Complete set of Matlab procedures for achieving uniform ray generation

Benoit Beckers[1] and Pierre Beckers[2]

[1]ISA-BTP
Université de Pau et des Pays de l'Adour
benoit.beckers@univ-pau.fr

[2] *University of Liège*
7, rue des Erables, 4122 Plainevaux - Belgique
Pierre.Beckers@ulg.ac.be

1. Introduction

The goal of this work is to provide efficient and accessible Matlab functions for anyone using ray tracing. The proposed methods and functions are considered as the most suitable ones to achieve uniform rays distribution. The foundations of these developments are explained in details in [Beckers & Beckers, 2014]. The figures of this document are giving both a Matlab statement in the header and the related examples below. The tables are displaying complete functions that can be used by a simple "copy paste" operation with the name shown in the table header. The red headers correspond to the equal solid angles technique (*ESA*), while the blue headers refer to the equal view factors (*EVF*) method.

2. Equal solid angle cells generation on the hemisphere

The first illustration consists in generating an equal area or equal solid angle (*ESA*) mesh on the hemisphere. The cells, except the polar one, are defined between two longitudes and two latitudes. In *Figure 1*, we see a dome composed of 145 cells. To generate it, we have just to call the function Bsamd (*Table 1*) with a single input: the number of requested cells (here 145) and a single output: the array *S*, which contains the cumulated number of cells in the nested spherical caps: *S* = [1 8 20 38 60 86 115 145] (*Figure 1*, bottom right).

```
clear all;S=Bsamd(145);figure;Bsa3D(S);
```

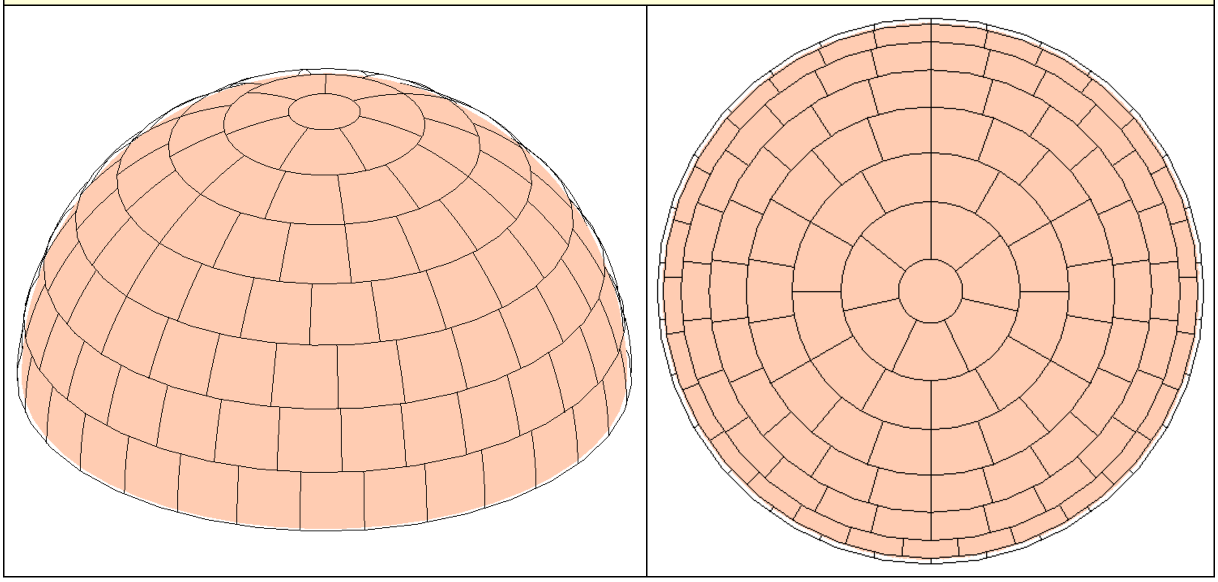


Figure 1: Generation and representation of a 145 ***ESA*** *cells dome, using* **Bsamd** *&* **Bsa3D** *functions*

The statement of *Figure 1* is also calling another function: Bsa3D (*Table 4*) which is producing the drawing, its input is the output of the function Bsamd.

Bsamd (solid angle mesh dome)

```
function [S] = Bsamd(idep)
if nargin == 0, idep = 145; end                    % Default value of the input
Ims    = 1;ccp=7;                                         % Tuning parameters
nring = floor(sqrt(idep));% Defining a tentative realistic number of rings
tp     = ones(1,nring)*pi/2;Sp=ones(1,nring)*idep;nan = 0;% initializations
for i         = 2:nring;
    tp(i)    = tp(i-1)-sqrt(2*pi/idep);
    Sp(i)    = round(Sp(i-1)*(sin(tp(i)/2)/sin(tp(i-1)/2))^2);
    if Ims > 0;tp(i)=2*asin( sin(tp(i-1)/2)*sqrt(Sp(i)/Sp(i-1)));end;
    if Sp(i-1)<ccp;Sp(i-1)=1;Sp(i)=0;end% Forcing presence of central disk
    if Sp(i-1) == 1;if nan == 0;nan = i-1;end; end
end
S=zeros(1,nan);for i=nan:-1:1;S(nan-i+1)=Sp(i);end; % Re-ordering the caps
if size(S,2) < 11;
 disp(['SA Sequence of ',num2str(size(S,2)),' nested caps:  ',num2str(S)])
end
end
```

Table 1: Function **Bsamd***. Equal solid angle cells sequence generation on the hemisphere (ESA)*

3. Equal view factor cells generation on the hemisphere

Instead of generating equal area cells, we can also want to produce equal view factor cells (***EVF***). As before, we can do it with the statement of *Figure 2* header. We observe that this dome is different from the *ESA* one of *Figure 1*. According to the theory, its orthogonal projection in the base plane contains equal area cells (Nusselt analogy, [Beckers & Beckers, 2014]), what is not the situation in the *Figure 1* case. Like in the first example, this statement is using two functions: Bvfmd to generate the sequence of cumulated cells $S = [1 \quad 12 \quad 28 \quad 49 \quad 74 \quad 101 \quad 127 \quad 14]$ and the function Bvf3D (*Table 10*), which is producing the drawing; its input is the output of the function Bvfmd.

```
clear all;S=Bvfmd(145);figure;Bvf3D(S);
```

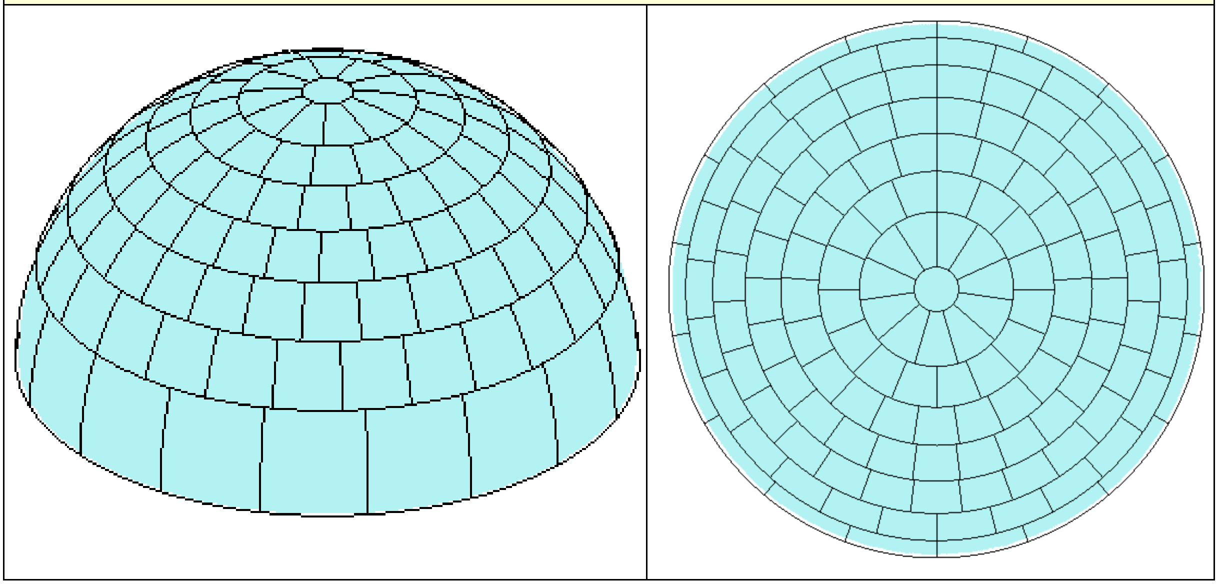


*Figure 2: Generation and representation of a 145 EVF cells dome using **Bvfmd** & **Bvf3D** functions*

Bvfmd (view factor mesh dome)[1]

```
function [S] = Bvfmd(idep)
if nargin == 0, idep = 145; end                   % Default value of the input
Ad     = pi/idep;                                        % Ad = disk cell area
nring = floor(sqrt(idep));% Defining a tentative realistic number of rings
tp     = ones(1,nring)*pi/2;Sp=ones(1,nring)*idep;         % initializations
r      = ones(1,nring);ncr=round(idep/nring);              % initializations
Sp(2) = Sp(1)-ncr;nan=0;                                   % initializations
for i         = 2:nring;
    if Sp(i-1) > 1;
        Sp(i)     = Sp(i-1)-10;
        ar        = 10;
        while ar > 1;
%             arp      = ar;                % Possible optimization Stat. 1
            Sp(i)    = Sp(i)-1;
            ncr      = Sp(i-1)-Sp(i);
            r(i)     = sqrt(r(i-1)^2-Ad*ncr/pi);
            tp(i)    = asin(r(i));
            acell    = 2*pi*(cos(tp(i))-cos(tp(i-1)))/ncr;
            ar       = acell/(tp(i-1)-tp(i))^2;
        end
%         if arp-1 < 1-ar;Sp(i)=Sp(i)+1;end% Possible optimization Stat. 2
        if Sp(i-1)<6;Sp(i-1)=1;Sp(i)=1;end % Forcing presence central disk
        if Sp(i-1) == 1;if nan == 0;nan = i-1;end;end;
    else
        nan=i-1;
    end
end
if Sp(nan) == Sp(nan-1);nan = nan-1;end
S=zeros(1,nan);for i = nan:-1:1;S(nan-i+1)=Sp(i);end; % Re-ordering caps
if S(1) <= 0;S(1)=1;end
if nan < 11;
disp(['VF Sequence of ',num2str(size(S,2)),' nested caps: ',num2str(S)])
end
end
```

*Table 2: Function **Bvfmd**. Equal view factor cells sequence generation on the hemisphere (EVF)*

[1] Procedure modified in April 2018 to remove useless argument.

4. Equal solid angle cells generation on the sphere

In the case of *Figure 3*, we are following the same scheme as before, but with two new functions Bsams and Bsa3Ds given in *Table 3* & *Table 7*.

```
clear all;S=Bsams(145);figure;Bsa3Ds(S);
```

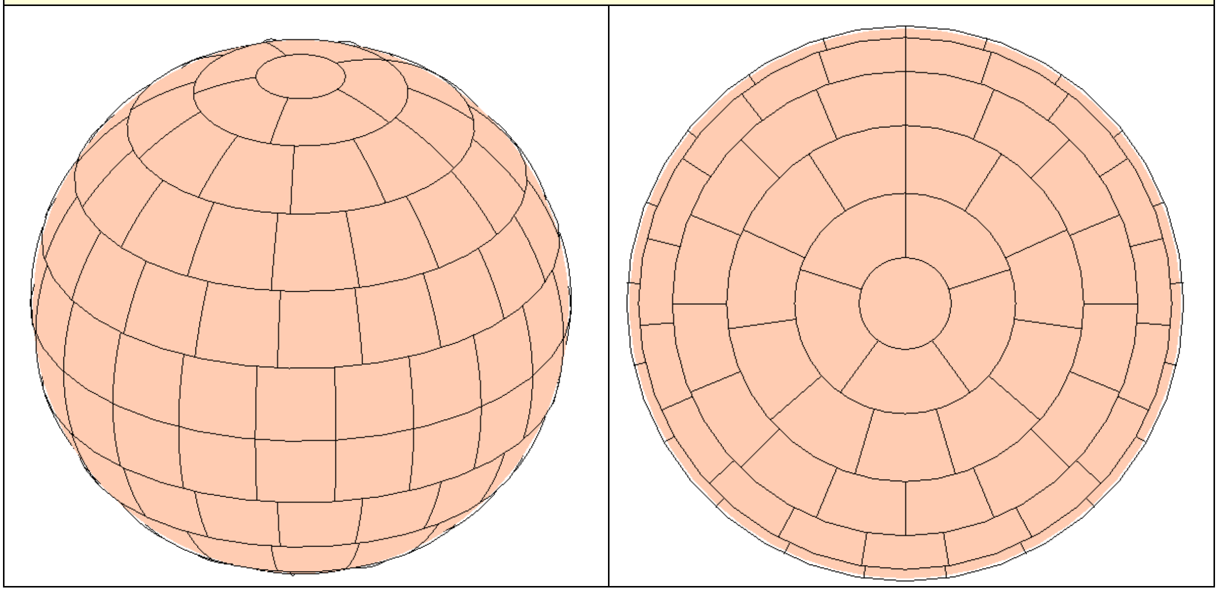


Figure 3: Generation and representation of a 145 ***ESA*** *cells sphere, with* **Bsams** *&* **Bsa3Ds** *functions*

Bsams (solid angle mesh sphere**)**

```
function [S] = Bsams(nsph)
if nargin == 0;nsph = 290;end;                          % Input default value
if nsph < 8;nsph=8;end
nsph  = round(nsph/2)*2;                       % Number of cells must be even
idep  = nsph/2;tim1=pi/2;rim1=sqrt(2);nim1=idep;          % Initializations
nring = floor(sqrt(idep));               % Estimated number of rings
n     = zeros(1,nring);nan = 0;          % initializations
for i      = 1:nring;                    % Loop on the rings or disks
    n(i)   = nim1;                       % Number of cells in disk i
    ti     = tim1-sqrt(2*pi/idep);       % Zenithal angle             (20)
    ri     = 2*sin(ti/2);                % equivalent projection      (16)
    ni     = round(nim1*(ri/rim1)^2);    % Number of cells            (1)
    nim1   = ni;rim1   = ri;tim1   = ti;
    if nim1 == 2;nim1 = 1;end            % Forcing presence of polar disks
    if nim1 == 0;nim1 = 1;end
    if nim1 == 1;if nan == 0;nan = i+1;end;end
end
S = [n(nan:-1:1) 2*n(1)-n(2:nan) nsph];
if size(S,2) < 13;
disp(['SA Sphere sequence of ',num2str(size(S,2)),' layers: ',num2str(S)])
end
end
```

Table 3: Function **Bsams**. *Equal solid angle cells sequence generation on the sphere (****ESA****)*

5. Ray generation

When we benefit from a uniform mesh created on the sphere or the hemisphere, we can easily define rays, either from particular positions in the elements, for instance the barycenter (deterministic rays) or from random points created inside the elements. For the second situation, we take advantage of the quadrangular shape of the elements. It is especially simple to generate random coordinates defined between two latitudes and two longitudes.

```
clear all;S=Bvfmd(600);figure;colo=Bvf3Dd(S);Bsrays(colo);figure;Bsrays(colo);
```

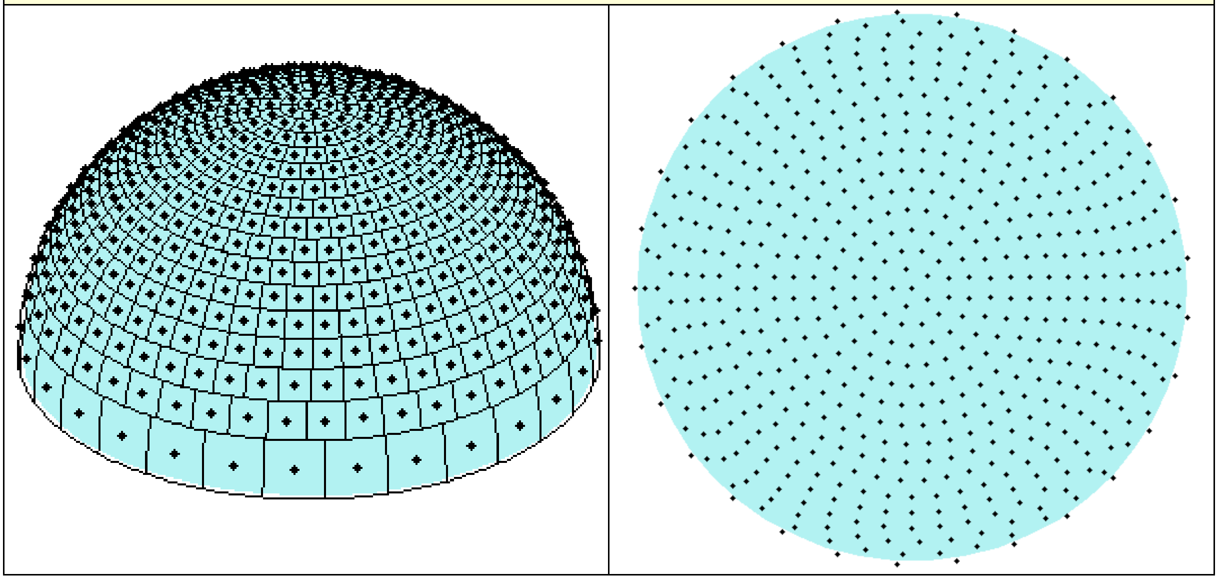


Figure 4: Generation of 600 ***EVF*** *cells & deterministic rays, with* Bvfmd**,** Bvf3Dd *&* Bvrays

```
clear all;S=Bvfmd(600);figure;colo=Bvf3D(S);Bsrays(colo);figure;Bsrays(colo);
```

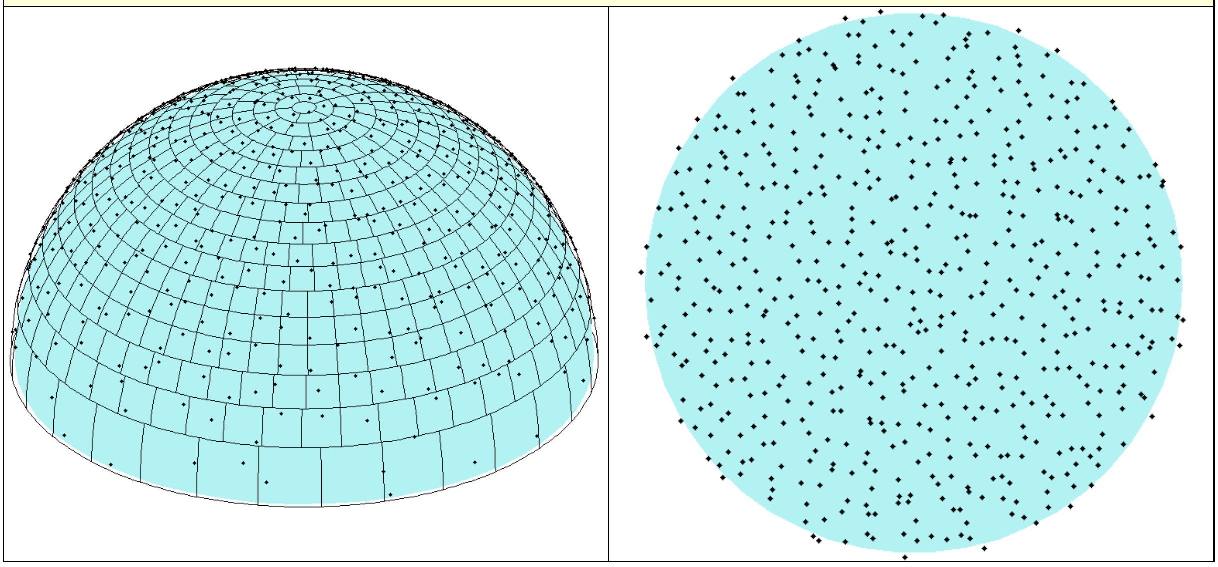


Figure 5: Generation of 600 ***EVF*** *cells and random rays, with* Bvfmd**,** Bvf3D *&* Bvrays *functions*

The left part of *Figure 6* shows an *ESA* mesh of the sphere composed of 490 elements. The right part also contains the random points generated in the cells. The statements used to produce both drawing are presented in the header of the figure. In the second one, we observe the instruction colo = Bsa3Ds(S) indicating that the spherical coordinates of the points are recovered in the array colo. Their representation is obtained throught the function Bvrays (*Table 16*).

Left: `clear all;S=Bsams(490);figure;Bsa3Ds(S);`
Right: `clear all;S=Bsams(490);figure;colo=Bsa3Ds(S);Bvrays(colo)`

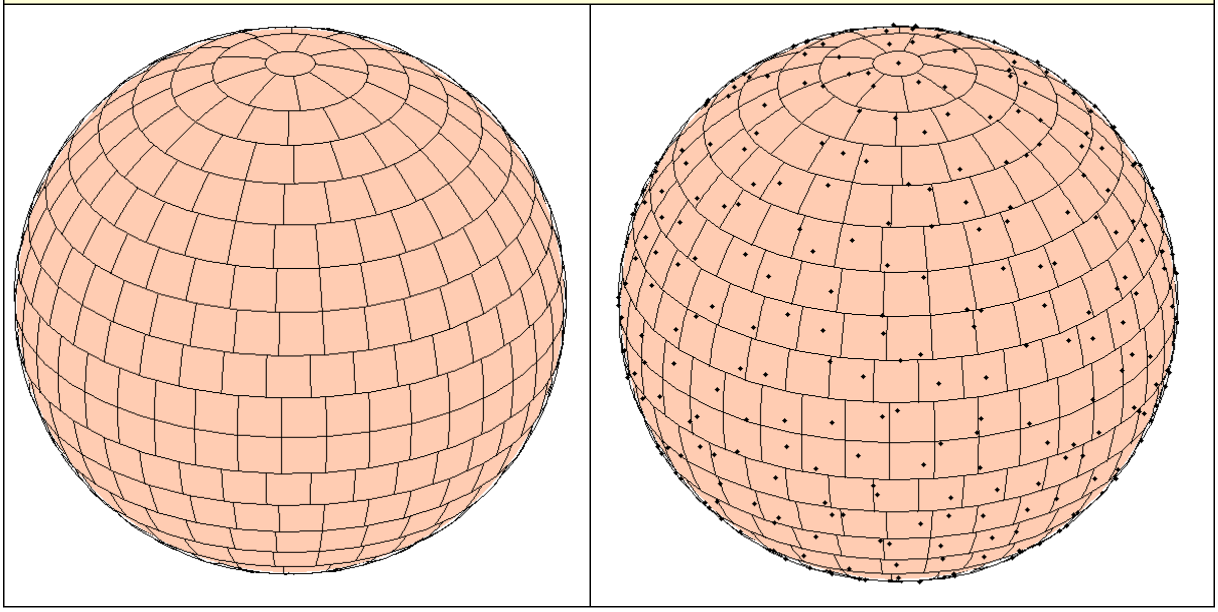


Figure 6: Sphere composed of 490 elements (left) and their 490 superposed random rays (right)

Left: `clear all;S=Bsams(490);figure;colo=Bsa3Ds(S);figure;Bvrays(colo)`
Right: `clear all;S=Bsams(490);figure;colo=Bsa3Dsd(S);figure;Bvrays(colo)`

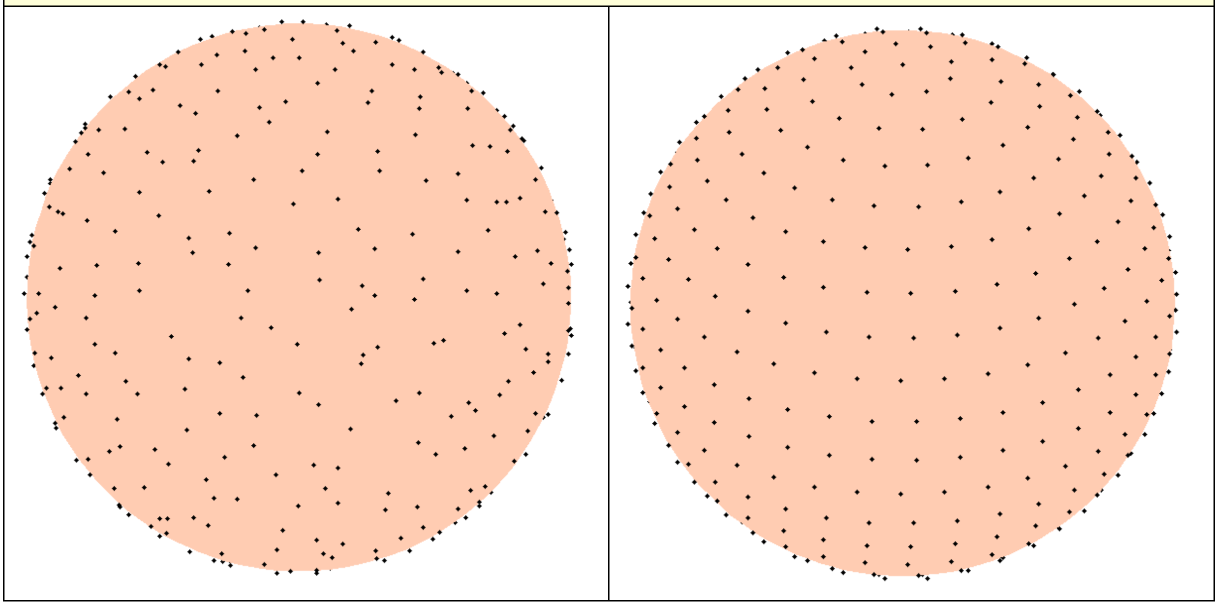


Figure 7: Sphere composed of 490 random rays (left) and 490 deterministic rays (right)

To represent the coordinates of the rays only, it is sufficient, like in *Figure 7*, to reset the drawing by calling the '*figure*' Matlab function. In *Figure 7*, the statements leading to both drawings only differ by calling Bsa3Ds (*Table 7*) for the random rays or Bsa3Dsd (*Table 8*) for the deterministic ones.

Evaluating the cost of a ray generation is performed with the following sequence, with the new function Bsas (*Table 9*), which only computes the spherical coordinates from the sequence *S*. In the header, we have also introduced statements to measure the elapsed time between "*tic*" and "*toc*".

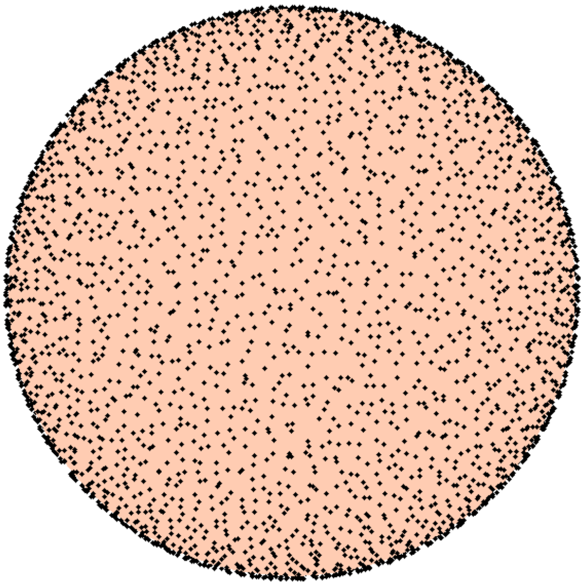
clear all;ini=tic;S=Bsams(5000);colo=Bsas(S);display(toc(ini));figure;Bvrays(colo)	
Cpu $_{\text{5000 rays}}$ = 2.6 seconds Cpu $_{\text{10000 rays}}$ = 10.7 seconds	

Figure 8: Sphere composed of 5000 ESA random rays

We can do the same with the domes by calling functions purely dedicated to the generation of rays. For the view factors, we use Bvf (*Table 11*).

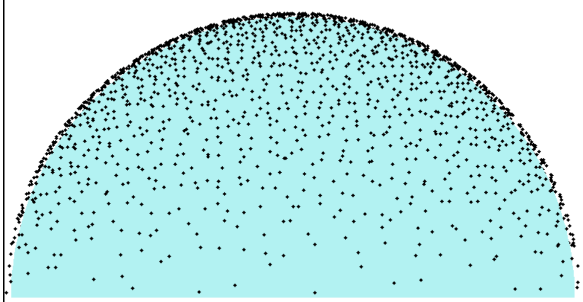
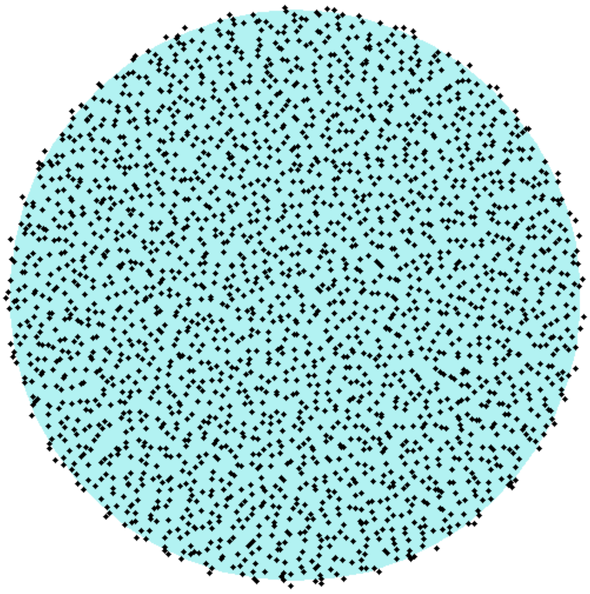
clear all;ini=tic;S=Bvfmd(2500);colo=Bvf(S);display(toc(ini));figure;Bsrays(colo)	
Cpu $_{\text{2500 rays}}$ = 0.8 seconds Cpu $_{\text{10000 rays}}$ =10.9 seconds 	

Figure 9: Dome composed of 2500 EVF random rays

When comparing *Figure 8* and *Figure 9*, we observe that the uniformly distributed points of the sphere seem to present higher density close to the limits of the disk of *Figure 8* that contains their orthogonal projection, while the density appears to be perfectly uniform in *Figure 9*, which corresponds to the orthogonal projection of a non uniform distribution of points on the hemisphere.

6. Conclusion

The procedures given in *Table 1* to *Table 3* are very compact and allow generating meshes composed of cells of equal area and aspect ratio close to 1 (*ESA*) or equal view factor and aspect ratio close to 1 (*EVF*). This set is completed by a set of procedures helping to draw the meshes or to list the coordinates of the rays.

7. Matlab procedures

7.1. Solid angles: rays generation and 3D drawing

The objective of the function of *Table 4* is to draw meshes defined by any array *S* and to compute the spherical coordinates of the set of random rays generated in the two colums array drays, when it is present in the call of the function like in *Figure 10*.

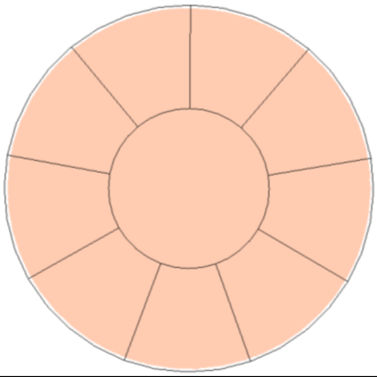
Bsa3D (solid angle 3D dome & random rays)

```
function [drays] = Bsa3D(S)
cs = 20;npas=8;b=(0:10:360)*pi/180;          % Parameter for the execution
Nc = size(S,2);Nt = S(Nc);R  = S(2:Nc)-S(1:Nc-1);% Definition of the rings
aleax=rand(1,Nt);aleay=rand(1,Nt);           % Generation of 2 x Nt random
numbers
t = acos(1-S/Nt);r = sin(t);h = cos(t);        % from disk equal area to 3D
for i= 1 : Nc;             % Drawing the parallels in axnometric projection
    x = r(i)*cos(b);y = r(i)*sin(b);z = ones(size(b,2))*h(i);
    plot3(x,y,z,'k'); hold on;
end
la=zeros(Nt,1);lo=zeros(Nt,1);n=1;
for i = 1 : Nc-1;                                % Drawing the meridians segments
    lp = 0.;
    for j = 1 : R(i);
        longi = j*2*pi/R(i);
        lai   = t(i);las   = t(i+1);pala  = (las-lai)/npas;
        mxi   = sin(lai:pala:las)*cos(longi);
        myi   = sin(lai:pala:las)*sin(longi);
        mzi   = cos(lai:pala:las);
        plot3(mxi',myi',mzi','k');hold on
        if nargout >0;
            n=n+1;
            la(n)=t(i)+aleax(n)*(t(i+1)-t(i));
            lo(n)=lp  +aleay(n)*(longi-lp);lp=longi;
        end
    end
end;
if nargout > 0;
    drays=[la lo];
end
title(['Equal solid angle dome composed of ',num2str(Nt),' elements'],...
    'fontsize',cs);
Bwdo;hold on;axis equal;axis off;
end
```

Table 4: Function **Bsa3D**. *Drawing of the mesh and eventually generation of random rays*

Using the procedures of *Table 1* and *Table 4*, the statement of the header of *Figure 10* is giving the 10 couples of spherical coordinates (in degrees) randomly positionned in the cells. This kind of output is perfectly suited to perform ray tracing based on stratified sampling.

```
clear all;S=Bsamd(10);figure;colo=Bsa3D(S);round(colo*180/pi)
```



0	0
54	24
79	76
65	116
59	144
81	173
32	234
84	258
33	316
59	321

Figure 10: Example of generation, drawing and random set of rays created for stratified sampling

Bsa (*ESA* random rays on a dome)

```
function [drays] = Bsa(S)
Nc = size(S,2);Nt = S(Nc);R  = S(2:Nc)-S(1:Nc-1);% Definition of the rings
aleax=rand(1,Nt);aleay=rand(1,Nt);    % Generation of 2 x Nt random numbers
t = acos(1-S/Nt);                                  % from disk equal area to 3D
la=zeros(Nt,1);lo=zeros(Nt,1);n=1;
for i = 1 : Nc-1;                            % Drawing the meridians segments
    lp = 0.;
    for j = 1 : R(i);
        longi = j*2*pi/R(i);
        n=n+1;
        la(n)=t(i)+aleax(n)*(t(i+1)-t(i));
        lo(n)=lp  +aleay(n)*(longi-lp);lp=longi;
    end
end;
drays=[la lo];
end
```

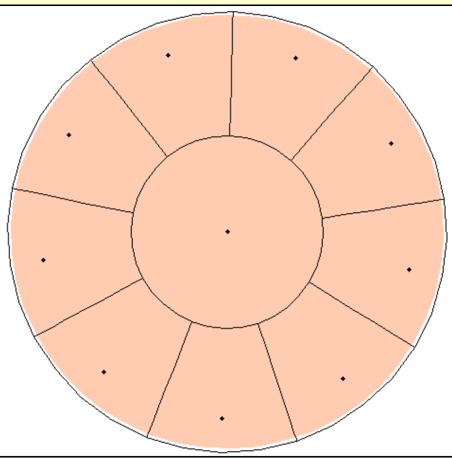
Table 5: Function **Bsa**. *Generation of ESA random rays on a dome*

Bsa3Dd (solid angle 3D dome & deterministic rays)

```
function [drays] = Bsa3Dd(S)
cs = 20;npas=8;b=(0:10:360)*pi/180;          % Parameter for the execution
Nc = size(S,2);Nt = S(Nc);R  = S(2:Nc)-S(1:Nc-1);% Definition of the rings
t = acos(1-S/Nt);                             % from disk equal area to 3D
r = sin(t);
h = cos(t);
for i= 1 : Nc;              % Drawing the parallels in axnometric projection
    x = r(i)*cos(b);y = r(i)*sin(b);z = ones(size(b,2))*h(i);
    plot3(x,y,z,'k'); hold on;
end
la=zeros(Nt,1);lo=zeros(Nt,1);n=1;
for i = 1 : Nc-1;                           % Drawing the meridians segments
    lp = 0.;
    for j = 1 : R(i);
        longi = j*2*pi/R(i);
        lai   = t(i);las   = t(i+1);pala  = (las-lai)/npas;
        mxi   = sin(lai:pala:las)*cos(longi);
        myi   = sin(lai:pala:las)*sin(longi);
        mzi   = cos(lai:pala:las);
        plot3(mxi',myi',mzi','k');hold on
        if nargout >0;
            n=n+1;la(n)=(t(i+1)+t(i))/2;lo(n)=(longi+lp)/2;lp=longi;
        end
    end
end;
if nargout > 0;drays=[la lo];end
title(['Equal solid angle dome composed of ',num2str(Nt),' elements'],...
    'fontsize',cs);Bwdo;hold on;axis equal;axis off;
end
```

Table 6: Function ***Bsa3Dd****. Drawing of the mesh and eventually generation of deterministic rays*

```
clear all; S=Bsamd(10); figure; colo=Bsa3Dd(S);
display(round(colo*180/pi));Bsrays(colo)
```



0	0
58	20
58	60
58	100
58	140
58	180
58	220
58	260
58	300
58	340

Figure 11: Example of generation of the drawing and deterministic set of rays

Bsa3Ds (solid angle 3D sphere & random rays)

```
function [drays] = Bsa3Ds(S)
cs = 20;npas=8;b=(0:10:360)*pi/180;          % Parameter for the execution
Nc = size(S,2)-1;                          % Size of the interior points set
Nt = max(S);                             % Number of cells in the hemisphere
aleax=rand(1,Nt);aleay=rand(1,Nt);                           % Random vectors
R  = S(2:Nc)-S(1:Nc-1) ;                             % Definition of the rings
t = acos(1-2*S/Nt);r = sin(t);h = cos(t);      % From disk equal area to 3D
for i= 1 : Nc;           % Drawing the parallels in axnometric projection
    x = r(i)*cos(b);y = r(i)*sin(b);z = ones(size(b,2))*h(i);
    plot3(x,y,z,'k'); hold on;
end
la=zeros(Nt,1);lo=zeros(Nt,1);n=1;
for i = 1 : Nc-1;                             % Drawing the meridians segments
    lp = 0.;
    for j = 1 : R(i);
        longi = j*2*pi/R(i);lai=t(i);las=t(i+1);pala=(las-lai)/npas;
        mxi   = sin(lai:pala:las)*cos(longi);
        myi   = sin(lai:pala:las)*sin(longi);
        mzi   = cos(lai:pala:las);
        plot3(mxi',myi',mzi','k');hold on
        if nargout >0;
            n=n+1;la(n)=t(i)+aleax(n)*(t(i+1)-t(i));
                  lo(n)=lp  +aleay(n)*(longi-lp);lp=longi;
        end
    end
end;
la(Nt)=pi;lo(Nt)=0;if nargout > 0;drays=[la lo];end
title(['Equal solid angle sphere composed of ',num2str(Nt),...
    ' elements'],'fontsize',cs);Bwba;hold on;axis equal;axis off;
end
```

Table 7: Function **Bsa3Ds**. *Sphere mesh drawing and eventually generation of random rays*

Bsa3Dsd (solid angle 3D sphere & deterministic rays)

```
function [drays] = Bsa3Dsd(S)
cs = 20;npas=8;b=(0:10:360)*pi/180;          % Parameter for the execution
Nc = size(S,2)-1;                          % Size of the interior points set
Nt = max(S);                             % Number of cells in the hemisphere
R  = S(2:Nc)-S(1:Nc-1) ;                             % Definition of the rings
t = acos(1-2*S/Nt);                                 % From disk equal area to 3D
r = sin(t);h = cos(t);
for i= 1 : Nc;           % Drawing the parallels in axnometric projection
    x = r(i)*cos(b);y = r(i)*sin(b);z = ones(size(b,2))*h(i);
    plot3(x,y,z,'k'); hold on;
end
la=zeros(Nt,1);lo=zeros(Nt,1);n=1;
for i = 1 : Nc-1;                             % Drawing the meridians segments
    lp = 0.;
    for j = 1 : R(i);
        longi = j*2*pi/R(i);
        lai   = t(i);las    = t(i+1);pala   = (las-lai)/npas;
        mxi   = sin(lai:pala:las)*cos(longi);
        myi   = sin(lai:pala:las)*sin(longi);
        mzi   = cos(lai:pala:las);
        plot3(mxi',myi',mzi','k');hold on
        if nargout >0;
            n=n+1;la(n)=(t(i+1)+t(i))/2;lo(n)=(longi+lp)/2;lp=longi;
        end
    end
end;
la(Nt)=pi;lo(Nt)=0;if nargout > 0;drays=[la lo];end
title(['Equal solid angle sphere composed of ',num2str(Nt),...
    ' elements'],'fontsize',cs);Bwba;hold on;axis equal;axis off;
end
```

Table 8: Function **Bsa3Dsd**. *Drawing of the mesh and eventually generation of deterministic rays*

Bsas (solid angle 3D sphere & random rays)

```
function [drays] = Bsas(S)
Nc = size(S,2)-1;                              % Size of the interior points set
Nt = max(S);                                  % Number of cells in the hemisphere
aleax=rand(1,Nt);aleay=rand(1,Nt);                                 % Random vectors
R  = S(2:Nc)-S(1:Nc-1) ;                                 % Definition of the rings
t = acos(1-2*S/Nt);                                   % From disk equal area to 3D
la=zeros(Nt,1);lo=zeros(Nt,1);n=1;
for i = 1 : Nc-1;                               % Drawing the meridians segments
    lp = 0.;
    for j = 1 : R(i);
        longi = j*2*pi/R(i);
        n=n+1;la(n)=t(i)+aleax(n)*(t(i+1)-t(i));
              lo(n)=lp  +aleay(n)*(longi-lp);lp=longi;
    end
end;
la(Nt)=pi;lo(Nt)=0;drays=[la lo];
end
```

Table 9: Function **Bsas***. Generation of random rays on the sphere*

7.2. View factors: ray generation and 3D representation

Bvf3D (view factor 3D dome & random rays)

```
function [drays] = Bvf3D(S)
cs = 20;npas=8;b=(0:360)*pi/180;                % Parameter for the execution
Nc = size(S,2);Nt = S(Nc);R  = S(2:Nc)-S(1:Nc-1);% Definition of the rings
aleax=rand(1,Nt);aleay=rand(1,Nt);
r  = sqrt(S/Nt);t  = asin(r);h  = cos(t);
for i= 1 : Nc;                 % Drawing the parallels in axnometric projection
    x = r(i)*cos(b);y = r(i)*sin(b);z = ones(size(b,2))*h(i);
    plot3(x,y,z,'k'); hold on;
end
la=zeros(Nt,1);lo=zeros(Nt,1);n=1;la(1)=aleax(1)*t(1);lo(1)=aleay(1)*2*pi;
for i = 1 : Nc-1;                               % Drawing the meridians segments
    lp=0.;
    for j = 1 : R(i);
        longi = j*2*pi/R(i);
        lai   = t(i);las=t(i+1);pala=(las-lai)/npas;
        mxi   = sin(lai:pala:las)*cos(longi);
        myi   = sin(lai:pala:las)*sin(longi);
        mzi   = cos(lai:pala:las);
        plot3(mxi',myi',mzi','k');hold on
        if nargout >0;
            n=n+1;
            la(n)=t(i)+aleax(n)*(t(i+1)-t(i));
            lo(n)=lp  +aleay(n)*(longi-lp);lp=longi;
        end
    end
end;
if nargout > 0;drays=[la lo];end
title(['Equal view factor dome composed of ',num2str(Nt),' elements'],...
    'fontsize',cs);Bwdo(50);hold on;axis equal;axis off;
end
```

Table 10: Function **Bvf3D***. Dome mesh and eventually generation of random rays*

Bvf (view factor random rays)

```
function [drays] = Bvf(S)
Nc = size(S,2);Nt = S(Nc);R  = S(2:Nc)-S(1:Nc-1);% Definition of the rings
aleax=rand(1,Nt);aleay=rand(1,Nt);
r  = sqrt(S/Nt);t  = asin(r);
la=zeros(Nt,1);lo=zeros(Nt,1);n=1;la(1)=aleax(1)*t(1);lo(1)=aleay(1)*2*pi;
for i = 1 : Nc-1;                           % Drawing the meridians segments
    lp=0.;
    for j = 1 : R(i);
        longi = j*2*pi/R(i);
        n=n+1;
        la(n)=t(i)+aleax(n)*(t(i+1)-t(i));
        lo(n)=lp  +aleay(n)*(longi-lp);lp=longi;
    end
end;
drays=[la lo];
end
```

Table 11: Function **Bvf**. *Generation of random rays on an EVF dome*

Bvf3Dd (Beckers view factor 3D dome & deterministic rays)

```
function [drays] = Bvf3Dd(S)
cs = 20;npas=8;b=(0:360)*pi/180;          % Parameter for the execution
Nc = size(S,2);Nt = S(Nc);R  = S(2:Nc)-S(1:Nc-1);% Definition of the rings
r  = sqrt(S/Nt);t  = asin(r);h  = cos(t);
for i= 1 : Nc;              % Drawing the parallels in axnometric projection
    x = r(i)*cos(b);y = r(i)*sin(b);z = ones(size(b,2))*h(i);
    plot3(x,y,z,'k'); hold on;
end
la=zeros(Nt,1);lo=zeros(Nt,1);n=1;
for i = 1 : Nc-1;                           % Drawing the meridians segments
    lp=0.;
    for j = 1 : R(i);
        longi = j*2*pi/R(i);
        lai   = t(i);las   = t(i+1);pala  = (las-lai)/npas;
        mxi   = sin(lai:pala:las)*cos(longi);
        myi   = sin(lai:pala:las)*sin(longi);
        mzi   = cos(lai:pala:las);
        plot3(mxi',myi',mzi','k');hold on
        if nargout >0;
            n=n+1;la(n)=(t(i+1)+t(i))/2;lo(n)=(longi+lp)/2;lp=longi;
        end
    end
end;
if nargout > 0;drays=[la lo];end
title(['Equal view factor dome composed of ',num2str(Nt),' elements'],...
    'fontsize',cs);Bwdo(50);hold on;axis equal;axis off;
end
```

Table 12: Function **Bvf3Dd**. *Dome mesh and eventually generation of deterministic rays*

7.3. General procedures

The two following procedures are the same, except for the color of the dome. The blue is used for *EVF* situations and the red for *ESA* ones.

Bbdo (blue opaque dome)

```
function [xx,yy,zz] = Bwdo(n)
%   Bwdo generate a white opaque dome (Beckers white dome)
%   [X,Y,Z] = Bwdo(n) generates three (n+1)-by-(n+1)
%   matrices so that SURF(X,Y,Z) produces a unit hemispherical dome.
%
%   [X,Y,Z] = Bwdo uses n = 20.
%
%   Bwdo(n) and just Bwdo graph the sphere as a SURFACE
%   and do not return anything (nargout = 0).
%
if nargin == 0, n = 50; end             % tests the presence of argument
theta      = (-n:2:n)/n*pi;phi       = (0:1:n)'/n*pi/2;
cosphi     = cos(phi)   ; cosphi(1)     = 1; cosphi(n+1) = 0;
sintheta   = sin(theta); sintheta(n+1) = 0;
scal       = .99;                              % radius of the white dome
x          = scal*cosphi*cos(theta);
y          = scal*cosphi*sintheta;
z          = scal*sin(phi)*ones(1,n+1);
blue=[0.7 0.95 0.95];colormap(blue);%blue dome or white: colormap(white)
if nargout == 0                             % computed or returned result
   surf(x,y,z,'EdgeColor','none')
else
   xx = x; yy = y; zz = z;
end
```

Table 13: Function **Bbdo***: creates a blue opaque dome for hidden lines elimination (EVF)*

Bodo (red opaque dome**)**

```
function [xx,yy,zz] = Bodo(n)
%   Bwdo generate a white opaque dome (Beckers white dome)
%   [X,Y,Z] = Bwdo(n) generates three (n+1)-by-(n+1)
%   matrices so that SURF(X,Y,Z) produces a unit hemispherical dome.
%
%   [X,Y,Z] = Bwdo uses n = 20.
%
%   Bwdo(n) and just Bwdo graph the sphere as a SURFACE
%   and do not return anything (nargout = 0).
%
if nargin == 0, n = 50; end              % tests the presence of argument
theta      = (-n:2:n)/n*pi;phi       = (0:1:n)'/n*pi/2;
cosphi     = cos(phi)   ; cosphi(1)     = 1; cosphi(n+1) = 0;
sintheta   = sin(theta); sintheta(n+1) = 0;
scal       = .985;                             % radius of the white dome
x          = scal*cosphi*cos(theta);
y          = scal*cosphi*sintheta;
z          = scal*sin(phi)*ones(1,n+1);
ora=[1 0.8  0.7];colormap(ora);                            % orange dome
if nargout == 0                              % computed or returned result
   surf(x,y,z,'EdgeColor','none')
else
   xx = x; yy = y; zz = z;
end
```

Table 14: Function **Bodo***: creates a red opaque dome for hidden lines elimination (***ESA***)*

Bwba (red opaque sphere)

```
function [xx,yy,zz] = Bwba(n)
%   Bwba generate a white opaque sphere (Beckers white ball)
%   [X,Y,Z] = Bwba(n) generates three (n+1)-by-(n+1)
%   matrices so that SURF(X,Y,Z) produces a unit sphere.
%
%   [X,Y,Z] = Bwba uses n = §0.
%
%   Bwba(n) and just Bwba graph the sphere as a SURFACE
%   and do not return anything (nargout = 0).
%
if nargin == 0, n = 50; end              % test of the presence of argument
theta      = (-n:2:n)/n*pi;sintheta  = sin(theta);
phi        = (-n:2:n)'/n*pi/2;cosphi = cos(phi);
scal       = .99;                                 % radius of the white dome
x          = scal*cosphi*cos(theta);
y          = scal*cosphi*sintheta;
z          = scal*sin(phi)*ones(1,n+1);
ora=[1 0.8  0.7];colormap(ora);
if nargout == 0                                 % computed or returned result
   surf(x,y,z,'EdgeColor','none')    % To see the sphere, replace none by k
else
   xx = x; yy = y; zz = z;
end
```

Table 15: Function **Bwba***: creates an red opaque sphere for hidden lines elimination (***ESA***)*

The two following procedues are also the same except for the color of the dome supporting the rays. The blue is used for *EVF* domes (**Bsrays**) and the red for *ESA* spheres or domes (**Bvrays**).

Bvrays (visualization of rays on spherical surface for *ESA* meshes)

```
function [] = Bvrays(colo)
b=(0:5:360)*pi/180;                         % Definition of the base disk
x  = cos(b);y = sin(b);z = zeros(size(b,2));plot3(x,y,z,'k'); hold on;
la = colo(:,1);lo=colo(:,2);
plot3(sin(la).*cos(lo),sin(la).*sin(lo),cos(la),'.k'); hold on;
if la(size(la,1)) > pi/2;
    Bwba(50);hold on;axis equal;axis off;                    % White sphere
else
    Bwdo(50);hold on;axis equal;axis off;                    % White dome
end
end
```

Table 16: Function **Bvrays***. Representation of the spherical coordinates of the ESA rays*

Bsrays (visualization of rays on spherical surface for EVF meshes)

```
function [] = Bvrays(colo)
b=(0:5:360)*pi/180;                         % Definition of the base disk
x  = cos(b);y = sin(b);z = zeros(size(b,2));plot3(x,y,z,'k'); hold on;
la = colo(:,1);lo=colo(:,2);
plot3(sin(la).*cos(lo),sin(la).*sin(lo),cos(la),'.k'); hold on;
if la(size(la,1)) > pi/2;
    Bwba(50);hold on;axis equal;axis off;                    % White sphere
else
    Bwdo(50);hold on;axis equal;axis off;                    % White dome
end
end
```

Table 17: Function **Bsrays***. Representation of the spherical coordinates of the EVF rays*

7.4. Procedures

Name	Object	Input	Output
Generation of cells sequence			
Bsamd	**Equal solid angle (*ESA*)** dome mesh sequence	Number of cells default value: 145	Sequence of cells
Bsams	***ESA*** sequence for a sphere	Number of cells default value: 290	Sequence of cells
Bvfmd	**Equal view factor**, equal aspect ratio (***EVF***) sequence	Number of cells, default value: 145	Sequence of cells
Generation of rays and 3D representation of the mesh			
Bsa3D ***ESA***	Displays the dome and generates random rays	Sequence of cells	Image, matrix of colatitudes & longitudes
Bsa ***ESA***	Generates random rays on the dome	Sequence of cells	matrix of spherical coordinates colatitudes & longitudes
Bsa3Dd ***ESA***	Displays the dome and generates deterministic rays	Sequence of cells	Image, matrix of colatitudes & longitudes
Bsa3Ds ***ESA***	Displays the sphere and generates random rays	Sequence of cells	Image, matrix of colatitudes & longitudes
Bsas ***ESA***	Generates random rays on the sphere	Sequence of cells	Matrix of spherical coordinates colatitudes & longitudes
Bsa3Dsd ***ESA***	Displays the sphere and generates deterministic rays	Sequence of cells	Image, matrix of colatitudes & longitudes
Bvf3D ***EVF***	Displays the dome and generates random rays	Sequence of cells	Image, matrix of colatitudes & longitudes
Bvf ***EVF***	Generates random rays	Sequence of cells	Matrix of spherical coordinates colatitudes & longitudes
Bvf3Dd *EVF*	Displays the dome and generates deterministic rays	Sequence of cells	Image, matrix of colatitudes & longitudes

Name	Object	Input	Output
General procedures			
Bbdo *EVF*	Creates a blue dome for hidden lines elimination	Size of the mesh Default value: 50	Blue dome drawing
Bodo ***ESA***	Creates an red dome for hidden lines elimination	Size of the mesh Default value: 50	Red dome drawing
Bwba ***ESA***	Creates a red sphere for hidden lines elimination	Size of the mesh Default value: 50	Red sphere drawing
Bvrays ***ESA***	Draws previously computed rays	Matrix colatitudes & longitudes	Points on the dome or the sphere
Bsrays *EVF*	Draws previously computed rays	Matrix colatitudes & longitudes	Points on the dome for *EVF* ***vis***ualizations

8. References

[Beckers & Beckers, 2012], Benoit Beckers & Pierre Beckers, "A general rule for disk and hemisphere partition into equal area cells", *Computational Geometry*, vol. 45, no. 7, pp. 275–283, **2012**.

[Beckers & Beckers, 2013], Benoit Beckers & Pierre Beckers, "Sky vault partition for computing daylight availability and shortwave energy budget on an urban scale", *Lighting Research & Technology*, December **2014** 46: 716-728.

[Beckers & Beckers, 2014], Benoit Beckers & Pierre Beckers, "Reconciliation of Geometry and Perception in Radiation Physics", John Wiley and Sons, Inc., 192 pages, July **2014**.

9. List of tables