Stochastic Simulation
Performance and Dependability

Paulo Maciel

Centro de Informática - UFPE

Objective

- To study the fundaments of stochastic simulation, its methods, and applying simulation for solving performance and dependability problems.

Program

- Introduction
- Problems and Mistakes in Simulation
- System and Model
- Classification of Models
- Types of Simulation
- Discrete Event Simulation: an Overview
- Random Number Generation
- Random Variates Generation
- Output Analysis – Steady State Simulation
- Output Analysis – Transient Simulation
- Verification and Validation

Methodology

- Expositive classes
- Practical classes

Evaluation

- List resolutions
- Write a draft paper

Basic References

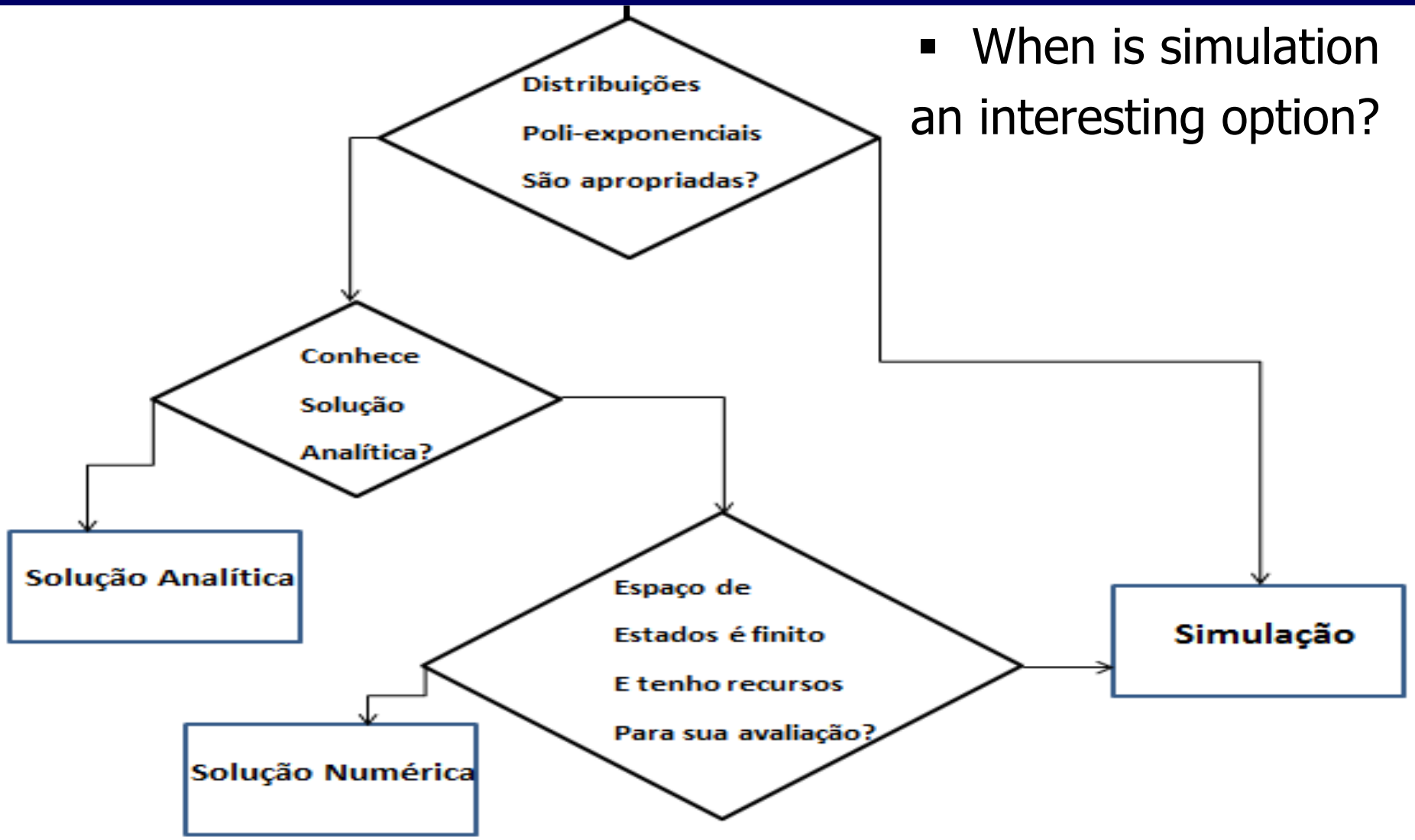
- Art of Computer Systems Performance Analysis Techniques For Experimental Design Measurements Simulation And Modelingm, Raj Jain, Wiley Computer Publishing, John Wiley & Sons, Inc,1991.
- Simulation modeling handbook : a practical approach, Christopher A. Chung. 2003.
- Stochastic Simulation: Algorithms and Analysis, Søren Asmussen, Peter W. Glynn. Springer, 2007.
- Introduction to Discrete Event Systems, Christos G. Cassandras, Stéphane Lafortune. Springer. 2008

Introduction

- Simulation
- Analysis
- Evaluation

Introduction

- When is simulation an interesting option?



Introduction

- Common mistakes in simulation projects:
  - Not considering numerical or analytical methods as possible alternative
  - Inappropriate level of detail
  - Not taking into account the modeling and coding costs
  - Unverified and invalid models
  - Improperly handled initial conditions
  - Results without confidence level and error margin
  - Poor random-number generators
  - Poor interpretation of results

System and Model

The Concept of System

- An aggregation or assemblage of things so combined by nature or man as to form an integral or complex whole (Encyclopedia Americana).
- An interacting or interdependent group of items forming a unified whole (Webster's Dictionary)
- A combination of components that act together to perform a function (IEEE Standard Dictionary).

- There are two features in these definitions:
  1. a system consists of interacting "components", and
  2. a system is associated with a "function".

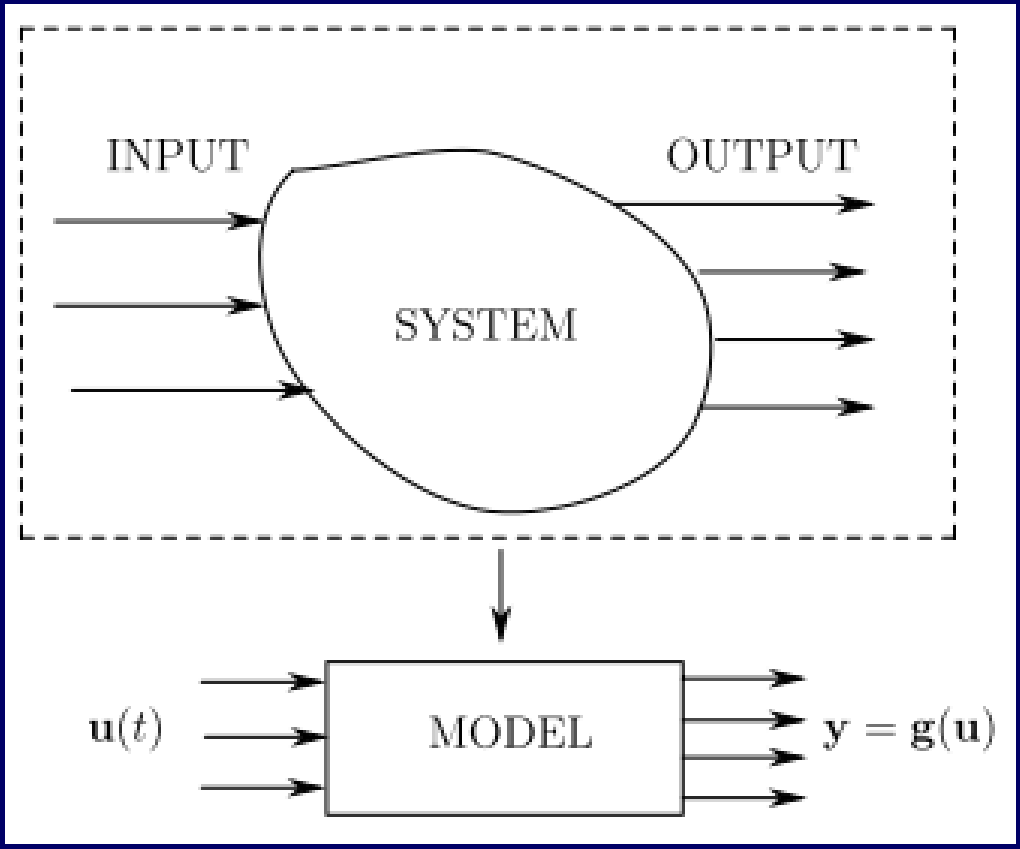
System and Model

Model

- Abstraction level
- Perspective

System and Model

Model



System and Model

State and state variable

The state of a system at time t_0 is the information required at t_0 such that the output $y(t)$, for all $t \geq t_0$, is uniquely determined from this information and from $u(t)$, $t \geq t_0$.

Like the input $u(t)$ and the output $y(t)$, the state is also generally a vector, which we shall denote by $x(t)$. The components of this vector, $x_1(t), \ldots, x_n(t)$, are called state variables.

System and Model

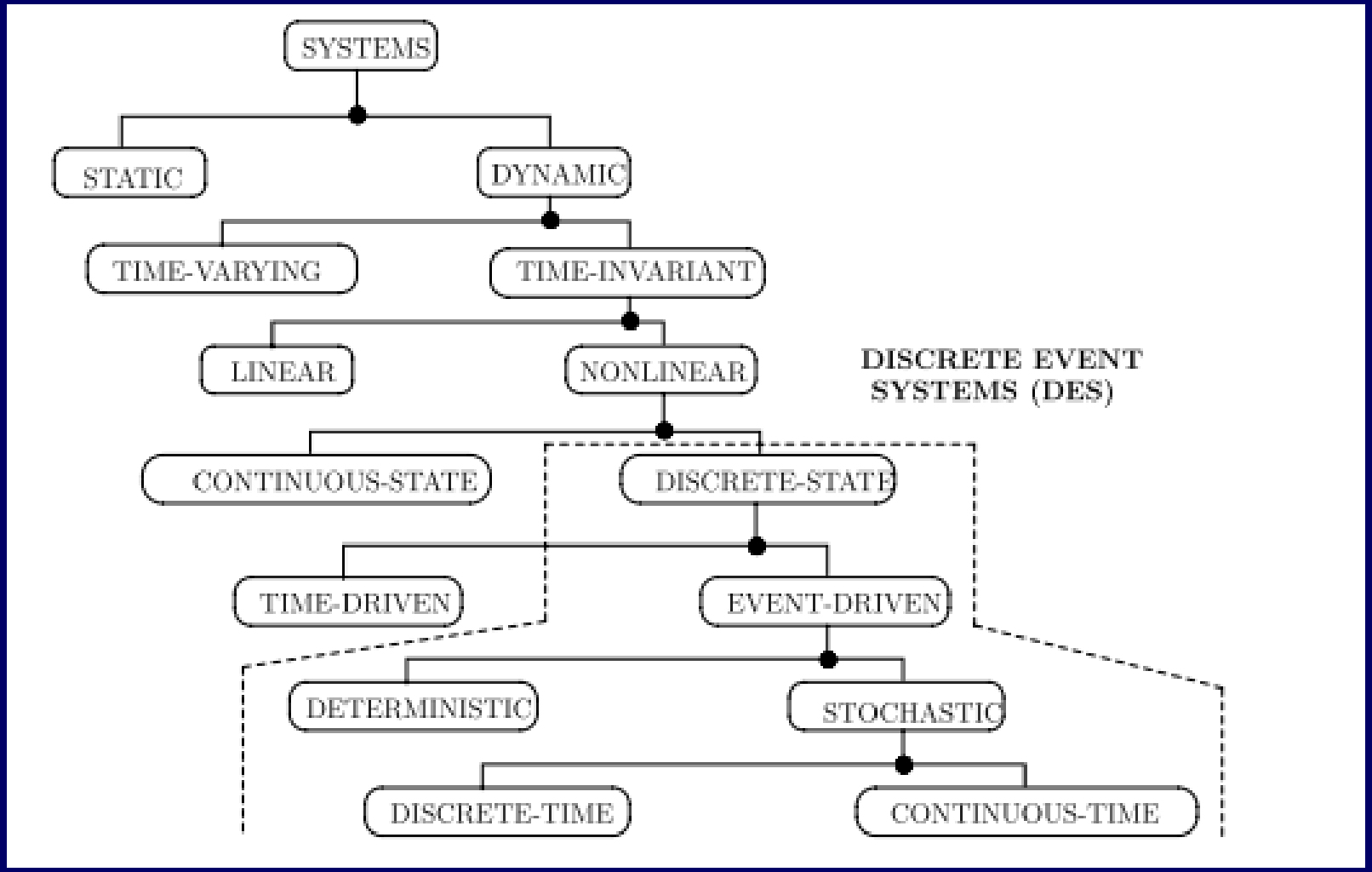
Event

An event may be identified with a specific action taken by a person or system or

It may be viewed as a spontaneous occurrence dictated by nature or

it may be the result of several conditions which are suddenly all met.

System classification



Types of Simulation

Static vs Dynamic Simulation

- Static simulation is adopted for representing a system in which time does not play a role or to describe a system at a particular time.
- Dynamic simulation is used for depicting a system that evolves over time.

Types of Simulation

Deterministic vs Stochastic Simulation

- Deterministic simulation considers models that do not contain any random variable, that is, the output is determined once the input quantities and the model relations are defined.
- Stochastic simulation adopts models that have at least one random variable.

Types of Simulation

Continuous vs Discrete Model Simulation

- Continuous model simulation relies on continuous models for representing the system whereas discrete model simulation is supported by discrete models.

Types of Simulation

- Trace driven (Deterministic simulation)
- Monte Carlo (Stochastic static model simulation)
- Discrete event (Stochastic dynamic discrete model simulation)

Trace-Driven Simulation

- Trace-driven simulation uses traces (time-ordered record of events on a real system) as its input applied to the simulation model which depicts the system behavior. This sort of simulation strategy has been applied for evaluating resource management algorithms, such as paging algorithms, cache analysis, CPU scheduling algorithms, and algorithms for dynamic allocation of storage.

Monte Carlo Simulation

- Monte Carlo simulations are used to model probabilistic phenomena that do not change characteristics over time.

- Monte Carlo simulations are also used for evaluating non-probabilistic expressions using probabilistic methods.

Monte Carlo Simulation

Steps in the Monte Carlo Simulation

1. Define the Model (Transfer Equation)
2. Define the Input Parameters
3. Create Random Input Data
4. Apply the Input Data to the Model
5. Analyze the Output

Monte Carlo Simulation

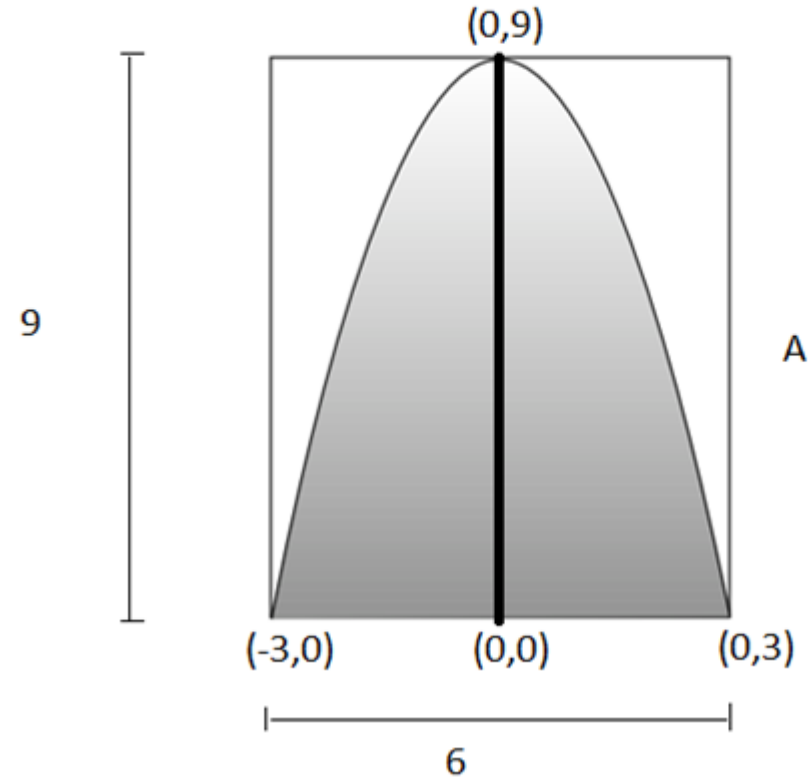
Excel

Mathematica

Integration

Now, let's try to find the area bounded by $y = 9 - x^2$ and $y = 0$.

We will envelop this area by a rectangular larger area that measures 6 × 9.



$A_R = 9 \times 6 = 54$

Model:

$$\tilde{A}_F := \left(\frac{\sum_{k=1}^{n} C(x_k, y_k)}{n}\right) \times A_R$$

$$C(x,y) = \begin{cases} 1 & if \quad x^2 + y \leq 9 \\ 0 & otherwise \end{cases}$$

Input:

$$x_k = 6 \times U(0,1) - 3$$

$$y_k = 9 \times U(0,1)$$

Exact Area:

$$\int_{-3}^{3} (9 - x^2)\, dx = \left(9x - \frac{x^3}{3}\right)\Big|_{-3}^{3} = (27 - 9) - (-27 + 9) = 36$$

Monte Carlo Simulation

Minitab | Mathematica | Excel | Mathematica V2

- Estimating π - Model

Since the area of quarter circle is:

$$A_{QC} = \frac{\pi \times r^2}{4}$$

Then

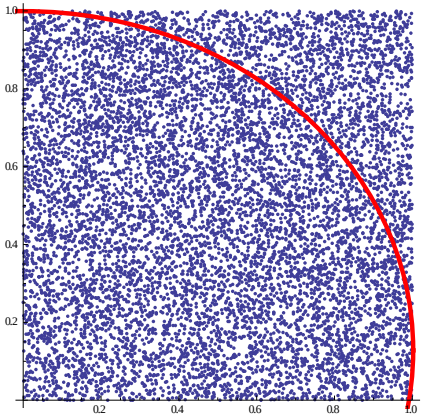
$$\pi = \frac{4 \times A_{QC}}{r^2}$$

As

$$r^2 = 1$$

Hence

$$\pi = 4 \times A_{QC}$$



An approximation of π can be estimated by:

$$\tilde{\pi} = 4 \times \widetilde{A_{QC}}$$

where

$$\widetilde{A_{QC}} = \frac{\sum_{k=1}^{n} C_k(i,j)}{n} \times A_R$$

$$C(i,j) = \begin{cases} 1 & if\ d(i,j) \le 1 \\ 0 & otherwise \end{cases}$$

$$d(i,j) = \sqrt{U_i^2 + U_j^2}$$

and

$$(i,j) = (U,U)$$

Input parameters

$$A_R = r^2 = 1$$

Therefore

Model

$$\tilde{\pi}(n) = 4 \times \frac{\sum_{k=1}^{n} C_k(i,j)}{n}$$

Monte Carlo Simulation

Piston pump

A manufacturing company needs to evaluate the design of a piston pump that must pump 12 ml of fluid per minute (flow rate).

You want to estimate the mean flow rate over thousands pumps, given specified variation in piston diameter (D), stroke length (L), and strokes per minute (frequency - RPM).

Ideally, the pump flow rate across thousands of pumps will have a standard deviation no greater than 0.2 ml/min.

Based on the historical data of pumps your facility has manufactured, you can say that piston diameter is normally distributed with a mean of 0.8 cm and a standard deviation of 0.003 cm, the Stroke length is normally distributed with a mean of 2.5 cm and a standard deviation of 0.15 cm. Finally, strokes per

Monte Carlo Simulation

Cont.

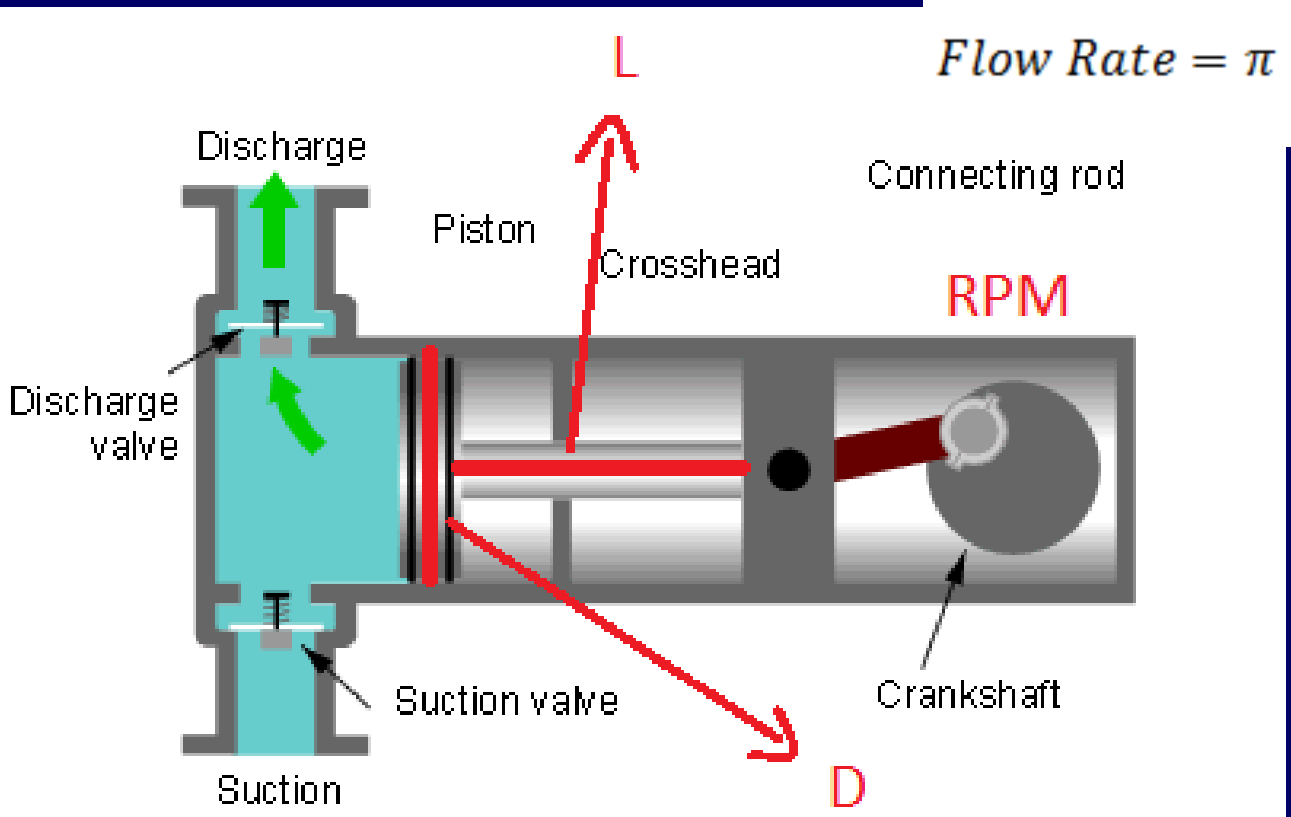
minute is normally distributed with a mean of 9.549 RPM and a standard deviation of 0.17 RPM.

Estimate the mean flow rate and the respective standard deviation, considering a sample of 10,000 pumps with the above input definition.

Monte Carlo Simulation

Model

$$Flow\ Rate = \pi \times \left(\frac{D}{2}\right)^2 \times L \times RPM$$



Monte Carlo Simulation

Input data parameters

$$D \sim N(0.8\ cm, 0.003\ cm)$$

$$L \sim N(2.5\ \text{cm}, 0.15\ \text{cm})$$

$$Frequecy\text{: } RPM \sim N(9.549\ \text{RPM}\ , 0.17\ \text{RPM})$$

Monte Carlo Simulation

- Create random input data
- Apply the input data to the model
- Analyze the output

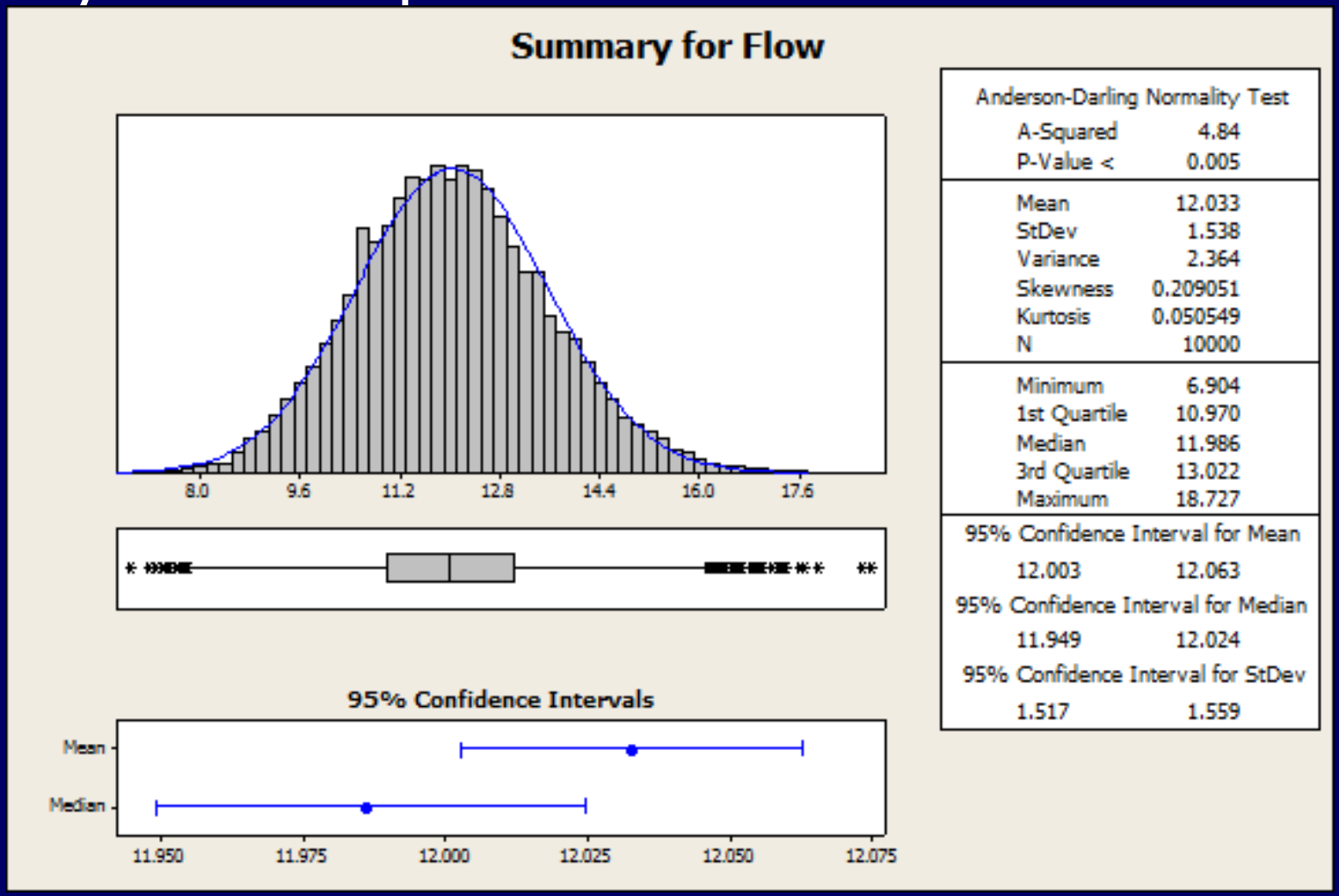
Minitab

Excel

Mathematica

Monte Carlo Simulation

Analyze the output



Anderson-Darling Normality Test	
A-Squared	4.84
P-Value <	0.005
Mean	12.033
StDev	1.538
Variance	2.364
Skewness	0.209051
Kurtosis	0.050549
N	10000
Minimum	6.904
1st Quartile	10.970
Median	11.986
3rd Quartile	13.022
Maximum	18.727
95% Confidence Interval for Mean	
12.003	12.063
95% Confidence Interval for Median	
11.949	12.024
95% Confidence Interval for StDev	
1.517	1.559

Monte Carlo Simulation

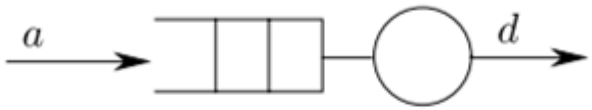
Analyze the output

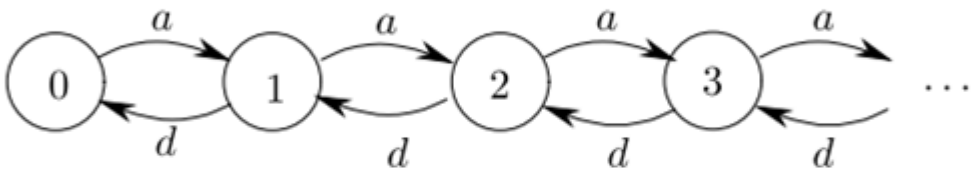
$$\mu \in (12.0025 \quad , \quad 12.0628)$$
$$\sigma \in (1.52, \quad 1.56)$$

The average flow rate is pretty close to the specified value, but the standard deviation far exceed 0.2 ml/min.

Discrete Event Simulation: an Overview

Let's consider the Stochastic Timed Automaton (STA) that represents a G/G/1 queue model:





$$\mathcal{E} = \{a, d\} \quad X = \{0, 1, 2, \cdots\}$$

$$\Gamma(x) = \{a, d\} \text{ for all } x > 0 \quad \Gamma(0) = \{a\}$$

$$f(x, e') = \begin{cases} x + 1 & \text{if } e' = a \\ x - 1 & \text{if } e' = d \text{ and } x > 0 \end{cases}$$

Given: $p_0(x), x \in \mathcal{X}, \quad G_a(\cdot), G_d(\cdot)$

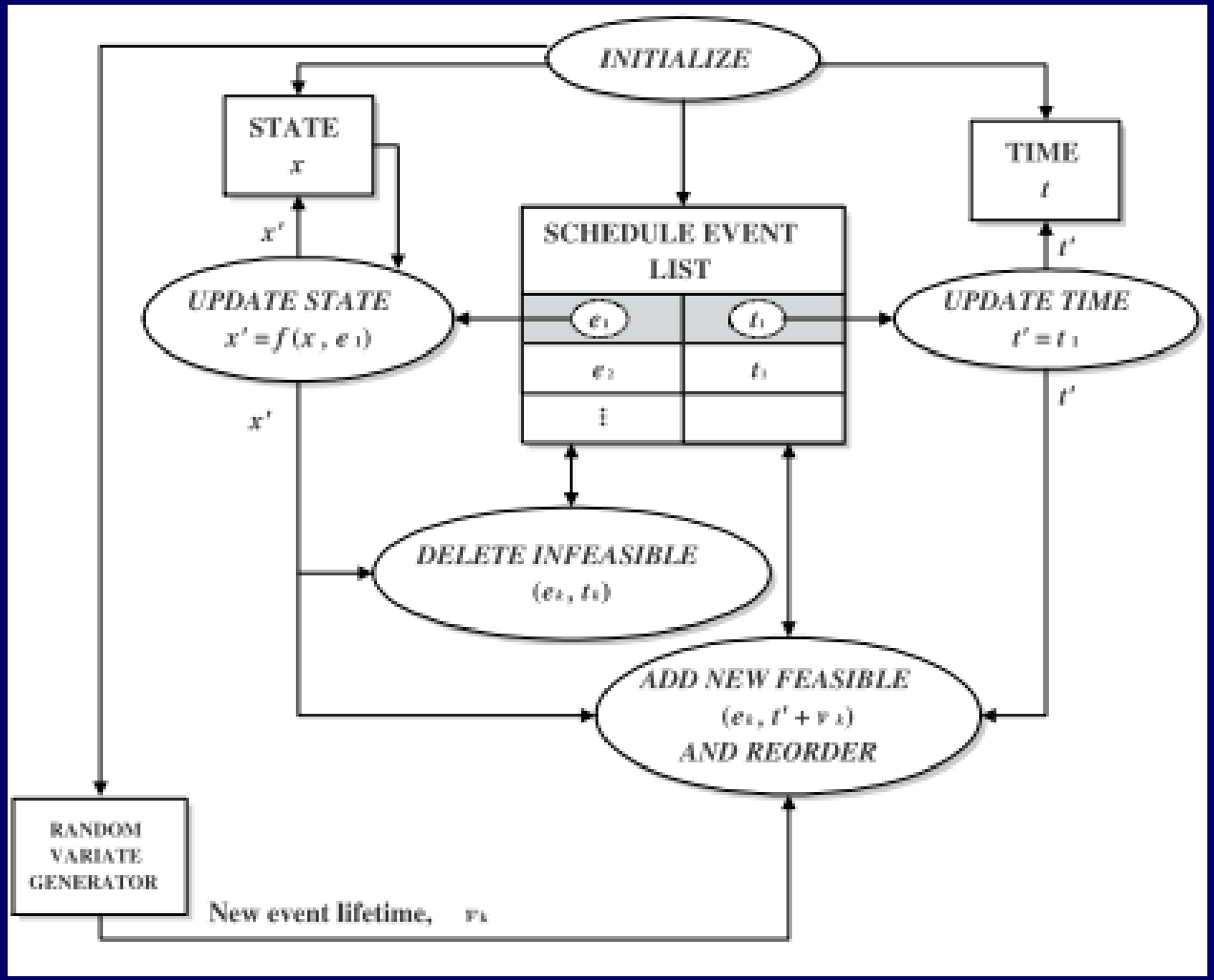
Recall that $\mathcal{E}$ is always a countable set of events, and that X is a denumerable state space.

Discrete Event Simulation: an Overview

We denote a denumerable set as a set that is countably infinite, or countable.

Let's recall that G of the stochastic timed automaton model is the set of probability distributions $G = \{G_i : i \in \mathcal{E}\}$ characterizing event lifetimes. This information is combined with the computer's Random Number Generator (RNG) to supply event lifetime samples as required.

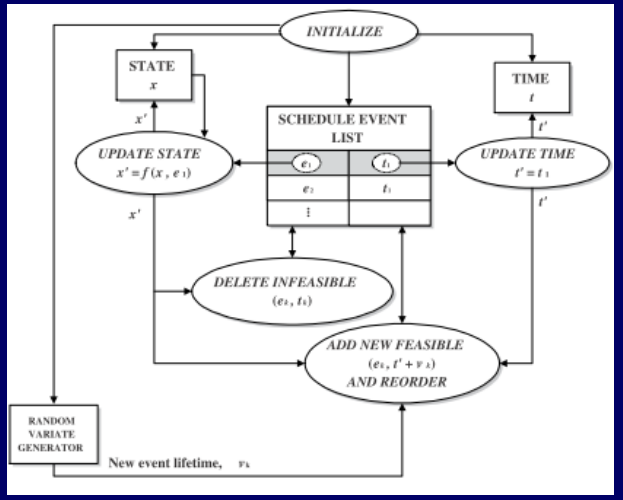
Discrete Event Simulation: an Overview

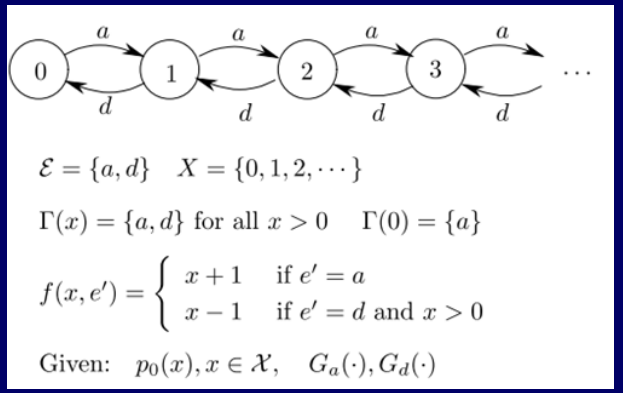


Discrete Event Simulation: an Overview

1. Let's initialize the time at t = 0 and assume that an initial state $x_0 \in X$ is given. Considering the G/G/1 model, let's consider $x_0 = 0$.
2. For the state $x = 0$, the set of feasible events is $\Gamma(0) = \{a\}$. This is the only event which may occur at this state.
3. Associate a a clock value, which represents the amount of time required until a occurs.

   This amount of time is supplied by the random variate generating mechanism that considers $G_a(.)$. We denote clock values for event a by y_a, and their lifetimes by v_a. Hence $y_a = v_a$.





$\mathcal{E} = \{a, d\} \quad X = \{0, 1, 2, \cdots\}$

$\Gamma(x) = \{a, d\}$ for all $x > 0 \quad \Gamma(0) = \{a\}$

$$f(x, e') = \begin{cases} x + 1 & \text{if } e' = a \\ x - 1 & \text{if } e' = d \text{ and } x > 0 \end{cases}$$

Given: $p_0(x), x \in \mathcal{X}, \quad G_a(\cdot), G_d(\cdot)$

Discrete Event Simulation: an Overview

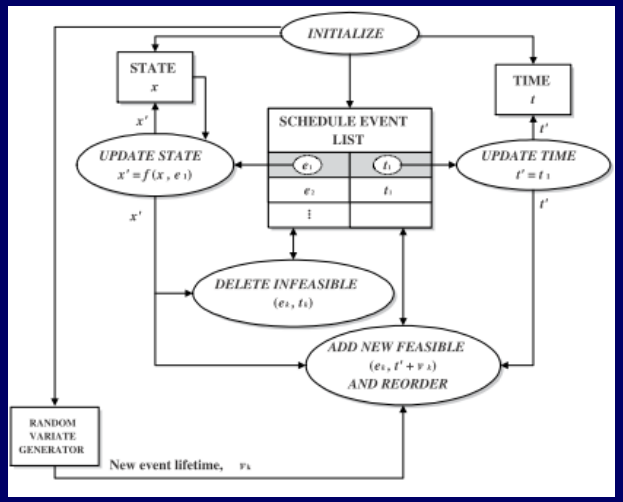
4. Now given that the current state is $x = 0$, we look at all clock values y_i where $i \in \Gamma(x = 0)$.

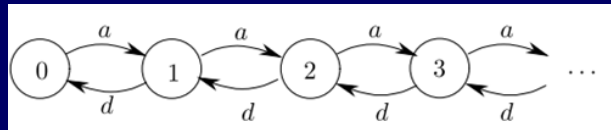
   The triggering event e is the event which occurs next at that state, that is, the event with the smallest clock value:

$$e' = \arg \min_{i \in \Gamma(x=0)} \{y_i\}$$

As the only of feasible event is a, the triggering event e' is a:

$$e' = \arg \min_{i \in \{a\}} \{v_a\} = a$$





$\mathcal{E} = \{a, d\} \quad X = \{0, 1, 2, \cdots\}$

$\Gamma(x) = \{a, d\}$ for all $x > 0 \quad \Gamma(0) = \{a\}$

$$f(x, e') = \begin{cases} x + 1 & \text{if } e' = a \\ x - 1 & \text{if } e' = d \text{ and } x > 0 \end{cases}$$

Given: $p_0(x), x \in \mathcal{X}, \quad G_a(\cdot), G_d(\cdot)$

Discrete Event Simulation: an Overview

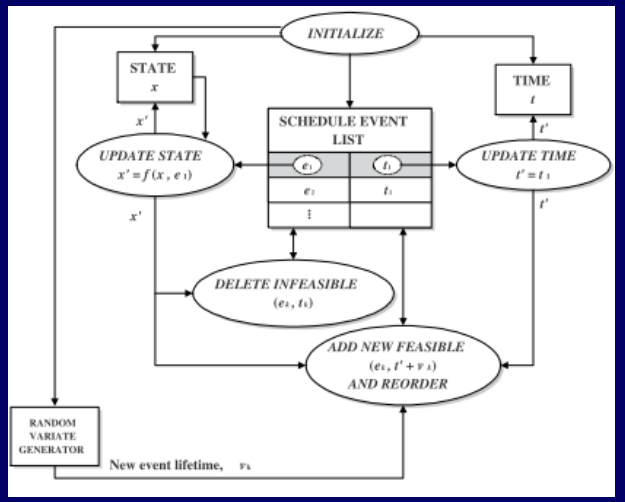
At this point we can update the state based on a given state transition mechanism.

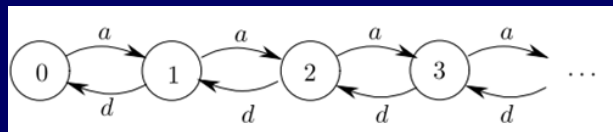
5. As there is only one possible new state (the adopted STA has a deterministic state transition mechanism) when event a occurs at state 0, we have $f(0, a) = 1$.

6. The amount of time spent at state x defines the interevent time y^*:

$$y^* = \min_{i \in \Gamma(x=0)} \{y_i\}$$

Since $\Gamma(x = 0) = \{a\}$, the amount of time spent at state 0 is $y^* = y_a = v_a$





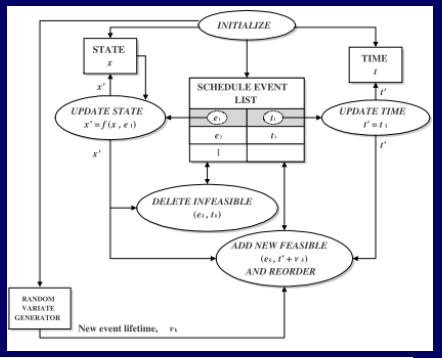
$\mathcal{E} = \{a, d\} \quad X = \{0, 1, 2, \cdots\}$

$\Gamma(x) = \{a, d\}$ for all $x > 0 \quad \Gamma(0) = \{a\}$

$$f(x, e') = \begin{cases} x + 1 & \text{if } e' = a \\ x - 1 & \text{if } e' = d \text{ and } x > 0 \end{cases}$$

Given: $p_0(x), x \in \mathcal{X}, \quad G_a(\cdot), G_d(\cdot)$

Discrete Event Simulation: an Overview



7. We can then update time by

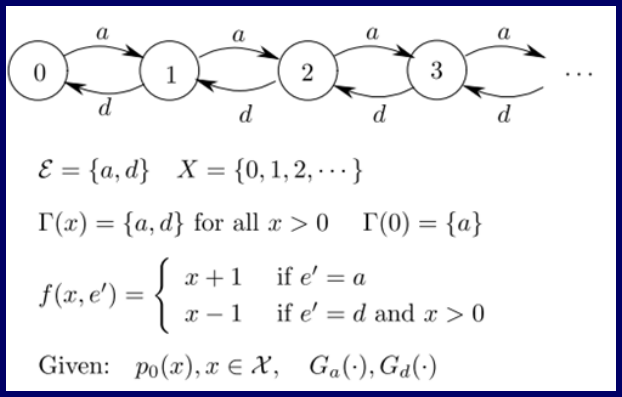
$$t' = t + y^* = 0 + v_a$$

8. We also update clock values for all feasible events in the new state $x = f(0, a) = 1$ as follows:

- There are two cases to consider.
  - First, if an event $i \in \Gamma(x)$ such that $i \neq e'$ remains feasible in the new state x, the time remaining until its occurrence (the new clock value) is simply given by

$$y_i' = y_i - y^*$$

  - The second case applies to e' itself if $e' \in \Gamma(x) - x$ is next state $-$ and to all other events which were not feasible in x (previous state), but become feasible in x (next state). For all such events, we need new lifetimes. These new lifetimes are supplied once again by the computer through the random variate generating mechanism.



$\mathcal{E} = \{a, d\} \quad X = \{0, 1, 2, \cdots\}$

$\Gamma(x) = \{a, d\}$ for all $x > 0 \quad \Gamma(0) = \{a\}$

$$f(x, e') = \begin{cases} x + 1 & \text{if } e' = a \\ x - 1 & \text{if } e' = d \text{ and } x > 0 \end{cases}$$

Given: $p_0(x), x \in \mathcal{X}, \quad G_a(\cdot), G_d(\cdot)$

Discrete Event Simulation: an Overview

At $x = 1, \Gamma(1) = \{a, d\}$:

$y_a = v_a,$

$y_d = v_d,$

since $a = e'$ at $\Gamma(0)$, and $d \notin \Gamma(0)$

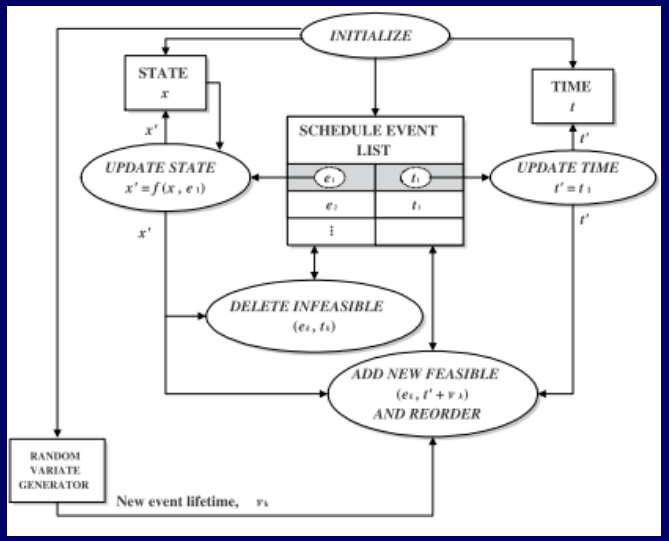
9. $e' = \arg\min_{i \in \Gamma(1)} \{y_i\} = \arg\min_{i \in \{a,d\}} \{v_a, v_d\}$

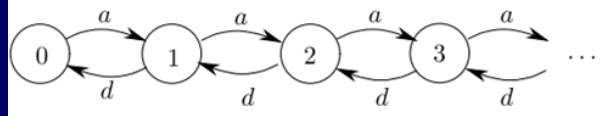
$y^* = \min_{i \in \{a,d\}} \{v_a, v_d\}$

Let's assume that $v_a < v_d$, then:

$e' = \arg\min_{i \in \{a,d\}} \{v_a, v_d\} = a$ and

$y^* = \min_{i \in \{a,d\}} \{v_a, v_d\} = v_a.$





$\mathcal{E} = \{a, d\} \quad X = \{0, 1, 2, \cdots\}$

$\Gamma(x) = \{a, d\}$ for all $x > 0 \quad \Gamma(0) = \{a\}$

$$f(x, e') = \begin{cases} x + 1 & \text{if } e' = a \\ x - 1 & \text{if } e' = d \text{ and } x > 0 \end{cases}$$

Given: $p_0(x), x \in \mathcal{X}, \quad G_a(\cdot), G_d(\cdot)$

Discrete Event Simulation: an Overview

10. Hence we can then update time by

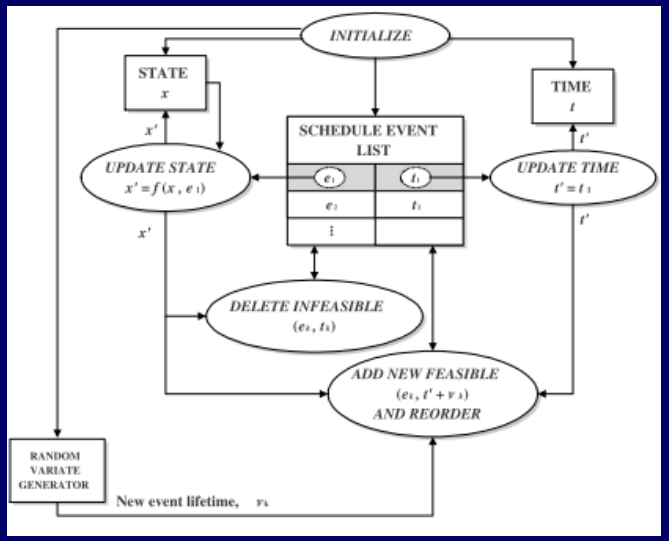
$$t' = t + y^* = v_a + v_a = 2 \times v_a.$$

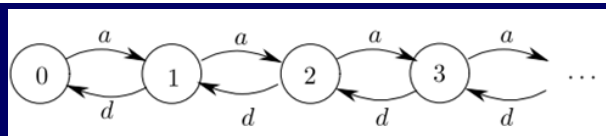
11. We also update clock values for all feasible events in the new state $x = f(1, a) = 2$ as follows:

First, since $d \in \Gamma(2)$ and $d \neq e'$ (that is, the triggered event e' was a), then the time remaining until its occurrence (the new clock value) is simply given by

$$y_d{}' = y_d - y^* = v_d - v_a$$

The second case applies to a itself since $a \in \Gamma(2)$. For a, we need a new lifetime. The new lifetime is supplied again by the computer through the random variate generating mechanism.





$\mathcal{E} = \{a, d\} \quad X = \{0, 1, 2, \cdots\}$

$\Gamma(x) = \{a, d\}$ for all $x > 0 \quad \Gamma(0) = \{a\}$

$$f(x, e') = \begin{cases} x + 1 & \text{if } e' = a \\ x - 1 & \text{if } e' = d \text{ and } x > 0 \end{cases}$$

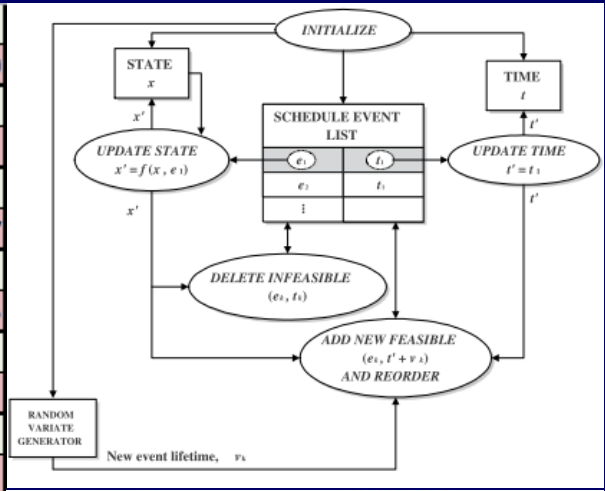
Given: $p_0(x), x \in \mathcal{X}, \quad G_a(\cdot), G_d(\cdot)$

Excel
N/N/1

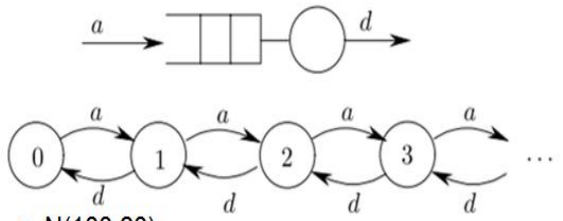
Discrete Event Simulation: an example

	x	x	x	x	x	x	x
State	0	1	0	1	2	1	0
	x'=f(x,e')	x'=f(x,e')	x'=f(x,e')	x'=f(x,e')	x'=f(x,e')	x'=f(x,e')	x'=f(x,e')
Next state	1	0	1	2	1	0	
	y*	y*	y*	y*	y*	y*	y*
Sojourn time at x	109.7491	71.3786	19.87684	92.07155	23.56192	67.53348	7.426077
	t	t	t	t	t	t	t
Global time	0	109.7491	181.1277	201.0046	293.0761	316.6381	384.1715
	e'	e'	e'	e'	e'	e'	e'
Scheduled event	a	d	a	a	d	d	a
	Γ(x)	Γ(x)	Γ(x)	Γ(x)	Γ(x)	Γ(x)	Γ(x)
Set of enabled events	a	a,d	a	a,d	a,d	a,d	a
	va	va	va	va	va	va	va
Random variate generated for a	109.7491	91.25544		92.07155	98.52148		
		vd		vd	vd	vd	vd
Random variate generated for d		71.3786		115.6335		67.53348	
	ya	ya	ya	ya	ya	ya	ya
Remaining time of a	109.7491	91.25544	19.87684	92.07155	98.52148	74.95956	7.426077
		yd		yd	yd	yd	yd
Remaining time of d		71.3786		115.6335	23.56192	67.53348	
	ymin	ymin	ymin	ymin	ymin	ymin	ymin
Minimal remaining time	109.7491	71.3786	19.87684	92.07155	23.56192	67.53348	7.426077

Initial state	x0	0
Arrival event	a	N(100,20)
Departure event	d	N(90,20)



G/G/1 queue model: N/N/1

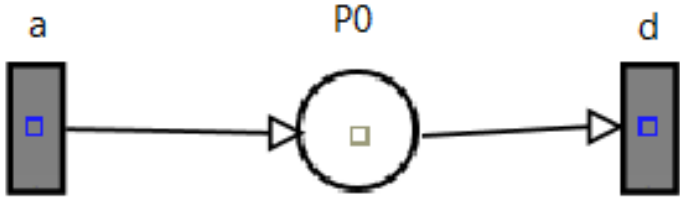


a~N(100,20)
d~N(90,20)

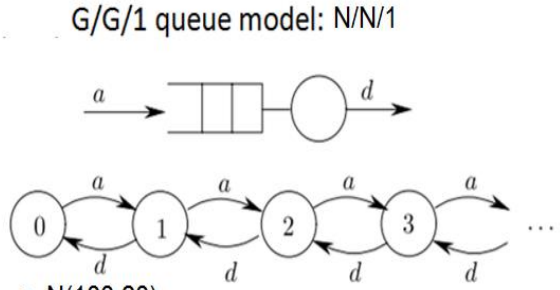
	P{s0}	P{s0}	P{s0}	P{s0}	P{s0}	P{s0}	P{s0}
	1	0.605921	0.644891	0.442295	0.409382	0.349982	
	P{s1}	P{s1}	P{s1}	P{s1}	P{s1}	P{s1}	P{s1}
	0	0.394079	0.355109	0.557705	0.516205	0.589849	
	P{s2}	P{s2}	P{s2}	P{s2}	P{s2}	P{s2}	P{s2}
Metrics	0	0	0	0	0.074413	0.060169	
Σ P{si} =	1	1	1	1	1	1	

C:\Users\Paulo Maciel\Dropbox\Models\Mercury\Mercury 4.4.3\Mercury 4.4.3.2\Examples

Discrete Event Simulation: an example



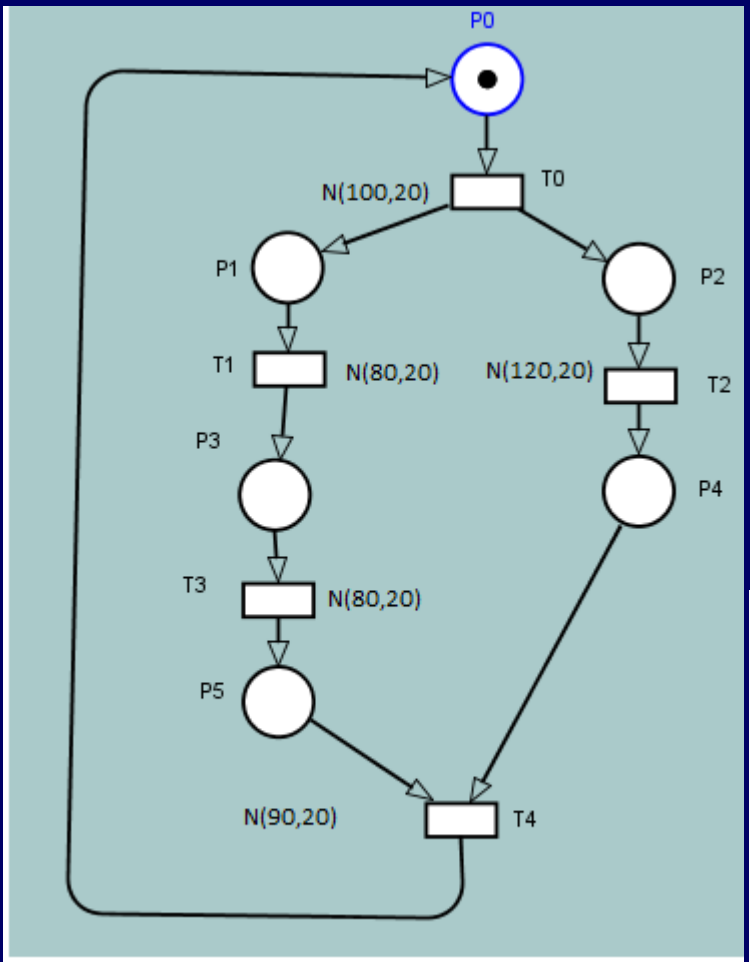
M0: 0.79566154894109..
M1: 0.1059998450720813
M2: 0.0810818630981709




```
Metric : M0, P{#P0=0}
Result: 0.7956615489410949
Confidence Interval: [0.7878510268176973,0.8034720710644925]
Error %: 0.9816387550450609
Metric : M1, P{#P0=1}
Result: 0.10599984507208135
Confidence Interval: [0.10494687340687012,0.10705281673729258]
Error %: 0.9933709473775153
Metric : M2, P{#P0=2}
Result: 0.08108186309817098
Confidence Interval: [0.08027313531127094,0.08189059088507102]
Error %: 0.9974213171703543
Run size: 1000
Number of Runs 403000
Total Runs 403000000
```

C:\Users\Paulo Maciel\Dropbox\Models\Mercury\Mercury 4.4.3\Mercury 4.4.3.2\Examples\StudyOnSimulation.xml

Discrete Event Simulation: a SPN example



State 0: (M(P0)=1, M(P1)= M(P2)= M(P3)= M(P4)= M(P5)=0)

Enabled Transitions	Firable Transitions
T0	T0

Transitions	Delay
T0	91.55042574784241

Transition Fired = T0

Firing Delay = 91.55042574784241

Global Time = 91.55042574784241

State 1: (M(P2)= M(P3)==1, M(P0)= M(P3)= M(P4)= M(P5)=0)

Enabled Transitions	Firable Transitions
T1,T2	T1,T2

Transitions	Delay
T1	108.9958909819219
T2	121.0819483992443

Transition Fired $=\min_{i \in Firable}\{Delay(T_i)\}$

Transition Fired = T1

Firing Delay =108.9958909819219

Global Time =200.5463

State 2: (M(P1)= M(P2)==1, M(P0)= M(P3)= M(P4)= M(P5)=0)

Enabled Transitions	Firable Transitions
T2,T3	T2,T3

Transitions	Delay
T2	Delay(T2)- Firing Delay=121.0819483992443 - 108.9958909819219 = 12.08606
T3	67.76778453445709

Transition Fired $=\min_{i \in Firable}\{Delay(T_i)\}$

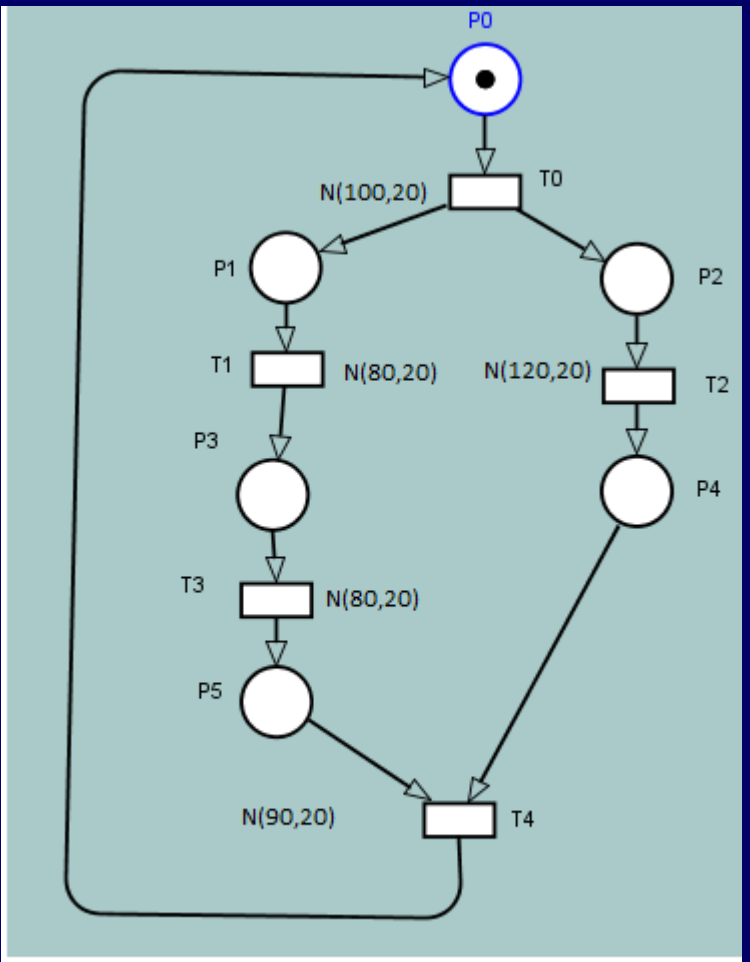
Transition Fired = T2

Firing Delay =12.08606

Global Time =212.6324

C:\Users\Paulo Maciel\Dropbox\Models\Mercury\Mercury 4.4.3\Mercury 4.4.3.2\Examples\StudyOnSimulation.xml

Discrete Event Simulation: a SPN example



State 0: (M(P0)=1, M(P1)= M(P2)= M(P3)= M(P4)= M(P5)=0)

State 3: (M(P3)= M(P4)==1, M(P0)= M(P1)= M(P2)= M(P5)=0)

Enabled Transitions	Firable Transitions
T3	T3

Transitions	Delay
T3	Delay (T3)- Firing Delay = 67.76778453445709 - 12.08606 = 55.68172

Transition Fired = T3
Firing Delay =55.68172
Global Time =268.3141

State 4: (M(P4)= M(P5)==1, M(P0)= M(P1)= M(P2)= M(P3)=0)

Enabled Transitions	Firable Transitions
T4	T4

Transitions	Delay
T4	103.1182638502188

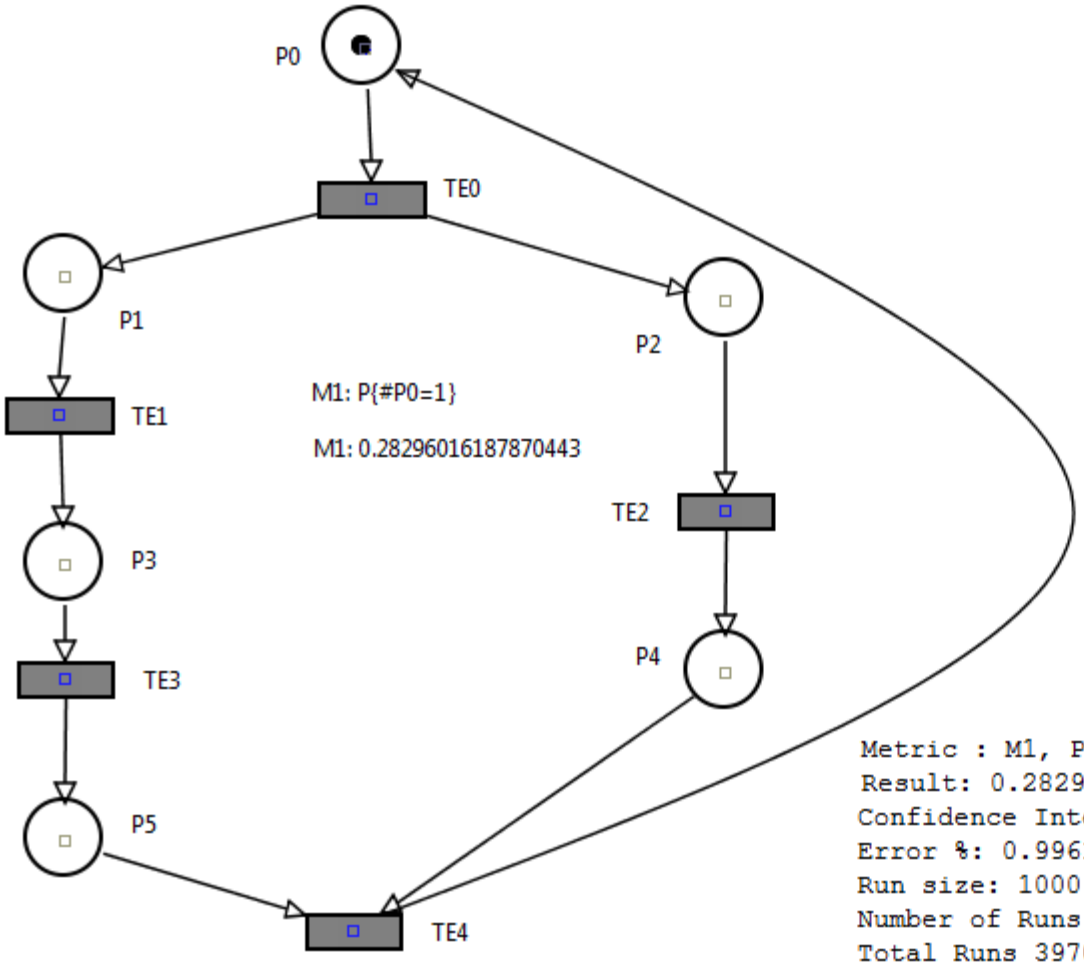
Transition Fired = T4
Firing Delay =103.1182638502188
Global Time =371.4324

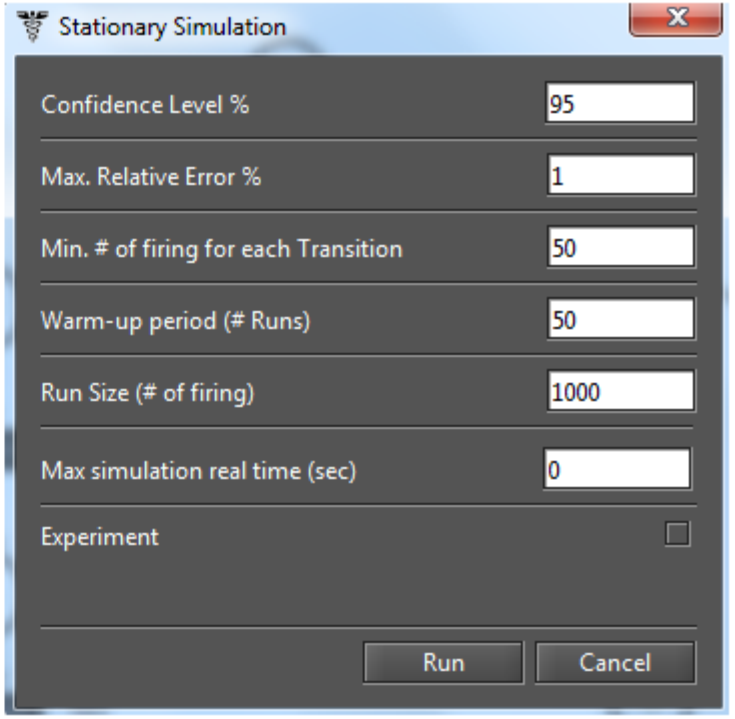
State 0: (M(P0)=1, M(P1)= M(P2)= M(P3)= M(P4)= M(P5)=0)

The process repeats until a stop criterion is reached.

C:\Users\Paulo Maciel\Dropbox\Models\Mercury\Mercury 4.4.3\Mercury 4.4.3.2\Examples\StudyOnSimulation.xml

Discrete Event Simulation: a SPN example






```
Metric : M1, P{#P0=1}
Result: 0.28296016187870443
Confidence Interval: [0.2801415896149667,0.28577873414244215]
Error %: 0.996102152693122
Run size: 1000
Number of Runs 397000
Total Runs 397000000
```

Random Number Generator

Stochastic simulation is deeply based on sequences of values generated from of random variates. Random variates require methods for generating random numbers.

Since the generation methods should be computationally efficient as well accurate, a balance between efficiency and the accuracy is very important requirement.

Excel

Random Number Generator

Generation of Uniform Random Numbers

The most common method for generating a random number is to use a recursive function in which the next number in the sequence is obtained from the last one or two numbers, that is:

$$x_n = f(x_{n-1}, x_{n-2}, \ldots)$$

For instance:

$$x_n = 5x_{n-1} + 1 \bmod 16$$

Starting with $x_0 = 5$, x_1 is obtained as follows:

$$x_1 = 5(5) + 1 \bmod 16$$
$$x_1 = 26 \bmod 16$$
$$x_1 = 10$$

Random Number Generator

Excel

Generation of Uniform Random Numbers

The first 32 numbers obtained by this method are:

10, 3, 0, 1, 6, 15, 12, 13, 2, 11, 8, 9, 14, 7, 4, 5
10, 3, 0, 1, 6, 15, 12, 13, 2, 11, 8, 9, 14, 7, 4, 5.

The *x*s are integers between 0 and 15.

By dividing *x*s by 16, a sequence of random numbers between 0 and 1 is obtained, hence:

0.6250, 0.1875, 0.0000, 0.0625, 0.3750, 0.9375, 0.7500, 0.8125, 0.1250, 0.6875, 0.5000, 0.5625, 0.8750, 0.4375, 0.2500, 0.3125, 0.6250, 0.1875, 0.0000, 0.0625, 0.3750, 0.9375, 0.7500, 0.8125, 0.1250, 0.6875, 0.5000, 0.5625, 0.8750, 0.4375, 0.2500, 0.3125.

Excel

Random Number Generator

Generation of Uniform Random Numbers

It is obvious that we can regenerate the sequence provided the starting value (seed) x_0 is given. Hence, $f(x_{n-1}, x_{n-2},\ldots)$ is deterministic. However, as the sequence would pass statistical tests for randomness, these numbers are called **pseudo-random**.

In simulations, pseudo-random numbers are preferable to *random* numbers because it is often desirable to be able to repeat an experiment exactly as it was done before. And if a different result is needed, change the seed before the next simulation.

Random Number Generator

Excel

Generation of Uniform Random Numbers

Notice only the first 16 numbers are unique, the 17th number is the same as the first and the remaining sequence is a repetition of the first 16 numbers. Hence the random-number generator has **period** of 16.

0.6250, 0.1875, 0.0000, 0.0625, 0.3750, 0.9375,
0.7500, 0.8125, 0.1250, 0.6875, 0.5000, 0.5625,
0.8750, 0.4375, 0.2500, 0.3125, 0.6250, 0.1875,
0.0000, 0.0625, 0.3750, 0.9375, 0.7500, 0.8125,
0.1250, 0.6875, 0.5000, 0.5625, 0.8750, 0.4375,
0.2500, 0.3125.

Excel

Random Number Generator

Generation of Uniform Random Numbers

The desired properties of the generator:

1. *efficiently computable.*

2. *large period.*

3. *successive values should be independent and uniformly distributed.*

The third requires a lot of test.

Some random-number generators:

- Linear-congruential generators
- Tausworthe generators
- Extended Fibonacci generators
- Combined generators

Excel

Random Number Generator

Generation of Uniform Random Numbers

$$X(k) = [aX(k-1) + c] \bmod M$$

where M is the modulus, $M > 0$, a large (prime) integer value; a is the multiplier, $0 < a < M$; c is the increment, usually = 1 or 0; and $X(0)$ is the seed, $0 < X(0) < M$. The algorithm is executed in integer arithmetic. The seed value $X(0)$ is provided by the user.

The equation produces a random sequence of integer values in the range $0 \le X(k) \le M-1$ and this sequence can be converted into a random sequence of uniform random numbers in the interval [0, 1] by executing the floating-point operation

$$U(k) = \text{Float}[X(k)/M]$$

a, c, and M should be uncorrelated.

Example:

$$X(k) = (16{,}807 \times X(k-1)\ ) \ Mod\ (2^{31} - 1)$$

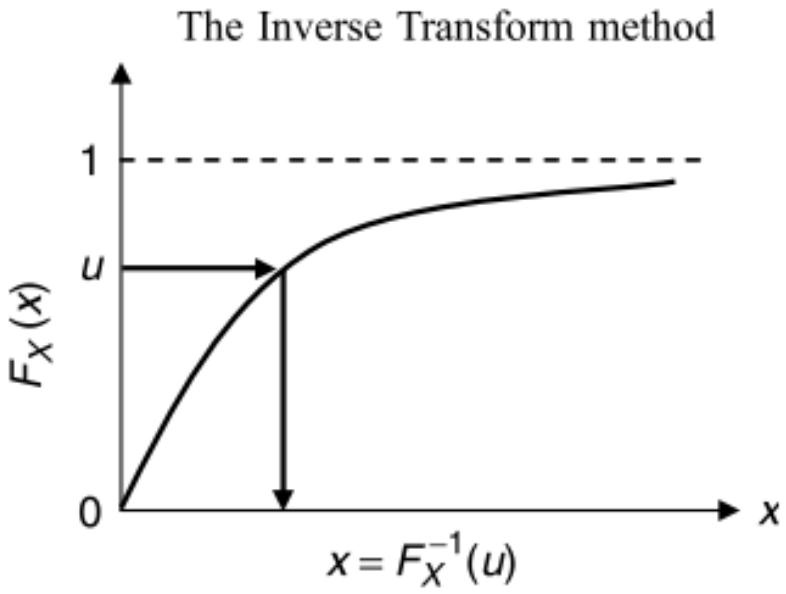
Random Variate Generator

- A random variable is a function.
- A random variate is an outcome provided by a random variable.

Random Variate Generator

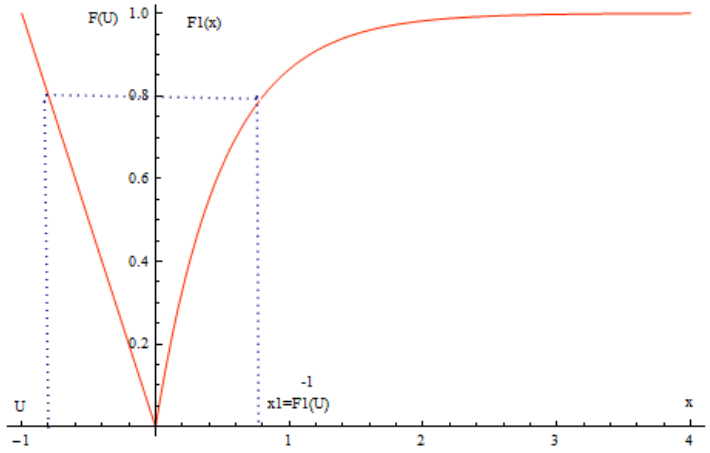
- Basic methods
  - Inversion transform
  - Convolution
  - Composition
  - Characterization
  - Acceptance-Rejection

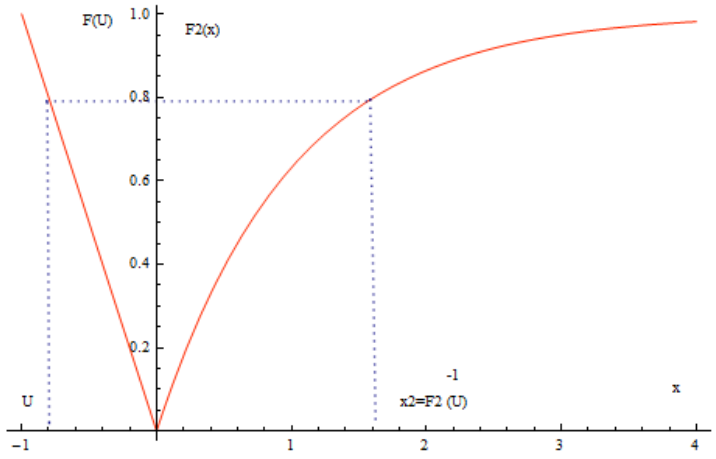
Random Variate Generator
Inversion transform



Transform method as a two-step algorithm:

1. Use your favorite RNG to generate a realization u from a variate $U \sim \text{Unif}(0, 1)$.
2. Compute $x = F_X^{-1}(u)$ as a realization of X.





Random Variate Generator
Inversion transform

Excel Exponential

Mathematica Exponential

The sequence of random numbers will not be identical, but will be statistically equivalent.

Generation of Exponential Variate

Suppose we wish to generate a realization x from the exponential distribution with rate $\lambda = 0.5$.

$$u = F_{U(0,1)}(u) = F_X = 1 - e^{-\lambda x}$$

$$u = 1 - e^{-\lambda x}$$

$$1 - u = e^{-\lambda x} =$$

$$\ln(1-u) = \ln e^{-\lambda x} =$$

$$\ln(1-u) = -\lambda x$$

Since u was generated from $U(0,1)$, then $1-u$ is also obtained from $U(0,1)$, so:

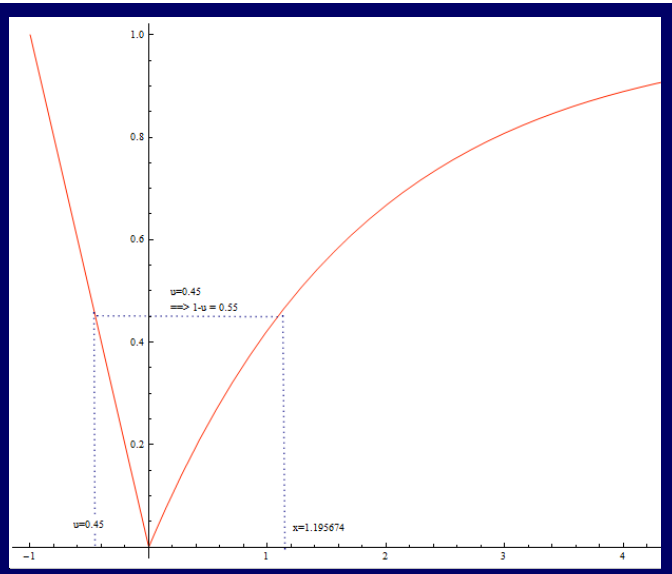
$$\ln u = -\lambda x$$

Hence:

$$x = -\frac{\ln u}{\lambda}$$

If $u = 0.55$ was obtained and since $\lambda = 0.5$, hence:

$$x = 1.195674$$



Random Variate Generator
Inversion transform

Excel Exponential

Mathematica Exponential

Generation of Exponential Variate

Example: to generate an exponential variate with λ, follow this procedure:

$$F(t) = \begin{cases} 1 - e^{-\lambda t} & t \geq 0 \\ 0 & t < 0 \end{cases}$$

Through its inversion, $t_i = -\frac{1}{\lambda} \times \ln(1 - u_i)$ is obtained.

Exponential variate t_i can be generated by a uniform variate u_i and the preceding equation.

Since u_i is uniformly distributed between 0 and 1, $1 - u_i$ is also uniformly distributed between 0 and 1, then:

$$t_i = -\frac{1}{\lambda} \times \ln(u_i) \Leftrightarrow t_i = -\frac{1}{\lambda} \times \ln(1 - u_i)$$

Random Variate Generator
Inversion transform

Excel Exponential

Mathematica Exponential

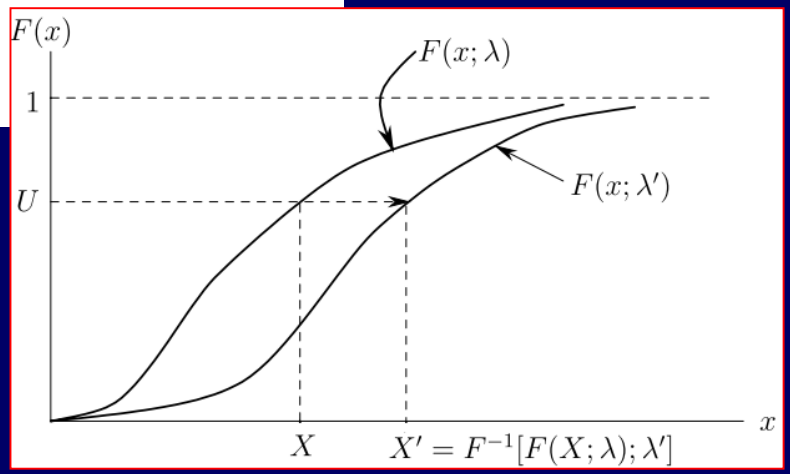
Generation of Exponential Variate

What would the random variate have been if the parameter λ had been replaced by a new value λ'? We can answer this question by arguing as follows. If X had been generated through the inverse transform technique, there would have been some random number U giving rise to X through $U = F(U)$, such that:

$$X = F^{-1}(U, \lambda)$$

Now if this same random number U had been used to generate a random variate X from the new $F(U, \lambda')$, we would get

$$X' = F^{-1}(U, \lambda')$$



Random Variate Generator
Inversion transform

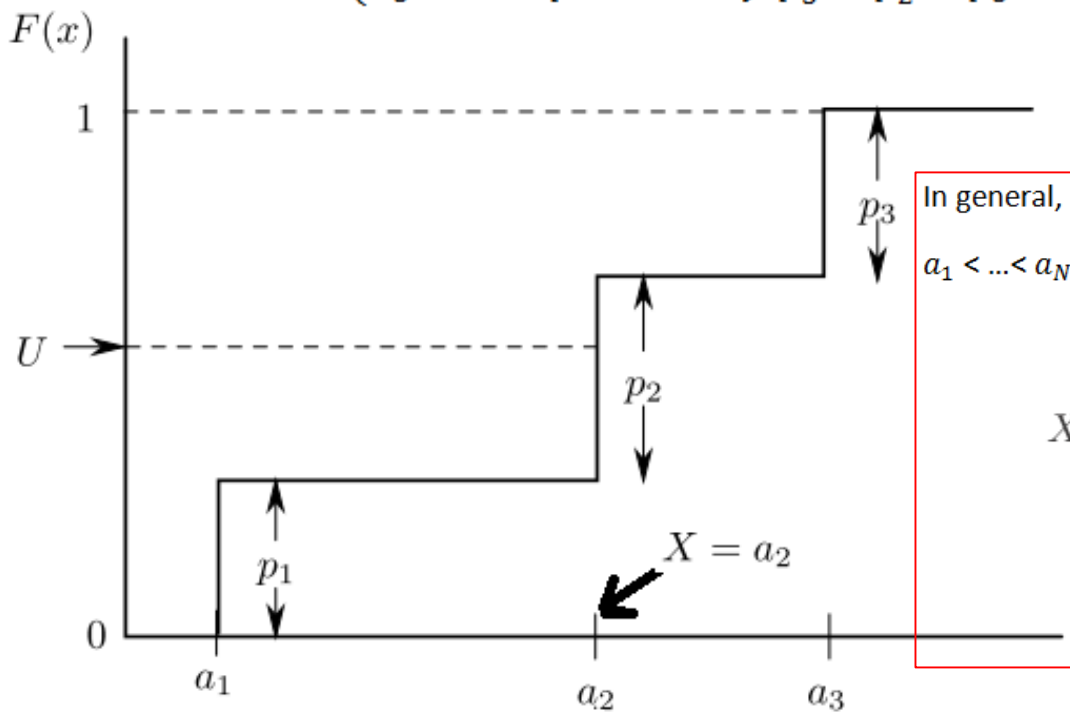
Excel Discrete

Mathematica Discrete

Discrete random variate

The inverse transform technique is quite general. It can be used, for example, to sample from distributions of discrete random variables. Figure shows a random variable X that can take three values $a_1 < a_2 \leq a_3$:

$$X = \begin{cases} a_1 & \text{with probability } p_1 \\ a_2 & \text{with probability } p_2' = p_1 + p_2 \\ a_3 & \text{with probability } p_3' = p_2' + p_3 \end{cases}$$



In general, for a discrete random variable X that can take N values $a_1 < \ldots < a_N$ with corresponding probabilities $p_1, \ldots, p_N$, we have

$$X = \begin{cases} a_1 & \text{if } 0 \leq U \leq p_1 \\ a_2 & \text{if } p_1 < U \leq p_1 + p_2 \\ \vdots & \\ a_N & \text{if } p_1 + p_2 + \ldots + p_{N-1} < U \leq 1 \end{cases}$$

or, equivalently,

$$X = \min\{a_n : F(a_n) \geq U, n = 1, \ldots, N\}$$

Random Variate Generator
Inversion transform

Excel Discrete

Mathematica Discrete

Discrete random variate

	a1=	1	0≤U≤p1	0	0.2		U	FALSE	RV
	a2=	4	p1<U≤p2	0.2	0.5		0.296579	TRUE	4
X	a3=	6	p2<U≤1	0.5	1			FALSE	

Specification
Unif. Rand Gen.
Generated RV - output

In general, for a discrete random variable X that can take N values

$a_1 < \ldots < a_N$ with corresponding probabilities $p_1, \ldots, p_N$, we have

$$X = \begin{cases} a_1 & \text{if } 0 \le U \le p_1 \\ a_2 & \text{if } p_1 < U \le p_1 + p_2 \\ \vdots & \\ a_N & \text{if } p_1 + p_2 + \ldots + p_{N-1} < U \le 1 \end{cases}$$

or, equivalently,

$$X = \min\{a_n : F(a_n) \ge U, n = 1, \ldots, N\}$$

Random Variate Generator
Inversion transform

TABLE: Applications of the Inverse-Transform Technique

Distribution	CDF $F(x)$	Inverse
Exponential	$1 - e^{-x/a}$	$-a \ln(u)$
Extreme value	$1 - e^{-e^{(x-a)/b}}$	$a + b \ln \ln u$
Geometric	$1 - (1-p)^x$	$\left\lceil \frac{\ln(u)}{\ln(1-p)} \right\rceil$
Logistic	$1 - \frac{1}{1 + e^{(x-\mu)/b}}$	$\mu - b \ln\left(\frac{1}{u} - 1\right)$
Pareto	$1 - x^{-a}$	$1/u^{1/a}$
Weibull	$1 - e^{(x/a)^b}$	$a(\ln u)^{1/b}$

One drawback of the inverse transform technique is that it may not always be possible to evaluate $F^{-1}(U)$ in closed form.

This problem arises with a number of common distributions (e.g., the normal distribution).

Random Variate Generator
Convolution

Excel Erlang

Mathematica Erlang

Convolution

This technique can be used if the random variable X can be expressed as a sum of n random variables $Y_1, Y_2, \ldots, Y_n$, such that:

$$X = Y_1 + Y_2 + \ldots + Y_n$$

If X is a sum of two random variables Y_1 and Y_2, then the pdf of X can be obtained analytically by a convolution of the pdf's of Y_1 and Y_2.

This is why the technique is called convolution, although no convolution is required in random-number generation.

An example: an Erlang-k variate is the sum of k exponential variates. so, it can be obtained by generating k exponential variates and summing them.

U1	0.382	λ	100
Y1	0.009623	k	4
U2	0.449568		RV specification
Y2	0.007995		U[0,1] Random generation
U3	0.879696		Internal computation
Y3	0.001282		Result
U4	0.967559		
Y4	0.00033		
X	0.01923		

Random Variate Generator
Convolution

Excel Erlang

Mathematica Erlang

Examples of applications of this technique:

- An Erlang-k variate is the sum of k exponential variates. so, it can be obtained by generating k exponential variates and summing them.
- A binomial variate with parameters n and p is a sum of n Bernoulli variates with success probability p. Thus, a binomial variate generated by n U(O, 1) random numbers and returning the number of random numbers that are less than p.
- The chi-square distribution with v degrees of freedom is a sum of squares of v unit normal $N(0,1)$ va. iates.
- The sum of two gamma variates with parameters $(a, b1)$ and $(a, b2)$ is a gamma variate with parameter$(a, b1 + b2)$. Thus, a gamma variate with a noninteger value of b parameter can be obtained by adding two gamma variates—one with integer b and the other with the fractional b.
- The sum of a large number of variates from any distribution has a normal distribution. This fact is used to generate normal variates by adding a suitable number of $U(0,1)$ variates.
- The sum of m geometric variates is a Pascal variate.
- The sum of two uniform variates has a triangular density.

Random Variate Generator
Composition

Excel
Hyper-Exponential

Mathematica
Hyper-Exponential

Mixing or Composition Method: Suppose that the distribution $F_Y(y)$ or the density function $f_Y(y)$ can be represented by either of the following forms:

1. $F_Y(y) = \alpha_1 F_{X_1}(y) + \alpha_2 F_{X_2}(y) + \ldots + \alpha_k F_{X_k}(y)$,
2. $f_Y(y) = \alpha_1 f_{X_1}(y) + \alpha_2 f_{X_2}(y) + \ldots + \alpha_k f_{X_k}(y)$,

where $\alpha_1, \ldots, \alpha_k$ are non-negative and sum to one. Given that the $X_i's$ are relatively easy to generate, these kind of variates can be generated using the composition method.

Random Variate Generator
Composition

Excel
Hyper-Exponential

Mathematica
Hyper-Exponential

The *hyperexponential* distribution is given by

$$F(x) = \sum \alpha_i (1 - e^{-\lambda_i x}) \qquad x \geq 0,\ \lambda_i,\ \alpha_i > 0 \text{ and } \sum \alpha_i = 1\,.$$

The random variate for the hyperexponential can be generated in two steps. Consider, for example, a three-stage hyperexponential distribution with parameters α_1, α_2, α_3 and λ_1, λ_2, λ_3. First a uniform random number u is generated and the following inverse function is used in the first step:

$$\Phi(u) = \begin{cases} 1\,, & 0 < u \leq \alpha_1\,, \\ 2\,, & \alpha_1 < u \leq \alpha_1 + \alpha_2\,, \\ 3\,, & \alpha_1 + \alpha_2 < u \leq 1\,. \end{cases}$$

The desired variate is then given by

$$x = \mathbf{1}_{\{u \leq \alpha_1\}} \left(-\frac{\ln(u_1)}{\lambda_1} \right) + \mathbf{1}_{\{\alpha_1 < u \leq \alpha_1 + \alpha_2\}} \left(-\frac{\ln(u_1)}{\lambda_2} \right) + \mathbf{1}_{\{\alpha_1 + \alpha_2 < u \leq 1\}} \left(-\frac{\ln(u_1)}{\lambda_3} \right)$$

Random Variate Generator
Composition

Excel Hyper-Exponential

Mathematica Hyper-Exponential

	1	0<u≤α1	0	0.2		u	FALSE	Φ(u)
	2	α1<u≤α1+α2	0.2	0.5		0.224158	TRUE	2
Φ(u)	3	α1+α2<u≤1	0.5	1			FALSE	

u1	0.555162	λ1	50	λ2	100	λ3	150	
x	0.005885							

Specification
Unif. Rand Gen. - Input
Internal
Generated RV - output

The *hyperexponential* distribution is given by

$$F(x) = \sum_i \alpha_i (1 - e^{-\lambda_i x}) \qquad x \geq 0,\ \lambda_i,\ \alpha_i > 0 \text{ and } \sum \alpha_i = 1 .$$

$$\Phi(u) = \begin{cases} 1, & 0 < u \leq \alpha_1 , \\ 2, & \alpha_1 < u \leq \alpha_1 + \alpha_2 , \\ 3, & \alpha_1 + \alpha_2 < u \leq 1 . \end{cases}$$

The desired variate is then given by

$$x = \mathbf{1}_{\{u \leq \alpha_1\}} \left(-\frac{\ln(u_1)}{\lambda_1} \right) + \mathbf{1}_{\{\alpha_1 < u \leq \alpha_1 + \alpha_2\}} \left(-\frac{\ln(u_1)}{\lambda_2} \right) + \mathbf{1}_{\{\alpha_1 + \alpha_2 < u \leq 1\}} \left(-\frac{\ln(u_1)}{\lambda_3} \right)$$

Random Variate Generator
Characterization

Excel
Normal

Mathematica
Normal

Characterization

Special characteristics of some distributions allow the respective random variates to be generated by specially tailored algorithms. These algorithms are generally classified as characterization methods.

Example: the Polar Method (Marsaglia and Bray (1964)) allows generating two independent Normal variates.

1. Generate two U(0, 1) variates u_1 and u_2.
2. Let $v_1 = 2u_1 - 1$ and $v_2 = 2u_2 - 1$
3. $r = v_1^2 + v_2^2$
4. If $r \leq 1$, go back to step 1; otherwise let $s = \sqrt{-2 \times \frac{\ln r}{r}}$ and return:

$$x_1 = \mu + \sigma \times v_1 \times s$$

$$x_2 = \mu + \sigma \times v_2 \times s$$

Random Variate Generator
Characterization

Excel
Normal

Mathematica
Normal

u1	0.508988	μ	100
u2	0.074252	σ	20
v1	0.017975		
v2	-0.1485		
r	0.022376		
s	1.949296		
x1	100.7008		
x2	94.21047		

RV specification

U[0,1] Random generation

Internal computation

Result

1. Generate two U(0, 1) variates u_1 and u_2.
2. Let $v_1 = 2u_1 - 1$ and $v_2 = 2u_2 - 1$
3. $r = v_1^2 + v_2^2$
4. If $r \leq 1$, go back to step 1; otherwise let $s = \sqrt{-2 \times \frac{\ln r}{r}}$ and return:

$$x_1 = \mu + \sigma \times v_1 \times s$$

$$x_2 = \mu + \sigma \times v_2 \times s$$

Random Variate Generator
Acceptance-Rejection

Excel

The Acceptance-Rejection method generates a random variate X from the density function $f_X(t)$.

The first step is to identify some function $g_Y(t)$ with the property:

$g(t) \geq f_X(t), \forall t.$

$g(t)$ is called a majorizing function, and it is clearly not unique.

Since $g(t)$ is generally not a density function, we determine next a normalization constant, c, which allows us to transform it into a density function. So, consider

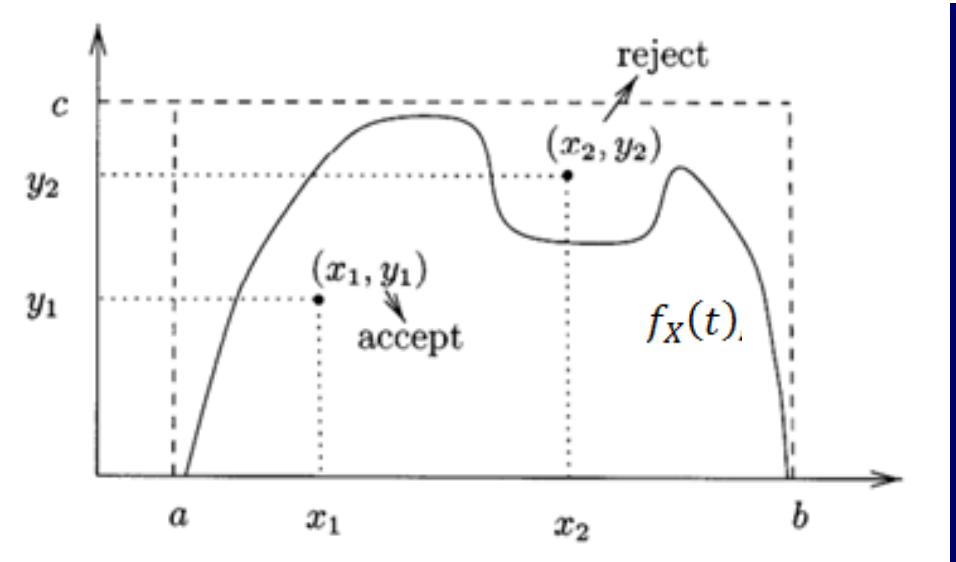
$$c = \int_{-\infty}^{+\infty} g(t)\, dt$$

and that $g_Y(t)$ was chosen such that $< \infty$, it possible defining a density function $h_Y(t) = \frac{g(t)}{c}$.

Random Variate Generator
Acceptance-Rejection

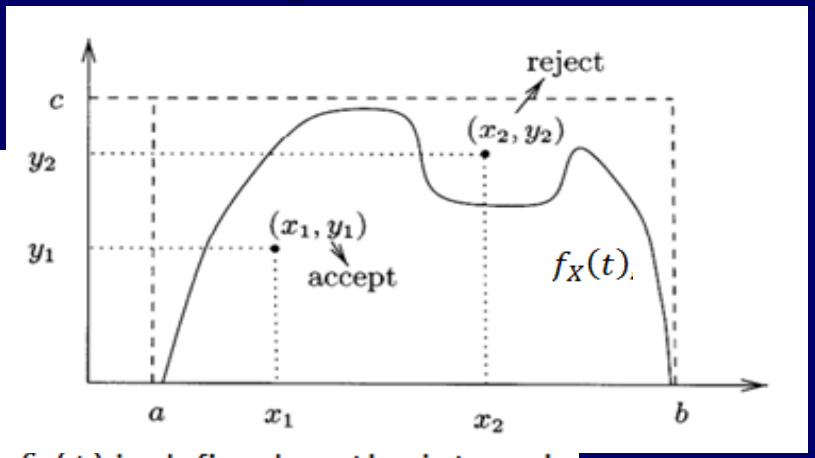
Excel

Owing to the complex shape of $f_X(t)$, the simplest bounding function is the rectangle. Random points within this box can be easily generated by selecting a random value x that is uniformly distributed on the interval $[a\,, b]$ and a value y that is uniformly distributed on $[0\,, c]$. Then, if $y \leq f_X(t)$, we accept x as a sample from the desired distribution. Otherwise, we reject x and repeat the process, beginning with the generation of new values for x and y.



Random Variate Generator
Acceptance-Rejection

Excel



Therefore: $g(t) \geq f_X(t), \forall t$

$$c \times h_Y(t) \geq f_X(t), \forall t$$

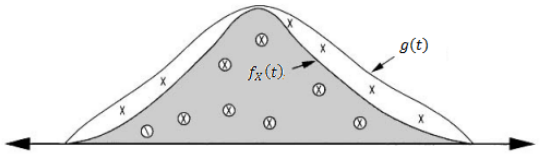
$$\frac{f_X(t)}{h_Y(t)} \leq c, \forall t$$

Let us assume that the density function $f_X(t)$ is defined on the interval $[a, b]$.

Pseudo-code

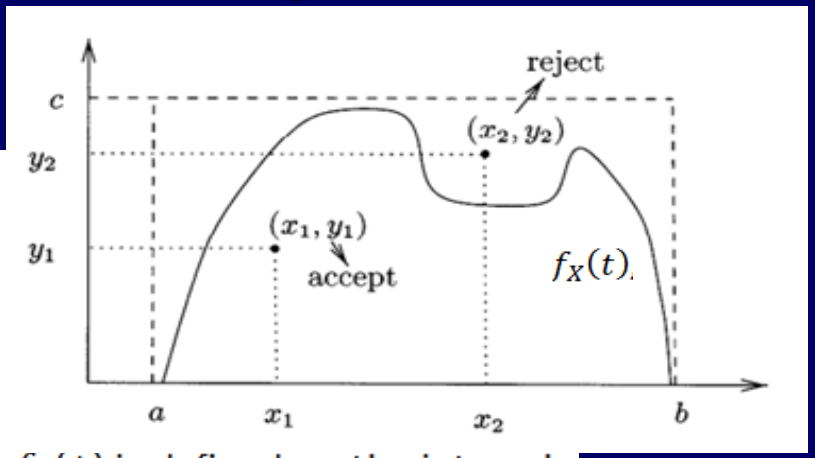
1. generate two uniformly distributed random numbers, u_1 and u_2 on [0.1]
2. set x_1 =a+(b-a) u_1
3. set $y_1 = cu_2$
4. (x_1, y_1) = (a+(b-a) u_1 , cu_2) // The tuple x_1, y_1) is a randomly // selected point in the rectangle // $[a, b] \times [0, c]$.
5. IF $y_1 \leq f_X(x_1)$
   5.1. $y = y_1$ (Accept)
6. Else

In fact, choosing a choosing a convenient majorizing function can enhance the efficiency of the method, since the percentage of rejection may be reduced.



Random Variate Generator
Acceptance-Rejection

Excel



Therefore: $g(t) \geq f_X(t), \forall t$

$$c \times h_Y(t) \geq f_X(t), \forall t$$

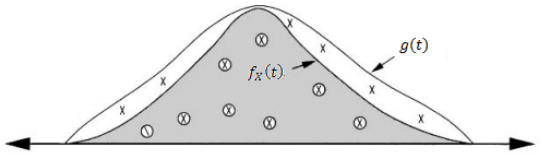
$$\frac{f_X(t)}{h_Y(t)} \leq c, \forall t$$

Let us assume that the density function $f_X(t)$ is defined on the interval $[a, b]$.

Pseudo-code

1. generate two uniformly distributed random numbers, u_1 and u_2 on [0.1]
2. set x_1 =a+(b-a) u_1
3. set $y_1 = cu_2$
4. (x_1, y_1) = (a+(b-a) u_1 , cu_2) // The tuple x_1, y_1) is a randomly // selected point in the rectangle // $[a, b] \times [0, c]$.
5. IF $y_1 \leq f_X(x_1)$
   5.1. $y = y_1$ (Accept)
6. Else

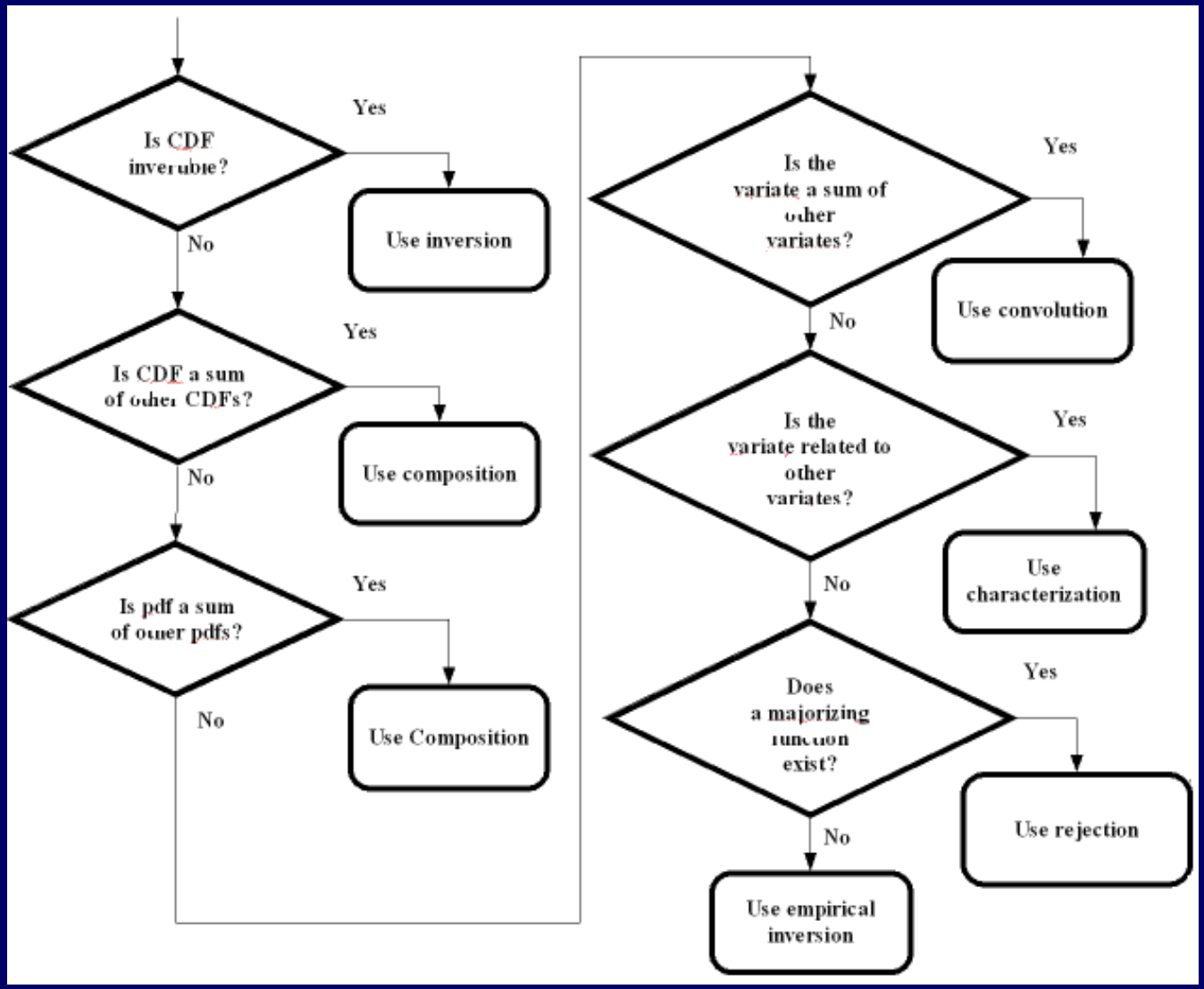
In fact, choosing a choosing a convenient majorizing function can enhance the efficiency of the method, since the percentage of rejection may be reduced.



Random Variate Generator

Deciding which method to adopt

Excel



Excel

Output Analysis

The output data of simulations can be thought as estimate some quantity of interest θ.

For example, θ may be the mean of residence time distribution of customers in a bank, and X_k is the observed residence time of the k^{th} customer.

A point estimate of θ, denoted by $\hat{\theta}$ is a number that represents a "guess" of θ based on collected data $X_1, X_2, \ldots, X_n$.

An interval estimate of θ provides a range of numbers defined by

$[\hat{\theta} - \alpha_1, \hat{\theta} + \alpha_2]$ in which the true value of θ lies within with a given probability.

Excel

Output Analysis

Point estimation

The simplest and most common estimation problem arises when we collect a sequence of *iid* random variables $X_1, X_2, \ldots, X_n$ characterized by a single probability distribution function. Let θ be the mean of that distribution.

To obtain a point estimate of θ, $\hat{\theta}_n$, based on n samples, we use the sample mean:

$$\hat{\theta}_n = \bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

Excel

Output Analysis

Point estimation

It is important to keep in mind that $\hat{\theta}_n$ is a random variable. So, we are using a random variable to estimate a real number θ.

Therefore, an unbiased estimator of θ is $E[\hat{\theta}_n]$ $(E[\hat{\theta}_n] = \theta)$.

$Var[\hat{\theta}_n] = \frac{\sigma^2}{n}$ is an unbiased estimator of the variance of the distribution of $\hat{\theta}_n$. As σ^2 is unknown, we adopt the sample variance of our collected data, defined by

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

as an estimate of σ^2.

So:

$$Var[\hat{\theta}_n] = \frac{S^2}{n} \qquad \sigma[\hat{\theta}_n] = \sqrt{Var[\hat{\theta}_n]} = \frac{S}{\sqrt{n}}$$

Output Analysis

Excel

Point estimation

Summary:

$$\bar{X} = \sum_{i=1}^{n} X_i \quad \text{(estimator of } \theta\text{)}$$

$$\sigma[\hat{\theta}_n] = \frac{S}{\sqrt{n}} \text{ (standard deviation of our estimator } \hat{\theta}_n\text{)}$$

Adopt the following approach:

1. Choose a value of n (where n is at least 30) and generate n samples $X_1, X_2, \ldots, X_n$.
2. Compute the sample mean

$$\bar{X} = \sum_{i=1}^{n} X_i$$

3. If $\frac{S}{\sqrt{n}} \leq \epsilon$, then stop. ϵ – error.
4. Otherwise generate additional samples and incorporate into the recursive formulae until the desired accuracy is attained. $\left(n \geq \left(\frac{S}{\epsilon}\right)^2\right)$

Output Analysis

Excel

Estimation considering confidence interval

If our output data $X_1, X_2, \ldots, X_n$ form an iid sequence, the unbiased point estimator $\theta_n \longrightarrow \theta$ as $n \longrightarrow \infty$ with probability 1.

As our data set is finite, we would like to develop define an interval that contains θ with some level of "confidence."

The basic tool for accomplishing this task is the Central Limit Theorem, where for large enough n, we have:

$$P\left[\hat{\theta}_n - Z_{\frac{\alpha}{2}}\sqrt{\sigma^2/n} \leq \theta \leq \hat{\theta}_n + Z_{\frac{\alpha}{2}}\sqrt{\sigma^2/n}\right] \approx 1 - \alpha$$

σ^2 may be replaced by the sample variance S_n^2, since S_n^2 approaches σ^2 as $n \longrightarrow \infty$.

Thus: $$P\left[\hat{\theta}_n - Z_{\frac{\alpha}{2}}\sqrt{S^2/n} \leq \theta \leq \hat{\theta}_n + Z_{\frac{\alpha}{2}}\sqrt{S^2/n}\right] \approx 1 - \alpha$$

Output Analysis

Excel

Estimation considering confidence interval

The obvious difficulty is the determination of how large n should be for the central limit theorem to hold. To partly overcome this difficulty, the t distribution can be adopted so that points denoted by $-t_{n-1,\alpha/2}$ and $t_{n-1,\alpha/2}$ can be determined such that

$$P\left[\hat{\theta}_n - t_{n-1,\frac{\alpha}{2}}\sqrt{S^2/n} \leq \theta \leq \hat{\theta}_n + t_{n-1,\frac{\alpha}{2}}\sqrt{S^2/n}\right] \approx 1-\alpha$$

Output Analysis

Types

- **Transient or terminating simulation**

  In transient simulation (terminating simulation) there is a particular point that limits the length of a run, for instance: a condition, a specified time instant, the completion of a trace etc.

- **Steady state or non-terminating simulation**

  Steady-state simulation (a non-terminating simulation) does not have this particular simulation end point. In such an evaluation, there is no “natural” reason why a simulation should finish other than measures’ accuracy.

Output Analysis

Issues in Obtaining Accurate Simulation Results

Before discussing the issues in obtaining accurate simulation results, a key difference between model performance and system performance needs to be highlighted.

The concern (here) is obtaining accurate results from the performance of the model, assuming the model is accurate.

The aim (here) is NOT evaluating how accurate the model predicts the system performance. This is the concern of model validation.

Output Analysis

Issues in Obtaining Accurate Simulation Results

Therefore, it is essential to stress that if the data used as input to the model is inaccurate, then accurate prediction of the system performance will not be provided.

Two key issues in assuring the accuracy of the estimates:

- Removal of any initialization bias:
  - warm-up and
  - initial conditions
- Ensuring that enough output data have been obtained to obtain accurate estimates:
  - long runs and
  - multiple replications

Output Analysis

Transient or terminating simulation

Excel

Initial number of replications

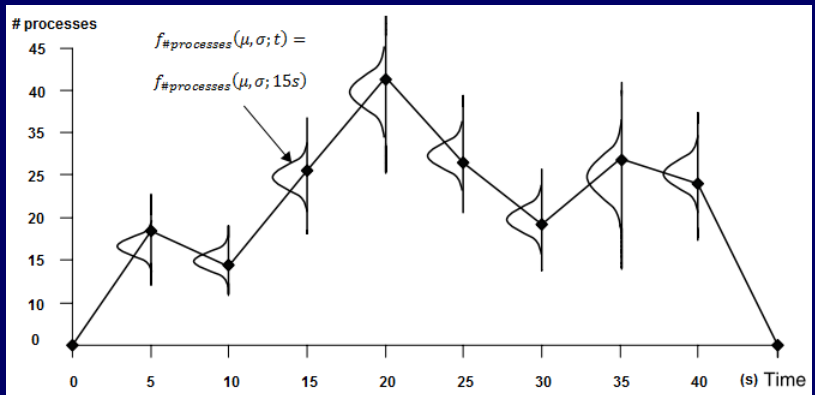
Independent replication simulation must begin with some arbitrary number of replications to begin the replication analysis. A small initial number of replications may be insufficient so that additional replications might be required. On the other hand, too much time may be wasted on unnecessary simulation runs if a too large number of initial replications is adopted.

A good practice is setting a reasonably small initial number of replications, such as ten. In general, this is sufficient to have statistical confidence for estimating the additional number of replications needed.

Output Analysis

Excel

Transient or terminating simulation



This type of simulation should be applied when we are interested in the transient value of some measure, i.e., number of processes in the system after 15 seconds of operation or the reliability of a system at 4000 hours.

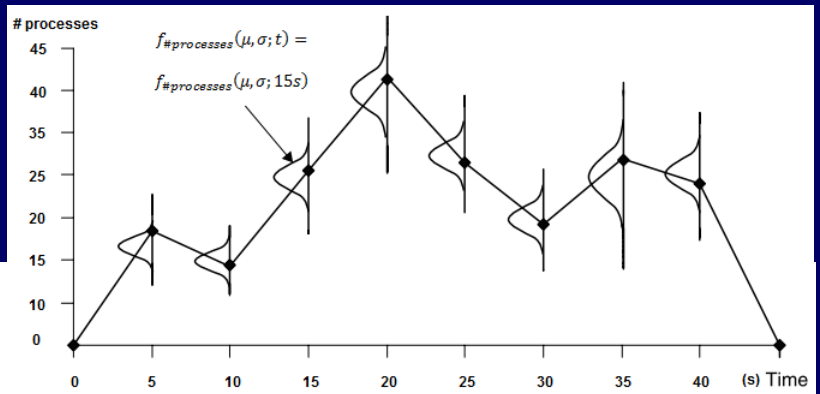
- In these cases each simulation run is conducted until the required simulated time and
- from each run a single sample value of the measure is collected.

Transient simulation is usually conducted by making m independent simulation runs, from which point and interval estimates of the required measure are obtained using the basic methods described before.

Output Analysis

Transient or terminating simulation

Excel



Independent Replication Method

A replication is a run of a simulation that uses a specific *iid* sequence $(X_1, X_2, \ldots, X_n)$. Multiple replications are conducted by changing the random number seed and re-running the simulation. The aim is to produce several samples for estimating the measure mean. An important issue is: how many replications need to be carried out?

A confidence interval for the mean for shows how precise the average of a value is being estimated.

Output Analysis

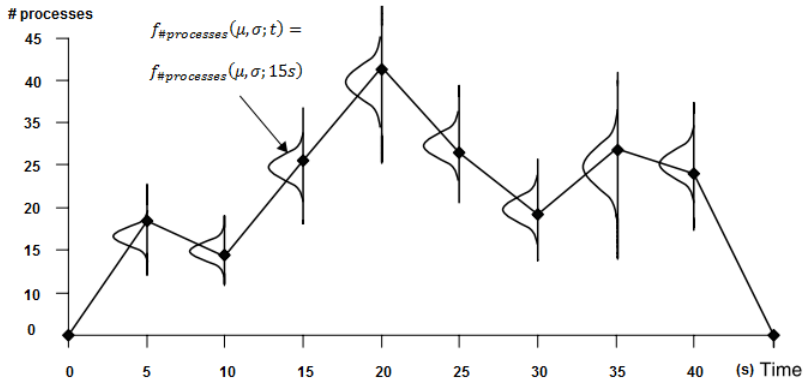
Transient or terminating simulation

Excel

Independent Replication Method

$$CI = \overline{X} \pm t_{n-1,\alpha/2}\frac{S}{\sqrt{n}}$$

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



$\overline{X}$ = mean of the output data from the replications
S = standard deviation of the output replications
n = number of replications
$t_{n-1,\alpha/2}$ = value from Student's t-distribution with $n-1$ degree of freedom and a significance level of $\alpha/2$
X_i = the result from replication i

The narrower the interval the more precise the estimate is. In general, the more sample data that are included in the interval, the narrower it becomes. When applying confidence intervals to simulation output, more replications are performed until the interval becomes sufficiently narrow to satisfy the user requirement.

Output Analysis

Transient or terminating simulation

Excel

Number of Replications required

$$\bar{X} - t_{n-1,\alpha/2} \times \frac{S}{\sqrt{n}} \leq \theta \leq \bar{X} + t_{n-1,\alpha/2} \times \frac{S}{\sqrt{n}}$$

$$t_{n-1,\alpha/2} \times \frac{S}{\sqrt{n}} \leq \epsilon$$

$$\left(t_{n-1,\alpha/2} \times \frac{S}{\sqrt{n}}\right)^2 \leq \epsilon^2$$

$$n \geq \left(t_{n-1,\alpha/2} \times \frac{S}{\epsilon}\right)^2$$

ϵ – absolute precision error required.

$$\%\epsilon = \frac{\frac{\epsilon}{\bar{S}}}{100} = \frac{100 \times \epsilon}{S}$$

$$\epsilon = \frac{S \times \%\epsilon}{100}$$

$$n \geq \left(t_{n-1,\alpha/2} \times \frac{S}{\epsilon}\right)^2$$

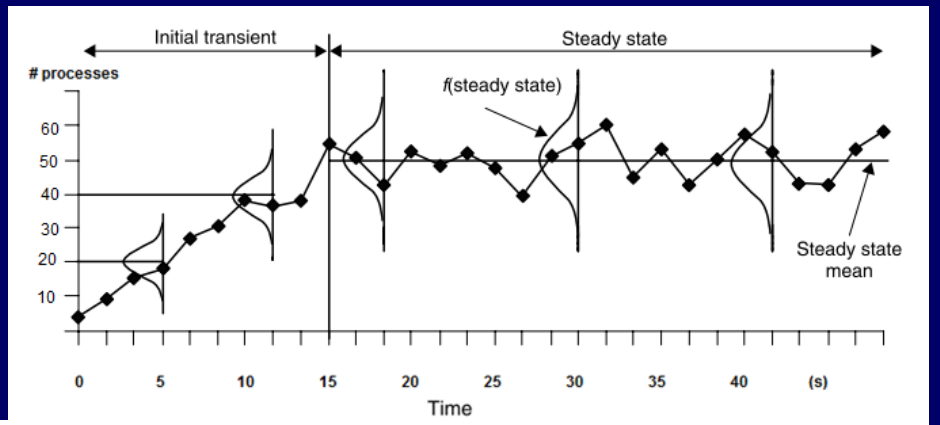
$$n \geq \left(t_{n-1,\alpha/2} \times \frac{S}{\frac{S \times \%\epsilon}{100}}\right)^2$$

$$n \geq \left(t_{n-1,\alpha/2} \times \frac{100}{\%\epsilon}\right)^2$$

$\%\epsilon$ – relative precision error required, that is the percentage deviation of the confidence interval about the mean.

Output Analysis

Steady State or Non-terminating Simulation



In steady-state simulation we are interested in estimating measures in steady state.

Steady state simulations are, in general, more computationally intensive than transient simulations.

A fundamental difference between steady-state and transient simulations is that, in the former, the transient phase should not affect the result. As transient phase length is unknown, this is an issue to be overcome.

Output Analysis

Steady State or Non-terminating Simulation

In principle, independent replications may be considered for steady-state simulation, but since the transient phase needs to be removed, this approach is not cost-effective.

A first problem to be addressed is estimating the length of the transient phase in order to remove it from the steady state evaluation. A basic guideline for accomplishing that is simply removing the first k samples.

Another approach is to observer the measure values until they seem to approach a regular pattern. A basic guideline is to simulate long enough so that the initial transient period becomes negligible. Obviously, this is not an efficient method.

Output Analysis

Steady State or Non-terminating Simulation

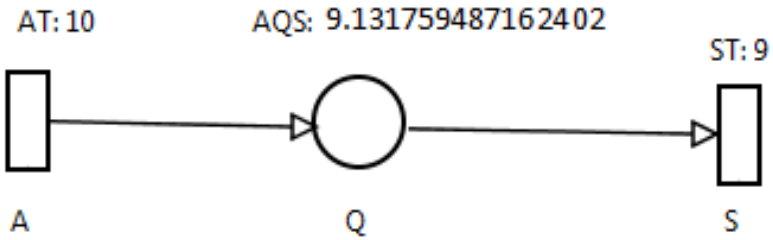
Three methods:

- Intelligent initialization
- Warm-up
- Batch means method

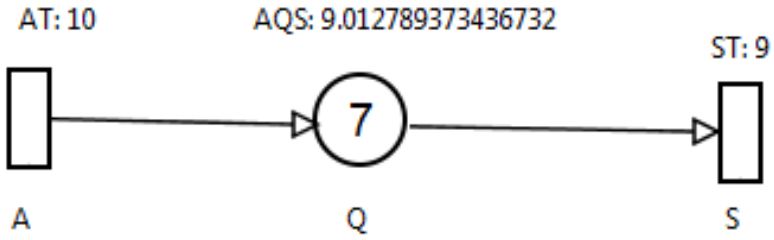
Output Analysis

Steady State or Non-terminating Simulation

- Intelligent initialization: it involves initialization of the simulation in a state that is more representative of long-run conditions.



Metric : AQS, E{#Q}
Result: 9.131759487162402
Confidence Interval: [9.040637417842197,9.222881556482607]
Error %: 0.9978588403287031
Run size: 1000
Number of Runs 721000
Total Runs 721000000

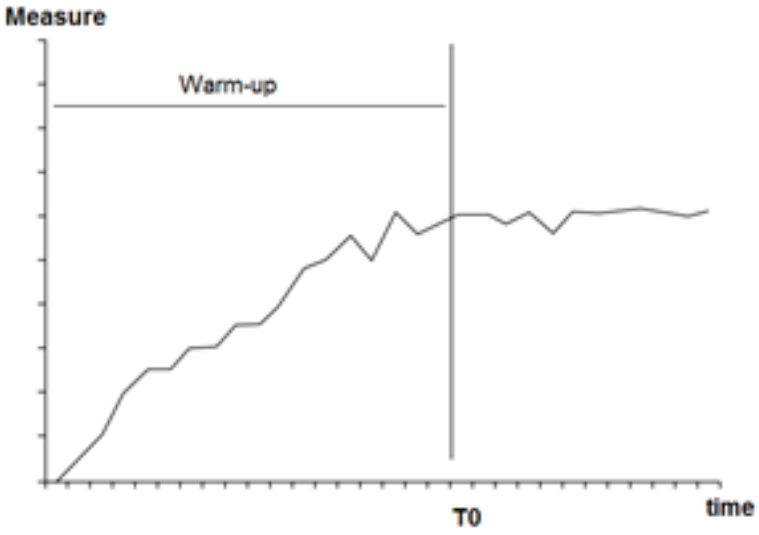


Metric : AQS, E{#Q}
Result: 9.012789373436732
Confidence Interval: [8.922897362211062,9.102681384662402]
Error %: 0.9973828023831057
Run size: 1000
Number of Runs 465000
Total Runs 465000000

Output Analysis

Steady State or Non-terminating Simulation

- Warm-up: a second method involves dividing the simulation into two phases. One of them is called the initialization phase (warm-up) from time 0 to T_0 and the other is called the data-collection phase from T_0 to $T_0 + T_E$.



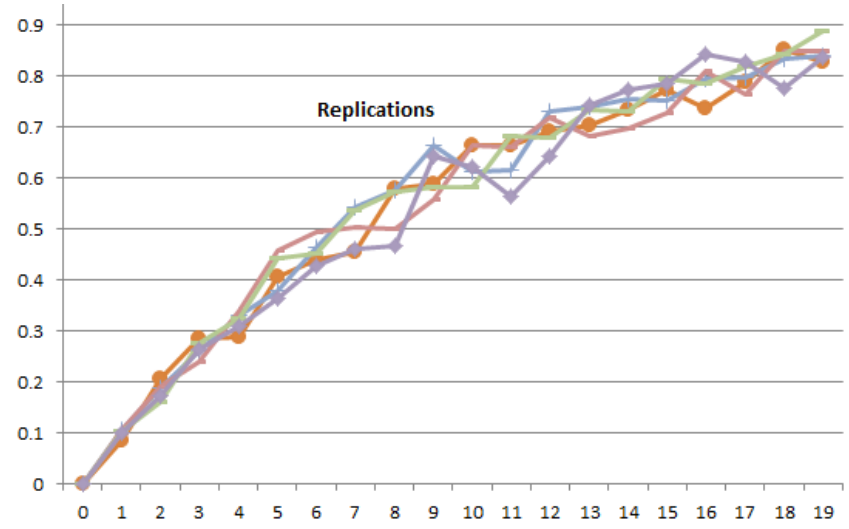
Identify the warm-up period as the point where the time-series becomes flat.

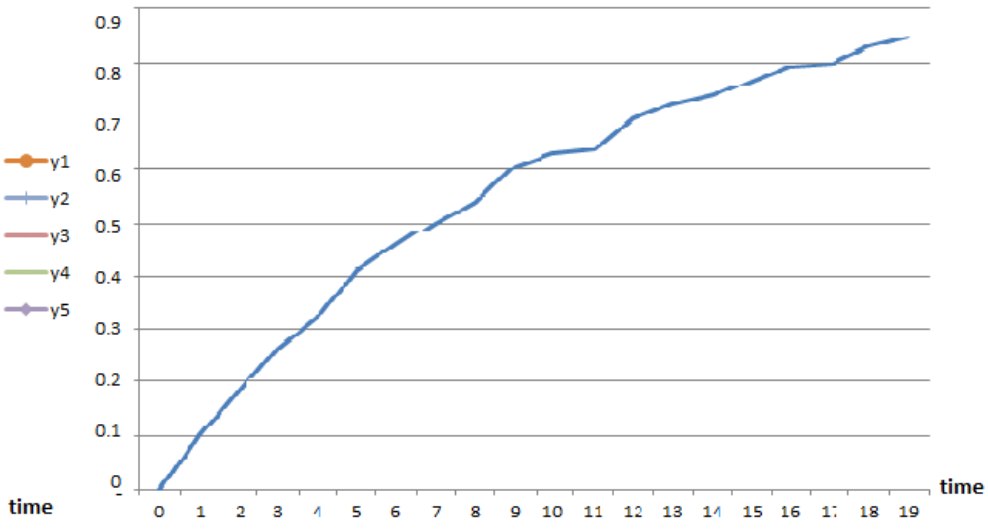
Output Analysis

Steady State or Non-terminating Simulation

Excel

- Warm-up
  - *Ensemble average* (for independent runs) becomes smoother and more precise as the number of replications increases. This method may additionally consider:
    - *Cumulative average* or
    - *Moving average* or
    - *Linear Regression* methods.





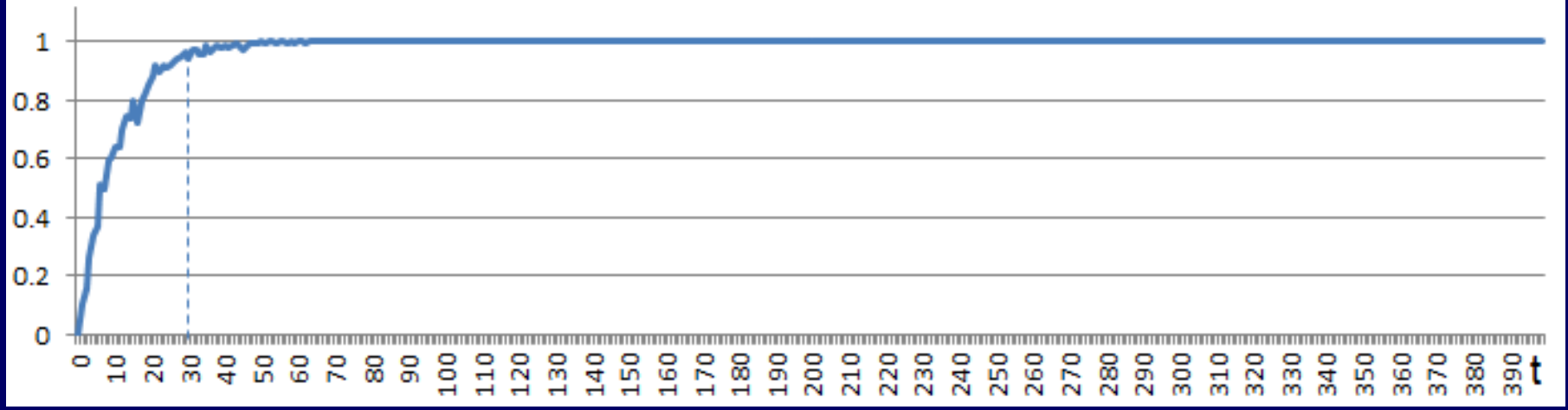
Output Analysis

Steady State or Non-terminating Simulation

Excel

t	y1	$\bar{y}$	w	ε<	knee CUSUM
0	0	0	1	0.01	30
1	0.103451	0.051725	2		
2	0.1556	0.08635	3		
3	0.26272	0.130443	4		
4	0.340089	0.172372	5		
27	0.938762	0.648694	28		
28	0.945423	0.658926	29		
29	0.962281	0.669038	30		
30	0.937825	0.677709	31		
31	0.969615	0.686831	32		

- Warm-up
  - *Cumulative average* (of long runs) becomes less variable as more data are averaged.



Output Analysis

Steady State or Non-terminating Simulation

Excel

- Warm-up
  - *Moving averages* (averages over a single run, considering a window size) become less variable as data reach a steady state.

The moving averages are calculated using the following formula:

$$\overline{Y}_i(w) = \begin{cases} \dfrac{\sum_{s=-(i-1)}^{i-1} Y_{i+s}}{2i-1} & \text{if } i = 1, \ldots, w \\ \dfrac{\sum_{s=-w}^{w} Y_{i+s}}{2w+1} & \text{if } i = w+1, \ldots, m-w \end{cases}$$

where:

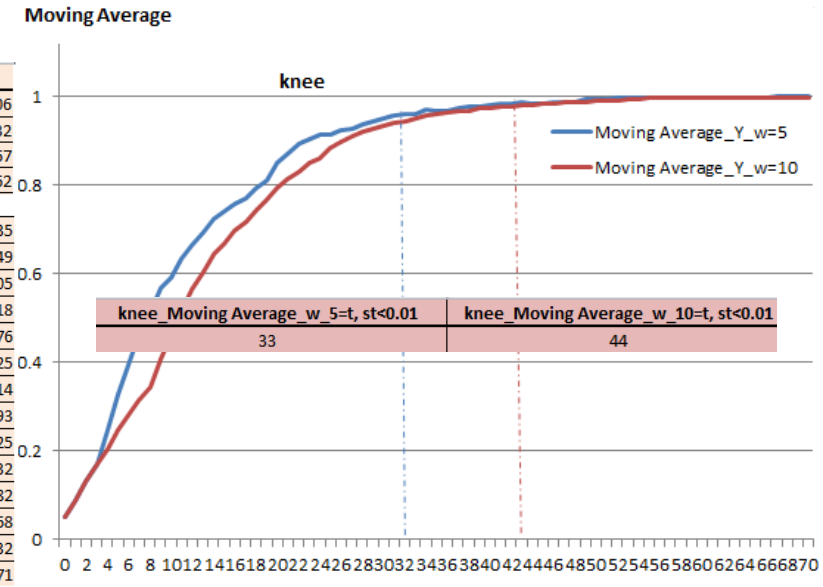
$\overline{Y}_i(w)$ = moving average of window size w

Y_i = time-series of output data

i = period number

m = number of periods in the simulation run

t	y1	Moving Average_Y_w=5	Moving Average_Y_w=10
0	0	0.051725306	0.051725306
1	0.103451	0.086350332	0.086350332
2	0.1556	0.130442667	0.130442667
3	0.26272	0.172371962	0.172371962
32	0.971854	0.960028123	0.944207535
33	0.958565	0.958810413	0.948554449
34	0.956193	0.968504045	0.955784705
35	0.986293	0.966758709	0.958770018
36	0.960888	0.967279171	0.962339476
37	0.974457	0.972458327	0.966243225
38	0.98446	0.976661215	0.967735814
39	0.977207	0.97692214	0.972713093
40	0.987598	0.980541142	0.973649925
41	0.978983	0.983266294	0.975272732
42	0.988082	0.984445836	0.978452082
43	0.990358	0.98609592	0.981378568
44	0.985458	0.982732323	0.979827232
45	0.97078	0.984422279	0.98248171



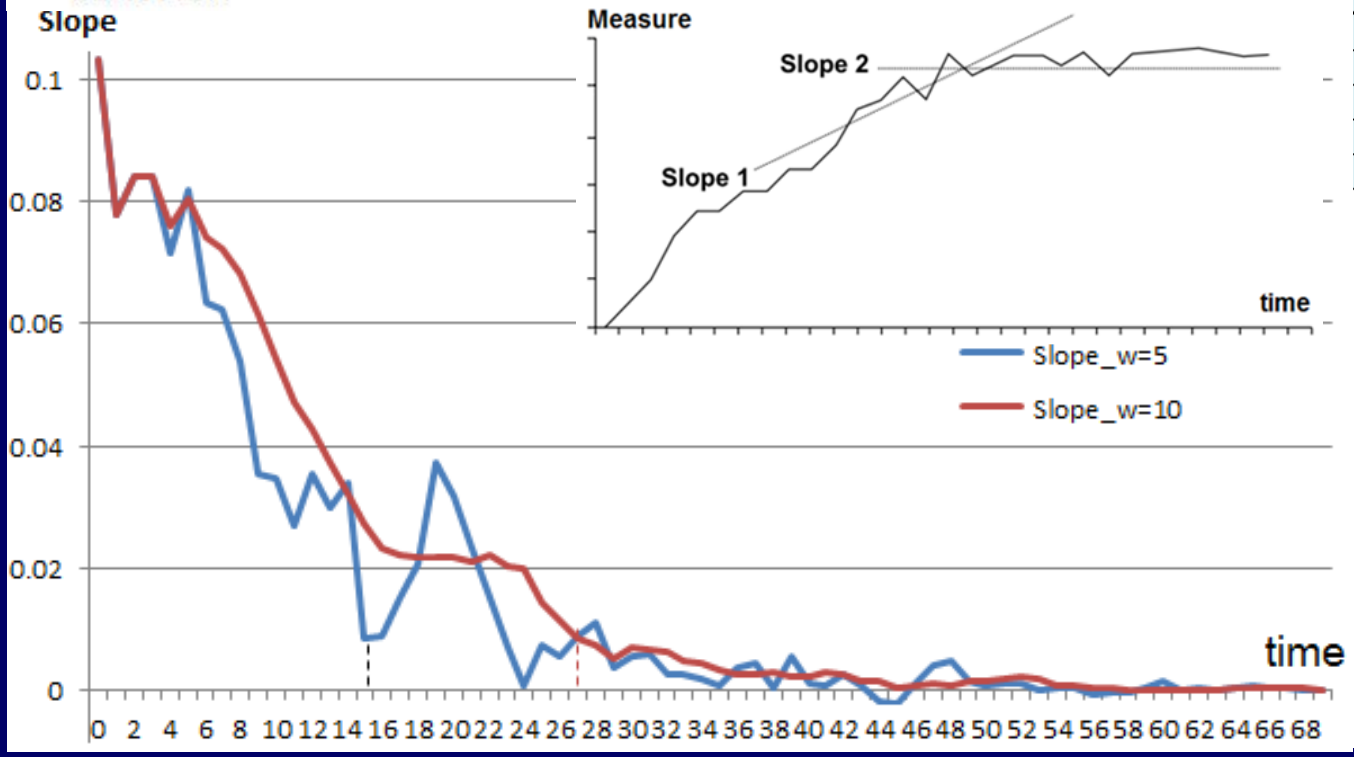
Output Analysis

Steady State or Non-terminating Simulation

Excel

- *Linear Regression Approach*

The linear regression approach adopts the least-squares method to determine if the linear regression slope coefficient is approaching zero. If the slope for a given range of observations is not close enough to zero, then the range to a later set of observations is considered until reaching the slope coefficient constraint. At this point, the simulation seems to have reached the steady-state behavior.



t	y1	Slope_w=5	Slope_w=10
0	0	0.103450613	0.103450613
1	0.103451	0.077800192	0.077800192
2	0.1556	0.084030878	0.084030878
3	0.26272	0.083944734	0.083944734
4	0.340089	0.071444327	0.075976406
14	0.736594	0.033836086	0.032198087
15	0.793684	0.00861901	0.027467758
16	0.71951	0.008806412	0.023061324
17	0.795387	0.01495626	0.022255441
18	0.810524	0.02082943	0.02163266
19	0.852324	0.037148218	0.021868104
20	0.876782	0.031845881	0.021936743
21	0.921487	0.023170696	0.02099498
22	0.891796	0.015146668	0.022050114
23	0.920551	0.007094706	0.020109304
24	0.912724	0.000593557	0.019908747
25	0.913991	0.007191863	0.014234654
26	0.931035	0.00547341	0.011476546
27	0.938762	0.009016998	0.008353011
28	0.945423	0.011096871	0.00730059

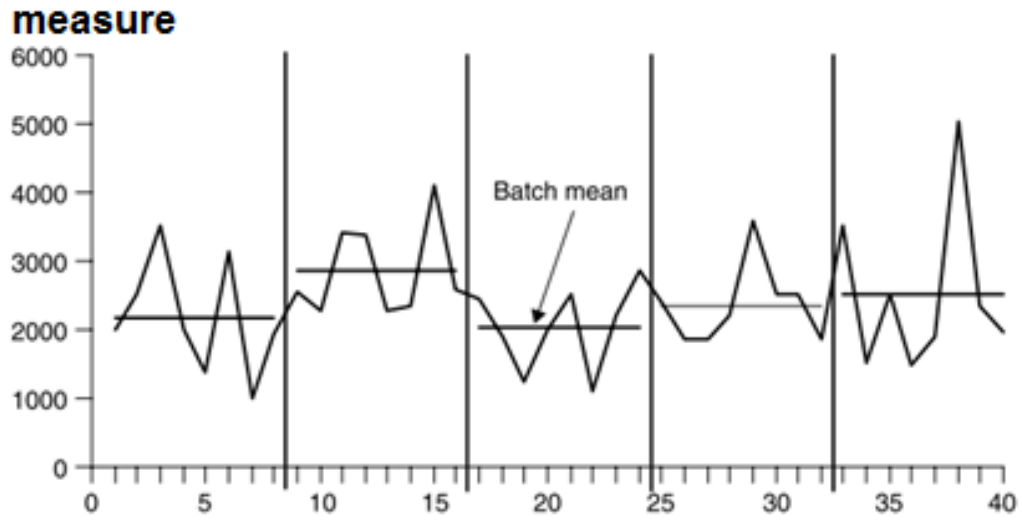
Output Analysis

Steady State or Non-terminating Simulation

Excel

Batch means method

Batch means method considers only a single long simulation run. We can also use a warmup interval, during which no data are collected. Then, the remaining observations are split into distinct batches (subsamples). These batches are then considered as individual replications.



Given a long run of $N + n_0$ observations, where n_0 is the number of observations that belong to the transient interval and are discarded, the remaining N observations are divided into $m = [N/n]$ batches of n observations each.

Output Analysis

Steady State or Non-terminating Simulation

Excel

Correlation

Batch means method

1. Compute the means for each batch:

$$\bar{X}_i = \frac{1}{n}\sum_{j=1}^{n} x_{ij}, \qquad i = 1,2,\dots,m$$

2. Calculate the mean of the batch means:

$$\bar{\bar{X}} = \frac{1}{m}\sum_{i=1}^{m} x_{ij},$$

3. Compute the standard deviation of the batch

$$Std(\bar{X}) = \sqrt{\frac{1}{m-1}\sum_{i}^{m}(\bar{X}_i - \bar{\bar{X}})^2}$$

4. The confidence interval for the mean is

$$\bar{\bar{X}} - t_{m-1,\alpha/2} \times \frac{Std(\bar{X})}{\sqrt{m}} \le \theta \le \bar{\bar{X}} + t_{m-1,\alpha/2} \times \frac{Std(\bar{X})}{\sqrt{m}}$$

5. The required number of batches is:

$$t_{m-1,\alpha/2} \times \frac{Std(\bar{X})}{\sqrt{m}} \le \epsilon$$

$$\left(t_{m-1,\alpha/2} \times \frac{Std(\bar{X})}{\sqrt{m}}\right)^2 \le \epsilon^2$$

$$m \ge \left(t_{m-1,\alpha/2} \times \frac{Std(\bar{X})}{\epsilon}\right)^2$$

ϵ – absolute precision error required.

$$\%\epsilon = \frac{\frac{\epsilon}{Std(\bar{X})}}{100} = \frac{100 \times \epsilon}{Std(\bar{X})}$$

$$\epsilon = \frac{Std(\bar{X}) \times \%\epsilon}{100}$$

$$m \ge \left(t_{m-1,\alpha/2} \times \frac{Std(\bar{X})}{\frac{Std(\bar{X}) \times \%\epsilon}{100}}\right)^2$$

$$m \ge \left(t_{m-1,\alpha/2} \times \frac{100}{\%\epsilon}\right)^2$$

$\%\epsilon$ – relative precision error required, that is the percentage deviation of the confidence interval about the mean of means.

Output Analysis

Steady State or Non-terminating Simulation

Excel

Correlation

Notice that the computation is essentially the same as it is in the method of independent replications.

However, the method of batch means incurs less waste, since only n_0 observations are discarded.

The confidence interval width is inversely proportional to $\sqrt{m \times n}$, and it can be reduced by increasing either the number of batches m or the batch size n. The batch size n must be large so that the batch means have little correlation.

One way to find a suitable n is to compute the correlation of successive batch means.

As $$Var[X+Y] = Var[X] + Var[Y] + 2Cov[X,Y],$$

where $2Cov[X,Y]$ is defined as:

$$Cov[X,Y] = E[(X - E[X]) \times (Y - E[Y])]$$

$$\Leftrightarrow$$

$$Cov[X,Y] = \frac{1}{m-2} \sum_{i=1}^{m-1} (X - \bar{X}) \times (Y - \bar{Y})$$

Output Analysis

Steady State or Non-terminating Simulation

Excel

Correlation

Then, If X and Y and independent, then $Cov[X, Y] = 0$. Therefore:

$$Var[X + Y] = Var[X] + Var[Y].$$

Independent random variables have correlation 0, but correlation 0 does not guarantee independence!

Correlation is the scale free covariance, that is:

$$Cor[X, Y] = \frac{Cov[X, Y]}{\sigma_X \sigma_Y}$$

Output Analysis

Steady State or Non-terminating Simulation

Excel

Correlation

As $X = \bar{X}_i$ and $Y = \bar{X}_{i+1}$, where $\bar{X}_i$ and $\bar{X}_{i+1}$ are obtained from the same set X, considering a batch size n, we have:

$$Cov[\bar{X}_i, \bar{X}_{i+1}] = \frac{1}{m-2} \sum_{i=1}^{m-1} \left(\bar{X}_i - \bar{\bar{X}}\right) \times \left(\bar{X}_{i+1} - \bar{\bar{X}}\right)$$

$$Cor[\bar{X}_i, \bar{X}_{i+1}] = \frac{Cov[\bar{X}_i, \bar{X}_{i+1}]}{\sigma_{\bar{X}_i} \sigma_{\bar{X}_{i+1}}}$$

These quantities are called the autocovariance and autocorrelation.

The prefix *auto* denotes that the fact that both random variables $\bar{X}_i$ and $\bar{X}_{i+i}$ are member (obtained) of the same set.

This process is repeated by increasing the batch size (n) until the autocorrelation is small.

Output Analysis

Steady State or Non-terminating Simulation

Excel

Mathematica Correlation

Minitab Correlation

Excel Correlation

1	0.97078
2	0.987433
3	0.9949
4	0.992916
5	0.99118
6	0.997277
7	0.995647
8	0.997065

345	1
346	1
347	1
348	1
349	1
350	1
351	1
352	1
353	1
354	1
355	1

Lag	Correlation	σx	σy	Cov(x,y)	Correlation
1	0.81395439	0.001946	0.00119362	1.89075E-06	0.81395439
2	0.675835448	0.001949	0.00099995	1.31698E-06	0.675835448
4	0.74829857	0.001954	0.000897061	1.31174E-06	0.74829857
8	0.629564925	0.001965	0.000897061	8.83294E-07	0.501110822
16	0.611706872	0.001987	0.000423098	5.14296E-07	0.611706872
32	0.785329537	0.002034	0.000215402	3.44057E-07	0.785329537
64	0.442689826	0.002138	7.84794E-06	7.42859E-09	0.442689826
71	0.476804843	0.002163	5.32864E-06	5.4962E-09	0.476804843
	Direct use of Excel formula				From Cov (x,y), Std(x) and Std(y)

Output Analysis

Steady State or Non-terminating Simulation

Excel

Mathematica Correlation

Excel Correlation

Minitab Correlation