5.21 モンテカルロシミュレーション　Convolution 法　　ver.0.1

もう１つ乱数の生成法として、普段はそれとは気がつかずに用いているものとして、ある分布の乱数を足し合わせるという convolution がある。この方法をもう一度再確認してみる。

ベルヌーイ試行と２項分布

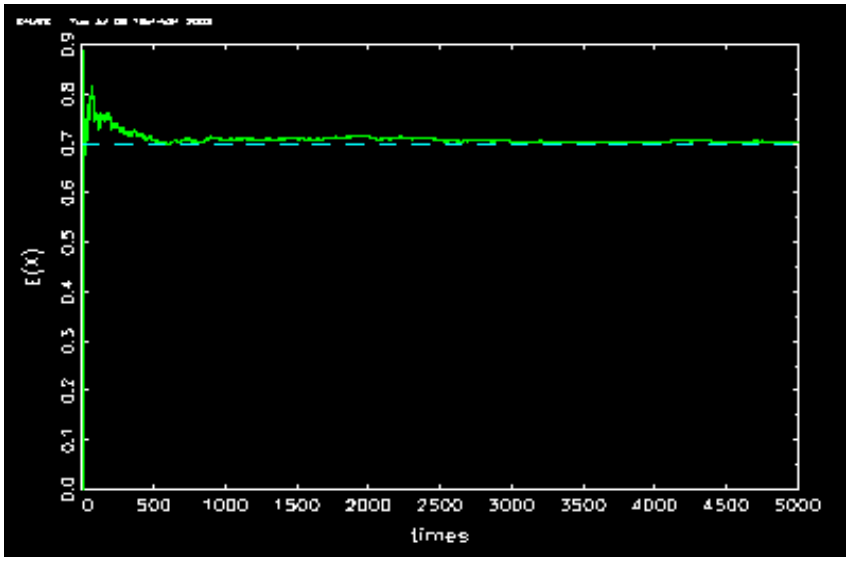
まずは、成功確率ｐとしたときＸ＝１となるベルヌーイ試行をやってみよう。以下では、それを times 回繰り返した時に、その平均が成功確率ｐに近づいていくことを示す。

プログラム

```
new; cls;
p=0.7;
times=5000;
call bernoulli(p,times);

proc bernoulli(p,times);
    local m,i,e;
    m=zeros(times,1);
    i=1;
    do while i<=times;
        if rndu(1,1)<p;
            m[i]=1;             /* success */
        else;
            m[i]=0;             /* failure */
        endif;
        i=i+1;
    endo;
/* E[X] at each time */
    print "E[X]=" sumc(m)/times;
    e=cumsumc(m)./seqa(1,1,times);
/* Plot */
    library pgraph;
    graphset;
    xlabel("times");
    ylabel("E(X)");
    xy(seqa(1,1,times),e~p*ones(times,1));
```

```
    retp(m);
endp;
```

グラフ表示



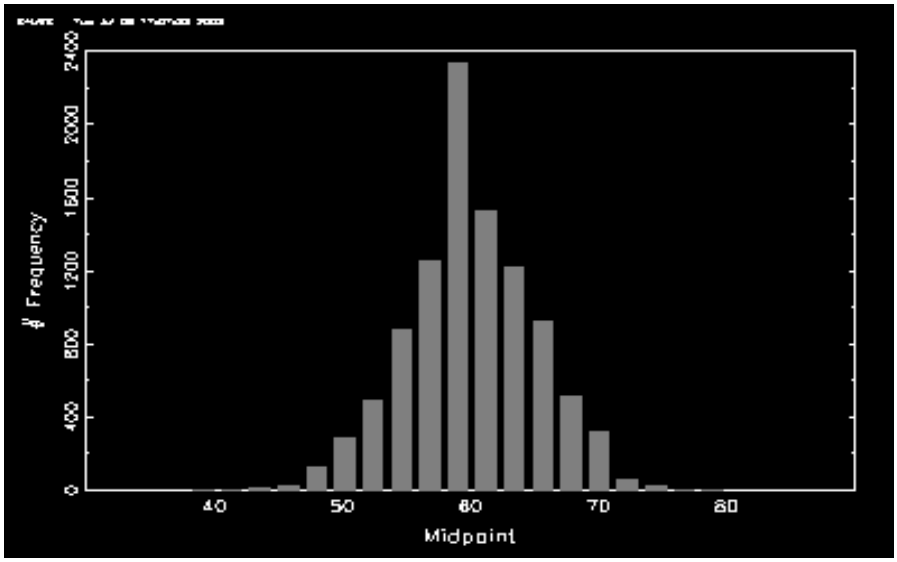
次にこのベルヌーイ試行をｎ回足し合わせた分布を考える。すなわち、２項分布 binomial である。いま、パラメーターｎおよびｐに対して、乱数をｒ個だけ作成してみる。

プログラム

```
new; cls;
r=10000;
n=100;
p=0.6;
call rndbino(r,n,p);

proc rndbino(r,n,p);
    local X,m,i,j;
    X=zeros(r,1);
    j=1;
    do while j<=r;
        m=zeros(n,1);
        i=1;
        do while i<=n;
            if rndu(1,1)<p;
                m[i]=1;                 /* success */
            else;
                m[i]=0;                 /* failure */
```

```
            endif;
            i=i+1;
        endo;
        X[j]=sumc(m);                   /* sum of n trials */
        j=j+1;
    endo;
/* Histogram of X */
    library pgraph;
    graphset;
    call hist(X,19);
    retp(X);
endp;
```

グラフ表示



実際にこうしたベルヌーイ試行のｎ個の足し算が２項分布のパラメーターと平均および分散の関係にあてはまっているかをシミュレーションで調べてみる。

プログラム

```
new; cls;
n=100;
p=0.6;
times=10000;
call binosim(n,p,times);

proc binosim(n,p,times);
    local X,m,i,j;
```

```
    X=zeros(times,1);
    j=1;
    do while j<=times;
        m=zeros(n,1);
        i=1;
        do while i<=n;
            if rndu(1,1)<p;
                m[i]=1;                 /* success */
            else;
                m[i]=0;                 /* failure */
            endif;
            i=i+1;
        endo;
        X[j]=sumc(m);                   /* sum of n trials */
        j=j+1;
    endo;
/* Display statistics */
    print/lz "n=" n;
    print/lz "p=" p;
    print "throretical     E[X]=n*p=          " n*p;
    print "empirical        E[X]=             " meanc(X);
    print "theoretical Var[X]=n*p*(1-p)=" n*p*(1-p);
    print "empirical     Var[X]=             " vcx(X);
    retp(X);
endp;
```

画面表示

```
n= 100
p= 0.6
throretical    E[X]=n*p=              60.000000
empirical      E[X]=                  60.040700
theoretical Var[X]=n*p*(1-p)=         24.000000
empirical    Var[X]=                  23.955639
```

確かに多くの回数（上では 5000 回）繰り返すと、理論上の平均ｎｐおよび分散ｎｐ(1－ｐ)にシミュレーションの実験値もたいへん近い値をとることがわかる。

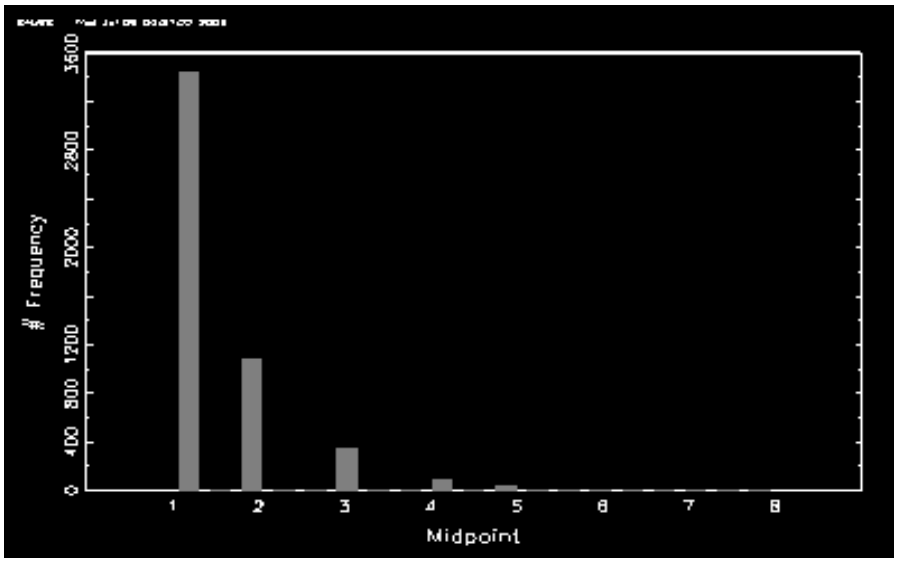
幾何分布とパスカル分布

　ベルヌーイ試行をｎ回行なったときの成功回数を数えたものが２項分布であった。幾何分布は、成功回数ではなくて初めに成功したのは何番目であるかを数えるものである。まずは、この幾何分布を乱数生成してみる。

プログラム

```
p=0.7;
times=5000;
call rndgeom(times,p);

proc rndgeom(times,p);
    local X,i,n;
    X=zeros(times,1);
    i=1;
    do while i<=times;
        n=1;
        do while n<=10000;        /* cutoff value */
            if rndu(1,1)<p;
                X[i]=n;             /* first success after n bernoulli trials */
                break;
            endif;
            n=n+1;
        endo;
        i=i+1;
    endo;
/* Histogram of X */
    library pgraph;
    graphset;
    call hist(X,19);
    retp(X);
endp;
```

グラフ表示



以下では、実際にこの幾何分布にしたがう乱数がその理論上の平均と分散に近い値となるような分布になっているのかをシミュレーションで調べる。

プログラム

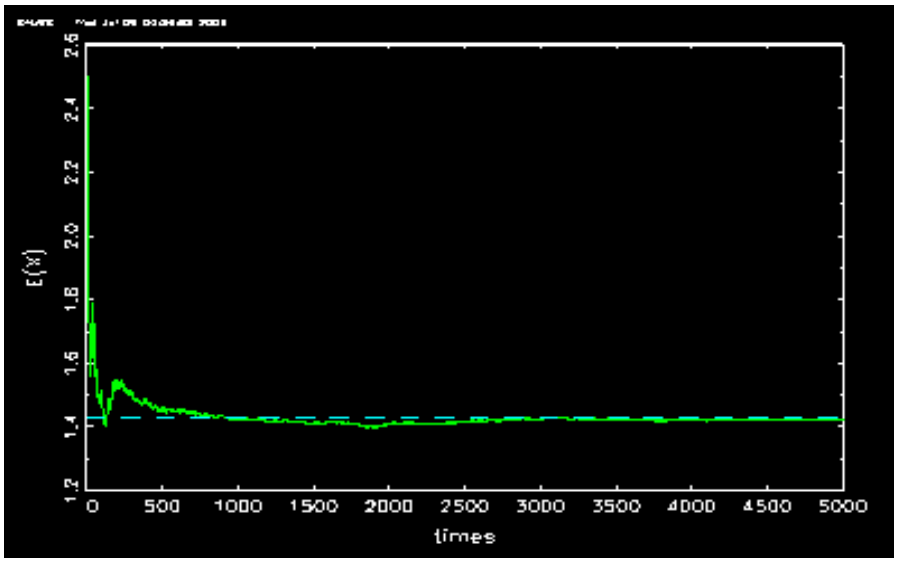
```
new; cls;
p=0.7;
times=5000;
call simgeom(p,times);

proc simgeom(p,times);
    local X,i,n,e;
    X=zeros(times,1);
    i=1;
    do while i<=times;
        n=1;
        do while n<=10000;        /* cutoff value */
            if rndu(1,1)<p;
                X[i]=n;             /* first success after n bernoulli trials */
                break;
            endif;
            n=n+1;
        endo;
        i=i+1;
```

```
    endo;
/* Display statistics */
    print "Theoretical E[X]=1/p=" 1/p;
    print "Empirical     E[X]=       " sumc(X)/times;
    print "Theoretical Var[X]=(1-p)/p^2=" (1-p)/p^2;
    print "Empirical     Var[X]=                " vcx(X);
    e=cumsumc(X)./seqa(1,1,times);
/* Plot */
    library pgraph;
    graphset;
    pqgwin auto;
    xlabel("times");
    ylabel("E(X)");
    xy(seqa(1,1,times),e~1/p*ones(times,1));
    retp(X);
endp;
```

画面表示

```
Theoretical E[X]=1/p=          1.4285714
Empirical   E[X]=                1.4242000
Theoretical Var[X]=(1-p)/p^2=        0.61224490
Empirical   Var[X]=                  0.61157668
```

グラフ表示



上のグラフは平均が理論上の平均に近づいていっていることを示している。

次に、この幾何分布にしたがう変数をｒ個足し合わせた分布を考える。これはｒ次のパスカル分布（または負の２項分布）と呼ばれる。

プログラム

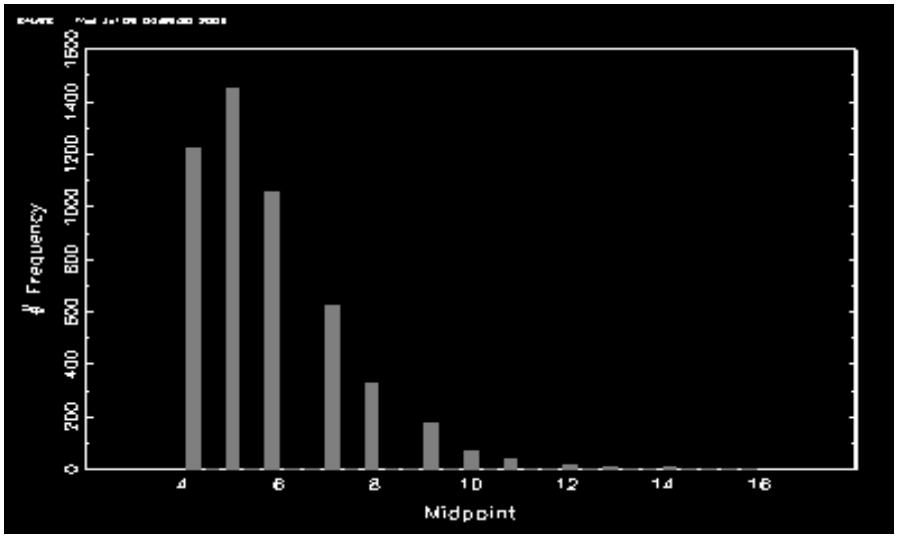
```
new; cls;
times=5000;
p=0.7;
r=4;
call rndpascal(times,r,p);

proc rndpascal(times,r,p);
    local Y,X,i,j,n;
    Y=zeros(times,1);
    j=1;
    do while j<=times;
        X=zeros(r,1);
        i=1;
        do while i<=r;
            n=1;
            do while n<=10000;        /* cutoff value */
                if rndu(1,1)<p;
                    X[i]=n;             /* first success after n bernoulli trials */
                    break;
                endif;
                n=n+1;
            endo;
            i=i+1;
        endo;
        Y[j]=sumc(X);
        j=j+1;
    endo;
/* Display statistics */
    print "Theoretical E[X]=r/p=" r/p;
    print "Empirical     E[X]=       " meanc(Y);
    print "Theoretical Var[X]=r*(1-p)/p^2=" r*(1-p)/p^2;
    print "Empirical     Var[X]=                  " vcx(Y);
/* Histogram of Y */
```

```
    library pgraph;
    graphset;
    call hist(Y,29);
    retp(Y);
endp;
```

画面表示

```
Theoretical E[X]=r/p=          5.7142857
Empirical   E[X]=              5.7108000
Theoretical Var[X]=r*(1-p)/p^2=          2.4489796
Empirical   Var[X]=                      2.5472728
```

グラフ表示



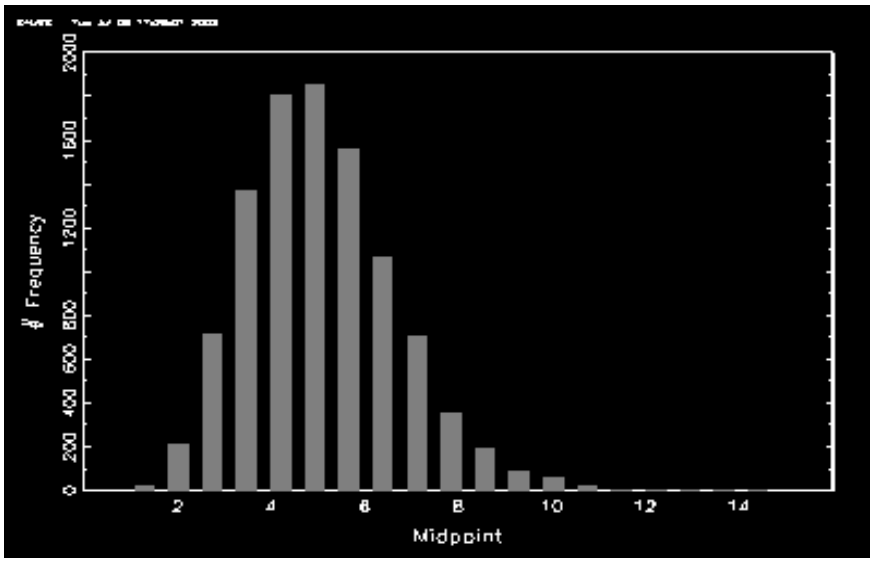
指数分布と Erlang 分布

ここでは離散分布ではなくて、連続分布のケースとして指数分布 exponential を足し合わせる分布である Erlang 分布を乱数生成してみる。ここでは、それぞれのパラメータλに対する Exponential の乱数生成は前章で行なったように逆関数法で生成する。それをm個だけ足し合わせたものを考える。

プログラム

```
new; cls;
r=10000;
lambda=2;
m=10;
n=10000;
x=rnderlang(r,lambda,m);

proc rnderlang(r,lambda,m);
    local U,X,Z,j,i;
    X=zeros(r,1);
    j=1;
    do while j<=r;
        U=rndu(m,1);
        Z=zeros(m,1);
        i=1;
        do while i<=m;
            Z[i]=-1/lambda*ln(U[i]);     /* Zi=-1/lambda*ln(Ui) */
            i=i+1;
        endo;
        X[j]=sumc(Z);                    /* sum of Z's */
        j=j+1;
    endo;
/* Histogram of X */
    library pgraph;
    graphset;
    call hist(X,19);
    retp(X);
endp;
```

グラフ表示



なお、プログラム中、共通のλの逆数がかかっている ln(Ui)を足し合わせるところは、ログの計算のきまりから ln(U1U2U3 … Um)であるから、その部分を書きなおすこともできる。

Convolution **とは何か**

　これまで、いくつかの例を見てきた。そこでは対象となる分布のランダム変数Ｙは、いくつかのランダム変数の和の形になっているものであった。すなわち、アルゴリズムは

基本的なアルゴリズム

１）ある分布にしたがう独立なランダム変数Ｘ1,Ｘ2,...，Ｘn を発生させる。

２）Ｙ＝Ｘ1＋Ｘ2＋ … ＋Ｘn なるランダム変数を作成する。

３）１から２の作業を繰り返して、それぞれのＹを並べたものを新しい分布の乱数とする。

というものでった。なお、元の分布にしたがうランダム変数Ｘ1,Ｘ2, … , Ｘn は、例えば逆関数などの方法や組込み関数から生成するものとする。何であってもよい。要は、これらをｎ個足し合わせるということである。この和をとる作業は、数学的には次のように考えられている。すなわち、単純化のためＹ＝Ｘ＋Ｘa という２つのランダム変数の和を考える。ただし、ＸとＸa は、同じ分布にしたがうが、互いに独立なランダム変数とする。

離散ランダム変数のケース

$$f*g(Y)=\Sigma_x f(X)\,g(Y-X) \qquad \{X\in R: X\in S \text{かつ } Y-X\in T\}$$

であるときｆ＊ｇは、ｆとｇの discrete convolution と呼ばれる。

ただし、上のＳとＴは実数Ｒのサブセットとする。

連続ランダム変数のケース

$$f*g(Y)=\int f(X)\,g(Y-X)\,dX$$

であるときｆ＊ｇは、ｆとｇの continuous convolution と呼ばれる。

この独立なランダム変数の足し合わせという作業は、数学的には density の重畳（または、くりこみ）積分という表現に相当している。そういうわけで convolution と呼ばれている。この convolution という方法Ｙ＝Ｘ1＋Ｘ2＋ … ＋Ｘn は一般的なＹ＝ｈ(Ｘ1,Ｘ2, …,Ｘn)という形の関数ｈの部分がＸ1＋Ｘ2＋ … ＋Ｘn というふうに１次の線形になった形であるとも解釈できる。このことは、自由度ｎのχ^2分布にしたがう乱数が定義にしたがえば標準正規乱数を２乗したもののｎ個の和によってできあがることからもわかるであろう。

ただし、このアルゴリズムは定義にしたがって直接的であってよいのだが、幾何分布からパスカル分布を作成するように、幾何分布がさらにベルヌーイ試行から作成されているとなるとプログラム的には３重ループが必要となり、そのことは、１つのランダム変数を作り出すのに、それぞれについて、何個か（場合によっては途方もない数の）ランダム変数を必要としており、場合によってはさらにその前の段階があることになって、大変な計算量を必要とする。そういう意味で非効率的である。時間がかかる。しかしながら、この方法も、計算機が十分に発達した今日では、基礎的な分布の作成法をループをまわして作るという観点からは教育的に有意義であるばかりか、ランダム変数とは何かを実感できる方法と言えるであろう。

以上、４章をかけて次のような一様乱数から様々な乱数を作り出す方法を取り扱った。

１．逆関数法（inverse transform）
２．採択棄却法（acceptance rejection）
３．Composition 法
４．Convolution 法