Additional notes

These lecture notes summarizes several issues that do not appear in the lecture slides that were dealt with on the white board in class. Most but not all of this material is covered in Chapters 1-2 of the textbook.

1 Simulation Classification

Besides the three classification criteria for simulations (discrete time/continuous time, deterministic/stochastic, offline/real-time) there is a fourth: static/dynamic. A static simulation involves no time at all, whereas any simulation which evolves a system in time is called dynamic. Examples of static simulations are stress calculation in building structures, most Monte Carlo simulations, electrostatics, etc.

2 More Terminology

When an identical simulation is repeated several times, this is called replication. This is sometimes necessary for stochastic simulations, as the randomness in the system will result in a different result for whatever you are computing every time you run the simulation. By replication we can get an idea of the statistical spread in the computed quantities and compute their average.

When we start a discrete event simulation such as the queueing system example, we say we perform a cold start if the system is initially empty. It then takes a while for the system to fill up and stabilize into its steady state. A warm start involves an initial state with several entities already in the system. Quite often a warm start is implemented by first simulating the system from a cold start for a while, then use the end state of that simulation as the warm start state.

3 Random number generation

Quite often we need to write a function that returns one of several possible values (e.g., arrival or service times), with specified probabilities, as in the examples in Chapter 2. Suppose the probability to return a particular value $T_i$, $i = 1, \ldots, n$ is specified as $p_i$. The trick is to generate a uniform random number $r$, $0 \leq r \leq 1$, and assign an interval of width $p_i$ to each return value $T_i$. After generating $r$ we then check in which interval $r$ ended up, and we return that value. See sim1.m for an example.

4 Stable/Unstable/Critical Systems

These concepts are defined in the lecture slides. To locate approximately a critical point we can follow two methods. In the first method we plot a quantity like the average queue length
<q> as a function of the total simulation time. We do this for several values of some system parameter x (which was the “arrivalFactor” in example sim1.m). From eyeballing the plots we determine if <q> grows indefinitely over time or not. The second method consists of plotting <q> for a fixed but large value of the total simulation time T as a function of x. This function should go to ∞ at the critical point, in the limit T → ∞. Of course, we can only realize large but finite T, in which case <q> will merely get large near the critical point. By plotting <q> versus x we can estimate where it gets large. Note that every data point for this plot requires a full simulation.

In the 1 queue 1 server example (sim1.m, ex. 2.1 in the book) it is clear that the system will be stable if the average service time is less than the average interarrival time. The average interarrival time is < T_a >= 4.5 * arrFactor and the average service time (from table 2.7 in the book) is < T_s >= 3.2. We therefore expect the critical value for arrFactor to be that value where < T_a >= < T_s >. Solving for arrFactor gives arrFactor_critical = 3.2/4.5 = 0.7111.

5 Averages

The sample code sim1.m contained a deliberate mistake. The average queue length is a time average. The time average of a quantity q over an interval [0 T] is defined as

\[ \int_0^T q(t)dt/T. \]

In our case q changes at the discrete times t_i and the integral becomes a weighted sum. The sample code sim1.m omitted this time weighting and instead averaged over events, which is not a meaningful average.

In class we added code to sim1.m (made available online as sim1b.m) to verify Little’s law. We computed the average arrival rate λ correctly as (“total number of arrivals” / “total time”). The average interarrival time can be calculated easily from table 2.6 in the textbook, < T_a >= 4.5 * arrFactor. You may wonder if we can now simply calculate λ (the arrival rate) from < T_a >. If T_a was a constant we would obviously have λ = 1/T_a. If it is a stochastic variable is it λ = 1/ < T_a > or λ = < 1/T_a >? That these are not the same can be seen from he definition of average. Suppose P(T_a) is the probability density function (pdf) for T_a. We have

\[ < T_a > = \int_{-\infty}^{\infty} T_a P(T_a)dT_a \]

and

\[ \lambda = < 1/T_a > = \int_{-\infty}^{\infty} \frac{P(T_a)}{T_a}dT_a. \]

It is clear that < T_a >≠ 1/ < T_a > unless the pdf P(T_a) has some special properties. So which average should we use for λ? Suppose we have generated N arrivals with interarrival times T_{a1},...,T_{aN}. At this point we have simulated up to time T = T_{a1} + T_{a2} + ... + T_{aN}. λ is
just the total number of arrivals \( (N) \) per time \((T)\) so we have \( \lambda = N/T \). Writing this as

\[
\lambda = 1/((T_a^1 + T_a^2 + \ldots + T_a^N)/N)
\]

we recognize the denominator as \(<T_a>\), so the correct answer is \( \lambda = 1/ <T_a> \).