รายงานวิจัย

เรื่อง
ซอฟแวร์ในการผลิตตัวแปรสุ่ม ซึ่งมีการแจกแจงแบบต่อเนื่อง และไม่ต่อเนื่อง

โดย
อิว ไอยราคาญจนกุล
สมชัย ยืนนาน

เรื่อง ซอฟแวร์ในการผลิตตัวแปรสุ่ม ซึ่งมีการแจกแจงแบบต่อเนื่องและไม่ต่อเนื่อง

ผู้วิจัย *อิว ไลยราภาญจนกุล
*สมชัย ยืนนาน

บทคัดย่อ

การศึกษาครั้งนี้ ได้รวบรวมวิธีการผลิตตัวแปรสุ่ม แล้วนำมาเขียนโปรแกรมย่อยภาษาปาสคาล ตัวแปรสุ่มที่ผลิตมีทั้งชนิดไม่ต่อเนื่อง ได้แก่ตัวแปรสุ่มที่มีการแจกแจงแบบ ทวินาม ปัวส์ซง เรขาคณิต ทวินามนิเสธ และตัวแปรสุ่มชนิดต่อเนื่อง ได้แก่ตัวแปรสุ่มที่มีการแจกแจงแบบ เอกซ์โพเนนเชียล แกมมา (เฉพาะกรณีที่พารามิเตอร์ α เป็นเลขจำนวนเต็ม) บีตา (เฉพาะกรณีที่พารามิเตอร์เป็นเลขจำนวนเต็ม) ปกติ ลอกนอร์แมล ไวบูล ไคสแควร์ สติวเด้นที เอฟ และมัลติแวริเอตนอร์แมล

---

*อาจารย์ ภาควิชาคณิตศาสตร์ คณะวิทยาศาสตร์ มหาวิทยาลัยสงขลานครินทร์
วิทยาเขตหาดใหญ่ สงขลา

Software for Generating Discrete and Continuous Random Variates

By * Mr. Iew Ayaragarnchanakul

* Mr. Somchai Yuennan

Abstract

In this study, various methods for generating discrete and continuous random variates are collected and implemented using functions or procedures written in Pascal. For discrete distribution, random variates for Binomial, Poisson, Geometric, and Negative Binomial distributions are generated. For continuous distribution, random variates for Exponential, Gamma (for integer parameter α), Beta (for integer parameters), Normal, Lognormal, Weibul, Chi-Square, Student-t, F, Multi-variate Normal distributions will be generated.

---

* Department of Mathematics, Faculty of Science, Prince of Songkla University, Hat Yai Campus, Thailand.

สารบาญ

บทที่ 1

บทนำ

1.1. วัตถุประสงค์

งานวิจัยครั้งนี้ มีวัตถุประสงค์เพื่อจัดทำซอฟแวร์ เกี่ยวกับการผลิตตัวแปรสุ่มชนิดต่างๆ ซอฟแวร์นี้สามารถนำไปใช้ในงานวิจัยอื่นๆ หรือเป็นสื่อการเรียนการสอนในวิชาสถิติ การวิจัยดำเนินงาน การจำลอง และคอมพิวเตอร์ เป็นต้น

1.2. ขอบเขตของการวิจัย

ในกรณีของตัวแปรสุ่มแบบไม่ต่อเนื่อง จะเขียนโปรแกรมย่อยภาษาปาสคาล เพื่อผลิตตัวแปรสุ่มที่มีการแจกแจง ดังนี้

- ทวินาม(Binomial)
- ปัวส์ซง(Poisson)
- เรขาคณิต(geometric)
- ทวินามนิเสธ(Negative Binomial)

ในกรณีของตัวแปรสุ่มแบบต่อเนื่อง จะเขียนโปรแกรมย่อยภาษาปาสคาล เพื่อผลิตตัวแปรสุ่มที่มีการแจกแจง ดังนี้

- เอกซ์โพเนนเชียล (Exponential)
- แกมมา(Gamma) เฉพาะกรณีที่พารามิเตอร์ α เป็นเลขจำนวนเต็ม
- บีตา (Beta) เฉพาะกรณีที่พารามิเตอร์เป็นเลขจำนวนเต็ม
- ปกติ(Normal)
- ลอกนอร์มอล(Lognormal)
- ไวบูล(Weibul)
- ไคสแควร์(Chi-square)
- สติวเด้นที่(Student's t)
- เอฟ(F)
- มัลติแวริเอตนอร์มอล(Multi-variate Normal)

บทที่ 2

วิธีการผลิตตัวแปรสุ่ม และโปรแกรม

2.1 เลขสุ่ม(Random number) และตัวแปรสุ่มแจกแจงสม่ำเสมอแบบต่อเนื่องในช่วง [0,1]

ก่อนที่จะกล่าวถึงการผลิตตัวแปรสุ่มแบบต่างๆ จะขอกล่าวถึง การผลิตเลขสุ่ม และตัวแปรสุ่มแจกแจงสม่ำเสมอแบบต่อเนื่องในช่วง [0,1] ก่อน เพราะจำเป็นต้องนำไปใช้ ในการผลิตตัวแปรสุ่มแบบต่างๆ

การผลิตเลขสุ่มอาจจะกระทำได้ 2 วิธี[1] คือ

ก. การผลิตเลขสุ่มโดยการโปรแกรม

ข. การผลิตเลขสุ่มโดย RND (Random Number Dervice)

วิธีแรกเป็นการผลิตเลขสุ่มจากความสัมพันธ์ ที่ซ้ำซาก (recurrence relation) อนุบรรพ (sequence) ของเลขที่ผลิตในลักษณะนี้ย่อมมีคาบ แต่อย่างไรก็ตาม เลขที่ผลิตออกมานี้จะผ่านการทดสอบความเป็นเลขสุ่มเชิงสถิติก่อนจึงจะนำไปใช้ จึงเรียกเลขเหล่านี้ว่าเลขคล้ายสุ่ม (pseudo-random number) แต่เพื่อความสะดวกในการเรียก เราจะใช้คำว่า "เลขสุ่ม" แทน

ส่วนวิธีที่สอง คือการผลิตเลขสุ่มโดย RND เป็นการแปลงผลที่ได้จากกระบวนการทางกายภาคที่เป็นสุ่ม (random physical process) เช่นจำนวน particle ที่หลุดจากสารกัมมันตภาพรังสี ณ ขณะใดขณะหนึ่งเป็นต้น มาเป็นอนุบรรพของเลขสุ่ม เลขสุ่มที่ได้จากการผลิตด้วย RND จะเป็นเลขสุ่มในความหมายที่แท้จริง

ในการศึกษาครั้งนี้ จะกล่าวถึงเฉพาะเลขสุ่มแบบวิธีแรกเท่านั้น การผลิตเลขสุ่มโดยการโปรแกรมมีข้อดีหลายประการ ที่สำคัญคือ วิธีนี้สามารถผลิตอนุบรรพเลขสุ่มชุดเดิมออกมาได้ ซึ่งเป็นสิ่งจำเป็นอย่างยิ่งในกรณีที่ ทดสอบแบบจำลองและมีความประสงค์ที่จะทบทวนการคำนวณ

วิธีที่ใช้ในการผลิตเลขสุ่มคือ การใช้เศษจากการหาร (multiplicative congruential method) โดยเลขสุ่มจะผลิตจากความสัมพันธ์ $x_{i+1} \equiv kx_i \pmod{m}$ เมื่อ กำหนดค่า x_0 ,k และ m ซึ่งเป็นเลขจำนวนเต็มบวกให้ ส่วนตัวแปรสุ่มแจกแจงสม่ำเสมอแบบต่อเนื่องในช่วง [0,1] ได้จากการหารเลขสุ่มที่ผลิตจากการใช้เศษของผลหารด้วยค่า $m - 1$ และจะแทนด้วยสัญลักษณ์ U

จากวิธีการที่กล่าวมาข้างต้น นำมาเขียนโปรแกรมย่อยด้วยภาษาปาสคาล ในรูปของฟังก์ชัน เพื่อผลิตตัวเลขสุ่มที่มีการแจกแจงสม่ำเสมอแบบต่อเนื่องในช่วง [0,1] ได้ดังนี้

```
function psurandom:real
 begin {** psurandom **}
  seed:=k*seed-m*trunc((1.0*k*seed)/m);
  psurandom:=seed/(m-1.0);
end; {** psurandom **}
```

เหตุผลที่เขียนเป็นฟังก์ชัน และไม่มีการส่งพารามิเตอร์ เพราะง่ายต่อการเรียกใช้ ผู้ใช้เพียงแต่กำหนดค่าเริ่มต้นให้กับตัวแปร seed,k และ m เพียงครั้งเดียว แล้วเรียกใช้ โดยการเรียกชื่อ psurandom เท่านั้นก็จะได้ตัวเลขสุ่มที่มีการแจกแจงสม่ำเสมอแบบต่อเนื่องในช่วง [0,1] หนึ่งตัวทันที

ตัวแปร k และ m มีความหมายดังที่กล่าวมาแล้วข้างต้น ส่วนตัวแปร seed เป็นค่าของ x_i ค่าเริ่มต้นของ seed คือค่า x_0 เมื่อเรียกใช้ function psurandom ครั้งที่หนึ่ง ค่าของ seed จะเป็นค่า x_1 ,ครั้งที่สอง ค่าของ seed จะเป็นค่า x_2 เช่นนี้ไปเรื่อย ๆ ฉะนั้นผู้ใช้ที่ต้องการใช้ function psurandom จะต้องกำหนดตัวแปร ในโปรแกรมที่จะเรียกใช้ ให้มีชื่อเป็น seed,k และ m ด้วยเสมอ โดยที่ตัวแปรทั้งสามนี้จะต้องเป็นตัวแปรเดียวกันกับตัวแปรใน function psurandom ตามหลักภาษาปาสคาล

ข้อแนะนำในการเลือกค่าเริ่มต้น x_0 ,k และ m จากการสังเกตุถ้า x_0 เป็นเลขคู่ เลขหลักสุดท้ายของผลคูณ kx_i ในเลขฐานสอง จะเป็นเลขศูนย์เสมอ จึงเป็นการเสียเลขฐานสองไปหนึ่งหลักโดยไม่ได้ประโยชน์อะไร ดังนั้นเพื่อใช้ประโยชน์ของเลขฐานสองทุกหลัก x_0 จึงควรเป็นเลขคี่ ในทำนองเดียวกันหาก k เป็นเลขคู่ เลขฐานสองในหลักท้ายๆของผลคูณ kx_i จะกลายเป็นศูนย์เพิ่มขึ้นเมื่อ i มีค่าเพิ่มขึ้น จนกระทั่งเลขทั้งหมดของผลคูณ kx_i กลายเป็นศูนย์หมด เพื่อหลีกเลี่ยงปัญหานี้ k จึงควรเป็นเลขคี่ และจากทฤษฎีบทที่ว่า "ถ้า k มีค่าเท่ากับ $8t \pm 3$ โดยที่ t เป็นเลขจำนวนเต็มบวกใดๆ แล้วอนุบรรพของเลขสุ่มที่ผลิตจากความสัมพันธ์ $x_{i+1} \equiv kx_i \pmod{m}$ เมื่อ m มีค่าเท่ากับ 2^n จะมีคาบเท่ากับ 2^{n-2}"

การบวกลบคูณหารด้วยเลขจำนวนเต็ม ก็สามารถทำได้ด้วยเลขจำนวนจริงที่มีค่าเท่ากัน ดังนั้นตัวแปร seed,k และ m ควรเป็นเลขจำนวนจริง

ตัวอย่าง โปรแกรมเรียกใช้ function psurandom

```
program example1(input,output);
 var seed,k,m:real;
     n,i:integer;
 function psurandom:real;
  begin {** psurandom **}
   seed:=k*seed-m*trunc((1.0*k*seed)/m);
   psurandom:=seed/(m-1.0);
 end; {** psurandom **}
 begin {** example1 **}
  writeln('Enter seed k and m');
  readln(seed,k,m);
  writeln('Enter number of UNIFORM RANDOM NUMBER');
  readln(n);
  for i:=1 to n do
   write(psurandom);
 end.
```

ตัวอย่างโปรแกรมข้างต้น กำหนดค่าเริ่มต้นให้กับตัวแปร seed,k และ m ด้วยการอ่านค่าเข้ามา เช่นป้อนข้อมูลเข้าเป็น 5 131 32768 ค่าเริ่มต้นของตัวแปร seed,k และ m จะเป็น 5 131 และ 32768 ตามลำดับ ส่วนคำสั่ง write(psurandom) แสดงให้เห็นการเรียกใช้ function psurandom

2.2 ตัวแปรสุ่มที่มีการแจกแจงแบบต่อเนื่อง

2.2.1 ตัวแปรสุ่มที่มีการแจกแจงแบบเอกซ์โพเนนเชียล[2,7]

การแจกแจงแบบเอกซ์โพเนนเชียลมี p.d.f.

$$f_X(x) = \frac{1}{\beta} e^{-x/\beta} \quad , \quad 0 \leq x \leq \infty \ , \ \beta > 0$$

โดยวิธีผกผัน (inverse method) $U = F_X(X) = 1 - e^{-X/\beta}$ ดังนั้น $X = -\beta \ln(1-U)$

ฉะนั้นวิธีผลิตเลขสุ่มที่มีการแจกแจงแบบเอกซ์โพเนนเชียล ทำได้ตามขั้นตอนต่อไปนี้

1. ผลิตตัวเลขสุ่มที่มีการแจกแจงสม่ำเสมอแบบต่อเนื่องในช่วง [0,1] คือค่า U
2. ค่า $-\beta \ln(1-U)$ ที่คำนวณได้จะเป็นตัวเลขสุ่มที่มีการแจกแจงแบบเอกซ์โพเนนเชียล ตามต้องการ

เขียนเป็นโปรแกรมย่อยภาษาปาสคาลในรูปของฟังก์ชัน ซึ่งมีพารามิเตอร์ α ได้ดังนี้

```
function exponentialrandom(beta:real):real;
 begin {** exponentialrandom **}
  exponentialrandom:=-beta*ln(1-psurandom);
end; {** exponentialrandom **}
```

function exponentialrandom เรียกใช้ function psurandom ดังนั้น โปรแกรมที่เรียกใช้ function exponentialrandom จะต้องมี function psurandom เขียนมาก่อนด้วย เสมอตามหลักภาษาปาสคาล ในหัวข้อต่อไปก็ใช้หลักการเดียวกัน คือ ฟังก์ชัน ที่ถูกเรียกใช้ จะต้องเขียนมาก่อน ฟังก์ชัน ที่เรียกใช้เสมอ

ตัวอย่าง โปรแกรมเรียกใช้ function exponentialrandom

```
program example2(input,output);
 var seed,k,m,beta:real;
 function psurandom:real;
  begin {** psurandom **}
   seed:=k*seed-m*trunc((1.0*k*seed)/m;
   psurandom:=seed/(m-1.0);
  end; {** psurandom **}
 function exponentialrandom(beta:real):real;
  begin {** exponentialrandom **}
   exponentialrandom:=-beta*ln(1-psurandom);
  end; {** exponentialrandom **}
```

```
begin {** example2 **}
 readln(seed,k,m,beta);
 writeln(exponentialrandom(beta));
end.
```

2.2.2 ตัวแปรสุ่มที่มีการแจกแจงแบบแกมมา[2,7]

การแจกแจงแบบแกมมา มี p.d.f.

$$f_X(x) = \frac{x^{\alpha-1} e^{-x/\beta}}{\beta^{\alpha} \Gamma(\alpha)} \quad , \quad 0 \leq x \leq \infty \quad , \alpha > 0 \quad , \quad \beta > 0$$

ในที่นี้จะสนใจเฉพาะกรณีพารามิเตอร์ α เป็นเลขจำนวนเต็มเท่านั้น โดยคุณสมบัติของการแจกแจงแบบแกมมา ถ้า X_i , $i = 1,\ldots,n$ เป็นตัวแปรสุ่มที่มีการแจกแจงแบบแกมมา ซึ่งมีพารามิเตอร์ α_i และ β เป็นอิสระ (independent) ต่อกัน แล้วจะได้ว่า $X = \sum_{i=1}^{n} X_i$ จะเป็นตัวแปรสุ่ม ที่มีการแจกแจงแบบแกมมา ซึ่งมีพารามิเตอร์ α และ β โดยที่ $\alpha = \sum_{i=1}^{n} \alpha_i$

กรณีที่ α มีค่าเท่ากับ 1 การแจกแจงแบบแกมมา ก็จะมีการแจกแจงแบบเอกซ์โพเนนเชียล นั่นคือ $X = \sum_{i=1}^{\alpha}(-\beta \ln(1-U_i)) = \beta \sum_{i=1}^{\alpha}(-\ln(1-U_i))$

ขั้นวิธีสลิดเลขสุ่มที่มีการแจกแจงแบบแกมมา ซึ่งมีพารามิเตอร์ α เป็นเลขจำนวนเต็มบวก และ β ทำได้ตามขั้นตอนต่อไปนี้

1. กำหนดให้ $X = 0$
2. ทำซ้ำๆกัน จำนวน α ครั้ง ดังนี้
   2.1 ผลิตตัว**เลข**สุ่มที่มีการแจกแจงแบบเอกซ์โพเนนเชียล**ซึ่ง**มีพารามิเตอร์ $\beta = 1.0$
   2.2 นำค่าที่ได้ในข้อ 2.1 บวกเพิ่มให้กับ X
3. ค่าที่ได้จากผลคูณระหว่าง β กับ X จะเป็นตัว**เลข**สุ่มที่มีการแจกแจงแบบแกมมา ตามต้องการ

เขียนเป็นโปรแกรมย่อยภาษาปาสคาลในรูปของ ฟังก์ชัน ซึ่งมีพารามิเตอร์ α และ β ได้ดังนี้

```
function gammarandom(alpha:integer;beta:real):real;
 var x:real;
     i:integer;
 begin {** gammarandom **}
  x:=0.0;
  for i:=1 to alpha do
   x:=x+exponentialrandom(1.0);
  gammarandom:=beta*x;
 end; {** gammarandom **}
```

ตัวอย่าง โปรแกรมเรียกใช้ function gammarandom

```
program example3(input,output);
 var seed,k,m,beta:real;
     alpha:integer;
 function psurandom:real;
  begin {** psurandom **}
   seed:=k*seed-m*trunc((1.0*k*seed)/m);
   psurandom:=seed/(m-1.0);
  end; {** psurandom **}
 function exponentialrandom(beta:real):real;
  begin {** exponentialrandom **}
   exponentialrandom:=-beta*ln(1-psurandom);
  end; {** exponentialrandom **}
 function gammarandom(alpha:integer;beta:real):real;
  var x:real;
      i:integer;
  begin {** gammarandom **}
   x:=0.0;
   for i:=1 to alpha do
    x:=x+exponentialrandom(1.0);
```

```
    gammarandom:=beta*x;
  end; {** gammarandom **}
begin {** example3 **}
  readln(seed,k,m,alpha,beta);
  writeln(gammarandom(alpha,beta));
end.
```

2.2.3 ตัวแปรสุ่มที่มีการแจกแจงแบบบีตา[7]

การแจกแจงแบบบีตา มี p.d.f.

$$f_X(x) = \frac{\Gamma(\alpha+\beta)x^{\alpha-1}(1-x)^{\beta-1}}{\Gamma(\alpha)\,\Gamma(\beta)} \quad , \quad \alpha > 0 \ , \ \beta > 0 \ , \ 0 \leq x \leq 1$$

ในที่นี้จะสนใจเฉพาะกรณีพารามิเตอร์ α และ β เป็นเลขจำนวนเต็มเท่านั้น โดยทฤษฎีของ order statistics ถ้า $U_1, U_2, \ldots, U_{\alpha+\beta-1}$ เป็นตัวแปรสุ่มที่มีการแจกแจงสม่ำเสมอแบบต่อเนื่องในช่วง [0,1] แล้วจะได้ว่า U_α เป็นตัวแปรสุ่มที่มีการแจกแจงแบบบีตา

ฉะนั้นวิธีผลิตเลขสุ่มที่มีการแจกแจงแบบบีตา ซึ่งมีพารามิเตอร์ α และ β เป็นเลขจำนวนเต็มบวก ทำได้ตามขั้นตอนต่อไปนี้

1. ผลิตตัวเลขสุ่มที่มีการแจกแจงสม่ำเสมอแบบต่อเนื่องในช่วง [0,1] จำนวน $\alpha+\beta-1$ ตัว
2. เรียงลำดับตัวเลขสุ่มที่ได้ในข้อ 1. จากค่าน้อยไปมาก
3. ตัวเลขสุ่ม ณ ตำแหน่ง α ในข้อ 2. จะเป็นตัวเลขสุ่มที่มีการแจกแจงแบบบีตา ตามต้องการ

เขียนเป็นโปรแกรมย่อยภาษาปาสคาลในรูปของ ฟังก์ชัน ซึ่งมีพารามิเตอร์ α และ β ได้ดังนี้

```
function betarandom(alpha,beta:integer):real;
 const maxorder=100;
 type typearray=array[1..maxorder] of real;
 var u:typearray;
     n,i:integer;
 procedure bubblesort(var u:typearray;n:integer);
  var i,j:integer;
      flag:boolean;
      temp:real;
```

```
begin {** bubblesort **}
 i:=1;
 flag:=false;
 while (i<n) and (not flag) do
  begin
   flag:=true;
   for j:=1 to n-1 do
    if u[j] > u[j+1] then
     begin
      temp:=u[j+1];
      u[j+1]:=u[j];
      u[j]:=temp;
      flag:=false;
     end;
   i:=i+1;
  end;
end; {** bubblesort **}
begin {** betarandom **}
 n:=alpha+beta-1;
 for i:=1 to n do
  u[i]:=psurandom;
 bubblesort(u,n);
 betarandom:=u[alpha];
end; {** betarandom **}
```

function betarandom ประกอบด้วย procedure bubblesort ซึ่งใช้ในการเรียงตัวเลข โดยที่ตัวเลขที่จะนำมาเรียง จะต้องมีจำนวนไม่เกินค่า maxorder ในที่นี้กำหนดค่าไว้ 100 ถ้าค่า $\alpha+\beta-1$ มีค่ามากกว่า 100 ผู้ใช้จะต้องเปลี่ยนแปลงค่านี้ ค่าสูงสุดของ maxorder ขึ้นอยู่กับขนาดของหน่วยความจำ และขนาดของโปรแกรม ความเร็วในการเรียงตัวเลขขึ้นอยู่กับ procedure bubblesort ในที่นี้ใช้วิธีที่เรียกว่า bubble sort[5] ผู้ใช้ที่ต้องการเขียนโปรแกรม เพื่อเรียงตัวเลขเอง สามารถเปลี่ยน procedure bubblesort ใหม่ได้ตามต้องการ

ตัวอย่าง โปรแกรมเรียกใช้ function betarandom

```
program example4(input,output);
 var seed,k,m:real;
     alpha,beta:integer;
 function psurandom:real;
  begin {** psurandom **}
   seed:=k*seed-m*trunc((1.0*k*seed)/m);
   psurandom:=seed/(m-1.0);
  end; {** psurandom **}
 function betarandom(alpha,beta:integer):real;
  const maxorder=100;
  type typearray=array[1..maxorder] of real;
  var u:typearray;
      n,i:integer;
  procedure bubblesort(var u:typearray;n:integer);
   var i,j:integer;
       flag:boolean;
       temp:real;
   begin { ** bubblesort **}
    i:=1;
    flag:=false;
    while (i<n) and (not flag) do
     begin
      flag:=true;
      for j:=1 to n-1 do
       if u[j] > u[j+1] then
        begin
         temp:=u[j+1];
         u[j+1]:=u[j];
         u[j]:=temp;
```

```
          flag:=false;
         end;
       i:=i+1;
      end;
    end; {** bubblesort **}
    begin {** betarandom **}
     n:=alpha+beta-1;
     for i:=1 to n do
      u[i]:=psurandom;
     bubblesort(u,n);
     betarandom:=u[alpha];
    end; {** betarandom **}
  begin {** example4 **}
   readln(seed,k,m,alpha,beta);
   writeln(betarandom(alpha,beta));
  end.
```

2.2.4 ตัวแปรสุ่มที่มีการแจกแจงแบบปกติ[7]

การแจกแจงแบบปกติ มี p.d.f.

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad , \quad -\infty < x < \infty$$

โดยทฤษฎีของ central limit theorem ถ้า X_i , $i = 1,2,\ldots,n$ เป็นตัวแปรสุ่มที่เป็นอิสระต่อกัน ซึ่งมี $E(X_i) = \mu$ และ $var(X_i) = \sigma^2$ แล้วจะได้ว่า $Z = \frac{\sum_{i=1}^{n} X_i - n\mu}{n^{1/2}\sigma}$ จะลู่เข้า(converges) หาการแจกแจงแบบปกติ ซึ่งมี $E(Z) = 0$ และ $var(Z) = 1$

พิจารณากรณีพิเศษ เมื่อ X_i , $i = 1,2,\ldots,n$ เป็นตัวแปรสุ่มที่มีการแจกแจงสม่ำเสมอแบบต่อเนื่องในช่วง $[0,1]$

X_i จะมีค่า $\mu = \frac{1}{2}$ $\sigma = \frac{1}{\sqrt{12}}$ และ $Z = \frac{\sum_{i=1}^{n} U_i - n/2}{\sqrt{n/12}}$ กรณี $n = 12$

จะได้ $Z = \sum_{i=1}^{12} U_i - 6$ เป็นตัวแปรสุ่มประมาณที่ดี

ฉะนั้นวิธีผลิตเลขสุ่มที่มีการแจกแจงแบบปกติ ทำได้ตามขั้นตอนต่อไปนี้

1. ผลิตตัวเลขสุ่มที่มีการแจกแจงสม่ำเสมอแบบต่อเนื่องในช่วง [0,1] จำนวน 12 ตัว คือ $U_1,\ldots,U_{12}$
2. คำนวณค่า $Z = \sum_{i=1}^{12} U_i - 6$
3. ค่า $\mu + \sigma Z$ ที่คำนวณได้จะเป็นตัวเลขสุ่มที่มีการแจกแจงแบบปกติ ซึ่งมีพารามิเตอร์ μ และ σ ตามต้องการ

เขียนเป็นโปรแกรมย่อยภาษาปาสคาล ในรูปของ ฟังก์ชัน ซึ่งมีพารามิเตอร์ μ และ σ ได้ดังนี้

```
function normalrandom(mue,zigma:real):real;
 var sumu:real;
     i:integer;
 begin {** normalrandom **}
  sumu:=0.0;
  for i:= 1 to 12 do
   sumu:=sumu+psurandom;
  normalrandom:=(sumu-6)*zigma+mue;
 end; {** normalrandom **}
```

ตัวอย่าง โปรแกรมเรียกใช้ function normalrandom

```
program example5(input,output);
 var seed,k,m,mue,zigma:real;
 function psurandom:real;
  begin {** psurandom **}
   seed:=k*seed-m*trunc((1.0*k*seed)/m);
   psurandom:=seed/(m-1.0);
  end; {** psurandom **}
 function normalrandom(mue,zigma:real):real;
  var sumu:real;
      i:integer;
```

```
 begin {** normalrandom **}
  sumu:=0.0;
  for i:= 1 to 12 do
   sumu:=sumu+psurandom;
  normalrandom:=(sumu-6)*zigma+mue;
 end; {** normalrandom **}
begin {** example5 **}
 readln(seed,k,m,mue,zigma);
 writeln(normalrandom(mue,zigma));
end.
```

2.2.5 ตัวแปรสุ่มที่มีการแจกแจงแบบลอกนอร์มอล[2,7]

การแจกแจงแบบลอกนอร์มอล มี p.d.f.

$$f_X(x) = \frac{1}{\sqrt{2\pi}\,\sigma x} e^{-\frac{(\ln x-\mu)^2}{2\sigma^2}}, \quad 0 \le x \le \infty$$

จะเห็นได้ว่าถ้า Y เป็นตัวแปรสุ่มที่มีการแจกแจงแบบปกติ แล้วจะได้ว่า $X = e^Y$ เป็นตัวแปรสุ่มที่มีการแจกแจงแบบลอกนอร์มอล

ฉะนั้นวิธีผลิตเลขสุ่มที่มีการแจกแจงแบบลอกนอร์มอล ทำได้ตามขั้นตอนต่อไปนี้

1. ผลิตเลขสุ่มที่มีการแจกแจงแบบปกติ
2. นำค่าที่ได้ในข้อ 1. มาหาค่าเอกซ์โพเนน ค่าที่ได้ จะเป็นตัวเลขสุ่มที่มีการแจกแจงแบบลอกนอร์มอลตามต้องการ

เขียนเป็นโปรแกรมย่อยภาษาปาสคาลในรูปของ ฟังก์ชัน ซึ่งมีพารามิเตอร์ μ และ σ ได้ดังนี้

```
function lognormalrandom(mue,zigma:real):real;
 begin {** lognormalrandom **}
  lognormalrandom:=exp(normalrandom(mue,zigma));
 end; {** lognormalrandom **}
```

ตัวอย่าง โปรแกรมเรียกใช้ function lognormalrandom

```
program example6(input,output);
 var seed,k,m,mue,zigma:real;
 function psurandom:real;
  begin {** psurandom **}
   seed:=k*seed-m*trunc((1.0*k*seed)/m);
   psurandom:=seed/(m-1.0);
  end; {** psurandom **}
 function normalrandom(mue,zigma:real):real;
  var sumu:real;
      i:integer;
  begin {** normalrandom **}
   sumu:=0.0;
   for i:= 1 to 12 do
    sumu:=sumu+psurandom;
   normalrandom:=(sumu-6)*zigma+mue;
  end; {** normalrandom ** }
 function lognormalrandom(mue,zigma:real):real;
  begin {** lognormalrandom **}
   lognormalrandom:=exp(normalrandom(mue,zigma));
  end; {** lognormalrandom **}
 begin {** example6 **}
  readln(seed,k,m,mue,zigma);
  writeln(lognormalrandom(mue,zigma));
 end.
```

2.2.6 ตัวแปรสุ่มที่มีการแจกแจงแบบไวบูล[2,7]

การแจกแจงแบบไวบูล มี p.d.f.

$$f_X(x) = \frac{\alpha}{\beta^{\alpha}} x^{\alpha-1} e^{-(x/\beta)^{\alpha}} \quad , \quad 0 \leq x < \infty \quad , \alpha > 0 \quad , \beta > 0$$

โดยวิธีผกผัน $U = F_X(X) = 1 - e^{-(x/\beta)^{\alpha}}$ ดังนั้น $X = \beta(-\ln(1-U))^{1/\alpha}$

ฉะนั้นวิธีผลิตเลขสุ่มที่มีการแจกแจงแบบไวบูล ทำได้ตามขั้นตอนต่อไปนี้

1. ผลิตเลขสุ่มที่มีการแจกแจงสม่ำเสมอแบบต่อเนื่องในช่วง [0,1] ก็อ ค่า U
2. ค่า $\beta(-\ln(1-U))^{1/\alpha}$ ที่คำนวณได้ จะเป็นตัวเลขสุ่มที่มีการแจกแจงแบบไวบูล ตามต้องการ

เขียนเป็นโปรแกรมย่อยภาษาปาสคาลในรูปของ ฟังก์ชัน ซึ่งมีพารามิเตอร์ α และ β ได้ดังนี้

```
function weibulrandom(alpha,beta:real):real;
 var u:real;
 begin {** weibulrandom **}
  weibulrandom:=beta*exp((1-alpha)*ln(-ln(1-psurandom)));
 end; {** weibulrandom **}
```

ตัวอย่าง โปรแกรมเรียกใช้ function weibulrandom

```
program example7(input,output);
 var seed,k,m,alpha,beta:real;
 function psurandom:real;
  begin {** psurandom **}
   seed:=k*seed-m*trunc((1.0*k*seed)/m);
   psurandom:=seed/(m-1.0);
  end; {** psurandom **}
 function weibulrandom(alpha,beta:real):real;
  var u:real;
  begin {** weibulrandom **}
   weibulrandom:=beta*exp((1-alpha)*ln(-ln(1-psurandom)));
  end; {** weibulrandom **}
 begin {** example7 **}
  readln(seed,k,m,alpha,beta);
  writeln(weibulrandom(alpha,beta));
 end.
```

2.2.7 ตัวแปรสุ่มที่มีการแจกแจงแบบไคสแควร์[7]

ถ้า $Z_1,\ldots,Z_k$ เป็นตัวแปรสุ่มที่มีการแจกแจงแบบปกติ ซึ่งมี $\mu = 0$ และ $\sigma = 1$ แล้วจะได้ว่า $Y = \sum_{i=1}^{k} Z_i^2$ เป็นตัวแปรสุ่มที่มีการแจกแจงแบบไคสแควร์ ซึ่งมีองศาแห่งความอิสระ (degrees of freedom) เป็น k

ฉะนั้นวิธีผลิตเลขสุ่มที่มีการแจกแจงแบบไคสแควร์ ทำได้ตามขั้นตอนต่อไปนี้

1. ผลิตตัวเลขสุ่มที่มีการแจกแจงแบบปกติ ซึ่งมี $\mu = 0$ และ $\sigma = 1$ จำนวน k ตัว คือ $Z_1,\ldots,Z_k$
2. คำ $\sum_{i=1}^{k} Z_i^2$ ที่คำนวณได้ จะเป็นตัวเลขสุ่มที่มีการแจกแจงแบบไคสแควร์ ตามต้องการ

เขียนเป็นโปรแกรมย่อยภาษาปาสคาล ในรูปของ ฟังก์ชัน ซึ่งมีพารามิเตอร์ k ได้ดังนี้

```
function chisquarerandom(k:integer):real;
 var sumz:real;
     j:integer;
 begin {** chisquarerandom **}
  sumz:=0.0;
  for j:= 1 to k do
   sumz:=sumz+sqr(normalrandom(0.0,1.0));
  chisquarerandom:=sumz;
 end; {** chisquarerandom **}
```

ตัวอย่าง โปรแกรมเรียกใช้ function chisquarerandom

```
program example8(input,output);
 var seed,k,m:real;
     degree:integer;
 function psurandom:real;
  begin {** psurandom **}
   seed:=k*seed-m*trunc((1.0*k*seed)/m);
   psurandom:=seed/(m-1.0);
  end; {** psurandom **}
```

```
function normalrandom(mue,zigma:real):real;
 var sumu:real;
     i:integer;
 begin {** normalrandom **}
  sumu:=0.0;
  for i:= 1 to 12 do
   sumu:=sumu+psurandom;
  normalrandom:=(sumu-6)*zigma+mue;
 end; {** normalrandom **}
function chisquarerandom(k:integer):real;
 var sumz:real;
     j:integer;
 begin {** chisquarerandom **}
  sumz:=0.0;
  for j:= 1 to k do
   sumz:=sumz+sqr(normalrandom(0.0,1.0));
  chisquarerandom:=sumz;
 end; {** chisquarerandom **}
begin {** example8 **}
 readln(seed,k,m,degree);
 writeln(chisquarerandom(degree));
end.
```

2.2.8 ตัวแปรสุ่มที่มีการแจกแจงแบบสติวเด้นที[7]

ถ้า Z เป็นตัวแปรสุ่มที่มีการแจกแจงแบบปกติ ซึ่งมี $\mu = 0$ และ $\sigma = 1$ และ Y เป็นตัวแปรสุ่มที่มีการแจกแจงแบบไคล์แควร์ ซึ่งมีองศาแห่งความอิสระเท่ากับ k โดยที่ Z กับ Y เป็นอิสระต่อกันแล้วจะได้ว่า $X = \frac{Z}{\sqrt{Y/k}}$ เป็นตัวแปรสุ่มที่มีการแจกแจงแบบสติวเด้นที ซึ่งมีองศาแห่งความอิสระเป็น k

ฉะนั้นวิธีผลิตเลขสุ่มที่มีการแจกแจงแบบสติวเด้นที ทำได้ตามขั้นตอนต่อไปนี้

1. ผลิตตัวเลขสุ่มที่มีการแจกแจงแบบปกติ ซึ่งมี $\mu = 0$ และ $\sigma = 1$
2. ผลิตตัวเลขสุ่มที่มีการแจกแจงแบบไคสแควร์ ซึ่งมีองศาแห่งความอิสระเท่ากับ k
3. ค่า $\frac{Z}{\sqrt{Y/k}}$ ที่คำนวณได้ จะเป็นตัวเลขสุ่มที่มีการแจกแจงแบบสติวเด้นต์ ตามต้องการ

เขียนเป็นโปรแกรมย่อยภาษาปาสคาล ในรูปของ ฟังก์ชัน ซึ่งมีพารามิเตอร์ k ได้ดังนี้

```
function studenttrandom(k:integer):real;
 begin {** studenttrandom **}
  studenttrandom:=normalrandom(0.0,1.0)/sqrt(chisquarerandom(k)/k);
 end; {** studenttrandom **}
```

ตัวอย่าง โปรแกรมเรียกใช้ function studenttrandom

```
program example9(input,output);
 var seed,k,m:real;
     degree:integer;
 function psurandom:real;
  begin {** psurandom **}
   seed:=k*seed-m*trunc((1.0*k*seed)/m);
   psurandom:=seed/(m-1.0);
  end; {** psurandom **}
 function normalrandom(mue,zigma:real):real;
  var sumu:real;
      i:integer;
  begin {** normalrandom **}
   sumu:=0.0;
   for i:= 1 to 12 do
    sumu:=sumu+psurandom;
   normalrandom:=(sumu-6)*zigma+mue;
  end; {** normalrandom **}
```

```
function chisquarerandom(k:integer):real;
 var sumz:real;
     j:integer;
 begin {** chisquarerandom **}
  sumz:=0.0;
  for j:= 1 to k do
   sumz:=sumz+sqr(normalrandom(0.0,1.0));
  chisquarerandom:=sumz;
 end; {** chisquarerandom **}
function studenttrandom(k:integer):real;
 begin {** studenttrandom **}
  studenttrandom:=normalrandom(0.0,1.0)/sqrt(chisquarerandom(k)/k);
 end; {** studenttrandom **}
begin {** example9 **}
 readln(seed,k,m,degree);
 writeln(studenttrandom(degree));
end.
```

2.2.9 ตัวแปรสุ่มที่มีการแจกแจงแบบเอฟ[7]

ถ้า Y_1 และ Y_2 เป็นตัวแปรสุ่มที่มีการแจกแจงแบบไคสแควร์ ซึ่งมีองศาแห่งความอิสระเท่ากับ k_1 และ k_2 ตามลำดับ โดยที่ Y_1 กับ Y_2 เป็นอิสระต่อกันแล้วจะได้ว่า $X = \frac{Y_1/k_1}{Y_2/k_2}$ เป็นตัวแปรสุ่มที่มีการแจกแจงแบบเอฟ ซึ่งมีองศาแห่งความอิสระเป็น k_1 และ k_2

ฉะนั้นวิธีผลิตเลขสุ่มที่มีการแจกแจงแบบเอฟ ทำได้ตามขั้นตอนต่อไปนี้

1. ผลิตตัวเลขสุ่มที่มีการแจกแจงแบบไคสแควร์ ซึ่งมีองศาแห่งความอิสระเท่ากับ k_1 คือ Y_1
2. ผลิตตัวเลขสุ่มที่มีการแจกแจงแบบไคสแควร์ ซึ่งมีองศาแห่งความอิสระเท่ากับ k_2 คือ Y_2
3. ค่า $\frac{Y_1/k_1}{Y_2/k_2}$ ที่คำนวณได้ จะเป็นตัวเลขสุ่มที่มีการแจกแจงแบบเอฟ ตามต้องการ

เขียนเป็นโปรแกรมย่อยภาษาปาสคาล ในรูปของ ฟังก์ชัน ซึ่งมีพารามิเตอร์ k_1 และ k_2 ได้ดังนี้

```
function frandom(k1,k2:integer):real;
 begin {** frandom **}
  frandom:=(chisquarerandom(k1)/k1)/(chisquarerandom(k2)/k2);
 end; {** frandom **}
```

ตัวอย่าง โปรแกรมเรียกใช้ function frandom

```
program example10(input,output);
 var seed,k,m:real;
     degree1,degree2:integer;
 function psurandom:real;
  begin {** psurandom **}
   seed:=k*seed-m*trunc((1.0*k*seed)/m);
   psurandom:=seed/(m-1.0);
  end; {** psurandom **}
 function normalrandom(mue,zigma:real):real;
  var sumu:real;
      i:integer;
  begin {** normalrandom **}
   sumu:=0.0;
   for i:= 1 to 12 do
    sumu:=sumu+psurandom;
   normalrandom:=(sumu-6)*zigma+mue;
  end; {** normalrandom **}
 function chisquarerandom(k:integer):real;
  var sumz:real;
      j:integer;
  begin {** chisquarerandom **}
   sumz:=0.0;
   for j:= 1 to k do
    sumz:=sumz+sqr(normalrandom(0.0,1.0));
```

```
  chisquarerandom:=sumz;
 end; {** chisquarerandom **}
function frandom(k1,k2:integer):real;
 begin {** frandom **}
  frandom:=(chisquarerandom(k1)/k1)/(chisquarerandom(k2)/k2);
 end; {** frandom **}
begin {** example10 **}
 readln(seed,k,m,degree1,degree2);
 writeln(frandom(degree1,degree2));
end.
```

2.2.10 ตัวแปรสุ่มที่มีการแจกแจงแบบมัลติเวริเอตนอร์มอล (7)

เวกเตอร์สุ่ม $X = (X_1, \ldots, X_n)$ มีการแจกแจงแบบมัลติเวริเอตนอร์มอล ถ้า p.d.f. คือ

$$f_X(x) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} e^{\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right)}$$

เมื่อ $\mu = (\mu_1, \ldots, \mu_n)$ และ

$$\Sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1n} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2n} \\ \vdots & & & \\ \sigma_{n1} & \sigma_{n2} & \cdots & \sigma_{nn} \end{bmatrix}$$

เนื่องจาก Σ เป็นเมตริกซ์บวกแน่นอน (positive definite matrix) และสมมาตร (symmetric matrix) ดังนั้นจะมีเมตริกซ์สามเหลี่ยมเฉียงล่าง (lower triagular matrix)

$$C = \begin{bmatrix} c_{11} & 0 & \cdots & 0 \\ c_{21} & c_{22} & \cdots & 0 \\ \vdots & & & \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{bmatrix}$$

เพียงเมตริกซ์เดียวเท่านั้น (unique) ที่มีคุณสมบัติว่า $\Sigma = CC^T$ และสามารถแทนเวกเตอร์สุ่ม X ได้ด้วย $X = CZ+\mu$ เมื่อ $Z = (Z_1,\ldots,Z_n)$ เป็นเวกเตอร์สุ่มที่มีการแจกแจงแบบมัลติแวริเอตนอร์มัล ซึ่งมี $\mu = (0,\ldots,0)$ และ

$$\Sigma = \begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & 1 & \ldots & 0 \\ \vdots & & & \\ 0 & 0 & \ldots & 1 \end{bmatrix}$$

วิธีหาเมตริกซ์ C ทำได้โดยวิธี Cholesky factorisation[6] ซึ่งมีขั้นตอนดังนี้

1. ก) คำนวณค่า $C_{11} = \sqrt{\sigma_{11}}$

   ข) คำนวณค่า $C_{j1} = \frac{\sigma_{j1}}{C_{11}}$ เมื่อ $j = 2,3,\ldots,n$

2. ก) คำนวณค่า $C_{ii} = \sqrt{\sigma_{ii} - \sum_{k=1}^{i=1} C_{ik}^2}$ เมื่อ $i = 2,3,\ldots,n$

   ข) คำนวณค่า $C_{ji} = (\sigma_{ji} - \sum_{k=1}^{i-1} C_{jk}C_{ik})/C_{ii}$ เมื่อ $i = 2,3,\ldots,n$ และ $j = i+1,i+2,\ldots,n$

ฉะนั้นวิธีผลิตเลขสุ่มที่มีการแจกแจงแบบมัลติแวริเอตนอร์มัล ทำได้ตามขั้นตอนต่อไปนี้

1. คำนวณเมตริกซ์ C
2. ผลิตเวกเตอร์สุ่ม $Z = (Z_1,\ldots,Z_n)$ โดยที่ Z_i มีการแจกแจงแบบปกติ ซึ่งมี $\mu = 0$ และ $\sigma = 1$
3. ค่า $CZ+\mu$ ที่คำนวณได้จะเป็นเวกเตอร์สุ่มที่มีการแจกแจงแบบมัลติแวริเอตนอร์มัลตามต้องการ

เขียนเป็นโปรแกรมย่อยภาษาปาสคาลได้ดังนี้

```
procedure findc(n:integer;covariance:matrix;var c:matrix;
                var error:integer);
 label 99;
 var i,j,k:integer;
     sum:real;
 begin {** findc **}
```

```
error:=0;
{ Find C such that CCT = covariance }
if covariance[1,1] <= 0 then
 begin
  error:=-2;
  goto 99;
 end;
c[1,1]:=sqrt(covariance[1,1]);
for j:= 2 to n do
 c[j,1]:=covariance[j,1]/c[1,1];
for i:= 2 to n do
 begin
  sum:=covariance[i,i];
  for k:= 1 to i-1 do
   sum:=sum-c[i,k]*c[i,k];
  if sum <= 0.0 then
   begin
    error:=-3;
    goto 99;
   end;
  c[i,i]:=sqrt(sum);
  for j:= i+1 to n do
   begin
    sum:=covariance[j,i];
    for k:= 1 to i-1 do
     sum:=sum-c[j,k]*c[i,k];
    c[j,i]:=sum/c[i,i];
   end;
 end;
99:
end; {** findc **}
```

```
procedure multivariatenormalrandom(n:integer;c:matrix;mue:vector;
                                   var x:vector);
var z:vector;
    i,j:integer;
begin {** multivariatenormalrandom **}
 { Generate Z = (Z1,...,Zn) from N(0.0,1.0) }
 for i: 1 to n do
  z[i]:=normalrandom(0.0,1.0);
 { Find X = CZ+mue }
 for i:= 1 to n do
  begin
   x[i]:=mue[i];
   for j:= 1 to i do
    x[i]:=x[i]+c[i,j]*z[j];
  end;
end; {** multivariatenormalrandom **}
```

เหตุผลที่แยกเขียน procedure findc ออกต่างหากจาก procedure multivariatenormalrandom และไม่มีการคำนวณเมตริกซ์ C ใน procedure multivariatenormalrandom เพราะการคำนวณเมตริกซ์ C สามารถทำได้เพียงครั้งเดียว ถ้ามีการคำนวณเมตริกซ์ C ใน procedure multivariatenormalrandom จะต้องคำนวณทุกครั้ง ที่มีการเรียกใช้ ทำให้เสียเวลา ฉะนั้นโปรแกรมที่จะเรียกใช้ procedure multivariatenormalrandom จะต้องมีการเรียกใช้ procedure findc มาก่อนหนึ่งครั้งเสมอ

พารามิเตอร์ของ procedure findc มีดังนี้ n,covariance,c และ error หมายถึงค่า n, เมตริกซ์ Σ , เมตริกซ์ C และ ค่าผิดพลาด ตามลำดับ ค่าผิดพลาดในกรณีนี้มีไว้เพื่อตรวจสอบว่าเมตริกซ์ Σ เป็นเมตริกซ์บวกหนาแน่นและสมมาตรหรือไม่ กรณีที่ไม่เป็นจะไม่สามารถคำนวณเมตริกซ์ C และจะกำหนดค่า error ให้มีค่าน้อยกว่า 0 แต่กรณีที่เป็น ค่า error จะมีค่าเป็น 0 ดังนั้นผู้ใช้สามารถตรวจสอบจากค่า error ดังกล่าวได้

ส่วนพารามิเตอร์ของ procedure multivariatenormalrandom มีดังนี้ n,c,mue และ x หมายถึง ค่า n, เมตริกซ์ C , เวกเตอร์ μ และเวกเตอร์สุ่ม X ตามลำดับ

พารามิเตอร์ข้างต้น มี type ที่ผู้ใช้จำเป็นต้องกำหนดในโปรแกรมที่เรียกใช้ procedure ทั้งสอง ดังนี้

```
type matrix=array[1..n,1..n] of real;
     vector=array[1..n] of real;
```

และจะต้องกำหนดค่าคงที่ n ด้วยเสมอ เช่น n = 3 ให้กำหนด ดังนี้ const n=3; เป็นต้น

ตัวอย่าง โปรแกรมเรียกใช้ procedure multivariatenormalrandom

```
program example11(input,output);
 const n=2;
 type matrix=array[1..n,1..n] of real;
 var seed,k,m:real;
     i,j,error:integer;
     c,covariance:matrix;
     mue,x:vector;
 procedure findc(n:integer;covariance:matrix;var c:matrix;
                 var error:integer);
  label 99;
  var i,j,k:integer;
      sum:real;
  begin {** findc **}
   error:=0;
   { Find C such that CCT = covariance }
   if covariance[1,1] <= 0 then
    begin
     error:=-2;
     goto 99;
    end;
   c[1,1]:=sqrt(covariance[1,1]);
   for j:= 2 to n do
    c[j,1]:=covariance[j,1]/c[1,1];
   for i:= 2 to n do
```

```
  begin
   sum:=covariance[i,i];
   for k:= 1 to i-1 do
    sum:=sum-c[i,k]*c[i,k];
   if sum <= 0.0 then
    begin
     error:=-3;
     goto 99;
    end;
   c[i,i]:=sqrt(sum);
   for j:= i+1 to n do
    begin
     sum:=covariance[j,i];
     for k:= 1 to i-1 do
      sum:=sum-c[j,k]*c[i,k];
     c[j,i]:=sum/c[i,i];
    end;
  end;
99:
end; {** findc **}
procedure multivariatenormalrandom(n:integer;c:matrix;mue:vector;
                                   var x:vector);
var z:vector;
    i,j:integer;
begin {** multivariatenormalrandom **}
 { Generate Z = (Z1,...,Zn) from N(0.0,1.0) }
 for i: 1 to n do
  z[i]:=normalrandom(0.0,1.0);
 { Find X = CZ+mue }
 for i:= 1 to n do
  begin
```

```
      x[i]:=mue[i];
      for j:= 1 to i do
       x[i]:=x[i]+c[i,j]*z[j];
     end;
  end; {** multivariatenormalrandom **}
begin {** example11 **}
 readln(seed,k,m);
 for i:= 1 to n do
  readln(mue[i]);
 for i:= 1 to n do
  for j:= 1 to n do
   readln(covariance[i,j]);
 findc(n,covariance,c,error);
 if error = 0 then
  begin
   for i:= 1 to 100 do
    begin
     multivariatenormalrandom(n,c,mue,x);
     writeln(' ');
     for j:= 1 to n do
      write(x[j]);
    end;
  end
 else
  writeln('Error cannot find C');
end.
```

โปรแกรมตัวอย่างข้างต้น เป็นการหาตัวเลขสุ่มที่มีการแจกแจงแบบไบแวริเอตนอร์มอล (Bivariate Normal) จำนวน 100 ตัว

2.3 ตัวแปรสุ่มที่มีการแจกแจงแบบไม่ต่อเนื่อง

2.3.1 ตัวแปรสุ่มที่มีการแจกแจงแบบทวินาม

ตัวแปรสุ่ม X มีการแจกแจงแบบทวินาม

$$Pr(X=x) = P_x = \binom{n}{x} p^x(1-p)^{n-x},\ 0 < p < 1,\ x = 0,1,\ldots,n,\ n>0$$

วิธีง่ายๆ สามารถทำได้โดยการจำลองการทดลอง n ครั้ง ด้วยการผลิตเลขสุ่มที่มีการแจกแจงสม่ำเสมอแบบต่อเนื่องในช่วง [0,1] จำนวน n ตัว แล้วนับจำนวนตัวเลขสุ่มดังกล่าวที่มีค่าน้อยกว่า p ตัวเลขที่นับได้จะเป็นตัวเลขสุ่มที่มีการแจกแจงแบบทวินาม ตามต้องการ

ฉะนั้นวิธีผลิตตัวเลขสุ่ม ที่มีการแจกแจงแบบทวินาม ทำได้ตามขั้นตอนต่อไปนี้

1. ผลิตตัวเลขสุ่มที่มีการแจกแจงสม่ำเสมอแบบต่อเนื่องในช่วง [0,1] จำนวน n ตัว
2. นับจำนวนตัวเลขสุ่มในข้อ 1. ที่มีค่าน้อยกว่า p ตัวเลขที่นับได้ จะเป็นตัวเลขสุ่มที่มีการแจกแจงแบบทวินาม ตามต้องการ

เขียนเป็นโปรแกรมย่อยภาษาปาสคาลในรูปของ ฟังก์ชัน ซึ่งมีพารามิเตอร์ n และ p ได้ดังนี้

```
function binomialrandom(n:integer;p:real):real;
 var i,k:integer;
 begin {** binomialrandom **}
  k:=0;
  for i:= 1 to n do
   if psurandom < p then
    k:=k+1;
  binomialrandom:=k;
 end; {** binomialrandom **}
```

ตัวอย่าง โปรแกรมเรียกใช้ function binomialrandom

```
program example12(input,output);
 var seed,k,m,p:real;
     n:integer;
```

```
function psurandom:real;
 begin {** psurandom **}
  seed:=k*seed-m*trunc((1.0*k*seed)/m);
  psurandom:=seed/(m-1.0);
 end; {** psurandom **}
function binomialrandom(n:integer;p:real):real;
 var i,k:integer;
 begin {** binomialrandom **}
  k:=0;
  for i:= 1 to n do
   if psurandom < p then
    k:=k+1;
  binomialrandom:=k;
 end; {** binomialrandom **}
begin {** example11 **}
 readln(seed,k,m,n,p);
 writeln(binomialrandom(n,p));
end.
```

2.3.2 ตัวแปรสุ่มที่มีการแจกแจงแบบปัวส์ซง[7]

ตัวแปรสุ่ม X มีการแจกแจงแบบปัวส์ซง

$$Pr(X=x) = P_x = \frac{\lambda^x e^{-\lambda}}{x!} \quad , \quad \lambda > 0 \quad , \quad k = 0,1,2,...$$

เป็นที่รู้กันดีแล้วว่า ถ้าการแจกแจงความน่าจะเป็นของเวลาระหว่างการเกิดขึ้นของเหตุการณ์สองเหตุการณ์ที่อยู่ติดกัน (interarrival time) มีการแจกแจงแบบเอกซ์โพเนนเชียล ซึ่งมีพารามิเตอร์ 1/λ แล้วจะได้ว่า จำนวนครั้งของการเกิดขึ้นของเหตุการณ์สุ่มอย่างหนึ่งในหนึ่งหน่วยเวลาดังกล่าว จะมีการแจกแจงแบบปัวส์ซง ซึ่งมีพารามิเตอร์ λ เขียนเป็นอสมการทางคณิตศาสตร์ได้ ดังนี้

$$\sum_{i=0}^{x} T_i \leq 1 \leq \sum_{i=0}^{x+1} T_i$$

เมื่อ $T_i = -\frac{1}{\lambda} \ln U_i$ นั่นคือ $-\sum_{i=0}^{x} \ln U_i \leq \lambda \leq -\sum_{i=0}^{x+1} \ln U_i$, $x = 0,1,...$

หรือ $\prod_{i=0}^{x} U_i \geq e^{-\lambda} \geq \prod_{i=0}^{x+1} U_i$, $x = 0,1,...$

โดยที่ U_i เป็นตัวแปรสุ่มที่มีการแจกแจงสม่ำเสมอแบบต่อเนื่องในช่วง [0,1]

ฉะนั้นวิธีผลิตเลขสุ่ม ที่มีการแจกแจงแบบปัวส์ซง ทำได้ตามขั้นตอน ต่อไปนี้

1. กำหนดให้ $A = 1$
2. กำหนดให้ $k = 0$
3. ผลิตตัวเลขสุ่ม ที่มีการแจกแจงแบบต่อเนื่องในช่วง [0,1] คือ U_k
4. คำนวณค่า $A = U_k A$
5. ถ้า $A < e^{-\lambda}$ จริงแล้วจะได้ว่า k เป็นเลขสุ่มที่มีการแจกแจงแบบปัวส์ซง ตามต้องการ หยุดทำ มิฉะนั้น ให้ทำข้อ 6.
6. คำนวณค่า $k = k + 1$
7. กลับไปทำข้อ 3.

เขียนเป็นโปรแกรมย่อยภาษาปาสคาลในรูปของ ฟังก์ชัน ซึ่งมีพารามิเตอร์ λ ได้ดังนี้

```
function poissonrandom(lemda:real):integer;
 var k:integer;
     a,elemda:real;
 begin {** poissonrandom **}
  a:=1.0;
  k:=0;
  elemda:=exp(-lemda);
  repeat
   a:=psurandom*a;
    if a >= elemda then
     k:=k+1;
  until a < elemda;
 end; {** poissonrandom **}
```

ตัวอย่าง โปรแกรมเรียกใช้ function poissonrandom

```
program example13(input,output);
 var seed,k,m,lemda:real;
 function psurandom:real;
  begin {** psurandom **}
   seed:=k*seed-m*trunc((1.0*k*seed)/m);
   psurandom:=seed/(m-1.0);
  end; {** psurandom **}
 function poissonrandom(lemda:real):integer;
  var k:integer;
      a,elemda:real;
  begin {** poissonrandom **}
   a:=1.0;
   k:=0;
   elemda:=exp(-lemda);
   repeat
    a:=psurandom*a;
    if a >= elemda then
     k:=k+1;
   until a < elemda;
  end; {** poissonrandom **}
 begin {** example13 **}
  readln(seed,k,m,lemda);
  writeln(poissonrandom(lemda));
 end.
```

2.3.3 ตัวแปรสุ่มที่มีการแจกแจงแบบเรขาคณิต[7]

ตัวแปรสุ่ม X มีการแจกแจงแบบเรขาคณิต

$$Pr(X=x) = P_x = p(1-p)^x \quad , 0 < p < 1 \; ; \; x = 0,1,\ldots$$

เนื่องจาก $g_k = Pr(X < k) = \sum_{i=0}^{k} P_i$ จะได้ว่า $g_{k+1} = g_k + P_{k+1}$ โดยวิธีผกผัน

$$Pr(g_{k-1} < U < g_k) = \int_{g_{k-1}}^{g_k} du = g_k - g_{k-1} = P_k \quad \text{เมื่อ} \quad g_{-1} = 0$$

โดยที่ U มีการแจกแจงสม่ำเสมอแบบต่อเนื่องในช่วง [0,1] นั่นคือ $X = \min \{x: g_{k-1} < U < g_k\}$

กรณีการแจกแจงแบบเรขาคณิต จะได้ว่า $P_0 = p$ และ $P_{k+1} = (1-p)P_k$

ฉะนั้นวิธีผลิตตัวเลขสุ่มที่มีการแจกแจงแบบเรขาคณิต ทำได้ตามขั้นตอนต่อไปนี้

1. กำหนดให้ C = p (นั่นคือ $C = P_0$)
2. กำหนดให้ B = C (นั่นคือ $B = g_0 = P_0$)
3. กำหนดให้ k = 0
4. ผลิตตัวเลขสุ่ม ที่มีการแจกแจงสม่ำเสมอแบบต่อเนื่องในช่วง [0,1] คือ U
5. ถ้า U < B (นั่นคือ $U < g_k$) จริง จะได้ว่า k เป็นตัวเลขสุ่มที่มีการแจกแจงแบบเรขาคณิต ตามต้องการ หยุดทำ มิฉะนั้น ให้ทำข้อ 6.
6. คำนวณค่า k = k + 1
7. คำนวณค่า C = (1-p)C (นั่นคือ $P_{k+1} = (1-p)P_k$)
8. คำนวณค่า B = B + C (นั่นคือ $g_{k+1} = g_k + P_{k+1}$)
9. กลับไปทำข้อ 5.

เขียนเป็นโปรแกรมย่อยภาษาปาสคาลในรูปของ ฟังก์ชัน ซึ่งมีพารามิเตอร์ p ได้ดังนี้

```
function geometricrandom(p:real):integer;
 var k:integer;
     b,c,u:real;
 begin {** geometricrandom **}
  c:=p;
  b:=c;
  k:=0;
  u:=psurandom;
  while u > b do
   begin
    k:=k+1;
    c:=(1-p)*c;
```

```
    b:=b+c;
   end;
 end; {** geometricrandom **}
```

ตัวอย่าง โปรแกรมเรียกใช้ function geometricrandom

```
program example14(input,output);
 var seed,k,m,p:real;
 function psurandom:real;
  begin {** psurandom **}
   seed:=k*seed-m*trunc((1.0*k*seed)/m);
   psurandom:=seed/(m-1.0);
  end; {** psurandom **}
 function geometricrandom(p:real):integer;
  var k:integer;
      b,c,u:real;
  begin {** geometricrandom **}
   c:=p;
   b:=c;
   k:=0;
   u:=psurandom;
   while u > b do
    begin
     k:=k+1;
     c:=(1-p)*c;
     b:=b+c;
    end;
  end; {** geometricrandom **}
begin {** example14 **}
 readln(seed,k,m,p);
 writeln(geometricrandom(p));
end.
```

2.3.4 ตัวแปรสุ่มที่มีการแจกแจงแบบทวินามนิเสธ[7]

ตัวแปรสุ่ม X มีการแจกแจงแบบทวินามนิเสธ

$$Pr(X=x) = P_x = \binom{x+r-1}{x} p^r (1-p)^x \quad , \quad 0 < p < 1 \quad , \quad x = 0,1,2,\ldots$$

โดยคุณสมบัติ ที่เรียกว่า reproductive property ถ้า X_i , $i=1,\ldots,r$ เป็นตัวแปรสุ่มที่มีการแจกแจงแบบเรขาคณิต ซึ่งมีพารามิเตอร์ p แล้วจะได้ว่า $X = \sum_{i=1}^{r} X_i$ จะเป็นตัวแปรสุ่ม ที่มีการแจกแจงแบบทวินามนิเสธ ซึ่งมีพารามิเตอร์ r และ p

ฉะนั้นวิธีผลิตเลขสุ่มที่มีการแจกแจงแบบทวินามนิเสธ ทำได้ตามขั้นตอนต่อไปนี้

1. ผลิตตัวเลขสุ่ม ที่มีการแจกแจงแบบเรขาคณิต จำนวน r ตัว
2. นำตัวเลขสุ่มทั้งหมดในข้อ 1. มารวมกัน ผลลัพธ์ที่ได้ จะเป็นตัวเลขสุ่มที่มีการแจกแจงแบบทวินามนิเสธ ตามต้องการ

นำมาเขียนเป็นโปรแกรมย่อยภาษาปาสคาล ในรูปของ ฟังก์ชัน ซึ่งมีพารามิเตอร์ r และ p ได้ดังนี้

```
function negativebinomialrandom(r:integer;p:real):integer;
 var i,x:integer;
 begin {** negativebinomialrandom **}
  x:=0;
  for i:= 1 to r do
   x:=x+geometricrandom(p);
  negativebinomialrandom:=x;
 end; {** negativebinomialrandom **}
```

ตัวอย่าง โปรแกรมเรียกใช้ function negativebinomialrandom

```
program example15(input,output);
 var seed,k,m,p:real;
     r:integer;
 function psurandom:real;
  begin {** psurandom **}
```

```
  seed:=k*seed-m*trunc((1.0*k*seed)/m);
  psurandom:=seed/(m-1.0);
 end; {** psurandom **}
function geometricrandom(p:real):integer;
 var k:integer;
     b,c,u:real;
 begin {** geometricrandom **}
  c:=p;
  b:=c;
  k:=0;
  u:=psurandom;
  while u > b do
   begin
    k:=k+1;
    c:=(1-p)*c;
    b:=b+c;
   end;
 end; {** geometricrandom **}
function negativebinomialrandom(r:integer;p:real):integer;
 var i,x:integer;
 begin {** negativebinomialrandom **}
  x:=0;
  for i:= 1 to r do
   x:=x+geometricrandom(p);
  negativebinomialrandom:=x;
 end; {** negativebinomialrandom **}
begin {** example15 **}
 readln(seed,k,m,r,p);
 writeln(negativebinomialrandom(r,p));
end.
```

บทที่ 3

สรุป

วิธีการผลิตตัวแปรสุ่มแบบต่างๆในบทที่ 2 เป็นวิธีการที่ไม่สลับซับซ้อน จึงเหมาะอย่างยิ่งที่จะใช้ในการศึกษา โดยเฉพาะอย่างยิ่งจะเห็นได้ว่า การผลิตตัวแปรสุ่มที่มีการแจกแจงสม่ำเสมอแบบต่อเนื่องในช่วง [0,1] นั้นเป็นวิธีที่ง่าย และเป็นพื้นฐานที่ใช้ในการเรียนการสอนมาช้านานแล้ว ผู้สอนอาจจะเขียนโปรแกรมเพื่อเรียกใช้ function psurandom แล้วทดลองเปลี่ยนค่า seed, k และ m ดู เพื่อแสดงให้ผู้เรียนเห็นว่าตัวเลขสุ่มที่ผลิตได้ มีคาบของอนุกรรมตามทฤษฎีจริง เป็นต้น

วิธีการผลิตตัวแปรสุ่มในการศึกษาครั้งนี้ ได้ทดสอบความเป็นสุ่ม ของตัวแปรสุ่มที่มีการแจกแจงสม่ำเสมอแบบต่อเนื่องในช่วง [0,1] โดยวิธี runs tests[3] และทดสอบตัวแปรสุ่มว่ามีการแจกแจงแบบต่างๆ ด้วยการทดสอบสารูปสนิทดี (goodness of fit tests)[3] ปรากฏว่ามีคุณภาพเชิงสุ่มตามหลักเกณฑ์

บทที่ 4

เอกสารอ้างอิง

1. วิชิต หล่อจีระชุณห์กุล,2524, เอกสารประกอบการสอนรายวิชา สป. 644 การจำลอง, สถาบันบัณฑิตพัฒนบริหารศาสตร์, กรุงเทพมหานคร
2. George S. Fishman, "Principles of Discrete Event Simulation", John Wiley & Sons, Inc., 1978
3. Jerry Banks and John S. Carson, II, "Discrete-Event System Simulation", Prentice-Hall, Inc., 1984
4. Jim Welsh and John Elder, "Introduction to Pascal", Prentice-Hall, Inc., 1979
5. Knuth,D.E., "The Art of Computer Programming, vol. 3, Searching and Sorting", Addison-Wesley Publishing Company, Reading, Mass., 1973
6. Michael E. Fisher, "Introductory Numerical Methods for Scientists and Engineers", Department of Mathematics, The University of Western Australi
7. Reuven Y. Rubinstein, "Simulation and the Monte Carlo Method", John Wiley & Sons, Inc., 1981

ภาคผนวก

ตัวอย่างที่ 1 ผลการทดสอบการแจกแจงของตัวแปรสุ่มสม่ำเสมอแบบต่อเนื่องในช่วง [0, 1]

x_0 = 5 k = 131 m = 32768

จำนวนตัวเลขสุ่มที่ผลิต 30 ตัว ดังนี้

0.01999	0.61858	0.03146	0.12174	0.94726	0.08780
0.50169	0.71990	0.30418	0.84613	0.83911	0.91961
0.46580	0.01846	0.41868	0.84588	0.80712	0.72991
0.61547	0.62371	0.70312	0.10538	0.80444	0.37809
0.52879	0.26994	0.36058	0.23405	0.65917	0.34855

โดยวิธีการทดสอบสารูปสนิทดี

ชั้นที่	ขีดจำกัดล่าง	ขีดจำกัดบน	ความถี่ค่าสังเกตุ	ความถี่ค่าคาดหวัง
1	0.00000	0.20000	6.00000	6.00000
2	0.20000	0.40000	6.00000	6.00000
3	0.40000	0.60000	4.00000	6.00000
4	0.60000	0.80000	7.00000	6.00000
5	0.80000	1.00000	7.00000	6.00000

x^2 = 1.000 องศาแห่งความอิสระ = 4

ค่า P = 0.91

ตัวอย่างที่ 2 ผลการทดสอบความเป็นสุ่ม ของตัวแปรสุ่มสม่ำเสมอแบบต่อเนื่องในช่วง [0,1]

x_0 = 5 k = 131 m = 32768

จำนวนตัวเลขสุ่มที่ผลิต 30 ตัว ดังตัวอย่างที่ 1 โดยวิธี runs tests

RUNS UP AND RUNS DOWN

+-++-++-+-+--++---++-+-+-+-+-

NUMBER OF RUNS = 22

μ = 19.667 σ = 2.238 Z0 = 1.042

ค่า P = 0.15

RUN LENGTH	Oi		Ei	
1	16		12.58333	
2	5		5.26667	
3	1	6	1.45278	7.08334
>3	0		0.36389	

χ^2= 1.093 องศาแห่งความอิสระ = 1

ค่า P = 0.30

RUNS ABOVE AND BELOW THE MEAN

-+--+-++-+++---++++++-+-+---+-

n1 = 16

n2 = 14

NUMBER OF RUNS = 17

μ = 15.433 σ = 2.678 Z0 = 0.585

ค่า P = 0.28

RUN LENGTH	Oi		Ei	
1	11		7.40059	
2	2		3.70029	
3	3	6	1.85837	8.03274
>3	1		2.47408	

χ^2= 2.265 องศาแห่งความอิสระ = 1

ค่า P = 0.13

ตัวอย่างที่ 3 ผลการทดสอบการแจกแจงของตัวแปรสุ่มแบบเอกซ์โพเนนเชียล

x_0 = 5 k = 131 m = 32768 β = 1.5

จำนวนตัวเลขสุ่มที่ผลิต 30 ตัว ดังนี้

0.03029	1.44578	0.04796	0.19472	4.41369	0.13785
1.04481	1.90892	0.54399	2.80743	2.74052	3.78138
0.94049	0.02795	0.81369	2.80505	2.46855	1.96351
1.43359	1.46608	1.82162	0.16704	2.44781	0.71245
1.12869	0.47194	0.67078	0.39995	1.61455	0.64284

โดยวิธีการทดสอบสารูปสนิทดี

ชั้นที่	ขีดจำกัดล่าง	ขีดจำกัดบน	ความถี่ค่าสังเกตุ	ความถี่ค่าคาดหวัง
1	0.00000	0.50000	8.00000	8.50406
2	0.50000	1.50000	11.00000	10.45956
3	1.50000	2.50000	6.00000	5.37012
4	>2.50000		5.00000	5.66627

χ^2= 0.210 องศาแห่งความอิสระ = 3

ค่า P = 0.98

ตัวอย่างที่ 4 ผลการทดสอบการแจกแจงของตัวแปรสุ่มแบบแกมมา

x_0 = 5 k = 131 m = 32768 α = 4 β = 1.3

จำนวนตัวเลขสุ่มที่ผลิต 30 ตัว ดังนี้

1.48957	6.50456	8.55687	3.97556	6.35416	4.46239
2.31518	3.84957	3.54576	8.96296	3.87315	4.78600
3.88369	3.54323	9.36044	6.30687	6.89135	7.84221
4.89147	3.13231	6.65363	3.84020	2.61372	4.90358
6.36485	3.12708	4.21443	7.94492	6.76315	9.67962

โดยวิธีการทดสอบสารูปสนิทดี

ชั้นที่	ขีดจำกัดล่าง	ขีดจำกัดบน	ความถี่ค่าสังเกตุ	ความถี่ค่าคาดหวัง
1	0.00000	3.15000	5.00000	6.78343
2	3.15000	4.75000	9.00000	8.10174
3	4.75000	6.50000	6.00000	7.16406
4	>6.50000		10.00000	7.95078

χ^2= 1.286 องศาแห่งความอิสระ = 3

ค่า P = 0.73

ตัวอย่างที่ 5 ผลการทดสอบการแจกแจงของตัวแปรสุ่มแบบบีตา

x_0 = 5 k = 131 m = 32768 α = 2 β = 5

จำนวนตัวเลขสุ่มที่ผลิต 30 ตัว ดังนี้

0.03146	0.50169	0.41868	0.37809	0.26994	0.31657
0.33396	0.30363	0.01553	0.03488	0.41844	0.22068
0.17502	0.47868	0.21525	0.19651	0.56829	0.09299
0.37394	0.45231	0.24577	0.20334	0.48552	0.15909
0.13932	0.14371	0.26609	0.26627	0.20939	0.19602

โดยวิธีการทดสอบสารูปสนิทดี

ชั้นที่	ขีดจำกัดล่าง	ขีดจำกัดบน	ความถี่ค่าสังเกตุ	ความถี่ค่าคาดหวัง
1	0.00000	0.20000	10.00000	10.33920
2	0.20000	0.40000	13.00000	12.66240
3	>0.40000		7.00000	6.99840

χ^2= 0.020 องศาแห่งความอิสระ = 2

ค่า P = 0.99

ตัวอย่างที่ 6 ผลการทดสอบการแจกแจงของตัวแปรสุ่มแบบปกติ

x_0 = 5 k = 131 m = 32768 μ = 0.0 σ = 1.0

จำนวนตัวเลขสุ่มที่ผลิต 30 ตัว ดังนี้

-0.04254	0.51607	-0.86288	0.82076	-0.43318	1.37547
0.24653	-0.81991	0.17621	1.23484	-0.64412	0.53951
-1.21445	0.09418	-0.53475	0.89889	1.39500	-1.04648
-0.42537	-0.74178	1.00436	-1.18711	0.68404	0.61763
-2.38636	1.67235	0.79342	-1.02304	1.22312	-0.46834

โดยวิธีการทดสอบสารูปสนิทดี

ชั้นที่	ขีดจำกัดล่าง	ขีดจำกัดบน	ความถี่ค่าสังเกตุ	ความถี่ค่าคาดหวัง
1	<-0.50000		10.00000	9.25613
2	-0.50000	0.00000	4.00000	5.74387
3	0.00000	0.50000	3.00000	5.74387
4	>0.50000		13.00000	9.25613

χ^2= 3.414 องศาแห่งความอิสระ = 3

ค่า P = 0.33

ตัวอย่างที่ 7 ผลการทดสอบการแจกแจงของตัวแปรสุ่มแบบลอกนอร์มอล

x_0 = 5 k = 131 m = 32768 μ = 1.5 σ = 2.1

จำนวนตัวเลขสุ่มที่ผลิต 30 ตัว ดังนี้

4.09866	13.24662	0.73195	25.11854	1.80457	80.51837
7.52108	0.80107	6.48862	59.92920	1.15875	13.91494
0.34981	5.46180	1.45796	29.59702	83.88965	0.49777
1.83442	0.94389	36.93519	0.37049	18.84953	16.39589

0.02986150 19612 23.71677 0.52289 58.47233 1.67614

โดยวิธีการทดสอบสารูปสนิทดี

ชั้นที่	ขีดจำกัดล่าง	ขีดจำกัดบน	ความถี่ค่าสังเกตุ	ความถี่ค่าคาดหวัง
1	0.00000	5.00000	14.00000	15.62342
2	5.00000	20.00000	7.00000	7.23196
3	>20.00000		9.00000	7.14462

χ^2 = 0.658 องศาแห่งความอิสระ = 2

ค่า P = 0.72

ตัวอย่างที่ 8 ผลการทดสอบการแจกแจงของตัวแปรสุ่มแบบไวบูล

x_0 = 5 k = 131 m = 32768 α = 1.1 β = 2.2

จำนวนตัวเลขสุ่มที่ผลิต 30 ตัว ดังนี้

0.06334	2.12759	0.09618	0.34383	5.86844	0.25117
1.58360	2.73906	0.87492	3.88949	3.80513	5.09890
1.43918	0.05889	1.26165	3.88650	3.46024	2.81017
2.11127	2.15473	2.62494	0.29909	3.43379	1.11810
1.69877	0.76890	1.05849	0.66150	2.35222	1.01832

โดยวิธีการทดสอบสารูปสนิทดี

ชั้นที่	ขีดจำกัดล่าง	ขีดจำกัดบน	ความถี่ค่าสังเกตุ	ความถี่ค่าคาดหวัง
1	0.00000	1.00000	9.00000	10.29023
2	1.00000	2.00000	7.00000	7.51838
3	>2.00000		14.00000	12.19139

χ^2 = 0.466 องศาแห่งความอิสระ = 2

ค่า P = 0.79

ตัวอย่างที่ 9 ผลการทดสอบการแจกแจงของตัวแปรสุ่มแบบไคสแควร์

x_0 = 5 k = 131 m = 32768 องศาแห่งความอิสระ = 3

จำนวนตัวเลขสุ่มที่ผลิต 30 ตัว ดังนี้

1.01270	2.75322	0.76408	2.23079	1.76972	3.84915
1.73993	2.25861	9.12102	2.76198	2.62910	4.27105
3.65744	4.70294	1.19977	1.65406	5.25961	1.75396

0.83141 4.17641 1.66895 7.42806 2.52239 2.13184

0.80515 5.36841 7.65376 1.20990 2.30794 3.13089

โดยวิธีการทดสอบสารูปสนิทดี

ชั้นที่	ขีดจำกัดล่าง	ขีดจำกัดบน	ความถี่ค่าสังเกตุ	ความถี่ค่าคาดหวัง
1	0.00000	2.43547	15.00000	15.38800
2	2.43547	4.10685	7.00000	7.10736
3	>4.10685		8.00000	7.50464

χ^2= 0.044 องศาแห่งความอิสระ = 2

ค่า P = 0.98

ตัวอย่างที่ 10 ผลการทดสอบการแจกแจงของตัวแปรสุ่มแบบสติวเด้นที

x_0 = 5 k = 131 m = 32768 องศาแห่งความอิสระ = 4

จำนวนตัวเลขสุ่มที่ผลิต 30 ตัว ดังนี้

-0.06218 1.81830 -0.89739 0.92562 0.71227 1.81620

1.00788 1.58995 -3.54070 0.81396 -0.76572 -0.48172

0.09868 2.32835 -0.00538 -0.40200 0.25724 -0.38381

-0.53858 0.09096 -0.82052 -1.61848 0.09303 0.68671

-0.81319 0.25034 -2.09418 0.46550 0.53828 0.38313

โดยวิธีการทดสอบสารูปสนิทดี

ชั้นที่	ขีดจำกัดล่าง	ขีดจำกัดบน	ความถี่ค่าสังเกตุ	ความถี่ค่าคาดหวัง
1	<-0.50000		8.00000	9.64995
2	-0.50000	0.00000	5.00000	5.35005
3	0.00000	0.50000	7.00000	5.35005
4	>0.50000		10.00000	9.64995

χ^2= 0.827 องศาแห่งความอิสระ = 3

ค่า P = 0.84

ตัวอย่างที่ 11 ผลการทดสอบการแจกแจงของตัวแปรสุ่มแบบเอฟ

x_0 = 5 k = 131 m = 32768 k_1 = 2 k_2 = 4

จำนวนตัวเลขสุ่มที่ผลิต 30 ตัว ดังนี้

0.15332	0.64817	0.71764	0.44757	5.00753	1.21003
1.54426	0.73034	4.81967	0.25687	0.03923	1.58213
0.00184	12.40178	0.09693	2.89881	3.21614	0.28976
1.62975	0.24783	0.70697	1.25110	0.00448	8.14237
0.35917	0.60046	1.00757	3.51647	0.65097	0.03408

โดยวิธีการทดสอบสารูปสนิทดี

ชั้นที่	ขีดจำกัดล่าง	ขีดจำกัดบน	ความถี่ค่าสังเกตุ	ความถี่ค่าคาดหวัง
1	0.00000	0.50000	11.00000	10.80000
2	0.50000	1.00000	6.00000	5.86667
3	>1.00000		13.00000	13.33333

x^2 = 0.015 องศาแห่งความอิสระ = 2

ค่า P = 0.99

ตัวอย่างที่ 12 ผลการทดสอบการแจกแจงของตัวแปรสุ่มแบบมัลติแวริเอตนอร์มอล

x_0 = 5 k = 131 m = 2147483648

$\mu = (0,0)$

$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

จำนวนเวกเตอร์สุ่มที่ผลิต 1500 ตัว

โดยวิธีการทดสอบสารูปสนิทดี

ความถี่ค่าสังเกตุ

ขีดจำกัด	<-1.5	-1.5 - 0.0	0.0 - 1.5	>1.5
<-1.5	8.00000	43.00000	45.00000	9.00000
-1.5 - 0.0	41.00000	270.00000	278.00000	37.00000
0.0 - 1.5	45.00000	269.00000	295.00000	56.00000
>1.5	6.00000	46.00000	48.00000	4.00000

ความถี่ค่าคาดหวัง

ขีดจำกัด	<-1.5	-1.5 - 0.0	0.0 - 1.5	>1.5
<-1.5	6.42700	42.53343	42.53343	6.42700
-1.5 - 0.0	42.53343	281.48393	281.48393	42.53343
0.0 - 1.5	42.53343	281.48393	281.48393	42.53343
>1.5	6.42700	42.53343	42.53343	6.42700

X^2 = 10.389 องศาแห่งความอิสระ = 15

ค่า P = 0.79

ตัวอย่างที่ 13 ผลการทดสอบการแจกแจงของตัวแปรสุ่มแบบทวินาม

x_0 = 5 k = 131 m = 32768 n = 4 p = 0.2

จำนวนตัวเลขสุ่มที่ผลิต 30 ตัว ดังนี้

3 1 0 1 0 1 0 0 1 0 1 0 1 2 1

1 0 1 1 1 0 0 2 1 0 1 2 1 0 0

โดยวิธีการทดสอบสารูปสนิทดี

x	ความถี่ค่าสังเกตุ	ความถี่ค่าคาดหวัง
0	12.00000	12.28800
1	14.00000	12.28800
>1	4.00000	5.42400

X^2 = 0.619 องศาแห่งความอิสระ = 2

ค่า P = 0.73

ตัวอย่างที่ 14 ผลการทดสอบการแจกแจงของตัวแปรสุ่มแบบปัวส์ซง

x_0 = 13949 k = 131 m = 32768 λ = 1.7

จำนวนตัวเลขสุ่มที่ผลิต 30 ตัว ดังนี้

2 1 1 2 4 2 2 0 3 3 3 3 1 1 1

1 2 3 2 1 1 1 2 0 1 0 1 1 1 2

โดยวิธีการทดสอบสารูปสนิทดี

x	ความถี่ค่าสังเกตุ	ความถี่ค่าคาดหวัง
0	3.00000	5.48051
1	13.00000	9.31686
2	8.00000	7.91933
>2	6.00000	7.28330

x^2= 2.806 องศาแห่งความอิสระ = 3

ค่า P = 0.42

ตัวอย่างที่ 15 ผลการทดสอบการแจกแจงของตัวแปรสุ่มแบบเรขาคณิต

x_0 = 5 k = 131 m = 32768 p = 0.6

จำนวนตัวเลขสุ่มที่ผลิต 30 ตัว ดังนี้

0 1 0 0 3 0 0 1 0 2 1 2 0 0 0

2 1 1 1 1 1 0 1 0 0 0 0 0 1 0

โดยวิธีการทดสอบสารูปสนิทดี

x	ความถี่ค่าสังเกตุ	ความถี่ค่าคาดหวัง
0	16.00000	18.00000
1	10.00000	7.20000
>1	4.00000	4.80000

x^2= 1.444 องศาแห่งความอิสระ = 2

ค่า P = 0.49

ตัวอย่างที่ 16 ผลการทดสอบการแจกแจงของตัวแปรสุ่มแบบทวินามนิเสธ

x_0 = 5 k = 131 m = 32768 r = 5 p = 0.5

จำนวนตัวเลขสุ่มที่ผลิต 200 ตัว ดังนี้

5 4 5 7 4 1 1 9 4 4 3 10 6 9 4 4 5 2 4 5

4 4 10 10 6 2 2 6 4 3 3 2 5 6 2 4 8 7 2 8

8 4 3 4 4 2 5 3 5 11 5 2 3 6 5 7 3 4 4 0

9 7 5 6 8 3 6 3 7 11 5 5 3 4 8 8 3 5 3 7

3 6 7 4 12 11 13 7 5 11 5 9 5 1 5 3 2 1 5 3

3 7 2 3 7 11 5 3 6 5 7 5 1 6 5 5 2 12 10 11

7 1 3 6 8 7 4 0 6 3 11 3 0 5 6 5 6 12 2 2

6 6 9 6 2 3 4 6 4 6 3 2 1 8 3 2 8 11 6 7

5 3 2 3 3 7 2 9 3 5 8 9 6 5 7 2 9 1 3 4

7 3 3 3 8 1 0 2 2 3 5 21 11 2 6 3 4 9 2 3

โดยวิธีการทดสอบสารูปสนิทดี

X	ความถี่ค่าสังเกตุ	ความถี่ค่าคาดหวัง
0	4.00000	6.25000
1	9.00000	15.62500
2	23.00000	23.43750
3	35.00000	27.34375
4	23.00000	27.34375
5	29.00000	24.60938
6	22.00000	20.50781
7	17.00000	16.11328
8	11.00000	12.08496
9	9.00000	8.72803
10	4.00000	6.10962
>10	14.00000	11.84692

χ^2 = 8.627 องศาแห่งความอิสระ = 11

ค่า P = 0.66