

# Pramod Parajuli

# Simulation and Modeling, CS-331

---

## Course Curriculum

(C) Pramod Parajuli, 2004

Source: [www.csitnepal.com](http://www.csitnepal.com)

1  
Csitnepal

# **Simulation and Modeling**

## Introduction to Simulation

- System Concepts
- System Modeling
- Mathematical Models: their nature and assumptions
- Calibration and Validation

## Discrete and Continuous Systems

- Queuing System
- Markov Chains
- Differential and Partial Differential Equations

# **Simulation and Modeling**

## **Generation of Random Variables**

- Uniform Random Generators
- Testing of Uniform Random Generators
- Methods of Generating Non-uniform Variables
- Inversion, Rejection, composition

## **Analysis of Simulated Output**

- Estimation Methods
- Simulation Run Statistics
- Replication of Internal Bias

# Simulation and Modeling

## Simulation Languages

- Discrete Systems Modeling and Simulation with GPSS
- Basic Concepts
- Resources in GPSS, GPSS Programs, Applications
- Continuous Systems Modeling and Simulation with CSMP
- Structural Data and Control Statements, Hybrid Simulation
- Feedback System, Typical Applications

# Simulation and Modeling

## Text Books

- Gordon, Geoffrey – *System Simulation*, 2<sup>nd</sup> Edition - PHI (must)
- Law, Kelton – *Simulation Modeling and Analysis*, 3<sup>rd</sup> Edition - Tata McGraw-Hill (must)
- Banks, Carson, Nelson, Nicol – *Discrete Event System Simulation*, 3<sup>rd</sup> Edition – PHI
- Deo, Narsingh – *System Simulation with Digital Computer* – PHI
- Books, Manuals, Tutorials covering GPSS and CSMP.

# Simulation and Modeling

## Note

- 4-periods per week
- Practical – using GPSS and CSMP. If time allows, we will see Vensim and Arena also.
- Project - simulation model development and testing of some real world scenario
- Website:
  - <http://192.168.100.46/sim>

# Pramod Parajuli

---

## GPSS World - Background

## **Introduction**

- IBM – 1961
  - Block Oriented Simulation System
  - Originally, restricted to 1000 blocks
- 
- GPSS/World – Minuteman Software

## **Additional Features**

- Integration and differentiation
- Random numbers generation following desired probability distributions
- Sensitivity – to provide high sensitivity requires high precision real numbers

In queuing systems,

- Arrival – Poisson, Exponential, Hyper-exponential etc.
- Service – Normal
- Departure – None

## GPSS World

- Continuous and discrete modeling capabilities
- Underlying PLUS (Programming Language Under Simulation)
- Untyped data (implicit conversion)
- Mathematical and Probability distributions are available implicitly
- User defined PLUS functions (use CONDUCT)
- New multi-way ANOVA matrix library
- Support for file and database access, external functions from .exe or .dll files directly (Call(), Call\_Integer(), Call\_real() etc..)

# Objects

- Model objects – 50 blocks
- Simulation objects
- Report objects
- Text objects

Translator – Creates simulation objects from model objects

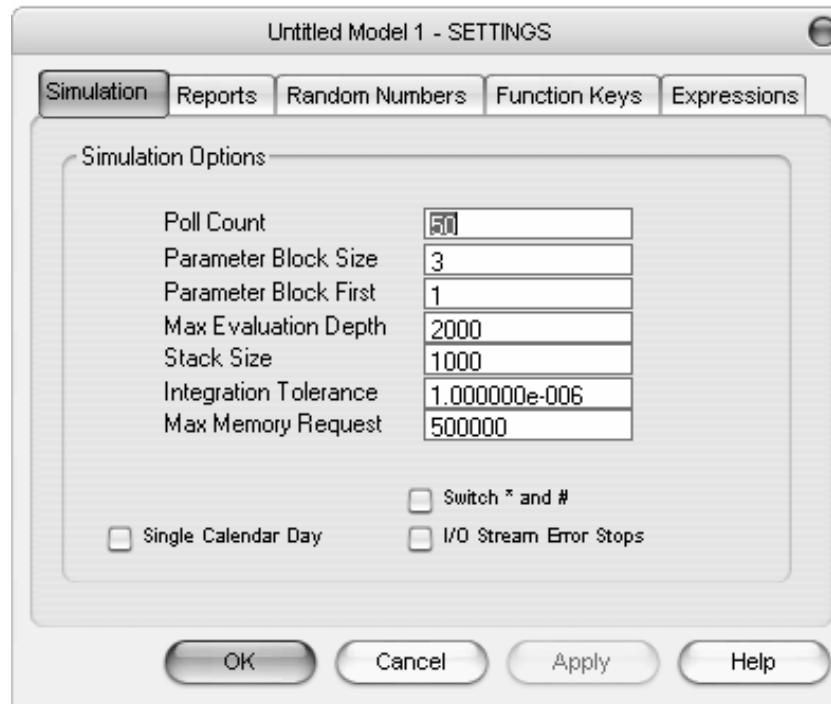
Errors (if any) are uncovered during the translation process

# Conventions

# for multiplication

\* for pointer

Can be changed by using model setting



- Length of function name is unrestricted

# Architecture

- Multi-tasking
- Use of virtual memory
- Interactivity
- Visualization
  - Snapshots
  - Dynamic Window

Loc	Block Type	Current Count	Entry Count	Retry Chain	Line Number	Include-file
1 GEN	GENERATE	0	102	0	8	0
2 TES	TEST	0	102	0	9	0
3 SAV	SAVEVALUE	0	55	0	11	0
4 ASN	ASSIGN	0	55	0	12	0
5 QUE	QUEUE	1	55	0	13	0
6 SEI	SEIZE	1	54	0	14	0
7 DEP	DEPART	0	53	0	15	0
8 ADV	ADVANCE	0	53	0	16	0
9 REL	RELEASE	0	53	0	17	0
FINIS	TERMINATE	0	100	0	18	0

(C) Pramod Parajuli, 2004

# Architecture

- Simulation clock
- Animation
  - Abstract
  - Post processing
  - Online animation
- 50 blocks and 25 commands
- All of state variables are global

## PLUS

- Polymorphic data types
- Multidimensional matrices
- Expressions

DoCommand("SHOW "("GPSS Presentation")");

Every command is parsed by DoCommand  
function

SHOW PolyCatenate("The ","time ","is ",AC1)

"The time is 0"

SHOW Length("ABC") 3

SHOW Find("ABC","123ABC789") 4

## PLUS

```
Accumulator=1;  
Counter=1;  
WHILE (Counter<=X_Integer) DO BEGIN  
    Accumulator=Accumulator#Counter;  
    Counter=Counter+1;  
END;
```

```
IF ( Expression ) THEN Statement1 ELSE  
    Statement2  
GOTO Label ;
```

# Pramod Parajuli

# Simulation and Modeling, CS-331

---

## Chapter 1

System concepts

System modeling

Mathematical models: their nature and  
assumptions

Calibration and validation

(C) Pramod Parajuli, 2004

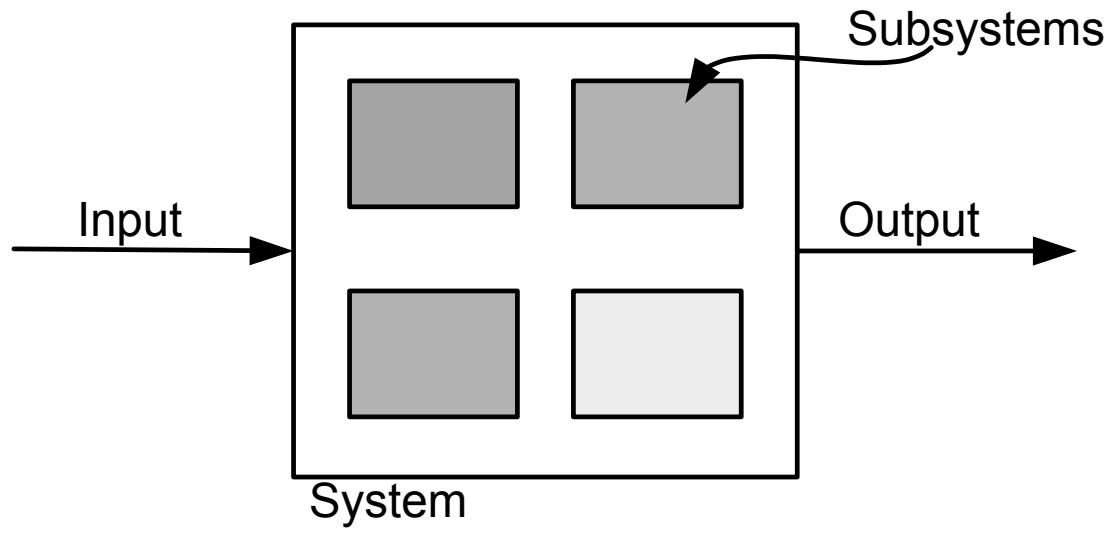
Source: [www.csitnepal.com](http://www.csitnepal.com)

CSitnepal  
1

# System Concepts

## System

- Assembly of objects joined in some regular interaction or interdependence.
- A system exists and operates in time and space.
- bounded inside system boundary



Environment

(C) Pramod Parajuli, 2004

# **System Concepts - terms**

## **State**

A variable characterizing an attribute in the system such as no. of jobs waiting for processing.

## **Activity/event**

An occurrence at a point in time which may change the state of the system

- Endogenous – activities occurring within the system
- Exogenous – activities occurring in the environment that affect the system

## **Entity**

An object that passes through the system

e.g. cars

Entity is associated with an event

# **System Concepts - terms**

## **Queue**

- physical queue of entity/task
- any place where entities are waiting for something to happen for any reason

## **Creating**

Causing an arrival of a new entity to the system at some point in time

## **Scheduling**

The act of assigning a new future event to an existing entity

## **Random variable**

A random variable is a quantity that is uncertain, such as inter-arrival time between two incoming flights or number of defective parts in a shipment

(C) Pramod Parajuli, 2004

## **System Concepts - terms**

### **Random variate**

A random variate is an artificially generated random variable.

### **Distribution**

A distribution is the mathematical law which governs the probabilistic features of a random variable.

# System Concepts

## System environment

The changes occurring outside the system that affect the system is system environment

Defining a ***system boundary*** is an important step

- if small system boundary, may not include necessary components
- if larger boundary, high degree of error propagation, management difficulties etc.

Closed system – that is not affected by the exogenous events. Practically, unrealizable

Open system – affected by the exogenous events

## System - A Simple Example

Building a simulation of a gas station with a single pump served by a single service man. Assume that arrival of cars as well their service times are random.

At first identify the

states: number of cars waiting for service and number of cars served at any moment

events: arrival of cars, start of service, end of service

entities: these are the cars

queue: the queue of cars in front of the pump, waiting for service

random realizations: inter-arrival times, service times

distributions: we shall assume exponential distributions for both the inter-arrival time and service time.

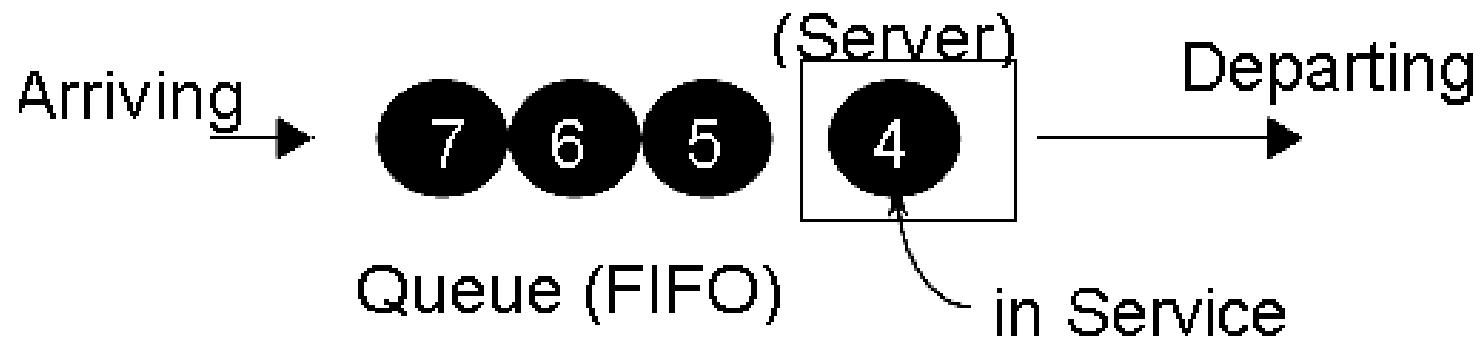
## System - A Simple Example

Let's represent the system as;

1. Upon the entity arrival event, create next arrival
2. If the server is free, send entity for start of service.
3. Otherwise put it on the queue.
4. At event of service start: Server becomes occupied.  
Schedule end of service for this entity.
5. At event of service end: Server becomes free. If any entities waiting in queue: remove first entity from the queue; send it for start of service.

Here, we must consider some initial conditions for example, the creation of the first arrival.

## System - A Simple Example



## Stochastic activities

If the output of an activity is determined by the inputs, it is said to be ***deterministic***.

But if the output of an activity varies then it is known as ***stochastic activity***. Such random occurrence of activity itself constitutes part of the environment.

Truly stochastic – no known explanation for the activities randomness

# Stochastic activities - classification

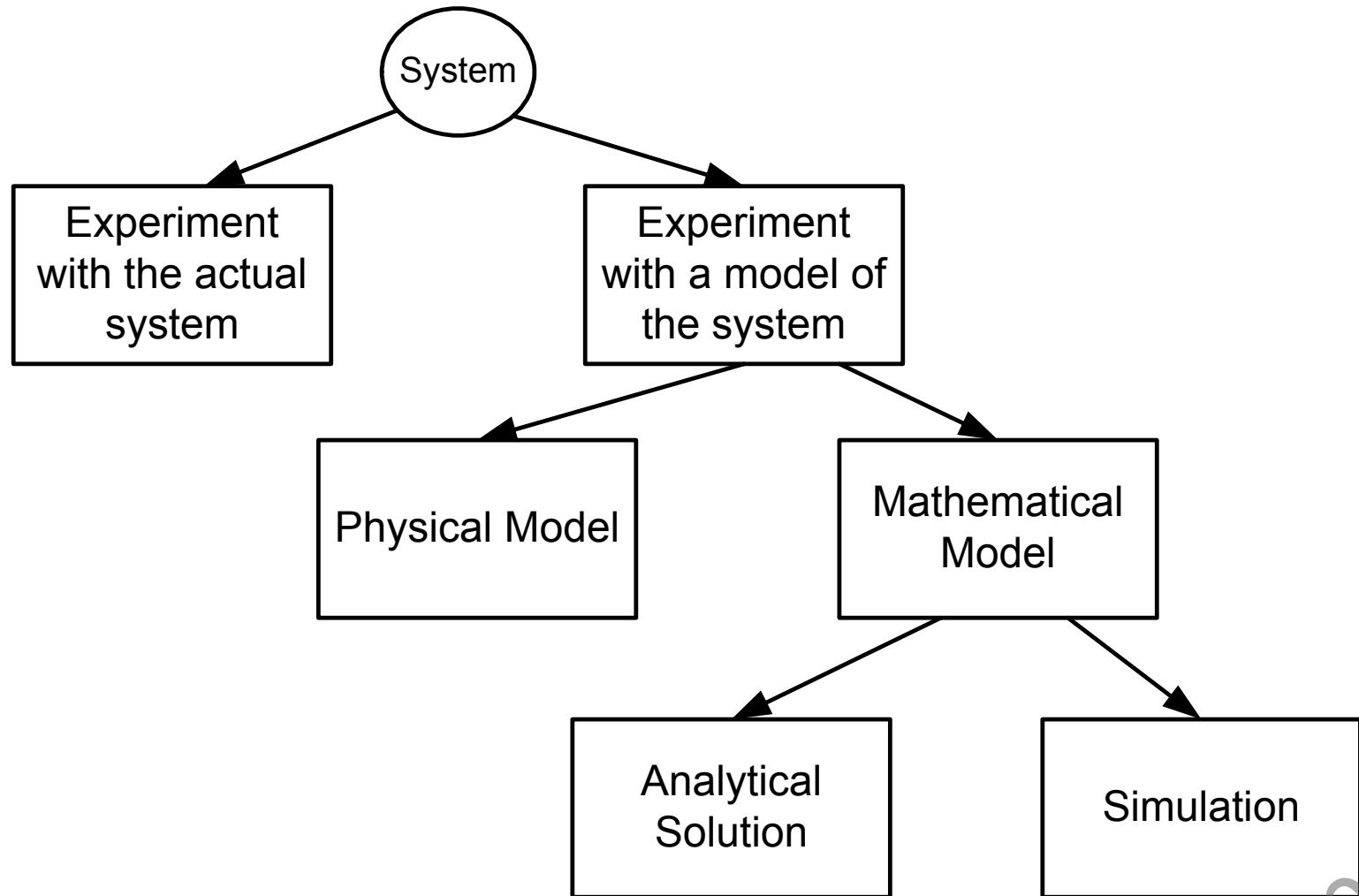
		Change in the Sate of the System	
		Continuous	Discrete
Time	Continuous	Level of water behind dam	Number of customers in the bank
	Discrete	Weekdays' range of temperature	Sales at the end of the day

## **Types of Systems**

Discrete – system for which the state variables change instantaneously at separated points in time

Continuous – system for which the state variables change continuously with respect to time

# System Study



## Types of Systems

Experiment with actual system vs. Experiment a model of the system

If possible and cost effective, only then do the actual system experiment. Most of the systems are rarely feasible. E.g. test of bridge

Physical model vs. Mathematical model

Physical models – iconic model (like small samples). They do not represent the internal properties.

Whereas the mathematical model represents a system in terms of logical and quantitative relationships that are manipulated and changed to see how the model reacts.

## Types of Systems

If model is simple enough, use traditional mathematical analysis ... get exact results, lots of insight into model

Mathematical Analysis includes;

- Queuing theory
- Differential equations
- Linear programming

## Types of Systems

### Analytical solution vs. simulation

How the mathematical model can be used to answer question of interest about the system. If the model is simple enough, can be studied analytically.

e.g.  $d = rt$ . If we know  $d$  and  $r$ , then we can find  $t$ .

But, situations might come in which the mathematical model is highly complex so that analytical study is not sufficient. In such case, simulation is used.

# Simulation

Broadly interpreted, computer simulation refers to methods for studying a wide variety of models of systems

- Numerically evaluate on a computer
- Use software to imitate the system's operations and characteristics, often over time

Can be used to study simple models but should not use it if an analytical solution is available

Real power of simulation is in studying complex models

Simulation can tolerate complex models

## Simulation - Advantages

- Flexibility to model things as they are (even if messy and complicated)
- Allows uncertainty, non-stationarity in modeling
  - Danger of ignoring system variability
  - Model validity
  - Advances in computing/cost ratios
- Estimated that 75% of computing power is used for various kinds of simulations
- Dedicated machines
- Far easier to use (GUIs)
- Statistical design & analysis capabilities

## Simulation - Problems

- Don't get exact answers, only approximations, estimates
  - Also true of many other modern methods
  - Can bound errors by machine round-off
- Get random output (RIRO) from stochastic simulations
  - Statistical design, analysis of simulation experiments
  - Exploit: noise control, replicability, sequential sampling, variance-reduction techniques
  - Catch: **"standard" statistical methods seldom work**

## Kinds of Simulation

### Static vs. Dynamic

Does time have a role in the model?

### Continuous-change vs. Discrete-change

Can the “state” change continuously or only at discrete points in time?

### Deterministic vs. Stochastic

Is everything for sure or is there uncertainty?

### Most operational models:

Dynamic, Discrete-change, Stochastic

# Using Computers to Simulate

## Simulation languages

- GPSS, SIMSCRIPT, SLAM, SIMAN
- Popular, still in use
- Learning curve for features, effective use, syntax

## High-level simulators

- Very easy, graphical interface
- Domain-restricted (manufacturing, communications)
- Limited flexibility — model validity?

# **System Analysis**

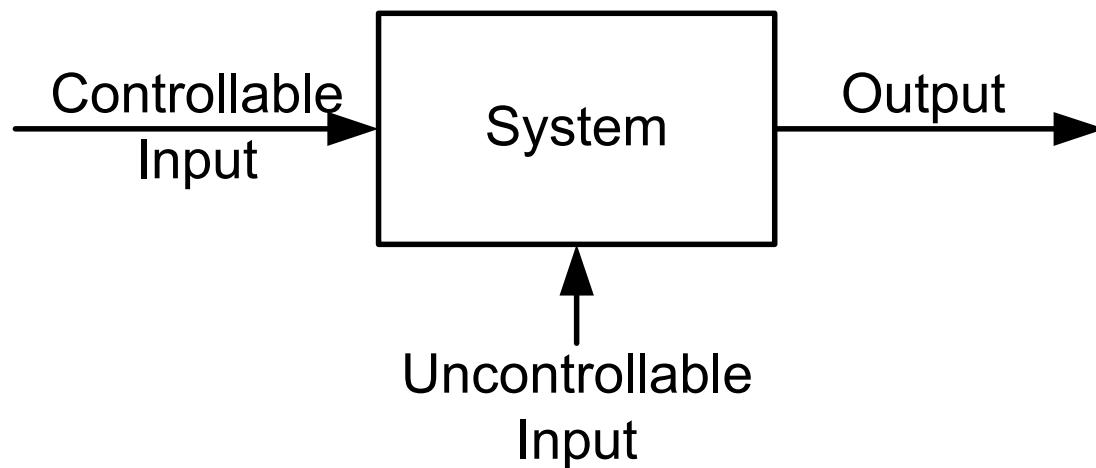
- Descriptive Analysis
- Prescriptive Analysis
- Post-prescriptive Analysis

# System Analysis

Descriptive Analysis includes:

## 1. Problem Identification & Formulation

Identify controllable and uncontrollable inputs.



Identify constraints on the decision variables. Define measure of system performance and an objective function. Develop a preliminary model structure to interrelate the inputs and the measure of performance.

# **System Analysis**

## **2. Data Collection and Analysis**

Regardless of the method used to collect the data, the decision of how much to collect is a trade-off between cost and accuracy.

## **3. Computer Simulation Model Development**

Acquiring sufficient understanding of the system to develop an appropriate conceptual, logical and then simulation model is one of the most difficult tasks in simulation analysis.

# System Analysis

## 4. Model Validation, Verification, and Calibration

Verification focuses on the internal consistency of a model

Validation is concerned with the correspondence between the model and the reality.

The term validation is applied to those processes which seek to determine whether or not a simulation is correct with respect to the "real" system. More prosaically, validation is concerned with the question "Are we building the right system?".

Verification, on the other hand, seeks to answer the question "Are we building the system right?" Verification checks that the implementation of the simulation model (program) corresponds to the model. Validation checks that the model corresponds to reality.

# **System Analysis**

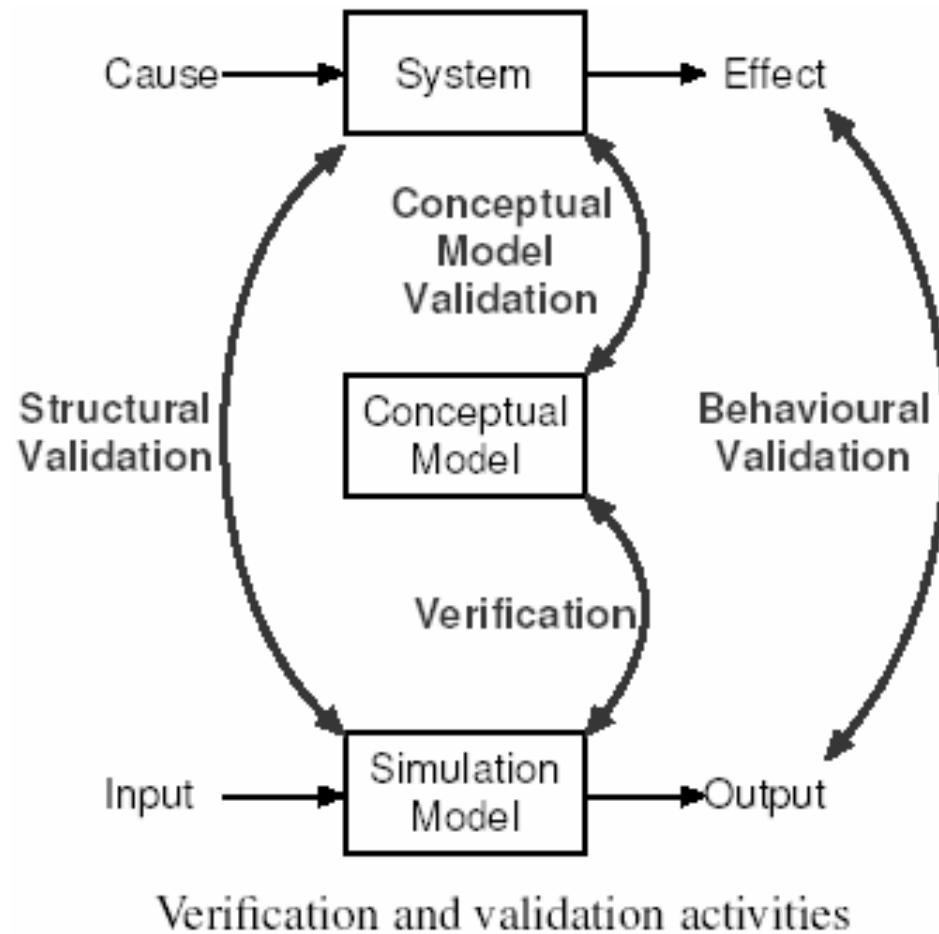
## **Validation**

The process of comparing the model's output with the behavior of the phenomenon. In other words: comparing model execution to reality (physical or otherwise)

## **Verification**

The process of comparing the computer code with the model to ensure that the code is a correct implementation of the model.

# System Analysis



# System Analysis

## Validation & Verification – 3 approaches

- The development team make the decision as to whether the model is valid. This is a subjective decision based on the results of the various tests and evaluations conducted as part of the model development process.
- “independent verification and validation” (IV&V), uses a third (independent) party to decide whether the model is valid (extremely costly)
- determining whether a model is valid is to use a scoring model. Scores (or weights) are determined subjectively when conducting various aspects of the validation process and then combined to determine category scores and an overall score for the simulation model. (infrequently used)

# **Validation techniques**

## **Animation**

The model's operational behavior is displayed graphically as the model moves through time.

## **Comparison to Other Models**

Various results of the simulation model being validated are compared to results of other (valid) models.

## **Degenerate Tests**

The degeneracy of the model's behavior is tested by appropriate selection of values of the input and internal parameters.

## **Event Validity**

The "events" of occurrences of the simulation model are compared to those of the real system to determine if they are similar.

(C) Pramod Parajuli, 2004

29

# Validation techniques

## Extreme Condition Tests

The model structure and output should be plausible for any extreme and unlikely combination of levels of factors in the system.

## Face Validity

“Face validity” is asking people knowledgeable about the system whether the model and/or its behavior are reasonable.

## Fixed Values

Fixed values (e.g., constants) are used for various model input and internal variables and parameters. This should allow the checking of model results against easily calculated values.

# **Validation techniques**

## **Historical Data Validation**

If historical data exist (or if data are collected on a system for building or testing the model), part of the data is used to build the model and the remaining data are used to determine (test) whether the model behaves as the system does.

## **Internal Validity**

Several replications (runs) of a stochastic model are made to determine the amount of (internal) stochastic variability in the model.

## **Turing Tests**

People who are knowledgeable about the operations of a system are asked if they can discriminate between system and model outputs

# **Validation techniques**

## Multistage Validation

consists of

- (1) developing the model's assumptions on theory, observations, general knowledge, and function,
- (2) validating the model's assumptions where possible by empirically testing them,
- (3) and comparing (testing) the input-output relationships of the model to the real system.

and many more...

## **Validation and verification - 4 musts**

### Data Validity

Data are needed for three purposes: for building the conceptual model, for validating the model, and for performing experiments with the validated model. In model validation we are concerned only with the first two types of data.

### Conceptual model validity

Conceptual model validity is determining that

- (1) the theories and assumptions underlying the conceptual model are correct, and
- (2) the model representation of the problem entity and the model's structure, logic, and mathematical and causal relationships are "reasonable" for the intended purpose of the model

## **Validation and verification - 4 musts**

### **Model verification**

Computerized model verification ensures that the computer programming and implementation of the conceptual model are correct.

### **Operational validity**

Operational validity is concerned with determining that the model's output behavior has the accuracy required for the model's intended purpose over the domain of its intended applicability. This is where most of the validation testing and evaluation takes place.

### **Documentation - required and very important.**

# System Analysis

## Calibration

It is an iterative process of comparing the model to the real system, making adjustments or major changes to the model, comparing the revised model to reality, making additional adjustments, comparing again and so on.

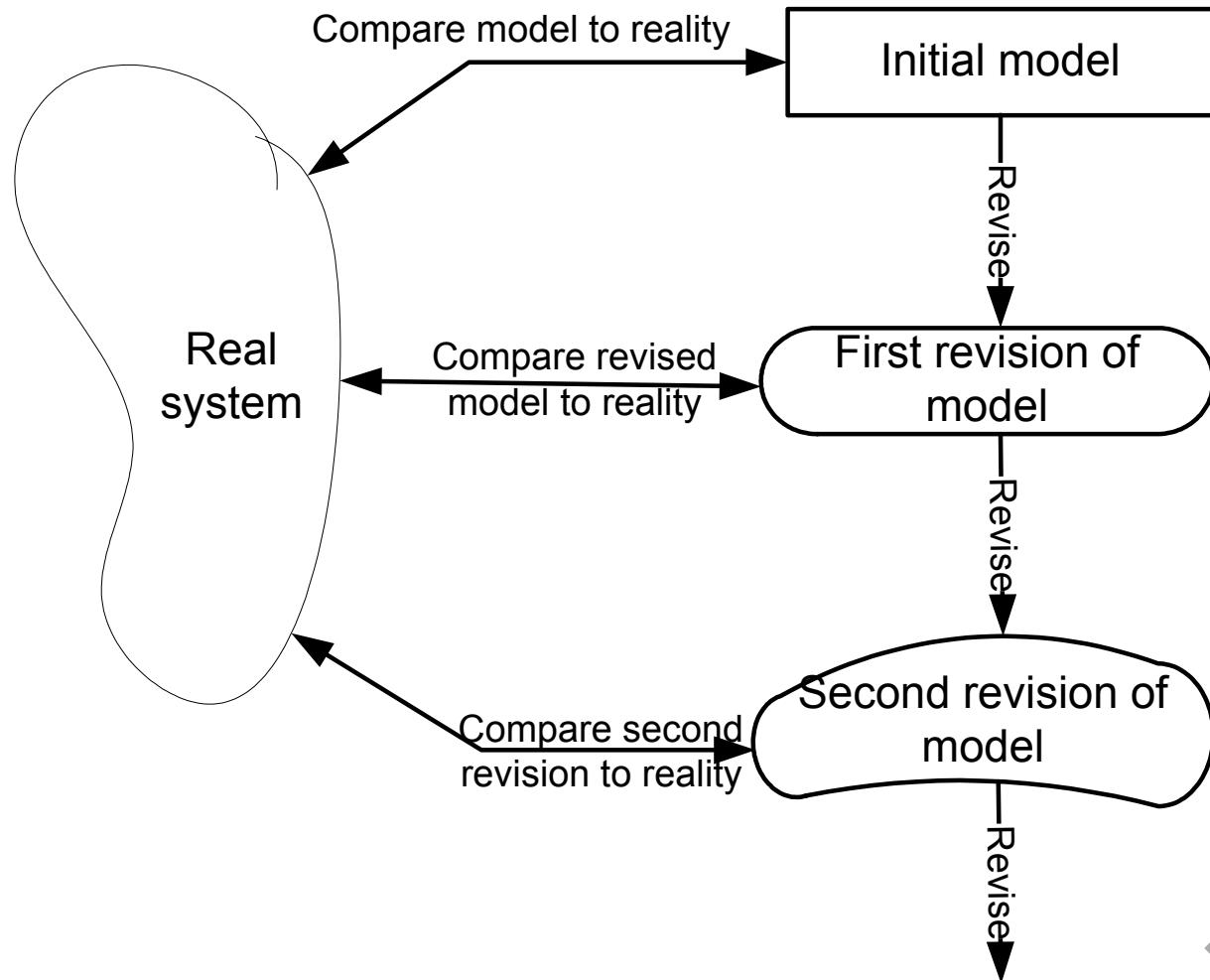
Checks that the data generated by the simulation matches real (observed) data.

The process of parameter estimation for a model.

Calibration is a tweaking/tuning of existing parameters and usually does not involve the introduction of new ones, changing the model structure. In the context of optimization, calibration is an optimization procedure involved in system identification or during experimental design.

# System Analysis

## Calibration – verification and validation relationship



(C) Pramod Parajuli, 2004

# System Analysis

## 5. Input and Output Analysis

Discrete-event simulation models typically have stochastic components that mimic the probabilistic nature of the system under consideration.

Successful input modeling requires a close match between the input model and the true underlying probabilistic mechanism associated with the system.

The input data analysis is to model an element (e.g., arrival process, service times) in a discrete-event simulation given a data set collected on the element of interest.

This stage performs intensive error checking on the input data, including external, policy, random and deterministic variables. System simulation experiment is to learn about its behavior.

# **System Analysis**

## 6. Performance Evaluation.

# **System Analysis**

## Prescriptive Analysis

### Optimization or Goal Seeking

Traditional optimization techniques require gradient estimation. As with sensitivity analysis, the current approach for optimization requires intensive simulation to construct an approximate surface response function.

# **System Analysis**

## Post-prescriptive Analysis

1. Sensitivity: Users must be provided with affordable techniques for sensitivity analysis if they are to understand which relationships are meaningful in complicated models.
2. What-If Analysis: The 'what-if' analysis is at the very heart of simulation models

## Report Generating

Report generation is a critical link in the communication process between the model and the end user.

## **System Modeling**

Model – body of information about a system gathered for the purpose of studying the system. A model is a representation of the construction and working of some system of interest. A model is similar to but simpler than the system it represents. One purpose of a model is to enable the analyst to predict the effect of changes to the system.

A model is a simplified representation of a system at some particular point in time or space intended to promote understanding of the real system.

Abstraction also applies

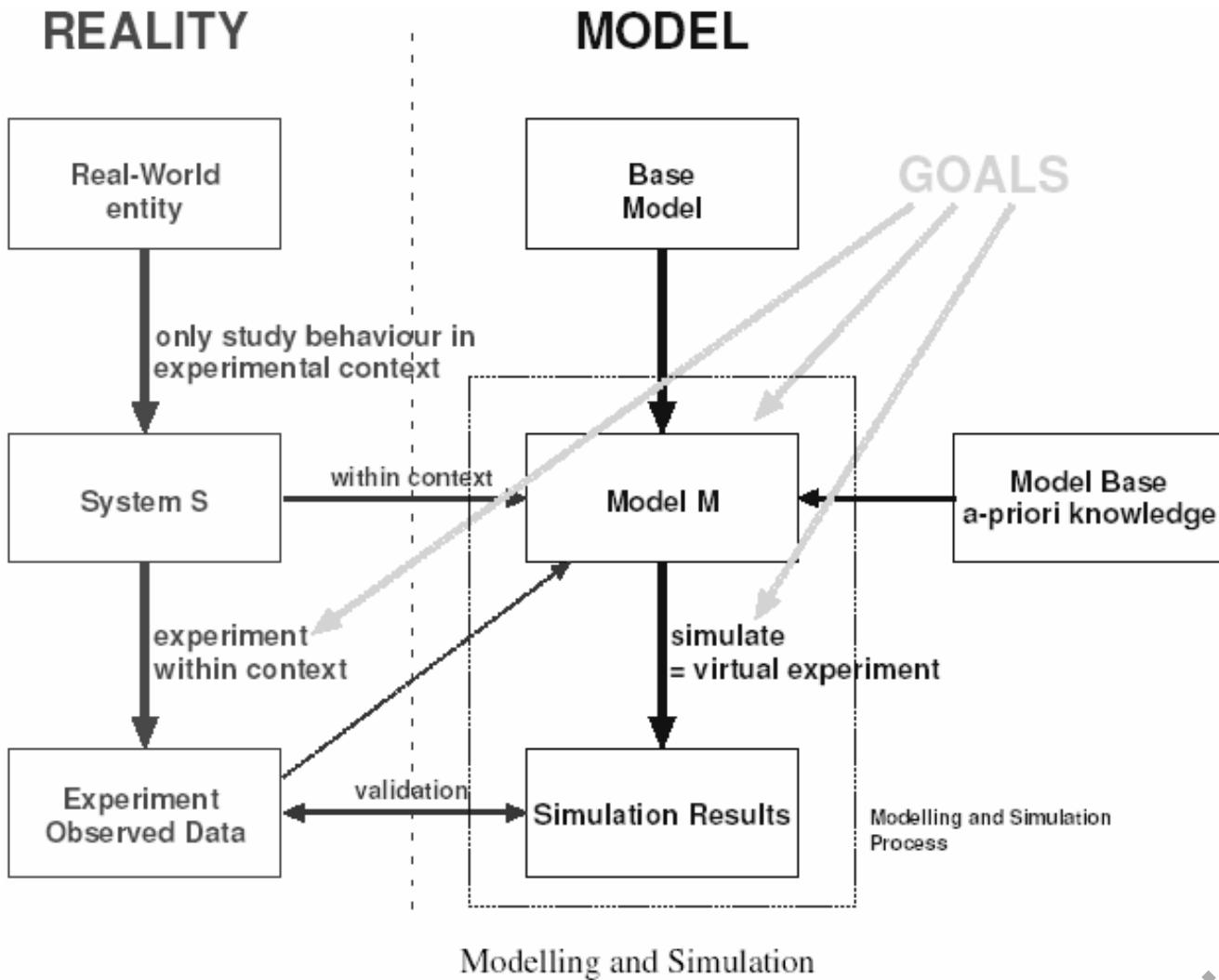
## **System Modeling**

The model should be sufficiently detailed to permit valid conclusions to be drawn about the real system.

Modeling is the process of producing a model; On the one hand, a model should be a close approximation to the real system and incorporate most of its salient/relevant features. On the other hand, it should not be so complex that it is impossible to understand and experiment with it.

A good model is a judicious tradeoff between realism and simplicity.

# System Modeling



## System Modeling

An important issue in modeling is model validity. Model validation techniques include simulating the model under known input conditions and comparing model output with system output.

Generally, a model intended for a simulation study is a mathematical model developed with the help of simulation software. Mathematical model classifications include **deterministic** (input and output variables are fixed values) or **stochastic** (at least one of the input or output variables is probabilistic); **static** (time is not taken into account) or **dynamic** (time-varying interactions among variables are taken into account). Typically, simulation models are stochastic and dynamic.

## More on simulation

A simulation of a system is the operation of a model of the system. The model can be reconfigured and experimented with; usually, this is impossible, too expensive or impractical to do in the system it represents. The operation of the model can be studied, and hence, properties concerning the behavior of the actual system or its subsystem can be inferred. In its broadest sense, simulation is a tool to evaluate the performance of a system, existing or proposed, under different configurations of interest and over long periods of real time.

A simulation is the manipulation of a model in such a way that it operates on time or space to compress it, thus enabling one to perceive the interactions that would not otherwise be apparent because of their separation in time or space.

(C) Pramod Parajuli, 2004

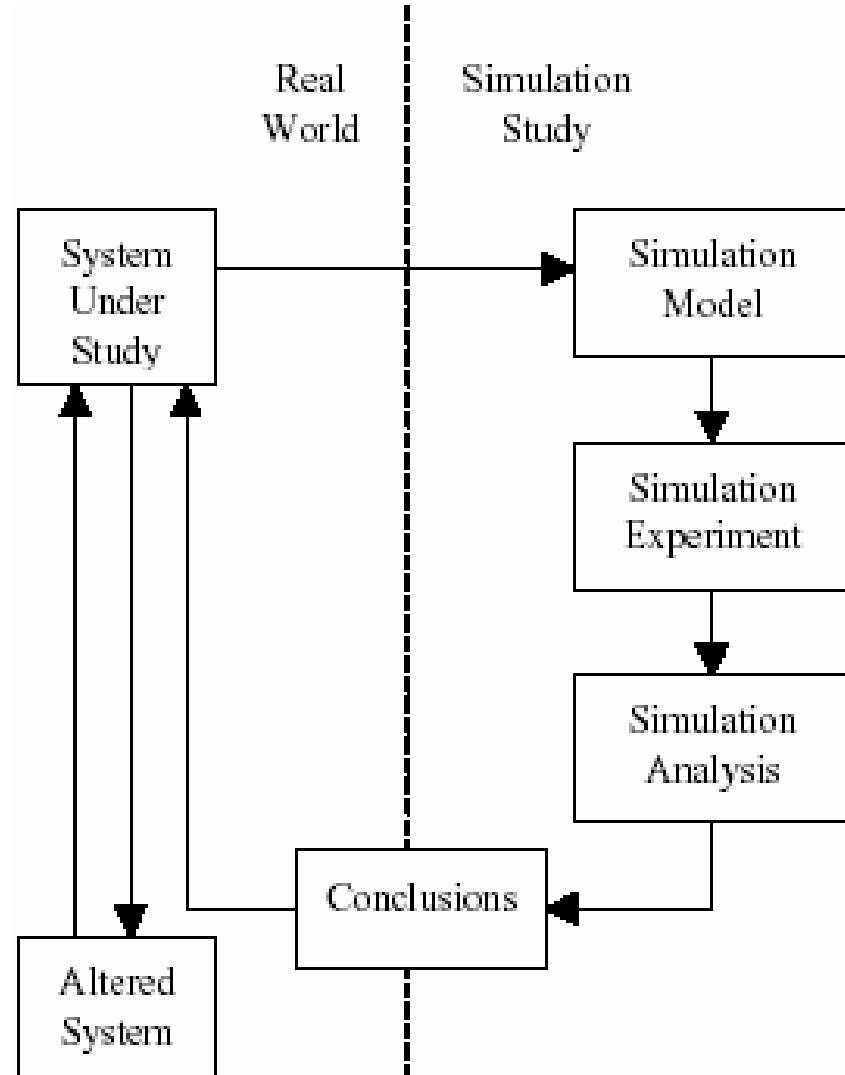
45

## More on simulation

System Simulation is the mimicking of the operation of a real system, such as the day-to-day operation of a bank, or the value of a stock portfolio over a time period, or the running of an assembly line in a factory, or the staff assignment of a hospital or a security company, in a computer.

Simulation is used before an existing system is altered or a new system built, to reduce the chances of failure to meet specifications, to eliminate unforeseen bottlenecks, to prevent under or over-utilization of resources, and to optimize system performance.

# Simulation - study



(C) Pramod Parajuli, 2004

## **Simulation Model**

The steps involved in developing a simulation model, designing a simulation experiment, and performing simulation analysis are:

How to develop a simulation model

Step 1. Identify the problem.

Step 2. Formulate the problem.

Step 3. Collect and process real system data.

Step 4. Formulate and develop a model.

Step 5. Validate the model.

Step 6. Document model for future use.

# Pramod Parajuli

# Simulation and Modeling, CS-331

---

## Chapter 2

## System Studies

(C) Pramod Parajuli, 2004

Source: [www.csitnepal.com](http://www.csitnepal.com)

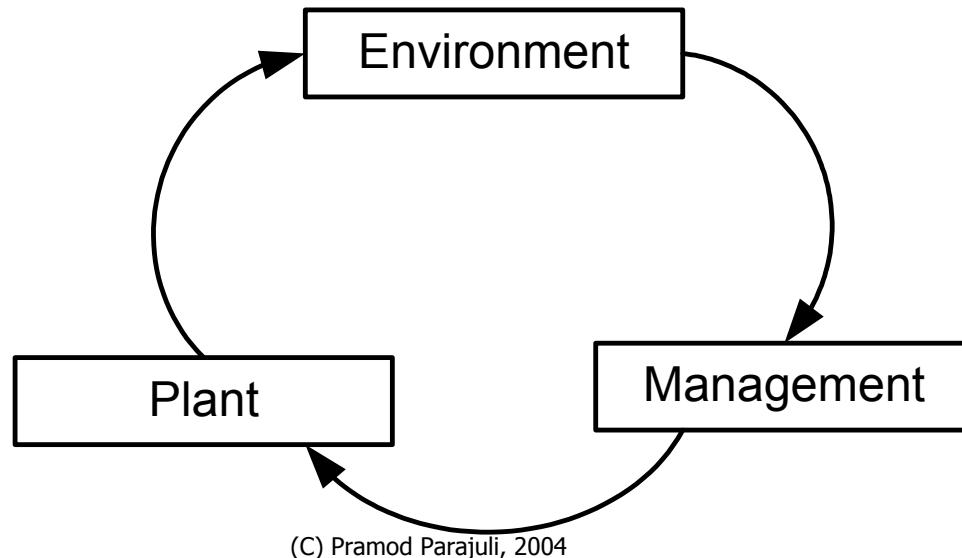
1  
Csitnepal

## **Sub Systems**

- System consists of interacting ***subsystems***
- Each subsystem has its own inputs and outputs
- Similarly, a model can also be broken into ***sub-models***
- Sometimes, the terms subsystem, sub-model, and blocks are interchangeable

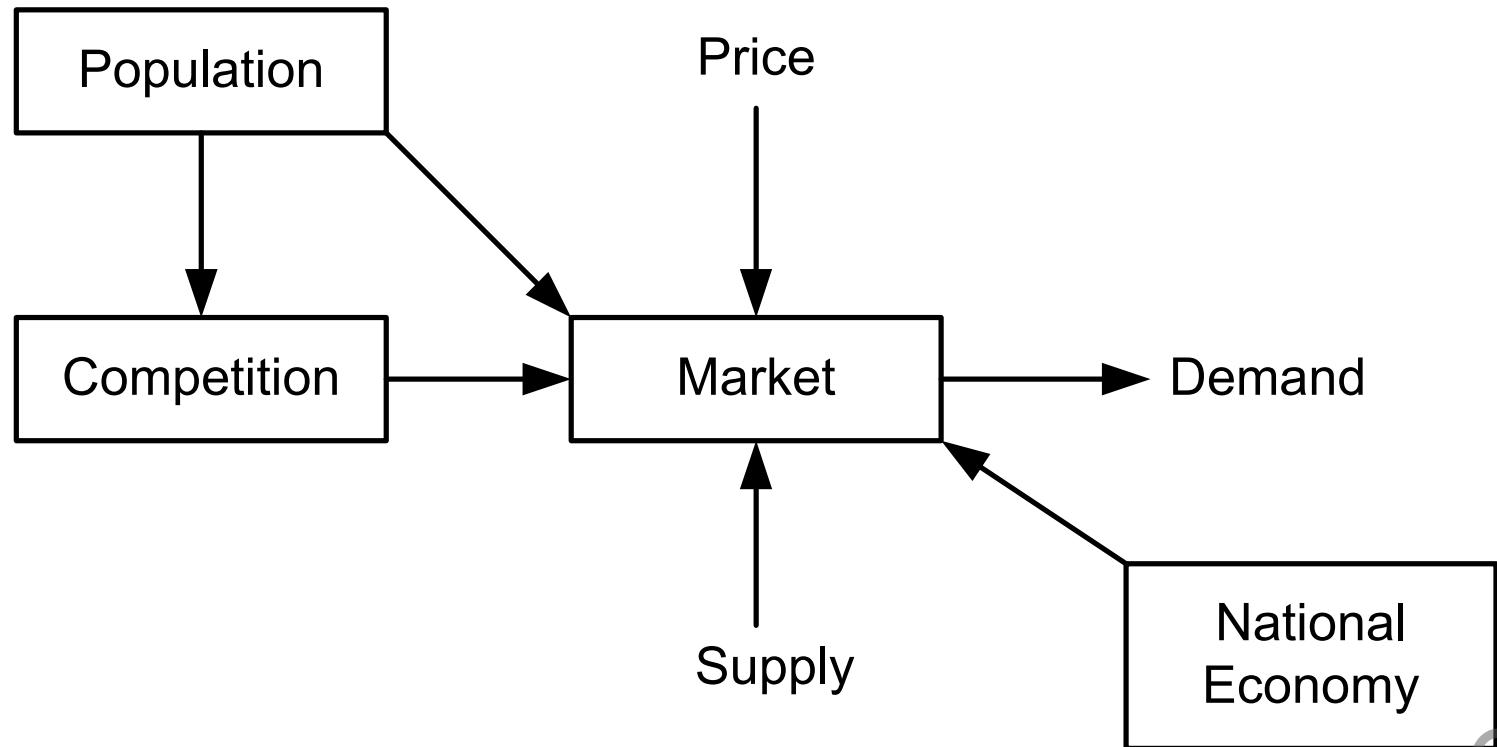
## A Corporate Model

- Model that exhibits the property of a corporation
- 3 essentials
  - Environment: the corporation exists within the environment
  - Management: responsible for all of the planning, risk analysis, scheduling, marketing etc.
  - Plant: the real implementation of the plan and design

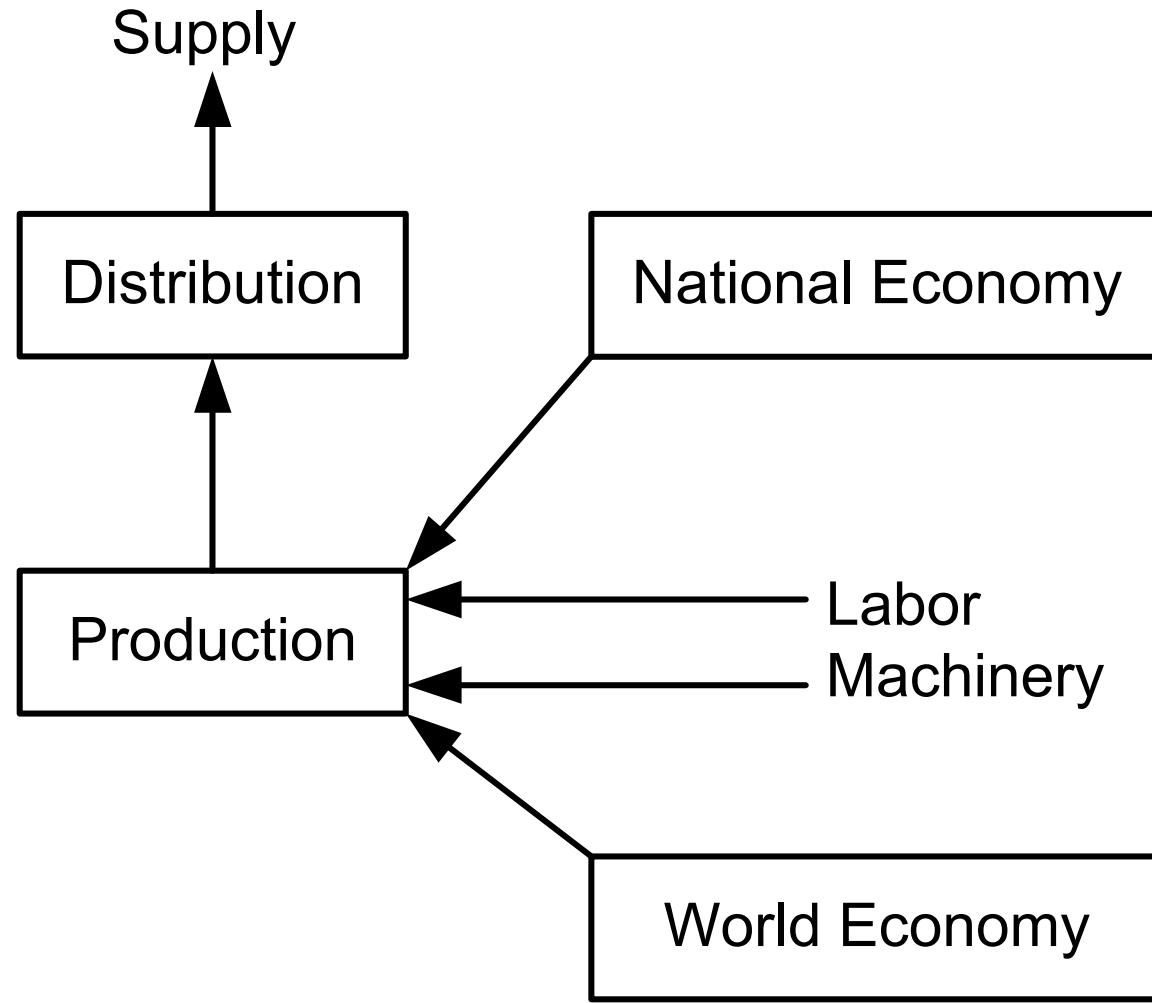


# Environment

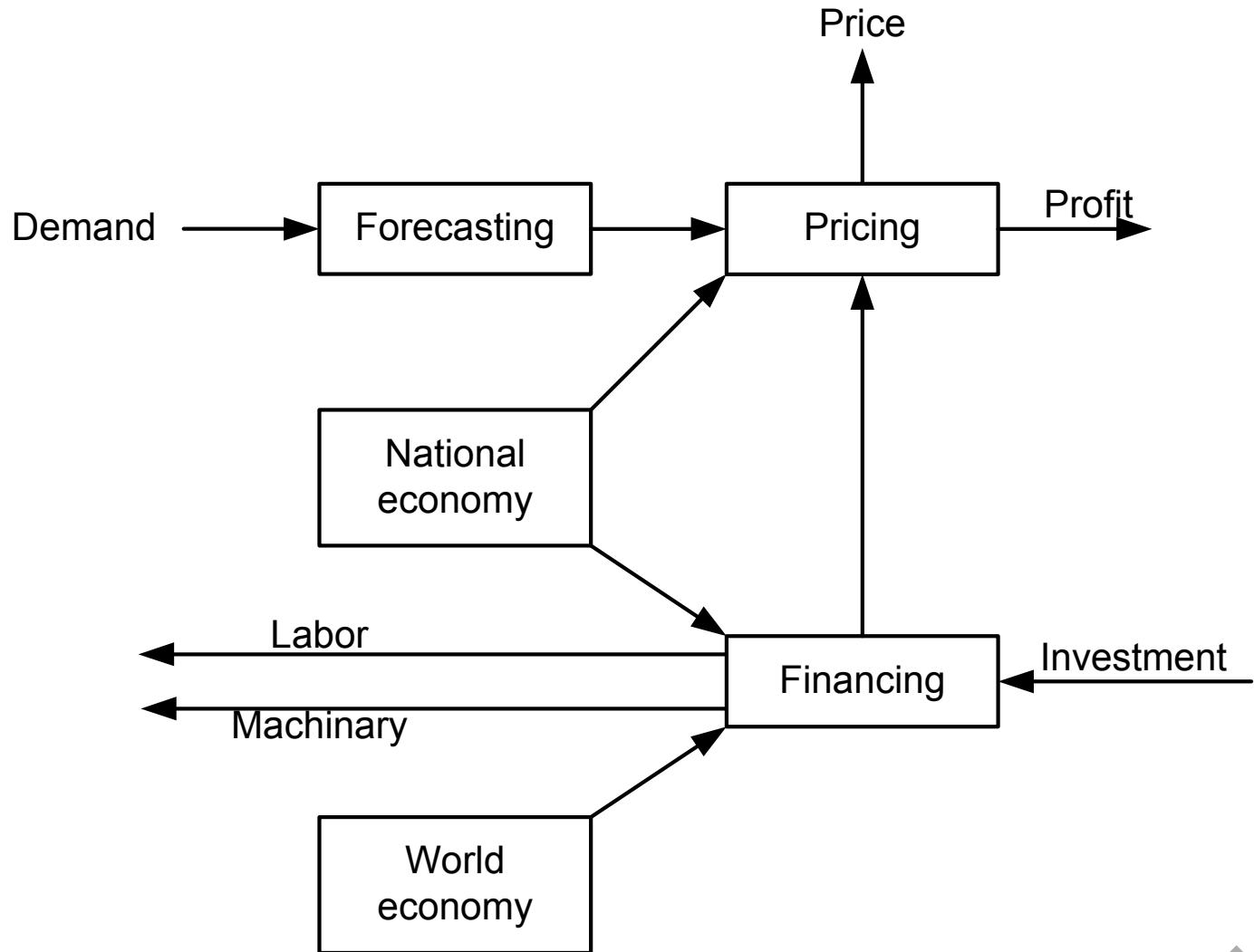
## market model



# Production



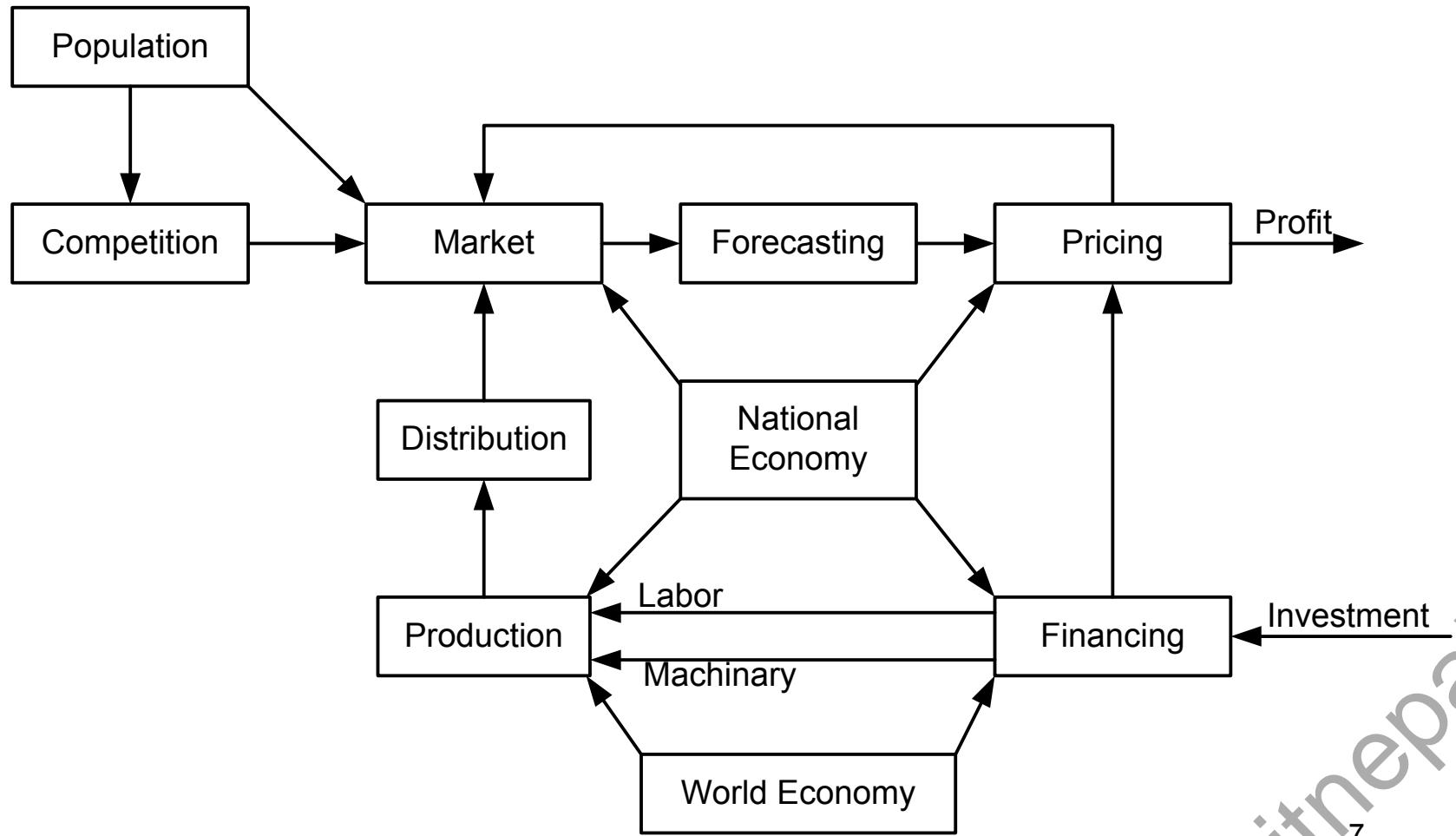
# Management



(C) Pramod Parajuli, 2004

# Full Corporate Model

## Combine



(C) Pramod Parajuli, 2004

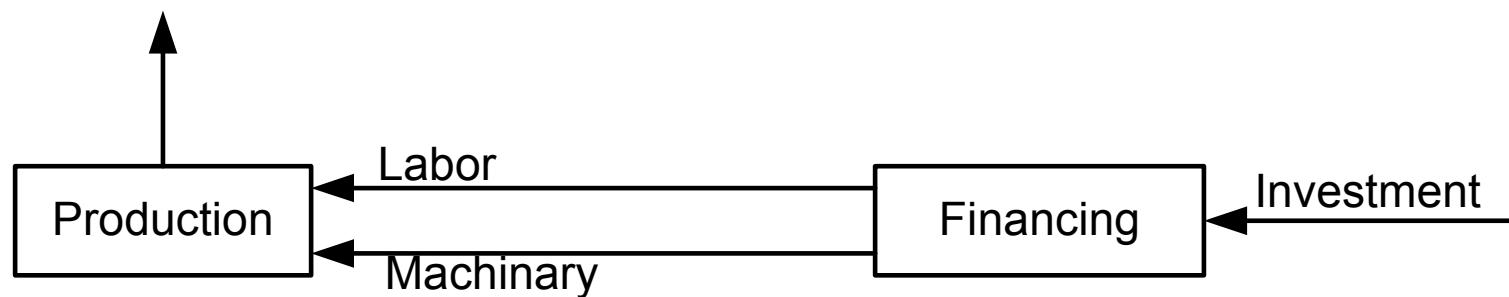
# **Types of System Studies**

- System analysis
  - understand the system under study
  - focus on system performance
- System design
  - develop the logical objects that meet some specification
  - calculate the performance and compare with the prediction
  - if not satisfied, then redesign and repeat again
- System postulation
  - Characteristic of the way models are employed in other disciplines

# System Analysis

## Considerations;

K	Capital investment
L	Labor
M	Machinery
S	Supply



$$S = f \cdot L^{a_1} \cdot M^{a_2}$$

*Cobb-Douglas model.* For simplicity consider;  $a_1=a_2=1$

## System Analysis

As far as financial model is considered, it is modeled as;

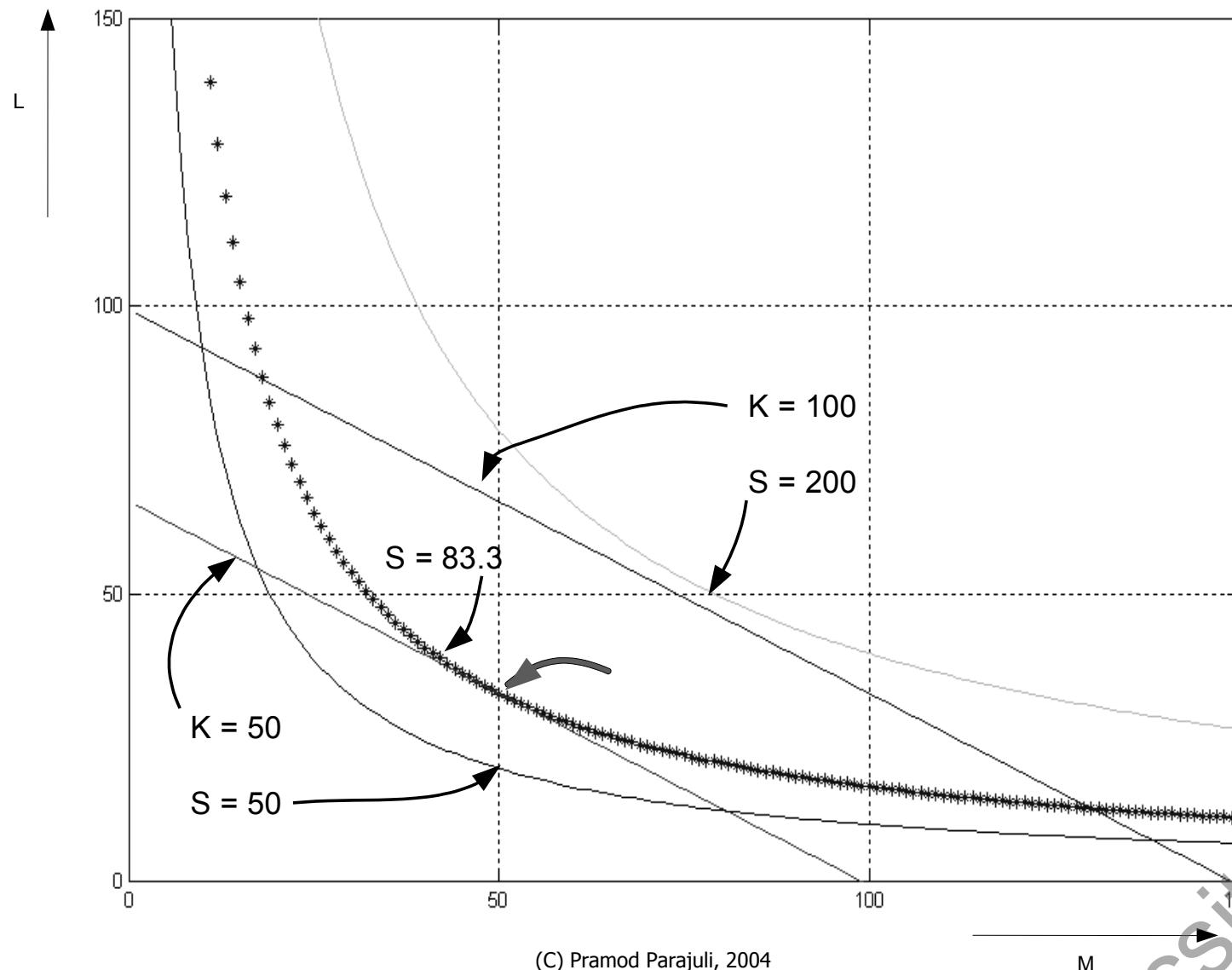
$$K = eL + M$$

Since the interdependency of K, L, M, and S has been found, can we find an assignment of L and M that maximizes supply for a given investment?

Let's see the graph plot of the relationships.

Let's consider that  $e = 0.75$  and  $f = 0.1$

# System Analysis – The optimal solution



## System Analysis – The optimal solution

The optimal point is;

$$M = 50$$

$$L = 33.3$$

Depending on the samples taken, the optimal point changes.

Here, only a part of the corporate model is optimized and hence known as ***sub-optimization***.

In general, when the system is giving the maximum performance, most of the sub-systems are at their peak performance.

## **System Analysis – The optimal solution**

The example we saw is a static one.

If the market competition, economy influence etc are to be considered, then the system will be dynamic and required even more sophisticated study

# System Design

## Online computer system

- '**M**' number of messages are received in a second
- '**m**' number of characters in one message
- A buffer that can hold '**b**' number of characters at a time
- Fraction '**k**' of the messages are replied
- The replied messages have '**r**' number of characters on the average
- Same buffer used for sending and receiving
- **2,000** instructions are required to process a message
- **10,000** instructions require to generate the reply message
- To read and reply the messages requires **1,000** instructions each

# System Design

## Online computer system

- 3 computer systems,
  - 25,000 instruction per second
  - 50,000 instruction per second
  - 100,000 instructions per second
- Buffer size can be
  - 1 character buffer
  - 2 character buffer
  - 5 character buffer
  - 10 character buffer
- Altogether, 12 different configurations
- Which computer with which buffer size will be the best for the implementation given the price of the systems

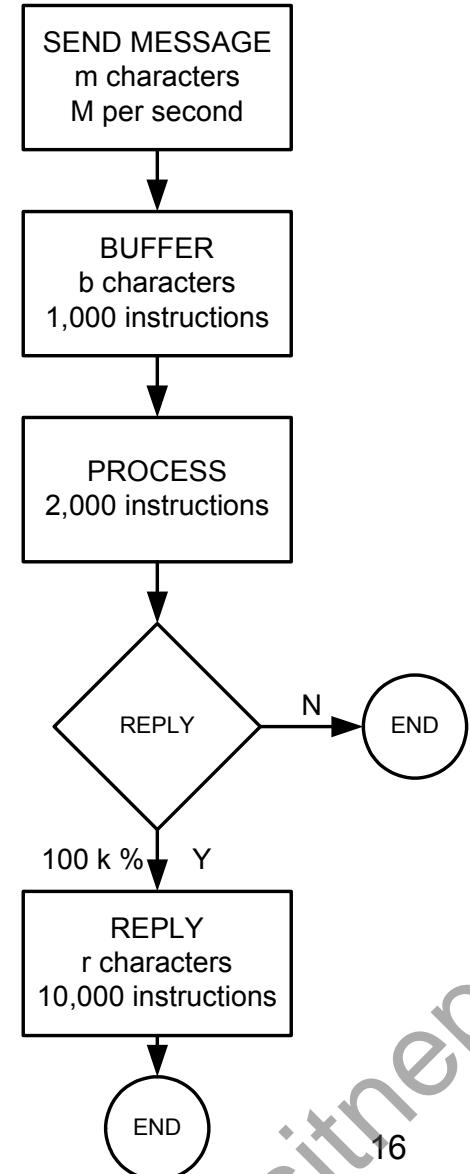
# System Design

Here,

- M messages per second
- kM messages replied per second
- (M.m + k.M.r) characters pass thro buffer
- Since the buffer can hold 'b' chars,  $M(m+k.r)/b$  incomings per second

Altogether, the total number of instructions required;

$$N = 2,000M + 10,000M.k + \frac{1,000M(m + kr)}{b}$$



## System Design

- If 's' is the number of instructions per second that the computer can execute then, the value of 's' to be able to process the given messages M, must be

$$N \leq s$$

- Let's consider that,

$$M = 5 \quad m = 15$$

$$k = 0.1 \quad r = 50$$

- When simplified,  $N \leq s$  becomes;

$$\frac{20}{b} \leq \frac{s}{5,000} - 3$$

- Plot b against  $20/b$

# System Design

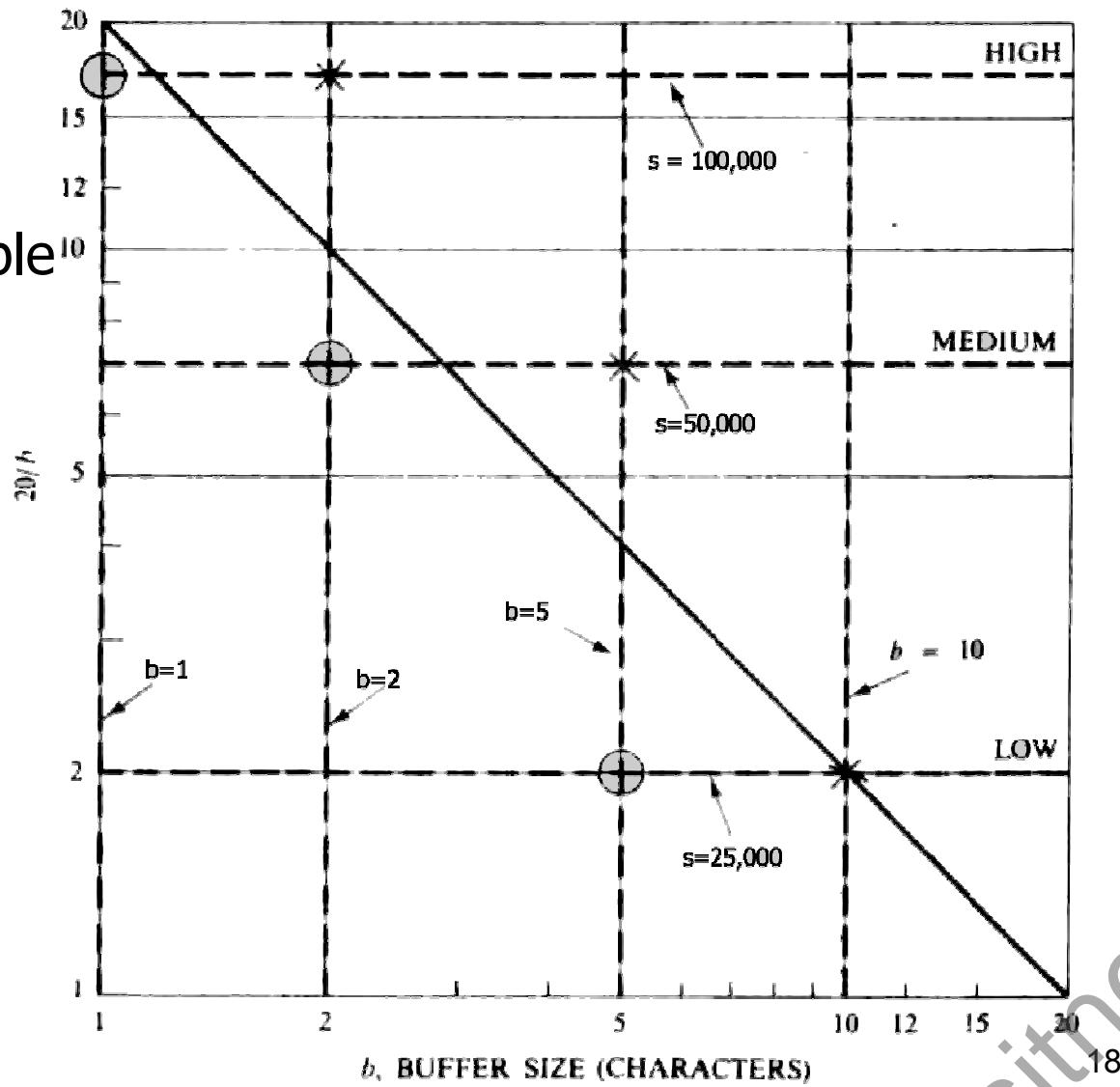
- Plot

- + are impossible

- \* are possible

Analyze for:

Cost

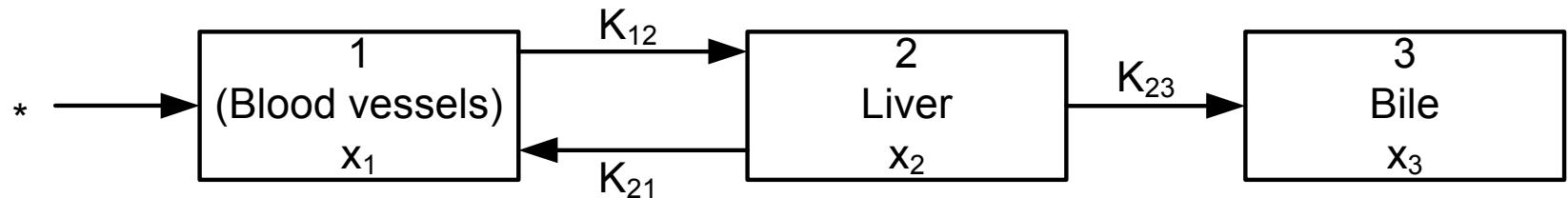


(C) Pramod Parajuli, 2004

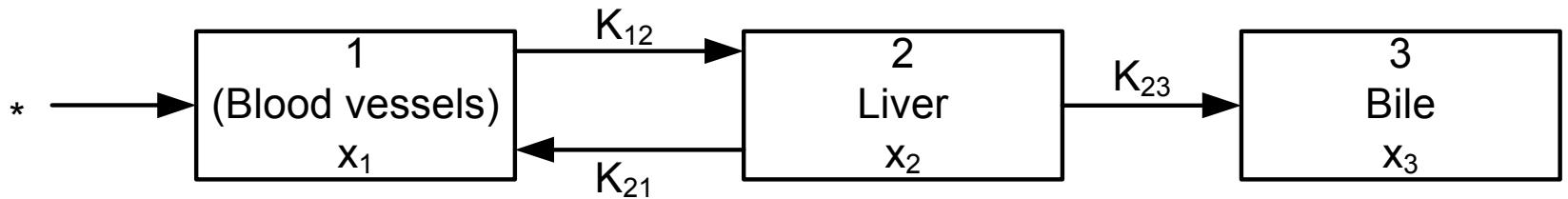
# System Postulation

## Blood filtration in human body

- Bloods from vessels come into the liver
- Liver filters the blood and re-circulates the refined blood to vessels and extracts bile
- The amount of blood filtered and re-circulated can be observed (medical studies) using thyroxine.
- Use of compartments (**compartmental model**) to represent the components



# System Postulation



The overall function can be represented as;

$$\frac{dx_1}{dt} = -k_{12} \cdot x_1 + k_{21} \cdot x_2$$

$$x_1 = C_{11} \cdot e^{-b_1 t} + C_{12} \cdot e^{-b_2 t}$$

$$\frac{dx_2}{dt} = k_{12} x_1 - (k_{21} + k_{23}) \cdot x_2$$

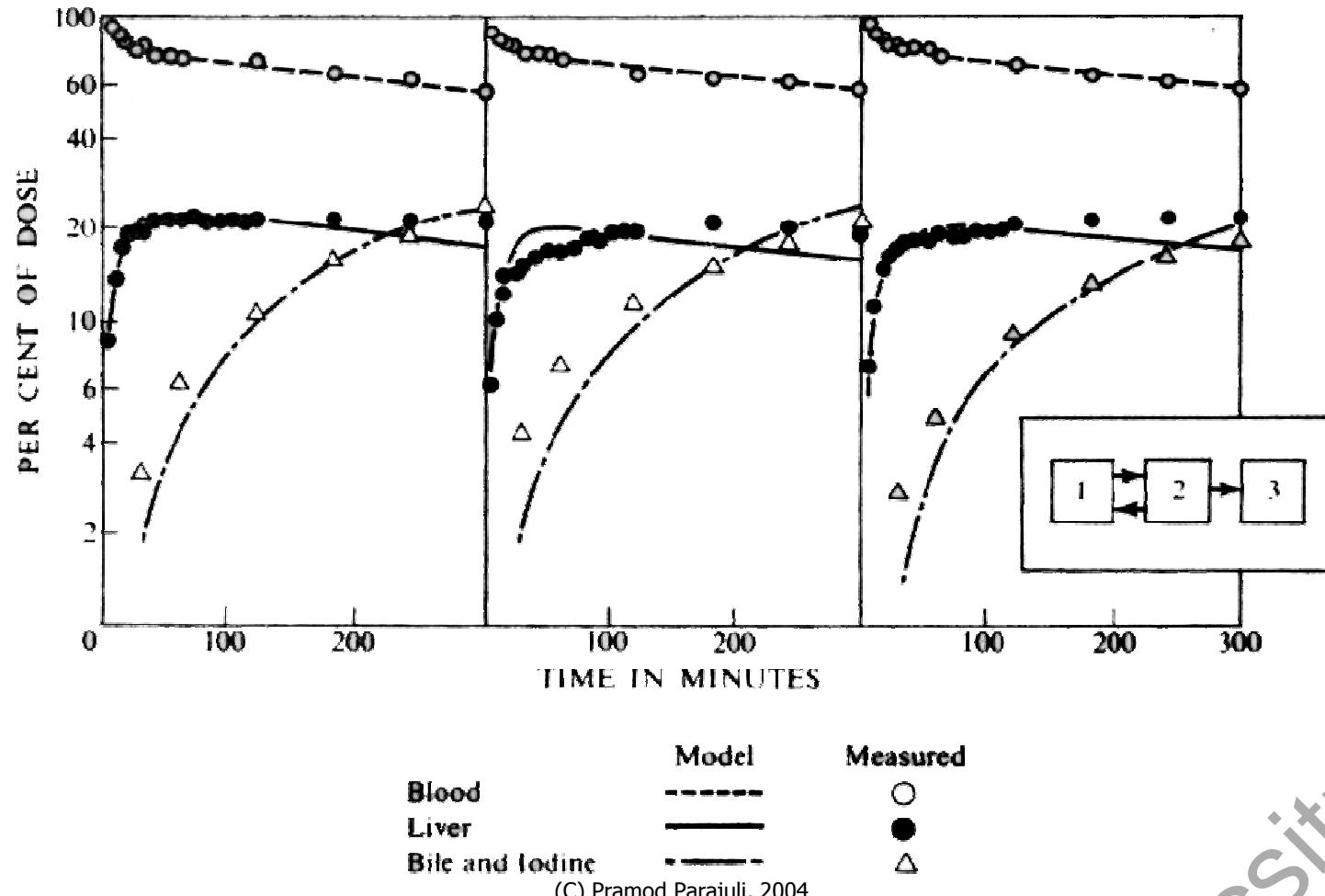
$$x_2 = C_{21} \cdot e^{-b_1 t} + C_{22} \cdot e^{-b_2 t}$$

$$\frac{dx_3}{dt} = k_{23} \cdot x_2$$

$$x_3 = C_{31} + C_{32} \cdot e^{-b_1 t} + C_{33} \cdot e^{-b_2 t}$$

## System Postulation

If the real world readings and the output of the models are plotted, then;



# Pramod Parajuli

# Simulation and Modeling, CS-331

---

## Chapter 3

## System Simulation

(C) Pramod Parajuli, 2004

Source: [www.csitnepal.com](http://www.csitnepal.com)

1  
Csitnepal

# System Simulation

- Analytical methods directly produce general solutions.
- Numerical methods produce solutions in steps; each step gives the solution for one set of conditions.  
The steps are repeated to expand the range of the solution
- In case of static models like corporate model, the distinction between terms 'simulation' and 'numerical computation' can not be distinguished however in dynamic models, it can be
- Solving the equations of the model, step by step, with increasing values of time - ***simulation***

# **System Simulation**

- System simulation – the technique of solving problems by the observation of the performance over time, of a dynamic model of the system
- Not only the time constraint, but randomized selection in a step-by-step transition system is also considered as system simulation

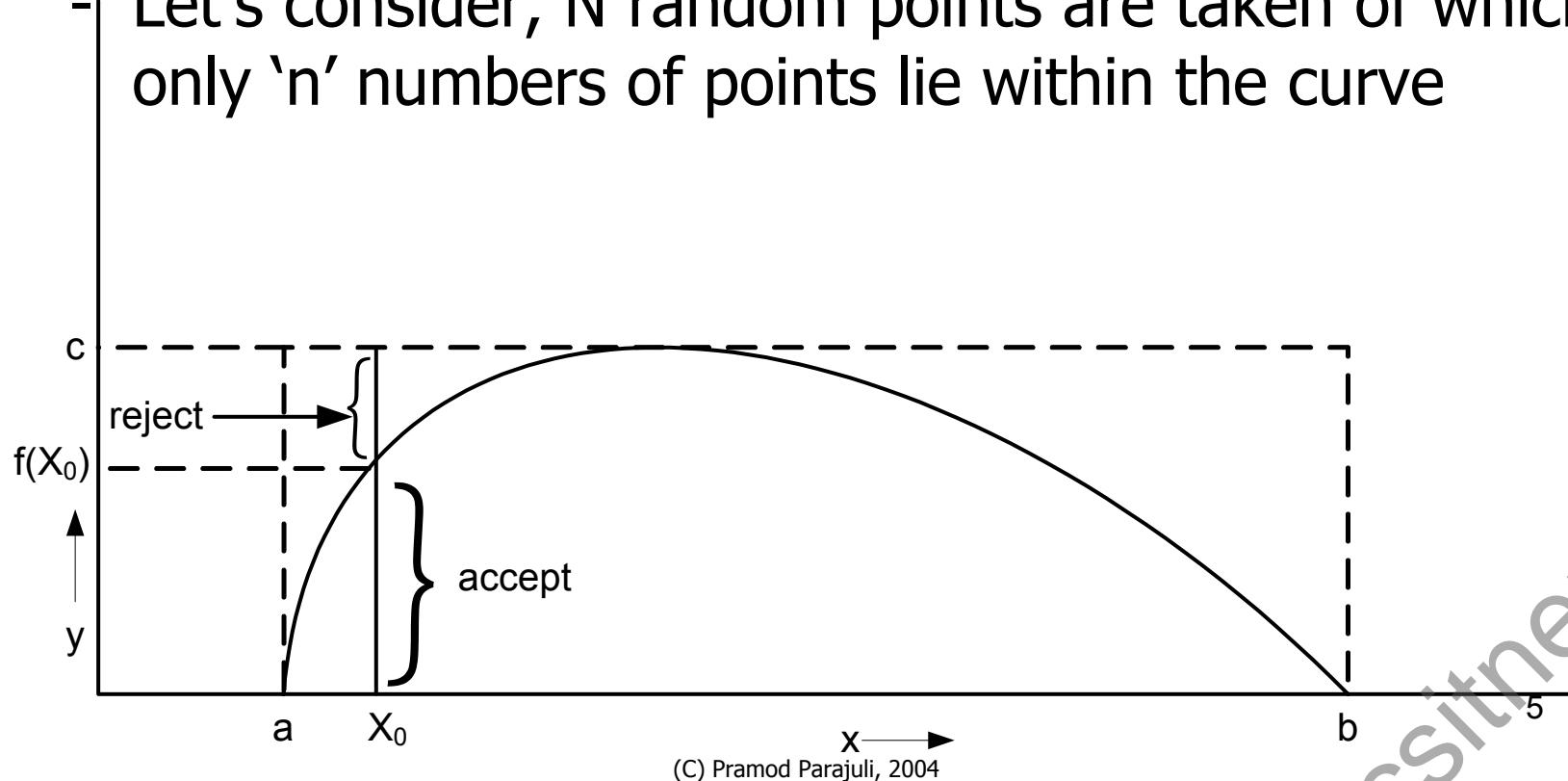
## ***Black board example***

## The Monte Carlo Method

- Originally used for variable reduction
  - Uses random numbers as the input sample
  - Computational technique applied to static models
- 
- Define a function and select random numbers
  - Test the function against the random numbers
  - The good accuracy can be achieved by increasing the number of random samples

# The Monte Carlo Method

- Let  $f(x)$  be a function defined and has lower and upper bounds  $a$ ,  $b$  and  $c$ .
- The function will be bounded by the area  $c*(b-a)$
- Let's consider,  $N$  random points are taken of which only ' $n$ ' numbers of points lie within the curve



## The Monte Carlo Method

- The ratio will lead to;

$$\frac{n}{N} = \int_a^b \frac{f(x)dx}{c(b-a)}$$

- It seems like when the value of 'n' is increased, the accuracy will be increased, but since the value of 'n' again depends on 'N', the accuracy is increased for greater value of N
- Only the points satisfying  $Y \leq f(x_0)$  are accepted otherwise rejected
- If rejected then another point is selected randomly

# **Simulation vs. Analytical methods**

- Simulation gives specific solution for given set of data whereas analytical method gives solution for all of the conditions
- To find the general solution, the simulation needs to be repeated many times
- For analytical study, the system should be modeled by using some specific format
- Simulation consists of lots of step by step execution. If the simulation is run for several times, then there will be increase in detail complexity
- Ideal way for simulation-use extension of mathematical sol<sup>n</sup>

## **Experimental Nature of Simulation**

- Simulation: observe the way in which all variables of the model change with time
- Instead of using analytical solutions, a system might be tested against lots of input samples and the output is observed
- The relationship between input and output can be discovered
- Requires lots of simulation runs
- Popular in system study
- Must be planned as a series of experiments

## **CSMP III – statements**

Printing

PRINT X, Y

Print-plotted

PRTPLT X

Heading of the printed output

TITLE MySimulation System

Heading of the print-plotted output

LABEL Plot Output of MySimulation System

Closing all of the job

ENDJOB

## CSMP III – Example

Example for automobile suspension system

TITLE AUTOMOBILE SUSPENSION SYSTEM

PARAM D = (5.656, 16.968, 39.592, 56.56, 113.12)

X2DOT = (1.0/M) \* (K\*F – K\*X – D\*XDOT)

XDOT = INTGRL(0.0, X2DOT)

X = INTGRL(0.0, XDOT)

CONST M = 2.0, F = 1.0, K = 400.0

TIMER DELT=0.005, FINTIM=1.5, PRDEL=0.05, OUTDEL=0.05

PRINT X, XDOT, X2DOT

PRTPLT X

LABEL DISPLACEMENT VERSUS TIME

END

STOP

## **Combined / hybrid simulation**

Three fundamental types of interactions that can occur between discretely changing and continuously changing state variables;

1. A discrete event may cause a discrete change in the value of a continuous state variable
2. A discrete event may cause the relationship governing a continuous state variable to change at a particular time
3. A continuous state variable achieving a threshold value may cause a discrete event to occur or to be scheduled

# Feedback Systems

- The system takes feedback from the output i.e. input is ***coupled*** with output
- Example can be; heat monitoring and control system
- Issues – amplification and correction of feedback
- ***Negative feedback*** – control variable is proportional with output
- ***Positive feedback*** – control variable and output are inversely proportional
- Other examples;
  - Aircraft system
  - Error Correction mechanism

## Error Correction Mechanism

$Y(t)$  – real output

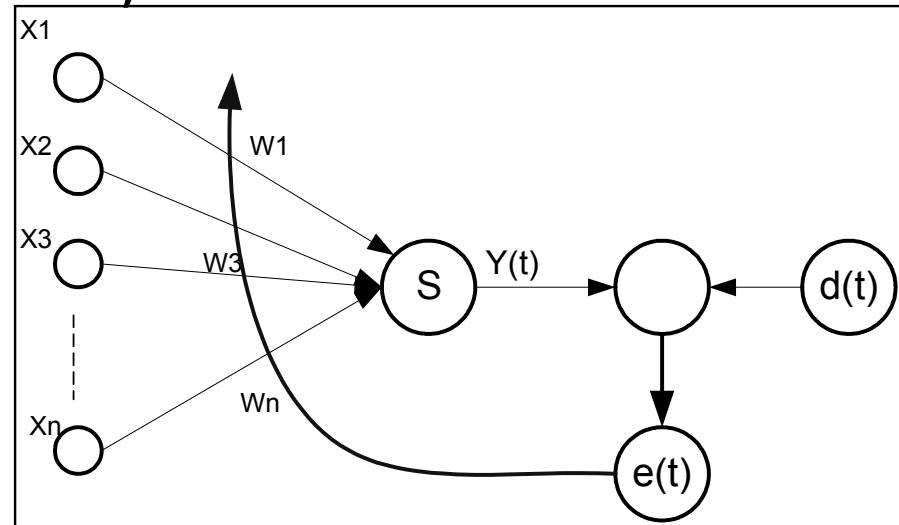
$D(t)$  – desired output

Let error at that instant be;

$$e(t) = D(t) - Y(t)$$

$Y(t)$  is governed as;

$$\sum_{i=0}^n w_i(t) \cdot x_i(t)$$



$e(t)$  is provided as feedback so that it adjusts the values of  $w(t)$

The process is stopped if  $Y(t) = D(t)$

## Interactive Systems

- Systems that support input and output events even the simulation is going on
- Most of the continuous simulation packages provide feature to display output on the screen in the form of text and plots
- Some simulation packages provide facility to change the input parameters as well

## Real-time Simulation

- Some devices and their functions are so critical that any model of such devices/sub-systems could not achieve desirable output
- For such systems, the real devices are used
- The real devices are used to provide input/feedback to the simulation programs. The output is also deployed on the real devices
- Such kind of simulation is known as ***real-time*** simulation
- The main reason behind real-time simulation is to simulate the real world events on the real time i.e. the job must be done within predefined time

## **Homework - 4**

Solve problems  
4-2, and 4-5  
from Chapter – 4 of 'System Simulation' by  
Gordon

# Pramod Parajuli Simulation and Modeling, CS-331

---

## Chapter 5 System Dynamics

(C) Pramod Parajuli, 2004

Source: [www.csitnepal.com](http://www.csitnepal.com)

1  
Csitnepal

# **Introduction**

Control theory – analysis of response of a system for given input signal

## Dynamics

Dynamic change in lots of parameters of systems

Study of influence of parameters on the stability or growth of the system

# Exponential Growth and Decay Models

- Rate of change
- Let k is the fraction of the capital that determines the interest

$$\frac{dx}{dt} = kx \quad x = x_0 e^{kt}$$

$$x = x_0 \quad \text{at } t = 0$$

- In other words, logarithm of the variable increases linearly with time

Decay - same as growth model except the initial value decays

$$\frac{dx}{dt} = -kx \quad x = x_0 e^{-kt}$$

$$x = x_0$$

(C) Pramod Parajuli, 2004

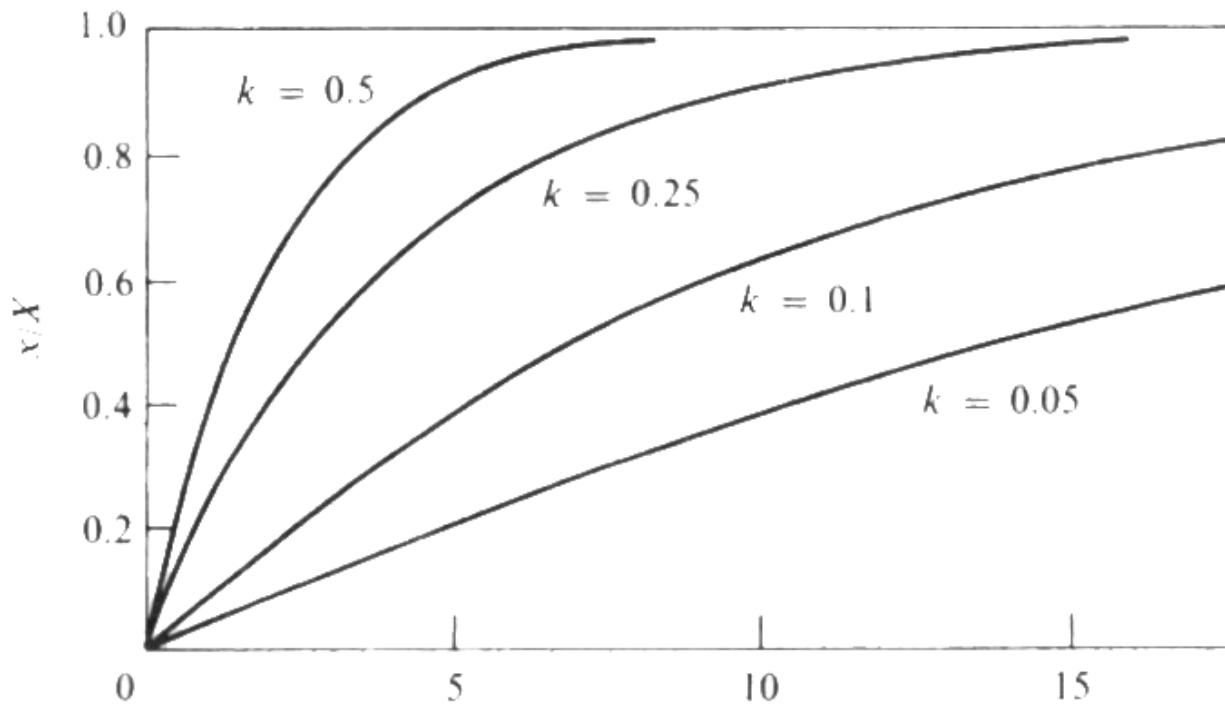
## Modified exponential growth models

If the max./min. value is limited by some amount

e.g. house holds and

$$\frac{dx}{dt} = k(X - x)$$

$$x = X(1 - e^{-kt})$$

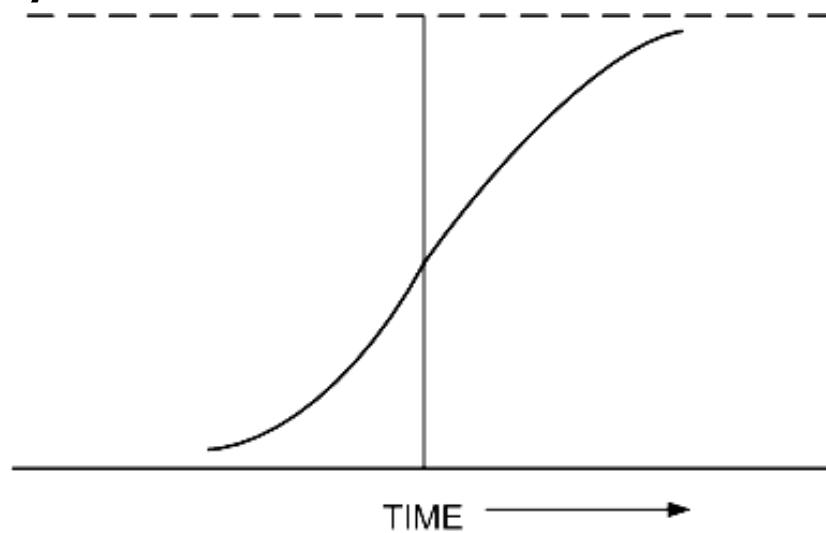


(C) Pramod Parajuli, 2004

## Logistic Curves

The modified exponential growth model gives large slope at beginning

But the reality is somehow different



A logistic function is a combination of exponential and modified exponential function that describes the real process

## Logistic Curves

Modeled using;

$$\frac{dx}{dt} = k \cdot x \cdot (X - x)$$

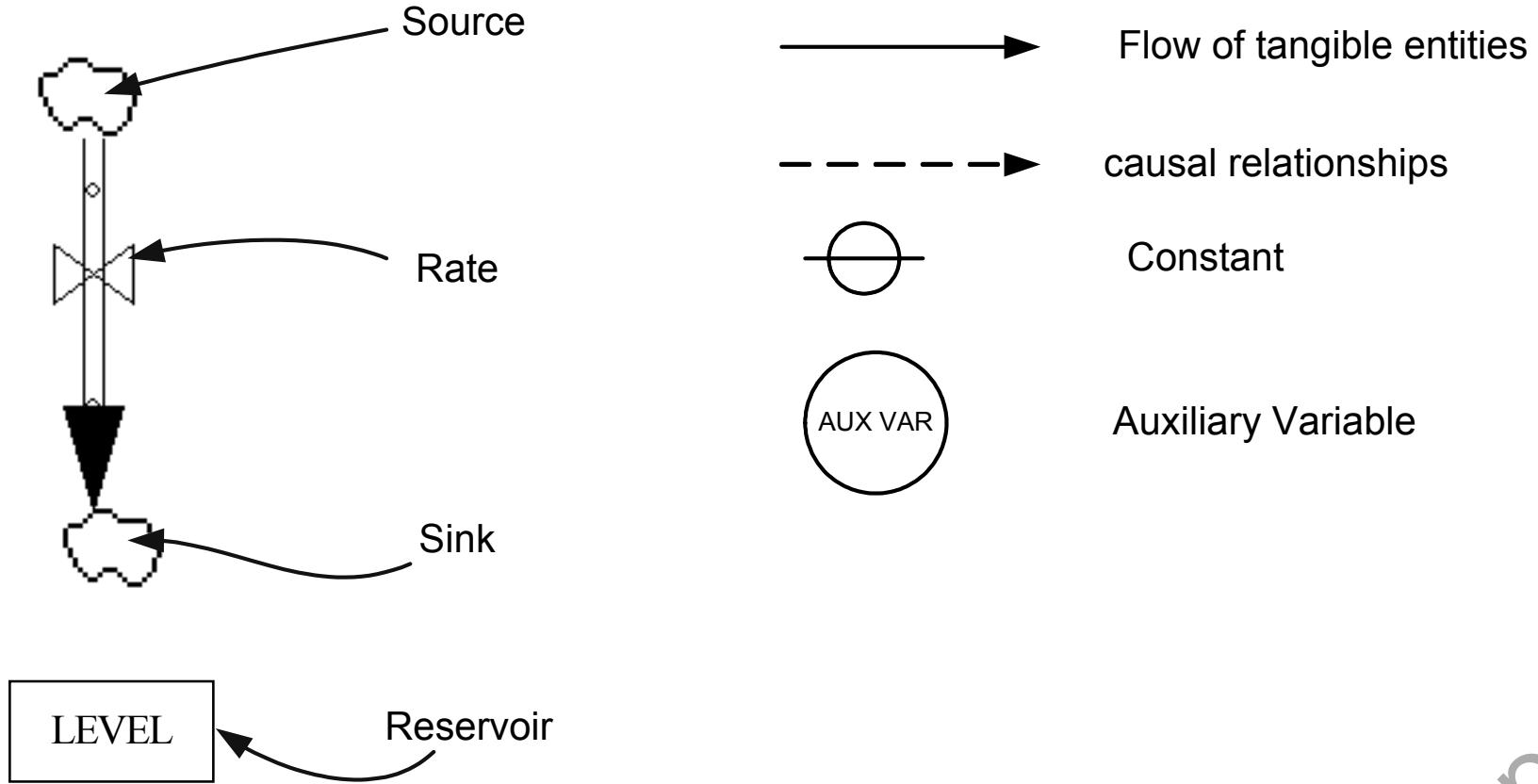
If  $x$  is very smaller than  $X$ , then  $(X-x)$  will be  $x$ ,  
(starting point)

$$\frac{dx}{dt} = k \cdot X \cdot x$$

Again, if  $x$  is very much near to  $X$ , (saturation point)

$$\frac{dx}{dt} = k \cdot X \cdot (X - x)$$

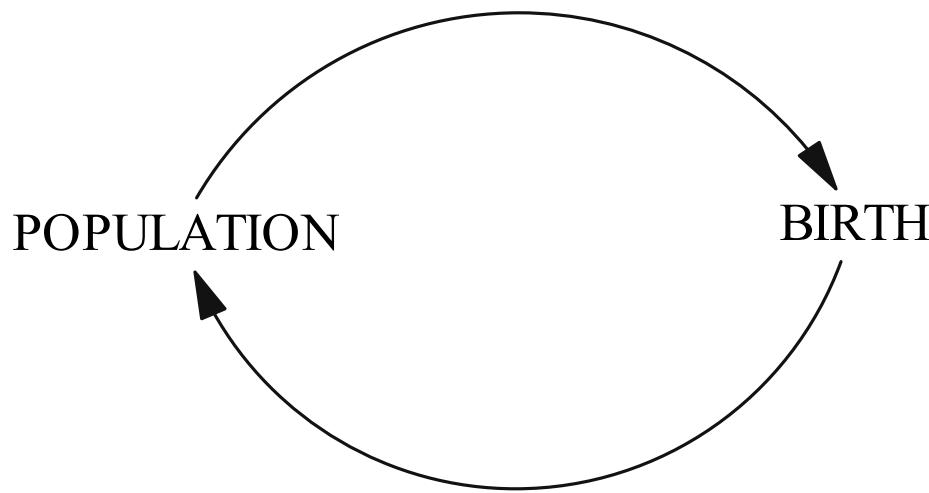
# System Dynamics Diagrams



(C) Pramod Parajuli, 2004

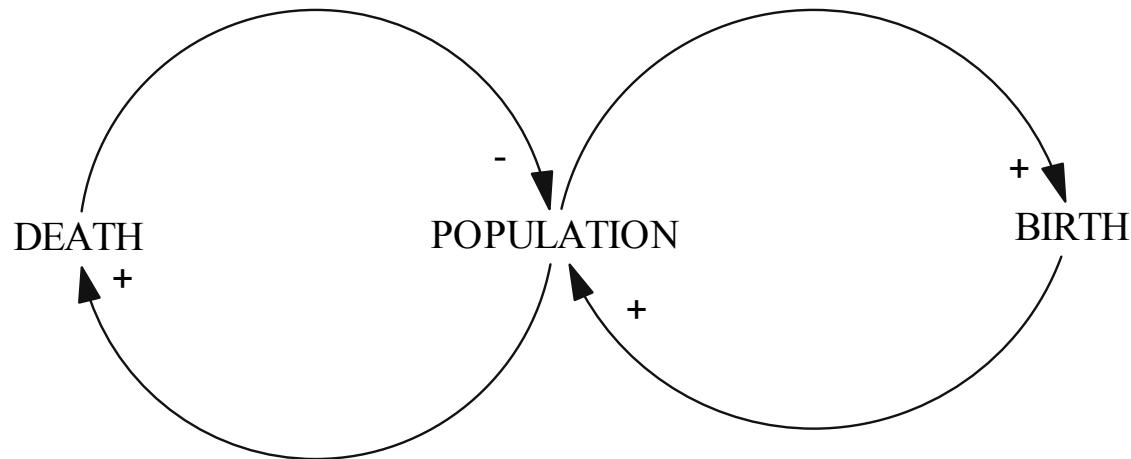
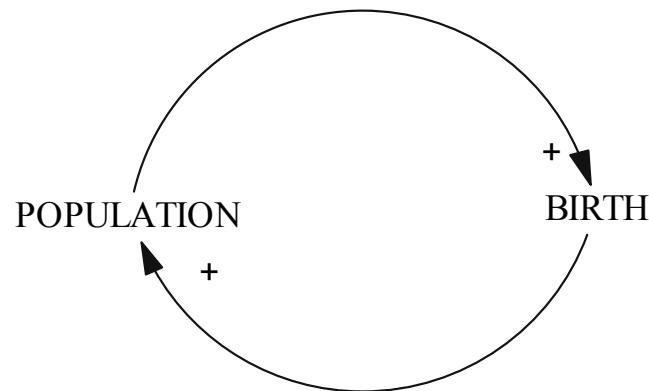
# Causal Loop Diagrams in Systems

Causal loop diagrams (CLD's) are a way of attempting to understand some of the inter-relationships which occur within all systems and processes. They trace cause and effect through a system and, in particular, try to model feedback.



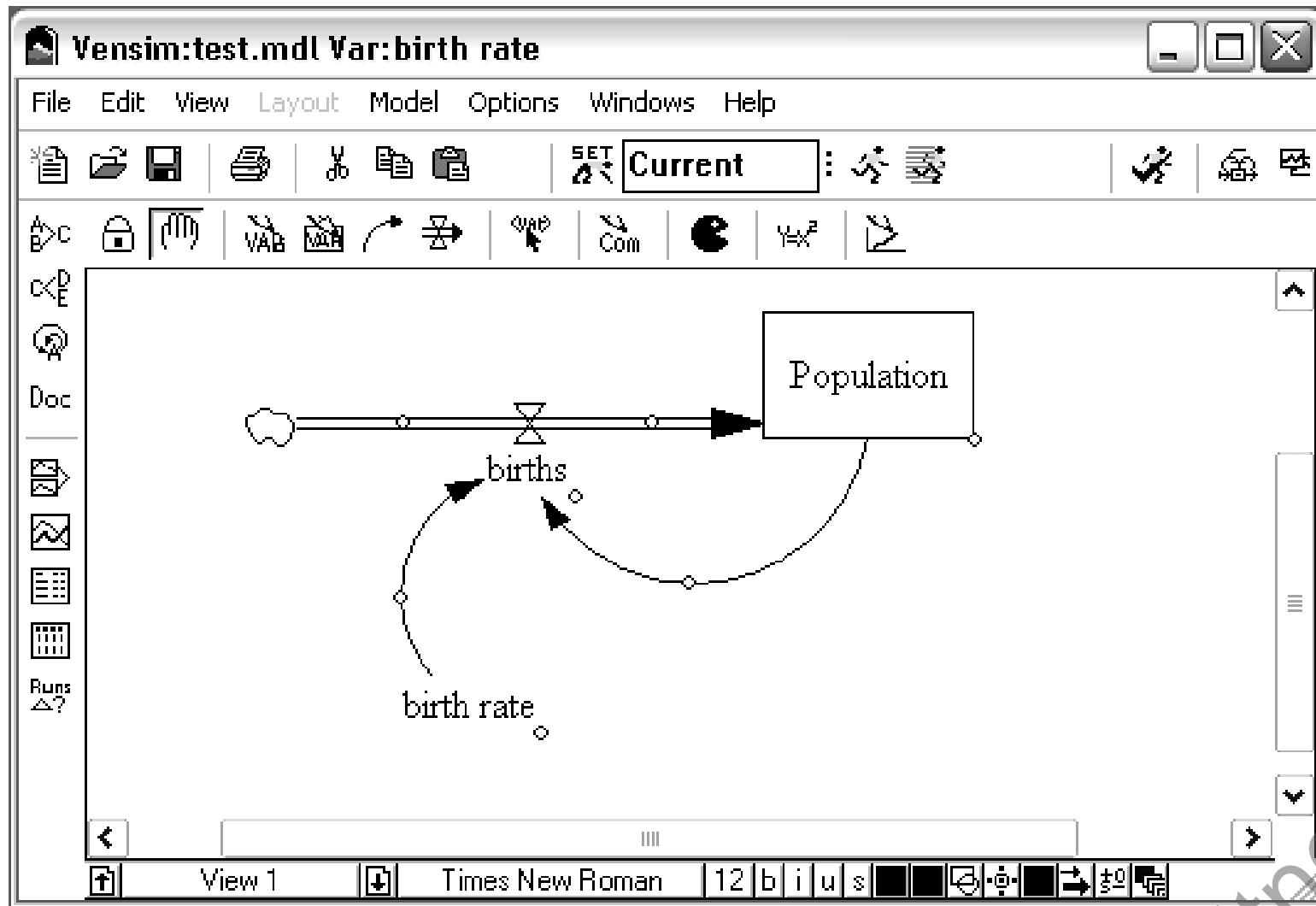
(C) Pramod Parajuli, 2004

# Causal Loop Diagrams in Systems



(C) Pramod Parajuli, 2004

# Vensim PLE – Simple dynamic model



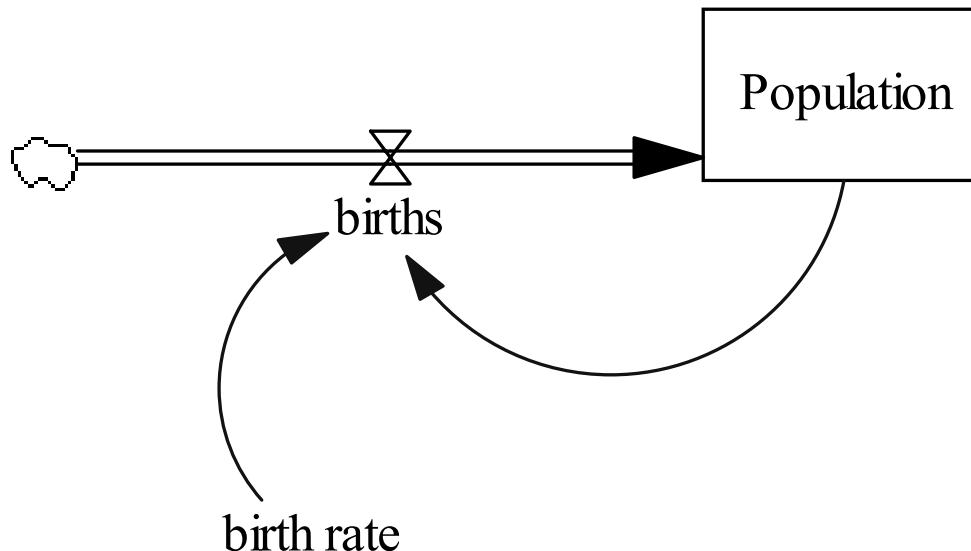
## Vensim PLE – Simple dynamic model

Block variable (level)– **Population** (initial value 1000)

Rate – ***births***

Births – governed by ***birth rate*** – (Population \* birth rate)

Constant - ***Birth rate*** – constant factor (0.12)



(C) Pramod Parajuli, 2004

## Vensim PLE – Simple dynamic model

### Steps – drawing the diagrams

- Use block variable tool to draw '***Population***' block
- Use rate tool to draw '***births***' rate control
- Use variable tool to draw '***birth rate***'
- Draw arrow from '***birth rate***' to '***births***' to indicate the influence of '***birth rate***' on number of births
- Draw arrow from '***Population***' level to '***births***' to indicate the production of '***Population***' and '***birth rate***' results into the actual number of '***births***'

## Vensim PLE – Simple dynamic model

### Steps – the equations

- Click on 'equations' tool
- All the variables that need an equation are highlighted. If the variable is already been defined by an equation, then the variable will not be highlighted
- Click on the variable and enter the equations and other preferences as required
  
- For 'Population' enter the initial value as 1000
- For births, enter the equation as;  
*Population\*birth rate*
- For birth rate, put 0.12 (i.e. the fraction)

13

## Vensim PLE – Simple dynamic model

### Steps – checking equations and units

- Use 'Model->Check model' to check the validity of the model
- If any problem, the Vensim will present you a dialog box for the variable in which there is some problem. You need to modify the equation/unit representation of the variable
- Use 'Model->Units Check' to check validity of each and every units
- If errors are there, then Vensim will present a document which lists the problematic variables and conditions

## **Vensim PLE – Simple dynamic model**

Steps – running and analyzing the simulation

- Simulate using 'Tools -> Simulate'
- It will simulate the dynamic model
- But to have interactive simulation, use 'Tools ->Synthesim'
  
- While using 'Synthesim' you will be able to view the graph plotted against the time in real time

## Vensim PLE – Simple dynamic model

### Steps –analyzing the simulation

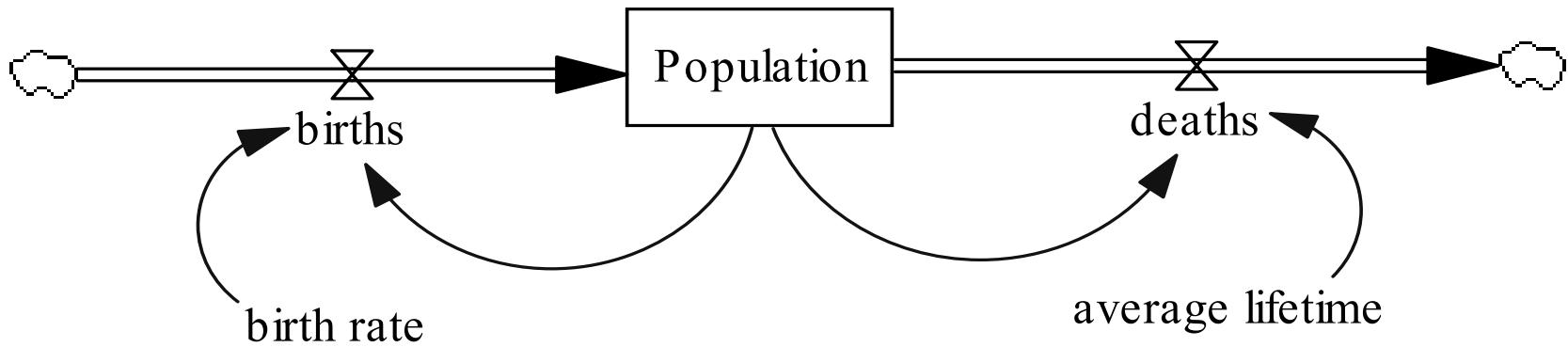
- View causes of the variance using 'Causes Tree' tool
- To view how the variable values are being used, use 'Uses Tree' tool
- Use 'Loops' tool to view iterations if any
- For summary of simulation, use 'Document' tool
- Use 'causes strip' to view the values of causes
- Use 'graph' to visualize the rate of change
- Use 'table' to view tabular data of simulation run
- Use 'runs' to compare two or more runs

## **Vensim PLE – Simple dynamic model**

### Tuning and customizing

- Change the 'initial time', 'fin time', 'step' etc. while building new model
- Change the unit of the time according to your model
- Use 'causes tree', 'loops', 'uses tree' etc. to verify the model

## Dynamic model - example



Initial time = 0

Time step = 0.125

Final time = 100

Units for time = Year

Box variable = Population

Constant variable= birth rate

Constant variable= average lifetime

Rate = births

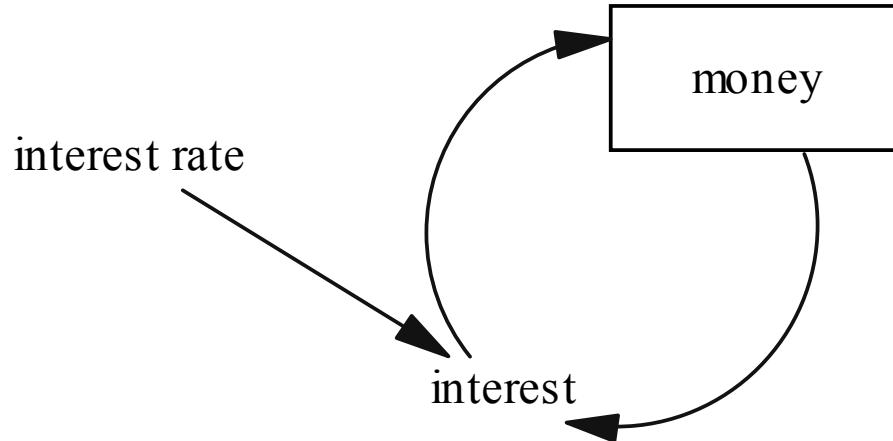
Rate = deaths

## Dynamic model - example

Object type	Name	Int <sup>n</sup>	Equation	Initial value	Unit
Box variable	Population	births-deaths		1000	People
Rate	births		Population*birth rate		People/year
Rate	deaths		Population/average lifetime		People/year
Constant variable	birth rate			0.23	Fraction/year
Constant variable	average lifetime			70	year

# Exponential Growth

Suppose that you deposit \$100 in the bank. If the interest rate is 10% per year (compounded daily) and you wait 100 years what will happen?

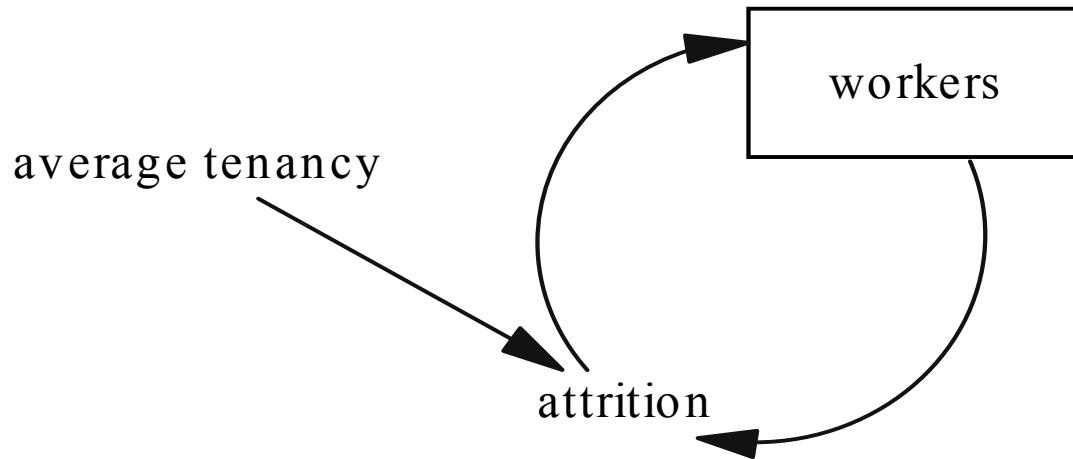


Classroom demonstration

20

## Exponential Decay

Suppose that you have 100 people working for you and you decide never to hire anyone again. Your average worker hangs around for 10 years. What will happen to your workforce?

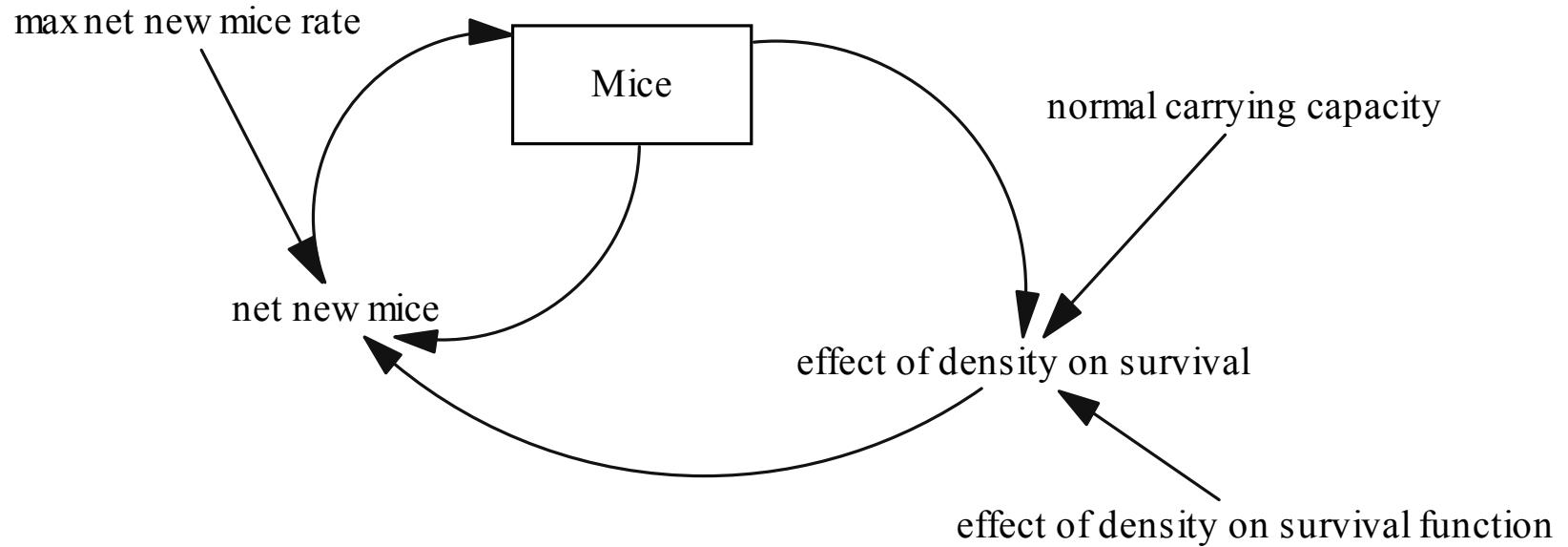


## Homework

21

# Logistic Curves

Suppose that you let some mice loose in your house and don't try to kill them. What will happen to the mouse population?

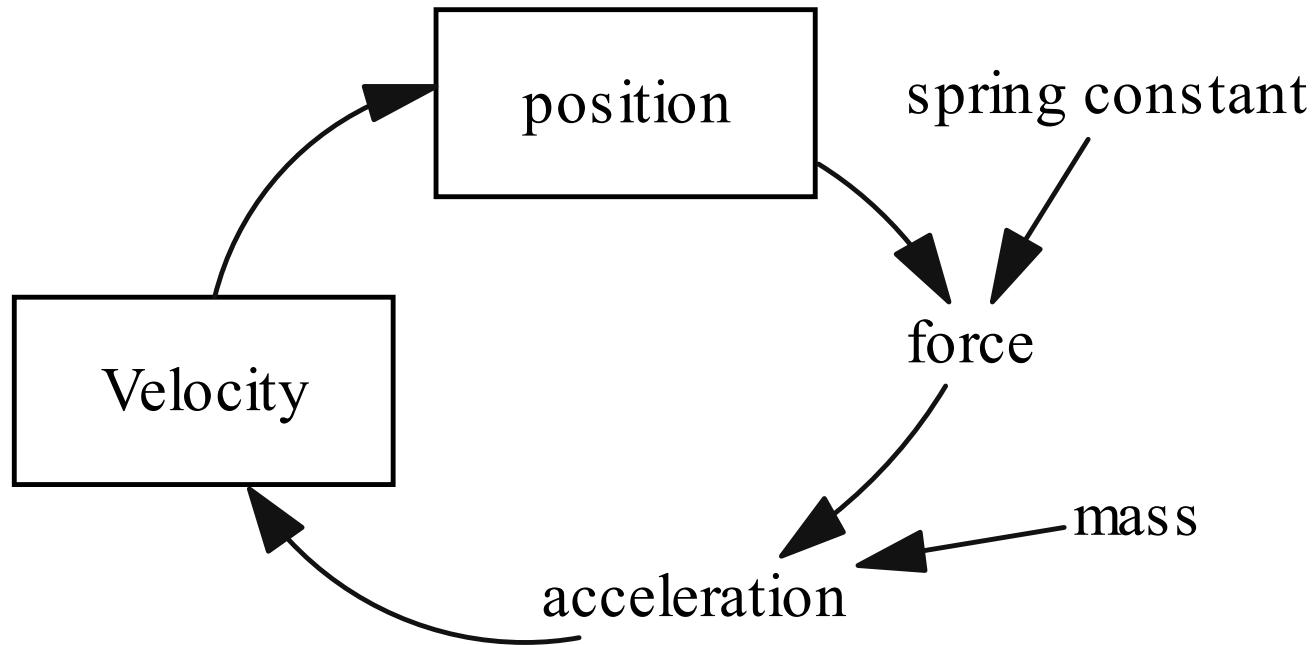


## Homework

22

# Oscillation

Consider the problem of a spring pushing a weight on a frictionless horizontal surface



## Homework

## **Home Work**

Develop Vensim model for figure

5-9 (a, b)

5-10

5-11

5-12

## Time delays

The Constants can be expressed as time constants

$$\text{e.g. } C_1 = 1/T_1$$

which represents that there is time lag of  $T_1$  time unit before the variable associated with  $C_1$  changes

To represent a delay; Let  $x$  is the unit of a level.  $D$  is average time units to process a unit of  $x$ . Then  $dx/dt$  is defined as;

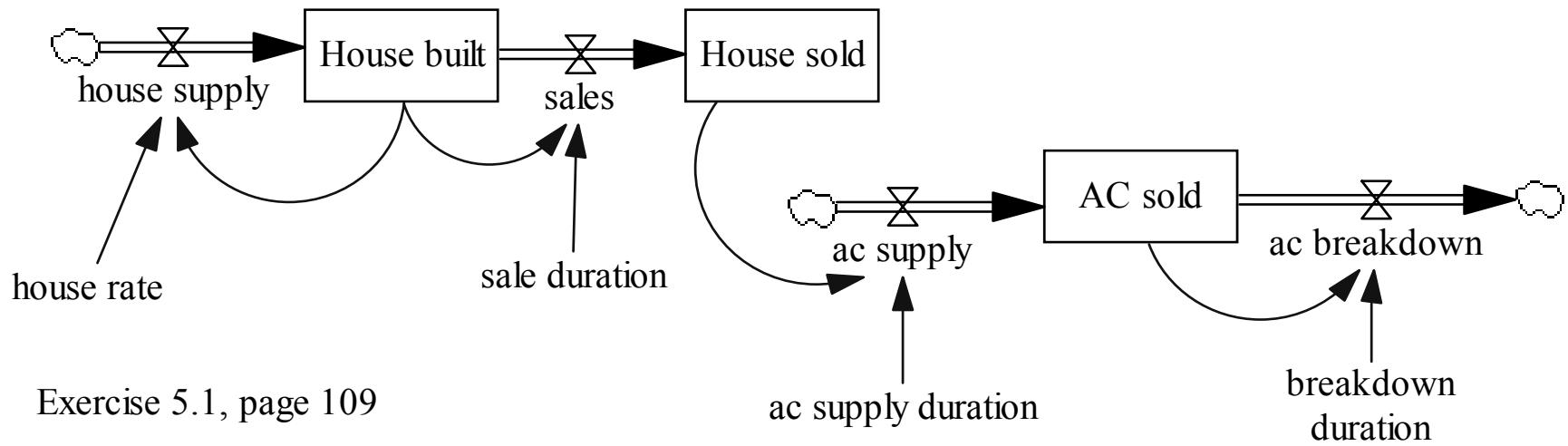
$$\frac{dx}{dt} = \frac{x}{D}$$

$$rate = \frac{x}{D}$$

$$Delay = \frac{Level}{Rate}$$

# Exercises – Chapter 5, Gordon

Q. 5-1



## **Exercise – 5.1**

- (01) ac breakdown=AC sold/breakdown duration  
Units: unit/Month
- (02) AC sold=INTEG (+ac supply-ac breakdown, 0)  
Units: unit
- (03) ac supply = House sold/ac supply duration  
Units: unit/Month
- (04) ac supply duration = 10  
Units: Month
- (05) breakdown duration = 25  
Units: Month
- (06) FINAL TIME = 100  
Units: Month  
The final time for the simulation
- (07) House built= INTEG (house supply-sales,1000)  
Units: unit

## **Exercise – 5.1**

- (08) house rate=0.125  
Units: 1/Month
- (09) House sold= INTEG (sales, 0)  
Units: unit
- (10) house supply=house rate\*House built  
Units: unit/Month
- (11) INITIAL TIME = 0  
Units: Month  
The initial time for the simulation
- (12) sale duration=5  
Units: Month
- (13) sales=House built/sale duration  
Units: unit/Month

# Pramod Parajuli Simulation and Modeling, CS-331

---

## Chapter 6 Probability Concepts

(C) Pramod Parajuli, 2004

Source: [www.csitnepal.com](http://www.csitnepal.com)

1  
Csitnepal

# **Stochastic Variables**

- Ordered set of random values
- Stochastic process give rise to stochastic variable
- Stochastic process can be discrete or continuous

# Discrete Probability Functions

## Probability Mass Function (PMF)

-Distribution of probabilities

-If a variable can take  $n$  different values  $x_n$  ( $n = 1, 2, 3, \dots N$ ), and probability of  $x_n$  being taken is  $p(x_n)$ , then the set of numbers comprising of  $p(x_n)$  is probability mass function

$$\sum_{n=1}^N p(x_i) = 1$$

-In certain cases like dice roll, the PMF values may be known (e.g. 1/6). But most of the time, PMF is counted from input sample

-Example – table 6-1

# Discrete Probability Functions

## Cumulative Distribution Function (CDF)

-For a continuous variable, probability of any one specific value occurring is considered logically to be zero

$$F(x) = P(-\infty < X \leq x)$$

$$F(x) = P(X \leq x) \quad F(-\infty) = 0$$

$$\text{gives : } 0 \leq F(x) \leq 1 \quad F(+\infty) = 1$$

-Let  $f(x)$  is PDF and the probability that 'x' falls in the range  $x_1$  to  $x_2$  is given by

$$= \int_{x_1}^{x_2} f(x) dx \quad = \int_{-\infty}^{\infty} f(x) dx = 1$$

# Discrete Probability Functions

## Cumulative Distribution Function

-As we have seen already, value of 'infinity' is implementation dependent

-But, the cumulative distribution function (CDF) will be

$$F(x) = \int_{-\infty}^x f(x)dx$$

gives,  $0 \leq F(x) \leq 1$

-Therefore, probability of x falling in the range  $x_1$  to  $x_2$  is,

$$F(x_2) - F(x_1)$$

# Discrete Probability Functions

## Probability Density

-If,  $P(x < X \leq x + \Delta x) = f(x).\Delta x$

then,

$$f(x) = \frac{P(x < X \leq x + \Delta x)}{\Delta x}$$

is said to be probability density/density function in continuous system

In discrete system, it is called frequency function

# Measure of Probability Functions

## Mean (Expectation)

$$\text{mean} = \frac{1}{N} \sum_{i=1}^N x_i \quad x_i (i = 1, 2, 3, 4, \dots, N)$$

If observation fall into I groups i.e.  $x_i$  element have  $n_i$  occurrences

$$m = \frac{1}{N} \sum_{i=1}^I n_i \cdot x_i \quad m = \sum_{i=1}^I p(x) \cdot x_i$$

For continuous variable;

$$m = \int_{-\infty}^{\infty} x \cdot f(x) dx$$

$$\begin{aligned}\sum (aX) &= a \sum (x) \\ \sum (x_1 + x_2) &= \sum (x_1) + \sum (x_2)\end{aligned}$$

# **Measure of Probability Functions**

## **Mean (Expectation)**

- Most of the times, the selection of expectation has no occurrence in the sample
- Therefore, it is better to take the nearest sample
- Note; if a value is being multiplied in future, then it must not be corrected. Only after the multiplication it should be corrected. (example)

## **Mode**

Peak probability density function

## **Median**

- Half of the random values will fall below this point
- Normally, cumulative function,  $F(x) = 0.5$

# Measure of Probability Functions

## Variance

- Standard deviation
- Measure of the degree to which data are dispersed from the mean value
- Computed as +ve square root of variance

$$s = \left[ \frac{1}{(N-1)} \sum_{i=1}^N (m - x_i)^2 \right]^{1/2}$$

- If I-groups

$$s = \left[ \sum_{i=1}^I p(x_i) \cdot (m - x_i)^2 \right]^{1/2}$$

# Measure of Probability Functions

## Variance

-For continuous

$$s^2 = \int_{-\infty}^{\infty} f(x)x^2 dx - m^2$$

-If mean 'm' and data are at same point, standard deviation = 0

# Measure of Probability Functions

## Coefficient of Variance

$$\frac{s \tan dard - deviation}{mean - value}$$

-Better than variance as it is expressed in relative terms

## Auto-correlated

-All the data are supposed to be independent of each other

-But if data at  $i^{th}$  step is related to previous data using any kind of relationship they are said to be auto-correlated

-In such case, only the 'mean' holds the meaning, others do not

# Numerical Evaluation of probability functions

- Required for random number generation
- For computation, numerical methods required

## Representation

- Frequency distribution
- Relative frequency distribution

If  $x_i < x \leq x_{i+1}$

then relative frequency distribution=  $p_i = \int_{x_i}^{x_{i+1}} f(x)dx$

density function =  $\frac{p_i}{(x_{i+1} - x_i)}$

For cumulative

$$F_n = \sum_{i=0}^n p_i$$

# Pramod Parajuli

# Simulation and Modeling, CS-331

---

## Chapter 7

Uniform Random Generators

Testing of Uniform Random Generators

Methods of Generating Non-Uniform  
Variables

Inversion, Rejection, Composition

# Introduction

Random numbers

R<sub>1</sub>, R<sub>2</sub>, R<sub>3</sub>, . . .

Must have two essential properties;

- Uniformity
- Independence

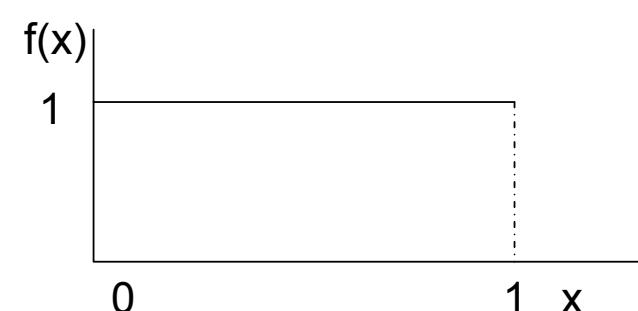
Continuous random variable – Random variable X is continuous if its sample space is a range or collection of ranges of real values

## Continuous Uniformly Distributed RNs

Each random number  $R_i$  is an independent sample from a continuous uniform distribution between 0 and 1

Therefore, PDF =  $f(x) = \begin{cases} 1, & 0 \leq x \leq 1 \\ 0, & \text{otherwise} \end{cases}$

Expected value,  $E(R) = \int_0^1 x dx$   
 $= \left| \frac{x^2}{2} \right|_0^1$   
 $= \frac{1}{2}$



Variance

$$V(R) = \int_0^1 x^2 dx - [E(R)]^2 = 1/12$$

(C) Pramod Parajuli, 2004

## **Generation of RNs – RNGs (RN Generators)**

- Should be fast
- Should be portable
- Should have sufficiently long cycle
- Should be replicable
- Should hold ideal statistical properties (uniformity and independence)
- Linear Congruent Generator

## Linear Congruent Generator (LCG)

-Produces integers between 0 and  $m-1$

$$x_{i+1} = (a \cdot x_i + c) \bmod m$$
$$i = 0, 1, 2, \dots$$

Initial value i.e.  $x_0$  is called seed

$a$  = constant multiplier

$c$  = constant increment

Example;

$$m = 100, x_0 = 27, a = 17, c = 43$$

For random number between 0 and 1,

$$= R_i/m$$

# Multiplicative Congruent Generator (MCG)

-If  $c = 0$

-Example;

$$A = 13, m = 2^6 = 64, x_0 = 1, 2, 3, \text{ and } 4$$

Look for the period/cycle length

Maximum period is  $64/4 = 16$

Here, 4 is the gap between the elements

## Guidelines for selecting the values

Choose 'm' be one more than largest integer that can be stored in one word of the computer being used. Problem, how to represent such a number? Therefore, choose  $(2^{\text{maxword}-1} - 1)$  i.e. in 32 bit number system, choose  $(2^{31}-1)$

The seed  $x_0$  must be relatively prime to m

The constant multiplier 'a' must be selected properly (prime). Normally, it is selected as;

$$a = k \cdot 8 + 3$$

or       $a = k \cdot 8 + 5$

or       $a = 2^{\text{max wordsize}/2} + 3$

or       $a = 65,539$

# Testing Uniform Random Generators

The general idea is to eliminate undesirable characteristics

- Empirical tests – statistical tests, generates uniform random numbers for tests
  - Theoretical tests – uses numerical parameters but do not generate numbers
- 
- Frequency test (uniformity test)
  - Runs test
  - Autocorrelation test
  - Gap test
  - Poker test

## Frequency Tests (uniformity test)

- Counts how often numbers in a given range occur in the sequence to ensure that the numbers are uniformly distributed

Example;

If there are 5,000 3-digit numbers from 000 to 999, then we expect that there are about 500 numbers in the range of 00 to 99. If there are exact 500 numbers in each 10 segments, then the system is not a random one

Therefore, how much deviation should we allow from expected value of 500 and still accept the sequence as uniformly distributed?

Use chi-square (goodness of fit) test

## Chi-square $\chi^2$ test

- Statistical test
- Checks how well certain observed data fit the theoretically expected data
- First divide the observed data (here, the random numbers) into k non-overlapping classes, k must be  $>= 3$
- Then count  $O_i$ , the number of times the observed data falls in each class i,  $i = 1, 2, 3, 4, \dots$
- Then measure how far the observed frequency deviates from the expected

$$chi^2 = \sum_{i=1}^k \frac{(O_i - E_i)^2}{E_i}$$

(C) Pramod Parajuli, 2004

## Chi-square $\chi^2$ test

$$chi^2 = \sum_{i=1}^k \frac{(O_i - E_i)^2}{E_i}$$

- $O_i$  is the observed number of sequences of value  $x_i$  and  $E_i$  is the expected number of occurrences of  $x_i$
- This is showing the square of the differences the observed values and the expected values (divided by the expected values to normalize it)
- Now consider two hypotheses:
  - NULL Hypothesis,  $H_0$  : Y matches distribution of X
  - Hypothesis  $H_1$ : Y does not match distribution of X
- If  $\chi^2$  is too large, we reject the null hypothesis (i.e. the distributions do not match)
- If  $\chi^2$  is small(ish) we do not reject the null hypothesis (i.e. the distributions may match)

## Chi-square test

- If the observed data depart more from the expected value, the value of  $\chi^2$  will be larger
- In our example,  $E_i = 500$  for  $i=1, 2, 3, \dots, 10$

i	Range	Oi no. of observed occurrences	No. of expected occurrences	$(O_i - E_i)^2$	$(O_i - E_i)^2 / E_i$
1	000-099	468	500	1024	2.048
2	100-199	519	500	361	0.722
3	200-299	480	500	400	0.8
4	300-399	495	500	25	0.05
5	400-499	508	500	64	0.128
6	500-599	426	500	5476	10.952
7	600-699	497	500	9	0.018
8	700-799	515	500	225	0.45
9	800-899	463	500	1369	2.738
10	900-999	529	500	841	1.682

(C) Pramod Parajuli, 2004

## Chi-square test

- $\text{Chi}^2 = 19.588$
- For large values, the hypothesis is rejected if

$$X^2 > X^2_{k-1,1-\alpha}$$

where  $X^2_{k-1,1-\alpha}$  is upper critical point

$$X^2_{k-1,1-\alpha} \approx (k-1) \left\{ 1 - \frac{2}{9(k-1)} + z_{1-\alpha} \sqrt{\frac{2}{9(k-1)}} \right\}^3$$

Where  $Z_{1-\alpha}$  is the upper  $1-\alpha$  critical point of the  $N(0,1)$  distribution

Degree of freedom  $v = k - 1$ , where  $k$  is no. of sets

$\alpha = \Pr(\text{reject } H_0 \mid H_0 \text{ true})$ , literally – level of significance

## Chi-square test

- Let's consider, probability of neglecting H<sub>0</sub> is 0.05
- Then,

$$\alpha = 0.05$$

$$1 - \alpha = 0.95$$

$$k = 10$$

$$k - 1 = 9$$

??

$$\alpha = 0.1$$

$$1 - \alpha = 0.9$$

$$k = 10$$

$$k - 1 = 9$$

??

Are we going to reject the hypothesis or not?

## Kolmogorov-Smirnov Test

- Another kind of frequency test
- Compares continuous cdf,  $F(x)$ , of the uniform distribution to the empirical cdf,  $S_N(x)$ , of the sample of the  $N$  observations

$$F(x) = x, \quad 0 \leq x \leq 1$$

- If sample from the random-number generator is  $R_1, R_2, \dots, R_N$ , then the empirical cdf,  $S_N(x)$  is defined by

$$S_N(x) = \frac{\text{number of } R_1, R_2, \dots, R_N \text{ which are } \leq x}{N}$$

- As  $N$  becomes larger,  $S_N(x)$  should become a better approximation to  $F(x)$ , provided that the null hypothesis is true

## Kolmogorov-Smirnov Test

- This test is based on the largest absolute deviation between  $F(x)$  and  $S_N(x)$  over the range of the random variable

Steps

- Rank the data from smallest to largest. E.g.

$$R_{(1)} \leq R_{(2)} \leq R_{(3)} \leq \dots \leq R_{(N)}$$

- Compute

$$D^+ = \max_{1 \leq i \leq N} \left\{ \frac{i}{N} - R_{(i)} \right\}$$

$$D^- = \max_{1 \leq i \leq N} \left\{ R_{(i)} - \frac{i-1}{N} \right\}$$

## Kolmogorov-Smirnov Test

3. Compute  $D = \max(D^+, D^-)$
4. Determine the critical value  $D\alpha$  from Kolmogorov-Smirnov Critical values table
5. If  $D > D\alpha$ , the null hypothesis is rejected, otherwise accepted

Example 7.6, page 267

## Runs test – runs up and runs down

- Direct test of independence (only) assumption
- Chi-square test might show some numbers are uniformly distributed but the ordering matters in case of independence
- Look at example on page 270 of Discrete Event System Simulation, Banks, Nicol, Nelson
- The general idea is to look at the patterns of runs up and runs down
- E.g.

0.08 0.18 0.23 0.36 0.42 0.55 0.63 0.72 0.89 0.91

One run – run up

0.08 0.93 0.15 0.96 0.26 0.84 0.28 0.79 0.36 0.57

Nine run – five up, four down

## Runs test – runs up and runs down

- The run pattern is likely to be somewhere between two extremes
- If N is the number of numbers in the sequence, the maximum number of runs is N-1 and minimum number of runs is one
- Now, if  $a$  is total number of runs in truly random sequence, the mean and variance of  $a$  are given by

$$\mu_a = \frac{2N - 1}{3}$$

$$\sigma_a^2 = \frac{16N - 29}{90}$$

- For  $N > 20$ , the distribution of  $a$  is approximated by normal distribution

$$N(\mu_a, \sigma_a^2)$$

(C) Pramod Parajuli, 2004

## Runs test – runs up and runs down

- Therefore, standardized normal test parameter

$$Z_0 = \frac{a - \mu_a}{\sigma_a}$$

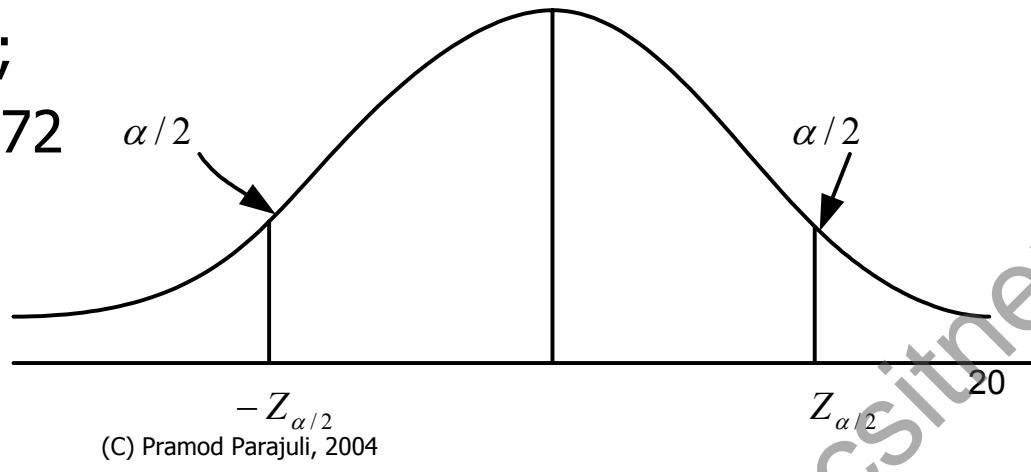
$$Z_0 = \frac{a - [(2N-1)/3]}{\sqrt{(16N-29)/90}}$$

- Where,  $Z_0$  is approximately equal to  $N(0,1)$
- Therefore, the hypothesis can not be rejected if

$$-Z_{\alpha/2} \leq Z_0 \leq Z_{\alpha/2}$$

Class-demonstration;

Example 7.8, page 272



(C) Pramod Parajuli, 2004

## Runs test – runs above and below the mean

- Runs up and down don't indicate any absolute organization of the data (or lack thereof)
- We may also want to test the number of runs (i.e consecutive values) above and below the mean
  - Too many or too few is not a good sign
  - Calculation for mean and variance is more complex, but we handle this test in more or less the same way as the runs up and runs down test
    - Don't want the our number to be too far from the mean
    - Use the standard normal distribution to test

## Runs test – runs above and below the mean

If  $n_1$  and  $n_2$  is the number of individual observation above and below the mean and  $b$  is total number of runs,

mean  $\mu_b = \frac{2n_1 n_2}{N} + \frac{1}{2}$

variance of b  $\sigma_b^2 = \frac{2n_1 n_2 (2n_1 n_2 - N)}{N^2 (N - 1)}$

If  $n_1$  or  $n_2$  is greater than 20,  $b$  is approximately normally distributed

## Runs test – runs above and below the mean

Let's define standardized variable  $Z_0$  for testing

$$Z_0 = \frac{b - (2n_1 n_2 / N) - 1/2}{\left[ \frac{2n_1 n_2 (2n_1 n_2 - N)}{N^2 (N-1)} \right]^{1/2}}$$

Failure to reject the hypothesis of independence

occurs when  $-Z_{\alpha/2} \leq Z_0 \leq Z_{\alpha/2}$

where  $\alpha$  is the level of significance

Class-demonstration

Example 7.9, page 274

## Runs test – length of runs

Given a sequence of random numbers, we can expect a certain number of runs of each length 1, 2, 3, ...

If the run lengths in our empirical tests differ too much from the expected distribution, we can reject the generator

Idea is to determine the empirical distribution of runs of various lengths, then compare it with the expected distribution given random data

The distributions are compared using the Chi-Square test

Sample from book, page 274

In the example, we saw that run with length two is occurring. In reality, it must be random

## Runs test – length of runs

As in chi-square test for the samples, the runs themselves should also poses uniformity

Determined by;  $\chi_0^2 = \sum_{i=1}^L \frac{[O_i - E(Y_i)]^2}{E(Y_i)}$

where,

$Y_i$  = number of runs of length  $i$  in a sequence of  $N$  numbers

$L = N - 1$  for runs up and down

$L = N$  for runs above and below the mean

Derivations – on blackboard

## **Runs test – length of runs**

If null hypothesis of independence is true, then  $\chi_0^2$  is approximately chi-square distributed with L-1 degrees of freedom

Example 7.10 – page 275

## Autocorrelation test

- Tests relationship of items  $m$  (lag) locations apart
- Autocorrelation is determined for
$$R_i, R_{i+m}, R_{i+2m}, \dots, R_{i+(M+1)m}$$
- The value of  $M$  is the largest integer that satisfies;

$$i + (M+1)m \leq N$$

where  $N$  is total number of values in the sequence

- A nonzero autocorrelation implies a lack of independence. Let

$$H_0 : \rho_{im} = 0$$

$$H_1 : \rho_{im} \neq 0$$

(C) Pramod Parajuli, 2004

## Autocorrelation test

- For large value of M, the distribution of the estimator of  $\rho_{im}$  ( $\hat{\rho}_{im}$ ) is approximately normal if the values  $R_i, R_{i+m}, R_{i+2m}, \dots, R_{i+(M+1)m}$  are uncorrelated
- Therefore;

$$Z_0 = \frac{\hat{\rho}_{im}}{\sigma\hat{\rho}_{im}}$$

where

$$\hat{\rho}_{im} = \frac{1}{M+1} \left[ \sum_{k=0}^M R_{i+km} R_{i+(k+1)m} \right] - 0.25$$

$$\sigma\hat{\rho}_{im} = \frac{\sqrt{13M+7}}{12(M+1)}$$

## Autocorrelation test

- Do not reject the null hypothesis of independence if  $-Z_{\alpha/2} \leq Z_0 \leq Z_{\alpha/2}$
- If  $\rho_{im} > 0$ , the subsequence is said to exhibit positive autocorrelation. High random numbers in the subsequence followed by high, and low followed by low)
- If  $\rho_{im} < 0$ , the subsequence is said to exhibit negative autocorrelation. Low random numbers tend to be followed by high ones, and vice versa
- For desired property of independence, which implies zero autocorrelation

## Gap test

- The gap test is used to determine the significance of the interval between the recurrences of the same digit
- The gap of length  $x$  occurs between the recurrences of some specified digit
- E.g.

4, 1, 3, 5, 1, 7, 2, 8, 2, 0, 7, 9, 1, 3, 5, 2, 7, 9,  
4, 1, 6, 3, 3, 9, 6, 3, 4, 8, 2, 3, 1, ..

For digit 3, the first gap is 10 and second gap is 7.

$$P(\text{gap of } 10) = \overbrace{P(\text{no 3}), \dots, (P \text{ no 3})}^{\text{P(3)}} P(3)$$

(C) Pramod Parajuli, 2004

## Gap test

- The probability of the digits not being 3 is 0.9  
Therefore,  $P(\text{gap of } 10) = (0.9)^{10}(0.1)$
- The theoretical frequency distribution for randomly ordered digits is given by

$$P(\text{gap} \leq x) = F(x) = 0.1 \sum_{n=0}^x (0.9)^n = 1 - 0.9^{x+1}$$

### Steps

1. Specify the cdf for the theoretical frequency distribution given by  $P(\text{gap} \leq x)$  based on the selected class interval width
2. Arrange the observed sample of gaps in a cumulative distribution with these same classes

## Gap test

### Steps

3. Find D, the maximum deviation between  $F(x)$  and  $S_N(x)$  as in  $D = |F(x) - S_N(x)|$
4. Determine the critical value,  $D_\alpha$ , from Kolmogorov-Smirnov Critical Values table for the specified value of  $\alpha$  and the sample size N
5. If the calculated value of D is greater than the tabulated value of  $D_\alpha$ , the null hypothesis of independence is rejected

Example 7.13, page 282

## Poker test

Test for interdependence based on the frequency with which certain digits are repeated in a series of numbers

e.g.

0.255, 0.577, 0.331, 0.414, 0.828, 0.909, 0.303, 0.001..

Here, there are three possibilities

1. The individual numbers can all be different
2. The individual numbers can all be the same
3. There can be one pair of like digits

The probability associated with each of these possibilities is given as;

$$\begin{aligned} P(\text{three different digits}) &= P(\text{second different from the first}) * \\ &\quad P(\text{third different from the first and second}) \\ &= (0.9)*(0.8) \\ &= 0.72 \end{aligned}$$

## Poker test

$$\begin{aligned} P(\text{three like digits}) &= P(\text{second digit same as the first}) * \\ &\quad P(\text{third digit same as the first}) \\ &= (0.1) * (0.1) = 0.01 \end{aligned}$$

$$P(\text{exactly one pair}) = 1 - 0.72 - 0.01 = 0.27$$

Alternatively, the last result can be obtained as follows;

$$P(\text{exactly one pair}) = \binom{3}{2}(0.1)(0.9) = 0.27$$

Normally used with chi-square test.

Example 7.14, page 283

## Code snippet

```
/* Returns values between 0 and 1*/
/* Depends on 32 bit representation for integers */
double randu( seed )
    long int *seed;
{
    long int a = 16807,
            mod = 2147483647; /* 2^31 - 1 */
    double dmod = 2147483647.0; /* 2^31 - 1 */

    *seed = *seed * a;
    if ( *seed < 0 )
        *seed += mod;
    return( ( double ) *seed / dmod );
}
```

## Discrete distributions

In real life, the probability of occurring one kind of event/data has different probability than other kind of event/data

Normally, there is a need for discrete but non-uniform distribution

Therefore, while generating a uniform random number 'U', the value is compared with the values of y. If the value falls in a interval  $y_i < U \leq y_{i+1}$  ( $i = 0, 1, \dots, 4$ ) the corresponding value of  $x_{i+1}$  is taken as the desired output

No. of items (x)	Probability p(x)	Cumulative prob. (y)
0	0	0
1	0.10	0.10
2	0.51	0.61
3	0.19	0.80
4	0.15	0.95
5	0.05	1.00

(C) Pramod Parajuli, 2004

36

## Discrete distributions

When the number U is being tested, only 10% of the numbers are validated by using the first entry where as only 61% of numbers are validated by using second entry

For such cases, the table can be rearranged as the ordering doesn't matter for testing

Cumulative prob. (y)	No. of items (x)
0.51	2
0.70	3
0.85	4
0.95	1
1.00	5

Now, the first entry can satisfy 51% of data

## **Non-uniform continuously distributed RNs**

- Random numbers drawn from a distribution that is continuous & non-uniform
- These methods are based on use of sequences of uniformly distributed random numbers
- The algorithm must satisfy
  - Exactness – the random variates must follow the desired distribution
  - Efficient – in terms of time (setup time, marginal execution time) and space
  - Less Complexity
- Inverse transformation method – most widely used
- Rejection, Composition, Convolution etc.

## Non-uniform continuously distributed RNs

### Inverse transformation method

If  $U_i$  ( $i = 1, 2, 3, 4, \dots$ ) are independent variables, uniformly distributed over the interval 0 to 1, and  $F^{-1}(x)$  is the inverse of the cumulative distribution function for random variable  $x$ , then the random variables defined by  $x_i = F^{-1}(U_i)$  are the random sample of  $x$ .

If desired PDF is

$$f(x) = \begin{cases} a(1-x) & (0 \leq x \leq 1) \\ 0 & otherwise \end{cases}$$

Cumulative;

$$F(x) = a\left(x - \frac{x^2}{2}\right)$$

(C) Pramod Parajuli, 2004

## Non-uniform continuously distributed RNs

Let  $U = F(x)$

$$x = 1 - (1 - U)^{1/2}$$

<u>U</u>	<u>X</u>
0.1009	69.23
0.3754	88.48
0.0842	66.65
0.9901	142.33

## Non-uniform continuously distributed RNs

For exponential distribution

CDF

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

now, find U(0,1)

then,

$$F(x) = R$$

or

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

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

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

$$x = \left( -\frac{1}{\lambda} \ln(R) \right)$$

$$\beta = \frac{1}{\lambda} \text{ mean value}$$

R is cumulative

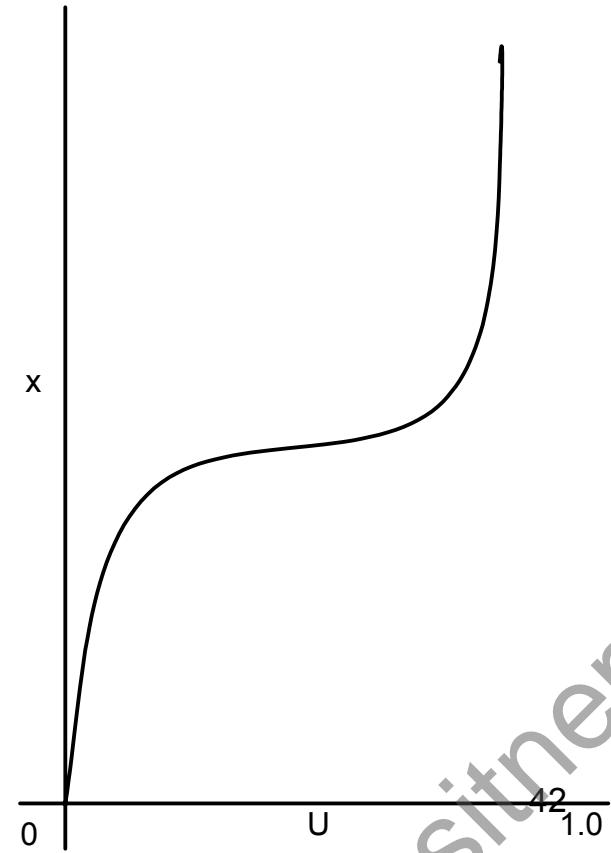
## **Non-uniform continuously distributed RNs**

In case of discrete data, if the result falls between two values, then upper value is taken

But in case of continuous system, any value might occur

In such case, interpolation is applied

The plot of uniform random numbers and the input data ( $x$ ) is;

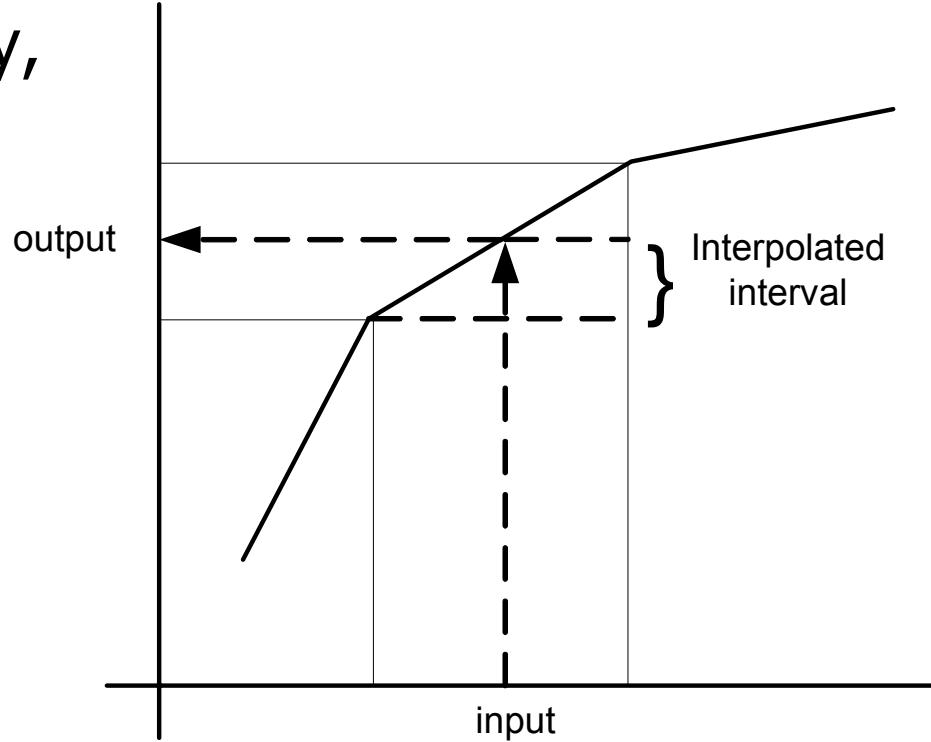


(C) Pramod Parajuli, 2004

## Non-uniform continuously distributed RNs

The general idea is, if a value falls in between two values, the result is calculated by using lower value plus portion of the output that is divided by the input

Graphically,



(C) Pramod Parajuli, 2004

## Non-uniform continuously distributed RNs

The standard curve and its points do not change

Let's define the slope of the segment as;

$$a_i = \frac{x_{i+1} - x_i}{y_{i+1} - y_i} \quad (i = 0, 1, 2, 3, \dots, N-1)$$

now,

$$\text{if } Y_i < U < Y_{i+1} \quad (i = 0, 1, 2, 3, \dots, N-1)$$

then,

$$x = x_i + a_i(U - Y_i)$$

<u>U</u>	<u>X</u>
0.1009	69.29
0.3754	68.48
0.6842	66.65 . .

## **Non-uniform continuously distributed RNs**

### Disadvantage of inverse-transform

1. For normal and gamma distribution, there is no mathematical expression of  $F^{-1}$
2. For given distribution, it may not be the fastest way

### Advantages

1. Even if the distribution is truncated, can generate the output
2. Facilitates variance reduction that rely on correlation

## Non-uniform continuously distributed RNs

### Composition

Applies when the distribution function  $F$  from which we wish to generate can be expressed as a convex combination of other distributions  $F_1, F_2, \dots$

For all  $x$ ,

$$F(x) = \sum_{j=1}^{\infty} p_j F_j(x)$$

$$p_j \geq 0$$

$$\sum_{j=1}^{\infty} p_j = 1$$

$F_j$  is distribution function

## Non-uniform continuously distributed RNs

If,  $p_k > 0, p_j = 0, j > k$   
then, it will be finite

$$\text{pdf} = f(x) = \sum_{j=1}^{\infty} p_j \cdot f_j(x)$$

### Algorithm

1. Generate positive random number  $J$  such that
$$P(J - j) = p_j \quad \text{for } j = 1, 2, 3, 4, \dots$$
2. Return  $X$  with distribution function  $F_J$

# Non-uniform continuously distributed RNs

## Convolution

Probability distribution of a sum of two or more independent random variables is a convolution of the distributions of the original variables

## Algorithm

Let  $F$  be the distribution function of  $X$  and  $G$  be the distribution function of  $Y_j$

1. Generate  $Y_1, Y_2, Y_3, \dots, Y_m$ , independent and identically distributed random variables each with distribution function  $G$
2. Return,  $X = Y_1 + Y_2 + Y_3 + \dots + Y_m$

# Non-uniform continuously distributed RNs

## Rejection method

Applied when the PDF  $f(x)$  has a lower and upper limit to its range  $a$  and  $b$  and upper bound  $c$ .

However, the upper bound 'c' is not very much relevant

## Steps

1. Compute the values of two independent uniformly distributed variables  $U_1$  and  $U_2$
2. Compute,  $X_0 = a + U(b - a)$
3. Compute,  $Y_0 = c.U_2$
4. If  $Y_0 \leq f(X_0)$  accept  $X_0$  otherwise repeat with two new uniform variates

## **Non-uniform continuously distributed RNs**

Recall the Monte Carlo method

The probability density function is enclosed in a rectangle i.e. if a point is accepted, 'n' is incremented by 1

The curve  $f(x)$  represents probability density function. Therefore, the area under the curve must be 1. But,

$$c.(a-b) = 1$$

In such case, the graph must be chosen so that  $c.(a-b)$  and  $f(x)$  both are 1

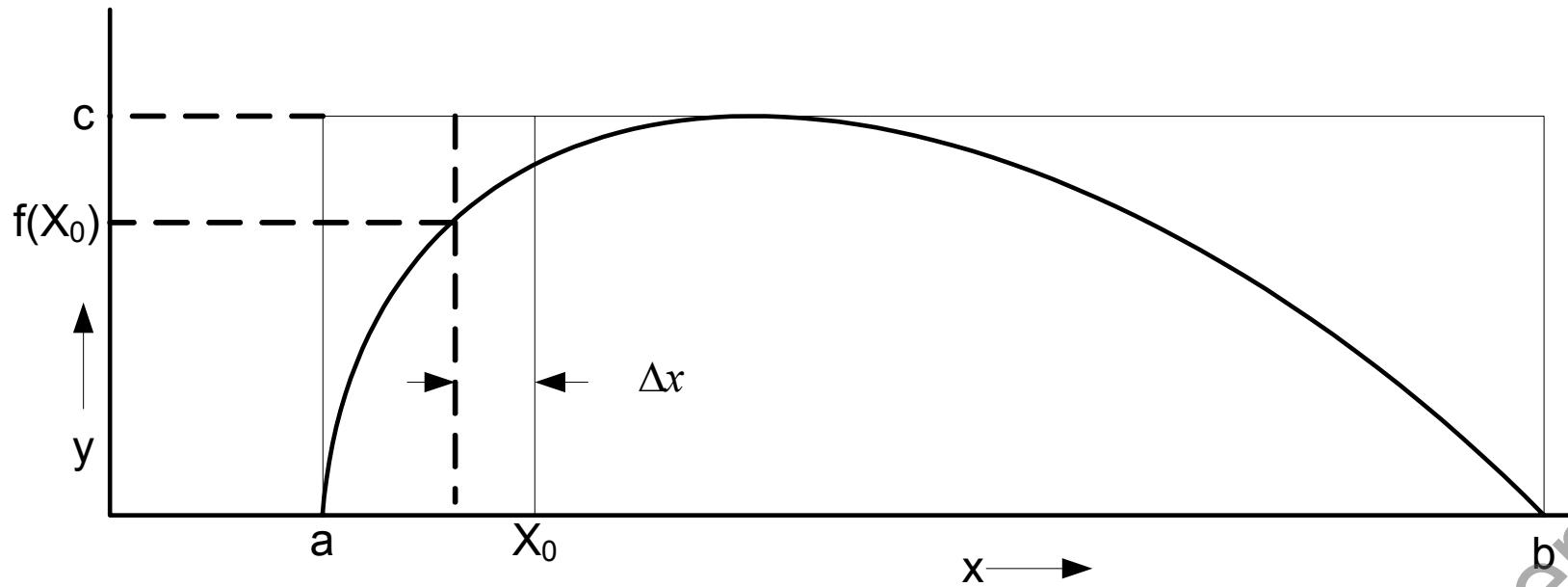
Therefore, is this a valid method?

## Non-uniform continuously distributed RNs

Given,  $X_0$ ,

$X_0$  is accepted if,

$$Y_0 \leq f(X_0)$$



## Non-uniform continuously distributed RNs

Let  $\Delta x$  be small area. If  $Y_0$  still is inside the small rectangle bounded by  $\Delta x$ , under the curve, now output range will be,

$X_0 - \Delta x$  to  $X_0$

In such case,

$$P(X \leq X_0) = P(Y_0 \text{ bounded by left of } X_0)$$

$$F(X_0) = \frac{\int_a^{X_0} f(x)dx}{c(x_0 - a)} \cdot \frac{(x_0 - a)}{(b - a)}$$

Therefore,

$$F(X_0) = \int_a^{X_0} f(x)dx$$

## **Non-uniform continuously distributed RNs**

### Disadvantage

- Two uniform variates must be calculated
- The situation will be even more complex if more iterations need to be performed

Therefore, it is usable only if,

The probability density function is given by a mathematical function

The PDF must be limited i.e.  $F(x) = 0$  for  $x < a$  and  $x > b$ . If not, then use inverse transformation method

## Sample function

```
/* C implementation Park & Miller's random function      */
double random ( seed )
    long int    *seed;
{
    long int  a = 16807,                      /* 7^5 */
              m = 2147483647,                /* 2^31 - 1 */
              q = 127773,                   /* m / a (int divide) */
              r = 2836,                     /* m % a */
              lo, hi, test;
    double dm = 2147483647;

    hi = *seed / q;
    lo = *seed % q;
    test = a * lo - r * hi;
    *seed = test > 0 ? test : test + m;
    return( (double) *seed / dm );
}
```

# Pramod Parajuli Simulation and Modeling, CS-331

---

## Chapter 8 Queueing Systems

(C) Pramod Parajuli, 2004

Source: [www.csitnepal.com](http://www.csitnepal.com)

1  
Csitnepal

# Queueing Systems

A queueing system consists of one or more servers that provide service of some kind to arriving customers

If the servers are busy, the customers join one or more queues in front of the servers, hence queueing system

<b>System</b>	<b>Servers</b>	<b>Customers</b>
Bank	Tellers	Customers
Hospital	Doctors, Nurses, beds	Patients
Computer System	CPU, I/O devices	Jobs
Manufacturing systems	Machines, Workers	Parts
Airport	Runways, Gates	Airplanes, travelers
Communications network	Nodes, links	Messages, Packets

# Queueing Systems

State of the system; the number of units in the system

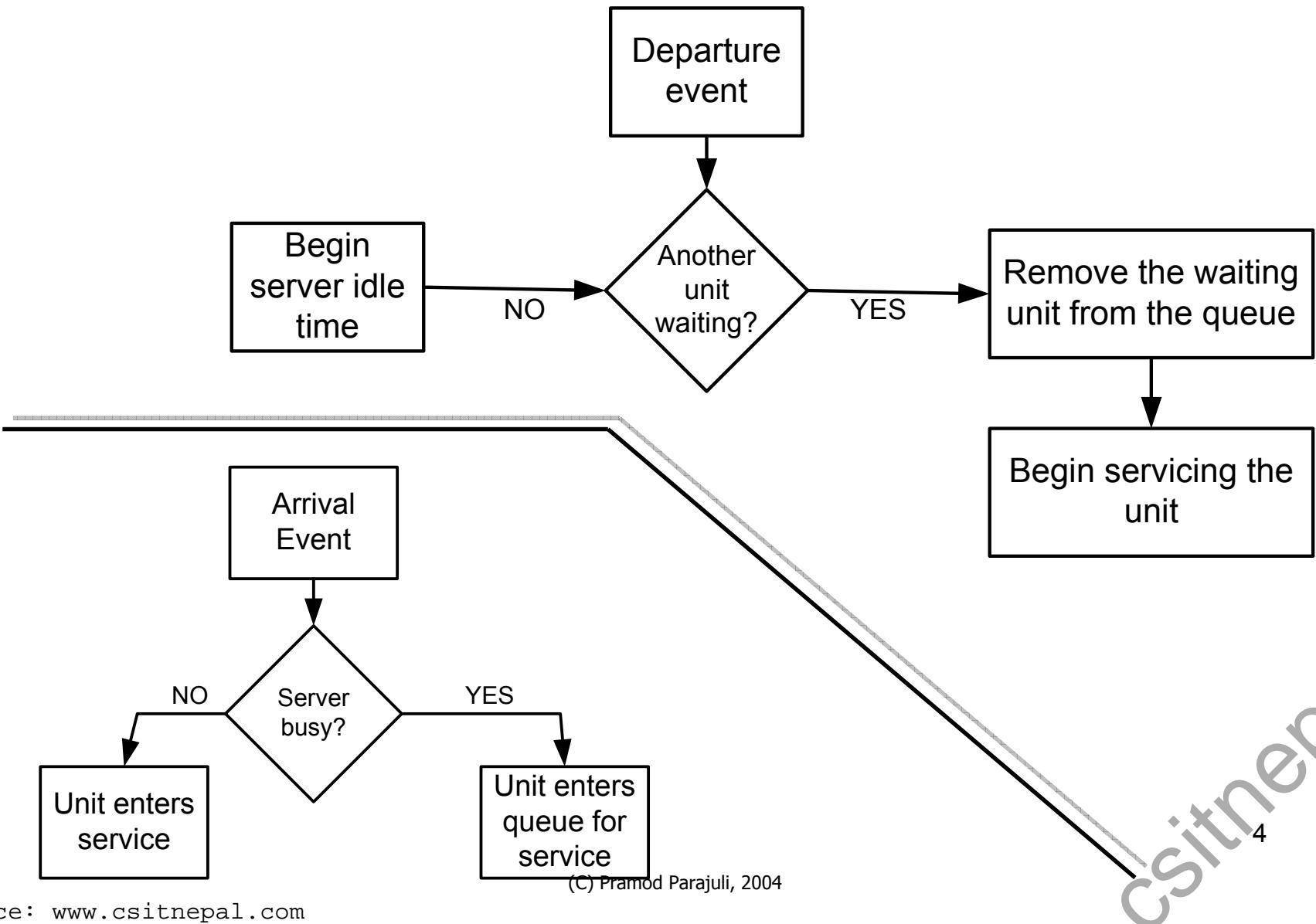
Queues that grow without bound are known as explosive or unstable queues

Queueing system includes; the server, the unit being services (?), and units in the queue

Simulation time is determined by simulation clock

		Queue status	
		Not empty	Empty
Server status	Busy	Enter queue	Enter queue
	Idle	Impossible	Enter service

# Queueing Systems



# **Characteristics of Queueing Systems**

## The calling population

- The population of potential customers
- In systems with a large population of potential customers, the calling population is considered to be infinite
- Although the population is finite, if very small fraction of the potential customers are served by the system, then the population is supposed to be infinite

## System capacity

- Number of customers waiting in the queue is limited

## The arrival process

## Queue behavior and discipline

## Service times and service mechanism

# **Components Queueing Systems**

- Arrival process (pattern)
- Service mechanism
- Queue discipline

## Arrival Process

The simulation time start when first customer arrives to the system

The arrival time is random and hence inter-arrival time is also random

Customer	Inter-arrival time	Arrival time on clock
1	-	0
2	2	2
3	4	6
4	1	7
5	2	9
6	6	15

## Arrival Process

If arrivals vary stochastically, it is necessary to define the probability function of the inter-arrival times

Two or more arrivals may be simultaneous. If n variables are simultaneous, then n-1 of them have zero inter-arrival times

Notation;

$T_a$  Mean inter-arrival time

$\lambda$  Mean arrival time

They are related as;  $\lambda = \frac{1}{T_a}$

(C) Pramod Parajuli, 2004

## Arrival Process

Example - office;

- five days a week
- Eight-hour day
- 800 calls a week
- Model the office using time scale of minutes

$$\begin{aligned}1 \text{ week} &= 40 \text{ working hours. Therefore inter-arrival time} \\&= (40 * 60) / 800 \\&= 3 \text{ minutes}\end{aligned}$$

i.e. 0.333 calls per minute

---

- In infinite population model, the arrival rate is not affected by the number of customers
- If the arrival process is homogenous (no rush-hours) then arrival is linear

## Arrival Process

On the other hand, the arrival time is very much uncertain and hence its distribution is important

Let's define  $A_{0(t)}$  be the arrival distribution to be defined as; *(different writers use different notations)*

It is the probability that an inter-arrival time is greater than 't'

Again, if the cumulative distribution function  $F(t)$  is the probability that an inter-arrival time is less than t, then

$$A_{0(t)} = 1 - F(t)$$

Therefore, takes maximum value of '1' at  $t = 0^{10}$

## Poisson Arrival Patterns

Often used to model arrival processes with constant arrival rates

Gives the number of events that occur in a given period

Probability of an arrival in an interval  $dt$  is proportional to  $dt$

If  $\lambda$  is the mean number of arrivals per unit time, then probability of an arrival in  $dt$  is

$$\lambda \cdot \Delta t$$

Therefore, PDF of inter arrival time is given by;

$$f(t) = \lambda \cdot e^{-\lambda t} \quad (t \geq 0)$$

## Poisson Arrival Patterns

In other words, a probability function is said to have poisson pattern if

$$p(x) = \begin{cases} \frac{e^{-\alpha} \alpha^x}{x!}, & x = 0, 1, \dots \\ 0, & \text{otherwise} \end{cases}$$

- Where  $\alpha$  is the mean rate and must be positive
- $E(X) = V(X) = \alpha$

## Poisson Arrival Patterns

So now, arrival distribution will be;

$$A_0(t) = e^{-\lambda t}$$

$$\begin{aligned} A &= \frac{f(t)}{\lambda} \\ A &= \frac{\lambda \cdot e^{-\lambda t}}{\lambda} \\ A &= e^{-\lambda t} \end{aligned}$$

Here,  $\lambda$  is mean no. of arrivals per unit time and arrivals in a period of time 't' is again a random variable.

So now, probability of 'n' arrivals occurring in a period of length 't' is given by (exponential)

$$p(n) = \frac{(\lambda t)^n e^{-\lambda t}}{n!}$$

# Poisson Arrival Patterns

Example;

Arrival Number	Arrival Time	Inter-arrival Time	Arrival Number	Arrival Time	Inter-arrival time
1	12.5	12.5	11	136.4	21.4
2	15.1	2.6	12	142.7	6.3
3	44.1	29.0	13	151.2	8.5
4	62.6	18.5	14	162.5	11.3
5	65.3	2.7	15	167.2	4.7
6	67.6	2.3	16	172.9	5.7
7	71.0	3.4	17	179.8	6.9
8	92.5	21.5	18	181.6	1.8
9	106.5	14.0	19	185.0	3.4
10	115.0	8.5	20	194.9	9.9

## Poisson Arrival Patterns

Example;

The inter arrival time is ranging from 1.8 to 29.0.

Look at graph plot at page 149 of text book.

Taking the average of the inter arrival time;

9.74

Therefore, estimated arrival rate is =  $\lambda = \frac{1}{T_a}$   
= 0.103

Now, let's group the time period into 10 different range of size 20

## Poisson Arrival Patterns

Example;

Time Period	No. of Arrivals	Time Period	No. of Arrivals
0-20	2	100-120	2
20-40	0	120-140	1
40-60	1	140-160	2
60-80	4	160-180	4
80-100	1	180-200	3

$\lambda = 0.103$ ,  $t = 20$ , and value of  $p(n)$  for  $n = 0, 1, 2, 3, 4, 5$  will be;

# Poisson Arrival Patterns

Example;

No. of arrivals in 20 mins	Actual No. of Occurrences	$p(n)$	Expected number of occurrences
0	1	0.128	1.3
1	3	0.266	2.7
2	3	0.272	2.7
3	1	0.187	1.9
4	2	0.096	1.0
5	0	0.040	0.4

Compare the values in 2<sup>nd</sup> and 4<sup>th</sup> column

# Poisson Arrival Patterns

Example;

Number of people logging onto a computer per minute is Poisson Distributed with mean 0.7

What is the probability that 5 or more people will log onto the computer in the next 10 minutes?

Solution?

Must first convert the mean to the 10 minute period – if mean is 0.7 in 1 minute, it will be  $(0.7)(10) = 7$  in a ten minute period

$$\begin{aligned} P(X \geq 5) &= 1 - P(0) - P(1) - P(2) - P(3) - P(4) \\ &= 1 - F(4) \quad (\text{where } F \text{ is the cdf}) \\ &= 1 - 0.173 \quad (\text{from the table in the text}) \\ &= 0.827 \end{aligned}$$

# The Exponential Distribution

A random variable is said to be exponentially distributed with  $\lambda > 0$  if its pdf is given by

$$f(x) = \begin{cases} \lambda e^{-\lambda x}, & x \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

Since the exponent is negative, the pdf will decrease as  $x$  increases

$\lambda$  is the arrival rate: number of occurrences per time unit

Ex: arrivals per hour; failures per day

Note that  $1/\lambda$  can thus be considered to be the time between events or the duration of events

Ex: 10 arrivals per hour  $\rightarrow 1/10$  hr (6min) average between arrivals

Ex: 20 customers served per hour  $\rightarrow 1/20$  hr (3min) average service time

## The Exponential Distribution

The CDF of exponential distribution is given by;

$$y = 1 - e^{-\lambda t} \quad \text{after solving; } \lambda t = -\ln(1 - y)$$

Since  $y$  is representing cumulative distribution, the term  $(1-y)$  gives values between 0 and 1

Within the range of 0-1, the natural logarithm gives –ve value

If numbers  $y$  are uniformly distributed, the numbers  $1-y$  are also uniformly distributed

$$t = \frac{-\ln(y)}{\lambda} = -T_a \ln(y)$$

It shows that exponential distribution is dependent on mean value

## The Exponential Distribution

The significance about  $T_a$  is that it is in multiplicative form. Numbers with any mean can be derived from a generator of mean value 1 by multiplying its output by the required mean

$$E(X) = \frac{1}{\lambda} \quad V(X) = \frac{1}{\lambda^2}$$

$$F(x) = \begin{cases} 0, & x < 0 \\ \int_0^x \lambda e^{-\lambda t} dt = 1 - e^{-\lambda x}, & x \geq 0 \end{cases}$$

# The Exponential Distribution

## Example – Generation;

Let's consider we have 5 random numbers 'y' in the range (0,1). The numbers are 0.135, 0.639, 0.424, 0.010, 0.843

Given  $\lambda = 3.8$

$$1/\lambda = 1/3.8$$

The numbers we get are;

$$0.526, 0.118, 0.226, 1.213, 0.045$$

## Coefficient of Variation

Applies to arrival times

Ratio of standard deviation of the inter-arrival time  
to the mean arrival time

Zero value means no variation. If the coefficient  
increases the data become more dispersed

If exponential distribution of mean value is  $T$  and  
standard deviation is also  $T$ , then variance is 1  
i.e. if the variance is close to 1, then the  
exponential distribution may fit the data. In  
such case, chi-square test should be carried out  
to test the level of significance

## Erlang Distribution

- Originally derived to analyze telephone traffic
- PDF of such distribution is given by;

$$f(t) = (k\lambda)^k \left[ \frac{e^{-k\lambda t}}{(k-1)!} \right] t^{k-1}$$

- And arrival distribution will be;

$$A_0(t) = e^{-k\lambda t} \cdot \sum_{n=0}^{k-1} \frac{(k\lambda t)^n}{n!}$$

$$F(x) = \begin{cases} 1 - \sum_{i=0}^{k-1} \frac{e^{-k\theta x} (k\theta x)^i}{i!}, & x > 0 \\ 0, & x \leq 0 \end{cases}$$

Where k is a positive integer

## Erlang Distribution

Setting  $k=1$ , gives exponential distribution.

Standard deviation of Erlang Distribution =  $T/k^{1/2}$

Therefore coefficient of variation is =  $1/k^{1/2}$

If  $k > 1$ , the data cluster around the mean value

If  $k = \infty$ , the distribution will be step function

If variation coefficient of variation is significantly less than 1,  
it is desirable to represent the data using Erlang  
distribution for which 'k' is closest to the value  $(T/s)^2$

In general terms, there are 'k' stages that an entity have to  
pass and total time required is 'T' then,

$$E(x) = \frac{1}{\lambda} + \frac{1}{\lambda} + \frac{1}{\lambda} + \frac{1}{\lambda} + \dots + \frac{1}{\lambda}$$

k-terms

## Erlang Distribution

If  $\beta$  is average time delay of each 'k' constituent exponential distributions, then  $k\beta$  is the average value of the Erlang distribution.

In this case;

$$f(t) = \frac{1}{(k-1)!} \cdot \left(\frac{1}{\beta}\right)^k \cdot t^{k-1} \cdot e^{-t/\beta}$$

If  $k = 1$ , then

$$f(t) = \frac{1}{\beta} \cdot e^{-t/\beta}$$

i.e. exponential distribution

# Erlang Distribution

## Example - Generation;

- Generate 'k' random samples from an exponential distribution with mean time of  $\beta$
- Add all of the terms

```
float prod = 1.0;
for(counter = 0; counter < k; counter++) {
    rn = generateRandom(seed);
    prod *= rn;
}
return -beta * ln(prod);
```

# Erlang Distribution

## Exercise 5.21

Time to serve a customer at a bank is exponentially distributed with mean 50 sec

1. Probability that two customers in a row will each require less than 1 minute for their transaction
2. Probability that the two customers together will require less than 2 minutes for their transactions

For 1) we are looking at the probability that each of two independent events will be < 1 minute

In this case, the probability overall is the product of the two probabilities, each of which is exponential

$$\lambda = \text{rate} = 1/\text{mean} = 1/50$$

$$\text{We want } P(X < 60) = F(60) = 1 - e^{-(1/50)(60)} = 0.6988$$

$$\text{Thus the total probability is } (0.6988)^2 = 0.4883$$

## Erlang Distribution

For 2) we are looking at the probability that two events together will be < 2 minutes

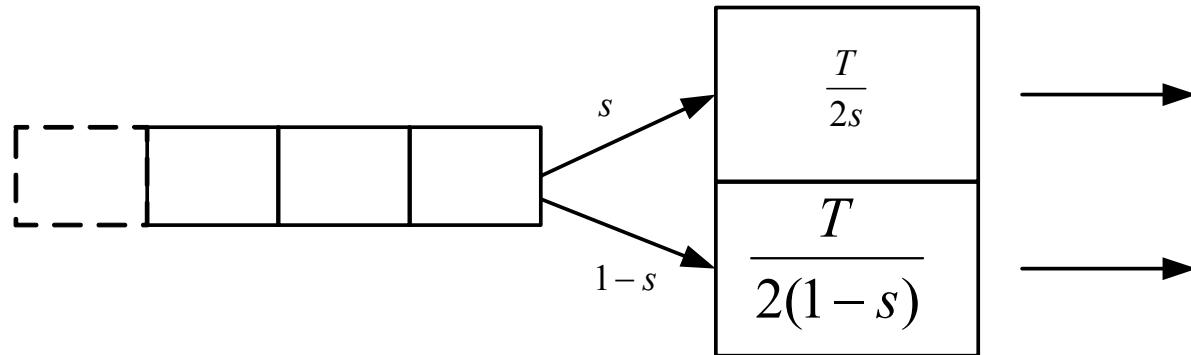
In this case the probability overall is an Erlang distribution with  $k = 2$  and  $k\theta = 1/50$ , and we want to determine  $P(X < 120) = F(120)$

Substituting into our equation for  $F$ , we get

$$\begin{aligned}F(120) &= 1 - \sum_{i=0}^{2-1} \frac{e^{-(120/50)} (120/50)^i}{i!} \\&= 1 - (0.0907) - (0.2177) = 0.6916\end{aligned}$$

# Hyper Exponential Distribution

If a set of data have two different exponential patterns, then it known as hyper-exponential distribution



Here, mean service time for first server =  $T/2s$

mean service time for second server =  $T/2(1-s)$

$$0 < s \leq \frac{1}{2}$$

And therefore, the arrival pattern

$$A_0(t) = se^{-2s\lambda t} + (1-s)e^{-2(1-s)\lambda t}$$

# Hyper Exponential Distribution

The variance

$$k = \frac{(1 - 2s + 2s^2)}{2s(1 - s)}$$

when  $s = 1/2$ , k becomes 1 and hence gives exponential distribution

Steps;

- Generate exponentially distributed rn from uniformly distributed random number with mean value of 1
- Compare the second uniformly distributed random number to 's'
- If less than s, multiply by T/2s
- Else multiply by T/2(1-s)

## Service Times

Normally, service time of a process is constant  
But if it varies stochastically, it is described by a probability function

$T_s$  = Mean service time

$\mu$  = mean service rate

$S_0(t)$  = Probability that service time > t

Normally, exponential distribution is used to represent the service time but if the service time is fluctuating by large variance, it is represented by normal distribution

## Service Times

Articulated by specifying the number of servers (denoted by  $s$ ) whether each sever has its own queue or there is one queue feeding all servers and probability distribution of customers' service times

If  $S_i$  is the service time of the  $i^{\text{th}}$  arriving customer,  $E(S)$  is the mean service time of the customer then service rate of the server is

$$= 1/E(S)$$

Example : Class demonstration (from page 210 of Banks, Carson, Nelson, Nicol)

## Normal Distribution

A random variable X with mean  $\mu$  ( $-\infty < \mu < \infty$ ) and variance  $\sigma^2 > 0$  has a normal distribution if it has the pdf

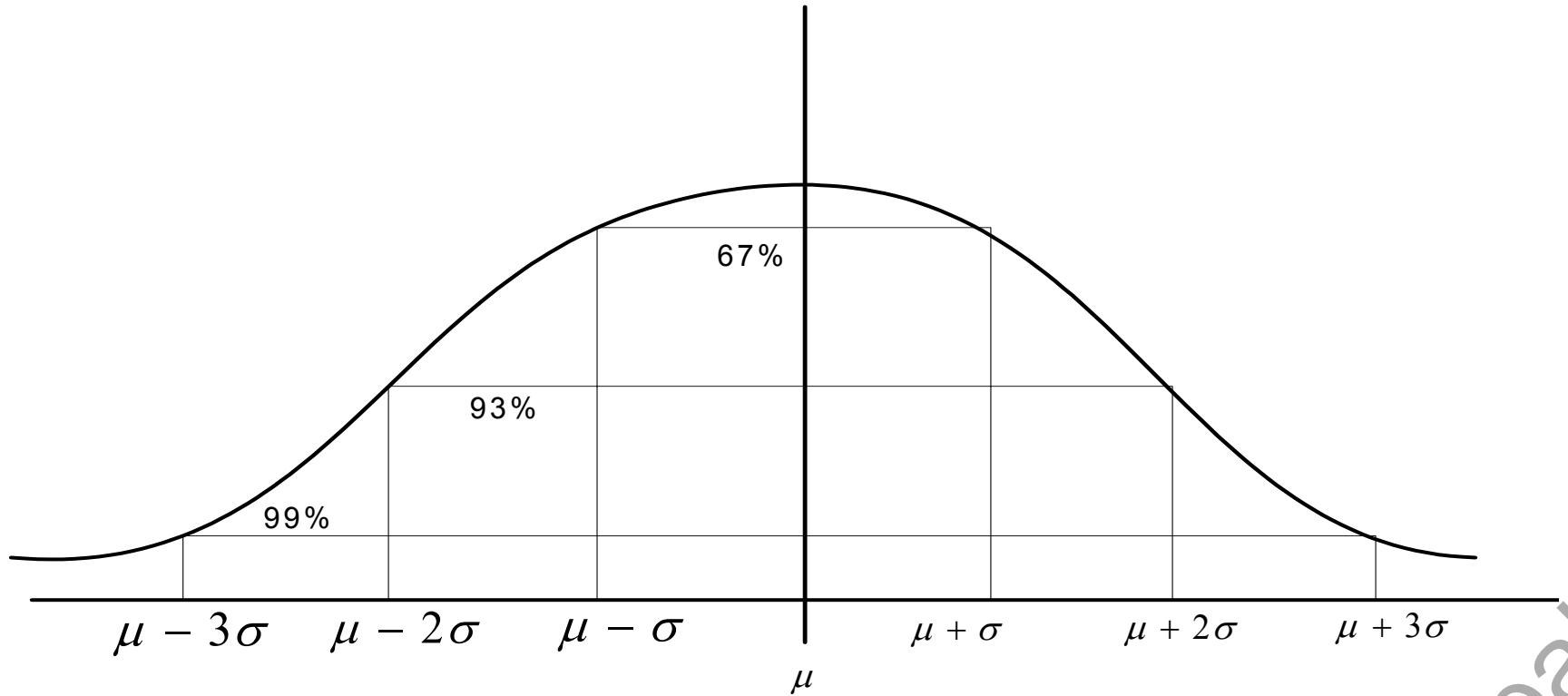
$$f(x) = \frac{1}{\sigma\sqrt{2\Pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}, -\infty < x < \infty$$

In normal form

$$f(z) = \frac{1}{\sqrt{2\pi}} \cdot e^{-z^2/2}$$

# Normal Distribution

-Property;



# Normal Distribution

To generate numbers distributed according to  $f(z)$ , we use

$$x = z \cdot \sigma + \mu, \quad z = \text{standard normal variate}$$

If  $r_1$  and  $r_2$  are two uniform random numbers in the range  $(0, 1)$ , then to generate random samples from a standardized normal distribution is to use 'Box Mueller Transformation':

$$z = (-2 \cdot \ln r_1)^{1/2} \cdot \cos(2\pi r_2)$$

For a set of several uniformly distributed random numbers,

standard deviation of this equation reaches normal distribution as  $k$  approaches infinity

$$y = \frac{\sum_{i=1}^k x_i - \frac{k}{2}}{\left(\frac{k}{12}\right)^{1/2}}$$

$x_i$  = uniformly distributed random numbers  
 $k$  = small values of  $k$  give good accuracy (normally 12)

$$y = \sum_{i=1}^{12} x_i - 6.0$$

## Normal Distribution

The random numbers  $x_i$  are generated using inverse transformation method

A more accurate method can be;

Given two independent uniform variates, the method generates two outputs which are independent and is from a normal distribution with a mean of 0 and deviation of 1. It provides more accuracy than the previous one

$$X_1 = (-2 \ln U_1)^{1/2} \cdot \cos 2\pi U_2$$

$$X_2 = (-2 \ln U_1)^{1/2} \cdot \sin 2\pi U_2$$

In this case, the numbers X1 and X2 will be IID (independent and identically distributed)

# Queueing Notation

Standard notation

A / B / c / N / K

A – represents interarrival-time distribution

B – represents the service-time distribution

A and B can follow any distribution

D : Constant or deterministic

M : Exponential or Markov

E<sub>k</sub> : Erlang of order k

G : Arbitrary or General

H : Hyper exponential

GI : General independent

## Queueing Notation

c – represents the number of parallel servers

N – represents the system capacity

Could be "infinite" if the queue can grow arbitrarily large (or at least larger than is ever necessary)

Ex: a queue to go up the Eiffel Tower

Space could be limited in the system

Ex: a bank or any building

This can affect the effective arrival rate, since some arrivals may have to be discarded

K – represents the size of the calling population

- How large is the pool of customers for the system?
- It could be some relatively small, fixed size

Ex: The computers in a lab that may require service

## Queueing Notation

It could be very large (effectively infinite)

Ex: The cars coming upon a toll booth

The size of the calling population has an important effect on the arrival rate

If the calling population is infinite, customers that are removed from the population and enter the queueing system do not affect the arrival rate of future customers ( $\infty - 1 = \infty$ )

If the calling population is finite, removal of customers from the population (and putting them into and later out of the system) must affect future arrival rates

Ex: In a computer lab with 10 computers, each has a 10% chance of going down in a given day. If a computer goes down, the repair takes a mean of 2 days

## Queueing Notation

In the first day, the expected number of failures is  $10 * 0.1 = 1$   
However, once a failure occurs, the faulty computer is out of  
the calling population, so the expected number of  
failures in the next day is  $9 * 0.1 = 0.9$

Clearly this changes again if another computer fails

# Queueing Notation

Example;

1. there are 's' servers in parallel and one FIFO queue feeding all servers
2.  $A_1, A_2, \dots$  are IID random variables
3.  $S_1, S_2, \dots$  are IID random variables
4.  $A_i$ 's and  $S_i$ 's are independent

Such system is represented by GI/G/s queue

GI – distribution of  $A_i$

G – distribution of  $S_i$

M/M/1/ $\infty/\infty$

Here, the interarrival time and service time are exponentially distributed, the server is of single-server, the system have infinite capacity and infinite population of potential arrivals

This kind of server often represented by M/M/1

# Queueing Disciplines

The queueing discipline describes in which order next entity is selected from the queue for service

- FIFO – first in first out
- LIFO – Last in first out
- SIRO – service in random order
- SPT – shortest processing time first
- PR – service according to priority

The order of picking entities from the queue does not mean that the entities will leave the system in the same order. This is all because of random service time for each entity

# Queueing Disciplines

Sometimes, the entity may leave the queue even before being served. Such process is known as '**reneging**'.  
The rules for reneging must be defined clearly

If an entity refuses to join the queue (because the queue is relatively long), the process is known as '**balking**'

If there are multiple lines i.e. queues but a single server, the entities are selected by the process of polling

In case of priority queue, if the recently arrived entity have the greatest priority, then the new arrival will interrupt or preempt the service

## Measure of Performance for Queues

Let  $T_a$  be mean arrival time,  $T_s$  be mean service time,  $\lambda$  be arrival rate, and  $\mu$  be service rate then;

The ratio of mean service time to the mean inter-arrival time is called the ***traffic intensity (u)***

$$u = T_s / T_a$$

Let's denote rate of all of the arrivals (including losses) by  $\lambda'$  and  $\lambda$  to denote arrival rate of entities that are served.  
In this case;

$$u = \lambda' \cdot T_s$$

And ***server utilization ( $\rho$ )*** is defined as;

$$\rho = \lambda' \cdot T_s = \lambda / \mu$$

If 'n' number of servers;

$$\rho = \lambda / (n \cdot \mu)$$

# Measure of Performance for Queue

## Notations for parallel server systems

$P_n$	steady state probability of having 'n' customers in system
$P_n(t)$	Probability of 'n' customers in system at time 't'
$\lambda$	Arrival rate
$\lambda_e$	Effective arrival rate
$\mu$	Service rate of one server
$\rho$	Server utilization
$A_n$	Inter-arrival time between customers 'n-1' and 'n'
$S_n$	Service time of the nth arriving customer

## Measure of Performance for Queue

### Notations for parallel server systems

$W_n$  Total time spent in system by the nth arriving customer

or  $W$  Mean time to complete service (including the wait for service)

$W_n^Q$  Total time spent in the waiting line by customer 'n'

$L(t)$  The no. of customers in system at time 't'

or  $W_w$  Mean time spent waiting for service to begin

$L_Q(t)$  The no. of customers in queue at time 't'

$L$  Long-run time-average no. of customers in system (Mean no. of entities in the system)

## Measure of Performance for Queue

Notations for parallel server systems

$L_Q$  Long-run time-average no. of customers in queue  
or  $L_w$  Mean number of entities in the waiting line

$W$  Long-run average time spent in system per customer

$w_Q$  Long-run average time spent in queue per customer

The probability that an entity will have to wait more than a given time is known as ***delay distribution***

$P(t)$ - Probability that the time to complete the service is greater than 't'

$P_w(t)$ -Probability that the time spent waiting for service to begin is greater than 't'

# Mathematical Solutions of Queuing Problems

Some times, the queuing problems can be formulated by some mathematical solutions

Single server, poisson arrival

The mean number of entities in the system when  $\rho < 1$  are;

$$L = \frac{\rho(2-\rho)}{2(1-\rho)}$$

Constant

$$L = \frac{\rho}{1-\rho}$$

Exponential

$$L = \frac{2k\rho - \rho^2(k-1)}{2k(1-\rho)}$$

Erlang

$$L = \frac{\rho^2 + \rho(1-\rho)4s(1-s)}{4s(1-s)(1-\rho)}$$

Hyper-Exponential

(C) Pramod Parajuli, 2004

49

## Mathematical Solutions of Queuing Problems

In above cases, the total number of entities waiting for service ( $L_w$ ), mean times spent waiting for service to begin ( $W$ ), and mean times spent waiting for service to be completed ( $W_w$ ) can be derived as;

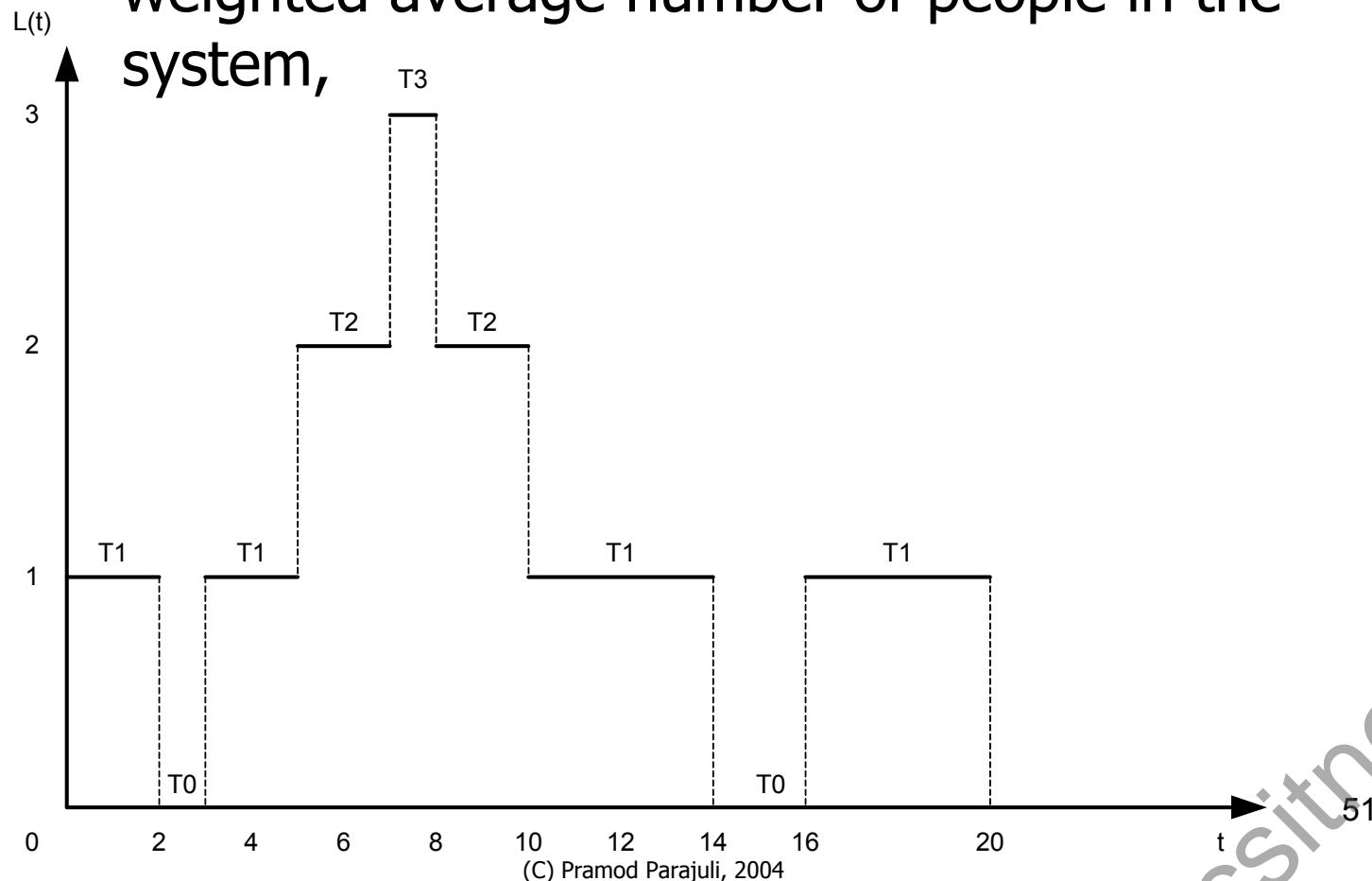
$$L_w = L - \rho$$

$$W = \frac{L}{\lambda} = L.T_a$$

$$W_w = \frac{L_w}{\lambda} = L_w T_a$$

## Time-Average Number in the System, L

Given a queueing system operating for some period of time, T, we'd like to determine the time-weighted average number of people in the system,



## Time-Average Number in the System, $\hat{L}$

We'd also like to know the time-weighted average number of people in the queue,

Note that for a single queue with a single server, if the system is always busy,

However, it is not the case when the server is idle part of the time

The "hats" indicate that the values are "estimators" rather than analytically derived long-term values

## Time-Average Number in the System, $\hat{L}$

We can calculate  $\hat{L}$  for an interval  $[0, T]$  in a fairly straightforward manner using a sum

$$\hat{L} = \frac{1}{T} \sum_{i=0}^{\infty} i T_i$$

Note that each  $T_i$  here represents the total time that the system contained exactly  $i$  customers

These may not be contiguous

'i' is shown going to infinity, but in reality most queuing systems (especially stable queuing systems) will have all  $T_i = 0$  for  $i >$  some value

In other words, there is some maximum number in the system that is never exceeded

## Time-Average Number in the System, L

Let's think of this value in another way:

Consider the number of customers in the system at any time, t

$L(t)$  = number of customers in system at time t

This value changes as customers enter and leave the system

We can graph this with t as the x-axis and  $L(t)$  as the y-axis

Consider now the area under this plot from  $[0, T]$

It represents the sum of all of the customers in the system over all times from  $[0, T]$ , which can be determined with an integral

$$Area = \int_0^T L(t) dt$$

## Time-Average Number in the System, $L$

Now to get the time-average we just divide by  $T$ ,  
or

$$\hat{L} = \frac{1}{T} \sum_{i=0}^{\infty} iT_i = \frac{1}{T} \int_0^T L(t) dt$$

For many stable systems, as  $T \rightarrow \infty$  (or, practically speaking, as  $T$  gets very large)  $\hat{L}$  approaches  $L$ , the long-run time-average number of customers in the system

However, initial conditions may determine how large  $T$  must be before the long-run average is reached

The same principles can be applied to  $\hat{L}_Q$ , the time-average number in the queue, and  $L_Q$ , the long-run time average number in the queue

## Time-Average Number in the System, L

From fig. in slide no. 51,

$$\begin{aligned}\hat{L} &= [0(3) + 1(12) + 2(4) + 3(1)] / 20 \\ &= 23/20 \\ &= 1.15 \text{ customers}\end{aligned}$$

## Average Time in System Per Customer, $\hat{w}$

This is also a straightforward calculation during our simulations

$$\hat{w} = \frac{1}{N} \sum_{i=1}^N W_i$$

where  $N$  is the number of arrivals in the period  $[0, T]$

where each  $W_i$  is the time customer  $i$  spends in the system during the period  $[0, T]$

If the system is stable, as  $N \rightarrow \infty$ ,  $\hat{w} \rightarrow w$

$w$  is the long-run average system time

We can do similar calculations for the queue alone to get the values  $\hat{w}_Q$  and  $w_Q$

We can think of these values as the observed delay and the long-run average delay per customer

*Example: from 216 - Nicol*

## Average Time in System Per Customer, $w$

For simple queueing systems such as those we have been examining, stability can be determined in a fairly easy way

The arrival rate,  $\lambda$ , must be less than the service rate i.e. customers must arrive with less frequency than they can be served

Consider a simple single queue system with a single server  
 $(G/G/1/\infty/\infty)$

- Define the service rate to be  $\mu$
- This system is stable if  $\lambda < \mu$

If  $\lambda > \mu$ , then, over a period of time there is a net rate of increase in the system of  $\lambda - \mu$

This will lead to increase in the number in the system ( $L(t)$ ) without bound as  $t$  increases

## Average Time in System Per Customer, $w$

Note if  $\lambda == \mu$  some systems (ex: deterministic) may be stable while others may not be

If we have a system with multiple servers (ex:  $G/G/c/\infty/\infty$ ) then it will be stable if the net service rate of all servers together is greater than the arrival rate

If all servers have the same rate  $\mu$ , then the system is stable if  $\lambda < c\mu$

## Average Time in System Per Customer, $w$

Note that if our system capacity or calling population (or both) are fixed, our system can be stable even if the arrival rate exceeds the service rate

Ex: G/G/c/k/ $\infty$

With a fixed system capacity, in a sense the system is unstable until it "fills" up to k. At this point, excess arrivals are not allowed into the system, so it is stable from that point on

The idea here is that the net arrival rate decreases once the system has filled

Ex: G/G/c/ $\infty$ /k

With a fixed calling population, we are in effect restricting the arrival rate

As arrivals occur, the arrival rate decreases and the system again becomes stable

## **Conservation Law**

An important law in queueing theory states

$$L = \lambda w$$

where  $L$  is the long-run number in the system,  $\lambda$  is the arrival rate and  $w$  is the long-run time in the system

Often called "Little's Equation"

This holds for most queueing systems

## Server Utilization

What fraction of the time is the server busy?

Clearly it is related to the other measures we have discussed (we will see the relationship shortly)

As with our other measures we can calculate this for a given system (G/G/1/ $\infty/\infty$ )

$$\hat{\rho} = \frac{1}{T} \sum_{i=1}^{\infty} T_i$$

We assume that if at least 1 customer is in the system, the server will be busy (which is why we start at  $T_1$  rather than  $T_0$ )

However, we can also calculate the server utilization based on the arrival and service rates

# Server Utilization

G/G/1/ $\infty/\infty$  systems

Consider again the arrival rate  $\lambda$  and the service rate  $\mu$

Consider only the server (without the queue)

With a single server, it can be either busy or idle

If it is busy, there is 1 customer in the "server system", otherwise there are 0 customers in the "server system" (excluding the queue)

Thus we can define  $L_s = \rho$  = average number of customers in the "server system"

Using the conservation equation this gives us

$$L_s = \lambda_s w_s$$

where  $\lambda_s$  is the rate of customers coming into the server and  $w_s$  is the average time spent in the server

## Server Utilization

For the system to be stable,  $\lambda_s = \lambda$ , since we cannot serve faster than customers arrive and if we serve more slowly the line will grow indefinitely

The average time spent in the server is simply  $1/\mu$  (i.e.  $1/(rate\ of\ the\ server)$ )

Putting these together gives us

$$\rho = L_s = \lambda_s W_s = \lambda (1/\mu) = \lambda/\mu$$

Note that this indicates that a stable queueing system must have a server utilization of less than 1

The closer we get to one, the less idle time for the server, but the longer the lines will be (probably)

Actual line length depends a lot not just on the rates, but also on the variance

## Server Utilization

### G/G/c/ $\infty$ / $\infty$ systems

Applying the same techniques we did for the single server, recalling that for a stable system with  $c$  servers:

$$\lambda < c\mu$$

we end up with the the final result

$$\rho = \lambda/c\mu$$

# Pramod Parajuli Simulation and Modeling, CS-331

---

## Chapter 9 Discrete System Simulation

(C) Pramod Parajuli, 2005

Source: [www.csitnepal.com](http://www.csitnepal.com)

1  
Csitnepal

# Introduction

## Event

Change in the state of the system

## Discrete Systems

- Systems having discrete nature
- Can be of two types
  1. Fixed time step – a timer or clock exists
  2. Event-to-event (discrete event simulation) – the event works as the clock i.e. self clocking  
(figure 1.2, Law & Kelton, page 9)

The state of the system is not altered while the transition is being occurred.

## Clock time

Number recorded time

# Introduction

## Attributes

Properties of a given entity

## Event notice

A record of an event to occur at the current or some future time, along with any associated data necessary to execute the event

## Event list

A list of event notices for future events, ordered by time of occurrence; also known as the future event list (FEL)

## Delay (conditional wait)

A duration of time of unspecified indefinite length which is not known until it ends.

# Introduction

## Activity (unconditional wait)

A duration of time of specified length which is known when it begins

Completion of an activity is an event (primary event)

Completion of a delay is conditional/secondary event.

Such events are not represented by event notices nor do they appear on the FEL

## Duration of activity

- Deterministic (given exact duration)
- Statistical (random draw from specified set with equal probability of picking an element)
- Functional (defined by a function of any kind)

# **Introduction**

**Significant event simulation**

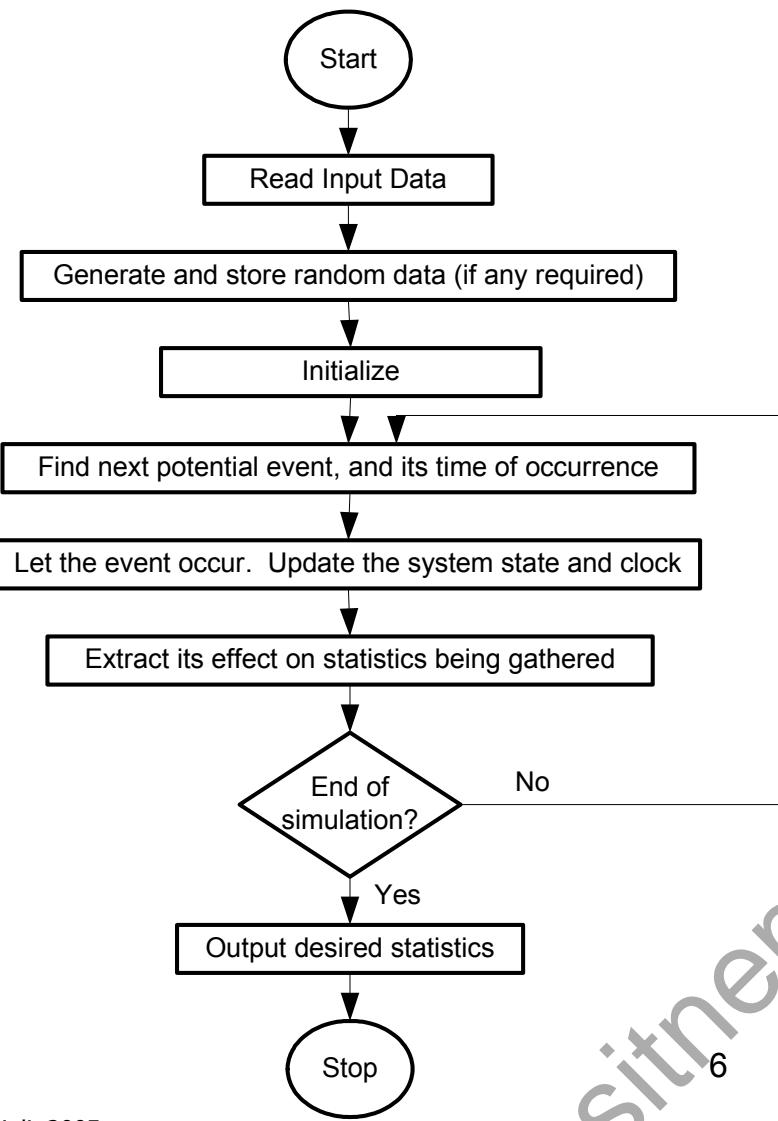
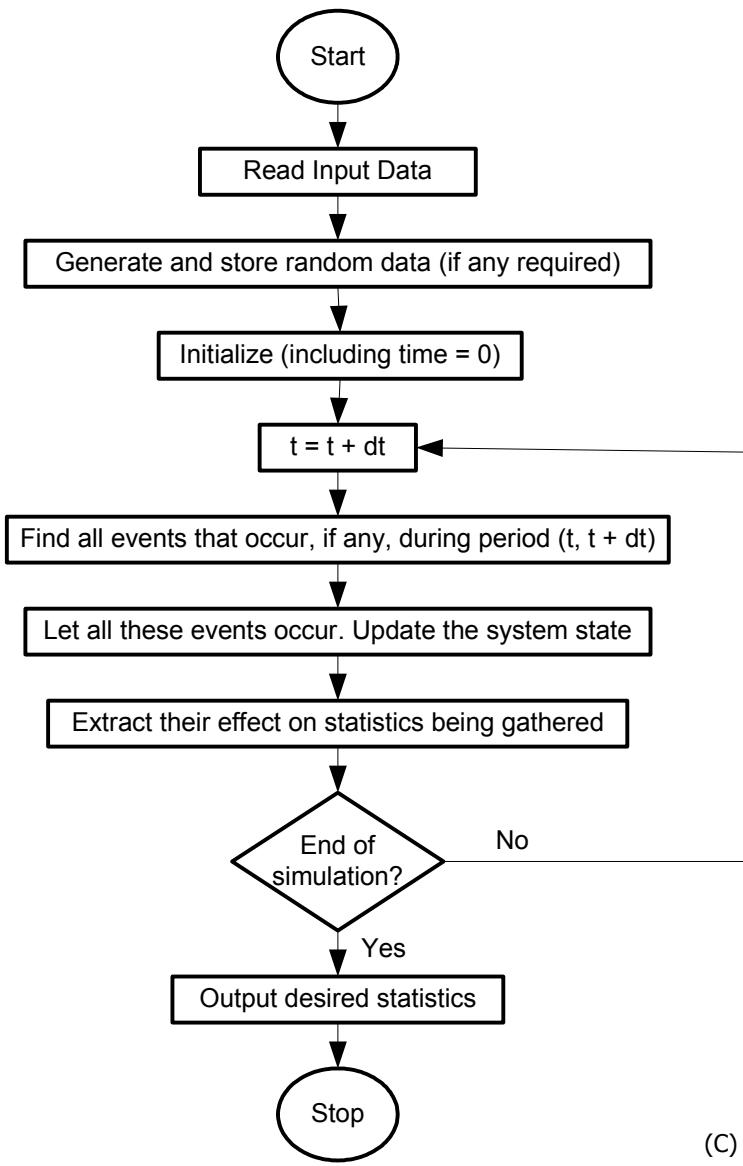
Continuous systems with quiescent periods (interval between events)

Low order polynomials are used to represent the quiescent periods

# Fixed time-step

vs.

# Next-event



(C) Pramod Parajuli, 2005

# **Components of next-event time-advance approach**

## **System state**

The collection of state variables necessary to describe the system at a particular time

## **Simulation clock**

A variable giving the current value of simulated time

## **Event list**

A list containing the next time when each type of event will occur

## **Statistical counters**

Variables used for storing statistical information about system performance

## **Initialization routine**

A subprogram to init the simulation model at time 0<sup>7</sup>

# **Components of next-event time-advance approach**

## **Timing routine**

A subprogram that determines the next event from the event list and then advances the simulation clock to the time when that event is to occur

## **Event routine**

A subprogram that updates the system state when a particular type of event occurs i.e. one event routine for each event type

## **Library routines**

A set of subprograms used to generate random observations from probability distributions that were determined as part of the simulation model

# **Components of next-event time-advance approach**

## Report generator

A subprogram that computes estimates (from the statistical counters) of the desired measures of performance and produces a report when the simulation ends

## Main program

A subprogram that invokes the timing routine to determine the next event and then transfers control to the corresponding event routine to update the system state appropriately.

## Generation of Arrival Patterns

- Generation of exogenous arrivals
- Trace driven simulation - the sequence of inputs may also be generated from observations on a system
- If there is no interaction between the exogenous arrivals and the endogenous events of the system, then the creation of a sequence of arrivals in preparation for the simulation is permissible otherwise arrivals are generated as required
- Arrival of exogenous entity is an event and the arrival time of the next entity is recorded as one of the event times
- The event of entering the entity into the system is executed and the arrival time of next entity is calculated from inter-arrival distribution

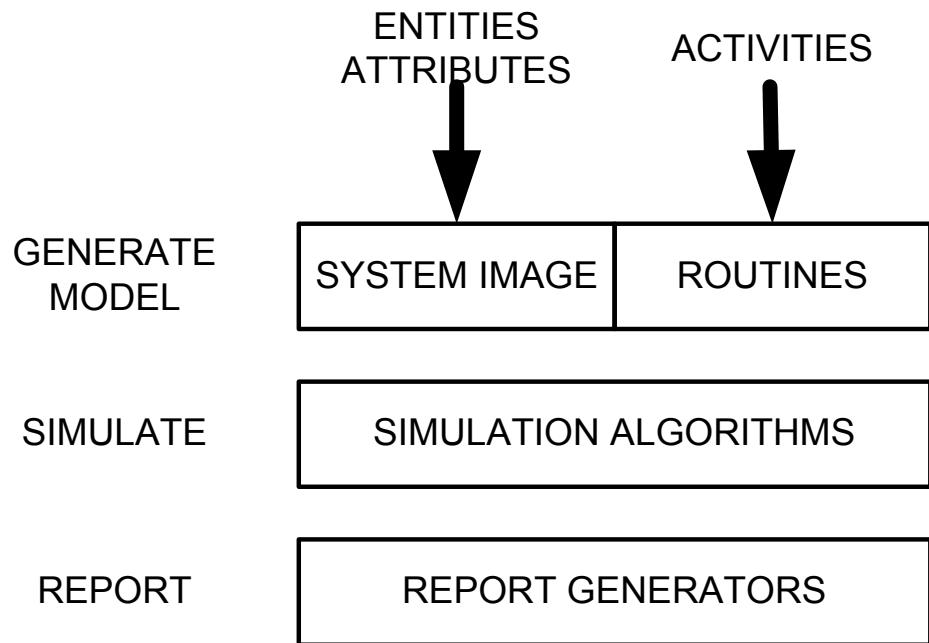
## Generation of Arrival Patterns

- **Bootstrapping** – the previous process of counting the arrival of next entity as current entity is executed is known as bootstrapping. It is a process of making one entity create its successor.
- In bootstrapping, the simulation program have to keep record of arrival of next entity only. Therefore, it is the most preferred method of generating arrivals.
- The attributes for arriving entity can be assigned at the generation time or later on depending on the requirement.
- If the attributes do not affect the event execution, then the attributes can be assigned later on.
- But if attributes do affect the event execution, they must be assigned at the arrival time and retained until the event execution is finished.

# Simulation Programming Tasks

- 3 basic tasks

1. Generate model
2. Simulate
3. Report



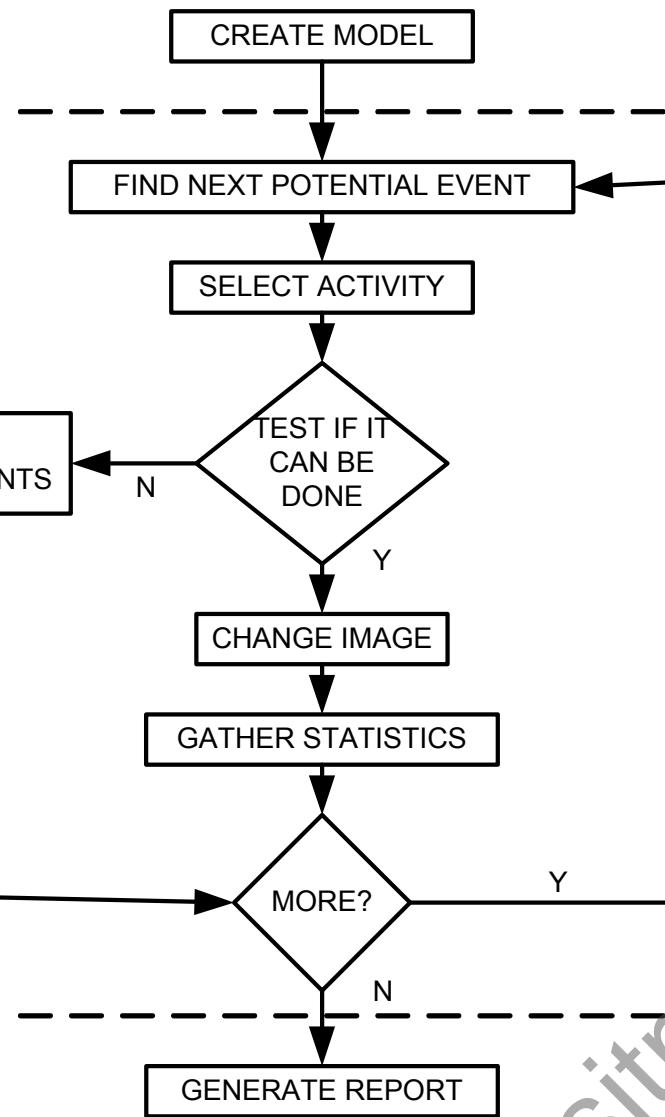
System image – Set of numbers created to represent the state of the system

Simulation algorithms – procedures that execute the cycle of actions

# Simulation Algorithm

## Steps

1. *Find* the next potential event
2. *Select* an activity
3. *Test* if the event can be executed
  - If direct measure, either success or abandon
  - If the event can be executed later, then it is known as conditional event
4. *Change* the system image
5. *Gather* statistics



# Gathering Statistics

Commonly used statistics parameters (included in report);

1. *Counts* – no. of entities of particular type or no. of times some event occurred
2. *Summary measures* – extreme values, mean values, standard deviations
3. *Utilization* – fraction of time some entity is engaged
4. *Occupancy* – fraction of a group of entities in use on the average
5. *Distributions* – queue lengths, waiting times etc.
6. *Transit times* – time taken for an entity to move from one part of the system to some other part

# Counters and Summary Statistics

- Counters used to accumulate totals, some level etc.
- Mean of N observations  $x_i$ ,

$$mean = \frac{1}{N} \sum_{i=1}^N x_i$$

- Standard deviation

$$s = \left[ \frac{1}{(N-1)} \sum_{i=1}^N (m - x_i)^2 \right]^{1/2}$$

# Measuring Utilization and Occupancy

Measuring the load on some entity is most common requirement of a simulation

What fraction of the time the item was engaged during the simulation run?

Measured by *utilization* factor

Let time  $t_b$  be the time the item becomes busy, and  $t_f$  be the time when the item becomes free

(fig. 8-11) Gordon

The interval  $t_f - t_b$  is added to a counter. At the end, the utilization is derived as:

$$U = \frac{1}{T} \cdot \sum_{r=1}^N (t_f - t_b)_r$$

(C) Pramod Parajuli, 2005

## Measuring Utilization and Occupancy

A discrete simulation program, updating time as events occur, will measure the intervals  $t_f - t_b$  directly

A continuous simulation program updating time in small intervals need to build up the count by counting the number of intervals in which the item is busy. For example, if for 'x' number of counts item was busy then the fact must be preserved.

If the items was busy at the end of the simulation, then add a count according to the amount of time left in the current count as the trail of previous run

Same holds for if the item was busy at the beginning of the simulation run

For a group of such items; 
$$A = \frac{1}{T} \cdot \sum_{r=1}^N n_r \cdot (t_{r+1} - t_r)$$

## Measuring Utilization and Occupancy

If there is an upper limit on the number of entities, then the occupancy indicates the average number in use as a ration to the maximum

If 'M' is the maximum number of entities, and the quantity  $n_r$  is the number busy in the interval  $t_r$  to  $t_{r+1}$ , the average occupancy, assuming the number  $n$ , changes N times is;

$$= \frac{1}{NM} \sum_{r=1}^N n_r (t_{r+1} - t_r)$$

Utilization – timing information for each individual entity

Occupancy – count of a class of entities and last time the count changed

For large number of active entities, utilization cost more (time and space) as compared with occupancy

## Recording Distributions and Transit Times

To determine the distribution of a variable, we need to count the no. of times the value of the variable falls within specific intervals

For each new value, it is compared with predefined intervals and for that interval, counted up once

All information is kept in a table that requires;

1. The lower limit of tabulation (L)
2. The interval size ( $dx$ )
3. The number of intervals (N)

(Fig. 8-13 Gordon)

To measure transit times, the clock is used in the manner of a time stamp i.e. each entity holds time stamp. When entity reaches a point from which transit time is started, the arrival time is noted. Later when the entity reaches to end point, the difference is computed

# Pramod Parajuli

# Simulation and Modeling, CS-331

---

## Chapter 10

## Analysis of Simulation Output

(C) 2005, Pramod Parajuli

Source: [www.csitnepal.com](http://www.csitnepal.com)

CSITNEPAL

## Nature of the Problem

- Making a variable stochastic changes the properties of remaining variables to stochastic since the endogenous events make one variable depend upon another
- While characterizing the property of a variable, in every time unit the value changed is compared to predefined intervals known as ***confidence interval*** to find similarity between estimated and simulated value
- Simulation results are not mutually independent

## Central Limit Theorem

- The average of a sample of observations drawn from some population with any shape-distribution is approximately distributed as a normal distribution if certain conditions are met
- Most dominant conditions are; distribution of the parent population and the actual sampling procedure
- Simple application is;  
If random sample size is 'n' ( $n > 30$ ) from an infinite population  
Finite standard deviation  
Then, the standard sample mean converges to a standard normal distribution

(Look in the handout given for the details)

## Estimation Methods

- IID – independently and identically distributed, random numbers are mutually independent
- Central limit theorem

It states that the sum of  $n$  IID variables, drawn from a population that has a mean of  $\mathbf{m}$  and a variance of  $s^2$ , is approximately distributed as a normal variable with a mean of  $n\mathbf{m}$  and variance of  $ns^2$ .

$$X_{norm} \equiv \frac{\sum_{i=1}^N x_i - \sum_{i=1}^N \mathbf{m}}{\sqrt{\sum_{i=1}^N s_i^2}}$$

where,  $X_1, X_2, X_3, \dots, X_N$  : set of independent random variates  
each  $X_i$  has arbitrary probability distribution of  $P(x_1, x_2, x_3, \dots, x_N)$   
with mean  $\mathbf{m}_i$ , and variance  $s_i^2$

## Estimation Methods

- Any normal distribution can be transformed into a standard normal distribution (mean = 0, variance 1)
- Therefore, the normal variate will be

$$z = \frac{\sum_{i=1}^n x_i - n\bar{m}}{\sqrt{n} s}$$

- If, sample mean

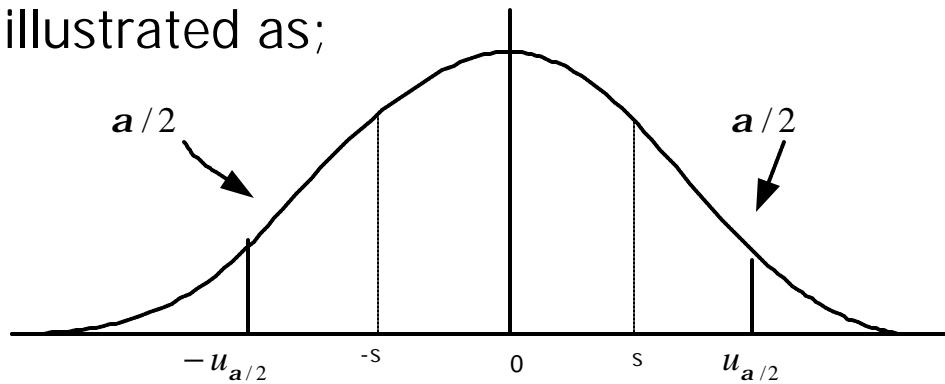
$$\bar{x} = \frac{1}{n} \cdot \sum_{i=1}^n x_i$$

Then,

$$z = \frac{\bar{x} - \bar{m}}{s / \sqrt{n}}$$

## Estimation Methods

- The probability distribution of standard normal variate can be illustrated as;



- The integral from  $-\infty$  to a value  $u$  is the probability that  $z$  is less than or equal to  $u$
- This integral is generally denoted by  $F(u)$
- Let for a chosen value of  $u$  is satisfying

$$\Phi(u) = 1 - \frac{a}{2}$$

**a** =some constant less than 1

## Estimation Methods

- Let the value of  $\alpha$  be  $u_{\alpha/2}$  i.e. the probability that  $z$  is greater than  $u_{\alpha/2}$  is  $\alpha/2$ .
- Same holds for  $-u_{\alpha/2}$
- The probability that  $z$  lies between  $u_{\alpha/2}$  and  $-u_{\alpha/2}$  is

$$\Pr ob\{-u_{\alpha/2} \leq z \leq u_{\alpha/2}\} = 1 - \alpha$$

- In terms of sample mean, the probability can be written as;

$$\Pr ob\left\{ \bar{x} - \frac{s}{\sqrt{n}} u_{\alpha/2} \leq m \leq \bar{x} + \frac{s}{\sqrt{n}} u_{\alpha/2} \right\} = 1 - \alpha$$

- The constant  $1 - \alpha$  is confidence level and

$\bar{x} + \frac{s}{\sqrt{n}} \cdot u_{\alpha/2}$  is confidence interval

## Estimation Methods

- The size of the confidence interval depends upon the confidence level
- If confidence level is 90%, then the value of  $u_{\alpha/2}$  is 1.65
- That is,  $\mu$  will be covered by the confidence interval

$$\bar{x} \pm \frac{1.65s}{\sqrt{n}}$$

with probability 0.9. i.e. greater the value of 'n' (more no. of simulation runs), the confidence interval can be expected to cover the value  $\mu$  on 90% of the repetitions

- In practice population variance  $s^2$  is not known, in such case, it is replaced by an estimate calculated using;

$$s^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$$

## Estimation Methods

- $s^2$  has Student-t distribution with  $n-1$  degrees of freedom  
i.e.  $n-1$  number of parameters may be independently varied
- Useful if samples drawn are normally distributed

## Simulation Run Statistics

- For CLT, two more considerations exists;
  1. The observations are mutually independent
  2. The distribution from which they are drawn is stationary
- But in most of the cases, both of these considerations do not hold
- E.g. (mutually independent)

Let's consider a system with 'n' successive entities. Let  $x_i$  represents wait time for  $i^{\text{th}}$  entity. To represent mean wait time;

$$\bar{x}(n) = \frac{1}{n} \cdot \sum_{i=1}^n x_i$$

As wait time of  $i^{\text{th}}$  entity affect wait time of remaining entities, they are not independent. Therefore, they are auto-correlated

(C) 2005, Pramod Parajuli

## Simulation Run Statistics

- In most of simulation run, the sample mean of auto-correlated data is approximated to a normal distribution as the sample size increases
- In such case, the equation in slide-10 holds
- However, variance of auto-correlated data will not exhibit same property as variance of independent data
- Therefore, some adjustment is made for auto-correlated data. Failure to adjust will result into under-estimation or over-estimation
- E.g. (stationary)
  - Early arrivals get service quickly, later arrivals have to wait
  - Large number of simulation runs reduce the bias

(C) 2005, Pramod Parajuli

## **Simulation Run Statistics**

- If distribution for particular sample size is taken, it might differ from another sample size
- This leads to dynamic distribution
- E.g. mean wait time
  - **fig( 14.2) Gordon**
- Even after 2,000 samples, the mean is not in steady state. For low server utilization, steady state is reached very fast but our concern is for higher server utilization<sup>12</sup>

(C) 2005, Pramod Parajuli

## Replication of Runs

- Repeated no. of simulations generate independent results
- For each simulation run, a different random numbers are used for same sample size 'n' and the simulation gives a set of independent determinations of sample mean

$$\bar{x}(n)$$

- Even though the sample mean depends on degree of auto-correlation, the independent determinations can be used to estimate the variance of the distribution
- Let the simulation is run for 'p' times with independent random number series
- Let  $x_{ij}$  be the  $i^{th}$  observation in the  $j^{th}$  run and let the sample mean and variance for the  $j^{th}$  run be;

$$\bar{x}_j(n) \qquad s_j^2(n)$$

(C) 2005, Pramod Parajuli

## Replication of Runs

$$\bar{x}_j(n) = \frac{1}{n} \sum_{i=1}^n x_{ij} \quad s_j^2(n) = \frac{1}{n-1} \sum_{i=1}^n [x_{ij} - \bar{x}_j(n)]^2$$

- Since there are 'p' number of runs,

$$x = \frac{1}{p} \sum_{j=1}^p \bar{x}_j(n) \quad s^2 = \frac{1}{p} \sum_{j=1}^p s_j^2(n)$$

- Look into figure 14-3 in Gordon, p = 100

For sample size, n = 5, and for  $\theta = 0.2, 0.3, 0.4$

For n = 10,  $\theta = 0.5, 0.6$

- 'p' repetitions of 'n' observations involve a total of  $N = p.n$  observations.
- Dividing 'N' observations into number of ranges is quite difficult
- It is suggested that number of repetitions be as low as possible → reduction in the approximation of normal distribution with sample means

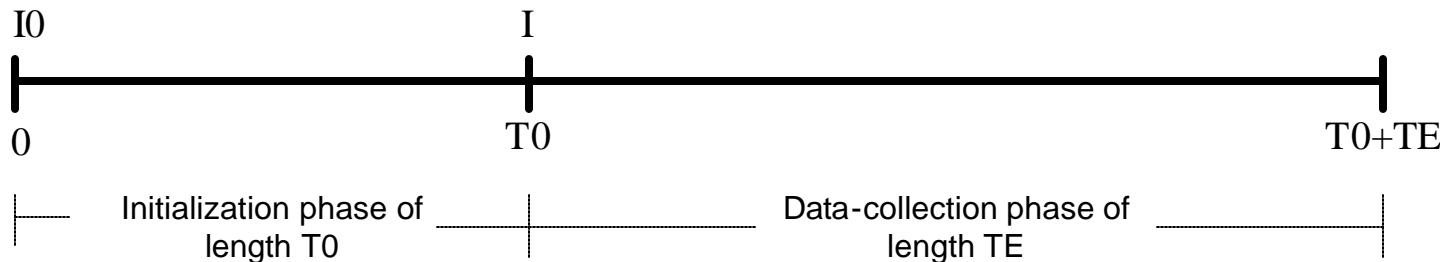
## Initialization Bias

- Initial system state representation is more representative of long-run conditions
- This step is also known as intelligent-initialization
- Two ways to specify initial conditions;
  1. If the system exists, collect data on it and use these data to specify more typical initial conditions.
    - It requires a large data collection effort
    - If system being modeled does not exist, this method is impossible to implement
    - Collecting system data from a second model which is a simplified model is suggested for complex systems
  2. Reduce the impact of initial conditions, by dividing each simulation run into two phases;
    1. Initialization phase from time 0 to  $T_0$
    2. Data collection phase from time  $T_0$  to stopping time  $T_0 + T_E$

# Initialization Bias

Specified Initial  
Conditions

“Steady-state” initial  
conditions,



- In such case, simulation begins at time 0 under specified initial conditions  $I_0$ , and runs for specified period of time  $T_0$ .
- Data collection on the response variables of interest starts from  $T_0$  and until time  $T_0+T_E$ .
- The length  $T_E$  of the data collection phase should be long enough to guarantee sufficiently precise estimates of steady-state behavior

## Elimination of Initial Bias

Two approaches to eliminate initial bias;

1. System can be started in a more representative state than the empty state
2. Or, first part of simulation run can be ignored i.e. take care of those data that come only after steady-state

Ideal situation to remove initial bias is to select initial conditions from steady state distribution.

More common approach to remove initial bias is to eliminate an initial section of the run. The run is started from an idle state and stopped after certain period of time. The run is restarted with statistics being gathered from the point of restart

## **Elimination of Initial Bias**

Pilot runs – set of runs which are conducted to judge how long the initial bias remains

### Disadvantages

- More number of pilot runs required
- Eliminating the first part of a simulation introduces variance estimate, confidence limit etc. be based on less information

# Pramod Parajuli

## Simulation and Modeling, CS-331

---

### Chapter 11

#### Introduction to GPSS

## General Description

- Blocks → activities
- Lines → sequence of activities
- For a choice, more than one line leaves a block and the condition for the choice is stated at the block
- Entities move through the system e.g. messages in a communication system, motor vehicles in transportation ..

### Block

- Name of block
- Set of data fields (A,B, C, ...)

# General Description

## Resources

GPSS Entities	Expressions	Queues
Block statements	Logic Switches	Storages
Reports	Plot	Data streams
Savevalue	Tables	User chains
Math libraries (RNGs, Statistical Analysis, etc.)		

## GPSS Entities

Transaction	Block
Facility	Function
Logic switch	Matrix
Queue	Storage
Save value	Table
User chain	Variable
Numeric group	Transaction group

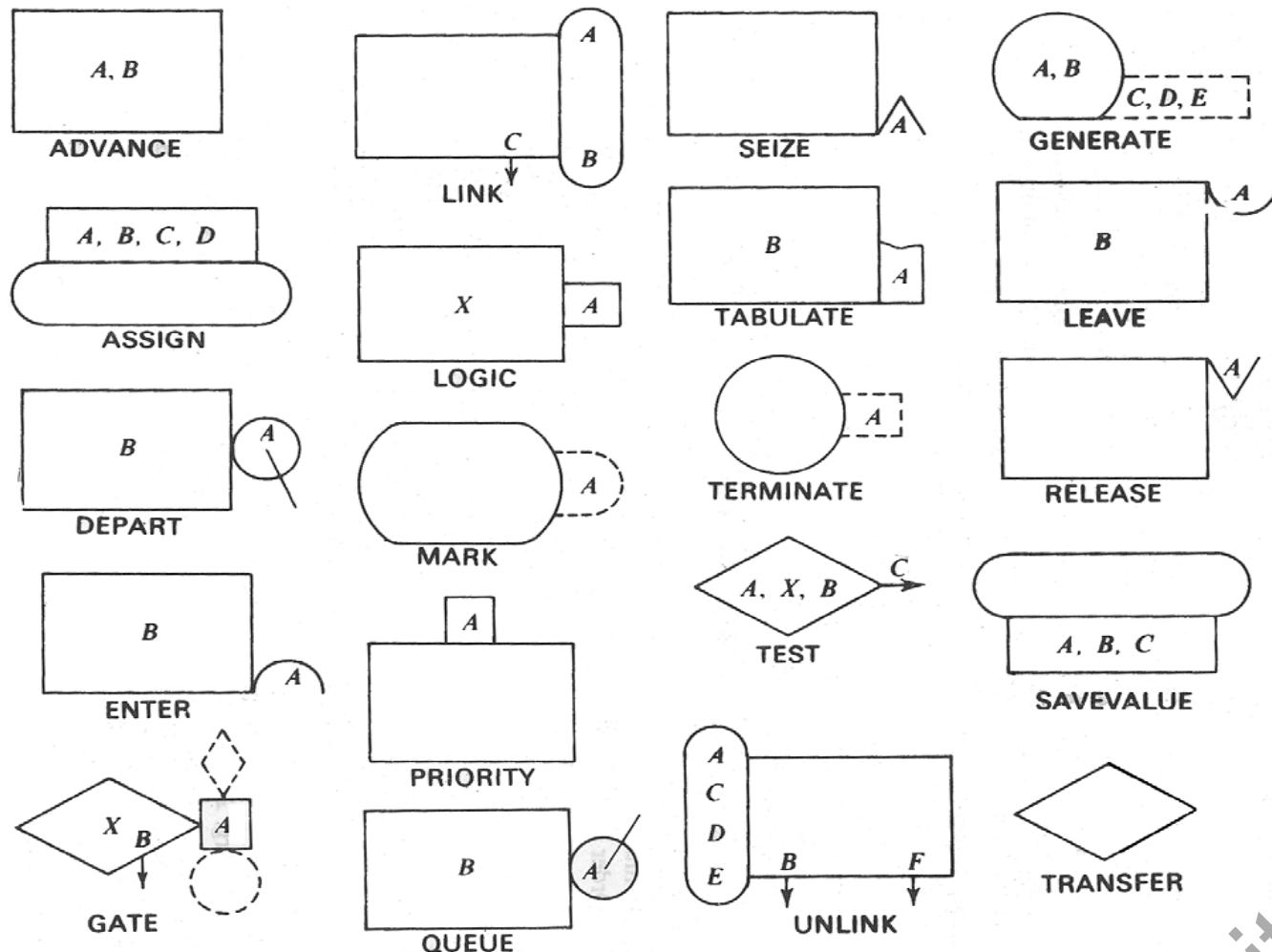
# General Description

## GPSS – Basic commands

CLEAR	reset statistics and remove transaction
CONTINUE	resume the simulation
EXIT	end the GPSS world session
HALT	stop the simulation and delete all queued commands
INCLUDE	read and translate a secondary model file
INTEGRATE	automatically integrate a time differential in a use variable
REPORT	set the name of the report file or request an immediate report
RESET	reset the statistics of the simulation
SHOW	evaluate and display expression
START	set the termination count and begin a simulation
STEP	attempt a limited number of block entities
STOP	set a stop condition based on block entry attempts
STORAGE	define a storage entity
TABLE	define a table entity
VARIABLE	define a variable entity

# General Description

## GPSS – Blocks



(C) 2005, Pramod Parajuli

## General Description

### GPSS – Blocks

- GPSS block diagram consists of many blocks
- An identification number called a '**location**' is given to each block
- Movement of transactions is usually from one block to the block with the next highest location
- Blocks can be given names for the identification if required

# Transaction

- Transaction move from block to block
- Transaction attempting block to block is called '***active transaction***'
- If a transaction fails to find favorable conditions while entering a block, it may come to rest. Then, another transaction is chosen to begin
- Transactions are numbered sequentially throughout a session starting with 1
- CLEAR statement begins the numbering of transaction at 1
- Behavior of transaction is determined by its attributes;  
    GPSS World Reference Manual – Chapter 4 (r4.html), page 3
- ASSIGN, MARK, TRANSFER SUB, SELECT, SPLIT, COUNT blocks create a transaction parameter

# **Transaction**

## Active transaction

- Active transaction go as far as it can
- When it can not move further, another transaction is chosen to be active
- There can be no more than one active transaction

# GPSS Blocks

A, B

## ADVANCE

An **ADVANCE** Block delays the progress of a Transaction for a specified amount of simulated time.

### ADVANCE A,B

#### Operands

**A** - The mean time increment. Required. The operand must be *Name, Number, String, ParenthesizedExpression, SNA* or *SNA\*Parameter*.

**B** - The time half-range or, if a function, the function modifier. Optional. The operand must be *Null, Name, Number, String, ParenthesizedExpression, SNA*, or *SNA\*Parameter*.

#### Example

#### ADVANCE 101.6, 50.3

This example creates a Block which chooses a random number between 51.3 and 151.9, inclusively (i.e. 101.6 plus or minus 50.3), and delays the entering Transaction that amount of simulated time.

# GPSS Blocks

A, B, C, D

## ASSIGN

**ASSIGN** Blocks are used to place or modify a value in a Transaction Parameter.

**ASSIGN A,B,C**

### Operands

**A - Parameter number of the Active Transaction. Required.**

The operand must be *Name*, *PosInteger*, *ParenthesizedExpression*, *SNA*, or *SNA\*Parameter*, followed by +, -, or *Null*.

**B - Value. Required. the operand must be *Name*, *Number*, *String*, *ParenthesizedExpression*, *SNA*, or *SNA\*Parameter*.**

**C - Function number. Optional. The operand must be *Null*, *Name*, *PosInteger*, *ParenthesizedExpression*, *SNA* or *SNA\*Parameter*.**

### Examples

**ASSIGN 2000, 150.6**

This is the simplest way to use the **ASSIGN** Block. The value 150.6 is assigned to Parameter number 2000 of the entering Transaction. If no such Parameter exists, it is created.

## GPSS Blocks

### DEPART

A DEPART Block registers statistics which indicate a reduction in the content of a Queue Entity.

DEPART A,B

#### Operands

A - Queue Entity name or number. Required. The operand must be *Name, PosInteger, ParenthesizedExpression, SNA* or *SNA\*Parameter*.

B - Number of units by which to decrease content of the Queue Entity. Default value is 1. Optional. The operand must be *Null, Name, PosInteger, String, ParenthesizedExpression, SNA*, or *SNA\*Parameter*.

#### Example

DEPART WaitingLine

In this example the content of the Queue Entity named WaitingLine is reduced by one and the associated statistics accumulators are updated.

# GPSS Blocks

## ENTER

When a Transaction attempts to enter an ENTER Block, it either takes or waits for a specified number of storage units.

ENTER A,B

### Operands

A - Storage Entity name or number. Required. The operand must be *Name, PosInteger, ParenthesizedExpression, SNA* or *SNA\*Parameter*.

B - Number of units by which to decrease the available storage capacity. Default value is 1. Optional. The operand must be *Null, Name, PosInteger, ParenthesizedExpression, SNA*, or *SNA\*Parameter*.

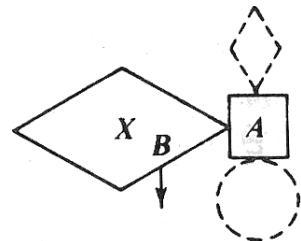
### Example

ENTER Toolkit, 2

In this example the Active Transaction demands 2 storage units from the storage units available at the Storage Entity named Toolkit. If there are not enough storage units remaining in the Storage Entity, the Transaction comes to rest on the Delay Chain of the Storage Entity.

12

# GPSS Blocks



## GATE

A GATE Block alters Transaction flow based on the state of an entity.

GATE O A,B

### Operands

O - Conditional operator. Condition required of entity to be tested for successful test. Required. The operator must be FNV, FV, I, LS, LR, M, NI, NM, NU, SE, SF, SNE, SNF, SNV, SV, or U.

A - Entity name or number to be tested. The entity type is implied by the conditional operator. Required. The operand must be *Name*, *PosInteger*, *ParenthesizedExpression*, *SNA*, or *SNA\*Parameter*.

B - Destination Block number when test is unsuccessful. Optional. The operand must be *Null*, *Name*, *PosInteger*, *ParenthesizedExpression*, *SNA*, *SNA\*Parameter*.

### Examples

GATE SNF MotorPool

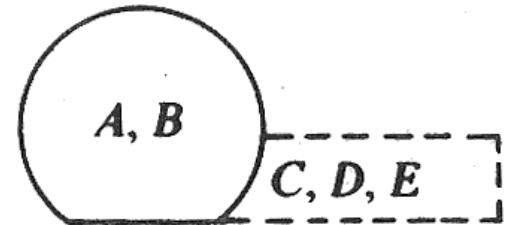
In this example of a "Refuse Mode" GATE Block, the Active Transaction enters the GATE Block if the Storage Entity named MotorPool is not full (i. e. if at least 1 unit of storage is available). If the Storage is full, the Active Transaction is blocked until 1 or more storage units become available.

# GPSS Blocks

## GENERATE

A GENERATE Block creates Transactions for future entry into the simulation.

GENERATE A,B,C,D,E



### Operands

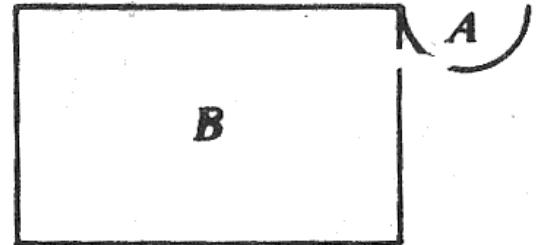
- A - Mean inter generation time. Optional. The operand must be *Null, Name, Number, String, ParenthesizedExpression, or DirectSNA*. You may not use Transaction Parameters.
- B - Inter generation time half-range or Function Modifier. Optional. The operand must be *Null, Name, Number, String, ParenthesizedExpression, or DirectSNA*. You may not use Transaction Parameters.
- C - Start delay time. Time increment for the first Transaction. Optional. The operand must be *Null, Name, Number, String, ParenthesizedExpression, or DirectSNA*. You may not use Transaction Parameters.
- D - Creation limit. The default is no limit. Optional. The operand must be *Null, Name, PosInteger, String, ParenthesizedExpression, or DirectSNA*. You may not use Transaction Parameters.
- E - Priority level. Optional. Zero is the default. The operand must be *Null, Name, integer, String, ParenthesizedExpression, or DirectSNA*. You may not use Transaction Parameters.

### Example

GENERATE 0.1

This is the simplest way to use the GENERATE Block. This Block causes a priority zero Transaction to enter the simulation every tenth of a time unit.

## GPSS Blocks



### LEAVE

A LEAVE Block increases the accessible storage units at a Storage Entity.

LEAVE A,B

#### Operands

A - Storage Entity name or number. Required. The operand must be *Name, PosInteger, ParenthesizedExpression, SNA, or SNA\*Parameter*.

B - Number of storage units. The default is 1. Optional. The operand must be *Null, Name, PosInteger, ParenthesizedExpression, SNA, or SNA\*Parameter*.

#### Example

LEAVE RepairMen,10

In this example, when a Transaction enters the LEAVE Block, the available storage units at the Storage Entity named RepairMen is increased by 10.

# GPSS Blocks

## LINK

A LINK Block controls the placement of the Active Transaction on the User Chain of a Userchain Entity.

LINK A,B,C

### Operands

A - Userchain number. The Userchain Entity which may receive the entering Transaction. Required. The operand must be *Name, PosInteger, ParenthesizedExpression, SNA, or SNA\*Parameter*.

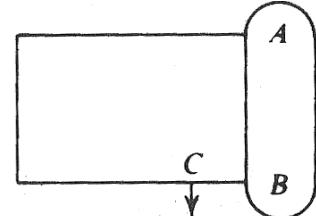
B - Chain ordering. The placement of new Transactions on the Userchain. Required. The operand must be LIFO, FIFO, *ParenthesizedExpression, SNA, or SNA\*Parameter*.

C - Next Block location. The destination Block for Transactions which find the Link Indicator of the Userchain in the off state (reset). Optional. The operand must be *Null, Name, PosInteger, ParenthesizedExpression, SNA or SNA\*Parameter*.

### Example

LINK OnHold,FIFO

In this example, the Active Transaction is placed at the end of the User Chain Entity named OnHold. It is removed from all chains except Transaction Groups and Interrupt Chains. In other words, preemptions are not cleared. The Transaction remains on the User Chain until some other Transaction enters an UNLINK Block and specifically removes it. In the present example, the Transaction is placed at the end of the User Chain named OnHold.



## GPSS Blocks

### LOGIC

A LOGIC Block changes the state of a Logicswitch entity.

LOGIC O A

#### Operands

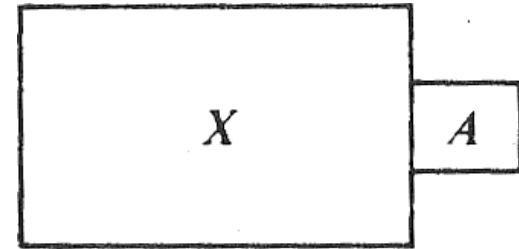
O - Logic operator. Required. The operator must be S, R, or I.

A - Logicswitch Entity number. Required. The operand must be *Name*, *PosInteger*, *ParenthesizedExpression*, *SNA*, or *SNA\*Parameter*.

#### Example

LOGIC S PowerSwitch

In this example, the Logicswitch Entity named PowerSwitch is left in the true or "set" state.



# GPSS Blocks

## MARK

A MARK Block places an absolute clock time stamp into the Active Transaction or into its Parameter.

MARK A

### Operand

A - Parameter number. Parameter to receive value of system clock.

Optional. The operand must be *Null*, *Name*, *PosInteger*,  
*ParenthesizedExpression*, *SNA*, or *SNA\*Parameter*.

### Examples

MARK Beginning

In this example, when a Transaction enters the MARK Block, its Transaction Parameter named Beginning is given a value equal to the value of the absolute system clock, AC1.

MARK

In this example, when a Transaction enters the MARK Block, its Mark Time is set equal to the value of the absolute system clock.

## GPSS Blocks

### PRIORITY

A PRIORITY Block sets the priority of the Active Transaction.

PRIORITY A,B

#### Operands

A - New priority value. Required. The operand must be *Name*, *integer*, *String*, *ParenthesizedExpression*, *SNA*, or *SNA\*Parameter*.

B - Buffer option. Places Active Transaction behind priority peers on CEC.  
Optional. The operand must be *BU* or *Null*.

#### Example

PRIORITY 10

In this example any entering Transaction is made to have a priority of 10.

## GPSS Blocks

### QUEUE

A QUEUE Block updates Queue Entity statistics to reflect an increase in content.

QUEUE A,B

#### Operands

A - Queue Entity name or number. Required. The operand must be *Name*, *PosInteger*, *ParenthesizedExpression*, *SNA*, or *SNA\*Parameter*.

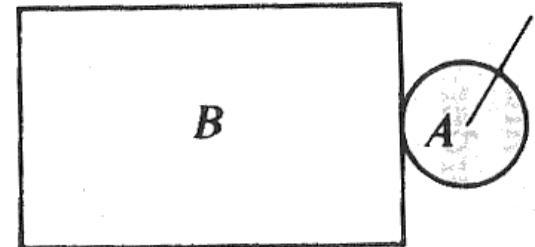
B - Number of units by which to increase the content of the Queue Entity. Default value is 1. Optional. The operand must be *Null*, *Name*, *PosInteger*, *ParenthesizedExpression*, *SNA*, or *SNA\*Parameter*.

#### Example

QUEUE WaitingLine

In this example the content of the Queue Entity named WaitingLine is increased by one and the associated statistics accumulators are updated.

20



## GPSS Blocks

### RELEASE

A RELEASE Block releases ownership of a Facility, or removes a preempted Transaction from contention for a Facility.

RELEASE A

#### Operand

A - Facility number. Required. The operand must be *Name*, *PosInteger*, *ParenthesizedExpression*, *SNA*, or *SNA\*Parameter*.

#### Example

RELEASE Teller1

In this example, when a Transaction which owns the Facility Entity named Teller1 enters the RELEASE Block, it gives up ownership to the Facility.

# GPSS Blocks

## SAVEVALUE

A SAVEVALUE Block changes the value of a Savevalue Entity.

SAVEVALUE A,B

### Operands

A - Savevalue Entity number. Required. May be followed by + or - to indicate addition or subtraction to existing value. Required. The operand must be *Name*, *PosInteger*, *ParenthesizedExpression*, *SNA*, or *SNA\*Parameter*, followed by +, -, or *Null*.

B - The value to be stored, added, or subtracted. Required. The operand must be *Name*, *Number*, *String*, *ParenthesizedExpression*, *SNA*, or *SNA\*Parameter*.

### Examples

SAVEVALUE Account,99.95

In this example, the Savevalue Entity named Account takes on the value 99.95.

SAVEVALUE The\_Bard,"A rose by any other name ..."

In this example, the Savevalue Entity named The\_Bard is assigned a string. If the Savevalue Entity does not exist, it is created.

# GPSS Blocks

## SEIZE

When the Active Transaction attempts to enter a SEIZE Block, it waits for or acquires ownership of a Facility Entity.

SEIZE A

Operand

A - Facility name or number. Required. The operand must be *Name*, *PosInteger*, *ParenthesizedExpression*, *SNA*, or *SNA\*Parameter*.

Example

SEIZE Teller1

In this example, when a Transaction attempts to enter the SEIZE Block, the state of the Facility named Teller1 is tested. If it is idle, ownership is given to the Active Transaction, which is allowed to enter the SEIZE Block and proceed to the Next Sequential Block (NSB). If the Facility is busy (owned), the Active Transaction comes to rest on the Delay Chain of the Facility.

## TABULATE

A TABULATE Block triggers the collection of a data item in a Table Entity.

TABULATE A,B

### Operands

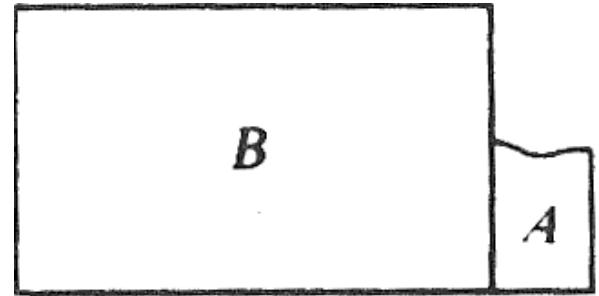
A - Table Entity name or number. Required. The operand must be *Name*, *PosInteger*, *ParenthesizedExpression*, *SNA*, or *SNA\*Parameter*.

B - Weighting factor. Optional. The operand must be *Null*, *Name*, *Number*, *ParenthesizedExpression*, *SNA*, or *SNA\*Parameter*.

### Example

TABULATE Sales

When a Transaction enters this TABULATE Block, the Table Entity named Sales is found. Sales must have been defined in a TABLE Command. Then the statistics associated with the table are updated with no weighting.



## GPSS Blocks

### TERMINATE

A TERMINATE Block removes the Active Transaction from the simulation and optionally reduces the Termination Count.

TERMINATE A

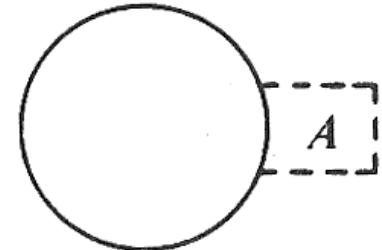
#### Operand

A - Termination Count decrement. Default is 0. Optional. The operand must be *Null*, *Name*, *PosInteger*, *ParenthesizedExpression*, *SNA*, or *SNA\*Parameter*.

#### Example

TERMINATE 1

In this example, when a Transaction enters the TERMINATE Block it is removed from the simulation. Also, the Termination Count of the simulation, which is set by a START Command is decremented by 1.



# GPSS Blocks

## TEST

A TEST Block compares values, normally SNAs, and controls the destination of the Active Transaction based on the result of the comparison.

TEST O A,B,C

### Operands

O - Relational operator. Relationship of Operand A to Operand B for a successful test. Required. The operator must be E, G, GE, L, LE, or NE.

A - Test value. Required. The operand must be *Name, Number, String, ParenthesizedExpression, SNA, or SNA\*Parameter*.

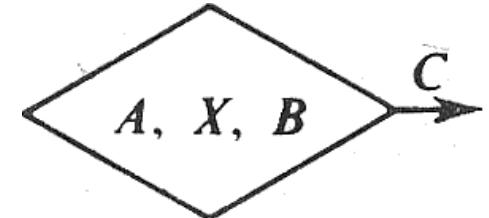
B - Reference value. Required. The operand must be *Name, Number, String, ParenthesizedExpression, SNA, or SNA\*Parameter*.

C - Destination Block number. Optional. The operand must be *Null, Name, PosInteger, ParenthesizedExpression, SNA, or SNA\*Parameter*.

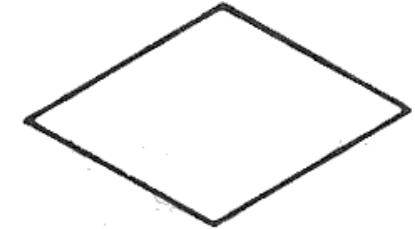
### Example

TEST G C1, 70000

In this example of a "Refuse Mode" TEST Block, the Active Transaction enters the TEST Block if the relative system clock value is greater than 70000. Otherwise, the Transaction is blocked until the test is true.



# GPSS Blocks



## TRANSFER

A TRANSFER Block causes the Active Transaction to jump to a new Block location.

TRANSFER A,B,C,D

### Operands

- A - Transfer Block mode. Described below. Optional. The operand must be BOTH, ALL, PICK, FN, P, SBR, SIM, *fraction*, *Name*, *PosInteger*, *ParenthesizedExpression*, *SNA*, *SNA\*Parameter*, or *Null*.
- B - Block number or location. Parameter name or number when in P Mode. Optional. The operand must be *Null*, *Name*, *PosInteger*, *ParenthesizedExpression*, *SNA*, or *SNA\*Parameter*.
- C - Block number or location. Increment value in FN or P Mode. Optional. The operand must be *Null*, *Name*, *PosInteger*, *ParenthesizedExpression*, *SNA*, or *SNA\*Parameter*.
- D - Block number increment for ALL Mode. Default is 1. Optional. The operand must be *Null*, *Name*, *PosInteger*, *ParenthesizedExpression*, *SNA*, or *SNA\*Parameter*.

### Example:

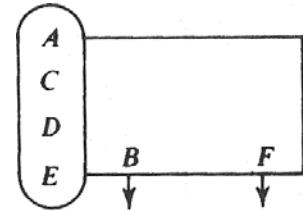
Look into the modes.

# GPSS Blocks

## UNLINK

An UNLINK Block removes Transactions from the User Chain of a Userchain Entity.

UNLINK O A,B,C,D,E,F



### Operands

O - Relational operator. Relationship of D to E for removal to occur. These choices are explained below. Optional. The operator must be *Null*, E, G, GE, L, LE or NE.

A - User Chain number. User Chain from which one or more Transactions will be removed. Required. The operand must be *Name*, *PosInteger*, *ParenthesizedExpression*, *SNA*, or *SNA\*Parameter*.

B - Block number. The destination Block for removed Transactions. Required. The operand must be *Name*, *PosInteger*, *ParenthesizedExpression*, *SNA*, or *SNA\*Parameter*.

C - Removal limit. The maximum number of Transactions to be removed. If not specified, ALL is used. Optional. The operand must be ALL, *Null*, *Name*, *PosInteger*, *ParenthesizedExpression*, *SNA*, or *SNA\*Parameter*.

D - Test value. The member Transaction Parameter name or number to be tested, a Boolean variable to be tested, or BACK to remove from the tail of the chain. Optional. The operand must be *Null*, *Name*, *PosInteger*, *ParenthesizedExpression*, *SNA*, *SNA\*Parameter* or BACK.

E - Reference value. The value against which the D Operand is compared. Optional. The operand must be *Null*, *Name*, *Number*, *String*, *ParenthesizedExpression*, *SNA*, or *SNA\*Parameter*. Operand E is not used if Operand D is a Boolean Variable.

F - Block number. The alternate destination for the entering Transaction. Optional. The operand must be *Null*, *Name*, *PosInteger*, *ParenthesizedExpression*, *SNA*, or *SNA\*Parameter*.

### Example

UNLINK OnHold,Reentry,1

This is the simplest way to use the UNLINK Block. The first Transaction at the head of the Userchain Entity named OnHold, if any, is taken off the chain and is directed to the Block labeled Reentry. It is put on the CEC behind Transactions of the same priority. The Transaction entering the UNLINK Block proceeds to the Next Sequential Block. 28

# GPSS Control Statements

## CLEAR

A CLEAR Command returns the simulation to the unused state.

CLEAR A

### Operand

A - ON or OFF. If the A Operand is omitted, ON is assumed. Optional. The operand must be ON, OFF or Null.

## END

The ENDControl Statement has been replaced by EXIT, which can terminate a Session. END is now a keyword in the PLUS Language.

# GPSS Control Statements

## FUNCTION

A FUNCTION Command defines the rules for a table lookup.

There are several types of Function Entities. Each has its own rules pertaining to the table lookup. For each, the lookup table is specified in one or more Function Follower Statements. Type C Functions are a special case. They use a table lookup, followed by a linear interpolation.

The use of Function Commands to define probability distributions has been largely supplanted by the built-in distributions in the Procedure Library. This is discussed in Chapter 8. The old Function Types are still supported by GPSS World.

NAME FUNCTION A,B

Label / Operands

NAME - Entity Label this entity. Required. The field must be Name.

A - Function argument. Required. The operand must be Name, PosInteger, String, ParenthesizedExpression, SNA, or SNA\*Parameter.

B - Function type (one letter) followed immediately by the number of data pairs in the Function Follower Statements. Required.

30

# GPSS Control Statements

## INITIAL

An INITIAL Command initializes a Matrix Entity, a Logicswitch Entity, Savevalue Entity, or an element of a Matrix Entity.

INITIAL A,B

Operands

A - Logicswitch, Savevalue, or Matrix element specified as SNA, or the name of a Matrix Entity. Operand A must have the form of an LS, X, or MX class SNA, or a Matrix Name. Required. The operand must be Name, LSPosInteger, LS\$Name, XPosInteger, X\$Name, MXPosInteger(m,n) or MX\$ Name(m,n). Coordinates (m,n) must be Name or PosInteger.

B - Value to be assigned, or "UNSPECIFIED" The default is 1. Optional. The operand must be Null, Number, String, Name, or UNSPECIFIED.

# **GPSS Control Statements**

## **RESET**

A RESET Command marks the beginning of a measurement period.

**RESET**

Operands

None.

# GPSS Control Statements

## START

A START Command begins a simulation.

```
START A,B,C,D
```

### Operands

- A - Termination count. Required. The operand must be PosInteger.
- B - Printout operand. NP for “no printout”. Default is to print a standard report. Optional. The operand must be NP or Null.
- C - Not used. Kept for compatibility with older dialects of GPSS.
- D - Chain printout. 1 to include the CEC and FEC in the standard report. Optional. The operand must be Null, or PosInteger.

# GPSS Control Statements

## STORAGE

**A STORAGE Command defines a Storage Entity.**

NAME    STORAGE    A

Label / Operand

NAME - Entity Label for this entity. Required. The field must be Name.

A - Total storage capacity. Required. The operand must be PosInteger.

Example

MotorPool    STORAGE    20

This Command defines a Storage Entity named MotorPool with a total capacity of 20 units.

**Note:** Storage name must be defined before the simulation program and name of storage must start with 3 alphabets.

# GPSS Control Statements

## TABLE

**A TABLE Command initializes a frequency distribution table.**

**NAME TABLE A,B,C,D**

Label / Operands

NAME - Entity Label for this entity. Required. The field must be Name. The length of a Table name is limited to 32 characters.

A - Table argument. The data item whose frequency distribution is to be tabulated. Optional. The operand must be Name, Number, String, ParenthesizedExpression, or SNA. Ignored by ANOVA, but must be specified when used by TABULATE Blocks.

B - Upper limit of first frequency class. The maximum argument which causes the first frequency class to be updated. Required. The operand must be Number or String.

C - Size of frequency classes. The difference between the upper limit and lower limit of each frequency class. Required. The operand must be Number or String.

D - Number of frequency classes. Required. The operand must be PosInteger.

# **Entities**

**Transaction entities:**

GENERATE, SPLIT, TRANSFER, TERMINATE ..

**Facilities entities:**

SEIZE, RELEASE ..

**Queue entities:**

QUEUE, DEPART

**Storage entities:**

ENTER, LEAVE

## Action Times

- An interval of time is called an ***action time***
- Represented by an integral number
- All times are in terms of same unit
- GENERATE and ADVANCE blocks employ action times
- GENERATE block controls the interval between successive arrivals of transaction
- ADVANCE is concerned with representation of the expenditure of time
- Action time may be fixed interval or a random variable
- Action time is defined by giving ***mean (A)*** and a ***function modifier (B)***
- If 'B' is zero, the action time is constant equal to the mean

## Action Times

- The modifier can be specified as a ***function*** that controls the action time
  - The action time is derived by multiplying the mean by the value of the function
  - Function might take various parameters
- 
- As seen already, GENERATE block begins creating transactions from zero time and continues
  - The 'C' field can be used to specify an offset time as the time when the first transaction will arrive (Ref. 14)
  - Transactions have priority level and they can carry items of data called parameters
  - 'E' field determines the priority of the transaction

## Action Times

- The parameters can exist in four formats
  - Signed integers of fullword
  - Signed integers of halfword
  - Signed integers of byte
  - Signed floating-point numbers
- If not specified, the program creates transactions with 12 halfword parameters.

## Succession of Events

- The program maintains records of when each transaction in the system is due to move
- If more than one transaction due to move, transactions are processed according to their priority
- If same priority, then first-come, or first-served basis is applied
- No time is spent to enter into ADVANCE block
- If the transaction can not enter a block due to some conditions, it is monitored and;
  - Another transaction is started from that point or,
  - It enters TERMINATE block and it is removed from the simulation or,
  - It is kept on a chain
- When processing of a transaction is finished, the program looks into other transactions at that instant

## Choice of Paths

- TRANSFER block allows some other location than the next sequential location to be selected
- The choice between two blocks is referred to as next blocks A and B
- The ***selection factor*** in field A of TRANSFER block guides the selection
- It can be set to indicate one of nine choices (coming..)
- Next blocks 'A' and 'B' are placed in fields 'B' and 'C'
- If no choice, the selection factor is left blank
- Random choice can be set by using selection factor 'S'
- Probability of selecting 'A' is then ' $1-S$ ' and next block 'B' is ' $S$ '
- Setting field 'A' to BOTH, allows a transaction to select an alternate path depending upon existing conditions.<sup>41</sup>

## Choice of Paths

- If move is possible, transaction goes to A otherwise to B
- If both are impossible, transaction waits for the first to become possible, giving preference to A in the event of simultaneity

## Facilities and Storages

- Facilities and Storages are permanent entities that operates on the transactions
- Facility is defined as an entity that can be engaged by a single transaction at a time
- Storage is defined as an entity that can be occupied by many transactions at a time up to some predefined limit
- A transaction controlling a facility can be interrupted or preempted by another transaction
- Both facilities and storages can be made unavailable if the equipment they represent breaks down and made available again if repaired
- Entities are numbered starting from 1

# Facilities and Storages

## Example

Type of system	Transaction	Facility	Storage
Communications	Message	Switch	Trunk
Transportation	Car	Toll booth	Road
Data processing	Record	Key punch	Computer memory

In SEIZE, RELEASE, ENTER, LEAVE blocks, field 'A' indicates which facility or storage is intended

- SEIZE block allows the transaction to engage a facility if it is available
- The RELEASE block allows the transaction to disengage the facility
- ENTER block allows a transaction to occupy space in storage
- LEAVE block allows it to give up the space

# Facilities and Storages

## Example

GENERATE	5	
SEIZE	1	
ADVANCE	4 , 3	
RELEASE	1	
TRANSFER	0 . 1 , ACC , REJ	
ACC	TERMINATE	1
REJ	TERMINATE	1

Sample program - 2

Using facility only

Get one inspector

myStorage	STORAGE	3
Beg	GENERATE	5
	ENTER	myStorage
	ADVANCE	12 , 9
	LEAVE	myStorage
	TRANSFER	0 . 1 , ACC , REJ
ACC	TERMINATE	1
REJ	TERMINATE	1

Sample program - 3

Using storage

Get three inspectors

## Gathering Statistics

- Some blocks are used to gather statistics
- E.g. QUEUE, DEPART, MARK, TABULATE
- These blocks introduce two other entities, queues and tables
- When conditions are not favorable to enter a block, transactions may be kept waiting at a block. When conditions are favorable, they are allowed to move on
- QUEUE block increases and DEPART block decreases the queue numbered in field 'A'
- If field B is blank, the change is a unit change otherwise the value of field B
- The program measures average and maximum queue lengths, distribution of time spent on the queue etc.

## Gathering Statistics

- To measure the length of time taken by transactions to move through the system or part of the system, MARK and TABULATE blocks are used.
- MARK block notes the time of arrival on the transaction
- TABULATE block subtracts the time noted by a MARK block from the time of arrival at TABULATE block
- The time, '***transit time***', is entered in a table whose number or name is indicated in field 'A' of TABULATE block
- If transaction entering a TABULATE block has not passed through a MARK block, the transit time is derived by using as a base the time at which the transaction was created
- Therefore, MARK block can be considered to reset the transit time to zero

# Gathering Statistics

## Example

myStorage	STORAGE	3
myTable	TABLE	M1 , 5 , 5 , 10
Beg	GENERATE	5
	QUEUE	1
	ENTER	myStorage
	DEPART	1
	MARK	
	ADVANCE	12 , 9
	LEAVE	myStorage
	TABULATE	myTable
	TRANSFER	0 . 1 , ACC , REJ
ACC	TERMINATE	1
REJ	TERMINATE	1

**TABLE A,B,C,D**

**A = transit time tabulation**

**B = lower limit of the table**

**C = tabulation interval size**

**D = no. of intervals**

# Conditional Transfers

- TRANSFER block – conditional, and unconditional

```
myTable      TABLE M1,5,5,10
Beg   GENERATE      5
      ADVANCE       2
      TRANSFER
      BOTH,,CONV1
      SEIZE 1
      ADVANCE      12,9
      RELEASE       1
TAB    TABULATE      myTable
      TERMINATE     1
CONV1  ADVANCE      2
      TRANSFER
      BOTH,,CONV2
      SEIZE 2
      ADVANCE      12,9
      RELEASE       2
      TRANSFER     ,TAB
```

Gordon – 215  
Explain

```
CONV2 ADVANCE      2
      TRANSFER
      BOTH,,CONV2
      SEIZE 3
      ADVANCE      12,9
      RELEASE       3
      TRANSFER     ,TAB
CONV3 TERMINATE
```

## Program Control Statements

CLEAR - wipe out statistics and transactions. Re-run starts simulation from beginning. But do not reset random number generator seeds

RESET – clears the statistics gathered, reset relative clock

START – continues with previous statistics if not wiped out

START

CLEAR

START

Given sequence of statements run the same simulation twice but the second run would use a different set of random numbers

JOB – instructs the program to wipe out all preceding model and proceed with the following problem

## **Program Control Statements**

JOB – resets the random number seeds so each simulation finds the program in exactly the same form as does the first problem

## Reports

Title

GPSS World Simulation Report - SAMPLE9.1.1

Tuesday June 6, 2000 13:20:07

The title line of the standard report is taken from the name of the Model File that produced the report. The Date and Time of the running of the model is also included.

# Reports

## General Information

START TIME	END TIME	BLOCKS	FACILITIES	STORAGES
0.000	11359.609	32	3	1

**START TIME.** The absolute system clock at the beginning of the measurement period. Utilizations and space-time products are based on the START TIME. The START TIME is set equal to the absolute system clock by a RESET or CLEAR statement.

**END TIME.** The absolute clock time that the termination count became 0.

**BLOCKS.** The number of Block entities in the simulation at the end of the simulation.

**FACILITIES.** The number of Facility entities in the simulation at the end of the simulation.

**STORAGES.** The number of Storage entities in the simulation at the end of the simulation.

# Reports

## Names

NAME	VALUE
ADDUP	10007.000
CHAIN1	10012.000
COLLECT	10017.000

- NAME. User assigned names used in your GPSS World model since the last Translation.
- VALUE. The numeric value assigned to the name. System assigned numbers start at 10000.

# Reports

## Blocks

LABEL	LOC	BLOCK TYPE	ENTRY COUNT	CURRENT-COUNT	RETRY
1		GENERATE	61	1	0
2		JOIN	60	0	0
3		JOIN	60	0	0
..	..		..	.	.
11		LINK	51	0	0
NXTBLK	12	SEIZE	51	0	0

LABEL. Alphanumeric name of this Block if given one.

LOC. Numerical position of this Block in the model. “Location”.

BLOCK TYPE. The GPSS Block name.

ENTRY COUNT. The number of Transactions to enter this Block since the last RESET or CLEAR statement or since the last Translation.

CURRENT COUNT. The number of Transactions in this Block at the end of the simulation.

RETRY. The number of Transactions waiting for a specific condition depending on the state of this Block Entity.

# Reports

Refer to Chapter – 11 of 'Reference Manual'  
For

Facilities	Queues	Storages
Tables and Qtables	Userchains	Transaction Groups
Numeric Groups	Logic Switches	Savevalues
Matrix Entities		
The Current Events Chain		
The Future Events Chain		

# Pramod Parajuli Simulation and Modeling, CS-331

---

## Chapter 12 GPSS Examples

(C) 2005, Pramod Parajuli

Source: [www.csitnepal.com](http://www.csitnepal.com)

1  
csitnepal

# Priorities and Parameters

- All Transactions are processed in same manner
- Transactions have two types of attribute which influence the way they are processed
  - 128 level of priority (0 – 127), 0 - lowest priority
  - Parameters

## Priority

- Priority can be set up or down to any levels by PRIORITY block
- Priority can be set by using 'E' field of generate block. Blank field denotes zero priority
- If there is competition between transactions to occupy a block, priorities are measured. Within same priority, first in, first out

# Priorities and Parameters

## Parameters

- Parameters are characterized by numerical data
- The values are identified by the notation **Pxn** where 'n' is parameter number and 'x' is the type.
- x can be; F, H, B, L for full word, half word, byte, floating point respectively
- If no declaration made, 12 halfword parameters are assigned – PHn
- All parameter values are zero at the time a transaction is created
- A value is given to the transaction when it enters an ASSIGN block
- The number of parameter is given in field 'A' of ASSIGN block.

# Priorities and Parameters

## Parameters

- The value of parameter is given in field 'B'
- Value can be specific number or SNA
- Field 'C' indicates type – F, H, B, L
- + sign adds to, - sign subtract from, and number replace the value of the parameter

# Standard Numerical Attributes (SNA's)

## Contains

- One or two letter code
- A number

'Storage number 5' is denoted by S5

'Length of queue number 15' is denoted by Q15

SNA's combined with operators can form variable statements

myVariable VARIABLE S6+5\*(Q12+Q17)

Similarly, to define floating point, FVARIABLE

Recall 'M1' parameter to field 'A' of TABULATE – it is SNA

Current value of SNA can be used as the value of almost any  
field of a block

Value of SNA's are calculated at the time they are required.

The value is never maintained for future use

## Standard Numerical Attributes (SNA's)

- To save value of SNA, SAVEVALUE is used.
- Field 'A' provides the number of one of many savevalue locations, and field B gives SNA to be saved.
- The roles of + sign and - sign are same as ASSIGN block
- Field 'C' specifies type – 'XF', 'XH', 'XB', 'XL'

SAVEVALUE 10, S6, XH

- Save contents of storage number 6 n halfword savevalue number 10
- Notation Xxn
- INITIAL can set the value of a savevalue at the beginning of the simulation. For the purpose, notation Xxn is put in field A and initial value in field B

# Standard Numerical Attributes (SNA's)

C1	Current value of clock
CHn	Number of transactions on chain 'n'
Fn	Current status of facility number 'n'. 1- if facility is busy, 0 otherwise
FNn	Value of function 'n'.
Kn	Integer 'n'
M1	Transit time of a transaction
Nn	Total number of transactions that have entered block 'n'
Pxn	Parameter number 'n' of transaction, of type 'x'
Qn	Length of queue 'n'
Rn	Space remaining in storage 'n'
RNn	A computed random number having one of the values 1 to 999 with equal probability. 'n' = 1, 2, . . . , 8
Sn	Current occupancy of storage 'n'
Vn	Value of variable statement number 'n'
Wn	Number of transactions currently at block 'n'
Xxn	Value of savevalue location 'n' of type 'x'

# Functions

- Introduce functional relationships in a model
- Each function defined by giving two or more pairs of numbers that relate an input  $x$  to an output  $y$
- Function can be continuous or discrete
- In continuous, program interpolates between defined points to approximate the continuous function
- In discrete, function shows staircase property

Fig 10-1, Gordon

- Functions can take SNA's as parameter which can be treated as following;
  - By using uniformly distributed random number  $RN_n$ , as input, any other distribution can be obtained
  - By using clock time  $C_1$ , action times can be made to depend upon clock time, thereby simulating the effects of peak loads

# Functions

- By using the current contents of a storage, can action time can depend upon the current load on some part of the system
- By using a parameter, an action time can depend on each particular transaction
- The choice of parameter is made at function definition time
- Format of FUNCTION

Field	Contents
Location	Function number
Operation	FUNCTION
A	SNA to be used as input
B	Cn for continuous mode Dn for discontinuous mode where 'n' is the number of points to be defined

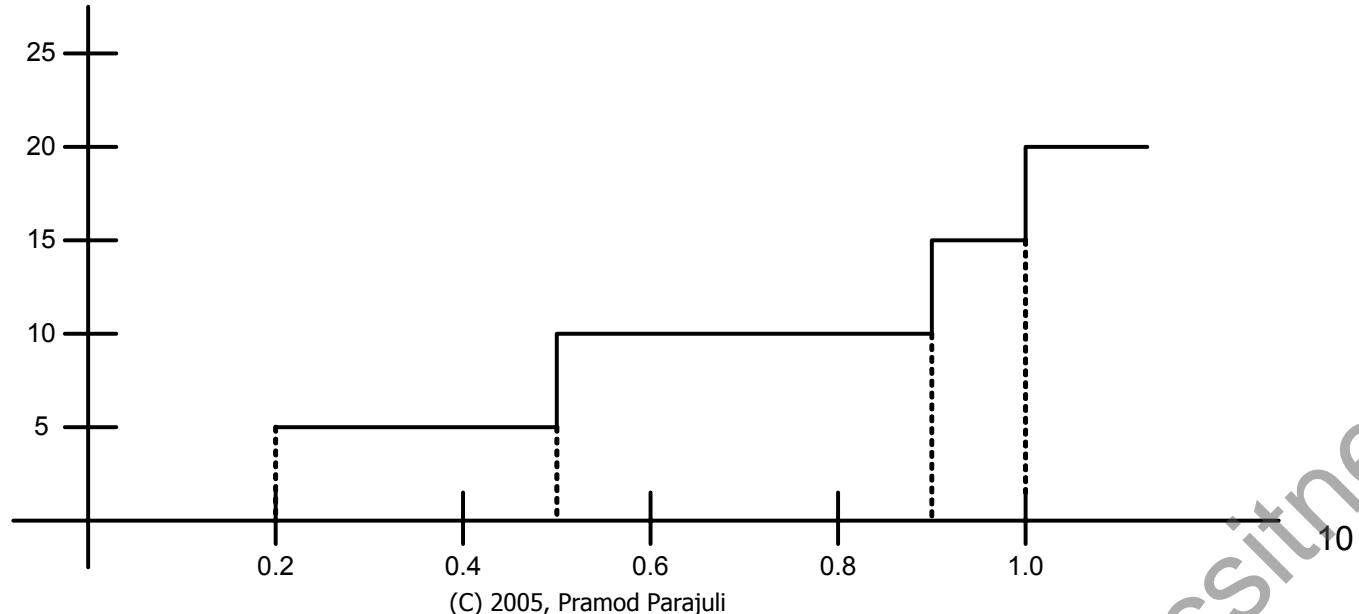
# Functions

## Example

myFunction     FUNCTION     RN1, D4  
0.2, 5/0.5, 10/0.9, 15/1, 20

i.e.  $x_1, y_1/x_2, y_2/x_3, x_4$

A plot of 'myFunction' will be;



## Transfer Modes

- To use parameter mode, Px is used in the 'A' field of the TRANSFER block, and parameter number is put in field 'B'
- For function mode, FN is put in field 'A' and function number in field 'B'
- If a number is put in field 'C', it is added to the parameter or function value
- A list mode function assumes that the input is an integer 'n' and it returns nth listed vlaue of the function
- E.g. if transactions are to be sent to one of four locations called LOCA, LOCB, LOCC, and LOCD, and suppose one of the numbers 1,2,3,4, let's say 3, is placed in a parameter,

## Transfer Modes

- Then;

TRANSFER FN, myFunc

.....

myFunc      FUNCTION PH3, L4  
1,LOCA/2,LOCB/3,LOCC/4,LOCD

L4 signifies that the function is in the list mode and has four listed values

## Logic Switches

- Represent two state conditions in a system
- Each switch is either on or off
- Block – LOGIC
- Transaction entering the block either set the switch ON or OFF or invert
- S – set, R – reset, I – inverse
- Can be useful when status need to be checked before entering into next block

## Testing Conditions

- Control flow of transactions
- GATE block is used for the purpose
- Symbols to represent conditions

LS n      Logic switch 'n' set

LR n      Logic switch 'n' reset

U n      Facility 'n' in use

NU n      Facility 'n' not in use

SF n      Storage 'n' full

SNF n      Storage 'n' not full

SE n      Storage 'n' empty

SNE n      Storage 'n' not empty

## Testing Conditions

- Transaction enters GATE block if the condition being tested is true
- If condition is not true, there is choice of action
- If alternative block is specified, the transaction will be sent to the alternative block immediately
- If no alternative block, transaction waits for favorable condition
- TEST block can be used to test variety of relationships between any two SNA's

G	Greater than	GE	Greater than or equal
E	Equal	NE	Not equal
LE	Less than or equal	L	Less than

## Set Operations

- An important requirement – capability of handling sets of temporary entities which have some common property
- Blocked transactions are automatically entered and removed from sets with a FIFO discipline
- Userchains are available and a transaction is placed on a chain when it enters a LINK block
- Field 'A' carries name/number of chain and field B indicates queuing discipline (FIFO,LIFO)
- If Pxn is used, transactions are ordered by ascending values of parameter number n, with FIFO rule for transactions having the same value
- While on chain, the transactions remain at the LINK block

## Set Operations

- UNLINK block allows a transaction to remove transactions from the chain
- Field A – block name, field B – location to which unlinked transactions are to go. Field C specifies how the transactions are to be removed
- The count can be an integer, SNA, or 'ALL' to remove all transactions
- If only A,B,C are specified, program removes transactions from beginning of the chain
- JOIN allows a transaction to make itself a member of a group
- REMOVE allows a transaction to remove itself from the group

## **Set Operations**

- SPLIT – allows one transaction to produce many copies which are automatically linked as a set but may proceed independently
- ASSEMBLE – gathers a given number of copies and merge them into one
- MATCH – synchronizes the movement of copies

## Examples

Simulation of a Supermarket  
Gordon – 227

GPSS Model of a Simple Telephone System  
Gordon - 236

# Pramod Parajuli

## Simulation and Modeling, CS-331

---

### Markov Chains

## Markov Chains

A Markov chain is a sequence of random values whose probabilities at a time interval depends upon the value of the number at the previous time

A simple example is the nonreturning random walk, where the walkers are restricted to not go back to the location just previously visited

In this case, each possible position is known as state or condition

The controlling factor in a Markov chain is the ***transition probability***, it is a conditional probability for the system to go to a particular new state, given the current state of the system

(C) Pramod Parajuli, 2004

# Markov Chains

Since the probability of moving from one state to another depends on probability of the preceding state, transition probability is a conditional probability

*(blackboard example)*

Markov chain (process) has following properties;

1. The set of all possible states of stochastic (probabilistic) system is finite
2. The variables move from one state to another and the probability of transition from a given state is dependent only on the present state of the system, not in which it was reached
3. The probabilities of reaching to various states from any given state are measurable and remain constant over a time (i.e. throughout the system's operation)

# Markov Chains

Markov chains are classified by their order

- If the probability occurrence of each state depends only upon the immediate preceding state, then it is known as ***first order Markov chain***
- The zero order Markov chain is memory less chain

## Matrix of Transition Probabilities

Let  $s_j$ , ( $s_1, s_2, \dots, s_m$ ;  $j = 1, 2, \dots, m$ ) be state of a system and

$p_{ij}$ , ( $p_{0,0}, p_{0,1}, p_{0,2}, \dots, p_{m,m}$ ) be probability of moving from state  $s_i$  to state  $s_j$

So now, the square matrix of size  $m \times m$

$$P = [p_{ij}]_{m \times m} = \begin{array}{c} & & & \text{Succeeding state} \\ & & & \begin{matrix} s_1 & s_2 & \dots & \dots & s_m \end{matrix} \\ \begin{matrix} s_1 \\ s_2 \\ \dots \\ \dots \\ s_m \end{matrix} & \left[ \begin{matrix} p_{11} & p_{12} & \cdot & \cdot & p_{1m} \\ p_{21} & p_{22} & \cdot & \cdot & p_{2m} \\ \dots & \dots & \dots & \cdot & \dots \\ \dots & \dots & \dots & \cdot & \dots \\ p_{m1} & p_{m2} & \cdot & \cdot & p_{mm} \end{matrix} \right] \end{array}$$

(C) Pramod Parajuli, 2004

## Matrix of Transition Probabilities

If there is no transition between  $s_i$  and  $s_j$ , then  $p_{ij} = 0$

If only one state is selected while advancing, then

$$p_{ij} = 1$$

The probability is distributed over the elements in the row. Therefore;

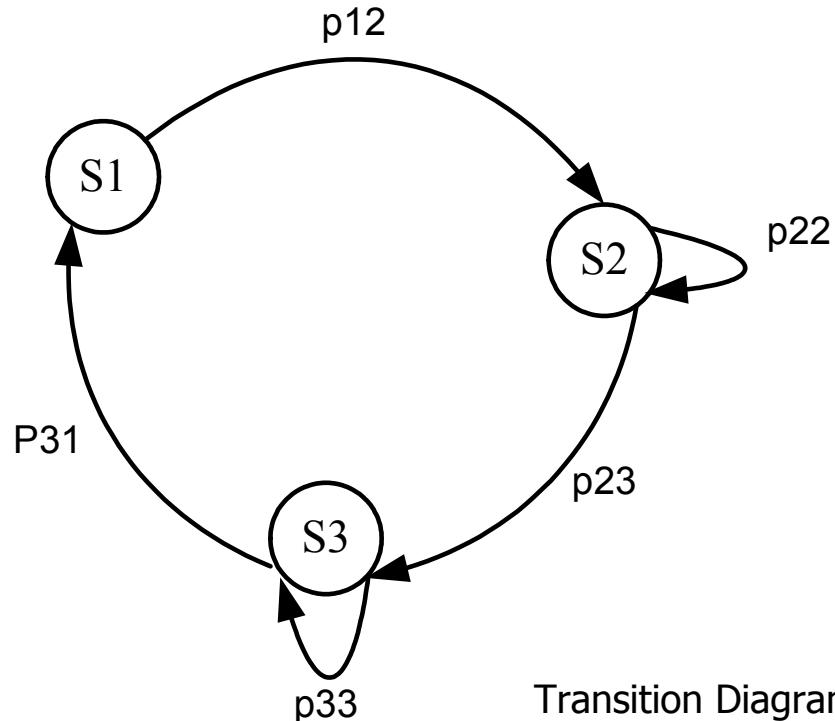
$$\sum_{j=1}^m p_{ij} = 1 \text{ for all } i$$

and  $0 \leq p_{ij} \leq 1$

# Diagrams

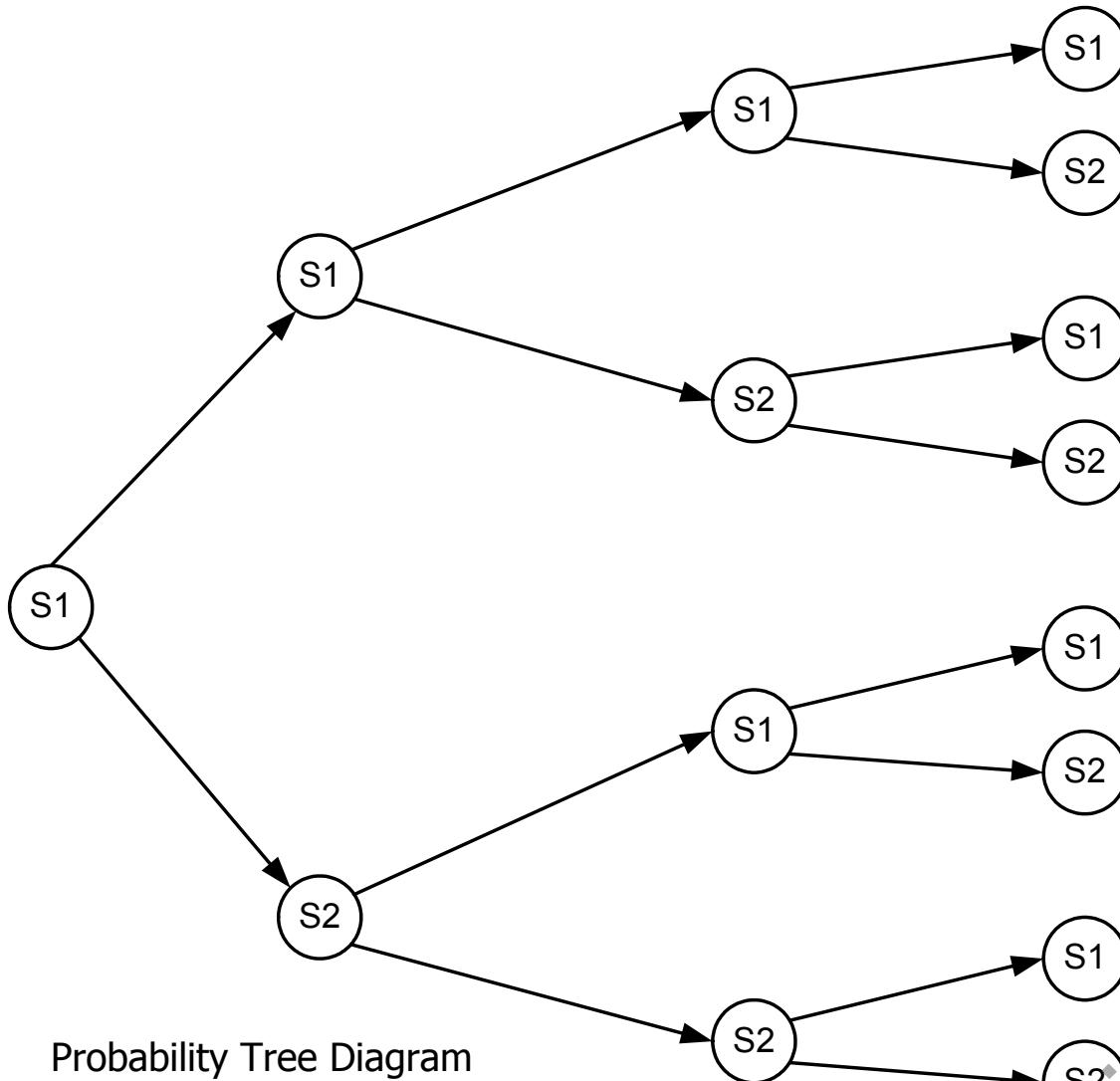
Two types;

1. *Transition Diagram*
2. *Probability Tree Diagram*



		Succeeding state		
		<b><math>S_1</math></b>	<b><math>S_2</math></b>	<b><math>S_3</math></b>
<b><math>S_1</math></b>	0	$P_{12}$	0	
<b><math>S_2</math></b>	0	$P_{22}$	$P_{23}$	
<b><math>S_3</math></b>	$P_{31}$	0	$P_{33}$	

# Diagrams



(C) Pramod Parajuli, 2004

## Diagrams

The probabilities at various states can be determined by using;

If state1 is the initial state;

$$\begin{aligned} P(\text{state 1, shift } n+1 \mid \text{state 1, shift 1}) \\ = 0.7 * P(\text{state 1, shift } n \mid \text{state 1, shift 1}) + \\ 0.8 * P(\text{state2, shift } n \mid \text{state 1, shift 1}) \end{aligned}$$

$$n = 1, 2, 3$$

Ex;

$$\begin{aligned} P(\text{state 1, shift 3} \mid \text{state 1, shift 1}) \\ = 0.7 (0.7) + 0.8 (0.3) = 0.73 \end{aligned}$$

## Diagrams

$$\begin{aligned} P(\text{state 1, shift 4} \mid \text{state 1, shift 1}) \\ = 0.7 P(\text{state 1, shift 3} \mid \text{state 1, shift 1}) \\ + 0.8 P(\text{state 1, shift 3} \mid \text{state 1, shift 1}) \\ = 0.727 \end{aligned}$$

$$\begin{aligned} P(\text{state 1, shift 4} \mid \text{state 1, shift 1}) \\ = 0.7 (0.7) (0.7) + 0.7 (0.3)(0.8) + 0.3 (0.8) \\ (0.7) + 0.3 (0.2)(0.8) \\ = 0.727 \end{aligned}$$

## n-Step Transition Probabilities

Let's represent the initial situation by  $R_0$  as;

$$R_0 = [p_{11}, p_{12}, p_{13}, \dots, p_{1m}]$$

And  $P = [p_{ij}]_{m \times m}$  be transition probabilities matrix  
at time period  $n = 0$

Further, let  $R_1$  represent the situation after one  
execution of the experiment i.e.  $n = 1$

$$R_1 = R_0 \times P$$

Similarly,

$$R_1 = R_0 \times P = R \times P^2$$

.

.

.

.

$$R_n = R_{n-1} \times P = R_0 \times P^n$$

## n-Step Transition Probabilities

$$R_n = R_{n-1} \times P = R_0 \times P^n$$

Here,  $P^n$  denotes the n-steps transition matrix

Sample calculation – Blackboard demonstration

1. Example 17.1 from page 716 (handouts)
2. Samples from PC Quest

## **Steady-state (equilibrium conditions)**

When the number of periods (stage) increases, the probabilities approaches to steady state (equilibrium). At this state, the system becomes independent of time

A Markov chain reaches to equilibrium state if;

1. The transition matrix elements remains positive from one period to the next (regular property of Markov chain)
2. It is possible to go from one state to another in a finite number of steps, regardless of the present state (ergodic property)

Note: All regular Markov chains must be ergodic Markov chains but the converse is not true.

# Steady-state behavior of Markovian Systems

- Steady state solution solved mathematically

## Inifinite – Population

- Are assumed to follow Poisson process with  $\lambda$  arrivals per unit time and inter-arrival time is exponentially distributed with mean  $1/\lambda$
- The queue discipline is FIFO
- Mathematically, a system is said to be in steady state, provided the probability that the system is in a given state is not time dependent;

$$P(L(t) = n) = P_n(t) = P_n$$

## Steady-state behavior of Markovian Systems

- For simple models, steady-state behavior parameter 'L' (time-average of customers in the system) can be computed as;

$$L = \sum_{n=0}^{\infty} n.P_n$$

Ref. Lecture 8, Slide – 46

- If L is given, then other steady-state parameters can be computed by using Little's equation;

$$L = \lambda \cdot w$$

$$w_Q = w - (1/\mu)$$

$$L_Q = \lambda \cdot w_Q$$

# M/G/1

Single server queues with poisson arrivals and unlimited capacity

Mean service time =  $1/\mu$

Variance =  $\sigma^2$

If  $\mu < 1$ , then M/G/1 queue has a steady-state probability distribution

If  $\lambda < \mu$ , then  $\rho$  will be server utilization

$$\rho = \frac{\lambda}{\mu}$$

$$L = \rho + \frac{\rho^2(1+\sigma^2\mu^2)}{2(1-\rho)}$$

$$L_Q = \frac{\rho^2(1+\sigma^2\mu^2)}{2(1-\rho)}$$

Single server queues with poisson arrivals and unlimited capacity

Let's look at these a bit more closely

Consider first if  $\sigma^2 = 0$

i.e. the service times are all the same (= mean)

In this case the equations for L and  $L_Q$  greatly simplified to

$$L_Q = \frac{\rho^2(1 + 0^2\mu^2)}{2(1 - \rho)} = \frac{\rho^2}{2(1 - \rho)}$$

In this case  $L_Q$  is dependent solely upon the server utilization,  $\rho$

Note as  $\rho \rightarrow 0$  (low server utilization)  $L_Q \rightarrow 0$

Note as  $\rho \rightarrow 1$  (high server utilization)  $L_Q \rightarrow \infty$

# **M/G/1**

Single server queues with poisson arrivals and unlimited capacity

Again,  $\rho = L - L_Q$  is the time average number of customers being served

Example;

Supplements from Banks and Nicol, Page 226, 227

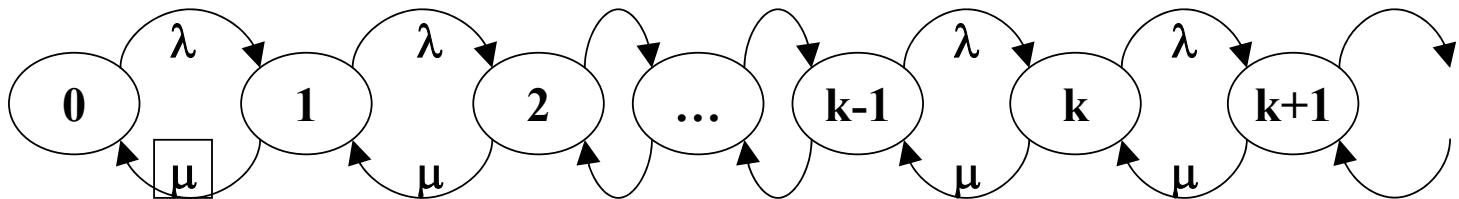
## M/G/1 (service times are exponential)

- Let's look at the simplest case, an M/M/1 queue with arrival rate  $\lambda$  and service rate  $\mu$
- Consider the state of the system to be the number of customers in the system
- We can then form a state-transition diagram for this system
  - A transition from state  $k$  to state  $k+1$  occurs with probability  $\lambda$
  - A transition from state  $k+1$  to state  $k$  occurs with probability  $\mu$

# M/M/1

M/G/1 (service times are exponential)

► From this we can obtain



► We also know that

$$P_k = P_0 \prod_{i=0}^{k-1} \frac{\lambda}{\mu} = P_0 \left( \frac{\lambda}{\mu} \right)^k$$

- Since the sum of the probabilities in the distribution must equal 1
- This will allow us to solve for  $P_0$

$$\sum_{k=0}^{\infty} P_k = 1$$

## M/M/1

M/G/1 (service times are exponential)

Before completing the derivation,  $\sum_{k=0}^{\infty} P_k = 1$   
we must note an important  
requirement: to be stable,  
the system utilization  
 $\lambda/\mu < 1$  must be true

$$\sum_{k=0}^{\infty} P_0 \left( \frac{\lambda}{\mu} \right)^k = 1$$

$$P_0 \left( 1 + \sum_{k=1}^{\infty} \left( \frac{\lambda}{\mu} \right)^k \right) = 1$$

$$P_0 = \frac{1}{1 + \sum_{k=1}^{\infty} \left( \frac{\lambda}{\mu} \right)^k}$$

## M/M/1

M/G/1 (service times are exponential)

$$P_0 = \frac{1}{\left(1 + \frac{\lambda/\mu}{1-(\lambda/\mu)}\right)} = \frac{1}{\left(\frac{1-(\lambda/\mu)}{1-(\lambda/\mu)} + \frac{\lambda/\mu}{1-(\lambda/\mu)}\right)}$$

$$P_0 = \frac{1}{\left(\frac{1}{1-(\lambda/\mu)}\right)} = 1 - \frac{\lambda}{\mu}$$

- Which is the solution for  $P_0$  from the M/G/1 Queue in Table 6.3
- ▶ Utilizing these, we can substitute back to get

$$P_k = P_0 \left(\frac{\lambda}{\mu}\right)^k = \left(1 - \frac{\lambda}{\mu}\right) \left(\frac{\lambda}{\mu}\right)^k = (1 - \rho) \rho^k$$

- Which is the formula indicated in Table 6.4
- The other values can also be derived in a similar manner

# **Homework**

Read about 'Steady state behavior for finite population model' and write an article about it.

Deadline – February 14, 2005