ICS 2307 SIMULATION AND MODELLING

Course Outline

- Systems modelling – discrete event simulation
- Design of simulation experiments simulation
- Language probability and distribution theory
- Statistical estimation, inference and random number generators
- Sample event sequences for random number generation
- Translation of models for simulation application

References

- Simulation modelling and analysis

Introduction

Computers can be used to *imitate* (simulate) the operations of various kinds of real world facilities or processes. *The facility or process of interest is usually called a* ***system*** and in order to *study it scientifically*, we often have to make a set of *assumptions* about how it works.

These **assumptions** which usually *take the form of mathematical or logical relationships constitute a model* that is used to try to gain some understanding of how the corresponding system behaves. *If the relationships that propose the model are simple enough, it may be possible to use mathematical methods to obtain exact information on questions of interest*. This is called an ***analytic solution***. However, most real world systems are too complex to allow realistic models to be evaluated analytically. Such models must be studied by means of ***simulation***.

In a simulation, we use a computer to *evaluate a model numerically* and *data is gathered* in order to estimate the desired true characteristics of the model. For example: consider a manufacturing firm that is contemplating building a large extension onto one of its plants but is not sure if the potential gain in productivity would justify the construction costs. A careful simulation study would shed some light on the question by simulating the operation of the plant as it currently exists and as it would be if the plant were expanded.

Simulation has been found to be a useful tool in a number of areas:

i. Designing and analysing manufacturing systems.
ii. Evaluating hardware and software requirements for a computer system.
iii. Evaluating new military weapons systems or tactics.
iv. Determining ordering policies for an inventory system.
v. Designing communication systems and message protocols.
vi. Designing and operating transportation facilities e.g. highways, airports etc.
vii. Evaluating designs for service organizations such as hospitals, post offices or fast food restaurants.
viii. Analysing financial or economic systems.

Systems, Models and Simulations

Definitions

A **system** is a *collection of entities* e.g. people and machines *that act and interact together* towards the *accomplishment of some logical end*

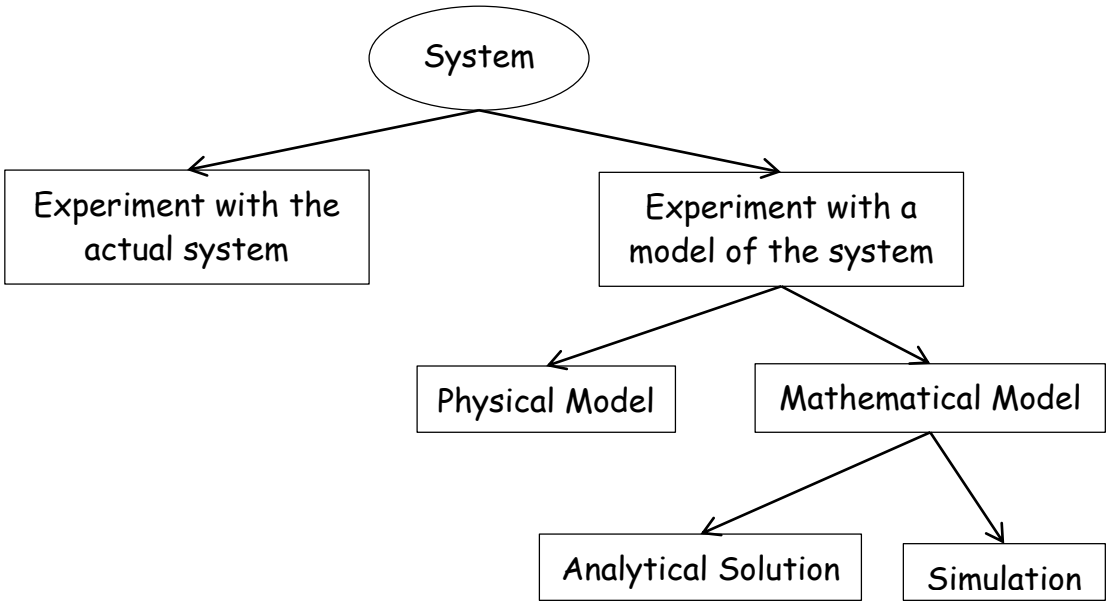
The **state of a system** is the *collection of variables* necessary to *describe* a system at a particular time *relative to the objects of a study.*

Systems can be categorized to be of two types:

i. **Discrete system -** is one for which the *state variables change instantaneously at separate points in time* e.g. in a bank, the number of customers changes only when a customer arrives or finishes being served and departs

ii. C**ontinuous system** - is one for which the *state variables change continuously with respect to time* e.g. an airplane moving through the air has its state variables like position and velocity changing continuously with time

There is need to study most systems to gain some insight into the relationships among various components or to predict performance under some new conditions being considered; this could be done in the following ways:



Simulation models can be classified along three different dimensions:

1) **Static Vs. Dynamic simulation models**

A ***static simulation*** model is a representation of the system *at a particular time* or one that may be used to represent *a system in which time plays no role* e.g. Monte Carlo systems.

A ***dynamic simulation*** model represents a system as it *evolves over time* such as a conveyer system in a factory.

2) **Deterministic Vs. Stochastic**

If a simulation *does not contain any probabilistic (random) components*, it is called ***deterministic*** e.g. a complicated system of differential equations describing a chemical reaction.

In deterministic models, the output is "determined" once the set of input quantities and relationships in the models have been specified. However, many systems must be modelled as having at least some random input components and this gives rise to stochastic simulation models e.g. queuing and inventory systems are modelled stochastically.

3) **Continuous Vs. Discrete simulation models**

Discrete and continuous simulation models are defined analogously to the way discrete and continuous systems are defined. But a discrete model is NOT always used to model a discrete system and vice versa. The decision whether to use a discrete or continuous model for a particular system depends on the specific objectives of the study e.g. a model of traffic flow on a highway would be discrete if the characteristics and movements of individual cars are important. Alternatively if the cars can be treated in the aggregate, the flow of traffic can be described by differential equations in a continuous model.

Discrete Event Simulation

Discrete Event Simulation concerns the modelling of a system as it evolves over time by a representation in which the state variables change instantaneously at separate points in time. These points in time are the ones at which an event occurs where an event is defined as an instantaneous occurrence that may change the state of a system.

Example

Consider a service facility with a single server e.g. a one-operator barber shop or an information desk at an airport, for which we would like to estimate the expected (average) delay in queue (line) of arriving customers where the delaying queue of a customer is the length of time interval from the instant of his/her arrival at the facility to the instance he/she begins to be served.

The state variables for a discrete event simulation model of the facility would be:

- Status of the server – Idle or busy
- The number of customers waiting in queue to be served, if any
- Time of arrival of each person waiting in queue.

The status of the server is needed to determine, upon the customer's arrival, whether the customer can be served immediately or MUST join the end of the queue.

When the server completes serving a customer, the number of customers in the queue determines whether the server will be idle or will begin serving the first customer in the queue.

The time of arrival of a customer is needed to compute his/her delay in queue which is the time he begins to be served less his arrival time

$$delay\ in\ queue = time\ he\ begins\ being\ served - time\ of\ arrival$$

There are two types of events in the system:

- Arrival of a customer
- Completion of services for a customer which results in the customer's departure

An arrival is an event since it causes the (state variable) server status to change from idle to busy or the number of the customer in the queue to increase by one.

Correspondingly, a departure is an event because it causes the server status to change from busy to idle or the number of customers in the queue to decrease by 1.

In this example, both types of events actually changed the states of the system, however, in some discrete-event simulation model, events are used for purposes that do not actually affect such a change e.g. an event might be used to schedule the end of a simulation run at a particular time or to schedule a decision about a systems operation at a particular time and might not actually result in a change in the state of the system.

After an arrival/departure; the system status is described as:
server - idle/busy & queue - size increases/decreases by 1

Time Advance Mechanisms

Because of the dynamic nature of discrete-event simulation models, we must keep track of the current value of simulated time as the simulation proceeds and we also need a mechanism to advance simulated time from one value to another.

We call the variable in a simulation model that gives the converted value of simulated time the **simulation clock**.

The unit of time for the simulation time is never stated explicitly when a model is written in a general purpose language. It is assumed to be in the same units as the input parameters.

Also, there is generally no relationship between simulated time and the time needed to run a simulation on the computer. Two principal approaches have been used to advance the simulation clock:

i. Next-event time advance approach
ii. Fixed-increment time advance approach

i. Next Event Time Approach

[ONE SERVER!!!]

In this case, a simulation clock is initialized to zero and times of occurrence of future events are determined. The simulation clock is then advanced to the time of occurrence of the most imminent (first) of these future events at which point the state of the system is updated to account for the fact that an event has occurred, and our knowledge of times of occurrence of future events is also updated. Then the simulation clock is advanced to the (new) most imminent event, the state of the system is updated and future event times are determined. This process of advancing the simulation clock from one event time to another is continued until eventually some pre-specified stopping condition is satisfied.

Since all state changes occur only at event times for a discrete event simulation model, periods of inactivity are skipped over by jumping the clock from event time to event time. Successive jumps of the simulation clock are generally variable (unequal) in size.

Example

An illustration of the next event time advance approach using a single server queuing system. We use the notations:

t_i – Time of arrival of the i[th] customer and $t_0 = 0$.

$A_i = t_i - t_{i-1}$ – Inter-arrival time between the $(i-1)^{th}$ and i^{th} arrivals of customers.

S_i – The time that the server actually spends serving the i[th] customer (exclusive of customer delay on the queue).

D_i – The delay in queue of the i[th] customer

$C_i = t_i + D_i + S_i$ – Time that the i[th] customer completes services and departs

e_i – The time of occurrence of i[th] event of any type (i[th] value the simulation clock takes time $e_0 = 0$

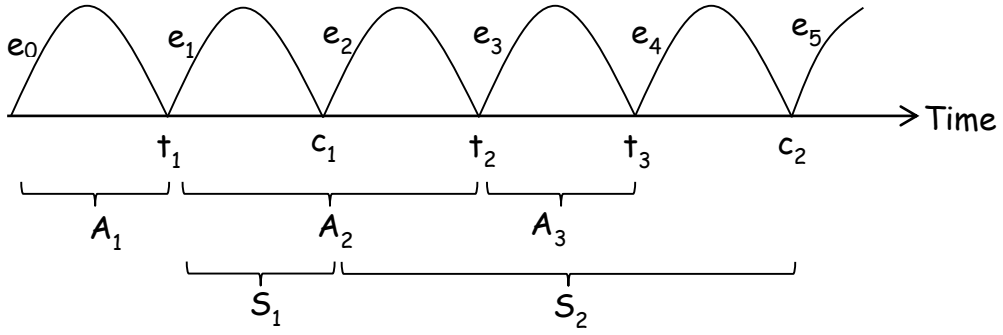
Each of these defines quantities that would generally be random variables.

Assume that the probability distribution of the inter-arrival times A, A ,...and service times S_1, S_2,… are known and have cumulative distribution functions denoted by F_A and F_S respectively (in general, F_A and F_S will be determined by collecting data from the system of interest and then fitting distributions to this data using assessing sample techniques).

At time e_0=0, the status of the server is idle and t_1 of the 1[st] arrival is determined by generating A_1 from F_A and adding it to 0. The simulation clock is then advanced from C_0 to the time of the next (first) event $e_1 = t_1$

ti implies an arrival - server changes from idle to busy BUT;

ci implies completion/departure so server changes from busy to idle



Link to a modified diagram with t4, t5, c3, t6:

[s3 should start at c2; the lec's diagram has an issue] ***https://prnt.sc/u24kov***

When describing system status, MUST:
both server status & queue length eg empty

Since the customer arriving at t_1 finds the server idle, he/she immediately has serving and has a delay of t_1 and the status of the server is changed from idle to busy. The time C_1 when the arriving customer will complete service is computed by generating S_1 from F_S and adding it to t_1.

delays only affect events in the queue - during arrival, there are no delays

The time of the 2nd arrival t_2 is computed as $t_2 = t_1 + A_2$ where A_2 is generated from F_A. If t_2 is less than C_1, the simulation clock is advanced from e_1 to the time of the next event, $e_2 = t_2$. If C_1 is less than t_2, the simulation clock is advanced from e_1 to C_1. Since the customer arriving at t_2 finds the server idle, he enters service immediately and the number in the queue remains zero.

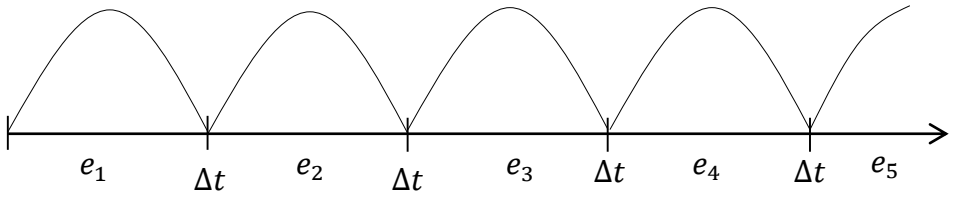
Time of third arrival t_3 is computed as $t_3 = t_2 + A_3$. If C_1 is less than t_3, the simulation clock is advanced from e_2 to the next event. The arriving customer finds the server idle and goes for service. The simulation clock is then advanced to time t_3. The arriving customer here finds the server busy and joins the queue. The process continues in this manner.

ii. Fixed-Increment Time Advance Approach

In this approach, the simulation is advanced in increments of exactly Δt time units for some appropriate choice of Δt. After each update of the clock, a check is made to determine if any events should have occurred during the previous interval of length Δt.

If one or more events were scheduled to have occurred during this interval, these events are considered to occur at the end of the interval and the system state (and statistical count is updated accordingly).

The figure below defines the fixed increment time approach where the curved arrows represent the advancing of the simulation clock and $e_i,\ i = 1, 2, 3, \ldots$ is the actual time of occurrence of the i^{th} event of any type in the interval $[0, \Delta t]$.



An event occurs at time e_1 but is said to occur at time Δt in the model. No events occur in the model $[\Delta t, \Delta t]$ but the model checks to determine if this is the case. Events occur at the times e_2 and e_3 in the interval $[2\Delta t, 3\Delta t]$ but both events are considered to occur at time $3\Delta t$.

A set of rules must be built into the model to decide in what order to process events when two or more events are considered to occur at the same time by the model.

Two disadvantages of fixed increment time advance are:

i. The errors by processing events at the end of the interval in which they occur and
ii. The necessity of deciding which event to process first when events that are not simultaneous in reality are treated as such by the model.

These problems can be made less severe by making Δt smaller but this increases the amount of checking for event occurrences that must be done and results in an increase in execution time. Because of these considerations, fixed increment time advance is generally not used for discrete events simulation models when the times between successive events can vary greatly.

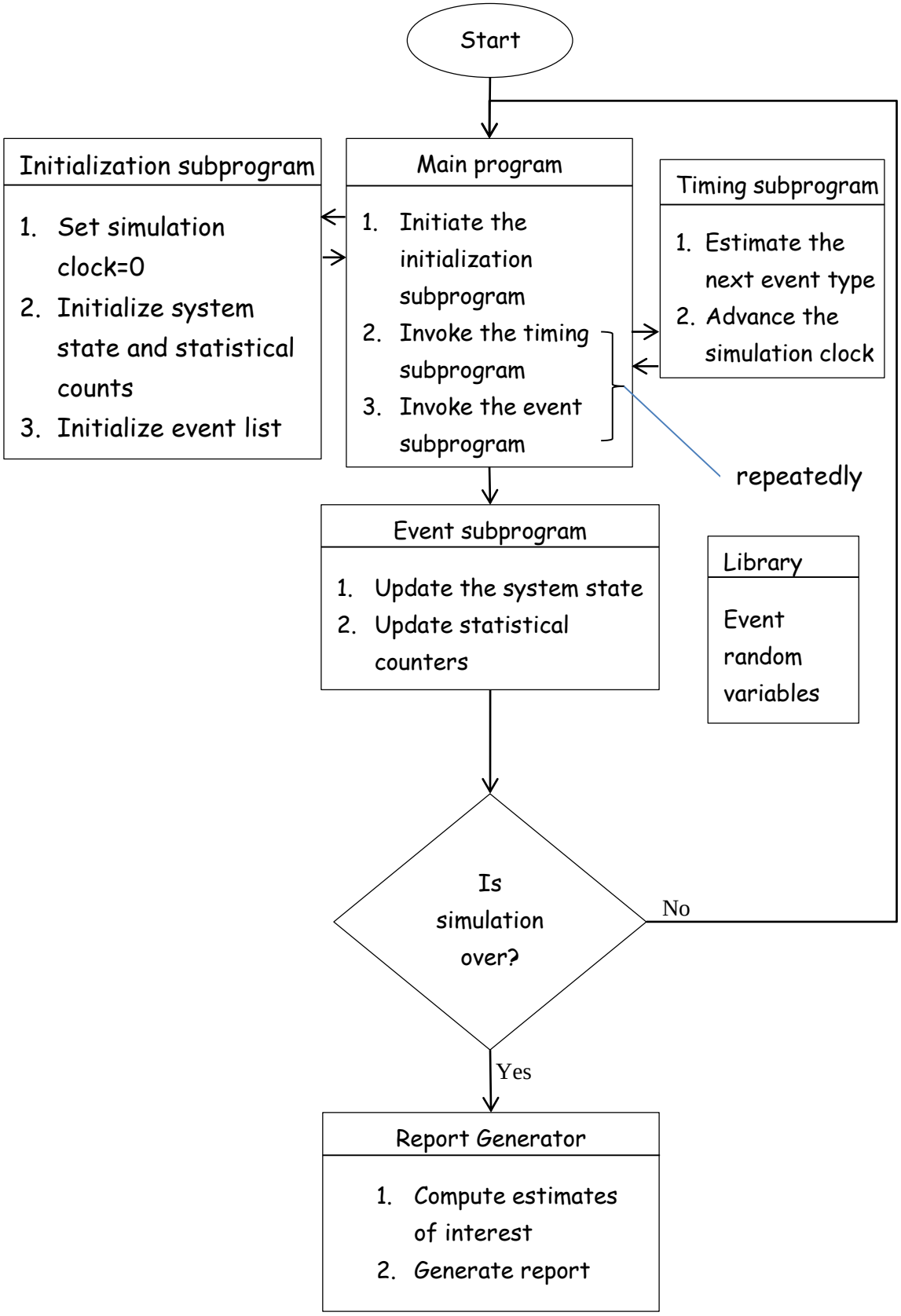
The primary use of this approach is for systems where it can reasonably be assumed that all events actually occur at one of the $n\Delta t,\ n = 0, 1, 2, \ldots$ and for an appropriately chosen Δt. For example, data in economic systems are often available only on an annual basis and it is natural in a simulation model to advance the simulation clock in increments of one year.

Components and Organization of a Discrete-Event Simulation Model

The following components will be found in most discrete-event simulation models using the next event time advance approach:

- **System state**: The collection of state variables necessary to describe the system at a particular time.
- **Simulation clock**: A variable giving the current value of simulated time.
- **Statistical counters**: Variables used for storing statistical information about system performance.
- **Initialization subprogram**: A subprogram to initialize the simulation model at time zero (0).
- **Timing subprogram**: A subprogram that determines the next event from the event list and then advances the simulation clock to the time when that event is to occur.
- **Event subprogram**: A subprogram that updates the system state when a particular type of event occurs (there is one event subprogram for each event type).
- **Library subprogram**: A set of subprograms used to generate random observations from probability distributions that were determined as part of the simulation model
- **Report generator**: A subprogram that computes estimates of the desired measures of performance and produces a report when the simulation ends.
- **Main program**: A subprogram that invokes the timing subprogram to determine the next event and then transfers control to the corresponding event subprogram to update the system state appropriately.
  The main program may also check for termination and invoke report generator when the simulation is over.

The logical relationships (flow of control) among these components is as shown below.



Simulation of a single server Queuing System

Problem statement

Consider a single server queuing system, for which the inter-arrival times A_1, A_2 … are independently identically distributed (IID) random variables.

A customer who arrives and finds the server idle enters service immediately and the service times S_1, S_2 … of the successive customers are IID random variables that are independent of the inter-arrival times.

A customer who arrives and finds the server busy joins the end of a single queue. Upon completing service for a customer, the server choses a customer from the queue (if any) in a first in first out (FIFO) manner.

The simulation will begin in the "empty" and "idle" state. At time zero (0), we will begin waiting for the arrival of the first customer which will occur after the first inter-arrival time A_1. We wish to simulate the system until a fixed number (n) of customers have completed their delays in queue i.e. the simulation will stop when the n^{th} customer enters service.

Thus the time the simulation ends is a random variable depending on the observed values of the inter-arrival and service time random variables.

To measure the performance of the system, we will look at estimates of three quantities:

First, we will estimate the expected average delay in queue of the n customers completing their delays during the simulation. We denote this by $d(n)$. On a given run of the actual simulation, the actual average delay observed for the n customers depends on the inter-arrival and service time random variable observation that happen to have been obtained. On another run of the simulation, there would probably be arrival at different times and the service times would probably also be different. This would give rise to a different value for the average of the n delays. Thus the average delay of a given random simulation is always regarded as a random variable of the simulation itself. We seek to estimate the expected value of $d(n)$. One interpretation of this is that $d(n)$ is the average of a large (infinite) number of n customers' average delays. From a single run of the simulation resulting in customer delays D_1, D_2, D_3… D_n an obvious estimator of $d(n)$ is

$\hat{d}(n) = \frac{\sum_{i=1}^{n} D_i}{n}$

which is the average of the n D_is observed in the simulation [so that $\hat{d}(n)$ could also be denoted by $\overline{D}(n)$] [a hat ($\widehat{\ }$) denotes an estimator]. The expected average number of customers in the queue is denoted by $q(n)$. This is a different kind of "average" because it is taken over "continuous" time.

Let $Q(t)$ denote the number of customers in queue at time, t for every real number $t \geq 0$.

Let $T(n)$ be the time required to observe n delays in queue.

Then for any time $0 \leq t \leq t_n$, $Q(t)$ is a non-negative integer

> Q(t) can never be a negative value because it reps no of customers - clients on queue can either be 0 or more.

Further, let P_i be the expected proportion of time that $Q(t)$ is equal to i, then a reasonable definition of $q(n) = \sum_{i=0}^{\infty} iP_i$ thus $q(n)$ is a weighted average of the possible values I for the queue length $Q(t)$ with the weights being the expected proportion of time the queue spends at each of its possible lengths.

To estimate $q(n)$ from a simulation, we replace the p_is with their estimates and get

$$\hat{q}(n) = \sum_{i=0}^{\infty} i\hat{p}_i ,$$

where $\hat{p}_i$ is the observed (rather than expected)proportion of timed during the simulation when there were i customers in the queue

If we let T_i be the total time during the simulation that the queue is of length i, then $T(n) = T(0) + T1 + T2 + \cdots$

And

$$\hat{p}_i = \frac{T_i}{T(n)}$$

and so rewrite

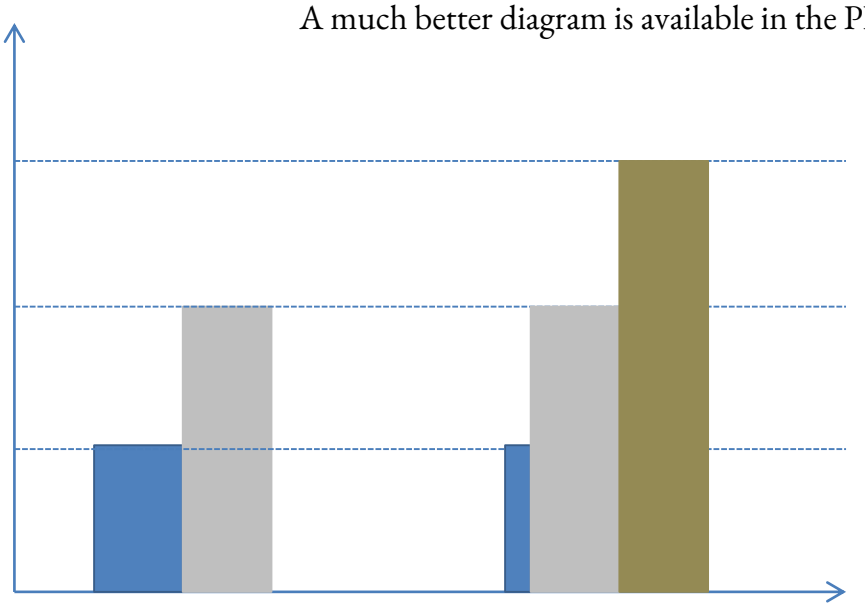
$$\hat{q}(n) = \frac{\sum_{i=0}^{\infty} iT_i}{T(n)}$$

A Numerical Illustration

Consider a single server queuing simulation model where arrivals come at 0.4, 1.6, 2.1 3.8 4.0 5.6 5.8 and 7.2, departures (since completions) occur at times 2.4 3.1 3.3 4.9 and 8.6 and the simulation ends at time $t960 = 8.6$

The figure below illustrates possible time path or realization of $Q(t)$ for the system with $n = 6$.

A much better diagram is available in the PDF lec sent [pp.16]



System state when experiment ends - Queue length is 2 [Q(t) = 2] while the Server is busy

Q(t) does not rise at 0.4(check diagram in PDF) - At 0.4 first customer arrives and starts being served. So the queue remains empty - Server is idle, Queue is empty
Therefore, describe each instance using both SERVER and QUEUE circumstances when answering such questions.

NB: Shadings are necessary when computing averages - similar shadings when the queue is of the same length

In the above figure $Q(t)$ does not count the customer in this server (if only) between times 0.4 and 1.6 since tere is one customer in the system being served even though the queue is empty i.e. $Q(t) = 0$. The same is true between times 3.1 and 3.3 and 3.8 and 4.0 and 4.9 and 5.6.

Between times 3.3 and 3.8, the system is empty and the server is idle. This is also the case at $time = 0$ and $time = 0.4$. To compute $\hat{q}(n)$, we must first compute T_is which can be read of the figure at the intervals at which $Q(t)$ is equal to 0, 1, 2, 3…

$T_0 = (1.6 - 0.0) + (4.0 - 3.1) + (5.6 - 4.9) = 3.2$ (when the queue is empty)

$T_1 = (2.1 - 1.6) + (3.1 - 2.4) + (4.9 - 4.0) + (5.8 - 5.6) = 2.3$ Departure time

$T_2 = (2.4 - 2.1) + (7.2 - 5.8) = 1.7$ queue of length 2

$T_3 = (8.6 - 7.2) = 1.4$ queue of length 3

Queue is of length 0 (happens 3 times)

Queue is of length 1 (happens 4 times so 4 entries)

$T_i = 0$ for $i \geq 4$ since the queue never grew to those lengths in this realization thus

$$\sum_{i=0}^{\infty} iT_i = (0 \times 3.2)(1 \times 2.3)(2 \times 1.7)(3 \times 1.4) = 9.9$$

And so the time estimate from this particular simulation run is:

$$\hat{q}(n) = \frac{9.9}{8.6} = 1.15$$

Each of the non-zero on the RHS of the above equation corresponds to one of the shaded areas in the figure hence the summation in this equation is just the area under the $Q(t)$ curve between the beginning and the end of the simulation.

Since area under curve is an integral

$$\sum_{i=0}^{\infty} iT_i = \int_0^{\pi n} Q(t)dt$$

The other output measure of performance for this system is a measure of how busy the server is. The expected utilization of the server is the expected proportion of time during the simulation (from time to $T(n)$) that the server is busy (not idle) and is thus a number between 0 and 1 which we donate by $U(n)$.

From a single simulation, our estimate of $U(n)$ is:

$$U(n) = observed\ proportion\ of\ time\ during\ simulation\ that\ the\ server\ is\ busy$$

Now $\widehat{U}(n)$ could be computed directly by noting the times the server changes status (idle to busy or viz-a-viz) and then doing the appropriate subtractions and division.

However, it is easier to look at this quantity as a continuous time average similar to the average queue length by defining the “busy function”):

B(t) drops to 0 at 3.3 because the server is idle as all the arrivals have been served into completion and departed.

Check PDF sent by lec for the plotted graph of the busy function - B(t)

$$B(t) = \begin{cases} 1 \; if \; the \; server \; is \; busy \; at \; time \; t \\ 0 \; if \; the \; server \; is \; idle \; at \; time \; t \end{cases}$$

And so $\widehat{U}(n)$ could be expressed as the proportion of time $B(t) = 1$

At 5.8, what is the state of the system? - Server is busy hence the arrival goes to the queue which now has a value of 2

$$\widehat{U}(n) = \frac{(3.3 - 0.4) + (8.6 - 3.8)}{8.6} = 0.90$$

This indicates that the server is busy 90% of the time during this simulation.

Another expression of $\widehat{U}(n)$ is:

$$\widehat{U}(n) = \frac{\int_0^{\pi n} B(t)dt}{T(n)}$$

Where $\widehat{U}(n)$ is the continuous average of the $B(t)$ function

Note - - - For many simulations involving "servers" of some sort, utilization statistics are quite informative in identifying bottlenecks (utilizations near a 100% coupled with heavy congestion measures for the queue leading in) or excess capacity (low utilizations). This is true if the servers are expensive items such as robots or large computers in a data processing system.

The three measures of performance are:

i. The average delay in queue $\hat{d}(n)$
ii. The time average in queue $\hat{q}(n)$
iii. The proportion of time the server is busy $\widehat{U}(n)$

The average delay in queue is an example of a **discrete time statistic** since it is defined relative to the collection of random variables D_i that have a discrete time index $i = 1,2,3 \ldots$

What is the proportion of time that the queue had 1 or more customers? - time/8.6 * 100% - where time is obtained by adding all time instances where queue is not empty [not 0] - in the example above, the answer is 62.79%

The average of number in queue and proportion of time the server is busy is an example of continuous time statistics since they are defined on the collection of random variables $\{Q(t)\}$ and $\{B(t)\}$ each of which is indexed at a continuous time parameter $l \in [0, \infty]$.

Both discrete and continuous time statistics are common in simulation and can have meaning beyond averages e.g. we might be interested in the maximum of all the delays in queue observed (discrete time statistic) or the proportion of time during the simulation that the queue contained at least 5 customers (continuous time statistic).

IID - Independent and Identically distributed - All the variables have the same probability distribution as the others and all are mutually independent

Simulation of an Inventory System

A simulation can be used to compare alternative ordering policies for an inventory system.

Problem statement

The company sells a single product. It would like to decide how many items it should have in inventory for each of the n months; the times between demands are IID exponential random variables with mean of 0.1 months. The sizes of the demands, D, are IID random variables (independent of when the demand occurs) with $D = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} \begin{matrix} w.p \\ w.p \\ w.p \\ w.p \end{matrix} \begin{pmatrix} 1/6 \\ 1/3 \\ 1/3 \\ 2/6 \end{pmatrix}$

Should be 1/6!!!

Where w.p means “with probability”

At the beginning of each month, the company reviews the inventory level and decides how many items to order from its suppliers. If the company orders z items, it incurs a cost of $k = iz$ where k is the set up cost and i is the implemental cost per item order (if z is zero, no cost is incurred).

i = no of items ordered

When an order is placed, the time required for it to arrive (called the **delivery lag or lead time**) is a random variable distributed uniformly between 0.5 and 1 month. The company uses a stationary (s, S) policy to decide how much to order i.e.

$$z = \begin{cases} S - I, & if\ I < s \\ 0, & if\ I \geq s \end{cases}$$

Where I is the inventory level at the beginning of the month

When a demand occurs, it is satisfied immediately if the inventory level is at least large as the demand.

If the demand exceeds the inventory level, the excess of demand over supply is backlogged and satisfied by future deliveries (in this case, the new inventory level is equal to the old inventory level minus the demand size resulting in a negative inventory level).

When an order arrives, it is first used to eliminate as much of the backlog (if any) as possible, the remainder of the order (if any) is added to the inventory.

Beyond the ordering cost, most real inventory systems also incur holding and shortage costs.

Check Textbook!!! Section 1.5

Why will I-(t) always be positive or zero [I-(t) >= 0]? Because I+(t) is negative, meaning I-(t) is actually -*- [double negative] which is positive

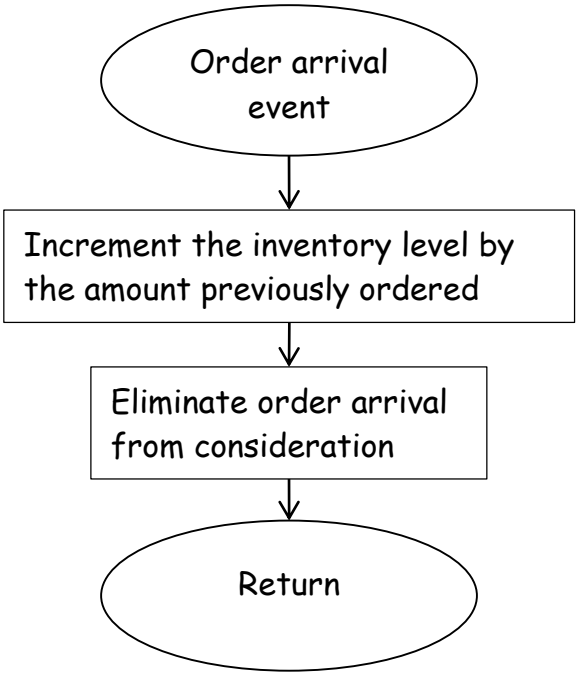
Let $I(t)$ be the inventory level at time (t). $I(t)$ Could be positive, negative or zero.

let $I^+(t) = \max\{I(t), 0\}$ be the number of items physically on hand in the inventory at time t ($I^+(t) \geq 0$).

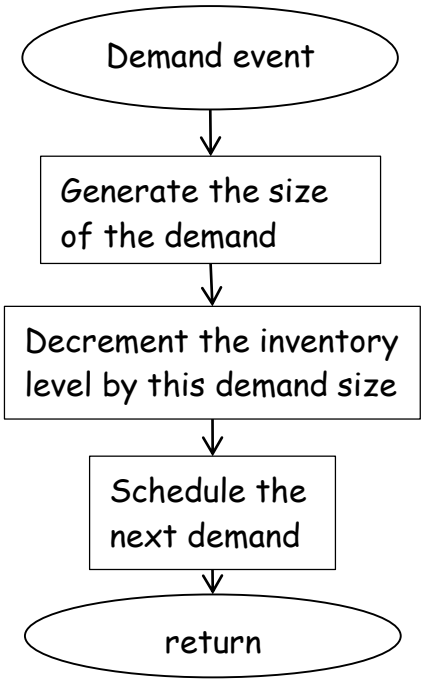
Let $I^-(t) = \max\{-I(t), 0\}$ be the backlog at time t ($I^-(t) \geq 0$).

The delivery lags are uniformly distributed but not over the interval [0, 1]. We can generate a random variate distributed over [a, b] by generating $U(0, 1)$ random number u and then returning $a + U(b - a)$.

Of the four events, only three actually involve state changes. The order arrival event is as shown in the flow chart below and must make the changes necessary when an order arrives from the supplier. The inventory level is increased by the amount of the order and the order arrival event must be eliminated from consideration.

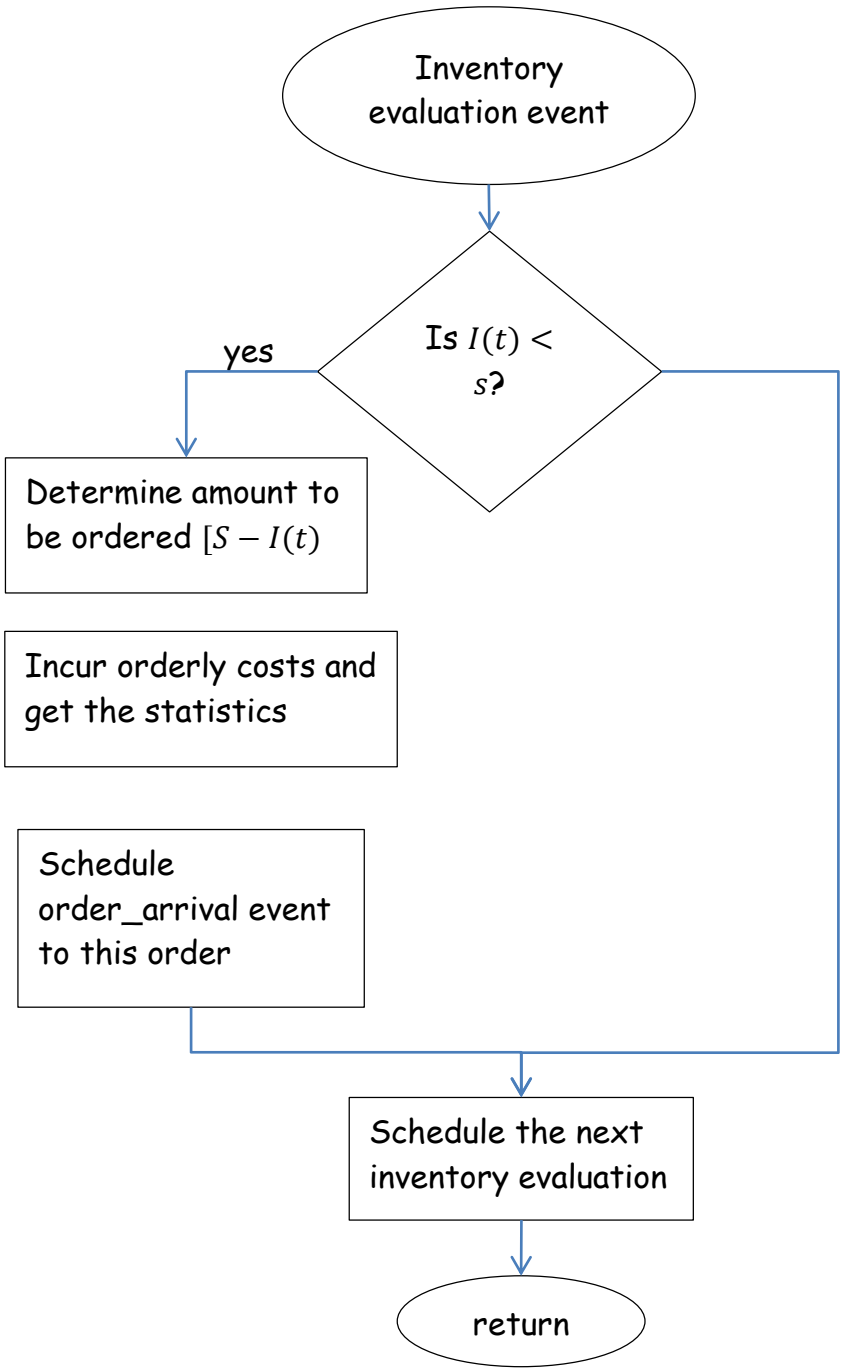


The flowchart for the demand event is as shown below:



First, the demand size is generated and then the inventory level is decremented by this amount. The time of the next demand is then scheduled into the event list.

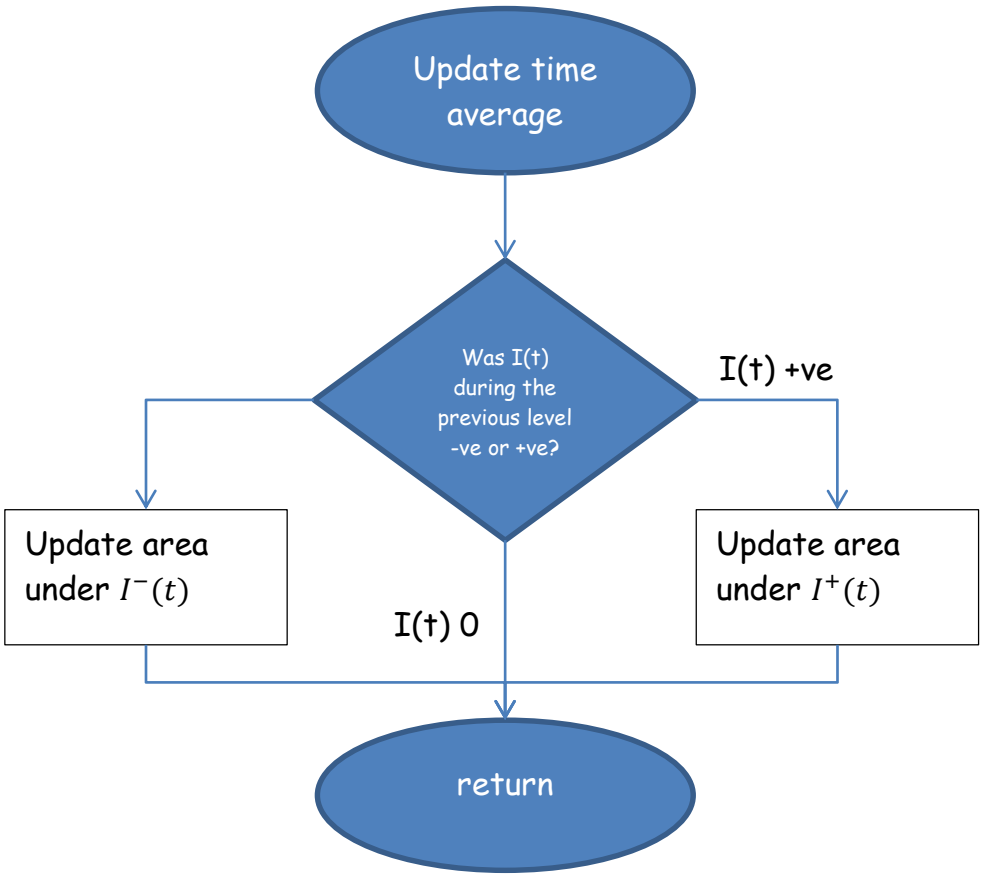
The inventory evaluation event which takes place at the beginning of each time period (month) is as shown in the flowchart below:



If the inventory level $I(t)$ at the time of evaluation is at least s, then no order is placed and nothing is done except to schedule the next evaluation into the event list.

If $I(t) < s$ for $S - I(t)$ items, this is done by storing the amount of the order $[S - I(t)]$ until the order arrives and scheduling its arrival time. In this case as well, we want to schedule the next inventory evaluation event.

A continuous time statistical accumulator flow chart is as shown below:



If the inventory level since the last event has been negative, then we have been in backlog so the area under $I^-(t)$ only should be updated.

If the inventory level has been positive, we need only update the area under $I^+(t)$.

If the inventory level has been zero, then neither update is necessary.

The code in each language for this subprogram also brings the variable for the time of the last event up to the present time. This subprogram will be invoked from the main program just after returning from the timing subprogram regardless of the event type or whether the inventory level is actually changing at this point. This provides a computationally efficient way of updating integrals for continuous time statistics.

Monte Carlo Simulation

Monte Carlo simulation is a scheme employing random numbers that are $u(0,1)$ random variates. It is used for solving certain stochastic or deterministic problems where the passage of time plays no substantive role. Monte Carlo simulations are generally static rather than dynamic.

Example

Suppose that we want to evaluate the integral

$I = \int_a^b g(x)dx$

where $g(x)$ is a real valued function that is not analytically integrable.

This is a deterministic problem and can be approached by Monte Carlo simulation method in the following way:

$let\ Y\ be\ the\ random\ variable\ (b-a)g(X)$
$where\ X\ is\ a\ continuous\ random\ variable\ distributed\ uniformly\ on\ [a,b]$
$then\ the\ expected\ value\ of\ Y\ is$

$$\begin{aligned} E(y) &= E[(b-a)g(X)] \\ &= (b-a)E[g(X)] \\ &= (b-a)\int_a^b g(x)f_x(x)dx \\ &= (b-a)\frac{\int_a^b g(x)dx}{b-a} \\ &= I \end{aligned}$$

$Where\ f_x(x) = \frac{1}{b-a}\ is\ probability\ density\ function\ of\ U(a,b)\ random\ variable.$

The problem of evaluating the integral has been reduced to one of estimating the expected value $E(Y)$. We estimate $E(Y) = I$ by the sample mean

$$\bar{Y}(n) = \frac{\sum_{i=1}^n Y_i}{n} = \frac{(b-a)\sum_{i=1}^n g(X_i)}{n}$$

$Where\ x_1, x_2, \dots, X_n\ \ are\ IID\ U(a,b) random\ variables$

$\bar{Y}(n)$ can be seen as the estimate of a rectangle that has a base of length $b-a$ and height $\frac{I}{b-a}$ which is a continuous average of $g(X)$ over $[a,b]$. Further, it can be shown that:

$E[\bar{Y}(n)] = I$

i.e. $\bar{Y}(n)$ is an unbiased estimator of I and

$$var[\bar{Y}(n)] = \frac{var(Y)}{n}$$

Assuming that $var(Y)$is finite it follows that $\bar{Y}(n)$ will be arbitrarily close to I for sufficiently large n.

As an illustration of the Monte Carlo method, suppose that we would like to evaluate the integral

$$I = \int_0^\pi \sin(x)dx$$

This can be shown by elementary calculus to have a value of two (2).

The table below shows the results of applying Monte Carlo simulation to the estimation of this integral

n	10	20	40	80	160
$\bar{Y}(n)$	2.213	1.951	1.948	1.989	1.993

Input Probability Distributions

In order to carry out simulations using random inputs such inter-arrival times or demand sizes, we have to specify their probability. For example in the simulation of the single server, the inter-arrival times were taken to be IID exponential random variables with mean probability of 1 minute. The demand sizes in the inventory simulation were specified to be 1, 2, 3 or 4 items with respective probabilities 1/6, 1/3, 1/3, and 1/6. Then given that the input random variables to a simulation model follow a particular distribution, the simulation proceeds through time by generating random values from these distributions.

Almost all real systems contain one or more sources of randomness. It is necessary to represent each source of system randomness by a probability distribution in the simulation model.

Table 1: Sources of Randomness for Common Simulation Applications

Type of System	**Source of Randomness**
Manufacturing systems	• Processing time. • Machine operating times before a down time • Machine repair time. • Etc.
Computer based systems	• Inter-arrival times of jobs. • Job types. • Processing requirements of jobs.
Communication systems	• Inter-arrival times of messages. • Message types. • Message lengths.
Defence-related systems	• Arrival times and payloads if missiles or airplanes. • Outcome of an engagement. • Miss distances for munitions.

The following example shows that failure to choose the “correct” distribution can affect the accuracy of a model’s result:

A single-server queuing system has exponential inter-arrival times with a mean of one minute. Suppose that 200 service times are available from the system but their underlying probability distribution is unknown, we can fit the best exponential, gamma, weibull, lognormal and normal distributions to the observed service time data. We then make 100 independent simulation runs (i.e. different random numbers were used for each run) of the queuing system using each of the five fitted distributions. Each of the 500 simulation runs were continued until 1000 delays in queue were collected.

A summary of the results from these simulation values is given in the table below:

Service-time distribution	**Average delay in queue**	**Average number in queue**	**Proportion of delays $\geq$ 20**
Exponential	6.71	6.78	0.064
Gamma	4.54	4.60	0.019
Weibull	4.36	4.41	0.013
Lognormal	7.19	7.30	0.078
Normal	6.04	6.13	0.045

Note that in column two, the average of the 100,000 delays is given for each of the service time distributions. The Weibull distribution actually provides the best model for the service time data. Thus the average delay for the real system should be close to 4.36 minutes. On the other hand, the average delays for the normal and lognormal distributions are 6.04 and 7.19 minutes respectively corresponding to output errors of 39% and 65%. Thus the choice of probability distributions can have a large impact on the simulation output and potentially on the quality of the decisions made with the simulation results.

Using Collected Data to Specify a Distribution

It is possible to collect data on an input random variable of interest. This data can be used in one of the following approaches to specify a distribution:

i. The data values themselves are used directly in the simulation e.g. if the data represents service times, then one of the data values is used whenever a service time is needed in the simulation. This is called a ***trace driven simulation***.
ii. The data values are used to define an empirical distribution function in some way. If this data represents service times, we would sample from this distribution when a service time is needed in the simulation.
iii. Standard techniques of statistical inferences are used to 'fit' a theoretical form such as exponential or poison to the data and to perform a hypothesis test to determine the goodness of fit. If a particular theoretical distribution with certain values for its parameters is a good model for the service time data, then we will sample from this distribution when a service time is needed in the simulation.

Two drawbacks of approach one are that:

- The simulation can only reproduce what has happened historically.
- There is seldom enough data to make all the desired simulation runs.

Approach two avoids these shortcomings since at least for continuous data any value between the minimum and maximum observed data points can be generated. Thus approach two is generally preferable over approach one. However, approach one is recommended for model validation when comparing model output for an existing system with the corresponding output for the system itself.

If a theoretical distribution can be found that fits the observed data, reasonably well (approach 3) then this will, generally, be preferable to using an empirical distribution (approach two) for the following reasons:

- An empirical distribution function may have certain irregularities particularly if only a small number of data values is available. But a theoretical distribution "smooths-out" the data and may provide information for the overall underlying distribution.
- If empirical distributions are used in their usual way, it is not possible to generate values outside the range of the observed data in the simulation. This is unfortunate since many measures of performance for simulated systems depend heavily on the possibility of an "extreme" event occurring e.g. generation of a very large service time. With a fitted theoretical distribution, values outside the range of the observed data can be generated.
- There may be compelling physical reasons in some situations for using a certain theoretical distribution form as a model for a particular input random value. Even when we have this kind of information, it is good to use observed data to provide emphirical support for the use of this particular distribution.
- A theoretical distribution is a compact way of representing a set of data values. Conversely, if n data values are available for the provided distribution, then 2n (data and cumulative probabilities) must be entered and stored in the computer to represent an emphirical distribution in many simulation simulations. Thus the use of an empirical distribution will be cumbersome if the data set is large.

Random Number Generators

Properties of a Good Arithmetic Random Number Generator

A good arithmetic random number generator should possess the following properties:

i. The numbers produced should appear to be ***distributed uniformly*** on $[0,1]$ and should not exhibit any correlation with each other; otherwise simulation results may be completely invalid.
ii. The generator should ***be fast*** to avoid the need for a lot of storage.
iii. It should be ***possible to reproduce a given stream*** of random numbers exactly. This makes debugging and verification of the program easier. Also, it may be necessary to use identical random numbers in simulating different systems in order to obtain more precise comparisons.
iv. The generation should be able to ***produce several separate streams*** of random numbers. **A stream *is a sub-segment of the numbers produced by the generator with one stream beginning where the previous stream ends***. Different streams can be seen as separate and independent generators provided that the whole stream is not used.

The usercan dedicate a particular stream to a particular source of randomness in the simulation. E.g. in the single server queuing model, stream one can be used for generating inter-arrival times and stream two for generating service times. Using separate streams for separate purposes facilitates reproducibility and comparability of simulation results.

v. The generator should be ***portable*** i.e. produce the same sequence of random numbers for all standard computers.

i. Mid-square Method

Start with a four-digit positive integer, Z_0 and square it to obtain an integer with up to eight digits; if necessary, append zeros to the left to make it exactly eight digits. Take the middle four digits of this eight digit number as the next four digit number, Z_1.

Place a decimal point at the left of Z_1 to obtain the first $U(0,1)$ random number U_1. Let Z_2 be the middle four digits of Z_1^2 and let U_2 be Z_2 with a decimal point at the left and so on e.g.

i	Zi	Ui	Z_i^2
0	7182	-	51581124
1	5811	0.5811	33767721
2	7677	0.7677	58936329
3	9363	0.9363	87665769
4	6657	0.6657	44315649
5	3156	0.3156	09960336
6	9603	0.9603	92217609

ii. Linear Congruential Generators (LCGs)

In LCGs, a sequence of integers $Z_1, Z_2, \ldots$ is defined by the recursive formula,

$$Z_i = (aZ_{i-1} + c)(mod\ m),$$

$$where\ \begin{array}{lll} m & - & modulus \\ a & - & multiplier \\ c & - & increment \\ Z_0 & - & the\ seed\ (non\ negative\ integer) \\ \therefore & \rightarrow & 0 \leq Z_i \leq m-1 \end{array}$$

The desired random number, U_i is

$$U_i = \frac{Z_i}{m}$$

Also $a < m, 0 < m, c < m\ and\ Z_0 < m$

Example: consider the LCG defined by $m = 16, a = 5, c = 3\ and\ z_0 = 7$

i	Z_i	U_i
0	7	
1	6	0.375
2	1	0.063
3	8	0.500
4	11	0.688

5	10	0.625
6	5	0.313
7	12	0.750
8	15	0.938
9	14	0.875
10	9	0.563
11	0	0.000
12	3	0.188
13	2	0.125
14	13	0.813
15	4	0.250
16	7	0.438
17	6	0.375
18	1	0.063
19	8	0.500
20	11	0.688

The LCG defined above has full period if and only if the following three conditions hold:

i. The only negative integer that exactly divides both m and c is 1.
ii. If q is a prime number that divides m then q divides $a - 1$.
iii. If 4 divides m then 4 divides $a - 1$.

iii. Tausworth Generators

They are related to cryptographic methods and operate directly on bits to form random numbers. They define a sequence b_1, b_2, … of binary digits by the recurrence relation,

$$b_i = (c_1 b_{i-1} + c_2 b_{i-2} + \cdots + c_q b_{i-q})(mod\ 2),$$
$$where\ c_1, c_2, \ldots, c_q are\ constants\ that\ are\ either\ 0\ or\ 1, the\ minimum\ period\ is\ 2^q - 1$$

Most Tausworth generators are such that only two of the c_i coefficients are non-zero. The above equation becomes:

$$b_i = (b_{i-r} + b_{i-q})(mod\ 2)$$

$$For\ integers\ r\ and\ q\ satisfying\ o < r < q$$

Addition of mod 2 is equivalent to the XOR instruction on bits. The above equation can be expressed as:

$$b_i = \begin{cases} 0, if\ b_{i-r} = b_{i-q} \\ 1, if\ b_{i-r} \neq b_{i-q} \end{cases}$$

To initialize the $\{b_i\}$ sequence, the 1st q bits must be specified.

Example: let $r = 3$, and $q = 5$ in the above equation for $i \geq 6$, b_i is the XOR of b_{i-3} with b_{i-5}. The first 42 bits are:

1111 1000 1101 1101 0100 0010 0101 1001 1111 0001 10…

The period of the bits is equal to

$31 = 2^5 - 1$

To obtain the $U(0,1)$ random numbers, string together the consecutive bits to form an l-bit binary integer between zero and $2^l - 1$ then divide by 2^l. For $l = 4$, the sequence of random numbers is:

$$\frac{15}{16}, \frac{8}{16}, \frac{13}{16}, \frac{13}{16}, \frac{4}{16}, \frac{2}{16}, \frac{5}{16}, \frac{9}{16}, \frac{15}{16}, \frac{1}{16}, \ldots$$

Tausworth generators offer a number of advantages over LCGs which include:

a) They are independent of the computer used and its word size and one can readily obtains periods of great lengths such as $2^{521} - 1 > 10^{156}$ or more.
b) They have appealing theoretical properties.

Generating Random variates

A simulation that has any random aspects must involve generating random variates from probability distributions. The term 'generating random variates' refers to the activity of obtaining an observation on (or realization of) a random variable from the derived distribution.

These distributions are specified as a result of fitting some appropriate distributional form e.g. exponential, gamma or poison to observed data.

The basic ingredient needed for every method of generating random variates from any distribution or random process is a source of IID $U(0,1)$ random numbers. There are several alternative algorithms for generating random variates from a given distribution.

Factors to Consider

The following factors are important when choosing which algorithm to use in a particular simulation study:

i. Exactness

The algorithm should result in random variates with exactly the desired distribution within the un-avoidable external limitations of machine accuracy and exactness of the $U(0,1)$ random number generator.

ii. Efficiency

The algorithm should be efficient in terms of both storage space and execution time. Some algorithms require storage of large number of constants or large tables. The execution time has two factors:

a. **Marginal execution time**: generation of each random variable is done in a short time.
b. **Setup time**: any initial computing required on particular distribution parameters.

iii. Overall complexity

These include conceptual as well as implementation factors. The potential gain in efficiency that might be experienced by using a more complicated algorithm should be carefully weighed against the extra effort needed to understand and implement it.

iv. Robustness

Some algorithms rely on a source of random variables from distributions other that $U(0,1)$. Also a given algorithm may be efficient for some parameter values but costly for others. A good algorithm should be efficient for a wide range of parameter values (robust).

v. Ability to use variance-reduction techniques

Two commonly used techniques are:

- Common random variables
- Antithetic variates (contrast to common)

These techniques require synchronization of the basic $U(0,1)$ input random variates used in the simulation of the system under study. This synchronization is more easily accomplished for certain types of random variate generation algorithms. In particular, the universe transform method can be very helpful in facilitating the desired synchronization and variance reduction.

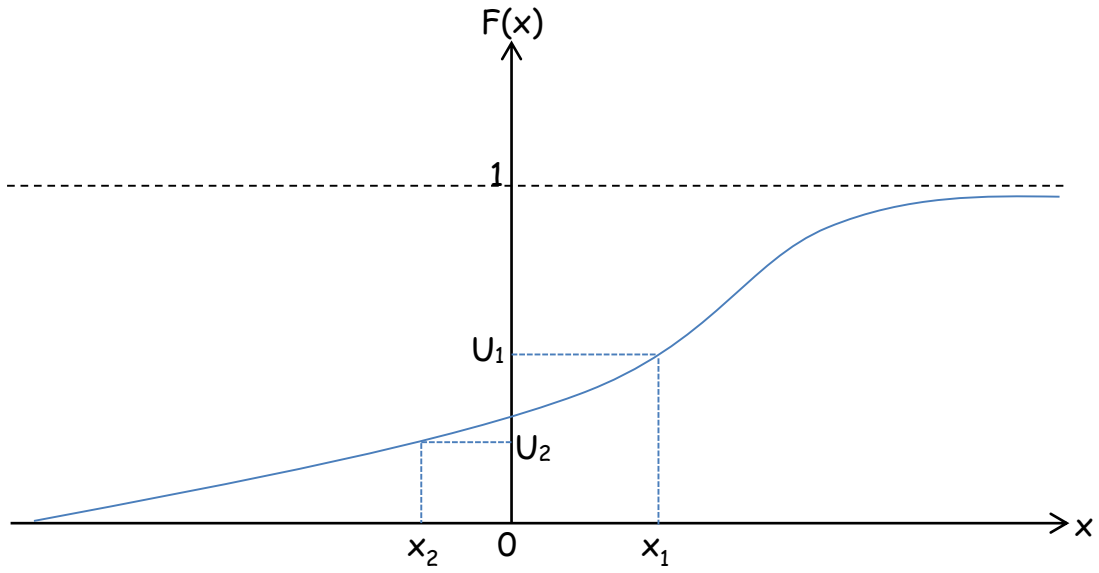
The Inverse Transform Method

Suppose that we wish to generate a random variate X that is continuous and has a distribution function, F that is also continuous and strictly increasing, when $0 < F(x) < 1$ i.e. if $x_1 < x_2 \; and \; 0 < F(x_1) \leq F(x_2) < 1$ then actually $F(x_1) < F(x_2)$.

Let F^{-1} denote the inverse of function F, then an algorithm for generating a random variate X having distribution function F is:

i. Generate $U \sim U(0,1)$ Note: ~ is read “is distributed as”
ii. Return $X = F^{-1}(U)$

$F^{-1}(U)$ will always be defined since $0 \leq U < 1$ and the range of F is $[0,1]$.



The random variable corresponding to this distribution function can take on either positive or negative values. This is determined by the particular value of U. to show that the value X returned by this method has the desired distribution F, we show that $\forall x \in \mathbb{R},\ p(X \leq x) = F(x)$. Since F(x) is invertible, we have $P(X \leq x) = P(F^{-1}(U) \leq x) = P(U \leq F(x) = F(x)$ where the last equality follows since $U\ \sim (0,1)\ and\ 0 \leq F(x) \leq 1.$

Example

Let X have the exponential distribution with mean β. The distribution function is

$$F(x) = \begin{cases} 1 - e^{-\frac{x}{\beta}}, & x \geq 0 \\ 0, & otherwise \end{cases}$$

To find F^{-1} we set $U = F(x)$ and solve for x to obtain $F^{-1}(U) = -\beta \ln(1-U)$

$$U = 1 - e^{-\frac{x}{\beta}}$$
$$\ln(U-1) = \ln\left(-e^{-\frac{x}{\beta}}\right)$$

$You\ can\ use\ U\ instead\ of\ 1-U\ since\ U\ and\ 1-$
$U\ \ have\ the\ same\ U(0,1)\ distribution.This\ saves\ a\ subtraction.$

To generate the desired random variate, we first generate $U \sim U(0,1)$ and then let $X = -\beta \ln U$

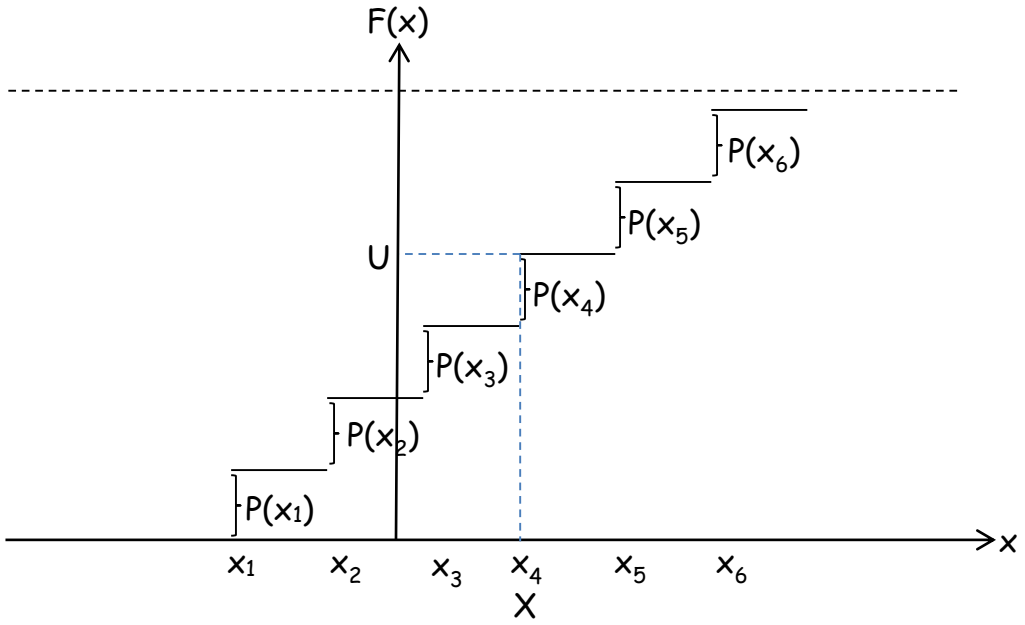
The inverse transform method can also be used when X is discrete. Here, the distribution function is

$$F(x) = F(X \leq x) = \sum_{x_i \leq x} P(x_i)\,,$$
$$where\ P(x_i)\ is\ the\ probability\ mass\ function\ P(x_i) = P(X = x_i)$$

Assume that X can take on only the values $x_1, x_2, \ldots, x_n$ where $x_1 < x_2 < \cdots < x_n$.

Then the algorithm is as follows

i. Generate $U \sim U(0,1)$.
ii. Determine the smallest positive integer I such that $U \leq F(X_I)$ and return $X = x_i$.



To verify that the discrete inverse transform method is valid, we need to show that $P(X = x_i) = P(x_i)\ \forall i.$

$For\ i = 1, x = x_1\ iff\ u \leq P(x_1) = P(x_1)$

Since we have arranged x_is in increasing order

Since $U \sim U(0,1), P(X = x_1) = P(x_1)$ as desired

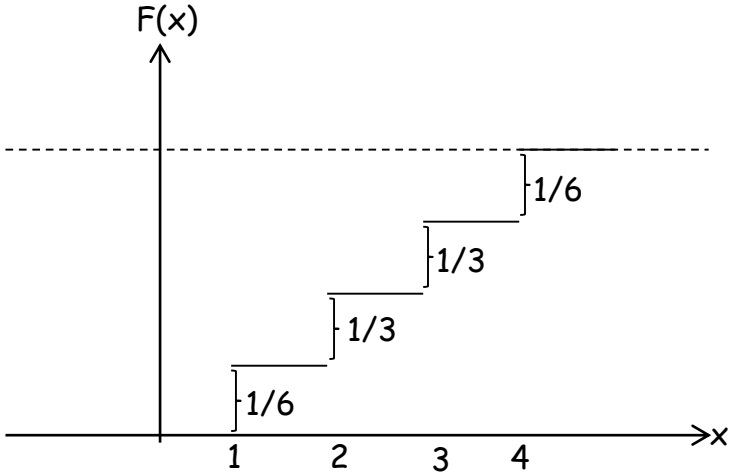
For $i \geq 2$ the algorithm sets $X = x_i\ iff\ F(X_{i-1} < U < F(x_i)$ since the I chosen by the algorithm is the smallest positive integer such that $U \leq F(x\_i)$. further, since $U \sim U(0,1) and\ 0 \leq F(x_{i-1}) < F(x_i) \leq 1,$

$$P(X = x_i) = P[F(x_{i-1}) < U < F(x_i)]$$

$$= F(x_i) - F(x_{i-1}) = P(x_i)$$

Example

In the inventory example, the demand size random variable X is discrete taking on the values $1, 2, 3, or\ 4$ with respective probabilities, $\frac{1}{6}, \frac{1}{3}, \frac{1}{3}, \frac{1}{6}$. The distribution function is as follows



To generate an X, first generate $U \sim U(0,1)$ and set X to either $1, 2, 3, or\ 4$ depending on the sub-interval in $[0,1]$ into which U falls.

If $U \leq \frac{1}{6}$, $\quad then\ let\ x = 1$

If $\frac{1}{6} < U \leq \frac{1}{2}$, $\quad let\ x = 2$

If ½ $< \frac{U\backslash leq5}{6}$, $let\ x = 3$

If $\frac{5}{6} < U \leq 1$, $\quad let\ x = 4$

Some simulation Software

1. GPSS – general purpose simulation system
2. SIMAN – simulation analysis
3. SIMSCRIPT – simulation scripting
4. SLAM – simulation language for alternative modelling
5. MODSIM – based on OOP
6. AUTOMOD
7. WITNESS
8. SIMFACTORY
9. XCELL
10. PROMODEL

(6–10: simulation package for manufacturing systems.)

11. NETWORK – simulation for computer systems and LAN
12. COMMET – simulation for wide area telecommunication networks
13. POWERSIM
14. VENSIM

(13–14: used in system dynamics)

15. SIMULA – object oriented