1.8 グラフィックスの実践(プレゼンテーションのために)　ver.0.1

この章では、GAUSS をファイルコンバートやプレゼンテーションといった実践的な事例に適用することを試みます。初心者はこの章をスキップして、自分でもプログラムを書けるようになってから後で戻ってきてください。

グラフィックファイルのコンバート

GAUSS で図を描くと TKF Viewer が起動して、テンポラリーなファイル graphic.tkf に現在描かれている図が入っています。このテンポラリーファイルを Postscript ファイルに変換するには、プログラム中に tkf2ps または tkf2eps を呼び出してやれば、プログラムの一連の流れとして ps ファイルまたは eps ファイルが自動コンバートされて作成されます。

プログラム

```
new; cls;
library pgraph;
graphset;
x=seqa(1,1,200); y=rndu(200,1);
xy(x,y);
call tkf2ps("graphic.tkf","d:/mine1.ps");
/* call tkf2eps(“graphic.tkf”,”d:/mine1.eps”); */
```

上のように、ランダムな 200 個の軌跡を１から 200 のシークエンスに対して作図するには上のように、まず new; cls;の後に pgraph のライブラリを呼び出して、graphset でそのグローバル変数をリセットしてやります。そして、例えばｘｙ関数によって描画させます。そして、その直後に第１引数にコンバート元の”graphic.tkf”を第２引数にコンバート先のファイル名”d:mine1.ps”としてやり、関数 tkf2ps を呼び出します。この「２」というのは英語の to の意味です。これにより、tkf ファイルという特殊なファイルとして保存せずとも自動的に Postscript 形式でファイルが保存できます。なお、EPS 方式でも保存は可能ですがバージョンとＯＳの関係でバグが報告されていることもあるので、うまくいかない場合には、上の注釈で無効にしてあるところを実行するのではなくて、tkf2ps を実行して Postscript ファイルを一度作成してから、そこから EPS 形式に Ghostview など各種ソフトウエアを使って変換するようにして TeX 関連のソフトウエア、例えば Scientific Workplace 等に読み込んでみてください。なお、上のプログラムで、

```
call tkf2ps("graphic.tkf","d:/mine1.ps");
```

または call tkf2eps(“graphic.tkf”,”d:/mine1.eps”);

の関数の第１引数”graphic.tkf”に相当する部分には、自分で tkf 形式で

```
_ptek=”filename.tkf”;
```

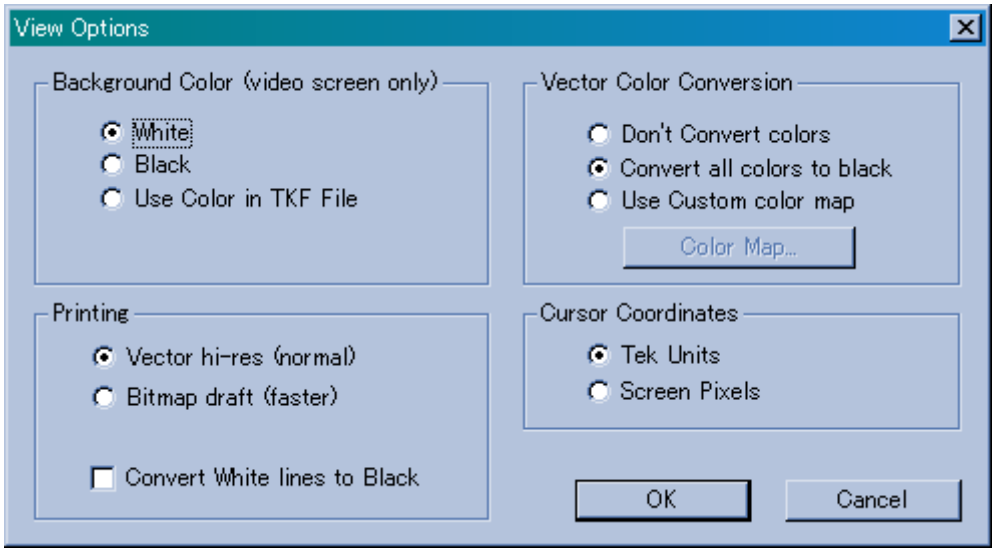
として保存したファイルを、関数の第 1 引数に代入して、変換することもできます。

グラフィックスの白黒化

通常 GAUSS のグラフは黒地に枠は白、グラフの軌跡は緑で描かれます。この色を尊重することが大切なことは、古きよき時代の PC の世界を知っている方にはどうしてなのかが歴史をもってわかることでしょう。しかしながら、その他の一般的な計量ソフトのように白地に描くこともプレゼンテーション上必要になってくるかもしれません。また、白地に黒だけを使ってモノクロで印刷上表示させたいこともあります。そんな場合には、機械的に TKF Viewer の設定を変更してしまう方法と、プログラム的に出力方法を変える方法の２つが考えられます。

【機械的な方法】

下のように、一度なにがしかのグラフを描いてから、TFK Viewer の中から View Options のところの Use Color in TKF File を White に変更し、その隣の Don't convert colors を Convert all colors to black に合わせて変更します。OK を押すとそれ以降あらたに設定を変更するまでは白黒でグラフが描かれ続けます。



【プログラム的方法】

機械的ではなくて、プログラムとして見かけの黒地はそのままで、コンバートする時にファイルのみを白黒にするには、次のようなグローバル変数設定をします。

プログラム

```
new; cls;
library pgraph;
graphset;
x=seqa(1,1,200); y=rndu(200,1);
xy(x,y);
_pqg_ps_Monochrome = 1;
call tkf2ps("graphic.tkf","d:/mine1.ps");
```

ポストスクリプトの場合に色がついて表示されるグラフをすべて黒にするためには、先ほどの Postscript ファイルを作成する関数 tkf2ps を呼び出す前に_pqg_ps_Monochrome = 1; を挿入します。これにより、緑色の軌跡は黒色に変ります。

グラフィックスの各種ファイル形式読み出し

今度は、何かのファイルをコンバートするのではなくて、graphprt という関数を使って直接、各種ファイル形式で読み出します。

プログラム

```
new; cls;
library pgraph;
graphset;
x=seqa(1,1,200); y=rndu(200,1);
xy(x,y);
graphprt("-c=5 -cf=d:/mygauss/mine2.bmp");
rerun;
/*  graphprt("-c=1 -cf=d:/mygauss/mine2.eps "); */
/*  rerun;                                      */
```

上の方法は、Windows の bmp 形式のファイルに読み出す方法です。関数 graphprt の丸括弧の中味は、パラメータも含めて、引用符に包まれた「文字列」として代入します。

-c=1　　eps 形式

-c=5　　Windows の bmp 形式

となり、-cf=のオプションのあとにはファイル名がきます。上の例の設定では、ｄドライブの mygauss というフォルダの mine2.bmp というファイルにグラフが読み出され保存されます。ファイル名には拡張子を正確につけた方が、ファイル認識上、より無難です。

注意：本書の説明では、初心者でも分かりやすいように、一貫してディレクトリを表すのにスラッシュ”/”を用いています。もちろん正式な方法であるバックスラッシュのアジアファンと表記の”¥”を使うことができるのは言ううまでもありません。しかしながら、バックスラッシュを用いて文字列としてディレクトリを表記する場合、その区切りのバックスラッシュは２つずつ用いる必要があります。例えば、上の場合、

```
graphprt("-c=5 -cf=d:¥¥mygauss¥¥mine2.bmp");
```

というふうにダブルにする必要があります。これは文字列としてファイル名をディレクトリつきで指定する際にあらゆる局面で共通してなされる方法です。繰り返しますが、この場合の¥のマークはバックスラッシュを打って GAUSS ではバックスラッシュとして表記されますが、カットアンドペーストでアジアフォントシステムの上でワープロの上では

￥と表示されるものです。スラッシュの場合はシングルで問題ありません。また、バージョン５からは多少のファイルハンドリングに変更があり、トップにある場合でも

d:/mine2.bmp　　または　　d:￥￥mine2.bmp

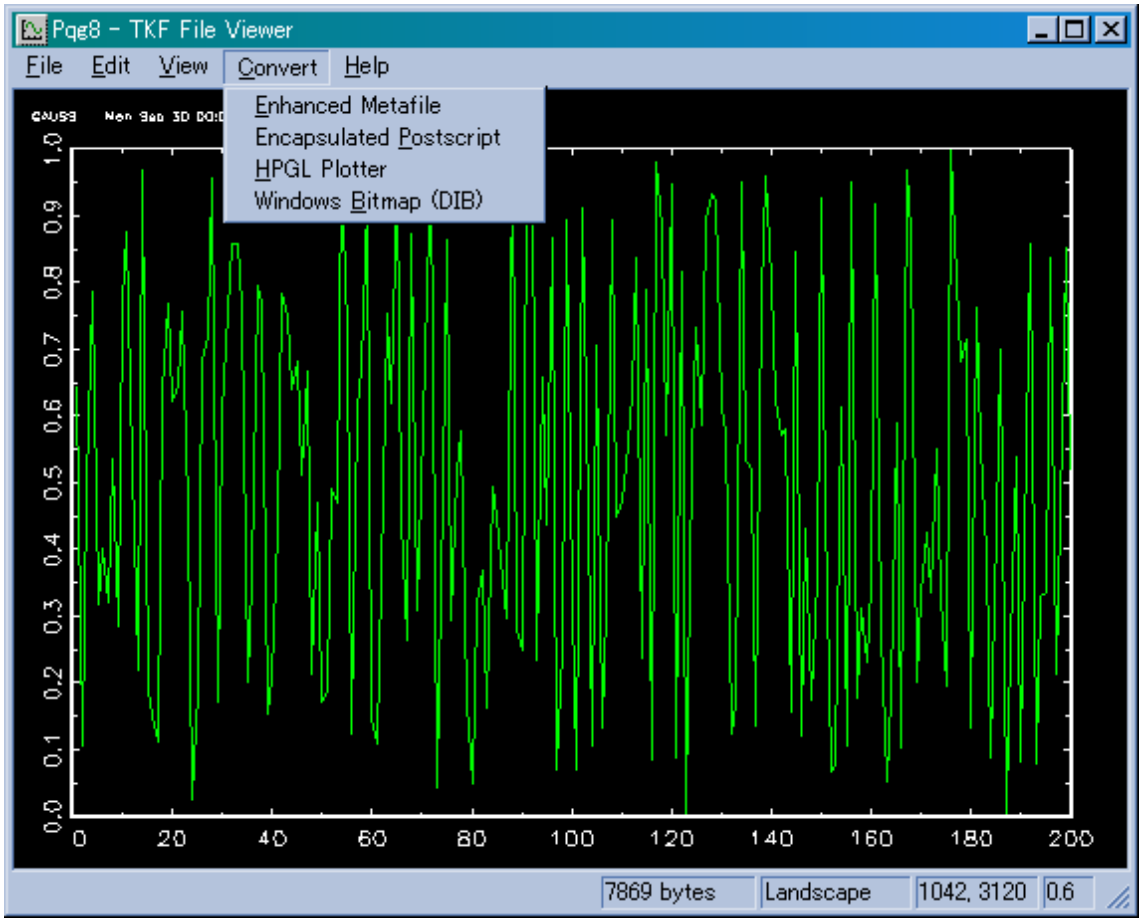
などとしなくては、内部でエラーとなり代替的に Working Directory に同名のファイルが生成される結果になります。したがって、どのようなバージョンにも動くファイル名設定は、これまで行なってきた I/O のところやこれからの章で扱われているディレクトリは

d:/　　または　　d:￥￥

などとしないとバージョン５以上では思ったドライブのトップレベルにファイルが保存できない可能性もありますので注意してください。上のいずれかの指定をトップレベルにしてやると、それ以前のバージョンでも問題なくファイルは指定ドライブに保存できます。

TKF Viwer 上からのファイル保存

上のようにファイル名のディレクトリー指定に悩まされなくとも、TKF Viwer 上から同じようなことができます。上の-c=1 のケースが Encapsulated Postscrip で、-c=5 のケースが Windows Bitmap に相当します。なお参考までに言うと、-c=3 が HPGL Plotter に相当し、-c=8 が Enhanced Metafile に相当します。



一旦グラフを描いてしまってからは、こちらで保存する方が上級者も含めて、自由に好き

なディレクトリに保存できます。マニュアルでオプションを選択するのが、イメージ的には、プログラム上の rerun の動作に相当すると考えても差し支えはないでしょう。

表示の一時停止またはグラフの１枚ごとの表示

プレゼンテーションをやる際には、GAUSS のようにグラフが連続して表示されてはこまることや計算結果が一度に表示されては困ることがあります。そんな時によく用いられるのが、waitc というコマンドです。これは、Command Window で何かキーが押されるまで次に続く一連の計算や表示またはグラフ表示を一時的に中断するものです。以下の例は、グラフを２つ描くのに、この waitc のコマンドを用いて中断するものです。

プログラム

```
new; cls;
library pgraph;
graphset;
x=seqa(1,1,200); y=rndu(200,1);
call xy(x,y);
print "Close the TKF viewer.";
print "And then hit any key inside Command Window.";
waitc;
cls;
call hist(y,10);
```

上のプログラムの場合、xy グラフのところまでは以前のものと同じです。ただし、xy グラフや hist グラフは、後の節で述べるようにリターンがある porcedure なので、そのリターン（計算結果の付属物）を出力されないようにするために call 文で呼び出しています。こうすることによって、余計なリターン出力がなくなります。まず、call xy(x,y);で第１のグラフを描かせて、print 文で「TKF Viewer を閉じて、Command Window 内で何かキーを押してください」という２行にわたる文字表示をさせます。そして、waitc;でその時点でプラグラムを一時停止させます。実際、描かれたグラフを閉じてから、プログラムが書かれているエディターの部分ではなくて、Command Window の部分で何かキーを押すと、その後の画面をクリアーにし、ヒストグラムを描かせる call hist(y,10);にプログラムは移行します。ここで、hist の第２要素は、区切りの数のことで、10 本からなるヒストグラムが描かれます。ここでも call で止めていますから、付属する計算結果は出力されなくなります。この waitc というのは、この他、説明だけの print 文や、実際の計算結果を分割して表示するなど、自分の思うとおりのプレゼンテーションがこれによって可能になります。

そうではなくて、画面にたくさんのグラフィックウインドウを開く（実際には１枚以外のグラフ画面は後ろに隠れますが）のには、pqgwin many;または pqgwin auto;とします。元に戻すには、pqgwin one;または pqgwin manual;とします。すなわち、

プログラム

```
new; cls;
library pgraph;
graphset;
x=seqa(1,1,200); y=rndu(200,1);
pqgwin many;
call xy(x,y);
call hist(y,10);
```

というように、pqgwin many;の文を適当な場所におくと、それ以降の描画は独立した画面がたくさんでてくるようになります。ただし、この方法は、動作を止める命令ではありませんから、動作を止めるには、その前に示した waitc;と組み合わせればよいでしょう。

グラフのスケールの変更

グラフは一旦描いた際に右下端のところをつまんで大きくしたり小さくしたり自由に縮尺は変更できます。以降そのウインドウの「見た目の」大きさででグラフは描かれます。そうではなくて、グラフの目盛りの最小最大をきっちりと指定できないものでしょうか？その方法が、関数 scale による指定です。以下のように、ｘｙグラフなどのグラフの前にこれを指定すれば、各変数の最小最大値をきっちりと指定できます。

プログラム

```
new; cls;
library pgraph;
graphset;
x={1,2,3,4,5};   y=x^3;
xscale={0,6};
yscale={0,150};
scale(xscale,yscale);
xy(x,y);
```

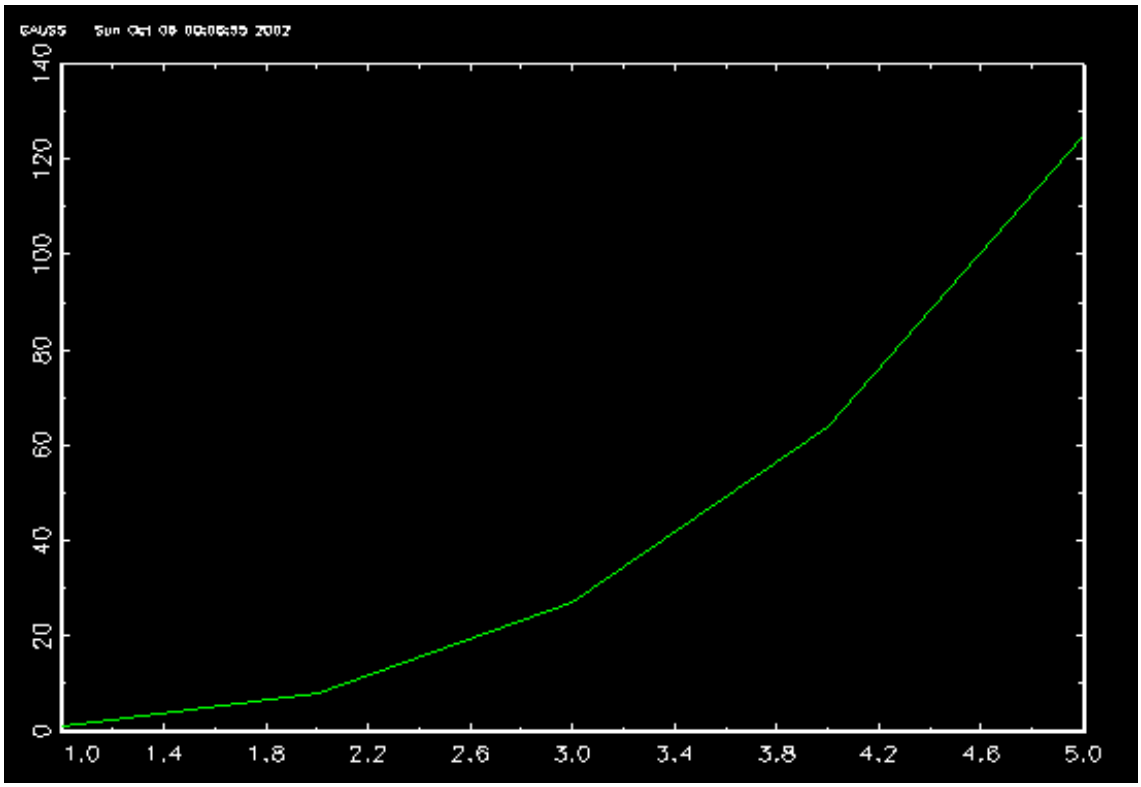
上のように、関数 scale の第 1 要素に２×１の最小値と最大値の入った変数を、第２要素に２×１の同じく最小値と最大値の入った変数を代入すると、グラフの各軸の目盛りの最小値と最大値がマニュアルで設定できます。なお、上の xscale と yscale という変数は任意です。何でもかまいませんが、GAUSS では、{0,6}と{0,150}などの｛　｝括弧を関数の中に直接入れることはできません。上の太字の３行なしに通常どおりグラフを描けば、スケールは自動的に計算されます。この関数はグローバル変数の一種です。したがって、graphset をそれぞれの直前において、グラフごとにリセットして変更も可能です。なお、軸が３本ある３Ｄグラフのケースには、関数 scale3d でもって、同様に３つの引数にそれぞれ最小値と最大値を割り当てます。この他に、次のように、それぞれの軸に最小値と最大値、それ

にステップ数や小目盛りの数を設定する xtics や ytics というのもあります。
プログラム

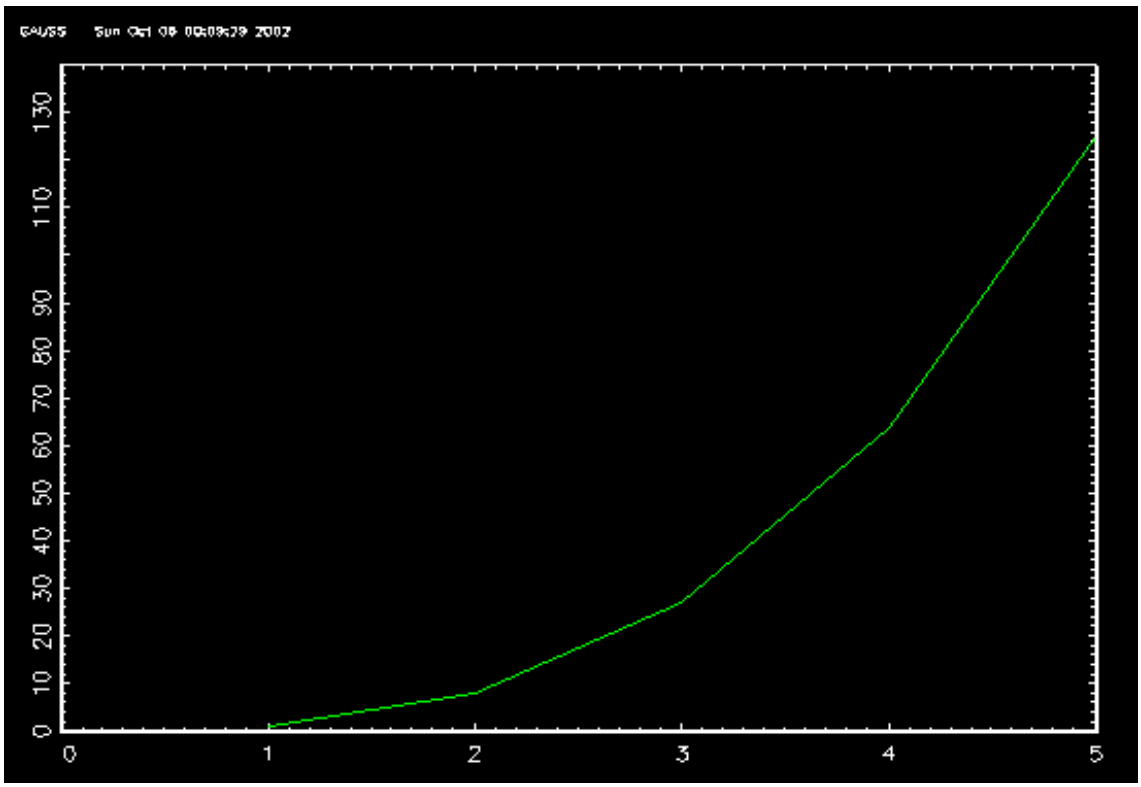
```
new; cls;
library pgraph;
graphset;
x={1,2,3,4,5};   y=x^3;
xtics(0,5,1,10); ytics(0,140,10,5);
xy(x,y);
```

書式は、今度は行列設定ではなくてスケラー数値設定で、次のようになります。
ｘ座標は　　xtics(最小値,最大値,ステップの値,１ステップ内の小目盛数)
ｙ座標は　　ytics(最小値,最大値,ステップの値,１ステップ内の小目盛数)

グラフ表示（元のグラフ）



グラフ表示（xtics と ytics を指定したもの）



すなわち、上のグラフでｘ座標は、最小０最大５の範囲で、目盛りは１ごとのステップでその中の小さな目盛は 10 刻み、ｙ座標は最小０最大 140 で、目盛りは 10 ごとのステップ

で、その中の小さな目盛りは５刻みということを設定しています。

グラフの縮小と右上移動

黒いバックグラウンドの大きさを固定しておいて、グラフの表示できる範囲を小さくしたり、それを右上方向に移動したりできます。グラフィックパネルで高度に操作したように、グラフが表示できる範囲は、９×6.855 インチの範囲にあります。この数をもとに、横の長さと縦の長さをインチで指定します。次に、それを右にある長さ、上にある長さだけ移動します。具体的には、_pagesiz と_pageshf というそれぞれのグローバル変数にそれらの数値を指定します。

プログラム

```
new; cls;
library pgraph;
graphset;
x={1,2,3,4,5};
y=x^3;
_pagesiz={4.5,3.427};
_pageshf={2.25,1.7};
xy(x,y);
```

上のプログラムでは、グラフの表示できる範囲の９×6.855 インチの約半分にして、さらにその半分を移動右上に移動することによって、グラフを半分にするものです。すなわち、

```
_pagesiz={横の長さ,縦の長さ};
_pageshf={右方向シフト,上方向シフト};
```

をインチで指定できます。下線から始まるものはグローバル変数ですが、一文字でもスペリングが間違っていたとしても、自分が定義した変数とみなされるだけで、何も動作しないので名前を確認してください。一つ前のスケールのマニュアルの設定とこれらのグローバル変数の設定によって、グラフを自由に変形できます。

特殊文字のグラフ内表示

特殊文字は、関数 fonts によってフォントをロードしてから、¥201 や¥202 などのあとに文字を書くことによって、それぞれの番目のフォント構成にもとづく表示ができます。

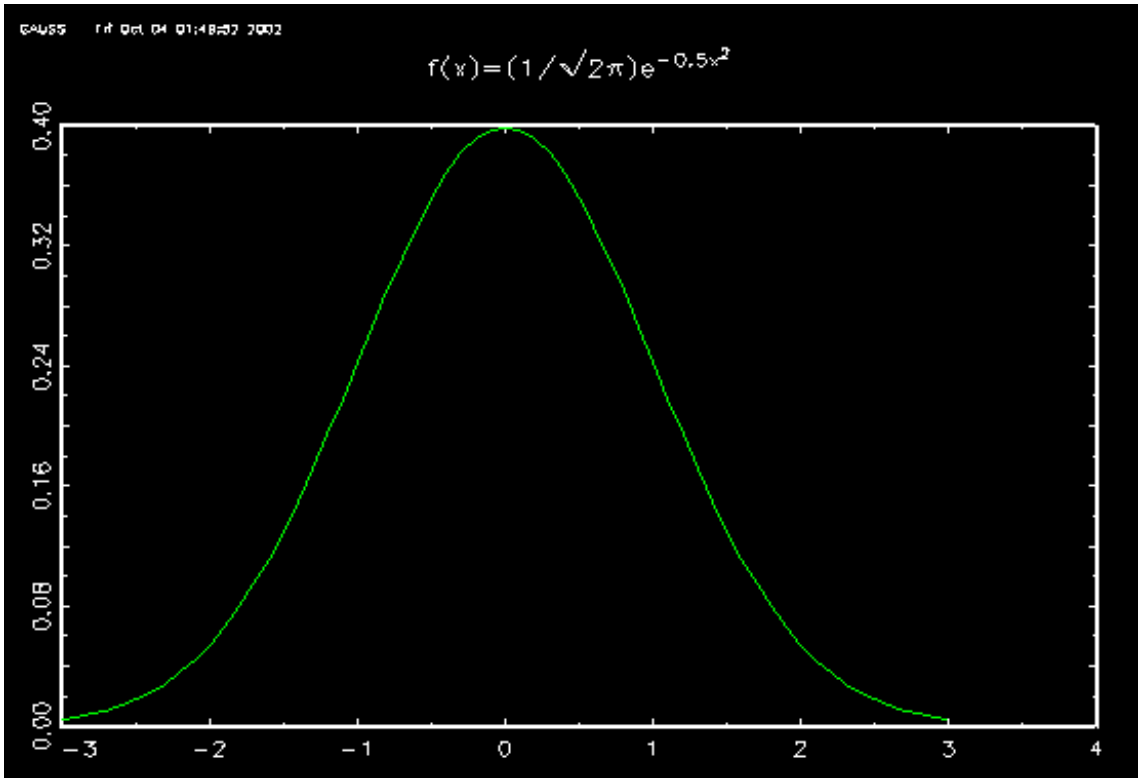
プログラム

```
new; cls;
library pgraph;
graphset;

x=seqa(-3,0.1,61);
```

```
y=pdfn(x); /* y=(1/sqrt(2*pi))*exp(-0.5*x^2); */
fonts("simplex simgrma");
title("f(x)=(1/¥2021¥2012¥202p¥201)e[-0.5x[2]]");
xy(x,y);
```

画面表示



上のプログラムで注意すべきことは、fonts 関数でフォント名をロードしなければ、アルファベットから何も変らないということです。マニュアルの Appendix にあるように、

Simplex,Simgrma,Microb,Complex

のフォントが利用できます。そのうち、Simgrma はギリシャ文字や特殊フォントを含みます。ロードする順番によって、１番目にロードしたものが¥201、２番目にロードしたものが¥202 などとなります。使わないのであれば、わざわざ４つすべてをロードすることは必ずしもありません。ここで、Legend 凡例の区切りに使った¥000 のような１つの制御文字のかたまりを用います。例えば、title の中に特殊文字を含む文字を設定するには、通常の文字はそのまま、または、この場合¥201 の後に書いていきます。特殊文字にする場合のみこのフォントのロードの順番の場合¥202 の後に書きます。ロードの順番や数によって、この制御コードは前後します。なお、四角括弧[　]の中には上付の文字を入れます。この例では、タイトルのところですが、その他の部分にも同じようにして特殊文字を書けます。

上のプログラムの制御文字は、¥201 や¥202 を１つのかたまりと見てください。そのあとに数字が来ようと文字が来ようとそれらの制御コードの後で変更されたフォントと考えて

その変更は、あらたに制御コードのかたまりが来るまで続きます。
　以下、主に使われるギリシャ文字と特殊（数学）文字と通常のキーボードのアルファベットの対応表の主なものをまとめておきます。

ａ ｂ ｃ ｄ ｅ ｆ ｇ ｈ ｉ ｊ ｋ ｌ ｍ ｎ ｏ ｐ ｑ ｒ ｓ ｔ ｕ ｖ ｗ ｘ
α β γ δ ε φ γ χ ι　　κ λ μ ν ο π θ ρ σ τ υ　　ω ξ

，　￥ １ Ｓ ＜ ＝ ＞ ＾ Ｄ ７ ０ ＃ ［ ］ ２ ６ （ただし￥はバックスラッシュ）
∫ ∂ √ Σ ≦ ≠ ≧ ∩ Δ ∥ ∇ ≡ ⊂ ⊃ ∮ ∃