情報科学概論最終レポート

060302208　　樋掛雅則

1-(1)-(a)

ある明るさと、それより明るい星の個数との関係から、夜空の明るさをシミュレーションすることを考える。いま、その関係は図のようになっているとする。

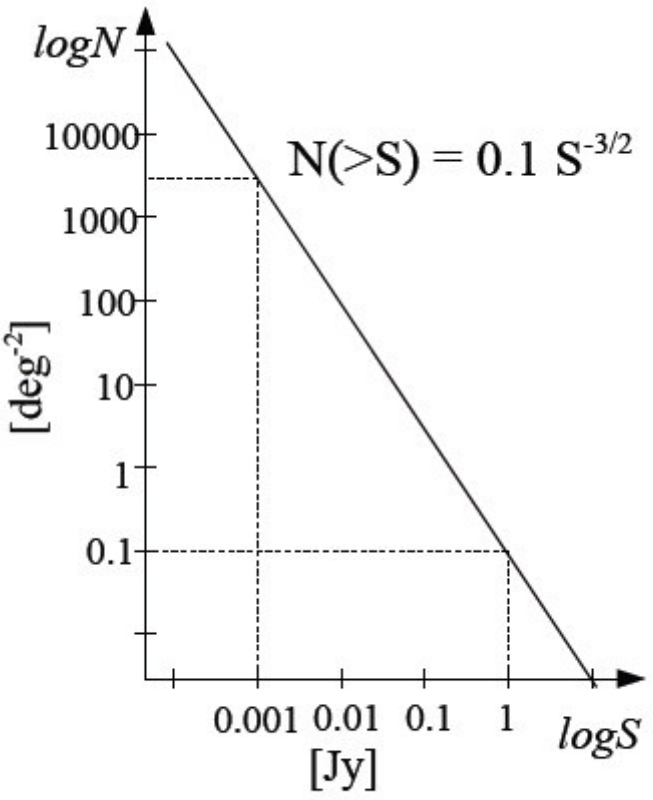


図1：logN−logS 関係

このグラフはいってみれば“積分”された状態にあるので、あるステップごとに値を読み取り、差をとる、という“微分”のような作業をする必要がある。

そうしてわかったある明るさの範囲にある個数にその明るさをかけて、指定した範囲で足しあげるようなプログラムを作る。また、上の関係式は“平均値”であるので、実際に値を作成するときは、「ある明るさの範囲にある個数」を平均値とするポアソン分布に従う乱数を用いる。(わかりにくい説明ではあるが…)。

このことを考えて、次のような基本 source を作成した。

```
/*********************************************************************
 *   Sample plogram to generate random value according to the
 *        Poisson distribution
 *********************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define PI 3.141592654 // pai
#define rand2() ((double)rand()/(RAND_MAX+1.0))

/*********************************************************************
 * ガンマ関数（自然対数）: Numerical Recipes in C
 * ln Γ(x)の値を計算する。xx>0 のとき ln Γ(xx)を返す。
 *********************************************************************/
float gammln(float xx)
{
   double x, y, tmp, ser;
   static double cof[6] = {76.18009172947146, -86.50532032941677,
                           24.01409824083091, -1.231739572450155,
                           0.1208650973866179e-2, -0.5395239384953e-5};
   int     j;

   y = x = xx;
   tmp = x + 5.5;
   tmp -= (x + 0.5) * log(tmp);
   ser = 1.000000000190015;
   for (j = 0; j <= 5; j++) ser += cof[j] / ++y;
   return(-tmp + log(2.5066282746310005 * ser / x));
}
```

```
/*************************************************************
 * Poisson 乱数：Numerical Recipes in C
 * 平均 xm の Poisson 分布の乱数（整数値）を浮動小数点で返す。一様乱数
 * 源として rand2()[0~1]を使用。
 *************************************************************/
float poidev(float xm)
{
   static float sq, alxm, g;
   static float oldm = (-1.0);
   float em, t, y;

   if (xm < 12.0) {
      if (xm != oldm) {
         oldm = xm;
         g = exp(-xm);
      }
      em = -1;
      t = 1.0;
      do {
         ++em;
         t *= rand2();
      } while (t > g);
   } else {
      if (xm != oldm) {
         oldm = xm;
         sq = sqrt(2.0 * xm);
         alxm = log(xm);
         g = xm * alxm - gammln(xm + 1.0);
      }
      do {
         do {
            y = tan(PI * rand2());
            em = sq * y + xm;
         } while (em < 0.0);
         em = floor(em);
         t = 0.9 * (1.0 + y * y) * exp(em * alxm - gammln(em + 1.0) - g);
```

```
    } while (rand2() > t);
  }
  return(em);
}

/*************************************************************************/

/*************************************************************************
 *logN-logS 関係
*************************************************************************/
double nsl(double s)
{
        double a=-1.0,r=-1.5;


        return a+r*s;
}
/*************************************************************************
*"微分"
*************************************************************************/
double ns(double sj,double ds)
{
    double x1,x2;

    x1=pow(10.0,nsl(sj));
    x2=pow(10.0,nsl(sj+ds));
    return x1-x2;
}

/*************************************************************************
*MAIN
*************************************************************************/
int main(void)
{
  int j,k;
  double ds=0.1 ,x3,kido,sj;
```

```
    printf("Plese input Smax= ");
    scanf("%lf",&smax);
    printf("Plese input Smin= ");
    scanf("%lf",&smin);
    printf("Plese input Omega= ");
    scanf("%lf",&ome);
  k=(smax-smin)/ds;
     kido=0.0;
      for(j=0;j<k+1;j++)
          {
           sj=smin+j*ds;
           kido+=pow(10.0,sj)*(double)poidev((ome*ns(sj,ds)));
          }
       x3=kido/ome;
       printf("%f¥n",x3);

 return 0;
}
```

（一応、ばらまく星の明るさの最大値 Smax,最小 Smin,領域 Omega を変更可能にしてあるが、申し訳程度なので、プログラムに書いてしまった方が便利と思われる。）
この source で、星をばら撒く広さ 1 平方度、ばら撒く星の明るさの最大値 1、最小値 0.001 の場合の平均表面輝度を求めてみる。実行結果は次のようになる。

	1 平方度
平均表面輝度	8.287427

以降、この source をベースに若干の変更を加えることで、課題に取り組む。
なお、余談ではあるが、このレポートの作成に関わるすべての source の実行に於いて、MicrosoftVisualC++6.0 を使用した。これは、休暇中はなかなか学校の計算機を使用できない、というやむにやまれぬ事情によるものである。

1-(1)-(b)

先に示した source を用い、ばら撒く天体の明るさの最大値を 1[Jy](logS_{max}=0)最小値を 0.001[Jy](logS_{min}=3)とし（変更しないで）、領域の広さ 1 平方度、10 平方度それぞれについて平均表面輝度を求める。結果は次のようになる。

	1 平方度	10 平方度
平均表面輝度	8.287427	8.514892

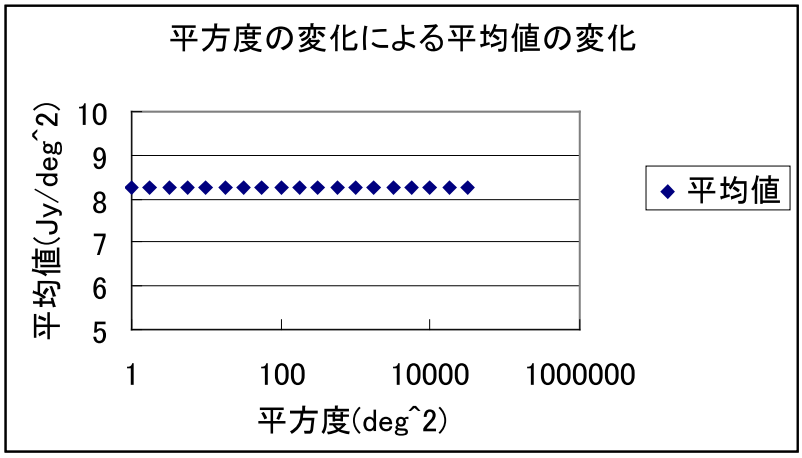
1-(1)-(c)

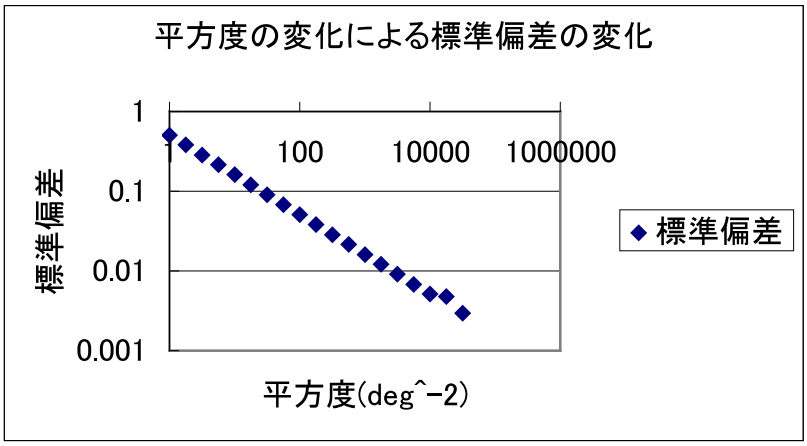
次に、1 平方度についてこの計算を 10000 回行い、平均表面輝度の平均値と、標準偏差を求める。ばら撒く天体の明るさの設定に 1-(1)-(b)のものを用いると、以下の結果が得られた。

	平均値(Jy/deg^-2)	標準偏差
1 平方度	8.254343	0.504902

1-(1)-(d)

次に、この平均値・標準偏差が領域の広さを変えることによってどのように変化するかを調べる。全天球が 41252.96deg^2 であることに配慮し、平方度の範囲としては対数で 0 から 4.5 まで、0.25 刻みで測定することにする。結果を以下に示す。

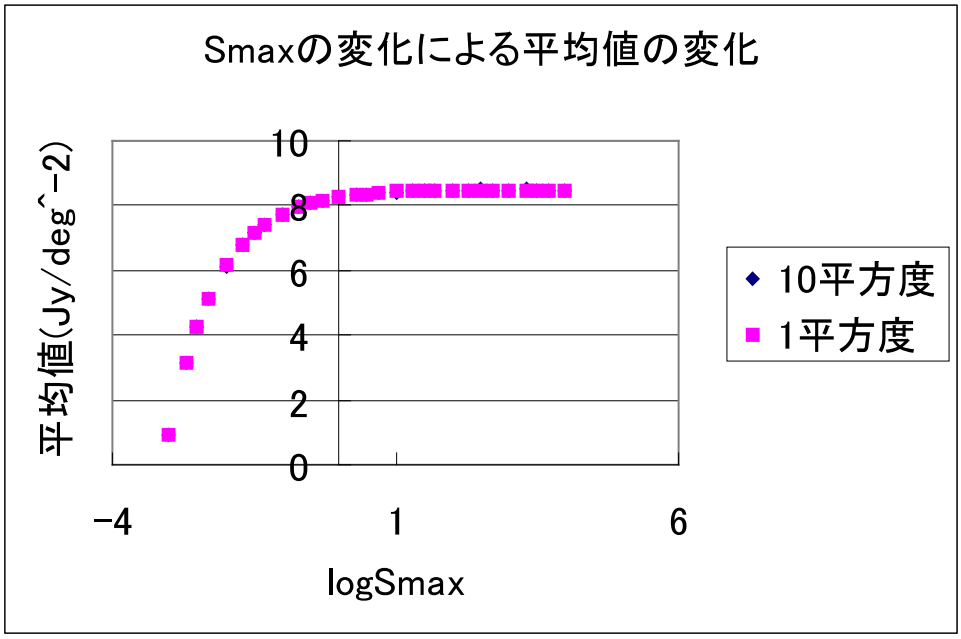


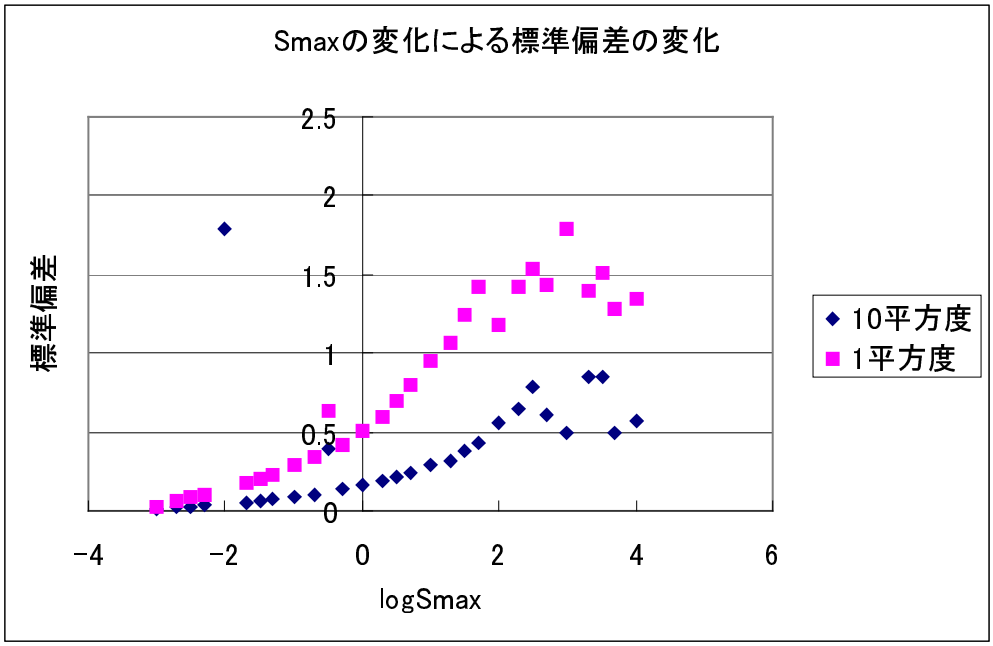


このとき、平方度を変化させても平均値は 8.25Jy/deg2 でほぼ一定である。
それに対し、標準偏差は平方度が大きくなるにつれて減少することがわかる。
狭い領域の測定の場合、数の少ない明るい星が、領域に入ってくるかどうかで大きく表面輝度が変化するのに対し、領域が広ければある程度コンスタントに明るい星が入るようになるため、揺らぎが減ると考えられる。
あるいは、たとえば一平方度の場合は、ばら撒いてある程度揺らいでいるものを 1 で割るのに対し、10 平方度の場合はそれを 10 回異なる領域に対して行い、10 で割っている、と考えれば、揺らぎは減少することが妥当と考えることができる。

1-(1)-(e)

今度はばら撒く星の明るさを変化させる。まずは、最小値を 0.001 に固定し、最大値を 0.001 から 10000 まで変化させて 1-(c)と同様の処理を行い、平均値と標準偏差を求める。なお、領域の広さは 1 平方度に固定する。
結果として以下のグラフが得られた。





平均値は、明るさの最大値が 0.001-10Jy までは徐々に増加するが、1Jy を越えたあたりからはほぼ横ばいで推移した。標準偏差は最大値 100Jy までは対数関数的な増加を示すが、それ以降は振動しながら、ほぼ横ばいで推移している様子がわかる。

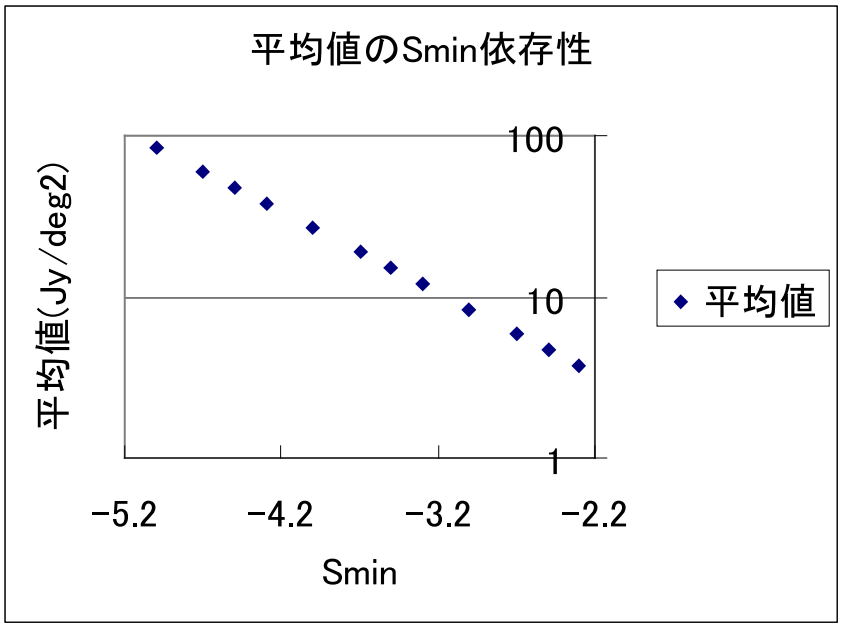
これは、星の明るさの最大値がある程度暗い段階では、その明るさの領域にある個数が十分多く、今回の“平均値が減る”という形で計算結果に大きく寄与するが、ある程度明るくなると、存在確率が減るため、あまり寄与しなくなってくる、ということに起因してくるのではないかと思われる。

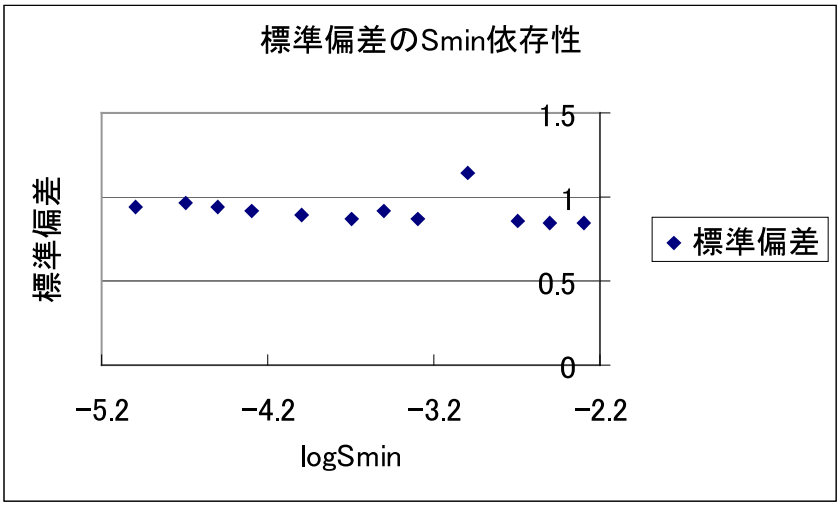
標準偏差に関しては、個人的には最大値が 100Jy を越えても、本来は指数関数的に増加するように思えてならず、「おそらく 10000 回では明るい星が入る場合と入らない場合が出てきている」と考え、試行回数を 50000 回にして実行してみたが、大きな変化は見られず、100000 回としてみたものの実行に膨大な時間を要したため、途中で断念した。そのため、このことに関しては未だに自分の中では解決を見ていない。

1-(1)-(f)

次に、最小値を変化させる。最大値を 1Jy に固定し、最小値を 0.0001 から 1 まで変化させて 1-(1)-(c)と同様の処理を行い、平均値と標準偏差を求める。なおこのときも、領域の広さは 1 平方度に固定する。

結果として以下のグラフが得られた。





平均値は、最小値を増加させることで対数関数的に減少する。暗い星はその絶対量が多いため、実行結果への寄与が大きい。
標準偏差は、最小値を増加させることで若干現象する傾向が見られるが、ほぼ横ばいで推移している。

1-(2)-(a)

次に、logN-logS 関係を次のように変更して、1-(1)-(c)のような処理を行う。

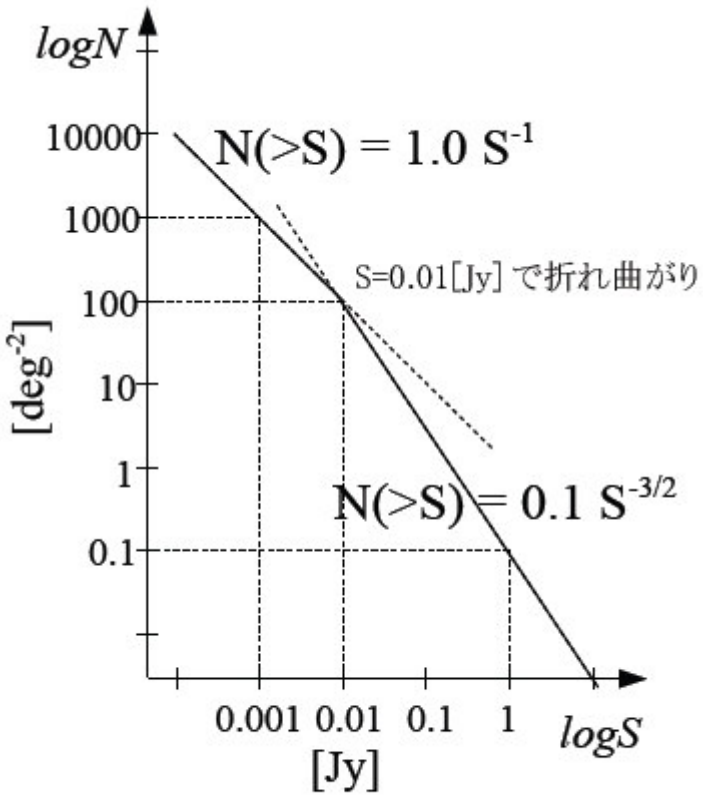


図2：logN-logS 関係（2）

実行結果は、次のようになった。

	平均値	標準偏差
1 平方度	4.512194	0.504421

標準偏差はあまり変化していないが、平均値が大幅に減少している。「星の数」自体が減るため、もっともな結果であるといえる。

1-(2)-(b)

logN-logS 関係(2)を用い、関係(1)のような平均値を得るには、ばら撒く星の最小値をいくつにすればよいかを考える。先ほどの source に、(1)-(e)のような処理を加えることで、次

のデータが得られた。

logSmin	Smin	平均値	標準偏差
-4.9	1.25893E-05	8.567322	0.901434
-4.85	1.41254E-05	8.461107	0.879667
-4.8	1.58489E-05	8.372164	0.925207
-4.75	1.77828E-05	8.252304	0.860865
-4.7	1.99526E-05	8.155148	0.896377
-4.65	2.23872E-05	8.063516	0.969627
-4.6	2.51189E-05	7.943889	0.866686

LogSmin を-4.8 程度に設定することで、ほぼ 1-(c)と同じ平均値が得られることがわかる。

1-(2)-(d)

さらに、logN-logS 関係を次のようにして、1-2-(b)と同様の計算を行う。

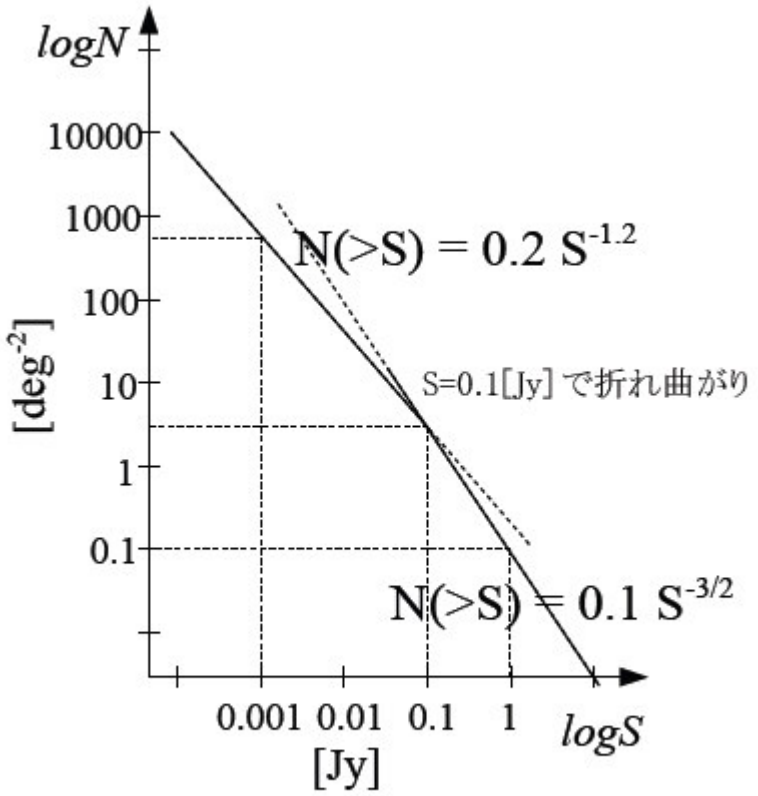


図3：logN-logS 関係（3）

結果は、次のようになる。

logSmin	Smin(Jy)	平均値	標準偏差
-4.75	1.77828E-05	8.634534	0.887381
-4.7	1.99526E-05	8.411429	0.889577
-4.65	2.23872E-05	8.201771	0.911691
-4.6	2.51189E-05	7.99597	0.899999
-4.55	2.81838E-05	7.766759	0.820994
-4.5	3.16228E-05	7.601607	0.900862
-4.45	3.54813E-05	7.400256	0.884214

この場合は LogSmin を-4.7 程度に設定することで、1-(c)とほぼ同じ平均値が得られることがわかる。
このとき、関係(2)、関係(3)について、標準偏差はほとんど同じ値(0.9)となっているが、関係(1)の値(0.5)と比較すると、異なる値と言わざるを得ない。
今回のような条件下では、標準偏差はばら撒く領域によるところが大きく、関係の違いで大きく変動することはない、と予想していた。そのため、この結果には正直驚いた。
友人に聞いてみたところ、演習でやったように Linux でやってみると、あまり差は出てこない、という。
そんな具合で、この点に関しても、未だ自分の中での解決を見ていない。