SA421 – Simulation Modeling — Fall 2015
Asst. Prof. Nelson Uhan

Lesson 10. Random Number Generation, Testing for Independence

1 Overview

- A **random number** is a random observation from a Uniform$[0,1]$ distribution
- How do we tell a computer to generate random numbers: i.e., sample independent values from the Uniform$[0,1]$ distribution?
  - e.g. the `uniform` function in `numpy.random`
- It is very difficult to get a computer to do something randomly
  - A computer, by design, follows its instructions blindly, and is therefore completely predictable
  - A computer that doesn't do this is broken!
- One approach: **pseudo-random number generators**

2 Pseudo-random number generators (PRNGs)

- "Psuedo" means having a deceptive resemblance
- PRNGs are (deterministic) algorithms that use mathematical formulas or precalculated tables to produce sequences of numbers that appear random

2.1 Desirable properties of a PRNG

Efficient. Can produce many numbers in a short amount of time

Deterministic. A given sequence of numbers can be reproduced at a later date if the starting point in the sequence is known

- Useful for comparing different systems

Long cycle. If the PRNG is periodic (generates a sequence that eventually repeats itself), the cycle length should be sufficiently long

- Modern PRNGs have a period so long that it can be ignored for most practical purposes

Pass statistical tests for uniformity and independence. Most importantly: these numbers should not be statistically differentiable from a sequence of truly independently sampled values from the Uniform$[0,1]$ distribution

- We can test for uniformity using goodness-of-fit tests (e.g. Kolmogorov-Smirnov)
- We will discuss testing for independence later

3 The linear congruential generator

- Produces sequence of integers $X_1, X_2, \ldots$ using the following recursion:

  - The initial value X_0 is called the
  - The minimum possible value of $X_1, X_2, \ldots$ is
  - The maximum possible value of $X_1, X_2, \ldots$ is

- The **stream**, or the sequence of generated pseudo-random numbers is

- The modulus is often chosen to be a power of 2: binary computations are fast on a computer
- If $c = 0$, this is a **multiplicative congruential generator**
- If $c \neq 0$, this is a **mixed congruential generator**

3.1 Period length

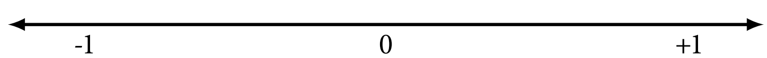
- The **period** of a linear congruential generator (LCG) is the smallest integer n such that $X_0 = X_{n-1}$ (how many iterations of the LCG take place before the sequence starts to repeat itself)
- An LCG has **full period** if its period is m (Why?)
- **Theorem.** An LCG has full period if and only if:
  (i) c and m are relatively prime: the only positive integer that divides both c and m is 1
  (ii) If m is a multiple of 4, then $a - 1$ is a multiple of 4
  (iii) If p is a prime number dividing m, then $a - 1$ is a multiple of p

Example 1. Consider the LCG with modulus 16, increment 11, and multiplier 9. Confirm that this LCG has full period.

4 Testing for independence

- Many tests have been devised to determine whether a sequence of random variates are independent: testing for independence is a deep problem
- One simple, quick-and-dirty way to test whether these variates are independent is to plot the **autocorrelation** of the sequence
- Roughly speaking, autocorrelation helps us detect repeating patterns in a sequence of values
- Let $x_0, \ldots, x_{n-1}$ and $y_0, \ldots, y_{n-1}$ be sequences of observed random variates
- The **observed sample correlation coefficient** between $(x_0, \ldots, x_{n-1})$ and $(y_0, \ldots, y_{n-1})$ is

  - Also known as the **Pearson correlation coefficient**
  - Ranges between -1 and $+1$:



- The **lag-k autocorrelation** of $(y_0, \ldots, y_{n-1})$ is the observed sample correlation coefficient between $(y_0, \ldots, y_{n-k-1})$ and $(y_k, \ldots, y_{n-1})$
- In other words, the lag-k autocorrelation helps us detect if there is a pattern between every k observations