Statistically Sound Discrete Event Simulations

Varsha Apte CS 681

Reference: Simulation Modeling & Analysis by Law & Kelton

Outline

- Conducting proper simulations of random systems
- Analyzing output of simulations of random systems

Conducting Simulations of Random Systems

Using random Number Generators, streams, seeds, repetitions, generating random variates of distributions

Random Experiments

- Often, quantitative questions have to be asked about “stochastic” or random systems
  - E.g. Probability that a coin toss results in a head
  - Average score on a particular test by 5^{th} graders
  - Etc…

…Random Experiments

- To answer these questions, we have to do *experiments* which have *random outcomes*
- Consider the problem of finding :

p_h = Prob[Coin toss results in a head]

for the new 10-rupee coins

Example 1: Coin Toss Experiment

One way

- Get a coin – toss it m times
- Note m_h = number of times result was head
- Estimate probability as m_h/m

Problem

- There might be minor differences in different coins (some randomness). Only one coin may not give a good estimate

…Coin Toss Experiment

Better way:

- Get *n* coins (10-15)
- Do *m* (100-200) tosses with each coin
- Note m_h^i for each coin *i*
- The experiment with each coin *i* will result in a fraction of heads seen $p_h^i = m_h^i/m$
- Estimate required probability p_h as:

$(p_h^1 + p_h^2 + p_h^3 + \ldots + p_h^n)/n$

…Coin Toss Experiment

- This method takes care of random differences between coins
- Should give a better estimate than using just one coin
- Intuitively, we know that increasing *n* and *m* will improve the estimate
- But still – how do we know how close we are to the “real” probability of head?

Example 2: M/M/1 queue

- Questions we want to ask. For a given arrival rate λ and average service time (τ)
  - What is the average utilization?
  - What is the average delay (waiting time) of a customer? …etc
- M/M/1 queue is a "random" system – how do we determine these quantities? (In this case, we know the theoretical answer – but what if we didn't?)
  - Observe an actual M/M/1 queue (like we actually did coin tosses) with the given parameters – not always possible
  - Estimate the quantities using a discrete event simulation program

M/M/1 Simulation

- Simulation should be functionally correct
- Simulation should be *statistically* correct.
- Since original system has randomness, simulation model should also incorporate randomness.
- Specifically, in this case, inter-arrival times, and service times should be EXP

…Randomness in Simulations

How do we generate EXP times ?

- Use pseudo-random number generators available as library calls in most languages
- “rand” or similar function returns a random number (called a random *variate)* from a Uniform (0,1) distribution
  - If R ~ Uniform(0,1), P[R < r] = r.

…Generating Random Variates

- But we want EXP random variates
- Note code in the queuesim.c is as follows

```
float expon(float mean)  /* Exponential variate generation function. */
{
    /* Return an exponential random variate with mean "mean". */
    return -mean * log ( lcgrand(stream) );
}
```

- Here “mean” is mean of the EXP distribution
- And lcgrand(stream) returns a Uniform (0,1) pseudo-random number

…Generating EXP random variates

- If R ~ Uniform (0,1) and
- X = - m ln(R) //this the formula in the code
- What is the distribution of X ?

Pr [X < x] = P[-m ln R < x]

= P[ln R > -x/m] = P [R > $e^{-x/m}$]

= 1 – P[R < $e^{-x/m}$]

= 1 – $e^{-x/m}$ → EXP (*1/m*)

- Exponentially distributed with mean *m*

Using pseudo-random numbers

- Given Uniform(0,1) random numbers, we can generate EXP random numbers
- Random number generator (RNG) functions have multiple "streams" of random numbers that they can generate
  - Stream can be given as argument to the RNG function call
  - In each stream "seed" specifies the point of starting the stream
    - Seed of a stream is set using a related function

    (lcgrandst (seed, stream) in the queuesim.c code)

…Using pseudo-random numbers

- If we do not change stream or seed, we will get the *same results* for a simulation for a given arrival rate and service time (see code & run it)
  - This is not very useful
- We must do simulations in a way that even for the same arrival rate and service time, we should get different results in different runs
- → Set different streams and seeds in different runs

…Using pseudo-random numbers

- Using different streams in different runs is important (random numbers from the same stream may be correlated)
- If we carry out multiple runs with different seeds/streams, we get varied estimates for the same metric (e.g. average delay), even for the same arrival rate and service time
  - This is what we want (it is similar to using 10 different coins and doing 100 tosses with each coin)
- See the code & run it

Summary of part I

- Run each simulation to collect one sample point of your metric (e.g. one M/M/1 simulation run gives one sample point of average delay)
- *Do repetitions (runs)* of your simulation
- In each repetition, ensure that random number generator with different stream and seed is being used
- This will ensure output is *random* and *different* for each run

Analyzing Simulation Output Data

Please note in the rest of these slides $\underline{X}(n)$ and $\overline{X}(n)$ are one and the same thing. (Slides are using $\underline{X}(n)$ because it is tedious to get “X-bar” in powerpoint!)

M/M/1 queue example Average delay estimates

- For $\lambda = 1$, $\tau = 0.8$, number delays recorded in each run is 2000. These were averaged for each run. Five such runs (with different seeds/streams) resulted in these values:
  - 3.240
  - 3.354
  - 3.807
  - 3.885
  - 3.557
- Average of these is 3.57

Estimating Average Delay

- But how do we know whether this is “correct” (close to the real average delay?)

…Estimating Average Delay

- In the M/M/1 case, we know that the correct answer (calculated theoretically) is:

$$\frac{\tau}{1-\rho} - \tau$$

- Where τ is service time and ρ is utilization.
- For τ =0.8 and λ =1, ρ = 0.8, this value is 3.2
- The simulation gave us 3.57, which seems close enough

…estimating average delay

- But how do we know close/far from the "real" value that we are estimating, when we do not know the real value?
- Use confidence intervals

Confidence intervals

Theory of sample mean and variance, central limit theorem, how it applies to simulation output

Note: in the following slides the symbol μ is used to denote population mean. This is the convention in statistics. Do not confuse this with service rate μ of queueing theory.

Sample mean & variance

- Let X_1, X,…X_n be i.i.d. random variables with mean μ and variance σ^2 (they are like sample points from a population with mean μ and variance σ^2)
- Then sample mean $\overline{X}(n)$ given by,

$$\overline{X}(n) = \frac{\sum_{i=1}^{n} X_i}{n}$$

is an unbiased estimator of μ. i.e. $E[\ \underline{X}(n)\ ] = \mu$

- And sample variance $S^2(n)$ given by:

$$S^2(n) = \frac{\sum_{i=1}^{n} [X_i - \overline{X}(n)]}{n-1}$$

is an unbiased estimator of σ^2 , i.e. $E[S^2(n)] = \sigma^2$

Variance & mean of Sample Mean

- Note that the sample mean $\underline{X}(n)$ itself is a random variable.
  - If we get a different set of *n* sample points from the same population, the sample mean is different for each set → $\underline{X}(n)$ itself is random variable
- We already know that expectation of the sample mean is μ ($E[\underline{X}(n)] = \mu$)
- What is the *variance* of the sample mean?
  - $Var[\underline{X}(n)] = \sigma^2/n$
  - You can see that Variance of the sample mean *decreases* as the size of the sample increases!

Important Note

Note that the following two quantities are totally different

- *Sample Variance $S^2(n)$*
  - This is the variance of the sample, an estimator of population variance σ^2. If the sample size (n) is large , this should be close to the population variance σ^2
- *Variance of the Sample Mean*
  - ($Var[X(n)] = \sigma^2/n$)
  - This is the variance of the random variable that is the sample mean. This variance DECREASES (to 0!!) if the sample is large

Central Limit Theorem

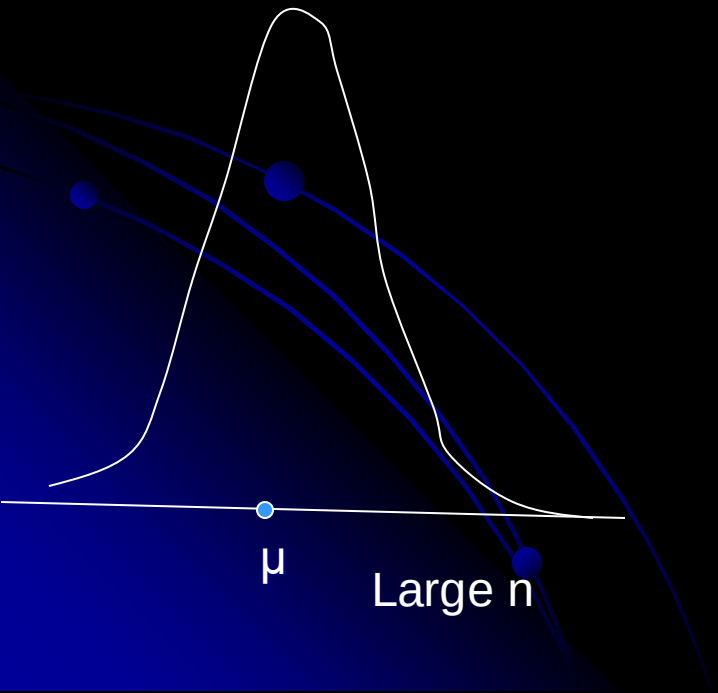
- If *n* is large then $\overline{X}(n)$ (the random variable that is the sample mean) has normal distribution
  - With Mean: μ
  - And variance σ^2/n
- Equivalently,

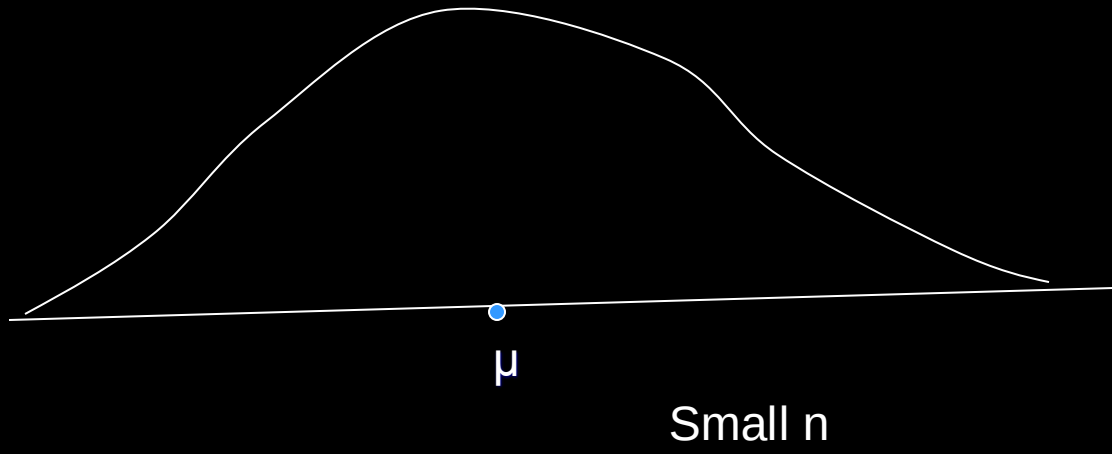
$$Z_n = \frac{\overline{X}(n) - \mu}{\sqrt{\sigma^2 / n}}$$

is Normal(0,1) – has standard normal distribution

…Central Limit Theorem

In other words, if we take many different samples of size *n* – and calculate sample mean $\underline{X}(n)$ for each sample and plot the histogram, the shape will look “normal” if *n* is large. Also, as *n* increases the variance will decrease, so the chance of $\underline{X}(n)$ to be closer to μ will be higher.





…Central Limit Theorem

- Now, in Central Limit Theorem, the result still applies if we replace σ^2 by *sample variance,* because $S^2(n) \rightarrow \sigma^2$ as *n* grows to infinity
- So we can re-state the theorem as:

For a large *n*

$$Z_n = \frac{\overline{X}(n) - \mu}{\sqrt{S^2(n)/n}} \sim Normal(0,1)$$

Applying Central Limit Theorem to Discrete Event Simulations

Confidence intervals!

Applying Central Limit Theorem

- When we do multiple simulation runs (also called *repetitions*), each run *i* generates a statistic X_i. This is one sample point.
  - For the same system, with the same parameters, we can assume that $X_1, X_2, \ldots, X_n$ are identically distributed.
  - We can assume that they are *independent* since we ensure we used a different stream/seed for each run

So we can assume X_i's are i.i.d. So they can be the *n sample points* of the Central Limit Theorem.

…applying Central Limit Theorem

- Sample mean is defined as usual ($\underline{X}(n)$)
- Sample variance ($S^2(n)$)is defined as usual
- Central Limit theorem may help us answer the question of how close is $\underline{X}(n)$ to the actual mean μ?

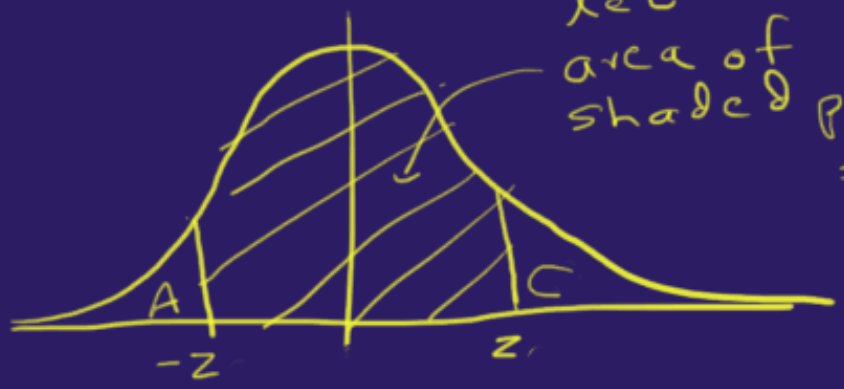
…using Normal Distribution

- Now, by central limit theorem $Z_n = \frac{\bar{X}(n) - \mu}{(S^2(n)\ /n)}$ has normal distribution.
- So, for any given value β, where 0< β <1, there is a point *z* such that

P[-z <= Z_n <= z] = β.

- This is because normal distribution is *symmetric* around the mean (mean in this case is 0).
- Given a β , can we figure out what the value of this *z* is going to be?
- (See explanation next slide)

Normal pdf



let area of shaded portion $= \beta$, $0 < \beta < 1$

Area of regions $A + C = 1 - \beta$

Area of region A = Area of region $C = \frac{1-\beta}{2}$

$$P[Z < z] = \beta + \left(\frac{1-\beta}{2}\right)$$

$$= \frac{1+\beta}{2}$$

This z is denoted by $z_{\frac{1+\beta}{2}}$

i.e. In general for $Z \sim N(0,1)$

$$P[Z < z_c] = c$$

z_c is called the upper c-critical point

…using Normal Distribution

- Now for standard normal distribution, CDF values have been tabulated. So given some *p,* we can find a z_p such that
  - $P[Z_n < z_p] = p$, where $Z_n \sim \text{Normal}(0,1)$
- This z_p is called the *upper p critical point* of the standard normal distribution.
- E.g. from Table C.3 in KST book, $z_{0.95}$ is 1.55 i.e. $P[Z_n < 1.55] = 0.95$

…using Normal Distribution

- Coming back to given a value β, $0 < \beta < 1$, the z that will give us

$$P[-z < Z_n < z] = \beta$$

Will be equal to $z_{(1+\beta)/2.}$(as explained earlier).

…Confidence Intervals

- So given any β, we can find $z_{(1+\beta)/2}$ (from standard normal distribution tables), such that $P[-z_{(1+\beta)/2} < Z_n < z_{(1+\beta)/2}] = \beta$. Now:

$$Given, \quad Z_n = \frac{\bar{X}(n) - \mu}{\sqrt{S^2(n)/n}}$$

$$and \quad P[-z_{(1+\beta)/2} \le Z_n \le z_{(1+\beta)/2}] = \beta$$

$$\Rightarrow P[-z_{(1+\beta)/2} \le \frac{\bar{X}(n) - \mu}{\sqrt{S^2(n)/n}} \le z_{(1+\beta)/2}] = \beta$$

$$\Rightarrow P[-z_{(1+\beta)/2}\sqrt{S^2(n)/n} \le \bar{X}(n) - \mu \le z_{(1+\beta)/2}\sqrt{S^2(n)/n}] = \beta$$

$$\Rightarrow P[z_{(1+\beta)/2}\sqrt{S^2(n)/n} \ge -\bar{X}(n) + \mu \ge -z_{(1+\beta)/2}\sqrt{S^2(n)/n}] = \beta$$

$$\Rightarrow P[\bar{X}(n) - z_{(1+\beta)/2}\sqrt{S^2(n)/n} \le \mu \le \bar{X}(n) + z_{(1+\beta)/2}\sqrt{S^2(n)/n}] = \beta$$

…Confidence Interval

- The equation says that the mean μ lies in this interval with probability β :

$$(\overline{X}(n) - z_{(1+\beta)/2}\sqrt{S^2(n)/n},\quad \overline{X}(n) + z_{(1+\beta)/2}\sqrt{S^2(n)/n})$$

- This is also called the 100β % confidence interval for the mean μ.
  - This means that there is a 100β % chance that the real μ lies in this interval.

Confidence Interval

- This answers the question of how good is our estimate of the mean?
  - Note now: we do NOT give a single point estimate, we now give an interval (the confidence interval) as our "interval estimate"
- We say that we are 100 β% confident that the actual mean lies in this interval

Confidence Interval

- Width of the interval is large if $S^2(n)$, i.e., sample variance, is large
- To reduce the width of the interval, we have to increase n, then variance of the sample mean (σ^2/n), estimated by $S^2(n)/n$, will decrease.
- The formula shows that to halve the width of the interval, you have to do four times as many repetitions.

Confidence Interval

- Now we will show a step by step method to generate confidence intervals, with the example of the M/M/1 simulation. You should extrapolate this to your project/assignment.
- Metric being estimated is average delay.
- Suppose you will keep average service time τ constant and vary λ from $0.1/\tau$ to $0.9/\tau$ in steps of $0.1/\tau$

Confidence Interval Method

1. Initialize
    1. *n* (number of runs) ~ around 20
    2. *m* (number of samples in each run) ~ long enough for queue to achieve steady state (1000s)
    3. Initialize $\lambda = 0.1/\tau$
    4. Initialize confidence level β (say 0.95)
    5. Initialize $z_{(1+\beta)/2}$, the $(1+\beta)/2$ upper critical point of the standard normal distribution
2. While $\lambda <= 0.9/\tau$
    1. Carry out the *n* runs with different seeds/streams. From each run $i = 1, 2 \ldots n$, you will get an average delay X_i

…Confidence Interval

2. For the sample of average delays $X_1, X_2, \ldots X_n$
   1. Calculate sample mean $\underline{X(n)}$
   2. Calculate sample variance $S^2(n)$
3. Calculate confidence interval as:

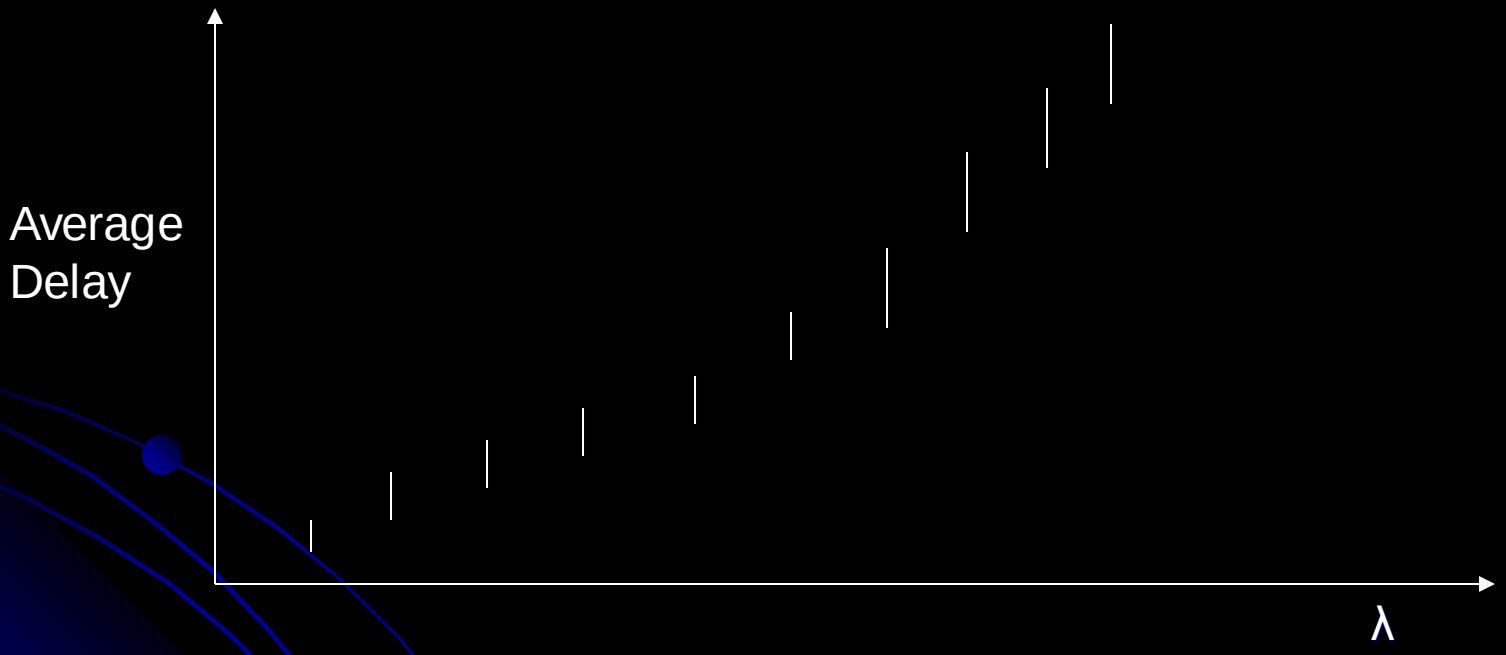
$$\left(\overline{X}(n) - z_{(1+\beta)/2}\sqrt{S^2(n)/n},\quad \overline{X}(n) + z_{(1+\beta)/2}\sqrt{S^2(n)/n}\right)$$

4. This is the confidence interval of the average delay for this value of λ. Save these values.
5. Increment λ, and go to step 2.1

4. Output the values (λ, confidence interval) for all λ, and plot them.

Confidence Intervals

- Your final plot will look like this:



Note: the width of the interval will generally increase with increasing load. Try to do enough number of repetitions so that confidence interval is not too wide (10-20% around midpoint)

Numerical Example

- See spreadsheet.

Appendix (code)

- Download queuesim.c, lcgrand.c and lcgrand.h. Also input file queuesim.in
- Inputs
  - Demo? (1 If you want detailed trace, 0 if not)
  - Mean interarrival time, mean service time, Number of delays per run
  - Number of runs (repetitions)
  - <seed,stream> for each run
- Compile as
  - gcc –o queuesim queuesim.c lcgrand.c –lm
- Output of all runs in file queuesim.out – average utilization, delay etc. for each run.