Unit 3: Congruential Random Number Generator

Contents

1.0 Introduction

As has been stated earlier, if you want to write a simulation program and you neither have a simulation language at your disposal nor a programming language with a random number generating function, you must design and write a random number generating function and call it whenever you need it.

Classical uniform random number generators have some major defects, such as, short period length and lack of higher dimension uniformity. However, nowadays there are a class of rather complex random number generators which are as efficient as the classical generators which enjoy the property of a much longer period and of higher dimension uniformity.

2.0 Intended Learning Outcomes (ILOs)

By the end of this unit, the reader should be able to:

- Explain the use Congruential method for generating Random numbers;
- Choose appropriate parameters for congruential method;
- Translate the method to computer programs;
- Use other very similar random number generating methods such as: Mid square, Mid product, Fibonacci

3.0 Main Content

3.1 The Congruential Method

The Congruential Method is widely used. The method is based on modulus arithmetic, which we now discuss briefly.

We say that two numbers x and y are congruent modulo m if (x-y) is an integral multiple of m. Thus we can write: $x = y(\text{modulo } m)$

For example, let m = 10, then we can write:

$13 \equiv 3$ (modulo 10)

$84 \equiv 4$ (modulo 10)

The ***congruential method*** generates random numbers by computing the next random number from the last random number obtained, given an initial random number say, X_0, called the seed.

The method uses the formula:

$X_{n+1} = (aX_n + c)(\text{modulo } m)$ where X_0 = Seed and; $a, c < m$,

Where a, c and m are carefully chosen positive integer constants of which a and c must be less than m, X_0 is the seed or the last random number generated in the sequence. Stated in the computer language, the above formula becomes:

$X_{(N+1)} = (A * X_{(N)} + C)$ MOD M (FORTRAN)

or

R = (A* SEED + C) MOD M (QBASIC)

From the above formula, it follows that the random number generated must be between 0 and (m-1) since MOD (modulo) produces remainder after division. Hence the above formula will produce the remainder after dividing ($aX_n + C$) by m. So to generate a random number between p and m we use:

$X_{n+1} = (aX_n + C)(\text{modulo } m + 1 - p) + p$, for $m > p$.

If the value of c is zero, the congruential method is termed Multiplicative Congruential Method. If the value of c is not zero, the method is called Mixed Congruential Method.

The **multiplicative congruential** method is very handy. It is obtained using the general formula:

$r_n = ar_{n-1}$ (modulo m)

where the parameters a, m and the seed r_0 are specified to give desirable statistical properties of the resultant sequence. By virtue of modulo arithmetic, each r_n must be one of the numbers 0,1,2,3… m-1. Clearly, you must be careful about the choice of a and r_0. The values of 'a' and r_0 should be chosen to yield the largest cycle or period, that is to give the largest value for n at which $r_n = r_0$ for the first time.

Example 4
To illustrate the technique, suppose you want to generate ten decimal place numbers u_1, u_2, u_3,…. It can be shown that if you use
$u_n = r_n \times 10^{-1}$
where $r_n = 100003r_{n-1}$ (modulo 10^{10}), and r_0 = any odd number not divisible by 5,
then the period of the sequence will be 5×10^8, that is $r_n = r_0$ for the first time at $n = 5 \times 10^8$ and the cycle subsequently repeats itself.

As an example, using our mixed congruential formula
$X_{n+1} = (aX_n+c)$ (modulo m),
And suppose m = 8, a = 5, c = 7 and X_0 (seed) = 4 we can generate a random sequence of integer numbers thus:

n	$X_{n+1} = (5X_n+7)$mod 8
0	X1 = (5*X0+7)mod 8 = (5*X4+7)mod 8 = 27 mod 8 = 3
1	X2 = (5*X1+7)mod 8 = (5*X3+7)mod 8 = 22 mod 8 = 6
2	X3 = (5*X2+7)mod 8 = (5*X6+7)mod 8 = 37 mod 8 = 5
3	X4 = (5*X3+7)mod 8 = (5*X5+7)mod 8 = 32 mod 8 = 0
4	X5 = (5*X4+7)mod 8 = (5*X0+7)mod 8 = 7 mod 8 = 7
5	X6 = (5*X5+7)mod 8 = (5*X7+7)mod 8 = 42 mod 8 = 2
6	X7 = (5*X6+7)mod 8 = (5*X2+7)mod 8 = 17 mod 8 = 1
7	X8 = (5*X7+7)mod 8 = (5*X1+7)mod 8 = 12 mod 8 = 4

Note that the value of X_8 is 4, which is the value of the seed X_0. So if we compute X_9, X_{10}, etc the same random numbers 3,6,5,0,7,2,1,4 will be generated once more.

Note also that if we divide the random integer values by 8, we obtain random numbers in the range $0 \leq X_{n+1} < 1$ which is similar to using the RND function of BASIC.

3.1 Choice of a, c and m

The method of this random number generation by *linear congruential method*, works by computing each successive random number from the previous. Starting with a seed, Xo, the linear congruential method uses the following formula:

X_{i+1} = (A*Xi + C) mod M

In his book, *The Art of Computer Programming*, Donald Knuth presents several rules for maximizing the length of time before the random number generator comes up with the same value as the seed. This is desirable because once the random number generator comes up with the initial seed, it will start to repeat the same sequence of random numbers (which

will not be so random since the second time around we can predict what they will be). According to Knuth's rules, if M is prime, we can let C be 0.
The LCM defined above has full period if and only if the following conditions are satisfied:

a) m and c are relatively prime
b) If q is a prime number that divides m , then q divides a-1
c) If 4 divides m, then 4 divides a-1

Therefore, the values for a, c and m are not generated randomly, rather they are carefully chosen based on certain considerations. For a binary computer with a word length of r bits, the normal choice for m is m = 2^{r-1}. With this choice of **m**, **a** can assume any of the values 1, 5,9,13, and **c** can assume any of the values 1, 3, 5, 7… However, experience shows that the congruential method works out very well if the value of a is an odd integer not divisible by either 3 or 5 and c chosen such that c mod 8 = 5 (for a binary computer) or c mod 200 = 21 (for a decimal computer).

Example 5
Develop a function procedure called RAND in QBASIC which generates a random number between 0 and 1 using the mixed congruential method. Assume a 16-bit computer.

Solution
```
FUNCTION RAND (SEED)
CONST M = 32767, A = 2743, C = 5923
IF SEED < 0 THEN SEED = SEED + M
SEED = (A* SEED + C) MOD M
RAND = SEED/M
END FUNCTION
```

Note that in the main program that references the above function in (a), the TIMER function can be used to generate the SEED to be passed to the function RAND as illustrated in example 2.

Example 5
Write a program that can generate that can generate 20 random integer number distributed between 1 and 64 inclusive using mixed congruential method.

Solution
QBASIC

```
DECLARE RAND (X)
CLS: REM Mixed Congruential Method
DIM SHARED SEED
SEED = TIMER
FOR K% = 1 TO 20
```

```
        SEED = RAND (SEED)        ‘Call of function RAND PRINT SEED: SPC(2)
NEXT K%
END ‘End of main program

FUNCTION RAND (SEED) ‘Beginning of function subprogram
CONST M = 64 A = 27, C = 13
IF SEED = THEN SEED = SEED + M
SEED = (a* SEED + C) MOD M + 1
RAND = SEED
END FUNCTION ‘End of the function program RAND
```

(b) Using FORTRAN

```
PROGRAM RANDNUM
COMMON SEED
CLS     ‘Clear screen
DO 50 K = 1, 25
WRITE (*, 5)
FORMAT(/)
50      CONTINUE
WRITE(*,*) ‘Enter the seed’
READ(*,*) SEED
DO 30 J = 1, 20
SEED = RAND
WRITE (*, 6) SEED
6       FORMAT (I4)
30      CONTINUE
END

FUNCTION RAND
COMMON SEED
PARAMETER (M = 64, A = 27, C =13)
IF (SEED.LT.0) SEED = SEED + M
HOLD = (A*SEED + C)
SEED = AMOD (HOLD,M) + 1
RAND = SEED
RETURN END
```

3.3 RANECU Random Number Generator

A FORTRAN code for generating uniform random numbers on [0,1]. RANECU is multiplicative linear congruential generator suitable for a 16-bit platform. It combines three simple generators, and has a period exceeding 81012.

It is constructed for more efficient use by providing for a sequence of such numbers (Length), to be returned in a single call. A set of three non-zero integer seeds can be supplied, failing which a default set is employed. If supplied, these three seeds, in order, should lie in the ranges [1,32362], [1,31726] and [1,31656] respectively. The program is given below.

```
        SUBROUTINE RANECU (RVEC,LEN)
C Portable random number generator for 16 bit computer.
C Generates a sequence of LEN pseudo-random numbers, returned
C in RVEC.
        DIMENSION RVEC(*)
        SAVE ISEED1,ISEED2, ISEED3
        DATA ISEED1,ISEED2,ISEED3/1234, 5678, 9876/
C Default values, used if none is suppliednvia an ENTRY
C call at RECUIN
        DO 100 I = 1,LEN
        K=ISEED1/206
ISEED1 = 157 * (ISEED1 - K * 206) - K * 21
IF(ISEED1.LT.0) ISEED1=ISEED1+32363
K=ISEED2/217
ISEED2 = 146 * (ISEED2 - K*217) - K* 45
IF(ISEED2.LT.O) ISEED2=ISEED2+31727
K=ISEED3/222
ISEED3 = 142 * (ISEED3 - K *222) - K * 133
IF(ISEED3.LT.0) ISEED3=ISEED3+31657
IZ=ISEED1-ISEED2
IF(IZ.GT.706)IZ = Z - 32362 IZ = 1Z+ISEED3
IF(IZ.LT.1)IZ = 1Z + 32362
RVEC(I)=REAL(IZ) * 3.0899E – 5
100     CONTINUE
RETURN
ENTRY RECUIN(IS1, IS2, IS3)
ISEED1=IS1
ISEED2=IS2
ISEED3=IS3
RETURN
ENTRY RECUUT(IS1,IS2,IS3)
IS1=ISEED1
IS2=ISEED2
IS3=ISEED3
RETURN
END
```

3.4 Other Methods of Generating Random Numbers

Some other methods of generating random numbers are Quadratic Congruential Method, Mid-square method, Mid-product Method and Fibonnachi Method.

3.4.1 The Quadratic congruential method

This method uses the formula:

$X_{n=1} = (dX_n^2 + cX_n + a)$ modulo m

Where d is chosen in the same way as c and m should be a power of 2 for the method to yield satisfactory results.

3.4.2 The Mid-square method

The first random number is generated from the seed by squaring the seed and discarding all the digits except the middle four digits. This number is subsequently used as the new seed to generate the next random number in the same manner and so on.

The formula is: $X_{n+1} = X_n^2$

The mid-square method is rarely used these days as it has the tendency to degenerate rapidly. Also, if the number zero is ever generated, then all subsequent numbers generated will be zero. Furthermore, the method is slow when simulated in the computer since many multiplications and divisions are required to access the middle four digits.

3.4.3 The mid-product method

This method is similar to the mid-square method, except that a successive random number is obtained by multiplying the current number by a constant c, and taking the middle digits.

The formula is: $X_{n+1} = cX_n$

The mid-product method has a longer period and it is more uniformly distributed than the mid-square method.

3.4.4 The Fibonacci method

Fibonacci method uses the formula: $X_{n+1} = (X_n + X_{n-1})$ modulo m

In this method, two initial seeds need to be provided. However, experience has shown that the random numbers generated by using Fibonacci method fail to pass tests for randomness. Therefore, the method does not give satisfactory results.

From the foregoing discussions, it is obvious that the last three methods – mid-square, mid-product and Fibonacci are of historical significance and have detrimental and limiting characteristics.

4.0 Self-Assessment Exercise(s)

1. Write a QBASIC program using Quadratic congruential method to generate 15 random integer numbers between 1 and 50.
2. Produce a table of random numbers using multiplicative congruential method, using a =5, m =8 and $X_0 = 4$. Draw an inference from your solution.
3. Write a QBASIC function that can be referenced as computer random number between 30 and 100 using mixed congruential method.
4. Use the mixed congruential method to generate the following sequences of random numbers:

a. A sequence of 10 one-digit random numbers given that $X_{n+1} \equiv (X_n + 3)(\text{modulo } 10)$ and $X_0 = 2$
b. A sequence of eight random numbers between 0 and 7 given that $X_{n+1} \equiv (5X_n + 1)(\text{modulo } 8)$ and $X_0 = 4$
c. A sequence of two-digit random numbers such that $X_{n+1} \equiv (61X_n + 27)(\text{modulo } 100)$ and $X_0 = 40$
d. A sequence of five-digit random numbers such that $X_{n+1} \equiv (21X_n + 53)(\text{modulo } 100)$ and $X_0 = 33$

5. Define a methods period and state how to improve a period. Show two examples of such improvement.
6. Consider the multiplicative congruential method for generating random digits. In each part below, assume modulo 10 arithmetic and determine the length of the cycle:
   a. Let a = 2 and r_0 = 1, 3 and 5
   b. Let a = 3 and r_0 = 1,2 and 5

5.0 Conclusion

In this unit, you have been introduced to Random Numbers generation. You have also learnt how to design random number generator.

6.0 Summary

What you have learnt in this unit concern:

- The Congruential methods of generating random numbers,
- The use of QBasic RND function to simulate randomness,
- The other Random number generating methods,
- The properties of good random number generator.

7.0 Further Readings

- Devore, J. L. (2018). *Probability and statistics for engineering and the sciences*. Toronto, Ontario: Nelson.
- Georgii, H. (2013). *Stochastics: Introduction to probability and statistics*. Berlin: De Gruyter.
- Giri, N. C. (2019). *Introduction to probability and statistics*. London: Routledge.
- Johnson, R. A., Miller, I., & Freund, J. E. (2019). *Miller & Freunds probability and statistics for engineers*. Boston: Pearson Education.
- Laha, R. G., & Rohatgi, V. K. (2020). *Probability theory*. Mineola, NY: Dover Publications.
- Mathai, A. M., & Haubold, H. J. (2018). *Probability and statistics: A course for physicists and engineers*. Boston: De Gruyter.

- Pishro-Nik, H. (2014). *Introduction to probability, statistics, and random processes*. Blue Bell, PA: Kappa Research, LLC.
- Spiegel, M. R., Schiller, J. J., & Srinivasan, R. A. (2013). *Schaums outline of probability and statistics*. New York: McGraw-Hill.