

陰影を付けた立体図の K_εTpic による描画

長野工業高等専門学校・一般科 前田 善文 (Yoshifumi Maeda)

Fuculty of General Education,

Nagano National College of Technology

東邦大学・薬学部 高遠 節夫 (Setsuo Takato)

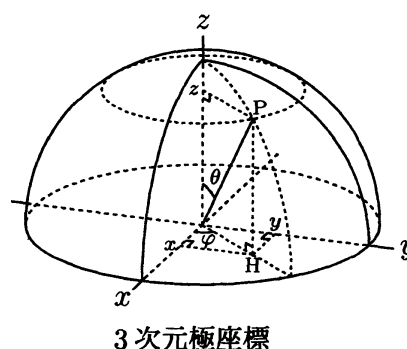
Fuculty of Pharmaceutical Science,

Toho University

1 はじめに

教材によっては、図やグラフがあれば容易に理解を深めることができるものがある。教材に必要な図やグラフを手軽に作成でき、簡単に位置を指定して文書中に張り付けることができるツールとして K_εTpic はとても便利である。K_εTpic には豊富なコマンドがあり、さらに図を作成する上でのきめ細かな設定も行うこともでき、平面図形は勿論のこと立体図形も簡単に思い通りの作図をすることができる。

Scilab 版の K_εTpic を使って、まず始めに正確な立体図があれば理解が深まるような教材について作成を行った。例として、3次元極座標の説明のために作成した図を右に示す。次に2重積分の求積問題における立体図を作成したが、立体感が少し乏しかったため、立体図に陰影を付けることを試みた。



2 研究の内容

陰影を付けるにあたって、考慮したことは次の2点である。

- (1) 印刷教材の作成を目的とする。
- (2) すべてを T_EX 文書として作成する。

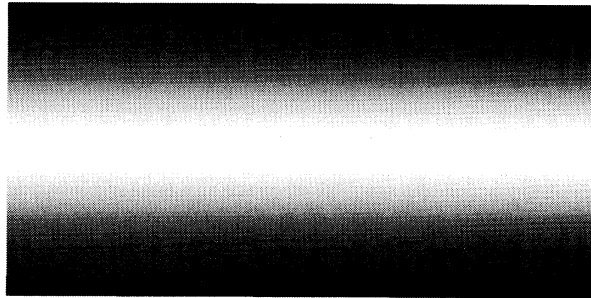
3次元コンピュータグラフィックス (CG) のような精緻な陰影のついた図 (画面) の作成を目的とするのではなく、学生が印刷された図を見て、立体的に見えることによって内容の理解が深まるような立体図を作成することが目的である。

また、すべてが T_EX 文書という作成のための制限があるが、逆に大きな画像ファイルを取り込む必要がなく、作成された PDF ファイルの大きさは小さなものになるという利点がある。

陰影を付けるための方法として、`KpTpic` のコマンド `Shade` を用いた。いくつかの閉曲線のリストを作り `Shade` によって閉曲線の内部を濃さを変えながら塗りつぶしていくという方法で陰影を付けることにした。立体図に陰影を付ける基本的な方法として円柱と球面に陰影を付ける例で3通りの方法を紹介する。

2.1 方法1

外側から内側に向かって明度を上げながら塗りつぶす方法
(図は水平に光があたっている円柱を正面から見た場合)



Scilab のプログラムを次に示す。

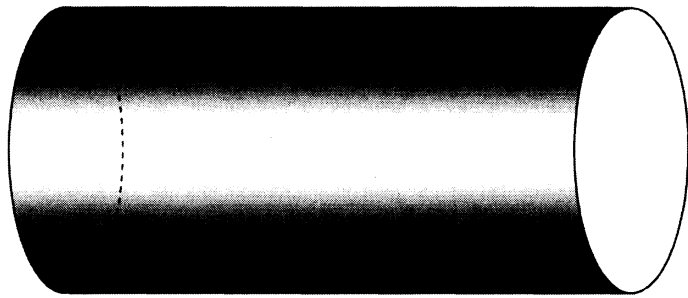
```
Setwindow([-5,5],[-3,3]);
LnX=4; LnY=2;           : 横幅 8, 円の半径 2
Shd=1; Shd0=0.1; Shd1=0.5; Shd2=0.4; Spw=2;           : 明度を計算する定数
L0=list();               : 図形データを保存するリスト
K0=50;                   : 分割数
Th=%pi/2; T0=Th/K0;      : T0 は分割された角度
for k=0:K0-1,
    Py=LnY*cos(k*T0);     : 長方形の縦の長さの半分
    L0($+1)=Framedata([-LnX,LnX],[-Py,Py]);           : 長方形の図形データ
end
Openfile('shade1.tex');  : ファイル shade1.tex をオープン
Beginpicture('1cm');
for k=0:K0-1,
    Ts=%pi/2*(1-k/(K0-1)); : 円柱の法線と光のなす角
    Tp=2*Ts;               : 反射光と視線のなす角
    if Tp>%pi/2 then Tp=%pi/2; end
    Mc=Shd0+Shd1*cos(Ts)+Shd2*(cos(Tp))^Spw;           : 明度 (注参照)
    Mc=Shd*(1-Mc);         : 塗りつぶす濃さ
    Texcom('{ ¥ color[cmyk]0,0,0,'+string(Mc)+'}');    : 色指定 (黒の濃さ)
    Shade(L0(k+1),1);      : L0(k+1) は長方形のデータ, 内部を塗りつぶす
    Texcom('{}');
end
Endpicture(0); Closefile();
```

《注》明暗のコントラストにより立体感を与える技法としての Phong shading に準じて明度の計算を行っているが、教材作成を主目的としているためこの方法によらず立体の特徴が分かるような簡単な shading 方法でよいと考える。

2.2 方法2

区分求積法と同様に立体の表面を細長い帯状(短冊)に切って塗りつぶす方法

(水平に光があたっている円柱を視点を22°横にずらして見た場合)



```

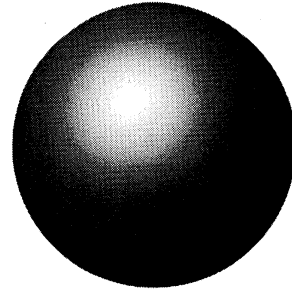
LnX=4; LnY=2; Phi=22/180*%pi;           : 視角 Phi= 22°
A=LnY; B=tan(Phi)*LnY;                   : B は楕円の短軸の長さ
Shd=1; Shd0=0.1; Shd1=0.5; Shd2=0.4; Spw=2; : 明度を計算する定数
L0=list();                               : 図形データを保存するリスト
K0=50;                                   : 分割数
Th=%pi/2; T0=Th/K0;                     : T0 は分割された角度
X0=LnX; Y0=LnY;                         : 初期値…分割点 0 に設定
for k=0:2*K0-1,
    Ts0=k*T0; Ts1=(k+1)*T0;
    X1=LnX-B*sin(Ts1); Y1=A*cos(Ts1);    : 分割点 1
    G=Listplot([X0,Y0],[X1,Y1]);          : 2 点を結ぶ線分
    G=Joincrvs(G,Listplot([X1,Y1],[X1-2*LnX,Y1])); : 線分の連結
    G=Joincrvs(G,Listplot([X1-2*LnX,Y1],[X0-2*LnX,Y0]));
    G=Joincrvs(G,Listplot([X0-2*LnX,Y0],[X0,Y0])); : 閉曲線を作成
    L0($+1)=G;                            : 短冊形のデータ
    X0=X1; Y0=Y1;                         : 分割点 1 を分割点 0 に設定
end
Openfile('shade2.tex'); Beginpicture('1cm');
for k=0:K0-1,
    Ts=%pi/2*(1-k/(K0-1)); Tp=2*Ts;
    if Tp>%pi/2 then Tp=%pi/2; end
    Sc2=cos(Tp)*cos(Phi);                  : 反射光と視線のなす角の余弦
    Mc=Shd0+Shd1*cos(Ts)+Shd2*Sc2^Spw;    : 明度
    Mc=Shd*(1-Mc);                        : 塗りつぶす濃さ
    Texcom('{ ¥ color[cmymk]0,0,0,'+string(Mc)+'}'); : 色指定 (黒の濃さ)
    Shade(L0(k+1),1); Shade(L0(2*K0-k),1); : 内部を塗りつぶす
    Drwline(L0(k+1));                     : 注参照
    Drwline(L0(2*K0-k)); Texcom('}');
end
Endpicture(0); Closefile();

```

《注》 幅の狭い図形の内部を塗りつぶすとき、塗り残しが無いように境界線も描画する。

2.3 方法3

2重積分と同様に領域を微小領域(矩形)に分割し,
対応する微小曲面を塗りつぶす方法
(左上から光があたっている球を正面から見た場合)



```

R=2;                                : 球面の半径
Th0=60; Phi0=-20;                   : 光がくるの方向
Th=Th0/180*%pi; Phi=Phi0/180*%pi;   (θ = 60°, φ = -20°)
Vd=[sin(Th)*cos(Phi),sin(Th)*sin(Phi),cos(Th)];      : 光線の方向ベクトル
Ve=[1,0,0];                          : 視線の方向ベクトル
Shd=1; Shd0=0.2; Shd1=0.45; Shd2=0.35; Spw=2;       : 明度を計算する定数
K0=35; K1=90;                        : θ, φ の2方向の分割数
    θ は x 軸と動径のなす角, φ は yz 平面への正射影と y 軸とのなす角を表す
Dth=%pi/2/K0; Dph=2*%pi/K1;          : 分割された角度 dθ, dφ
Openfile('shade3.tex'); Beginpicture('1cm');
Dth0=0; R0=R*sin(Dth0);               : 初期値 θ₀ = 0, r₀ = 0
for j=1:K0,
    Dph0=0; Dth1=Dth0+Dth; R1=R*sin(Dth1);          : 初期値
                                                φ₀ = 0, θ₁ = θ₀ + dθ, r₁ = R sin(θ₁)
    P0=R0*[cos(Dph0),sin(Dph0)]; P1=R1*[cos(Dph0),sin(Dph0)];
    for k=1:K1,
        Dph1=Dph0+Dph;                        : φ₁ = φ₀ + dφ
        P2=R1*[cos(Dph1),sin(Dph1)]; P3=R0*[cos(Dph1),sin(Dph1)];
        G=Listplot(P0,P1,P2,P3,P0);            : 閉曲線を作成
        P0=P3; P1=P2; Dph0=Dph1;
        Vn=[cos(Dth0),sin(Dth0)*cos(Dph0),sin(Dth0)*sin(Dph0)];
        Kd=Dotprod(Vn,Vd);                     : Vn は球面の法線ベクトル
        Vs=2*Kd*Vn-Vd;                         : Vs は反射光の方向ベクトル
        if Kd<0 then Kd=0; end
        Ks=Dotprod(Ve,Vs); if Ks<0 then Ks=0; end : Ve は視線方向
        Mc=Shd0+Shd1*Kd+Shd2*Ks^Spw;           : 明度
        Mc=Shd*(1-Mc);                         : 塗りつぶす濃さ
        Texcom('{ ¥ color[cmymk]0,0,0,'+string(Mc)+'}'); : 色指定 (黒の濃さ)
        Shade(G,1); Drwline(G);                : G の内部と境界線を塗りつぶす
        Texcom('}');
    end
    R0=R1; Dth0=Dth1;                        : 次のループのための初期設定
end
Endpicture(0); Closefile();

```

《注》 この例では、2重に for 文を実行しているため、分割数の 35 と 90 に対して 35×90 すなわち 3150 回の繰り返しが必要となる。ファイルをオープンする前に図形のデータを作成してリストに保存していると、繰り返しの回数が 2 倍になり、実行時間がその分長くなる。実行時間を短縮するために閉曲線を作成すると同時に内部と周の塗りつぶしを行って陰影を付けている。作成された $\text{T}_\text{E}\text{X}$ ファイルは 887KB、実行時間はパソコンの機種により異なるが 1 分程度である。作成された図をよく見ると、鱗のように筋が入っている。これは分割数を増やせば目立たなくなるが、作成された $\text{T}_\text{E}\text{X}$ ファイルが大きくなり、実行時間も長くなる。印刷したときの状態をみて分割数を決めればよい。

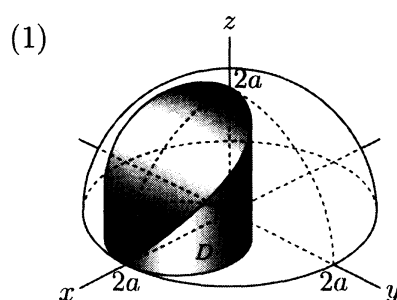
2.4 各方法の利点と欠点

方法 1 と方法 2 は円柱や円錐などの簡単な図形の場合には、プログラムの作成も容易であり、印刷教材に立体感を表現するためには適しているといえる。球面の場合も視線の方向と光源の方向が一致している場合は方法 1 または方法 2 で作成することができる。また、方法 1 と方法 2 では $\text{K}_\text{E}\text{T}_{\text{pic}}$ によって作成された $\text{T}_\text{E}\text{X}$ ファイルの大きさも小さなものとなる。しかし、複雑な立体図形には対応できない。

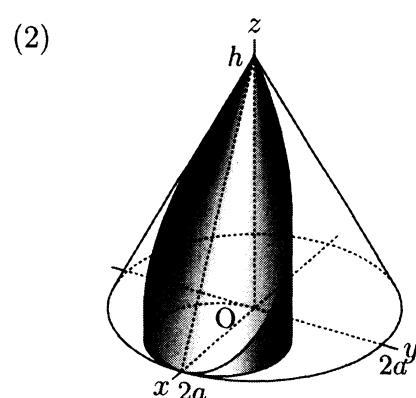
方法 3 は複雑な立体図形や光の方向・光源の数にも対応できるが、2重のループを考える必要があるため、Scilab のプログラムも複雑になる。また、作成された $\text{T}_\text{E}\text{X}$ ファイルの大きさも画像ファイル程ではないが方法 1 や方法 2 と比べてはるかに大きくなる。

このことから、印刷教材を作成するにあたって、どの程度に陰影を付けて立体感を表現する必要があるかを見極めて、必要に応じて陰影を付ける方法を選択することが大切であると考ええる。

3 陰影を付けた立体図の作成例

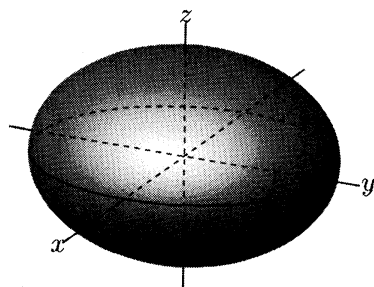


上面と側面の陰影は光の方向が異なるが、立体の特徴を表すために敢えて方向を変えて方法 1 によって作図した。



線画では上面が凹んで見えるが、陰影を付けることによって円錐面と同じように膨らんで見えるようになる。

(3)

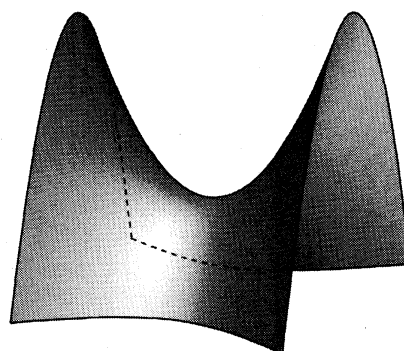


$$\text{曲面: } \frac{x^2}{4^2} + \frac{y^2}{5^2} + \frac{z^2}{3^2} = 1$$

視線の方向角: $\theta = 70^\circ$, $\varphi = 25^\circ$,

分割数: 35×90

(4)



$$\text{曲面: } z = \frac{x^2}{4} - \frac{y^2}{4}$$

視線の方向角: $\theta = 75^\circ$, $\varphi = -70^\circ$

分割数: 70×25

4 考察と今後の課題

K_ETpicを使う以前には、教材作成をするにあたり、図を挿入すると理解が深まると思われる内容についても作図が困難であったため、文章だけで説明するか教科書等の図をコピーして貼り付けて使用していた。K_ETpicを使い始めてからは、教材の内容に適した図を工夫して簡単に作成することができるようになり、わかり易い図入り教材を作成することが可能になった。さらに、教材作成の負担も軽減され、新しい教材作成の意欲も増して楽しく教材作成ができるようになったといえる。

これまで比較的単純な立体図形に陰影を付けることを試みたが、今後はさらに様々な立体図形に陰影を付けることと、より立体的に感じられるための方法について検討していきたいと考えている。さらに、現在は立体ごとに立体の特徴から陰影を付けることを行ってきたが、一般に曲面の方程式等を与えることによって自動的に陰影を付ける手法についても考えていきたい。

また、学生たちに立体感覚を養わせるためにも、K_ETpicは有効であると考えられる。作図をすることによって、立体や数学への理解が深まり、数学に対する興味や関心も強くなると思われる。機会を捉えK_ETpicを紹介し、興味を持った学生に指導をしていきたいと考えている。

参考文献

- [1] CAST_EX 応用研究会:『K_ETpic で楽々T_EX グラフ』, イーテキスト研究所, 2011.