0.S4 ジュニア版 特別章(4)　等間隔と数列そして準乱数

　ここでは、等間隔な点のプログラミングや２進数などのプログラミングから始めて、それらが乱数や準乱数と呼ばれるものに発展することを学習しましょう。単純なことではありますが、極めて最先端であると言われている分野への掛け橋にもなる話です。

0 と１の間に等間隔な点をとる

　例えば、０と１の間を 10 等分をして、その始点と終点も含めて 11 の点を作成してやりましょう。すなわち、0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0 の 11 の点です。

プログラム

```
new;
cls;
x=seqa(0,1,11);
x=x/10;
print x;
```

画面表示

```
 0.00000000
 0.10000000
 0.20000000
 0.30000000
 0.40000000
 0.50000000
 0.60000000
 0.70000000
 0.80000000
 0.90000000
  1.0000000
```

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　スクリーンをクリアする。
３行目　組込み関数 seqa で始点０、増分１ステップで 11 個の数列{0,1,2,3,4,5,6,7,8,9,10}を作成して、変数ｘに代入する。
４行目　ベクトル変数ｘのおのおのの要素を 10 で割ったものをｘに代入する。
５行目　ｘの内容を画面表示させる。

組込み関数seqaは等間隔の数列を作成する関数です。その第１要素が「始点」、第２要素が「各ステップの増分」、そして第３要素が「全体の個数」です。seqa(0,1,11)とあるのは、始

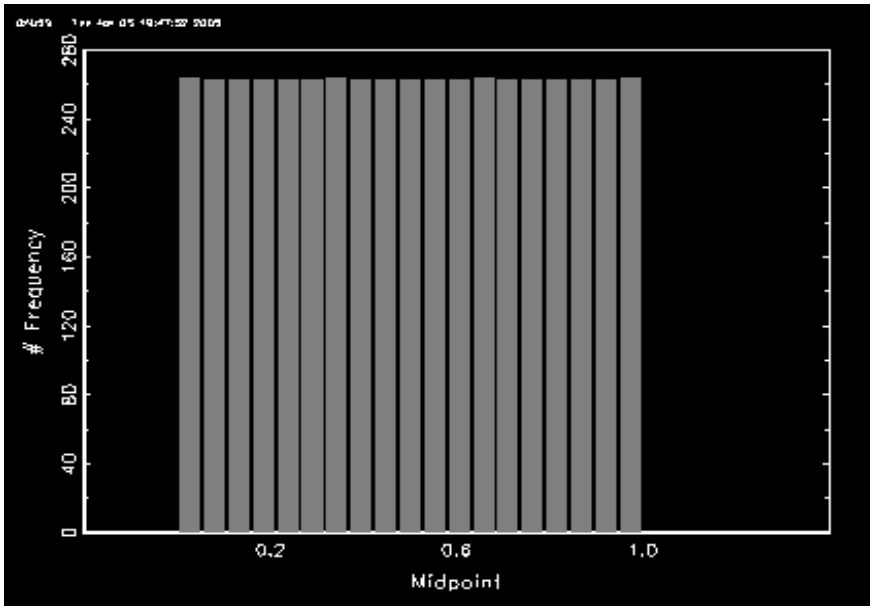
点０から始めて１ずつの増分で11個分ということです。始点と終点の両方を含むので、11個分を考えています。結果は、0,1,2,3,4,5,6,7,8,9,10のベクトルになります。それを10で割ると、最初が０で最後が１、その間が0.1,0.2, … , 0.8, 0.9のように等間隔に並ぶ小数となります。これらは、当然のことながら等間隔に並んでいます。

　さらに間隔をせばめて、それらの点が０と１の間に均等に分布しているか、すなわち一様分布にしたがっているかを見ましょう。ここでは、一般的な個数ｎに十分に大きい数、例えば、5000を与えて、結果をグラフ表示させてみましょう。

プログラム

```
new;
cls;
n=5000;
x=seqa(0,1,n+1);
x=x/n;
library pgraph;
graphset;
call hist(x,19);
```

グラフ表示



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　スクリーンをクリアする。

３行目　変数ｎに 5000 を代入する。

４行目　組込み関数 seqa で始点０、増分１でｎ＋１個の数列を作成し、変数ｘに代入。
５行目　ベクトル変数ｘのおのおのの要素をｎで割ったものをｘに代入する。
６行目　グラフィックライブラリpgraphを呼び出す。
７行目　そのグローバル変数を初期化する。
８行目　そのライブラリ内にある関数histを呼んで、19の棒でヒストグラムを描く。

グラフで表されるように、きれいに一様分布にしたがっていることがわかります。ここで用いたhistというpgraphグラフィックライブラリに属している関数は、変数ベクトルｘのヒストグラムを作成する関数です。その第２要素の19という数は、整数であれば何でもよいのですが、ここでは19を用いているので、19個の棒で表すという意味です。ここが９であれば、もう少し粗い９個の棒で、29であれば、もう少し細かい29の棒で表されます。

一様乱数U(0,1)を生成する

上にプログラムでは、０と１の間に一様に点は散らばっていますが、「乱数」ではありません。なぜならば、常に最初が０で最後が１、ｎの数によって細分化される間隔は変化しますが、それでも０から１へと小さいものから大きいものへと並んでいます。これを乱数にする、すなわち、でたらめに並べ替えるにはどうしたらよいでしょうか？

それには、変数のインデックスナンバーを混ぜます。上のグラフを描くプログラムに、

```
print x;
```

の１行を加えてみましょう。そうすると、ｘの内容の画面表示は、０から１へと細かく並んでいることがわかると思います。ここで、

```
index={1,9,20};
print x[index];
```

としてやりましょう。そうすると、画面表示はもともとのｘの内容の１番目と９番目と20番目だけが表示されます。こうしたGAUSSの変数のインデックスナンバーの操作をする簡単なプログラムで乱数にしてみます。すなわち、

プログラム

```
new;
cls;
n=10;
x=seqa(0,1,n+1);
x=x/n;
index=rankindx(rndu(n+1,1),1);
x=x[index];
print x;
```

画面表示

0.20000000
0.70000000
0.80000000
0.40000000
0.90000000
0.60000000
0.00000000
1.0000000
0.10000000
0.50000000
0.30000000

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　スクリーンをクリアする。
３行目　変数ｎに 10 を代入する。
４行目　組込み関数 seqa で始点０、増分１でｎ＋１個の数列を作成し、変数ｘに代入。
５行目　ベクトル変数ｘのおのおのの要素をｎで割ったものをｘに代入する。
６行目　組込み関数rndu(n+1,1)でn+1個の一様乱数を作成しておいて、その結果を組込み関数rnkindex(x,1)でｘのところに入れて（１というのは１列目を判断するという意味、ここでは１列しかないので1）順位づけする。その結果の順位のインデックスを変数indexに代入する。
７行目　変数ｘのインデックス番目の数をまとめて変更したものを変数ｘに代入する。
８行目　変数ｘの内容を画面表示する。

という具合のプログラムになります。ここでは実際に乱数になっていることを確認するためにｎを10程度に小さくしていますが十分に大きい数にしても同じことです。

　なお、この乱数は、間隔のエッジを採用する乱数の作り方です。もし、この一様乱数をあとで何かの変換するなど、０と１を含んでは都合が悪い場合には、最初の０と最後の１を取り除くプログラムにしてみてください。

プログラム

```
new;
cls;
n=9;
x=seqa(1,1,n);
```

```
x=x/(n+1);
index=rankindx(rndu(n,1),1);
x=x[index];
print x;
```

結果の表示とプログラムの解説は省略しますが、ここではｎの数自体が、乱数全体の個数に等しくなっています。１から９までの数列を作成しておいて、それをｎ＋１の10で割るということをしています。これにより、そもそもが１からはじまっていますから０は含まれませんし、ｎ＋１の１つ手前のｎで数列は終わっていますから、一番大きい数はn/(n+1)の9/10であるはずです。これらのｎ個のインデックスを混ぜて、乱数完成となります。

0と１の間に等間隔の中点をとる

上のようなエッジを採用するものではなくて、短冊状に等間隔にしたものの中点を採用する方法を考えて見ましょう。そうすることによって、そもそも、最初の０と最後の１については考えなくてもよくなります。

プログラム

```
new;
cls;
n=10;
x=seqa(1,1,n);
x=(x-0.5)/n;
print x;
```

画面表示

```
    0.050000000
     0.15000000
     0.25000000
     0.35000000
     0.45000000
     0.55000000
     0.65000000
     0.75000000
     0.85000000
     0.95000000
```

今度は、ｎが 10 の時には、０と 1 の間を 10 等分し、その短冊の中点を採用していることが、結果からわかると思います。すらわち、0 と 0.1 の間の点は 0.05、その次の 0.1 と 0.2 の中点は 0.15、以下同様にして、最終は 0.9 と１の中点の 0.95 まで計 10 の短冊の中点の 10 点が計算されています。

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　スクリーンをクリアする。
３行目　変数ｎに 10 を代入する。
４行目　組込み関数 seqa で始点１、増分１でｎ個の数列を作成し、変数ｘに代入する。
５行目　ｘのおのおのから 0.5 を引いたものをさらにｎで割ったものをｘに代入する。
６行目　変数ｘの内容を画面表示する。

これをエッジを採用した以前の乱数の時と同じようにすると、以下のようなプログラムになります。index=rankindx(rndu(n,1),1);とx=x[index];の２行が加わるだけです。

プログラム

```
new;
cls;
n=10;
x=seqa(1,1,n);
x=(x-0.5)/n;
index=rankindx(rndu(n,1),1);
x=x[index];
print x;
```

結果は、それぞれの回ごとに、以前の結果の数字がランダムになって出てきます。グラフにして、これらが０と１の間に一様に並んでいるのかを確かめるためには、以前と全く同様にして、以下の３行をプログラムに加えてみてください。

```
library pgraph;
graphset;
call hist(x,19);
```

なお、繰り返しになりますが、この各等間隔の短冊の中点を採用する方法では、始点の０と終点の１は最初から含まれておらず、あとでそれらの処理をする必要は全くありません。

これら２つの方法（エッジを採用する方法と中点を採用する方法）だけでも、乱数として相当の力を発揮します。GAUSSの組込み関数であるrndu(n,1)とともに、これからも使っていってください。これらは、１変数の場合、学術的にも十分な信頼性をもって使用することができます。

２進数の数列を作成する

今度は趣向を変えて、等間隔という概念から一様に散らばった乱数を作成するのではな

くて、N進法、手始めに２進数を作成して、究極的には、その性質を用いて０と１の間に一様に散らばる乱数を生成を目指します。

プログラム

```
new;
cls;
n=8;
k=sumc(n.>=(2^seqa(0,1,16)));
x=zeros(1,k);
xx=zeros(n,k);
j=1;
do while j<=n;
    x[1]=x[1]+1;
    i=2;
    do while i<=cols(x);
        if x[i-1]==2;
            x[i-1]=0;
            x[i]=x[i]+1;
        endif;
        i=i+1;
    endo;
    xx[j,.]=x;
    j=j+1;
endo;
print/rz xx;
```

画面表示

1	0	0	0
0	1	0	0
1	1	0	0
0	0	1	0
1	0	1	0
0	1	1	0
1	1	1	0
0	0	0	1

上では、それぞれの行が（横向きに）０と１からなる２進数になっています。ただし、左から数えて２進数になっています。リバースすれば簡単に右と左とを交換できますが、こ

こでは次なるステップのために、あえてこのままにしておきます。

プログラム

```
new;
cls;
n=8;
k=sumc(n.>=(2^seqa(0,1,16)));
x=zeros(1,k);
xx=zeros(n,k);
j=1;
do while j<=n;
    x[1]=x[1]+1;
    i=2;
    do while i<=k;
        if x[i-1]==2;
            x[i-1]=0;
            x[i]=x[i]+1;
        endif;
        i=i+1;
    endo;
    xx[j,.]=x;
    j=j+1;
endo;
c=2^(-seqa(1,1,k));
xx=xx*c;
print xx;
```

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　スクリーンをクリアする。

３行目　変数ｎに８を代入する。

４行目　2^seqa(0,1,16)で{1,2,4,8,16,32,64,128,256,…}を作成しておいて、それが n 以下であるものの数を数えて、その合計を変数ｋに代入する。すなわち n＝８のときそれ以下であるのは{1,2,4,8}の４つだからｋは８になる。

５行目　変数初期化のため、変数ｘに１行ｋ列の零からなる行ベクトルを代入する。

６行目　変数初期化のため、変数xxにｎ行ｋ列の零行列を代入する。

７行目　変数ｊに１を代入して、以下のｊループをここから始める。

８行目～20行目　ｊを１ずつ増す間に、ｊがｎ以下であるかぎりループを回す。この間、

行ベクトルｘをその都度計算をし、それを行列xxのｊ行目に置く。

９行目　行ベクトルｘの第１番目の要素に１加えたものを、あらためてｘの第１番目の要素とする。

10行目　変数ｉに２を代入して、以下のｉループをここから始める。

11行目～17行目　ｉを１ずつ増す間に、ｉがｋい家であるかぎりループを回す。この間もし１つ前のｘ[i-1]が２になれば、１つ桁上がりをして１つ前のx[i-1]を０にそしてx[i]を１加えたものにそれぞれ設定し直す。すなわちこの作業を実際の桁上がりと同じように、ｉが２からｋまですべて１つずつずらせて行う。

21行目　変数ｃに{2^{-1},2^{-2},2^{-3},2^{-4}}を代入する。

22行目　行列xxの後ろから列ベクトルｃをかけて、その答えをxxに代入する。

23行目　xxの内容（もともとの行列xxのそれぞれの行の左端を小数点の場所とした場合の２進数表示である列ベクトル）を画面表示する。

画面表示

0.50000000
0.25000000
0.75000000
0.12500000
0.62500000
0.37500000
0.87500000
0.062500000

これは、まえの左から読む２進数の結果の行列xxに、最初が2^{-1}、次が2^{-2}、その次が2^{-3}、最終ｋ＝４のときのこの場合の最後が2^{-4}からなる列ベクトルｃを後ろからかけたxx＊ｃの計算結果を表示したものです。ちょうど前の結果行列の左端に小数点があるものとして、それぞれ、２進法で

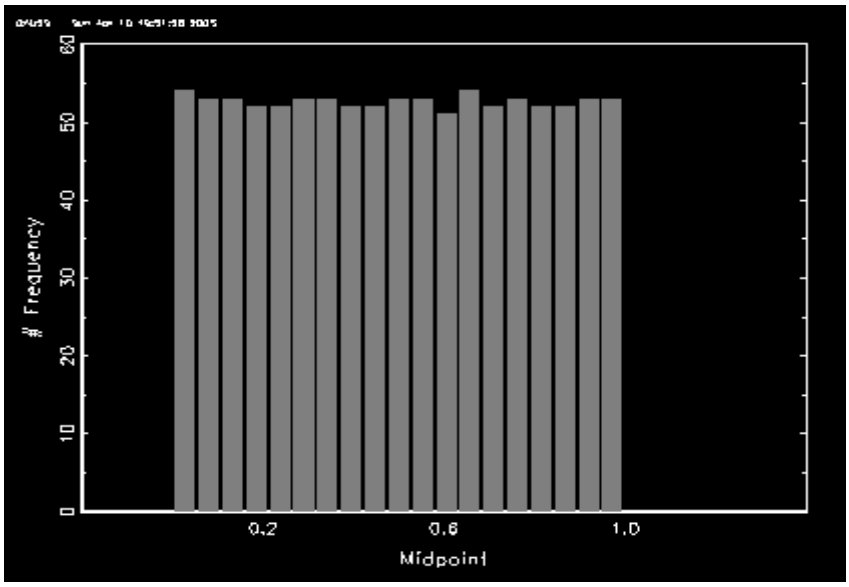
0.1000
0.0100
0.1100
0.0010
0.1010
0.0110
0.1110
0.0001

というふうに読んだ時に、それを10進法に換算した結果に相当します。なお、これをｎが1000という十分に大きな数に直したものを、プログラム末尾を

```
library pgraph;
graphset;
call hist(xx,19);
```

というふうにxxの内容をグラフ表示させるようにすると

グラフ表示



という具合に０と１の間の一様乱数になっていることがわかります。この場合、実際に、最初の計算で２進数の０ではなくて１から始まっているので、０は最初から除かれていますし、計算上結果としての１も出てこないプログラムになっています。これは、すべての準乱数の基礎となるvan der Corput列と呼ばれるものです。

　さらに、上のプログラムの２であるとことをi=2のところを除いて、すべてbに直して、冒頭で例えばb=3;などとすると３進数によるものになります。

プログラム

```
new;
cls;
n=1000;
b=3;
k=sumc(n.>=(b^seqa(0,1,16)));
x=zeros(1,k);
xx=zeros(n,k);
j=1;
do while j<=n;
    x[1]=x[1]+1;
    i=2;
```

```
    do while i<=cols(x);
        if x[i-1]==b;
            x[i-1]=0;
            x[i]=x[i]+1;
        endif;
        i=i+1;
    endo;
    xx[j,.]=x;
    j=j+1;
endo;
c=b^(-seqa(1,1,k));
xx=xx*c;
library pgraph;
graphset;
call hist(xx,19);
```

グラフは省略しますが、ｂ＝２のときの２進数のケースと同様に乱数のパターンは異なりますが０と１の間にうまく散らばった一様乱数にしたがうものとなります。なお、そのほかの４以上の整数に対しても同様に０と１の間にうまく散らばることがわかっています。

詳しい計算はここでは扱いませんが、予備的な知識としてお教えしておくと、ｂが2,3,5,7,11,…などの素数の数であらわされる結果の組、例えば、３次元の場合、１次元目がb=2で、２次元目がb＝3で、３次元目がb=5などとなっているものをHalton列と呼びます。また、そのHalton列の最初の１次元を冒頭で扱った等間隔のエッジからなる数列とし、以下同様に素数の数から計算される数列を重ねていったものをHammersley列と呼びます。

フィボナッチ数とその黄金比を用いた乱数

まず、数学やプログラムの基礎や株式のテクニカル分析でも有名なフィボナッチ数のプログラムを行います。これは、ｘの第１番目と第２番目をそれぞれ１としてやったときに、２つ前の数と１つ前の数の和がその数となるように再帰的に考えていく数列です。

ｘ1＝1
ｘ2＝1
ｘ3＝ｘ1＋ｘ2＝1＋1＝2
ｘ4＝ｘ2＋ｘ3＝1＋2＝3
ｘ5＝ｘ3＋ｘ4＝2＋3＝5
．．．．．．．．．．．．．．．．．．．．．

となっていく数列のことです。プログラムは至って簡単で次のようになります。

プログラム

```
new;
cls;
k=10;
y=zeros(k,1);
y[1]=1;
y[2]=1;
i=3;
do while i<=k;
    y[i]=y[i-1]+y[i-2];
    i=i+1;
endo;
print y;
```

画面表示

```
        1.0000000
        1.0000000
        2.0000000
        3.0000000
        5.0000000
        8.0000000
        13.000000
        21.000000
        34.000000
        55.000000
```

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　スクリーンをクリアする。
３行目　変数ｋに 10 を代入する。
４行目　変数初期化のため、変数ｙにｋ行１列の零ベクトルを代入する。
５行目　変数ｙの第１番目に１を代入する。
６行目　変数ｙの第２番目に１を代入する。
７行目　変数ｉに３を代入し、ここからｉループを始める。
８行目～11行目　１ずつ増すｉループをｉがｋ以下であるかぎり回す。
12行目　ｙの２つ前とｙの１つ前の数を足し合わせて、現在ｉ番目のｙの要素のところにその答えを代入する。

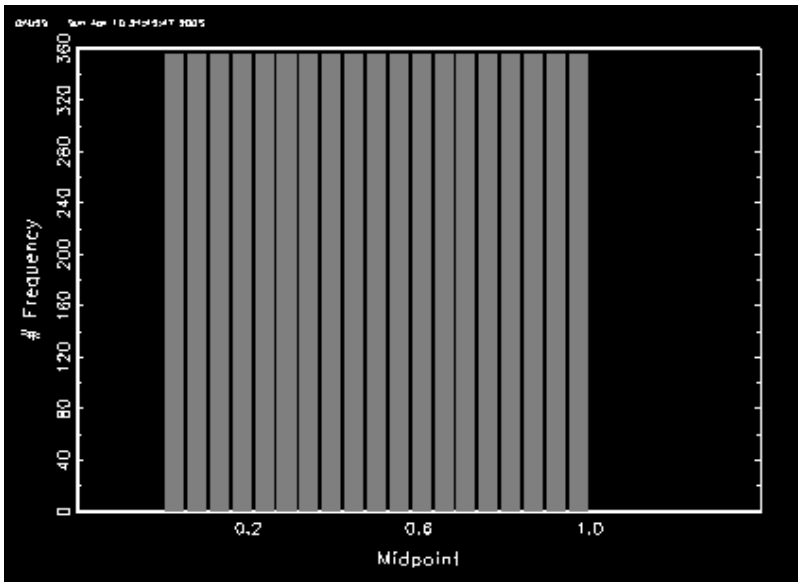
ここでプログラムしたフィボナッチ数列とは、「隣り合う２つの数を加えると次の数なる」

という数列のことでした。それら隣り合う数の比、すなわち**大きい方の数／小さい方の数**は、ｋが増していくにしたがって、1.6..なにがしらの数に近づいて行きます。この比のことを黄金比などと呼ぶことがありますが、ここでは、その黄金比を形成する隣り合う数の比の逆数を用いて、それに数列をずらせて代入することで０と１の間に一様に散らばる性質をもつフィボナッチ乱数を生成することができます。簡単化のため、k=30までのフィボナッチ数をあらかじめ計算したものを数列Fkとして与えておきましょう。

プログラム

```
new;
cls;
k=20;
Fk={1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597,2584,4181,6765,
      10946,17711,28657,46368,75025,121393,196418,317811,514229,832040};
r=Fk[k-1]/Fk[k];
i=seqa(1,1,Fk[k]-1);
x=(i*r)-floor(i*r);
library pgraph;
graphset;
call hist(x,19);
```

グラフ表示



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　スクリーンをクリアする。
３行目　変数ｋに 20 を代入する。
４行目～５行目 変数 Fk にフィボナッチ数を列ベクトルとしてあらかじめ代入しておく。
６行目　ｒに隣り合うフィボナッチ数の比の逆数を設定する
７行目　変数ｉに１から１の増分でFk[k]-1までの{1,2,3,4,…,Fk[k]-1}の数列を代入。
８行目　ｉ＊ｒとその整数部分の差の値を計算したものをｘに代入する。
９行目　グラフィックライブラリpgraphを呼び出す。
10行目　そのグローバル変数を初期化する。
11行目　変数ｘの内容を19本の棒でヒストグラム表示する。

この場合、乱数の個数はｋによりますが、０と１の間に一様に散らばっていることが見て取れると思います。ここで、組込み関数floorはその数の整数部分（すなわちfloorして小数点以下を切り捨てたもの）を計算するものです。

なお、このフィボナッチ乱数も、等間隔のエッジの数列と組み合わせれば、２次元のフィボナッチ数列となります。

練習問題

【問１】van der Corput の乱数において、b=2 と b=3 のそれぞれの場合の乱数を合成することによって、Halton 列と呼ばれる２次元の乱数を作成せよ。
ただし、n＝1000 とする。
(ヒント)b=3 と設定したプログラムの冒頭の部分とグラフィックライブラリを呼び出す以降の部分以外の中間部分を２回同じもの書く。そして、最初の部分の変数 xx をすべて xx1 とし、２回目の部分の変数 xx をすべて xx2 とし、それぞれのｂ＝のところを最初を２、２回目の部分を３とする。最後に、グラフィックライブライを呼び出す直前で、xx＝xx1~xx2;とマージしてやる。

【問２】最後で行ったフィボナッチ乱数に i/(n+1)の数列(ｉ＝1,2,3,4,…y の列数)を合成することにより、２次元フィボナッチ乱数を完成させよ。
(ヒント)変数ｙの列数は rows(y)でわかる。x=seqa(1,1,rows(y))/(rows(y)+1);
として、等間隔の数列を作成し、フィボナッチ乱数ｙに対して xx＝x~y;としてやれば、２次元となる。この変数 xx の内容をグラフ表示させる。