

Lesson 11. Variance Reduction – Common Random Numbers, Generating Randomness in JaamSim

0 Warm up

Example 1. Let X be a continuous uniform random variable on $[a, b]$. Recall that the cdf of X is

$$F_X(x) = \begin{cases} 0 & \text{if } x < a \\ \dfrac{x-a}{b-a} & \text{if } a \le x \le b \\ 1 & \text{if } x > b \end{cases}$$

Find a random variate generator for X.

1 Motivation

- Consider the Fantastic Dan problem once again: single server with a single queue
- Suppose now that customer interarrival times are uniformly distributed between 10 and 25 minutes
- Consider the following alternatives:

  System 1. In the current system, service times are uniformly distributed between 30 and 45 minutes

  System 2. Fantastic Dan is considering a new haircutting technique. In this hypothetical system, service times are uniformly distributed between 15 and 20 minutes

- Fantastic Dan wants to know what effect this new haircutting technique would have on the time average number of customers waiting to get a haircut (in the queue)

2 The effect of using a single stream of random numbers

- Remember that random variates are generated using a stream of random numbers
- At first, it seems like a good idea to generate all the distribution samples we need from a single stream of numbers: more efficient, less bookkeeping
- Let's examine this idea in JaamSim
- The files `11-system1-onestream.cfg` and `11-system2-onestream.cfg` contain simulations of Systems 1 and 2, respectively, using a single stream of random numbers
- That is, both the interarrival times and service times are sampled using the same stream of random numbers
- In addition, the same stream of random numbers is used in both simulations
- Run the simulations, and take a look at the arrival times in both systems
- What do you observe?
- Why does this happen?
- Intuitively, we don't want this to happen: we want our analysis to focus on the difference between the two systems, which is the service time distribution, not the interarrival time distribution
- This in turn will help us reduce the variance in our estimates of the difference between the two systems

3 Using separate streams and common random numbers

- By setting up separate streams for interarrival times and service times, we can ensure that the interarrival times are the same across both systems
- This is called the **common random number** technique: ensuring that common random numbers are used to generate the same random variates for matching parts of alternate systems
  - Other names: **correlated sampling**, **matched streams**, matched pairs
- It turns out that JaamSim generates a separate stream for each probability distribution object
- As a result, by modeling probability distributions the "natural" way in JaamSim, you implement the common random number technique automatically!
- The files `11-system1-sepstreams.cfg` and `11-system2-sepstreams.cfg` contain simulations of Systems 1 and 2, respectively, using a single stream of random numbers
- Run the simulations, and take a look at the arrival times in both systems
- What do you observe?
- Why does this happen?

4 Generating randomness in JaamSim

- You may have already noticed that every probability distribution object has an input called **RandomSeed**
- RandomSeed is the seed for the random number generator associated with that probability distribution
    - RandomSeed can be set to any nonnegative integer
- When replicating a simulation many times, we need to change the seeds of the probability distributions
    - Otherwise, we'd just get the same output over and over
- The easiest way to achieve this is to make sure **Simulation** → **GlobalSubstreamSeed** changes from replication to replication
    - GlobalSubstreamSeed can be set to any nonnegative integer
    - An easy choice is simply to set GlobalSubstreamSeed to the appropriate RunIndex
- The seed of each probability distribution object is based on a combination of the value of its RandomSeed input, as well as the GlobalSubstreamSeed input