0.9 ジュニア版　　乱数とその利用

　これまでも少しでてきましたが、コンピュータ利用と乱数は密接な関係があります。ここでは、乱数の基本となる「一様分布」「標準正規分布」「ガンマ分布」の乱数を取り上げて、特に一般的な一様分布及び正規分布への拡張とその実際的な使い方を解説します。

GAUSS における乱数

プログラム

```
new;
cls;
    x=rndu(10000,1);
    y=rndn(10000,1);
    a=2;
    z=rndgam(10000,1,a);
library pgraph;
graphset;
    pqgwin many;
    call hist(x,19);
    call hist(y,19);
    call hist(z,19);
```

プログラム各行の説明

１行目　メモリを初期化する

２行目　画面をクリアする

３行目　組込み関数 rndu で０と１の間の一様乱数を 10000×１（すなわち 10000 個）だけ発生させて、その結果を変数ｘに代入する。

４行目　組込み関数 rndu で０と１の間の一様乱数を 10000×１（すなわち 10000 個）だけ発生させて、その結果を変数ｙに代入する。

５行目　変数ａに２を代入する。

６行目　組込み関数 rndgam でパラメータａのガンマ分布にしたがう乱数を 10000×１(すなわち 10000 個)だけ発生させて、その結果を変数ｚに代入する。

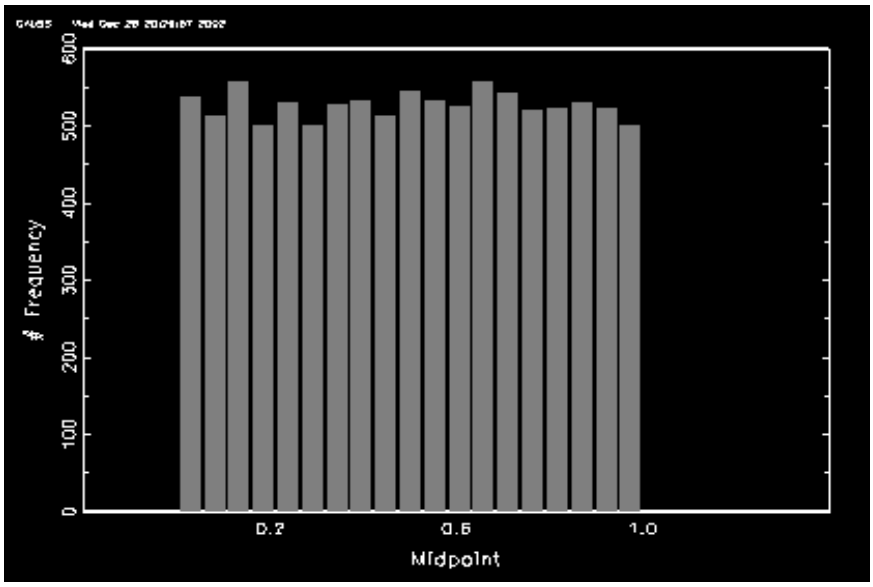
７行目　グラフを描く pgraph のライブラリを呼び出す。

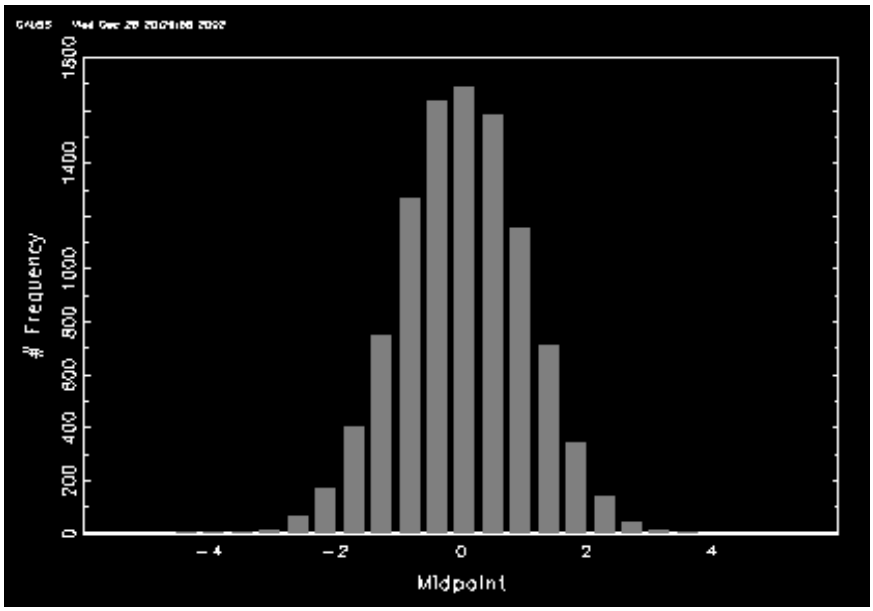
８行目　その pgraph のライブラリのグローバル変数（初期設定）を初期化する。

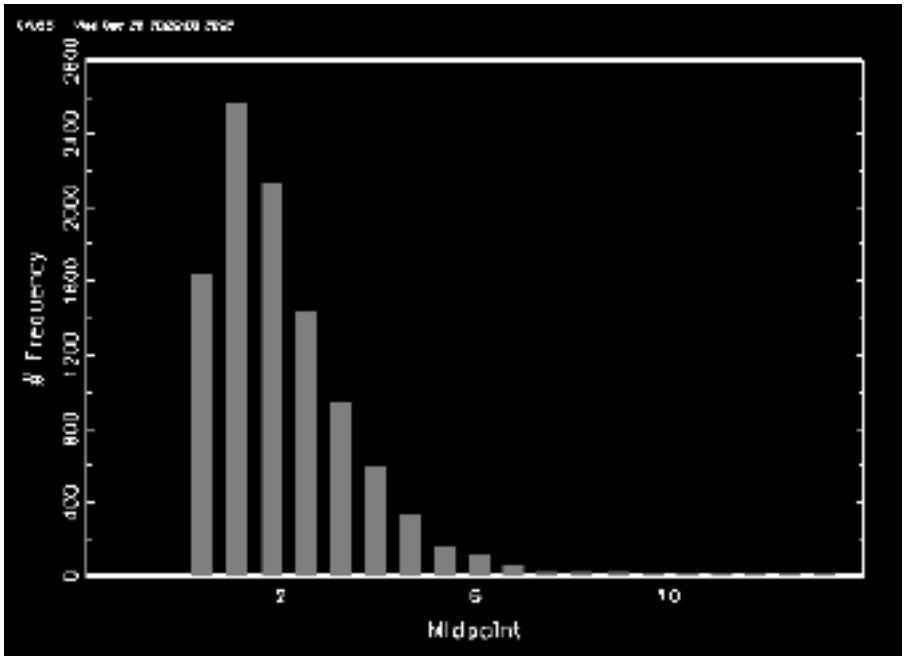
９行目　グラフの描かれる PQG ウインドウが一度に複数開けるように設定を変更する。
（この設定がなければ、GAUSS のデフォルトは一度に１枚のグラフのみを表示）

１０～１２行目　ｘ（ｙおよびｚ）のヒストグラムを19本の棒グラフで描く。

グラフ表示


GNUPLOT
Frequency
0
100
200
300
400
500
600
0.2
0.6
1.0
Midpoint


GNUPLOT
Frequency
0
200
400
600
800
1000
1400
1600
-4
-2
0
2
4
Midpoint



上のグラフの作り方と同じものについて、それぞれの平均と標準偏差を表示するプログラムは次のようになる（ただし、乱数に依存しているのでその値はその時々で異なる）。

プログラム

```
new;
cls;
    x=rndu(10000,1);
    y=rndn(10000,1);
    a=2;
    z=rndgam(10000,1,a);
print "      U[0,1)" meanc(x) stdc(x);
print "      N(0,1)" meanc(y) stdc(y);
print "Gamma(a=2)" meanc(z) stdc(z);
```

プログラム各行の説明

１～６行目　（　同上　）

７～９行目　引用符内のメッセージの文字列とともに、ｘ（ｙおよびｚ）の平均と標準偏差を計算結果の２つを画面表示する。

画面表示

```
     U[0,1)       0.50191778       0.28951933
     N(0,1)    0.0053559613        1.0028955
Gamma(a=2)        1.9946959        1.4112971
```

一様乱数の引き伸ばしと平行移動

プログラム

```
new;
cls;
    x=6*rndu(10000,1);
library pgraph;
graphset;
    call hist(x,19);
```

プログラム各行の説明

１行目　メモリを初期化する

２行目　画面をクリアする

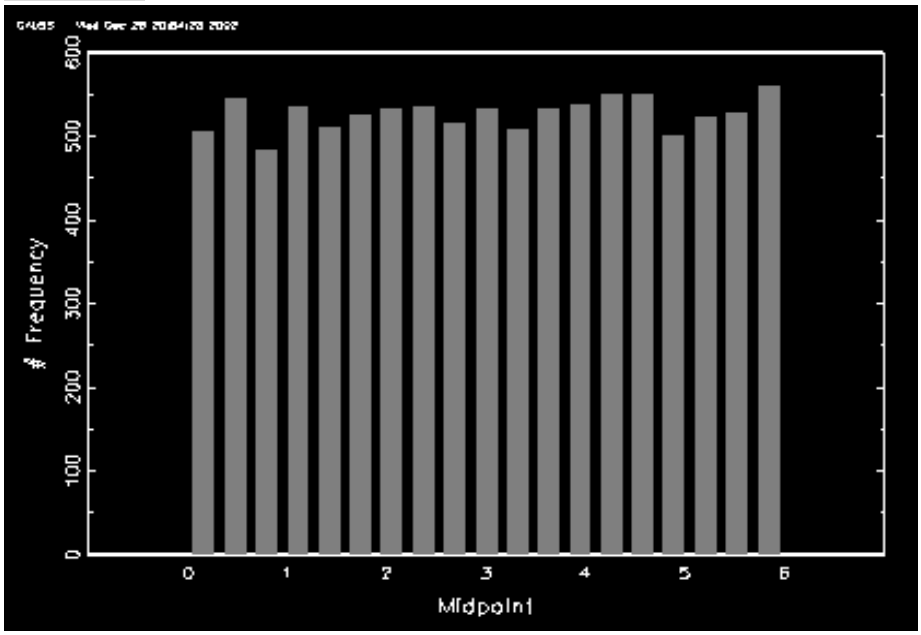
３行目　組込み関数 rndu で０と１の間の一様乱数を 10000×１（すなわち 10000 個）だけ発生させて、それに６倍したその結果（すなわち、０と６の間の一様乱数）を変数ｘに代入する。

４行目　グラフを描く pgraph のライブラリを呼び出す。

５行目　その pgraph のライブラリのグローバル変数（初期設定）を初期化する。

６行目　ｘのヒストグラムを 19 本の棒グラフで描く。

グラフ表示



上のように、これまでは一様乱数は０と１の間であったが、それが６倍されてその分一様に引き伸ばされて０と６の間の一様乱数になっていることがわかる。

今度は、一様乱数の引き伸ばしに加えて平行移動させてみる。

プログラム

```
new;
cls;
    x=2*rndu(10000,1)-1;
library pgraph;
graphset;
    call hist(x,19);
```

プログラム各行の説明

１行目　メモリを初期化する

２行目　画面をクリアする

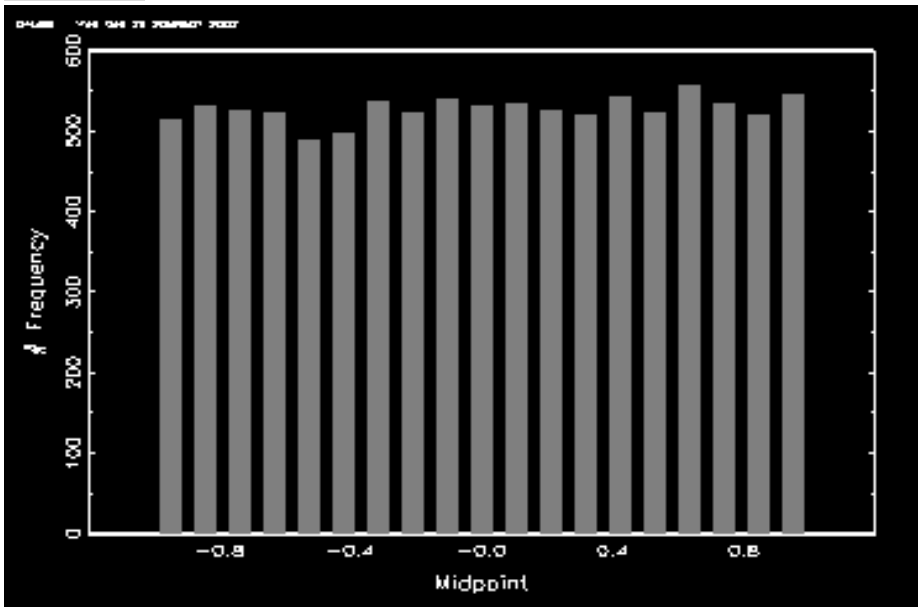
３行目　組込み関数 rndu で０と１の間の一様乱数を 10000×１（すなわち 10000 個）だけ発生させて、それに２倍したその結果（すなわち、０と２の間の一様乱数）から１を引いたもの（すなわち、－１と１の間の一様乱数）を変数ｘに代入する。

４行目　グラフを描く pgraph のライブラリを呼び出す。

５行目　その pgraph のライブラリのグローバル変数（初期設定）を初期化する。

６行目　ｘのヒストグラムを 19 本の棒グラフで描く。

グラフ表示



標準正規乱数の引き伸ばしと平行移動（一般的な正規乱数の作成）

記述統計量の章でデータの標準化を少し扱いました。データの標準化とは、データXを

$$Z = \frac{X - \mu}{\sigma}$$

というふうに、平均μを引くことで平均を0に中央化して、標準偏差σで割ることによって新しく作成されたデータの散らばりを表す標準偏差を1に標準化するものでした。したがって、どのような分布のデータも上のμを引いてσで割れば、平均0標準偏差1になります。これを逆方向に利用しましょう。例えば、N(0,1)の標準正規分布から始まって、その乱数を作成する関数自体にある数をかけたとしましょう。例えば、下の例では5をかけます。そうするとデータの標準化では5で割りましたが、ここではすでに標準正規分布に5をかけます。そうすると、標準化の類推からも容易に見当がつくように標準偏差が5に引き伸ばされた分布ができるはずです。

プログラム

```
new;
cls;
    x=5*rndn(10000,1);
print "N(0,25)" meanc(x) stdc(x);
library pgraph;
graphset;
    call hist(x,19);
```

プログラム各行の説明

1行目　メモリを初期化する

2行目　画面をクリアする

3行目　今度は組込み関数 rndn で平均0と分散1の標準正規乱数を 10000×1（すなわち 10000 個）だけ発生させて、それに5倍したその結果（すなわち、平均0と標準偏差5（分散5^2）の正規乱数）を変数xに代入する。

4行目　引用符の中のメッセージとともに、xの平均と標準偏差を計算した結果を2つ並べて画面表示する。

5行目　グラフを描くpgraphのライブラリを呼び出す。

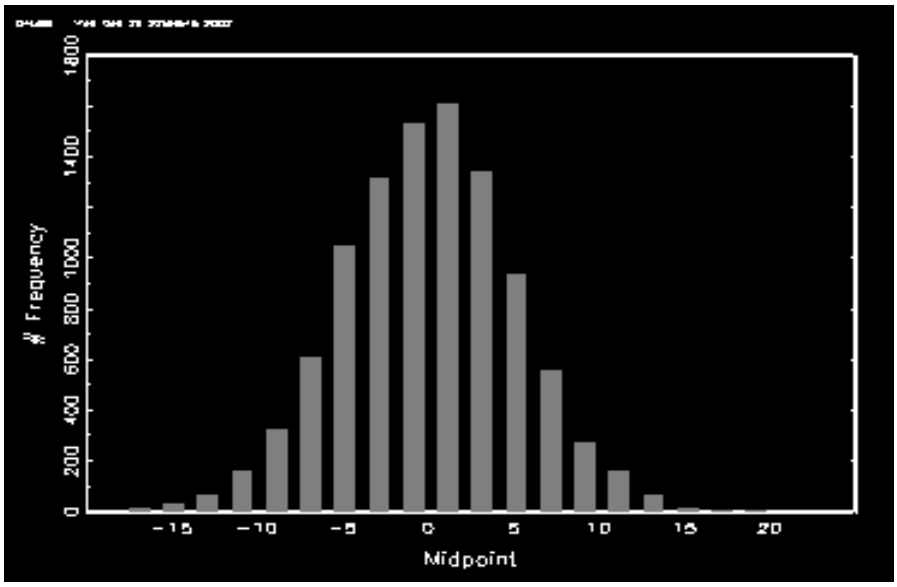
6行目　そのpgraphのライブラリのグローバル変数（初期設定）を初期化する。

7行目　xのヒストグラムを19本の棒グラフで描く。

画面表示

```
N(0,25)        0.019377683           4.9931309
```

グラフ表示



同様にして、今度は上の操作に加えて、平均を平行移動することによって変更してみましょう。標準偏差は５に伸ばしたままで、平均を０から１０に平行移動させます。ちょうど、平均が１０のデータを１０を引くことによって標準化する場合の逆方向を考えるわけです。すなわち、平均を１０にするにはただ単に引き伸ばした分布にその平均の数１０を足すだけです。括弧をつけて５で引き伸ばす前の乱数を平行移動させてはいけません。なぜならば、万が一そうすると引き伸ばす標準偏差を最後にかけることになってしまって、平行移動の数が１０ではなくなっていまうからです。したがって、ただ単にその数を足して平行移動するだけで足ります。

プログラム

```
new;
cls;
    x=5*rndn(10000,1)+10;
print "N(10,25)" meanc(x) stdc(x);
library pgraph;
graphset;
    call hist(x,19);
```

プログラム各行の説明

１行目　メモリを初期化する

２行目　画面をクリアする

３行目　今度は組込み関数 rndn で平均０と分散１の標準正規乱数を 10000×１（すなわち 10000 個）だけ発生させて、それに５倍して 10 加えた結果（すなわち、平均 10 と標準偏差５（分散 5^2）の正規乱数）を変数ｘに代入する。

４行目　引用符の中のメッセージとともに、ｘの平均と標準偏差を計算した結果を２つ並べて画面表示する。

５行目　グラフを描くpgraphのライブラリを呼び出す。

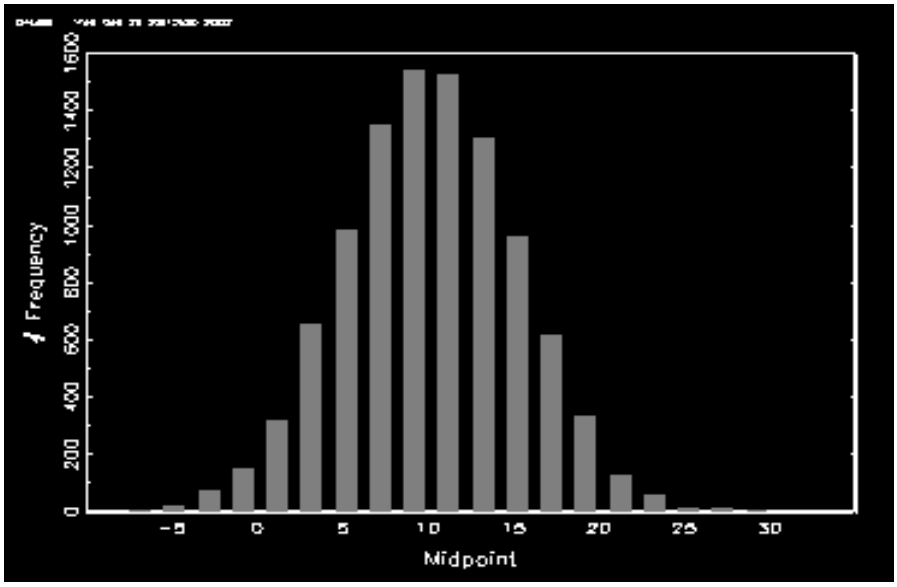
６行目　そのpgraphのライブラリのグローバル変数（初期設定）を初期化する。

７行目　ｘのヒストグラムを19本の棒グラフで描く。

画面表示

N(0,25)　　10.088126　　5.0189733

グラフ表示



上のように、ヒストグラムの中心は０から 10 に移動していて、乱数の平均がほぼ 10 になっていることがわかります。

すなわち、標準偏差σをかけて平均を加える

$$N(0,1) \to \sigma = 5, \mu = 10 \to \sigma N(0,1) + \mu$$

という操作を標準正規乱数 N(0,1)に加えていることになります。

一般的な一様乱数および正規乱数の関数化

プログラムのなかでこれまでのような操作を直接してもかまいませんが、見通しをよくするために１行定義関数 fn の機能を用いてそれぞれの一般形を関数として作成しておきましょう。正規乱数の場合は簡単で、上の変換関係をそのまま関数で表します。すなわち、関数に適当に名前をつけて）ここでは rndnorm としてやろう）

```
fn rndnorm(r,c,m,s)=s*rndn(r,c)+m;
```

というふうに、組込み関数 rndn のインプットの r と c（乱数の行数と列数）とともに平均

mと標準偏差ｓの合計４つをインプットにしてやって、もともとの rndn(r,c)に標準偏差ｓをかけて平均mを加えたものを関数名にイコールサインで定義しています。ここでmがマイナスの値をとるとその値がそのまま新しく作成された正規乱数の平均になります。なお標準偏差は分布のちらばりを表す尺度であって、マイナスの値をとることはできません。
一般化した一様乱数は少し面倒です。このようなときにこそ、関数を作っておけば後で簡単に利用できます。ここで関数名を適当に定めて（ここでは rndunif としてやろう）その新しい一様乱数の下限を low、上限を up としてやりましょう。少し前に行なった一様乱数の変換は、０と１の間の一様乱数を２倍して１を引いたもの（－１を加えたもの）が、結果として－1 と１の間の一様乱数になるというものでした。同様に考えて、もとの０と１の間の一様乱数を３倍して１を加えると、０と３の間の一様乱数を１平行移動したもの、すらわち、１と４の間の一様乱数になるはずです。これらを新しく作られる一様乱数の下限 low と上限 up に注目してあらためて考えなおしてやると、1 と４の距離（すなわち up-low）倍したものを下限 low だけ平行移動したものになるはずです。すなわち、rndu のもともとのインプットｒとｃ（行数と列数）とともに下限 low と上限 up の４つをインプットとして

fn rndunif(r,c,low,up)=(up-low)*rndu(r,c) +low;

この関係を rndu(r,c)に対して up と low の距離(up-low)をかけて、下限 low を加えるという(up-low)*rndu(r,c) +low という式を関数名にイコールサインでつないで定義しています。
実際、最初の例では、下限は－1 上限は１でその距離は２で、下限の－１だけ平行移動したものと等しくなっていて、この考え方は合っていることがわかります。

プログラム

```
new;
cls;
    x=rndunif(10000,1,5,10);
    y=rndnorm(10000,1,10,5);
print " U[5,10)" meanc(x) stdc(x) minc(x) maxc(x);
print "N(10,25)" meanc(y) stdc(y);

fn rndunif(r,c,low,up)=(up-low)*rndu(r,c) +low;
fn rndnorm(r,c,m,s)=s*rndn(r,c)+m;
```

画面表示

```
U[5,10)        7.5129421        1.4416948        5.0001635        9.9997065
N(10,25)        9.9674280        5.0209938
```

一様乱数の応用（判断分岐）

ふたたび０と１の間の一様乱数に戻って、これを用いて二者択一の判断分岐をしよう。すなわち、一様乱数で出てきた数が 0.5 よりも小さければ第１の選択をし、そうでなければ（0.5 以上であれば）第２の選択をするというものである。この章の一番最初に示した０と１の間の一様乱数のヒストグラムを見ても明らかなように、ここでは、第１の選択と第２の選択の起こる確立は回数を重ねてならしてみれば、ほぼ二分の一ずつになるはずであることを利用している。

プログラム

```
new;
cls;
x=rndu(1,1);
if x<0.5;
    print "Decision 1";
else;
    print "Decision 2";
endif;
```

プログラム各行の説明

１行目　メモリを初期化する
２行目　画面をクリアする
３行目　一様乱数を１×１の行列分（すなわち１個）作成して変数ｘに入れる
４行目　ｘが 0.5 よりも小さければ、次の命令を実行する
５行目　Decision 1 と画面表示する
６行目　そうでなければ（ｘが 0.5 以上であれば）、次の命令を実行する
７行目　Decision 2 と画面表示する
８行目　Ｉｆ条件分岐の最後の部分（Ｉｆを使えば必ず必要）

一様乱数の応用（整数乱数の作成）

サイコロを振ると１から６までのなかの整数のどれかの目がでる。しかも回数を重ねてならしてみるとそれぞれの目の出る確率は６分の１になるはずである。このことを一様乱数でやってみよう。やり方は、０と１の間の一様乱数を６倍することによって、０と６の間の一様乱数にしておいて、最後に小数点以下を切り上げしてやれば、{1,2,3,4,5,6}のなかのどれかの整数になるはずである。

プログラム

```
new;
cls;
x=6*rndu(10,1);
```

```
print ceil(x);
```

画面表示

```
3.0000000
5.0000000
5.0000000
3.0000000
2.0000000
2.0000000
3.0000000
1.0000000
4.0000000
4.0000000
```

プログラム各行の説明

１行目　メモリを初期化する
２行目　画面をクリアする
３行目　１０×１のディメンション（すなわち１０個の要素からなる列ベクトル）の０と１の間の一様乱数を６倍したもの（０と６の間の一様乱数のこと）をｘに入れる
４行目　ｘを切り上げの組込み関数 ceil によって変換されて出てきたものを画面表示する

単純であるので、

```
print ceil(6*rndu(10,1));
```

などと１行で示してやってもよいだろう。１から６までの整数ではなくて、１からｎまでの整数がほしい場合には６のところをｎに変更するだけでよい。なお、ceil は英語で天井を意味する。同様にして、GAUSS では切捨ては floor、四捨五入は round をそれぞれ使う。

一様乱数の応用（データの中から１つピックアップする）

実際のデータの要素がｎ個あって、その中から１つランダムにピックアップするにはどうしたらよいだろうか？それは、１からｎまでの整数の乱数を作成して、実際のデータのその番目（行番号）のデータを示せばよい。

プログラム

```
new;
cls;
data={78,68,89,92,60,42,74,80,78};
n=rows(data);
```

```
index=ceil(n*rndu(1,1));
print data[index];
```

プログラム各行の説明

１行目　メモリを初期化する

２行目　画面をクリアする

３行目　変数 data に９行１列の{78,68,89,92,60,42,74,80,78}を入れる

４行目　data の行数を調べて、それを変数ｎに入れる

５行目　（１×１のディメンションの）１つの０と１の間の一様乱数にｎをかけることで１からｎまでの一様乱数にしておいて、それをさらに組込み関数 ceil で切り上げることによって、１からｎまでの整数の乱数にして、その結果を index に入れる

６行目　変数 data の index 番目の数を表示する

上の例では乱数が実際のデータと結びついてデータをランダムにピックアップするという作業ができることを示しています。乱数はシミュレーションためだけのものではありません。

一様乱数の応用（ウエイトのミックス）

今、点$x_1(1,2)$と点$x_2(3,4)$の間の間を１－ａ対ａに分割する点を

$$(1-a)x_1+a x_2$$

と考えて、ａを適当に０から１までの数で（すなわち一様乱数で）ランダムにミックスしたものにしてみましょう。同様に、ｙについても同じａを用いてランダムにミックスします。乱数ａを１００個発生させて、100 通りの軌跡をｘｙ平面上にプロットしてみます。

プログラム

```
new;
cls;
x1=1; y1=2;
x2=3; y2=4;
a=rndu(100,1);
x=a.*x1+(1-a).*x2;
y=a.*y1+(1-a).*y2;
library pgraph;
graphset;
    xtics(0,5,1,0);
    ytics(0,5,1,0);
```

```
xy(x,y);
```

プログラム各行の説明

１行目　メモリを初期化する

２行目　画面をクリアする

３行目　ｘ１に１を代入する。ｙ１に２を代入する。

４行目　ｘ２に３を代入する。ｙ２に４を代入する。

５行目　組込み関数 rndu で０と１の間の一様乱数を 100×１（すなわち 100 個）だけ発生させて、その結果を変数ａに代入する。ここではウエイトａをランダムにする。

６行目　ウエイトａを用いて、ａx1+(ａ－１)x2 を変数ｘに代入する（すなわち、ｘとおく)。掛け算にドットがついているのは要素ごとの計算の意味。

７行目　ウエイトａを用いて、ａy1+(ａ－１)y2 を変数ｙに代入する（すなわち、ｘとおく)。掛け算にドットがついているのは要素ごとの計算の意味。

８行目　グラフを描く pgraph のライブラリを呼び出す。

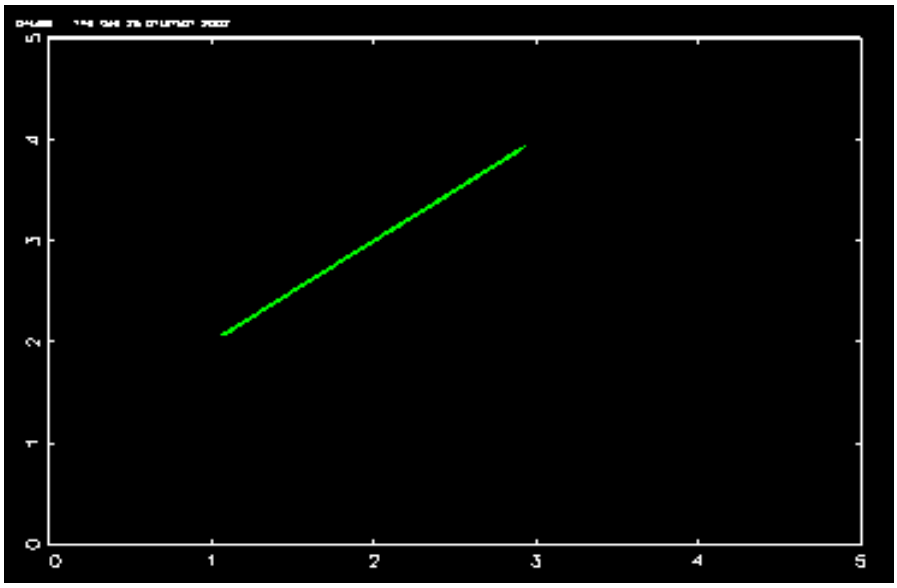
９行目　その pgraph のライブラリのグローバル変数（初期設定）を初期化する。

１０行目　グラフのｘ軸は最小値０最大値５で１ずつの大目盛、小目盛は０で無し。

１１行目　グラフのｙ軸は最小値０最大値５で１ずつの大目盛、小目盛は０で無し

１２行目　ｘとｙをｘｙグラフでプロットする。

グラフ表示



乱数のシードについて

プログラム

```
new;
cls;
rndseed 1000;
print rndu(1,1);
print rndn(1,1);
```

プログラム各行の説明

１行目　メモリを初期化する
２行目　画面をクリアする
３行目　乱数のシードを 1000 に強制的に変更して同じ結果が出るようにする
４行目　組込み関数 rndu で０と１の間の一様乱数を１×１（すなわち１個）だけ発生させて、その結果を画面表示する
５行目　組込み関数 rndn で標準正規乱数を１×１（すなわち１個）だけ発生させて、その結果を画面表示する

GAUSS ではその立ち上げ時刻に応じて乱数を発生させるおおもとのシードという数を内部で自動的に決定しています。同じマシンでは同じ乱数結果が出ることはまずありませんが、異なるマシンではタイミングによって同じ乱数結果もでることもあります。ただし、同じ状況を作り出すには、この乱数のおおもとの数を強制的に変更してだれでもいつでもどのマシンでも一定の乱数を出す必要があります。そのために設定するのがシードとよばれるものです。乱数の組込み関数を呼び出すよりも前に、

```
rndseed 整数;
```

というふうに書いて設定をします。この他に、乱数のおおもとになる係数部分と定数部分も設定を変えることができますが、学術論文レベルでも、通常の使用でも、まずそれらの変更はしません。このシードの変更だけをすることになります。

注意事項

１）シードの設定は冒頭で１回限りで十分である。多くとも、各乱数につき１つずつのシード設定をするのみで足りる。しかし、その場合もまとめて１回でも足りる。
２）旧式の GAUSS では rndus と rndns というふうにｓの字がついてその第３パラメータにシードを設定できるものもあったが、現在は廃止された。
３）シードを１から順に増していくなどといった方法では、ステートの変化を表すことはできないばかりか、そうしてしまうと理論上は乱数とは言いがたい。

練習問題

【問１】

１万個からなる－２と２と間に一様に分布する乱数を作成して、それをグラフにしよう。

【問２】

１万個からなる平均－２「分散」２の正規分布にしたがう乱数を作成して、それが実際に平均－２分散２に近い値になっているかを確認した上、それをグラフにしよう。

【問３】

標準の GAUSS の組込み関数として、他にどんな乱数があるのかを調べてみよう。

発展

⇒　Luenberger　サイコロの例：　6.2a　6.2b　6.2c

回転盤の例：　6.3a　6.3b