Lecture 7

10.4 Using surfaces to make objects

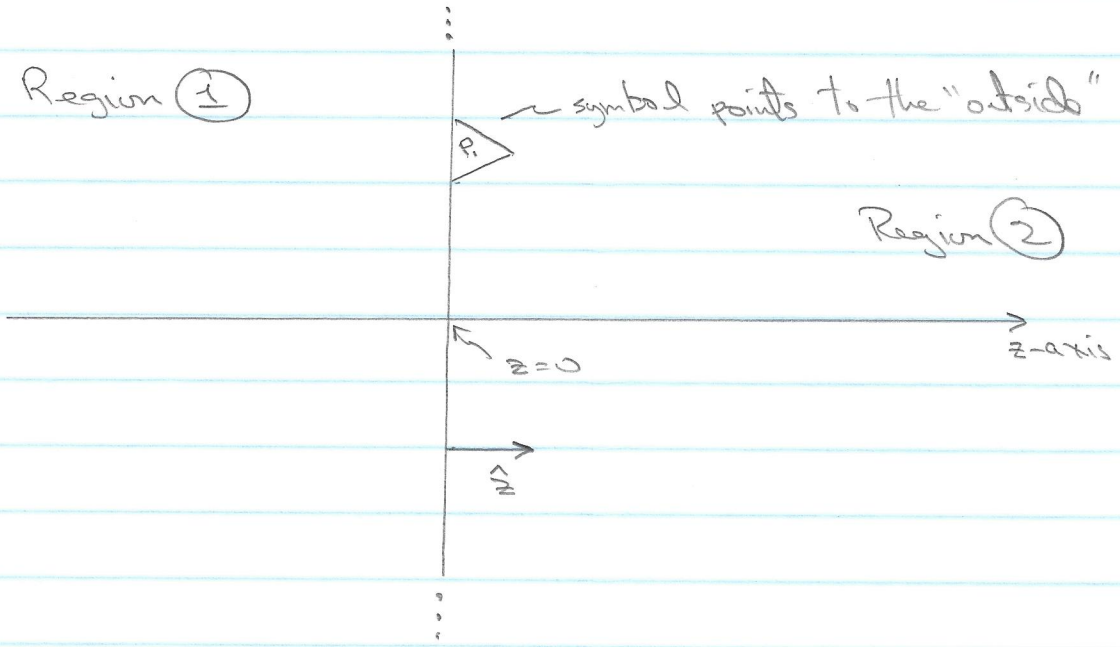
Elemental volume

A region of space that is specified implicitly as inside or outside a collection of surfaces. (We'll restrict this to planes or quadrics)

E.g.

A single plane

$$f = z = 0$$
$$\Rightarrow \nabla f = \hat{z}$$



Particles in (1) are "inside" P1
" " (2) " "outside" P1

```
% track2plane.n Class Example
% Track to the plane, z = 0, from a random start within a 2x2x2 box
% centered at the origin, with sides = 2

clear all
close all

Nh = input('Number of histories: ');

Hits = 0; % Count the number of hits

for n = 1:Nh

   % pick a starting position
   % The following code picks the initial position randomly with a 2x2 box
   % ccentered at the origin
   x = [1 - 2*rand,1 - 2*rand,1 - 2*rand];

   % pick a starting random direction vector
   costhe = 1 - 2*rand;
   sinthe = sqrt(1 - costhe^2);
   phi = 2*pi*rand;
   u = [sinthe*cos(phi),sinthe*sin(phi),costhe];

   % Determine inside or outside
   if     (x(3) <  0) inside = true;
   elseif (x(3) >  0) inside = false;
   else
      if (u(3) > 0) % On surface headed outside
         inside = false; % Place it outside
      elseif (u(3) < 0) % On surface headed inside
         inside = true; % Place it inside
      else % On surface and tangent to it, unable to determine logical location.
         if (rand <= 0.5) % So make a random choice!
            inside = false;
         else
            inside = true;
         end
      end
   end

   [hit,s] = zplane(0,x(3),u(3),inside); % See zplane.m for explanation of I/O

   if (hit)
      Hits = Hits + 1; % Count the hits
      xhit(Hits) = x(1) + s*u(1); % Track to the surface
      yhit(Hits) = x(2) + s*u(2);
      zhit(Hits) = x(3) + s*u(3);
   end

end % End of loop over histories

disp(sprintf('Minimum z = %g, Maxiumum z = %g', min(zhit), max(zhit)))

plot3(xhit, yhit, zhit,'k.') % Plot the surface intercepts
axis([-10,10,-10,10,-1,1])
```

```
function [hit,s] = zplane(zp,z0,w,inside)

% Calculates the distance of a particle to a planar surface normal to the
% z-axis

% Inputs
%%%%%%%%
%  double
%     zp:  z-coordinate that the plane passed through
%     z0:  z-coordinate of the particle
%     w :  z-axis direction cosine of the particle
%  logical
%     inside: == true   particle thinks it is inside
%           : == false  particle thinks it is outside
%

% Outputs
%%%%%%%%%
%  logical
%     hit: == true   particle would hit  the surface
%        : == false  particle would miss the surface
%
%  double
%     s: distance to the surface (if hit)

if ((inside & w > 0) | (~inside & w < 0)) % headed towards the surface
   hit = true;
   s = max(0,(zp-z0)/w);
   return
else
   hit = false;
   s = inf;
   return
end
```

```
% track2sphere.n Class Example
% Tracking to a unit sphere, radius = 1, centered at the origin
% from a random starting position within a minimum bounding box.

clear all, close all

Nh = input('Number of histories: ');

Hits = 0; % Count the number of hits

for n = 1:Nh

   % Pick a random starting position within the box that contains the sphere
   x = [1 - 2*rand,1 - 2*rand,1 - 2*rand];

   % Pick a random starting direction vector
   costhe = 1 - 2*rand;
   sinthe = sqrt(1 - costhe^2);
   phi = 2*pi*rand;
   u = [sinthe*cos(phi),sinthe*sin(phi),costhe];

   % Determine wherther inside or outside
   if     (x*x' <  1) inside = true;  % Starts inside
   elseif (x*x' >  1) inside = false; % Starts outside
   else % x*x' = 1, on the sphere, need another way to determine inside/outside
      xdotu = x*u';      % Dot product of x[] aand u[]
      if (xdotu > 0)     % On surface headed outside
         inside = false; % Place it outside
      elseif (xdotu < 0) % On surface headed inside
         inside = true;  % Place it inside
      else % On surface and tangent to it
         inside = false; % Place it outside (this works for spheres)
      end
   end

   [hit,s] = sphere(1,[0,0,0],x,u,inside); % See sphere.m for explanation of I/O

   if (hit)
      Hits = Hits + 1; % Count the hits
      xhit(Hits) = x(1) + s*u(1); % Track to the surface
      yhit(Hits) = x(2) + s*u(2);
      zhit(Hits) = x(3) + s*u(3);
      r2(Hits) = xhit(Hits)^2 + yhit(Hits)^2 + zhit(Hits)^2; % Diagnostic
   end

end % End of loop over histories

disp(sprintf('Minimum radius^2 - 1 = %g, Maxiumum radius^2 - 1 = %g', min(r2) - 1, max(r2) - 1))

plot3(xhit, yhit, zhit,'k.') % Plot the surface intercepts
axis equal
```

```
function [hit,s] = sphere(R,xs,x0,u,inside)

% Calculates the distance to a sphere of radius R centered at (xs,ys,zs)

% Inputs
%%%%%%%%
%  double
%     R :  Radius of the sphere (scalar)
%     xs:  coordinates of the center of the sphere (row vector)
%     x0:  coordinates of the particle (row vector)
%     u :  direction cosines of the particle (row vector)
%  logical
%     inside: == true   particle thinks it is inside
%           : == false  particle thinks it is outside
%

% Outputs
%%%%%%%%%
%  logical
%     hit: == true   particle would hit  the surface
%        : == false  particle would miss the surface
%
%  double
%     s: distance to the surface (if hit)
%
% Internal variables
%  double
%     xr: relative coordinates of the particle wrt the sphere (row vector)
%     A:  Quadratic coefficient A (communicated to quadric) (scalar)
%     B:  Quadratic coefficient B (communicated to quadric) (scalar)
%     C:  Quadratic coefficient C (communicated to quadric) (scalar)

% Compute relative coordinates

xr = x0 - xs;

% Compute the quadratic coefficients
A = 1; % i.e. assuming u**2 + v**2 + w**2 = 1
B = u*xr';
C = xr*xr' - R.^2;

% Call the quadratic solver
[hit,s] = quadric(A,B,C,inside);

return
```

```
function [hit,s] = quadric(A,B,C,inside)
% Calculates the first positive distance to an arbitrary quadric surface
% Inputs:
%  double
%     A:  Quadratic coefficient A (communicated to quadric)
%     B:  Quadratic coefficient B (communicated to quadric)
%     C:  Quadratic coefficient C (communicated to quadric)
%  logical
%     inside: == true   particle thinks it is inside
%           : == false  particle thinks it is outside
% Outputs:
%  logical
%     hit: == true   particle would hit  the surface
%        : == false  particle would miss the surface
%  double
%     s: distance to the surface (if hit)

% Internal variables
%  double
%     Q: quadratic coefficient

Q = B.^2 - A.*C;
if (Q < 0)
    hit = false;
    s = nan;
    return
end

if (inside)           % inside the surface
   if (B <= 0)        % headed away from surface
      if (A > 0)      % but, surface curving up
         hit = true;  % always a hit in this case
         s = (sqrt(Q) - B)./A;
         return
      else            % surface curving away and headed in
         hit = false; % never a hit in this case
         s = inf;
         return
      end
   else               % headed toward the surface
      hit = true;     % always a hit in this case
      s = max(0, -C./(sqrt(Q) + B));
      return
   end
end

% Must be outside the surface
if (B >= 0)          % headed away
    if (A >= 0)      % surface curves away
        hit = false; % never a hit in this case
        s = inf;
        return
    else             % surface curves up
       hit = true;  % always a hit in this case
       s = -(sqrt(Q) + B)./A;
       return
    end
else                 % headed toward the surface
    hit = true;      % always a hit in this case
    s = max(0, C./(sqrt(Q) - B));
    return
end
return
```

```
% track2quadric.n Class Example
% Tracking to all the 9 quadrics, plus parallel and crossing
% planes, coincident planes, and even a single plan
% from a random starting position within a 2x2x2x box.

clear all
close all
format compact

Ns = input('Input the surface option [1...13] (out of range value will give help): ');
if (Ns < 1 || Ns > 13)
   disp('1 is x^2 + y^2 + z^ 2 - 1 = 0, a sphere')
   disp('2 is x^2 + y^2 - z^ 2 = 0, a cone')
   disp('3 is x^2 + y^2 - 1 = 0, a cylinder')
   disp('4 is x^2 + y^2 - z^ 2 - 1 = 0, a hyperboloid of one sheet')
   disp('5 is x^2 + y^2 - z^ 2 + 1 = 0, a hyperboloid of two sheets')
   disp('6 is x^2 + y^2 + z = 0, an elliptic paraboloid')
   disp('7 is x^2 - y^2 + z = 0, a hyperbolic paraboloid')
   disp('8 is x^2 - y^2 + 1 = 0, a hyperbolic cylinder')
   disp('9 is x^2 + z = 0, a parabolic cylinder')
   disp('10 is z*(z-1) = 0, two parallel planes')
   disp('11 is xy = 0, two crossed planes')
   disp('12 is z^2 = 0, two coincident planes at z = 0')
   disp('13 is z = 0, a single plane at z = 0')
   return
end

Nh = input('Number of histories: ');

Hits = 0; % Count the number of hits

xin  = []; % Tally for starting points that are inside
xout = []; % Tally for starting points that are outside

for n = 1:Nh

   % pick a starting position
   % The following code picks the initial position randomly with a 2x2x2 box
   % containing the sphere
   x = [1 - 2*rand, 1 - 2*rand, 1 - 2*rand];

   % pick a starting direction vector
   costhe = 1 - 2*rand;
   sinthe = sqrt(1 - costhe^2);
   phi = 2*pi*rand;
   u = [sinthe*cos(phi),sinthe*sin(phi),costhe];

   if (Ns == 1)
      if (n == 1) disp('Surface: x^2 + y^2 + z^ 2 - 1 = 0, a sphere'); end
      A = 1;
      B = x*u';
      C = x*x' - 1;
   elseif (Ns == 2)
      if (n == 1) disp('Surface: x^2 + y^2 - z^ 2 = 0, a cone'); end
      A = u(1)*u(1) + u(2)*u(2) - u(3)*u(3);
      B = x(1)*u(1) + x(2)*u(2) - x(3)*u(3);
      C = x(1)*x(1) + x(2)*x(2) - x(3)*x(3);
   elseif (Ns == 3)
      if (n == 1) disp('Surface: x^2 + y^2 - 1 = 0, a cylinder'); end
      A = u(1)*u(1) + u(2)*u(2);
      B = x(1)*u(1) + x(2)*u(2);
      C = x(1)*x(1) + x(2)*x(2) - 1;
   elseif (Ns == 4)
      if (n == 1) disp('Surface: x^2 + y^2 - z^ 2 - 1 = 0, a hyperboloid of one sheet'); end
```

```
   A = u(1)*u(1) + u(2)*u(2) - u(3)*u(3);
   B = x(1)*u(1) + x(2)*u(2) - x(3)*u(3);
   C = x(1)*x(1) + x(2)*x(2) - x(3)*x(3) - 1;
elseif (Ns == 5)
   if (n == 1) disp('Surface: x^2 + y^2 - z^ 2 + 1 = 0, a hyperboloid of two sheets'); end
   A = u(1)*u(1) + u(2)*u(2) - u(3)*u(3);
   B = x(1)*u(1) + x(2)*u(2) - x(3)*u(3);
   C = x(1)*x(1) + x(2)*x(2) - x(3)*x(3) + 1;
elseif (Ns == 6)
   if (n == 1) disp('Surface: x^2 + y^2 + z = 0, an elliptic paraboloid'); end
   A = u(1)*u(1) + u(2)*u(2);
   B = x(1)*u(1) + x(2)*u(2) + 0.5*u(3);
   C = x(1)*x(1) + x(2)*x(2) + x(3);
elseif (Ns == 7)
   if (n == 1) disp('Surface: x^2 - y^2 + z = 0, a hyperbolic paraboloid'); end
   A = u(1)*u(1) - u(2)*u(2);
   B = x(1)*u(1) - x(2)*u(2) + 0.5*u(3);
   C = x(1)*x(1) - x(2)*x(2) + x(3);
elseif (Ns == 8)
   if (n == 1) disp('Surface: x^2 - y^2 + 1 = 0, a hyperbolic cylinder'); end
   A = u(1)*u(1) - u(2)*u(2);
   B = x(1)*u(1) - x(2)*u(2);
   C = x(1)*x(1) - x(2)*x(2) + 1;
elseif (Ns == 9)
   if (n == 1) disp('Surface: x^2 + z = 0, a parabolic cylinder'); end
   A = u(1)*u(1);
   B = x(1)*u(1) + 0.5*u(3);
   C = x(1)*x(1) + x(3);
elseif (Ns == 10)
   if (n == 1) disp('Surface: z*(z-1) = 0, two parallel planes'); end
   A = u(3)*u(3);
   B = x(3)*u(3) - 0.5*u(3);
   C = x(3)*(x(3) - 1);
elseif (Ns == 11)
   if (n == 1) disp('Surface: xy = 0, two crossed planes'); end
   A = u(1)*u(2);
   B = 0.5*(x(1)*u(2) + x(2)*u(1));
   C = x(1)*x(2);
elseif (Ns == 12)
   if (n == 1) disp('Surface: z^2 = 0, two coincident planes'); end
   A = u(3)*u(3);
   B = x(3)*u(3);
   C = x(3)*x(3);
elseif (Ns == 13)
   if (n == 1) disp('Surface: z = 0, single plane'); end
   A = 0;
   B = 0.5*u(3);
   C = x(3);
end
% Determine inside or outside
   if  (C < 0)
   inside = true;
elseif (C > 0)
   inside = false;
else
   if (B > 0) % On surface headed outside
      inside = false; % Place it outside
   elseif (B < 0) % On surface headed inside
      inside = true; % Place it inside
   else % On surface and tangent to it
      if (A > 0)
         inside = false; % Place it outside
      elseif (A < 0)
         inside = true; % Place it inside
      else
```

```
            % It is going along a ruling. Do it the MC way
            if (rand < 0.5)
               inside = false; % Place it outside
            else
               inside = true; % Place it inside
            end
         end
      end
   end

   [hit,s] = quadric(A,B,C,inside); % See sphere.m for explanation of I/O

   if (hit)
      Hits = Hits + 1; % Count the hits
      xhit(Hits) = x(1) + s*u(1); % Track to the surface
      yhit(Hits) = x(2) + s*u(2);
      zhit(Hits) = x(3) + s*u(3);
      if      (inside) xin(end + 1,:)  = x;
      elseif (~inside) xout(end + 1,:) = x;
      end
   end

end % End of loop over histories



figure(1)
plot3(xhit, yhit, zhit,'k.') % Plot the surface intercepts
xlabel('x')
ylabel('y')
zlabel('z')
if (Ns == 1)
   axis equal
elseif(Ns == 5)
   axis([-3,3,-3,3,-3,3])
elseif(Ns == 10)
   axis([-2,2,-2,2,-0.5,1.5])
else
   axis([-2,2,-2,2,-2,2])
end

figure(3)
plot3(xout(:,1),xout(:,2),xout(:,3),'r.') % View results
axis equal
xlabel('x')
ylabel('y')
zlabel('z')

figure(2)
plot3(xin(:,1),xin(:,2),xin(:,3),'b.') % View results
axis equal
xlabel('x')
ylabel('y')
zlabel('z')
```

An elemental volume made up of several surfaces

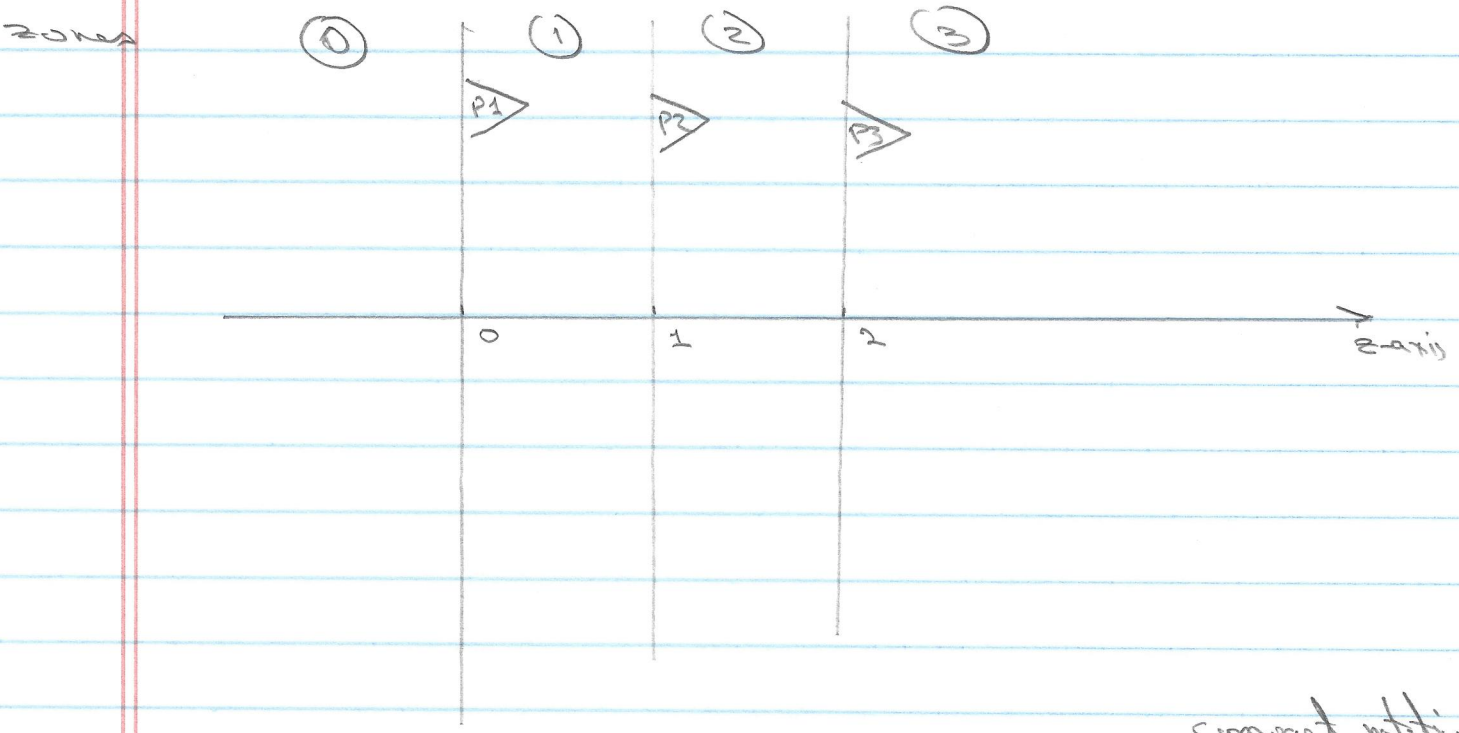
example 3 z-planes, making two planar zones

code <zplane3.m>

P1 z-plane z=0

P2 z-plane z=1

P3 z-plane z=2



compact notation

zone 0:	escape zone	inside P1	P_1
1:	transport zone	outside P1, inside P_2	$\overline{P}_1 \cap P_2$
2:	" "	outside P2, inside P_3	$\overline{P}_2 \cap P_3$
3:	escape zone	outside P3	$\overline{P}_3$

see code

```
Nr = 2 ;   % # of physical volume
z1 = 0 ;   %P1
z2 = 1 ;   %P2
z3 = 2 ;   %P3

% source (circularly collimated beam)
Rbeam = 1;   %radius
Rbeam2 = Rbeam^2;   % used later
Nh = some number;   % number of histories

for n = 1:Nh
    r = 1 ;   % starting zone (region) #
    x(3) = 0;   %starts @ z=0
    u = [0,0,1];   % directed along the +z-axis
    repeat = true;   % set up a rejection loop
    while (repeat)
        x(1) = 1 - 2*rand;   } 2x2 box
        x(2) = 1 - 2*rand;   } centered w
                                the xy plane
        if ( x(1)^2 + x(2)^2 <= Rbeam2)
            repeat = false;
        end
    end
    % Surviving x(1) & x(2) are x,y
    % within a circle
```



```
% Transport loop

while (r>0 ; r<= Nr)   % i.e. is in the physical geometry
   td = inf;   % start this way, geometry code will shorten it

   if (r==1)   % Particle is in zone 1
      [hit, d] = zplane(z1, x(3), u(3), false);
                  plane pos'n   particle loc'n   particle dir'n   not inside P1
      if (hit ; (d<=td))   % hits plane & would take us there
         r=0;   % escape zone is the next
         td=d;   % set the transport distance
      end   } checked one bounding surface
      [hit, d] = zplane(z2, x(3), u(3), true);
                                          inside P2
      if (hit ; d<=td)
         r=2;   % next zone
         td=d;
      end   } checked other bounding surface
   end % end of zone 1 check
   if (r==2)   % similar coding
      ...
   end % end of zone 2 check
   x = x + u*td;   % transport
end   % end of transport loop
```

```
% zplanes3.m Class Example - 3 parallel planes, non-interacting beam,
% a very simple implementation of multiple planes

clear all, close all

% Geometry definition %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Nr = 2; % Number of elemental subvolumes

% Region 0 (escape zone) definition, bounded by:
z1 = 0; % z = z1; z-plane

% Region 1 definition, bounded by z1 defined above plus
z2 = 1; % z = z2; z-plane

% Region 2 definition, bounded by z2 defined above plus
z3 = 2; % z = z3; z-plane

% Region 3 definition (escape zone), bounded by z3 defined above

% Source definition %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This example is a beam incident from the left, normal incidence
Rbeam = 1; % Radius of collimation of the beam

% Tally set up %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Below defines the tally vectors used for plotting later
xspot = [];

% Simulation %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Nh = 200; % Number of histories to simulate
for n = 1:Nh
   % Sample the source
   r = 1; % Source particle starts in region 1
   % Define the starting position vector
   repeat = true; % This is a rejection loop for circular collimation
   while (repeat)
      x(1) = Rbeam*(2*rand - 1);
      x(2) = Rbeam*(2*rand - 1);
      if (x(1)^2 + x(2)^2 <= Rbeam^2) repeat = false; end
   end
   x(3) = z1;            % Starts on leftmost plane
   u = [0, 0, 1];        % Normal incidence
   xspot(end + 1,:) = x; % Record the starting point

   % Transport through the geometry
   while (r > 0 & r <= Nr) % True if particle is inside the geometry
      t0 = inf;   % Start with an impossibly big transport distance
      if (r == 1) % Particle is in region 1
         [hit,t] = zplane(z1,x(3),u(3),false);   % Outside z1
         if (hit & (t <= t0)) r = 0; t0 = t; end
         [hit,t] = zplane(z2,x(3),u(3),true);    % Inside z2
         if (hit & (t <= t0)) r = 2; t0 = t; end
      elseif (r == 2) % Particle is in region 2
         [hit,t] = zplane(z2,x(3),u(3),false);   % Outside z2
         if (hit & (t <= t0)) r = 1; t0 = t; end
         [hit,t] = zplane(z3,x(3),u(3),true);    % Inside z3
         if (hit & (t <= t0)) r = 3; t0 = t; end
      end
      x = x + u*t0;         % Transport the particle
      xspot(end + 1,:) = x; % Record the intercept point
   end % End of the transport loop
   plot3(xspot(:,1),xspot(:,2),xspot(:,3),'k.') % View results
   pause(0.001) % The pause enables a dynamic view as points are added

end % End of the history loop
```

rejection loop to provide a circularly

easily modified to an arbitrary collection of z-planes

essential modification

```
Nr = input  # of planes
zp = linspace(0, Nr, Nr+1);   % Nr+1 planes
%eg. for Nr = 10;
% zp = [0,1,2,3,4,5,6,7,8,9,10];

% much the same as <zplanes3.m>

% Transport through the geometry
while(r>0 & r<=Nr)  %inside the physical geom
   tφ = inf;
   [hit,t] = zplane(zp(r), x(3), u(3), false);
   if(hit & t<=tφ) r=r-1; tφ=t; end   % <- less volume
   [hit,t] = zplane(zp(r+1), x(3), u(3), true);
   if(hit & t<=tφ) r=r+1; tφ=t; end
   x = x + u*tφ;  % transport
end  % by virtue of r=0 or r=Nr+1 (escape)
```

check both plane

sketch the logic diagram

generalize To N planes

```
% zplanesN.m Class Example - N parallel planes, non-interacting beam,
% a very simple implementation of multiple planes

clear all, close all

% Geometry definition %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Nr = 10;                  % Number of elemental subvolumes
zp = linspace(0,Nr,Nr+1); % Nr zones require Nr + 1 planar surfaces


% Source definition %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Rbeam = 1; % Radius of collimation of the beam

% Tally set up %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
xspot = []; % Tally vector used for plotting later

% Simulation %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Nh = 200; % Number of histories to simulate
for n = 1:Nh

   % Sample the source
   r = 1; % Source particle starts in region 1
   % Define the starting position vector
   repeat = true; % This is a rejection loop for circular collimation
   while (repeat)
      x(1) = Rbeam*(2*rand - 1);
      x(2) = Rbeam*(2*rand - 1);
      if (x(1)^2 + x(2)^2 <= Rbeam^2) repeat = false; end
   end
   x(3) = zp(1);          % Starts on leftmost plane
   u = [0, 0, 1];         % Normal incidence
   xspot(end + 1,:) = x; % Record the starting point

   % Transport through the geometry
   while (r > 0 & r <= Nr) % True if particle is inside the geometry
      t0 = inf; % Start with an impossibly big transport distance
      [hit,t] = zplane(zp(r),x(3),u(3),false); % Outside lower plane
      if (hit & (t <= t0)) r = r - 1; t0 = t; end
      [hit,t] = zplane(zp(r+1),x(3),u(3),true); % Inside z2
      if (hit & (t <= t0)) r = r + 1; t0 = t; end
      x = x + u*t0; % Transport the particle
      xspot(end + 1,:) = x; % Record the intercept point
   end % End of the transport loop
   plot3(xspot(:,1),xspot(:,2),xspot(:,3),'k.') % View results
   pause(0.001) % The pause enables a dynamic view as points are added

end % End of the history loop
```

How to add interactions?

Instead of $t\phi = \inf$ as the initial proposed transport distance

$$t\phi = -\log(\text{rand}) / \Sigma;$$

↑ can be region dependent

<Zplane3a.m>

Show results first.

Discuss general CG

Then, show the code

```
% zplanes3a.m Class Example - 3 parallel planes, 1 cylinder, interacting beam
% a very simple implementation of multiple planes
% This is an adaptation of zplane3.m adding a cylinder and interactions

clear all, close all

% Source definition %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This example is a beam incident from the left, normal incidence
Rbeam = 1; % Radius of collimation of the beam

% Geometry definition %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Nr = 2; % Number of elemental subvolumes

% Region 0 (escape zone) definition, bounded by:
z1 = 0; % z = z1; z-plane

% Region 1 definition, bounded by z1 defined above plus
z2 = 1; % z = z2; z-plane at z2
r1 = 2* Rbeam; % Cylinder with radius 2x that of the beam

% Region 2 definition, bounded by z2 and r1 defined above plus
z3 = 2; % z = z3; z-plane at z3

% Region 3 definition (escape zone), bounded by z3 defined above

% Region 4 definition, bounded by (escape zone)
% r1 defined above

% Physics set up %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

S = linspace(0.1,10,Nr); % Macroscopic Sigma in the exponential interaction law
% Note that it is region dependent and hence we are modelling an inhomogeneous
% object. Region 1 will have little scattering, Region 2 will have lots.
% For Nr = 2, S = [0.1,10];

% Tally set up %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
xspot = []; % Tally vector used for plotting later

% Simulation %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Nh = 500; % Number of histories to simulate
for n = 1:Nh

   % Sample the source
   r = 1; % Source particle starts in region 1
   % Define the starting position vector
   repeat = true; % This is a rejection loop for circular collimation
   while (repeat)
      x(1) = Rbeam*(2*rand - 1);
      x(2) = Rbeam*(2*rand - 1);
      if (x(1)^2 + x(2)^2 <= Rbeam^2) repeat = false; end
   end
   x(3) = z1;            % Starts on leftmost plane
   u = [0, 0, 1];        % Normal incidence
   xspot(end + 1,:) = x; % Record the starting point

   % Transport through the geometry

   while (r > 0 & r <= Nr) % True of particle is inside the geometry
   t0 = -log(rand)/S(r); % Distance to an interaction, note region dependence
   ri = r; % Current region

      % zplane() and ccylz() are NERS544 library routines

      if (r == 1) % Particle is in region 1
```

```
      [hit,t] = zplane(z1,x(3),u(3),false); % Outside z1
      if (hit & (t <= t0)) r = 0; t0 = t; end
      [hit,t] = zplane(z2,x(3),u(3),true); % Inside z2
      if (hit & (t <= t0)) r = 2; t0 = t; end
      [hit,t] = ccylz(r1,x,u,true); % Inside r1
      if (hit & (t <= t0)) r = 4; t0 = t; end

   elseif (r == 2) % Particle is in region 2
      [hit,t] = zplane(z2,x(3),u(3),false); % Outside z2
      if (hit & (t <= t0)) r = 1; t0 = t; end
      [hit,t] = zplane(z3,x(3),u(3),true); % Inside z3
      if (hit & (t <= t0)) r = 3; t0 = t; end
      [hit,t] = ccylz(r1,x,u,true); % Inside r1
      if (hit & (t <= t0)) r = 4; t0 = t; end
   end

   x = x + u*t0; % Transport the particle
   if (r == ri) % The particle stays in the same region thus must interact
      ws = 2*rand - 1; % Isotropic scattering
      u = rotate(u,ws); % NERS544 library routine
   end

   xspot(end + 1,:) = x; % Record the intercept or interaction point

end

pause(0.001) % Matlab has a multithread error causing me to do this
             % Plotting starts before the data is written!
plot3(xspot(:,1),xspot(:,2),xspot(:,3),'k.') % View results

end % End of simulation
```