Le corps noir

Benoit Beckers & Pierre Beckers

Architecture et Physique Urbaine
Université de Pau et des Pays de l'Adour (France)
Université de Liège (Belgique)

Résumé

Juste un siècle après la découverte du rayonnement infrarouge par William Herschel, en 1800, Max Planck en donne l'explication, par la loi de rayonnement du corps noir. Nous commençons par interpréter la courbe décrite par cette loi, ainsi que sa dérivée (loi de Wien) et son intégrale (loi de Stefan-Boltzmann). La grandeur principale du rayonnement est la radiance, une puissance par unité de surface et par unité d'angle solide. Cette dernière notion est introduite afin de déduire la constante solaire à partir de la température moyenne de la surface du Soleil et de la distance moyenne de cet astre à la Terre. Pour étudier les échanges radiatifs entre les surfaces d'une scène quelconque, il faut ensuite rappeler la notion de facteur de vue, ainsi que l'équation de la radiosité. La distribution des radiosités sur les surfaces d'un cube est finalement montrée sur la projection de Mollweide, qui permet de présenter l'ensemble des résultats sur un seul graphique aux propriétés remarquables : équivalence des aires et horizontalité des parallèles.

1. Le rayonnement électromagnétique

Dans ce document, on se réfère à plusieurs constantes universelles et classiques (*Table 1*).

Constantes du corps noir				
Constante de Planck :	h	=	$6.626069934 \times 10^{-34}$	Js
Vitesse de la lumière dans le vide :	c	=	299792458	ms^{-1}
Constante de Boltzmann :	k	=	$1.3806488 \times 10^{-23}$	JK^{-1}
Constante de déplacement de Wien	b	=	2.8977729×10^{-3}	$m\ K$

Table 1 : Constantes physiques

En 1900, Max Planck postule que l'énergie électromagnétique n'est pas émise en continu (comme par les oscillateurs vibrants), mais par portions discrètes ou quanta [Planck 1900]. Ce résultat est précurseur de la physique moderne et de la théorie quantique. Initialement, la loi de Planck a été développée avec des constantes empiriques, mais il a été montré par la suite que c'est la seule distribution d'énergie stable pour le rayonnement thermique à l'équilibre thermodynamique.

1.1. Loi de Planck

La **loi de Planck** [Planck 1901, 1909] décrit la densité spectrale du rayonnement électromagnétique émis par un corps noir en équilibre thermique à une température T donnée lorsqu'il n'y a pas de flux net de matière ou d'énergie entre le corps et son environnement.

La relation de *Planck – Einstein* relie l'énergie photonique particulière E à sa fréquence d'onde associée ν :

$$E = h\ \nu \tag{1}$$

L'énergie d'un photon de pulsation $\omega = 2\pi\ \nu$ est donnée par $E = \hbar\ \omega$. Dans cette expression, nous utilisons la constante de Dirac (ou constante de Planck réduite) $\hbar = h\ /\ (2\pi)$ exprimée en *Js*. En fonction de la longueur d'onde, la puissance rayonnante spectrale de la fonction de Planck $L_T\ (\lambda)$ est (*Figure 1*) :

$$L_T(\lambda) = \frac{2hc^2}{\lambda^5}\frac{1}{e^{\frac{hc}{\lambda kT}} - 1} \qquad Wm^{-3}sr^{-1} \tag{2}$$

Intégrée sur le domaine des longueurs d'onde, la ***puissance spectrale radiante*** $L_T\ (\lambda)$ ou **radiance spectrale** devient une **radiance** exprimée en $Wm^{-2}sr^{-1}$. Dans le cas d'une onde électromagnétique diffuse homogène émise sur le plan tangent à la surface de l'émetteur, le résultat est multiplié par le facteur π et devient l'**exitance** du corps noir.

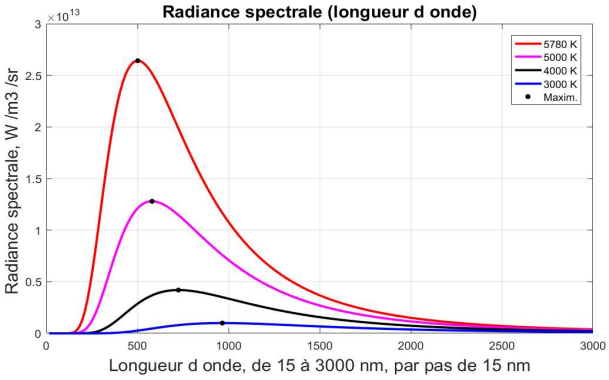

Figure 1 : Radiances spectrales $L_T\ (\lambda)$, Planck_f.m (Table 19)

La fréquence $\nu = c\ /\lambda$ s'exprime en *Hz* ou s^{-1}. La radiance spectrale peut aussi s'écrire comme une fonction de la fréquence $N_T(\nu)$, (*Figure 2*). **N.B. :** Js $s^{-3}m^{-2}$ s^2=W s s $s^{-3}m^{-2}$ s^2= W m^{-2} s

$$N_T(\nu) = \frac{2h\nu^3}{c^2}\frac{1}{e^{\frac{h\nu}{kT}} - 1} \qquad Wm^{-2}s\ sr^{-1} \tag{3}$$

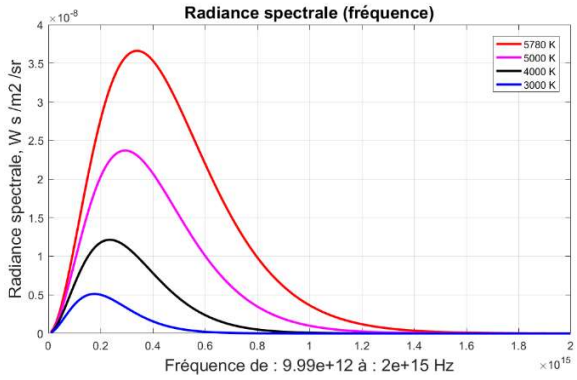

Figure 2 : Radiances spectrales $N_T(\nu)$, Planck_f.m (Table 19)

Les résultats numériques correspondant aux représentations des radiances spectrales sont donnés dans la *Table 2*.

Planck_f.m

```
L 17, Const. de Planck : 6.6261e-34 Js
L 18, Vitesse lumière  : 299792458  ms-1
L 19, Const. Boltzmann : 1.3806e-23 JK-1
L 20, Cst. dépl. Wien  : 0.0028978  mK
L 21, Stefan-Boltzmann : 5.6704e-08 Wm-2K-4
L 22, Tempér. imposées : 5780  5000  4000  3000 K
L 23, Stef.-Boltz. sT4 : 63.3 35.4 14.5 4.59 MW m-2
L 24, Long. onde max.  : 3000 nm
L 25, Nombre de pas    : 200
L 26, Amplitude de pas : 15   nm
L 41, Integrale LT(la) : 62   34.3 13.7 4.09 MW m-2
L 42, Max. rad. LT(la) : 2.64e+13 1.28e+13 4.19e+12 9.95e+11 W/(m2 sr m)
L 43, Position du max. : 501 580 724 966     nm
L 68, Intégrale NT(nu) : 63.3 35.4 14.5 4.59 MW m-2
L 69, Maximum NT(nu)   : 3.66e-08 2.37e-08 1.21e-08 5.12e-09 W/(m2 sr s)
L 75, Date, CPU, 18-Apr-2024, 2.3143 s
```

Table 2 : Résultats de la procédure Planck_f.m (Table 19), extension : 3000 nm

1.2. Loi de Wien

La loi de Wien [Buckingham 1912], [Soffer *et al.* 1999], [Heald 2003] et [Overduin 2003] établit que :

$$\lambda_{max} T = b = 2.897771955... \; 10^{-3} \quad mK \tag{4}$$

Cette loi stipule que la courbe de rayonnement du corps noir pour différentes températures culmine à une longueur d'onde inversement proportionnelle à la température. Cette relation se déduit facilement de (2) en calculant le maximum de la courbe ; *b* est appelée **constante de déplacement de Wien**. Grâce à cette relation, il est aisé de situer la courbe de luminance spectrale dans le spectre.

Les températures affichées en *L 22* (*Table 2*, *Table 5*) correspondent aux pics donnés en *L 42* et localisés aux positions λ_{max} calculées en (4) et affichées en *L 43* (*Table 2*, *Table 5*).

Pour une température donnée, la paramétrisation par fréquence implique une longueur d'onde maximale différente de la paramétrisation par longueur d'onde, car les définitions des fonctions (2) et (3) ne sont pas les mêmes. En effet, on peut facilement vérifier à partir de (2) et (3) que :

$$\frac{L_T(\lambda)}{N_T(\nu)} = \frac{2hc^2}{\lambda^5} \frac{c^2}{2h\nu^3} = \frac{c^4}{\lambda^5 \nu^3} = \frac{c}{\lambda^2} = \frac{\nu}{\lambda} \tag{5}$$

On en déduit :

$$\lambda \, L_T(\lambda) = \nu \, N_T(\nu) \tag{6}$$

Cette relation montre explicitement la relation non linéaire entre $L_T(\lambda)$ et $N_T(\nu)$. Elle confirme qu'il n'y a aucune raison de trouver les maximas des deux fonctions aux mêmes positions.

Pour la même température, mais en paramétrant par fréquence, la fréquence de radiance spectrale maximale est $\nu = 352{,}735$ *THz* avec la longueur d'onde correspondante $\lambda = 849{,}907$ *nm*.

Pour la température du Soleil T = 5780 K, la loi de Wien (4), donne : λ_{max} = 5.013 10-^{7} m (≈ 501 nm (nanomètres). Les diagrammes peuvent être affichés de différentes manières, comme, par exemple, dans les *Figure 3* & *Figure 4*, où les zones sont séparées par une médiane qui correspond à la fréquence ou à la longueur d'onde telle que la moitié de l'intensité totale du rayonnement tombe de chaque côté [Heald 2003]. Comme prévu, la radiance spectrale maximale affichée dans le titre de la *Figure 3* est identique à *L 42* de la *Table 2*.

Les *Figure 3* & *Figure 4* sont créées dans la procédure Matlab© *Separation_radiance.m* (*Table 22*). En considérant les positions des médianes dans les courbes ci-dessus : 710,5 *nm* sur la *Figure 3*, 421946 *GH* sur la *Figure 4*, on vérifie que leur produit est égal à la vitesse de la lumière : 7,105 10^{-7} *m* x 4,21946 10^{14} *Hz* = 299792633 *m/s*. La position de la médiane est la même dans les deux diagrammes. Cependant, comme commenté plus haut, les positions des luminances spectrales maximales sont différentes et ne se situent pas dans les mêmes zones (rouge ou bleue) des diagrammes.

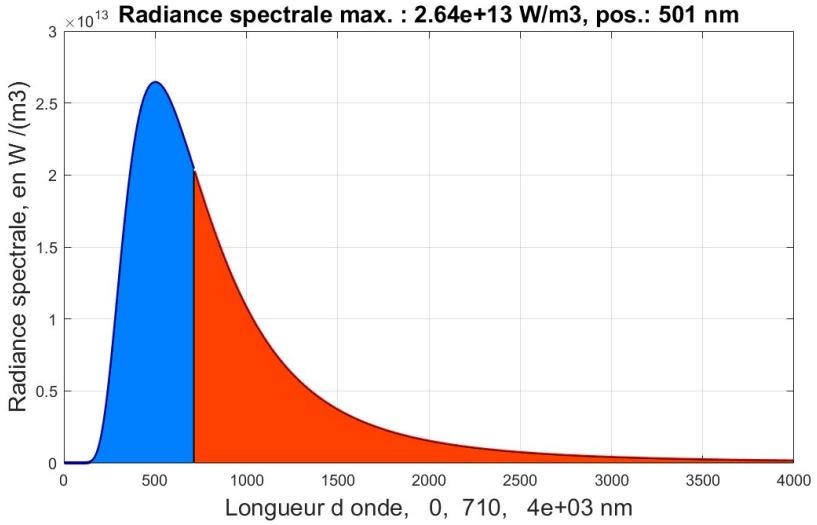

Figure 3 : Rad. spectrale L_T (λ), T = 5780 K, zones séparées par la médiane, Separation_radiance.m

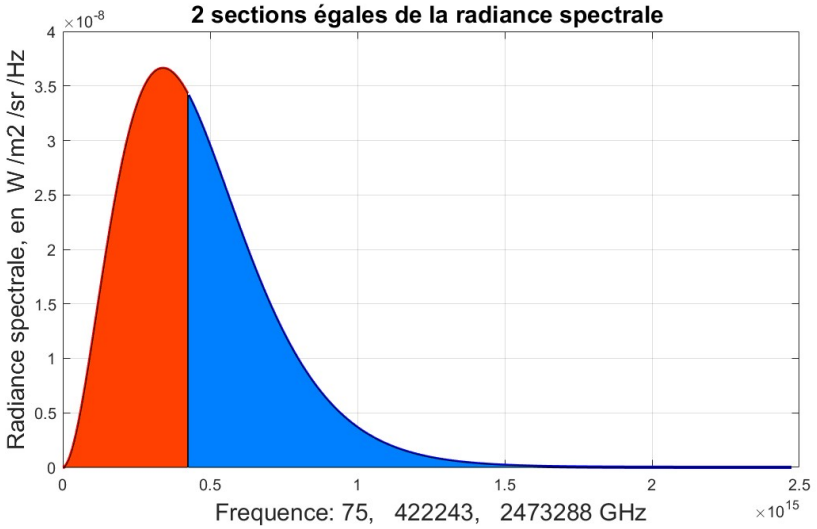

Figure 4 : Rad. spectrale N_T(ν), T = 5780 K, zones séparées par la médiane, Separation_radiance.m

Separation_radiance.m

```
L 12, Const. de Planck : 6.62607e-34 Js
L 13, Vitesse lumière  : 2.99792e+08 m/s
L 14, Cst de Boltzmann : 1.38065e-23 J/K
L 15, Stefan-Boltzmann : 5.6704e-08  W/(m2K4)
L 16, Cst depl de Wien : 0.00289777  mK
L 17, Nombre de pas    : 1000
L 18, Temp. corps noirs: 5780      K
L 19, Long. onde min.  : 0         m
L 20, Long. onde max.  : 4e-06     m
L 21, Interv lon. ondes: 4e-09     m
L 31, Maximum radiance : 2.64e+13 W/m3
L 32, Pos. of the max. : 501       nm
L 43, Est. / exact SB  : 0.99027
L 49, xmin lam(m) xmax : 0 7.1e-07 4e-06  m
L 72, Minimum frequ.   : 75        GHz
L 73, Maximum frequ.   : 2.47e+06 GHz
L 74, freq at max. rad.: 4.22e+14 Hz
L 75, Est. / exact SB  : 1
L 80, Date, CPU, 18-Apr-2024, 0.23498 s
```

Table 3 : Résultats de la procédure Separation_radiance.m (Table 22), 5780 K

Des diagrammes similaires sont construits pour l'émission à 288 *K* (*Figure 5* & *Figure 6*). Ici aussi, le produit des positions médianes satisfait la relation : 14265 10^{-9} *m* x 21016 10^{9} *Hz* = 299793240 *m/s* (vitesse de la lumière).

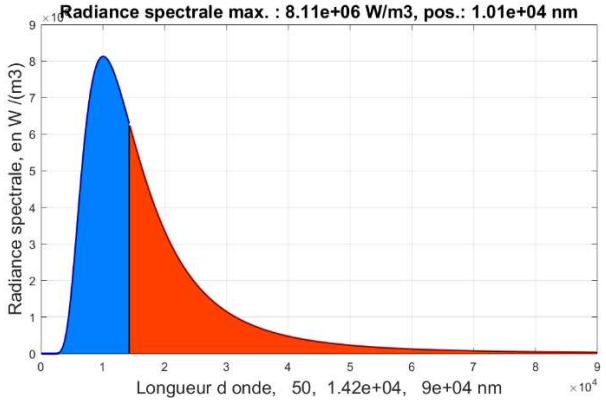

Figure 5 : Radiance spectrale $L_T(\lambda)$, T = 288 K, séparées par la médiane, Separation_radiance.m

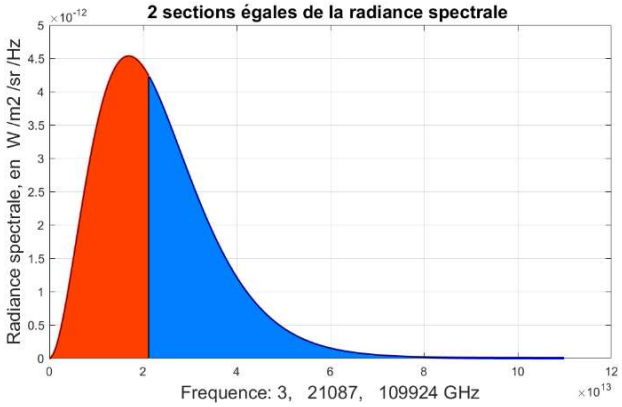

Figure 6 : Radiance spectrale $N_T(\nu)$, T = 288 K, séparées par la médiane, Separation_radiance.m

Separation_radiance.m

```
L 12, Const. de Planck : 6.62607e-34 Js
L 13, Vitesse lumière  : 2.99792e+08 m/s
L 14, Cst de Boltzmann : 1.38065e-23 J/K
L 15, Stefan-Boltzmann : 5.6704e-08  W/(m2K4)
L 16, Cst depl de Wien : 0.00289777  mK
L 17, Nombre de pas    : 1000
L 18, Temp. corps noirs: 288       K
L 19, Long. onde min.  : 5e-08     m
L 20, Long. onde max.  : 9e-05     m
L 21, Interv lon. ondes: 8.995e-08 m
L 31, Radiance maximum : 8.11e+06  W/m3
L 32, Position du max. : 1.01e+04  nm
L 43, SB  est. / exact : 0.99291
L 49, xmin lam(m) xmax : 5e-08 1.4217125e-05 9e-05  m
L 72, Minimum frequ.   : 3        GHz
L 73, Maximum frequ.   : 1.1e+05  GHz
L 74, Freq at max. rad.: 2.11e+13 Hz
L 75, SB est. / exact  : 1
L 80, Date, CPU, 18-Apr-2024, 0.20044 s
```

Table 4 : Résultats de la procédure Separation_radiance.m (Table 22), 288 K

Les intégrations des radiances spectrales définies en (2) et (3) donnent les mêmes exitances.

$$\pi\int_0^\infty L_T(\lambda)\,d\lambda=\pi\int_0^\infty \frac{2hc^2}{\lambda^5}\frac{1}{e^{\frac{hc}{k\lambda T}}-1}\,d\lambda=\pi\int_0^\infty N_T(\nu)\,d\nu=\pi\int_0^\infty \frac{2h\nu^3}{c^2}\frac{1}{e^{\frac{h\nu}{kT}}-1}\,d\nu \tag{7}$$

Le développement de la première intégrale donne :

$$\pi\int_0^\infty L_T(\lambda)\,d\lambda=\pi\int_0^\infty \frac{2hc^2}{\lambda^5}\frac{1}{e^{\frac{hc}{k\lambda T}}-1}\,d\lambda \tag{8}$$

La longueur d'onde λ est remplacée par la grandeur adimensionnelle $x=\frac{hc}{\lambda kT}$. Cela donne :

$$\pi\int_0^\infty \frac{2hc^2}{\lambda^5}\frac{1}{e^{\frac{hc}{k\lambda T}}-1}\,d\lambda=\pi\int_0^\infty \frac{2hc^2}{\lambda^5}\frac{1}{e^x-1}\,d\lambda=-\pi\int_0^\infty -\frac{2ckT}{\lambda^3}\frac{dx}{e^x-1} \qquad \text{comme } d\lambda=-\lambda^2\frac{kT}{hc}dx \tag{9}$$

On sépare la partie à intégrer qui est fonction de x:

$$\pi\int_0^\infty \frac{2ckT}{\lambda^3}\frac{dx}{e^x-1}=\frac{2\pi(kT)^4}{h^3c^2}\int_0^\infty \frac{x^3}{e^x-1}dx \tag{10}$$

Cette expression contient une fonction qu'il est possible d'intégrer explicitement[1].

$$\int_0^\infty \frac{x^3}{e^x-1}\,dx=\frac{\pi^4}{15} \tag{11}$$

[1] https://math.stackexchange.com/questions/99843/contour-integral-for-x3-ex-1

$$\pi\int_0^\infty L_T(\lambda)\,d\lambda = \frac{2\pi(kT)^4}{h^3c^2}\int_0^\infty \frac{x^3}{e^x-1}dx = \frac{2\pi^5k^4}{15c^2h^3}T^4 \quad Wm^{-2} \tag{12}$$

On intègre maintenant la radiance spectrale définie en (3) en y incluant le facteur π :

$$\pi\int_0^\infty N_T(\nu)\,d\nu = \pi\int_0^\infty \frac{2h\nu^3}{c^2}\frac{1}{e^{\frac{h\nu}{kT}}-1}d\nu = \pi\int_0^\infty \frac{2h}{c^2}\frac{\nu^3}{e^{\frac{h\nu}{kT}}-1}d\nu \quad Wm^{-2} \tag{13}$$

Comme précédemment, on remplace la fréquence ν par la variable adimensionnelle $x = \frac{h}{kT}\,\nu$:

$$\pi\int_0^\infty \frac{2h}{c^2}\frac{\nu^3}{e^{\frac{h\nu}{kT}}-1}d\nu = \pi\int_0^\infty \frac{2h}{c^2}\frac{\nu^3}{e^x-1}\frac{kT}{h}dx = \frac{2\pi(kT)^4}{c^2h^3}\int_0^\infty \frac{x^3}{e^x-1}dx \tag{14}$$

En introduisant ce résultat dans (1.9), on a :

$$\pi\int_0^\infty N_T(\nu)\,d\nu = \frac{2(kT)^4}{c^2h^3}\pi\int_0^\infty \frac{x^3}{e^x-1}dx = \frac{2(kT)^4}{c^2h^3}\frac{\pi^5}{15} = \frac{2}{15}\frac{\pi^5k^4}{c^2h^3}T^4 \quad Wm^{-2} \tag{15}$$

Selon (12) ou (14) et la relation définissant la constante de Dirac ou de Planck réduite $\hbar = h/(2\pi)$ = 1.054571783 x 10^{-34} *Js*, on obtient la constante de Stefan-Boltzmann.

$$\sigma = \frac{2}{15}\frac{\pi^5k^4}{c^2h^3} \quad \text{ou} \quad \sigma = \frac{\pi^2k^4}{60\,c^2\hbar^3} \qquad Wm^{-2}K^{-4} \tag{16}$$

1.3. Loi de Stefan-Boltzmann

La loi de Stefan-Boltzmann établit la valeur de l'*exitance* produite par le rayonnement d'un corps noir :

$$Q = \sigma T^4 \quad Wm^{-2} \quad \text{avec : } \sigma = 5.670373\ 10^{-8}\ Wm^{-2}K^{-4} \tag{17}$$

L'intégration de l'expression (2) est effectuée numériquement en (18). Le résultat Q_{is} est comparé à celui de (17) :

$$Q_{is} = \pi\int_{\lambda_i}^{\lambda_s} L_T(\lambda)\,d\lambda = \pi\int_{\lambda_i}^{\lambda_s} \frac{2hc^2}{\lambda^5}\frac{1}{e^{\frac{hc}{k\lambda T}}-1}d\lambda \quad Wm^{-2} \tag{18}$$

Le rapport Q_{is}/Q est calculé dans la procédure Matlab *Exitance.m* (*Table 21*). A la *ligne 6*, on introduit les données du calcul : la température (en *K*), ainsi que les longueurs d'onde λ_i et λ_s (en nm) limitant l'intervalle. A la *ligne 18* de la procédure, on évalue l'intégrant de la relation *(18)* au milieu des *n* segments utilisés pour découper l'intervalle λ_i - λ_s. En additionnant ces valeurs, en les multipliant par la largeur des intervalles et par π, puis en divisant par σT^4 (*ligne 20*), on obtient le rapport souhaité.

Ligne	λ (Nanomètre)	Q_{is}/Q	Température (K)
1	380 - 760	0.44741	5780
2	400 - 700	0.36675	5780
3	0 - 4000	0.99027	5780

4	4000 - 100000	0.00972	5780
5	4000 - 100000	0.99327	288
6	7500 - 13000	0.33887	288
7	7500 - 13000	0.00132	5780
8	0 - 50000	0.96545	288
9	0 - 100000	0.99471	288

Table 5 : Résultats de la procédure Exitance.m : noir : soleil, bleu : terre, rouge : 2 bandes

Quelques cas sont présentés à la *Table 5*. Les deux lignes affichées en rouge correspondent aux températures moyennes de la surface du Soleil (5780 *K*) et de la Terre (15 °C, soit 288 *K*). La longueur d'onde de 4 *microns* (4000 *nm*) sépare presque parfaitement le rayonnement solaire (à 99% sous les 4 microns, ce sont les *ondes courtes*, *ligne 3*) du rayonnement terrestre (à 99% au-dessus de 4 *microns*, ce sont les *ondes longues*, *ligne 5*).

Dans la littérature, l'intervalle de la lumière visible se situe entre 380 *nm* et 760 *nm* (45 % du rayonnement solaire), ou bien entre 400 *nm* et 700 *nm* (37% du rayonnement solaire). L'affichage simultané des radiances spectrale du Soleil et d'une planète comme la Terre est réalisé à la *Figure 7* dans la procédure *Planck_biban.m* de la *Table 20*.

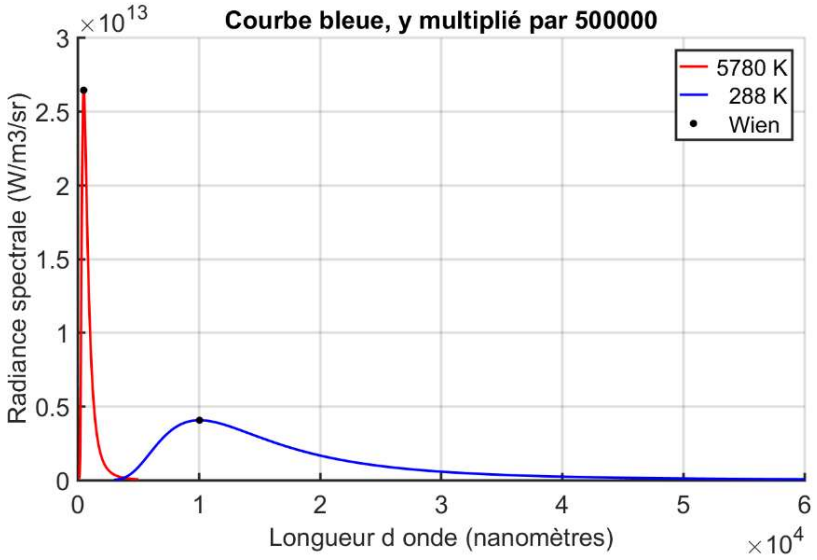

Figure 7 : Radiance spectrale du soleil (rouge) et de la terre (bleu), Planck_biban.m

```
Planck_biban
L 16, Use log. 1: yes, 0: no: 1
L 23, Interv. long. ondes    : 150, 60000 nm
L 25, Stefan-Boltzmann       : 5.6704e-08 W/(m2K4)
L 49, Wien, lamax visible    : 501.3841   nm
L 50, Wien, lamax infra rou : 10062.5    nm
L 51, Irradiance visible     : 6.33e+07   W/m2
L 52, Irradiance infra rou   : 390        W/m2
L 79, Rapp. max vi / max ir : 3300000
L 80, Coeff. courbe bleue    : 500000
L 81, log10(4*1e-6)          : -5.4
L 82, Date, CPU, 18-Apr-2024, 1.0815 s
```

Table 6 : Résultats de la procédure Planck_biban.m (Table 20 et Figure 7)

Températures du soleil et de la terre	
Exitance L 10, Temperature : 5780 K L 11, Intervalle : 4000 100000 nm L 12, Nombre de pas : 200 L 13, Stef.-Boltz. sT4 : 63.3 MW/m2 L 14, Max. wavelength : 100000 nm L 21, Int.(L) / st4 : 0.0097189 L 22, Date, CPU, 18-Apr-2024, 0.005717 s	Exitance L 10, Temperature : 288 K L 11, Intervalle : 4000 100000 nm L 12, Nombre de pas : 200 L 13, Stef.-Boltz. sT4 : 0.00039 MW/m2 L 14, Max. wavelength : 100000 nm L 21, Int.(L) / st4 : 0.99327 L 22, Date, CPU, 18-Apr-2024, 0.013973 s

Table 7 : Résultats de la procédure Matlab© Exitance.m (Table 21) : 5780 et 288 K

La norme ISO 20473:2007 définit l'infrarouge lointain dans la fourchette comprise entre 50 *µm* et 1 *millimètre*. Le rayonnement terrestre n'y intervient que pour un peu plus de 3 %.

Avec la variable *loga* = 1 (*ligne 15*), la procédure *Planck_biban.m* donne la *Figure 8*.

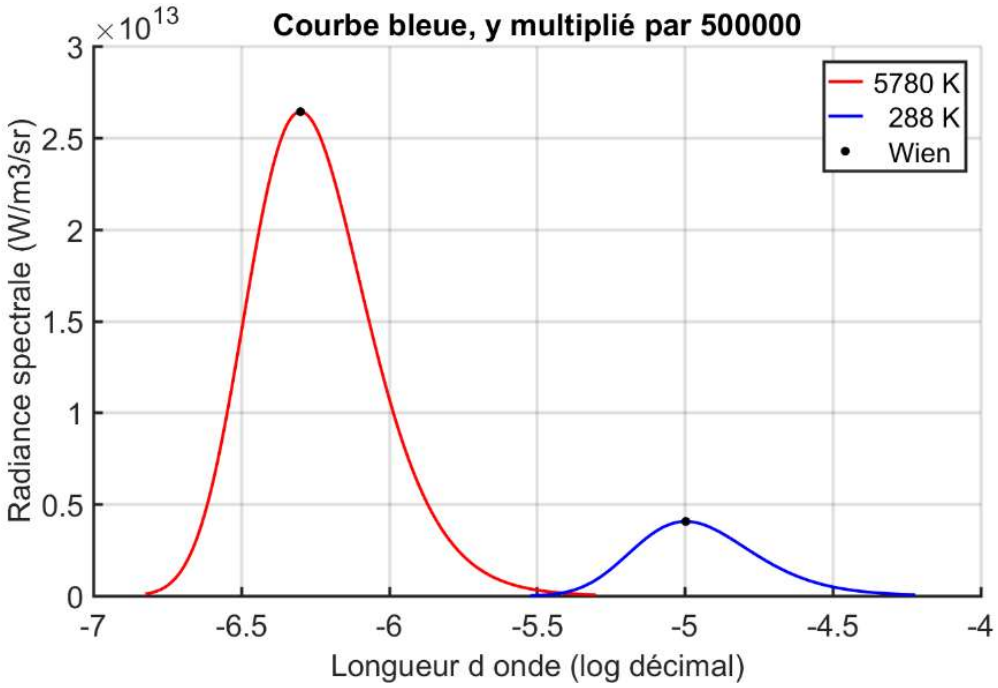

Figure 8 : Rad. spectrale du soleil (rouge) et de la terre (bleu), Planck_biban.m

1.4. Loi de Kirchhoff

Pour un carreau dont la surface est grise et diffuse, la ***loi de Kirchhoff*** établit la relation entre émission et réflexion :

$$\varepsilon_i = \alpha_i \qquad \rho_i = 1 - \varepsilon_i \tag{19}$$

Les coefficients introduits en (19) représentent les grandeurs suivantes : α_i est le coefficient d'absorption, ε_i, l'émissivité, et ρ_i, la réflectivité. Toutes ces grandeurs sont sans dimension. Dans le cas du corps noir, α_i - ε_i = 1 et ρ_i = 0.

2. Interactions par rayonnement

2.1 Angle solide

L'aire de la sphère de rayon r est égale à $2\pi r^2$. L'angle solide d'un objet de l'espace par rapport à un point est donné par l'aire de sa projection sur une sphère de rayon unitaire entourant le point.

L'élément d'angle solide $d\omega$ d'un petit élément d'aire dA est fonction de l'orientation du carreau et du carré de la distance entre le point de vue et le carreau :

$$d\omega = \frac{dA \cos\theta_{carreau}}{r^2} \tag{20}$$

La variable $\theta_{carreau}$ représente l'angle entre la normale au carreau et la droite qui joint son centre de gravité au point d'observation. La variable r est la distance entre ces deux points.

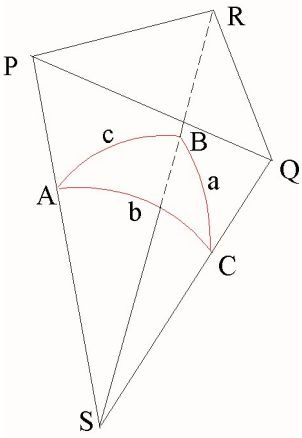

Figure 9 : Projection ABC du triangle PQR sur la sphère unitaire centrée en S

On obtient une solution approchée de l'angle solide d'un carreau par une formule proche de (21) :

$$\text{Angle solide} = \frac{Aire_{carreau} \cos\theta_{carreau}}{r^2} \tag{21}$$

La qualité de cette approximation est de moins en moins bonne quand la distance du point d'observation au carreau diminue.

La notion d'angle solide est illustrée à la *Figure 9*, par le calcul de l'angle solide d'un triangle. Le triangle *PQR* constitue la moitié de la face supérieure d'un cube unitaire centré à l'origine des axes. La procédure *Angl_sol_triangle.m* (*Table 18*) donne les résultats de la *Table 8*.

```
Angl_sol_triangle
Position du point de vue S     : 0  0  0 m
Définition du triangle point P : -0.5        -0.5         0.5 m
Définition du triangle point Q :  0.5        -0.5         0.5 m
Définition du triangle point R : -0.5         0.5         0.5 m
Angles triangle sphér.   A-B-C : 60           60          120 deg
Angle solide tr. sph. A+B+C-pi : 1.0472  str
Angle solide PQR / hémisphère  : 16.6667 %
Angle solide PQR + sym. / hém. : 33.3333 %
An. sol. faces vert. demi cube : 16.6667 %
L 28, Date, CPU, 18-Apr-2024, 0.22311 s
```

Table 8 : Résultats de la procédure Angl_sol_triangle.m (Table 18)

2.2 Constante solaire

Pour le calcul de la constante solaire, on définit l'unité astronomique (*UA*), égale à 149 597 870 700 mètres exactement, selon le système de constantes astronomiques en vigueur depuis 2009. La distance terre soleil est de 1 *UA,* soit environ 150 millions de *km*. Le rayon de la Terre, considérée comme une sphère légèrement aplatie aux pôles, est approximativement de 6370 *km*. Le rayon de la sphère solaire est de 696 342 *km*.

La Terre tourne autour de son axe en un peu moins de 24 *h* (1 jour) et autour du Soleil, dans le plan de l'écliptique, avec une période de 365.25 jours. L'axe de la première rotation (qui est défini par les deux pôles) est incliné de 23.5° par rapport à la normale à l'écliptique. Cette obliquité détermine les latitudes particulières des tropiques (23.5° par rapport à l'équateur) et des cercles polaires (23.5° par rapport aux pôles et donc 66.5° par rapport à l'équateur).

La surface du Soleil se comporte approximativement comme celle d'un corps noir à la température de 5780 *K*. Selon la loi de Stefan-Boltzmann, son émittance vaut $\sigma T^4 = 63.29$ MWm^{-2}. Le produit de cette émittance par l'aire de la sphère solaire est égal au produit de la constante solaire par la sphère de 150 millions de *km*. La constante solaire résultante (1371 Wm^{-2}) est équivalente à l'irradiance moyenne capturée au voisinage de la terre. Si cette irradiance est répartie, non sur un disque de même rayon que la terre perpendiculaire au rayon solaire, mais sur la sphère terrestre, on obtient une valeur quatre fois plus petite, soit 342 Wm^{-2} [Beckers 2012].

```
Constante_solaire
 SB : 5.67e-08      W/(m2K4
 rs : 6.963e+08     m
 ds : 149597900000  m
 ss : 6.093e+18     m2
 t  : 5780          K
 em : 6.329e+07     W/m2
 cs : 1371          W/m2
 as : 342.8         W/m2
```

Table 9 : Résultats de la procédure Constante_solaire .m (Table 23)

2.3 Facteur de vue

Suivant la terminologie utilisée dans le domaine de la CAO, on appelle *carreau* (en anglais, *patch*) une surface limitée le plus souvent par quatre côtés et définie sous forme paramétrique. Cette surface est de forme quelconque, elle n'est pas nécessairement plane, pas plus que les courbes qui la bordent

Le facteur de vue (également appelé facteur de forme) est à la base des études de transfert de chaleur par rayonnement [Sillion *et al*. 1994, Beckers *et al*. 2012, Beckers 2023]. C'est une variable purement géométrique, mais sa définition s'appuie sur des notions énergétiques : F_{ij} est la proportion de la *puissance totale* quittant l'élément A_i qui est reçue par l'élément A_j.

Dans la formule (22), les angles θ_i et θ_j correspondent aux angles entre le vecteur reliant les éléments différentiels et les vecteurs normaux à ces éléments ; r est la distance entre les deux éléments différentiels.

$$F_{ij} = \frac{1}{A_i} \int_{A_i} \int_{A_j} \frac{\cos\theta_i \cos\theta_j}{\pi r^2} V(y_i, y_j)\, dA_i dA_j \qquad (22)$$

Le symbole V représente la fonction de visibilité, elle est égale à 0 si le rayon qui joint les deux carreaux est interrompu par un obstacle, sinon elle est égale à 1. L'expression (22) est symétrique par rapport aux deux carreaux (ils y jouent le même rôle). On en déduit la ***relation de réciprocité***.

$$A_i \, F_{ij} = A_j \, F_{ji} \tag{23}$$

En exprimant le différentiel de l'aire dAj en fonction du différentiel d'angle solide $d\omega$ de la formule (20), on obtient :

$$F_{ij} = \frac{1}{A_i} \int_{A_i} \int_{\Omega} \frac{\cos\theta_i}{\pi} \, V(y_i, y_j) \, dA_i \, d\omega \tag{24}$$

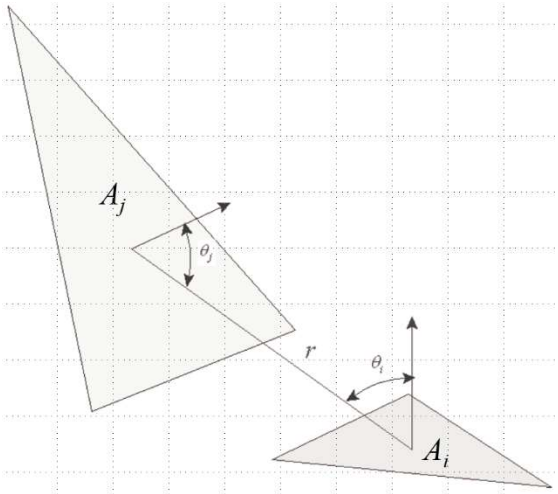

Figure 10 : Définitions géométriques des facteurs de vue

Nous venons d'introduire la définition classique de facteur de vue. Ce terme est constitué d'une intégrale double qui ne peut se calculer analytiquement que dans un petit nombre de situations particulières [Howell *et al.* 2010]. Une difficulté supplémentaire apparaît en présence d'obstructions qui sont représentées dans (22) par la fonction de visibilité $V(y_i, y_j)$.

Il est plus facile de calculer le ***facteur de vue point - surface*** en supprimant l'intégration sur A_i, laquelle, pour réaliser l'évaluation du facteur de vue entre surfaces, sera prise en compte dans une deuxième étape en utilisant, par exemple, la méthode de quadrature gaussienne. Le ***facteur de vue différentiel*** en un point situé sur l'élément d'aire dS est donné par :

$$F_{dS-A_j} = \int_{A_j} \frac{\cos\theta_i \, \cos\theta_j}{\pi r^2} V(y_i, y_j) \, dA_j \tag{25}$$

Il en résulte que le ***facteur de vue point - surface*** d'un élément A_j vu du point X_i appartenant à l'élément A_i est donné par :

$$F_{X_i,A_j} = \int_{A_j} \frac{\cos\theta_i \, \cos\theta_j}{\pi r^2} V(Y_i, Y_j) \, dA_j = \frac{1}{\pi} \int_{\Omega_{X_i-A_j}} \cos\theta_i \, V(Y_i, Y_j) \, d\omega \tag{26}$$

Le ***facteur de vue point - surface*** lie un point X du carreau A_i au carreau A_j. L'angle entre le rayon r et la normale au carreau A_i est noté θ_i, l'angle entre le rayon r et la normale au carreau

A_j est noté θ_j. Dans le second terme de l'équation (26), l'intégrale de surface est transformée en intégrale sur l'angle solide. On peut l'interpréter comme la projection orthogonale sur le plan tangent à l'élément A_i au point X_i de la projection de l'élément A_j sur l'hémisphère unitaire, divisée par la surface du cercle sous-tendu. Cette interprétation du facteur de vue est connue sous le nom **d'*analogie de Nusselt*** [Beckers *et al*. 2009].

Une des premières méthodes utilisées pour calculer le ***facteur de vue point - surface*** en présence de masques a été celle de l'hémicube [Cohen *et al*. 1985]. Pour passer du facteur de vue point - surface au facteur de vue entre surfaces, il suffit de réaliser une quadrature de Gauss sur le carreau A_i où on effectue l'évaluation point – surface.

Afin de résoudre efficacement le problème d'interaction par rayonnement, on établit une formulation discrète dérivée de l'équation d'éclairement global en faisant l'hypothèse que l'environnement est une collection d'un nombre fini N de petits carreaux réfléchissant de manière diffuse, avec chacun une radiosité uniforme [Sillion *et al*. 1994].

$$F = \begin{pmatrix} F_{11} & F_{12} & \cdots & F_{1N} \\ F_{21} & F_{22} & & \vdots \\ \vdots & & & \vdots \\ F_{N1} & \cdots & \cdots & F_{NN} \end{pmatrix} \tag{27}$$

La matrice F, (27), contient les N facteurs de vue entre les carreaux i et j. Lorsque les carreaux sont des polygones plans, les termes F_{ii} sont nuls. Si l'environnement complet, soit la scène et le ciel, est pris en compte, les coefficients de la matrice F vérifient la ***propriété de fermeture*** :

$$\sum_{j=1}^{N} F_{ij} = 1 \qquad i = 1 : N \tag{28}$$

Depuis le centre d'une des faces d'un cube, les facteurs de vue des six faces, calculés selon (26) et mémorisés sous la forme (27), sont donnés dans la *Table 10*. Pour obtenir ce résultat, la procédure *Rayocube.m* (*Table 25*) fait appel à la fonction *fvuelamb.m* (*Table 38*).

Facteurs de vue point - surface depuis le centre de la base du cube					
0	0.2395	0.1901	0.1901	0.1901	0.1901
0.2395	0	0.1901	0.1901	0.1901	0.1901
0.1901	0.1901	0	0.2395	0.1901	0.1901
0.1901	0.1901	0.2395	0	0.1901	0.1901
0.1901	0.1901	0.1901	0.1901	0	0.2395
0.1901	0.1901	0.1901	0.1901	0.2395	0

Table 10 : Facteurs de vue dans un cube (fonction fvuelamb.m)

Dans les deux exemples suivants les facteurs de vue sont déduits directement des propriétés géométriques des espaces considérés.

Hémisphère sur plan. Soit P_d un disque d'aire A_d et P_h l'hémisphère qui le surplombe, d'aire A_h égale au double de celle de son disque de base : $A_h = 2A_d$. Le facteur de vue de P_h est F_{dh} = 1. Par la ***relation de réciprocité*** (23) : $A_d F_{dh} = A_h F_{hd}$, on a : $F_{hd} = A_d F_{dh} / A_h = A_d F_{dh} / 2 A_d = 0.5$.

Sphère dans sphère. Soit deux sphères concentriques, le facteur de vue de la sphère intérieure vers l'extérieure est $F_{ie} = 1$, en effet, elle ne se voit pas elle-même et ses éléments voient la sphère extérieure. Le facteur de vue de la sphère extérieure (d'aire A_e) vers l'intérieure (d'aire A_i) est, selon la ***relation de réciprocité*** (23) : $A_i F_{ie} = A_e F_{ei}$, donné par $F_{ei} = A_i / A_e$. Le facteur de vue de la sphère extérieure vers cette même sphère extérieure est $F_{ee} = 1 - A_i/A_e$. Ceci correspond à la ***relation de fermeture*** : dans une enceinte fermée, la somme des facteurs de vue est égale à 1.

2.4 Equation de radiosité pour des carreaux réfléchissant : $\rho_i > 0$.

Si les équations de radiosité peuvent être écrites sous la forme d'un système linéaire reposant sur une matrice symétrique définie positive, elles peuvent être résolues par la méthode de Cholesky [van de Geijn 2011]. Cette méthode, numériquement stable et très rapide, peut être mise en concurrence avec les méthodes itératives classiques.

Lorsque tous les coefficients de réflexion sont supérieurs à zéro, les équations de radiosité [Beckers 2012] s'écrivent sous la forme suivante.

$$B_i = E_i + \rho_i \sum_{k=1}^{n} B_k F_{ik} \qquad (29)$$

Les termes B_i et E_i dénotent les composantes des vecteurs de radiosité B ($W\ m^{-2}$) et de puissance émise par unité de surface E ($W\ m^{-2}$). La radiosité B_i est la puissance totale par unité de surface quittant le carreau i. Le coefficient F_{ik} de la matrice F représente le facteur de vue du carreau i vers le carreau k.

Les caractéristiques physiques et géométriques des carreaux sont introduites dans les deux matrices diagonales du tableau suivant.

Matrices diagonales utilisées dans les formulations		
Réflectances	R	$R_{ij} = \rho_i\ \delta_{ij}$
Aires des carreaux	A	$A_{ij} = A_i\ \delta_{ij}$

Table 11 : Matrices diagonales utilisées dans la solution des équations de radiosité

Soit I la matrice unité : $I_{ij} = \delta_{ij}$, sous forme matricielle, l'équation de radiosité (29) s'écrit :

$$\begin{aligned} (I - RF)\,B &= E \\ MB &= E \end{aligned} \qquad (30)$$

La matrice $M = I - RF$ est la **matrice de radiosité**.

Le test suivant consiste à réaliser des représentations du cube de la *Figure 11, gauche*. Cette projection est réalisée avec la procédure *cartsphe.m* (*Table 24*), dont l'affichage du déroulement est présenté à la *Table 12*. La numérotation des nœuds est telle que l'arête 1–2 est parallèle à l'axe des x, l'arête 1-4 à l'axe des y et la 1–5, à celui des z. L'origine des axes est située au centre du cube, qui est aussi le centre de la sphère sur laquelle on le projette (*Figure 11, droite*). L'avantage de la projection sur la sphère est de pouvoir utiliser n'importe quel planisphère pour visualiser l'entièreté du cube sur un plan en respectant certaines propriétés, comme les aires égales dans la

projection de Mollweide (*Figure 12* et *Figure 13, droite*) réalisée grâce à la fonction *Mollcube.m* (*Table 26*). Dans la *Figure 12*, le plan du méridien de référence, de latitude 0°, passe par les arêtes 1 – 5 et 3 – 7. On voit cette position à gauche de la *Figure 12*. La projection sphérique du cube est le cercle rouge, les deux faces avant sont entièrement visibles, les deux arrières sont totalement invisibles et les deux faces horizontales sont à moitié visibles. Dans cette configuration, les quatre faces verticales sont représentées en entier ? tandis que les deux faces horizontales apparaissent sur le dessus et le dessous de l'ellipse de la projection.

```
cartsphe
T  04, Num. hexa.: 1
T  05, Cube dim. : 1 m
T  07, Cube vol. : 1 m3
gm3 2, Vert. num.: 1  2  3  4  5  6  7  8
gms 2, Vert. num.: 1  2  3  4  5  6  7  8
gms 9, size xyz  : 8  3
gms10, size lK   : 12   2
gms11, size lel  : 1  8
T   9, Date, CPU : 07-Feb-2024, 0.5 s
```

Table 12 : Résultats de la procédure cartsphe.m (Table 24)

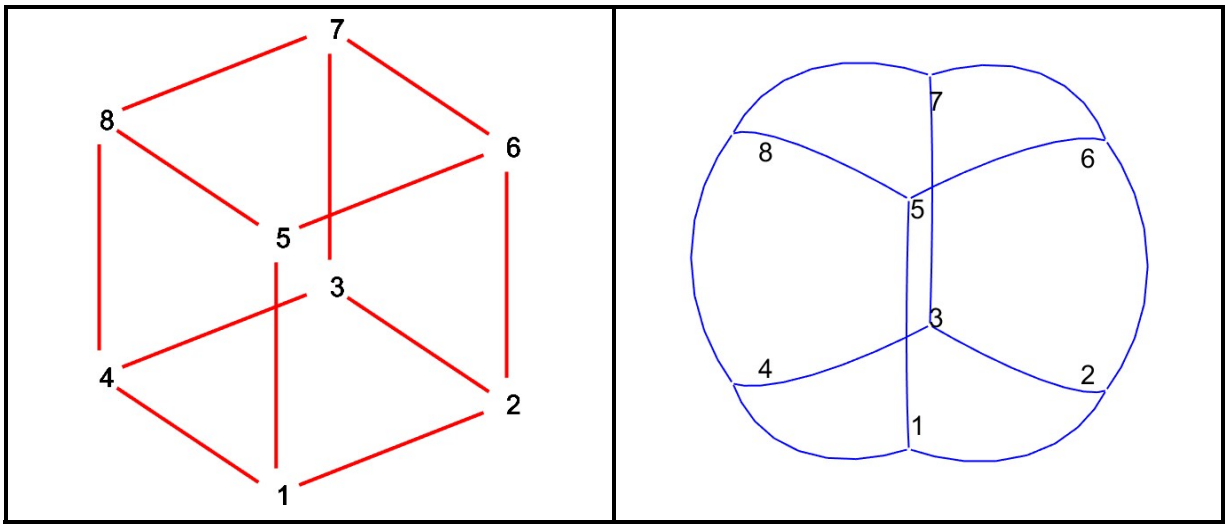

Figure 11 : Axonométries du cube et de sa projection sphérique

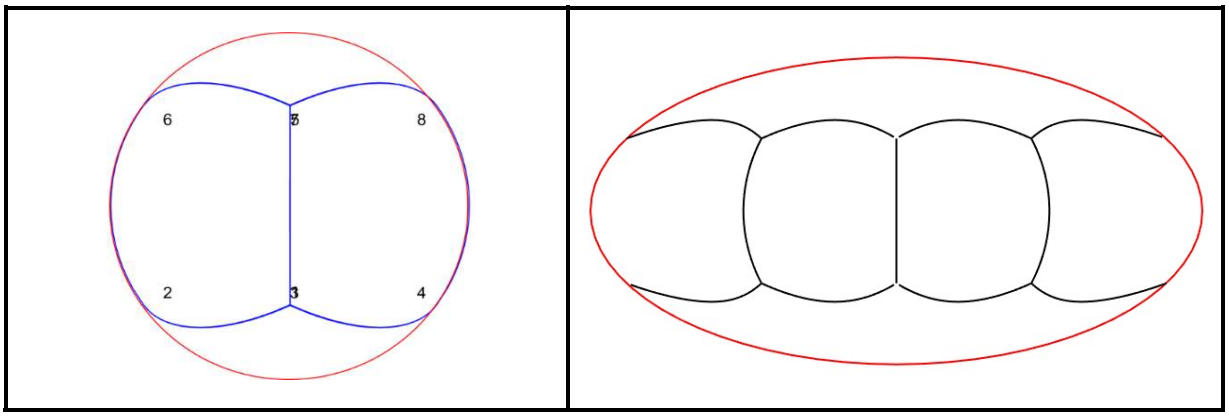

Figure 12 : Projection orthogonale et projection de Mollweide du cube sphérique

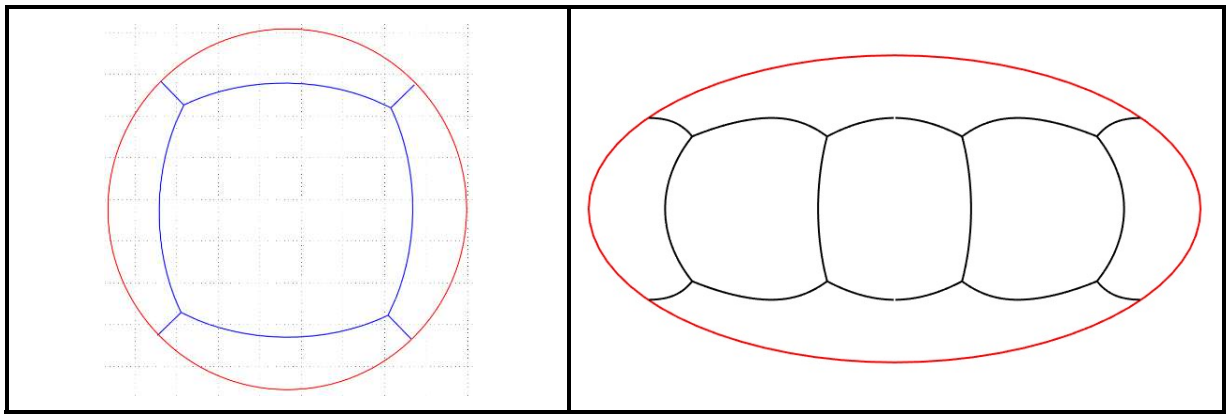

Figure 13 : Projections orthogonale et de Mollweide du cube sphérique

L'activation des *lignes 34* à *75* de la procédure de la *Table 26* permet d'ajouter les médianes des six faces qui avaient été obtenues *Figure 12* et *Figure 13* sur les projections du cube (*Figure 14*)

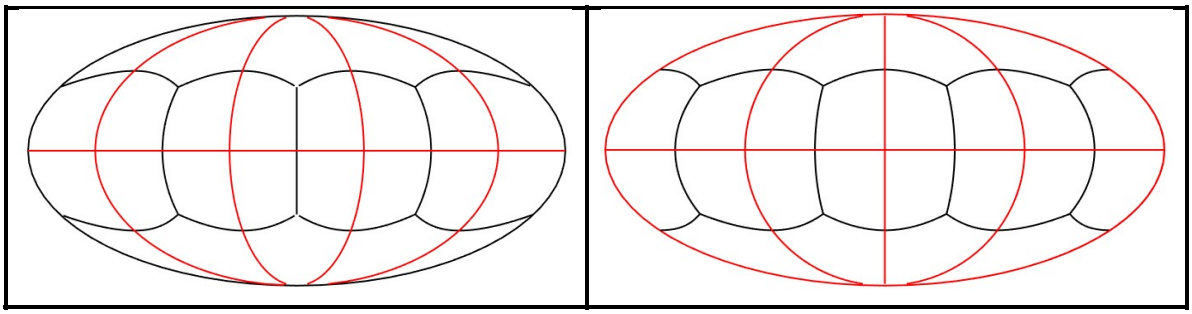

Figure 14 : Projection de Mollweide du cube (noir) avec médianes des faces (rouge)

Pour passer de la *Figure 14* de gauche à celle de droite : dans *Mollcube.m*, il faut activer la *ligne 18* et désactiver la *19* ; dans *lalo.m*, il faut activer la *47* et désactiver la *48*.

Pour un cube dont chaque face est constituée d'un seul carreau et dans lequel l'exitance est égale à 1 *W* sur une des faces, on obtient la solution de la *Table 13*. La ***propriété de fermeture*** est parfaitement vérifiée : la somme des termes divisée par le coefficient de réflexion ρ est égale à 1 pour toutes les lignes de la matrice.

M						***B***(Wm^{-2})	***E***(Wm^{-2})
1.0000	-0.1197	-0.0951	-0.0951	-0.0951	-0.0951	1.0913	1
-0.1197	1.0000	-0.0951	-0.0951	-0.0951	-0.0951	0.1982	0
-0.0951	-0.0951	1.0000	-0.1197	-0.0951	-0.0951	0.1776	0
-0.0951	-0.0951	-0.1197	1.0000	-0.0951	-0.0951	0.1776	0
-0.0951	-0.0951	-0.0951	-0.0951	1.0000	-0.1197	0.1776	0
-0.0951	-0.0951	-0.0951	-0.0951	-0.1197	1.0000	0.1776	0

Table 13 : Matrice de radiosité et solution pour les 6 faces du cube (Rayocube.m)

Comme toutes les aires sont égales, la matrice de radiosité est symétrique. Grâce à la symétrie de la géométrie et du chargement, les radiosités des 4 faces verticales sont égales. Pour la face inférieure, on a une contribution unitaire due à l'exitance et une contribution réfléchie égale environ à la moitié de celles des faces verticales. On vérifie également que le produit scalaire du

vecteur de radiosités par le vecteur des aires des carreaux, soit dans la notation de la procédure Matlab© utilisée : 'B'*aires', est égal à 2 Watts.

```
Rayocube
L   2, coeff. refl. : 0.5
L   6, Lambert, oui : 1
L   9, Maillage     : 4 x 4 x 4
L  13, Dim. struc   : 384    3
L  20, N.carr. = nel: 96
L  74, B min & + del: 0.1121      0.2921
L  81, a, b ellipse : 5.6569      2.8284
L  95, Dessin élém. : 1 - 96
L 139, N. cotés des.: 347
L 148, Ferm. moyenne: 1
L 149, B*aires      : 2 W
L 150, CPU total    : 13.92 s
(Pierre\Benoit\Radiosité_2017) (Table 25)
```

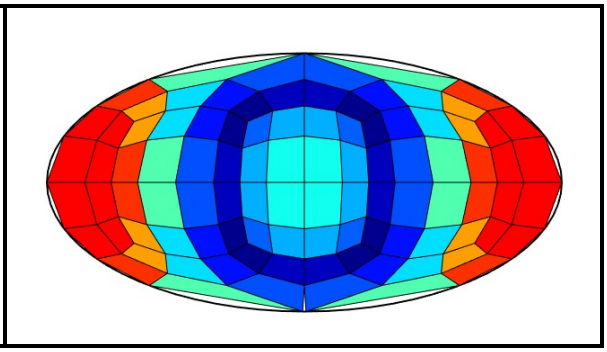

Table 14 : Représentation des radiosités sur les 6 faces du cube subdivisées en 16 carrés

Pour un maillage de 16 éléments par face du cube, on obtient les résultats de la *Table 14*, qui montrent les radiosités sur les 96 éléments couvrant le cube. En quadruplant le nombre d'éléments, on obtient la *Figure 15*. Ici, la procédure s'exécute en 92 secondes.

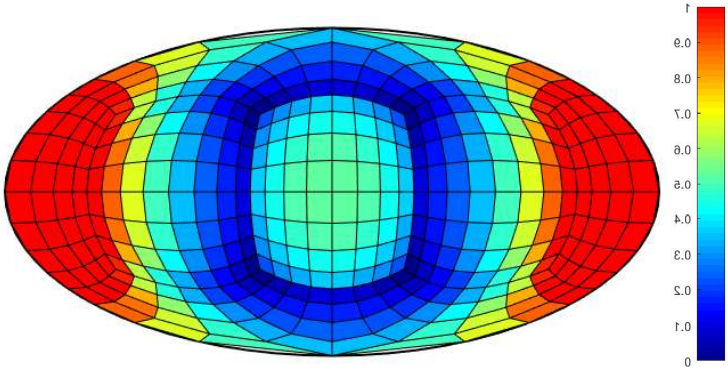

Figure 15 : Radiosités sur les 384 éléments carrés couvrant le cube

Calcul des facteurs de vue par lancer de rayons

Les représentations de la *Figure 16* sont générées par les instructions : *figure ; Bsphe (Bsams (300),0,1)* pour la sphère de gauche (*Table 29*, *Table 30*) et *BMollw ([1 9 22 41 64 91 120 150])* pour l'ellipse de droite (*Table 32*). Elles font apparaitre les 300 cellules d'aires égales, sur la sphère définie en *3D* et sur sa projection de Mollweide. En pratique, 150 cellules sont définies sur un hémisphère dont on crée ensuite le symétrique, cette symétrie est visible sur les équateurs des deux figures.

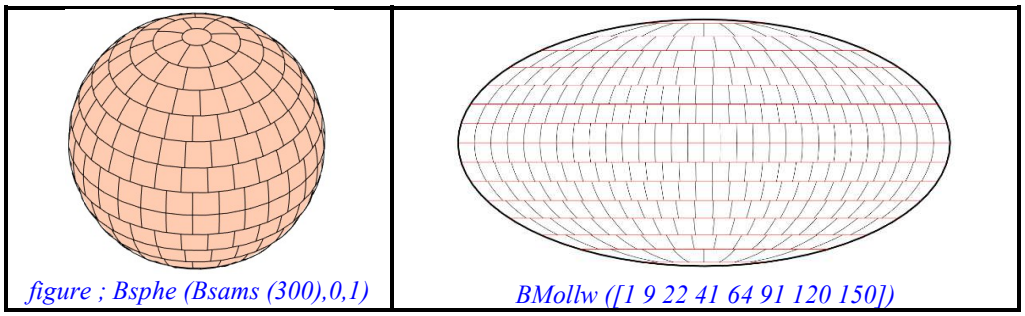

Figure 16 : Une sphère et sa projection de Mollweide comportant 300 cellules d'aires égales

Chacune des faces du cube contient l'équivalent de 50 cellules, voir *Figure 17*, où apparait également la projection « fil de fer » du cube (procédure *Mollcube.m*, *Figure 12*).

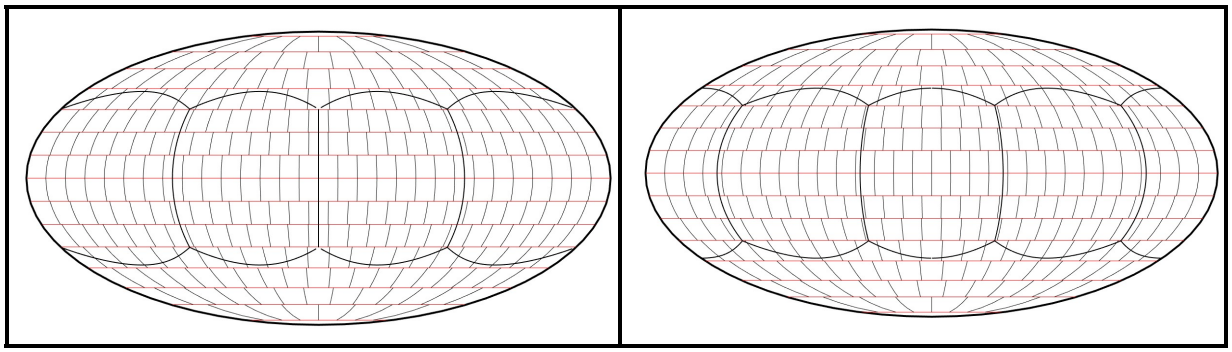

Figure 17 : Projections de Mollweide des arêtes du cube superposées à 300 cellules d'aires égales

Les procédures décrites dans le rapport [Beckers 2016] permettent de générer les coordonnées sphériques des rayons à lancer pour une évaluation des facteurs de vue. Le rapport entre le nombre de rayons atteignant un carreau et le nombre de rayons lancés définit le facteur de vue du carreau.

Si, à partir de chaque cellule, on tire le même nombre de rayons de manière aléatoire et qu'on compte le nombre de rayons arrivant sur chaque carreau du maillage, le rapport entre ce nombre et le nombre total de rayons définit le facteur de vue du carreau.

Dans cette méthode, la condition de fermeture est automatiquement vérifiée. Un premier calcul de la matrice des facteurs de vue est présenté à la *Table 15*, où, avec 3000 rayons, les résultats sont très probants.

Facteurs de vue depuis le centre de la base du cube, 3000 rayons					
0	0.2393	0.1893	0.1893	0.1900	0.1920
0.2393	0	0.1893	0.1893	0.1900	0.1920
0.1893	0.1893	0	0.2393	0.1900	0.1920
0.1893	0.1893	0.2393	0	0.1920	0.1900
0.1900	0.1920	0.1893	0.1893	0	0.2393
0.1920	0.1900	0.1893	0.1893	0.2393	0

Table 15 : Facteurs de vue dans un cube (Rayocube.m, *lancer de rayons)*

Dans la *Table 16*, on calcule les radiosités avec 300 rayons et on compare aux résultats de la *Table 13*.

M						*B* (Wm^{-2})	*E* (Wm^{-2})
1.0000	-0.1217	-0.0950	-0.0933	-0.0950	-0.0950	1.0914	1
-0.1217	1.0000	-0.0950	-0.0933	-0.0950	-0.0950	0.1998	0
-0.0950	-0.0933	1.0000	-0.1217	-0.0950	-0.0950	0.1775	0
-0.0933	-0.0950	-0.1217	1.0000	-0.0950	-0.0950	0.1762	0
-0.0950	-0.0950	-0.0950	-0.0933	1.0000	-0.1217	0.1776	0
-0.0950	-0.0950	-0.0933	-0.0950	-0.1217	1.0000	0.1776	0

Table 16 : Facteurs de vue calculés par lancer de 300 rayons. Fermeture vérifiée exactement.

Les résultats sont très bons et restent satisfaisants avec un tir de 100 rayons (*Table 17*). La somme des produits des radiosités par les aires y est égale à 2.001 *W*.

M						*B(Wm^{-2})*	*E(Wm^{-2})*
1.0000	-0.1350	-0.0900	-0.0900	-0.1000	-0.0850	1.0922	1
-0.1350	1.0000	-0.0900	-0.0900	-0.1000	-0.0850	0.2111	0
-0.0900	-0.0900	1.0000	-0.1350	-0.1000	-0.0850	0.1732	0
-0.0900	-0.0900	-0.1350	1.0000	-0.0850	-0.1000	0.1730	0
-0.1000	-0.0850	-0.0900	-0.0900	1.0000	-0.1350	0.1812	0
-0.0850	-0.1000	-0.0900	-0.0900	-0.1350	1.0000	0.1696	0

Table 17 : Facteurs de vue calculés par lancer de 100 rayons

2.5 Traitement des carreaux de type corps noir

Pour un corps noir, $[R] = 0$, les émissivités ε_i sont égales à 1 (loi de Kirchhoff, (19)). Le vecteur E est proportionnel à la constante de Stefan-Boltzmann et à la quatrième puissance de la température de surface.

$$E \rightarrow E_i = \sigma T_i^4 \tag{31}$$

Les flux de chaleur J dépendent des températures de surface des éléments bordant le maillage éléments finis. On ne peut cependant définir une température de surface que s'il existe un solide qui la porte. Idéalement, ce solide est modélisé par un maillage éléments finis [Beckers 2013, Beckers & *al*. 2014].

Le vecteur de mise en charge par rayonnement Q (Wm^{-2}) est obtenu en soustrayant le flux entrant J (Wm^{-2}) à la radiosité sortante B (Wm^{-2}) :

$$Q = B - J \tag{32}$$

Sachant que :

$$J = [F]\, B \tag{33}$$

Le bilan des exitances sortant du carreau i est la matrice uni-ligne Q :

:

$$\begin{aligned} Q &= B - [F]B \\ &= \{[I] - [F]\}B \end{aligned} \tag{34}$$

Comme $E = B$, on a finalement :

$$J = \{[I] - [F]\}\, E \tag{35}$$

Lorsqu'on aborde le rayonnement des ondes longues, on ne peut plus se limiter aux réflexions diffuses. En effet, dans ce domaine, les réflexions spéculaires sont loin d'être négligeables [Rushmeier & *al.* 1990], [He & *al.* 1991]. Le traitement qui s'impose est celui de l'utilisation des facteurs de vue étendus [Sillion & *al*, 1994], [Bugeat & *al*, 2020].

Références

Beckers Benoit, Masset Luc & Beckers Pierre, *Commentaires sur l'analogie de Nusselt*, Rapport Helio_004_fr, http://www.heliodon.net/heliodon/documents.html, **2009**.

Beckers Benoit, Beckers Pierre, Radiative Simulation Methods, in *Solar Energy at Urban Scale, chap. 10, Ed. B. Beckers, John Wiley and Sons*, Inc., pp 205-236, **2012**

Beckers Benoit, Worldwide Aspects of Solar Radiation Impact, in *Solar Energy at Urban Scale, chap. 5, Ed. B. Beckers, John Wiley and Sons*, Inc., pp 99 - 448, **2012**

Beckers Benoit, *Taking Advantage of Low Radiative Coupling in 3D Urban Models*, Eurographics Workshop on Urban Data Modelling and Visualization, May 6-10, **2013**, Girona, Spain. http://www.heliodon.net/downloads/Beckers_Benoit_2013_EG_Gerona_Taking_Advantage

Beckers Benoit, Beckers Pierre, *Reconciliation of Geometry and Perception in Radiation Physics*, Focus Series in Numerical Methods in Engineering, Wiley-ISTE, 192 pages, July **2014**.

Beckers Benoit, Beckers Pierre, *Super element technique for solar energy optimization at urban level*, OPT-*I,* An International Conference on Engineering and Applied Sciences Optimization, M. Papadrakakis, M.G. Karlaftis, N.D. Lagaros (eds.) Kos Island, Greece, 4-6 June **2014**

Beckers Benoit, *Multiscale Analysis as a Central Component of Urban Physics Modeling*, In: Computational Methods for Solids and Fluids, Multiscale Analysis, Probability Aspects and Model Reduction, Adnan Ibrahimbegovic (Ed.), Springer International Publishing, **2016**, Pp.1- 27. http://www.springer.com/fr/book/9783319279947

Beckers Benoit, Beckers Pierre, *Fast and accurate view factor generation*, FICUP An International Conference on Urban Physics B. Beckers, T. Pico, S. Jimenez (Eds.) Quito - Galápagos, Ecuador, 26 - 30 September **2016**, www.heliodon.net

Beckers Benoit, "*Fiammetta, Finite Element Method Applied to Heat Transfer*", **2023** www.heliodon.net,

Boya Luis J., "The Thermal Radiation Formula of Planck", Rev. Real Academia de Ciencias. Zaragoza. 58: 91-114, (**2003**).

Buckingham Edgar, "*On the deduction of Wien's Displacement law*", Journal of the Washington Academy of Sciences, Vol. 2, No. 7 (April 4, **1912**), pp. 180-182 (3 pages), https://www.jstor.org/stable/24520831

Buckingham Edgar, "O*n the deduction of Wien's Displacement law*", Bulletin of the Bureau of Standards, **1912**

Bugeat Antoine, Beckers Benoit, Fernandez Eduardo, "*Improving the daylighting performance of residential light wells by reflecting and redirecting approaches*", Solar Energy 207 (**2020**) 1434-1444

Bunker P.R., Mills Ian M., Jensen Per, "*The Planck constant and its units*", Journal of Quantitative Spectroscopy & Radiative Transfer 237 (**2019**) 106594

Campbell G.S., Norman J.M., *An Introduction to Environmental Biophysics*, 2nd ed., New York, Springer, **1998**

M. Cohen & D. Greenberg, *The hemicube: A radiosity solution for complex environments*, SIGGRAPH'85, Volume 19, Number 3 (**1985**) 31-40

Diaz Medina Jorge Omar **2021** Cienciorama La radiación de cuerpo negro y el nacimiento de la mecánica cuántica

Gebhart B., *Heat Transfer*, McGraw-Hill, **1961**

He Xiao D., Torrance Kenneth E., Sillion François X., Greenberg Donald P., "*A Comprehensive Physical Model for Light Reflection*", Computer Graphics, Volume 25, Number 4, July **1991**

Heald M.A. "*Where is the "Wien peak*" ?", Am. J. Phys. 71 (12), December **2003**

Lambert J.H., "Photometria sive de mensura et gradibus luminis, colorum et umbrae", **1760**, German translation by Anding E., in Ostwald's *Klassiker der Exakten Wissenschaften*, vol. 31‑33, Leipzig, 1892. Cited by Schröder P. and Hanrahan P., A closed form expression for the form factor between two polygons, Research ReportCS-TR-404-93, January **1993**.

Wilhelm Nusselt, *"Détermination graphique du rapport d'angle pour le rayonnement thermique*", in B. Beckers, L. Masset & P. Beckers, *Commentaires sur l'analogie de Nusselt*, Rapport Helio_004_fr, www.heliodon.net, **2009**.

Overduin J. M., *Eyesight and the solar Wien peak*, Am. J. Phys. 71 (3), March **2003**

Planck Max, *"On an Improvement of Wien's Equation for the Spectrum",* Verhandl. Dtsch. Phys. Ges., 2, 202, **1900**. English translation from *"The Old Quantum Theory",* édité par D. ter Haar, Pergamon Press, **1967**, p. 79.

Planck Max, *"On the Law of Distribution of Energy in the Normal Spectrum", Annalen der Physik vol. 4, p. 553 ff (1901)*, **1901**

Planck Max, "Eight Lectures on Theoretical Physics. By Max Planck, Professor of Theoretical Physics in the University of Berlin. *A course of lectures delivered at Columbia University in* **1909**, translated by A. P. Wills, Professor of Mathematical Physics in Columbia University" .

Rushmeier Holly E. and Torrance Kenneth E. "*Extending the Radiosity Method to Include Specularly Reflecting and Translucent Materials*", ACM Transactions on Graphics, Vol. 9, No. 1, January **1990**, Pages l-27.

Sillion François, Puech Claude, "*Radiosity and Global Illumination"*, Morgan Kaufmann Publishers Inc, **1994**

Soffer B.H. & Lynch D.K., "*Some paradoxes, errors, and resolutions concerning the spectral optimization of human vision*", Am. J. Phys. **67** (11), November **1999**

van de Geijn Robert, "*Notes on Cholesky Factorization"*, Report TX 78712 University of Texas at Austin ‑ **2011**

Annexes

Procédure *Angl_sol_triangle.m* : angle solide

```
CPU  = tic;
P    = [-.5 -.5 .5];Q = [.5 -.5 .5];R = [-.5 .5 .5]; S    = [0 0 0];
SA   = (P-S)/norm(P-S); % Distance des sommets au centre de projection
SB   = (Q-S)/norm(Q-S);
SC   = (R-S)/norm(R-S);
nSAB = cross(SA,SB)/norm(cross(SA,SB));
nSBC = cross(SB,SC)/norm(cross(SB,SC));
nSCA = cross(SC,SA)/norm(cross(SC,SA));
A    = acos(-dot(nSAB,nSBC));
B    = acos(-dot(nSBC,nSCA));
C    = acos(-dot(nSCA,nSAB));
As   = A+B+C-pi;As1 = As*100/pi;
disp(['Position du point de vue S     : ',num2str(S),' m']);
disp(['Définition du triangle point P : ',num2str(P),' m']);
disp(['Définition du triangle point Q : ',num2str(Q),' m']);
disp(['Définition du triangle point R : ',num2str(R),' m']);
disp(['Angles triangle sphér.   A-B-C : ',num2str([A B C]*180/pi),' deg'])
disp(['Angle solide tr. sph. A+B+C-pi : ',num2str(As),' str'])
disp(['Angle solide PQR / hémisphère  : ',num2str(As*100/(2*pi)),' %'])
disp(['Angle solide PQR + sym. / hém. : ',num2str(As*100/pi),' %'])
disp(['An. sol. faces vert. demi cube : ',num2str((100-As1)/4),' %'])
figure;plot3([P(1) Q(1) R(1) P(1)], [P(2) Q(2) R(2) P(2)],...
    [P(3) Q(3) R(3) P(3)],'b')
axis equal; grid on;hold on
plot3([0 SA(1)],[0 SA(2)],[0 SA(3)],'r');hold on
plot3([0 SB(1)],[0 SB(2)],[0 SB(3)],'r');hold on
plot3([0 SC(1)],[0 SC(2)],[0 SC(3)],'r');hold on
disp (['L 28, Date, CPU, ',num2str(date),', ',num2str(toc(CPU)),' s',])
```

Table 18 : Procédure Matlab© Angl_sol_triangle.m : calcul angle solide triangle

Procédure *Planck_f.m*, voir *Figure 1* et *Figure 2*.

```
CPU    = tic;
%Fi 2 & 3 Beckers 20230226 Comments about blackbody radiation Planck's law
h      = 6.62606957 * 10^(-34);  % Constante de Planck   J.s
c      = 299792458;              % Vitesse de la lumière m/s
k      = 1.3806488 * 10^(-23);   % Boltzmann constant    J/K
b      = 2.897771955*10^(-3);    % Wien's displacement constant
sSB    = 5.6704 *10^(-8);        % Stefan-Boltzmann      en W/(m2K4)
np     = 200;                    % Nombre de points
xmax   = 3/1000000;              % Longueur maxi en metres
incr   = xmax/np;                % wavelenght interval en nanometres
taca   = 15;epli   = 2;          % Taille et épaisseur des caractères
t      = [5780 5000 4000 3000];  % Températures
col    = ['r'  'm'  'k'  'b'];   % Colors used for the curves
lam    = zeros(np,1);L = zeros(np,1);                    % Initialization
lamax  = zeros(4,1);maxL = zeros(4,1);                   % Initialization
Q      = sSB*t.^4*1e-6;Ql = zeros(1,4);Qn = Ql;              % Irradiance
disp (['L 17, Const. de Planck : ' ,num2str(h)  ,' Js'])
disp (['L 18, Vitesse lumière  : ' ,num2str(c)  ,' ms-1'])
disp (['L 19, Const. Boltzmann : ' ,num2str(k)  ,' JK-1'])
disp (['L 20, Cst. dépl. Wien  : ' ,num2str(b)  ,' mK'])
disp (['L 21, Stefan-Boltzmann : ' ,num2str(sSB),' W/(m2K4)'])
disp (['L 22, Tempér. imposées : ' ,num2str(t)  ,' K'])
disp (['L 23, Stef.-Boltz. sT4 : ' ,num2str(Q,3),' MW/m2'])
disp (['L 24, Long. onde max.  : ' ,num2str(xmax*1e9),' nm'])
disp (['L 25, Nombre de pas    : ' ,num2str(np)])
```

```
disp (['L 26, Amplitude de pas : ' ,num2str(incr*1e9),' nm'])

figure   ('Position',[100 100 800 450]);                  % First drawing
axes     ('FontSize',taca,'LineWidth',epli)
for j  = 1:4                           % loop on the 4 imposed temperatures
    T  = t(j);
    for i          = 1:np                 % Integration of the Planck's law
        lam(i)     = i*incr;
        L(i)       = (2*h*c^2./lam(i)^5)*(1/(exp(h*c/(k*lam(i)*T))-1));
    end
    plot (lam*10^9,L,col(j),'LineWidth',epli);hold on; % plot the 4 curves
    maxL (j) = max(L);            % Maximum spectral radiances in W /m3/sr
    Ql(j)    = sum(L)*incr*pi;                      % Integrated irradiance
    lamax(j) = b / t(j);    % Wien's law : positions of the max. radiances
end
disp(['L 41, Integrale LT(la) : ', num2str(Ql*1e-6,3),' MW/m2'])
disp(['L 42, Max. rad. LT(la) : ',num2str(maxL',3),' W/(m2 sr m)'])
disp(['L 43, Position du max. : ',num2str(lamax'*1e9,3),' nm'])
plot   (lamax*1e9,maxL,'.k','MarkerSize',taca);hold on;% Draw the 4 points
legend ('5780 K','5000 K','4000 K','3000 K','Maxim.')
xlabel (['Longueur d onde, de ',num2str(min(lam)*10^9),' à ',...
    num2str(max(lam)*10^9),' nm, par pas de ',num2str(incr*10^9),...
    ' nm',],'FontSize', taca)
ylabel ('Radiance spectrale, W /m2 /sr /m' ,'fontsize',taca)
title  ('Radiance spectrale (longueur d onde)',...
    'FontSize', taca);grid on;

figure ('Position',[100 100 800 450]);taca=15;            % Second drawing
axes   ('FontSize',taca,'LineWidth',epli)
nu    = zeros(np,1);L = zeros(1,np);xmax = xmax *10;
numin = c/xmax;numax = c*np/xmax;                    % numin = numax/200;
incr  = (numax-numin)/(np-1);
for j     = 1:4                    % Loop on the four imposed temperatures
    T     = t(j);
    for i = 1:np                                     % Loop on the np steps
       nu(i) = numin + (i-1)*incr;
       L(i)  = (2*h*nu(i)^3/c^2)*(1/(exp(h*nu(i)/(k*T))-1));
    end
    plot(nu,L,col(j),'MarkerSize',taca,'LineWidth',epli);hold on;
    maxL(j)  = max(L);                                 % Maximum radiance
    Qn(j)    = sum(L)*incr*pi;                      % Integrated irradiance
end
disp  (['L 68, Intégrale NT(nu) : ', num2str(Qn*1e-6,3),' MW/m2'])
disp  (['L 69, Maximum NT(nu)   : ' ,num2str(maxL',3),' W/(m2 sr s)'])
legend('5780 K','5000 K','4000 K','3000 K')
xlabel(['Fréquence de : ',num2str(numin,3),' à : ',num2str(np*incr,3),...
    ' Hz'],'fontsize',taca)
ylabel('Radiance spectrale, W s /m2 /sr' ,'fontsize',taca);grid on;
title ('Radiance spectrale (fréquence)','fontsize',taca)
disp  (['L 75, Date, CPU, ',num2str(date),', ',num2str(toc(CPU)),' s',])
```

Table 19 : Procédure Matlab© Planck_f.m

Procédure *Planck_biban.m*, voir *Figure 7*.

```
CPU    = tic; y = [1 2 3 4 5];
x      = [11 99 1009 10020 10006];        % figure; semilogx(x,y);grid on;
h      = 6.62606957 * 10^(-34);
c      = 299792458;                                % Light speed in km/s
k      = 1.3806488 * 10^(-23);
cSB    = 5.6704 *10^(-8);                       % Stefan Boltzmann cste
np     = 1000;
xmax   = 60  *10^(-6);
```

```
xmin    = 0.15*10^(-6);
xmil    = 5*10^(-6);
pas     = (xmil-xmin)/(np-1);
taca    = 20;
tter    = 288;                                          % Earth temperatures
epli    = 2;
loga    = 1;
disp(['L 16, Use log. 1: yes, 0: no: ' ,num2str(loga)])
lambda = zeros(np,1);xscale= zeros(np,1);
L       = zeros(np,1);
t       = [5780 tter];                          % Sun and Earth temperatures
lamax   = zeros(2,1);
maxL    = zeros(2,1);
col     = ['r'  'b'];
disp(['L 23, Interv. long. ondes    : ' ,num2str(xmin*10^9),', ',...
    num2str(xmax*10^9),' nm'])
disp(['L 25, Stefan-Boltzmann       : ' ,num2str(cSB),    ' W/(m2K4)'])
% First curve : high temperature ========================================
figure('Position',[1 50 1000 600]);
axes('FontSize', taca,'LineWidth', epli)
T               = t(1);
lamax(1)        = 2.898*10^(-3)/t(1);                           % Wien s law
for i           = 1:np
    lambda(i) = xmin+(i-1)*pas;
    xscale(i) = log10(lambda(i));
    L(i)        = (2*h*c^2./lambda(i)^5)*(1/(exp(h*c/(k*lambda(i)*T))-1));
    hold on;
end
maxL(1)      = max(L);
if loga      == 0
    lamgra   = lambda*10^9;                         % displayed in nanometers
    semilogx(lamgra,L,'r','LineWidth', epli);hold on;grid on;
else
    plot(xscale,L,'r','LineWidth', epli);hold on;
end
% Second curve : low temperature ========================================
xmil      = 3*10^(-6);
pas       = (xmax-xmil)/(np-1);
T         = t(2);
lamax(2) = 2.898*10^(-3)/t(2);                                  % Wien s law
disp (['L 49, Wien, lamax visible    : ' ,num2str(lamax(1)*10^9),' nm'])
disp (['L 50, Wien, lamax infra rou : ' ,num2str(lamax(2)*10^9),' nm'])
disp (['L 51, Irradiance visible     : ' ,num2str(cSB*t(1)^4,3) ,' W/m2'])
disp (['L 52, Irradiance infra rou   : ' ,num2str(cSB*t(2)^4,3) ,' W/m2'])
for i     = 1:np
    lambda(i) = xmil+(i-1)*pas;
    xscale(i) = log10(lambda(i));
    L(i)        = (2*h*c^2./lambda(i)^5)*(1/(exp(h*c/(k*lambda(i)*T))-1));
    hold on;
end
maxL(2)    = max(L);
ratio12    = round(maxL(1)/maxL(2)/10^5)*10^5;              % rounded ratio
scale      = 500000;
maxL(2)    = maxL(2)*scale;
if loga == 0
    lamgra   = lambda*10^9;                         % displayed in nanometers
    semilogx(lamgra,L*scale,'b','LineWidth', epli);hold on;
    semilogx(lamax*10^9,maxL,'.k','MarkerSize',taca);hold on;
else
    plot(xscale,L*scale,'b','LineWidth', epli);hold on;
    plot(log10(lamax),maxL,'.k','MarkerSize',taca);hold on;
end
legend(' 5780 K',['    ',num2str(tter),' K'],'   Wien')
```

```
72 if loga ==0
73     xlabel('Longueur d onde (nanomètres)','fontsize',taca)
74 else
75     xlabel('Longueur d onde (log décimal)','fontsize',taca)
76 end
77 ylabel ( 'Radiance spectrale (W/m3/sr)' ,'fontsize',taca);grid on
78 title  (['Courbe bleue, y multiplié par ',num2str(scale)],'fontsize',taca)
79 disp   (['L 79, Rapp. max vi / max ir : ' ,num2str(ratio12)])
80 disp   (['L 80, Coeff. courbe bleue   : ' ,num2str(scale)])
81 disp   (['L 81, log10(4*1e-6)         : ' ,num2str(log10(4*1e-6),3)])
82 disp   (['L 82, Date, CPU, ',num2str(date),', ',num2str(toc(CPU)),' s',])
```

Table 20 : Procédure Matlab© Planck_biban.m

Procédure *Exitance.m* : exitance d'un corps noir

```
 1 CPU    = tic;
 2 h      = 6.62606957 * 10^(-34);          % Constante de Planck    en J.s
 3 c      = 299792458;                      % Vitesse de la lumière  en m/s
 4 k      = 1.3806488 * 10^(-23);           % Constante de Boltzmann en J/K
 5 sSB    = 5.6704 *10^(-8);                % Stefan-Boltzmann    en W/(m2K4)
 6 don    = [288 4000*1.e-9 100000*1.e-9];  % Teùperature, intervalle lambda
 7 n      = 1000;np = 200;T = don(1);xmin = don(2);xmax = don(3);
 8 incr   = (xmax-xmin)/n;
 9 sT4    = sSB*T^4*1e-6;                          % Stefan-Boltzmann solution
10 disp (['L 10, Temperature      : ', num2str(T ), ' K'])
11 disp (['L 11, Intervalle       : ', num2str([xmin*1.e9 xmax*1.e9]),' nm'])
12 disp (['L 12, Nombre de pas    : ', num2str(np)])
13 disp (['L 13, Stef.-Boltz. sT4 : ', num2str(sT4,3),' MW/m2'])
14 disp (['L 14, Max. wavelength  : ', num2str(xmax*1e9),' nm'])
15 L          = zeros(1,np);lam=zeros(1,np);
16 for i      = 1:n
17     lam(i) = xmin + incr/2 + (i-1)*incr;
18     L(i)   = (2*h*c^2./lam(i)^5)*(1/(exp(h*c/(k*lam(i)*T))-1));
19 end
20 siSB       = sum(L)*incr*pi/10^6/sT4;                          % Qis/Q
21 disp (['L 21, Int.(L) / st4    : ', num2str(siSB)])
22 disp (['L 22, Date, CPU, ',num2str(date),', ',num2str(toc(CPU)),' s',])
```

Table 21 : Procédure Matlab© Exitance.m : calcul de l'exitance d'un corps noir

Procédure *Separation_radiance.m*, voir *Figure 3* à *Figure 6* et *Table 3*

```
 1 CPU    = tic;
 2 taca   = 15;epli   = 2;            % Taille des caractères, Epais. traits
 3 h      = 6.62606957 * 10^(-34);    % Planck   constant in J.s
 4 c      = 299792458;                % Light velocity in m/s
 5 k      = 1.3806488  * 10^(-23);    % Boltzmann constant in J/K
 6 cSB    = 5.6704 *10^(-8);          % Stefan-Boltzmann constant in W/(m2K4)
 7 b      = 2.897771988* 10^(-3);     % Wien's displacement constant
 8 n      = 1000; T  = 288;  % 5780;  % Nombre de points et température (K)
 9 if T == 5780;xmin = 0       ;xmax=4.e-06  ;inwa = xmax/n       ;end
10 if T ==  288;xmin = 0.5*1e-7;xmax = 9*1e-5;inwa = (xmax-xmin)/n;end
11 numin  = round(c/xmax/n); numax=round(c/xmax*33);   % min. max frequencies
12 disp     (['L 12, Const. de Planck : ' ,num2str(h,6),          ' Js'])
13 disp     (['L 13, Vitesse lumière  : ' ,num2str(c,6),          ' m/s'])
14 disp     (['L 14, Cst de Boltzmann : ' ,num2str(k,6),          ' J/K'])
15 disp     (['L 15, Stefan-Boltzmann : ' ,num2str(cSB),     ' W/(m2K4)'])
16 disp     (['L 16, Cst depl de Wien : ' ,num2str(b,6),          ' mK'])
17 disp     (['L 17, Nombre de pas    : ' ,num2str(n)])
18 disp     (['L 18, Temp. corps noirs: ' ,num2str(T),            ' K'])
19 disp     (['L 19, Long. onde min.  : ' ,num2str(xmin),         ' m'])
20 disp     (['L 20, Long. onde max.  : ' ,num2str(xmax),         ' m'])
```

```
disp     (['L 21, Interv lon. ondes: ' ,num2str(inwa ),          ' m'])
figure   ('Position',[100 100 800 450]);              % ===== L (lam, T )
axes     ('FontSize',taca,'LineWidth',epli)
L          = zeros(1,n);lam=zeros(1,n);cum=zeros(1,n);  % initialisations
for i      = 1:n                          % Calcul de la radiance spectral
    lam(i)  = xmin + inwa/2 + (i-1)*inwa;
    L(i)    = (2*h*c^2./lam(i)^5)*(1/(exp(h*c/(k*lam(i)*T))-1));
end                     % pm=lm*inwa*1e9
[lalo,lm]   = max(L);m = 0;pm = b/T*1e9;    % Val. & pos. max rad. spectr.
disp     (['L 31, Radiance maximum : ' ,num2str(lalo,3),      ' W/m3'])
disp     (['L 32, Position du max. : ' ,num2str(pm,3) ,' nm'])
for i      = 2:n % Loop on L(i) to detect when exitance = 0.5 ==> median
    cum(i)  = sum(L(1:i))/T^4*inwa*pi/cSB;
    if m    == 0;if cum(i) > .5;m = i;end;end
end
plot (lam(1:m)  *10^9,L(1:m)  ,'b','LineWidth',epli);hold on;
plot (lam(m+1:n)*10^9,L(m+1:n),'r','LineWidth',epli);hold on;
xg = [lam(1:m) lam(m) 0]*10^9;                                  % blue zone
yg = [L(1:m) 0 0]    ;fill(xg,yg,[0  0.5  1. ])                 % blue zone
xg = [lam(m+1:n) lam(n) lam(m+1)]*10^9;                         % red zone
yg = [L(m+1:n) 0 0]  ;fill(xg,yg,[1.  0.25  0])                 % red zone
r1          = sum(L)/T^4*inwa*pi/cSB;               % Mean radianc. / T^4
disp (['L 43, SB  est. / exact  : ', num2str(r1)])
xlabel (['Longueur d onde, ',num2str(xmin*10^9,3),', ',...
    num2str(lam(m)*10^9,3),', ',num2str(xmax*10^9,3),' nm'],...
    'FontSize',taca)
ylabel ('Radiance spectrale, en W /(m3)' ,'fontsize',taca)
title  (['Radiance spectrale max. : ',num2str(lalo,3),' W/m3, pos.: ',...
    num2str(pm,3),' nm'],'FontSize', taca);grid on;
disp(['L 49, xmin lam(m) xmax : ',num2str([xmin lam(m) xmax],8),'  m'])
figure('Position',[100 100 800 450])                % ===== L ((?, T )
axes  ('FontSize',taca,'LineWidth',epli)
nu    = zeros(n,1);L = zeros(n,1);
incr  = (numax-numin)/(n-1);
numil = c/lam(m);                                  % Frequence médiane
ps    = 0;
for i = 1:n
    nu(i) = numin+(i-1)*incr;
    if ps ==0;if nu(i) >= numil;ps=i;end;end
    L(i)  = (2*h*nu(i)^3/c^2)*(1/(exp(h*nu(i)/(k*T))-1));
end
plot(nu(1  :ps),L(1  :ps),'r','MarkerSize',taca,'LineWidth',epli);hold on;
plot(nu(ps+1:n),L(ps+1:n),'b','MarkerSize',taca,'LineWidth',epli);hold on;
xxg = [nu(1:ps); nu(ps) ; 0];yyg=[L(1:ps)'  0   0];
fill(xxg',yyg',[ 1. 0.25 0]);hold on                              % rouge
xxg = [nu(ps+1:n); nu(n); nu(ps+1)];yyg=[L(ps+1:n)' 0 0];
fill(xxg',yyg',[0 0.5 1.])                                         % bleu
[maxL ,nn]= max(L);                                  % Radiance maximum
pom       = nu(nn);        % Fréquence correspondant à la radiance maximum
siSB      = round((sum(L)/T^4*incr*pi)/cSB*100)/100;
fi = round(min(nu/10^9));fm=round(numil/10^9);fs=round(max(nu/10^9));
disp     (['L 72, Minimum frequ.   : ' ,num2str(fi,3),' GHz'])
disp     (['L 73, Maximum frequ.   : ' ,num2str(fs,3),' GHz'])
disp     (['L 74, Freq at max. rad.: ', num2str(fm*1e9,3),' Hz'])
disp     (['L 75, SB est. / exact  : ', num2str(siSB)])
title  ('2 sections égales radiance spectrale ','FontSize', taca);grid on;
xlabel (['Frequence: ',num2str(fi), ',   ',num2str(fm),',   ',...
    num2str(fs),' GHz'],'fontsize',taca)
ylabel( 'Radiance spectrale, en  W /m2 /sr /Hz' ,'fontsize',taca);grid on;
disp (['L 80, Date, CPU, ',num2str(date),', ',num2str(toc(CPU)),' s',])
```

Table 22 : Procédure Matlab© Separation_radiance.m pour créer les Figure 3 à Figure 6

Procédure Matlab© *Constante_solaire.m* – affichage de constantes

```
1  SB = 5.6704e-8;   disp([' SB : ',num2str(SB,4),' W/(m2K4'])  % Stef.-Boltz.
2  rs = 696342000;   disp([' rs : ',num2str(rs,4),' m'])       % rayon soleil
3  ds = 149.5979e9;  disp([' ds : ',num2str(ds),' m'])    % dist. soleil terre
4  ss = 4*pi*rs^2;   disp([' ss : ',num2str(ss,4),' m2'])% aire surf. soleire
5  t  = 5780;        disp([' t  : ',num2str(t ),' K'])    % température soleil
6  em = SB*t^4;      disp([' em : ',num2str(em,4),' W/m2'])% Emittance soleil
7  cs = em*(rs/ds)^2;disp([' cs : ',num2str(cs,4),' W/m2'])     % cste solaire
8  as = cs/4;        disp([' as : ',num2str(as,4),' W/m2'])  % apport solaire
```

Table 23 : Procédure Matlab© Constante_solaire.m – affichage de constantes

La procédure *cartsphe.m* (*Table 24*) fait appel aux fonctions *gra_mel3.m* (*Table 35*) et *gra_mels.m* (*Table 36*). Elle fournit les projections axonométriques du cube et du cube sphérique, qui est la projection du cube sur la sphère.

Procédure Matlab© *cartsphe.m* – affichage des axonométries du cube

```
1  CPU   = tic; a =1; b = a; c = a;vol=a*b*c;     % \Radiosité_2017\cartsphe.m
2  V     = [0 0 0;a 0 0;a b 0;0 b 0;0 0 c;a 0 c;a b c;0 b c];%8 cube vertices
3  lel   = [1 2 3 4 5 6 7 8];k = 12; deb = 1; % Element definition 8 vertices
4  fs    = 1;
5  disp(['T  05, Num. hexa.: ',num2str(size(lel,1))])
6  disp(['T  06, Cube dim. : ',num2str(a),' m'])
7  disp(['T  07, Cube vol. : ',num2str(vol),' m3'])
8  if deb==1; disp('T  08, Call function: ..... gra_mel3 .....');end
9  gra_mel3(V,lel,fs);
10 if deb==1; disp('T  10, Call function: ..... gra_mels .....');end
11 gra_mels(V-[.5 .5 .5],lel,fs);
12 disp(['T  12, Date, CPU : ',num2str(date),', ',num2str(toc(CPU)),' s',])
```

Table 24 : Procédure Matlab© cartsphe.m – affichage des axonométries

La procédure *Rayocube.m* (*Table 25*) fait appel aux fonctions *fmesh_cube_6f.m* (*Table 37*), *vecprouni.m* (*Table 27*), *fvuelamb.m* (*Table 38*), *br86.m* (*Table 40*), *lalo.m* (*Table 33*), *clos.m* (*Table 41*)

Procédure *Matlab© Rayocube.m*.

```
1  tini   = tic;deb=0;nit=50;n56 = 56;
2  rho    =.5;disp(['L   2, coeff. refl. : ',num2str(rho)])
3  nx=4;ny=4;nz=4;% nx=8;ny=8;nz=8;%  nx=2;ny=2;nz=2;% nx = 1;ny=1;nz=1;%
4  % cou  = {'k','k','k','k','b','r'};% xx ='k';cou   = {xx,xx,xx,xx,xx,xx};
5  cx     = 1;cy = 1;cz = 1;                              % Dimensions du cube
6  ila    = 1;     % 1: utilisation de la méthode de Lambert - 1 point de Gauss
7  nr     = 300; if ila == 1;nr = 0;end                % nr = nombre de rayons
8  if ila > 0;disp(['L   6, Lambert, oui : ',num2str(ila)]);end
9  if nr  > 0;disp(['L   7, Nomb. rayons : ',num2str(nr )]);end
10 disp(['L   9, Maillage     : ',num2str(nx),' x ',num2str(ny),' x ',...
11     num2str(nz)])
12 if deb==1;disp('L  12, Call function: ..... fmesh_cube_6f .....');end
13 struc     = fmesh_cube_6f(cx,cy,cz,nx,ny,nz);% Maillage des 6 faces du cube
14 disp(['L  13, Dim. struc   : ',num2str(size(struc))])
15 nel       = size(struc,1)/4; % Calcul norm. & centres grav. des "nel" carr.
16 np        = zeros(nel,3);cg=zeros(nel,3);aires=zeros(nel,1);
17 Rad       = zeros(nel,nel);                          % Matrice de radiosité
18 disp(['L  20, N.carr. = nel: ',num2str(nel)])
19 if deb==1;disp('L  28, Call function: ..... vecprouni .....');end
20 for ktri         = 1 : nel
21     c1           = struc((ktri-1)*4+2,:)-struc((ktri-1)*4+1,:);     % s2-s1
22     c2           = struc((ktri-1)*4+3,:)-struc((ktri-1)*4+1,:);     % s3-s1
23     c3           = struc((ktri-1)*4+4,:)-struc((ktri-1)*4+1,:);     % s4-s1
24     np(ktri,:)   = vecprouni(c1,c2);     % Normale unitaire du carreau ktri
```

```
    jk          = (ktri-1)*4;   % Nombre de sommets des carreau précédents
    aa          = norm(c1);                             % Longueur du coté 1-2
    bb          = norm(c3);                             % Longueur du coté 1-4
    aires(ktri) = aa*bb;                                % Aire du carreau ktri
    for jj      = 1:4                               % Boucle sur les 4 sommets
        cg(ktri,:) = cg(ktri,:)+struc(jk+jj,:)/4; % Centre gravité carreau
    end
end
if nel < 7;disp(['L  49, Aires        : ',num2str(aires'),' m2']);end
if ila == 1  % Calcul de la matrice de radiosité par la méthode de Lambert
    som = zeros(3,5);nor = zeros(3,1);
    if deb==1;disp('L  53, Call function: ..... fvuelamb  .....')
              disp('L  53, Call function: ..... vecprouni .....');end
    for ktri     = 1:nel                        % voir formule 1.14 du rapport
        for ktrj = 1:nel % Beckers 20200815 - Angle solide et Fact. de vue
            Rad (ktri,ktrj) = fvuelamb(struc,ktri,ktrj);
        end;
    end;
    Rad = Rad     - eye(nel);                                       % disp(Rad)
    M   = eye(nel)+ rho*Rad ;                                       % disp(M  )
          % Fin du calcul matrice de radiosité M par la méthode de Lambert
else             % Calcul de la matrice de radiosité  par lancer de rayons
    if deb==1;disp('L  64, Call function: ..... fvlr .....');end
    M     = rho*fvlr(nr,cx,cy,cz,nel);
    for i = 1 : size(M); M(i,i) = 1;end      % disp(fvlr(nr,cx,cy,cz,nel))
    disp(['L  66, size(M)      : ',num2str(size(M))])             % disp(M)
end                                     % Fin calcul matrice de radiosité Rad
emit         = zeros(nel,1);                                % air=zeros(nel,1);
emit(1:nx*ny)= 1;                % Exitance constante sur la face supérieure
if nel < 7;disp(['L  67, Emittances E : ',num2str(emit'),' W/m2']);end
B      = M\emit;                           % Calcul du vecteur des radiosités
if nel < 7;disp(['L  69, Radiosit. B  : ',num2str(B'   ),' W/m2']);end
% ravi(1:nel-nx*ny) = B((nx*ny+1):nel);          % Dessin des radiosités B
fvmin        = min(B);
fvmax        = max(B);
deltafv      =.281-.101;%.281-.112;%.229-.133;%deltafv=0.169;%fvmax-fvmin;
disp(['L  74, B min & + del: ',num2str([fvmin fvmin+deltafv])])
disp( 'L  76 - 81, Surrounding ellipse in Mollweide projection')
figure
bd       = -pi : 5*pi/180 :pi;        % Discretization angulaire du cercle
aMol     = 2*sqrt(2)*cos(bd);
bMol     = sqrt(2)  *sin(bd);
plot(aMol,bMol,'k','LineWidth',1);hold on;axis equal;grid on
disp(['L  81, a, b ellipse : ',num2str([max(aMol)*2 max(bMol)*2])])
if deb==1;disp('L  84, Call function: ..... br56 .....');end
coul  = br56;  br56; colormap(br56);
Quad         = zeros(4,3);
dis          = zeros(5,1);
polyx        = zeros(5,1);
polyy        = zeros(5,1);
tc           = zeros (5,3);
sph = zeros(5,3);spd = zeros(5,3);lalor=zeros(5,2);xy=zeros(5,2);
if deb==1;disp('L  92, Call function: ..... lalo .....');end
elde         = 0;
k3=0;k1 =1; k2 = nel;%k1 = 90;k2=91;k3=90;
disp(['L  95, Dessin élém. : ',num2str(k1),' - ',num2str(k2)])
for ktri     = k1 : k2
    cg       = zeros(1,3);
    for jj   = 1 : 4                                    % Un quadrilatère
        Quad(jj,:) = struc((ktri-1)*4+jj,:);
        cg(1,:)    = cg(1,:) +Quad(jj,:)/4;
    end
    cg       = cg - [0.5 0.5 0.5];
```

```
88      if ktri == k3;disp(['L 103, cg            : ',num2str(cg)]);end
89      tc(1,:)  = Quad(1,:)-[0.5 0.5 0.5];
90      tc(2,:)  = Quad(2,:)-[0.5 0.5 0.5];
91      tc(3,:)  = Quad(3,:)-[0.5 0.5 0.5];
92      tc(4,:)  = Quad(4,:)-[0.5 0.5 0.5];
93      tc(5,:)  = Quad(1,:)-[0.5 0.5 0.5];  % 4 sommets du quad. + le premier
94      if ktri == k3;disp('L 109 Quad');disp(Quad);end
95      if ktri == k3;disp('L 110   tc');disp(tc);end
96      for k    = 1:5                  % Coordonnées sphériques de l'élément
97          sph(k,:)     = lalo(tc(k,:));% avec rot. de pi/4, autour axe vert.
98          lalor(k,1:2) = sph(k,2:3);
99          lalor(k,2)   = norm(lalor(k,2))*sign(cg(2));    % !!!!!!!!!!!!!!!
100         beta         = BMinr(lalor(k,1),nit)/2;
101         xy(k,1)      = 2*sqrt(2)*cos(beta)*lalor(k,2)/pi;
102         xy(k,2)      =   sqrt(2)*sin(beta);
103     end
104     if ktri == k3
105         disp('L 120 min latitude of elem. : ')
106         disp([ktri min(lalor(:,1)*180/pi)])
107         disp('L 122 max latitude of elem. : ')
108         disp([ktri max(lalor(:,1)*180/pi)])
109     end
110     if ktri == k3;disp('L 127 lalor ');disp(lalor*180/pi);end
111     if ktri == k3;disp('L 128 xy ');disp(xy);end
112     for n = 2 : 5
113         if norm(xy(n,1)) < 0.00001 ; xy(n,1) = 0;end
114         if sign(xy(n-1 ,1)) == sign(xy(n ,1))
115         plot([xy(n-1,1) xy(n ,1)],[xy(n-1,2) xy(n,2)],'k');hold on
116         grid on;axis equal
117         elde = elde + 1;
118         end
119     end
119     vf           = B(ktri);
120     icouleur     = min(floor((vf-fvmin)*n56/deltafv)+1,n56);
121     fill(xy(:,1),xy(:,2),coul(icouleur,:));hold on % ,'LineStyle','none'
122     colorbar;grid off
123 end
124 disp(['L 139, N. cotés des.: ',num2str(elde)])
125 title(['Eléments ',num2str(k1),' - ',num2str(k2)]);hold on,axis off
126 % title(['Mesh: ',num2str(nx),' x ',...
127 %     num2str(ny),' x ',num2str(nz),', Som B * aires : ',...
128 %     num2str(B'*aires),' W'],'fontsize', 15);hold on;axis on
129 % xlabel(['ila ',num2str(ila),', B min + Del : ',num2str(fvmin+deltafv,...
130 %     '%0.3g'),', B min : ',num2str(fvmin,'%0.3g'),' W/m2'],'fontsize',...
131 %     15);hold on;axis off
132 if deb==1;disp('L 144, Call function: ..... clos .....');end
133 disp(['L 148, Ferm. moyenne: ',num2str(clos(M,nel,rho))])
134 disp(['L 149, B*aires      : ',num2str(B'*aires),' W'])
135 disp(['L 150, CPU total    : ',num2str(toc(tini),'%0.4g'),' s'])
```

Table 25 : Procédure Matlab© Rayocube.m

La procédure *Mollcube.m* (*Table 26*) appelle les fonctions *lalo.m* (*Table 33*), *Mintff.com* (*Table 34*), *BMinr.com* (*Table 28*)

Procédure Matlab© *Mollcube.m* : dessin du cube en projection de Mollweide

```
1   xyz =[0 0 0 ;1 0 0 ;1 1 0 ;0 1 0 ; 0 0 1;1 0 1 ;1 1 1 ;0  1 1;]-[.5 .5 .5];
2   disp(['L 02, Nb. nodes size(xyz,1): ',num2str(size(xyz,1))])
3   sph = zeros(size(xyz,1),3);spd=zeros(size(xyz,1),3);
4   coul         = ['k' 'r' 'g' 'b'];gc=0;deb=1;
5   % disp('cartesian coord. of the cube, origin in its center');disp(xyz);
```

```
 6  mi1=[(xyz(1,:)+xyz(2,:))/2;(xyz(3,:)+xyz(4,:))/2];
 7  mi2=[(xyz(4,:)+xyz(1,:))/2;(xyz(2,:)+xyz(3,:))/2];
 8  disp(['L 07; Mediane inf 1-2 3-4  : ',num2str([mi1(1,:) mi1(2,:)])])
 9  disp(['L 08; Mediane inf 2-3 4-1  : ',num2str([mi2(1,:) mi2(2,:)])])
10  if deb==1;disp('L 12, Call function: ..... lalo .....');end
11  for i = 1 : size(xyz,1)  % Spher. coord. transf. : radius, latitude, long.
12      sph(i,:) = lalo(xyz(i,:));
13  end
14  for i            = 1 : size(xyz,1)     % Transforming radians into degrees
15      spd(i,1)     = sph(i,1);
16      for j        = 2 : 3; spd(i,j) = sph(i,j)*180/pi; end
17  end                          ; % disp('spherical coordinates spd');disp(spd)
18  % cu = [1 2;2 3;3 4;4 1;5 6;6 7;7 8;8 5;1 5;2 6;3 7;4 8]; % Edges sequence
19    cu = [1 2;2 3;3 4;4 1;5 6;6 7;7 8;8 5;    2 6;3 7;4 8]; % Edges sequence
20  disp(['L 20, Nb. arêtes size(cu,1): ',num2str(size(cu,1))])
21  figure        % BMollw ([1 9 22 41 64 91 120 150]) % 300 equal areas cells
22  % disp( 'L 21, 23 - 26 : Surrounding ellipse in Mollweide projection')
23  bd       = -pi : 5*pi/180 :pi;        % Angular discretization of a circle
24  aMol     = 2*sqrt(2)*cos(bd);
25  bMol     = sqrt(2)  *sin(bd);
26  plot(aMol,bMol,coul(2),'LineWidth',1);hold on;axis equal;grid on
27  % disp('L 27, Mollweide projection of the cube')
28  if deb==1;disp('L 28, Call function: ..... Mintff .....');end
29  for i = 1 :  size(cu,1)
30      lalod =[spd(cu(i,1),2) spd(cu(i,1),3);spd(cu(i,2),2) spd(cu(i,2),3)];
31      Mintff(lalod,36,coul(1),gc)
32  end;axis off
33
34  sfm   = zeros(size(mi1,1),3);  % Prem. médianes des 2 faces horizontales
35  if deb==1;disp('L 37, Call function: ..... lalo .....');end
36  for i = 1 : size(mi1,1)% Spher. coord. transf. : radius, latitude, long.
37      sfm(i,:) = lalo(mi1(i,:));
38  end ;                                                  % disp(sfm*180/pi)
39  if deb==1;disp('L 41, Call function: ..... Mintff .....');end
40  lalod =[sfm(1,2) sfm(1,3);sfm(2,2) sfm(2,3)]*180/pi;
41  Mintff(lalod,36,coul(2),gc);hold on;
42  lalod =[-sfm(1,2) sfm(1,3);-sfm(2,2) sfm(2,3)]*180/pi;
43  Mintff(lalod,36,coul(2),gc);hold on; %  disp(sfm);disp(lalod)
44  lalod =[sfm(1,2) sfm(1,3);-sfm(1,2) sfm(1,3)]*180/pi; % 2 médianes vert.
45  Mintff(lalod,36,coul(2),gc);hold on;
46  lalod =[sfm(2,2) sfm(2,3);-sfm(2,2) sfm(2,3)]*180/pi;
47  Mintff(lalod,36,coul(2),gc);hold on;
48  sfm   = zeros(size(mi2,1),3);%Secondes médianes des 2 faces horizontales
49  if deb==1;disp('L 51, Call function: ..... lalo .....');end
50  for i = 1 : size(mi2,1)% Spher. coord. transf. : radius, latitude, long.
51      sfm(i,:) = lalo(mi2(i,:));
52  end ;                                                  % disp(sfm*180/pi)
53  if deb==1;disp('L 56, Call function: ..... Mintff .....');end
54  lalod =[sfm(1,2) sfm(1,3);sfm(2,2) sfm(2,3)]*180/pi;
55  if lalod(2,2) == lalod(1,2);lalod(2,2) = lalod(1,2)+180;end
56  Mintff(lalod,36,coul(2),gc);hold on;
57  lalod =[-sfm(1,2) sfm(1,3);-sfm(2,2) sfm(2,3)]*180/pi;
58  if lalod(2,2) == lalod(1,2);lalod(2,2) = lalod(1,2)+180;end
59  Mintff(lalod,36,coul(2),gc);hold on; % disp(sfm);disp(lalod)
60  lalod =[lalod(1,1) lalod(1,2);-lalod(1,1) lalod(1,2)];
61  Mintff(lalod,36,coul(2),gc);hold on;
62  lalod =[lalod(2,1) lalod(2,2)+180;-lalod(2,1) lalod(2,2)+180];
63  Mintff(lalod,36,coul(2),gc);hold on
64  lalod        = [0 -180 ; 0 180];nit=5 ;             % Tracé de léquateur
65  disp(['L 65, Equateur extrémit.   : ',num2str([lalod(1,:) lalod(2,:)])])
66  lalor        = lalod*pi/180;
67  xy           = zeros(size(lalor,1),2);
68  if deb==1;disp('L 68, Call function: ..... BMinr .....');end
```

```
for i         = 1 : size(lalor,1)
    beta      = BMinr(lalor(i,1),nit)/2;
    xy(i,1)   = 2*sqrt(2)*cos(beta)*lalor(i,2)/pi;
    xy(i,2)   =   sqrt(2)*sin(beta);
end
plot([xy(1,1) xy(2,1)],[xy(1,2) xy(2,2)],coul(2),'LineWidth',1);hold on
if deb==1;disp('L 75, End of procedure Mollcube.m .....');end
```

Table 26 : Procédure Matlab© Mollcube.m : projection cube et médianes des faces

Fonction *vecprouni.m* (*Rayocube.m*)

```
function [n] = vecprouni(c1, c2)
n = cross(c1,c2)';
n = n/norm(n);
end
```

Table 27 : Fonction Matlab© vecprouni.m (Rayocube.m)

Fonction *BMinr.m* (*BMollw.m*

```
function[laMo] = BMinr(lat,nit)  % Compute the latitude in Mollweide proj.
th      = zeros(1,nit);la = lat;    % the latitude is expressed in radians
th(1)   = la -(la+sin(la)-pi*sin(la))/(1+cos(la));
for i   = 2:nit % nit = iterations used to solve the Newton Raphson scheme
  th(i) = th(i-1)-(th(i-1)+sin(th(i-1))-pi*sin(la))/(1+cos(th(i-1)));
end
laMo    = th(nit);    % laMo = new latitude in Mollweide projection (rad.)
end
```

Table 28 : Fonction Matlab© BMinr.m (BMollw.m)

Cette fonction appelle *Bwba.m* (*Table 31*)

Fonction *Bsphe.m* : dessin de cellules égales sur une sphère

```
function [drays] = Bsphe(S,tr,di)            % cells drawing on the sphere
npas  = 8;b=(0:10:360)*pi/180;               % Parameter for the execution
Nc    = size(S,2);                        % Size of the interior points set
Nt    = max(S);                         % Number of cells in the hemisphere
aleax = rand(Nt);aleay=rand(Nt);     % Generation of 2 x Nt random numbers
R     = [S(1) S(2:Nc)-S(1:Nc-1)] ;              % Definition of the rings
t     = [0 acos(1-2*S/Nt)];r=sin(t);h=cos(t); % From disk equal area to 3D
for i = 2 : Nc      % Drawing the Nc -1 parallels in axnometric projection
    x = r(i)*cos(b);y = r(i)*sin(b);z = ones(size(b,2))*h(i);
    if di > 0 ;plot3(x,y,z,'k'); hold on;end
end
if tr > 0;la = zeros(Nt,1);lo = zeros(Nt,1);n=1;end
if S(1)==1; p=2; q=Nc-1; else; p=1; q=Nc; end
for i     = p : q                          % 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);
if di>0; plot3(mxi',myi',mzi','k');hold on;end  % Drawing meridian segment
        if tr == 1                        % If tr == 1: random rays generation
            n     = n+1;la(n) = t(i)+aleax(n)*(t(i+1)-t(i));
                        lo(n) = lp  +aleay(n)*(longi-lp);lp=longi;
        end
        if tr == 2              % If tr == 2: deterministic rays generation
            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 tr>0 ;drays=[la(2:Nt+1,1) lo(2:Nt+1,1)];end                    % Output
if di>0 ;Bwba;hold on;axis equal;axis off;end                     % or. sp.
end
```

Table 29 : Fonction Matlab© Bsphe.m : dessin de cellules égales sur la sphère

Fonction *Bsams.m* : dessin de cellules égales sur une sphère

```
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 30 : Fonction Matlab© Bsams.m : calcul de séquence de cellules égales sur la sphère

Fonction *Bwba.m* : dessin d'une sphère blanche opaque

```
function [xx,yy,zz] = Bwba(n)
%   Bwba generate a white opaque sphere
%   [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 = 50.
%   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);                             % orange sphere
if nargout == 0                                 % computed or returned result
   surf(x,y,z,'EdgeColor','none')     % To see the mesh, replace none by k
else
   xx = x; yy = y; zz = z;
end
```

Table 31 : Fonction Matlab© Bwba.m : dessin d'une sphère blanche opaque (Bsphe.m)

La fonction *BMollw.m* fait appel plusieurs fois à *BMinr.m* (*Table 28*)

Fonction *BMollw.m* : dessin de cellules égales en projection de Mollweide

```
function[] = BMollw(S)
t0       = tic;if max(S)> 20;npas=3;else;npas=8;end
figure('Position',[1 1 1024 512])
disp([' S  ',num2str(S)])
bd       = -pi : 5*pi/180 :pi ;              % angular discret. of a circle
ax       = 2*sqrt(2)*cos(bd); ay = sqrt(2)*sin(bd);
plot(ax,ay,'k','LineWidth',2);hold on                % Drawing the ellipse
nit      =   50;                     % nit = numb.iterations to compute beta
nan      = size(S,2);                                % nan = Number of rings
nan1     = nan + 1;                                    % Number of parallels
vr       = zeros(nan1,1); vr(2) = 1;theta = ones(nan1,1)*pi/2;
for i    = 3:nan1; vr(i) = vr(i-1)*sqrt(S(i-1)/S(i-2)); end;
vr       = vr/vr(nan1);                % Radius of the external circle = 1
for i        = 1:nan1
    theta(i) = pi/2-(2*asin(vr(i)*sqrt(2)/2));% latitudes of the parallels
    latro    = theta(i);
    beta     = BMinr(latro,nit)/2;
    xx       = (sqrt(2)*cos(beta)/pi*[-pi pi]*2);
    yy       = sqrt(2)*sin(beta);
                plot(xx, [yy yy],'r');hold on;axis equal;grid on    % North
    if i < nan1;plot(xx,-[yy yy],'r');hold on;end                   % South
end
for i            = 1:nan
    if i == 1; n = S(1);else;n = S(i)-S(i-1);end;k = -pi- 2*pi/n;
    for j        = 1 : n+1
        k        = k + 2*pi/n;             % Drawing the meridian sections
        latro    = theta(i);latrp = theta(i+1);
        palat    = (latrp-latro)/npas;
        for m      = 1:npas
            beta   = BMinr(latro+(m-1)*palat,nit)/2;
            xx1    = 2*sqrt(2)*cos(beta)*k/pi;
            yy1    = sqrt(2)*sin(beta);
            beta   = BMinr(latro+(m  )*palat,nit)/2;
            xx2    = 2*sqrt(2)*cos(beta)*k/pi;
            yy2    = sqrt(2)*sin(beta);
            plot( [xx1 xx2], [yy1 yy2],'k');hold on;axis equal;grid on
            plot( [xx1 xx2],-[yy1 yy2],'k');hold on;axis equal;grid on
        end
    end
end
axis off
% title(['Mollweide projection ; ',num2str(max(S)*2),' cells'])
disp (['Mollweide projection ; ',num2str(max(S)*2),' cells'])
disp (['Mollweide iter. cpu  = ',num2str([nit toc(t0)]),' sec'])
end
```

Table 32 : Fonction Matlab© BMollw.m : cellules égales en projection de Mollweide

Fonction *lalo.m*, appelée par *Rayocube.m (Table 25)* et *Mollcube.m (Table 26)*

```
function [sph] = lalo(xyz)
x      = xyz(1);y = xyz(2);z = xyz(3);
r      = sqrt(x^2 + y^2 + z^2);                      % r = distance radiale
th     = acos(z / r);                  % si z = 0, th = 90°, latitude = 0
if th  > pi   ; th = 2*pi-th ; end
if th  < - pi ; th = 2*pi+th ; end
ho     = r * sin(th);
```

```
 8 if ho == 0
 9     th = 0;fi = 0;
10 end
11 if norm(ho) > 0
12 fi     = asin(y/ho);              % fi     = asin(y/ho);fi     = acos(x/ho)
13 if  ho > 0                      % si on n'est pas sur l'axe vertical
14     if x == 0                                % x = 0
15         if y > 0                             % y > 0
16             fi =  pi/2;
17         else
18             fi = -pi/2;
19         end
20     else                                     % x > 0 ou x < 0
21         if y > 0                             % y > 0
22             fi = acos(x/ho);
23         else                                 % y = 0 ou y < 0
24             if x > 0                         % x > 0 et y < 0
25                 fi = - acos(x/ho);
26             else
27                 fi =   acos(x/ho) + pi;
28             end
29         end
30     end
31     if x < 0
32 %         fi=sign(y)*acos(x/ho);
33 %         if y <  0;fi = -acos(x/ho);end
34 %         if y == 0;fi =  acos(x/ho);end
35         if y == 0;fi =sign(x)*acos(x/ho);end
36     end
37     if y < 0
38          fi = -acos(x/ho);
39     end
40 %         if fi > 0
41 %             fi = -fi;
42 %         end
43 %     end
44 end
45 % if fi > pi;fi=fi-2*pi;end
46 end
47 sph = [r pi/2-th fi];   % sph = latitude et longitude décalée de pi/4
48 % sph = [r pi/2-th fi-pi/4]; % sph = latitude et longitude décalée de pi/4
49 end
```

Table 33 : Fonction Matlab© lalo.m

La fonction *Mintff.m fait* appel à *BMinr.m* (*Table 28*)

Fonction *Mintff.m* : dessins d'arcs de grands cercles

```
 1 function[]     = Mintff(lalod,ni,coul,gc)            % Arc de grand cercle
 2 if gc > 0
 3     lalor          = [lalod(1:2) ;lalod(3:4)]*pi/180;
 4 else
 5     lalor          = lalod*pi/180;
 6 end
 7 P1  = [sin(pi/2-lalor(1,1))*cos(lalor(1,2)) ...
 8        sin(pi/2-lalor(1,1))*sin(lalor(1,2)) (cos(pi/2-lalor(1,1)))];
 9 P2  = [sin(pi/2-lalor(2,1))*cos(lalor(2,2)) ...
10        sin(pi/2-lalor(2,1))*sin(lalor(2,2)) (cos(pi/2-lalor(2,1)))];
11 pt              = zeros(ni,3);
12 pt(1,:)         = P1;
13 pt(ni+1,:)      = P2;
```

```
for i           = 1:ni-1
    aa          = norm(P1+i*(P2-P1)/ni);
    pt(i+1,:)   = (P1+i*(P2-P1)/ni)/aa;
end
% if gc == 1;disp (pt);end
laalo           = zeros(size(pt,1),2);
for i           = 1:size(pt,1)
    laalo(i,1)  = acos(pt(i,3));
    laalo(i,2)  = acos(pt(i,1)/sin(laalo(i,1)))*sign(pt(i,2));
    laalo(i,1)  = pi/2-laalo(i,1);
end
xy            = zeros(size(laalo,1),2);
nit           = 5;
for i         = 1 : size(pt,1)
    beta      = BMinr(laalo(i,1),nit)/2;
    xy(i,1)   = 2*sqrt(2)*cos(beta)*laalo(i,2)/pi;
    xy(i,2)   =   sqrt(2)*sin(beta);
    if norm(xy(i,1)) < 0.00001 ; xy(i,1) = 0;end
    if i > 1
        if sign(xy(i-1,1)) == sign(xy(i,1))
        plot([xy(i-1,1) xy(i ,1)],[xy(i-1,2) xy(i,2)],coul,'LineWidth',1);
        end
    end
end
hold on;axis equal;
end
```

Table 34 : Fonction Matlab© Mintff.m appelée par Mollcube.m (Table 26)

Fonction *gra_mel3.m* : dessin des arêtes et des labels (*Table 24*)

```
function [] = gra_mel3(xyz,lel,fs)            % Drawing the edges & labels
disp(['gm3 2, Vert. num.: ',num2str(lel)])% disp(xyz)
sh     = .1;figure
for i  = 1:size(lel,1)
    lK = [lel(i,1) lel(i,2);lel(i,2) lel(i,3);lel(i,3) lel(i,4); ...
          lel(i,4) lel(i,1);lel(i,5) lel(i,6);lel(i,6) lel(i,7); ...
          lel(i,7) lel(i,8);lel(i,8) lel(i,5);lel(i,1) lel(i,5); ...
          lel(i,2) lel(i,6);lel(i,3) lel(i,7);lel(i,4) lel(i,8)];
for k  = 1 : size(lK,1)           % ne = size(lK,1) is the number of edges
    for j  = 1 : size(lK,1)                          % Drawing the ne edges
        n1 = lK(j,1);
        n2 = lK(j,2);
        x1 = xyz(n1,:)+(xyz(n2,:)-xyz(n1,:))*sh;
        x2 = xyz(n2,:)-(xyz(n2,:)-xyz(n1,:))*sh;
        plot3([x1(1,1),x2(1,1)],[x1(1,2),x2(1,2)],[x1(1,3),x2(1,3)],...
            'Color','r','LineWidth',fs)
    end
    for l  = 1:size(xyz,1)                                  % Nodes labels
        xx = xyz(l,:);
        text(xx(1),xx(2),xx(3),num2str(lel(l)),'Color','k','Fontsize',15)
    end
    for n = 1:size(lel,1)                                % Elements labels
        xx(n,1)=sum(xyz(lel(n,:),1))/size(lel,2);
        xx(n,2)=sum(xyz(lel(n,:),2))/size(lel,2);
        xx(n,3)=sum(xyz(lel(n,:),3))/size(lel,2);
    end
    for n  = 1:size(xx,1)
        text(xx(n,1),xx(n,2),xx(n,3),num2str(n),'Fontsize',15,'Color','b')
    end
    hold on;grid on;axis equal;axis off
end
```

```
32  end
33  end
```

Table 35 : Fonction Matlab© gra_mel3.m : dessin des arêtes et des labels (cartsphe.m)

Fonction *gra_mels.m* : arêtes et labels en coordonnées sphériques (*Table 24*).

```
1   function [] = gra_mels(xyz,lel,fs)          % Drawing the edges & labels
2   disp(['gms 2, Vert. num.: ',num2str(lel)]) % lel = element local vector
3   ne     = size(lel,1);
4   for i  = 1  : ne
5       lK = [lel(i,1) lel(i,2); lel(i,2) lel(i,3); lel(i,3) lel(i,4); ...
6             lel(i,4) lel(i,1); lel(i,5) lel(i,6); lel(i,6) lel(i,7); ...
7             lel(i,7) lel(i,8); lel(i,8) lel(i,5); lel(i,1) lel(i,5); ...
8             lel(i,2) lel(i,6); lel(i,3) lel(i,7); lel(i,4) lel(i,8)];
9       disp(['gms 9, size xyz  : ',num2str(size(xyz))])
10      disp(['gms10, size lel  : ',num2str(size(lel))])
11      disp(['gms11, size lK   : ',num2str(size(lK))])
12      disp(['gms12, arc thick.: ',num2str(fs)])
13      figure                                    % Drawing the ne = 12 edges
14      nj     = 18;     % Numbeer of imposed segments on the circular arcs
15      for j  = 1 : size(lK,1)
16          n1 = lK(j,1);
17          n2 = lK(j,2);
18          P1 = xyz(n1,:);aa=norm(P1);P1=P1/aa; % Proj. on the unit sphere
19          P2 = xyz(n2,:);aa=norm(P2);P2=P2/aa;
20          pt             = ones(3,1)*P1;
21          pt(1,:)        = P1;
22          pt(nj+1,:)     = P2;
23          for k          = 2:nj
24              aa         = norm(P1+k*(P2-P1)/nj);% Proj. on the unit sphere
25              pt(k+1,:) =      (P1+k*(P2-P1)/nj)/aa;
26              plot3([pt(k,1),pt(k+1,1)],[pt(k,2),pt(k+1,2)],...
27                  [pt(k,3),pt(k+1,3)],'Color','b','LineWidth',fs);hold on
28          end
29      end
30  %     for l  = 1 : size(xyz,1)             % Displaying the node labels
31  %         xx = xyz(l,:);
32  text(xx(1),xx(2),xx(3),num2str(lel(l)),'Color','k','Fontsize',15)
33  %     end
34      hold on;grid on;axis equal;axis off
35  end
36  end
```

Table 36 : Fonction Matlab© gra_mels.m Arêtes projetées sur la sphère unitaire (cartsphe.m)

Fonction *Matlab© fmesh_cube_6f.m*, maillage des 6 faces du cube (*Table 25*)

```
1   function [struc]= fmesh_cube_6f(cx,cy,cz,nx,ny,nz)
2   nel  = 0;
3   Quad = zeros(3,nx*nz*8+ny*nz*8+nx*ny*8);
4   for j=1:ny
5       for k=1:nz
6           Quad(1,nel*4+1)=0;
7           Quad(2,nel*4+1)=(j-1)*cy/ny;
8           Quad(3,nel*4+1)=(k-1)*cz/nz;
9           Quad(1,nel*4+2)=0;
10          Quad(2,nel*4+2)=j*cy/ny;
11          Quad(3,nel*4+2)=(k-1)*cz/nz;
12          Quad(1,nel*4+3)=0;
13          Quad(2,nel*4+3)=j*cy/ny;
14          Quad(3,nel*4+3)=k*cz/nz;
15          Quad(1,nel*4+4)=0;
```

```
16          Quad(2,nel*4+4)=(j-1)*cy/ny;
17          Quad(3,nel*4+4)=k*cz/nz;
18          nel=nel+1;
19      end
20  end
21  for j=1:ny
22      for k=1:nz
23          Quad(1,nel*4+1)=cx;
24          Quad(2,nel*4+1)=(j-1)*cy/ny;
25          Quad(3,nel*4+1)=(k-1)*cz/nz;
26          Quad(1,nel*4+4)=cx;
27          Quad(2,nel*4+4)=j*cy/ny;
28          Quad(3,nel*4+4)=(k-1)*cz/nz;
29          Quad(1,nel*4+3)=cx;
30          Quad(2,nel*4+3)=j*cy/ny;
31          Quad(3,nel*4+3)=k*cz/nz;
32          Quad(1,nel*4+2)=cx;
33          Quad(2,nel*4+2)=(j-1)*cy/ny;
34          Quad(3,nel*4+2)=k*cz/nz;
35          nel=nel+1;
36      end
37  end
38  for j=1:nx
39      for k=1:nz
40          Quad(1,nel*4+1)=(j-1)*cx/nx;
41          Quad(2,nel*4+1)=0;
42          Quad(3,nel*4+1)=(k-1)*cz/nz;
43          Quad(1,nel*4+4)=j*cx/nx;
44          Quad(2,nel*4+4)=0;
45          Quad(3,nel*4+4)=(k-1)*cz/nz;
46          Quad(1,nel*4+3)=j*cx/nx;
47          Quad(2,nel*4+3)=0;
48          Quad(3,nel*4+3)=k*cz/nz;
49          Quad(1,nel*4+2)=(j-1)*cx/nx;
50          Quad(2,nel*4+2)=0;
51          Quad(3,nel*4+2)=k*cz/nz;
52          nel=nel+1;
53      end
54  end
55  for j=1:nx
56      for k=1:nz
57          Quad(1,nel*4+1)=(j-1)*cx/nx;
58          Quad(2,nel*4+1)=cy;
59          Quad(3,nel*4+1)=(k-1)*cz/nz;
60          Quad(1,nel*4+2)=j*cx/nx;
61          Quad(2,nel*4+2)=cy;
62          Quad(3,nel*4+2)=(k-1)*cz/nz;
63          Quad(1,nel*4+3)=j*cx/nx;
64          Quad(2,nel*4+3)=cy;
65          Quad(3,nel*4+3)=k*cz/nz;
66          Quad(1,nel*4+4)=(j-1)*cx/nx;
67          Quad(2,nel*4+4)=cy;
68          Quad(3,nel*4+4)=k*cz/nz;
69          nel=nel+1;
70      end
71  end
72  for j= 1:nx
73      for k=1:ny
74          Quad(1,nel*4+1)=(j-1)*cx/nx;
75          Quad(2,nel*4+1)=(k-1)*cy/ny;
76          Quad(3,nel*4+1)= 0;
77          Quad(1,nel*4+2)=j*cx/nx;
78          Quad(2,nel*4+2)=(k-1)*cy/ny;
```

```
79          Quad(3,nel*4+2)= 0;
80          Quad(1,nel*4+3)=j*cx/nx;
81          Quad(2,nel*4+3)=k*cy/ny;
82          Quad(3,nel*4+3)= 0;
83          Quad(1,nel*4+4)=(j-1)*cx/nx;
84          Quad(2,nel*4+4)=k*cy/ny;
85          Quad(3,nel*4+4)= 0;
86          nel=nel+1;
87      end
88  end
89  for j=1:nx
90      for k=1:ny
91          Quad(1,nel*4+1)=(j-1)*cx/nx;
92          Quad(2,nel*4+1)=(k-1)*cy/ny;
93          Quad(3,nel*4+1)= cz;
94          Quad(1,nel*4+4)=j*cx/nx;
95          Quad(2,nel*4+4)=(k-1)*cy/ny;
96          Quad(3,nel*4+4)= cz;
97          Quad(1,nel*4+3)=j*cx/nx;
98          Quad(2,nel*4+3)=k*cy/ny;
99          Quad(3,nel*4+3)= cz;
100         Quad(1,nel*4+2)=(j-1)*cx/nx;
101         Quad(2,nel*4+2)=k*cy/ny;
102         Quad(3,nel*4+2)= cz;
103         nel=nel+1;
104     end
105 end
106 struc = Quad';
107 end
```

Table 37 : Fonction Matlab© fmesh_cube_6f.m (Rayocube.m)

La fonction *fvuelamb.m* fait appel à la fonction *vecprouni.m* (*Table 27*).

Fonction *fvuelamb.m* : calcul de facteurs de vue (*Rayocube.m*)

```
1   function [fvue]= fvuelamb(struc,ktri,ktrj)
2   cgi=(struc((ktri-1)*4+1,:)+struc((ktri-1)*4+2,:)+struc((ktri-1)*4+3,:)+...
3       struc((ktri-1)*4+4,:))/4;
4   c1          = struc((ktri-1)*4+2,:)-struc((ktri-1)*4+1,:);          % s2-s1
5   c2          = struc((ktri-1)*4+3,:)-struc((ktri-1)*4+1,:);          % s3-s1
6   np(:,ktri)  = vecprouni(c1,c2);          % normale unitaire du carreau ktri
7   som = zeros(3,5);nor = zeros(3,4);fvue = 0;g = zeros(1,4);
8       for i = 1:4
9           som(:,i) = (struc((ktrj-1)*4+i,:)-cgi);
10          som(:,i) = som(:,i)/sqrt(dot(som(:,i),som(:,i)));
11      end
12      som(:,5)     = som(:,1);
13      for i        = 1:4
14          j        = i+1;
15          nor(:,i) = vecprouni(som(:,i),som(:,j));
16          g(i)     = acos(dot(som(:,i),som(:,j)));
17          fvue     = fvue + 1/(2*pi)*dot(np(:,ktri),nor(:,i))*g(i);
18      end
19  end
```

Table 38 : Fonction Matlab© fvuelamb.m (Rayocube.m), calcul de facteurs de vue

Fonction *fvlr.m* : (*Rayocube.m*) calcul de *M* par lancer de rayons

```
1   function [rf]=fvlr(nr,cx,cy,cz,nel) % nr = nomb. rayons, nel = nomb. elem.
2   lalo = Bvf(Bvfmd(nr));% disp(lalo*180/pi) % Coord. sphériqus des nr rayons
3   cg   = [cx/2 cy/2  0  ;cx/2 cy/2 cz  ;0    cy/2 cz/2;... % Barycentres des
4           cx   cy/2 cz/2;cx/2  0   cz/2;cx/2 cy   cz/2];      % nel éléments
```

```
hem  = [1  2  3 ;1  2 -3 ;3  2  1;-3 -2 -1;1  3  2 ;-1 -3 -2 ];
bem  = sign(hem);
ray  = zeros(nel,3);
rf   = zeros(nel,nel);   % rf = matrice des fact. de vue, sortie de fvlr.m
disp('fvlr, nel');disp(nel)
disp('fvlr, Barycenters, cg');disp(cg)
disp('fvlr, hem');disp(hem)
% disp('fvlr, size(ray)');disp(size(ray))
% disp(rf)
for j = 1:nel %
%     disp(['fvir, hem ',num2str(hem(j,:))])
    fv    = zeros(1,nel);
    for i = 1 : nr
        d     = ones(nel,1)*100;
        rx    = sin(lalo(i,1))*cos(lalo(i,2));   % Coord. cartés. du rayon
        ry    = sin(lalo(i,1))*sin(lalo(i,2));
        rz    = cos(lalo(i,1));
        ra    = [rx ry rz];                  % Rayon en coord. xartésiennes
        ray(hem(j,1)*bem(j,1)) = ra(1)*bem(j,1);
        ray(hem(j,2)*bem(j,2)) = ra(2)*bem(j,2);
        ray(hem(j,3)*bem(j,3)) = ra(3)*bem(j,3);
% Calcul de la longueur du rayon pour atteindre les 6 faces
        t     = (  -cg(j,3))/ray(3); if t > 0.; d(1) = t;end % Face z = 0
        t     = (cz-cg(j,3))/ray(3); if t > 0.; d(2) = t;end % Face z = cz
        t     = (  -cg(j,1))/ray(1); if t > 0.; d(3) = t;end % Face x = 0
        t     = (cx-cg(j,1))/ray(1); if t > 0.; d(4) = t;end % Face x = cx
        t     = (  -cg(j,2))/ray(2); if t > 0.; d(5) = t;end % Face y = 0
        t     = (cy-cg(j,2))/ray(2); if t > 0.; d(6) = t;end % Face y = cy
        [c,k] = min(d);       % k est la face du cube atteinte par le rayon
        fv(k) = fv(k)+1;         % Comptage des intersections sur la face k
    end
    rf(j,:)   = -fv/nr;
%     rf(j,j)   = 1;
end
disp('fvlr,  rf');disp(eye(nel,nel)-rf)
disp('fvlr, ray');disp(ray)
if nel < 6;disp([c,k]);end
end
```

Table 39 : Fonction Matlab© fvlr.m (Rayocube.m), calcul de M par lancer de rayons

Fonction Matlab© *br56.m*

```
function [bbr] = br56
bbr=[ 0   0         0.5625
0         0         0.6250
0         0         0.6875
0         0         0.7500
0         0         0.8125
0         0         0.8750
0         0         0.9375
0         0         1.0000
0         0.0625    1.0000
0         0.1250    1.0000
0         0.1875    1.0000
0         0.2500    1.0000
0         0.3125    1.0000
0         0.3750    1.0000
0         0.4375    1.0000
0         0.5000    1.0000
0         0.5625    1.0000
0         0.6250    1.0000
0         0.6875    1.0000
0         0.7500    1.0000
0         0.8125    1.0000
0         0.8750    1.0000
```

```
24  0          0.9375    1.0000
25  0          1.0000    1.0000
26  0.0625     1.0000    0.9375
27  0.1250     1.0000    0.8750
28  0.1875     1.0000    0.8125
29  0.2500     1.0000    0.7500
30  0.3125     1.0000    0.6875
31  0.3750     1.0000    0.6250
32  0.4375     1.0000    0.5625
33  0.5000     1.0000    0.5000
34  0.5625     1.0000    0.4375
35  0.6250     1.0000    0.3750
36  0.6875     1.0000    0.3125
37  0.7500     1.0000    0.2500
38  0.8125     1.0000    0.1875
39  0.8750     1.0000    0.1250
40  0.9375     1.0000    0.0625
41  1.0000     1.0000         0
42  1.0000     0.9375         0
43  1.0000     0.8750         0
44  1.0000     0.8125         0
45  1.0000     0.7500         0
46  1.0000     0.6875         0
47  1.0000     0.6250         0
48  1.0000     0.5625         0
49  1.0000     0.5000         0
50  1.0000     0.4375         0
51  1.0000     0.3750         0
52  1.0000     0.3125         0
53  1.0000     0.2500         0
54  1.0000     0.1875         0
55  1.0000     0.1250         0
56  1.0000     0.0625         0
57  1.0000          0         0];
58  end
```

Table 40 : Fonction Matlab© br56.m Couleurs utilisées pour la barre de couleurs

Fonction Matlab© *clos.m* – vérification des conditions de fermeture

```
1  function [clo]= clos(M,nel,rho)
2  clov          = zeros(1,nel);
3  for i         = 1 : nel
4      clov(i)   = (1-sum(M(i,1:nel)))/rho;
5  end
6  clo           = mean(clov);
7  end
```

Table 41 : Fonction Matlab© clos.m – conditions de fermeture

Table des matières