Appendix

A)

Calculates the solid angle, efficiency, and percentage absorbed in the sample and in the detector as a function of energy

Uses parameter for actual detector and Marinelli beaker geometry.

Mass attenuation coefficient for A RANGE OF ENERGY for detector & soil sample (
Units in cm).

Detector radius is 3.89 cm and the height is 4.72cm.
Calculates the solid angle, efficiency, percentage absorbed in the sample and in the detector as a function of energy.
'%' Sign symolizes comment and is not excuted by MATLAB.

MATLAB Code

```
r=1.9450;      % inner radius of detector in cm
thickness=3;   % thickness of the sample
E=[60,100,150,200,400,500,600,800,1000,1500]; % matrix for different energies
DectectorAttp=[0.1058,0.4896,1.6697,3.9679,29.2855,53.0225,83.8852,164.1814,263.7230,570.5800];% PE in
detector
DectectorAttc=[1.4973,1.5518,1.6684,1.7926,2.2573,2.4642,2.6586,3.0200,3.3557,4.1240]; % Compton effect in
detector
SampleAttp=[10.8685,56.4051,207.7590,517.8235,4170.1592,7693.4380,12307.7378,24324.8042,39119.6514,84633.014
3];% PE in sample
SampleAttc=[4.8290,5.1961,5.7047,6.1985,7.9137,8.6582,9.3564,10.6438,11.8350,14.5600];% Compton effect in
sample

n=20000; % no of trials
EfficiencyE=[];  % matrix of efficiency as a function of energy
SolidAngleE=[];  % matrix of solid angle as a function of energy
AbsorbedInSample=[]; % matrix of the percentage of gammas absorbed in the sample
EfficiencyDetector=[]; % matrix of the per of gamma rays detected by the detector that survived the
attenuation in the sample


% Choosing an energy and corresponding attenuation
```

```
for q=1:10
i=1;
Energy= E(q);
Hits=0;           % keeps track of number of gammas that actually hits the detector after attenuation
PotentialHits=0;
SampleSurvivor=0; % no of gamma rays that survived the attenuation in the sample

% creating gamma rays
while i<=n

    % getting random points
    theta1=randi([0,359])+rand();
    z1=(randi([0,649])+rand())/100; % in cm units
    R1=4.25+thickness*(rand())^.5;  % creates random uniform distribution between the range of the thickness
of the sample
    x1=R1*cosd(theta1);
    y1=R1*sind(theta1);

    % getting random trajectory
    R2=15;                           % the length of the trajectory
    theta= acosd(1-2*rand());        % uniform theta distribution in degrees. REMEMBER this is in FLOATING
POINT
    Phi= randi([0,359])+rand();      % uniform Phi distribution in degrees. REMEMBER this is in FLOATING
POINT.
    x2p= R2*sind(theta)*cosd(Phi);   % measured in the source coordinate
    y2p= R2*sind(theta)*sind(Phi);   % measured in the source coordinate
    z2p= R2*cosd(theta);             % measured in the source coordinate
    x2=x2p+x1;                       % measured in the cylinder coordinate
    y2=y2p+y1;                       % measured in the cylinder coordinate
    z2=z2p+z1;                       % measured in the cylinder coordinate

    % counting hits
    Condition=0;                    % initialization

    % checking for exception conditions where Phi blows up

    if (ceil(Phi*10000)/10000== 90.0000) && (ceil(theta*10000)/10000 == 90.0000 ) % comparing up to four
sig-fig
        if z1>=1.78 && z1 <=6.50
            xa=x1;
            xb=x1;
            xSource=xa-x1;     % always zero

            %check for the absolute value of x
            if abs(xa)== r
                % tangent condition so does not matter which direction the
                % trajectory goes. But we are not going to count this as a hit.
                Condition=0;
```

```
elseif abs(xa)<r
    % intersects the circle at two points
    Condition=1;
    % since xaa=x1 the y of the source and the cylinder coordinate is
    % the same
    ya1=(r^2-(xa)^2)^.5;
    yb1=-(r^2-(xb)^2)^.5;

    ySourceCoa1=ya1-y1;    % converting into source coordinate
    ySourceCoa2=yb1-y1;

    % checking for direction

    % for this condition the aPhiPrime1 and aPhiPrime3 will be positive
    % 90 no matter what.
    PhiPrime1= atand(ySourceCoa1/xSource)*1.0000001; % finds the Phi in the source coordinate.
    PhiPrime3= atand(ySourceCoa2/xSource)*1.0000001;

    % for PhiPrime1
    if xa>=0 && ySourceCoa1>=0
        PhiPrime1=PhiPrime1+0;
    elseif xa<0 && ySourceCoa1>=0
        PhiPrime1=PhiPrime1 +180;
    elseif xa<0 && ySourceCoa1<0
        PhiPrime1=PhiPrime1+180;
    else
        PhiPrime1=PhiPrime1+360;
    end

    % for PhiPrime3
    if xa>=0 && ySourceCoa2>=0
        PhiPrime3=PhiPrime3+0;
    elseif xa<0 && ySourceCoa2>=0
        PhiPrime3=PhiPrime3+180;
    elseif xa<0 && ySourceCoa2<0
        PhiPrime3=PhiPrime3+180;
    else
        PhiPrime3=PhiPrime3+360;
    end

    % rounding to four sig digits
    Phi=ceil(Phi*10000)/10000;
    PhiPrime1=ceil(PhiPrime1*10000)/10000;
    PhiPrime3=ceil(PhiPrime3*10000)/10000;

    % check if the PhiPrime == Phi
    if (PhiPrime1==Phi) && (PhiPrime3==Phi)
        Condition=3;
```

```
                PhiPrime2=0;
                PhiPrime4=0;
            end
        else
            % it does not intersect the circle at all
            Condition=0;
        end
    end

else

    %getting x and y projection
    %r= radius of the cylinder

    a=1+(tand(Phi))^2;
    b=2*tand(Phi)*(y1-x1*tand(Phi));
    c=(x1^2)*(tand(Phi))^2-2*x1*y1*tand(Phi)+(y1^2)-r^2;

    xa= (-b+((b^2)-4*a*c)^.5)/(2*a);
    xb= (-b-((b^2)-4*a*c)^.5)/(2*a);

    if (isreal(xa)&& isreal(xb))==1
        Condition=1;

        %Checking for condition 2 (if it hits the detector)
        xSourceCo1=xa-x1;          % finding new value of x prime in the source coordinate
        xSourceCo2=xb-x1;
        zSourceCo1=(xSourceCo1 * cosd(theta))/(sind(theta)*cosd(Phi)); % finds value of z prime
        zSourceCo2=(xSourceCo2 * cosd(theta))/(sind(theta)*cosd(Phi));
        zCylinderCo1=zSourceCo1+z1;        % getting the value of z prime in the cylinder coordinate
        zCylinderCo2=zSourceCo2+z1;

        if  (zCylinderCo1>=1.78 && zCylinderCo1<= 6.5) || (zCylinderCo2>=1.78 && zCylinderCo2<=6.5)
            Condition=2;

            % checking for condition 3 (direction of the trajectory)
            % getting all the values of y

            ya1=((r^2)-(xa^2))^.5;         % finds the y when the trajectory intersects the cylinder
            ya2=-ya1;
            yb1=((r^2)-(xb^2))^.5;
            yb2=-yb1;

            ySourceCo11=ya1-y1;            % converting into source coordinate
            ySourceCo12=ya2-y1;
            ySourceCo21=yb1-y1;
            ySourceCo22=yb2-y1;
```

```
% getting all the values of PhiPrime

PhiPrime11= atand(ySourceCo11/xSourceCo1); % finds the Phi in the source coordinate
PhiPrime12= atand(ySourceCo12/xSourceCo1);
PhiPrime21= atand(ySourceCo21/xSourceCo2);
PhiPrime22= atand(ySourceCo22/xSourceCo2);

% checking for the different quadrant condition
% for PhiPrime 1

if xSourceCo1>=0 && ySourceCo11>=0
    PhiPrime1=PhiPrime11;
elseif xSourceCo1<0 && ySourceCo11>=0
    PhiPrime1=PhiPrime11+180;
elseif xSourceCo1<0 && ySourceCo11<0
    PhiPrime1=PhiPrime11+180;
else
    PhiPrime1=PhiPrime11+360;
end

% for PhiPrime 2

if xSourceCo1>=0 && ySourceCo12>=0
    PhiPrime2=PhiPrime12;
elseif xSourceCo1<0 && ySourceCo12>=0
    PhiPrime2=PhiPrime12+180;
elseif xSourceCo1<0 && ySourceCo12<0
    PhiPrime2=PhiPrime12+180;
else
    PhiPrime2=PhiPrime12+360;
end

% for PhiPrime 3

if xSourceCo2>=0 && ySourceCo21>=0
    PhiPrime3=PhiPrime21;
elseif xSourceCo2<0 && ySourceCo21>=0
    PhiPrime3=PhiPrime21+180;
elseif xSourceCo2<0 && ySourceCo21<0
    PhiPrime3=PhiPrime21+180;
else
    PhiPrime3=PhiPrime21+360;
end

% for PhiPrime 4

if xSourceCo2>=0 && ySourceCo22>=0
    PhiPrime4=PhiPrime22;
```

```
            elseif xSourceCo2<0 && ySourceCo22>=0
                PhiPrime4=PhiPrime22+180;
            elseif xSourceCo2<0 && ySourceCo22<0
                PhiPrime4=PhiPrime22+180;
            else
                PhiPrime4=PhiPrime22+360;
            end

            Phi=ceil(Phi*10000)/10000;
            PhiPrime1=ceil(PhiPrime1*10000)/10000;
            PhiPrime2=ceil(PhiPrime2*10000)/10000;
            PhiPrime3=ceil(PhiPrime3*10000)/10000;
            PhiPrime4=ceil(PhiPrime4*10000)/10000;

            % checking if the trajectory is in the right direction
            if (PhiPrime1==Phi) || (PhiPrime2==Phi) || (PhiPrime3==Phi) || (PhiPrime4==Phi)
                Condition=3;

            end
        end
    end
end
% finding the two right PhiPrime to get the values of y and x
% there is two correct value for PhiPrime=Phi. One for xa and other for xb.

if Condition==3
    PotentialHits=PotentialHits+1;        % keeps track of how many times it fulfilled condition 3

    % PhiPrime1==PhiPrime3 and PhiPrime2==PhiPrime4 so
    % checking for anyone of them is fine
    if Phi==PhiPrime1
        yf1=ya1;
        zf1=zCylinderCo1;
    end
    if Phi==PhiPrime2
        yf1=yb1;
        zf1=zCylinderCo1;
    end
    if Phi==PhiPrime3
        yf2=yb1;
        zf2=zCylinderCo2;
    end
    if Phi==PhiPrime4
        yf2=yb2;
        zf2=zCylinderCo2;
    end

    % finding the distance between the intersection points
```

```
        % and the source point
        distanceSample1 =((xa-x1)^2 + (yf1-y1)^2 + (zf1-z1)^2)^.5; % distance from the source to the first
intersection point
        distanceSample2 =((xb-x1)^2 + (yf2-y1)^2+ (zf2-z1)^2)^.5;  % distance from the source to the second
intersection point


        % Finds which intersection point is the first by
        % comparing the distance from the source to the two
        % intersection points

        if distanceSample2>distanceSample1
            FirstIntersectionDistance= distanceSample1;            % subustracting the gap, 2.3 cm, between
the beaker and the detector
        else
            FirstIntersectionDistance= distanceSample2;
        end


        % attenuation due to compton scattering
        Scs=-(log(1-rand()))*SampleAttc(q);       % For sample
        Sj=-(log(1-rand()))*SampleAttp(q);        % For sample.

        PhotoSample=0;
        ComptonSample=0;
        if Scs>Sj %Photoelectric effect
            if FirstIntersectionDistance < Scs && FirstIntersectionDistance >= Sj
                PhotoSample=1;
                ComptonSample=0;
            end
            if FirstIntersectionDistance < Scs && FirstIntersectionDistance <Sj
                PhotoSample=0;
                ComptonSample=0;
            end
            if FirstIntersectionDistance > Scs
                PhotoSample=1;
                ComptonSample=0;
            end

        end

        if Sj>Scs %Compton Scatter

            if FirstIntersectionDistance < Sj && FirstIntersectionDistance > Scs
                PhotoSample=0;
                ComptonSample=1;
            end
            if FirstIntersectionDistance < Scs && FirstIntersectionDistance <Sj
```

```
            PhotoSample=0;
            ComptonSample=0;
        end
        if FirstIntersectionDistance > Sj
            PhotoSample=0;
            ComptonSample=1;
        end

    end


    if PhotoSample==0 && ComptonSample==0                     % survives the attenuation in the detector
        SampleSurvivor=SampleSurvivor+1;
        distance =((xa-xb)^2 + (yf1-yf2)^2 + (zf1-zf2)^2)^.5; % distance travelled inside the detector
        Sc=-(log(1-rand()))*DectectorAttc(q);                 % For detector
        Si=-(log(1-rand()))*DectectorAttp(q);                 % For detector
        PhotoDetector=0;
        ComptonDetector=0;
        if Sc>Si % Photoelectric effect

            if distance < Sc && distance > Si
                PhotoDetector=1;
                ComptonDetector=0;
            end
            if distance < Sc && distance <Si
                PhotoDetector=0;
                ComptonDetector=0;
            end
            if distance > Sc
                PhotoDetector=1;
                ComptonDetector=0;
            end

        end

        if Si>Sc % Compton Scatter

            if distance < Si && distance > Sc
                PhotoDetector=0;
                ComptonDetector=1;
                Si2=-(log(1-rand()))*13.06;
                if (distance-Sc)>Si2
                    PhotoDetector=1;
                end
            end
            if distance < Sc && distance <Si
                PhotoDetector=0;
                ComptonDetector=0;
```

```
                end
                if distance > Si
                    PhotoDetector=0;
                    ComptonDetector=1;
                    if (distance-Sc)>Si
                        PhotoDetector=1;
                    end
                end

            end

            % Checking if the gamma ray satisfied both conditions
            if PhotoDetector==1
                Hits= Hits+1;           % number of gammas that did not scattered in the detector
            end

        end
    end

    i=i+1;
end
Efficiency=Hits/n;
SolidAngle=PotentialHits/n;
PerAbsorbedInSample=((PotentialHits-SampleSurvivor)/PotentialHits)*100;
PerDetectedByDetector=((SampleSurvivor-Hits)/SampleSurvivor)*100;
EfficiencyE(end+1)=Efficiency;
SolidAngleE(end+1)=SolidAngle;
AbsorbedInSample(end+1)=PerAbsorbedInSample;
EfficiencyDetector(end+1)=PerDetectedByDetector;
end

SolidAngleE
EfficiencyE
AbsorbedInSample
EfficiencyDetector
```