臺灣二 OO 三年國際科學展覽會

科　　別：數學科

作品名稱：隨機遞迴數列及渾沌現象

得獎獎項：大會獎第三名

　　莫斯科二〇〇三年科學博覽會

學　　校：台北市立建國高級中學

作　　者：吳孟軒

隨機遞迴數列及渾沌現象

研究摘要

給定一個 $p\in(0, 1)$，令 $k_0 = 0$, p_0 在(0, 1)間隨機分布,定義 k_1 爲能使 $\frac{p^k}{1-p_0} \leq 1$ 的最小正整數 k，而 $p_1 = \frac{p^{k_1}}{1-p_0}$ ；相同的，對於給定的 k_{n-1}，k_n 爲能使$\frac{p^k}{1-p_{n-1}} \leq 1$ 的最小正整數 k，$p_n = \frac{p^{k_n}}{1-p_{n-1}}$ 。若存在 k_n 使得 $\frac{p^{k_n}}{1-p_{n-1}} = 1$，則稱 $p \in I_n$ ； 若對於所有的 n 與 k_n，$\frac{p^{k_n}}{1-p_{n-1}} < 1$，則稱 $p \in I_\infty$。如此區間(0, 1)可分解成集合 $I_1, I_2, \ldots, I_\infty$。

本主題研究當 p 爲定值且 p_0 在(0, 1)間隨機分布時，p_n 的分布情形，是否會發生渾沌現象，以及當 p_0 集中分布在同一點 p 時，分別考慮所有 $k_n=1$ 及 $k_n=2$，求出能使 $p_n=1$ 的 p 值。本主題也探討對於當 p 固定時，p_0 在哪些範圍會產生 Lyapunov Exponent 的最小值。主要的結果如下：

(1)當 p_0 在(0, 1)間隨機分布時，p_1 的機率密度函數 $f_1(p_1)$爲$\frac{p}{(1-p)p_1^2}$ ；

p_2 的機率密度函數 $f_2(p_2)$爲$\frac{p}{(1-p)p_2^2}\sum_{k=k_{2(\min)}}^{\infty}\frac{p^k}{\left(1-\frac{p^k}{p_2}\right)^2}$；當 $n\geq 2$ 時，

$$f_n(p_n) = \sum_{k=k_{n(\min)}}^{\infty}\left[f_{n-1}\left(1-\frac{p^k}{p_n}\right)\cdot\frac{p^k}{p_n^2}\right]。$$

代入次數增加時，若 $p\leq\frac{1}{4}$ ，則 $f_n(p_n)$的分布逐漸集中於一點，且此點的位置趨於固定，而其他位置的分布則逐漸減少。

若 $p > \frac{1}{4}$ ，則 $f_n(p_n)$在(p, 1)間各處都有分布，產生渾沌現象。當 p 固定時，函數 $f_n(p_n)$分布較周圍高的點也並非只有一點，而是隨代入次數增加。但當 p 接近 1 時，$f_n(p_n)$在各處的分布差異較不明顯，接近均勻分布。

(2)我去年研究時，已發現若固定 $p_0 = p$，且考慮 $k_n = 1$，把表示 p_n 的分數展開，則得到的結果可用以下遞迴式表示：$a_0(p) = 1$,

$a_1(p) = 1 - p$, $a_{n+1}(p) = a_n - p\,a_{n-1}$ $(n \geq 1)$, $p_n = \dfrac{pa_{n-1}(p)}{a_n(p)}$ 。

要使 $p_n = 1$，就要使分母與分子相等，亦即 $a_n(p) - pa_{n-1}(p) = 0$，也就是 $a_{n+1}(p) = 0$。此多項式 $a_n(p)$與 Fibonacci 多項式 $f_0(x) = 1$, $f_1(x) = x$, $f_{n+1}(x) = x\,f_n(x) + f_{n-1}(x)$ $(n \geq 1)$相關。經過轉換後，求得 $p_n = 1$ 的解爲 $\dfrac{1}{4\cos^2 \dfrac{\pi}{n+3}} \in I_n$。

(3)當 $p_0 = p$，且考慮 $k_n=2$ 時，$p_n=1$ 的解爲 $\dfrac{1}{2\cos \dfrac{\pi}{2n+3}}$，此數值同時也是當 $k_n = 1$ 時 p_{2n} 的解之正平方根。

(4)如果將遞迴式改爲 $p_n = \dfrac{p_{n-1}{}^{k_n}}{1-p_{n-1}}$，則：

若所有 $p_n > p_{n-1}$ 時，數列 p_n 收斂到 1；

當存在 $p_n = p_{n-1}$，且 k_n 爲 k，設 p_n 收斂到 x，則 x 爲 $x^{k-1} + x - 1 = 0$ 在 $0 < x < 1$ 範圍內的唯一根

(5)當 p 固定且 p_0 在(p, 1)間分布時，給定代入次數 n 並指定 k_{n+1} 的值(設爲 k)，尋找(p, 1)中的最長連續區間(a, b)，使得區間中的每個數值代入 n 次後，得到的 p_n 再代入一次，產生的 k_{n+1} 不是預期的值，此時(a, b)是 Lyapunov Exponent 的下限出現之處。

$$\text{Lyapunov Exponent}_{(\min)} = \frac{\log_2 \dfrac{(1-p)-(p^{k-1}-p^k)}{b-a}}{n}$$

關鍵字：隨機遞迴數列，渾沌，機率密度函數，Lyapunov Exponent

Random recursive sequences and chaos phenomena

Abstract

For a given fixed number $p \in (0,1)$, let $k_0=0$ and p_0 be randomly distributed in (0, 1). Define

$$k_1 := \inf\{k \geq 1: \frac{p^k}{1-p} \leq 1\},$$

and

$$p_1 := \frac{p^{k_1}}{1-p}.$$

Similarly, for given p_{n-1}, we define

$$k_n = \inf\{k \geq 1: \frac{p^k}{1-p_{n-1}} \leq 1\},$$

and

$$p_n := \frac{p^{k_n}}{1-p_{n-1}}.$$

Then $\{p_n, n \geq 0\}$ is a sequence of numbers belongs to (p,∞). For each $n=1,2,\ldots$, let $I_n := \{p \in (0,1)$; there exists k_n such that $p^{k_n}/(1-p_{n-1}) = 1\}$, and denote $I_\infty = \{p \in (0,1)$; $p^{k_n}/(1-p_{n-1}) < 1$ for all k and $k_n\}$. In this way the open interval (0,1) can be decomposed into union of sets $I_1,\ldots, I_\infty$.

In this article, we study the distribution of p_n when p is fixed and p_0 is randomly distributed in (0, 1), and we decide whether p_n will form chaotic phenomena. We also study the cases when all the p_0's concentrate on the same point p, when $k_n = 1$ or $k_n = 2$, and we calculate the value of p which let $p_n = 1$. Also we discuss which range of p_0 produce the smallest value of Lyapunov Exponent when p is fixed. Our main results show that

1. When p_0 is randomly distributed in (0, 1), the probability density function of p_1, $f_1(p_1)$, is $\frac{p}{(1-p)p_1^2}$. And $f_2(p_2)$, the probability

density function of p_2, is $\frac{p}{(1-p)p_2^2}\sum_{k=k_{2(\min)}}^{\infty}\frac{p^k}{\left(1-\frac{p^k}{p_2}\right)^2}$. When $n\geqq 2$,

$$f_n(p_n)=\sum_{k=k_{n(\min)}}^{\infty}\left[f_{n-1}\left(1-\frac{p^k}{p_n}\right)\cdot\frac{p^k}{p_n^2}\right].$$

2. Last year I found that if we let $k_n=1$, and expand the ratio of p_n, then we have the recursive relation $a_0(p) = 1$, $a_1(p) = 1-p$, $a_{n+1}(p) = a_n(p) - p\,a_{n-1}(p)$ $(n\geq 1)$, $p_n = \frac{p\,a_{n-1}(p)}{a_n(p)}$.

In order to make $p_n = 1$, we let $a_n(p) - p\,a_{n-1}(p) = 0$, that is $a_{n+1}(p) = 0$. Then the polynomial $a_n(p)$ relate with the Fibonacci polynomial $f_0(x) = 1$, $f_1(x) = x$, $f_{n+1}(x) = x\,f_n(x) + f_{n-1}(x)$ $(n\geqq 1)$. A simple transformation leads into

$\frac{1}{4\cos^2\frac{\pi}{n+3}} \in I_n$, for n=1,2,....

3. When $p_0 = p$ and $k_n = 2$, the solution of $p_n = 1$ is $p = \frac{1}{2\cos\frac{\pi}{2n+3}}$. This value is also the positive square root of the solution of $p_{2n} = 1$ when $k_n = 1$.

4. If we change the recursive formula to $p_n = \frac{p_{n-1}^{k_n}}{1-p_{n-1}}$, then

(a) If $p_n > p_{n-1}$ for all $n > 0$, the sequence p_n converges to 1.
(b) If there exists an integer n in which $p_n = p_{n-1}$, and let $k_n = k$, p_n converges to the unique number x between 0 and 1 which is the root of $x^{k-1} + x - 1 = 0$.

5. When p is fixed and p_0 is uniformly distributed in (0, 1), for given n, times of substitution, and k_{n+1}, we find the largest interval (a, b)

so that after n+1 substitutions of any number in (a, b), the value k_{n+1} is not the expected value. We find that the smallest Lyapunov Exponent occurs at (a, b). And the smallest Lyapunov Exponent is

$$\frac{\log_2 \dfrac{(1-p)-(p^{k-1}-p^k)}{b-a}}{n}.$$

Keywords. random recursive sequence, chaos,
probability density function, Lyapunov Exponent.

一、前言

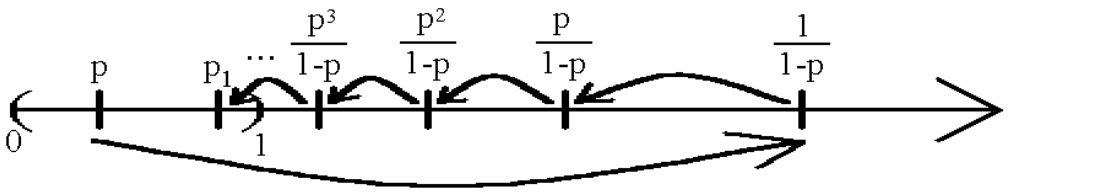
給定 p 值使 $0 < p < 1$，而 p_0 在(0, 1)間隨機分布，令 $p_n = \frac{p^{k_n}}{1 - p_{n-1}}$，其中 k_n 是能使 $p_n \leqq 1$ 的最小正整數，本主題研究

(1) p_n 的分布情形，以及是否會發生渾沌現象。

(2)當 p_0 集中分布在同一點 p 時，分別考慮所有 $k_n=1$ 及 $k_n=2$，求出能使 $p_n=1$ 的 p 值。

(3)本主題同時探討對於當 p 固定時，p_0 在哪些範圍會產生 Lyapunov Exponent 的最小值。

二、研究過程



給定一個 $p \in (0, 1)$，令 $k_0 = 0, p_0 = p$，定義 k_1 爲能使 $\frac{p^k}{1-p_0} \leqq 1$ 的最小正整數 k，而 $p_1 = \frac{p^{k_1}}{1-p_0}$；相同的，對於給定的 k_{n-1}，k_n 爲能使 $\frac{p^k}{1-p_{n-1}} \leqq 1$ 的最小正整數 k，$p_n = \frac{p^{k_n}}{1-p_{n-1}}$。若存在 k_n 使得 $\frac{p^{k_n}}{1-p_{n-1}} = 1$，則稱 $p \in I_n$；若對於所有的 n 與 k_n，$\frac{p^{k_n}}{1-p_{n-1}} < 1$，則稱 $p \in I_\infty$。如此區間(0, 1)可分解成集合 $I_1, I_2, \ldots, I_\infty$。

本主題研究當 p 爲定值且 p_0 在(0, 1)間隨機分布時，p_n 的分布情形，以及當 p_0 集中分布在同一點 p 時，分別考慮所有 $k_n=1$ 及 $k_n=2$，求出能使 $p_n=1$ 的 p 值。本主題也探討對於當 p 固定時，p_0 在哪些範圍能產生 Lyapunov Exponent 的最小值。

（一）考慮當 p 值固定，p_0 在(0, 1)間均勻分布時，各次代入所產生的 p_n 在(p, 1)間的分布情形：

首先由於 p_0 爲隨機分布，故 p_0 的機率密度函數 $f_0(p_0)$爲：

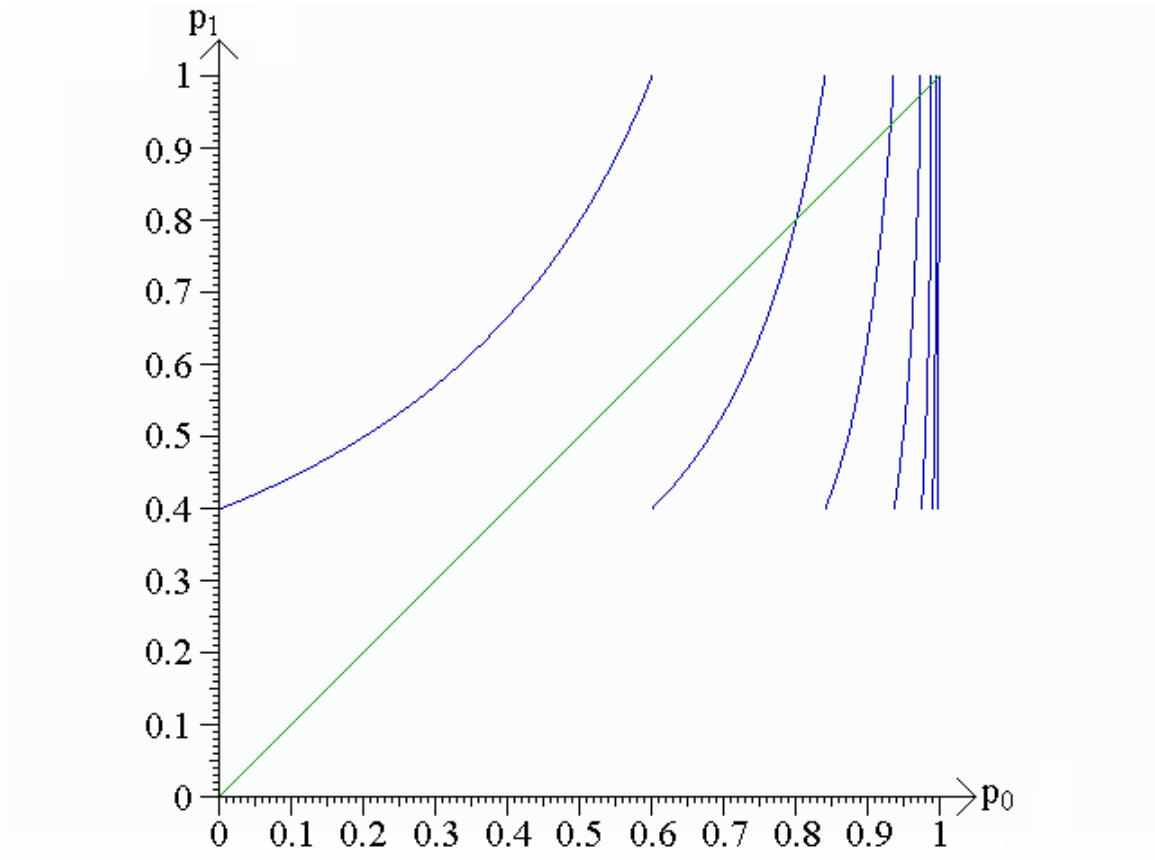
$$f_0(p_0)=\begin{cases} 1 & 0\leqq p_0<1 \\ 0 & p_0<0\text{或}p_0\geqq 1 \end{cases}$$

以下計算 p_n 的分配情形 $f_n(p_n)$：

1. 所有的 p_1 都滿足 $p\leqq p_1<1$，故只需考慮這段範圍內的情形。雖然對於這段範圍內的每個 p_1，都存在無限多個 p_0 代入後可得到此 p_1 值，但由 p_0 與 p_1 的關係圖（下圖以 p=0.4 爲例）中可看出，由左算起的第一條曲線對應的 p_0 範圍是 $0\leqq p_0<p$，往右每一條曲線對應的 p_0 範圍大小均是前一條的 p 倍。所以每一個 p_1 的機率密度函數等於只考慮第一條曲線時的情形乘以

$$1+p+p^2+\ldots=\frac{1}{1-p}$$ 倍。

圖 1



2. 由第一條曲線 $p_1 = \frac{p}{1-p_0}$ ，如改用 p_1 表示 p_0 則爲：

$p_0 = \frac{p_1 - p}{p_1} = 1 - \frac{p}{p_1}$

任何在 $p \leqq p_1 < 1$ 範圍內的 p_1，其機率密度函數值爲 $p_0 = 1 - \frac{p}{p_1}$

對 p_1 微分之值(p 爲常數)，而 $1 - \frac{p}{p_1}$ 對 p_1 的微分是 $\frac{p}{p_1^2}$ ，

故考慮所有曲線時，p_1 的機率密度函數爲：

$f_1(p_1) = \frac{p}{p_1^2} \times \frac{1}{1-p} = \frac{p}{(1-p)p_1^2}$

3. 對於 $p \leqq p_2 < 1$ 範圍內的 p_2，計算其機率密度函數 $f_2(p_2)$：

(1) 先計算哪些 p_1 代入後可以得到特定的 p_2：

由於 $p_2 = \frac{p^{k_2}}{1-p_1}$ ，故 $p_1 = 1 - \frac{p^{k_2}}{p_2}$ ，

其中 k_2 的最小值爲能使 $\frac{p^{k_2}}{p_2} < 1-p$ 的最小正整數，在此記爲 $k_{2(\min)}$。

(2) 令 $Q_k(p_2) = \frac{d}{dp_2}\left(1 - \frac{p^k}{p_2}\right)$，則某一特定的 p_2 值之機率密度函數 $f_2(p_2)$爲：

$$f_2(p_2) = \sum_{k=k_{2(\min)}}^{\infty} \left[f_1\left(1 - \frac{p^k}{p_2}\right) \cdot Q_k(p_2) \right]$$

其中 $f_1\left(1 - \frac{p^k}{p_2}\right) = \frac{p}{(1-p)\left(1 - \frac{p^k}{p_2}\right)^2}$ ，而 $Q_k(p_2) = \frac{p^k}{p_2^2}$

故 $f_2(p_2)$ 可化簡爲 $\sum_{k=k_{2(\min)}}^{\infty} \frac{p \cdot p^k}{(1-p)\left(1 - \frac{p^k}{p_2}\right)^2 p_2^2}$

提出相同的部分 $\frac{p}{(1-p){p_2}^2}$ ，

可得 $f_2(p_2)=\frac{p}{(1-p){p_2}^2}\sum_{k=k_{2(\min)}}^{\infty}\frac{p^k}{\left(1-\frac{p^k}{p_2}\right)^2}$

當 p 與 p_2 爲有理數時，$f_2(p_2)$爲無窮級數的和，目前尚未確定是否爲無理數，尚需時間證明。

4. $p_n(n\geqq 2)$的機率密度函數 $f_n(p_n)$可比照前面計算 $f_2(p_2)$的方法，列出遞迴式：

$$f_n(p_n)=\sum_{k=k_{n(\min)}}^{\infty}\left[f_{n-1}\left(1-\frac{p^k}{p_n}\right)\cdot Q_k(p_n)\right]$$

其中 $Q_k(p_n)=\frac{d}{dp_n}\left(1-\frac{p^k}{p_n}\right)=\frac{p^k}{{p_n}^2}$ ，

故 $f_n(p_n)=\sum_{k=k_{n(\min)}}^{\infty}\left[f_{n-1}\left(1-\frac{p^k}{p_n}\right)\cdot\frac{p^k}{{p_n}^2}\right]$

5. 機率密度函數 $f_1(p_1)\sim f_4(p_4)$在不同 p 值$(0.1\leqq p\leqq 0.9)$的分布圖：

（註：以下四張圖當中，x 軸代表 p_n，y 軸代表 p，z 軸代表 p_n 在此 p 值的機率密度函數值 $f_n(p_n)$，由於當 p 值接近 0 或 1 時，某些 $f_n(p_n)$將會過大，超出圖形外，此處爲方便觀察，故圖中 p 值的範圍是 $0.1\leqq p\leqq 0.9$）

圖 2

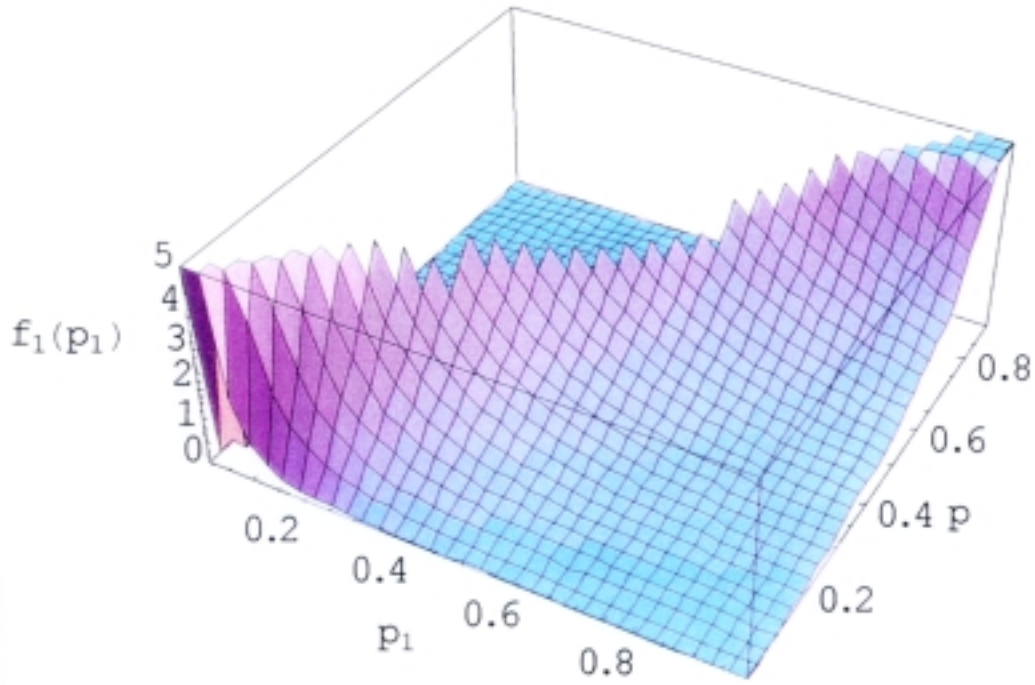

f1(p1)
5
4
3
2
1
0
0.2
0.4
0.6
0.8
p1
0.2
0.4 p
0.6
0.8

圖 3

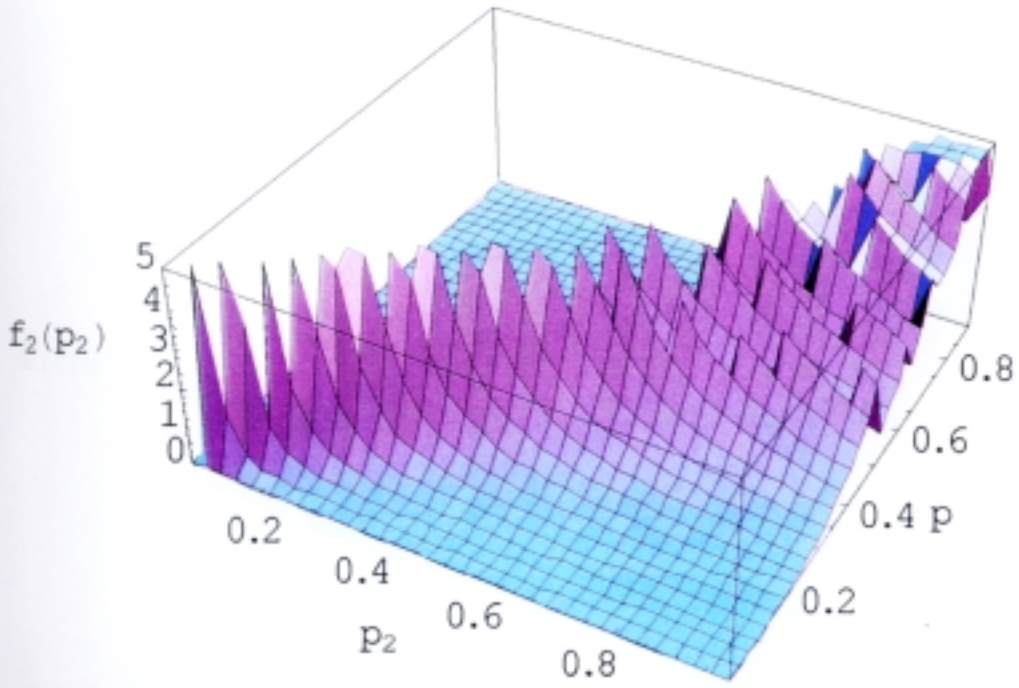

f2(p2)
5
4
3
2
1
0
0.2
0.4
0.6
0.8
p2
0.2
0.4 p
0.6
0.8

圖 4

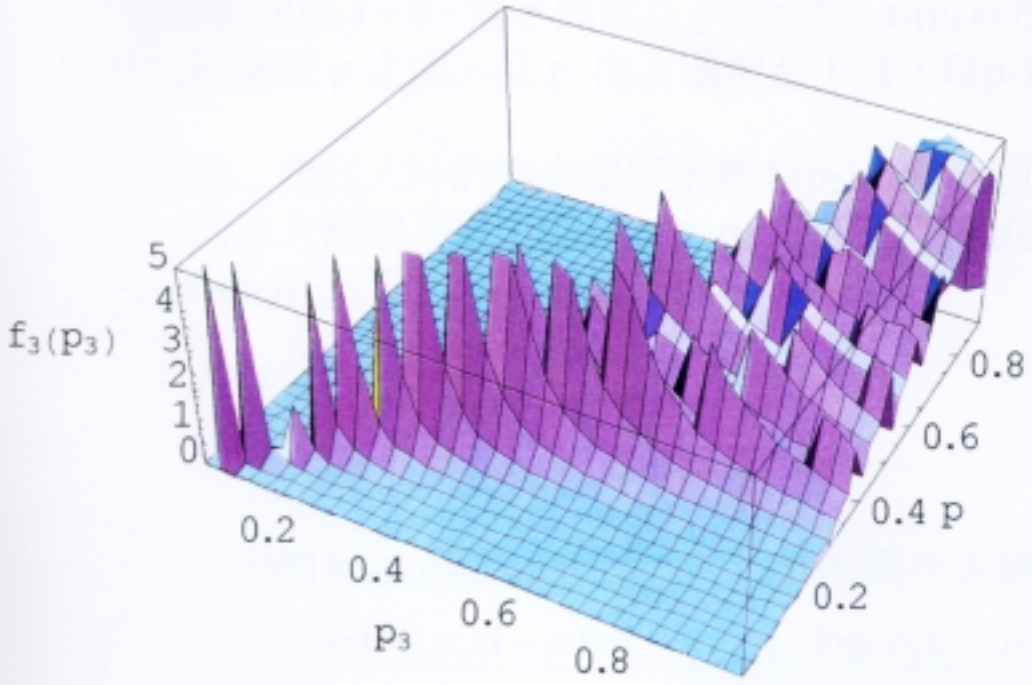
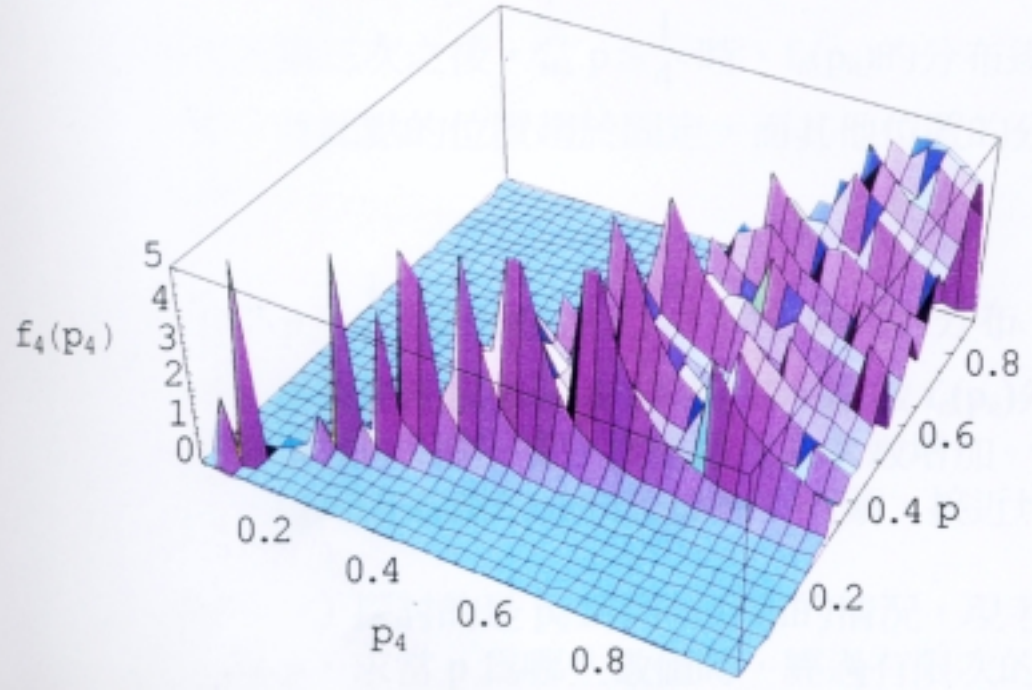

f3(p3)
5
4
3
2
1
0
0.2
0.4
0.6
0.8
p3
0.2
0.4 p
0.6
0.8

圖 5


f4(p4)
5
4
3
2
1
0
0.2
0.4
0.6
0.8
p4
0.2
0.4 p
0.6
0.8

6. 從上圖中觀察到的現象：

(1) 由於任何 p_0 在第一次代入得到的 p_1 必大於 p，故當 $p_1 \leqq p$ 時，$f_1(p_1) = 0$。當 p 固定且 $p_1 > p$ 時，$f_1(p_1)$的最大值出現在 p_1 趨近 p 之處，且 p_1 由 p 遞增至 1 時，$f_1(p_1)$將遞減。

(2) 第二次代入時的機率密度函數 $f_2(p_2)$，當 p 固定時，$f_2(p_2)$ 的最大值並不是出現在 $p_2 = p$，而是在其他位置，而函數 $f_2(p_2)$在此點出現不連續的情況，這是由於

$$f_2(p_2) = \frac{p}{(1-p)p_2^{\,2}} \sum_{k=k_{2(\min)}}^{\infty} \frac{p^k}{\left(1-\frac{p^k}{p_2}\right)^2}$$

式中的 $k_{2(min)}$等於能使 $\frac{p^{k_2}}{p_2} < 1 - p$ 的最小正整數，也就是 $k_{2(\min)} = \lceil \frac{\log(p_2(1-p))}{\log p} \rceil$，必須把$\frac{\log(p_2(1-p))}{\log p}$無條件進入成整數。當 p 固定時，兩個相近的 p_2 代入後得到的 k_2 可能爲不同的值，則在計算 Σ 內的無窮和時，兩者的起始 k 值不相同，造成函數 $f_2(p_2)$不連續。

(3) 代入第三次之後，當 $p \leqq \frac{1}{4}$ 時，$f_n(p_n)$的分布逐漸集中於一點，且此點的位置趨於固定，而其他位置的分布則逐漸減少。

(4) 但當 $p > \frac{1}{4}$ 時，$f_n(p_n)$在(p, 1)間各處都有分布，而非集中分布，產生渾沌現象。當 p 固定時，函數 $f_n(p_n)$分布較周圍高的點也並非只有一點，而是隨代入次數增加。但當 p 接近 1 時，$f_n(p_n)$在各處的分布差異較不明顯，接近均勻分布。

（二）前面（一）探討的是 p_0 爲均勻分布的情況，現考慮所有 p_0 都分布於同一點 p，求當 p 爲哪些數值時，經過有限次的代入，會使 $p_n = 1$？

去年曾經研究出當所有 $k_n = 1$ 時，$p_n = 1$ 的解爲 $\dfrac{1}{4\cos^2\dfrac{\pi}{n+3}}$ ：

當 $k_n = 1$ 時，p_n 表示成分數就相當於展開連分數 $\cfrac{p}{1-\cfrac{p}{1-\cfrac{p}{1-\cdots}}}$（共 n 層），

將此連分數化簡成分母、分子均爲多項式的形式後，則分母、分子均可由 Fibonacci 多項式演變而得：

先把 Fibonacci 多項式 $f_n(x)$相鄰兩項的次數差由 2 次改爲 1 次，如果最低次項爲 1 次項而非常數項，則把各項都降低 1 次。再將各項的符號改爲正負相間，其中最高次項爲正。最後把係數倒轉，即可得到多項式 $a_n(x)$，而 $p_n = \dfrac{p\, a_{n-1}(p)}{a_n(p)}$ 。

要求出 $p_n=1$ 的解，就令 $p\, a_{n-1}(p) = a_n(p)$，由於此遞迴式爲

$a_{n+1}(p) = a_n(p) - p\, a_{n-1}(p)$，故 $a_{n+1}(p) = 0$ 的解即爲 $p = 1$ 的解。

已知 Fibonacci 多項式的解爲 $x \in \{2i \cos\dfrac{j\pi}{n}\ , j = 1, 2, \ldots, n-1\}$，

故可推出 $p_n = 1$ 即 $a_{n+1}(p) = 0$ 的解爲

$p \in \{\dfrac{1}{4\cos^2\dfrac{j\pi}{n+3}}\ , j = 1, 2, \ldots, [\dfrac{n+2}{2}]\}$，但只有 $\dfrac{1}{4\cos^2\dfrac{\pi}{n+3}}$ 在合理範圍內，

因此 $\dfrac{1}{4\cos^2\dfrac{\pi}{n+3}} \in I_n$。

（三）將（二）中的 k_n 值由 1 改爲 2，探討當所有 $k_n = 2$ 時，哪些 p 值能使 $p_n = 1$？

1. 設k_n=2 時,q 代入 n 次可得q_n=1,即

$$\cfrac{q^2}{1-\cfrac{q^2}{1-\cfrac{\ddots}{1-\cfrac{q^2}{1-q}}}}=1 \qquad (1)$$

上式(1)中的連分數共有 n 層

2. 令 p=q^2,則

$$p_{2n}=\cfrac{p}{1-\cfrac{p}{1-\cfrac{\ddots}{1-\cfrac{p}{1-p}}}} \qquad (2)$$

上式(2)中的連分數共有 2n 層

3. 已知當k_n=1 時,將p_n的連分數展開,可得$\frac{p\,a_{n-1}(p)}{a_n(p)}$,其中
$a_0(p)=1, a_1(p)=1-p$,當 p≥2 時,$a_n(p)=a_{n-1}(p)-p\,a_{n-2}(p)$。
現把(2)式中連分數的下面 p 層展開:

$$p_{2n}=\cfrac{p}{1-\cfrac{p}{1-\cfrac{\ddots}{1-p_n}}}=\cfrac{p}{1-\cfrac{p}{1-\cfrac{\ddots}{1-\cfrac{p\,a_{n-1}(p)}{a_n(p)}}}} \qquad (3)$$

(3)式中央的連分數共有 n 層。右端各多項式$a_n(p)$的係數如下表:

（表 1 請見下頁）

表 1

	常數	p	p^2	p^3	p^4
$a_0(p)$	1				
$a_1(p)$	1	–1			
$a_2(p)$	1	–2			
$a_3(p)$	1	–3	+1		
$a_4(p)$	1	–4	+3		
$a_5(p)$	1	–5	+6	–1	
$a_6(p)$	1	–6	+10	–4	
$a_7(p)$	1	–7	+15	–10	+1
$a_8(p)$	1	–8	+21	–20	+5

4. 將(1)式左端q_n的連分數展開,可得$q_n = \dfrac{q^2 b_{n-1}(q)}{b_{n(q)}}$,其中

$b_0(q)$=1,$b_1(q)$=1–q,當 p≥2 時,$b_n(q) = b_{n-1}(q) - q^2 b_{n-2}(q)$。

因q_n=1,故分子與分母相等,$q^2 b_{n-1}(q) = b_n(q)$,即$b_{n+1}(q)$=0。下表爲各多項式$b_n(q)$的係數:

（表 2 請見下頁）

表 2

	常數	q	q^2	q^3	q^4	q^5	q^6	q^7	q^8
$b_0(q)$	1								
$b_1(q)$	1	–1							
$b_2(q)$	1	–1	–1						
$b_3(q)$	1	–1	–2	+1					
$b_4(q)$	1	–1	–3	+2	+1				
$b_5(q)$	1	–1	–4	+3	+3	–1			
$b_6(q)$	1	–1	–5	+4	+6	–3	–1		
$b_7(q)$	1	–1	–6	+5	+10	–6	–4	+1	
$b_8(q)$	1	–1	–7	+6	+15	–10	–10	+4	+1

5. 觀察表 1 與表 2,得知:

$$b_{n+1}(q) = a_n(q^2) - q\, a_{n-1}(q^2) \tag{4}$$

當 q_n=1 時,$b_{n+1}(q)$=0,代入(4)式得:

$$a_n(q^2) = q\, a_{n-1}(q^2) \tag{5}$$

由於q^2=p,且 $a_n(q^2)$ 不爲 0,故

$$\frac{a_n(p)}{q\, a_{n-1}(p)} = 1 \tag{6}$$

將(6)式代入(3)式,可得:

$$p_{2n}=\cfrac{p}{1-\cfrac{p}{1-\cfrac{\ddots}{1-\cfrac{p}{q}}}} \tag{7}$$

(7)式的連分數共 n 層。再將q^2=p 代入(7)式,得:

$$p_{2n}=\cfrac{q^2}{1-\cfrac{q^2}{1-\cfrac{\ddots}{1-\cfrac{q^2}{1-q}}}} \tag{8}$$

(8)式右端與(1)式右端完相同,故$p_{2n}=q_n$=1

6. 已知p_{2n}=1 的解爲$p=\dfrac{1}{4\cos^2\dfrac{\pi}{2n+3}}$,又因$q^2=p$,故

$$q=\sqrt{\frac{1}{4\cos^2\dfrac{\pi}{2n+3}}}=\frac{1}{2\cos\dfrac{\pi}{2n+3}} \tag{9}$$

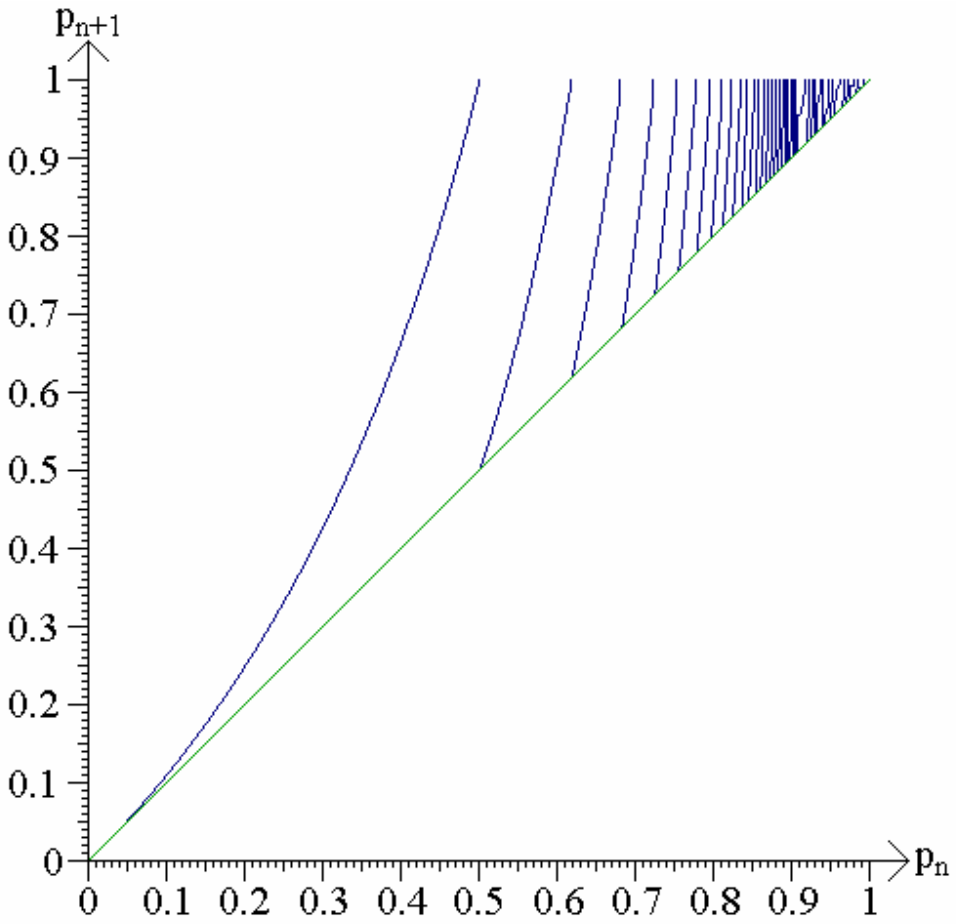
由(9)式,當k_n=2 時,$q=\dfrac{1}{2\cos\dfrac{\pi}{2n+3}}$可使$q_n$=1,即$\dfrac{1}{2\cos\dfrac{\pi}{2n+3}}\in I_n$,得證。

（四）變更遞迴式

如果將遞迴式改爲 $p_{n+1}=\frac{p_n^{k_n}}{1-p_n}$ ，即分子部分由p^{k_n} 改爲$p_n^{k_n}$ ，則與原來遞迴式的差異爲：

1. 原本 p_{n+1} 值受到 p 與 p_n 兩個值的影響，而現在 p_{n+1} 只由 p_n 來決定，因此無論原來的 p 值爲何， p_n 與 p_{n+1} 所形成的圖形只有一種：

圖 6



2. 由於分子爲$p_n^{k_n}$，故 p_{n+1} 的範圍是 $p_n \leqq p_{n+1} < 1$，其中最小值爲 p_n 而非 p，所以數列 p_n 爲非遞減，有兩種情形：

   (1) 若存在某個 n 使 $p_n = p_{n+1}$，則此項以後的每個數都將等於 p_n，故此數列收斂到 p_n。

   (2) 若所有的 p_n 與 p_{n+1} 都不相同，則 p_n 遞增且有上界 1，所以 p_n 收斂到 1。

   因此數列 p_n 在所有情況下均爲收斂。

3. 當存在 $p_n = p_{n+1}$ 時，設 $p_n = x$，則：

$$x = \frac{x^k}{1-x}$$

即 $x^k + x^2 - x = 0$

$\because x \neq 0,$

$\therefore x$ 爲 $x^{k-1} + x - 1 = 0$ 之一根

又 $\because$ $x^{k-1} + x - 1$ 在 $0 < x < 1$ 時遞增

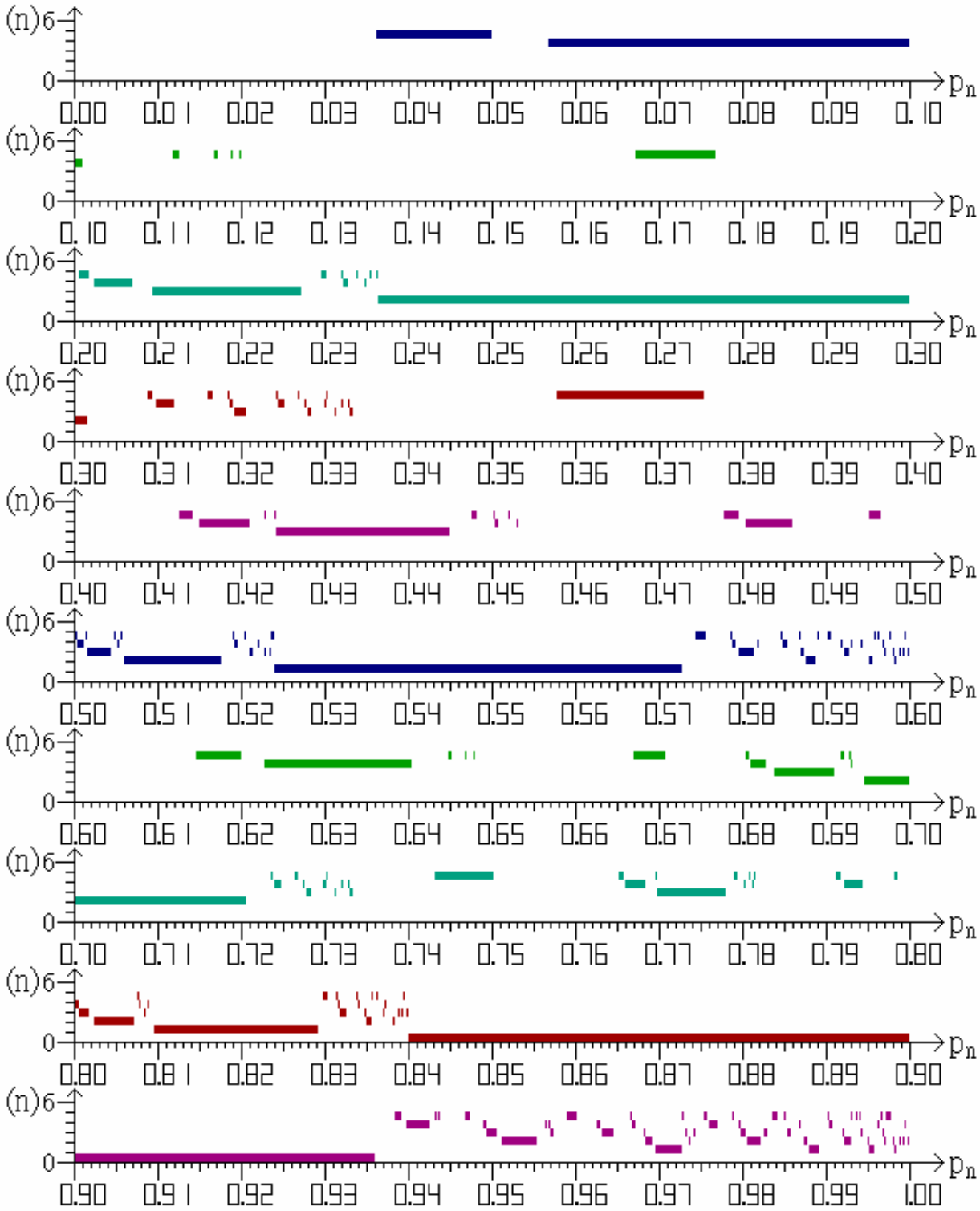
$\therefore x$ 爲 $x^{k-1} + x - 1 = 0$ 在 $0 < x < 1$ 範圍內唯一的根

（五）爲判斷 p_n 是否爲稠密分布，嘗試證明對於任何一個 (p, 1)內的區間(a, b)，所有在(p, 1)內的數 p_{n_1} 經過有限次的代入後，都可得到另一 p_{n_2} 在(a, b)當中：

1. 先從 a 與 b 分別爲 $1\text{-}p^{m-1}$ 與 $1\text{-}p^m$(m 爲正整數)形式的數開始，因爲$(1\text{-}p^{m-1}, 1\text{-}p^m)$中的任何一個數 p_n 再代入一次，都需要取 $k_{n+1}=m$，才可得到 p_{n+1}，亦即任何一個(p, 1)間的數，代入直到進入(a, b)中所需要的次數，將等於使 $k_{n+1}=m$ 需要的次數減 1，故此情況只須觀察每次代入取用的 k_n 值就行了。
下圖以 p=0.4, m=3 爲例，橫軸爲 0~1 間的數值(爲使圖形完整，一併加上 0~0.4 這一段)，縱軸爲此數值代入得到(0.84, 0.936)間的數所花費的最少次數（爲方便觀察，圖中最大刻度爲 6）。

（圖 7 請見下頁）

圖 7


(n)6
0
pn
0.00 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.10
0.10 0.11 0.12 0.13 0.14 0.15 0.16 0.17 0.18 0.19 0.20
0.20 0.21 0.22 0.23 0.24 0.25 0.26 0.27 0.28 0.29 0.30
0.30 0.31 0.32 0.33 0.34 0.35 0.36 0.37 0.38 0.39 0.40
0.40 0.41 0.42 0.43 0.44 0.45 0.46 0.47 0.48 0.49 0.50
0.50 0.51 0.52 0.53 0.54 0.55 0.56 0.57 0.58 0.59 0.60
0.60 0.61 0.62 0.63 0.64 0.65 0.66 0.67 0.68 0.69 0.70
0.70 0.71 0.72 0.73 0.74 0.75 0.76 0.77 0.78 0.79 0.80
0.80 0.81 0.82 0.83 0.84 0.85 0.86 0.87 0.88 0.89 0.90
0.90 0.91 0.92 0.93 0.94 0.95 0.96 0.97 0.98 0.99 1.00

2. 由上圖觀察到的現象：

(1) 區間(0, 1)當中，只有其中一段在第一次代入時，k_n 就達到給定的值(在此例中 $k_1=3$)，這是由於 k_1 的值隨 p_0 遞增，所以只有一段連續的 p_0 可以產生特定的 k_1。

(2) 第二次以後的每一次代入，都有無限多段區間的 p_0，在第 n 次代入時產生的 k_n 是給定的值，因爲前一次代入時的 k_{n-1} 有無限多種可能，而每個不同的 k_{n-1} 值代表不同的區間。

(3) 每次代入後，從(0, 1)中取出 k_n 是給定值的 p_0 區間，則剩下的部分將形成 “空隙”，如果再繼續代入，則每個空隙都將分成無限多個更小的空隙，如此不斷反覆下去， “最大空隙”的長度也越來越小。

3. 撰寫電腦程式，尋找當 p 與預期達到的 k_n 固定時， “最大空隙”出現的位置以及大小：

(1) 程式 chaos32.exe 是觀察 p 值固定時(例如 p=0.4)，p_0 反覆代入 n 次(例如 n=1,此處 p_0 可與 p 不同,可在 p~1 間變化)，得到的 $p_1,p_2,...,p_n$ 與 $k_1,k_2,...,k_n$，判斷哪些區間的 p_0 並不能使代入 n 次得到的 p_n 下一次代入所產生的 k_{n+1} 是預期的值(如 $k_{n+1}=3$)。由於得到的結果將是由無限多個小區間組成，所以要輸出的是這些片段當中較長的幾段。

(2) 例如固定 p=0.4, n=1 且要使 $k_2=3$，可逆推得到：
$0.84 \leqq p_1 < 0.936$,
再往回推 p_0 的值，得到以下幾個區間：
$0.5238... \leqq p_0 < 0.5726...$，此時 $k_1=1,k_2=2$；
$0.8095... \leqq p_0 < 0.8290...$，此時 $k_1=2,k_2=2$；
另外還有 $k_1=3,k_2=2$； ...，
其中 k_1 可以是任何正整數，只要 $k_2=2$ 就行了。
相反的，0.4~1 之間不能使 $k_2=3$ 的 p_0 值範圍，即爲：

$0.4 \leqq p_0 < 0.5238...$；
$0.5726... \leqq p_0 < 0.8095...$；
...；
若列出前三段區間起訖點的各 k_n 值，以及該起訖點的實際數值和區間大小，則是：

from 0 to 1 2	0.4000000000	0.5238095238	0.1238095238
from 1 3 to 2 2	0.5726495726	0.8095238095	0.2368742369
from 2 3 to 3 2	0.8290598291	0.9238095238	0.0947496947

例如第二列左邊的 “1 3”與 “2 2”指的是 0.5726...與 0.8095...兩值分別作爲 p_0 代入所產生的 k_1 與 k_2。由於第一行的 0.4 是整個觀察範圍的起始點，所以此數值用 “0”來表示，而不列出其 k_n 值。

(3) 這些起訖點的 k_n 值有一個特性，就是當起始值 p_0 越大時，代入所產生的各 $k_i(1 \leqq i \leqq n)$也會越大。因此要判斷兩串 k_i 值分別代表的 p_0 值何者較大，只要看第一個不同的值何者較大就行了，例如上表中 k_i= “2 3”的 p_0=0.8290...，就大於 k_i= “2 2”的 p_0=0.8095...。因此在嘗試的過程中，是把各 k_i 由小到大分別檢驗，如此就不會有遺漏了。
另外當某一 k_i 越大而其他 k_i 固定時，所形成的區間也越來越小(例如第二列的 “1 3”至 “2 2”與第三列的 “2 3”至 “3 2”只有 k_1 增加而 k_2 不變，故後者比前者小)，但第一個區間有可能例外，因爲這個區間被起始值 0.4 切到了，是不完整的區間，同樣的情形也可能發生在其他的 k_i 值。因此要找出哪個區間最小，只要每一個 k_i 取最小的兩個值檢驗就可以了。上面取了三個區間，但第三個必定比第二個小，所以實際上不需要檢驗這個區間。因此代入次數爲 n 時，檢驗的總區間數不會超過 2^n 個。

(4) 但 “k_i 取最小的兩個值”並不是隨時都能取 1 和 2，當區間被切到時就不是了。再舉 p=0.4, n=2, k_2=3 的例子如下：

from 0 to 1 2 2	0.4000000000	0.5058823529	0.1058823529

from 1 2 3 to 1 3 2	0.5175257732	0.5670103093	0.0494845361
from 1 to 2 1 2	0.6000000000	0.6945454545	0.0945454545
from 2 1 3 to 2 2 2	0.7205970149	0.8023529412	0.0817559263

第三列的區間終止值是“2 1 2”，但第一列不是“1 1 2”而是“1 2 2”，原因是當 p=0.4 時，連續兩個 k_i 是“1 1”並不合理，若一串 ki 中有連續兩個值 k_a 與 k_{a+1} 爲“1 1”，表示若把 p_a 當作 p_0 重新開始代入，則前兩個 k_i 值 k_1 與 k_2 都爲 1。前面提及當 p_0 增大時，各 k_i 也增大，但當 p=0.4 時，p_0 最小爲 0.4，此時各 k_i 值依序爲 1,2,1,2,2,...，表示 k_1=1 時，k_2 最小爲 2，不可能出現連續兩個 k_i 都是 1。

(5) 要判斷某一串 k_i 值是否合理，就先把 p_0 爲最小值時的前 n 個 k_i 值存起來，名稱分別爲 $k_{n(1)},k_{n(2)},...,k_{n(n)}$，並設一個變數 warning，以警示各 k_i 值是否太小。剛開始 warning=0，代表沒有問題，接下來從 k_1 開始依序判斷每一 k_i 值，如果 $k_i>k_{n(warning+1)}$，則表示此 k_i 值通過檢驗，warning 值歸零；如果 $k_i<k_{n(warning+1)}$，則表示此 k_i 值太小故不合理，立刻停止檢驗；當 $k_i=k_{n(warning+1)}$時，表示這個 k_i 值正好是可以接受的下限，將 warning 的值加 1，繼續檢驗下一項 k_{i+1} 是否合理。例如 p=0.4, $k_{n(i)}$依序爲 1,2,1,2,2,...時，k_1~k_7 分別爲 2,3,1,2,1,2,3 是合理的，但爲 2,3,1,2,1,2,1 就不正確了。因爲 k_3~k_6 連續 4 項與 $k_{n(1)}$~$k_{n(4)}$相同，所以 warning=4，到了 k_7，因 $k_7=1<k_{n(5)}=2$，故不合理。

(6) 本程式共需輸入五個數值，分別爲 p, 預期的 k_{n+1} 值, 每個 k_i 檢驗的數值範圍 k_range(例如 k_i 最小爲 2，檢驗範圍設爲 3，則 k_i 就從 2 一直試驗到 5), 代入次數 n, 以及最多輸出列數(輸入 0 表示不限制)，結果將輸出在檔案 chaos32.out，螢幕並顯示輸出總列數或錯誤訊息(如果有輸入值不正確)。

(7) 改進後的程式 chaos33.exe 則是把 k_range 設爲 2，試驗完所有區間後再輸出最大的那一個區間，以及試驗過的區間數量。此程式可以一次檢驗一段範圍的代入次數 n(例如 1 ≦n≦5，就連續輸出 5 筆結果)，共需輸入四個值：p, 預期

的 k_{n+1} 值, n 的起始值, 以及 n 的終止值。

4. 程式執行結果：

(1) 固定 p=0.4, k=3，以 n=1~22 分別代入的情形：

註①：除 n=1 時起始值爲 p=0.4 以外，其餘起始值的 k_1~k_{n-1} 與終止值相同，僅 k_n 改爲 1，k_{n+1} 改爲 3

註②：以 p_0=0.4 代入，產生的各 k_i 值依序爲

1, 2, 1, 2, 2, 1, 2, 5, 1, 2, 2, 1, …

表 3

n	終止值之 k_1~k_{n+1}
1	1 2
2	1 2 2
3	2 1 2 2
4	2 1 2 2 2
5	1 2 2 1 2 2
6	2 1 2 2 1 2 2
7	2 [1 2 1 2 2 2] 2
8	1 2 2 1 2 2 1 2 2
9	2 [1 2 1 2 2 2] 1 2 2
10	2 1 2 2 [1 2 1 2 2 2] 2
11	1 2 2 [1 2 1 2 2 2] 1 2 2
12	2 1 2 2 [1 2 1 2 2 2] 1 2 2
13	2 [1 2 1 2 2 2] [1 2 1 2 2 2] 2
14	1 2 2 1 2 2 [1 2 1 2 2 2] 1 2 2
15	2 [1 2 1 2 2 2] [1 2 1 2 2 2] 1 2 2
16	2 1 2 2 [1 2 1 2 2 2] [1 2 1 2 2 2] 2
17	1 2 2 [1 2 1 2 2 2] [1 2 1 2 2 2] 1 2 2
18	2 1 2 2 [1 2 1 2 2 2] [1 2 1 2 2 2] 1 2 2
19	2 [1 2 1 2 2 2] [1 2 1 2 2 2] [1 2 1 2 2 2] 2
20	1 2 2 1 2 2 [1 2 1 2 2 2] [1 2 1 2 2 2] 1 2 2
21	2 [1 2 1 2 2 2] [1 2 1 2 2 2] [1 2 1 2 2 2] 1 2 2
22	2 1 2 2 [1 2 1 2 2 2] [1 2 1 2 2 2] [1 2 1 2 2 2] 2

(2) 從上面結果觀察到的現象：

當 $n \geqq 7$ 時，除 n=8 以外，$k_1 \sim k_{n+1}$ 的值當中，連續出現至少一組連續 6 個數爲 “1, 2, 1, 2, 2, 2”，且 n 每增加 6，$k_1 \sim k_{n+1}$ 就增加一組 “1, 2, 1, 2, 2, 2”。

(3) 試驗其他 p 值，觀察是否有同一組數值在 $k_1 \sim k_{n+1}$ 中連續重複出現的情況：

表 4

No.	以 p 代入得到的前幾個 k_i	p	重複出現的 k_i
1	1 2 3 4 5 6 7…	0.4404353704…	1 3
2	1 2 1 3 1 4 1 5…	0.4081535357…	1 2 2
3	1 1 2 1 1 2 1 1 2 3…	0.3389497828…	1 1 2 1 1 2 1 2
4	1 5 2 3 4 6…	0.4900663908…	2
5	2 3 4 2 3 4 5…	0.5730436618…	2 4
6	1 3 1 4 1 4 2…	0.450713228	1 3 2
7	4 9 8 8 7 5 5…	0.7182818284…	5
8	2 2 3 2 3 4 2…	0.5367813989…	2 2 3 3
9	2 2 4 2 2 4 3…	0.54399	2 3
10	2 4 2 5 2 5 3…	0.5827553856	2 4 3
11	2 2 4 2 5 2 8…	0.54607	2 3
12	1 3 3 2 2 2 1 4…	0.4676	其他規律
13	1 1 2 2 3 3 4…	0.363378818…	尚在尋找中

(4) 從上表觀察到的現象：

①上表中的第 1 ~11 列，重複出現的 k_i 都是以 p 代入所得到最初幾個 k_i 值組成，其中最後一個值要加 1，例如第 6 列右欄的 “1, 3, 2”是由左欄前 3 項 “1, 3, 1”的最後一項 “1”改爲 “2”而形成。

②重複出現的幾個 k_i 值，與用 p 代入產生的 k_1 相差小於等於 2，故這些 k_i 值如果取用以 p 代入所得到的前幾個 k_i 值，則在取到一個 $\geqq k_1+3$ 的數之前就必須停下來。

③但並非每次都能一直取到有 $\geqq k_1+3$ 的數出現爲止，有時遇到較小的數就停止了。例如第 10 列左欄的 “2, 4, 2,

5,…”，遇到 $k_1+3=2+3=5$ 才停下來，但第 9 列 “2, 2, 4, 2,…”只取到第 2 項就終止了，下一項是 $k_1+2=4$。

④有些例外的情況，例如第 12 列，當 p=0.4676 時，終止值的 k_i 並非以同一組數值反覆循環，而是中間出現 “1, 3, 3”，兩旁全部爲 “2”：
n=19, “1, 3, 3”的前面有 9 個 “2”，後面有 8 個 “2”。
n=22, “1, 3, 3”的前面有 10 個 “2”，後面有 10 個 “2”。
n=24, “1, 3, 3”的前面有 11 個 “2”，後面有 11 個 “2”。
可知 “1, 3, 3”大致出現在中央處。

⑤上表第 13 列的 p 值代入時，取 k=3, n=17~20 的情形如下：

表 5

n	終止值之 k_1~k_{n+1}
17	1 2 1 2 1 2 1 1 3 1 2 1 1 3 1 2 1 2
18	1 2 1 2 1 2 1 1 3 1 2 1 2 1 1 3 1 2 2
19	1 2 1 2 1 2 1 1 3 1 2 1 2 1 1 3 1 2 1 2
20	1 2 1 2 1 1 3 1 2 1 1 3 1 2 1 1 3 1 2 1 2

其中 n=17~19 的 k_1~k_{12} 完全相同，但到了 n=20，卻只有前面 5 項相同。另外 n=17 之結果連續出現 2 次 “1 1 3 1 2”，n=20 則出現 3 次，但與直接用 p 代入所得的 k_1~k_5 不合，且 n=18, 19 只分別出現 2 次不連續的 “1 1 3 1 2”，所以 “1 1 3 1 2”是否算作 “反覆循環”，目前還不確定。

三、研究結果與討論：

（一）使用機率密度函數來分析當 p 固定，而 p_0 爲均勻分布時，代入得到 p_n 的分布情形：

1. p_1 的機率密度函數 $f_1(p_1)$與 p_1^2 成反比。

2. 到了 p_2 時，$f_2(p_2)$就形成無窮級數$\frac{p}{(1-p)p_2^2}\sum_{k=k_{2(\min)}}^{\infty}\frac{p^k}{\left(1-\frac{p^k}{p_2}\right)^2}$的和。

3. 而 p_2 以後的分布情形可用遞迴式 $f_n(p_n)=\sum_{k=k_{n(\min)}}^{\infty}\left[f_{n-1}\left(1-\frac{p^k}{p_n}\right)\cdot\frac{p^k}{p_n^2}\right]$表示，此一部分目前尚在研究中。

（二）當 p_0 固定爲 p 時，令所有 $k_n=1$，則 $p_n=1$ 的解相當於把 p_n 以 p 表示的連分數，即

$$\cfrac{p}{1-\cfrac{p}{1-\cfrac{\cdots}{1-p}}}=1$$

的解（連分數共 n 層）。把上式左端展開，p_n 可轉換成 $\frac{p\,a_{n-1}(p)}{a_n(p)}$，其中 $a_0(p)=1$, $a_1(p)=1-p$, $a_{n+1}(p)=a_n(p)-p\,a_{n-1}(p)$。此多項式可經由 Fibonacci 多項式轉換而得：把 Fibonacci 多項式 $f_n(x)$相鄰兩項的次數差由 2 次改爲 1 次，如果最低次項爲 1 次項而非常數項，則把各項都降低 1 次。再將各項的符號改爲正負相間，其中最高次項爲正。最後把係數倒轉，即可得到多項式 $a_n(p)$。

由 Fibonacci 多項式的解爲 $x\in\{2i\cos\frac{j\pi}{n}, j=1, 2, \ldots, n-1\}$，可推

出 $p_n = 1$ 即 $a_{n+1}(p) = 0$ 的解爲 $\dfrac{1}{4\cos^2\dfrac{\pi}{n+3}} \in I_n$。

（三）令所有 $k_n=2$，則 $p_n=1$ 的解爲 $\dfrac{1}{2\cos\dfrac{\pi}{2n+3}}$，此爲當 $k_n = 1$ 時，$p_{2n} = 1$ 的解之正平方根。

（四）如果將遞迴式改爲 $p_n = \dfrac{p_{n-1}{}^{k_n}}{1-p_{n-1}}$，則 p_n 必大於等於 p_{n-1}，故 p_n 爲非遞減。又因 p_n 有上界 1，所以數列 p_n 收斂。
當所有 $p_n > p_{n-1}$ 時，數列 p_n 收斂到 1；
若存在 $p_n = p_{n-1}$，且 k_n 爲 k，設 p_n 收斂到 x，則

$$x = \frac{x^k}{1-x}$$

即 $x^k + x^2 - x = 0$

$\because x \neq 0,$

$\therefore x$ 爲 $x^{k-1} + x - 1 = 0$ 之一根

又 $\because$ $x^{k-1} + x - 1$ 在 $0 < x < 1$ 時遞增

$\therefore x$ 爲 $x^{k-1} + x - 1 = 0$ 在 $0 < x < 1$ 範圍內唯一的根

（五）當 p 固定時，給定代入次數 n 與下次代入時預期產生的 k_{n+1}(設爲 k)，尋找(p, 1)中的最長連續區間(a, b)，使得區間中的每個數值代入 n 次後，得到的 p_n 再代入一次，產生的 k_{n+1} 不是預期的值：

當 p_0 由 p 開始遞增至 1，同時 p_n 也遞增，一直到 1 之後再回到 p 重新開始遞增。現在取一段區間(a, b)，並把區間中的每個數都作爲 p_0 並代入 n 次，如果產生的 p_n 把(p, 1)完全涵蓋，則(p, 1)當中必有一段區間在下一次代入時 k_{n+1} 爲預期的值。故 p_n 的範圍不可能把(p, 1)的所有數都涵蓋，至少$(1-p^{k-1}, 1-p^k)$這段範圍下次代入時的 $k_{n+1}=k$，故不能包含此段。
假設區間(a, b)是所有滿足條件 “k_{n+1} 不是預期的值”的最大區

間，則因代入 n 次後，只能將(p, 1)除去($1-p^{k-1}$, $1-p^{k}$)的剩餘部分涵蓋，故區間(a, b)代入 n 次後，總共被放大$\frac{(1-p)-(p^{k-1}-p^{k})}{b-a}$ 倍。

由於(b-a)取的是最大值，所以在代入的過程中，此區間放大的倍數最小，也就是 Lyapunov Exponent 的下限出現之處。

$$\therefore \text{Lyapunov Exponent}_{(\min)} = \frac{\log_2 \frac{(1-p)-(p^{k-1}-p^{k})}{b-a}}{n}$$

四、結論與應用

（一）當 p 固定，p_0 在(0, 1)間均勻分布時，各 p_n 的機率密度函數 $f_n(p_n)$ 爲：

$f_1(p_1) = \frac{p}{(1-p)p_1^2}$ ，

$f_2(p_2) = \frac{p}{(1-p)p_2^2} \sum_{k=k_{2(\min)}}^{\infty} \frac{p^k}{\left(1-\frac{p^k}{p_2}\right)^2}$ ，

當 n≧2 時，$f_n(p_n) = \sum_{k=k_{n(\min)}}^{\infty} \left[f_{n-1}\left(1-\frac{p^k}{p_n}\right) \cdot \frac{p^k}{p_n^2} \right]$

代入次數增加時，若 $p \leq \frac{1}{4}$ ，則 $f_n(p_n)$的分布逐漸集中於一點，且此點的位置趨於固定，而其他位置的分布則逐漸減少。

若 $p > \frac{1}{4}$ ，則 $f_n(p_n)$在(p, 1)間各處都有分布，產生渾沌現象。當 p 固定時，函數 $f_n(p_n)$分布較周圍高的點也並非只有一點，而是隨代入次數增加。但當 p 接近 1 時，$f_n(p_n)$在各處的分布差異較不明顯，接近均勻分布。

（二）當 $p_0 = p$，爲固定值，且所有 $k_n=1$ 時，$p_n=1$ 的解爲 $\frac{1}{4\cos^2\frac{\pi}{n+3}}$ 。

（三）當 $p_0 = p$，且所有 $k_n=2$ 時，$p_n=1$ 的解爲 $\frac{1}{2\cos\frac{\pi}{2n+3}}$ 。

（四）將遞迴式改爲 $p_n = \frac{p_{n-1}^{k_n}}{1-p_{n-1}}$ ，則 p_n 收斂到 1 或 $x^{k-1}+x-1=0$ 在 $0<x<1$ 範圍內的唯一根，其中 k 爲正整數。

（五）當 p 固定時，給定代入次數 n 與下次代入時預期產生的 k_{n+1}(設

爲 k)，尋找(p, 1)中的最長連續區間(a, b)，使得區間中的每個數值代入 n 次後，得到的 p_n 再代入一次，產生的 k_{n+1} 不是預期的值：

當 n 逐漸增大時，$\frac{\log_2 \frac{(1-p)-(p^{k-1}-p^k)}{b-a}}{n}$ 趨近 Lyapunov Exponent 的最小值。

在某些 p 值代入時，以找出的 a 與 b 值作爲 p_0 實際代入求得 k_1~k_{n+1} 時，存在一組數值在 k_1~k_{n+1} 中反覆出現，而此組數值與用 p 作爲 p_0 代入時產生的最初幾個 k_i 相同(除了最後一個需要加 1 以外)，且這些 k_i 的最大值與最小值相差不超過 2。

另外有一些 p 值，例如 0.4676，以不同的 n 值代入後，k_1~k_{n+1} 的中央部分固定出現相同的幾個值(1, 3, 3)，兩端反覆出現相同數值(2)，且當 n 增大時，兩端出現此相同值的數量都增加。

當 p=0.363378818…，此時以 p_0 代入得到的最初幾個 k_n 分別是 1, 1, 2, 2, 3, 3, 4,…，但以幾個不同 n 值代入後，k_i 的循環的情況不完全相同，此部分尚在研究中。

五、參考文獻

（一）Alexandru Lupaş (1999). A Guide of Fibonacci and Lucas Polynomials. Octogon, Math. Magazine, vol.7, No.1, 2-12.

（二）James Gleick (1991). Chaos: Making a New Science. 天下文化。

（三）http://www.research.att.com/~njas/sequences/, The On-Line Encyclopedia Integer Sequences, A011973, Triangle of numbers $\{C_k^{n-k}, n \geq 0, 0 \leq k \leq [\frac{n}{2}]\}$; or, triangle of coefficients of Fibonacci polynomials.

（四）吳振奎，世界數學名題欣賞叢書(2)斐波那契數列。九章數學。

六、附錄

程式總表

程式名稱	用途
Chaos.bas	把輸入的 p 值代入 10,000 次，以 0.002 爲組距畫出 p_1~$p_{10,000}$ 在各組出現次數直方圖，並以 0.00001 爲單位印出每次代入的結果。圖形上方顯示 0~0.998(以 0.002 爲單位)分別代入 p_n 值計算的結果 p_{n+1}(最小單位 0.01)。
Chaos2.bas	把輸入的 p 值代入 80 次，以 15 位小數列出每次計算結果。
Chaos3.bas	用 3 種顏色在坐標圖上顯示出 p=0~399/400(以 1/400 爲單位)時，p_1~p_3 的值。
Chaos4.bas	修改 Chaos3.bas， p 值的間隔改爲 1/800，因輸出的圖表過大，故分左下、左上和右上等三部分輸出。(圖表右下沒有內容)
Chaos5.bas	設定一段 0~1 之間的範圍，在此範圍中，每隔一定的間隔，就取一個數當作 p 值，代入指定的次數，輸出得到的結果。
Chaos6.bas	改進 Chaos.bas，當直方圖長度超出 140 像素時，會自動縮小圖形的比例爲原本的一半，以避免超出範圍外。
Chaos7.bas	輸入正整數的分子及分母作爲 p 值，計算 p_1~p_{20} 的值，並分別以最簡分數和小數表示。
Chaos8.bas	改進 Chaos6.bas，$p<0.7$ 時代入 25,000 次，$p \geq 0.7$ 只代入 5,000 次(p 愈大時，計算花費時間愈長，且 p_n 出現的範圍也較小，只在 p 與 1 之間，故代入太多次，直方圖會太長)，圖形完成後，列出長條圖 500 根直條分別表示的數字(限於螢幕空間，每個數最多只能顯示 4 位數字，超過 9,999 就會溢位)。

Chaos9.bas	改進 Chaos8.bas，可改變觀察範圍，輸入 0~1 之間的任何區間作爲觀察範圍，即以此範圍的 1/500 作爲組距，統計 p 值代入特定次數(p<0.7 次數同 Chaos8.bas，p≧0.7 增加爲 10,000 次以利觀察)後每次的結果在這 500 組中出現的次數，並以觀察範圍的 1/100,000 爲最小單位，在方格上描出每次出現的數。最後與 Chaos8.bas 一樣列出長條圖各直條的確實數字，若數字超過 9,999，顯示不下時，即輸出 “E”和前 3 位數，例如 11,850 即爲 “E118”。
Chaos10.bas	改進 Chaos2.bas，先輸入 p 值與 p_0 值(可以不相同)，再計算 p_1~p_{80} 的值。
Chaos11.bas	輸入 p 值和代入次數，畫出 0 <p_n < 1 之間 $p_{n+1} = \frac{p^{k_n}}{p_n}$ 與 $p_{n+1} = p_n$ 的函數圖形，並用淺藍色折線表示 p 代入時的情形(直線表示由 p_n 得到 p_{n+1} 的運算，橫線表示把 p_{n+1} 由縱軸移到橫軸，以作爲新的 p_n，進行下一次運算)，解析度爲 400×400。
Chaos12.bas	改進 Chaos11.bas，可以設定座標圖中橫軸與縱軸的觀察範圍。
Chaos13.bas	將 Chaos11.bas 輸出圖形的解析度改爲 500×400，以配合 Chaos9.bas 的圖形觀察。
Chaos14.bas	把遞迴式改爲 $p_{n+1} = \frac{p_n{}^{k_n}}{p_n}$ 的函數圖形，此爲固定的圖形，不需輸入 p 值。
Chaos15.bas	輸入 p 值，代入 200 次，畫出圖形的橫軸爲代入次數 n，縱軸爲 p_n。
Chaos16.bas	輸入 p 值，反覆代入並把 0~1 以 0.002 爲單位分爲 500 個區間，統計 p_n 在各區間分布情況，當 p_n 在某一區間出現達 400 次時，程式即停止。
Chaos17.bas	輸入最多 15 個不同 p 值，各代入 100 次，每一 p 值已不同顏色畫出 p_1~p_{100} 的移動軌跡。
Chaos18.bas	輸入 p 與代入次數，把 p~1 分爲 10 個相等的區間，統計從每一區間移到另一區間的出現次數分布情形。

Chaos19.bas	輸入 p 值與代入次數，把 p~1 分爲 10 個相等區間，顯示與 Chaos18.bas 相同的統計結果，並產生以下檔案： (1) Number.txt：每次得到的 p_n 若小於 $\frac{p+1}{2}$ 則寫入 “0”；大於等於 $\frac{p+1}{2}$ 則寫入 “1”。 (2) Chaos.txt：將上面的 Number.txt 由二進位轉爲 256 進位，每一位元組表示一個 0~255 間的數。 (3) Decimal.txt：將(p, 1)分成的 10 個相等區間分別編號 0~9，此檔案依序紀錄每一個 p_n 所屬的區間編號(0~9)，作爲模擬亂數之用。
Chaos20.bas	改進 Chaos12.bas，縮短計算 p_n 所花費的時間，當 p 接近 1 時，程式執行速度大幅提升。
Chaos21.bas	輸入 p 及代入次數，在座標圖上畫出以每次代入前與代入後的數值(p_n, p_{n+1})爲座標的點。
Chaos22.bas	輸入 p 及代入次數，在座標圖上畫出當各 p 值代入後產生的 p_{n+1}，並用淺藍色折線表示 p_n 移動的軌跡。
Chaos23.bas	輸入 p 的起始值與終止值，把此區間內的各 p 值(以 $\frac{b-a}{500}$ 爲單位)代入預設的次數後，求出 p_n 軌跡的 Lyapunov Exponent，畫在座標圖上。
Chaos24.bas	輸入單一 p 值及代入次數，隨時計算 p_n 移動軌跡的 Lyapunov Exponent，p_n 每代入 100 次就更新螢幕顯示一次。
Chaos25.bas	同 Chaos23.bas，除畫出座標圖外，螢幕隨時更新顯示上一個 p 值計算的結果。
Chaos26.bas	輸入 p 值及代入次數，輸出各 k 值的出現次數。
Chaos27.bas	輸入 p 值，列出代入前 100 次所產生的 k_1~k_{100}。

Chaos28.bas	輸入 p 值(範圍 $0.25<p\leq 0.5$)、預期出現的 k_n 值以及 p_n 值的觀察範圍，在座標圖上畫出觀察範圍中的各 p_n 值(以觀察範圍的 $\frac{1}{500}$ 爲單位)分別最少需要代入多少次才能使 k_n 爲預期出現的值。 （若 $p\geq 0.5$，則 k_n 值不能輸入太小，否則永遠無法達到。下一程式 Chaos29.bas 具有檢驗 k_n 值合理性的功能）
Chaos29.bas	改進 Chaos28.bas，p 值範圍可爲 $0.25\leq p<1$，且觀察範圍可連續設定 6 次，把圖形的一部分逐漸放大。
Chaos30.bas	變更遞迴式：原遞迴式中，如果把 0~1 間的各 p_n 值與代入得到的 p_{n+1} 畫在座標圖上，則圖形是由曲線組成，現把各段曲線拉直，則遞迴式變爲 $p_{n+1}= p + \frac{(1-p)\,(p_n - (1-p^{k_n}))}{(p^{k_n}-p^{k_n+1})}$ 螢幕輸出內容同 Chaos20.bas。
Chaos31.c	輸入一個 p 值，然後輸入多組 $n, k_1, k_2, \ldots, k_n$，每輸入完一組，就輸出會產生這些 $k_1, k_2, \ldots, k_n$ 的 p_0 值。
Chaos32.c	觀察 p 值固定時(例如 p=0.4)，p_0 反覆代入 n 次(例如 n=1, 此處 p_0 可與 p 不同,可在 p~1 間變化)，得到的 $p_1,p_2,\ldots,p_n$ 與 $k_1,k_2,\ldots,k_n$，判斷哪些區間的 p_0 並不能使代入 n 次得到的 p_n 下一次代入所產生的 k_{n+1} 是預期的值(如 $k_{n+1}=3$)。由於得到的結果將是由無限多個小區間組成，所以要輸出的是這些片段當中較長的幾段，需設定輸出的片段數量。
Chaos33.c	把 k_range 設爲 2，試驗完所有區間後再輸出最大的那一個區間，以及試驗過的區間數量。此程式可以一次檢驗一段範圍的代入次數 n(例如 $1\leq n\leq 5$，就連續輸出 5 筆結果)，共需輸入四個值：p, 預期的 k_{n+1} 值, n 的起始值, 以及 n 的終止值。代入次數爲 n 時，最大試驗次數爲 2^n。
Chaos34.c	改正 Chaos33.c 的 bug。

Chaos35.c	前面 Chaos27.bas 是輸入 p 值，輸出各次代入的 k_n 值；此處輸入各 k_n 值，以二分搜尋法找出對應的 p 值。
Chaos36.c	設 p_0 在(0, 1)間均勻分布，輸入 p 與 p_n，輸出 p_n 的機率密度函數值。

以下是 chaos29.bas, chaos30.bas, chaos32.c,chaos34.c 的程式碼，謹供參考。

Chaos29.bas:

```
DIM R#(6, 2)
CLS
SCREEN 12
LINE (0, 0)-(639, 479), , BF
DO
    LOCATE 1, 1
    INPUT "P="; P#
LOOP UNTIL P# > .25 AND P# < .99
Kmin = 1
P1# = P# / (1 - P#)
DO WHILE P1# > 1
    Kmin = Kmin + 1
    P1# = P1# * P#
LOOP
DO
    LOCATE 2, 1
    INPUT "K="; K
LOOP WHILE K < Kmin
R#(0, 1) = 0: R#(0, 2) = 1
FOR L = 1 TO 5
    DO
        CHECK = 1
        LOCATE L + 2, 1
        PRINT USING "Range #"; L;
        INPUT R#(L, 1), R#(L, 2)
        IF R#(L, 1) >= R#(L, 2) THEN CHECK = 0
        REM IF R#(L, 1) < R#(L - 1, 1) OR R#(L, 2) > R#(L - 1, 2) THEN CHECK = 0
    LOOP WHILE CHECK = 0
    LINE (70, 150 + L * 50)-(570, 150 + L * 50), 0
    FOR I = 0 TO 500 STEP 5
        IF I MOD 50 = 0 THEN
            LINE (70 + I, 151 + L * 50)-(70 + I, 160 + L * 50), 0
        ELSE
            IF I MOD 25 = 0 THEN
                LINE (70 + I, 151 + L * 50)-(70 + I, 155 + L * 50), 0
            ELSE
                LINE (70 + I, 151 + L * 50)-(70 + I, 153 + L * 50), 0
            END IF
        END IF
    NEXT I
    REM IF L > 1 THEN
    REM       LINE (70 + 500 * (R#(L, 1) - R#(L - 1, 1)) / (R#(L - 1, 2) - R#(L - 1, 1)), 150 + 50 * (L - 1))-(70, 150
+ 50 * L), 8
    REM       LINE (70 + 500 * (R#(L, 2) - R#(L - 1, 1)) / (R#(L - 1, 2) - R#(L - 1, 1)), 150 + 50 * (L - 1))-(570, 150
+ 50 * L), 8
    REM END IF
    FOR I = 0 TO 499
        T = 0
        P0# = R#(L, 1) + (R#(L, 2) - R#(L, 1)) * I / 500
        IF P0# > 0 THEN
            DO
                T = T + 1
                A = 1
                P1# = P# / (1 - P0#)
                DO WHILE P1# > 1
                    P1# = P1# * P#
                    A = A + 1
                LOOP
                IF P1# = 1 THEN EXIT DO
```

```
                P0# = P1#
            LOOP UNTIL A = K
        END IF
        IF P0# > 0 AND T <= 6 THEN LINE (I + 70, 150 + L * 50 - T * 5)-(I + 70, 154 + L * 50 - T * 5), L
    NEXT I
NEXT L
END
```

Chaos30.bas:

```
CLS
SCREEN 12
LINE (0, 0)-(639, 479), 15, BF
DO
    INPUT "P0="; P#
LOOP WHILE P# <= 0 OR P# > .98
DO
    INPUT "Enter two numbers between 0 and 1 as the range"; B, E
    IF B < E AND B >= 0 AND E <= 1 THEN EXIT DO
LOOP
INPUT "Times:"; T
B# = B:E# = E
C# = (E# - B#) / 500
LINE (0, 0)-(640, 480), 15, BF
LINE (70, 100)-(570, 100), 6
FOR I = 0 TO 100
    IF I MOD 10 = 0 THEN
        LINE (70 + I * 5, 90)-(70 + I * 5, 99), 6
    ELSE
        IF I MOD 5 = 0 THEN
            LINE (70 + I * 5, 94)-(70 + I * 5, 99), 6
        ELSE
            LINE (70 + I * 5, 97)-(70 + I * 5, 99), 6
        END IF
    END IF
NEXT I
REM LOCATE 1, 1
REM PRINT "P0="; P#
REM PRINT "Range:"; B; "to"; E
REM LOCATE 6, 8
REM PRINT B
REM LOCATE 6, 71
REM PRINT E
DIM A(500)
P0# = P#
R = 1
FOR I = 1 TO T
    A = 0
    DO WHILE P0# < 1 - P# ^ A OR P0# >= 1 - P# ^ (A + 1)
        A = A + 1
    LOOP
    P1# = P# + (1 - P#) * (P0# - (1 - P# ^ A)) / (P# ^ A - P# ^ (A + 1))
    D = (P1# - B#) / C#
    IF 0 <= D AND D < 500 THEN
        A(INT(D)) = A(INT(D)) + 1
        IF A(INT(D)) = 375 / R THEN GOSUB SUB3 ELSE GOSUB SUB1
    END IF
    IF I MOD 1000 = 0 THEN
         LOCATE 3, 1
         PRINT USING "Times: ########,"; I;
    END IF
    IF P1# = 1# THEN S = I: LOCATE 25, 41: PRINT P1#: EXIT FOR
    P0# = P1#
NEXT I
S = I - 1
REM LOCATE 3, 1
REM PRINT USING "Times: ######,"; S;
FOR I = 0 TO 499
    P0# = B# + (E# - B#) * (I / 500)
```

```
    A = 0
    DO WHILE P0# < 1 - P# ^ A OR P0# >= 1 - P# ^ (A + 1)
        A = A + 1
    LOOP
    P1# = P# + (1 - P#) * (P0# - (1 - P# ^ A)) / (P# ^ A - P# ^ (A + 1))
    PSET ((70 + I), (100 - P1# * 100)), 12
NEXT I
IF P# - B# > -.0000001# AND P# < E# THEN LINE (70 + (P# - B#) / C#, 0)-(70 + (P# - B#) / C#, 100), 3
LINE (70, 100 - B# * 100)-(570, 100 - (B# + C# * 500) * 100), 1
REM LOCATE 29, 10
REM PRINT "Press any key to continue";
DO
LOOP WHILE INKEY$ = ""
CLS
PRINT "P0="; P#
FOR I = 0 TO 24
    FOR J = 0 TO 9
        LOCATE I + 2, J * 8 + 1
        IF P# - (B + (J * 25 + I + 1) * C#) >= -.0000001 THEN
            PRINT "      -"
        ELSE
            IF A(I + J * 25) < 1000000 THEN
                PRINT USING "    ######"; A(I + J * 25);
            ELSE
                PRINT "E"; MID$(STR$(A(I + J * 25)), 2, 5)
            END IF
        END IF
    NEXT J
NEXT I
DO
LOOP WHILE INKEY$ = ""
FOR I = 0 TO 24
    FOR J = 10 TO 19
        LOCATE I + 2, (J - 10) * 8 + 1
        IF P# - (B + (J * 25 + I + 1) * C#) >= -.0000001 THEN
            PRINT "      -"
        ELSE
            IF A(I + J * 25) < 1000000 THEN
                PRINT USING "    ######"; A(I + J * 25);
            ELSE
                PRINT "E"; MID$(STR$(A(I + J * 25)), 2, 5)
            END IF
        END IF
    NEXT J
NEXT I
END

SUB1:
PSET ((70 + INT(D)), (100 + A(INT(D)) * R)), 2
REM PSET ((70 + INT(D)), 245 + ((P1# - B#) / C# - INT((P1# - B#) / C#)) * 200), 0
RETURN
SUB3:
LINE (70, 101)-(570, 475), 15, BF
R = R / 2
FOR X = 0 TO 499
    IF A(X) > 0 THEN LINE (70 + X, 100)-(70 + X, 100 + A(X) * R), 2
NEXT X
LOCATE 5, 1
PRINT "Ratio: 1 :"; 1 / R
RETURN
```

Chaos32.c

```
#include <stdio.h>
#include <math.h>
double p,pa;
int kn[101],k[11],upper[12],lower[12];
int want,level,krange;
long work,limit;
FILE *fout;
void test(int depth,int warning);
void cal(int depth,int warning);
void range(int ka[11]);
void correct(void);

void test(int depth,int warning)
{
    int i;
    if(depth<=level){
        for(i=kn[warning+1];i<=kn[1]+krange;i++){
        k[depth]=i;
        if(i==kn[warning+1])
            test(depth+1,warning+1);
        else
            test(depth+1,0);
        }
    }
    else
        /*if(depth>=2)*/
        cal(depth,warning);
}

void cal(int depth,int warning)
{
    int i,pass=1,words;
    double pl,pu;
    if(warning==0)
        k[depth]=want-1;
    else{
        if(want-1>=kn[warning+1])
        k[depth]=want-1;
        else
        pass=0;
    }
    if(pass==1){
        for(i=1;i<=depth;i++)
        upper[i]=lower[i]=k[i];
        upper[0]=lower[0]=depth;
        lower[depth-1]--;
        lower[depth]++;
        if(warning>0){
        for(i=1;i<=warning+1;i++){
            lower[lower[0]]=0;
            lower[0]--;
        }
        lower[lower[0]]--;
        }
        if(want<kn[2]){
        correct();
        }
        if(work==limit){
        printf("The number of lines output exceeds which you required (%ld).\nProgram terminated.\n",limit);
```

```
            exit(0);
            }
            words=7;
            fprintf(fout,"from");
            for(i=1;i<=lower[0];i++){
            fprintf(fout," %d",lower[i]);
            words+=(int)log10(lower[i])+2;
            }
            if(lower[1]==0){
            fprintf(fout," %d",0);
            lower[0]=1;
            words+=2;
            }
            fprintf(fout," to");
            for(i=1;i<=upper[0];i++){
            fprintf(fout," %d",upper[i]);
            words+=(int)log10(upper[i])+2;
            }
            for(i=1;i<=40-words;i++)
            fprintf(fout," ");
            range(lower);
            pl=pa;
            range(upper);
            pu=pa;
            fprintf(fout,"%13.10lf\n",pu-pl);
            work++;
        }
}

void range(int ka[11])
{
    int i;
    pa=1;
    for(i=ka[0];i>=1;i--)
        pa=1-pow(p,ka[i])/pa;
    if(ka[1]==0)
        pa=p;
    fprintf(fout,"%13.10f",pa);
}

void correct(void)
{
    int i,wrong;
    wrong=lower[0]-1;
    while(lower[wrong]!=kn[1]&&wrong>0){
        wrong--;
    }
    if(wrong>0){
        for(i=lower[0];i>=wrong;i--){
        lower[i]=0;
        lower[0]--;
        }
        if(i>0)
        lower[i]--;
    }
    if(want==kn[1]){
        upper[upper[0]]=0;
        upper[0]--;
        upper[upper[0]]--;
    }
```

```
}

main()
{
    int kcount,n,i;
    double p0,p1;
    fout=fopen("c:\chaos32.out","w");
    scanf("%lf%d%d%d%ld",&p,&want,&krange,&level,&limit);
    if(p<=0||p>=1)
        printf("The value p you entered is incorrect. The range of p is 0 < p < 1.\n");
    else{
        if(limit==0)
        limit=-1;
        p0=p;
        n=0;
        do{
        p1=1/(1-p0);
        kcount=0;
        do{
            p1*=p;
            kcount++;
        }while(p1>1+(1e-13));
        n++;
        kn[n]=kcount;
        p0=p1;
        }while(p0<1-(1e-13)&&n<100);
        for(i=1;i<=100;i++)
        fprintf(fout," %d",kn[i]);
        fprintf(fout,"\n");
        if(p0>1-(1e-13))
        printf("The value n you entered will arrive 1 after %d step(s).\n",n);
        else{
        if(want<kn[1])
            printf("The value k you want is incorrect, it has to be at least %d.\n",kn[1]);
        else{
            work=0;
            test(1,0);
        }
        }
    }
    printf("Lines output: %ld\n",work);
}
```

Chaos34.c

```
#include <stdio.h>
#include <math.h>
int n,kwant,a[32],k[32],upper[32],lower[32],optl[32],optu[32];
long work;
double p,pa,maxgap,ol,ou;
void cal(int depth,int warning);
void range(int ka[31]);
void correct(void);

void test(int depth,int warning){
    int i;
    if(depth<=n){
        warning++;
        a[depth]=k[warning];
        test(depth+1,warning);
        a[depth]++;
        test(depth+1,0);
    }
    else{
        /*for(i=1;i<=depth-1;i++)
        printf("%d ",a[i]);
        printf("(%d)\n",warning);*/
        cal(depth,warning);
    }
}

void cal(int depth,int warning)
{
    int i,pass=1;
    double pl,pu;
    if(warning==0)
        a[depth]=kwant-1;
    else{
        if(n-1>=k[warning+1])
        a[depth]=kwant-1;
        else
        pass=0;
    }
    if(pass==1){
        for(i=1;i<=depth;i++)
        upper[i]=lower[i]=a[i];
        upper[0]=lower[0]=depth;
        lower[depth-1]--;
        lower[depth]++;
        if(warning>0){
        for(i=1;i<=warning+1;i++){
            lower[lower[0]]=0;
            lower[0]--;
        }
        lower[lower[0]]--;
        }
        if(n<k[2]){
        correct();
        }
        range(lower);
        pl=pa;
        range(upper);
        pu=pa;
        if(pu-pl>maxgap){
```

```
            ol=pl;
            ou=pu;
            maxgap=pu-pl;
            optl[0]=lower[0];
            for(i=1;i<=optl[0];i++)
                optl[i]=lower[i];
            optu[0]=upper[0];
            for(i=1;i<=optu[0];i++)
                optu[i]=upper[i];
            }
            work++;
        }
}

void range(int ka[31])
{
    int i;
    pa=1;
    for(i=ka[0];i>=1;i--){
        pa=1-pow(p,ka[i])/pa;
    }
    if(ka[1]==0)
        pa=p;
/*    for(i=1;i<=ka[0];i++)
        printf("%d ",ka[i]);
    printf("-->%.10lf\n",pa);*/
}

void correct(void)
{
    int i,wrong;
    wrong=lower[0]-1;
    while(lower[wrong]!=k[1]&&wrong>0){
        wrong--;
    }
    if(wrong>0){
        for(i=lower[0];i>=wrong;i--){
        lower[i]=0;
        lower[0]--;
        }
        if(i>0)
        lower[i]--;
    }
    if(n==k[1]){
        upper[upper[0]]=0;
        upper[0]--;
        upper[upper[0]]--;
    }
}

main()
{
    double p0;
    int i,count;
    while(scanf("%lf%d%d",&p,&kwant,&n)==3){
        p0=p;
        count=1;
        work=0;
        maxgap=0;
        do{
```

```
            k[count]=1;
            p0=p/(1-p0);
            while(p0>1){
                p0*=p;
                k[count]++;
            }
            count++;
            }while(count<=n+2&&p0<1-1e-13);
            count--;
            for(i=1;i<=count;i++)
            printf("%d ",k[i]);
            printf("\n");
            test(1,0);
            printf("from ");
            for(i=1;i<=optl[0];i++)
            printf("%d ",optl[i]);
            if(optl[0]==0)
            printf("0 ");
            printf("to ");
            for(i=1;i<=optu[0];i++)
            printf("%d ",optu[i]);
            printf("\n");
            printf("%.10lf %.10lf %.10lf\n",ol,ou,maxgap);
            printf("Case tested: %9ld\n",work);
        }
    }
```

評 語

1. 作者平日認真作數學的成果可放在網頁上供自己及他人回顧、分享。
2. 作者有豐富的旅遊經驗，期望能在俄國參展期間之所見所聞更能提升其數學涵養