1.6 グラフィックス表示（入門編その３）　　ver. 0.1

グラフィックスの入門編を締めくくるにあたって、User's Guide にあるような、３次元の Mathematica ライクなグラフや、グラフ画面の分割表示、グラフのオーバーラップなどおもしろいトピックスを取りあげたいと思います。プログラムや統計計量経済学などには直接は関係のないかもしれませんが、GAUSS に興味を持つには効果的でしょう。

その前に、surface グラフや contour グラフで頻繁に利用される GAUSS に特有な行列演算の規則を説明しておきましょう。GAUSS では、例えば、

$$\begin{bmatrix}1 & 2 & 3 & 4\end{bmatrix}+\begin{bmatrix}1\\2\\3\\4\end{bmatrix}=\begin{bmatrix}1 & 2 & 3 & 4\\1 & 2 & 3 & 4\\1 & 2 & 3 & 4\\1 & 2 & 3 & 4\end{bmatrix}+\begin{bmatrix}1 & 1 & 1 & 1\\2 & 2 & 2 & 2\\3 & 3 & 3 & 3\\4 & 4 & 4 & 4\end{bmatrix}=\begin{bmatrix}2 & 3 & 4 & 5\\3 & 4 & 5 & 6\\4 & 5 & 6 & 7\\5 & 6 & 7 & 8\end{bmatrix}$$

となります。１×４行列と４×１行列は、掛け合わすことはできても、足し合わせることはできないはずです。しかしながら、１×ＭとＮ×１の２つの行列を＋または－のマークでつないでやると、GAUSS では、ジオメトリーを配慮して、Ｍ×Ｎの行列が作られ、それぞれ同じ行ベクトルまたは列ベクトルが上のように繰りかえされます。同様に、

$$\begin{bmatrix}1 & 2 & 3 & 4\end{bmatrix}+\begin{bmatrix}0\\0\\0\\0\end{bmatrix}=\begin{bmatrix}1 & 2 & 3 & 4\\1 & 2 & 3 & 4\\1 & 2 & 3 & 4\\1 & 2 & 3 & 4\end{bmatrix}+\begin{bmatrix}0 & 0 & 0 & 0\\0 & 0 & 0 & 0\\0 & 0 & 0 & 0\\0 & 0 & 0 & 0\end{bmatrix}=\begin{bmatrix}1 & 2 & 3 & 4\\1 & 2 & 3 & 4\\1 & 2 & 3 & 4\\1 & 2 & 3 & 4\end{bmatrix}\equiv x$$

$$\begin{bmatrix}1\\2\\3\\4\end{bmatrix}+\begin{bmatrix}0 & 0 & 0 & 0\end{bmatrix}=\begin{bmatrix}1 & 1 & 1 & 1\\2 & 2 & 2 & 2\\3 & 3 & 3 & 3\\4 & 4 & 4 & 4\end{bmatrix}+\begin{bmatrix}0 & 0 & 0 & 0\\0 & 0 & 0 & 0\\0 & 0 & 0 & 0\\0 & 0 & 0 & 0\end{bmatrix}=\begin{bmatrix}1 & 1 & 1 & 1\\2 & 2 & 2 & 2\\3 & 3 & 3 & 3\\4 & 4 & 4 & 4\end{bmatrix}\equiv y$$

となることを利用して、ｘｙプレーンの上に３Ｄグラフとその等高線グラフを描きます。

プログラム

```
new;  cls;
library pgraph;
graphset;
x=seqa(1,1,51)';  y=seqa(1,1,51);
xm=x+zeros(51,1);  ym=y+zeros(1,51);
z=xm^(2/3).*ym^(2/3);
surface(x,y,z);
contour(x,y,z);
```

２つのグラフ表示、うまくできたでしょうか。ミクロ経済学でででてくる基本的なグラフが描けたと思います。ｘに１×５１の行（横）ベクトルを作り、１から５１までのシークエンスを代入します。ｙには５１×１の列（縦）ベクトルを作り、１から５１までのシークエンスを入れます。これらｘとｙはグラフのｘ座標とｙ座標になり surface グラフと contour グラフのｘとｙのところに入ります。ｚには、上のジオメトリーに対応した座標行列の作り方によって、５１×５１の行列を作ったあとに、xm のそれぞれの要素を２/３乗し ym のそれぞれの要素を２/３乗して、その両方の行列の要素を要素ごとにかけ合わせたものをｚに代入しています。その５１×５１行列を surface グラフと contour グラフのｚのところに代入して、それぞれのグラフを描いています。指数倍のよころの演算には＾のマークが、要素ごとの行列の積には.*が使われています。（*のマークだけでは行列対行列の積になってしまいます。要素対要素という意味でピリオドのマークが必要。）ｘは手前に横に座標をとるということで、Ｍ×１の横ベクトルが、ｙは奥に縦に座標をとるということで、１×Ｎの縦ベクトルが使われ、そのプレーン上に、Ｍ×Ｎの高さを表す行列データがｚにきています。ただし、contour グラフは等高線を示す性格上、縦横の座標の数はともに奇数でないとエラーが発生します。偶数個の座標は不可です。

ポイント　surface **グラフは** 3D **グラフを描き、**surface(x,y,z);

ｘは横軸（Ｍ×１の行ベクトル）、ｙは縦軸（１×Ｎの列ベクトル）、

ｚは高さを表すＭ×Ｎ行列、それぞれの座標においての高さ。

Contour **グラフは等高線グラフを描き、**contour(x,y,z);

ｘは横軸（Ｍ×１の行ベクトル）、ｙは縦軸（１×Ｎの列ベクトル）、

ｚは高さを表すＭ×Ｎ行列、それぞれの座標においての高さ。

ただし、Ｍ、Ｎともに奇数。

ポイント　＾のマークは指数乗の演算記号　　例　２の８乗は、2^8

．＊で行列の要素ごとの積の演算。ピリオドは要素ごとの演算を表す。

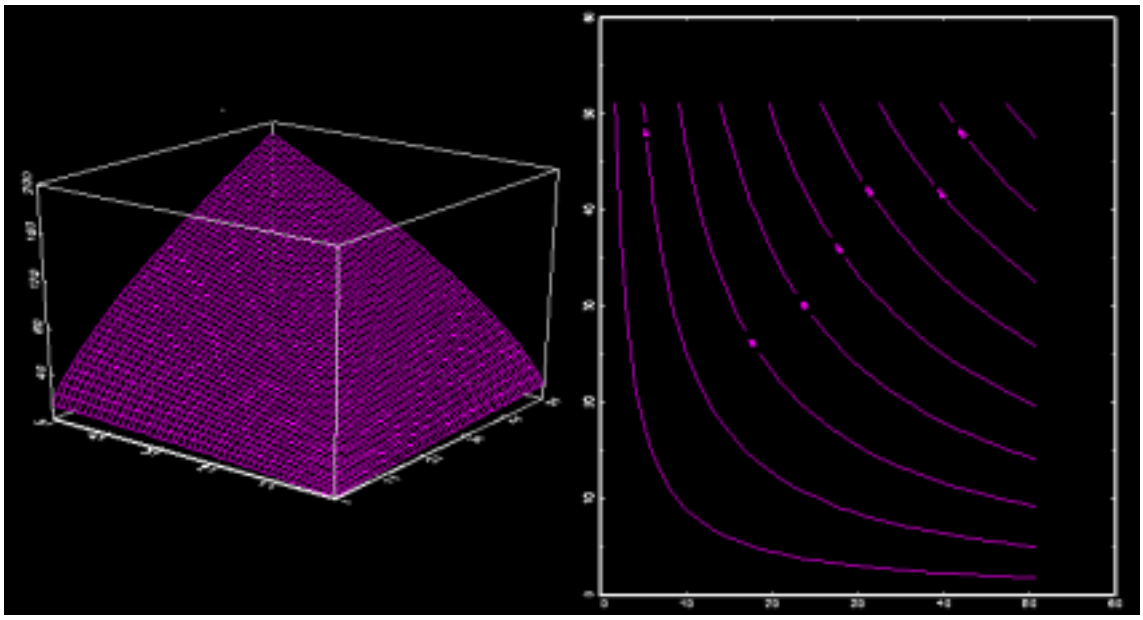
このままのグラフでは最初のグラフがすぐに消えて最後のグラフが上にきてしまうので、ここで、画面を左右に２分割して、まず 3D の surface グラフを、その横に等高線 contour グラフを描くように工夫をしてみます。

プログラム

```
new;  cls;
library pgraph;
graphset;
x=seqa(1,1,51)';  y=seqa(1,1,51);
xm=x+zeros(51,1);  ym=y+zeros(1,51);
z=xm^(2/3).*ym^(2/3);
begwind;
window(1,2,0);
```

```
setwind(1);
    surface(x,y,z);
setwind(2);          @ or nextwind; @
    contour(x,y,z);
endwind;
```

グラフ表示



ここでは GAUSS でパネルグラフというグラフを分割したり重ねたりする手法を用いるときの手順を説明しましょう。まず、begwind でパネル window を始めます(begin)。そして、window コマンドで、パネルウインドウを分割します。この場合、window(1,2,0)とは、パネルウインドウを１行２列に分割し、画面の属性を非透過にするという意味です。ちなみに、すけて見える透過にする場合は最終引数を１にしますが、この場合、グラフは重なり合わないので、０でも１でもどちらでもかまいません。縦に、上下に分割するときには、２行１列ですから window(2,1,0)となります。横３縦２に分割するには、window(2,3,0)となります。setwind(1)は、いまから描くグラフを１番目のグリッドに描くということ。setwind(2)は２番目のグリッドに描くということです。setwind(2)以降は、nextwind で番号なしですべて順送りにすることも可能です。なお、グリッドの番号は、最初の行の左から右へ、そして次の行の左から右へ数えられます。たとえば、グリッドがいくつかあって、setwind(3)とすると３番目のグリッドから描き始めることができます。setwind(1)の１は必ずしも１である必要はなく、グラフ描くグリッドの番号を表しています。そして、最後に、endwind でパネルウインドウ機能を終了(end)することによって、描画が始まります。

ポイント　画面を M 行N列に分割してグラフを描くには、

```
begwind;
window(M, N, 0);
setwind(1);
.............
.............
setwind(2);
.............
.............
setwind(3);
.
.
endwind;
```

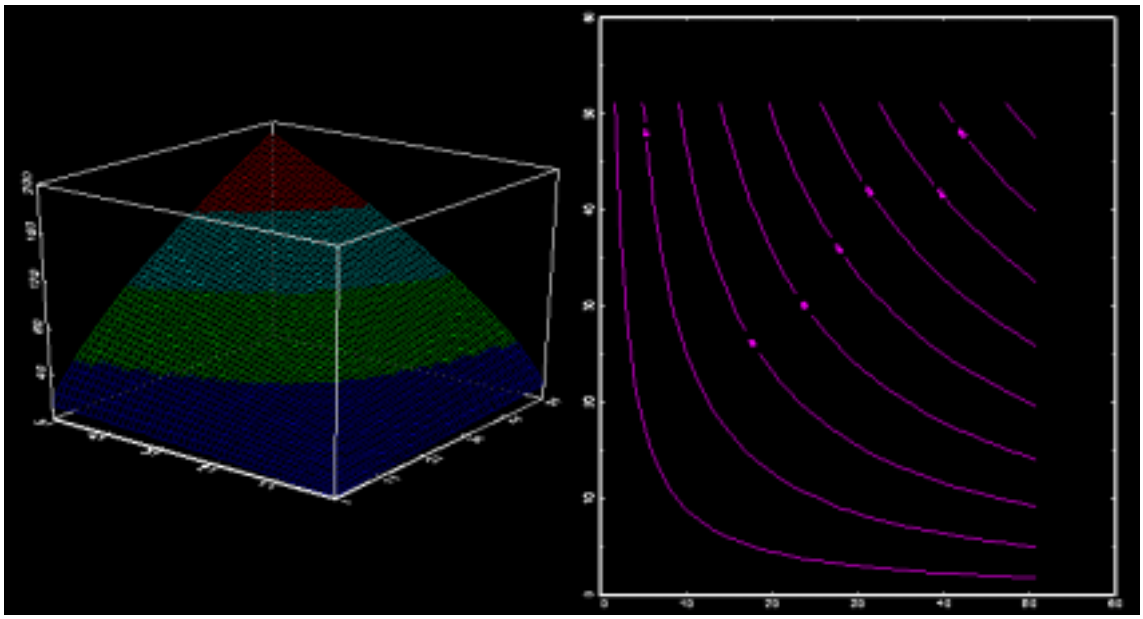
ただし setwind(2)**以降は、順にグラフを描くのであれば、**nextwind;**でも可**

すこしもの足りないので、グラフに色を変えてつけてみましょう。以前やったように、pgraph のライブラリーのなかのグローバル変数の設定値を変更します。とりあえず、surface グラフのみに適当にカラーをつけてみましょう。graphset;の挿入に注意して、太字のところを変更してみてください。

プログラム

```
new;  cls;
library pgraph;
x=seqa(1,1,51)';  y=seqa(1,1,51);
xm=x+zeros(51,1);  ym=y+zeros(1,51);
z=xm^(2/3).*ym^(2/3);
begwind;
window(1,2,0);
setwind(1);
    graphset;
    _zclr={1,2,3,4};
    surface(x,y,z);
setwind(2);
    graphset;
    contour(x,y,z);
endwind;
```

グラフ表示



どうでしょうか。_pzclr で、ｚ軸の color のグローバル変数を変更してみます。_のマークは、以前と同じく、グローバル変数を表し、p は GAUSS のグラフィックライブラリー環境である Publication Quality Graphics の頭文字のｐを表します。clr は color の略。ここで、_pzclr のグローバル変数のデフォルトは、13 番 light Magenta うす赤紫色です。（これは print _pzclr;を最初の graphset;のあとに入れると、13 という数を返すことで確かめられます。）このデフォルトを、ここでは、４つの領域に高さにおいて等間隔に分割して、それぞれに色番号 1,2,3,4 の各番号を割り当てたものです。ちなみに、色番号を示しますと、

０（black 黒）　１（blue 青）　２（green 緑）　３（cyan 青緑）
４（red 赤）　５（magenta 赤紫）　６（brown 茶）　７（grey 灰色）
８（dark grey 濃い灰色）　９（light blue うす青）　１０（light green うす緑）
１１（light cyan うす青緑）　１２（light red うす赤）　１３（light magenta うす赤紫）
１４（yellow 黄）　１５（white 白）　　[Command Reference　surface の項参照]

グラフごとにグローバル変数を変更する場合には（この場合、surface だけ変更していて、contour はデフォルト値を利用している）surface や contour、xy などの前のどこかに、その都度、graphset;を置いて、グローバル変数をリセットしてやらないといけません。もし１つしか graphset;がないとすると、２つ目のグラフも同じような色になります。これは、色以外の設定、例えば、グラフのタイトルや軸の名前を設定する際にも、その都度、グローバル変数を graphset;でリセットする必要があることを意味します。２つのグラフをデフォルトとは違った設定で描こうとする場合には、例えばこの場合、surface(x,y,z)よりも前

library graph よりも後に、１つ graphset が必要ですし、surface(x,y,z)と contour(x,y,z)の間にも１つ graphset が必要になります。この他に_pzclr の設定は、単色の場合、

_pzclr=2;

（この場合、色が緑一色になる）として色を一色全部変更する方法や、ある高さごとに色を変える方法がありあります。例えば、この surface グラフの目盛りの 40,80,120,160,200 の高さの間ごとに色をかえる場合には、

_pzclr={0 1,40 2, 80 3, 120 4, 160 5};

とします。M×N行列の値を代入する形となります。１列目は区切りの座標、２列目は色の番号が入ります。ｚの値 40 までを１の色、40 以上を２の色、80 以上を３の色、120 以上を４の色、160 以上を５の色という具合に色分けできます。（ただし、最初の０のところには、何の数が入ってもかまいません。普通は、ｚのとりうる値の最小値を入れてやると、プログラムを見たり移植する人にはわかりやすいでしょう。）なお、surface グラフも contour グラフも両者ともに、この_pzclr の設定を使うことができます。もちろん、二つのグラフを違ったｚ軸の高さに応じて違った色にすることもできます。その際にも、その都度、graphset;でグローバル変数をリセットする必要があります。

ポイント　surface グラフ、contour グラフのｚ軸（高さ）の色を変更するには、

[1] _pzclr=**色番号**; **単色変更の場合**

[2] _pzclr={ **色番号**, **色番号**, **色番号**, **色番号**, …… };

自動的に色番号の個数だけの色数に区切る場合

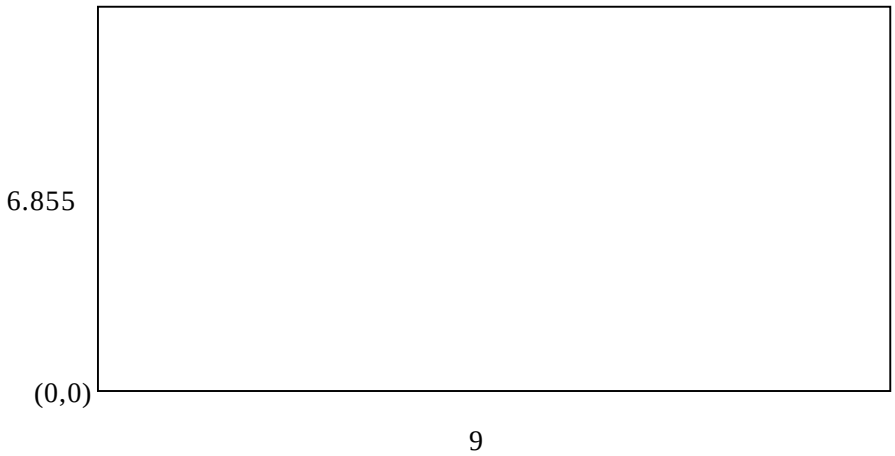
[3] _pzclr={0 **色番号**, **最初の区切り高さ　色番号**, **その次の高さ　色番号**,…};

区切りの高さを指定して色を変える場合

ポイント　グラフごとにデフォルトから設定を変える場合には、グラフを描く命令の前ごとに graphset;**でリセットが必要。**

画面を分割だけをするほかに、begwind から始まるグラフィックパネルを変更する方法の代表的なものとして、グラフをオーバーラップさせて重ね合わせたり、ずらして表示したりすることをつけ加えることも可能です。まずその前に、グラフが描かれる座標全体の大きさと、その座標のとり方を知っていなければなりません。フルサイズの座標は、横９インチ、縦 6.855 インチあり、その原点の座標は、左下のすみの値を(0,0)とします。以下の図のようなインチ単位の長さと、原点(0,0)の位置になっています。このことをよく知った上で、この値をもとに、重ねるべき（あるいは、ずらして表示する）ウインドウパネル

を作成します。

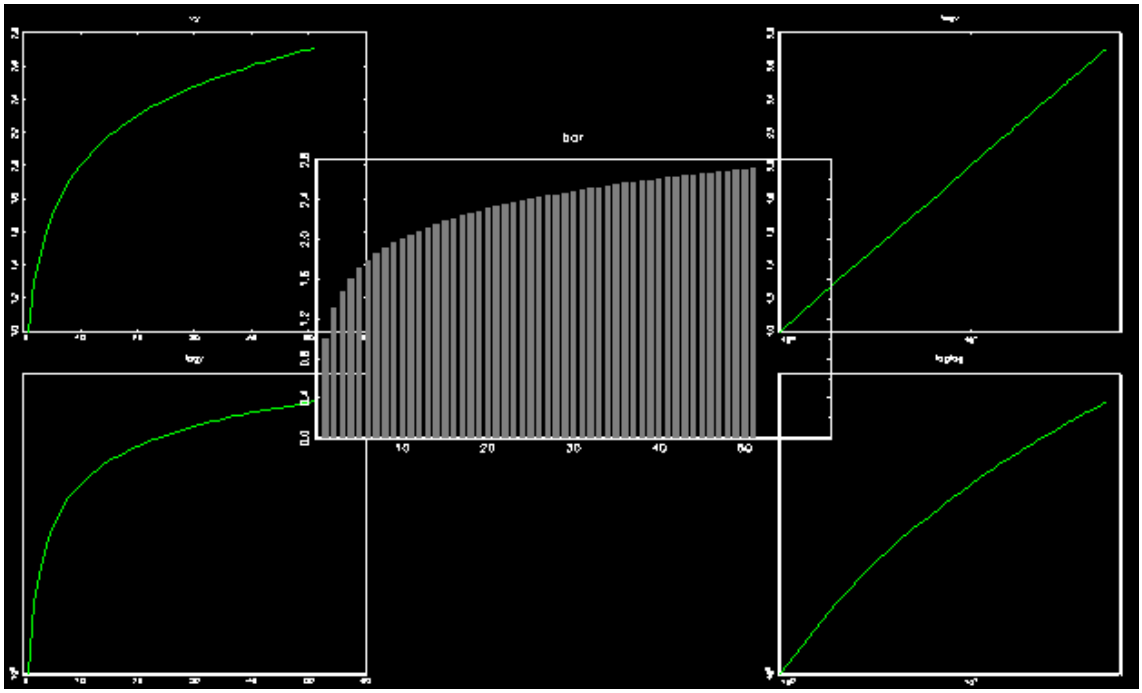


プログラム

```
new;  cls;
library pgraph;
x=seqa(1,1,51); y=1+log(x);
begwind;
window(2,3,0);
wide=9; hight=6.855;
makewind(wide/2,hight/2,wide/4,hight/3,1);
setwind(1);
    graphset;
    title("xy");
    xy(x,y);
setwind(3);
    graphset;
    title("logx");
    logx(x,y);
setwind(4);
    graphset;
    title("logy");
    logy(x,y);
setwind(6);
    graphset;
    title("loglog");
    loglog(x,y);
setwind(7);
```

```
    graphset;
    title("bar");
    bar(x,y);
endwind;
```

グラフ表示



上のグラフは、ウインドウを２×３に分割した上に、新しくウインドウを作って、透過オーバーラップさせたものです。具体的には、２×３に分割した後、四隅の位置に相当するウインドウ１番、３番、４番、６番、それに新しく作った７番にグラフを順に描いています。setwind の番号の値は、第１行の左から右へ、そしてその次の行の左から右へ、というふうにつけられていき、最後に、makewind で作ったオーバーラップさせるウインドウの番号がつきます。

このプログラムは、new; cls;の一連の手続きの後、pgraph ライブラリーを呼び出し、ｘに１から１刻みで５１個のシークエンスの列ベクトルデータを生成させ、さらに、そのデータをもとに１＋log(x)の値をｙにして、そのグラフを描いています。begwind;により、グラフィックパネルの手続きを初期化します。そして、window(2,3,0);で、そのパネルウインドウを２×３に分割します。そして、グラフィック画面の幅は９インチ、縦は 6.855 インチですから、その幅の２分の１、縦の２分の１の新しいウインドウを作り、その新しいウインドウの原点は、全体の幅の４分の１、全体の縦の３分の１のところに設定します。

```
wide=9; hight=6.855;
makewind(wide/2,hight/2,wide/4,hight/3,1);
```

それは、上のようになります。makewind の第１引数は新しいウインドウの幅の長さ、第２引数は縦の長さ、第３番目は、そのウインドウの原点のｘ座標、第４番目はそのｙ座標、そして最終第５番目の引数は、そのウインドウが透過(1)か非透過(0)かをそれぞれ表しています。もちろん、直接数値を入れてもかまいません。そして、ウインドウ番号１に xy グラフを。番号２をとばして、番号３に logx グラフを。番号４に logy グラフを。番号５をとばして番号６に loglog グラフを。そして新しく作成したウインドウ番号７に bar グラフを、それぞれタイトルラベルつきで、その都度、graphset;でリセットをしながら描いたものです。ここで、注意したいのは、makewind は常に window で画面を分割することと常に組みになっていることです。window で画面を分割だけすることはできますが、makewind 単独で、もとのグラフ１つに重ねるという作業は、そのままではできません。オーバーラップさせる makewind は、常に、window 分割の拡張機能ととらえてください。したがって、window(1,1,0)とともに makewind などとすることは不可能で、どうしても２つ以上に画面を分割した後で、さらにもう１つのウインドウを追加させることが必要です。

ポイント　ウインドウ分割に付け加えて、新しいウインドウを作成して重ねるには、
makewind(ウインドウの幅，　ウインドウの縦，　原点のｘ座標，　原点のｙ座標，　0 or1);
0:非透過　1:透過

nextwind を setwind(2)以降に代替することも可能ですが、より分かりやすいプログラム、また、より応用のきくプログラムという点からは、２番目以降も setwind(番号)で指定してやるとよいでしょう。なお、フルスクリーンのグラフに小さなグラフを重ねるという場合には、window(1,1,0)と makewind の組合せは不可です。その代わりに、setwind(番号)を応用して、あらかじめ小さくウインドウを分割して、その何番目かを setwind で指定してまずグラフを描き（そのあとの分割画面は使わない）、フルスクリーンのグラフの方を重ねる側に考えて、makewind(9,6.855,0,0,1);でフルスクリーンで、原点(0,0)のグラフとして透過オーバーラップさせることができます。

最後に、これらのグラフを保存する方法を説明しましょう。一番最初に描いた、surface グラフと contour グラフの場合、surface グラフの描画の後、すぐに contour グラフが描かれてしまって、最初の surface グラフは失われてしまいました。このようなときに、特に、グラフを保存するという作業は重要になってきます。なにもグラフを保存する指定をしない場合には、グラフは、gauss プログラムのあると同じディレクトリの graphic.tkf というファイルに一時的に保存されます。ただし、直前に描かれたグラフ１つしか graphic.tkf には保存されていません。これは、ライブラリーpgraph のなかの保存グラフに関するグローバル変数の設定が、デフォルトで graphic.tkf であるためです。したがって、グラフごとに保存するファイル名を指定する必要があります。

プログラム

```
new;  cls;
library pgraph;
x=seqa(1,1,51)';  y=seqa(1,1,51);
xm=x+zeros(51,1);  ym=y+zeros(1,51);
z=xm^(2/3).*ym^(2/3);
graphset;
    _ptek="d:surface.tkf";
    surface(x,y,z);
graphset;
    _ptek="d:contour.tkf";
    contour(x,y,z);
```

上のように、最初のプログラムを書きなおすと、１枚ずつグラフが指定するディレクトリのもとのファイル名に保存されます。グラフごとに graphset;でリセットしながら、_ptek というグローバル変数にファイル名の文字列をディレクトリつきで代入すると、複数のグラフが保存されます。tkf とは、Tektronix format というグラフィックフォーマットの一種の名前の略です。後で、GAUSS の外から開いて見ることができます。

ポイント　グラフの保存は　_ptek="ディレクトリつきファイル名";
ただしファイル名の拡張子は、tfk であること。
例　_ptek="d:/gauss/abcde.tkf";

なお、指定するファイル名は、tkf というドット以下の拡張子でなければならないきまりになっています。アウトプットのファイルへの出力や行列の fmt 形式のファイルへの出力と同様に、ドライブ名やディレクトリ名をファイル名の前につけて指定しなければなりません。_ptek="abcde.tkf";とだけ指定すると、GAUSS のプログラムがあるディレクトリ下に abcde.tkf というファイル名が作成されてしまいます。GAUSS プログラムのファイル群を誤って消したり編集してしまわないためにも、ディレクトリは、変更して自分のプログラムなりグラフィックファイルなりを保存する習慣をつけましょう。なお GAUSS は自動的にディレクトリを作成することはしませんので、あらかじめ作成しておきましょう。
tkf 形式のファイルを作りたくない場合には_ptek="";として空のファイル名を指定すればファイルは作成されません。