       % graphite x section ratio
          is_absorbedWood.leadgraph = 1;          % set absorb flag to 1
       end                                        % end graphite absorb if
    end                             % left lead graphite slab or absorbed
    while ((is_absorbedWood.leadwat==0) && (leadwat.x(q)>0) &&
                (leadwat.x(q)<0.2))               % still in lead water slab
       [xunit, yunit, zunit] = unitVector();      % call unitVector
       [dxwat, dywat, dzwat] = step(xunit, yunit, zunit, meanfree.wat);
                                                  % water steps
       if (runWood==1)                            % lead water run 1
          if ((q>0 && (q<7))                      % lead water particle 1-6
             pathleadwat.x(a.leadwat(q)+1,q) = pathleadwat.x(a.leadwat(q),q) +
                dxwat;                            % lead water particle 1-6 x path
             pathleadwat.y(a.leadwat(q)+1,q) = pathleadwat.y(a.leadwat(q),q) +
                dywat;                            % lead water particle 1-6 y path
             pathleadwat.z(a.leadwat(q)+1,q) = pathleadwat.z(a.leadwat(q),q) +
                dzwat;                            % lead water particle 1-6 z path
             a.leadwat(q) = a.leadwat(q)+1;% increment lead water particle 1-6 count
          end                                     % end lead water particles path if
       end                                        % end lead water run 1 if
       leadwat.x(q) = leadwat.x(q) + dxwat;       % lead water particles x path
       leadwat.y(q) = leadwat.y(q) + dywat;       % lead water particles y path
       leadwat.z(q) = leadwat.z(q) + dzwat;       % lead water particles z path
       while ((rand()>meanfree.wat/meanfree.lead) && (leadwat.x(q)>0) &&
                    (leadwat.x(q)<0.1))           % fictitious lead water step
          [dxwat, dywat, dzwat] = step(xunit, yunit, zunit, meanfree.wat);
                                                  % water steps
          if (runWood==1)                         % lead water run 1
             if ((q>0 && (q<7))                   % lead water particle 1-6
                pathleadwat.x(a.leadwat(q)+1,q) = pathleadwat.x(a.leadwat(q),q) +
                      dxwat;                    % lead water particle 1-6 x path
                pathleadwat.y(a.leadwat(q)+1,q) = pathleadwat.y(a.leadwat(q),q) +
                      dywat;                    % lead water particle 1-6 y path
                pathleadwat.z(a.leadwat(q)+1,q) = pathleadwat.z(a.leadwat(q),q) +
                      dzwat;                    % lead water particle 1-6 z path
                a.leadwat(1) = a.leadwat(1)+1;
                                          % increment lead water particle 1-6 count
              end                         % end lead water particles path if
            end                           % end lead water run 1 if
            leadwat.x(q) = leadwat.x(q) + dxwat;    % lead water particles x path
            leadwat.y(q) = leadwat.y(q) + dywat;    % lead water particles y path
            leadwat.z(q) = leadwat.z(q) + dzwat;    % lead water particles z path
          end                                       % end fictitious lead water loop
          if (rand()<sigma_a.lead/sigma_t.lead) && (leadwat.x(q)>0) && (lead
                wat.x(q)<0.1))                      % lead x section ratio
            is_absorbedWood.leadwat = 1;            % set absorb flag to 1
          elseif ((rand()<sigma_a.wat/sigma_t.wat) && (leadwat.x(q)>0.1) && (lead
                wat.x(q)< 0.2))                           % water x section ratio
            is_absorbedWood.leadwat = 1;            % set absorb flag to 1
          end                                       % end lead water absorb if
        end                                         % left lead water slab or absorbed
        while ((is_absorbedWood.graphlead==0) && (graphlead.x(q)>0) &&
                      (graphlead.x(q)<0.2))         % still in graphite lead slab
          [xunit, yunit, zunit] = unitVector();     % call unitVector
          [dxgraph, dygraph, dzgraph] = step(xunit, yunit, zunit, meanfree.graph);
                                                    % graphite steps
          if (runWood==1)                           % graphite lead run 1
            if ((q>0 && (q<7))                      % graphite lead particle 1-6
              pathgraphlead.x(a.graphlead(q)+1,q) =
pathgraphlead.x(a.graphlead(q),q) + dxgraph;        % graphite lead particle 1-6 x path
              pathgraphlead.y(a.graphlead(q)+1,q) =
pathgraphlead.y(a.graphlead(q),q) + dygraph;        % graphite lead particle 1-6 y path
              pathgraphlead.z(a.graphlead(q)+1,q) =
pathgraphlead.z(a.graphlead(q),q) + dzgraph;        % graphite lead particle 1-6 z path
              a.graphlead(q) = a.graphlead(q)+1;
                                    % increment graphite lead particle 1-6 count
            end                     % end graphite lead particles path if
          end                       % end graphite lead run 1 if
          graphlead.x(q) = graphlead.x(q) + dxgraph; % graphite lead particles x path
          graphlead.y(q) = graphlead.y(q) + dygraph; % graphite lead particles y path
          graphlead.z(q) = graphlead.z(q) + dzgraph; % graphite lead particles z path
          while (rand()>meanfree.graph/meanfree.lead) && (graphlead.x(q)>0.1) &&
                (graphlead.x(q)<0.2))         % fictitious graphite lead step
            [dxgraph, dygraph, dzgraph] = step(xunit, yunit, zunit, meanfree.graph);
                                              % graphite steps
                if (runWood==1)                      % graphite lead run 1
                   if ((q>0 && (q<7))                % graphite lead particle 1-6
                      pathgraphlead.x(a.graphlead(q)+1,q) =
pathgraphlead.x(a.graphlead(q),q) + dxgraph;    % graphite lead particle 1-6 x path
                      pathgraphlead.y(a.graphlead(q)+1,q) =
pathgraphlead.y(a.graphlead(q),q) + dygraph;    % graphite lead particle 1-6 y path
                      pathgraphlead.z(a.graphlead(q)+1,q) =
pathgraphlead.z(a.graphlead(q),q) + dzgraph;    % graphite lead particle 1-6 z path
                      a.graphlead(q) = a.graphlead(q)+1;
                                          % increment graphite lead particle 1-6 count
                  end                     % end graphite lead particles path if
                end                       % end graphite lead run 1 if
                graphlead.x(q) = graphlead.x(q) + dxgraph;
                                                     % graphite lead particles x path
                graphlead.y(q) = graphlead.y(q) + dygraph;
                                                     % graphite lead particles y path
                graphlead.z(q) = graphlead.z(q) + dzgraph;
                                                     % graphite lead particles z path
             end                                     % end fictitious graphite lead loop
             if ((rand()<sigma_a.lead/sigma_t.lead) && (leadgraph.x(q)>0.1) &&
                      (leadgraph.x(q)<0.2))          % lead x section ratio
                is_absorbedWood.leadgraph = 1;       % set absorb flag to 1
             elseif ((rand()<sigma_a.graph/sigma_t.graph) && (leadgraph.x(q)>0) &&
                      (leadgraph.x(q)<0.1))                    % graphite x section ratio
                is_absorbedWood.leadgraph = 1;       % set absorb flag to 1
             end                                     % end graphite lead absorb if
          end                                % left graphite lead slab or absorbed
          while ((is_absorbedWood.graphwat==0) && (graphwat.x(q)>0) &&
                           (graphwat.x(q)<0.2))      % still in slab
             [xunit, yunit, zunit] = unitVector();   % call unitVector
             [dxwat, dywat, dzwat] = step(xunit, yunit, zunit, meanfree.wat);
                                                     % water steps
             if (runWood==1)                         % graphite water run 1
                if ((q>0 && (q<7))                   % graphite water particle 1-6
                   pathgraphwat.x(a.graphwat(q)+1,q) = pathgraphwat.x(a.graphwat(q),q)
                           + dxwat;          % graphite water particle 1-6 x path
                   pathgraphwat.y(a.graphwat(q)+1,q) = pathgraphwat.y(a.graphwat(q),q)
                           + dywat;          % graphite water particle 1-6 y path
                   pathgraphwat.z(a.graphwat(q)+1,q) = pathgraphwat.z(a.graphwat(q),q)
                           + dzwat;          % graphite water particle 1-6 z path
                   a.graphwat(q) = a.graphwat(q)+1;
                                          % increment graphite water particle 1-6 count
               end                        % end graphite water particles path if
             end                          % end graphite water run 1 if
             graphwat.x(q) = graphwat.x(q) + dxwat;  % graphite water x move
             graphwat.y(q) = graphwat.y(q) + dywat;  % graphite water y move
```

```
            graphwat.z(q) = graphwat.z(q) + dzwat;    % graphite water z move
            while ((rand ()>meanfree.wat/meanfree.graph) && (graphwat.x(q)>0) &&
                        (graphwat.x(q)<0.1))          % fictitious graphite water steps
              [dxwat, dywat, dzwat] = step(xunit, yunit, zunit, meanfree.wat);
                                                      % water steps
              if (runWood==1)                         % graphite water run 1
                if ((q>0 && (q<7))                    % graphite water particle 1-6
                  pathgraphwat.x(a.graphwat(q)+1,q) =
      pathgraphwat.x(a.graphwat(q),q) + dxwat; % graphite water particle 1-6 x path
                  pathgraphwat.y(a.graphwat(q)+1,q) =
      pathgraphwat.y(a.graphwat(q),q) + dywat; % graphite water particle 1-6 y path
                  pathgraphwat(a.graphwat(q)+1,q) =
      pathgraphwat.z(a.graphwat(q),q) + dzwat; % graphite water particle 1-6 z path
                  a.graphwat(q) = a.graphwat(q)+1;
                                      % increment graphite water particle 1-6 count
                end                            % end graphite water particles path if
              end                              % end graphite water run 1 if
              graphwat.x(q) = graphwat.x(q) + dxwat;       % graphite water x move
              graphwat.y(q) = graphwat.y(q) + dywat;       % graphite water y move
              graphwat.z(q) = graphwat.z(q) + dzwat;       % graphite water z move
            end                                % end graphite water fictitious loop
            if ((rand()<sigma_a.graph/sigma_t.graph) && (graphwat.x(q)>0) &&
                        (graphwat.x(q)<0.1))                % graphite x section ratio
              is_absorbedWood.graphwat = 1;                 % set absorb flag to 1
            elseif ((rand()<sigma_a.wat/sigma_t.wat) && (graphwat.x(q)>0.1) &&
                        (graphwat.x(q)<0.2))                % water x section ratio
              is_absorbedWood.graphwat = 1;                 % set absorb flag to 1
            end                                             % end water absorb if
          end                                               % end graphwat while
          while ((is_absorbedWood.watgraph==0) && (watgraph.x(q)>0) &&
                        (watgraph.x(q)<0.2))                % still in slab
              [xunit, yunit, zunit] = unitVector();         % call unitVector
              [dxwat, dywat, dzwat] = step(xunit, yunit, zunit, meanfree.wat);
                                                            % water steps
              if (runWood==1)                               % water graphite run 1
                if ((q>0 && (q<7))                    % water graphite particle 1-6
                  pathwatgraph.x(a.watgraph(q)+1,q) =
      pathwatgraph.x(a.watgraph(q),q) + dxwat; % water graphite particle 1-6 x path
                  pathwatgraph.y(a.watgraph(q)+1,q) =
      pathwatgraph.y(a.watgraph(q),q) + dywat; % water graphite particle 1-6 y path
                  pathwatgraph.z(a.watgraph(q)+1,q) =
      pathwatgraph.z(a.watgraph(q),q) + dzwat; % water graphite particle 1-6 z path
                  a.watgraph(q) = a.watgraph(q)+1;
                                      % increment water graphite particle 1 count
               end                    % end water graphite particles path if
              end                     % end water graphite run 1 if
              watgraph.x(q) = watgraph.x(q) + dxwat;          % water graphite x move
              watgraph.y(q) = watgraph.y(q) + dywat;          % water graphite y move
              watgraph.z(q) = watgraph.z(q) + dzwat;          % water graphite z move
              while ((rand ()>meanfree.wat/meanfree.graph) && (watgraph.x(q)>0.1)
      && (watgraph.x(q)<0.2))                         % fictitious water graphite steps
                [dxwat, dywat, dzwat] = steps(xunit, yunit, zunit, meanfree.wat);
                                                      % water steps
                if (runWood==1)                     % water graphite run 1
                  if ((q>0 && (q<7))                % water graphite particle 1-6
                    pathwatgraph.x(a.watgraph(q)+1,q) =
      pathwatgraph.x(a.watgraph(q),q) + dxwat;        % water graphite particle 1-6 x path
                    pathwatgraph.y(a.watgraph(q)+1,q) =
      pathwatgraph.y(a.watgraph(q),q) + dywat;        % water graphite particle 1-6 y path
                    pathwatgraph.z(a.watgraph(q)+1,q) =
      pathwatgraph.z(a.watgraph(q),q) + dzwat;        % water graphite particle 1-6 z path
                    a.watgraph(q) = a.watgraph(q)+1;
                                      % increment water graphite particle 1-6 count
                  end                   % end water graphite particles path if
                end                     % end water graphite run 1 if
                watgraph.x(q) = watgraph.x(q) + dxwat;      % water graphite x move
                watgraph.y(q) = watgraph.y(q) + dywat;      % water graphite y move
                watgraph.z(q) = watgraph.z(q) + dzwat;      % water graphite z move
              end                                   % end water graphite fictitious loop
              if ((rand()<sigma_a.graph/sigma_t.graph) && (watgraph.x(q)>0.1) &&
                  (watgraph.x(q)<0.2))              % graphite x section ratio
                is_absorbedWood.watgraph = 1;       % set absorb flag to 1
              elseif (rand()<sigma_a.wat/sigma_t.wat) && (watgraph.x(q)>0) &&
                                  (watgraph.x(q)<0.1))          % water x section ratio
                is_absorbedWood.watgraph = 1;       % set absorb flag to 1
              end                                   % end water graphite absorb if
          end                                       % left water graphite slab or absorbed
          while ((is_absorbedWood.watlead==0) && (watlead.x(q)>0) &&
                  (watlead.x(q)<0.2))                        % still in slab
              [xunit, yunit, zunit] = unitVector();         % call unitVector
              [dxwat, dywat, dzwat] = step(xunit, yunit, zunit, meanfree.wat);
                                                      % water steps
              if (runWood==1)                       % water lead run 1
                if ((q>0 && (q<7))                  % water lead particle 1-6
                  pathwatlead.x(a.watlead(q)+1,q) = pathwatlead.x(a.watlead(q),q) +
                        dxwat;                      % water lead particle 1-6 x path
                  pathwatlead.y(a.watlead(q)+1,q) = pathwatlead.y(a.watlead(q),q) +
                        dywat;                      % water lead particle 1-6 y path
                  pathwatlead.z(a.watlead(q)+1,q) = pathwatlead.z(a.watlead(q),q) +
                        dzwat;                      % water lead particle 1-6 z path
                  a.watlead(q) = a.watlead(q)+1;
                                                      % increment water lead particle 1-6 count
                end                                   % end water lead particles path if
              end                                     % end water lead run 1 if
              watlead.x(q) = watlead.x(q) + dxwat;    % water lead x move
              watlead.y(q) = watlead.y(q) + dywat;    % water lead y move
              watlead.z(q) = watlead.z(q) + dzwat;    % water lead z move
              while ((rand()>meanfree.wat/meanfree.lead) && (watlead.x(q)>0.1) &&
                        (watlead.x(q)<0.2))           % fictitious water lead steps
                [dxwat, dywat, dzwat] = steps(xunit, yunit, zunit, meanfree.wat);
                                                      % water steps
                if (runWood==1)                       % water lead run 1-6
                  if ((q>0 && (q<7))                  % water lead particle 1-6
                    pathwatlead.x(a.watlead(q)+1,q) = pathwatlead.x(a.watlead(q),q) +
                        dxwat;                        % water lead particle 1-6 x path
                    pathwatlead.y(a.watlead(q)+1,q) = pathwatlead.y(a.watlead(q),q) +
                        dywat;                        % water lead particle 1-6 y path
                    pathwatlead.z(a.watlead(q)+1,q) = pathwatlead.z(a.watlead(q),q) +
                        dzwat;                        % water lead particle 1-6 z path
                    a.watlead(q) = a.watlead(q)+1;
                                                      % increment water lead particle 1-6 count
                       end                            % end water lead particles path if
                end                             % end water lead run 1 if
                watlead.x(q) = watlead.x(q) + dxwat; % water lead x move
                watlead.y(q) = watlead.y(q) + dywat; % water lead y move
                watlead.z(q) = watlead.z(q) + dzwat; % water lead z move
              end                               % end fictitious water lead loop
              if ((rand()<sigma_a.lead/sigma_t.lead) && (watlead.x(q)>0.1) &&
                        (watlead.x(q)<0.2))           % lead x section ratio
                is_absorbedWood.watlead = 1;          % set absorb flag to 1
              elseif ((rand()<sigma_a.wat/sigma_t.wat) && (watlead.x(q)>0) &&
                        (watlead.x(q)<0.1))           % water x section ratio
                is_absorbedWood.watlead = 1;          % set absorb flag to 1
              end                               % end water lead absorb if
          end                                   % left water lead slab or absorbed
          if (leadgraph.x(q)>0.2)               % transmitted lead graphite
            tallyout.leadgraph(runWood) = tallyout.leadgraph(runWood) + 1;
                                                      % add 1 to tallyoutleadgraph
          end                                   % end transmitted lead graphite if
          if (leadwat.x(q)>0.2)                 % transmitted lead water
            tallyoutWood.leadwat(runWood) = tallyoutWood.leadwat(runWood) + 1;
                                                      % add 1 to tallyoutleadwat
          end                                   % end transmitted lead water if
          if (graphlead.x(q)>0.2)               % transmitted graphite lead
            tallyoutWood.graphlead(runWood) = tallyoutWood.graphlead(runWood) + 1;
                                                      % add 1 to tallyoutgraphlead
          end                                   % end transmitted graphite lead if
          if (graphwat.x(q)>0.2)                % transmitted graphite water
```

```
         tallyoutWood.graphwat(runWood) = tallyoutWood.graphwat(runWood) + 1;
                                                % add 1 to tallyoutgraphwat
      end                                       % end transmitted graphite water if
      if (watgraph.x(q)>0.2)                    % transmitted water graphite
         tallyout.watgraph(runWood) = tallyout.watgraph(runWood) + 1;
                                                % add 1 to talloutwatgraph
      end                                       % end transmitted water graphite if
     if (watlead.x(q)>0.2)                      % transmitted water lead
         tallyout.watlead(runWood) = tallyout.watlead(runWood) + 1;
                                                % add 1 to tallyoutwatlead
      end                                       % end transmitted water lead if
   end                                          % end particles
end                                             % end runs
% ---------------------------------------------------------------------------------------------------------


% ---------------------------------------------------------------------------------------------------------
% Woodcock calculations
tallyoutavgWood = struct('leadgraph', {mean(tallyoutWood.leadgraph)}, 'leadwat',
         {mean(tallyoutWood.leadwat)}, 'graphlead', {mean(tallyoutWood.
         graphlead)}, 'graphwat', {mean(tallyoutWood.graphwat)}, 'watgraph',
         {mean(tallyoutWood.watgraph)}, 'watlead', {mean(tallyoutWood.watlead)});
                                   % Woodcock mean tallyout struct
errtallyoutWood = struct('leadgraph', {std(tallyoutWood.leadgraph)}, 'leadwat',
{std(tallyoutWood.leadwat)}, 'graphlead', {std(tallyoutWood.graphlead)}, 'graph-
wat', {std(tallyoutWood.graphwat)}, 'watgraph', {std(tallyoutWood.watgraph)},
'watlead', {std(tallyoutWood.watlead)});    % Woodcock error tallyout struct
syms h;                            % sum
leadtransgraphtrans =
double(trans.graph(T==0.1)*trans.lead(T==0.1)*symsum((reflect.graph(T==0.1)*ref
lect.lead(T==0.1))^h, 0, inf)); % trasnmission by 0.1m graphite & 0.1m lead
errleadtransgraphtrans = double(symsum(sqrt(transerr.graph(T==0.1)^2 + transer
r.lead(T==0.1)^2 + h*(reflecterr.graph(T==0.1)^2 + reflecterr.lead(T==0.1)^2)), 0,
         1));                      % error transmission by 0.1m graphite & 0.1m lead
wattransgraphtrans = double(trans.graph(T==0.1)*trans.wat(T==0.1)*symsum((re
                                   flect.graph(T==0.1)*reflect.wat(T==0.1))^h, 0, inf));
                                   % transmission by 0.1m water & 0.1m graphite
errwattransgraphtrans = double(symsum(sqrt(transerr.graph(T==0.1)^2 + transer-
r.wat(T==0.1)^2 + h*(reflecterr.graph(T==0.1)^2 + reflecterr.wat(T==0.1)^2)), 0,
1));                               % error transmission by 0.1m water & 0.1m graphite
wattransleadtrans = double(trans.lead(end)*trans.wat(T==0.1)*symsum((reflec
                                   t.lead(T==0.1)*reflect.wat(T==0.1))^h, 0, inf));
                                   % tranmission by 0.1m water & 0.1m lead
errleadtranswattrans = double(symsum(sqrt(transerr.lead(T==0.1)^2 + transer-
r.wat(T==0.1)^2 + h*(reflecterr.lead(T==0.1)^2 + reflecterr.wat(T==0.1)^2)), 0, 1));
                                   % error transmission by 0.1m water & 0.1m lead
% ------------------------------------------------------------------------------------------------
% plot lead graphite particles
leadgraphtransmit = tallyoutavgWood.leadgraph/particles;
                                                % fraction lead graphite transmitted
leadgraphtranserr = errtallyoutWood.leadgraph/particles;
                                                % error fraction lead graphite transmitted
minyleadgraph = min(leadgraph.y);                   % min lead graphite y val
maxyleadgraph = max(leadgraph.y);                   % max lead graphite y val
minzleadgraph = min(leadgraph.z);                   % min lead graphite z val
maxzleadgraph = max(leadgraph.z);                   % max lead graphite z val
figure;                                             % open fig 17
xlim([min(leadgraph.x) max(leadgraph.x)]);          % x axis limits
leadgraphtrans = struct('x', leadgraph.x(leadgraph.x>0.2), 'y', leadgraph.y(lead-
graph.x>0.2), 'z', leadgraph.z(leadgraph.x>0.2)); % lead graphite transmitted
leadgraphreflect = struct('x', leadgraph.x(leadgraph.x<0), 'y', leadgraph.y(lead
graph.x<0), 'z', leadgraph.z(leadgraph.x<0));             % lead graphite reflected
leadgraphabsorb = struct('x', leadgraph.x(leadgraph.x<0.1 & leadgraph.x>0), 'y',
leadgraph.y(leadgraph.x<0.1 & leadgraph.x>0), 'z', leadgraph.z(leadgraph.x<0.1 &
leadgraph.x>0));                                % lead graphite absorbed in lead
leadgraph = struct('x', leadgraph.x(leadgraph.x>0.1 & leadgraph.x<0.2), 'y', lead-
graph.y(leadgraph.x>0.1 & leadgraph.x<0.2), 'z', leadgraph.z(leadgraph.x>0.1 &
leadgraph.x<0.2));                              % lead graphite absorbed in graphite
hold on;                                        % plot on same fig
plot3(leadgraphtrans.x, leadgraphtrans.y, leadgraphtrans.z, '.');
                                                % plot transmitted lead graphite
plot3(leadgraphreflect.x, leadgraphreflect.y, leadgraphreflect.z, '.');
                                                % plot reflected lead graphite
plot3(leadgraph.x, leadgraph.y, leadgraph.z, '.');
                                                % plot graphite absorbed lead graphite particles
plot3(leadgraphabsorb.x, leadgraphabsorb.y, leadgraphabsorb.z, '.');
                                                % plot lead absorbed lead graphite particles
xlabel('x (m)', ‘FontSize’, 12);                % x axis label
ylabel('y (m)', ‘FontSize’, 12);                % y axis label
zlabel('z (m)', ‘FontSize’, 12);                % z axis label
set(gca, 'FontSize', 12);                       % axis font size
[yyleadgraph, zzleadgraph] = meshgrid(minyleadgraph:(maxyleadgraph-minylead
          graph):maxyleadgraph, minzleadgraph:(maxzleadgraph-
          minzleadgraph):maxzleadgraph);                       % lead graphite grid
surf(zeros(size(yyleadgraph)), yyleadgraph, zzleadgraph);      % x=0 edge
surf(0.1*ones(size(yyleadgraph)), yyleadgraph, zzleadgraph); % material boundary
surf(0.2*ones(size(yyleadgraph)), yyleadgraph, zzleadgraph);   % other edge
text(0, minyleadgraph, minzleadgraph, 'incident edge ', 'HorizontalAlignment',
          'Right', 'FontSize', 11);                            % label x=0 edge
text(0.05, maxyleadgraph, maxzleadgraph, 'material boundary', 'FontSize', 11);
                                      % label material boundary
```

```
text(0.2, minyleadgraph, minzleadgraph, ' transmitted edge', 'FontSize', 11);
                                   % label other edge
text(0.04, minyleadgraph, minzleadgraph, 'lead', 'FontSize', 11);
                                   % label where lead present
text(0.13, minyleadgraph, minzleadgraph, 'graphite', 'FontSize', 11);
                                   % label where graphite present
if isempty(leadgraph.x)            % none absorbed in graphite
   l = legend('transmitted', 'reflected', 'absorbed in lead');
                                   % label transmitted, reflected, absorbed in lead
else                               % some absorbed in graphite
   l = legend('transmitted', 'reflected', 'absorbed in graphite', 'absorbed in lead');
                                                 % label all plots
end                                              % end absorbed in graphite if
set(l, ‘FontSize', 10);                          % legend font size
hold off;                                        % turn hold off
print('lead graphite particles', '-depsc');      % save fig 17
% ----------------------------------------------------------------------------------------------


% ----------------------------------------------------------------------------------------------
% plot lead water particles
leadwattransmit = tallyoutavgWood.leadwat/particles;
                                                 % fraction lead water transmitted
leadwattranserr = errtallyoutWood.leadwat/particles;
                                                 % error fraction lead water transmitted
minyleadwat = min(leadwat.y);                    % min lead water y val
maxyleadwat = max(leadwat.y);                    % max lead water y val
minzleadwat = min(leadwat.z);                    % min lead water z val
maxzleadwat = max(leadwat.z);                    % max lead water z val
figure;                                          % open fig 18
xlim([min(leadwat.x) max(leadwat.x)]);           % x axis limits
leadwattrans = struct('x', leadwat.x(leadwat.x>0.2), 'y', leadwat.y(leadwat.x>0.2), 'z',
                                   leadwat.z(leadwat.x>0.2));   % lead water transmitted
leadwatreflect = struct('x', leadwat.x(leadwat.x<0), 'y', leadwat.y(leadwat.x<0), 'z',
                                   leadwat.z(leadwat.x<0));      % lead water reflected
leadwatabsorb = struct('x', leadwat.x(leadwat.x<0.1 & leadwat.x>0), 'y',
          leadwat.y(leadwat.x<0.1 & leadwat.x>0), 'z', leadwat.z(leadwat.x<0.1 &
                    leadwat.x>0));          % lead water absorbed in lead
leadwat = struct('x', leadwat.x(leadwat.x>0.1 & leadwat.x<0.2), 'y', leadwat.y(lead
          wat.x>0.1 & leadwat.x<0.2), 'z', leadwat.z(leadwat.x>0.1 & leadwat.x<0.2));
                                          % lead water absorbed in water
hold on;                                  % plot on same fig
plot3(leadwattrans.x, leadwattrans.y, leadwattrans.z, '.');
                                          % plot transmitted lead water
plot3(leadwatreflect.x, leadwatreflect.y, leadwatreflect.z, '.');
                                          % plot reflected lead water
```

```
plot3(leadwat.x, leadwat.y, leadwat.z, '.');
                                        % plot water absorbed lead water particles
plot3(leadwatabsorb.x, leadwatabsorb.y, leadwatabsorb.z, '.');
                                        % plot lead absorbed lead water particles
xlabel('x (m)', 'FontSize', 12);        % x axis label
ylabel('y (m)', 'FontSize', 12);        % y axis label
zlabel('z (m)', 'FontSize', 12);        % z axis label
set(gca, 'FontSize', 12);               % axis font size
[yyleadwat, zzleadwat] = meshgrid(minyleadwat:(maxyleadwat-
                minyleadwat):maxyleadwat, minzleadwat:(maxzleadwat-
                minzleadwat):maxzleadwat);                  % lead water grid
surf(zeros(size(yyleadwat)), yyleadwat, zzleadwat);         % x=0 edge
surf(0.1*ones(size(yyleadwat)), yyleadwat, zzleadwat);      % material boundary
surf(0.2*ones(size(yyleadwat)), yyleadwat, zzleadwat);      % other edge
text(0, minyleadwat, minzleadwat, 'incident edge ', 'HorizontalAlignment', 'Right',
        'FontSize', 11);                        % label x=0 edge
text(0.05, minyleadwat + 0.05, minzleadwat, 'material boundary', 'FontSize', 11);
                                                % label material boundary
text(0.2, minyleadwat, minzleadwat, {' transmitted';' edge'}, 'FontSize', 11);
                                                % label other edge
text(0.04, minyleadwat, minzleadwat, 'lead', 'FontSize', 11);
                                                % label where lead present
text(0.13, minyleadwat, minzleadwat, 'water', 'FontSize', 11);
                                                % label where water present
l = legend('transmitted', 'reflected', 'absorbed in water', 'absorbed in lead');
                                                % label plots
set(l, 'FontSize', 10);                         % legend font size
hold off;                                       % turn hold off
print('lead water particles', '-depsc');        % save fig 18
% ------------------------------------------------------------------------------------------------------


% ------------------------------------------------------------------------------------------------------
% plot graphite lead particles
graphleadtransmit = tallyoutavgWood.graphlead/particles;
                                        % fraction graphite lead transmitted
graphleadtranserr = errtallyoutWood.graphlead/particles;
                                        % error fraction graphite lead transmitted
minygraphlead = min(graphlead.y);                   % min graphite lead y val
maxygraphlead = max(graphlead.y);                   % max graphite lead y val
minzgraphlead = min(graphlead.z);                   % min graphite lead z val
maxzgraphlead = max(graphlead.z);                   % max graphite lead z val
figure;                                             % open fig 19
xlim([min(graphlead.x) max(graphlead.x)]);          % x axis limits
```

```
graphleadtrans = struct('x', graphlead.x(graphlead.x>0.2), 'y',
                    graphlead.y(graphlead.x>0.2), 'z', graphlead.z(graphlead.x>0.2));
                                                  % graphite lead transmitted
graphleadreflect = struct('x', graphlead.x(graphlead.x<0), 'y',
                    graphlead.y(graphlead.x<0), 'z', graphlead.z(graphlead.x<0));
                                                  % graphite lead reflected
graphleadabsorb = struct('x', graphlead.x(graphlead.x<0.1 & graphlead.x>0), 'y',
          graphlead.y(graphlead.x<0.1 & graphlead.x>0), 'z',
                    graphlead.z(graphlead.x<0.1 & graphlead.x>0));
                                        % graphite lead absorbed in graphite
graphlead = struct('x', graphlead.x(graphlead.x>0.1 & graphlead.x<0.2), 'y',
          graphlead.y(graphlead.x>0.1 & graphlead.x<0.2), 'z', graphlead.z(graphlead
                    .x>0.1 & graphlead.x<0.2));           % graphite lead absorbed in lead
hold on;                                          % plot on same fig
plot3(graphleadtrans.x, graphleadtrans.y, graphleadtrans.z, '.');
                                                  % plot transmitted graphite lead
plot3(graphleadreflect.x, graphleadreflect.y, graphleadreflect.z, '.');
                                        % plot reflected graphite lead
plot3(graphlead.x, graphlead.y, graphlead.z, '.');
                                        % plot graphite absorbed graphite lead particles
plot3(graphleadabsorb.x, graphleadabsorb.y, graphleadabsorb.z, '.');
                                        % plot lead absorbed graphite lead particles
xlabel('x (m)', 'FontSize', 12);        % x axis label
ylabel('y (m)', 'FontSize', 12);        % y axis label
zlabel('z (m)', 'FontSize', 12);        % z axis label
set(gca, 'FontSize', 12);               % axis font size
[yygraphlead, zzgraphlead] = meshgrid(minygraphlead:(maxygraphlead-miny
          graphlead):maxygraphlead, minzgraphlead:(maxzgraphlead-
                    minzgraphlead):maxzgraphlead);          % graphite lead grid
surf(zeros(size(yygraphlead)), yygraphlead, zzgraphlead);         % x=0 edge
surf(0.1*ones(size(yygraphlead)), yygraphlead, zzgraphlead); % material boundary
surf(0.2*ones(size(yygraphlead)), yygraphlead, zzgraphlead);   % other edge
text(0, minygraphlead, minzgraphlead, 'incident edge ', 'HorizontalAlignment',
          'Right', 'FontSize', 11);                         % label x=0 edge
text(0.05, minygraphlead + 0.05, minzgraphlead, 'material boundary', 'FontSize',
          11);                                        % label material boundary
text(0.2, minygraphlead, minzgraphlead, ' transmitted edge', 'FontSize', 11);
                                                      % label other edge
text(0.04, minygraphlead, minzgraphlead, 'graphite', 'FontSize', 11);
                                                  % label where graphite present
text(0.13, minygraphlead, minzgraphlead, 'lead', 'FontSize', 11);
                                                  % label where lead present
if (isempty(graphleadabsorb.x) && isempty(graphlead.x))         % no absorbed
   l = legend('transmitted', 'reflected');        % label transmitted & reflected
elseif isempty(graphleadabsorb.x)                 % no absorbed in graphite
```

```
   l = legend('transmitted', 'reflected', 'absorbed in lead');
                                          % label transmitted, reflected, absorbed in lead
elseif isempty(graphlead.x)               % no absorbed in lead
   l = legend('transmitted', 'reflected', 'absorbed in graphite');
                                    % label transmitted, reflected, absorbed in graphite
else                                % some absorbed in both
   l = legend('transmitted', 'reflected', 'absorbed in lead', 'absorbed in graphite');
                                                % label all plots
end                                             % end graphite lead absorb if
set(l, 'FontSize', 10);                         % legend font size
hold off;                                       % turn hold off
print('graphite lead particles', '-depsc');     % save fig 19
% ---------------------------------------------------------------------------------------------------------------


% ---------------------------------------------------------------------------------------------------------------
% plot graphite water particles
graphwattransmit = tallyoutavgWood.graphwat/particles;
                                          % fraction graphite water transmitted
graphwattranserr = errtallyoutWood.graphwat/particles;
                                          % error fraction graphite water transmitted
minygraphwat = min(graphwat.y);                 % min graphite water y val
maxygraphwat = max(graphwat.y);                 % max graphite water y val
minzgraphwat = min(graphwat.z);                 % min graphite water z val
maxzgraphwat = max(graphwat.z);                 % max graphite water z val
figure;                                         % open fig 20
xlim([min(graphwat.x) max(graphwat.x)]);              % x axis limits
graphwattrans = struct('x', graphwat.x(graphwat.x>0.2), 'y', graphwat.y(graph
          wat.x>0.2), 'z', graphwat.z(graphwat.x>0.2)); % graphite water transmitted
graphwatreflect = struct('x', graphwat.x(graphwat.x<0), 'y', graphwat.y(graph
          wat.x<0), 'z', graphwat.z(graphwat.x<0));          % graphite water reflected
graphwatabsorb = struct('x', graphwat.x(graphwat.x<0.1 & graphwat.x>0), 'y',
graphwat.y(graphwat.x<0.1 & graphwat.x>0), 'z', graphwat.z(graphwat.x<0.1 &
graphwat.x>0));        % graphite water absorbed in graphite
graphwat = struct('x', graphwat.x(graphwat.x>0.1 & graphwat.x<0.2), 'y', graph
          wat.y(graphwat.x>0.1 & graphwat.x<0.2), 'z', graphwat.z(graph
          wat.x>0.1 & graphwat.x<0.2));       % graphite water absorbed in water
hold on;                            % plot on same fig
plot3(graphwattrans.x, graphwattrans.y, graphwattrans.z, '.');
                                    % plot transmitted graphite water
plot3(graphwatreflect.x, graphwatreflect.y, graphwatreflect.z, '.');
                                    % plot reflected graphite water
plot3(graphwat.x, graphwat.y, graphwat.z, '.');
                                    % plot water absorbed graphite water particles
plot3(graphwatabsorb.x, graphwatabsorb.y, graphwatabsorb.z, '.');
                                    % plot graphite absorbed graphite water particles
```

```
xlabel('x (m)', 'FontSize', 12);        % x axis label
ylabel('y (m)', 'FontSize', 12);        % y axis label
zlabel('z (m)', 'FontSize', 12);        % z axis label
set(gca, 'FontSize', 12);               % axis font size
[yygraphwat, zzgraphwat] = meshgrid(minygraphwat:(maxygraphwat-minygraph
                    wat):maxygraphwat, minzgraphwat:(maxzgraphwat-
                    minzgraphwat):maxzgraphwat);            % graphite water grid
surf(zeros(size(yygraphwat)), yygraphwat, zzgraphwat);          % x=0 edge
surf(0.1*ones(size(yygraphwat)), yygraphwat, zzgraphwat); % material boundary
surf(0.2*ones(size(yygraphwat)), yygraphwat, zzgraphwat);     % other edge
text(0, minygraphwat, minzgraphwat, 'incident edge ', 'HorizontalAlignment',
          'Right', 'FontSize', 11);                         % label x=0 edge
text(0.05, minygraphwat + 0.05, minzgraphwat, 'material boundary', 'FontSize', 11);
                                                            % label material boundary
text(0.2, minygraphwat, minzgraphwat, {'transmitted';'edge'}, 'HorizontalAlign
          ment', 'Right', 'FontSize', 11);          % label other edge
text(0.04, minygraphwat, minzgraphwat, 'graphite', 'FontSize', 11);
                                                    % label where graphite present
text(0.13, minygraphwat, minzgraphwat, 'water', 'FontSize', 11);
                                                    % label where water present
if (isempty(graphwatabsorb.x) && isempty(graphwattrans.x))
                                                    % no absorbed graphite water
   l = legend(‘reflected’, ‘absorbed in water’);    % label transmitted & reflected
elseif isempty(graphwatabsorb.x)                    % no absorbed in graphite
   l = legend('transmitted', 'reflected', 'absorbed in water');
                                        % label transmitted, reflected, absorbed in water
elseif isempty(graphwattrans.x)         % no absorbed in water
   l = legend('reflected', ‘absorbed in water’, ’absorbed in graphite');
                                        % label transmitted, reflected, absorbed in water
else                                    % some absorbed in both
   l = legend('transmitted', 'reflected', 'absorbed in water', 'absorbed in graphite');
                                                    % label all plots
end                                                 % end graphite water absorb if
set(l, 'FontSize', 10);                             % legend font size
hold off;                                           % turn hold off
print('graphite water particles', '-depsc');        % save fig 20
% -----------------------------------------------------------------------------------------------


% -----------------------------------------------------------------------------------------------
% plot water graphite particles
watgraphtransmit = tallyoutavgWood.watgraph/particles;
                                        % fraction transmitted water graphite
watgraphtranserr = errtallyoutWood.watgraph/particles;
                                        % error fraction transmitted water graphite
minywatgraph = min(watgraph.y);                     % min y water graphite val
```

```
maxywatgraph = max(watgraph.y);                    % max y water graphite val
minzwatgraph = min([watgraph.z);                   % min z water graphite val
maxzwatgraph = max(watgraph.z);                    % max z water graphite val
figure;                                            % open fig 21
xlim([min(watgraph.x) max(watgraph.x)]);           % x axis limits
watgraphtrans = struct('x', watgraph.x(watgraph.x>0.2), 'y', watgraph.y(wat-
graph.x>0.2), 'z', watgraph.z(watgraph.x>0.2));  % water graphite transmitted
watgraphreflect = struct('x', watgraph.x(watgraph.x<0), 'y', watgraph.y(wat-
graph.x<0), 'z', watgraph.z(watgraph.x<0));        % water graphite reflected
watgraphabsorb = struct('x', watgraph.x(watgraph.x<0.1 & watgraph.x>0), 'y', wat
          graph.y(watgraph.x<0.1 & watgraph.x>0), 'z', watgraph.z(watgraph.x<0.1 &
                    watgraph.x>0));                % water graphite absorbed in water
watgraph = struct('x', watgraph.x(watgraph.x>0.1 & watgraph.x<0.2), 'y', wat
          graph.y(watgraph.x>0.1 & watgraph.x<0.2), 'z', watgraph.z(watgraph.x>0.1
                    & watgraph.x<0.2));            % water graphite absorbed in graphite
hold on;                                           % plot on same fig
plot3(watgraphtrans.x, watgraphtrans.y, watgraphtrans.z, '.');
                                          % plot transmitted water graphite
plot3(watgraphreflect.x, watgraphreflect.y, watgraphreflect.z, 'r.');
                                          % plot reflected water graphite
plot3(watgraph.x, watgraph.y, watgraph.z, 'g.');
                                          % plot water absorbed water graphite
plot3(watgraphabsorb.x, watgraphabsorb.y, watgraphabsorb.z, 'y.');
                                          % plot graphite absorbed water graphite
xlabel('x (m)', ‘FontSize’, 12);          % x axis label
ylabel('y (m)', ‘FontSize’, 12);          % y axis label
zlabel('z (m)', 'FontSize', 12);          % z axis label
set(gca, 'FontSize', 12);                 % axis font size
[yywatgraph, zzwatgraph] = meshgrid(minywatgraph:(maxywatgraph-minywat
          graph):maxywatgraph, minzwatgraph:(maxzwatgraph-
          minzwatgraph):maxzwatgraph);  % water graphite grid
surf(zeros(size(yywatgraph)), yywatgraph, zzwatgraph);          % x=0 edge
surf(0.1*ones(size(yywatgraph)), yywatgraph, zzwatgraph); % material boundary
surf(0.2*ones(size(yywatgraph)), yywatgraph, zzwatgraph);       % other edge
text(0, minywatgraph, minzwatgraph, 'incident edge ', 'HorizontalAlignment',
          'Right', 'FontSize', 11);       % label x=0 surface
text(0.14, minywatgraph, minzwatgraph, 'graphite', 'FontSize', 11);
                                          % label where water present
text(0.04, minywatgraph, minzwatgraph, 'water', 'FontSize', 11);
                                          % label where graphite present
if (isempty(watgraphtrans.x) && isempty(watgraph.x))            % none transmitted
   l = legend('reflected', 'absorbed in water');         % label reflected & absorbed
elseif isempty(watgraph.x);                              % no absorbed in graphite

   l = legend('transmitted', 'reflected', 'absorbed in water');
                                          % label transmitted, reflected, absorbed in water
   text(0.07, minywatgraph, minzwatgraph, 'material boundary', 'FontSize', 11);
                                             % label material boundary
   text(0.2, minywatgraph, minzwatgraph, 'transmitted edge ', 'HorizontalAlign
                  ment', 'Right', 'FontSize', 11);        % label other edge
elseif isempty(watgraphtrans.x)
                                             % no transmitted or absorbed in graphite
   l = legend('reflected', 'absorbed in graphite', 'absorbed in water');
                                             % label reflected, absorbed in water
   text(0.07, minywatgraph, minzwatgraph, 'material boundary', 'FontSize', 11);
                                             % label material boundary
else                                         % some transmitted & absorbed in graphite
   l = legend('transmitted', 'reflected', 'absorbed in graphite', 'absorbed in water');
                                             % label all plots
   text(0.2, minywatgraph, minzwatgraph, 'transmitted edge ', 'HorizontalAlign
                  ment', 'Right', 'FontSize', 11);        % label other edge
   text(0.07, minywatgraph, minzwatgraph, 'material boundary', 'FontSize', 11);
                                             % label material boundary
end                                          % end transmitted & absorbed in graphite if
set(l, 'FontSize', 10);                                 % legend font size
hold off;                                               % turn hold off
print('water graphite particles', '-depsc');            % save fig 21
% ------------------------------------------------------------------------------------------------


% ------------------------------------------------------------------------------------------------
% plot water lead particles
watleadtransmit = tallyoutavgWood.watlead/particles;
                                                   % fraction water lead transmitted
watleadtranserr = errtallyoutWood.watlead/particles;
                                                   % error fraction water lead transmitted
minywatlead = min(watlead.y);                      % min y water lead val
maxywatlead = max(watlead.y);                      % max y water lead val
minzwatlead = min(watlead.z);                      % min z water lead val
maxzwatlead = max(watlead.z);                      % max z water lead val
figure;                                            % open fig 22
xlim([min(watlead.x) max(watlead.x)]);             % x axis limits
watleadtrans = struct('x', watlead.x(watlead.x>0.2), 'y', watlead.y(watlead.x>0.2), 'z',
                               watlead.z(watlead.x>0.2));   % water lead transmitted
watleadreflect = struct('x', watlead.x(watlead.x<0), 'y', watlead.y(watlead.x<0), 'z',
                               watlead.z(watlead.x<0));     % water lead reflected
watleadabsorb = struct('x', watlead.x(watlead.x<0.1 & watlead.x>0), 'y',
                  watlead.y(watlead.x<0.1 & watlead.x>0), 'z', watlead.z(watlead.x<0.1
                       & watlead.x>0));              % water lead absorbed in water
watlead = struct('x', watlead.x(watlead.x>0.1 & watlead.x<0.2), 'y',
                  watlead.y(watlead.x>0.1 & watlead.x<0.2), 'z', watlead.z(watlead.x>0.1
                       & watlead.x<0.2));            % water lead absorbed in lead
```

```
hold on;                                          % plot on same fig
plot3(watleadtrans.x, watleadtrans.y, watleadtrans,.z '.');
                                                  % plot transmitted water lead
plot3(watleadreflect.x, watleadreflect.y, watleadreflect.z, '.');
                                                  % plot reflected water lead
plot3(watlead.x, watlead.y, watlead.z, '.');      % plot water absorbed water lead
plot3(watleadabsorb.x, watleadabsorb.y, watleadabsorb.z, '.');
                                                  % plot lead absorbed water lead
xlabel('x (m)', 'FontSize', 12);                  % x axis label
ylabel('y (m)', 'FontSize', 12);                  % y axis label
zlabel('z (m)', 'FontSize', 12);                  % z axis label
set(gca, 'FontSize', 12);                         % axis font size
[yywatlead, zzwatlead] = meshgrid(minywatlead:(maxywatlead-
                    minywatlead):maxywatlead, minzwatlead:(maxzwatlead-
                    minzwatlead):maxzwatlead);          % water lead grid
surf(zeros(size(yywatlead)), yywatlead, zzwatlead);     % x=0 edge
surf(0.1*ones(size(yywatlead)), yywatlead, zzwatlead);  % material boundary
surf(0.2*ones(size(yywatlead)), yywatlead, zzwatlead);  % other edge
text(0, minywatlead, minzwatlead, 'incident edge ', 'HorizontalAlignment', 'Right',
      'FontSize', 11);                      % label x=0 edge
text(0.14, minywatlead, minzwatlead, 'lead', 'FontSize', 11);
                                            % label where water present
text(0.04, minywatlead, minzwatlead, 'water', 'FontSize', 11);
                                            % label where lead present
if (isempty(watleadtrans.x) && isempty(watlead.x))
                                            % none transmitted & absorbed in lead
  l = legend('reflected', 'absorbed in water'); % label reflected & absorbed in water
elseif isempty(watlead.x);                  % no absorbed in lead
  l = legend('transmitted', 'reflected', 'absorbed in water');
                                            % label transmitted, reflected, absorbed in water
  text(0.07, minywatlead, minzwatlead, 'material boundary', 'FontSize', 11);
                                            % label material boundary
  text(0.2, minywatlead, minzwatlead, 'transmitted edge ', 'HorizontalAlignment',
            'Right');          % label other edge
elseif isempty(watleadtrans.x);             % none transmitted water lead
  l = legend('reflected', 'absorbed in lead', 'absorbed in water');
                               % label reflected, absorbed in lead, absorbed in water
  text(0.07, minywatlead, minzwatlead, 'material boundary', 'FontSize', 11);
                                            % label material boundary
else                           % some transmitted water lead
  l = legend('transmitted', 'reflected', 'absorbed in lead', 'absorbed in graphite');
                               % label all plots
  text(0.07, minywatlead, minzwatlead, 'material boundary', 'FontSize', 11);
                                            % label material boundary
  text(0.2, minywatlead, minzwatlead, 'transmitted edge ', 'HorizontalAlignment',
            'Right');          % label other edge
```

```
end                            % end water lead transmitted & absorbed in lead if
set(l, ‘FontSize', 10);                   % legend font size
hold off;                                 % turn hold off
print('water lead particles', '-depsc');  % save fig 22
% ---------------------------------------------------------------------------------------------


% ---------------------------------------------------------------------------------------------
% tranmission, absorption, reflection with thickness
% transmission
figure;                                               % open fig 23
errorbar(T, trans.wat, transerr.wat);                 % plot water transmission
xlabel('x (m)', 'FontSize', 12);                      % x axis label
ylabel('transmission fraction', 'FontSize', 12);      % y axis label
set(gca, 'FontSize', 12);                             % axis font size
hold on;                                              % plot on same fig
errorbar(T, trans.lead, transerr.lead, 'r');          % plot lead transmission
errorbar(T, trans.graph, transerr.graph, 'g');        % plot graphite transmission
l = legend('water', 'lead', 'graphite');              % label plots
set(l, ‘FontSize', 10);                               % legend font size
hold off;                                             % turn hold off
print('transmission', '-depsc');                      % save fig 23

% absorption
figure;                                                       % open fig 24
errorbar(T, absorb.wat, absorberr.wat);                       % plot water absorption
xlabel('x (m)', 'FontSize', 12);                              % x axis label
ylabel('absorption fraction', 'FontSize', 12);                % y axis label
set(gca, 'FontSize', 12);                                     % axis font size
hold on;                                                      % plot on same fig
errorbar(T, absorb.lead, absorberr.lead, 'r');                % plot lead absorption
errorbar(T, absorb.graph, absorberr.graph, 'g');              % plot graphite absorption
l = legend('water', 'lead', 'graphite');                      % label plots
set(l, ‘FontSize', 10);                                       % legend font size
hold off;                                                     % turn hold off
print('absorption', '-depsc');                                % save fig 24

% reflection
figure;                                               % open fig 25
errorbar(T, reflect.lead, reflecterr.lead);           % plot water reflection
xlabel('x (m)', 'FontSize', 12);                      % x axis label
ylabel('reflection fraction', 'FontSize', 12);        % y axis label
set(gca, 'FontSize', 12);                             % axis font size
hold on;                                              % plot on same fig
errorbar(T, reflect.lead, reflecterr.lead 'r');       % plot lead reflection
errorbar(T, reflect.graph, reflecterr.graph, 'g');    % plot graphite reflection
```

```
l = legend('water', 'lead', 'graphite');                % label plots
set(l, ‘FontSize', 10);                                 % legend font size
hold off;                                               % turn hold off
print('reflection', '-depsc');                          % save fig 25

% print statements
fprintf('Water %3.1f%% transmitted, %3.1f%% absorbed, %3.1f%% reflected\n',
trans.wat(T==0.1)*100, absorb.wat(T==0.1)*100, reflect.wat(T==0.1)*100);
                                                  % print water data
fprintf('Lead %3.1f%% transmitted, %3.1f%% absorbed, %3.1f%% reflected\n',
trans.lead(T==0.1)*100, absorb.lead(T==0.1)*100, reflect.lead(T==0.1)*100);
                                                        % print lead data
fprintf('Graphite %3.1f%% transmitted, %3.1f%% absorbed, %3.1f%% reflected
              \n', trans.graph(T==0.1)*100, absorb.graph(T==0.1)*100,
              reflect.graph(T==0.1)*100);               % print graphite data

% attenuation
trans.wat = trans.wat(trans.wat>0);                     % keep transwat if >0
transerr.wat = transerr.wat(trans.wat>0);  % keep transerrwat if transwat>0
T1wat = T(trans.wat>0);                                 % keep T if transwat>0
trans2.wat = trans2.wat(trans2.wat>0);                  % keep transwat2 if >0
transerr2.wat = transerr2.wat(trans2.wat>0);            % keep transerrwat2 if >0
T2wat = T(trans2.wat>0);                                % keep T if transwat2>0
trans.lead = trans.lead(trans.lead>0);                  % keep translead if >0
transerr.lead = transerr.lead(trans.lead>0);            % keep transerrlead if translead>0
T1lead = T(trans.lead>0);                               % keep T if translead>0
trans2.lead = trans2.lead(trans2.lead>0);               % keep translead2 if >0
transerr2.lead = transerr2.lead(trans2.lead>0);         % keep transerrlead2 if >0
T2lead = T(trans2.lead>0);                              % keep T if translead2>0
trans.graph = trans.graph(trans.graph>0);               % keep transgraph if >0
transerr.graph = transerr.graph(trans.graph>0);
                                                  % keep transerrgraph if transgraph>0
T1graph = T(trans.graph > 0);                     % keep T if transgraph>0
trans2.graph = trans2.graph(trans2.graph>0);                % keep transgraph2 if >0
transerr2.graph = transerr2.graph(trans2.graph>0);
                                                  % keep transerrgraph2 if >0
T2graph = T(trans2.wat > 0);                      % keep T if transgraph2>0
polywattrans1 = polyfitweighted(T1, log(trans.wat), 1, transerr.wat);
                                                  % water attenuation1 polyfit
polywattrans2 = polyfitweighted(T2, log(trans2.wat), 1, transerr2.wat);
                                                  % water attenuation2 polyfit
polyleadtrans1 = polyfitweighted(T, log(trans.lead), 1, transerr.lead);
                                                  % lead attenuation1 polyfit
polyleadtrans2 = polyfitweighted(T, log(trans2.lead), 1, transerr2.lead);
                                                  % lead attenuation2 polyfit
```

```
polygraphtrans1 = polyfitweighted(T, log(trans.graph), 1, transerr.graph);
                                                % graphite attenuation1 polyfit
polygraphtrans2 = polyfitweighted(T, log(trans2.graph), 1, transerr2.graph);
                                                % graphite attenuation2 polyfit
attenuate1 = struct('wat', -1/polywattrans1(1), 'lead', -1/polyleadtrans1(1), 'graph',
                    -1/polygraphtrans1(1));     % attenuate1 struct
attenuate2 = struct('wat', -1/polywattrans2(1), 'lead', -1/polyleadtrans2(1), 'graph',
                    -1/polygraphtrans2(1));     % attenuate2 struct
attenuate = struct('wat', (attenuate1.wat+attenuate2.wat)/2, 'lead', (attenuate1.lead
             +attenuate2.lead)/2, 'graph', (attenuate1.graph+attenuate2.graph)/2);
                                                % mean attenuate struct
errwatattenuate = abs(attenuate1.wat - attenuate2.wat);
                                                % error water attenuation length
errleadattenuate = abs(attenuate1.lead - attenuate2.lead);
                                                % error lead attenuation length
errgraphattenuate = abs(attenuate1.graph - attenuate2.graph);
                                                % error graphite attenuation length
figure;                                                 % open fig 26
xlabel('x (m)', 'FontSize', 12);                        % x axis label
ylabel('transmission fraction', 'FontSize', 12);        % y axis label
set(gca, 'FontSize', 12);                               % axis font size
semilogy(T1wat, exp(polyval(pwattrans1, T1wat)));       % plot water vector
hold on;                                                % plot on same fig
semilogy(T1lead, exp(polyval(pleadtrans1, T1lead)));    % plot lead vector
semilogy(T1graph, exp(polyval(pgraphtrans1, T1graph)), ‘y’);
                                                        % plot graphite vector
errorbar(T1wat, trans.wat, transerr.wat, 'bx'); grid
                                                  % plot water attenuation line
errorbar(T1lead, trans.lead, transerr.lead, 'rx');  % plot lead attenuation line
errorbar(T1graph, trans.graph, transerr.graph, 'yx');
                                                  % plot graphite attenuation line
if trans.lead(T==1)-transerr.lead(T==1) < 0;      % lead errorbar bottom -ve
  plot([1 1], [10^-7 trans.lead(T==1)+transerr.lead(T==1)], 'r');
                                                  % plot to graph bottom
else                                              % lead error bar bottom +ve
  plot([1 1], [trans.lead(T==1)-transerr.lead(T==1)
       trans.lead(T==1)+transerr.lead(T==1)], 'r');     % plot to errorbar bottom
end                                               % end lead errorbar if
if trans.graph(end)-transerr.graph(end) < 0;      % graphite errorbar bottom -ve
  plot([T1graph T1graph], [10^-7 trans.lead(end)+transerr.lead(end)], 'y');
                                                  % plot to graph bottom
else                                              % graphite errorbar bottom +ve
  plot([T1graph T1graph], [trans.lead(end)-transerr.lead(end) trans.lead(end)
             +transerr.lead(end)], 'y');                % plot to errorbar bottom
end                                               % end graphite errorbar if
l = legend('water', 'lead', 'graphite');          % label plots
```

```
set(l, ‘FontSize', 10);                          % legend font size
l.Location = ‘best’;                             % best location for legend
hold off;                                        % turn hold off
print('attenuation', '-depsc');                  % save fig 26
stop = struct('wat', attenuate.wat*log(particles*2), 'lead', attenuate.lead*log(parti
         cles*2), 'graph', attenuate.graph*log(particles*2));        % stop distance
% ---------------------------------------------------------------------------------------


% ---------------------------------------------------------------------------------------
% water walks
minypathwater = min(pathwat.y(:));         % min y water walk val
maxypathwater = max(pathwat.y(:));         % max y water walk val
minzpathwater = min(pathwat.z(:));         % min z water walk val
maxzpathwater = max(pathwat.z(:));         % max z water walk val
[yypathwater, zzpathwater] = meshgrid(minypathwater:(maxypathwater-miny
         pathwater):maxypathwater, minzpathwater:(maxzpathwater-
                  minzpathwater):maxzpathwater);           % water walk grid
figure;                                          % open fig 27
xlabel('x (m)', 'FontSize', 12);                 % x axis label
ylabel('y (m)', 'FontSize', 12);                 % y axis label
zlabel('z (m)', 'FontSize', 12);                 % z axis label
set(gca, 'FontSize', 12);                        % axis font size
hold on;                                         % plot on same fig
for o = 1:6                                      % water particles 1-6
   plot3(cat(1,0,nonzeros(pathwat.x(2:end,o))),
cat(1,zeros(2,1),nonzeros(pathwat.y(3:end,o))),
cat(1,zeros(2,1),nonzeros(pathwat.z(3:end,o)))); % plot water particle 1-6
end                                              % end water particles 1-6
text(0,0,0,'O');                                 % label origin
surf(zeros(size(yypathwater)), yypathwater, zzpathwater, 'FaceColor', 'none');
                                                 % label x=0 edge
surf(0.1*ones(size(yypathwater)), yypathwater, zzpathwater, 'FaceColor', 'none');
                                                 % label other edge
hold off;                                        % turn hold off
print('water walks', '-depsc');                  % save fig 27
% ---------------------------------------------------------------------------------------


% ---------------------------------------------------------------------------------------
% lead walks
minypathlead = min(pathlead.y(:));         % min y lead walk val
maxypathlead = max(pathlead.y(:));         % max y lead walk val
minzpathlead = min(pathlead.z(:));         % min z lead walk val
maxzpathlead = max(pathlead.z(:));         % max y lead walk val
```

```
[yypathlead, zzpathlead] = meshgrid(minypathlead:(maxypathlead-
                    minypathlead):maxypathlead, minzpathlead:(maxzpathlead-
                              minzpathlead):maxzpathlead);        % lead walk grid
figure;                                                           % open fig 28
xlabel('x (m)', 'FontSize', 12);                                  % x axis label
ylabel('y (m)', 'FontSize', 12);                                  % y axis label
zlabel('z (m)', 'FontSize', 12);                                  % z axis label
set(gca, 'FontSize', 12);                                         % axis font size
hold on;                                                          % plot on same fig
for p = 1:6                                                       % lead particles 1-6
   plot3(cat(1,0,nonzeros(pathlead.x(2:end,p))),
cat(1,zeros(2,1),nonzeros(pathlead.y(3:end,p))),
cat(1,zeros(2,1),nonzeros(pathlead.z(3:end,p))));  % plot lead particles 1-6
end                                                               % end lead particles 1-6
text(0,0,0,'O');                                                  % label origin
surf(zeros(size(yypathlead)), yypathlead, zzpathlead, 'FaceColor', 'none');
                                                                  % x=0 edge
surf(0.1*ones(size(yypathlead)), yypathlead, zzpathlead, 'FaceColor', 'none');
                                                                  % other edge
hold off;                                                         % turn hold off
print('lead walks', '-depsc');                                    % save fig 28
% ----------------------------------------------------------------------------------------------------


% ----------------------------------------------------------------------------------------------------
% graphite walks
minypathgraph = min(pathgraph.y(:));     % min y graphite walk val
maxypathgraph = max(pathgraph.y(:));     % max y graphite walk val
minzpathgraph = min(pathgraph.z(:));     % min z graphite walk val
maxzpathgraph = max(pathgraph.y(:));     % max z graphite walk val
[yypathgraph, zzpathgraph] = meshgrid(minypathgraph:(maxypathgraph-miny
                    pathgraph):maxypathgraph, minzpathgraph:(maxzpathgraph-minz
                              pathgraph):maxzpathgraph);          % graphite walk grid
figure;                                                           % open fig 29
xlabel('x (m)', 'FontSize', 12);                                  % x axis label
ylabel('y (m)', 'FontSize', 12);                                  % y axis label
zlabel('z (m)', 'FontSize', 12);                                  % z axis label
set(gca, 'FontSize', 12);                                         % axis font size
hold on;                                                          % plot on same fig
for r = 1:6                                                       % graphite particles 1-6
   plot3(cat(1,0,nonzeros(pathgraph.x(2:end,r))), cat(1,zeros(2,1),nonzeros(path
         graph.y(3:end,r))), cat(1,zeros(2,1),nonzeros(pathgraph.z(3:end,r))));
                                                                  % plot graphite particles 1-6
end                                                               % end graphite particles 1-6
text(0,0,0,'O');                                                  % label origin
```

```
surf(zeros(size(yypathgraph)), yypathgraph, zzpathgraph, 'FaceColor', 'none');
                                                   % x=0 edge
surf(0.1*ones(size(yypathgraph)), yypathgraph, zzpathgraph, 'FaceColor', 'none');
                                                   % other edge
hold off;                                          % turn hold off
print('graphite walks', '-depsc');                 % save fig 29
% -----------------------------------------------------------------------------------


% -----------------------------------------------------------------------------------
% lead graphite walks
minypathleadgraph = min(pathleadgraph.y(:));   % min y lead graphite walk val
maxypathleadgraph = max(pathleadgraph.y(:));   % max y lead graphite walk val
minzpathleadgraph = min(pathleadgraph.z(:));   % min z lead graphite walk val
maxzpathleadgraph = max(pathleadgraph.z(:));   % max z lead graphite walk val
[yypathleadgraph, zzpathleadgraph] = meshgrid(minypathleadgraph:(maxypath
        leadgraph-minypathleadgraph):maxypathleadgraph, minzpathleadgraph:
        (maxzpathleadgraph-minzpathleadgraph):maxzpathleadgraph);
                                        % lead graphite walk grid
figure;                                 % open fig 30
xlabel('x (m)', 'FontSize', 12);        % label x axis
ylabel('y (m)', 'FontSize', 12);        % label y axis
zlabel('z (m)', 'FontSize', 12);        % label z axis
set(gca, 'FontSize', 12);               % axis font size
hold on;                                % plot on same fig
for s = 1:6                             % lead graphite particles 1-6
  plot3(cat(1,0,nonzeros(pathleadgraph.x(2:end,s))), cat(1,zeros(2,1),nonzeros(path-
leadgraph.y(3:end,s))), cat(1,zeros(2,1),nonzeros(pathleadgraph.z(3:end,s))), 'o-',
        'MarkerSize', 2);               % plot lead graphite particles 1-6
end                                     % end lead graphite particles 1-6
text(0,0,0,'O');                        % label origin
surf(zeros(size(yypathleadgraph)), yypathleadgraph, zzpathleadgraph, 'FaceColor',
        'none');                        % x=0 edge
surf(0.1*ones(size(yypathleadgraph)), yypathleadgraph, zzpathleadgraph, 'FaceCol
        or', 'none');                   % material boundary
surf(0.2*ones(size(yypathleadgraph)), yypathleadgraph, zzpathleadgraph, 'FaceCol
        or', 'none');                   % other edge
hold off;                               % turn hold off
print('lead graphite walks', '-depsc'); % save fig 30
% -----------------------------------------------------------------------------------


% -----------------------------------------------------------------------------------
% lead water walks
minypathleadwat = min(pathleadwat.y(:));       % min y lead water walk val
maxypathleadwat = max(pathleadwat.y(:));       % max y lead water walk val
```

```
minzpathleadwat = min(pathleadwat.z(:));          % min z lead water walk val
maxzpathleadwat = max(pathleadwat.z(:));          % max z lead water walk val
[yypathleadwat, zzpathleadwat] = meshgrid(minypathleadwat:(maxypathleadwat-
          minypathleadwat):maxypathleadwat, minzpathleadwat:(maxzpathleadwat-
                    minzpathleadwat):maxzpathleadwat);   % lead water walk grid
figure;                                     % open fig 31
xlabel('x (m)', 'FontSize', 12);            % label x axis
ylabel('y (m)', 'FontSize', 12);            % label y axis
zlabel('z (m)', 'FontSize', 12);            % label z axis
set(gca, 'FontSize', 12);                   % axis font size
hold on;                                    % plot on same fig
for t = 1:6                                 % lead water particles 1-6
   plot3(cat(1,0,nonzeros(pathleadwat.x(2:end,t))), cat(1,zeros(2,1),nonzeros(path-
leadwat.y(3:end,t))), cat(1,zeros(2,1),nonzeros(pathleadwat.z(3:end,t))), 'o-', 'Mark
          erSize', 2);                      % plot lead water particles 1-6
end                                         % end lead water particles 1-6
text(0,0,0,'O');                            % label origin
surf(zeros(size(yypathleadwat)), yypathleadwat, zzpathleadwat, 'FaceColor', 'none');
                                            % x=0 edge
surf(0.1*ones(size(yypathleadwat)), yypathleadwat, zzpathleadwat, 'FaceColor',
          'none');                          % material boundary
surf(0.2*ones(size(yypathleadwat)), yypathleadwat, zzpathleadwat, 'FaceColor',
          'none');                          % other edge
hold off;                                   % turn hold off
print('lead water walks', '-depsc');        % save fig 31
% -----------------------------------------------------------------------------------------


% -----------------------------------------------------------------------------------------
% graphite lead walks
minypathgraphlead = min(pathgraphlead.y(:));  % min y graphite lead walk val
maxypathgraphlead = max(pathgraphlead.y(:));  % max y graphite lead walk val
minzpathgraphlead = min(pathgraphlead.z(:));  % min z graphite lead walk val
maxzpathgraphlead = max(pathgraphlead.z(:));  % max z graphite lead walk val
[yypathgraphlead, zzpathgraphlead] = meshgrid(minypathgraphlead:(maxypath
          graphlead-minypathgraphlead):maxypathgraphlead, minzpathgraphlead:
          (maxzpathgraphlead-minzpathgraphlead):maxzpathgraphlead);
                                            % graphite lead walk grid
figure;                                     % open fig 32
xlabel('x (m)', 'FontSize', 12);            % label x axis
ylabel('y (m)', 'FontSize', 12);            % label y axis
zlabel('z (m)', 'FontSize', 12);            % label z axis
set(gca, 'FontSize', 12);                   % axis font size
hold on;                                    % plot on same fig
for u = 1:6                                 % graphite lead particles 1-6
```

```
   plot3(cat(1,0,nonzeros(pathgraphlead.x(2:end,u))), cat(1,zeros(2,1),nonzeros(path-
graphlead.y(3:end,u))), cat(1,zeros(2,1),nonzeros(pathgraphlead.z(3:end,u))), 'o-',
         'MarkerSize', 2);                       % plot graphite lead particles 1-6
end                                              % end graphite lead particles 1-6
text(0,0,0,'O');                                 % label origin
surf(zeros(size(yypathgraphlead)), yypathgraphlead, zzpathgraphlead, 'FaceColor',
         'none');                                % x=0 edge
surf(0.1*ones(size(yypathgraphlead)), yypathgraphlead, zzpathgraphlead, 'FaceCol
         or', 'none');                           % material boundary
surf(0.2*ones(size(yypathgraphlead)), yypathgraphlead, zzpathgraphlead, 'FaceCol
         or', 'none');                           % other edge
hold off;                                        % turn hold off
print('graphite lead walks', '-depsc');          % save fig 32
% -----------------------------------------------------------------------------------------


% -----------------------------------------------------------------------------------------
% graphite water walks
minypathgraphwat = min(pathgraphwat.y(:));    % min y graphite water walk val
maxypathgraphwat = max(pathgraphwat.y(:));    % max y graphite water walk val
minzpathgraphwat = min(pathgraphwat.z(:));    % min z graphite water walk val
maxzpathgraphwat = max(pathgraphwat.z(:));    % max z graphite water walk val
[yypathgraphwat, zzpathgraphwat] = meshgrid(minypathgraphwat:(maxypath-
graphwat-minypathgraphwat):maxypathgraphwat, minzpathgraphwat:(maxzpath-
graphwat-minzpathgraphwat):maxzpathgraphwat);    % graphite water walk grid
figure;                                          % open fig 33
xlabel('x (m)', 'FontSize', 12);                 % label x axis
ylabel('y (m)', 'FontSize', 12);                 % label y axis
zlabel('z (m)', 'FontSize', 12);                 % label z axis
set(gca, 'FontSize', 12);                        % axis fonts size
hold on;                                         % plot on same fig
for v = 1:6                                      % graphite water particles 1-6
   plot3(cat(1,0,nonzeros(pathgraphwat.x(2:end,v))), cat(1,zeros(2,1),nonzeros(path-
graphwat.y(3:end,v))), cat(1,zeros(2,1),nonzeros(pathgraphwat.z(3:end,v))), 'o-',
         'MarkerSize', 2);                       % plot graphite water particles 1-6
end                                              % end graphite water particles 1-6
text(0,0,0,'O');                                 % label origin
surf(zeros(size(yypathgraphwat)), yypathgraphwat, zzpathgraphwat, 'FaceColor',
         'none');                                % x=0 edge
surf(0.1*ones(size(yypathgraphwat)), yypathgraphwat, zzpathgraphwat,
                'FaceColor', 'none');            % material boundary
surf(0.2*ones(size(yypathgraphwat)), yypathgraphwat, zzpathgraphwat,
                'FaceColor', 'none');            % other edge
hold off;                                        % turn hold off
print('graphite water walks', '-depsc');         % save fig 33
% -----------------------------------------------------------------------------------------
% ------------------------------------------------------------------------------------------------
% water graphite walks
minypathwatgraph = min(pathwatgraph.y(:));    % min y water graphite walk val
maxypathwatgraph = max(pathwatgraph.y(:));    % max y water graphite walk val
minzpathwatgraph = min(pathwatgraph.z(:));    % min z water graphite walk val
maxzpathwatgraph = max(pathwatgraph.z(:));    % max z water graphite walk val
[yypathwatgraph, zzpathwatgraph] = meshgrid(minypathwatgraph:(maxypathwat
graph-minypathwatgraph):maxypathwatgraph, minzpathwatgraph:(maxz
pathwatgraph-minzpathwatgraph):maxzpathwatgraph);  % water graphite grid
figure;                                   % open fig 34
xlabel('x (m)', 'FontSize', 12);          % x axis label
ylabel('y (m)', 'FontSize', 12);          % y axis label
zlabel('z (m)', 'FontSize', 12);          % z axis label
set(gca, 'FontSize', 12);                 % axis font size
hold on;                                  % plot on same fig
for w = 1:6                               % water graphite particles 1-6
   plot3(cat(1,0,nonzeros(pathwatgraph.x(2:end,w))),
cat(1,zeros(2,1),nonzeros(pathwatgraph.y(3:end,w))),
cat(1,zeros(2,1),nonzeros(pathwatgraph.z(3:end,w))), 'o-', 'MarkerSize', 2);
                                          % plot water graphite particles 1-6
end                                       % end water graphite particles 1-6
text(0,0,0,'O');                          % label origin
surf(zeros(size(yypathwatgraph)), yypathwatgraph, zzpathwatgraph, 'FaceColor',
        'none');                          % x=0 edge
surf(0.1*ones(size(yypathwatgraph)), yypathwatgraph, zzpathwatgraph,
                'FaceColor', 'none');     % material boundary
surf(0.2*ones(size(yypathwatgraph)), yypathwatgraph, zzpathwatgraph,
                'FaceColor', 'none');     % other edge
hold off;                                 % turn hold off
print('water graphite walks', '-depsc');  % save fig 34
% ------------------------------------------------------------------------------------------------


% ------------------------------------------------------------------------------------------------
% water lead walks
minypathwatlead = min(pathwatlead.y(:));        % min y water lead walk val
maxypathwatlead = max(pathwatlead.y(:));        % max y water lead walk val
minzpathwatlead = min(pathwatlead.z(:));        % min z water lead walk val
maxzpathwatlead = max(pathwatlead.z(:));        % max z water lead walk val
[yypathwatlead, zzpathwatlead] = meshgrid(minypathwatlead:(maxypathwatlead-
        minypathwatlead):maxypathwatlead, minzpathwatlead:(maxzpathwatlead-
                minzpathwatlead):maxzpathwatlead);      % water lead grid
figure;                                         % open fig 35
xlabel('x (m)', 'FontSize', 12);                % x axis label
ylabel('y (m)', 'FontSize', 12);                % y axis label
zlabel('z (m)', 'FontSize', 12);                % z axis label
```

```
set(gca, 'FontSize', 12);                                % axis font size
hold on;                                                 % plot on same fig
for x = 1:6                                              % water lead particles 1-6
   plot3(cat(1,0,nonzeros(pathwatlead.x(2:end,x))), cat(1,zeros(2,1),nonzeros(path-
watlead.y(3:end,x))), cat(1,zeros(2,1),nonzeros(pathwatlead.z(3:end,x))), 'o-', 'Mark
         erSize', 2);                                    % plot water particles 1-6
end                                                      % end water lead particles 1-6
text(0,0,0,'O');                                         % label origin
surf(zeros(size(yypathwatlead)), yypathwatlead, zzpathwatlead, 'FaceColor', 'none');
                                                         % x=0 edge
surf(0.1*ones(size(yypathwatlead)), yypathwatlead, zzpathwatlead, 'FaceColor',
         'none');                                        % material boundary
surf(0.2*ones(size(yypathwatlead)), yypathwatlead, zzpathwatlead, 'FaceColor',
         'none');                                        % other edge
hold off;                                                % turn hold off
print('water lead walks', '-depsc');                     % save fig 35
% -----------------------------------------------------------------------------------
% -----------------------------------------------------------------------------------



% -----------------------------------------------------------------------------------
% -----------------------------------------------------------------------------------
function [ xunit, yunit, zunit ] = unitVector( )
%unitVector Generates unit vectors
%   Generates 3D unit vector in x,y,z
      theta = asin(-1+2*rand());      % theta value
      phi = 2*pi*rand();              % phi value
      xunit = cos(theta)*cos(phi);    % x direction
      yunit = cos(theta)*sin(phi);    % y direction
      zunit = sin(theta);             % z direction
end                                   % end func
% ___________________________________________________________________________________

% -----------------------------------------------------------------------------------
function [ xstep, ystep, zstep ] = step( xunit, yunit, zunit, lamda )
%steps Generates exponentially distributed steps
%   Takes direction and outputs exponentially distributed step in 3D
      dist = -lamda*log(rand());      % exponential distribution
      xstep = dist.*xunit;            % step in x direction
      ystep = dist.*yunit;            % step in y direction
      zstep = dist.*zunit;            % step in z direction
end                                   % end func
% -----------------------------------------------------------------------------------
```