APPENDIX A

TOTAL FLUX DENSITY DISTRIBUTION FROM A HELIOSTAT

Flux distribution on cylindrical surface by multiple aimed points. The following algorithm explains the calculation involved in MATLAB language.

```
clear
% Define window on cylindrical surface
theta = [0:360];    % define dimension of  theta (0 - 360 degree)
L_theta = length (cet);
height = [800:1:1200];  % define dimension of tower height
L_height = length(height);
Flux = zeros (s_height, s_cet);
tower_H =  10;  % tower height 10 m above the ground
radius = 2;  % radius of cylinder
B = 0.3; A = 0.2;  %

% Calculate sun position
day  = 1; month = 3; LT = 10; % input day , month, time
Ls = -105;    %Longitude 0f Thailand
Lloc = -100.53;  %Longitude of Bangkok, Thailand
La = 13.735;    %Latitude of Bangkok
La  = La*pi/180;
```

```
i_d = day;

j_m  =  month;

if          j_m == 1

             dn = i_d;

    elseif j_m ==2

             dn = 31+i_d;

    elseif j_m == 3

            dn = 59+i_d ;

    elseif j_m == 4

            dn = 90+i_d;

    elseif j_m == 5

            dn = 120+i_d;

    elseif j_m == 6

            dn = 151+i_d;

    elseif j_m == 7

            dn = 181+i_d;

    elseif j_m == 8

            dn = 212+i_d;

    elseif j_m==9

            dn = 243+i_d;

    elseif j_m == 10

            dn = 273+i_d;

    elseif j_m == 11

             dn = 304+i_d;

    else

            dn = 335+i_d;

end
```

```
day_angle =  2*pi*(dn-1)/365;

T = day_angle;


Et=229.18*(0.000075+0.001868*cos(T)-0.032077*sin(T)-0.014615*cos(2*T)-
0.04089*sin(2*T))/60;

ST=LT+(4*(Ls-Lloc)/60)+(Et);

H=15*(12-ST)*pi/180;

Dec=23.45*sin(2*pi*(dn+284)/365)*2*pi/360;

Al=asin((cos(La1)*cos(Dec)*cos(H))+(sin(La1)*sin(Dec)));  %Altitude of sun

Az=acos((sin(Al)*sin(La1)-sin(Dec))/(cos(Al)*cos(La1)));  % Azimuth of sun

if    ST <=12

        A_z = Az;

else

     A_z =2*pi-Az ;

end

x = cos(Al)*cos(A_z);

y = cos(Al)*sin(A_z);

z = sin(Al);

s = [x,y,z];

J2 = sqrt((x)^2+(y)^2+(z)^2);

S = s/J2;  %unit vector of sun position relative to tower base


% position of heliostats

load (heli)

L_h = size(heli,1);

for quandant = 1 : 4

                for p = 1:L_h      % calculate (Xo,Yo,Zo) for each heliostat

                     if  quandant == 1
```

```
                    Xo=heli(p,1);

                    Yo=heli(p,2);

                    Zo=heli(p,3);

                elseif quandant == 2

                    Xo=-heli(p,1);

                    Yo=heli(p,2);

                    Zo=heli(p,3);

                elseif  quandant == 3

                    Xo=-heli(p,1);

                    Yo=-heli(p,2);

                    Zo=heli(p,3);

       else

                    Xo=heli(p,1);

                    Yo=-heli(p,2);

                    Zo=heli(p,3);

      end

 D = [ 0-Xo, 0-Yo, tower_H-Zo];

J1 = sqrt((0-Xo)^2+(0-Yo)^2+(tower_H-Zo)^2);

R = D/J1;  % unit vector from mirror to aim point

 Xd = R(1,1);

 Yd = R(1,2);

 Zd = R(1,3);

% find intercept point on cylindrical surface due to ray reflect from center of
mirror by ray tracing technique

F= Xd^2+Yd^2;

G=2*Xo*Xd + 2*Yo*Yd;

K=Xo^2 +Yo^2-radius^2;
```

```
 t10= (-G + sqrt(G^2-4*F*K))/(2*F);
 t20= (-G - sqrt(G^2-4*F*K))/(2*F);
      if   t10 > t20
           t30 = t20;
           t40 = t10;
 else
          t30 = t10;
     end
          t40 = t20;
     if    t30 > 0
           t0 = t30;
        elseif t40 > 0
          t0 =t40;
     else disp (' no t0 ')
    end
```

%% X0p, Y0p , Z0p are coordinate of intercept point on cylindrical surface

```
X0p = Xo + t0*R(1,1);
Y0p = Yo + t0*R(1,2);
Z0p =  tower_H;
Dop = [X0p-Xo,Y0p-Yo,tower_H-Zo];
Jop = sqrt((X0p-Xo)^2+(Y0p-Yo)^2+(tower_H-Zo)^2);
Rs=Jop*tan(0.25*pi/180); % Radius of solar disk
Rop = Dop/Jop; % unit vector from mirror to target
mirror_azi = atan2 (Y0p, X0p); % position azimuth of center point of mirror
if  mirror_azi >= 0
   mirror_azi  = mirror_azi ;
else
```

```
  mirror_azi = 2*pi+ mirror_azi;
end

% find  normal vector of each heliostat

N=(S+Rop);

N = N/abs(N);  %% unit normal vector of each heliostat

Alt = asin(N(1,3))*180/pi;  %Altitude of mirror in degree

Azm = atan2(N(1,2),N(1,1)); % Azimuth of each mirror

if Azm >= 0
  Azm = Azm;
else
  Azm = 2*pi+Azm;
end


%%% vector on mirror surface when mirror parallel with ground ( from
mirror center to four corner of mirror)

u_old1 = [A/2;B/2;0];

u_old2 = [-A/2;B/2;0];

u_old3 = [-A/2;-B/2;0];

u_old4 = [A/2;-B/2;0];

                Euler1 = [cos(pi/2-Alt)  0     sin(pi/2-Alt);
                            0            1    0;
                         -sin(pi/2-Alt)  0    cos(pi/2-Alt)  ] ;
               Euler2 = [cos(Azm)     -sin(Azm)  0;
                         sin(Azm)   cos(Azm)   0
                            0        0           1];
               Euler=Euler2*Euler1;
```

```
% vector on mirror surface when operate with Euler ( from mirror center to four corner of mirror)

u_new1 = Euler*u_old1  ;

u_new2 = Euler*u_old2;

u_new3 = Euler*u_old3  ;

u_new4 = Euler*u_old4;

%%%% position of mirror after operate with Euler

new_0 = [Xo;Yo;Zo];

new_1 = u_new1+[Xo;Yo;Zo];

new_2 = u_new2+[Xo;Yo;Zo];

new_3 = u_new3+[Xo;Yo;Zo];

new_4 = u_new4+[Xo;Yo;Zo];


% find intercept point on image plane  by ray tracing

d=-(Xod*X0p+Yod*Y0p+Zod*Z0p);

% point 0 ( center point of mirror)

t0=-(Xod*new_0(1,1)+Yod*new_0(2,1)+Zod*new_0(3,1)+d)/(Xod^2+Yod^2+Zod^2);

X00p= new_0(1,1) + t0*Rop(1,1);

Y00p= new_0(2,1) + t0*Rop(1,2);

Z00p= new_0(3,1) + t0*Rop(1,3);

% point 1

t1=-(Xod*new_1(1,1)+Yod*new_1(2,1)+Zod*new_1(3,1)+d)/(Xod^2+Yod^2+Zod^2);

X1p= new_1(1,1) + t1*Rop(1,1);

Y1p= new_1(2,1) + t1*Rop(1,2);

Z1p= new_1(3,1) + t1*Rop(1,3);
```

```
% point 2

t2=-(Xod*new_2(1,1)+Yod*new_2(2,1)+Zod*new_2(3,1)+d)/(Xod^2+Yod^2+Zod^2);

X2p= new_2(1,1) + t2*Rop(1,1);

Y2p= new_2(2,1) + t2*Rop(1,2);

Z2p= new_2(3,1) + t2*Rop(1,3);

% point 3

t3=-(Xod*new_3(1,1)+Yod*new_3(2,1)+Zod*new_3(3,1)+d)/(Xod^2+Yod^2+Zod^2);

X3p= new_3(1,1) + t3*Rop(1,1);

Y3p= new_3(2,1) + t3*Rop(1,2);

Z3p= new_3(3,1) + t3*Rop(1,3);

% point 4

t4=-(Xod*new_4(1,1)+Yod*new_4(2,1)+Zod*new_4(3,1)+d)/(Xod^2+Yod^2+Zod^2);

X4p= new_4(1,1) + t4*Rop(1,1);

Y4p= new_4(2,1) + t4*Rop(1,2);

Z4p= new_4(3,1) + t4*Rop(1,3);

%%% find effective area of principal inage

Area_i=1/2*abs(((X1p*Y2p)+(X2p*Y3p)+(X3p*Y4p)+(X4p*Y1p))-((X2p*Y1p)+(X3p*Y2p)+(X4p*Y3p)+(X1p*Y4p)));
```

% calculate principal image size (L_H,L_T) and θ*

```
%Azm_=Azimuth of position mirror measure from south to eastward direction (depend on position)

%Azm_M = Azimuth of mirror (depend on time)

%Azm_S= Azimuth of sun

%Al_S= Altitude of sun
```

```
%Al _M= Altitude of mirror (depend on time)

Azm_= atan2 (Yo,Xo);

if  Azm_>= 0

   Azm_ = Azm_;

else

   Azm_ = 2*pi+Azm_;

end

tower_al =acos ((tower_H-Zo)/Jop);

L_1=sqrt((sin(Azm_M-Azm_))^2*(cos(tower_al))^2+(cos(Azm_M-Azm_))^2);

b1=(sin((pi/2)-Al))^2+(sin(tower_al))^2-2*sin((pi/2)-
Al_S)*sin(tower_al)*cos(Azm_S-Azm_);

b2=2+2*cos(pi/2-Al_S)*cos(tower_al)-2*sin(pi/2-Al_S)*sin(tower_al)*cos(Azm_S-
Azm_);

b=sqrt(b1/b2);

L_2=sqrt((sin(Azm_M-Azm_))^2*(cos(Al_M))^2+(b*sin(tower_al)-cos(Azm_M-
Azm_)*cos(tower_al)*cos(Al_M))^2);

theta_11=asin((sin(Azm_M-Azm_)*cos(tower_al))/L_1);

theta_12=acos((cos(Azm_M-Azm_)/L_1));

sa = asin((b*sin(tower_al)-cos(Azm_M-Azm_)*cos(tower_al)*cos(Alt))/L_2);

sb = acos((sin(Azm_M-Azm_)*cos(Al_M))/L_2);

theta_sta r= (sa-theta_12);

theta_star_dd = theta_star*180/pi;

theta_star_d = (round(abs(theta_star_dd)/5))*5;

if theta_star_d == 50

   fluxXX=flux50;

elseif theta_star_d == 55

   fluxXX=flux55;

elseif theta_star_d == 60
```

```
    fluxXX=flux60;
 elseif theta_star_d == 65
    fluxXX=flux65;
 elseif theta_star_d == 70
    fluxXX=flux70;
 elseif theta_star_d == 75
    fluxXX=flux75;
 elseif theta_star_d == 80
    fluxXX=flux80;
 elseif theta_star_d == 85
    fluxXX=flux85;
 elseif theta_star_d == 90
    fluxXX=flux90;
 elseif theta_star_d == 95
    fluxXX=flux95;
 elseif theta_star_d == 100
    fluxXX=flux100;
 elseif theta_star_d == 105
    fluxXX=flux105;
 elseif theta_star_d ==110
    fluxXX=flux110;
 elseif theta_star_d == 115
    fluxXX=flux115;
 elseif theta_star_d == 120
    fluxXX=flux120;
 elseif theta_star_d == 125
    fluxXX=flux125;
```

```
elseif theta_star_d == 130
    fluxXX=flux130;
 elseif theta_star_d == 135
    fluxXX=flux135;
else
    fluxXX = flux140;
end
%change coordinate
Al_= acos (Rop(1,3));
                          Euler1=[cos(pi-Al_)  0  -sin(pi-Al_);
                                     0          1       0;
                                 sin(pi-Al_)    0   cos(pi-Al_)];

                         Euler2=[cos(Azm_) sin(Azm_)  0 ;
                                  -sin(Azm_) cos(Azm_) 0;
                                        0            0     1];
                         Euler=Euler1*Euler2;
point1=[X1p;Y1p;Z1p]-[X00p;Y00p;Z00p];
point2=[X2p;Y2p;Z2p]-[X00p;Y00p;Z00p];
point3=[X3p;Y3p;Z3p]-[X00p;Y00p;Z00p];
point4=[X4p;Y4p;Z4p]-[X00p;Y00p;Z00p];
Y1=Euler*point1;
Y2=Euler*point2;
Y3=Euler*point3;
Y4=Euler*point4;
Z =[Z1p Z2p Z3p Z4p];
Z = sort(Z);
```

```
Y1=[Y1(1,1) Y2(1,1) Y3(1,1) Y4(1,1)];
Y2=[Y1(2,1) Y2(2,1) Y3(2,1) Y4(2,1)];
Y1=sort(Y1);
Y2=sort(Y2);
h_1=(Y1(1,1)/Rs);
k_1=(Y1(2,1)/Rs);
h_2=(Y2(1,1)/Rs);
k_2=(Y2(2,1)/Rs);
h_3=(Y3(1,1)/Rs);
k_3=(Y3(2,1)/Rs);
h_4=(Y4(1,1)/Rs);
k_4=(Y4(2,1)/Rs);
h1=(abs(sin(theta_star))/sin(theta_star))*(h_1*sin(sa)-k_1*cos(sb));
k1=(abs(sin(theta_star))/sin(theta_star))*(k_1*cos(theta_12)-h_1*sin(theta_11));
h2=(abs(sin(theta_star))/sin(theta_star))*(h_2*sin(sa)-k_2*cos(sb));
k2=(abs(sin(theta_star))/sin(theta_star))*(k_2*cos(theta_12)-h_2*sin(theta_11));
h3=(abs(sin(theta_star))/sin(theta_star))*(h_3*sin(sa)-k_3*cos(sb));
k3=(abs(sin(theta_star))/sin(theta_star))*(k_3*cos(theta_12)-h_3*sin(theta_11));
h4=(abs(sin(theta_star))/sin(theta_star))*(h_4*sin(sa)-k_4*cos(sb));
k4=(abs(sin(theta_star))/sin(theta_star))*(k_4*cos(theta_12)-h_4*sin(theta_11));
x=[h1 h2 h3 h4];
x=sort(x);
h_min=x(1,1);
h_max=x(1,4);
y=[k1 k2 k3 k4];
y=sort(y);
k_min=y(1,1);
```

```
k_max=y(1,4);
Ax=h_min;
Ay=k_min;
Bx=h_max;
By=k_min;
Cx=h_max;
Cy=k_max;
Dx=h_min;
Dy=k_max;
```

% Move the origin of coordinate (ξ^{*},η^{*}) to the corner *A*, calculate Φ_{A}.

```
h1_s=Ax-Ax;
k1_s=Ay-Ay;
h2_s=Bx-Ax;
k2_s=By-Ay;
h3_s=Cx-Ax;
k3_s=Cy-Ay;
h4_s=Dx-Ax;
k4_s=Dy-Ay;
p01=0.05;
h=[h1_s-6*Rs:p01:h2_s+6*Rs];
k=[k1_s-6*Rs:p01:k3_s+6*Rs];
h=h';
l_h=length(h);
l_k=length(k);
inter_h = zeros(1,l_h);
inter_k = zeros(l_k,1);
 for j = 1:l_k
```

```
for i = 1:l_h


  if      h(i) > 1 & k(j) > 1
          flA(i,j) = 1;
  elseif  h(i) < -1 | k(j) < -1
          flA(i,j) =0;
  elseif  h(i) > 1 & k(j) < 1
          h(i)=1;
          n=(1+h(i))/0.05;
          m=(1+k(j))/0.05;
          n=round(n);
          m=round(m);
          flA(i,j) =fluxXX( n+1,m+1);
 elseif   h(i) < 1 & k(j) > 1
          k(j)=1;
          n=(1+h(i))/0.05;
          m=(1+k(j))/0.05;
          n=round(n);
          m=round(m);
          flA(i,j) =fluxXX( n+1,m+1);


  else
          n=(1+h(i))/0.05;
          m=(1+k(j))/0.05;
          n=round(n);
          m=round(m);
          flA(i,j) =fluxXX( n+1,m+1);
```

```
          end
       end
    end
```

% Move the origin of coordinate $\left(\xi^{*},\eta^{*}\right)$ to the corner *B*, calculate Φ_{B}

```
h1_s=Ax-Bx;
k1_s=Ay-By;
h2_s=Bx-Bx;
k2_s=By-By;
h3_s=Cx-Bx;
k3_s=Cy-By;
h4_s=Dx-Bx;
k4_s=Dy-By;
h=[h1_s-6*Rs:p01:h2_s+6*Rs];
k=[k1_s-6*Rs:p01:k3_s+6*Rs];
l_h=length(h);
l_k=length(k);
inter_h = zeros(l_h,1);
inter_k = zeros(1,l_k);
flB=zeros(l_h,l_k);
for j = 1:l_k
    for i = 1:l_h
        if      h(i) > 1 & k(j) > 1
                flB(i,j) = 1;
      elseif   h(i) < -1 | k(j) < -1
                flB(i,j) =0;
       elseif  h(i) > 1 & k(j) < 1
```

```
          h(i)=1;

          n=(1+h(i))/0.05;

          m=(1+k(j))/0.05;

          n=round(n);

          m=round(m);

          flB(i,j) =fluxXX( n+1,m+1);

      elseif  h(i) < 1 & k(j) > 1

          k(j)=1;

          n=(1+h(i))/0.05;

          m=(1+k(j))/0.05;

          n=round(n);

          m=round(m);

          flB(i,j) =fluxXX( n+1,m+1);

      else

          n=(1+h(i))/0.05;

          m=(1+k(j))/0.05;

          n=round(n);

          m=round(m);

          flB(i,j) =fluxXX( n+1,m+1);

       end

    end

  end
```

% Move the origin of coordinate $\left(\xi^{*},\eta^{*}\right)$ to the corner *C*, calculate Φ_C

```
h1_s=Ax-Cx;

k1_s=Ay-Cy;

h2_s=Bx-Cx;
```

```
k2_s=By-Cy;
h3_s=Cx-Cx;
k3_s=Cy-Cy;
h4_s=Dx-Cx;
k4_s=Dy-Cy;
k11=y(1,2)-Cy;
k12=y(1,3)-Cy;
h=[h1_s-6*Rs:p01:h2_s+6*Rs];
k=[k1_s-6*Rs:p01:k3_s+6*Rs];
l_h=length(h);
l_k=length(k);
inter_h = zeros(l_h,1);
inter_k = zeros(1,l_k);
flC=zeros(l_h,l_k);
for j = 1:l_k
   for i = 1:l_h


     if      h(i) > 1 & k(j) > 1
           flC(i,j) = 1;
      elseif   h(i) < -1 | k(j) < -1
           flC(i,j) =0;
     elseif  h(i) > 1 & k(j) < 1
           h(i)=1;
           n=(1+h(i))/0.05;
           m=(1+k(j))/0.05;
           n=round(n);
           m=round(m);
```

```
            flC(i,j) =fluxXX( n+1,m+1);
        elseif   h(i) < 1 & k(j) >1
            k(j)=1;
            n=(1+h(i))/0.05;
            m=(1+k(j))/0.05;
            n=round(n);
            m=round(m);
            flC(i,j) =fluxXX( n+1,m+1);

        else
            n=(1+h(i))/0.05;
            m=(1+k(j))/0.05;
            n=round(n);
            m=round(m);
            flC(i,j) =fluxXX( n+1,m+1);
        end
    end
  end
```

% Move the origin of coordinate $\left(\xi^{*},\eta^{*}\right)$ to the corner *D*, calculate Φ_D

```
h1_s=Ax-Dx;
k1_s=Ay-Dy;
h2_s=Bx-Dx;
k2_s=By-Dy;
h3_s=Cx-Dx;
k3_s=Cy-Dy;
h4_s=Dx-Dx;
```

```
k4_s=Dy-Dy;

k11=y(1,2)-Dy;

k12=y(1,3)-Dy;

h=[h1_s-6*Rs:p01:h2_s+6*Rs];

k=[k1_s-6*Rs:p01:k3_s+6*Rs];

l_h=length(h);

l_k=length(k);

inter_h = zeros(l_h,1);

inter_k = zeros(1,l_k);

flD=zeros(l_h,l_k);

for j = 1:l_k

    for i = 1:l_h


        if      h(i) > 1 & k(j) > 1

                flD(i,j) = 1;

         elseif   h(i) < -1 | k(j) < -1

                flD(i,j) =0;

        elseif  h(i) > 1 & k(j) < 1

                h(i)=1;

                n=(1+h(i))/0.05;

                m=(1+k(j))/0.05;

                n=round(n);

                m=round(m);

                flD(i,j) =fluxXX( n+1,m+1);

       elseif    h(i) < 1 & k(j) > 1

                k(j)=1;

                n=(1+h(i))/0.05;
```

```
            m=(1+k(j))/0.05;
            n=round(n);
            m=round(m);
            flD(i,j) =fluxXX( n+1,m+1);


        else
            n=(1+h(i))/0.05;
            m=(1+k(j))/0.05;
            n=round(n);
            m=round(m);
            flD(i,j) =fluxXX( n+1,m+1);
        end
    end
  end
fl=(flA+flC-flB-flD);
A_i = sum(fl);
flu_i = sum(A_i');
energy = flu_i*(0.05*Rs)*(0.05*Rs);
factor = Area_i/energy;
if quandant == 2
        if  Xo > -0.15 | Yo < 0.15
        fl = 0*fl;
      else
        fl= fl;
      end
    elseif  quandant == 4
      if  Xo  < 0.15 | Yo > -0.15
```

```
      fl = 0*fl;
     else
       fl= fl;
       end
       else
    fl= fl;
      end
h12=[Ax-6*Rs:0.05:Bx+6*Rs];
k12=[Ay-6*Rs:0.05:Cy+6*Rs];
l_h=length(h12);
l_k=length(k12);
km=[ cos(theta_12) cos(sb);
    sin(theta_11) sin(sa)];
Euler_INV=inv(Euler);
cylindrical=zeros(s_height,s_cet);

% transform image plane to x-y-z
for j=1:l_k
  for i=1:l_h
    xx(i,j)=(h12(i)*km(1,1)+k12(j)*km(1,2))*Rs;
    yy(i,j)=(k12(j)*km(2,2)+h12(i)*km(2,1))*Rs;
    Xx(i,j)=xx(i,j)*Euler_INV(1,1)+yy(i,j)* Euler_INV (1,2)+X0p;
    Yy(i,j)=xx(i,j)* Euler_INV (2,1)+yy(i,j)* Euler_INV (2,2)+Y0p;
    Zz(i,j)=xx(i,j)* Euler_INV (3,1)+yy(i,j)* Euler_INV (3,2)+Z0p;
    Xx(i,j)=Xx(i,j)*100;
    Yy(i,j)=Yy(i,j)*100;
    Zz(i,j)=round(Zz(i,j)*100);
```

```
  rho1(i,j)=atan2(Yy(i,j),Xx(i,j));
  rho1(i,j)=rho1(i,j)*180/pi;
  if rho1(i,j) >= 0
    rho1(i,j)= rho1(i,j);
  else
    rho1(i,j)= 360+rho1(i,j);
 end
 rho1(i,j)=round(rho1(i,j));
 tt(i,j)=(sqrt(Xx(i,j)^2+Yy(i,j)^2)-200)/sin(tower_al);
 X(i,j)=Xx(i,j)+tt(i,j)*Rop(1,1);
 Y(i,j)=Yy(i,j)+tt(i,j)*Rop(1,2);
 Z(i,j)=Zz(i,j)+tt(i,j)*Rop(1,3);
 Z(i,j)=round(Z(i,j));
 rho2(i,j)=atan2( Y(i,j),X(i,j));
 rho2(i,j)=rho2(i,j)*180/pi;
 if rho2(i,j) >= 0
   rho2(i,j)= rho2(i,j);
else
   rho2(i,j)= 360+rho2(i,j);
end
rho2(i,j)=round(rho2(i,j));
if fl(i,j) > 1
   fl(i,j) == 1;
else
   fl(i,j) = fl(i,j);
end
cylindrical(Z(i,j),rho2(i,j)+1)=fl(i,j)*sin(tower_al)*factor;
```

```
  end
end
Flux = cylindrical + Flux;
end
end
xlabel('theta(degree)')
ylabel('tower height (cm.)')
zlabel('flux dimensionless / Fo')
title('solar flux density distribution of 12/1/1')
surf(Flux(900:1100,1:360))
view([-27,66])
```

APPENDIX B

RELATION OF SOLAR INTENSITY AND SOLAR BEAM IRRADIANCE

The solar intensity is related to the solar beam irradiance at normal incidence G_{bn}. From equation (3.12) we can draw the relation between $S_o(W.m^{-2}.sr^{-1})$ and the normal flux density $G_{bn}(W.m^{-2})$ at the consider time.

$$G_{bn} = \int_0^{4\pi} S(\alpha) d\Omega$$

where Ω is solid angle

Solid angle is defined as the ration of the area *ds* of a spherical surface to the square of its radius *d*, it can be written as follows

$$d\Omega = \frac{ds}{d^2}$$

$$\int_0^{R_s} d\Omega = \int_0^{R_s} \frac{d\left(\pi R^2\right)}{d^2}$$

$$\Omega = \int_0^{R_s} \frac{2\pi R d(R)}{d^2}$$

$$\Omega = \int_0^{R_s} \frac{2\pi(d \tan\alpha) d(d \tan\alpha)}{d^2}$$

$$\Omega = \int_0^{\alpha_s} \frac{2\pi d^2 \tan\alpha \sec^2\alpha d\alpha}{d^2}$$

$$\Omega = \int_0^{\alpha_s} 2\pi \tan\alpha \sec^2\alpha d\alpha$$

then

$$G_{bn} = \int_{0}^{\alpha_s} S(\alpha) \cdot 2\pi \tan(\alpha)(1 + \tan^2 \alpha) d\alpha$$

REFERENCES

[1] World Energy Council. Measuring Solar Insolation [Online]. Available from http//www. Worldengy.org/wecgeis/publications/reports/ser [2005, Mar 2].

[2] DNT, NESDB, and oil traders in pursuance of section 7. Thailand Energy Situation 2003 [Online]. Available from http//www. dedg.go.th [2005, Mar 2].

[3] Solar PACES. Solar Thermal Power Plants [Online]. Available -

http//www.solarpaces.org/index.html [2005, Mar 2].

[4] เสริม จันทร์ฉาย และคณะ. แผนที่ศักยภาพพลังงานแสงอาทิตย์จากข้อมูลดาวเทียมสำหรับประเทศไทย.ใน แผนที่ศักยภาพพลังงานแสงอาทิตย์เฉลี่ยต่อปี , หน้า 82 กรุงเทพมหานคร: จิรังรัชต์,2542.

[5] Moustafa M. Elsayed and Kadry A. Fathalah. Estimation of percentage useful area of a heliostat when considering shadowing and blocking. Solar & Wind Technology **3** (1986): 199-205.

[6] Lipps, F.W., Vant-Hull, L.L. A cell wise method for the optimization of large central receiver systems. Solar Energy **20** (1978): 505-516.

[7] F.M.F. Siala, M.E. Elayeb. Mathematical formulation of a graphical method for a no-blocking heliostat field layout. Renewable Energy **23** (2001): 77-92.

[8] S. Puliaev, J.L. Penna, E.G. Jillnslki, and A.H. Andrei. Solar diameter observation at Observatorio Nacional in 1998-1999. Astron. Astrophys **143** (2006): 265-267.

[9] M.D. Walzel. F.W. Lipps and L.L. Vant-Hull. A solar flux density calculation for a solar tower concentrator using a two-dimensional Hermite function expansion. Solar Energy **19** (1977): 239-253.

[10] G.A. Smith. " Monte-Carlo ray trace simulation for solar central receiver system" UC-62, under ERDA Contract AT (29-1)-789, University of Houston.10-11 August 1977.

[11] F.W. Lipps and M.D. Walzel. An analytic evaluation of the flux density due to sunlight reflected from a flat mirror having a polygonal boundary. Solar Energy **21** (1978): 113.

[12] G.L. Schrenk. "The role of simulation in the development of solar-thermal energy conversion systems" Proc. 11th Intersociety Energy Conversion Engineering Conf. 12-17 September 1976.

[13] M.M.Elsayed and K.A. Fathalah. Solar flux density distribution using a separation of variable/superposition teachnique. Renewable energy **4** (1994): 77-87.

[14] J. C. Hennet and J.L Abatut. Ananalytical method for reflected flux density calculation. Solar Energy **32**(1984): 357-363.

[15] สมชาย เกียรติกมลชัย และคณะ. รายงานฉบับสมบูรณ์ โครงการศึกษา ออกแบบและสร้างต้นแบบเตาเผาสุริยะอุณหภูมิสูง. ใน การวางตำแหน่งฮีลิโอสแตต. หน้า 100-106 กรุงเทพมหานคร: ฝ่ายวิชาการ จุฬาลงกรณ์มหาวิทยาลัย, 2547.

[16] M.R.Riaz. A Theory of Concentrators of Solar Energy on a Central Receiver for Electric Power Generation. Transactions of ASME **32**(1976): 375-386.

VITAGE

Aparporn Sakulkalavek was born on 29 July 1979 in Petchaburi, Thailand. She has been a student in the Development and Promotion for Science and Technology Talents Project (DPST). She received her Bachelor degree of Science (Second Class Honors) in Physics from Silapakorn University in 2001, and study in Master's degree at Chulalongkorn University in 2002.

Conference Presentations:

1. Aparporn Sakulkalavek and Somchai Kiatgamolchai,"Design of Heliostat field for a Central Receiver System and Solar Flux Energy Calculation", *The 4th Conference on Science and Technology*, Thammasat University, Pathumthani,Thailand, March 16 (2005).

2. Aparporn Sakulkalavek and Somchai Kiatgamolchai, "A Calculation of Solar Flux Distribution on a Cylindrical Receiver Surface of a Central Receiver System", *31st Congress on Science and Technology of Thailand (STT 2005)*, Suranaree University of Technology, Nakhon Ratchasima, Thailand, October 18-20, (2005).