論　考

立体角の計算アルゴリズムの作成

Making of the calculation algorithm of the solid angle

Author : 志多　友史　(Yuji Shida)
Date : 2016/10/13
Keywords : Albert Girard, 球面過剰(spherical excess), 立体角(solid angle), 幾何学(geometry), C 言語(C programming language), 数値計算(numerical calculation)

Abstract:==========

本稿では科学や工学のあらゆる分野で登場する"立体角”について、その算出方法を検討したものである。検討する算出方法は、領域を微小な三角形に分割し、その三角形の面積及び法線ベクトルと位置ベクトルの内積を足し上げる方法、球面幾何学の分野に出てくる Girard の式を用いる方法の二種類である。これら二種類の方法を基に C 言語で計算コードを作成し、実行結果の比較を行った。

In this report, I examined the calculation method about “solid angle” which came up in science and every engineering field. I will introduce 2 calculation methods in this report. The 1st is, after the division of given area to small triangles, sum of the area of small triangle and inner product of normal vector and positional vector. The 2nd is, use Girard’s formula which comes out to the field of the spherical geometry. I made a calculation code by C programming language based on these 2 methods and compared the practice result.

==========

１．序論(Introduction)

二次元平面において 2 つのベクトルのなす角度は高校数学レベルの公式を用いることによって容易に求められる。しかし三次元空間において、ある点から見た平面の作る立体角を求めるには工夫が必要である。

立体角は様々な分野で用いられており、その計算アルゴリズム（プログラム）の構築は非常に意義のあるものなので、本稿では平面（三角形）を相似な微小三角形に分割し、それらの数値積分による算出と、球面幾何学・球面過剰の理論で示される Albert Girard の式を用いた算出方法について整理した。

２．理論(Theory)

基本的な微小三角形の数値積分と Albert Girard の式の概要について説明する。なお、立体角の算出のために与えられるものは、3 つのベクトル **a**,**b**,**c** の合計 9 つの成分のみである。

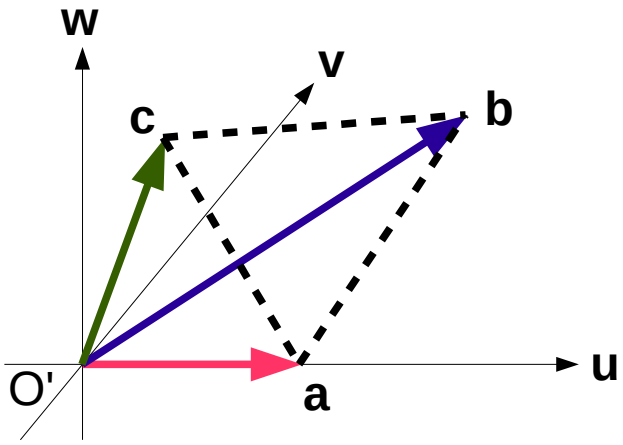

図２．１　局所座標系

図２．１中の局所座標系の軸 **u**,**v**,**w** は下記の通り定義される。

$$\boldsymbol{u}=\frac{\boldsymbol{a}}{|\boldsymbol{a}|},\boldsymbol{v}=\left(\frac{\boldsymbol{a}\times\boldsymbol{b}}{|\boldsymbol{a}\times\boldsymbol{b}|}\right)\times\frac{\boldsymbol{a}}{|\boldsymbol{a}|}=\boldsymbol{w}\times\boldsymbol{u},\boldsymbol{w}=\frac{\boldsymbol{a}\times\boldsymbol{b}}{|\boldsymbol{a}\times\boldsymbol{b}|}$$

この局所座標系はどちらの方法でも用いるので事前に定義しておく。

２．１．微小三角形の数値積分

まず立体角の基本的な定義を示す。

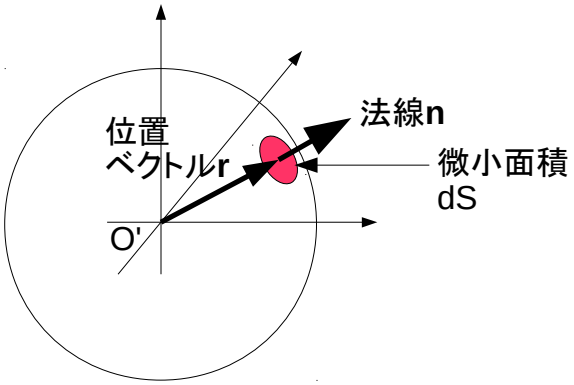

図２．１．１　微小立体角の定義

上図のように球面上の微小面積 dS を考え、O'から dS を見たときの立体角を dΩ とおくと、この間には $|\boldsymbol{r}|^2 d\Omega=\frac{\boldsymbol{r}\cdot\boldsymbol{n}}{|\boldsymbol{r}|}dS$ の関係がある。この単純なモデルならば位置ベクトルを規格化したものと法線ベクトルの内積について明記する必要は無いが、今後のためこのように表した。この式を整理すると下式が得

られる。

$$d\Omega=\frac{\boldsymbol{r}\cdot\boldsymbol{n}\,dS}{|\boldsymbol{r}|^3}$$

半径 R の球を考えた場合、球の中心を原点として、そこから球面全体を見たときの立体角を求めると次のようになる。

$$\Omega=\int_{surface} d\Omega=\int_{surface}\frac{\boldsymbol{r}\cdot\boldsymbol{n}\,dS}{|\boldsymbol{r}|^3}=\int_{surface}\frac{1}{R^2}\frac{\boldsymbol{R}\cdot\boldsymbol{n}}{R}dS=\frac{1}{R^2}\int_{surface}dS=\frac{1}{R^2}4\pi R^2=4\pi$$

ここでは法線ベクトルの存在が生かされていないが、次に示す微小三角形の数値積分を行う場合には重要になってくる。

この立体角の定義を 3 つのベクトルの終点で作られる三角形 abc に応用する。図２．１に示した三角形の各辺を N 分割し、作られた微小三角形に対して次図のように添字を定める。

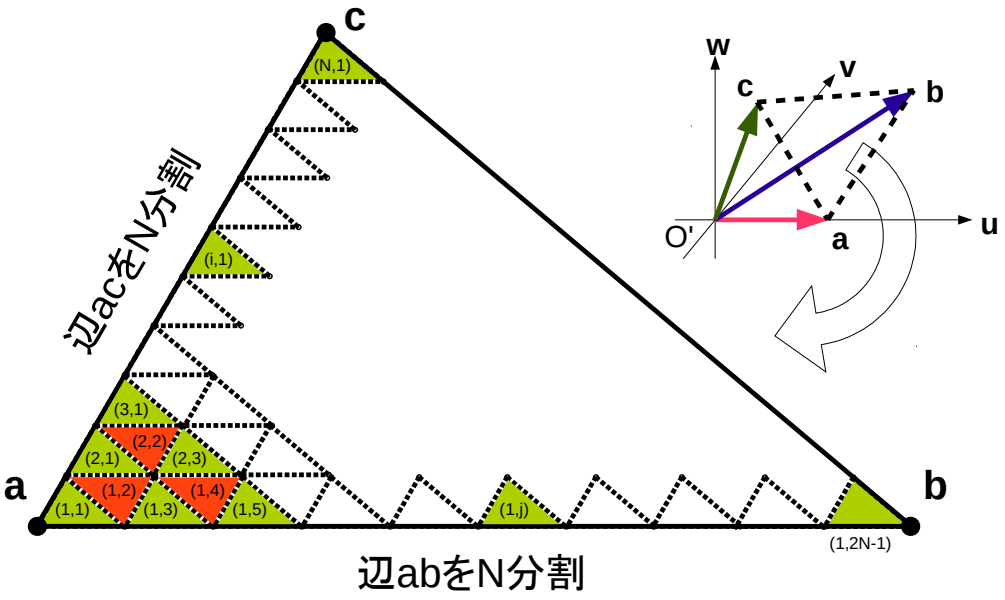

図２．１．２　三角形の分割と添字の定義

今後、簡単のため辺 ac 方向の分割を行、辺 ab 方向の分割を列と呼ぶことにする。作られる微小三角形は元の三角形 abc と相似であり、その総数は N^2 である。

$N_r(i)=2N+1-2i$　：i 行目の微小三角形の数

$$\sum_{i=1}^{N}N_r(i)=\sum_{i=1}^{N}(2N+1-2i)=(2N+1)N-2\sum_{i=1}^{N}i=2N^2+N-2\frac{N(N+1)}{2}=N^2$$

数値積分による立体角の算出方法を式で表すと次式のようになる。

$$\Omega=\sum_{i=1}^{N}\left\{\sum_{j=1}^{N_r(i)}\frac{\boldsymbol{r}_G(i,j)\cdot\boldsymbol{n}(i,j)}{|\boldsymbol{r}_G(i,j)|^3}dS(i,j)\right\}$$

式中の $\boldsymbol{r}_G(i,j)$ は i 行 j 列目の微小三角形の重心位置、 $\boldsymbol{n}(i,j)$ は同三角形の法線ベクトル（全ての微小三角形で等しい）、 $dS(i,j)$ は微小三角形の面積（全ての微小三角形で等しい）を表す。法線ベクトルと面積は行番号列番号(i,j)に依存しないので、式を整理すると次式のようになる。

$$\Omega=dS(i,j)\boldsymbol{n}(i,j)\cdot\sum_{i=1}^{N}\left\{\sum_{j=1}^{N_r(i)}\frac{\boldsymbol{r}_G(i,j)}{|\boldsymbol{r}_G(i,j)|^3}\right\}=dS\,\boldsymbol{n}\cdot\sum_{i=1}^{N}\left\{\sum_{j=1}^{N_r(i)}\frac{\boldsymbol{r}_G(i,j)}{|\boldsymbol{r}_G(i,j)|^3}\right\}$$

これより法線ベクトルの内積と微小面積の積は和算の後に行うことができ、数値積分における計算量を軽減できることが分かる。（四則演算において乗算は除算の次に演算負荷が大きい）また法線ベクトルと微小面積は次のように表されるので

$$\boldsymbol{n}=\frac{(\boldsymbol{b}-\boldsymbol{a})\times(\boldsymbol{c}-\boldsymbol{a})}{|(\boldsymbol{b}-\boldsymbol{a})\times(\boldsymbol{c}-\boldsymbol{a})|}$$

$$dS=\frac{1}{2}\frac{|\boldsymbol{b}-\boldsymbol{a}|}{N}\frac{|\boldsymbol{c}-\boldsymbol{a}|}{N}\sin a=\frac{1}{2N^2}|(\boldsymbol{b}-\boldsymbol{a})\times(\boldsymbol{c}-\boldsymbol{a})|$$

さらに式を整理すると次のようになる。

$$\Omega=\frac{1}{2N^2}(\boldsymbol{b}-\boldsymbol{a})\times(\boldsymbol{c}-\boldsymbol{a})\cdot\sum_{i=1}^{N}\left\{\sum_{j=1}^{N_r(i)}\frac{\boldsymbol{r}_G(i,j)}{|\boldsymbol{r}_G(i,j)|^3}\right\}$$

次に $\boldsymbol{r}_G(i,j)$ について列番号 j が奇数の場合と偶数の場合とで分けて考える。

（１）列番号 j が奇数の場合

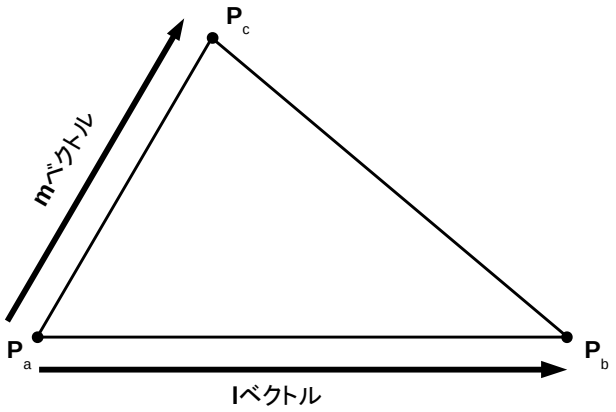

図２．１．３　微小三角形（正）の重心

列番号が奇数の場合、三角形は元の三角形 abc と同じ向きである。また上図中のベクトル **l** 及び **m** を次のように定義する。

$$\boldsymbol{l}=\frac{(\boldsymbol{b}-\boldsymbol{a})}{N},\boldsymbol{m}=\frac{(\boldsymbol{c}-\boldsymbol{a})}{N}$$

これよりベクトル $\mathbf{P}_a(i,j)$が与えられれば、$\mathbf{P}_b(i,j)$、$\mathbf{P}_c(i,j)$は $\boldsymbol{P}_b=\boldsymbol{P}_a+\boldsymbol{l},\boldsymbol{P}_c=\boldsymbol{P}_a+\boldsymbol{m}$ となり、この三角形の重心は

$$\boldsymbol{r}_G(i,j)=\frac{1}{3}(\boldsymbol{P}_a+\boldsymbol{P}_b+\boldsymbol{P}_c)=\frac{1}{3}(\boldsymbol{P}_a+\boldsymbol{P}_a+\boldsymbol{l}+\boldsymbol{P}_a+\boldsymbol{m})=\frac{1}{3}(3\boldsymbol{P}_a+\boldsymbol{l}+\boldsymbol{m})=\boldsymbol{P}_a+\frac{1}{3}(\boldsymbol{l}+\boldsymbol{m})$$

と表される。なお式中の $\mathbf{P}_a(i,j)$は

$$\boldsymbol{P}_a(i,j)=\boldsymbol{a}+\left(\frac{j+1}{2}-1\right)\boldsymbol{l}+(i-1)\boldsymbol{m}$$　：j は奇数

と表される。

（２）列番号 j が偶数の場合

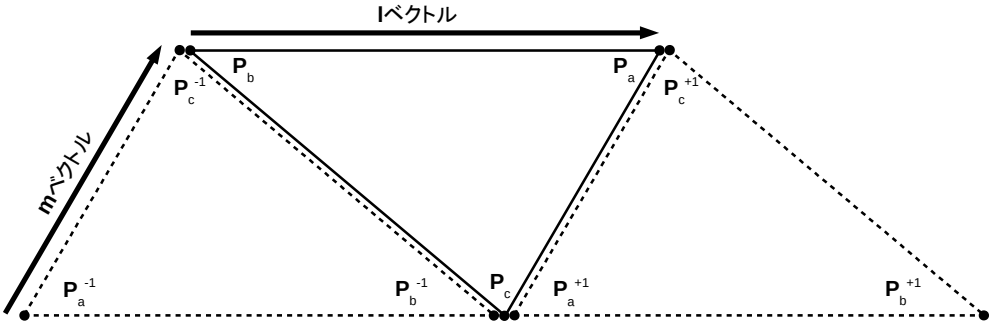

図２．１．４　微小三角形（逆）の重心

列番号が偶数の場合、三角形は元の三角形 abc を 180°回転させた形となる。この三角形の重心も先と同様に $\boldsymbol{r}_G(i,j)=\frac{1}{3}(\boldsymbol{P}_a+\boldsymbol{P}_b+\boldsymbol{P}_c)$ となるが、$\mathbf{P}_b$、$\mathbf{P}_c$は $\boldsymbol{P}_b=\boldsymbol{P}_a-\boldsymbol{l},\boldsymbol{P}_c=\boldsymbol{P}_a-\boldsymbol{m}$ となるため、この三角形の重心は

$$\boldsymbol{r}_G(i,j)=\frac{1}{3}(\boldsymbol{P}_a+\boldsymbol{P}_b+\boldsymbol{P}_c)=\boldsymbol{P}_a-\frac{1}{3}(\boldsymbol{l}+\boldsymbol{m})$$

と表され、また $\mathbf{P}_a(i,j)$ は隣り合う三角形（列番号が奇数）を用いて

$$\boldsymbol{P}_a(i,j)=\boldsymbol{P}_c(i,j-1)+\boldsymbol{l}=\boldsymbol{P}_a(i,j-1)+\boldsymbol{l}+\boldsymbol{m}$$

と表せるので

$$\boldsymbol{P}_a(i,j)=\boldsymbol{a}+\frac{j}{2}\boldsymbol{l}+i\,\boldsymbol{m}$$ ：j は偶数

となる。

以上(1),(2)をまとめると、微小三角形の重心の位置ベクトルは

$$\boldsymbol{r}_G(i,j_{odd})=\boldsymbol{a}+\left(\frac{j+1}{2}-\frac{2}{3}\right)\boldsymbol{l}+\left(i-\frac{2}{3}\right)\boldsymbol{m}$$

$$\boldsymbol{r}_G(i,j_{even})=\boldsymbol{a}+\left(\frac{j}{2}-\frac{1}{3}\right)\boldsymbol{l}+\left(i-\frac{1}{3}\right)\boldsymbol{m}$$

という形で表される。

従って、立体角 Ω は下記の 4 つの式を計算することによって求められる。

$$\Omega=\frac{1}{2N^2}(\boldsymbol{b}-\boldsymbol{a})\times(\boldsymbol{c}-\boldsymbol{a})\cdot\sum_{i=1}^{N}\left\{\sum_{j=1}^{N_r(i)}\frac{\boldsymbol{r}_G(i,j)}{|\boldsymbol{r}_G(i,j)|^3}\right\}$$ ：立体角の計算式

$$N_r(i)=2N+1-2i$$ ：i 行目の微小三角形の数

$$\boldsymbol{r}_G(i,j_{odd})=\boldsymbol{a}+\left(\frac{j+1}{2}-\frac{2}{3}\right)\boldsymbol{l}+\left(i-\frac{2}{3}\right)\boldsymbol{m}$$ ：奇数列目の微小三角形の位置ベクトル

$$\boldsymbol{r}_G(i,j_{even})=\boldsymbol{a}+\left(\frac{j}{2}-\frac{1}{3}\right)\boldsymbol{l}+\left(i-\frac{1}{3}\right)\boldsymbol{m}$$ ：偶数列目の微小三角形の位置ベクトル

２．２．Girard の定理の適用

球面過剰の理論は球面幾何学の分野に属するもので、本稿で求めたい立体角（球の表面積）が Albert Girard の式を用いる事によって容易に得られるというものである。

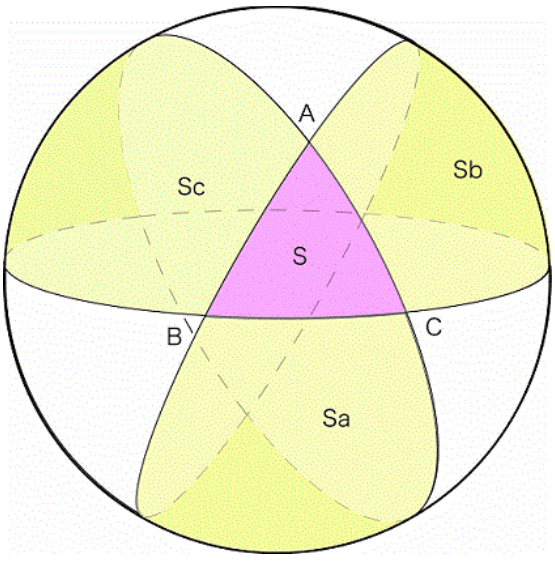

図２．２．１　球面過剰の理論
(http://www.irf.se/~futaana/Kiruna/50_Kyumen.html より引用)

本稿ではこの理論の詳細は割愛するが、上図のような半径 R の球面上に点 A,B,C が存在するとして、球面三角形の内角（平面三角形 ABC ではないことに注意）をそれぞれ α,β,γ とおくと球面上の面積 S は $S=R^2(\alpha+\beta+\gamma-\pi)$ で表されるというものである。この球の半径 R を 1 とすると S は立体角と同値

になる。

本節ではこの式を用いて、立体角の算出法について記す。なお、本節においても前節と同様に 3 つのベクトルが与えられているものとし、局所座標系におけるベクトル演算を行う。

まず、本方法で立体角を求めるには下記の手順を踏まなければならない。

（1）与えられた 3 つのベクトルを規格化し、各ベクトルの終点を単位球面上に設定する。
（2）各点における接平面を設定する。
（3）各点における接平面に 3 つのベクトルの終点が作る三角形の辺ベクトルを射影する。
（4）射影されたベクトルの内積より球面三角形の内角を算出する。
（5）得られた球面三角形の内角より前述の式を用いて、立体角を求める。

上記の短文だけでは、具体的に何を行うのか不明瞭なので、それぞれの内容を次に記す。

（1）与えられた 3 つのベクトルを規格化し、各ベクトルの終点を単位球面上に設定する。

3 つのベクトル **a**,**b**,**c** を規格化したベクトルを次のように定義する。

$$\hat{\boldsymbol{a}}=\frac{\boldsymbol{a}}{|\boldsymbol{a}|},\hat{\boldsymbol{b}}=\frac{\boldsymbol{b}}{|\boldsymbol{b}|},\hat{\boldsymbol{c}}=\frac{\boldsymbol{c}}{|\boldsymbol{c}|}$$

これらのベクトルの大きさは 1 なので、各ベクトルの終点は半径 1 の単位球上に存在することになる。

（2）各点における接平面を設定する。

局所座標系は直交ベクトル **u**,**v**,**w** で定められ、次のように定義される。

$$\boldsymbol{u}=\hat{\boldsymbol{a}},\boldsymbol{v}=\left(\frac{\hat{\boldsymbol{a}}\times\hat{\boldsymbol{b}}}{|\hat{\boldsymbol{a}}\times\hat{\boldsymbol{b}}|}\right)\times\hat{\boldsymbol{a}}=\boldsymbol{w}\times\boldsymbol{u},\boldsymbol{w}=\frac{\hat{\boldsymbol{a}}\times\hat{\boldsymbol{b}}}{|\hat{\boldsymbol{a}}\times\hat{\boldsymbol{b}}|}$$

これらの直交ベクトルと規格化されたベクトル $\hat{\boldsymbol{a}},\hat{\boldsymbol{b}},\hat{\boldsymbol{c}}$ を用いると各点における接平面を構成する 2 つの直交ベクトルは次のように求められる。

$$\boldsymbol{p}_{\hat{a}}=\boldsymbol{v},\boldsymbol{q}_{\hat{a}}=\boldsymbol{w}$$
$$\boldsymbol{p}_{\hat{b}}=\boldsymbol{w},\boldsymbol{q}_{\hat{b}}=\hat{\boldsymbol{b}}\times\boldsymbol{w}$$
$$\boldsymbol{p}_{\hat{c}}=\frac{\hat{\boldsymbol{c}}\times\boldsymbol{w}}{|\hat{\boldsymbol{c}}\times\boldsymbol{w}|},\boldsymbol{q}_{\hat{c}}=\hat{\boldsymbol{c}}\times\boldsymbol{p}_{\hat{c}}$$

これらを図示すると次のようになる。

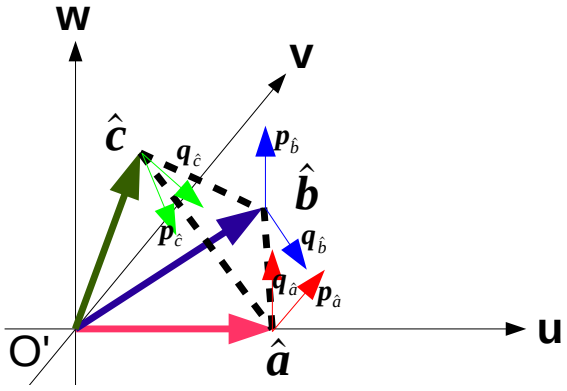

図２．２．２　局所座標系

（3）各点における接平面に 3 つのベクトルの終点が作る三角形の辺ベクトルを射影する。

まず、次図に示すような系を考える。

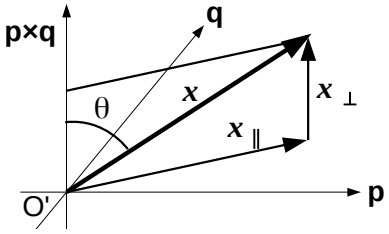

図２．２．３　接平面への射影

ここでは、ベクトル $\mathbf{p},\mathbf{q}$ が作る平面に含まれるベクトル $\boldsymbol{x}_{\parallel}$ を求めたい。このベクトル $\boldsymbol{x}_{\parallel}$ は $\boldsymbol{x}_{\parallel}=\boldsymbol{x}-\boldsymbol{x}_{\perp}$ と表されるので、平面に垂直なベクトル $\boldsymbol{x}_{\perp}$ を求めれば、平面上のベクトル $\boldsymbol{x}_{\parallel}$ も求められる。図より、

$$\boldsymbol{x}_{\perp}=|\boldsymbol{x}|\cos\theta\,\boldsymbol{p}\times\boldsymbol{q}=|\boldsymbol{p}\times\boldsymbol{q}||\boldsymbol{x}|\cos\theta\,\boldsymbol{p}\times\boldsymbol{q}=\{(\boldsymbol{p}\times\boldsymbol{q})\cdot\boldsymbol{x}\}\boldsymbol{p}\times\boldsymbol{q}$$

であるから、平面上のベクトルは

$$\boldsymbol{x}_{\parallel}=\boldsymbol{x}-\boldsymbol{x}_{\perp}=\boldsymbol{x}-\{(\boldsymbol{p}\times\boldsymbol{q})\cdot\boldsymbol{x}\}\boldsymbol{p}\times\boldsymbol{q}$$

と求められる。

この演算を単位球面上の各点に適用する場合、各ベクトルは次のように定義される。

- $\hat{\boldsymbol{a}}$ における各ベクトル

$$\boldsymbol{p}=\boldsymbol{p}_{\hat{a}}=\boldsymbol{v},\boldsymbol{q}=\boldsymbol{q}_{\hat{a}}=\boldsymbol{w},\boldsymbol{p}\times\boldsymbol{q}=\hat{\boldsymbol{a}}$$
$$\boldsymbol{x}_{\hat{a}1}=\hat{\boldsymbol{b}}-\hat{\boldsymbol{a}},\boldsymbol{x}_{\hat{a}2}=\hat{\boldsymbol{c}}-\hat{\boldsymbol{a}}$$

- $\hat{\boldsymbol{b}}$ における各ベクトル

$$\boldsymbol{p}=\boldsymbol{p}_{\hat{b}}=\boldsymbol{w},\boldsymbol{q}=\boldsymbol{q}_{\hat{b}}=\hat{\boldsymbol{b}}\times\boldsymbol{w},\boldsymbol{p}\times\boldsymbol{q}=\hat{\boldsymbol{b}}$$
$$\boldsymbol{x}_{\hat{b}1}=\hat{\boldsymbol{c}}-\hat{\boldsymbol{b}},\boldsymbol{x}_{\hat{b}2}=\hat{\boldsymbol{a}}-\hat{\boldsymbol{b}}$$

- $\hat{\boldsymbol{c}}$ における各ベクトル

$$\boldsymbol{p}=\boldsymbol{p}_{\hat{c}}=\frac{\hat{\boldsymbol{c}}\times\boldsymbol{w}}{|\hat{\boldsymbol{c}}\times\boldsymbol{w}|},\boldsymbol{q}=\boldsymbol{q}_{\hat{c}}=\hat{\boldsymbol{c}}\times\boldsymbol{p}_{\hat{c}},\boldsymbol{p}\times\boldsymbol{q}=\hat{\boldsymbol{c}}$$
$$\boldsymbol{x}_{\hat{c}1}=\hat{\boldsymbol{a}}-\hat{\boldsymbol{c}},\boldsymbol{x}_{\hat{c}2}=\hat{\boldsymbol{b}}-\hat{\boldsymbol{c}}$$

また、各系で求めたいベクトルは下記のように表される。

- $\hat{\boldsymbol{a}}$ における平面内のベクトル

$$\boldsymbol{x}_{\parallel\hat{a}1}=\hat{\boldsymbol{b}}-(\hat{\boldsymbol{a}}\cdot\hat{\boldsymbol{b}})\hat{\boldsymbol{a}},\boldsymbol{x}_{\parallel\hat{a}2}=\hat{\boldsymbol{c}}-(\hat{\boldsymbol{c}}\cdot\hat{\boldsymbol{a}})\hat{\boldsymbol{a}}$$

- $\hat{\boldsymbol{b}}$ における平面内のベクトル

$$\boldsymbol{x}_{\parallel\hat{b}1}=\hat{\boldsymbol{c}}-(\hat{\boldsymbol{b}}\cdot\hat{\boldsymbol{c}})\hat{\boldsymbol{b}},\boldsymbol{x}_{\parallel\hat{b}2}=\hat{\boldsymbol{a}}-(\hat{\boldsymbol{a}}\cdot\hat{\boldsymbol{b}})\hat{\boldsymbol{b}}$$

- $\hat{\boldsymbol{c}}$ における平面内のベクトル

$$\boldsymbol{x}_{\parallel\hat{c}1}=\hat{\boldsymbol{a}}-(\hat{\boldsymbol{c}}\cdot\hat{\boldsymbol{a}})\hat{\boldsymbol{c}},\boldsymbol{x}_{\parallel\hat{c}2}=\hat{\boldsymbol{b}}-(\hat{\boldsymbol{b}}\cdot\hat{\boldsymbol{c}})\hat{\boldsymbol{c}}$$

（4）射影されたベクトルの内積より球面三角形の内角を算出する。

（3）の演算によって各接平面に射影されたベクトルが 2 本ずつ求められるので、それぞれの内積から球面三角形の各内角を求める。各内角を $\hat{\boldsymbol{a}}\sim\hat{\boldsymbol{c}}$ の順に α,β,γ とおくと、これらの角度は下記のように表される。

$$\alpha=\cos^{-1}\left(\frac{\boldsymbol{x}_{\parallel\hat{a}1}\cdot\boldsymbol{x}_{\parallel\hat{a}2}}{|\boldsymbol{x}_{\parallel\hat{a}1}||\boldsymbol{x}_{\parallel\hat{a}2}|}\right)=\cos^{-1}\left(\frac{\hat{\boldsymbol{b}}\cdot\hat{\boldsymbol{c}}-(\hat{\boldsymbol{c}}\cdot\hat{\boldsymbol{a}})(\hat{\boldsymbol{a}}\cdot\hat{\boldsymbol{b}})}{|\hat{\boldsymbol{b}}-(\hat{\boldsymbol{a}}\cdot\hat{\boldsymbol{b}})\hat{\boldsymbol{a}}||\hat{\boldsymbol{c}}-(\hat{\boldsymbol{c}}\cdot\hat{\boldsymbol{a}})\hat{\boldsymbol{a}}|}\right)$$
$$\beta=\cos^{-1}\left(\frac{\boldsymbol{x}_{\parallel\hat{b}1}\cdot\boldsymbol{x}_{\parallel\hat{b}2}}{|\boldsymbol{x}_{\parallel\hat{b}1}||\boldsymbol{x}_{\parallel\hat{b}2}|}\right)=\cos^{-1}\left(\frac{\hat{\boldsymbol{c}}\cdot\hat{\boldsymbol{a}}-(\hat{\boldsymbol{a}}\cdot\hat{\boldsymbol{b}})(\hat{\boldsymbol{b}}\cdot\hat{\boldsymbol{c}})}{|\hat{\boldsymbol{c}}-(\hat{\boldsymbol{b}}\cdot\hat{\boldsymbol{c}})\hat{\boldsymbol{b}}||\hat{\boldsymbol{a}}-(\hat{\boldsymbol{a}}\cdot\hat{\boldsymbol{b}})\hat{\boldsymbol{b}}|}\right)$$
$$\gamma=\cos^{-1}\left(\frac{\boldsymbol{x}_{\parallel\hat{c}1}\cdot\boldsymbol{x}_{\parallel\hat{c}2}}{|\boldsymbol{x}_{\parallel\hat{c}1}||\boldsymbol{x}_{\parallel\hat{c}2}|}\right)=\cos^{-1}\left(\frac{\hat{\boldsymbol{a}}\cdot\hat{\boldsymbol{b}}-(\hat{\boldsymbol{b}}\cdot\hat{\boldsymbol{c}})(\hat{\boldsymbol{c}}\cdot\hat{\boldsymbol{a}})}{|\hat{\boldsymbol{a}}-(\hat{\boldsymbol{c}}\cdot\hat{\boldsymbol{a}})\hat{\boldsymbol{c}}||\hat{\boldsymbol{b}}-(\hat{\boldsymbol{b}}\cdot\hat{\boldsymbol{c}})\hat{\boldsymbol{c}}|}\right)$$

（5）得られた球面三角形の内角より前述の式を用いて、立体角を求める。

（4）で得られた内角 α,β,γ を始めに示した式 $S=R^2(\alpha+\beta+\gamma-\pi)$ に代入すると、立体角 Ω は $\Omega=\alpha+\beta+\gamma-\pi$ という形で求められる。

３．方法(Method)

前章で示した理論をC言語のプログラムで具体的に作り上げる。

３．１．微小三角形の数値積分

２．１の最後に示した4本の式を用いて計算すれば立体角を求めることは可能であるが、計算を行う上では若干不便である。ここで、i行目に含まれる正方向の微小三角形と逆方向の微小三角形の個数をそれぞれ $N_{ro}(i),N_{re}(i)$ とおくと、

$$N_{ro}(i)=\frac{N_r(i)+1}{2}=N-i+1$$

$$N_{re}(i)=\frac{N_r(i)+1}{2}-1=N-i$$

と表され、 $1\le j'_{odd}\le N_{ro}(i),0\le j'_{even}\le N_{re}(i)$ として微小三角形の重心の位置ベクトルをそれぞれ連番になるように書き換えると

$$\boldsymbol{r}_G(i,j'_{odd})=\boldsymbol{a}+\left(\frac{(2j'_{odd}-1)+1}{2}-\frac{2}{3}\right)\boldsymbol{l}+\left(i-\frac{2}{3}\right)\boldsymbol{m}=\boldsymbol{a}+\left(j'_{odd}-\frac{2}{3}\right)\boldsymbol{l}+\left(i-\frac{2}{3}\right)\boldsymbol{m}$$

$$\boldsymbol{r}_G(i,j'_{even})=\boldsymbol{a}+\left(\frac{2j'_{even}}{2}-\frac{1}{3}\right)\boldsymbol{l}+\left(i-\frac{1}{3}\right)\boldsymbol{m}=\boldsymbol{a}+\left(j'_{even}-\frac{1}{3}\right)\boldsymbol{l}+\left(i-\frac{1}{3}\right)\boldsymbol{m}$$

となる。従って立体角の式は次のようにまとめられる。

$$\Omega=\frac{1}{2N^2}(\boldsymbol{b}-\boldsymbol{a})\times(\boldsymbol{c}-\boldsymbol{a})\cdot\sum_{i=1}^{N}\left\{\sum_{j'_{odd}=1}^{N_{ro}(i)}\frac{\boldsymbol{r}_G(i,j'_{odd})}{\left|\boldsymbol{r}_G(i,j'_{odd})\right|^3}+\sum_{j'_{even}=1}^{N_{re}(i)}\frac{\boldsymbol{r}_G(i,j'_{even})}{\left|\boldsymbol{r}_G(i,j'_{even})\right|^3}\right\}$$

これまでの計算式をC言語のコード（関数）で表すと次のようになる。

この関数は3つのベクトルの成分 ax~az,bx~bz,cx~cz を受け取り、計算結果を返却するというものである。関数名は SoST とし、Sum of Small Triangles の頭文字を取ったものである。

```
double SoST(double ax,double ay,double az,double bx,double by,double bz,
                        double cx,double cy,double cz)
{
    int i,j;                                                    ※for 文用変数
    int nop;                                                    ※三角形の一辺の分割数
    int nro,nre;                                                ※for 文用変数
    double lx,ly,lz,mx,my,mz;                                   ※l,m ベクトルの成分
    double opx,opy,opz;                                         ※三角形の法線ベクトルの成分
    double buf1,buf2,buf3,buf4,buf5,buf6,buf7,buf8,buf9;        ※計算用バッファ変数
    double ans;                                                 ※計算結果用変数

    nop=1000;                                                   ※分割数の設定

    lx=(bx-ax)/(double)nop;                                     ※l ベクトルの計算
    ly=(by-ay)/(double)nop;
    lz=(bz-az)/(double)nop;
    mx=(cx-ax)/(double)nop;                                     ※m ベクトルの計算
    my=(cy-ay)/(double)nop;
    mz=(cz-az)/(double)nop;

    opx=(by-ay)*(cz-az)-(bz-az)*(cy-ay);                        ※法線ベクトル（外積）の計算
    opy=(bz-az)*(cx-ax)-(bx-ax)*(cz-az);
    opz=(bx-ax)*(cy-ay)-(by-ay)*(cx-ax);
```

```
	buf7=0.0e0;                                          ※buf7~9 に各計算結果を足し上げる
	buf8=0.0e0;
	buf9=0.0e0;

	for(i=1;i<=nop;i++){
		nro=nop-i+1;                                 ※i 行目の正方向の三角形の数
		for(j=1;j<=nro;j++){
			buf1=(double)j-2.0e0/3.0e0;
			buf2=(double)i-2.0e0/3.0e0;
			buf3=ax+buf1*lx+buf2*mx;
			buf4=ay+buf1*ly+buf2*my;
			buf5=az+buf1*lz+buf2*mz;
			buf6=sqrtl(buf3*buf3+buf4*buf4+buf5*buf5);
			buf6=buf6*buf6*buf6;
			buf7+=buf3/buf6;
			buf8+=buf4/buf6;
			buf9+=buf5/buf6;
		}

		nre=nop-i;                                   ※i 行目の逆方向の三角形の数
		for(j=1;j<=nre;j++){
			buf1=(double)j-1.0e0/3.0e0;
			buf2=(double)i-1.0e0/3.0e0;
			buf3=ax+buf1*lx+buf2*mx;
			buf4=ay+buf1*ly+buf2*my;
			buf5=az+buf1*lz+buf2*mz;
			buf6=sqrtl(buf3*buf3+buf4*buf4+buf5*buf5);
			buf6=buf6*buf6*buf6;
			buf7+=buf3/buf6;
			buf8+=buf4/buf6;
			buf9+=buf5/buf6;
		}
		if(i%100==0){printf("\nSoST=%d\n",i);}       ※現在の状況を表示
	}
	buf1=opx*buf7+opy*buf8+opz*buf9;                     ※法線ベクトルとの内積計算
	buf2=5.0e-1/((double)nop*nop);                       ※1/2N^2

	ans=buf1*buf2;                                       ※立体角の計算結果
	return ans;                                          ※値の返却
}
```

３．２．Girard の定理による計算

２．２の最後に示した 3 本の式を計算すれば容易に立体角を求めることができる。これをコードで表すと次のようになる。関数名は GSEF とし、Girard’s Spherical Excess Formula の頭文字を取ったものである。値の入出力に関しては先の SoST と同様である。

```
double GSEF(double ax,double ay,double az,double bx,double by,double bz,
		double cx,double cy,double cz)
{
	double nax,nay,naz,nbx,nby,nbz,ncx,ncy,ncz;          ※各規格化ベクトルの成分
	double iab,ibc,ica;                                  ※各規格化ベクトルの内積計算結果
	double alpha,beta,gamma,ans;                         ※各角度
	double buf1,buf2,buf3,buf4,buf5,buf6,buf17,buf18,buf19;  ※計算用バッファ変数

	buf17=sqrtl(ax*ax+ay*ay+az*az);                      ※各ベクトルの大きさ
	buf18=sqrtl(bx*bx+by*by+bz*bz);
```

```
    buf19=sqrtl(cx*cx+cy*cy+cz*cz);
    if(buf17==0.0e0) return -1.0e0;                    ※エラーチェック
    if(buf18==0.0e0) return -1.0e0;
    if(buf19==0.0e0) return -1.0e0;
    nax=ax/buf17;                                      ※a ベクトルの規格化
    nay=ay/buf17;
    naz=az/buf17;
    nbx=bx/buf18;                                      ※b ベクトルの規格化
    nby=by/buf18;
    nbz=bz/buf18;
    ncx=cx/buf19;                                      ※c ベクトルの規格化
    ncy=cy/buf19;
    ncz=cz/buf19;

    iab=nax*nbx+nay*nby+naz*nbz;                       ※内積計算
    ibc=nbx*ncx+nby*ncy+nbz*ncz;
    ica=ncx*nax+ncy*nay+ncz*naz;

    buf17=nbx-iab*nax;
    buf18=nby-iab*nay;
    buf19=nbz-iab*naz;
    buf1=sqrtl(buf17*buf17+buf18*buf18+buf19*buf19);
    buf17=ncx-ica*nax;
    buf18=ncy-ica*nay;
    buf19=ncz-ica*naz;
    buf2=sqrtl(buf17*buf17+buf18*buf18+buf19*buf19);

    buf17=ncx-ibc*nbx;
    buf18=ncy-ibc*nby;
    buf19=ncz-ibc*nbz;
    buf3=sqrtl(buf17*buf17+buf18*buf18+buf19*buf19);
    buf17=nax-iab*nbx;
    buf18=nay-iab*nby;
    buf19=naz-iab*nbz;
    buf4=sqrtl(buf17*buf17+buf18*buf18+buf19*buf19);

    buf17=nax-ica*ncx;
    buf18=nay-ica*ncy;
    buf19=naz-ica*ncz;
    buf5=sqrtl(buf17*buf17+buf18*buf18+buf19*buf19);
    buf17=nbx-ibc*ncx;
    buf18=nby-ibc*ncy;
    buf19=nbz-ibc*ncz;
    buf6=sqrtl(buf17*buf17+buf18*buf18+buf19*buf19);

    buf17=ibc-ica*iab;
    alpha=acos(buf17/(buf1*buf2));
    buf18=ica-iab*ibc;
    beta=acos(buf18/(buf3*buf4));
    buf19=iab-ibc*ica;
    gamma=acos(buf19/(buf5*buf6));

    ans=alpha+beta+gamma-M_PI;                         ※M_PI はプログラムの始めで定義
                                                         計算環境に依存するので注意
    return ans;                                        ※値の返却
}
```

４．結果(Results)

前章で作成したプログラムの実行結果を示す。なお計算環境は次の通りである。

CPU：Intel Core i5-4570 3.20GHz

MEM：DDR3 Dual Channel 32GB(8GB*4) 1333MHz

OS：Windows8 64bit

円周率の定義　M_PI=3.14159265358979323846

（１）**a**=(5,0,0),**b**=(0,5,0),**c**=(0,0,5)の場合

GSEF 解は理論解(π/2)と一致しているが、SoST 解は分割数(nop)を十分大きく取らなければ精度を上げられないという事が分かる。また SoST による演算時間はおおよそ分割数の二乗に比例するので、精度・演算負荷共に GSEF の方が優れているといえる。

表４．１　理論解と GSEF 解（１）

理論解	1.570796326794897
GSEF 解	1.570796326794897

表４．２　SoST 解（１）

分割数(nop)	SoST 解	計算時間(s)
1	2.598076211353317	<1
5	1.587640984001313	<1
10	1.574974500663912	<1
50	1.570963012118596	<1
100	1.570837994628075	<1
500	1.570797993463444	<1
1000	1.570796743461589	<1
5000	1.570796343461577	5
10000	1.570796330961223	15
15000	1.570796328647082	35
20000	1.570796327837590	54
50000	1.570796326963002	338

（２）**a**=(1,0,0),**b**=(0,2,0),**c**=(0,0,3)の場合

この条件でも理論解は（１）の場合と同じになるはずであり、実際 GSEF 解は理論解と一致しているが、SoST 解は（１）の時よりも精度が低下している事が分かる。

表４．３　理論解と GSEF 解・SoST 解（２）

理論解	1.570796326794897
GSEF 解	1.570796326794897
SoST 解 (nop=50000)	1.570796327053607 (1.570796326963002)

※括弧内は（１）での解

（３）**a**=(1,0,0),**b**=(1,1,0),**c**=(0,0,1)の場合

この条件での理論解は π/4 となるはずであり、実際 GSEF 解は理論解と一致している。SoST 解についても上位の数桁については一致している。

表４．４　理論解と GSEF 解・SoST 解（３）

理論解	$7.853981633974483\times10^{-1}$
GSEF 解	$7.853981633974483\times10^{-1}$
SoST 解 (nop=50000)	$7.8539816343724 02\times10^{-1}$

（４）**a**=(-1,-1,1),**b**=(1,-1,-1),**c**=(1,1,1)の場合

これらのベクトルは原点を中心とした正四面体の 4 つの頂点の内の 3 つを指し、この場合の理論解は π となる。ここで初めて GSEF 解が理論解とずれるが、これは計算機の丸め誤差によるものと考えられる。

表４．５　理論解と GSEF 解・SoST 解（４）

理論解	3.141592653589793
GSEF 解	3.141592653589794
SoST 解 (nop=50000)	3.141592653950766

５．考察(Discussion)

計算結果から GSEF 解、すなわち Albert Girard の式を用いた立体角の算出がいかに優れているかが分かった。微小三角形の和を求める計算アルゴリズム SoST に関しては改良の余地（情報落ち対策や高速化など）があるが、それでも Girard の式を用いたアルゴリズムには到底及ばないという事実に変わりはない。

６．結言(Summary)

数値計算において立体角の値を必要とする誰かの参考になればと思います。

７．文献(References)

［１］：http://www.irf.se/~futaana/Kiruna/50_Kyumen.html
［２］：https://en.wikipedia.org/wiki/Spherical_trigonometry
［３］：http://math.rice.edu/~pcmi/sphere/gos4.html
［４］：https://en.wikipedia.org/wiki/Albert_Girard
［５］：http://mathworld.wolfram.com/GirardsSphericalExcessFormula.html

８．著者(Author)

氏名：志多　友史（工学修士）
略歴：
2011 年：下位国立大学　工学部電気系学科卒業
2013 年：同大学大学院　工学研究科修了
2013 年：研究開発機関へ就職
興味：物理・数学・コンピュータ・電気電子工作

９．備考(Notes)

特になし。