# 9.2　陰線消去

## (1)考え方

等高線は，3 次元形状を数値的に正確に表示するという意味では有効ですが，直感的に図形を把握するのが困難です。そこで，普段，見慣れた見取り図で表示することを試みましょう。

曲線の XYZ 座標を 2 次元に平行投影するのが，最も簡単に見取り図を表示する方法です。図 9-3 に示す式が平行投影における変換式です。

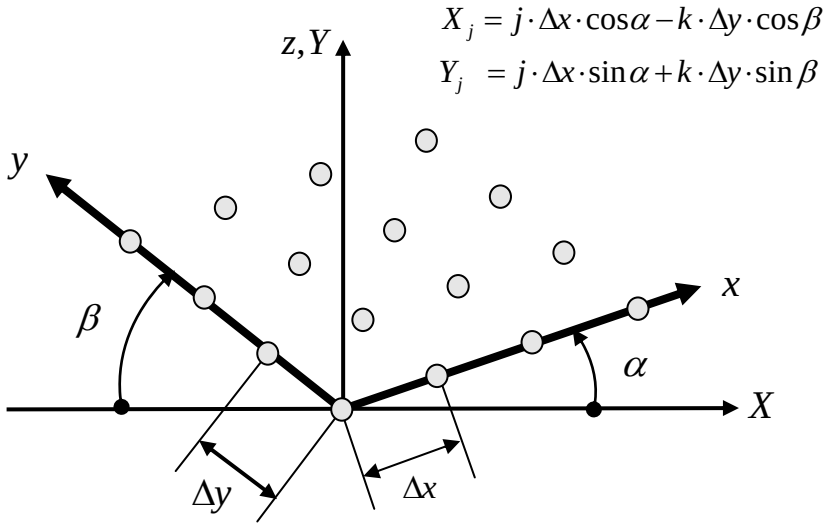


図 9-3　平行投影の変換式

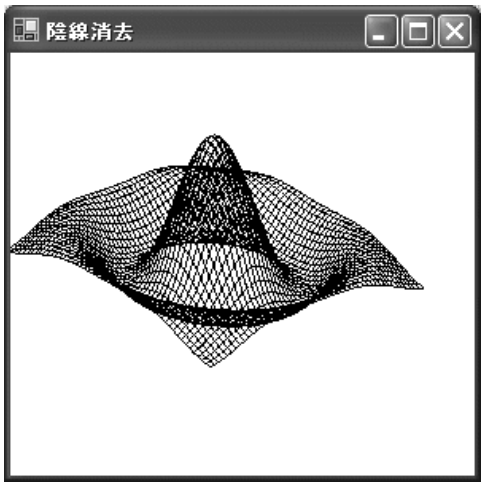


図 9-4　平行投影の例

たとえば，図 9-4 は，以下の式で得られた $z = f(x, y)$を平行投影した図です。

$$f(x,y)=\frac{\sin\sqrt{x^2+y^2}}{\sqrt{x^2+y^2}}$$

しかし，隠れた線まで表示されています。そこで，見えない線を消すと，より見やすくなるはずです。この見えない線を消すことを**陰線消去**と呼びます。図 9-5 は，図 9-4 に対して陰線消去を行った結果です。

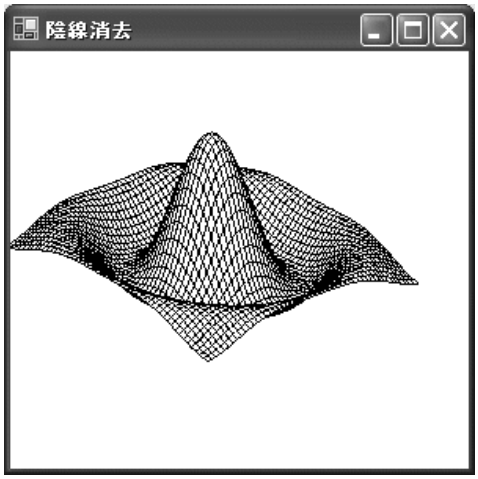


図 9-5　陰線消去の例

## (2)浮動水平線アルゴリズム

ここで示す方法は，**浮動水平線アルゴリズム**または**最大最小法**と呼ばれる方法です。その手順の概略は，以下のとおりです。

(a) 一番，手前から描く。
(b) 現在描いている曲線の 1 点の Y 座標が，それ以前に描かれた曲線の最大 Y 座標値より大きければ(水平線より上に位置すれば)，その点が見えるものとして描く。

すなわち，図 9-6 のように，実線部分は，先に描かれた 2 次元変換後の Y 座標より大きいので描かれ，点線の部分は小さいので描きません。

この手法では，描画の進行に伴って，水平線が Y の正の方向に上がっていきますので，浮動水平線アルゴリズムと呼ばれます。

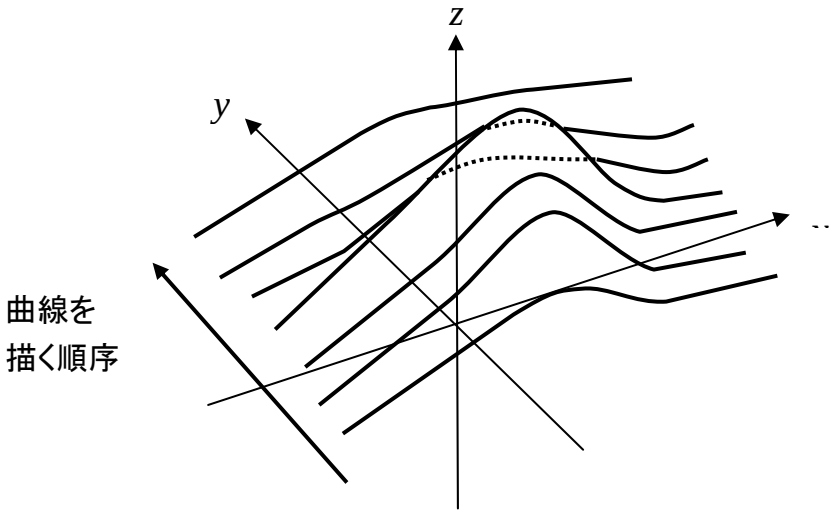


図 9-6　浮動水平線アルゴリズム

## (3)プログラムの説明

2 次元の絵を描きますので，以下の using を追加します。

```
using System.Drawing.Drawing2D;
```

【データ領域の宣言１】

```
public double Hidden_dlx;   // 表示刻み幅(dl)
public double Hidden_alpha; // x 軸と水平軸との角度(α)
public double Hidden_beta;  // y 軸と水平軸との角度(β)
public double Hidden_dx;    // x 軸方向の単位メッシュの長さ(dx)
public double Hidden_dy;    // y 軸方向の単位メッシュの長さ(dy)
public double[,]高さ= new double[51,51];// 高さ(z 値)
public double[] YMax= new double[2000] ;// Y 座標値最大値(上浮動水平線)
public double[] YMin= new double[2000] ;// Y 座標値最小値(下浮動水平線)
public double Hidden_Xlen; // 表示上の X 方向長さ = (numX-1)*dx*cos(α)
public double Hidden_Ylen; // 表示上の X 方向長さ = (numY-1)*dx*cos(β)
public int Hidden_NR;      // 浮動水平線用配列の長さ
public double beforX;      // 現在ペン位置 X
public double beforY;      // 現在ペン位置 Y
public int numX=51;        // x 方向メッシュ数
public int numY=51;        // y 方向メッシュ数
```

【データ領域の宣言 2】

```
public double Hidden_dxCosA;  // dx*cos(α)
public double Hidden_dyCosB;  // dy*cos(β)
public double Hidden_dxSinA;  // dx*sin(α)
public double Hidden_dySinB;  // dy*sin(β)
public double Hidden_dlxTanA; // dl*tan(α)
public double Hidden_dlxTanB; // dl*tan(β)
public Matrix matrix= new Matrix();
                          // グローバル座標系への変換マトリックス
```

【陰線消去の処理】

```
public bool Hidden_Draw(PaintEventArgs e,Pen pen,
          double px,double py, int p, bool Visible, bool Update)
{ // 陰線かどうかを判断し，陰線でない場合，線を描く。
  //
  // 関数値 :表示後の可視フラグ
  // e      :描画用引数
  // pen    :ペン属性
  // px     :補間された X 座標値(平面座標系)
  // py     :補間された Y 座標値(平面座標系)
  // p      :比較する浮動水平線の位置
  // Visible:現ペン位置が見えているかどうかを示す(可視フラグ)
  // Update :陰線でないとき，浮動水平線を更新するかどうかを示すフラグ
  if((py>=YMax[p])||(py<=YMin[p]))
  { if(Update && py >=YMax[p])YMax[p]=py;
    if(Update && py <=YMin[p])YMin[p]=py;
    if(Visible)
    { float fx1=(float)beforX; float fy1=(float)beforY;
      float fx2=(float)px    ; float fy2=(float)py;
      e.Graphics.DrawLine(pen,fx1,fy1,fx2,fy2);
    }
    beforX=px;beforY=py; return true;
  }
  else{ beforX=px;beforY=py; return false;}
}
```

【X0=Y0=0 の 2 次元座標値】

```
private double Hidden_GroundX(int j, int k) // X0=Y0=0 のときの X 座標
{ return (double)j * Hidden_dxCosA - (double) k * Hidden_dyCosB;}

private double Hidden_GroundY(int j, int k)// X0=Y0=0 のときの Y 座標
{ return (double)j * Hidden_dxSinA + (double) k * Hidden_dySinB;  }
```

【OnPaint のオーバライド】
ここでは，OnPaint で直接表示しています。

```
private int setDrawPos(int j,int k)
{ return (int)((0.5+(Hidden_Ylen+Hidden_GroundX(j,k))  /Hidden_dlx));}
private double setPx(int j, int k, double XO, double PH)
{ return (PH * Hidden_dlx + Hidden_GroundX(j,k) + XO);}
private double setPy(int j, int k, double YO, double PH, double fp)
{ return (PH * Hidden_dlxTanA + Hidden_GroundY(j,k) + fp + YO);}
protected override void OnPaint(PaintEventArgs e )
{  bool 可視フラグ=true; base.OnPaint(e);
   e.Graphics.Clear(Color.White);
   Pen pen = new Pen(Color.Black,0.02F);
   e.Graphics.Transform = matrix;              // 浮動水平線の初期化
   for (int j = 0;j < Hidden_NR; j++){ YMax[j] = -1E20; YMin[j] = 1E20;}
   double XO=80; double YO=100;                // 表示始点位置
   for(int k=0;k<numY;k++)
   {  可視フラグ=false;
      for(int j=0;j<numX-1;j++)                // X 軸方向描画
      {  int p1 = setDrawPos(j, k); int p2 = setDrawPos(j + 1, k);
         double H  = 高さ[j,k];
         double DH = (高さ[j + 1,k] - 高さ[j, k])
                     * Hidden_dlx / Hidden_dxCosA;
         for(int p = p1; p <= p2; p++)         // 補間
         {  double PH = (double)(p-p1); double fp = H + DH * PH;
            double px = setPx(j, k, XO, PH);
            double py = setPy(j, k, YO, PH, fp);
            if((j<numX-2 && p<p2) || (j == numX-2))
             可視フラグ=Hidden_Draw(e, pen, px, py, p,可視フラグ, true);
         }
      }
      for(int j=0;j<numX && k<numY-1;j++)      // Y 軸方向描画
      {  可視フラグ=false;
         int p1 = setDrawPos(j, k); int p2 = setDrawPos(j, k + 1);
         double H  = 高さ[j,k];
         double DH = (高さ[j, k + 1] - 高さ[j,k])
                     * Hidden_dlx / Hidden_dyCosB;
         for(int p = p1; p >= p2; p--)         // 補間
         {  double PH = (double)(p - p1); double fp = H - DH * PH;
            double px = setPx(j, k, XO, PH);
            double py = setPy(j, k, YO, PH, fp);
            可視フラグ=Hidden_Draw(e, pen, px, py, p,可視フラグ, p!=p2)
         }
      }
   }
 }
```

【変換マトリックス設定】

```
private void window(double X1, double Y1, double X2, double Y2)
{ float W  = this.ClientSize.Width;
  float H  = this.ClientSize.Height;
  float SX = W / ((float)(X2 - X1));
  float SY = H / ((float)(Y2 - Y1));
  matrix.Scale(SX,SY);
  matrix.Translate(-(float)X1, -(float)Y1);
}
```

【各変数の初期化】

高さ［j,k］の設定値を変えて，色々な３次元データを表示してみましょう。

```
  private void Form1_Load(object sender, System.EventArgs e)
  {
    double DNX2 = ((double)numX) / 2;
    double DNY2 = ((double)numY) / 2;
    double X,Y,R,fxy;
    // 高さデータの設定
    for(int j = 0; j < numX; j++)
    {
      X = 0.3 * ((double)j - DNX2);
      for(int k = 0; k < numY; k++)
      {
          Y = 0.3 * ((double)k - DNY2);
          R = Math.Sqrt(X * X + Y * Y);
          if(R == 0.0) fxy = 1.0; else fxy = Math.Sin(R) / R;
          高さ[j,k] = 40.0 * fxy;
       }
      }
      // 表示用パラメータの設定
      Hidden_dlx     = 0.1;
      Hidden_alpha   = Math.PI/12;
      Hidden_beta    = Math.PI/8;
      Hidden_dx      = 2; Hidden_dy = 1.4
      // 計算に用いる値の設定
      Hidden_dxCosA  = Hidden_dx  * Math.Cos(Hidden_alpha);
      Hidden_dyCosB  = Hidden_dx  * Math.Cos(Hidden_beta);
      Hidden_dxSinA  = Hidden_dx  * Math.Sin(Hidden_alpha);
      Hidden_dySinB  = Hidden_dx  * Math.Sin(Hidden_beta);
      Hidden_Xlen    = (numX-1)   * Hidden_dxCosA;
      Hidden_Ylen    = (numY-1)   * Hidden_dyCosB;
      Hidden_dlxTanA = Hidden_dlx * Math.Tan(Hidden_alpha);
      Hidden_dlxTanB = Hidden_dlx * Math.Tan(Hidden_beta);
      Hidden_NR      = (int)((Hidden_Xlen+Hidden_Ylen)/Hidden_dlx)+1;
      // 表示座標マトリックスの設定
      window(-10, 200, 200, 60);
  }
```