Discrete Event Simulation

Discrete Event Simulation

- In discrete-event simulation, the operation of a system is represented as a chronological sequence of events. Each event occurs at an instant in time and marks a change of state in the system [1]. For example, if an elevator is simulated, an event could be "level 6 button pressed", with the resulting system state of "lift moving" and eventually (unless one chooses to simulate the failure of the lift) "lift at level 6".
- A common exercise in learning how to build discrete-event simulations is to model a queue, such as customers arriving at a bank to be served by a teller. In this example, the system entities are CUSTOMER-QUEUE and TELLERS. The system events are CUSTOMER-ARRIVAL and CUSTOMER-DEPARTURE. (The event of TELLER-BEGINS-SERVICE can be part of the logic of the arrival and departure events.) The system states, which are changed by these events, are NUMBER-OF-CUSTOMERS-IN-THE-QUEUE (an integer from 0 to n) and TELLER-STATUS (busy or idle). The random variables that need to be characterized to model this system stochastically are CUSTOMER-INTERARRIVAL-TIME and TELLER-SERVICE-TIME.
- A number of mechanisms have been proposed for carrying out discrete-event simulation, among them are the event-based, activity-based, process-based and three-phase approaches (Pidd, 1998). The three-phase approach is used by a number of commercial simulation software packages, but from the user's point of view, the specifics of the underlying simulation method are generally hidden.

The Components of a Discrete-Event Simulation

- In addition to the representation of system state variables and the logic of what happens when system events occur, discrete event simulations include the following:

- Clock

- The simulation must keep track of the current simulation time, in whatever measurement units are suitable for the system being modeled. In discrete-event simulations, as opposed to real time simulations, time ‘hops’ because events are instantaneous – the clock skips to the next event start time as the simulation proceeds.

Components of a Discrete-Event Simulation

Events List

- The simulation maintains at least one list of simulation events. This is sometimes called the pending event set because it lists events that are pending as a result of previously simulated event but have yet to be simulated themselves. An event is described by the time at which it occurs and a type, indicating the code that will be used to simulate that event. It is common for the event code to be parameterized, in which case, the event description also contains parameters to the event code.
- When events are instantaneous, activities that extend over time are modeled as sequences of events. Some simulation frameworks allow the time of an event to be specified as an interval, giving the start time and the end time of each event.

Components of a Discrete-Event Simulation

Events List …

- Single-threaded simulation engines based on instantaneous events have just one current event. In contrast, multi-threaded simulation engines and simulation engines supporting an interval-based event model may have multiple current events. In both cases, there are significant problems with synchronization between current events.
- The pending event set is typically organized as a priority queue, sorted by event time.[2] That is, regardless of the order in which events are added to the event set, they are removed in strictly chronological order. Several general-purpose priority queue algorithms have proven effective for discrete-event simulation,[3] most notably, the splay tree. More recent alternatives include skip lists and calendar queues.[4]
- Typically, events are scheduled dynamically as the simulation proceeds. For example, in the bank example noted above, the event CUSTOMER-ARRIVAL at time t would, if the CUSTOMER_QUEUE was empty and TELLER was idle, include the creation of the subsequent event CUSTOMER-DEPARTURE to occur at time t+s, where s is a number generated from the SERVICE-TIME distribution.

Components of a Discrete-Event Simulation

Random-Number Generators

- The simulation needs to generate random variables of various kinds, depending on the system model. This is accomplished by one or more Pseudorandom number generators. The use of pseudorandom numbers as opposed to true random numbers is a benefit should a simulation need a rerun with exactly the same behavior.
- One of the problems with the random number distributions used in discrete-event simulation is that the steady-state distributions of event times may not be known in advance. As a result, the initial set of events placed into the pending event set will not have arrival times representative of the steady-state distribution. This problem is typically solved by bootstrapping the simulation model. Only a limited effort is made to assign realistic times to the initial set of pending events. These events, however, schedule additional events, and with time, the distribution of event times approaches its steady state. This is called bootstrapping the simulation model. In gathering statistics from the running model, it is important to either disregard events that occur before the steady state is reached or to run the simulation for long enough that the bootstrapping behavior is overwhelmed by steady-state behavior. (This use of the term bootstrapping can be contrasted with its use in both statistics and computing.)

Components of a Discrete-Event Simulation

Statistics

- The simulation typically keeps track of the system's statistics, which quantify the aspects of interest. In the bank example, it is of interest to track the mean waiting times.

Ending Condition

- Because events are bootstrapped, theoretically a discrete-event simulation could run forever. So the simulation designer must decide when the simulation will end. Typical choices are “at time t” or “after processing n number of events” or, more generally, “when statistical measure X reaches the value x”.

Simulation Engine Logic

The main loop of a discrete-event simulation is something like this:

- Start
  - Initialize Ending Condition to FALSE.
  - Initialize system state variables.
  - Initialize Clock (usually starts at simulation time zero).
  - Schedule an initial event (i.e., put some initial event into the Events List).
- "Do loop" or "While loop"

  While (Ending Condition is FALSE) then do the following:
  - Set clock to next event time.
  - Do next event and remove from the Events List.
  - Update statistics.
- End
  - Generate statistical report.

References

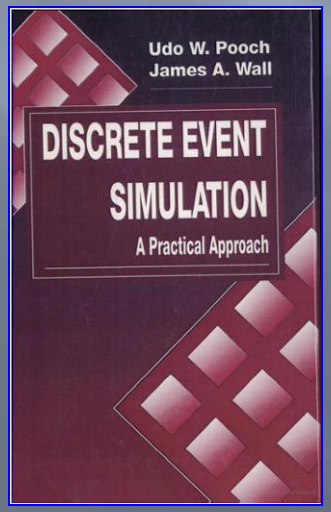
1. Stewart Robinson (2004). Simulation - The practice of model development and use. Wiley.
2. Douglas W. Jones, ed. Implementations of Time, Proceedings of the 18th Winter Simulation Conference, 1986.
3. Douglas W. Jones, Empirical Comparison of Priority Queue and Event Set Implementations, Communications of the ACM, 29, April 1986, pages 300-311.
4. Kah Leong Tan and Li-Jin Thng, SNOOPy Calendar Queue, Proceedings of the 32nd Winter Simulation Conference, 2000
5. Byrne, James; Heavey, Cathal; Byrne, P.J. (2006). "SIMCT: An Application of Web Based Simulation.". Proceedings of the 2006 Operational Research Society (UK) 3rd Simulation Workshop (SW06), 28-29th March, Royal Leamington Spa, UK..

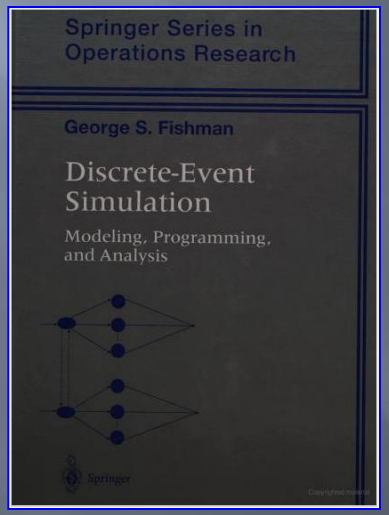
Further reading

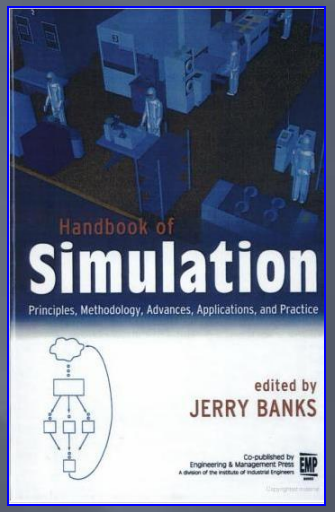
- Michael Pidd (1998). Computer simulation in management science - fourth edition. Wiley.
- Jerry Banks, John Carson, Barry Nelson and David Nicol (2005). Discrete-event system simulation - fourth edition. Pearson.
- Averill M. Law and W. David Kelton (2000). Simulation modeling and analysis - third edition. McGraw-Hill.
- Bernard P. Zeigler, Herbert Praehofer and Tag Gon Kim (2000). Theory of modeling and simulation: Integrating discrete event and continuous complex dynamic systems - second edition. Academic Press.
- Roger W. McHaney (1991). Computer Simulation: A Practical Perspective. Academic Press.
- William Delaney, Erminia Vaccari (1988). Dynamic Models and Discrete Event Simulation. Dekker INC.

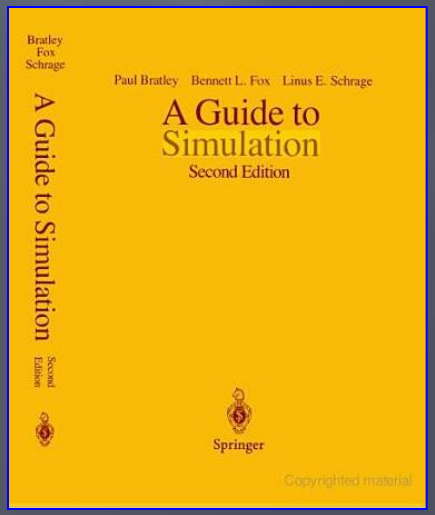
Books

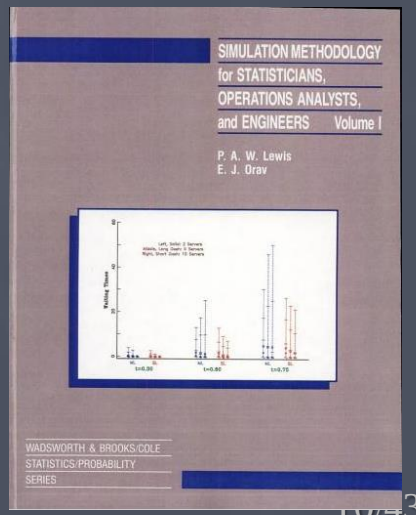
1. *Udo W. Pooch, James A. Wall,* Discrete event simulation: a practical approach.
2. *George S. Fishman*, Discrete-event simulation: modeling, programming, and analysis.
3. *Jerry Banks*, Handbook of simulation: principles, methodology, advances, applications, and Practice.
4. *Paul Bratley, Bennett L. Fox, Linus E. Schrage*, A guide to simulation.
5. *Peter A. W. Lewis, Endel John Orav,* Simulation methodology for statisticians, operations analysts, and engineers.











Bibliography

1. *Steve Park and Larry Leemis*, College of William and Mary, Discrete-Event Simulation: A First Course - PowerPoint Presentation. www.cs.wm.edu/~esmirni/Teaching/cs526/DESAFC-1.1.ppt .
2. *A. Udaya Shankar*, Discrete-Event Simulation - Department of Computer Science, University of MarylandCollege Park, Maryland 20742January, 1991. http://www.cs.umd.edu/~shankar/711-S98/DE-simulation.ps.
3. *Thomas J. Schriber, Daniel T. Brunner*, INSIDE DISCRETE-EVENT SIMULATION SOFTWARE:HOW IT WORKS AND WHY IT MATTERS, Proceedings of the 1997 Winter Simulation Conference, ed. S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.22.2215&rep=rep1&type=pdf
4. *Bikram Sharda, Scott J. Bury*, A DISCRETE EVENT SIMULATION MODEL FOR RELIABILITY MODELING OF A CHEMICAL PLANT , Proceedings of the 2008 Winter Simulation Conference. http://www.extendsim.com/downloads/papers/sols_papers_chemPlant.pdf

Model Validation

- What do we mean by "Model Validation"?
- There is no way in which we can PROVE that our simulator (model) matches reality - all we could prove is that it does NOT.
- What are the stages that we go through in order to determine that the simulator is - as far as we can tell - acceptable?

Design Phase

This consists of using

A) our intuitive and mathematical understanding of the process we are trying to model;

B) our experience in designing computer models of reality;

C) techniques of software engineering developed to aid the transition from A) and B) to a codable specification. These might simply include modular decomposition and top-down design, or more sophisticated specification languages and environments.

Validation of Design

a) Conceptual Phase: determine the logical flow of the system, and formulate the relationships between the various subsystems. Identify the factors likely to influence the performance of the model and decide how you will track them.

Validation by **review**: have a third party examine the design in detail.

Validation by **tracing from output to input**: this is just the opposite from the usual direction, and might indicate gaps and misunderstandings.

Validation of Design

b) Implementation Phase: selecting procedures for the model; coding the model; using the model.

Validation by checkpoints and milestones: at the end of the model design phase; at the end of the implementation of each module; at the integration of two or more modules.

The latter two require that individual module tests have been identified and designed, and that the flow of information between modules can be examined in detail. This blends into the next phase...

Verification Phase

Does the system that has been coded meet the specifications?

At this point, the design might well be "wrong", and this phase is **not** meant to catch design mistakes. It should catch any deviations of the actual implementation from the design specifications.

How: code walk-throughs, careful tracing of module interfaces and module dependencies, tests that were designed at specification time for this purpose.

Validation Phase

At this point we have a running program that we believe satisfies the design specifications and - because the design was achieved by people with experience in designing such simulation models using some design validation methods - is probably close enough to reality so that its inputs and outputs can be meaningfully compared with those of the "real thing".

We want to establish that the model is an "adequately accurate" representation of the reality we are interesting in modeling. If this cannot be established, our model will be useless for both explanation and prediction - both of which are goals of "good models".

What does Validation Consist of?

A) Comparison of results of simulation model with historical data;

B) Use of the simulation model to predict behavior of the real system, and compare prediction with actual behavior.

Both of these methods are standard when it comes to validating a physical theory: the theory must not contradict any "well-known facts"; it must provide new predictions (not available through established theory) that are subject to falsification by experiment.

How do you determine that the system has been validated?

Even in Physics, theoretical predictions usually do not match experimental results **exactly**.

Solution: run statistical tests, until you can conclude, from statistical considerations, that the probability of some competing explanation being true is smaller than a preassigned quantity.

This means that the system must be designed for multiple **controlled** observations and the collection of all appropriate data (= measures of performance).

Simulation Run:

This is an uninterrupted recording of the simulation system's performance given a specified combination of controllable variables: range of values, values of some parameter, queue arrival distributions, etc.

Simulation Duplication:

This is a recording of the simulation system's performance given the same or **replicated** conditions and/or combinations, but with different random variates.

This is also related to "regression testing": a correct model (or piece of software) must satisfy certain input/output relations. Any time the software is modified it must pass all the old tests; if it is "improved", it must pass all the old tests plus appropriate new ones.

Simulation Observation:

This is a simulation run or a segment of a simulation run that is sufficient for estimating the value of each of the performance measures.

Steady State or Stable State:

This state of a simulation system is achieved when successive system performance measurements are statistically indistinguishable - the second one provides no new information about the future behavior of the system.

Steady State corresponds, for example, to the constant solution of a differential equation. A Stable Steady State corresponds to a stable constant solution or, more likely, to an asymptotically stable one.

How do we identify a steady state of some performance measure?

By deciding on a "small" positive value, say ε, and deciding that we have reached steady state when, over a "long" period of time, the performance measure (or some appropriate function of it) has remained within $\pm\varepsilon$ of some value.

"small" and "long" are terms that are meaningful only in the context of the particular model being studied.

It may also be possible to actually predict, from the model, the constant value.

Transient State:

The time (and set of performance measurements) that correspond to the initial conditions becoming insignificant to the future behavior of the system.

Some systems may have no Steady State, so that no termination for a Transient State could be identified.

Most simulations appear to be interested in examining the steady state behavior. This is appropriate under some conditions - and in the case where modeling based on discrete queueing theory is being used: most such queueing theory results depend on our being able to obtain steady state predictions.

There are other situations where the transient state may be the most important, since that is where system queues (or buffers) might be overloaded. The "usual solution" is to provide enough system capacity to handle "most" transients. This requires that we find good prediction for the size of all "transients" in the measures of performance.

A further problem with transients is that the actual behavior depends on the exact initial conditions of the system.

As it turns out, some systems may have very complex initial conditions - possibly extending over long periods of time.

Some systems may exhibit very complex behavior - so that some initial conditions will tend to one steady state (or, more generally, an "attractor" which may exhibit a very complex geometry) while others will lead to another all without varying the parameters of the system. These are so-called bistable (or multistable) systems.

Studying transient behavior will thus require a large number of runs; a possibly complex geometric analysis of the "phase-space" of the system; the ability to describe and set arbitrary initial conditions for the system; and sophisticated statistical techniques to determine means, variances and other statistics of the relevant performance measures.

If one wishes to avoid dealing with transient behavior, one must be able to specify initial conditions near steady state: this may require being able to "load the model" with a specific history.

In simple cases this might just mean deferring data collection for a period of time; in others it might mean that consistent performance measure values have to be synthesized over a long enough time period and that these measures have to be "inserted" into the behavior of the model.

What is Validation?

How accurately is the simulation model representing the actual physical system being simulated?

Several terms have been introduced, corresponding to techniques that help in answering this question. One must always remember that there is no way to prove that the model is a faithful reproduction of reality.

All we can prove is that it is not. But we might be able to set up enough different experiments so that passing of all the experimental tests will allow us to conclude that the probability the model is inaccurate is very small.

Internal Validity.

This is affected by variability due to internal "noise" effects: stochastic models with high variance due to internal processing will provide outputs whose analysis may not be very useful: are the changes in the outputs due to the model or to incidentals of the implementation? (e.g., numerical approximation errors due to the presence of singularities in some of the functions used).

Face Validity.

Compare model output results with actual output results of the real system.

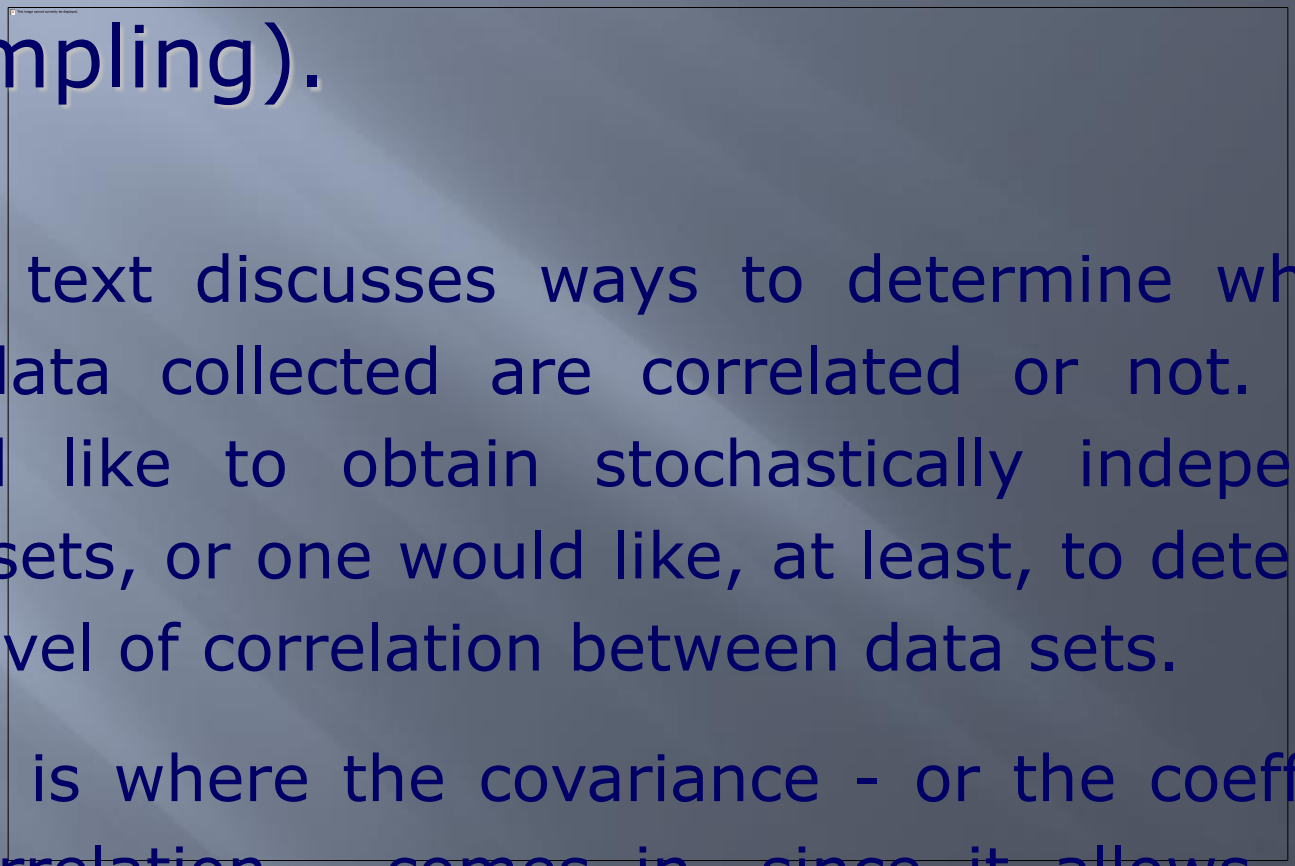
Variable-Parameter Validity.

Compare sensitivity to small changes in internal parameters or initial values with historical data. Compare model dependencies with historical data, looking for the same dependencies.

Event or Time-Series Validity.

Does the model predict observable events, event patterns and variations in output variables?

Some Ideas about Data Collection (Sampling).



The text discusses ways to determine whether the data collected are correlated or not. One would like to obtain stochastically independent data sets, or one would like, at least, to determine the level of correlation between data sets.

This is where the covariance - or the coefficient of correlation - comes in, since it allows us to determine something about dependence.

Repetition: how many runs and how long should they be?

Blocking: how do we avoid the contributions of transient periods (this assumes we are interested in steady-state behavior).

We can attempt to determine whether two runs are independent in the following way.

Let the x_i denote individual observations, n the number of observations. Let the average estimated performance measure be

$$\hat{\mu} = \sum_{i=1}^{n} \frac{x_i}{n}$$

If each if the x_i is independent, the confidence of this performance measure is just the estimate of the variance

$$\hat{\sigma}_{\mu}^{2} = \frac{\sigma^2}{n}$$

If $\{x_i;\ i = 1,\ldots,n/2\}$, $\{y_i;\ i = 1,\ldots,n/2\}$ are two sets of observations, we can try to find out whether they are correlated. We observe that the mean of the union of the two sets can be written as

$$\mu = \frac{1}{n/2}\sum_{i=1}^{n/2}\frac{x_i + y_i}{2}$$

The variance can be written as $\hat{\sigma}_{\mu}^{2} = \frac{\sigma^2}{n}(1+\alpha)$

Where α is the *replication correlation coefficient*. The formula can be derived through the following observations:

$$Var\left(\frac{X}{2}+\frac{Y}{2}\right) = Var\left(\frac{X}{2}\right) + Var\left(\frac{Y}{2}\right) + 2Cov\left(\frac{X}{2},\frac{Y}{2}\right)$$

Where $Cov(X,Y) = E(X,Y) - \mu_X \mu_Y = \alpha \sigma_X \sigma_Y$

and α is the coefficient of correlation. Using this in the original formula, and under the assumption that the two samples come from the same population (equal population variance):

$$Var\left(\frac{X}{2} + \frac{Y}{2}\right) = \frac{1}{4} Var(X) + \frac{1}{4} Var(Y) + 2\alpha \sigma_{\frac{X}{2}} \sigma_{\frac{Y}{2}}$$

$$= \frac{1}{4}\sigma^2 + \frac{1}{4}\sigma^2 + 2\alpha\left(\frac{1}{2}\sigma\right)\left(\frac{1}{2}\sigma\right) = 2\sigma^2(1+\alpha)$$

Since the variance of the means is given by the population variance divided by sample size, we have:

$$\hat{\sigma}^2_{\mu} = \frac{1}{n/2}\left(\sigma^2_{\frac{X}{2}+\frac{Y}{2}}\right) = \frac{1}{n/2}\,2\sigma^2(1+\alpha) = \frac{\sigma^2}{n}(1+\alpha)$$

If two runs (replications) are independent, then α=0, since they must behave exactly as one run of twice the length (i.e. n). We now have a way to test whether two successive runs are correlated or not.

To see what negatively correlated runs can do, assume we have obtained the following set of observations:

X = [.3211106933, .3436330737, .4742561436, .5584587190, .7467538305, .3206222209e-1, .7229741218, .6043056139, .7455800374, .2598119527, .3100754872, .7971794905, .3916959416e-1, .8843057167e-1, .9604988341, .8129204579, .4537470195, .6440313953, .9206249473, .9510535301]

from a uniform random number generator. The "complementary" set (1 - r, for each r in the first set) is

Y = [.6788893067, .6563669263, .5257438564, .4415412810, .2532461695, .9679377779, .2770258782, .3956943861, .2544199626, .7401880473, .6899245128, .2028205095, .9608304058, .9115694283, .395011659e-1, .1870795421, .5462529805, .3559686047, .793750527e-1, .489464699e-1]

The coefficient of correlation is given by Maple V as

α = describe[covariance](X,Y)/sqrt(describe[variance](X)*

describe[variance](Y)) = -.9999999996

Very close to -1. What is the actual variance for the joint population? The formula we use must lead us to a run of 20 items where each item is the mean of the corresponding two items in the separate populations.

But $(x_i + y_i)/2 = (r_i + (1 - r_i))/2 = 1/2$, μ = 1/2 and the sample variance is given by

$$\sum_{i=1}^{n/2} \frac{\left((x_i + y_i)/2 - \mu\right)^2}{n/2 - 1} \equiv 0$$

Negative correlation leads to a smaller variance than independence, while positive correlation leads to a larger variance.

If we want to estimate means and variances of different runs - i.e. runs with different conditions. We have estimated sample means and variances to be

$$\hat{\mu}_1, \hat{\mu}_2, \hat{\sigma}_1^2, \hat{\sigma}_2^2$$

The mean and variance of the difference are

$$\hat{\mu}_D = \hat{\mu}_1 - \hat{\mu}_1, \quad \hat{\sigma}_D^2 = \hat{\sigma}_1^2 + \hat{\sigma}_2^2 - 2\alpha\hat{\sigma}_1\hat{\sigma}_2$$

Where α is the usual coefficient of correlation. The difference in the formula is due to the difference for the statistics - rather than the sum. It is immediate to see that positively correlated runs will diminish the variance of the difference. To check whether $\hat{\mu}_D = 0$ positively correlated runs will make the variance small.

One of the problems is the elimination of transient information. In this case it will be advisable to have long runs, in which the early part has been ignored.

One method that helps find out a reasonable length for the run involves computation of the *autocorrelation function*. This is defined as:

$$\alpha(\Delta) = \frac{E[[x_t - \mu][x_{t+\Delta} - \mu]]}{\sigma^2}$$

Where x_t is an observation at time t; μ is the mean of the observations and σ^2 is their variance.

Note that $\Delta = 0$, gives that $\alpha(0) = 1$.

The sample mean is computed in the usual manner, while the variance of the sample means is given by the formula

$$\hat{\sigma}_{\mu}^{2} = \frac{\sigma^{2}}{n}\left[1 + 2\sum_{\Delta=1}^{n-1}\left(1 - \frac{\Delta}{n}\right)\alpha(\Delta)\right].$$

Notice that $\alpha(\Delta) = 0$ for all $\Delta > 0$ implies that the observations are independent - and this is reflected in the formula. Correlated observations will thus enlarge the variance of the sample means.

The Blocking Method

This simply consists of

a) Wait until transients are over;

b) collect successive blocks of observations of length k is such a way that the "block means" satisfy independence conditions.

c) Use the block means as “observations” to compute the sample mean and the variance of the sample means.

The independence conditions can be checked via any of the methods already mentioned.

Discrete Event Simulation

- În caz discret-simulare, operarea unui sistem este reprezentat ca o succesiune cronologică a evenimentelor. Fiecare eveniment are loc la un moment în timp şi marchează o schimbare de stare în cadrul sistemului [1]. De exemplu, dacă un lift este simulat, un eveniment ar putea fi "nivelul 6 apasat", cu starea sistemului rezultat de "ridicare în mişcare" şi în cele din urmă (cu excepţia cazului alege pentru a simula un eşec a ascensorului) "ridica la nivelul 6" este. Un exerciţiu comun în procesul de învăţare cum să-eveniment discret simulări pentru a construi modelul de o coadă, cum ar fi clienţii care sosesc la o bancă să fie deservită de un casier. În acest exemplu, entităţile sistemului sunt CLIENŢI-Coada şi observatori. Evenimentele din sistem sunt client sosire şi client plecare. (Caz de povestitor-incepe-serviciu poate fi o parte a logicii de sosire şi plecare evenimente) Sistemul de state, care sunt modificate de către aceste evenimente,. Sunt NUMĂRUL DE CLIENTI-IN-coada-(un număr întreg de la de la 0 la n) şi Teller-STARE (ocupat sau inactiv).
- Variabile aleatoare care trebuie să fie caracterizate de modelul de acest sistem stochastic sunt client INTERARRIVAL-time şi Teller-SERVICE-TIME. Un număr de mecanisme au fost propuse pentru efectuarea de simulare discret-eveniment, printre ele sunt bazate pe evenimente, activitate bazate pe proces şi bazate pe trei faze abordări (Pidd, 1998). Abordare în trei faze este utilizat de către un număr de pachete de software comercial de simulare, dar din punctul de vedere al utilizatorului, specificul metodei de simulare subiacente sunt, în general, ascunse.

Components of a Discrete-Event Simulation

- În plus faţă de reprezentarea variabilelor de stare sistem şi logică a ceea ce se întâmplă atunci când au loc evenimente de sistem, simularile cu evenimente discrete includ următoarele:

- Ceasul

- Simularea trebuie să ţină evidenţa timpului de simulare actual, în orice unităţi de măsură sunt adecvate pentru sistemul modelat. În simulările cu evenimente discrete, spre deosebire de simulările în timp real, timpul sare, deoarece evenimentele sunt instantanee - ceasul sare la urmatorul eveniment.

Components of a Discrete-Event Simulation

- Lista de evenimente

- Simularea menţine cel puţin o lista de evenimente de simulare. Acesta este numita *the pending event set*, deoarece listea de evenimente care sunt în aşteptarea rezultatelor evenimentelor anterioare, dar nu au fost încă simulate. Un eveniment este descris de la momentul la care apare aceasta, şi un tip, pentru indicarea codului, care vor fi utilizate pentru a simula acest eveniment. Se obisnuieste ca codul de eveniment să fie parametrizat, caz în care, descrierea evenimentului conţine, de asemenea, parametrii de la codul evenimentului. Atunci când evenimentele sunt instantanee, activităţile care se extind în timp sunt modelate ca secvenţe de evenimente. Unele cadre de simulare permite timpul unui eveniment să fie specificat ca un interval: ora de începere şi ora de încheiere a fiecărui eveniment.
- *Single threaded simulation engines based on instantaneous events have just one current event. In contrast, multi-threaded simulation engines and simulation engines supporting an interval-based event model may have multiple current events.* În ambele cazuri, există probleme semnificative cu sincronizare între evenimentele curente..

Components of a Discrete-Event Simulation

Random-Number Generators

- Simularea are nevoie pentru a genera variabile aleatoare de diferite feluri, în funcţie de modelul sistemului. Aceasta se realizează prin una sau mai multe generatoare de numărul pseudoaleatoare. Utilizarea de numere pseudoaleatoare, spre deosebire de numere aleatoare adevarate este un beneficiu daca simularea are nevoie de o reluare cu exact acelaşi comportament. Una din problemele cu distribuţiile de numere aleatorii utilizat la eveniment de simulare discretă este că *the steady-state distributions of event times* nu pot fi cunoscute în avans. Ca urmare, setul initial de evenimente plasate în setul în aşteptare a evenimentului nu vor avea sosire reprezentant ori de distribuţie la starea de echilibru. Această problemă este rezolvată de obicei, de procesul de *bootstrap* modelul de simulare. Numai un efort limitat se face pentru a atribui ori realist să setul initial de evenimente în curs. Aceste evenimente, însă, evenimente programul suplimentare, şi în timp, distribuirea ori eveniment abordări starea sa de echilibru. Aceasta se numeşte procesul de bootstrap modelul de simulare.
- În colectarea de statistici de la modelul de funcţionare, este important să nu ţină seama, fie evenimentele care au loc înainte ca statul echilibru este atins sau pentru a rula de simulare pentru suficient de lungi pentru ca procesul de bootstrap comportament este copleşit de comportament echilibru-stat.