光線追跡プログラミング [*] [†]

山田 光彦 [‡]

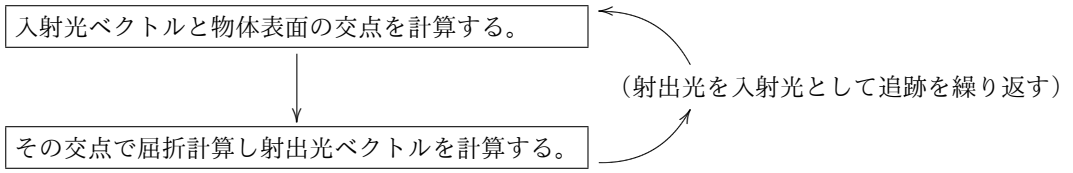
概要

光線追跡のプログラミングついて基本部分の解説をします。おもにシーケンシャルな追跡を扱います。「スネルの法則」以外の光学的な内容はありません。幾何計算、、、図形計算のみです。なるべく式を省略せず記述しています。また、屈折計算のみで、反射については扱っていません。ZEMAX ユーザー定義面 DLL サンプルソースを例とします。ただし それらはプログラム作成用であって実際の評価用ではないと思われ、詳しい動作説明は省きます。また最新の ZEMAX 機能についても承知しておらず内容不適合かもしれません
いちから光線追跡をやりたいとか、光学シミュレータへ独自形状面を追加して光線追跡をやりたいひと向けです。３ D グラフィック プログラミングであれば OpenGL などを調査したほうがよいです。 光学シミュレータ全般を構築するのであれば GNU Optical design and simulation library (`http://savannah.gnu.org/projects/goptical`) などを調査したほうがよいです。

目次

[*] 2014/05/31 Document Version 0.01a /released in commemoration of the N-th birthday
[†] 2014/06/02 Document Version 0.01b /correction of subsubsection 6.2.1 /brushup of expository writing
[‡] yees@nifty.com (http://homepage2.nifty.com/yees/)

1　シミュレーションの方法



この繰り返しです。ここで追跡方法を以下「シーケンシャル」、「ノンシーケンシャル」として分類します。シーケンシャル ノンシーケンシャル とも屈折計算そのものは同じです。

■**シーケンシャル**　光線の通過する面を定義し順次屈折計算します。ひとつのレンズは入射側と射出側のふたつの面を定義します。順次計算する点、光線行列同様です。精度を確保しながら、光線の進行とおりに追いかけていく処理は複雑で処理時間が大きくなります。それを回避するのに有効です。

■**ノンシーケンシャル**　光線の進行とおりに追いかけていくものです。

1.1　シーケンシャル

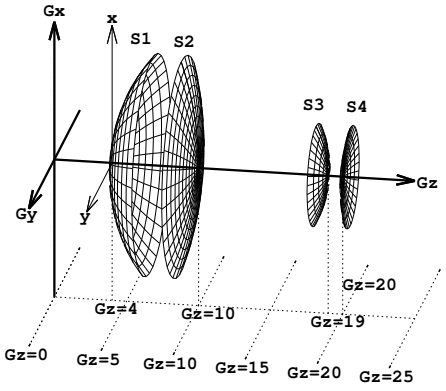


図 1　シーケンシャル 面配置

シミュレーション全体を扱う座標系 (Gx, Gy, Gz) の Gz 軸上に面を複数配置するものとします。面は

$$z = f(x, y)$$

の形式とします。面を定義する座標系 z 軸を、Gz 軸上に配置するものとします。各面で屈折計算を行いますが、その面を定義した座標系で計算することで処理が簡素になります。実際のシミュレーションには面の外形を定義する機能等必要です。

各面での屈折計算する際、入射光線の方向や入射位置は面定義座標で扱います。（最終的な追跡結果は、座標系 (Gx, Gy, Gz) で扱います）

ZEMAX ユーザー定義面 DLL の場合

- 仮想点光源の位置と方向余弦ではなく、面定義座標系 z=0 面上の通過位置 (x, y)　（引数 z 値は使用せず）
- 入射光方向余弦 (l, m, n)
- その他、入射側射屈折率、出側屈折率、形状パラメータ等

を受け取ります。屈折計算結果

- 面入射位置 (x, y, z) （プログラムの引数変数は上書き）
- 射出光方向余弦 (l, m, n) （プログラムの引数変数は上書き）
- 面入射位置の法線ベクトル (l_n, m_n, n_n)
- その他、入射光の $z = 0$ 面上の通過位置から定義面入射位置までの距離 $(path)$ 等

を返します。
ZEMAX のような入射光の扱いの場合、図 2 のように求めた射出光を次の面定義基準まで伸長し Q 座標 (x, y, z) を計算することになります。
図 3 のように、通過点 Q から入射光方向余弦 (l, m, n) とは逆方向の面探索も必要となります。
このような扱いは、面入射位置計算にニュートン法を採用する場合有利かもしれません。（$z = 0$ での通過点を初期位置として利用するとその面に近いという点で）
ニュートン法以外の方法では、P 位置を仮想点光源の位置とし入射方向余弦の向きにのみ光が進行するとしたほうがプログラム容易かもしれません。

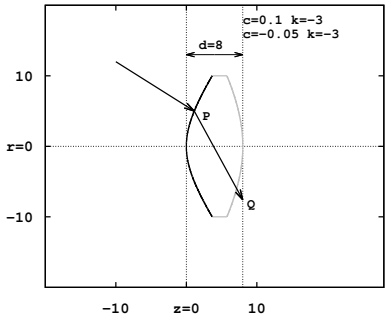


図 2　シーケンシャル 例 1

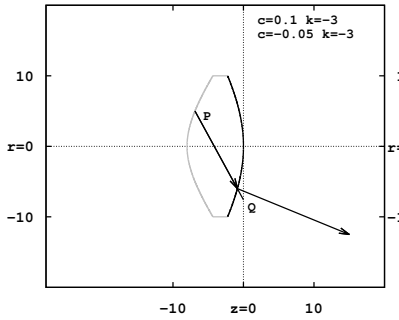


図 3　シーケンシャル 例 2

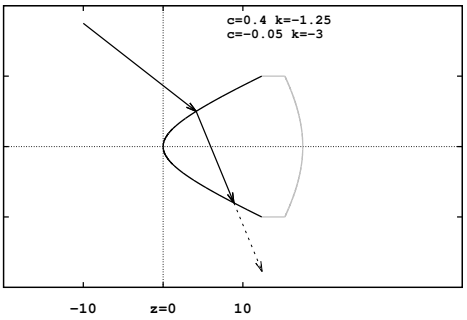


図 4　シーケンシャル 例 3

順次面ごとに屈折計算を行うため、面からの射出光は次の面への入射光となります。図 4 の場合適切な追跡ができません。
プログラム次第では対応可能ですが、ノンシーケンシャル追跡相等の処理となり処理時間は大きくなります。とくにアレイレンズのような面形状では、同一面で入射/射出が繰り返されることも考えられます。

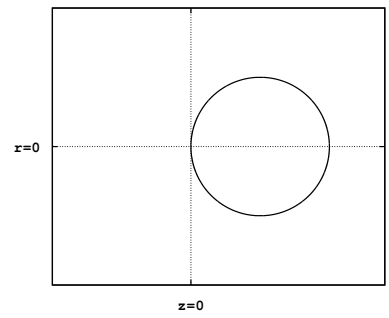


図 5　面 Ball

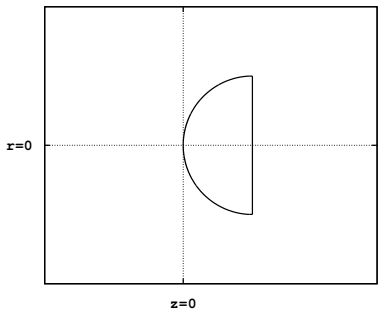


図 6　面 Hemispherical

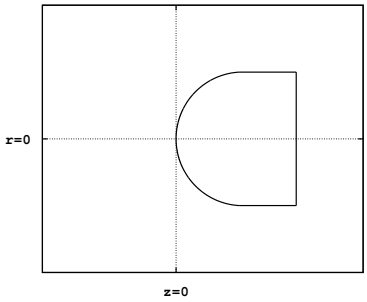
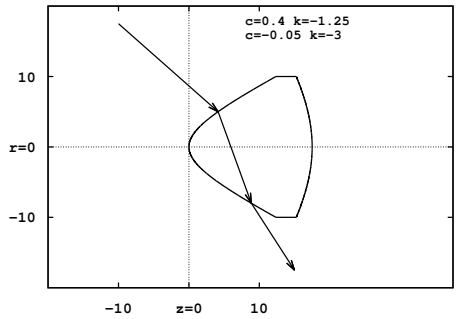


図 7　面 Bullet

球面は（図 5）２つの半球面（図 6）を定義することになります。
xy 平面に垂直な面がある場合（例 図 7）対応不可能です。$z = f(x, y)$ の形式で面定義するため、円筒部分の *z* 値が不連続値で特定できないためです。これは、面への光線入射位置（入射光と面との交点）を計算する際、「6.2.2　２分法 (３次元)」で対応可能です。
また、ニュートン法による入射位置計算では、半球面のように面傾きが大きいと失敗することがあります。

1.2　ノンシーケンシャル



光線の進行とおりに追いかけていくので、シーケンシャル追跡で困難な形状（図 4、図 7）にも対応します。シーケンシャル追跡プログラムは相当複雑で処理時間も大きくなります。次の 2 項目がその主な要因です。

図 8　ノンシーケンシャル 例

■立体物の定義

任意の形状の場合、STL ファイイルのような微小三角形の集合を考えます。精度確保のため三角形を小さくしていくと処理時間が膨大となります。または、三角形をある程度の大きさで扱い局所的に最小曲率半径でフィッティングする方法も考えられます。光学素子の評価の場合それでよいのか不明です。

数式定義された面の組み合わせや、球、楕円体、直方体などの AND/OR 論理演算による定義もあります。こちらのほうが精度確保は容易です。

■内外判定関数

任意の座標 (x, y, z) が立体物の内側か外側か判定する関数が必要となります。形状が複雑な場合この判定がいちばん厄介で処理時間が必要です。

2　ベクトル演算

2.1　内積

2.1.1　内積定義

$$\boldsymbol{A} = \left(A_x, A_y, A_z\right)$$
$$\boldsymbol{B} = \left(B_x, B_y, B_z\right)$$

のとき内積は

$$\boldsymbol{A} \cdot \boldsymbol{B} = A_x B_x + A_y B_y + A_z B_z \tag{1}$$
$$= |\boldsymbol{A}||\boldsymbol{B}| \cos\theta \tag{2}$$

2.2　外積

2.2.1　外積定義

$$\boldsymbol{A} = \left(A_x, A_y, A_z\right)$$
$$\boldsymbol{B} = \left(B_x, B_y, B_z\right)$$

のとき外積は

$$\boldsymbol{A} \times \boldsymbol{B} = \begin{vmatrix} \boldsymbol{i} & \boldsymbol{j} & \boldsymbol{k} \\ A_x & A_y & A_z \\ B_x & B_y & B_z \end{vmatrix} \tag{3}$$

$$= \left(A_y B_z - A_z B_y \,,\; A_z B_x - A_x B_z \,,\; A_x B_y - A_y B_x \right) \tag{4}$$

$$|\boldsymbol{A} \times \boldsymbol{B}| = |\boldsymbol{A}||\boldsymbol{B}| \sin\theta \tag{5}$$

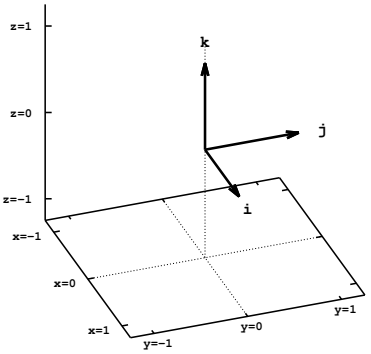


図 9　i，j，k

$\boldsymbol{A} \times \boldsymbol{B}$ の向きは、$\boldsymbol{A} \to \boldsymbol{B}$ に回したとき右ネジの進む方向などと表現されます。向きは、$\boldsymbol{i} \times \boldsymbol{j} = \boldsymbol{k}$ の関係で把握できると思います。$\boldsymbol{A}$ を $\boldsymbol{B}$ の位置まで回したとき、角度が 180 度を越えれば右ネジの進む方向とは逆になります。

2.2.2　外積の応用

数値計算により法線ベクトル計算する際に使用します。「7　法線ベクトル」
以下、図形計算でよく使用する例を示します。

■点と直線の位置関係判定（２次元）

２つの点 $A\left(A_x,\ A_y\right)$、$B\left(B_x,\ B_y\right)$ を通る直線と、点 $P\left(P_x,\ P_y\right)$ の位置関係を判定します。
点 A、B、C に、z 成分（$= 0$）を付け加え３次元化します。

$$A(Ax,\ Ay) \ \to\ A(Ax,\ Ay,\ 0)$$
$$B(Bx,\ By) \ \to\ B(Bx,\ By,\ 0)$$
$$P(Ax,\ Ay) \ \to\ P(Px,\ Py,\ 0)$$

$\overrightarrow{AB}$ と $\overrightarrow{AP}$ の外積の z 成分（Dz）を計算します。

$$Dz = (Bx - Ax)(Py - Ay) - (By - Ay)(Px - Ax)$$

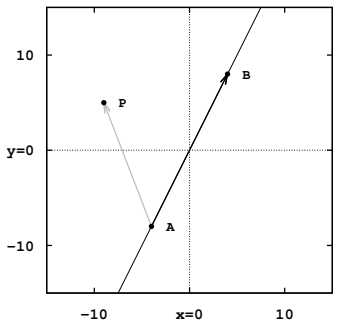


図 10　点と直線の位置関係判定

- $Dz = 0$ の場合、点 P は直線 AB 上にある。
- $Dz > 0$ の場合

  直線を $A \rightarrow B$ 方向に見たとき、点 P は直線の左側にある。
- $Dz < 0$ の場合

  直線を $A \rightarrow B$ 方向に見たとき、点 P は直線の右側にある。

図 27 の断面は、この判定を利用して描画しています。

■点と多角形の位置関係判定（２次元）

先の「点と直線の位置関係判定」を、すべての辺について行います。
すべての辺について点 P が同じ側にあれば、点 P は多角形内部にあります。
但し、凹型の多角形については三角形に分割して判定します。（図 12）

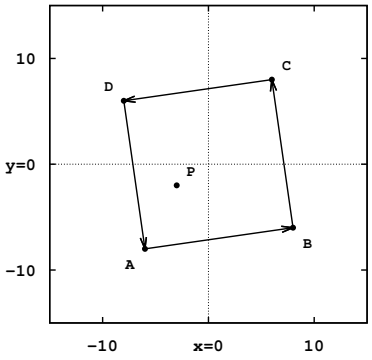


図 11　点と多角形の位置関係判定 1

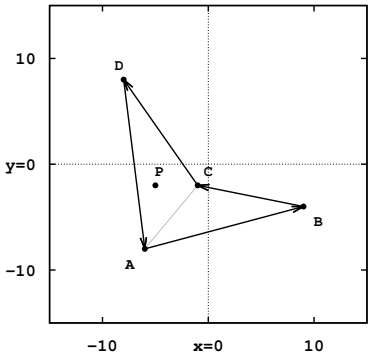


図 12　点と多角形の位置関係判定 2

3　スネルの法則

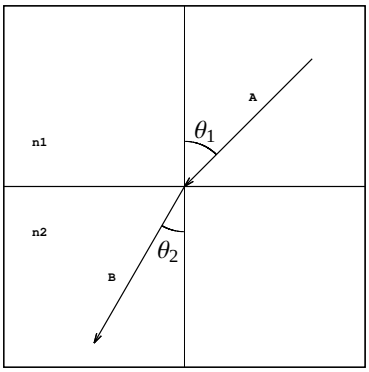


図 13　スネルの法則

$n1$ ： 入射光側 屈折率
$n2$ ： 射出光側 屈折率

$$\sin\theta_2 = \frac{n1}{n2}\sin\theta_1 \qquad (6)$$

3.1　光線屈折式

$\boldsymbol{A} = (l, m, n)$: 入射光 方向余弦
$\boldsymbol{B} = \left(\acute{l}, \acute{m}, \acute{n}\right)$: 射出光 方向余弦
$\boldsymbol{S} = (l_n, m_n, n_n)$: 法線ベクトル$\left(\text{入射光と面交点}\right)$
$n1$ ： 入射光側 屈折率
$n2$ ： 射出光側 屈折率

光線屈折式は

$$\boldsymbol{S} \times \boldsymbol{B} = \left(\frac{n1}{n2}\right) \boldsymbol{S} \times \boldsymbol{A} \tag{7}$$

スネルの法則との比較のため、絶対値をとると

$$|\boldsymbol{S} \times \boldsymbol{B}| = \frac{n1}{n2} |\boldsymbol{S} \times \boldsymbol{A}|$$

$$|\boldsymbol{S}|\,|\boldsymbol{B}| \sin \grave{\theta}_2 = \frac{n1}{n2} |\boldsymbol{S}|\,|\boldsymbol{A}| \sin \grave{\theta}_1 \tag{8}$$

$$\begin{cases} \text{内積 } \boldsymbol{S} \cdot \boldsymbol{A} \geq 0 \text{ の場合、} \grave{\theta}_1 = \theta_1 \\ \text{内積 } \boldsymbol{S} \cdot \boldsymbol{A} < 0 \text{ の場合、} \grave{\theta}_1 = \pi - \theta_1 = -\theta_1 \end{cases}$$

$$\sin \theta_2 = \frac{n1}{n2} \sin \theta_1$$

法線ベクトルを入射光と同じ方向で扱うか、逆方向で扱うかによって $\grave{\theta}$ は変わります。
これは法線ベクトルを、屈折率 $n_1 \to n_2$ 方向か、$n_2 \to n_1$ 方向か、どちらで扱うかということです。
どちらの方向であっても、式（6）は成立します。

$$\frac{n1}{n2}$$

は実数であり、 式（7）は、法線ベクトルと入射光ベクトルを含む平面に射出光ベクトルが含まれること、また その平面上でスネルの法則が成り立つことを表現しています。
法線ベクトルは、$n_1 \to n_2$ 方向、$n_2 \to n_1$ 方向、どちらでもよさそうですが、射出光方向余弦を算出する際は、考慮する必要があります。

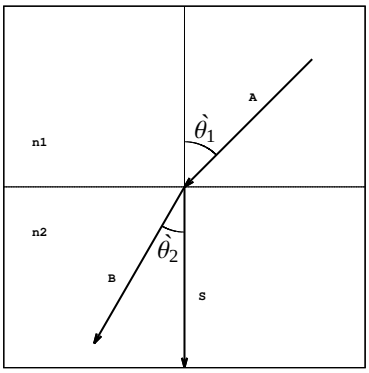


図 14　スネルの法則$\left(\text{法線ベクトル } n_1 \to n_2 \text{ 方向}\right)$

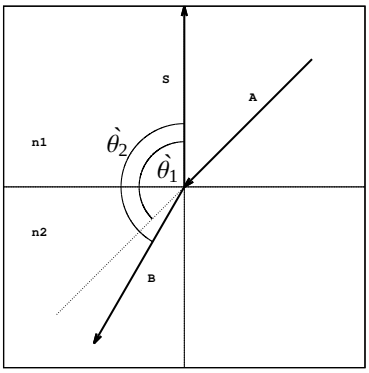


図 15　スネルの法則$\left(\text{法線ベクトル } n_2 \to n_1 \text{ 方向}\right)$

4　面定義 関数例

4.1　非球面 (偶数次項を含まない場合)

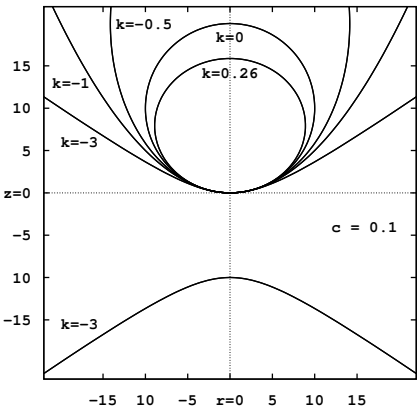


図 16　式（9） c = 0.1

$r^2 = x^2 + y^2$

c ： 曲率

k ： *Conic* 係数

陰関数形式

$$cr^2 - 2z + (k+1)cz^2 = 0 \tag{9}$$

極座標 (w, t)

$$w = \frac{2\sin t}{c\left(k\sin^2 t + 1\right)} \tag{10}$$

$(z = w\sin t \ ,\ \ r = w\cos t)$

面定義 $f = (x, y)$ は、$z = 0$ に近い側の曲線を採用して

$$z = \frac{cr^2}{1 + \sqrt{1 - (k+1)c^2r^2}} \tag{11}$$

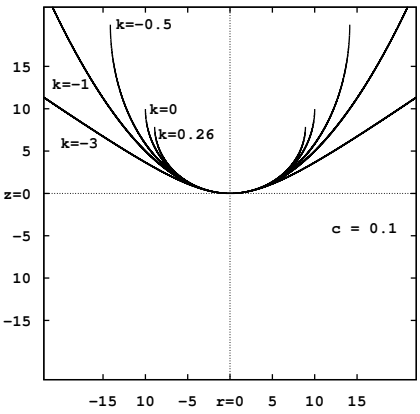


図 17　式（11） c = 0.1

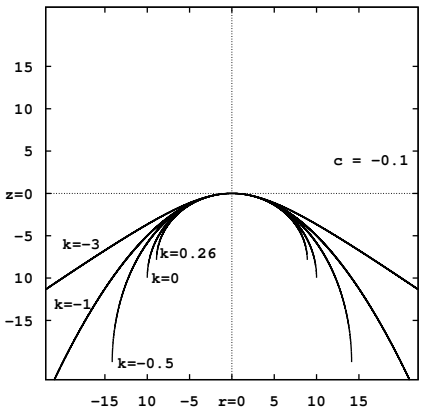


図 18　式（11） c = -0.1

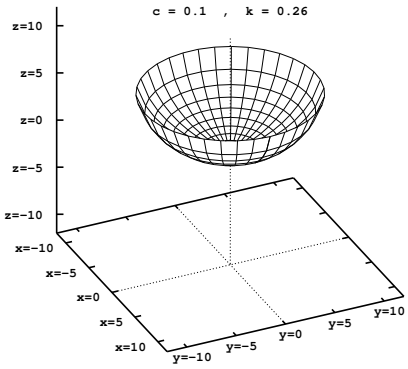


図 19　式（11） c = 0.1

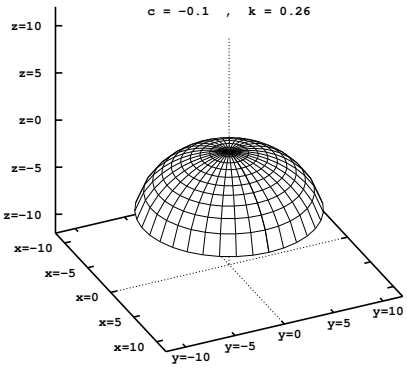


図 20　式（11） c = -0.1

4.2　非球面 (偶数次項を含む場合)

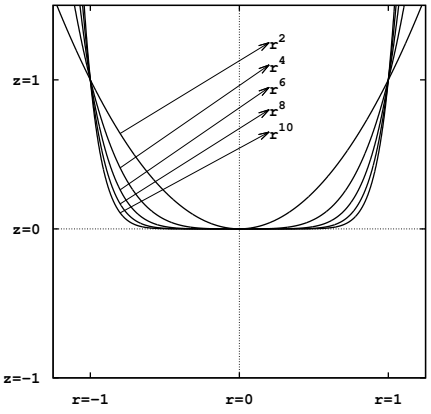


図 21　$r^2, r^4, r^6, r^8, r^{10}$

$r^2 = x^2 + y^2$

c : 曲率

k : *Conic* 係数

$$z = \frac{cr^2}{1 + \sqrt{1 - (k+1)c^2r^2}} + \sum_{i=1}^{n} a_i r^{2i} \qquad (12)$$

$$= \frac{cr^2}{1 + \sqrt{1 - (k+1)c^2r^2}} + a_1 r^2 + a_2 r^4 + a_3 r^6 + \cdots + a_n r^{2n}$$

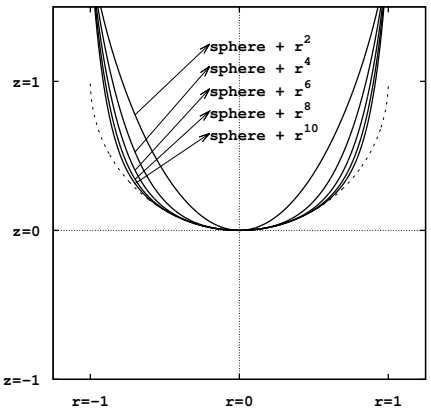


図 22　球面 $+ r^2, r^4, r^6, r^8, r^{10}$

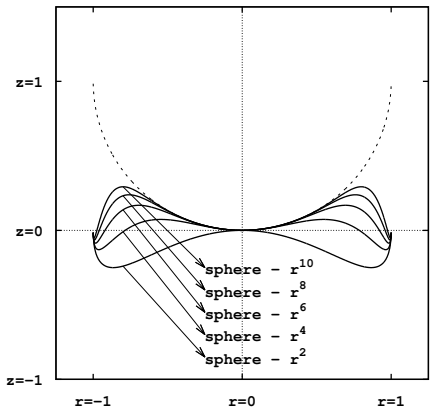


図 23　球面 $- r^2, r^4, r^6, r^8, r^{10}$

5　光線屈折計算

$\boldsymbol{A} = (l,\ m,\ n)$: 入射光 方向余弦

$\boldsymbol{S} = (l_n,\ m_n,\ n_n)$: 法線ベクトル

$n1$: 入射光側 屈折率

$n2$: 射出光側 屈折率

が与えられたとき

$\boldsymbol{B} = (\acute{l},\ \acute{m},\ \acute{n})$: 射出光 方向余弦

の算出方法です。

（入射光と面の交点、および、その位置の面法線ベクトルの算出については別記）

式（4）を使って、式（7）を書き換えれば、法線ベクトル方向余弦を算出できそうです。　この場合、*xyz* 各成分についての方程式を連立させて解くことになります。　解導出も面倒で計算時間もかかりそうです。

次の方法で、射出光 方向余弦 $\boldsymbol{B}$ を算出します。

$\boldsymbol{S}$、$\boldsymbol{A}$、$\boldsymbol{B}$ は同一平面上にあるので、

$$\boldsymbol{B} = \alpha\,\boldsymbol{A} + \gamma\,\boldsymbol{S} \qquad (13)$$

と表現し、α と γ を決定していきます。

■α の決定

式（13）の両辺について、$\boldsymbol{S}$ との外積をとり、さらに絶対値をとると、

$$
\begin{aligned}
|\boldsymbol{S}\times\boldsymbol{B}| &= |\boldsymbol{S}\times(\alpha\boldsymbol{A}+\gamma\boldsymbol{S})| \\
|\boldsymbol{S}\times\boldsymbol{B}| &= |\alpha\boldsymbol{S}\times\boldsymbol{A}+\gamma\boldsymbol{S}\times\boldsymbol{S}| \\
|\boldsymbol{S}|\,|\boldsymbol{B}|\sin\grave{\theta}_2 &= \alpha\,|\boldsymbol{S}|\,|\boldsymbol{A}|sin\grave{\theta}_1+0 \\
|\boldsymbol{S}| = |\boldsymbol{A}| &= 1 \text{ であり} \\
\sin\grave{\theta}_2 &= \alpha\ sin\grave{\theta}_1
\end{aligned}
$$

スネルの法則 式（6） と比較して、

$$\alpha = \frac{n1}{n2} \tag{14}$$

■γ の決定

式（13）の両辺について、$\boldsymbol{S}$ との内積をると、

$$
\begin{aligned}
\boldsymbol{S}\cdot\boldsymbol{B} &= \boldsymbol{S}\cdot(\alpha\,\boldsymbol{A}+\gamma\,\boldsymbol{S}) \\
\boldsymbol{S}\cdot\boldsymbol{B} &= \alpha\,\boldsymbol{S}\cdot\boldsymbol{A}+\gamma\,\boldsymbol{S}\cdot\boldsymbol{S} \\
|\boldsymbol{S}|\,|\boldsymbol{B}|\cos\grave{\theta}_2 &= \alpha\,|\boldsymbol{S}|\,|\boldsymbol{A}|\,cos\grave{\theta}_1+\gamma \\
\cos\grave{\theta}_2 &= \alpha\ \cos\grave{\theta}_1+\gamma
\end{aligned}
$$

$$\gamma = \cos\grave{\theta}_2 - \alpha\ \cos\grave{\theta}_1 \tag{15}$$

■$\grave{\theta}_1$、$\grave{\theta}_2$ の決定

式（15）内の $\grave{\theta}_1$、$\grave{\theta}_2$ を決定していきます。

手順① $\boldsymbol{S}$ と $\boldsymbol{A}$ の内積により $\grave{\theta}_1$ を求める。

$$
\begin{aligned}
|\boldsymbol{S}| = |\boldsymbol{A}| &= 1 \text{ であり} \\
\boldsymbol{S}\cdot\boldsymbol{A} &= \cos\grave{\theta}_1
\end{aligned} \tag{16}
$$

手順② $\cos^2\grave{\theta}_1+\sin^2\grave{\theta}_1=1$ の関係式により

$$\sin^2\grave{\theta}_1 = 1-\cos^2\grave{\theta}_1 \tag{17}$$

手順③ $\sin^2\grave{\theta}_1$ はスネルの法則により

$$\sin^2\grave{\theta}_2 = \left(\frac{n1}{n2}\right)^2\sin^2\grave{\theta}_1 \tag{18}$$

手順④ $\cos\grave{\theta}_2$ の決定

$\cos\grave{\theta}_1 \geq 0$ の場合、$\cos\grave{\theta}_2 \geq 0$ であり

$$\cos\grave{\theta}_2 = \sqrt{\cos^2\grave{\theta}_2} \tag{19}$$

$\cos\grave{\theta}_1 < 0$ の場合、$\cos\grave{\theta}_2 < 0$ であり

$$\cos\grave{\theta}_2 = -\sqrt{\cos^2\grave{\theta}_2} \tag{20}$$

5.1 C プログラム例

屈折率（入射側および射出側）、入射光方向余弦、法線ベクトル（面入射位置）が与えられたとき、射出光方向余弦を算出する C プログラム関数例を示します。

式（19）（20）の判定を含んでいます。呼び出し側プログラムで法線ベクトルと入射光線方向が一定に管理されていればその判定は不要です。

ZEMAX ユーザ定義面 DLL 呼び出し時は、式（20）の関係に管理されているようです。

引数	適用
n1	屈折率 入射側 n_1
n2	屈折率 射出側 n_2
*l	入射光方向余弦 l 戻り時射出光方向余弦 $\acute{l}$
*m	入射光方向余弦 m 戻り時射出光方向余弦 $\acute{m}$
*n	入射光方向余弦 l n 戻り時射出光方向余弦 $\acute{n}$
ln	法線ベクトル l_n
mn	法線ベクトル m_n
nn	法線ベクトル n_n

内部変数	適用
cost1	$\cos\grave{\theta}_1$
cos2t1	$\cos^2\grave{\theta}_1$
sin2t1	$\sin^2\grave{\theta}_1$
cos2t2	$\cos^2\grave{\theta}_2$
cost2	$\cos\grave{\theta}_2$
alpha	α
gamma	γ

戻り値	適用
0	正常終了
-1	全反射 中断 （反射方向余弦は算出しない）

（リスト開始）

```
int Refract(double n1, double n2, double *l, double *m, double *n, double ln, double mn, double nn)
{
    double cost1, cos2t1;
    double sin2t1, cos2t2, cost2;
    double alpha, gamma;

    if(n1 != n2) {    //n1=n2 なら、入射方向余弦 = 射出方向余弦
        alpha = n1 / n2;
        cost1 = (*l) * ln + (*m) * mn + (*n) * nn;
        cos2t1 = cost1 * cost1;
        if(cos2t1 > 1.0) {    // 面法線ベクトルおよび入射方向ベクトル 正規化されていれば、cos2t1 <= 1
            cos2t1 = 1.0;    // 誤差発生していれば、1 に丸める。
        }
        sin2t1 =1.0 -  cos2t1;
        sin2t2 =sin2t1 * alpha;
        cos2t2 =1.0 - sin2t2;

        if(cos2t2 < 0.0) {    // cos2t2 < 0 のとき屈折せず全反射
            return(-1);
        }
        cost2 = sqrt(cos2t2);
        gamma = cost2 - alpha * cost1;
        if(cost1<0.0){
            gamma = -gamma;
        }
        (*l) = (nr * (*l)) + (gamma * ln);
        (*m) = (nr * (*m)) + (gamma * mn);
        (*n) = (nr * (*n)) + (gamma * nn);
    }
    return 0;
}
```

（リスト終了）

6　入射光と面との交点計算

シーケンシャルに追跡する場合、面毎に光線ベクトルと面との交点を求めます。以下その方法です。

6.1　ニュートン法

今日マイコンの多くは C コンパイラなど利用でき、平方根、三角関数を数値計算することはありません。関数ライブラリが使えない非力なマイコンで、V_x、V_y から $\sqrt{V_x{}^2 + V_y{}^2}$ と $\tan^{-1}\left(V_y/V_x\right)$ を計算したいとき、数値テーブルと補間計算がお手軽です。もうちょと精度がほしいしけど、でかい数値テーブルは持ちたくない、、、
$\sqrt{\ }$ はニュートン法で、 $\tan^{-1}$ はマクローリン展開で、といった感じで使用されます。

6.1.1　ニュートン法概要

$\sqrt{a}$ 計算を例にし、ニュートン法概要を示します。

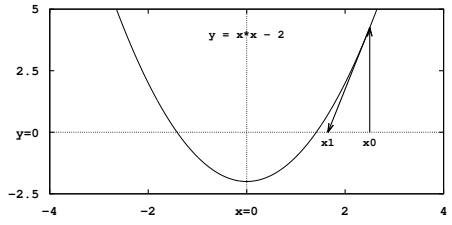


図 24　ニュートン法 $\sqrt{2}$

$\sqrt{a}$ を計算することは

$$\begin{cases} y &= f(x) = x^2 - a \\ y &= 0 \end{cases} \tag{21}$$

を解くことであり、曲線と x 軸の交点を求めます。

ニュートン法は、x_n における曲線の傾きを使い曲線を直線近似します。求めた近似解 x_{n+1} を x_n に置き換え繰り返します。$|x_{n+1} - x_n|$ が目標誤差数値に収まるまで繰り返します。　また、繰り返し回数限度を設定し越えた場合は中断します。

$$dy/dx = 2x$$
$$\Delta y_n = f(x_n) = {x_n}^2 - a$$
$$\Delta x_n = \frac{1}{[dy/dx]_{x=x_n}} \Delta y_n$$

Δx_n、Δy_n は、図 24 の三角形部の底辺、高さです。

$$\begin{aligned} x_{n+1} &= x_n - \Delta x_n = x_n - \frac{1}{[dy/dx]_{x=x_n}} \Delta y_n \\ &= x_n - \frac{1}{2x_n}\left({x_n}^2 - a\right) \\ &= \frac{1}{2}\left(x_n + \frac{a}{x_n}\right) \end{aligned} \tag{22}$$

$\sqrt{a}$ 計算の場合、a に対して x_n が大きく異なる状態では、繰り返し毎に約 1/2 となります。また、$x_n = \sqrt{a}$ となれば、$x_{n+1} = \sqrt{a}$ となります。

――――― ニュートン法の概要は ―――――
「x_n での誤差 Δy_n を 0 にするためには、x_n をどれだけ変化させればよいか、1 階の微分値で予測する。
その量が Δx_n で、x_n に Δx_n を付加し x_{n+1} とする。」です。

6.1.2　ニュートン法 (３次元)

曲面が、式（11）で定義されているのであれば、 ニュートン法のような繰り返し計算は不要です。　その場合、2 次方程式を解くことになります。 ZEMAX サンプル `us_stand.c` も解析的に計算しています。　陰関数形式の式（9）

と直線の交点を求めることになります。 但し解の存在を判別する際に、$z = 0$ 面から遠い側の解についても「解あり」判別してしまいます。 us_stand.c の場合そのあたり疑問です。

3 次元への応用ですが、光線ベクトルの大きさを変えながら、「光線ベクトル先端座標の z 値」と「その位置 (x,y) での面 $z = f(x,y)$ 」との誤差から、次に光線ベクトルの大きさを変える量を比例予測することになります。
$\sqrt{a}$ 計算の x 軸と違い、光線ベクトルは傾いていますが、比例予測する点同じです。

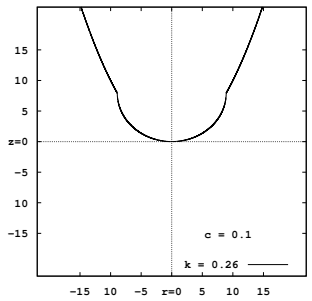


図 25　ニュートン法（3 次元） 1

ニュートン法は返し計算であり、微分値を扱うので、光線ベクトル初期値で面定義関数が計算できないと計算が始まりません。面定義関数が非球面式（11）のように未定義部分がある場合、未定義部分は適当な z を返すようにプログラムする必要があります。　図 25 はその例で、非球面式（11）の $\sqrt{\ }$ 内が負のとき 0 で計算するものです。

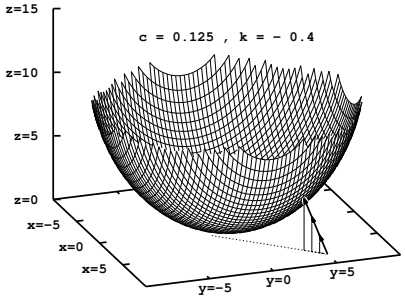


図 26　ニュートン法（3 次元） 2

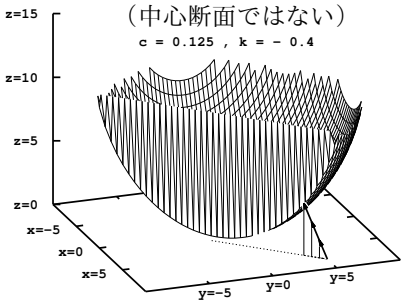


図 27　ニュートン法（3 次元） 3

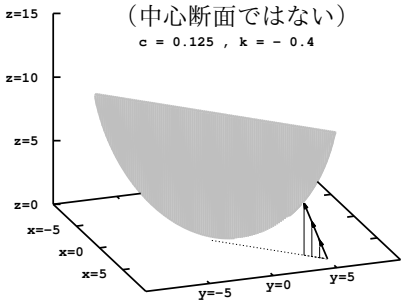


図 28　ニュートン法（3 次元） 4

図 27、図 28 のように、xy 平面に垂直で光線ベクトルを含む平面内で考えればよいのですが、一般的な面の断面曲線の微分は計算できません。
一般的な面の場合、予測のための微分値については次のとおりです。
①現在の光線ベクトルの大きさを少しだけ変えた $z = f(x,y)$ と今回 $z = f(x,y)$ の差分を使う。
②繰り返しの前回 $z = f(x,y)$ と今回 $z = f(x,y)$ の差分を使う。

②の場合、単純な曲面でも光線ベクトルとの位置関係により、収束に時間がかかったり収束しないこともあります。①のほうがよい点もありますが②より計算量が大きくなります。

ZEMAX サンプルで繰り返し計算を行っているのは、us_arrayeven.c のみです。アレイレンズですが、レンズ面は、式（12）で定義されるものです。
us_arrayeven.c では、収束判定は $1e-10$ で、最大繰り返し 1000 回以内に収束しなければ、行き先不明光、追跡中断します。
ZEMAX サンプル us_arrayeven.c では、②を行っています。
us_arrayeven.dll の実行では確認できませんが、us_arrayeven.c 内プログラムを VisualStudio 新規プロジェクトに移植し、さらにパラメータ類を設定するプログラムを作成することで内部変数や追跡中断の様子は確認できます。
収束問題以外にも理由があってニュートン法は追求せず、自身のプログラムには 2 分法を採用しています。

6.2　2 分法

6.2.1　2 分法概要

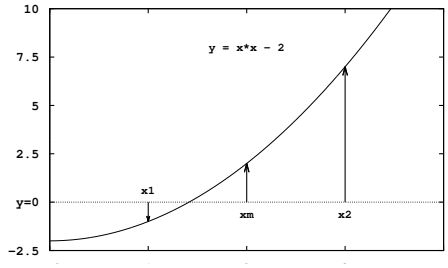


図 29　2 分法 $\sqrt{2}$

ニュートン法と同様に、代数方程式の一般的な数値計算法です。
$\sqrt{a}$ 計算を例にし、2 分法概要を示します。
次式曲線と x 軸との交点を求めることで $\sqrt{a}$ を算出します。

$$y = f(x) = x^2 - a$$

開始前に、解 x を含む範囲 $(x_1,\ x_2)$ を与える必要があります。

$x_1 < x < x_2$ または $x_2 < x < x_1$
　　その中間点を x_m とする

$\mathrm{sgn}(u)$ を u の符号を返す関数とすると

$$\begin{cases} \mathrm{sgn}(f(x_m)) \neq \mathrm{sgn}(f(x_1)) \text{ の場合、解 } x \text{ 含む範囲は } (x_m,\ x_1) \\ \mathrm{sgn}(f(x_m)) \neq \mathrm{sgn}(f(x_2)) \text{ の場合、解 } x \text{ 含む範囲は } (x_m,\ x_2) \end{cases}$$

選んだ範囲の中間点を作り、繰り返すことで範囲を絞り込んでいきます。 範囲の大きさが目標誤差数値に収まるまで繰り返します。　また、繰り返し回数限度を設定し越えた場合は中断します。

6.2.2　2 分法 (3 次元)

入射光線ベクトル軸上で、交点が存在する範囲を絞り込んでいきます。その際、「面関数 $f(x, y)$ の z 値」 と 「光線先端の z 値」 の大小関係を判定します。
開始前に交点を含む範囲を与える必要がありますが、入射光線が点光源位置 (x, y, z) と光線方向余弦 (l, m, n) で管理されている場合は、(x, y, z) から (l, m, n) 方向へ光線ベクトルを伸長（粗い間隔で）し検出できます。これはノンシーケンシャルな光線追跡に相当します。面の形状が複雑な場合、入射光線が最初に面に到達する位置を求めるのに有効的です。
ZEMAX ユーザー定義面 DLL のように、入射光線が $z = 0$ 面の通過位置 (x, y) と光線方向余弦 (l, m, n) で管理されている場合は、位置 $(x, y, 0)$ より十分後退した位置

$(x, y, 0) - t_0\,(l, m, n)$
t_0 (> 0) は、扱う座標範囲を考慮して決定する

から光線ベクトルを伸長していく必要があります。　ZEMAX サンプル us_arrayeven.c はニュートン法で交点を求めています。アレイレンズであることで、レンズ面形状パラメータや入射光方向余弦によっては、交点計算に失敗します。 ZEMAX のシミュレーション動作に影響があるかは不明です。

6.2.3　C プログラム例

（リスト開始）

```
// 内部使用関数 int sagInterpreter(double x, double y, double *z);
// は、面を定義する関数 z = f(x, y)
// 与えられた x,y に対して、z 計算不可能なら エラーコード (-1) を返す。
//
int IntersectionPoint(
                    double x, double y, double z,   // 点光源位置 (仮想)
                    double l, double m, double n,   // 光線方向余弦
```

```
                    double dPath,   //最初に交点存在を判定するとき、(l,m,n) を伸ばす変分量
                    double maxPath, //最初に交点存在を判定するとき、(l,m,n) を伸ばす最大量
                    double *pXc, double *pYc, double *pZc,      // 交点座標
                    double *pPath                          // 交点までの光線ベクトル大きさ
                    )
{
    int err;
    double tend;
    double t, tm, dt;
    double tx, ty, tz, ts;
    double tfar, txfar, tyfar;
    BOOL fIntersection, fInitialSag, fLessEqual;
    double t1, t2;
    double x1, y1, z1;
    double x2, y2, z2;
    int loop;
    double xc, yc, zc;
    double sag, sag1, sag2;
    *pXc = x;
    *pYc = y;
    *pZc = z;
    *pPath = 0.0;

    //粗サーチ ---------------------------------------------------------- 開始
    // 光線は、点光源 (x,y,z) から、(l,m,n) 方向に発光とする。
    // (x,y,z)+t(l,m,n) の t を  0 → +∞ とし、面との交点を求める
    // t は、0 から d Path 単位で、最大 maxPath まで増やす。
    //
    dt = fabs(dPath);
    tend = fabs(maxPath);
    if(dt < 1.0E-3) { dt = 1.0E-3; }

    fIntersection = FALSE;
    fInitialSag = FALSE;

    // 光線ベクトル伸ばしながら、
    // 面関数 z=f(x,y) が計算可能か判定する。
    // [面関数 f(x,y) の z 値] と [光線先端の z 値] の大小関係をメモしておく。
    // 面の曲率 c>0 か c<0 によって、面を通過判定が変わってくるので。
    for(t=0.0;t<tend;t+=dt){
        tx = x + l*t;
        ty = y + m*t;
        tz = z + n*t;
        err = sagInterpreter(tx, ty, &ts);
        if(!err){
            tm = t;
            fInitialSag = TRUE; //面関数 z=f(x,y) が計算可能になった
            if(tz<=ts){
                fLessEqual = TRUE;
            }else{
                fLessEqual = FALSE;
            }
            break;
        }
    }
    // 更に、光線ベクトルが面を通過するまで、伸ばしていく
    if(fInitialSag){
        for(t=tm;t<tend;t+=dt){
```

```
        tx = x + l*t;
        ty = y + m*t;
        tz = z + n*t;
        err = sagInterpreter(tx, ty, &ts);
        if(!err){
            // [面関数 f(x,y) の z 値] と [光線先端の z 値] の大小関係が反転したか？
            if( ((fLessEqual)&&(tz>ts)) || ((!fLessEqual)&&(tz<=ts)) ){
                tfar = t;
                txfar = x + l*t;
                tyfar = y + m*t;
                fIntersection = TRUE;   // 光線ベクトルが面を通過した（交点存在）
                break;
            }
        }
    }
}
//粗サーチ ------------------------------------------------------------ 終了
if(!fIntersection){
    //交点なし
    return(-1);
}

// 交点サーチ（２分法） ------------------------------------------- 開始
// 粗サーチで面の通過直後の光線ベクトル大きさ (tfar) を求めた。
// (tfar-dPath) と (tfar+dPath) の範囲で２分法を開始する。
// 最大繰り返し 1000 回
// 光線ベクトル大きさ変化が (1.0E-6) 以下で収束とする。
//  (x,y,z の精度 (1.0E-6) 以下で収束)
loop = 1000;
t1 = tfar - dt;
t2 = tfar + dt;

x1 = x + l*t1;
y1 = y + m*t1;
z1 = z + n*t1;
x2 = x + l*t2;
y2 = y + m*t2;
z2 = z + n*t2;
err  = sagInterpreter(x1, y1, &sag1);
err |= sagInterpreter(x2, y2, &sag2);

while(fabs(t1-t2)>1.0E-6){
    t = (t1 + t2) / 2.0;
    xc = x + l*t;
    yc = y + m*t;
    zc = z + n*t;
    err = sagInterpreter(xc, yc, &sag);
    if(n>=0.0){
        if(zc>sag){
            t2 = t;
            x2 = xc;
            y2 = yc;
            z2 = zc;
        }else{
            t1 = t;
            x1 = xc;
            y1 = yc;
            z1 = zc;
```

```
            }
        }else{
            if(zc<sag){
                t2 = t;
                x2 = xc;
                y2 = yc;
                z2 = zc;
            }else{
                t1 = t;
                x1 = xc;
                y1 = yc;
                z1 = zc;
            }
        }
        loop--;
        if(loop<0){
            return(-1);
        }
    }
    // 交点サーチ（２分法） ---------------------------------------- 終了
    *pXc = xc;
    *pYc = yc;
    *pZc = zc;
    *pPath = t;
    return(0);  //No Error
}
```

（リスト終了）

6.2.4　使用例

２分法を使用した光線追跡プログラム実行例です。C プログラム例そのものを使用しています。ニュートン法では困難（初期位置による）な面の例です。

ここでは、面に入射したした光線の屈折計算をしたあと、もう一度面に交差しないかチェックしています。（交差しているので破線で表示）「その交差位置で再度、屈折計算する」を繰り返せば、ノンシーケンシャル追跡となります。

ZEMAX ユーザー定義面 DLL 内で、このような処理を行っても ZEMAX 本体プログラムに知らせる手段がないので対応不可能です。ZEMAX では立体モデルを用意してノンシーケンシャル追跡を行うことになります。

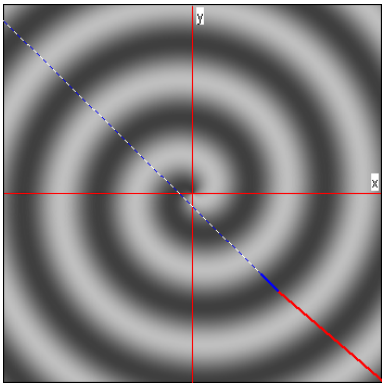


図 30　usSurfaceViewerver1.26 （１）

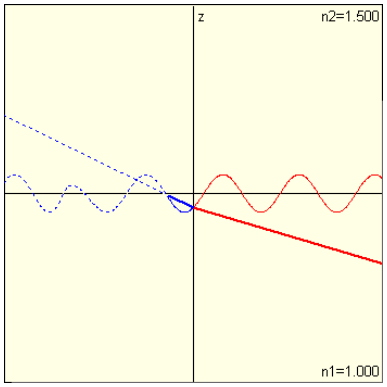


図 31　usSurfaceViewerver1.26 （２）

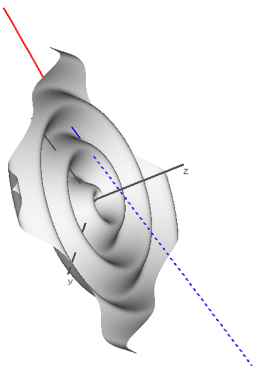


図 32　usSurfaceViewerver1.26 （５）

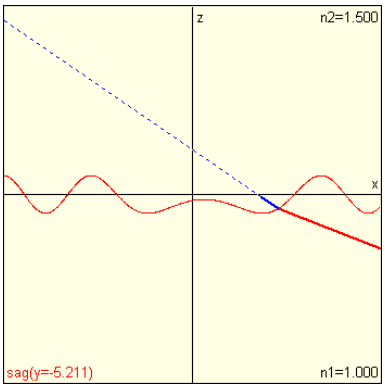


図 33　usSurfaceViewerver1.26 （３）

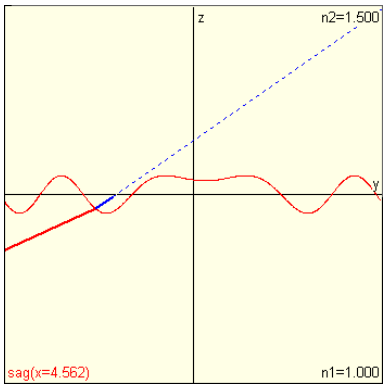


図 34　usSurfaceViewerver1.26 （４）

面定義関数
（半径方向の波形に sin を使っているだけです）
（リスト開始）

```
 e = 4;
 x0 = x / e;
 y0 = y / e;
 r = sqrt(x0*x0+y0*y0);
 t = 0;
 if(abs(x0)>0 && abs(y0)>0){
     if((x0>0) && (y0>0))
         t = atan(y0/x0);
     if((x0<0) && (y0>0))
         t = pi-atan(y0/-x0);
     if((x0<0) && (y0<0))
         t = pi+atan(-y0/-x0);
     if((x0>0) && (y0<0))
     t = 2*pi-atan(-y0/x0);
 }
 t = t / 2 / pi;
 z = sin((r-t)*2*pi);
```

（リスト終了）

7　法線ベクトル計算

7.1　非球面の場合

曲面が、式（11）または 式（12）で定義されているのであれば、解析的に計算可能です。ZEMAX サンプル us_stand.c 、us_arrayeven.c とも解析的に計算しています。
面が z 軸回転面であることから、法線ベクトル (l_n, m_n, n_n) のうち (l_n, m_n) は半径方向です。 (x_0, y_0, z_0) を面上の点とすると、半径方向のベクトル $(x_0, y_0, 0)$ を傾け正規化したものが法線ベクトルになります。
式（11）式（12）とも回転面を表し、$r > 0$ だけ考えればよいので、法線ベクトルの半径方向の傾き (dz/dr) を $-90°$ 傾けて

$$\tan\left(\tan^{-1}(dz/dr) - 90°\right) = -\tan\left(90° - \tan^{-1}(dz/dr)\right) = -\frac{1}{\tan\left(\tan^{-1}(dz/dr)\right)} = -\frac{1}{dz/dr}$$

法線方向のベクトル（$\boldsymbol{R}$ とする）を作ると

$$\boldsymbol{R} = \left(x_0,\ y_0,\ -\frac{1}{[dz/dr]_{r=r_0}} r_0\right)$$

ここで

$$r_0 = \sqrt{x_0^2 + y_0^2} \tag{23}$$

このベクトルの大きさ（R とする）は

$$R = |\boldsymbol{R}| = \sqrt{x_0^2 + y_0^2 + \left(-\frac{1}{[dz/dr]_{r=r_0}} r_0\right)^2} = \sqrt{r_0^2 + \left(\frac{1}{[dz/dr]_{r=r_0}}\right)^2 r_0^2} = r_0\sqrt{1 + \left(\frac{1}{[dz/dr]_{r=r_0}}\right)^2}$$

正規化して法線ベクトル $(l_n,\ m_n,\ n_n)$ は

$$(l_n,\ m_n,\ n_n) = \left(\frac{x_0}{R},\ \frac{y_0}{R},\ \frac{-\frac{1}{[dz/dr]_{r=r_0}} r_0}{R}\right) = \left(\frac{x_0}{R},\ \frac{y_0}{R},\ -\frac{r_0}{R\ \ [dz/dr]_{r=r_0}}\right) \tag{24}$$

(dz/dr) は式（9）を r で微分して

$$c2r - 2\frac{dz}{dr} + (k+1)\,c2z\frac{dz}{dr} = 0$$

$$\frac{dz}{dr} = \frac{cr}{1 - (k+1)\,cz} \tag{25}$$

以上で非球面式（11）の法線ベクトルが算出できます。
式（12）の場合は、式（25）に偶数次項の微分を付け加えればよさそうです。

7.2　一般的な曲面の場合

一般的な面の場合は、面上の点 (x_s, y_s, z_s) 近傍で面上の３点を求め、３点から２つのベクトルをつくり外積を計算します。さらにその外積を大きさ１に正規化します。
計算量を減らすためには、点 (x_s, y_s, z_s) を中心とする近傍３点を求めるのではなく

$$(x_s + \Delta x,\ y_s,\ z(x_s + \Delta x,\ y_s))$$
$$(x_s,\ y_s + \Delta y,\ z(x_s,\ y_s + \Delta y))$$

の２点を求め、(x_s, y_s, z_s) とともに３点とします。（ここで $z(x,y)$ は面を定義する関数）
Δx、Δy は目的精度にあわせ微小値を与えます。
２つのベクトルをつくる際に、３点からの選び方により法線ベクトルの z 軸方向の向きが変わるので、必要な向きにあわせ選択します。

$\Delta x > 0 \quad , \quad \Delta y > 0$
点 $Z_0\,(x_s,\ y_s,\ z_s)$
点 $Z_1\,(x_s + \Delta x,\ y_s,\ z\,(x_s + \Delta x,\ y_s))$
点 $Z_2\,(x_s,\ y_s + \Delta y,\ z\,(x_s,\ y_s + \Delta y))$

とすると

$\left(\overrightarrow{Z_0Z_1} \times \overrightarrow{Z_0Z_2} \right)$ の z 成分 > 0
$\left(\overrightarrow{Z_0Z_2} \times \overrightarrow{Z_0Z_1} \right)$ の z 成分 < 0

となります。