X.7 Mathematica to GAUSS の基礎　　ver.0.1

GAUSS と Mathematica の大きな違いは、GAUSS は微分積分の解や方程式の解を代数的には解くことはできないけれども、Mathematica では記号として解けることです。もっとも、GAUSS は計量のソフトウエアであるので、この点は数量的に数値に対応した最適値や解を見つけてくることになりますので手法が根本的に違います。また、Factor[　]などの Mathematica でできる因数分解などは、計量ソフトウエアの GAUSS ではできません。

微分

Mathematica	GAUSS
D[x^3, x]	fn f(x)=x^3; print gradp(&f,2);

まずもって違いを認識すべきことは、Mathematica では x^3 の微分が代数的に $3x^2$ と解けるのに対して、GAUSS は評価点での数値微分しかできないことです。上の例では、関数 f(x) を x^3 としてあらかじめ定義したものをポインタ呼び出しの＆のマークをつけてその関数を数値微分を行なう関数の中で呼び出して、その第２要素の２で評価することになります。最終的にその結果を画面表示させています。

グラフ表示

Mathematica	GAUSS

Plot[D[x^3, x],{x,-2,2}]	fn f(x)=x^3;
	library pgraph; graphset;
	x=seqa(-2,0.1,41);
	xy(x,diag(gradp(&f,x)));

微分結果をｘが－2 から２までの数で評価してグラフ表示するには、Mathematica では描画の関数 Plot の最初にその関数を、そしてカンマの後に{　}の中に変数名と始点と終点の３つを書き入れてやると表示される。GAUSS では、グラフ機能と GAUSS 本体とは別であって、１つの library として pgraph というものを呼び出して、graphset でそのグローバル変数を初期化してから、ｘｙグラフでプロットすることになる。ただし、GAUSS はあくまで計量のソフトウエアであるので、データ１つ１つに対したプロットしか想定されていな

い。しかがって、技術的には－２から例えば十分に小さい値 0.1 刻みに 41 個分のシークエンスを seqa によって発生させた後にこれをｘ軸として、gradp の関数の第２要素のところに代入してやる。これをｘｙグラフでその第１要素にはｘ軸の値の入った列変数を、そして第２要素には gradp の計算結果の入ったものを入れる。注意すべきことは上のプログラムで diag なしにｘｙグラフにしてやると少しおかしなことになる。これは、

```
print gradp(&x,seqa(-2,0.1,41));
```

としてやってもわかるように、結果は 41×41 の行列として微分結果が出てきて、その対角要素が我々の求めたいものになっているためである。これは GAUSS が１要素ごとに微分を計算していて、評価値間のクロス微分までわざわざ０として計算しているためです。そのため、diag( )の関数により対角要素をベクトルとしてピックアップしないとすべての行列値がグラフ表示されて色とりどりの結果になってしまうのです。

近似解と厳密解

Mathematica	GAUSS
3/17	3/17
N[3/17]	
3/17//N	

GAUSS には分数での厳密解の概念はありません。割りきれても割りきれなくても、すべて小数で表示されます。一方、Mathematica には、小数に直せない場合、分数は分数のままの厳密解が通常の操作では出てきます。近似解としての小数として表示するには、N[　]で与えるか、または、//N で与えると小数近似の解が得られます。なお、GAUSS では、1/2 の割りきれる場合も 0.50000000 のままなのですが、小数以下の０の部分を消すには、

```
print/rz 1/2;
```

のように print 文を/rz または/lz のオプションでもって表示させると 0.5 のように０の続きの部分はなくなります。

代入と定義

Mathematica	GAUSS
f[x_]:=x^3	fn f(x)=x^3;
x=2	x=2;

Mathematica では後ろのものを前に代入する時には x=2 のようにイコールサインを使用するのに対して、ユーザーが関数を定義する時には:=のコロンつきのイコールサインを使用します。また、関数の引数の名前に対して変数名のあとにアンダーラインをつけて区別します。これに対して、GAUSS では代入も定義も同じイコールサイン１つで対応します。また関数の引数も通常の変数と全く同様に扱います。また、上の例やその前の PLOT とｘｙグラフの例からもわかるように、Mathematica では関数の引数を包むのには四角括弧を使うのに対して、GAUSS では丸括弧で対応します。

リストと数列

Mathematica	GAUSS
{1, 2, 3, 4, 5}	{1, 2, 3, 4, 5}
{ {1, 2}, {3, 4}, {5, 6} }	{ 1 2, 3 4, 5 6}

Mathematica でも GAUSS では数値を数列として表現したり格納したりするのには{　}の括弧を使って、各要素をカンマで区切ります。Mathematica ではこれをリストと呼びます。リストはリストであって、GAUSS の{1,2,3,4,5}が５×１の行列（すなわち列ベクトル）を表しているのとは若干仕様的に異なります。GAUSS では{　}括弧のなかに{　}括弧をまた書いてリストのリストを使って要素をグループ化して擬似的に行と列の区別がついたようにする Mathematica のようなことはしません。カンマまでが１行で、その次のカンマまでが２行目というふうにカンマの位置で行数と列数を判断します。GAUSS では同一列にあるものは同じ系列のデータと考える計量特有の考え方を採用しています。つまり、上の場合１と３と５は１列目で、２と４と６は 2 列目です。このまま「列の束」として関数にほうり込んで列ごとの結果を出したりグラフ表示することになります。

四則演算等(Mathematica のドット:近似値 vs.GAUSS のドット:要素対要素の演算)

Mathematica	GAUSS
(1 + 1) – (2 – 3)	(1 + 1) – (2 – 3)
2 (5 – 2)	2*(5 – 2)
1/4 + 1/2	1/4 + 1/2
1./4 + 1/2	1./4 + 1/2

足し算と引き算は Mathematica も GAUSS もどうような扱いです。ただし、特にフォーマ

ットをしないかぎり、その結果は異なります。Mathematica では整数部分だけの 3 とそのものの答えが出されるのに対して、GAUSS では 3.0000000 などと小数点以下にゼロがついてきます。きわめて計量を念頭においた仕様です。Mathematica ではスペースを置いた数や変数はかけあわされます。特に括弧へ変数の直前には視覚的に実際の代数と同じ扱いになります。一方、GAUSS は転置記号の後では掛け算の＊印は省略できますが、それ以外には必ず＊印を掛け算の演算記号としてつけなくてはなりません。GAUSS の場合、＊の記号は行列と行列の積にも使われます。ただし、要素対要素で計算する際には、.＊のようにドットつきの計算をさせます。このように GAUSS ではドットは「要素対要素」という意味を持ちます。割り算も四則の優先順位も Mathematica も GAUSS も共通です。しかしながら、その結果はここでも異なります。すなわち、上の 1/4 + 1/2 の計算の場合、有理数で 3/4 と結果が出てくる Mathematica に対して、GAUSS では割りきれても割りきれなくてもとにかく小数で答えがでてきます。ただし、Mathematica では 1./4 + 1/2 のように計算の中で数字にドットがつくと近似計算に切り替わって、こたえは 0.75 になります。このように、Mathematica では数字にドットがつくと近似値を表し、演算結果も近似値で小数で表されます。なお、GAUSS では同じ 1./4 + 1/2 の式でも、意味自体が違います。すなわちドットの後ろの割り算の演算記号の方にかかって、ここでは要素対要素で割り算をするという意味になります。ここでは１×１の数値と数値の割り算ですからドットがついてもつかなくても要素対要素の計算をしていることになるので、1/4 + 1/2 も 1./4 + 1/2 も結果はまったく同じことになります。

組み込み関数

Mathematica	GAUSS
Sin[0.5Pi]	sin(0.5*pi)
Exp[1.5]//N	exp(1.5)

Mathematica では、組み込み関数の中味は通常四角括弧[　]の中に書きます。一方、GAUSS では丸括弧(　)中に書きます。四角括弧は、GAUSS では、配列を表します。なお、Mathematica では関数名や Pi などの予約語はすべて大文字から始まり、その他の文字は小文字にします。必ず大文字から始まらねければなりません。このように、Mathematica には大文字と小文字の区別が厳然と存在します。しかしながら、GAUSS には大文字小文字の区別はなく、通常、小文字で統一して書かれることが一般的です（もちろん、大文字で書いても同じです）。0.5 と予約語の pi の間には、GAUSS では掛け算の＊のマークが必ず必要です。Mathematica は省略できます。指数関数も、関数名は同じですが、同様な違いがあります。Mathematica は大文字から始まる予約語の関数名で四角括弧[　]を使います。

GAUSS は大文字小文字の区別はなく通常小文字で関数名を表記して丸括弧(　)を使いその中に値を入れます。Mathematica では近似値で解がほしい時には、//N をつけるか、または N[　]で式自体を囲みます。GAUSS では、いつでも表示桁数いっぱいいっぱいまで近似小数値で出てきます。

データの画面上でのインプット

Mathematica	GAUSS

data = {
{0.05, 0.04},
{0.10, 0.06},
{0.15, 0.10},
{0.20, 0.14},
{1.00, 1.42}
};
ListPlot[data]
	data={
	0.05 0.04,
	0.10 0.06,
	0.15 0.10,
	0.20 0.14,
	1.00 1.42
	};
	library pgraph; graphset;
	xy(data[.,1],data[.,2]);

Mathematica では、ｘとｙ座標の組を上のように{　}括弧の中にカンマで区切ってさらにその{　}括弧の中に書くことによってデータを格納することができます。GAUSS では、データは１列ずつ考えられますから、カンマまでが１行でｘとｙの組になります。これを、プロットするには、Mathematica では ListPlot を用いて、変数 data をその四角括弧の中に入れると自動作成されます。GAUSS では、前述のように、library pgraph;で 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初期化した後、xy の関数でｘ軸に相当する列ベクトルとｙ軸に相当する列ベクトルをカンマで区切って指定します。上では、ワイルドカードのドットマークを用いて、data[.,1]とは変数 data の１列目という意味で、data[.,2]とは変数 data の２列目という意味です。GAUSS ではグラフは自動作成されません。必ず列ベクトルで要素を指定します。GAUSS では座標をつなぐことがデフォルトになっています。ドット表示にするには、ｘｙのグラフの命令の前に、

```
_plctrl=-1;
```

というグローバル変数（pgraph グラフライブラリ関数のパラメータ設定）をします。

２乗誤差和

Mathematica	GAUSS
Sum[(f[data[[i,1]]] - data[[i,2]])^2, {i,1,Length[data]}]	sumc((data[.,1]-data[.,2])^2);

ｘとｙの差の２乗の和は Mathematica では、Sum[xi, {i,1,n}]というふうに、関数 xi を I について１からｎまで動かしたときの総和を Sum で求めます。上の場合、f が data[i,1]と data[i,2]の差としてその２乗を i について１から Length[data]まで動かした時の総和を求めています。ここで、Length[　]はその変数の長さを求める組み込み関数です。GAUSS では、rows(data)とします。GAUSS で、同じことをするには、「列として」計算ができて、前述と同様にワイルドカードのドットを用いて、列の長さを考えることなくして、その列の差を１つの結果列と考えて、その列の和を sumc(　)で求めます。組み込み関数 sumc は、ｃが付いていることでもわかるように、列ベクトルの和を求める関数です。

変数の長さ

Mathematica	GAUSS
Length[data]	rows(data) cols(data)

その前の計算で用いたように、Mathematica ではその変数の長さを計算するには、組み込み関数 Length[　]を用います。一方、GAUSS は行列言語であって、列と行の区別は厳然としていて、rows(　　)と cols(　)で行と列をそれぞれ求められます。２つ前の data の設定では、Mathematica の Length[data]は GAUSS の rows(data)に相当します。

関数および手続き定義

Mathematica	GAUSS
f[x_]:=x^2	fn f(x)=x^2;
rei[f_,x_]:=Block[{ },	proc rei(&f,x);
	local a,b,c,n,f:fn;
・・・・・・・・	・・・・・・・・・
Return[n]	retp(n);
]	endp;

Mathematica では、１行で定義できるものは、f[ ]:=式　の形で定義できます。GAUSS では同じようなことを fn で対応できます。GAUSS ではイコールサイン＝のままです。:= という形はありません。複数行にまたがるような手続きを定義するには、Mathematica では Block[ ]が使われます。リターンがある場合には、Return[ ]をその中に置きます。一方、GAUSS では、それに対応するものは、procedure の proc です。上では、手続き名は共通して rei になっています。リターンは retp( )の中に書きます。リターンが複数個または０個の場合には、その冒頭部分は proc(2)=rei( );や proc(0)=rei( );となり、括弧の中の数字とリターンの数を一致させます。GAUSS では手続きの最後には、endp;を置きます。またローカル宣言する場合には、local 文の後にカンマで区切って並べます。呼び出す「インプット」が関数の場合には、:proc や:fn などとします。また、インプットのところに、&のマークをつけて通常のインプット変数と関数を区別します。Mathematica の場合、関数や手続きの引数インプットの変数の後ろには通常の変数と区別して、x_のように、必ず__のマークをつけます。GAUSS では引数も通常の変数も同じです。ただし、ライブラリのグローバル変数（パラメータ設定）には変数の前に_variable というふうに下線がつきます。

Do ループ

Mathematica	GAUSS
	j=0;
Do[i=i+1;Print[i],{i,0,9,1}]	do while j<=9;
	i=j+1; print i;
	j=j+1;
	endo;

上のプログラムは、１から１０までの数列を画面表示するプログラムです。Mathematica では、上のようにすると i ＝0,1,2,3,4,5,6,7,8,9 と都合 10 回数値がそれぞれ設定され、計算されます。累積することはありません。GAUSS で同じような結果を出すには、カウンターは自分で管理して１回のループごとに１ずつ増しますから、別に例えば任意の変数 j を使います。これを do ループの前で j ＝0 と止めておいてから、ループで１ステップで９まで増します。それに１を足したものを変数 i に代入して、それを毎回画面表示させます。なお、i=j+1; print i;を消して、print j;にすると０から累積された和が毎回表示させます。

For ループ

Mathematica	GAUSS
For[スタート,テスト,インクリメント,式]	for i (スタート値,ストップ値,ステップ数); 式 endfor;

For ループの場合、Mathematica では四角括弧[　]の中にすべて書きます。スタートのところは、i=0 のように式で書きます。GAUSS では変数を指定して、その後に丸括弧の中に数値を３つカンマで区切って(0,10,1)などというように書きます。GAUSS では endfor;までの部分に式を書くことになります。

If 文

Mathematica	GAUSS
f[x_]:=If[x>2,1,0]	fn f(x)=(x>2);
f[3]	f(3);

Mathematica では、If 文は、If[テスト,True 式,False 式]というふうに関数の形になっています。上の場合、x>2 の評価をして、それが True なら１、False なら０とするようにしています。この場合、３ですから、３＞２は True ですから、１を返します。GAUSS で同じことをするには、(x>2)で真偽を判断して、GAUSS では True であれば１、False であれば０をその部分は返します。それをイコールサインで f(x)に代入しています。GAUSS では、最初から真偽の値は決まっています。If は、通常 GAUSS では、条件分岐に使われ、その後には必ず endif;で受けなくてはなりません。そうでない場合には else;を、複数の if がそうでない場合にある場合には、elseif 式;を続けて、最後には endif;を必ず置きます。