## 5.26 モンテカルロシミュレーション 分散減少法(4)　ver.0.1

### モーメント照合法 Moment Matching

ここでいうモーメントとは、平均や標準偏差、それに３次なら skewness、４次なら kurtosis を指します。作成された乱数や準乱数のそれらの値を一致させるのが、このモーメント照合法です。主に対象となるのは、標準正規分布にしたがう乱数です。一次元の系列の場合は、変数の標準化の作業と全く同じことを行ないます。標準正規分布にしたがう乱数に標準偏差をかけて平均を足し合わせれば、目的とする正規分布にしたがう乱数が作成される逆のことを行ないます。また、多次元（多変量）の場合には、Cholesky 行列をかけ合せて複数の標準正規分布にしたがう乱数から特定の分散共分散をもつ多次元乱数を作り出しましたが、そのちょうど逆のことをここでは行ないます。

３次４次のモーメントも照合する方法はありますが、ここでは平均と標準偏差（または分散）を一致させる方法のプログラムを見ていきます。

プログラム

```
new; cls;
n=10000;

X=rndn(n,1);
print "mean=" meanc(X);
print "s.d.=" stdc(X);

X=(X-meanc(X))/stdc(X);
print "after moment-matching:";
print "mean=" meanc(X);
print "s.d.=" stdc(X);
```

画面表示

```
mean=      0.0054755492
s.d.=         1.0023487
after moment-matching:
mean=   1.7030821e-017
s.d.=         1.0000000
```

通常の組込み関数の標準正規分布にしたがう乱数ではどんなにｎを大きくしようと、平均０標準偏差１にはなりません。しかしながら、いわゆる「標準化」の作業をこのすでに標

準正規分布を意図して作成された分布に施せば、計算上の若干の桁違いを除けば、完全に平均０標準偏差１になります。すなわち、Ｘという乱数列に対して、

$$\tilde{X} = \frac{X - \overline{X}}{\sigma_X}$$

といった全体からサンプルの平均を引いたものをさらにサンプルの標準偏差でわってやれば、「標準正規分布の標準化」がなされて、この分布を１次および２次のモーメントのみで判断をすれば完璧になります。

この作業は、既成の標準正規乱数のみならず、すでに何かの方法で分散減少がはかられた標準正規乱数や後の章で述べる準乱数にも簡単に適用できます。分散減少法の１つと言うよりは、むしろ「微調整」と言った方が適切であるかもしれません。

以下では、標準正規乱数段階でこうした Moment Match の「微調整」を施した上で、それを用いて、幾何ブラウン運動をシミュレーションしてみよう。

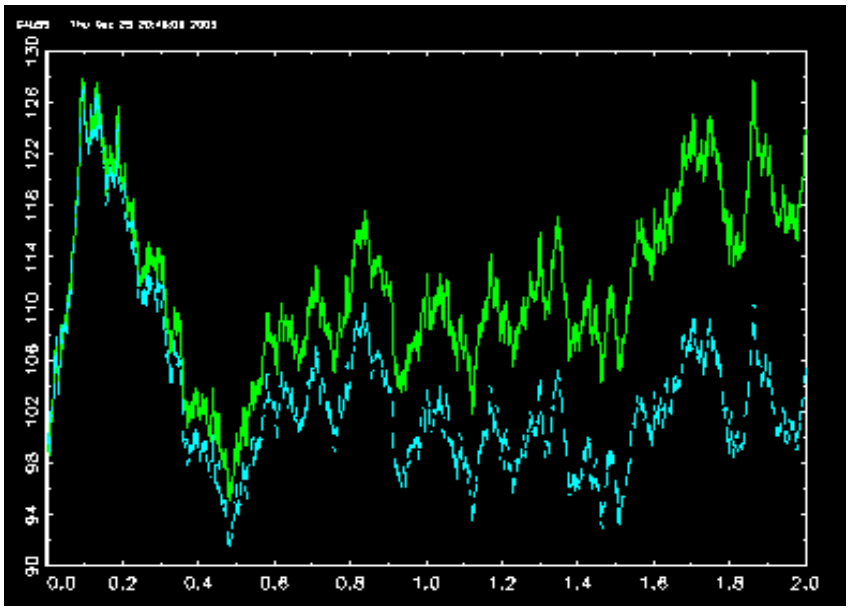
プログラム

```
new; cls;
n=2000;
S0=100;
r=0.05;
sig=0.20;
T=2;
call gbm(n,S0,r,sig,T);

proc gbm(n,S0,r,sig,T);
    local delt,e,e1,S,S1,i;
    delt=T/n;
    e=rndn(n,1);
    e1=(e-meanc(e))./stdc(e);                          /* after moment-match */
    S=zeros(n+1,1);
    S1=zeros(n+1,1);
    S[1]=S0;
    S1[1]=S0;
    i=1;
    do while i<=n;
        S[i+1]=exp((r-sig^2/2)*delt+sig*sqrt(delt)*e[i])*S[i];
        S1[i+1]=exp((r-sig^2/2)*delt+sig*sqrt(delt)*e1[i])*S1[i]; /* after moment-match */
        i=i+1;
    endo;
```

```
/* graph */
    library pgraph;
    graphset;
    xy(seqa(0,delt,n+1),S~S1);
    retp(S~S1);
endp;
```

グラフ表示




緑のパスは、もとになる標準正規乱数に GAUSS の組込み関数をそのまま使ったもの。水色のパスは同じ標準正規乱数をもとにしているのだが、Moment Match を施した標準正規乱数を用いた過程である。どちらが上にくるとか下にくるとかということはないが、総じて両者は平行して動く。微調整なのであるが、それが積み重なる単一の幾何ブラウン運動にしたがうパスは最終的にはかなり食い違ってくる。

　これをさらに応用したヨーロピアンコール解のシミュレーションによる導出にこの Moment Match の「微調整」を標準正規乱数段階で同様にして加えてプログラムしてみよう。なお、こちらのシミュレーションは最終T期のみのシミュレーションから解を導出することにする。

プログラム

```
new; cls;
S0=100;
K=100;
r=0.05;
sig=0.45;
T=1;
```

```
times=5000;
simn=1000;
C=zeros(simn,1);
Cmm=zeros(simn,1);
i=1;
do while i<=simn;
    C[i]=Ceuro(S0,K,r,sig,T,times);
    Cmm[i]=mmCeuro(S0,K,r,sig,T,times);
    i=i+1;
endo;
print "var(C)    =" vcx(C);
print "var(Cmm)=" vcx(Cmm);

proc Ceuro(S0,K,r,sig,T,times);
    local e,ST,CT,C;
    e=rndn(times,1);
    ST=exp(ln(S0)+(r-sig^2/2)*T+sig*e*sqrt(T));
    CT=maxc(((ST-K)~zeros(times,1))');
    C=CT*exp(-r*T);
    retp(meanc(C));
endp;

proc mmCeuro(S0,K,r,sig,T,times);
    local e,ST,CT,C;
    e=rndn(times,1);
    e=(e-meanc(e))./stdc(e);
    ST=exp(ln(S0)+(r-sig^2/2)*T+sig*e*sqrt(T));
    CT=maxc(((ST-K)~zeros(times,1))');
    C=CT*exp(-r*T);
    retp(meanc(C));
endp;
```

画面表示

```
var(C)    =          0.27115590
var(Cmm)=         0.010705629
```

mmCeuro(S0,K,r,sig,T,times)は単独で Moment Match を施したヨーロピアンコールの解

を得る procedure となっている。通常の解を得る Ceuro(S0,K,r,sig,T,times)とともに、同時シミュレーションを simn 回まわして、それらのコール解の分散を見ている。同一の乱数でシミュレーションを行なってはいないが、分散の減少の効果が明らかにあらわれている。

もっとも、Moment Match の「微調整」は平均と標準偏差を同時に両方行なうのであれば、乱数は標準正規分布にしたがうか、もしくは少なくとも正規分布である必要があるのであるが、平均だけの Moment Match を行なうのであれば、その分布の平均が厳密にわかっていれば、サンプル平均をまず差し引いておいてから、さらに真の平均値を足すことによって、乱数は平行移動する。すなわち、分布の真の平均値をμとすれば、乱数全体からサンプル平均を引いておいて、その平均をゼロにしておいてから、真の平均値μを足し合わせるのである。

$$\tilde{X} = X - \overline{X} + \mu$$

プログラム（ロジスティック分布（α，β）のケース、真の平均値α）

```
new; cls;
n=10000;
alpha=2;
beta=3;

X=rndu(n,1);
X=alpha+beta*ln(X./(1-X));
print meanc(X);
print "aftre moment-match:";
X=X-meanc(X)+alpha;
print meanc(X);
```

画面表示

```
        2.0535797
aftre moment-match:
        2.0000000
```

ただし、この平均値だけを一致させる方法を使っていけないケースも数多くある。ガンマ分布やベータ分布それに一様分布の場合のように、下限が決まっている分布や下限だけでなく上限も決まっている分布では、この調整によりその範囲からはみ出した値が出てくる可能性がある。したがって、この平均値の一致を行なってはいけない。また、整数値しか取り得ない離散分布も同様である。この調整によって、すべての値が整数たり得なくなるためである。したがって、こうした Moment Match のできるのは取りうる範囲が－∞から∞までの全域にわたって連続分布にしたがう乱数の場合のみである。注意したい。

**多変量標準正規乱数の場合**

多変量のケースも考え方は基本的に全く同様である。Multivariate な乱数を作成する際に、Cholesky 分解した分散共分散行列を後ろからかけた上で、平均の行ベクトルを足し合わせると、意図する分散共分散でかつそれぞれがそういう平均をとる Multivariate Normal にしたがう乱数が出来あがることはすでに Multivariate Normal の章で見た。その手順を全く逆転させて、一次元のケースと同じようにして、乱数の「さらなる標準化」をしてみよう。すなわち、ｎ×ｋのｋ次元（系列）の乱数があるものとして、それから行ベクトル（横ベクトル）である平均値をあらかじめ全体から引いておいて、それに元のｎ×ｋの乱数の分散共分散行列の Cholesky 分解の逆行列をかけてやればよい。ちょうど、一次元のケースで、平均を引いておいて、さらに標準偏差で割るのと同じである。以前お話したように、Cholesky 分解は行列の平方根のような演算であって、分散共分散行列の Cholesky 分解したものの逆行列をかけるというのは、いわば分散共分散行列の平方根で割るようなことに相当している。すなわち、ｎ×ｋのｋ系列の乱数データに対して、

$$\tilde{X} = (X - \bar{X})Chol(VCOV)^{-1}$$

という調整を行なう。GAUSS では、ｎ×ｋの行列から１×ｋの行ベクトルを引くと、各列どうしの計算が自動的に行なわれる。分散共分散行列 VCOV は組込み関数 vcx( )によって求められるので、それを inv 命令で逆行列としたものをかければよい。

プログラム

```
new; cls;
n=3000;
k=3;

X=rndn(n,k);
print meanc(X);
print vcx(X);
print;
X=(X-meanc(X)')*inv(chol(vcx(X)));
print "after moment-match:";
print meanc(X);
print vcx(X);
```

画面表示

```
      0.021670601
      0.010151406
     -0.030787529
```

```
      0.97706228       -0.016086383       -0.043093135
     -0.016086383       0.99203713        0.0049133773
     -0.043093135      0.0049133773        0.99277861
```

after moment-match:

```
 -1.1102230e-019
 -7.4755017e-017
  1.4573528e-016

         1.0000000   -4.2147013e-017   -2.5560303e-016
 -4.2147013e-017          1.0000000   -8.6626271e-018
 -2.5560303e-016   -8.6626271e-018          1.0000000
```

一次元の乱数の時と同様に、Moment Match が行なわれた結果の Multivariate Standard Normal な３次元（３系列）の乱数の分散共分散行列の対角成分はすべて１となっている。当然のことながら、非対角成分はすべてほぼゼロになるはずである。実際、10 のマイナス 16 乗より小さい値になっており、元の乱数よりも格段にゼロに近づいている。また、平均値も同じように、計算精度による桁の問題を除けば、ゼロにセットされていることがわかる。なお、さらにここからある特定の分散共分散と平均値をもつ多変量の系列にするのには、Multivariate Normal の章で行なった操作を施してやればよい。すなわち、もう一回意図する分散共分散行列の Cholesky 分解を後ろからかけてやって、意図する真の平均値を各列ごと足し合わせてやればよい。