三重大学教育学部研究紀要　第50巻　自然科学（1999）　69－77頁

桁落ち誤差領域における Horner 法の乱雑性

谷　口　礼　偉

Randomness of Horner's Rule in the Domain of Cancellation Error

Hirotake YAGUCHI

ABSTRACT

We report that we can obtain uniform random numbers easily by making use of the randomness of Horner's rule in the domain of the cancellation error under single-precision calculations. The chi-square test and the Kolmogorov-Smirnov test reveal that the properties of our random numbers are of the same level as those of random numbers generated by the function *random* attached to the software Delphi 2.0.

数値計算の桁落ち誤差領域における、Horner 法の乱雑性を利用することにより、複雑な議論を経ることなくかなり良質な一様乱数が簡便に得られることを報告する。よく知られている乱数系列以外に、新たな乱数系列が必要な場合に役立つであろう。

桁落ち誤差はコンピュータにおける数値計算において、注意をもって避けなければならない計算誤差の一つである。図1は $\sin x$ を

$$\sin x \fallingdotseq x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \cdots + (-1)^{n-1} \frac{x^{2n-1}}{(2n-1)!} \tag{1}$$

と近似し、$n = 30$ の場合において、

$$F_1(x) = x,\ F_k(x) = -F_{k-1}(x) \cdot \frac{x^2}{(2k-2)\ (2k-1)},\quad k = 2,\ldots,30, \tag{2}$$

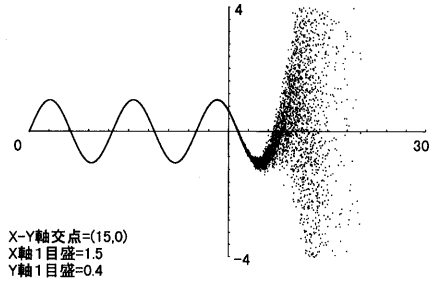
と単精度計算し、

$$\sin x \fallingdotseq F_1(x) + F_2(x) + F_3(x) + F_4(x) + F_5(x) + \cdots + F_{30}(x) \tag{3}$$

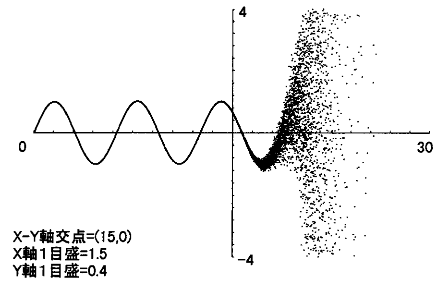
を求めた結果である。$x = 0$ から $x = 30$ までを 19999 等分し、各分点での計算値をグラフ化したものであるが、$x = 16.5$ 前後から桁落ち誤差によるグラフの乱れが顕著に観察される。単精度計算では有効桁数が 10 進で 7～8 桁[3]であるであることに留意して、例えば、$x = 22$ では、以下のような計算が行われている。各数値は、Delphi 2.0（Borland 社の Object Pascal 言語）の関

原稿受理日 1998年10月28日
三重大学教育学部数学教室
Department of Mathematics, Faculty of Education, Mie University
Kamihama 1515, 514-8507 Tsu, Mie, Japan

【図１】



【図２】



数 FloatToStrF（数値変数, ffExponent, 15, 3）による文字列を利用した。また、比較のため、倍精度計算（有効 15〜16 桁）による結果も同時に示した。

	単精度計算	倍精度計算
F_1	22.0000000000000	22.0000000000000
F_2	−1774.66662597656	−1774.66666666667
F_3	42946.9335937500	42946.9333333333
F_4	−494912.281250000	−494912.279365079
F_5	3326910.25000000	3326910.32239859
F_6	−14638405.0000000	−14638405.4185538
F_7	45416588.0000000	45416591.1703848
F_8	−104674424.000000	−104674429.173649
F_9	186258896.000000	186258910.735463
F_{10}	−263594464.000000	−263594481.859545
F_{11}	303761248.000000	303761260.047666
F_{12}	−290554240.000000	−290554248.741246
F_{13}	234380416.000000	234380427.317938
F_{14}	−161595616.000000	−161595622.253393
F_{15}	96320536.0000000	96320543.3135989
F_{16}	−50128108.0000000	−50128110.7137439
F_{17}	22975382.0000000	22975384.0771326
F_{18}	−9344609.00000000	−9344609.99439681
F_{19}	3395488.50000000	3395488.91688292
F_{20}	−1108918.00000000	−1108918.10780792
F_{21}	327266.031250000	327266.075718922
F_{22}	−87705.8437500000	−87705.8586090578
F_{23}	21439.2070312500	21439.2098822141
F_{24}	−4799.52636718750	−4799.52709666588
F_{25}	987.657653808594	987.657786898931
F_{26}	−187.461288452148	−187.461321121209
F_{27}	32.9213600158691	32.9213640865984
F_{28}	−5.36496257781982	−5.36496303633455
F_{29}	0.813484311103821	0.813484370171028
F_{30}	−0.115057393908501	−0.115057403612735
$\sum_{i=1}^{30} F_i$	−16.1806468963623	−0.0223788063394826

ここで、小サイズ数字の部分は、単精度の値を $\cdot\cdot a_3\, a_2 . a_1\, a_0\, b_0\, b_1\, b_2\, b_3 \cdot\cdot$、これに対応する倍精度の値を $\cdot\cdot \tilde{a}_3\, \tilde{a}_2 . \tilde{a}_1\, \tilde{a}_0\, \tilde{b}_0\, \tilde{b}_1\, \tilde{b}_2\, \tilde{b}_3 \cdot\cdot$ としたとき、3 桁の数値 $a_0\, b_0\, b_1$ と $\tilde{a}_0\, \tilde{b}_0\, \tilde{b}_1$ の距離が 50 以下となり、$b_0\, b_1\, b_2$ と $\tilde{b}_0\, \tilde{b}_1\, \tilde{b}_2$ の距離が 50 以下とならない場合の、$b_0\, b_1\, b_2\, b_3 \cdot\cdot$, $\tilde{b}_0\, \tilde{b}_1\, \tilde{b}_2\, \tilde{b}_3 \cdot\cdot$ に対応する部分である。

Horner 法は、桁落ち誤差を避けるのに有効な計算法とされ、(1) を

$$x\left(1-\frac{x^2}{2\cdot 3}\left(\cdots\left(1-\frac{x^2}{(2n-4)\ (2n-3)}\left(1-\frac{x^2}{(2n-2)\ (2n-1)}\right)\right)\cdots\right)\right) \tag{4}$$

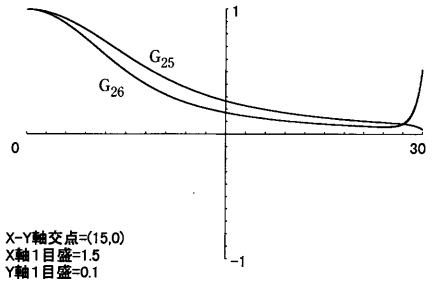
のように変形し、

$$G_0 = 1,\ G_k = 1-\frac{x^2}{(2n-2k)\ (2n-2k+1)} \times G_{k-1},\ k = 1, 2, \cdots, n-1, \tag{5}$$

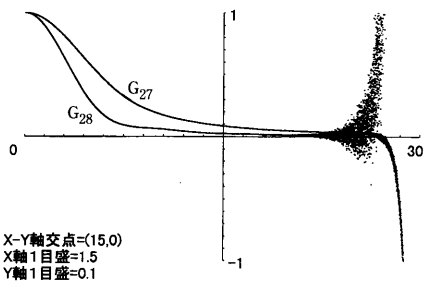
$$G_n = x \times G_{n-1}$$

と計算するものである。しかしながら前と同様な条件 ($n = 30$) でグラフを描いてみると、やはり突然グラフが乱れる現象が現れる（図 2）。計算途中の項 G_1〜G_{29} を見てみると、G_{27} あたりから顕著な乱れが観察される（図 3-1, 3-2）。

【図 3-1】

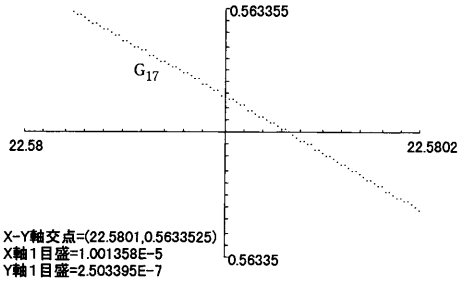


【図 3-2】

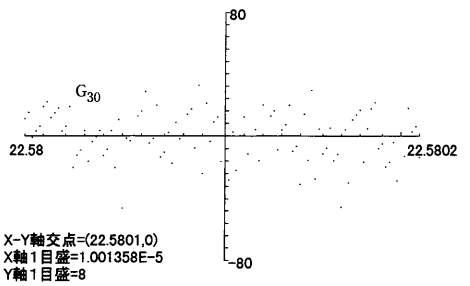


さらに、$x = 22.5$ あたりを拡大してみたものが図 4、図 5 である。

【図 4】



【図 5】



これらをみると、桁落ち領域にある x に対しては、最初のわずかな違いが繰り返し計算の終わり頃には次第に拡大し、最終的には乱雑な様相を呈することが観察される。前の計算法と比べ

るため、項数 $n=30, x=22$ で単精度計算を行い、G_0～G_{30} を記しておく（倍精度計算値も同時に記す）：

	単精度計算	倍精度計算
G_0	1.0	1.0
G_1	0.858562231063843	0.858562244301578
G_2	0.869817018508911	0.869817003057029
G_3	0.858252048492432	0.858252043946262
G_4	0.849276483058929	0.849276491556607
G_5	0.838804006576538	0.838803991406511
G_6	0.827388942241669	0.827388974557504
G_7	0.814775109291077	0.814775086176766
G_8	0.800832748413086	0.800832756712346
G_9	0.785380363464355	0.785380368633015
G_{10}	0.768217027187347	0.768217013159525
G_{11}	0.749111294746399	0.749111312841289
G_{12}	0.727800428867340	0.727800393832445
G_{13}	0.703987061977386	0.703987066710165
G_{14}	0.677339255809784	0.677339261091174
G_{15}	0.647492229938507	0.647492255518142
G_{16}	0.614056348800659	0.614056340306921
G_{17}	0.576633512973785	0.576633520358191
G_{18}	0.534848988056183	0.534848960244392
G_{19}	0.488405317068100	0.488405342374929
G_{20}	0.437171012163162	0.437170986406034
G_{21}	0.381313532590866	0.381313574793800
G_{22}	0.321486204862595	0.321486138969856
G_{23}	0.259050846099854	0.259050993993284
G_{24}	0.196278139948845	0.196277685302889
G_{25}	0.136376187205315	0.136378184667289
G_{26}	0.0832489654421806	0.0832355364032255
G_{27}	0.0406547784805298	0.0408095328771154
G_{28}	0.0161543600261211	0.0124093043738067
G_{29}	−0.303118377923965	−0.00101721948707606
G_{30}	−6.66860437393188	−0.0223788287156735

G_{26}, G_{27}, G_{28} あたりで（個々には甚だしいものではないが）桁落ち誤差がさみだれ的に発生していることに注目して欲しい。一応 $y=\sin 22=-6.6686$ と計算されているが、もちろんこの値は正しくない。本報告の目的は、このような x の領域における Horner 法計算値から乱数が生成されることを示し、検証することである。

乱 数 の 生 成

以下のようにして 0 以上 9999 以下の数値 20000 個を作成する。

1) $x_0=22$ と $x_{19999}=26$ の間を 19999 等分して、$x_i=x_0+\dfrac{x_{19999}-x_0}{19999}\times i, i=0,1,\cdots,19999$, を作り、
2) 各 x_i について、項数を 30 とする sin の近似式 (1)を、Horner 法(5)により単精度計算

し、数値 y_i を得る。

3) 計算結果 y_i を浮動小数点表示し、上位 3 桁の数字を捨て、残った桁から続けて 4 桁の数をとる。

y_0 ⇒ −6.66**8604**37393188E+000 ⇒ 8604

y_1 ⇒ −7.80**0577**16369629E+000 ⇒ 0577

y_2 ⇒ −7.64**5578**86123657E+000 ⇒ 5578

y_3 ⇒ 4.75**3845**21484375E+000 ⇒ 3845

得られる数値の最初の部分は次のようになる（付録「プログラムソース」参照）:

8604, 0577, 5578, 3845, 6527, 7537, 5051, 2681, 7372, 4805
8009, 6213, 4359, 2754, 1024, 7043, 3334, 9752, 0997, 3157
3356, 1578, 3988, 6373, 2061, 9528, 7992, 8025, 8380, 8188
5659, 9127, 9128, 7107, 7425, 5777, 7124, 6652, 1248, ・・・

一様乱数性の検証

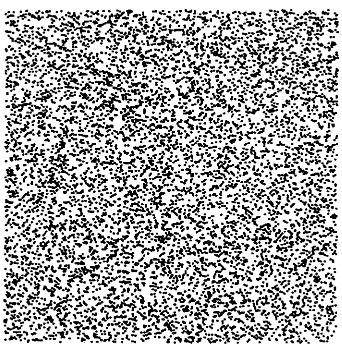
乱数性は多様な方法で検証されることが望ましいが、ここでは、基本的な、（Ⅰ）2 数の対による平面上の点の散布状態の図示ならびに Monte Calro 法による π の近似値の算出、（Ⅱ）数字 0〜9 の出現頻度の χ^2 検定、（Ⅲ）数字 0 の出現間隔の χ^2 検定、（Ⅳ）一様分布に対する Kolmogorov-Smirnov 検定、にとどめる。結果は必要に応じてコンピュータソフトウェア Delphi 2.0 付属の乱数関数 random による値と比較することにする。

（Ⅰ）発生した数値を 2 つずつ組み合わせ、XY 平面上の点

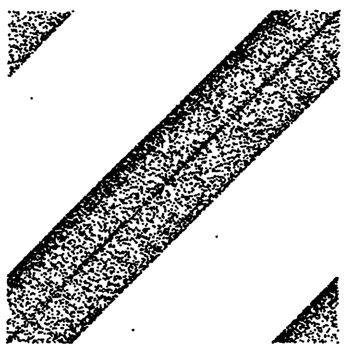
(0.8604, 0.0577), (0.5578, 0.3845), (0.6527, 0.7537), ・・・

を作り、プロットしたものが下の図 6 である。まんべんなく散らばっており一様乱数の性格を持っていることがうかがえる。同様のことを、$x=0$ と $x=4$ の間を 19999 等分して行うと図 7 になり、非桁落ち領域では一様乱数性が期待できないことがわかる。

【図 6】



【図 7】



上の左図の 10000 個の点のうち $x^2+y^2<1$ を満たすものの総数は 7852 個である。得られた数値データが、もし、一様に分布する乱数であれば、これらの点の総数の比 10000：7852 は、「一辺の長さ 1 の正方形の面積：半径 1 の四分円の面積」に近いと思われるから、「10000：7852 ≒ 1：$\pi/4$」となる。これより、$\pi \fallingdotseq 0.7852\times 4 = 3.1408$ と計算される。このとき、真値 $\pi = 3.1415926\ldots$ との相対誤差は $(3.1408-3.14159)/3.14159 = -2.51465\times 10^{-4}$ となる。同様の計算を、ソフトウェア付属の乱数関数 random(10000) で行うと（RandSeed = 1）、$x^2+y^2<1$ を満たすものの数は 7871 個であり、$\pi \fallingdotseq 3.1484$ となり、相対誤差は 2.16769×10^{-3} と計算される。

（Ⅱ）得られた数値データに含まれている文字 0～9 の個数は次の通りである（各数値に続く［ ］内の数値はソフトウェア付属の乱数関数によるもの；以下すべて同様）：

0	1	2	3	4	5	6	7	8	9
8022 [8074]	8021 [7927]	7897 [7929]	7992 [7932]	8067 [8047]	8107 [8061]	7953 [8085]	8016 [7970]	8022 [7939]	7903 [8036]

期待個数 8000 に対する標準偏差は 63.1617［62.8824］であり、その変動率は 63.1617/8000 = 0.007895［0.00786］となる。ちなみに、(3) の方法で同様に数値を発生させると、標準偏差は 68.7197、変動率は 0.00859 となって前二者とは明らかな差が見られるが、このデータだけでは有意差の判断はできない。数字 0～9 の出現頻度は一様（8000）であるという仮説の χ^2 検定を MS-EXCEL の関数 CHITEST で行うと、値 0.8355［0.8393］が得られ、0.8355 > 0.05 であることから、危険率 0.05 で仮説は棄却できない。

（Ⅲ）発生した 2 万個の数値には 8022［8074］個 の 0 があるが、最初に 0 が現れてから、以後順々に 0 が発生するまでの間隔の分布は、以下の通りである。

間　隔	0	1	2	3	4	5	6	7	8
回　数	787	745	655	600	506	471	400	386	369
理論回数	802.1	721.9	649.7	584.7	526.3	473.6	426.3	383.6	345.3

9	10	11	12	13	14	15	16	17	18
349	260	259	218	202	177	163	143	142	116
310.7	279.7	251.7	226.5	203.9	183.5	165.1	148.6	133.8	120.4

19	20	21	22	23	24	25	16	27	28
110	99	93	82	68	54	48	43	48	54
108.4	97.5	87.8	79.0	71.1	64.0	57.6	51.8	46.6	42.0

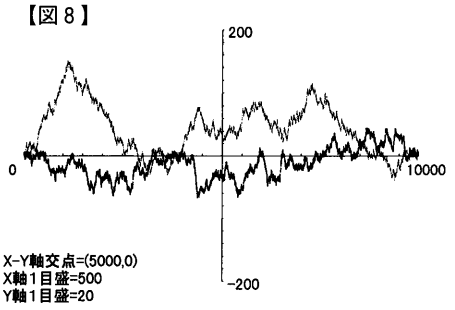
29	30	31	32	33	34	35	36	37	38以上
32	27	34	32	25	23	18	21	16	146
37.8	34.0	30.6	27.5	24.8	22.3	20.1	18.1	16.3	146.4

数字 0 が独立に発生すると仮定すれば出現間隔の理論分布は幾何分布であるので、$p = 0.1$ とおくと、0 が続けて発生する確率は p、間隔 k で発生する確率は $(1-p)^k\cdot p$、間隔 k 以上で発生

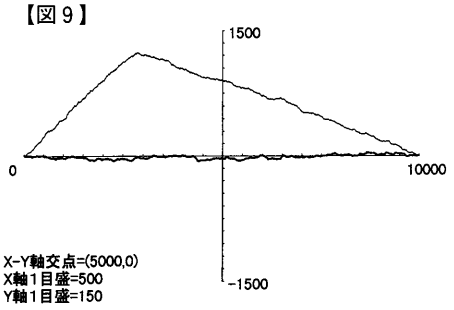
する確率は $(1-p)^k$ となる。これから、全部で 8022 [8074] 個ある 0 の間隔の理論出現回数を計算して（上表参照）χ^2 検定を MS-EXCEL で行えば（CHITEST)、値 0.9262 [0.8841] が得られ、0.9262 > 0.05 となり、危険率 0.05 で仮説は棄却できない。すなわち、0 が独立に発生しているという仮定は否定できない。

（Ⅳ）最後に、我々の数値を、区間 [0, 1] 上の一様分布にしたがう乱数とみなして、Kolmogorov-Smirnov 検定（KS 検定と略）[1, 2] を行う。[0, 1] 上の一様分布の分布関数 F は、$x \in [0, 1]$ では $F(x) = x$ で与えられる。発生した乱数値を順に $u_1, u_2, u_3, \ldots, u_n$, $n = 20000$, とおき、関数 F_n を次のように定める：$F_n(x) = n^{-1}\{i \mid u_i/10000 \leqq x\}$. KS 検定は、$K_n^+ = \sqrt{n}\max_x\{F_n(x) - F(x)\}$, $K_n^- = \sqrt{n}\max_x\{F(x) - F_n(x)\}$ を評価するものである。まず、$\#\{i \mid u_i \leqq k\} - 2(k+1)$, $k = 0, 1, \ldots, 9999$, を求めると（# = 集合の要素の個数)、以下図 8 の濃い線 [薄い線] のグラフになり、

【図 8】



【図 9】



その最大値は 43 [151]、最小値は −69 [−39] である。これから K_n^+, K_n^- を実際に計算すると、$K_n^+ = 43/\sqrt{20000} = 0.3041$ [1.0677], $K_n^- = 0.4879$ [0.2758] となる。この統計量の危険率 0.05 に対応する値の近似値は $\sqrt{-0.51\mathrm{n}0.05} = 1.2239$ であるので、分布が一様であるという仮説は棄却できない。

乱数の発生法として合同法がよく知られている。参考までに、C++Builder V.3 でその作成法が合同法であると明示されている [4]、関数 rand を使用した結果を図 9 に示しておく（srand の設定は 1、rand( )%10000 とした)。濃い線が我々の数値、薄い線が rand によるものである。また、$K_n^+ = 8.7752$, $K_n^- = 0.02121$ と計算された。（註：C++ Builder には、別途、乱数関数 random がある。）

以上の検定から、我々が発生した数値は一様乱数と呼んでよいであろう。評価の際現れた数値をみると、ソフトウェア付属の乱数関数 random と同等程度の性質を持つ乱数と考えられる。乱数発生法がかなり偶然的なものであるため、性質の解析は数学的手法に乗りにくいと思われる。しかしながら従来の乱数発生法とはかなり異なることから、種々の乱数系列が必要になった際に、もう 1 つの乱数系列として有用であろう。

付　　録

発生した 20000 個の乱数値を活字印刷で提供することは、ほぼ意味がないと思われるので、代りにプログラムソースの主要部を以下に示す。使用言語は Delphi 2.0 である。

まず、Horner 法による計算（5）は以下のように実現する：

```
function SIN_H(x:single):single;
  var
    i:integer;
    G:single;
  begin
    G:=1;
    for i:=29 downto 1 do
        G:=1-(G*x*x/((2*i)*(2*i+1)));
    SIN_H:=x*G;
  end;
```

$x=22$ と $x=26$ の間を19999等分して、上の関数SIN_Hによる計算結果を浮動小数点表示し、上位3桁の数字を捨て、残った桁から4桁の数をとるには以下のように行う（読者自身が最良と思われる方法を工夫されてもよい）。

```
procedure TForml. GenMyRan;
var
 x, y : single;
 XStart, XEnd, XStep : single;
 i, Kaisuu : longint;
 YStr : string[50];
 RanStr : string[20];
begin
 XStart:=22.0; XEnd:=26.0;
 Kaisuu:=19999;
 XStep:=(XEnd-XSart)/Kaisuu;
 for i:=0 to Kaisuu do
   begin
     x:=XStart+XStep*i;          {x座標}
     y:=abs(SIN_H(X));
     YStr:=FloatToStrF(y, ffExponent, 15, 3);
     if YStr[2]='.' then
         RanStr:=Copy(YStr, 5, 4)
       else
         RanStr:='Error';
     {必要に応じて、ここでRanStrを表示・保存する}
     if i<100 then              {とりあえず100個を画面に表示する}
         Canvas.TextOut((i mod 10)*48,(i div 10)*16,RanStr);
   end;
end;
```

留意点としては、使用する計算機および言語によって単精度計算の扱いが異なることである。このような場合、数学的に同じ条件であっても本報告とは違った乱数値が生成されることになる。また、最近の数値計算環境では、数値演算プロセッサとの関連で、単精度の数値を擬似的に倍精度扱いにして計算することが多々あるため、同じ計算環境であっても、コーディング方法の違いにより(途中の計算結果をいったん単精度の変数に保存するのかしないのか等)、有効桁数以下の数値が微妙に異なることがある。したがって、本報告内容の正確な再現を行う場合には、Delphi 2.0 (あるいは同等のもの)を使用し、上記のコーディングを用いることを希望する。

謝　　辞

岡山理科大学高嶋恵三氏との電子メールによる意見交換は、本論文発表の大きな励みになった。ここにお礼を申し上げたい。

参 考 文 献

1) Knuth, D. E. : The art of computer programming. Vol.2 (3rd ed.). Addison-Wesley, 1997.
2) 伏見正則：乱数．東京大学出版会，1989.
3) Borland 社（現 INPRISE 社）：Delphi 2.0 ユーザーズガイド．1996.
4) Borland 社（現 INPRISE 社）：C++Builder V3.0 オンラインマニュアル．1998.