- 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.

Unit 2: Random Numbers

Contents

1.0 Introduction

The use of Random numbers lies at the foundation of modelling and simulations. Computer applications such as simulations, games, graphics, etc., often need the ability to generate random numbers for such application.

The quality of a random number generator is proportional to its **period**, or the number of random numbers it can produce before a repeating pattern sets in. In large-scale simulations, different algorithms (called shift-register and lagged-Fibonacci) can be used, although these also have some drawbacks, combining two different types of generators may produce the best results.

2.0 Intended Learning Outcomes (ILOs)

By the end this unit, you should be able to:

- Describe how to generate pseudorandom numbers,
- Use QBasic RND function and describe how to simulate randomness,
- Use different Random number generators,
- Explain properties of good random number generator.

3.0 Main Content

Random Number can be defined as numbers that show no consistent pattern, with each number in a series and are neither affected in any way by the preceding number, nor predictable from it.

One way to get random digits is to simply start with an arbitrary number with a specified number of digits, for example 4 digits. The first number is called the **seed**. The seed is multiplied by a **constant** number of the same number of digits(length), and the desired number of digits is taken off the right end of the product. The result becomes the new seed. It is again multiplied by the original constant to generate a new product, and the process is repeated as often as desired. The result is a series of digits that appear randomly distributed as though generated by throwing a die or spinning a wheel. This type of algorithm is called a **congruential generator**.

Generating a random number series from a single seed works fine with most simulations that rely upon generating random events under the control of probabilities (Monte Carlo simulations). However, although the sequence of numbers generated from a given seed is randomly distributed, it is always the same series of numbers for the same seed. Thus, a computer poker game that simply used a given seed would always generate the same hands for each player.

What is needed is a large collection of potential seeds from which one can be more or less randomly chosen. If there are enough possible seeds, the odds of ever getting the same series of numbers become diminishingly small.

One way to do this is to read the time (and perhaps date) from the computer's system clock and generate a seed based on that value. Since the clock value is in milliseconds, there are millions of possible values to choose from. Another common technique is to use the interval between the user's keystrokes (in milliseconds). Although they are not perfect, these techniques are quite adequate for games.

The so-called true random number generators extract random numbers from physical phenomena such as a radioactive source or even atmospheric noise as detected by a radio receiver.

3.1 Pseudorandom Number Generation

In this section we look at how random numbers may be generated by human beings for use in simulating a system or by computer for use while simulating an event.

What we usually do is to take for instance ten pieces of papers and number them 0,1,2,3,4,5,6,7,8, and 9 , fold and place them in a box. Shake the box and thoroughly mix the slips of paper. Select a slip; then record the number that is on it. Replace the slip and repeat this procedure over and over. The resultant record of digits is a realized sequence of random numbers. Assuming you thoroughly mix the slips before every draw, the nth digit of the sequence has an equal or uniform chance of being any of the digits 0, 1, 2,3,4,5,6,7,8, 9 irrespective of all the preceding digits in the recorded sequence.

In some simulations, we use random numbers that are between 0 and 1. For example, if you need such numbers with four decimal digits, then you can take four at a time from the recorded sequence of random digits, and place a decimal point in front of each group of four. To illustrate, if the sequence of digits is 358083429261… then the four decimal placed random numbers are .3580, .8342, and .9261.

3.2 Random Numbers in Computer

How does computer generate a sequence of random numbers?

One way is to perform the above “slip-in-a-box” experiment and then store the recorded sequence in a computer-backing store.

The RAND Corporation using specially designed electronic equipment, to perform the experiment, actually did generate a table of a million random digits. The table can be obtained on tape, so that blocks of the numbers can be read into the memory of a high- speed computer, as they are needed. Their approach is disadvantageous since considerable computer time was expended in the delays of reading numbers into memory from a tape drive.

Experts in computer science have devised mathematical processes for generating digits that yield sequences satisfying many of the statistical properties of a truly random process. To illustrate, if you examine a long sequence of digits produced by deterministic formulas, each digit will occur with nearly the same frequency, odd numbers will be followed by even numbers about as often as by odd numbers, different pairs of numbers occur with nearly the same frequency, etc. Since such a process is not really random, it is called **pseudo-random number generator**.

The other ways of generating pseudo-random numbers are:

1. Computer simulation languages and indeed some programming languages such as BASIC have built-in pseudo-random number generators. In computer simulation situations where this facility is not available in the language you are using, you will have to write you own pseudo-random number generator (see how to do this later).
2. The results of experiments such as the one previously describe above are published in books of statistical tables. In hand simulation, it may be appropriate to use a published table of random numbers.
3. The conventional six-sided unbiased die may also be used to generate a sequence of random digits in the set (1, 2, 3, 4, 5, 6) where each digit has a probability 1/6 of occurrence.

Exercise

Suggest one or two experimental set-ups (analogous to the slip-in-a-box approach) for generating uniform random digits.

3.3 Using the RND Function in BASIC

The BASIC programming language has a numeric function named RND, which generates random numbers between 0 and 1. Each time RND is executed, a pseudo random number between 0 and 1 is generated. Using RND function at any time will always generate the same sequence of pseudo random numbers unless we vary the random number seed using the BASIC statement:

RANDOMIZE

This way, we can control the sequence of random numbers generated. RANDOMIZE will result to the following prompt on the VDU:

Random Number Seed (-32768 to 32767)?

Suppose your response to the above prompt is 100. Then the computer would use this number, 100, to generate the first random number. This number generated is used to generate the next random number. Thus by specifying the seed for the first random number, we are in a way controlling all random numbers that will be generated until the seed is reset. A control such as this can be very useful in validating a simulation program or other computer programs that use random numbers.
Consider the following BASIC program:

```
FOR K% = 1 TO 5
PRINT RND NEXT K%
END
```

If the above program is run, some seven-digit decimal numbers like the following will be displayed: .6291626, .1948297, .6305799, .8625749, .736353. The particular digits displayed depend on the system time.

Every time you run the above program, different sequence of numbers will be displayed.
Now add a RANDOMIZE statement to the program:

```
RANDOMIZE TIMER FOR K% = 1 TO 5
        PRINT RND NEXT K%
END
```

If you run this program with 300 as a response to the prompt for the random number seed, the following may be displayed: .1851404, .9877729, .806621, .8573399, .6208935

Exercise
Find out whether the same set of random numbers will be displayed each time the above program is run with seed 300.

3.4 Simulating Randomness

Suppose we want to simulate the throwing of a fair die. A random number between 0 and 1 will not always satisfy our needs. If the die is fair, throwing it several times will yield a series of uniformly distributed integers 1,2,3,4,5 and 6. Consequently we need to be able to generate a random integer with values in the range 1 and 6 inclusive.

Now the function RND generates a random number between 0 and 1. Specifically, the random variable X is in the range: $0 \leq X < 1$
The expression X = RND *6
Will generate a number in the range: $0 \leq X < 6$
We must convert these numbers to integers as follows: X% = INT (RND*6)
The expression produces an integer in the range: $0 \leq X < 5$

But we need the range: $0 \leq X < 6$

Therefore if we need to add 1 to the above expression in simulating the tossing of a die. Thus,

X% = INT (RND*6) + 1
In general, to generate a random integer between P and N we use the expression:
INT(RND*N+1-P) + P;
where N>P

While for integer number between 0 and N – 1 we use the expression INT (RND *N).

Example 1
A simple QBASIC program that will stimulate the tossing of two dice and display the value obtained after each toss, and the total value of the dice is shown below.

```
CLS
REM DI and D2 represent the individual dice. RANDOMIZE
DO
        D1% = INT(RND*6) + 1
        D2% = INT(RND*6) + 1
        TOTAL% = D1% + D2%
        PRINT "Die 1:"; D1%; "Die 2:"; D2%
        PRINT: PRINT
        INPUT "Toss Again (Y/N)?", Y$
        LOOP UNTIL UCASE$(Y$) = "N"

END
```

Exercise
Run the program of example 1several times using different random number seeds to

determine if the integers generated for the individual die are uniformly distributed between 1 and 6 inclusive.

If we want the computer to be generating the random number seed automatically, we use RANDOMIZE TIMER

In place of RANDOMIZE.

Example 2
Another QBASIC program to simulate the tossing of a fair coin 10 times. The program displays a H when a head appears and a T when a tail appears.

```
CLS
REM Program to simulate the tossing of a coin 10 times
REM and print the outcome
RANDOMIZE TIMER
FOR K% = 1 TO 10
RANDNO = RND
IF RANDNO <= 0.5 PRINT "H"
IF RANDNO > 0.5
PRINT "T"
NEXT K%
END
```

Example 3
Suppose the output of the program of example 3 is: HHTHHTTTHH
and that there are two players X and Y involved in the tossing of the coin. Given that player X wins N50.00 from player Y if a head appears and loses it to player Y if a tail appears. Determine who won the game and by how much.

Solution
From the output there are 6 heads and 4 tails.
Player X wins N50.00 x 6 = N300.00 from player Y. He loses N50.00 x 4 = N200.00 to player Y.
Thus, player X won the game with N300.00 – N200.00 = N100.00.

3.5 Properties of a Good Random Number Generator
The random numbers generated should;

a. have as nearly as possible a uniform distribution.
b. should be fast
c. not require large amounts of memory.
d. have a long period.
e. be able to generate a different set of random numbers or a series of numbers.
f. not degenerate.

4.0 Self-Assessment Exercise(s)

Write a QBASIC program to generate thirty random integer numbers distributed between 20 and 50. Your program should ensure that no number is repeated.
Write a QBASIC program to accept a set of characters from the keyboard and then move the characters randomly across the screen. The movement of the characters should stop once a key is pressed on the keyboard. The set of characters should also change colors randomly at the point of the movement.
What is a seed and explain how you can generate random numbers using a seed.
Define a period and state how to improve a period.

5.0 Conclusion

In this unit, you have been introduced to Random Numbers generation. You have also learnt the how to manipulate the RND function of QBASIC.

6.0 Summary

What you have learnt in this unit concern:

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

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.