2.1 行列 基本的な演算 ver. 0.1

セクション１では、GAUSS の I/O のあらゆる局面で、行列の概念が使われいることをプレビューしました。データの取り込みからグラフ表示の設定まで、ありとあらゆる設定が行列で書かれていることがわかったと思います。たとえそれがスケラーに見えてもたいていの場合、数値は、実は１×１行列として扱われています。セクション２では、この行列の操作に焦点をあてて説明していきます。一部くり返しになりますが、本格的なプログラムをする前に、ここで GAUSS 特有の行列操作についてウオームアップをしてください。

まずは、スケラー数値の四則演算をやってみましょう。

プログラム

```
new;  cls;
a=1; b=2;
x1=a+b;
x2=a-b;
x3=a*b;
x4=a/b;
print x1 x2 x3 x4;
```

画面結果

```
3.0000000      -1.0000000      2.0000000      0.50000000
```

メモリーをリフレッシュして画面をクリアするというお決まりの命令の後、ａに１を代入して、ｂに２を代入する。それぞれ１つずつの式なので、それぞれに；のマークが必要である。GAUSS では、１つの行に２つ以上の命令や式を；のマークをつければ書くことができる。また、２行以上にわたって命令や式を書いてもよい。その際に、１つの命令や式のかたまりは；のマークで判断される。それから、イコール＝のマークは、「同じ」という意味ではなくて、「後ろのものを前に代入格納する」という意味であるので注意されたい。

ｘ１にａ+ｂを、ｘ２にａ-ｂを、ｘ３にａ×ｂを、ｘ４にａ÷ｂの式の値をそれぞれ代入しています。そして最後の行では、ｘ１とｘ２とｘ３とｘ４を横にならべて画面表示しています。たいていのプログラム言語と同じように、掛け算の×のマークは＊のマークで、割り算の÷のマークは／のマークで代用されています。

さて、統計学でよく使われる exponent 指数倍や factorial 階乗はどうでしょうか。

プログラム

```
new;  cls;
a=10; b=11;
x1=a^b;
x2=a!;
```

```
x3=a*(a-1)*(a-2)*(a-3)*(a-4)*(a-5)*(a-6)
    *(a-7)*(a-8)*(a-9);
print x1 x2 x3;
```

画面結果

```
1.0000000e+011        3628800.0        3628800.0
```

四則の計算と同様に、メモリのリフレッシュと画面の消去のあとに、ａとｂの値を設定しています。ここでは、ａに 10、ｂに 11 をそれぞれ代入しています。a^bとａ！の式の値をｘ１、ｘ２とそれぞれして、画面表示させています。なお、ｘ３については、ａ！と同じ計算をしています。指数、階乗ともに、たいていのプログラムと同じく、＾のマークと！のマークが使われています。なお、ｘ３については、ｘ２のａ！の意味と同じ計算をしていますが、１行が長くなったので、２行に適当に分けて記述しています。分け方はどこで分けても、どこから始まってもよくて、要するに最後が；のマークであれば、スペースがいくら入ろうと、どこで切ろうと構いません。GAUSS に行の概念がなくて、;のマークまでが１命令または式であることをわかっていただけたでしょうか。ただし、ａのところがマイナスの数である場合には、ｂは必ず自然数でなくてはなりません。結果は、10 の 11 乗の値は、１×１０の１１乗という意味で 1.0000000e+011 という表示になっています。画面表示のデフォルトが/ro であるために、符号と小数点を含まない実質８桁の十進法表示の範疇で書ききれないので、自動的に、よりコンパクトな/re という指数表示に切り替わっています。指数表示が気にいらない場合には、/rd オプションを print 文につける必要があります。ｘ２もｘ３も同じことをしているので、当然同じ値になりました。こちらは実質８桁の表示がよりコンパクトということで自動選択されています。

もう少しだけスケラーの計算を続けます。統計計量でよく使われる演算としては、指数と対数があります。簡単な例で見ていきましょう。どちらも組込み関数を使います。

プログラム

```
new;  cls;
e=exp(1);  e2=exp(2);  e10=exp(10);
x1=ln(e);
x2=ln(e2);
x3=log(e10);
print e x1 x2 x3;
```

画面表示

```
2.7182818        1.0000000        2.0000000        4.3429448
```

GAUSSでは指数関数の指数部分はexpという組込み関数の括弧内の引数として表現されます。例えばｅまたはe^1は exp(1)として表されます。e^2なら exp(2)、e^{10}なら exp(10)と

いうことになります。それぞれの値をｅ１、ｅ２、ｅ３と順番に定義（または代入）してｅ１とｅ２は自然対数の組込み関数 ln の引数としたものをｘ１、ｘ２にそれぞれ代入しています。ｅ３は常用対数 $\log_{10}$ の引数としたものをｘ３に代入しています。最後に、もともとのｅ１の値の小数表示と、ｘ１、ｘ２、ｘ３の計算結果を画面表示しています。このように、自然対数と常用対数も組込み関数 ln と log の引数として括弧の中に数値を入れるようになっています。指数の場合も対数の場合も、総じて組み関数一般に、その引数は括弧（　）のなかに入れるように GUASS ではデザインされています。ちなみに、{　　}の中には行列のそれぞれの実際の要素値が入り、[　]の中には行列配列の何行何列目といういインデックス番号が入ります。この点、GAUSS は行列の計算を主眼におかれた言語なので区別ははっきりとしています。Matlab などの数学系の言語を知っている方には、少し戸惑いや、習慣からくる括弧の使い方の混同があるかもしれません。ちなみに、print 文も組込み関数なのですが、例外的に括弧（　）を使わずにそのまま数値を書くのが慣例になっています。ただし、オプションは括弧（　）の外、変数は括弧（　）の内に書いて print/rd(x1); など書けば、支障なく画面表示ができます。各自で試してみてください。

　その他の演算として考えられるのには、πとサインコサインの計算が残っています。πは組込み関数ではなくて、組込み定数 pi として処理されます。少し注意が必要なので例をあげておきましょう。

プログラム

```
new;  cls;
x1=sin(pi/2);
x2=sin(pi);
x3=cos(pi/2);
x4=cos(pi);
print pi x1 x2 x3 x4;
```

画面表示

```
3.1415927      1.0000000    1.2246064e-016      6.1230318e-017      -1.0000000
```

まず、GAUSS では、ごく一部の例外を除き、たいていの場合引数の中に計算式を書くことができます。ここでも、sin と cos の組込み関数の中にπを２で割るという計算式を書いています。GAUSS ではπは pi という 2 文字からなるキーワードとしてあらかじめ内部で定義されています。サインコサインの記号は sin と cos で全く同じです。数値は実際と同じように、括弧（　）の内に入れます。プログラムに書かれているとおり、πの値そのものと、π/２とπの時の sin と cos の値を求めて画面表示しています。πの値はデフォルトの/ro どおり実質８桁の数で表されています。π/２の時の sin の値は、当然１です。πのときの cos の値も当然－１です。しかしながら、０であるはずの sin(π)と cos(π/２)は、そうなっては

いません。ほとんど０なのですが、小数点以下０が 15 個から 16 個続く小数になってしまっています。これは明らかに誤りです。DOS の時代からこの仕様は改善されていないので何か理由があるのかもしれません。ただ、π自体を sin の中に入れても０にはならないので計算上の桁数の問題というよりも組込み関数 sin や cos 自体に端点のとき０になることが条件として書かれていないのでしょう。プログラムで sin や cos を使う場合にはその値が０の周辺では０には完全にはならないということを念頭においてプログラムをする必要があります。これを回避する 1 つの方法に、あらたにサインやコサインの procedure を自分で書いて、引数がどういう場合には、戻り値が０になるかを条件分岐で書いて定義して、sin や cos の組込み関数の代わりにその procedure を使う必要があるでしょう。フォーマットでは途中計算は制御されず、結果の見かけの出力しか制御できないので、フォーマットで調整してやろうというのは適切ではありません。GAUSS には、このほか０の周辺の計算が正確ではなかったり、インデックスが０になると計算されない場合など、通常の数学ソフトでは考えられない「仕様」がありますから、その際には、０周辺のいじわるテストでプログラムを確かめてから、不具合があれば、条件分岐を考えて計算をしなくてはなりません。

さて、本題の行列計算がどうなるのか、スケラーの場合と対比して見ていきましょう。

プログラム

```
new;  cls;
a={1 2,
   3 4};
b={5 6,
   7 8};
x1=a+b;  x2=a-b;  x3=abs(a-b);
print x1;  print x2;  print x3;
```

画面表示

```
      6.0000000       8.0000000
      10.000000       12.000000

     -4.0000000      -4.0000000
     -4.0000000      -4.0000000

      4.0000000       4.0000000
      4.0000000       4.0000000
```

細かい説明は省略しますが、足し算引き算は、スケラーと同じように上のように計算でき

ます。答えの絶対値を求めたいときには、組込み関数 abs を使って、その引数に値または計算式を与えてやれば、絶対値で表示されます。これはスケラーでも同じです。くりかえしになりますが、{　} の中には行列の実際の要素値が入り、カンマのマークまでが一列目その次のカンマまでが２列目というふうに表示します。{1,2,3,4} という行列は横（行）ベクトルではなくて、縦（列）ベクトルをさしますので注意が必要です。厳密です。なお、普通の代数の概念と同じく、行列の加減の際には、行列の縦横の大きさは同じ必要があります。もし、足し合わせたり引いたりする行列が前の行列と型が合わない場合には、計算はエラーを返してそこで止まります。行列の型があわないケースも含めて、いくつかの計算をさせたい場合には、型が合わない場合に分岐するプログラムを書いてやる必要があります。GAUSS はプログラム言語の一種ですから、エラーも含めてプログラムを制御してやることが必要になることもケースによってはあります。

ただし、３次元のグラフ surface と contour のところで述べましたように、行列の加減の演算の型の一致には例外がありました。それは、それぞれの行列が１行ベクトルと１列ベクトルの場合です。少々複雑なので、ここであらためて説明します。

プログラム

```
    new;  cls;
a={1 2 3 4};
b={0,
   0,
   0};
x1=a+b;
x2=a-b;
print x1; print x2;
```

画面表示

```
1.0000000     2.0000000     3.0000000     4.0000000
1.0000000     2.0000000     3.0000000     4.0000000
1.0000000     2.0000000     3.0000000     4.0000000

1.0000000     2.0000000     3.0000000     4.0000000
1.0000000     2.0000000     3.0000000     4.0000000
1.0000000     2.0000000     3.0000000     4.0000000
```

上のように M 行１列の横ベクトルと１行 N 列の縦ベクトルを足したり引いたりすると、M 行 N 列の行列が作成されます。その際には、横ベクトルがあたかも縦ベクトルの N 列分あるかのような行列になり、縦ベクトルのほうも横ベクトルの M 行分あるかの行列になり、

その上で加減の演算が行なわれます。上の例は横ベクトルが｛1 2 3 4｝というとき、すべての要素が０である縦ベクトルを足しても引いても｛1 2 3 4｝が縦ベクトルの行数分同じものが複製されてならんでいる例です。なお、横ベクトルと縦ベクトル、どちらを加減の後先にしてもかまいません。要するに、１行の横ベクトルと１列の縦ベクトルでそれぞれあればよいわけで、どちらか片方が２行以上や２列以上になっては型がちがうというエラーnot conformable になってしまいます。この演算の例外はジオメトリーを考えた、主に、３次元のグラフの座標を念頭においたものです。実際には｛1 2 3 4 .....｝と｛1,2,3,4,...｝という座標値をそれぞれプラスをマークで足し合わせることが行なわれるのが慣例です。例えば、ｚ＝2sin(x)+5cos(y)のｘｙ座標からｚの高さをもとめるとすると、

プログラム

```
new;  cls;
x={1 2 3 4};
y={1,
   2,
   3,
   4};
z=2*sin(x)+5*cos(y);
print x; print y; print z;
```

画面表示

```
1.0000000       2.0000000       3.0000000       4.0000000

1.0000000
2.0000000
3.0000000
4.0000000

 4.3844535       4.5201064       2.9837515       1.1879065
-0.39779221     -0.26213933     -1.7984942      -3.5943392
-3.2670205      -3.1313676      -4.6677225      -6.4635675
-1.5852761      -1.4496233      -2.9859781      -4.7818231
```

基本的に、この演算の例外はジオメトリーを念頭においてつくられているので、ｘ座標やｙ座標の値にそれぞれ何がかけあわさっていようが組込み関数によって変換されていようとかまいません。要するに＋のマーク（あるいは－のマーク）によってｘとｙから構成されるそれぞれの式がつなぎ合わさっていれば座標に対する高さを計算します。３つめの行列の結果の内容は、具体的には、

1	2	3	4	
2sin(1)+5cos(1)	2sin(2)+5cos(1)	2sin(3)+5cos(1)	2sin(4)+5cos(1)	1
2sin(1)+5cos(2)	2sin(2)+5cos(2)	2sin(3)+5cos(2)	2sin(4)+5cos(2)	2
2sin(1)+5cos(3)	2sin(2)+5cos(3)	2sin(3)+5cos(3)	2sin(4)+5cos(3)	3
2sin(1)+5cos(4)	2sin(2)+5cos(4)	2sin(3)+5cos(4)	2sin(4)+5cos(4)	4

となります。必ずしも縦横が同じ正方行列になる必要はありません。ｘに関する式とｙに関する式がそれぞれ＋のマークでつながれていればよいのです。

今度は、行列の乗除です。ここでも、

プログラム

```
new;  cls;
a={1 2,
   3 4,
   5 6,
   7 8};
b={1 2 3,
   4 5 6};
x1=a*b;
print x1;
```

画面表示

9.0000000	12.000000	15.000000
19.000000	26.000000	33.000000
29.000000	40.000000	51.000000
39.000000	54.000000	69.000000

行列ａは４×２、ｂは２×３ですから、conformable でかけ合わせることができます。ここでも、スケラーの計算と同様、＊のマークが行列の掛け算に使われています。ただし、GAUSS においては、行列が not conformable になるとその後のプログラムに誤りがなくても、そこでエラーを出して終了してしまいますから、加減同様に、not conformable が見込める場合には、条件分岐させて、その他の計算が実行されるように自分でプログラムしなくてはなりません。

行列の各要素のスケラーでの掛け算、割り算、指数乗などは、スケラー同士の計算と同様に＊、／、^のマークを用いて行なうことができます。

プログラム

```
new;  cls;
a={1 2,
   3 4};
x1=a*2;
x2=2*a;
x3=a./2;
x4=a^2;
print x1; print x2; print x3; print x4;
```

画面表示

```
      2.0000000      4.0000000
      6.0000000      8.0000000

      2.0000000      4.0000000
      6.0000000      8.0000000

     0.50000000      1.0000000
      1.5000000      2.0000000

      1.0000000      4.0000000
      9.0000000      16.000000
```

スケラーは後ろからかけても前からかけても、行列に対する対要素計算になります。割り算も割る数がスケラーの場合、スケラー対スケラーの計算と同様に行なえます。指数乗の場合も同様です。ただし統計計量ソフトの中には、要素対要素（スケラー）の指定をしなければエラーがでてきまうものもありますので、読みやすいプログラム、移植しやすいプログラムという観点からは、次にあげる方法を用いた方がよいかもしれません。特に、除数はその傾向があります。

プログラム変更箇所

```
x1=a.*2;
x3=a./2;
x4=a.^2;
```

要素対計算ということを表すのに、行列と演算記号の間にドットのマークを入れます。ただし、行列と行列の除算の場合には注意が必要です。/のマークの計算と./のドット付きのマークの計算は意味が違ってきます。

プログラム

```
new;  cls;
y={13 16,
   29 36};
a={1 2,
   3 4};
x1=y/a;
x2=y./a;
print x1; print x2;
```

画面表示

```
3.0000000        4.0000000
5.0000000        6.0000000

13.000000        8.0000000
9.6666667        9.0000000
```

２番目のｘ２の計算はｙの行列の中味をｚの行列の中味で要素対要素で割っています。したがって、

```
13/1             16/2
29/3             36/4
```

の値が計算結果として画面表示されています。しかし、１番目の結果の行列は何でしょうか。それは、ｙ＝ａｘとして、ａとｙが既知な場合のｘを求める計算、すなわちｘ＝ｙ/ａを計算しているのです。したがって、ドット付きの要素対要素の除算と、ドットなしの／のマークの演算は、行列と行列の場合には全く意味が異なりますから注意してください。
なお、掛け算の時にもドットを付けて要素対要素で計算することがあります。

プログラム

```
new;  cls;
a={1 2 3};
b={1,
   2,
   3};
c={0,
   0,
   0};
d={0 0 0};
```

```
x=a+c;
y=b+d;
z=x^(2/3).*y^(2/3);
print x; print y; print z;
```

画面表示

```
1.0000000       2.0000000       3.0000000
1.0000000       2.0000000       3.0000000
1.0000000       2.0000000       3.0000000

1.0000000       1.0000000       1.0000000
2.0000000       2.0000000       2.0000000
3.0000000       3.0000000       3.0000000

1.0000000       1.5874011       2.0800838
1.5874011       2.5198421       3.3019272
2.0800838       3.3019272       4.3267487
```

例えば、ｚ＝ｘ$^{2/3}$ｙ$^{2/3}$などの計算をして、ｘｙ平面に対するｚの高さをプロットする場合、横ベクトル縦ベクトルの足し算の例外のケースをつかって、まず座標を作成します。この際、０ベクトルを足し合わせる形で行なわれます。ｘはｘで指数乗して、ｙはｙで指数乗したものをかけ合わせる場合に、通常の行列の掛け算をしてしまったのでは、ｘとｙの式の掛け算になりません。そこで、行列の要素対要素の掛け算をドットのマークをつけて行なうのです。こうすれば、最初の行列のｉ行ｊ列目ともう一方の行列のｉ行ｊ列目がそれぞれ１対１でかけ合わせられ結果も２つと同じ行列の型になります。

当然のことながら、ln、log、exp、sin、cos などの組込み関数の中にスケラーばかりでなく行列も代入して計算することが自由にできます。（というよりも、むしろ GAUSS では組込み関数の引数は行列が想定されていて、スケラーが入る場合には１×１の行列として処理されています。）具体的な例については各自でたしかめてみてください。

行列の掛け算には、普通の行列の掛け算、それに要素対要素の掛け算のほかに、クロネッカー積があります。

プログラム

```
new;   cls;
a={1 2,
   3 4};
b={5 6,
```

```
    7 8};
x=a.*.b;
print x;
```

画面表示

5.0000000	6.0000000	10.000000	12.000000
7.0000000	8.0000000	14.000000	16.000000
15.000000	18.000000	20.000000	24.000000
21.000000	24.000000	28.000000	32.000000

このクロネッカー積の計算は、ご存じのように、後ろの行列が前のそれぞれの要素に行列ごとかかっている計算で、

$$\begin{pmatrix} 1*\begin{pmatrix}5&6\\7&8\end{pmatrix} & 2*\begin{pmatrix}5&6\\7&8\end{pmatrix} \\ 3*\begin{pmatrix}5&6\\7&8\end{pmatrix} & 4*\begin{pmatrix}5&6\\7&8\end{pmatrix} \end{pmatrix}$$

を行列内部でそれぞれの行列を展開した形である。（上の行列記述は便宜的なものであって、正しくはない。）以上が、GAUSS における基本的な四則の演算方法であった。そのほかの行列特有の組込み関数については、行列データの加工の仕方、行列データの作成の仕方の後でくわしく述べることにします。

ポイント　行列の要素は GAUSS では｛　｝の中に記述する。組込み関数の引数は（　）の中に、行列のインデックスは［　］の中に記述する。

ポイント　｛　　｝の中では、カンマのマークまでが１行目、次のカンマまでが２行目というふうに記述する。

ポイント　行列においても、＋－＊／の記号が加減乗除に使えるが、行列の型に注意しなければならないことと、乗除には要素対要素の計算にドットをつけることに注意しなければならないこともある。

ただし、次にあげる割り算の余りの整数部分を求める演算記号%には、行列対スカラーの場合にもドットをつけてはいけません。

プログラム

```
new;  cls;
a={1 2 3,
```

```
    4 5 6};
x=a%3;
print x;
```

画面表示

```
1.0000000    2.0000000    0.00000000
1.0000000    2.0000000    0.00000000
```

３で行列の各要素を割ったときの余りの整数部分の計算に%がつかわれているのですが、ここではドットの記号を%のマークの前につけるとエラーが出てしまいます。もともと%の対象とする計算に行列対行列の概念がないために、こういう仕様になっているものと思われます。