岡本安晴　2001.2.2；2001.8

関数の等高線図[1]

図１のような関数の等高線図の描き方について説明する。

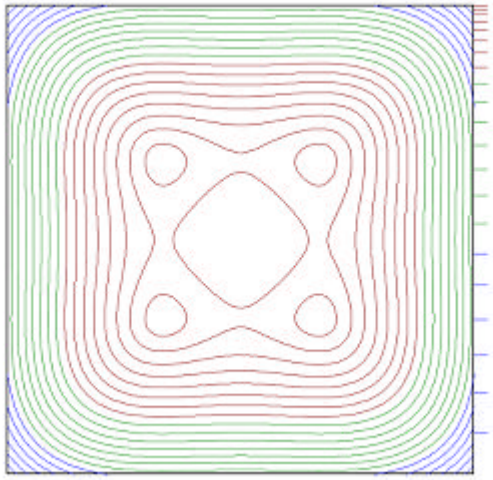

図１　関数 $f(x,y)$ の等高線図

図１は関数

$$z = f(x,y) = -x^4 - y^4 + 2(x^2 + y^2)$$

の等高線図を、手続き DrawContourV によって描いたものである（リスト１）。

リスト１　ユニット UCkDrawContV.pas の一部

```
function f1( x, y : Extended ) : Extended; //  等高線を引く曲面の関数
  begin
       f1:=x*x*x-3*x*y+y*y*y;              //  中程で密に設定
  end;

function f2( x, y : Extended ) : Extended; //  等高線を引く曲面の関数
  begin
       f2:=sqr(sqr(x))+sqr(sqr(y))-2*(sqr(x)+sqr(y)); //  下で密に設定
  end;
```

[1] 本解説は、TRY！PC、2000 年４月号「Delphi による数値計算プログラミングのすすめ：第２回 等高線図を描く」の原稿をもとにしたものである。

```
function f3( x, y : Extended ) : Extended; //  等高線を引く曲面の関数
  begin
       f3:=-sqr(sqr(x))-sqr(sqr(y))+2*(sqr(x)+sqr(y)); // 上で密に設定
  end;

procedure TForm1.DrawF1ButtonClick(Sender: TObject);
begin
          ExitButton.SetFocus;  UpDate;

          DrawContourV( f1, -3.0, 3.0, -3.0, 3.0 );
end;

procedure TForm1.DrawF2ButtonClick(Sender: TObject);
begin
          ExitButton.SetFocus;  UpDate;

          DrawContourV( f2, -3.0, 3.0, -3.0, 3.0 );
end;

procedure TForm1.DrawF3ButtonClick(Sender: TObject);
begin
          ExitButton.SetFocus;  UpDate;

          DrawContourV( f3, -3.0, 3.0, -3.0, 3.0 );
end;
```

関数 f3 を

```
function f3( x, y : Extended ) : Extended;
  begin
      f3:=-sqr(sqr(x))-sqr(sqr(y))+2*(sqr(x)+sqr(y));
  end;
```

のように宣言して、手続き DrawContourV を f3 を第１パラメータとして

```
DrawContourV( f3, -3.0, 3.0, -3.0, 3.0 );
```

のように呼出すと図１のような等高線を描くことができる。

等高線の描き方

図１の等高線は、次のような方法(1)で描かれている。

まず、関数値は、立体図の描画の場合と同じように、変数の変域を格子で区切って格子点における値を計算しておく。その後の計算においては、この格子点での値を使って計算を行う。格子点での値を使うことにより、複雑な関数であっても、その関数値の計算は最初の格子点での値を求めるときだけで済む。

次に、等高線の高さを決める。その値をcとすると、変数xとyの変域から$c = f(x, y)$となる点(x, y)を探す。その点から関数値がcである点を順番になぞっていくが、それを次のように格子点での関数値を使って行う。

いま、xの変域が$[a,b]$、yの変域が$[c,d]$であるとする。ここで、$[a,b]$および$[c,d]$は領域$a \le x \le b$および$c \le y \le d$を表わす。集合で表わすと$[a,b] = \{x \mid a \le x \le b\}$および$[c,d] = \{y \mid c \le y \le d\}$となります。これらの変域を、$a = x_0$、・・・、$x_n = b$、および$c = y_0$、・・・、$y_n = d$と、それぞれ点$x_i$および点$y_j$によって等分割して、矩形$[a,b] \times [c,d]$が格子状に分割されているものとする。格子状に分割された小矩形の辺の上で関数値がcになっているかどうかを調べる。

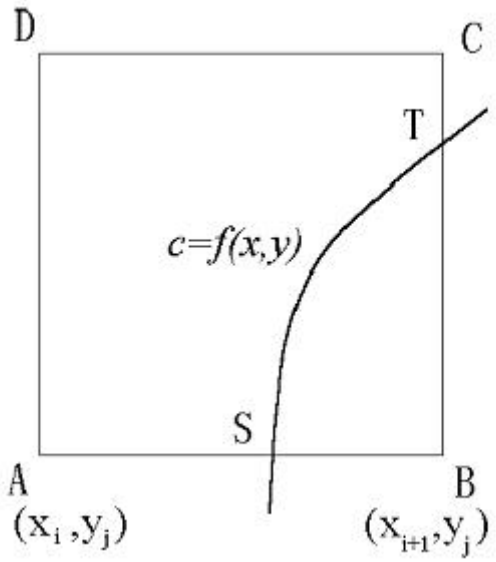

図２　矩形 ABCD 内の等高線$c = f(x, y)$

いま、隣接する格子点(x_i, y_j)と(x_{i+1}, y_j)をAとBで表わし（図２）、点 A と点 B の間で関数値が c であるとする。このとき、格子点の間隔が十分に小さく取られているならば、

$$f(x_i, y_j) < c < f(x_{i+1}, y_j) \quad \text{あるいは} \quad f(x_i, y_j) > c > f(x_{i+1}, y_j)$$

が成り立っていると考えられる。

すなわち、

$$(f(x_i, y_j) - c)(f(x_{i+1}, y_j) - c) < 0 \qquad (1)$$

が成り立つとき、等高線$c = f(x, y)$は点(x_i, y_j)と(x_i, y_{j+1})の間を通っていると考えられ

る。

辺 AB を通って矩形 ABCD の内部に入った等高線がどの辺を通って出て行くのか、その出て行く辺も各辺の端点での関数値と等高線上での関数値 c との差の積（式（1））を調べることによって求めることができる。図２の場合は、辺 BC を通っているので、点 B と C での関数値 $f(B)$ と $f(C)$ の間に

$$(f(B)-c)(f(C)-c)<0$$

が成り立っている。

辺上の等高線の通過点は、辺上における関数値の変化を１次式で近似して求める。すなわち、辺 AB 上の等高線の通過点 S、および BC 上の通過点 T は、辺の端点での関数値と等高線上での関数値 c との差で次のように内分して求める。

$$\frac{\overline{AS}}{\overline{SB}}=\frac{|f(A)-c|}{|c-f(B)|}、\qquad \frac{\overline{BT}}{\overline{TC}}=\frac{|f(B)-c|}{|c-f(C)|}$$

すなわち、点 S を辺 AB 上に $\overline{AS}:\overline{SB}=|f(A)-c|:|c-f(B)|$ となるようにとる。点 T についても同様に、辺 BC 上に $\overline{BT}:\overline{TC}=|f(B)-c|:|c-f(C)|$ となるようにとる。

点 S と点 T を直線で結んで矩形 ABCD での等高線とする。直線で結んでも、格子点の間隔を十分に小さくとっておけば、見かけ上滑らかな等高線を得ることができる。

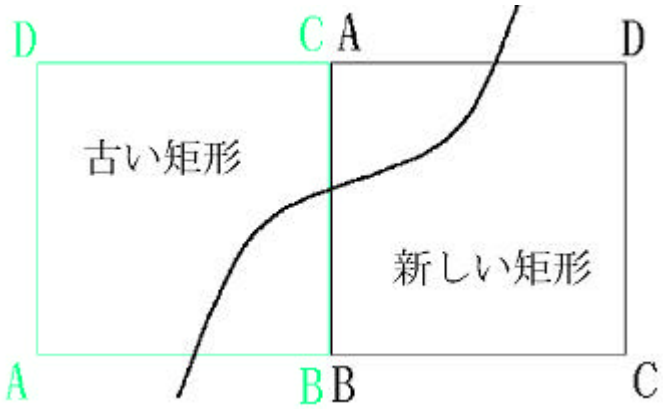

図３　新しい矩形で次の等高線を求める

次の通過点は、新しく求めた通過点 T を含む辺を改めて辺 AB として上の手続きを繰り返す。、上の場合では、点 C と B を改めて点 A と B とおいて矩形 ABCD を考える（図３）。

このように矩形 ABCD を曲線 $f(x,y)=c$ 上に沿って動かしながら等高線を描いていくと、変数の変域の端に突き当たるか、あるいは既に等高線を引いた矩形に戻るので、そこでその等高線の描画を終了する。

ユニットファイル UDrawContV.pas

上の方法で図１のような等高線を描く手続きが DrawContourV である。これはユニットファイル UDrawContV.pas に宣言されている。

手続き DrawContourV のヘッダーは、次のようになっている。

```
procedure DrawContourV( CheckF : TFunc;
                        pxmin, pxmax, pymin, pymax : Extended );
```

第１パラメータに等高線図を描く関数、第２、３パラメータに関数の第１変数の変域の下限と上限、第４、５パラメータに関数の第２変数の変域の下限と上限を設定する。

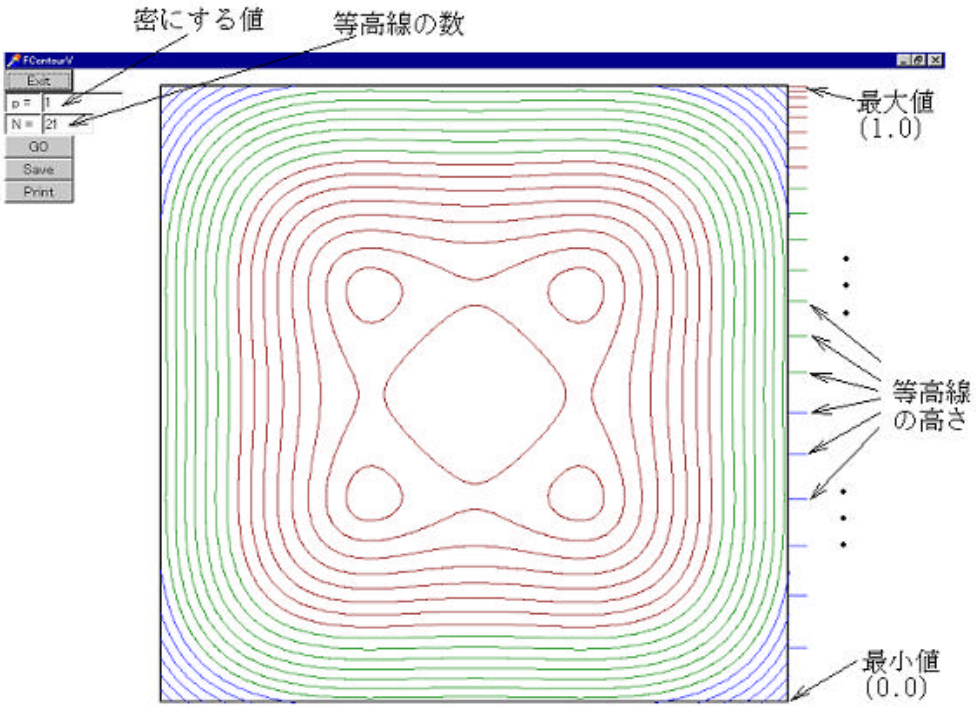

図４　等高線を密にとる値と等高線の数の指定

手続き DrawContourV では、等高線の高さｃの値が、関数の最小値と最大値の間の指定された値の近くで等高線が密になるように取られる。この等高線を密にとる値は、関数の最大値を 1.0、最小値を 0.0 としたときの 0.0 から 1.0 の間の相対値で指定する。等高線の描画画面の左上に Edit コンポーネントが表示されるが（図４）、ｐ＝の値が等高線が密になる値、Ｎ＝の値が等高線の数である。これらの値を設定してからＧＯボタンをクリックすると、設定した値による再描画が行われる。図１は、「Draw F3」ボタンのクリックによって表示される画面において、ｐ＝１と設定して「GO」ボタンのクリックを行い再描画す

ると得られる。

Save ボタンをクリックすると描画図形が*.bmp ファイルとして保存される。

Print ボタンをクリックすると描画図形がプリンタに出力される。

プロジェクト PCkDrawContV.dpr は、ユニット UDrawContV を使用して３つのタイプの関数 f1、f2、f3 の等高線図を描くものである。それぞれ、等高線が中程で密、最小値で密、最大値で密のときに関数の曲面の様子が良く分かるものである。このプロジェクトのフォームは、図５のように用意されている。

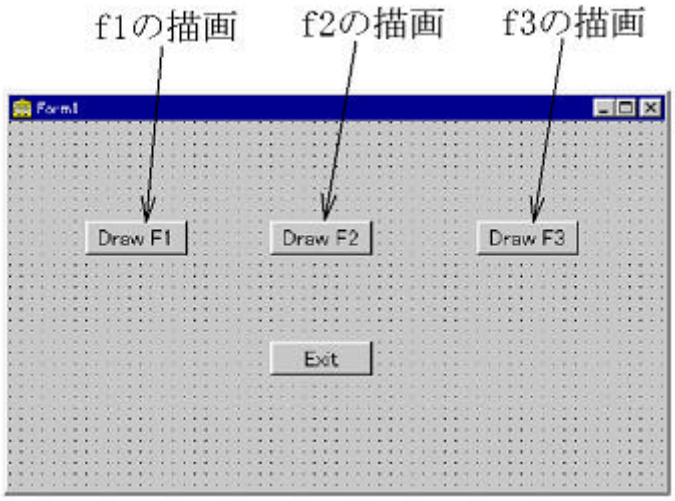

図５ プロジェクト PCkDrawContV 用のフォーム

３つのボタン「Draw F1」、「Draw F2」、「Draw F3」いずれかのクリックにより、それぞれ対応する関数 f1、f2、f3 を第１パラメータとして手続き DrawContourV が呼び出される（リスト１）。

ユニット UDrawContE.pas

等高線の高さの間隔を等間隔にとる場合は、ユニットファイル UDrawContE.pas の手続き DrawContourE を用いる。

手続き DrawContourE のヘッダーは、次のようになっている。

```
procedure DrawContourE( CheckF : TFunc;
                        pxmin, pxmax, pymin, pymax : Extended );
```

第１パラメータに等高線図を描く関数、第２、３パラメータに関数の第１変数の変域の下限と上限、第４、５パラメータに関数の第２変数の変域の下限と上限を設定する。

関数 $f(x,y)=x^2-2y^2$ の等高線を、高さを等間隔にとって変域、$-3.0\le x\le 3.0$、$-3.0\le y\le 3.0$、で描くときは、関数宣言を

```
function CheckF( x, y : Extended ) : Extended;
  begin
          CheckF:=x*x-2*y*y;
  end;
```

としておいて、手続き DrawContourE を

```
DrawContourE( CheckF, -3.0, 3.0, -3.0, 3.0 );
```

と呼出す。このときの図は、図６のようになる。

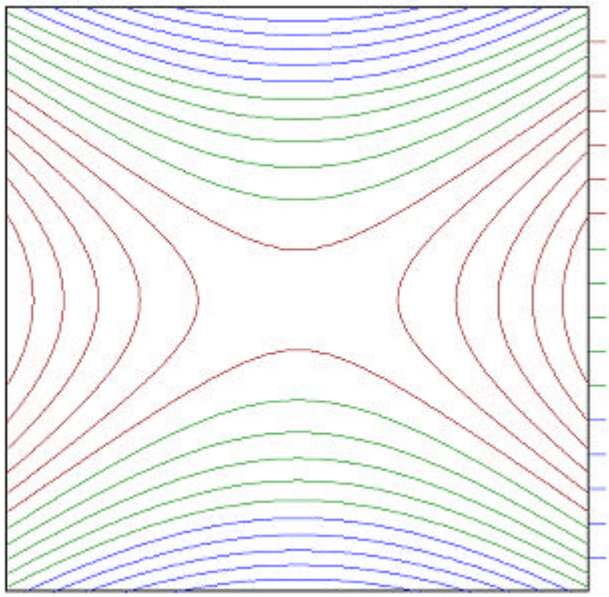

図６ ユニット UDrawContE.pas の手続き
DrawContourE による等高線の描画例

手続き DrawContourE によって等高線を描くサンプルプログラム PCkDrawContE.dpr のユニット UCkDrawContE のフォームは、図７のように用意されている。実行後表示されるフォームの GO ボタンのクリック（リスト２）で、図６のような等高線が描かれる。

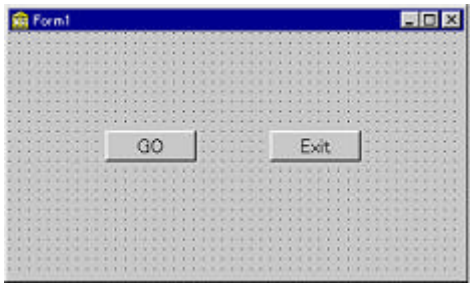

図７　ユニット UCkDrawContE.pas のフォーム

リスト２　ユニット UCkDrawContourE の一部

```
uses
  UDrawContE;

procedure TForm1.ExitButtonClick(Sender: TObject);
begin
              Close;
end;

function CheckF( x, y : Extended ) : Extended; //  等高線を描く曲面の関数
  begin
            CheckF:=x*x-2*y*y;
  end;

procedure TForm1.GOButtonClick(Sender: TObject);
begin
       ExitButton.SetFocus; UpDate;

       DrawContourE( CheckF, -3.0, 3.0, -3.0, 3.0 );
end;
```

等間隔の関数値に対して等高線を描く手続き DrawContE に等高線の数を再設定する機能を加えたものを同じ名前でユニットファイル UDrawContENvar.pas に用意した。等高線の数の設定後、「Re-Draw」ボタンをクリックすると、再設定した数で再描画が行われる。ユニット UDrawContENvar の手続き DrawContE を用いるサンプルプログラムは PCkDrawContENvar.dpr である。

参 考 文 献

（１）森 正武　「曲線と曲面」、Pp.150、教育出版株式会社、1974.