Radiography

Written by

Euan Foster

Feb 2019

Email: **e.foster@strath.ac.uk**
Tel: **07539 450593**

1 INDEX

Table of Contents

2 INTRODUCTION

A 1-inch thick steel plate with a cross section of 100 x 100 mm has a side drilled hole of 5 mm centred at its mid-plane and drilled parallel to the surface to a depth of 50 mm – See Figure 1 - Component Geometry

An x-ray source and detector are used to generate an image of the plate and the hole. The source and detector are separated by a distance of 240 mm. The hole located within the plate is located at a distance of 120 mm - See Figure 2 - Side view of Experimental Set Up.

A MATLAB code is presented in Appendix A that describes the generation a simulated image generated by this set up.

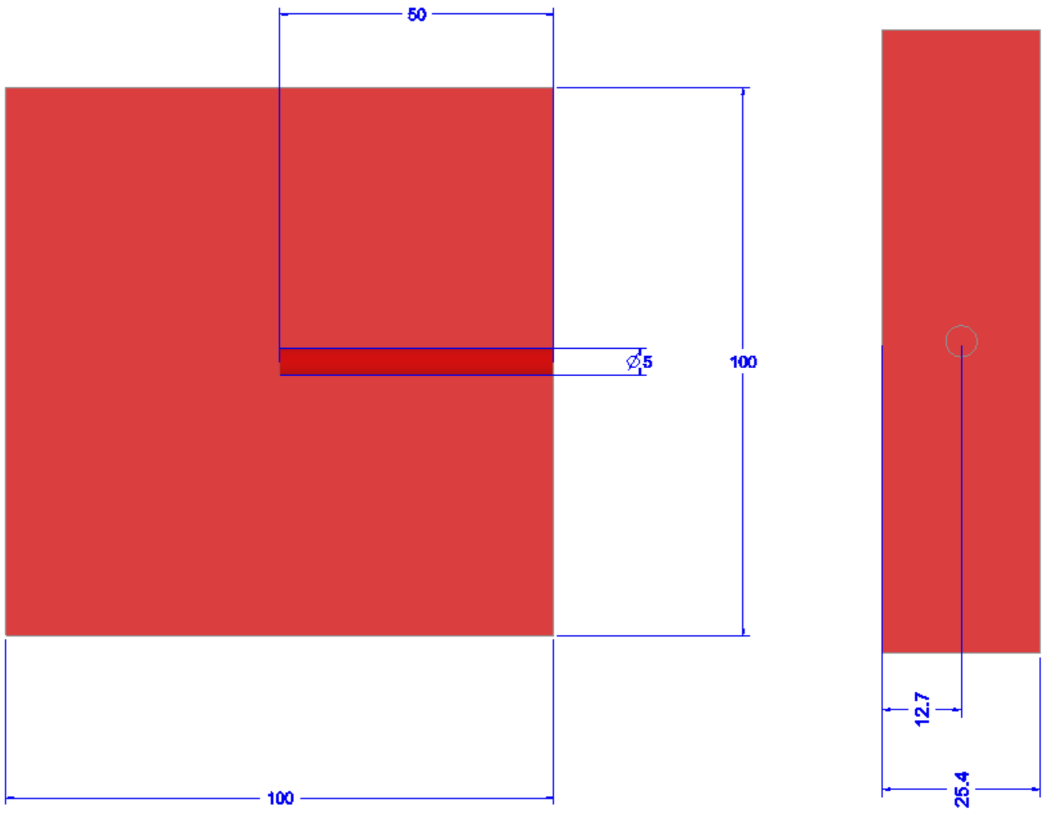


Figure 1 - Component Geometry

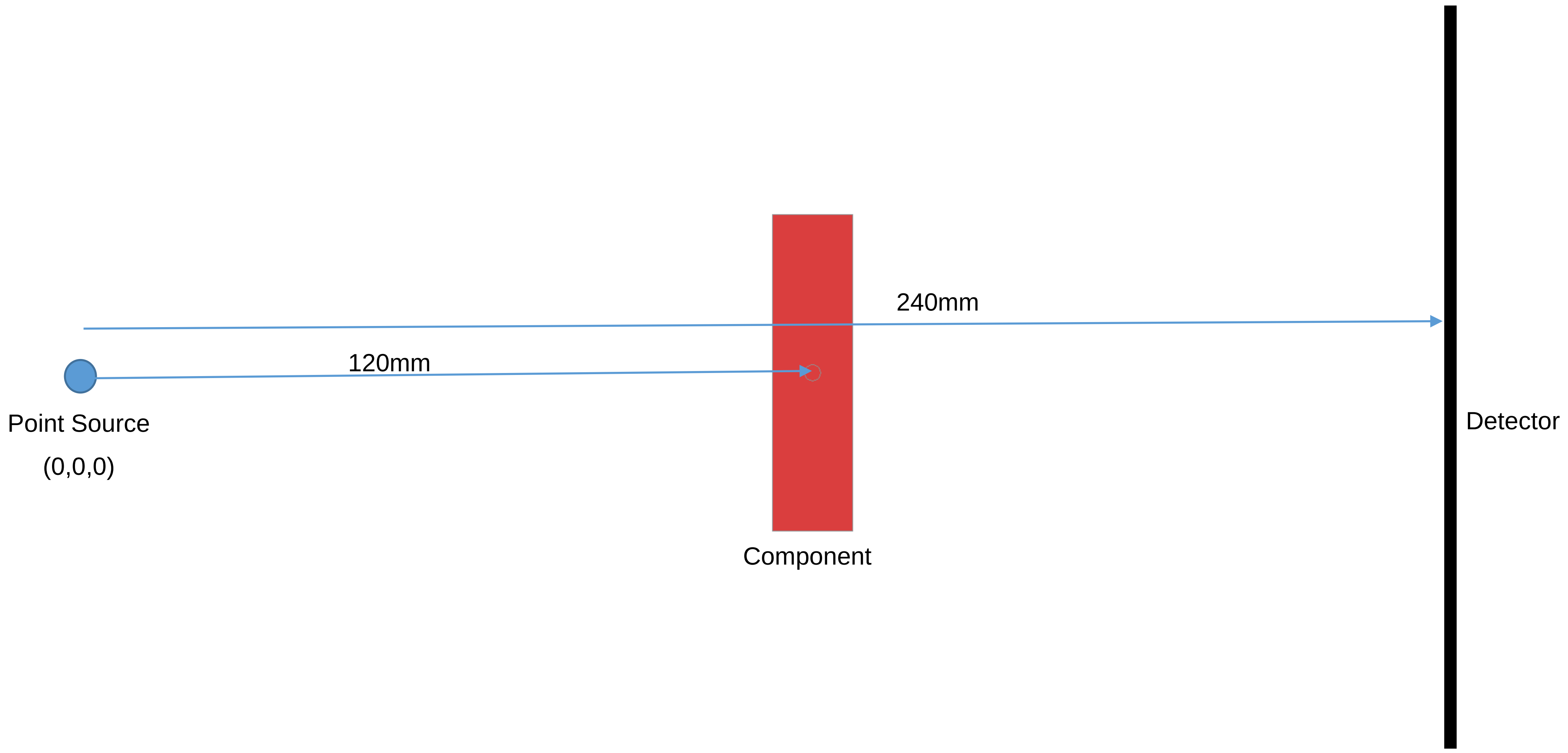


Figure 2 - Side view of Experimental Set Up

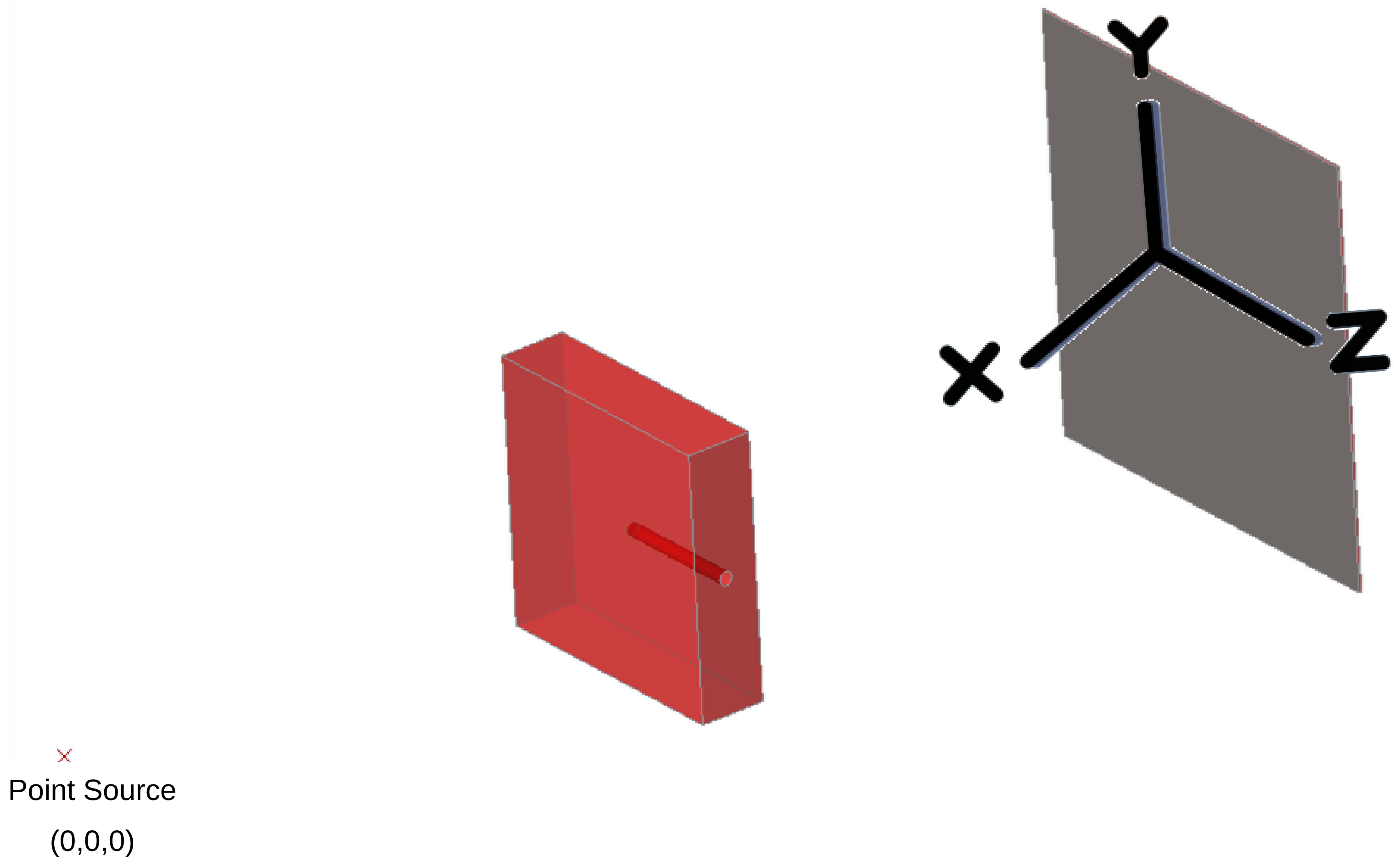


Figure 3 - Isometric View of Experimental Set Up

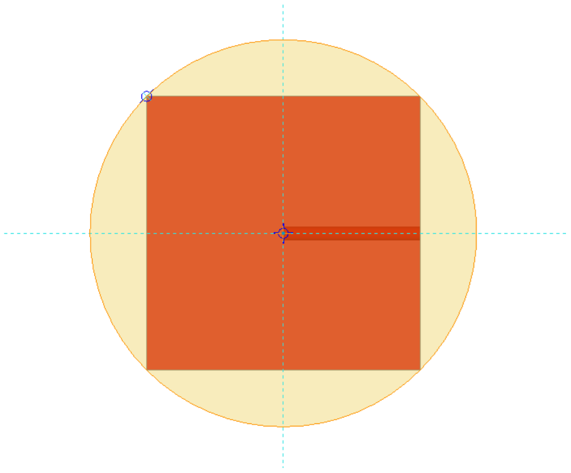
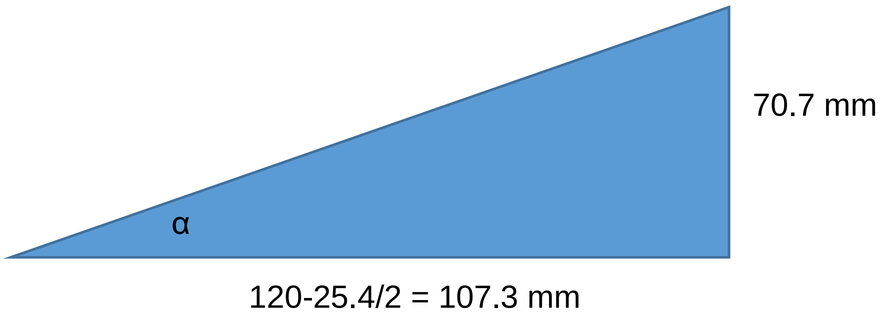
3 EXPLANATION OF MATLAB CODE

Whilst it could be argued that this document is not required, it is felt that a graphical explanation will make the code easier to understand and thus simpler to grade.

The author also acknowledges that he has made certain assumptions about the problem that make it harder than if more simplified assumptions were made.

Assumptions

- Point Source
- 1 MeV Source
- No Scattering
- Steel attenuation at 1MeV = 5.995x10^{-4} mm^2/g
    - https://physics.nist.gov/PhysRefData/XrayMassCoef/ElemTab/z26.html
- Air attenuation at 1MeV = 6.358x10^{-4} mm^2/g
    - https://physics.nist.gov/PhysRefData/XrayMassCoef/ComTab/air.html
- Steel Density = 7.87 g/cm^3
- Air Density = 0.001225 g/cm^3
- X axis is towards the source
- Y axis is the vertical axis of the detector
- Z axis is the horizontal axis of the detector
- The part is a 100 x 100 x 25.4 mm steel block with a 5 mm 50 mm deep side drilled hole.

- The field of view is restricted to be the following:
  - At the first face of the steel block from the source in the x axis, you can construct a circle where the FoV must be wide enough to encompass the part:
  - 
  - The points of intersection of the part with the FoV must be (107.3,50,-50), (107.3,-50,-50), (107.3,-50,50) & (107.3,50,50)
  - The Radius of the circle at this point can be calculated: $x^2 + y^2 = r^2$ at 70.7 mm
  - This can then be projected back along the x axis, to construct a triangle and hence a FoV.
  - 

  - FoV = α = 33.3849^o

- With the FoV defined, a cone can be projected from the point source to the detector. See the diagram below:

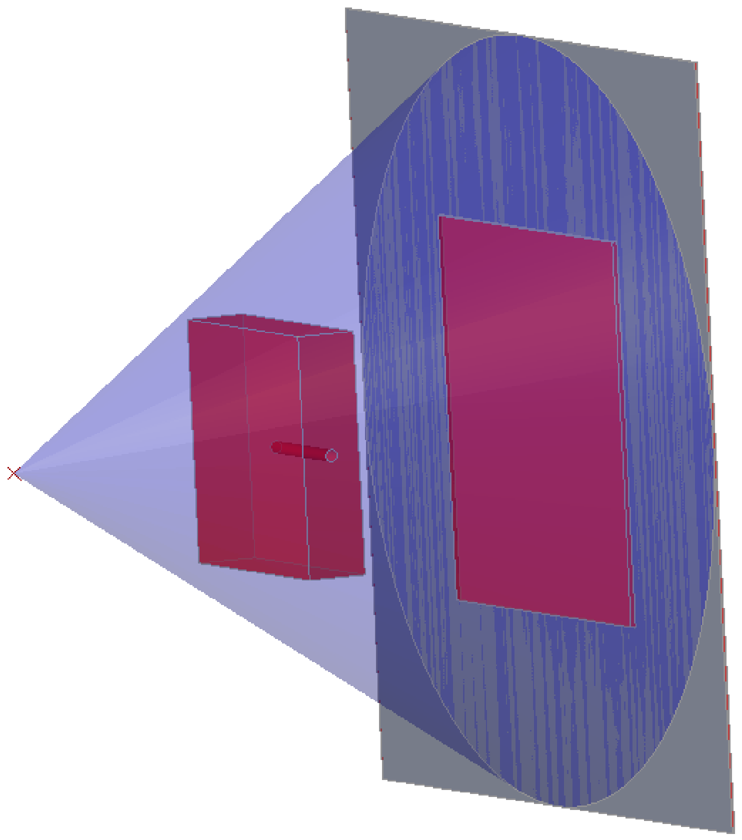
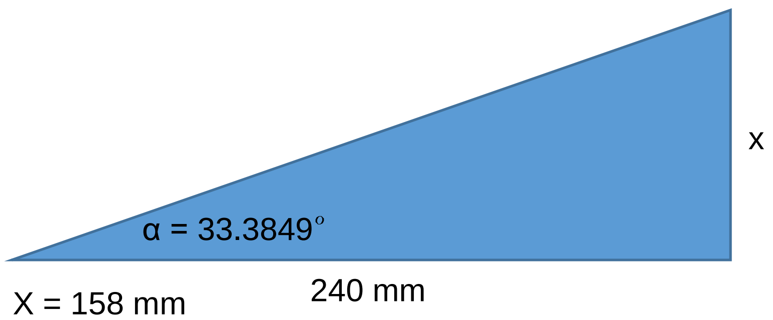


Figure 4 - Diagram of object, FoV and Magnified Object on Detector

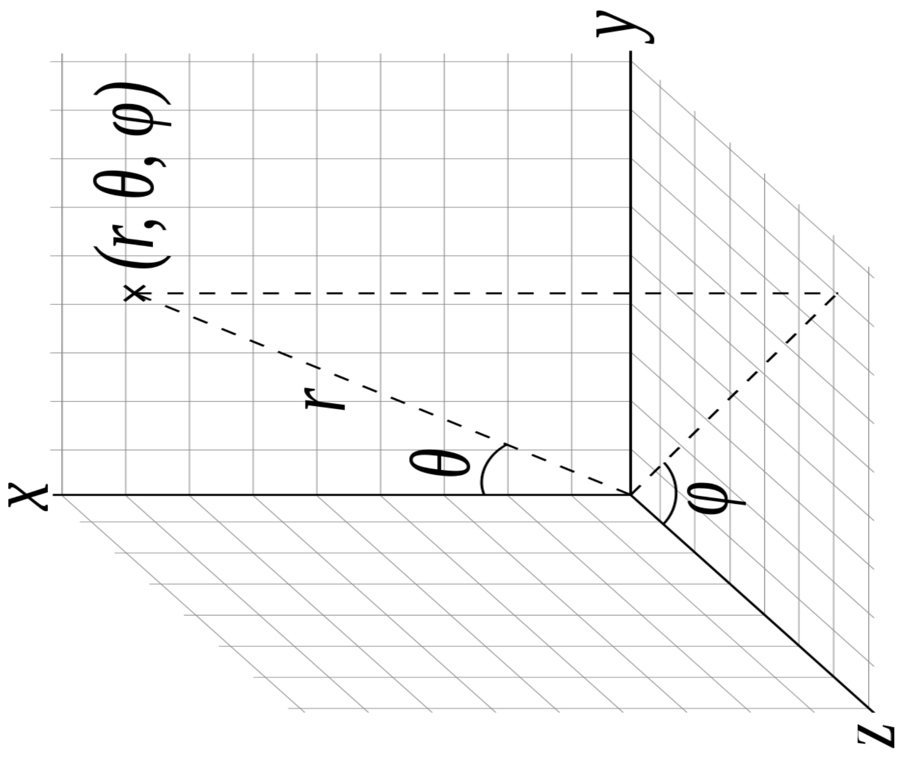
  - This allows for another triangle to be constructed at the detector plane to give the detector dimensions.
  - 

  - X = 158 mm
  - As the part is square, the detector can also be square at 316 x 316 mm. This square detector is tangential to our

- With the detector size defined, it can now be discretised. This is a user variable command in the program. For 500 x 500 pixels, the MATLAB code presented took approximately 218 seconds to solve
- It is suggested that a user try 100 x 100 pixels on the first attempt and up the resolution for future iterations. It is also advised that the resolution be keep the same for both Z and Y dimensions on the detector, otherwise aspect ratio issues will be introduced.

- Note: The author acknowledges that the FoV is not restricted to the above. The actual FoV in reality will be much bigger. This is a fictitious FoV used only to create a detector size for the problem at hand whilst assuming a point source.

- With the detector now discretised, this gives X, Y & Z co-ordinates of each pixel.
    - These X, Y, Z Cartesian coordinates can be converted to spherical coordinates of R,θ, ϕ to allow rays to be projected along.



    -

Where:

  - $x = r\cos(\theta)$
  - $y = rsin(\theta)\cos(\phi)$
  - $z = rsin(\theta)\sin(\phi)$
- Dividing z by y you get:
  - $\frac{z}{y} = \tan(\phi)$
  - $\phi = \operatorname{atan}\left(\frac{z}{y}\right)$
- This in combination with other equations allows for all the pixel X, Y, Z coordinates to be related to angles of θ & ϕ.

- With constant angles of θ & ϕ corresponding to pixels on the detector known, rays can be traced at increments of x and their respective Cartesian coordinates calculated. This effectively projects rays in a conical cuboid. A slice of this conical cuboid is taken at increments of x, their coordinates calculated and evaluated to see if they lie within or without the component.
- This allows the total length of each ray to be calculated in the component and free space, and subsequently allows the relative attenuation to be calculated to the following:
    - $\frac{I}{I_O} = \sum_n e^{-\mu_n l_n}$
    - $where$:
        - μ $is\ the\ attenuation\ coefficient\ of\ each\ material$
        - l $is\ the\ length\ of\ the\ ray\ in\ each\ material$
    - This allows the generation of an image like the one shown below:

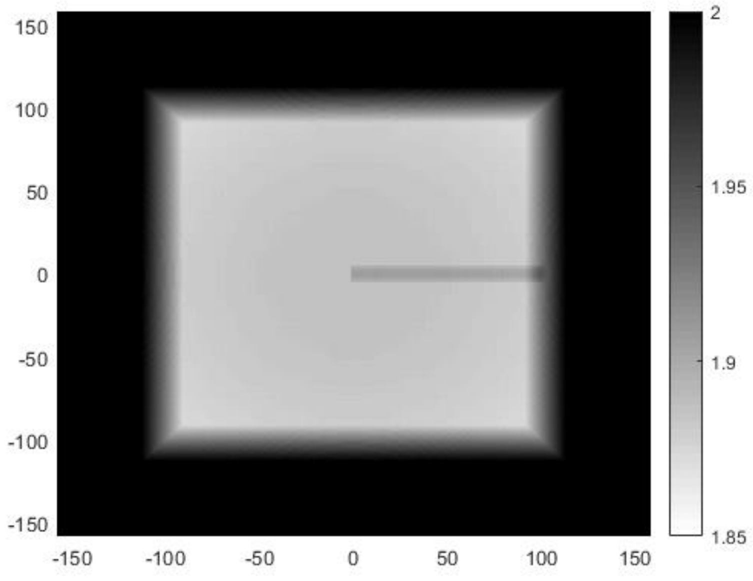


Figure 5 - Simulated X-Ray Image

4 APPENDIX A

```
clear all

%%
%Defining geometry

tic

%Defining the Steel Block Bounding Box
X1 = [107.3; 132.7; 132.7; 107.3; 107.3; 132.7; 132.7; 107.3];
Y1 = [50; 50; -50; -50; 50; 50; -50; -50];
Z1 = [50; 50; 50; 50; -50; -50; -50; -50];

%Defining Hole in Component
x_cyl = 120;
y_cyl = 0;
r_cyl = 2.5;
th = 0:pi/50:2*pi;
X2 = r_cyl * cos(th) + x_cyl;
Y2 = r_cyl * sin(th) + y_cyl;
%Getting arrays in correct format for alphaShape Command
Z2 = [true(length(X2));zeros(length(X2))];
Z2 = Z2(:,1);
Z2 = Z2*50;
X2 = [X2,X2];
X2 = X2.';
Y2 = [Y2,Y2];
Y2 = Y2.';

%Creating alphaShape of component
plate = alphaShape(X1,Y1,Z1);
hole = alphaShape(X2,Y2,Z2);

%Plotting component
figure(01)
clf
plot(plate)
hold on
plot(hole)
axis equal
title('Steel Plate with Hole')
xlabel('X Axis')
ylabel('Y Axis')
zlabel('Z axis')
%%
%Dicretising Detector and Ray Paths

%Calculating FOV and detector size
r_block = sqrt(max(Y1)^2 + max(Z1)^2);
FOV = atan(r_block/(min(X1)));
r_detector = round(240*tan(FOV));

%Discretising the detector in Cartesian Coordinates
prompt = {'Enter Detector Pixels in Y Direction:','Enter Detector Pixels in
Z Direction:'};
```

```
title = 'Input';
dims = [1 35];
definput = {'500','500'};
answer = inputdlg(prompt,title,dims,definput);
pix_y = str2double(answer(1,1)); %720;
pix_z = str2double(answer(2,1)); %1280;

detector_y = round(r_detector);
detector_z = round(r_detector*pix_z/pix_y);
detector_Y = linspace(-detector_y,detector_y,pix_y)';
detector_Z = linspace(-detector_z,detector_z,pix_z);
x = 240;

%Spherical Coordinates at the rectangulate detector
%projecting a rectangular beam through space along x axis
detector_R = round(sqrt(x.^2 + detector_Y.^2 + detector_Z.^2),0);
phi = atan(detector_Y./detector_Z);
theta = asin(detector_Z./(detector_R.*cos(phi)));

%creating 3D array of cartesian coordinates of the rectangular beam/array
inc = 0.1;
X_R = ones(pix_y,pix_z,240/inc);
for i = 1:size(X_R,3)
    X_R(:,:,i)= X_R(:,:,i)*inc*i;
end
Z_R = round((X_R./cos(theta)).*sin(theta).*cos(phi),0);
Y_R = round((X_R./cos(theta)).*sin(theta).*sin(phi)*-1,0);



%%
%Finding the coordinates of the rays that are in the block of steel
%Storing the ray's max/min coordinates in the block of steel and hole
%Calculating length of each ray in the steel and air



%Boolean 3D array of coordinates in and outside the block of steel
tf_plate = inShape(plate,X_R,Y_R,Z_R);
tf_hole = inShape(hole,X_R,Y_R,Z_R);

%Finding & storing the min indexes of the array when rays leave and enter
the block of steel
plate_ind_min = zeros(pix_y,pix_z);
summation_plate = zeros(pix_y,pix_z);
for i = 1:pix_z
    for j = 1:pix_y
           summation_plate(j,i) = round(sum(tf_plate(j,i,:)),0);
           if summation_plate(j,i) ~= 0
               plate_ind_min(j,i) = find(tf_plate(j,i,:),1,'first');
           else
               plate_ind_min(j,i) = 0;
           end
    end
end

%Finding & storing the max indexes of the array when rays leave and enter
the block of steel
plate_ind_max = zeros(pix_y,pix_z);
```

```
for i = 1:pix_z
    for j = 1:pix_y
           if summation_plate(j,i) ~= 0
               plate_ind_max(j,i) = find(tf_plate(j,i,:),1,'last');
           else
               plate_ind_max(j,i) = 0;
           end
    end
end

%Finding storing X, Y & Z Coordinates of rays entering and leaving the
block of steel
X_Max = zeros(pix_y,pix_z);
Y_Max = zeros(pix_y,pix_z);
Z_Max = zeros(pix_y,pix_z);
X_Min = zeros(pix_y,pix_z);
Y_Min = zeros(pix_y,pix_z);
Z_Min = zeros(pix_y,pix_z);

%Finding max coorindates of ray in the block of steel
for i = 1:pix_z
    for j = 1:pix_y
            if plate_ind_max(j,i) ~= 0
                X_Max(j,i) = X_R(j,i,plate_ind_max(j,i));
                Y_Max(j,i) = Y_R(j,i,plate_ind_max(j,i));
                Z_Max(j,i) = Z_R(j,i,plate_ind_max(j,i));
            else
                X_Max(j,i) = 0;
                Y_Max(j,i) = 0;
                Z_Max(j,i) = 0;
            end
    end
end

%Finding min coordinates of ray in the block of steel
for i = 1:pix_z
    for j = 1:pix_y
            if plate_ind_min(j,i) ~= 0
                X_Min(j,i) = X_R(j,i,plate_ind_min(j,i));
                Y_Min(j,i) = Y_R(j,i,plate_ind_min(j,i));
                Z_Min(j,i) = Z_R(j,i,plate_ind_min(j,i));
            else
                X_Min(j,i) = 0;
                Y_Min(j,i) = 0;
                Z_Min(j,i) = 0;
            end
    end
end

%Calculating length of each ray in the block of steel
deltax = X_Max-X_Min;
deltay = Y_Max-Y_Min;
deltaz = Z_Max-Z_Min;
length_steel = sqrt(deltax.^2 + deltay.^2 + deltaz.^2);

%Clearing variables for the hole
```

```
clear('X_Max','Y_Max','Z_Max','X_Min','Y_Min','Z_Min','deltax','deltay','de
ltaz')

%Finding & storing the min indexes of the array when rays leave and enter
the hole
hole_ind_min = zeros(pix_y,pix_z);
summation_hole = zeros(pix_y,pix_z);
for i = 1:pix_z
    for j = 1:pix_y
           summation_hole(j,i) = round(sum(tf_hole(j,i,:)),0);
           if summation_hole(j,i) ~= 0
               hole_ind_min(j,i) = find(tf_hole(j,i,:),1,'first');
           else
               hole_ind_min(j,i) = 0;
           end
    end
end

%Finding & storing the max indexes of the array when rays leave and enter
the hole
hole_ind_max = zeros(pix_y,pix_z);
for i = 1:pix_z
    for j = 1:pix_y
           if summation_hole(j,i) ~= 0
               hole_ind_max(j,i) = find(tf_hole(j,i,:),1,'last');
           else
               hole_ind_max(j,i) = 0;
           end
    end
end

%Finding storing X, Y & Z Coordinates of rays entering and leaving the hole
X_Max = zeros(pix_y,pix_z);
Y_Max = zeros(pix_y,pix_z);
Z_Max = zeros(pix_y,pix_z);
X_Min = zeros(pix_y,pix_z);
Y_Min = zeros(pix_y,pix_z);
Z_Min = zeros(pix_y,pix_z);

%Finding max coorindates of ray in the hole
for i = 1:pix_z
    for j = 1:pix_y
            if hole_ind_max(j,i) ~= 0
                X_Max(j,i) = X_R(j,i,hole_ind_max(j,i));
                Y_Max(j,i) = Y_R(j,i,hole_ind_max(j,i));
                Z_Max(j,i) = Z_R(j,i,hole_ind_max(j,i));
            else
                X_Max(j,i) = 0;
                Y_Max(j,i) = 0;
                Z_Max(j,i) = 0;
            end
    end
end

%Finding min coorindates of ray in the hole
for i = 1:pix_z
    for j = 1:pix_y
```

```
            if hole_ind_min(j,i) ~= 0
                X_Min(j,i) = X_R(j,i,hole_ind_min(j,i));
                Y_Min(j,i) = Y_R(j,i,hole_ind_min(j,i));
                Z_Min(j,i) = Z_R(j,i,hole_ind_min(j,i));
            else
                X_Min(j,i) = 0;
                Y_Min(j,i) = 0;
                Z_Min(j,i) = 0;
            end
    end
end

%Calculating length of each ray in the hole
deltax = X_Max-X_Min;
deltay = Y_Max-Y_Min;
deltaz = Z_Max-Z_Min;
length_hole = sqrt(deltax.^2 + deltay.^2 + deltaz.^2);

%clearing variables
clear('X_Max','Y_Max','Z_Max','X_Min','Y_Min','Z_Min','deltax','deltay','de
ltaz')

%calculating the length of each ray if it passes through the hole
length_steel = length_steel - length_hole;
%calculating the length of each reay in air
length_air = detector_R - length_steel;

%%
%Calculating the total attentuation and plotting result
%attenuation coefficients
alpha_air = 6.358e-04.*0.001225;
alpha_steel = 7.87.*5.995e-04;

%calculating the attenuation of the rays in air and steel
att_air = exp(-1.*alpha_air.*length_air);
att_steel = exp(-1.*alpha_steel.*length_steel);

%calculating total image
att_total = (att_air + att_steel);

toc

figure(02)
clf
pcolor(Z_R(:,:,2400),Y_R(:,:,2400),att_total)
set(gca, 'clim', [1.85,2]);
shading interp;
colormap gray
MOP = colormap;
MOP = flipud(MOP);
colormap(MOP);
colorbar
```