LOVELY PROFESSIONAL UNIVERSITY

SIMULATION AND MODELLING

Edited By

Sandeep Kumar

Printed by
EXCEL BOOKS PRIVATE LIMITED
A-45, Naraina, Phase-I,
New Delhi-110028
for
Lovely Professional University
Phagwara

SYLLABUS

Simulation and Modelling

Objectives: This course provides an introduction to system modelling using both computer simulation and mathematical techniques. The objective of this course is:

- To teach students methods for modelling systems.
- To emphasize of the course will be on modelling
- To use of simulation software

Sr. No.	Topics
1.	**Introduction**: Simulation of a pure-pursuit problem, System and its model, Simulation of an inventory problem, basic nature of simulation, When to simulate
2.	**Simulation of Continuous systems**: A chemical reactor, numerical integration Vs Continuous system simulation, Selection of integration formulas
3.	Simulation of a servo system, Simulation of water reservoir system, Analog Vs Digital Simulation
4.	**Discrete System Simulation**: Fixed time-step vs event-to-event model, Simulating randomness, generation of random numbers
5.	**Discrete System Simulation**: Generation of non-uniformly distributed random numbers, monte-carlo Vs stochastic simulation
6.	**Simulation of Queuing Systems**: Rudiments of queuing theory, Simulation of single-server queue, Simulation of two-server queue
7.	**Simulation of a PERT network**: Network model of a project, Analysis of an activity network, Critical path computation, Simulation of an activity network **Computer program for simulation**, resource allocation and cost considerations
8.	**Design and Evaluation of Simulation Experiments**: Length of simulation runs, variance reduction techniques, experimental layout, validation
9.	**Simulation Languages**: Continuous and Discrete system simulation languages, Continuous simulation languages
10.	**Simulation Languages**: Block-structured continuous simulation languages, Expression-based languages, Discrete-system simulation languages

CONTENT

Unit 1:	Introduction to System Simulation *Sandeep Kumar, Lovely Professional University*	1
Unit 2:	Simulation of Continuous System *Sandeep Kumar, Lovely Professional University*	33
Unit 3:	Simulation of Continuous System *Sandeep Kumar, Lovely Professional University*	43
Unit 4:	Discrete System Simulation *Sandeep Kumar, Lovely Professional University*	55
Unit 5:	Discrete System Simulation *Sandeep Kumar, Lovely Professional University*	68
Unit 6:	Discrete System Simulation *Sandeep Kumar, Lovely Professional University*	78
Unit 7:	Simulation of Queuing System *Nisha Sethi, Lovely Professional University*	96
Unit 8;	Simulation of Queuing System *Nisha Sethi, Lovely Professional University*	119
Unit 9:	Simulation of a PERT Network *Sandeep Kumar, Lovely Professional University*	139
Unit 10:	Simulation of a PERT Network *Sandeep Kumar, Lovely Professional University*	169
Unit 11:	Design and Evaluation of Simulation Experiments *Sandeep Kumar, Lovely Professional University*	186
Unit 12:	Design and Evaluation of Simulation Experiments *Sandeep Kumar, Lovely Professional University*	201
Unit 13:	Simulation Languages *Nisha Sethi, Lovely Professional University*	236
Unit 14:	Simulation Languages *Nisha Sethi, Lovely Professional University*	267

Unit 1: Introduction to System Simulation

Notes

CONTENTS

Objectives

Introduction

1.1 Modelling and Simulation Process

1.2 Simulation of a Pure-pursuit Problem

1.3 System and its Model

1.3.1 Definitions of Systems and Models

1.3.2 Some Reflections on Models

1.4 Types of Models

1.5 Simulation of an Inventory Problem

1.6 Basic Nature of Simulation

1.7 When to Simulate

1.8 Summary

1.9 Keywords

1.10 Self Assessment

1.11 Review Questions

1.12 Further Readings

Objectives

After studying this unit, you will be able to:

- Understand the meaning, system and its model
- Describe basic nature of simulation
- Explain when to simulate
- Discuss the simulation of Pure-pursuit problem
- Explain the simulation of inventory problem

Introduction

Simul8 Standard is an incorporated environment for functioning with simulation models. The influential language and model visualization potential facilitate you to create the accurate, flexible and robust simulations you require in less time. Simul8 is used in nearly all Six Sigma process enhancement initiatives.

Simul8 comes with everything you require to begin building simulations straight away. The context perceptive Help make it simple to discover answers to questions about **Simul8** features. You can get simulation aid from Simul8's distinctive simulation supporter.

Remaining competitive in today's worldwide economy means delivering superior quality services and products at enhanced prices. Your consumers demand nonstop improvement, lower costs and shorter delivery times. **Simul8** Corporation offers you with a tool that assists you

Notes

squeeze up your processes to fulfill those demands. Out-deliver your competition by means of **Simul8** to plan, model, authenticate, and converse better methods of getting the job done.

This unit discovers the use of modelling and simulation as a problem resolving tool. We commence this discussion within the structure of a modelling and simulation project. This project framework cuddles two key ideas; first there is the notion of a 'system context'; that is, there is a system that has been recognized for examination, and second, there is a difficulty relating to the identified system that requires to be solved.

The defining feature of the modelling and simulation approach is that it is founded on a computer-based experimental investigation that utilises an appropriate model for the SUI. The model is a representation or abstraction of the system. The use of models (in particular, mathematical models) as a basis for analysis and reasoning is well established in such disciplines as engineering and science. It is the emergence and widespread availability of computing power that has made possible the new dimension of experimentation with complex models and hence, the emergence of the modelling and simulation discipline.

1.1 Modelling and Simulation Process

An outline of the essential steps involved in carrying out a modelling and simulation study is provided in the discussion that follows. Although the initial steps can be effectively presented using various notions that have been previously introduced, there are several aspects of the latter stages that require extensive elaboration. This is provided in the discussions that follow. An overview of the process is provided in Figure 1.1.

The overview of Figure 1.1 does not include a preliminary phase during which solution alternatives for the problem are explored and a decision is made to adopt a modelling and simulation approach. We note that the option of also carrying out other complementary approaches is entirely reasonable and is often prudent for some portions of the problem. Although this preliminary phase is not explicitly represented in Figure 1.1 its existence and importance must nevertheless be recognised.

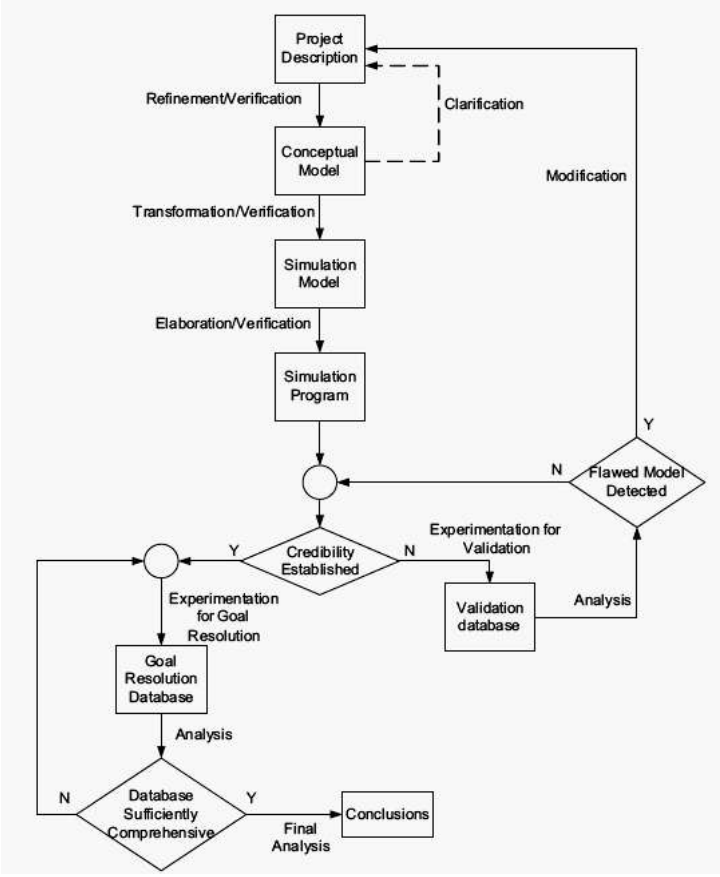
It should be emphasised that a modelling and simulation project of even modest size is often carried out by a team of professionals where each member of the team typically contributes some special expertise. There is, therefore, a need for communication among team members. Some facets of the discussion have their basis in this communication requirement.

Did u know? **Meaning of Modelling and Simulation**

Although the word 'modelling' has a meaning that is reasonably confined in its general usage, the same cannot be said for the word 'simulation'. Nevertheless, the phrase 'modelling and simulation' does have a generally accepted meaning and implies two distinct activities. The modelling activity creates an object (i.e., a model) that is subsequently used as a vehicle for experimentation. This experimentation with the model is the simulation activity.

Notes

Figure 1.1: Modelling and Simulation Process



Project Description

The process begins with the preparation of a document called the project description. This document includes a statement of the project goal(s) and a description of those behavioural features of the SUI that have relevance to the goals. These behaviour features are typically formulated in terms of the various entities that populate the space that the SUI embraces with particular focus on the interactions among these entities. It is, more or less, an informal description in as much as it relies mainly on the descriptive power of natural language. The language is, furthermore, often heavily coloured with jargon associated with the SUI. This jargon may not be fully transparent to all members of the project team and this can contribute to both an inherent lack of precision and, as well, to communication problems.

With few exceptions, the SUI also has structural features that provide the context for the interactions among the entities (e.g., the layout of the pumps at a gas station or the topology of the network of streets being serviced by a taxi company). Informal sketches are often the best means of representing these structural features. These are an important part of the presentation because they provide a contextual elaboration that can both facilitate a more precise statement of the project goals and as well, help to clarify the nature of the interaction among the entities. Because of these contributions to understanding, such sketches are often a valuable component of the project description.

Notes

Conceptual Model

The information provided by the project description is, for the most part, unstructured and relatively informal. Because of this informality it is generally inadequate to support the high degree of precision that is required in achieving the objective of a credible model embedded within a computer program. A refinement phase must be carried out in order to add detail where necessary, incorporate formalisms wherever helpful, and generally enhance the precision and completeness of the accumulated information. Enhanced precision is achieved by moving to a higher level of abstraction than that provided by the project description. The reformulation of the information within the project description in terms of parameters and variables is an initial step because these notions provide a fundamental means for removing ambiguity and enhancing precision. They provide the basis for the development of the simulation model that is required for the experimentation phase.

There is a variety of formalisms that can be effectively used in the refinement process. Included here are mathematical equations and relationships (e.g., algebraic and/or differential equations), symbolic/graphical formalisms (e.g., Petri nets, finite state machines), rule based formalisms, structured pseudo code, and combinations of these. The choice depends on suitability for providing clarification and/or precision. The result of this refinement process is called the conceptual model for the modelling and simulation project. The conceptual model may, in reality, be a collection of partial models each capturing some specific aspect of the SUI's behaviour. The representations used in these various partial models need not be uniform.

The conceptual model is a consolidation of all relevant structural and behavioural features of the SUI in a format that is as concise and precise as possible. It provides the common focal point for discussion among the various participants in the modelling and simulation project. In addition, it serves as a bridge between the project description and the simulation model that is essential for the experimentation activity (i.e., the simulation phase). As we point out below, the simulation model is a software product and its development relies on considerable precision in the statement of requirements. One of the important purposes of the conceptual model is to provide the prerequisite guidance for the software development task.

In Figure 1.1, a verification activity is associated with the transition from the project description to the conceptual model. As will transition under consideration because it involves a reformulation of the key elements of the model from one form to another and the integrity of this transformation needs to be confirmed.

In the modelling and simulation literature, the phrase 'conceptual model' is frequently reduced simply to 'model'. Our usage of the word 'model' without a modifier generally implies a composite notion that program successors where the latter two notions are described in the discussion that follows.

Notes It is worth pointing out that there is by no means a common understanding in the modelling and simulation literature of the nature and role of a conceptual model. The overview presented by Robinson gives considerable insight into the various perspectives that prevail.

Simulation Model

The essential requirement for the experimentation phase of a modelling and simulation project is an executable computer program that embodies the conceptual model. It evolves from a transformation of the conceptual model into a representation that is consistent with the syntax

Notes

and semantic constraints of some programming language. This program is the simulation model for the project. It is the execution of this program (or more correctly), Enhanced version of it; see following and simulation project is an executable computer program that embodies the system under investigation. The solution to the underlying problem is the simulation model for the project. It is the execution of this program that is embedded in the project goal(s) is obtained from the data reflected in this behaviour. As will the conceptual model. It evolves from a transformation of a conceptual model and its simulation model and simulation appeared in recent years; some examples are: SIMSCRIPT II.5, MODSIM, GPSS, SIMAN, ACSL, Modelica, Arena, CSIM, and SIMPLE ++. Such languages generally provide features to support the management of time, collection of data, and presentation of required output information. In the case of projects in the DEDS domain, additional features for the generation of random variates, management of queues, and the statistical analysis of data are also provided.

The simulation model is the penultimate stage of a development process that began with the decision to formulate a modelling and simulation project to resolve an identified problem. The Simulation model is a software product and as such, the process for its development shares many of the general features that characterise the development of any software product.

Notes In the Figure 1.1 the transition from the conceptual model to the simulation model is associated with two activities: namely, transformation and verification. As in the earlier transition from project description to conceptual model, verification is required here to confirm that the transformation has been correctly carried out.

Simulation Program

The outline of the simulation model provided above is idealised inasmuch as it suggests that the simulation model is directly capable of providing the behaviour-generating mechanism for the simulation activity. In reality this program code segment is never self-sufficient and a variety of auxiliary services must be superimposed. The result of augmenting the simulation model with complementary program infrastructure that provides these essential functional services is the simulation program.

The services in question fall into two categories: one relates to fundamental implementation issues whereas the other is very much dependent on the nature of the experiments that are associated with the realisation of the project goals. Included within the first category are such basic tasks as initialisation, control of the observation interval, management of stochastic features (when present), solution of equations (e.g., the differential equations of a continuous system model), data collection, and so on. Convenient programming constructs to deal with these various tasks are normally provided in software environments specifically designed to support the simulation activity. But this is certainly not the case in general-purpose programming environments where considerable additional effort is often required to provide these functional requirements.

The second category of functional services can include such features as data presentation (e.g., visualisation and animation), data analysis, database support, optimisation procedures, and the like. The extent to which any particular modelling and simulation project requires services from this second category can vary widely. Furthermore, modelling and simulation software environments provide these services only to varying degrees and consequently, when they are needed; care must be taken in choosing an environment that is able to deliver the required services at an adequate level.

Notes

The manner in which the support services to augment the simulation model are invoked varies significantly among software environments. Almost always there is at least some set of parameters that need to be assigned values in order to choose from available options. Often some explicit programming steps are needed. Considerable care must be taken when developing the simulation program to maintain a clear demarkation between the code of the simulation model and the code required to invoke the ancillary services. Blurring this separation can be detrimental because the resulting simulation program may become difficult to verify, understand, and/or maintain. It has, in fact, been frequently noted (e.g., Oren [2.10]) that an important quality attribute of a simulation software platform is the extent to which it facilitates a clear separation of the code for the simulation model from the infrastructure code required for the experimentation that is required for the achievement of the project goal(s).

Figure above indicates that a verification activity needs to be carried out in the transition from the simulation model to the simulation program. This need arises because this transition typically involves a variety of decisions relating to the execution of the simulation model and the correctness of these decisions must be confirmed. Consider, for example, a simulation model that incorporates a set of ordinary differential equations. Most modelling and simulation programming environments offer a variety of solute on methods for such equations and each has particular strengths and possibly weaknesses as well. If the equations in question have distinctive properties, then there exists a possibility of an improper choice of solution method. The verification process applied at this stage would uncover the existence of such a flaw when it exists.

Tasks Analyze the categories that occur in the functional services of simulation model.

Operational Phases

Thus far our outline of the modelling and simulation process has focused on the evolution of a series of interdependent representations of SUI. However, with the existence of the simulation program, the stage is set for two operational phases of the process that we now examine. The first of these is the validation phase whose purpose is to establish the credibility of each of the model realisations, from the perspective of the project goals.

The second phase, which can begin only after the model's credibility has been established, is the experimentation phase, or more specifically, the simulation phase. This activity is presented in Figure 1.1 as the task of 'goal resolution'. This is achieved via a sequence of experiments with the simulation program during which an ever-increasing body of data is collected and analysed until it is apparent that a 'goal resolution database' is sufficiently complete and comprehensive to permit conclusions relating to the goal(s) to be confidently formulated.

Tasks Analyze the operational phases that appear with the existence of simulation program.

1.2 Simulation of a Pure-pursuit Problem

Pure pursuit is a tracking algorithm that functions by scheming the curvature that will shift a vehicle from its existing position to some objective position. The entire point of the algorithm

Notes

is to select a goal position that is some distance forward of the vehicle on the path. Pure pursuit is a type of pursuit curve used in aerial combat in which an aircraft pursues another aircraft by pointing its nose directly towards it. This is in contrast with lead pursuit, in which the chasing aircraft points ahead of the aircraft it is following (typically used when attempting a gun attack) and lag pursuit, in which the chasing aircraft points behind the aircraft it is following (typically used when attempting a rear-aspect missile attack. The name pure pursuit occurs from the resemblance that we use to portray the method. We have a tendency to think of the vehicle as chasing a point on the path some distance ahead of it - it is chasing that moving point. That analogy is frequently used to evaluate this method to the manner in which humans drives. We have a propensity to look some distance in front of the car and head in the direction of that spot. This look ahead distance varies as we drive to reflect the twist of the road and visualization occlusions.

Pursuit Algorilthm

The execution of the pure pursuit algorithm itself is quite simple. The pure pursuit algorithm can be summarized as below:

1. Establish the current location of the vehicle.
2. Locate the path point closest to the vehicle.
3. Locate the goal point
4. Convert the goal pole to vehicle coordinates.
5. Compute the curvature and appeal the vehicle to set the routing to that curvature.
6. Update the vehicle's location.

1. ***Establish the current location of the vehicle.*** Both the HMMWV and NavLab have a central vehicle organizer that offers functions which account the vehicle's current location as (x,y,heading). The position is reposed with respect to the vehicle's location at initialization time. This original location is the global reference frame for the run.
2. ***Locate the path point closest to the vehicle.*** In the geometric origin it was confirmed that the goal point would be inside one look ahead distane of the vehicle. It is probable that there are numerous points one look ahead distance from the vehicle's existing location. The vehicle should steer toward the closest point one look ahead distance from its existing location. So, the path point closest to the vehicle will first be establish, and the search for a point 1 lookahead distance away from the vehicle will start at this point and initiate up the path.
3. ***Locate the goal point.*** The goal point is set up by moving up the path and computing the distance between that path point and the vehicle's existing location. Path point positions are recorded in the global frame; this computation is done in worldwide coordinates.
4. ***Convert the goal point to vehicle coordinates.*** Once the goal point has been establish, it must be converted to the vehicle's local coordinates. The derivation for the curvature was performed in vehicle coordinates and curvature commands to the vehicle make sense in vehicle. coordinates.
5. ***Compute the curvature.*** Using the curvature equation, compute the desired vehicle curvature. The curvature is converted into steering wheel angle by the vehicle's on board controller.
6. ***Update the vehicle's location.*** During simulation, it is essential to determine what effects the command has upon the vehicle's location and heading.

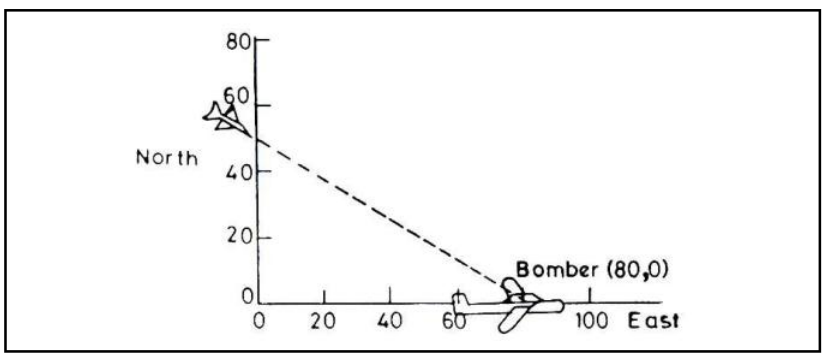
Notes

Example: Suppose a fighter aircraft sights an enemy bomber and flies directly toward it, in order to catch up with the bomber and destroy it. Here the target is the bomber, it continues flying along a specified curve so the fighter, which is the pursuer, has to change its direction to keep pointed toward the target. Now you are concerned in determining the attack course of the fighter and in knowing how long it would take for it to catch up with the bomber.

If the bomber flies along a straight line, the problem can be solved directly with analytic techniques. On the other hand, if the path of the bomber is curved, then the problem is much more complicated and generally cannot be solved directly. To solve this curved path problem you will use simulation under the following simplifying conditions:

1. The bomber and the fighter are flying in the same horizontal plane when the fighter first sights the bomber, and both stay in that plane. This makes the pursuit model two-dimensional.
2. The fighter's speed VF is constant (20 kms/minute).
3. The bomber's path (i.e., its position as a function of time) is specified.
4. After a fixed time span Δt the fighter changes its direction in order to point itself toward the bomber.

First we introduce a rectangular coordinate system, which is coincident with the horizontal plane in which the two aircraft are flying. Let we choose the point due south of the fighter and due west of the bomber at the beginning of the pursuit as the origin of this coordinate system. Let the distances be given 1(45-123/77) in kilometers and the time in minutes. We start measuring the time when the fighter first sights the bomber.



Now we will represent the path of the bomber which is known to us in advance by two arrays, the east coordinates and the north coordinates at specified moments or we can say each minute. We call these coordinates XB(t) and YB (t), respectively. They are presented in the form of a table (in kirometers) below.

Time, t	0	1	2	3	4	5	6	7	8	9	10	11	12
$XB(t)$	80	90	99	108	116	125	133	141	151	160	169	179	180
$YB(t)$	0	−2	−5	−9	−15	−18	−23	−29	−28	−25	−21	−20	−17

Likewise, we will represent the path of the fighter plane by two arrays XF(t) and YF(t). In this example, initially we are given

YF(O) = 50 kms, XF(O) = 0 kms.

Notes

Our purpose is to compute the positions of the pursuer, namely, XF(t), YF(t) for t = 1,2, ... , 12, or until the fighter catches up with the bomber.

Suppose we will assume that once the fighter is within IO kms of the bomber, the fighter shoots down its target by firing a missile, and the pursuit is over. If case the target is not caught up within 12 minutes, the pursuit is abandoned, and the target is considered escaped. From the time t = 0 till the target is shot down, the attack course is determined as follows:

The fighter uses the following simple strategy: It looks at the target at instant t, aligns its velocity vector with the line of sight means points itself toward the target. It continues to fly in that direction for one minute, till instant (t + 1). At time (t + 1) it looks at the target again and realigns itself.

The distance DIST (t) at a given time t between the bomber and the fighter is given by

$$\text{DIST (t)} = \sqrt{(\text{YB(t)} - \text{YF(t)})^2 + (\text{XB)(t)} - \text{XF(t)})^2} \quad ...(1)$$

The angle θ of the line from the figher to the target at a given time t is given by

$$\sin\theta = \frac{\text{YB(t)} - \text{YF(t)}}{\text{DIST(t)}} \quad ...(2)$$

$$\cos\theta = \frac{\text{XB(t)} - \text{XF(t)}}{\text{DIST(t)}} \quad ...(3)$$

Using this value of the position of the fighter at time (t + 1) is determined by

$$\text{XF(t + 1)} = \text{XF(t)} + \text{VF} \cos\theta \quad ...(4)$$

$$\text{YF(t + 1)} = \text{YF(t)} + \text{VF} \sin\theta \quad ...(5)$$

With these new coordinates of the pursuer, its distance from the target is again computed using Eq. (1). If this distance is 10 kms. or less the pursuit is over, otherwise q is recomputed, and the process continues.

A flowchart of the logic of this program is given below:

The following FORTRAN program (a format-free version) will implement the flowchart.

```
        DIMENSION XB (25), YB (25), XF (25), YF (25)
        INTEGER T, J
        READ, (XB (T), YB (T), T = 1,13)
        READ, XF (I), YF (I), VF
        T=1
100     DIST = SQRT ((YB (T) - YF (T)) **2 + (XB (T) - XF (T)** 2)
        IF (DIST. LE. 10.0) GO TO 110
        IF (T.GT.12) GO TO 120
        XF (T + I) = XF(T) + VF* (XB(T) – XF (T))/DIST
        YF (T + 1) = YF(T) + YF* (YB (T) – YF (T))/DIST
        T = T+1
        GO TO 100
110     PRINT 990, T, DIST
990     FORMAT (10X, 10H CAUGHT AT, 13, 8H MTS AND, F10.3, 4H KMS)
        STOP
120     PRINT 1000
```

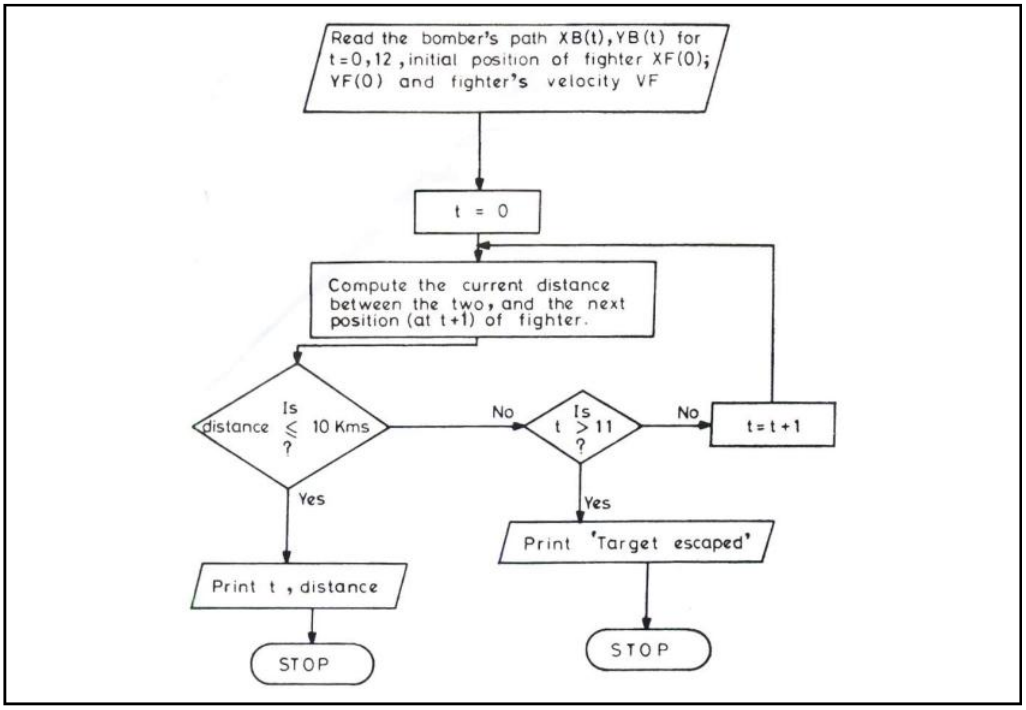
Notes

```
1000    FORMAT (l0X, 16H TARGET ESCAPED) STOP
        END
```

(Note that since Fortran does not permit 0 as an index we had to set T = 1 to correspond to t = 0 in the flowchart.)

The output of this program for the specified data is as fol1ows:

CAUGHT AT 10 MTS AND 2.969 KMS



By simulation you are able to make the computer go through the instant-to-instant predictions for as many instants as you required. This was possible only because we knew the basic process involved, namely, how the fighter plane behaves at any particular instant. Such knowledge of the basic process of the system under study is essential for all simulation experiments.

The simple strategy, of fighter redirecting himself toward the bomber at fixed intervals of time, while the target goes on its predetermined path without making any effort to escape the fighter, is called pure pursuit. Although in many situations, the strategy used by the fighter (pursuer) is more sophisticated, this basic approach can be used for any pursuit problem.

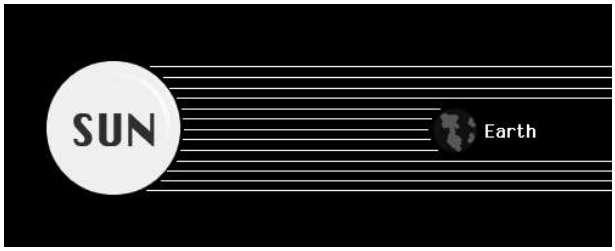
1.3 System and its Model

1.3.1 Definitions of Systems and Models

Systems are often visualized or modeled as component blocks that have connections drawn between them. For example, the illustration below describes the interception of solar radiation by the Earth. In this system, the Earth and Sun, the parts or component blocks, are represented by two colored circles of different size. The process of solar emission and the interception of the Sun's emitted radiation by the Earth (the connection) is illustrated by the drawn lines.

Notes

Figure 1.2: Simple Visual Model of Solar Radiation being Emitted from the Sun and Intercepted by the Earth



Did u know? **Definition of System**

A system is an assemblage of interrelated parts that work together by way of some driving process

Characteristics of a System

Most systems share the same common characteristics. These common characteristics include the following:

1. Systems have a structure that is defined by its parts and processes.
2. Systems are generalizations of reality.
3. Systems tend to function in the same way. This involves the inputs and outputs of material (energy and/or matter) that is then processed causing it to change in some way.
4. The various parts of a system have functional as well as structural relationships between each other.
5. The fact that functional relationships exist between the parts suggests the flow and transfer of some type of energy and/or matter.
6. Systems often exchange energy and/or matter beyond their defined boundary with the outside environment, and other systems, through various input and output processes.
7. Functional relationships can only occur because of the presence of a driving force.
8. The parts that make up a system show some degree of integration – in other words the parts work well together.

Properties of a System

Within the boundary of a system we can find three kinds of properties:

1. ***Elements*** are the kinds of parts (things or substances) that make up a system. These parts may be atoms or molecules, or larger bodies of matter like sand grains, rain drops, plants, animals, etc.
2. ***Attributes*** are characteristics of the elements that may be perceived and measured. For example: quantity, size, color, volume, temperature, and mass.
3. ***Relationships*** are the associations that occur between elements and attributes. These associations are based on cause and effect.

Notes

Notes We can define the state of the system by determining the value of its properties (the elements, attributes, and/or relationships).

Types of Systems

Scientists have examined and classified many types of systems. Some of the classified types include:

1. ***Isolated System:*** It is a system that has no interactions beyond its boundary layer. Many controlled laboratory experiments are this type of system.
2. ***Closed System:*** It is a system that transfers energy, but not matter, across its boundary to the surrounding environment. Our planet is often viewed as a closed system.
3. ***Open System:*** It is a system that transfers both matter and energy can cross its boundary to the surrounding environment. Most ecosystems are example of open systems.
4. ***Morphological System:*** This is a system where we understand the relationships between elements and their attributes in a vague sense based only on measured features or correlations. In other words, we understand the form or morphology a system has based on the connections between its elements. We do not understand exactly how the processes work to transfer energy and/or matter through the connections between the elements.
5. ***Cascading System:*** This is a system where we are primarily interested in the flow of energy and/or matter from one element to another and understand the processes that cause this movement. In a cascading system, we do not fully understand quantitative relationships that exist between elements related to the transfer of energy and/or matter.
6. ***Process-response System:*** This is a system that integrates the characteristics of both morphological and cascading systems. In a process-response system, we can model the processes involved in the movement, storage, and transformation of energy and/or matter between system elements and we fully understand how the form of the system in terms of measured features and correlations.
7. ***Control System:*** A system that can be intelligently manipulated by the action of humans.
8. ***Ecosystem:*** It is a system that models relationships and interactions between the various biotic and abiotic components making up a community or organisms and their surrounding physical environment.

There are four basic types of system depending on whether the parts and the whole can display choice, and therefore, be purposeful. The four kinds of system are shown in Table 1.1

Notes

Figure 1.3: Types of Systems

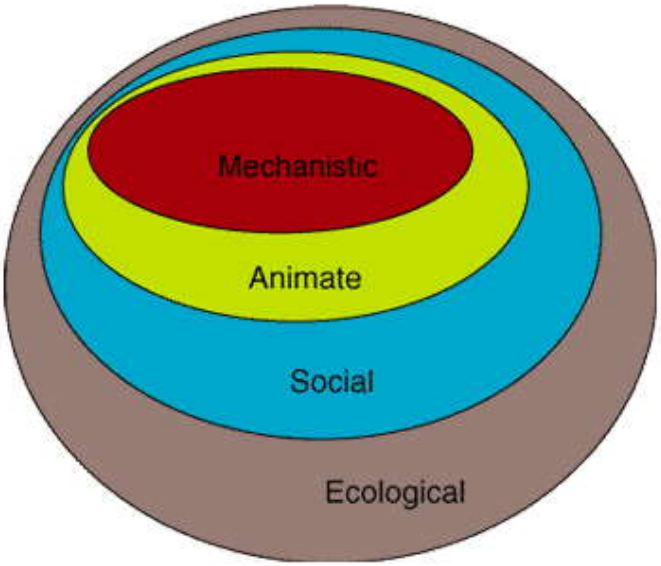


Table 1.1: Four Types of Systems

Types of System Model	Parts	Whole	Example
Mechanistic	No choice	No choice	Machines
Animate	No choice	Choice	Persons
Social	Choice	Choice	Corporations
Ecological	Choice	No Choice	Nature

These type form a hierarchy with ecological systems – the highest type. All but mechanistic systems can incorporate as parts other systems of the same or a lower type, but not of a higher type; for example, social systems (e.g., society) may incorporate animate systems (people) and mechanistic systems (machines), but a mechanistic system cannot incorporate either an animate or social system. Ecological systems can incorporate systems of all the other types.

Only animate and social systems can be said to be purposeful.

Now consider each type of system in a bit more detail.

1. ***Mechanistic Systems:*** Mechanistic systems and their parts have no purposes of their own, but their essential parts make possible the functioning of the whole. All mechanisms are mechanistic systems. Plants are also. Clocks are common examples of such systems; they operate with a regularity dictated by their internal structure and the causal laws of nature. Neither the whole nor the parts of a clock display choice, but they have functions. Similarly, an automobile is a mechanical system that has no purpose of its own but serves its driver's and passengers' purposes. In addition, an automobile's fuel pump (a mechanical system) has the function of supplying its fuel injector or carburetor with fuel, without which the automobile could not carry out its defining function.

Notes

Mechanistic systems are either open or closed, closed if their behavior is unaffected by any external conditions or events; open if they are so affected. The universe was conceptualized by Newton as a closed (self-contained) mechanical system, with no environment-like a hermetically sealed clock. On the other hand, the planet Earth is seen as an open system, one whose motion is influenced by other bodies in the solar system.

2. ***Animate Systems:*** These are conceptualized as purposeful systems whose parts have no purposes of their own. The principal purpose of such systems is survival. A person's lungs have no purpose of their own; but they function to enable a person to extract oxygen from the environment so as to survive. Animate systems are necessarily open; they must interact with their environments in order to survive Understanding these interactions are essential for understanding their properties and behavior.

   Animate systems are living systems. "Life" has been defined in many different Ways. The definition now most widely accepted by biologists involves the ways concept autopsies:

   "The maintenance of units and wholeness, while components themselves are being continuously or periodically disassembled and rebuilt, created and decimated, produced and consumed." (Zeleny, 1981, p. 5)

   From this definition it follows that social and ecological systems are also alive. (Many biologists are unhappy about this consequence of their definition of 'life.')

3. ***Social Systems:*** These are systems that (1) have purposes of their own, (2) consist of parts at least some of which are animate, hence have purposes of their own, and (3) are a part of one or more larger (containing) systems that may have purposes of their own and that may contain other social systems. For example, a local government viewed as a social system is part of a state government, which is also a social system, This, in turn, is part of a national government. Social systems can be and usually are nested.

4. ***Ecological Systems:*** Such systems contain mechanistic, animate, and social systems as parts and, therefore, containing some parts that have purposes of their own. However, these systems as a whole are conceptualized as having no purpose of their own. Nature, of course, is commonly taken to be an ecological system as is our environment.

Ecological systems serve the purposes of their animate and social parts, and provide necessary inputs to these and open deterministic systems. They also provide a receptacle for their waste as well as their useful products. Such service and support is their function. An ecological system can be affected mechanistically by the mechanical or purposeful behavior of its parts. For example, the purposeful use by people of fluoro-carbons as a propellant and the emissions of power plants affect the ozone layer mechanistically.

Caution Animate and social systems are frequently confronted with situations in which their choices can affect their effectiveness, either positively or negatively. Such situations are problematic. In other words, problems are situations in which a system's choice can make a significant difference to that system.

1.3.2 Some Reflections on Models

The use of models as a means of obtaining insight or understanding is by no means novel. One could reasonably claim, for example, that the pivotal studies in geometry carried out by Euclid were motivated by the desire to construct models that would assist in better understanding important aspects of his physical environment. It could also be observed that it is rare indeed for the construction of even the most modest of structures to be undertaken without some documented perspective (i.e., an architectural plan or drawing) of the intended form. Such a document

Notes

represents a legitimate model for the structure and serves the important purpose of providing guidance for its construction. Many definitions of a model can be found in the literature. One that we feel is especially noteworthy was suggested by Shannon. 'A model is a representation of an object, system or idea in some form other than itself.'

Although outside the scope of our considerations, it is important to recognise a particular and distinctive class of models called physical models. These provide the basis for experimentation activity within an environment that mimics the physical environment in which the problem originates. An example here is the use of scale models of aircraft or ships within a wind tunnel to evaluate aerodynamic properties; another is the use of 'crash-test dummies' in the evaluation of automobile safety characteristics. A noteworthy feature of physical models is that they can, at least in principle, provide the means for direct acquisition of relevant experimental data. However, the necessary instrumentation may be exceedingly difficult to implement.

A fundamental dichotomy among models can be formulated on the basis of the role of time; more specifically, we note that some models are dynamic whereas others are static. A linear programming model for establishing the best operating point for some enterprise under a prescribed set of conditions is a static model because there is no notion of time dependence embedded in such a model formulation. Likewise, the use of tax software to establish the amount of income tax payable by an individual to the government can be regarded as the process of developing.

Another important attribute of any model is the collection of assumptions that are incorporated into its formulation. These assumptions usually relate to simplifications and their purpose is to provide a means for managing the complexity of the model. Assumptions are invariably present but often they are not explicitly acknowledged and this can have very serious consequences. The assumptions embedded in a model place boundaries around its domain of applicability and hence upon its relevance not only to the project for which it is being developed but also to any other project for which its reuse is being considered.

Making the most appropriate choices from among possible assumptions can be one of the most difficult aspects of model development. The underlying issue here is identifying the correct balance between complexity and credibility where credibility must always be interpreted in terms of the goals of the project. It's worth observing that an extreme, but not unreasonable, view in this regard is that the development of any model is simply a matter of making the correct selection of assumptions from among the available options (often a collection of substantial size).

The assumptions embedded in a model are rarely transparent. It is therefore of paramount importance to ensure, via the documentation for the project, that all users of the model are cognisant of its limitations as reflected in the assumptions that underlie its development.

As might be expected, the inherent restricted applicability of any particular model as suggested above has direct and significant consequences upon the simulation activity. The implication is simply that restrictions necessarily emerge upon the scope of the experiments that can be meaningfully carried out with the model. This is not to suggest that certain experiments are impossible to carry out but rather that the value of the results that are generated is questionable. The phenomenon at play here parallels the extrapolation of a linear approximation of a complex function beyond its region of validity. The need to incorporate in simulation software environments a means for ensuring that experimentation remains within the model's range of credibility has been observed. Realisation of this desirable objective, however, has proved to be elusive.

Notes

Nature of a Model

A model is a specification for behaviour generation and the modelling process is concerned with the development of this specification. It is often suggested that the task is to ensure that the behaviour of the model is as indistinguishable as possible from the behaviour of the SUI. This assertion is only partially correct. A more appropriate statement of the task at hand is to develop the specification so that it captures the behaviour properties.

The challenge is to capture all relevant detail and to avoid superfluous features. (One might recall here the quotation from Albert Einstein, 'Everything should be made as simple as possible, but not simpler.') For example, consider a project concerned with evaluating strategies for improving the operating efficiency of a fast-food restaurant. Within this context it would likely be meaningless (and indeed, nonsensical) to incorporate into the model information about the sequence in which a server prepares the hot and cold drinks when both are included in a customer's order.

The notion of 'behaviour' is clearly one that is fundamental to these discussions and in particular, we have suggested that there is usually a need to evaluate behaviour. But what does this mean and how is it done? At this point we have to defer addressing these important questions until a more detailed exploration of the features of models has been completed.

Modelling is a constructive activity and this raises the natural question of whether the product (i.e., the model) is 'good enough.' This question can be answered only if there is an identified context and as we show in the discussions to follow, there are many facets to this key issue. One other words, a key question is always whether the model is good enough from the point of view of the project goals. The corollary of this assertion is that it is not meaningful to undertake any modelling study without a clear understanding of the purpose for which the model will be used. Perhaps the most fundamental implication of the above discussion is that it is never meaningful to undertake a study whose goal is simply 'to develop a model of . . . .'

There is a variety of ways in which the specification of behaviour can be formulated. Included here are: natural language, mathematical formalisms, rule-based formalisms, symbolic/graphical descriptions, and combinations of these. It is typical for several distinct formulations of the model (or perhaps only portions of it) to evolve over the course of the study. These alternatives are generally created in formats that are best suited to capturing subtleties or providing clarification.

A particular format that plays a very special role is a specification formulated as a computer program. The importance of such a specification arises because that computer program provides the means for actually carrying out the experiments that are central to the modelling and simulation approach. This illustrates, furthermore, the important fact that some realisations of the specification (which, after all, is the model) are actually executable and produce the behaviour we seek to observe. This legitimises the implications in our frequent use of the phrase 'the model's behaviour.

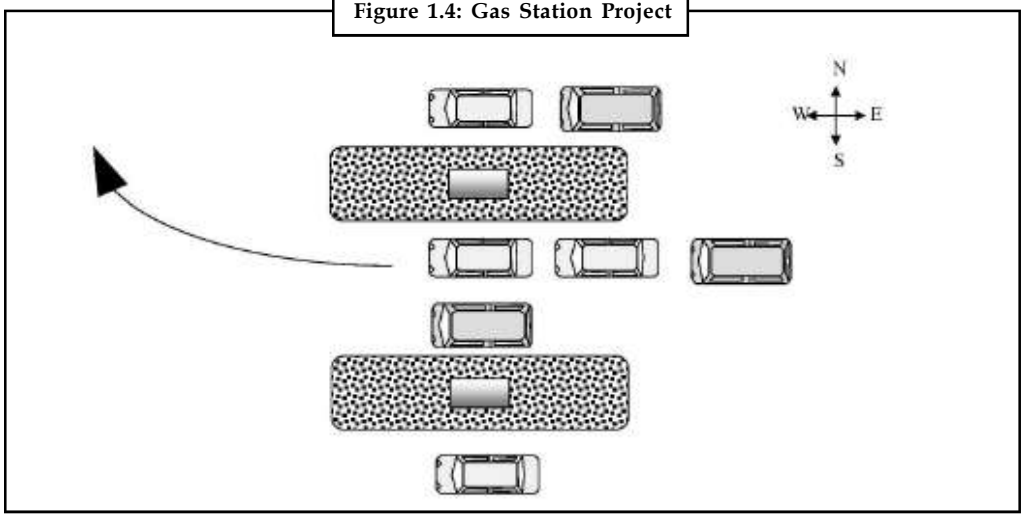
Example (Full-Service Gas Station)

To illustrate some facets of the discussion above, we consider a modelling and simulation project whose system context (SUI) is a 'full-service' gas station with two islands and four service lanes see Figure 1.4. A significant portion of the customers at this station drive small trucks and vans which typically have gas tank capacities that are larger than those of most passenger vehicles. Often the drivers of passenger cars find themselves queued behind these large-tank vehicles which introduce substantially longer wait times when they arrive at the gas pumps. This can cause aggravation and complaints. The station management is considering restricting these large-tank vehicles to two designated lanes. The goal of the modelling and simulation

Notes

project could be to obtain performance data that would assist in determining whether such a policy would improve the flow of vehicles through the station. Vehicles are obliged (via appropriate signage) to access the pumps from the east side and after their respective gas purchases they exit on the west side. Upon arrival, drivers always choose the shortest queue. In the case where two or more queues have the same shortest length, a random choice is made. An exception is when it is observed that a customer in one of the 'shortest queues' is in the payment phase of the transaction in which case that queue is selected by the arriving driver.

Figure 1.4: Gas Station Project



Depending on the time of day, one or two attendants are available to serve the customers. The service activity has three phases. During the first, the attendant determines the customer's requirement and begins the pumping of gas (the pumps have a preset delivery amount and automatically shut off when the preset amount has been delivered). In addition, any peripheral service such as cleaning of windshields and checking oil levels are carried out during this first phase. Phase two is the delivery phase during which the gas is pumped into the customer's gas tank. Phase three is the payment phase; the attendant accepts payment either in the form of cash or credit card. The duration of phase two is reasonably long and an attendant typically has sufficient time either to begin serving a newly arrived customer or to return to handle the phase three (payment) activity for a customer whose gas delivery is complete. The protocol is to give priority to a payment function before serving a newly arrived customer. It is standard practice for the payment function to be carried out by the same attendant who initiated the transaction.

The above text can be regarded as an initial phase in the model building process for this particular modelling and simulation project. Notice, however, that much detail remains to be added; for example, the specification of the arrival rate of vehicles, the proportion of vehicles that fall into the small truck/van category, service times for each of the three service phases, and so on (these correspond to data requirements). Nor should it be assumed that it is entirely complete and adequately comprehensive.

Refinements to this description are almost certain to be necessary; these may simply provide clarification (what are the conditions that govern the attendant's options during phase two) or may introduce additional detail; such as what happens when a pump becomes defective or, under what conditions does an arriving customer 'balk,' that is, decide the queues are too long and leave. Or, in fact, is balking even a relevant occurrence? What about accommodating the possibility that drivers (or passengers) may need to use the washroom facilities and thereby 'hold' the pump position longer than is otherwise necessary? The merits of introducing such refinements must always be weighed against their relevance in terms of achieving the goals of the modelling and simulation project. (It may be useful for the reader to dwell on other possible

Notes

refinements.) In fact, some refinement of the goals is most certainly necessary. (What exactly are the performance data that would enable a meaningful decision to be made?)

It is also important to observe that the model's features as outlined above have an orientation that is specific to the stated goal of the project. There is very little in the presentation that would allow a model formulated from the given description to be useful in, for example, an environmental assessment of the gas station's operation or indeed in an analysis of its financial viability.

1.4 Types of Models

System Modelling

1. System modelling helps the analyst to understand the functionality of the system and models are used to communicate with customers
2. Different models present the system from different perspectives
   (a) External perspective showing the system's context or environment
   (b) Behavioural perspective showing the behaviour of the system
   (c) Structural perspective showing the system or data architecture

Structured Methods

1. Structured methods incorporate system modelling as an inherent part of the method
2. Methods define a set of models, a process for deriving these models and rules and guidelines that should apply to the models
3. CASE tools support system modelling as part of a structured method

Method Weaknesses

1. They do not model non-functional system requirements
2. They do not usually include information about whether a method is appropriate for a given problem
3. The may produce too much documentation
4. The system models are sometimes too detailed and difficult for users to understand

Model Types

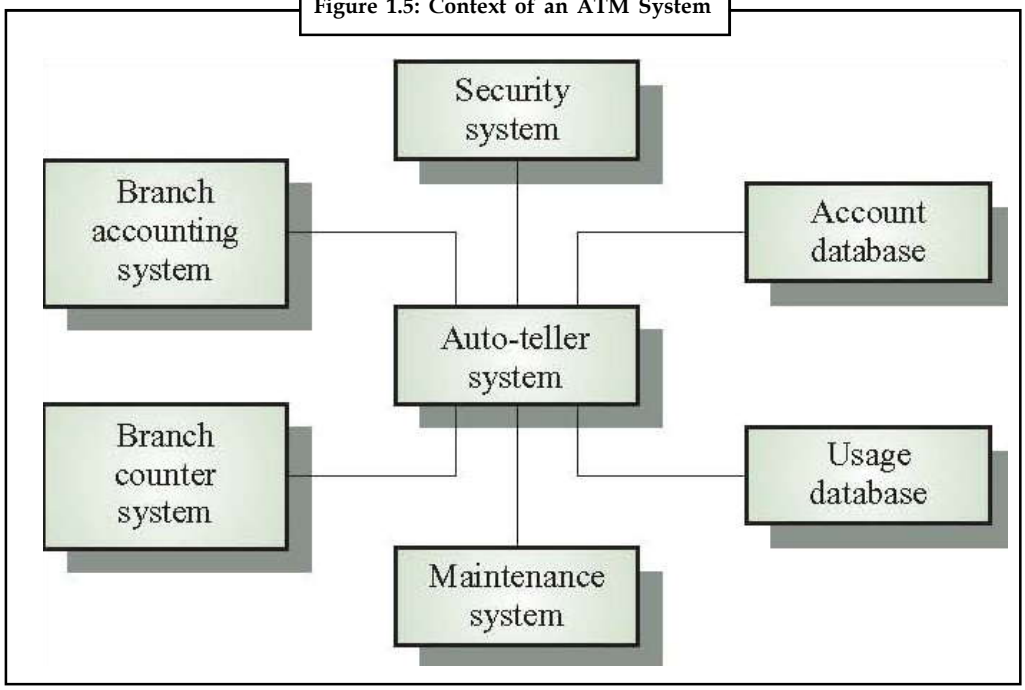
1. Data processing model showing how the data is processed at different stages
2. Composition model showing how entities are composed of other entities
3. Architectural model showing principal sub-systems
4. Classification model showing how entities have common characteristics
5. Stimulus/response model showing the system's reaction to events

Context Models

1. Context models are used to illustrate the boundaries of a system
2. Social and organisational concerns may affect the decision on where to position system boundaries
3. Architectural models show the system and its relationship with other systems

Notes

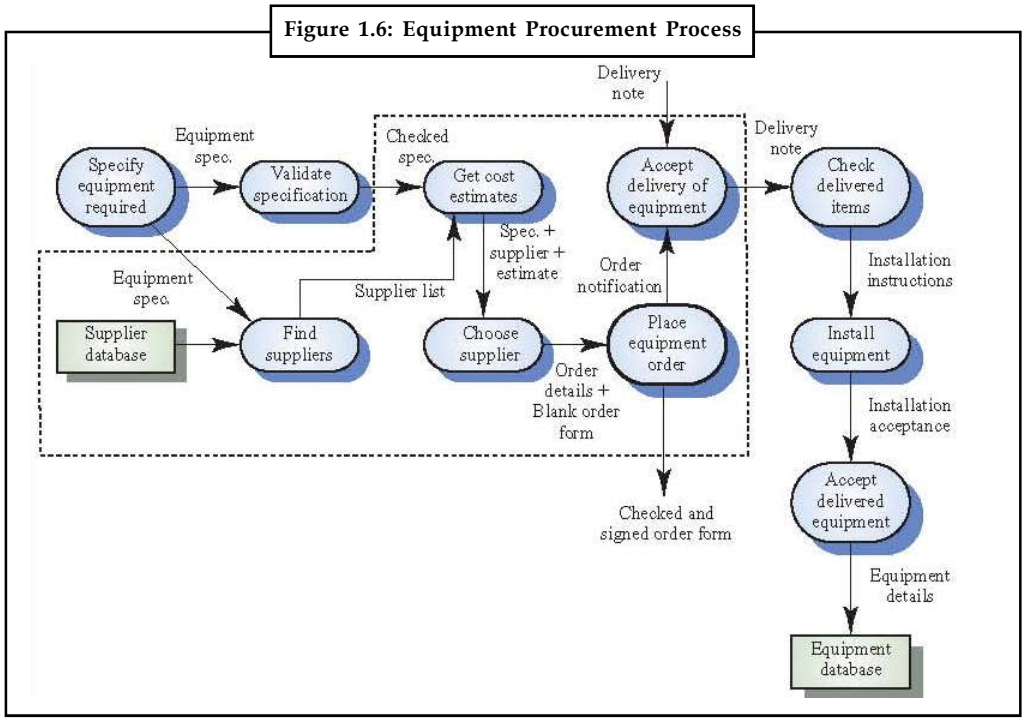
Figure 1.5: Context of an ATM System



Process Models

1. Process models show the overall process and the processes that are supported by the system
2. Data flow models may be used to show the processes and the flow of information from one process to another

Figure 1.6: Equipment Procurement Process



Notes

Behavioural Models

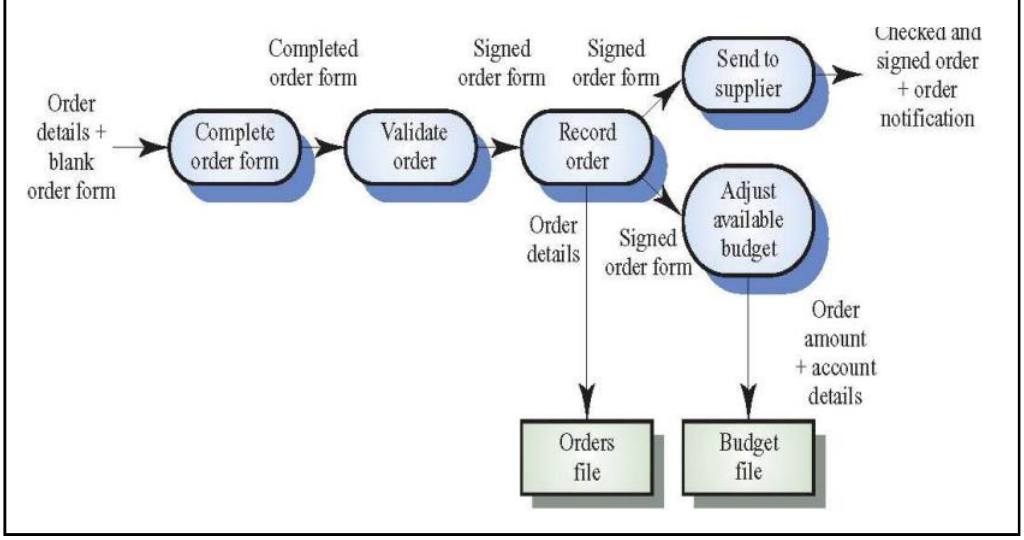
1. Behavioural models are used to describe the overall behaviour of a system
2. Two types of behavioural model are shown here
   (a) Data processing models that show how data is processed as it moves through the system
   (b) State machine models that show the systems response to events

Both of these models are required for a description of the system's behaviour

Data-processing Models

1. Data flow diagrams are used to model the system's data processing
2. These show the processing steps as data flows through a system
   (a) Intrinsic part of many analysis methods
   (b) Simple and intuitive notation that customers can understand
   (c) Show end-to-end processing of data
   (d) Order Processing DFD

Figure 1.7: Order Processing DFD

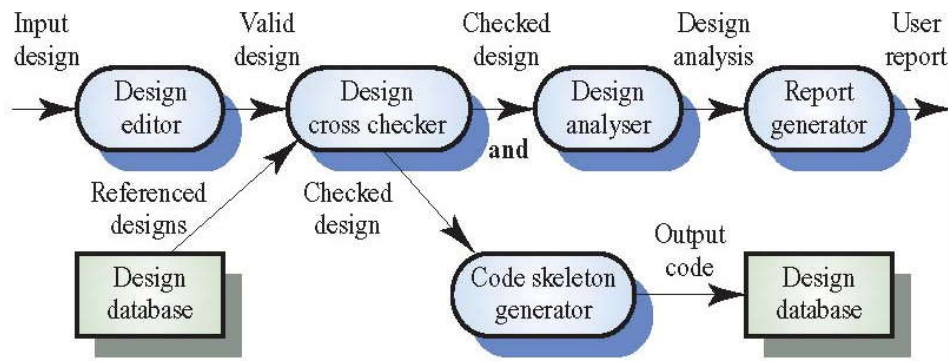


Data Flow Diagrams

1. DFDs model the system from a functional perspective
2. Tracking and documenting how the data associated with a process is helpful to develop an overall understanding of the system
3. Data flow diagrams may also be used in showing the data exchange between a system and other systems in its environment

Notes

Figure 1.8: CASE toolset DFD



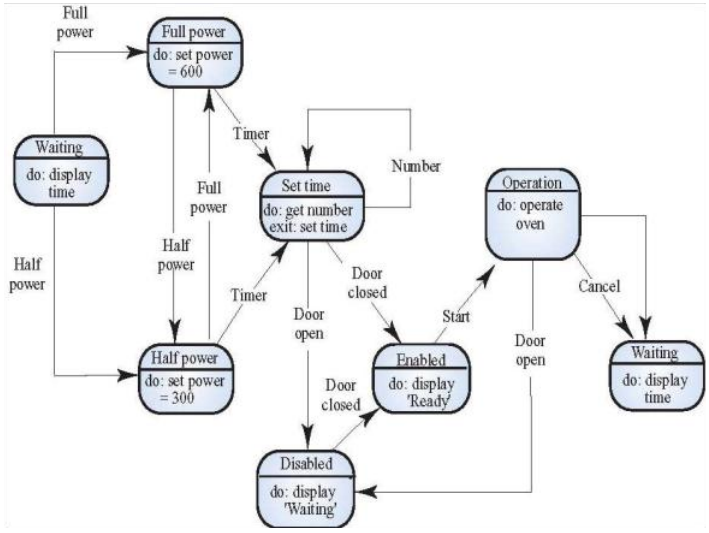
Did u know? **What is DFD?**

DFD is a chart that follows the movement of data in a computer system and displays how the data is to be processed to characterize data.

State Machine Models

1. These model the behaviour of the system in response to external and internal events
2. They show the system's responses to stimuli so are often used for modelling real-time systems
3. State machine models show system states as nodes and events as arcs between these nodes. When an event occurs, the system moves from one state to another
4. State charts are an integral part of the UML

Figure 1.9: Microwave Oven Model

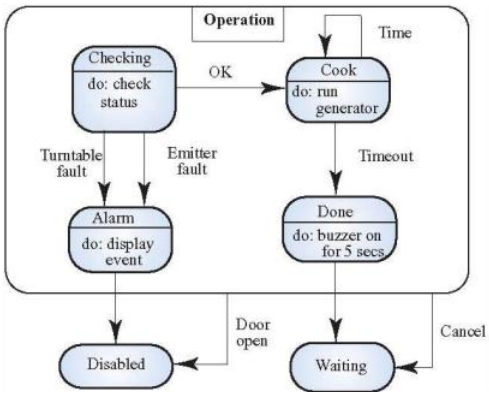


Notes

State Charts

1. Allow the decomposition of a model into sub-models (see following slide)
2. A brief description of the actions is included following the 'do' in each state
3. Can be complemented by tables describing the states and the stimuli

Figure 1.10: Microwave Oven Operation

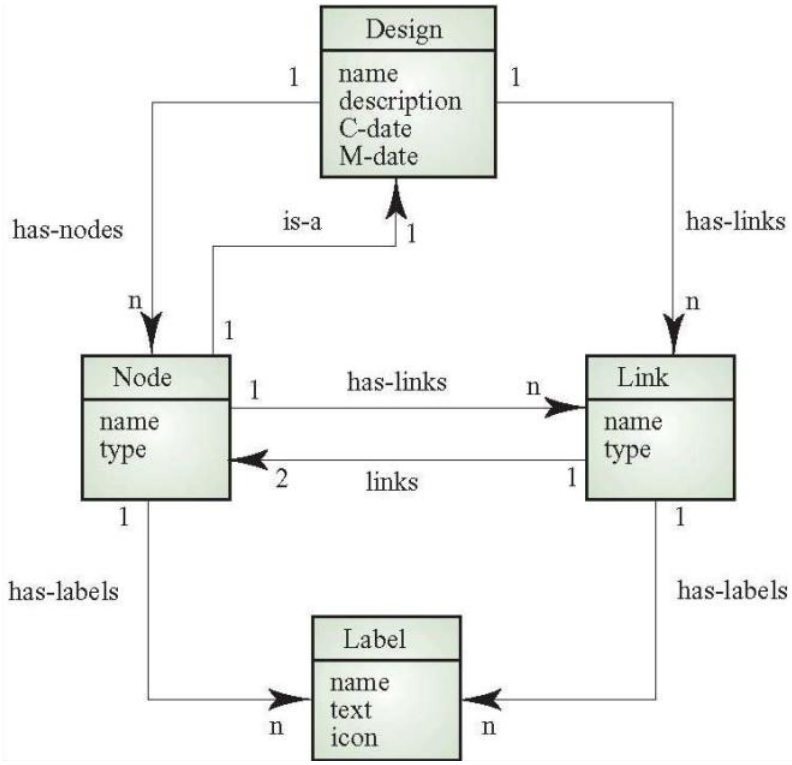


Semantic Data Models

1. Used to describe the logical structure of data processed by the system
2. Entity-relation-attribute model sets out the entities in the system, the relationships between these entities and the entity attributes
3. Widely used in database design. Can readily be implemented using relational databases
4. No specific notation provided in the UML but objects and associations can be used

Notes

Figure 1.11: Software Design Semantic Model



Data Dictionaries

1. Data dictionaries are lists of all of the names used in the system models. Descriptions of the entities, relationships and attributes are also included
2. Advantages
   (a) Support name management and avoid duplication
   (b) Store of organisational knowledge linking analysis, design and implementation
3. Many CASE workbenches support data dictionaries

Object Models

1. Object models describe the system in terms of object classes
2. An object class is an abstraction over a set of objects with common attributes and the services (operations) provided by each object
3. Various object models may be produced
   (a) Inheritance models
   (b) Aggregation models
   (c) Interaction models

Notes

4. Natural ways of reflecting the real-world entities manipulated by the system
5. More abstract entities are more difficult to model using this approach
6. Object class identification is recognised as a difficult process requiring a deep understanding of the application domain
7. Object classes reflecting domain entities are reusable across systems

Inheritance Models

1. Organise the domain object classes into a hierarchy
2. Classes at the top of the hierarchy reflect the common features of all classes
3. Object classes inherit their attributes and services from one or more super-classes. these may then be specialized as necessary
4. Class hierarchy design is a difficult process if duplication in different branches is to be avoided

Task Analyze the difference between various types of models.

1.5 Simulation of an Inventory Problem

An important group of simulation problems include inventory systems.

When simulating an inventory control systems project, it is essential to take stock of all the key elements of real-world inventory control without overseeing the serious real-world factors. A well-designed inventory control simulation should involve data depending on the suggestions of front-line employees who know where losses appear that might otherwise go un-observed.

Inventory Control Simulation Key Factors

An inventory control system should take into report key factors like demand fluctuation depending on market trends, spoilage in unstable goods like food or chemicals, shrinkage because of spills, product destroyed in shipping, and shrinkage caused by staff. Demand fluctuation based on market trends can only be calculated in a general sense by examining past precedences with similar products and how they connect to new items. In contrast, the spoilage of unstable goods is generally a very predictable procedure and can be diminished by effectively calculating how much product will be sold before the shelf-life of the item expires, thereby evading over-purchasing. Within inventory control, calculating the market demand for unstable goods and verifying that the company does not buy too much or too little of a product is within the most difficult of tasks, and must be supported by large volume data samples before an informed decision can be made. Shrinkage because of spills can be diminshed through the employee training programs, though the precise amount of shrinkage will change dramatically between locations and need some real-world data gathering. Shrinkage because of internal and external staff can be virtually eliminated by improving systemic security protocols, involving adequate security monitoring and loss-prevention method.

Determining an Inventory Control Strategy

Selecting an inventory control strategy for the simulation experiment needs an intimate knowledge of the particular nature of the business being examined. A small-scale greengrocer,

Notes

for instance, should focus their inventory control strategy on anticipating consumer demand and diminshing loss due to spoilage, whereas a large stable goods retailer like Wal-Mart, Kmart or Target can afford to make large purchase orders of items, store them in a warehouse and distribute them internally when receiving volume purchasing discounts.

The Effectiveness of Inventory Control Simulations

The effectiveness of business simulation software is not globally accepted, but it has gained important credibility by being used in at least three American universities. As per the Entrepreneur Magazine, simulation software is being used by students under the administration of professors or assistant professors at the University of Chicago, Seton Hall University and Michigan State University. In spite of the growing popularity of business simulation software, its correctness can only be as good as its programming, meaning that programs will be estimated on a case-by-case basis.

Example: Suppose you work in a retail store as a inventory manager in inventory mananagement department. Now as a manager it is your responsibility to keep restock certain item in, the 'store by ordering it from the wholesaler. For this you want to adopt a simple policy for ordering new supplies: you have to fix that 'When your stock goes down to P items here P called reorder point, you will order Q more items , here Q called as reorder quantity from the wholesaler.' If the demand on any day exceeds the amount of inventory on hand, the excess represents lost sale and loss of goodwill. On the other hand, overstocking implies increased carrying cost (i.e., cost of storage, insurance, interest, deterioration, etc.). Ordering too frequently will result in excessive reorder cost. Assume the following conditions:

1. There is a 3-day lag between the order and arrival. The merchandise is ordered at the end of the day and is received at the beginning of the fourth day. That is, merchandise ordered on the evening of the ith day is received on the morning of the (i + 3)rd day.
2. For each unit of inventory the carrying cost for each night is ₹ 0.75.
3. Each unit out of stock when ordered results into a loss of goodwill worth ₹ 2.00 per unit plus loss of ₹ 16.00 net income, that would have resulted in its sale, or a total loss of ₹ 18.00 per unit. Lost sales are lost forever; they cannot be backordered.
4. Placement of each order costs ₹ 75.00 regardless of the number of units ordered.
5. The demand in a day can be for any number of units between 0 and 99, each equiprobable.
6. There is never more than one replenishment order outstanding.
7. Initially we have 115 units on hand and no reorder outstanding.

With these conditions in force you have been asked to compare the following five replenishment policies and seJect the one that has the minimum total cost (i.e., reorder cost + carrying cost + lost sales cost).

	P (reorder pomt)	Q (reorder quantity)
Policy I	125	150
Policy II	125	250
Policy III	150	250
Policy IV	175	250
Policy V	175	300

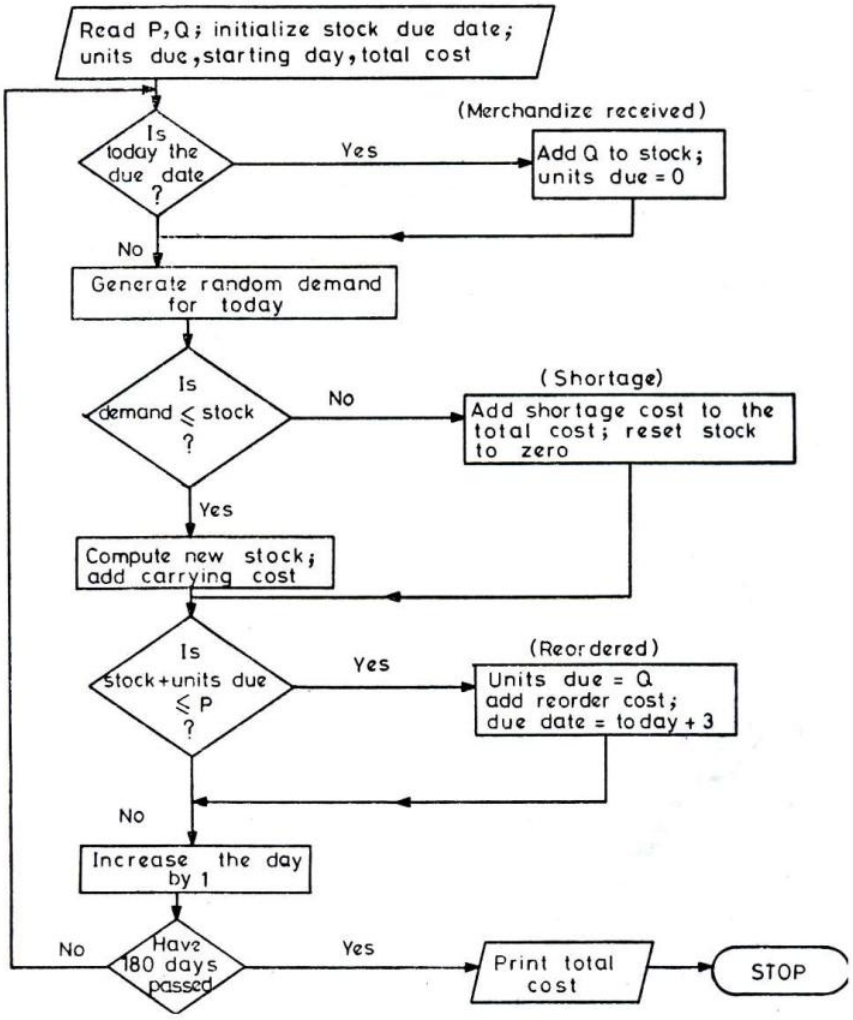
Notes

(Since we are intereste.d in simulation here and not in inventory theory, we will not investigate the reasonableness of the replenishment policy too critically. Ours is undoubtedly a very simplified model.)

The problem does not easily lend itself to an analytic solution; it is best therefore to solve it by ;imuiation. Let us simulate the running of the store for about six months (180 days) under each of the five policies and then compare their costs.

A simulation model of this inventory system can be easily constructed by stepping time forward in the fixed increment of a day, starting with Day 1, and continuing up to Day 180. On a typical day, Day i, first we check to see if merchandise is due to arrive today. If yes, then the existing stock S is increased by Q (the quantity that was ordered). If DEM is the demand for today, and DEM ≤ S, our new stock at the end of today will be (S – DEM) units. If DEM > S, then our new stock will be zero. In either case, we calculate the total cost resulting from today's transactions, and add it to the total cost C incurred till yesterday. Then we determine if the inventory on hand plus units on order is greater than P, the reorder point. If not, place

Cost of Inventory Policy (P, Q)



Notes

an order (to be delivered three days hence), by stating the amount ordered and the day it is due to be received. We repeat this procedure for 180 days'. Initially we set day number i = 1, stock S = 115, number of units due UD = 0 (because there is no outstanding order), and the day they are due DD=0.

The demand, DEM, for each day is not a fixed quantity but a random variable. It could assume any integral value from 00 to 99 and each with equal probability. We will use a special subroutine, which generates a 2-digit random integer, and will use the numbers thus produced as the daily demands.

The following Fortran program (format free) implements the flowchart for evaluating the total cost for a given replenishment policy (P, Q) for 180 days. Statement No. 110 in the program makes use of the subprogram RNDY1 (DUM) which is a subprogram to generate a real pseudorandom number between zero and one. The argument of this function can be any dummy number or variable.

Notice how condition 6, that there is no more than one reorder outstanding, has been taken care of. In Statement 130, we add the number of units due (if any) UD to the current stock S to get the equivalent stock, ES. It is this number which is compared to P before an order is placed. Since UD> P if we already have a replenishment order outstanding another order will not be placed.

This is an extremely simple model of an inventory-control system.

```
        INTEGER P, Q, S, ES, UD, DD. DEM
        READ, P, Q
        C = 0.0
        S = 115
        I = 1
        UD = O
        DD = O
100     IF (DD .NE. I) GO TO 110
        S = S + Q
        UD = 0
110     DEM = RNDY1 (DUM)* 100.0
        IF (DEM .LE. S) GO TO 120
        C = C + (FLOAT (DEM) – FLOAT (S))*18.0
        S = 0
        GO TO 130
120     S = S – DEM
        C = C + FLOAT (S)*.75
130     ES =  S + UD
        IF (ES .GT. P) GO TO 140
        UD = Q
        DD = I + 3
        C = C + 75.0
140     I = 1 + 1
        IF (I .LE. 180) GO TO 100
        PRINT, P, Q, C
        STOP
        END
```

Notes

The program yielded the following cost figures for the five inventory policies:

P	Q	Cost in ₹
125	150	38679.75
125	250	31268.25
150	250	29699.25
175	250	26094.00
175	300	27773.25

Thus, Policy IV (P = 175, Q = 250) is the best amongst the five considered.

Caselet

NeST Arm unveils Simulation Debugger

Ashling Microsystems, a division of Kerala-based diversified NeST Group, has launched simulation debug software for designers of electronics systems. Debugging is a methodical process of finding and reducing the number of bugs, or defects, in a computer programme or a piece of electronic hardware.

Named PathFinder-XD, the new software provides full support for debugging bare-metal and embedded Linux-based applications, a NeST Group spokesman said here on Monday. It operates on Linux x86 and Windows hosts and is now available for free download at www.ashling.com and www.nestsoftware.com.

Designers of embedded electronics systems are faced with the challenge of debugging the complex software that runs on them. Availability of efficient tools will help designers in quickly debugging their software and thereby getting their products to market faster, the spokesman said.

He quoted Mr N. Jehangir, Vice-Chairman and Managing Director of NeST Group, as saying that the PathFinder-XD simulator software is the first in a series of new embedded software and hardware tools that the NeST Group is planning to roll out from the Ashling division. More high-tech and effective tools of this genre will be brought out from the group's development facility in Thiruvananthapuram and manufacturing facility in Kochi and Bangalore, the spokesperson added.

1.6 Basic Nature of Simulation

The word 'simulation' is frequently used alone in a variety of contexts. For example, it is sometimes used as a noun to imply a specialized computer program (as in, 'A simulation has been developed for the proposed system.'). It is also used frequently as an adjective (as in, 'The simulation results indicate that the risk of failure is minimal,' or 'Several extensions have been introduced into the language to increase its effectiveness for simulation programming'). These wide-ranging and essentially inconsistent usages of the word 'simulation' can cause regrettable confusion for neophytes to the discipline. As a rule, we avoid such multiplicity of uses of this word but, as will become apparent, we do use the word as an adjective in two specific contexts where the implication is particularly suggestive and natural.

Simulation involves **Notes**

Inventory System

1. Discrete
2. Stochastic

Pure Pursuit

1. Continuous
2. Deterministic

Still, few things are general for all types of simulation:

1. We initiate with a mathematical model of the system under study
2. Some initial conditions (i.e., at t=0) are presumed
3. State change appears in agreement with some equations (rules or laws)
4. At the end, desired information concerning the system (i.e., problem solution) is composed
5. These rules can be coded in a computer program. Alteration in program variables mimics the state transform.
6. Hence, through simulation, we managed to get around the requirement of acquiring the analytic solution

1.7 When to Simulate

Simulation is a decision examination and support tool. Simulation software permits you to assess, compare and optimize alternative designs, plans and policies. As such, it offers a tool for explaining and protecting decisions to a variety of stakeholders.

Simulation should be used when the penalties of a proposed action, plan or design cannot be directly and right away observed (i.e., the consequences are postponed in time and/or dispersed in space) and/or it is just impractical or prohibitively luxurious to test the alternatives straightforwardly. For instance, when implementing a strategic plan for a company, the impacts are probable to take months (or years) to happen.

Simulation is mainly valuable when there is important uncertainty concerning the outcome or consequences of a particular substitute under consideration. Probabilistic simulation permits you deal with this vagueness in a quantifiable manner.

Perhaps most significantly, simulation should be used when the system under concern has complex interactions and necessitates the input from multiple disciplines. In this case, it is hard for any one person to easily recognize the system. A simulation model can perform as the framework to integrate the various components so as to better understand their communications. As such, it becomes a management tool that keeps you focused on the "big picture" without getting lost in insignificant details.

Task What do you think the best time for simulation to be valuable?

Notes

1.8 Summary

- Simul8 Standard is an incorporated environment for functioning with simulation models.
- The defining feature of the modelling and simulation approach is that it is founded on a computer-based experimental investigation that utilises an appropriate model for the SUI.
- Pure pursuit is a tracking algorithm that functions by scheming the curvature that will shift a vehicle from its existing position to some objective position.
- Systems are often visualized or modeled as component blocks that have connections drawn between them.
- Elements are the kinds of parts (things or substances) that make up a system. Attributes are characteristics of the elements that may be perceived and measured. Relationships are the associations that occur between elements and attributes.
- A model is a specification for behaviour generation and the modelling process is concerned with the development of this specification.
- When simulating an inventory control systems project, it is essential to take stock of all the key elements of real-world inventory control without overseeing the serious real-world factors.

1.9 Keywords

Attributes: Attributes are characteristics of the elements that may be perceived and measured. Relationships are the associations that occur between elements and attributes.

Elements: Elements are the kinds of parts (things or substances) that make up a system.

Model: A model is a representation of an object, system or idea in some form other than itself.

1.10 Self Assessment

Fill in the blanks:

1. ..... Standard is an incorporated environment for functioning with simulation models
2. ..... is a tracking algorithm that functions by scheming the curvature that will shift a vehicle from its existing position to some objective position.
3. Systems are often visualized or modeled as component blocks that have ..... drawn between them.
4. ..... are characteristics of the elements that may be perceived and measured.
5. Elements are the kinds of parts (things or substances) that make up a ..... .
6. The various entities that populate the space that the …… embraces with particular focus on the interactions among these entities.
7. Informal sketches are often the best means of representing these …… features.
8. A …… phase must be carried out in order to add detail where necessary, incorporate formalisms wherever helpful, and generally enhance the precision and completeness of the accumulated information.

Notes

9. The reformulation of the information within the project description in terms of …… and variables.
10. The …… model may, in reality, be a collection of partial models each capturing some specific aspect of the SUI's behavior.
11. The essential requirement for the …… phase of a modelling and simulation project is an executable computer program that embodies the conceptual model.
12. The simulation model is the …… stage of a development process that began with the decision to formulate a modelling and simulation project to resolve an identified problem.
13. Convenient programming constructs to deal with these various tasks are normally provided in software environments specifically designed to support the …… activity.
14. Systems have a structure that is defined by its …… and processes.
15. Functional relationships can only occur because of the presence of a ……

1.11 Review Questions

1. Outline the essential steps involved in carrying out a modelling and simulation study.
2. Analyze how conceptual model turns out to be the basis for the development of the simulation model.
3. Examine the impact of validation phase and experimentation phase on modelling and simulation process.
4. Modelling is a constructive activity and this raises the natural question of whether the product (i.e., the model) is 'good enough.' Comment.
5. What do you think would be the main factors regarding Inventory Control Simulation.
6. Simulation software permits you to assess, compare and optimize alternative designs, plans and policies. Is this statement true or false according to you. Justify your answer with proper reasoning.
7. The information provided by the project description is, for the most part, unstructured and relatively informal. Comment
8. Explain the Simulation Model with examples
9. The model's credibility has been established, is the experimentation phase. Explain with reasons
10. Do you think functional services can include such features as data presentation? If yes then why?

Answers: Self Assessment

1.	Simul8	2.	Pure pursuit
3.	Connections	4.	Attributes
5.	System	6.	SUI
7.	structural	8.	refinement
9.	parameters	10.	conceptual

Notes

11. experimentation
12. penultimate
13. simulation
14. parts
15. driving force

1.12 Further Readings

Books

Balci, O.,(1994), Validation, verification, and testing techniques throughout the life cycle of a simulation study, Annals of Operations Research, 53: 121–173.

Balci, O., (2001), *A methodology for certification of modelling and simulation applications, ACM Transactions on Modelling and Computer Simulation, 11*: 352–377.

Birta, L.G. and Ozmizrak, N.F., (1996), *A knowledge-based approach for the validation of simulation models: The foundation, ACM Transactions on Modelling and Computer Simulation*, 6: 67–98.

Boehm, B.W., (1979), *Software engineering: R&D trends and defense needs, in: P.* Wegner (Ed.), Research Directions in Software Technology, MIT Press, and Cambridge, MA.

Cellier, F.E., (1986), Combined discrete/continuous system simulation – application, techniques and tools, in Proceedings of the 1986 Winter Simulation Conference.

Department of Defense (DoD) Recommended Practices Guide (RPG) for Modelling and Simulation VV&A, Millennium Edition (available at http://vva.dmso.mil).

General Accounting Office, (1976), Report to the Congress: Ways to improve management of federally funded computerized models, report LCD-75-111, U.S. General Accounting Office, Washington, DC.

General Accounting Office, (1979), Guidelines for model evaluation, report PAD-79-17, U.S. General Accounting Office, Washington, DC.

General Accounting Office, (1987), DOD simulations: Improved assessment procedures would increase the credibility of results, report GAO/PEMD-88-3, U.S. General Accounting Office, and Washington, DC.

Ören, T.I., (1981), Concepts and criteria to access acceptability of simulation studies, Communications of the ACM, 24: 180–189.

Online link

http://www.bakker.org/cfm/webdoc9.htm

Unit 2: Simulation of Continuous System (I)

Notes

CONTENTS

Objectives

Introduction

2.1 Simulation of Continuous Systems

2.1.1 A Chemical Reactor

2.1.2 Numerical Integration vs Continuous System Simulation

2.2 Summary

2.3 Keywords

2.4 Self Assessment

2.5 Review Questions

2.6 Further Readings

Objectives

After studying this unit, you will be able to:

- Understand the simulation of continuous systems
- Describe Numerical integration vs continuous system simulation
- Explain the selection of integration formula

Introduction

CSSL(Continuous System Simulation Language) versions I, II, III, IV and V have been commercially available since 1968. CSSL-I was developed for Jet Propulsion Labs in 1968. CSSL-III was widely distributed from 1969-1975. CSSL-IV (interactive version) was developed by R. Nilsen and ran on over 30 different computers. Currently CSSL-V is marketed by Simulation Services International and available on PCs and workstations.

2.1 Simulation of Continuous Systems

Continuous System Simulation illustrates analytically and systematically how mathematical models of dynamic systems, typically described by sets of either ordinary or partial differential equations perhaps attached with algebraic equations, can be simulated on a digital computer.

Modern modelling and simulation environments alleviate the occasional user from having to understand how simulation actually works. Once a mathematical model of a process has been formulated, the modelling and simulation environment compiles and simulates the model, and curves of result trajectories occur magically on the user's screen. Yet, magic has a propensity to fail, and it is then that the user must recognize what went wrong, and why the model could not be simulated as predictable.

Notes

Continuous System Simulation is written by engineers for engineers, launching the partly symbolical and partly numerical algorithms that drive the procedure of simulation in terms that are familiar to simulation practitioners with an engineering background, and yet, the text is precise in its approach and inclusive in its coverage, providing the reader with a methodical and detailed understanding of the mechanisms that administer the simulation of dynamical systems.

Continuous System Simulation is a extremely software-oriented text, depending on MATLAB. Homework problems, suggestions for term project, and open research questions conclude every chapter to intensify the understanding of the student and enhance his or her inspiration.

2.1.1 A Chemical Reactor

As you all know that in a chemical reaction when two substances A and B are brought together they produce a third chemical substance C. It means if 1 gram of A combines with 1 gram of B then it produce 2 grams of C. Moreover, the rate of formation of C is proportional to the product of the amounts of A and B present. This type of reaction is called as forward reaction. In addition to this forward reaction there is also a backward reaction, means decomposing C back into A and B. The rate of decomposition of C is proportional to the amount of C present in the mixer.

In other words, we can say at any time t if a, b, and c are the quantities of the chemicals A, B and C present, respectively, then their rates of increases are described by the following three differential equations:

$$\frac{da}{dt} = k_2c - k_1ab, \qquad ...(1)$$

$$\frac{db}{dt} = k_2c - k_1ab, \qquad ...(2)$$

$$\frac{dc}{dt} = 2k_1ab - 2k_2c, \qquad ...(3)$$

Where k_1 and k_2 are the rate constants. these constants will vary with temperature and pressure, but we do not allow the temperature or pressure of the reaction to vary. Given the values of the constants k_1 and k_2 and the initial quantities of the chemicals A and B , now we wish to determine how much of C has been produced as a function of time. Determination of the rate of such chemical reactions is important in many industrial processes.

A straightforward method of simulating this system is to start at time zero and increment time in small steps of Δt. We assume that the quantities of chemicals remain unaltered during each step and only change 'instantaneously' at the end of the step. Thus the quantity of A (or B or C) at the end of one such step is given in terms of the quantity at the beginning of the step as:

$$a(t+\Delta t) = a(t) + \frac{da(t)}{dt}.\Delta t \qquad ...(4)$$

If Δ t is sufficiently small Eq. (4) is a reasonable representation. Identical equations can be written for $b(t + \Delta t)$ and $c(t + \Delta t)$.

Suppose we wish to simulate the system for a period T. We will divide this period T into a large number N of small periods Δt. This is:

$$T = N\Delta T$$

At time zero, we know a(0), b(0), c(0). From these initial values and the values k_1 and k_2 we compute the amounts of chemicals at time Δt as:

$$a(\Delta t) = a(0) + [k_2.c(0) - k_1.a(0).b(0)]\Delta t$$

Notes

$$b(\Delta t) = b(0) + [k_2.c(0) - k_1.a(0).b(0)]\Delta t$$

$$c(\Delta t) = c(0) + [2k_1.a(0).b(0) - 2k_2.c(0)]\Delta t$$

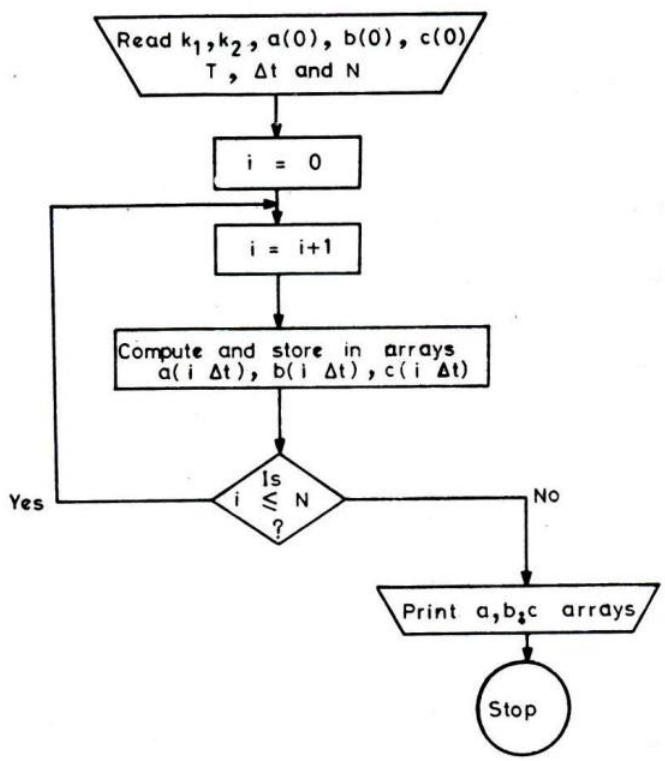
Using these values we calculate the next state of the system, i.e., at time 2t as

$$a(2\Delta t) = a(\Delta t) + [k_2.c(\Delta t) - k_1 a(\Delta t).b(\Delta t)].\Delta t$$

$$b(2\Delta t) = b(\Delta t) + [k_2.c(\Delta t) - k_1 a(\Delta t).b(\Delta t)].\Delta t$$

$$c(2\Delta t) = c(\Delta t) + [2k_1.a(\Delta t).b(\Delta t) - 2k_2.c(\Delta t)].\Delta t$$

Using the state of the system at 2 Δt, we determine its state at 3 Δt, and so on. We continue in this vein, moving time forward by Δt and determining the state of the system from the previous state, for N steps, at the end of which we have the desired result. This procedure is shown in the form of a flow-chart given below:



A FORTRAN (format free) program which simulates the system with k_1 = 0.008 gram^{-1} min^{-1}, k_2 = 0.002 min^{-1}, and a(0) = 100 grams, b(0) = 50 grams, c(0) = 0 grams, for a period of T = 5 minutes in steps of Δt = 0.1 min. is as follows (N = 50).

```
DIMENSION A (52), B(52), C(52),
REAL KI, K2
A(1) = 100.0
BC(l) = 50.0
C(l) = 0.0
2(45 – 123/77)
DELTA = 0.1
T = 0
Kl = 0.008 K2 = 0.002 DO 3 I=1,51
PRINT, T, A(I), B(I), C(I)
```

Notes

```
      A(I + 1) = A(I) + (K2*C(I) – Kl* A(I)*B(I))*DEL TA
      B(I + 1) = B(I) + (K2*C(I) – Kl* A(I)*B(I)*DELTA
      C(I + 1) = C(I) + 2. *(Kl * A(I)*B(I) – K2*C(I)*DELTA
      T = T + DELTA
3     CONTINUE
      STOP
      END
```

The following is the output of this program, which gives the state of the system (i.e., the values of a, b, and c) for 5 minutes at the intervals of 0.1 minute.

TIME	A(I)	B(I)	C(I)
0.00	100.00	50.00	0.00
0.10	96.00	46.00	8.00
0.20	92.47	42.47	15.06
0.30	89.33	39.33	21. 34
0.40	86.52	36.52	26.95
0.50	84.00	34.00	32.00
0.60	81.72	31. 72	36.55
0.70	79.66	29.66	40.69
0.80	77.77	27.77	44.45
0.90	76.05	26.05	47.89
1.00	74.48	24.48	51.04
1.10	73.03	23.03	53.94
1.20	71.70	21. 70	56.61
1.30	70.46	20.46	59.07
1.40	69.32	19.32	61.36
1.50	68.26	18.26	63.48
1.60	67.28	17.28	65.44
1. 70	66-.36	16.36	67.28
1.80	65.51	15.51	68.99
1.90	64.71	14.71	70.59
2.00	63.96	13.96	72.08
2.10	63.26	13.26	73.48
2.20	62.60	12.60	74.79
2.30	61.99	11.99	76.03
2.40	61.41	11.41	77.18
2.50	60.86	10.86	78.27
2.60	60.35	10.35	79.30
2.70	59.87	9.87	80.27
2.80	59.41	9.41	81.18
2.90	58.98	8.98	82.04

Contd...

Notes

3.00	58.57	8.57	82.86
3.10	58.19	8.19	83.63
3.20	57.82	7.82	84.36
3.30	57.48	7.48	85.05
3.40	57.15	7.15	85.70
3.50	56.84	6.84	86.32
3.60	56.55	6.55	86.91
3.70	56.27	6.27	87.46
3.80	56.00	6.00	87.99
3.90	55.75	5.75	88.50
4.00	55.51	5.51	88.97
4.10	55.29	5.29	89.43
4.20	55.07	5.07	89.86
4.30	54.86	4.86	90.27
4.40	54.67	4.67	90.66
4.50	54.48	4.48	91.03
4.60	54.31	4.31	91. 39
4.70	54.14	4.14	91.73
4.80	53.98	3.98	92.0'
4.90	53.82	3.82	92.35
5.00	53.68	3.68	92.65

2.1.2 Numerical Integration vs Continuous System Simulation

Continuous Simulation points to a computer model of a physical system that endlessly tracks system response over time as per the set of equations usually including differential equations.

In continuous simulation, the continuous time reply of a physical system is modeled using ODEs.

Newton's 2nd law, F = ma, is a superior example of a single ODE continuous system. Numerical integration methods like Runge Kutta, or Bulirsch-Stoer are used to elucidate the system of ODEs. By integrating the ODE solver with other numerical operators and techniques a continuous simulator can be used to model many different physical phenomena like flight dynamics, robotics, automotive suspensions, hydraulics, electric power, electric motors, human respiration, polar ice cap melting, steam power plants etc. There is practically no limit to the classes of physical incident that can be modeled by a system of ODE's. Some systems although can not have all derivative terms mentioned explicitly from known inputs and other ODE outputs. Those derivative terms are defined completely by other system constraints like Kirchoff's law that the flow of charge into a joint must equal the flow out.

Notes To resolve these implicit ODE systems a congregating iterative scheme like Newton-Rapson must be employed.

Notes

Selection of Integration Formulas

Block-oriented simulation languages are a dominant tool for solving simulation problems. A problem of major significance in simulation,yet, is the option of the integration formula to be employed in the integration block. Users of previous simulation languages had no problem in choosing a formula as these languages provided only a single formula. Advanced languages have a number of formulas available, but leave the choice to the user who does not have much to go by apart from the good names of the formulas. Troubles which were encountered in the use of multistep formulas made formulas of the Runge-Kutta type more well-liked. Experience has shown that they are less critical with respect to steadiness. Though, the calculating time for a Runge-Kutta formula is considerably larger and for this reason multistep formulas, which involve the predictor- corrector scheme, are however appealing.

Caselet

5 Ways to Automate Internal Controls

SCARF and EAM, snapshots, audit hooks, ITF and CIS. Quite an alphabet soup, but these are the five types of automated evaluation techniques discussed in Technical Guide on Information System Audit from the Committee on Information Technology of the Institute of Chartered Accountants of India.

The use of SCARF (system control audit review file) and EAM (embedded audit modules) involves planting specially-written audit software in the organisation's host application system so that the application systems are monitored on a selective basis, explains the publication.

The second technique, that is, snapshot involves taking "pictures of the processing path that a transaction follows from the input to the output stage. With the use of this technique, transactions are tagged by applying identifiers to input data and recording selected information about what occurs for the auditor's subsequent review."

Audit hooks function as red flags and induce information system auditors to act before an error or irregularity gets out of hand; ITF (integrated test facilities) gets the system to process test transactions and to update the records of a dummy entity; and CIS (continuous and intermittent simulation) mimics instruction execution of an application during a process run.

To those who wonder if all these are relevant to companies, the preface explains how information system audits are increasingly necessary in the context of the Clause 49 in corporate law regime, which requires certification by the directors about the existence and operation of sufficient internal controls.

Recommended Read for Security Professionals

Neural Networks

Essentially, when we respond to the daily stimuli of the environment that we already know from past interactions, we are using the same circuitry to define ourselves in our world, writes Joe Dispenza in Evolve Your Brain: The science of changing your mind (Westland).

"Most people spend a great deal of their day unconsciously feeling and thinking from past memories. They do this because they have hardwired those experiences by repeatedly thinking of them and by associating many other experiences with them."

Contd...

Notes

When we are processing the same thoughts repeatedly every day, the mind that is created from the same stimulation of the same neural networks will become automatic, unconscious, routine, familiar, common, and more habitual, the author reasons.

He argues that the neural networks we form from our repeated thoughts, actions, behaviours, feelings, emotions, skills and conditioned experiences are etched in the brain's hardware and they become 'an effortless, unconscious response stimulated by our environment.'

Healthy Study

Strategy Alignment

The success of a software application depends upon how it fits into the IT (information technology) strategy of the company, observes Pradeep Hari Pendse in Business Analysis: Visualizing business processes and effective software solutions (www.phindia.com).

He finds that organisations that are not very IT savvy tend to take up IT applications that are not necessarily part of the business strategy, even as organisations that are business leaders tend to pick up new technology as also apply it effectively to business strategy. "Where the application is a crucial part of the organisation's business strategy, the BA (business analyst) can expect a much higher level of commitment from the clients' top management as well as users at all levels. This greatly enhances the chances of success of the application and reduces the client-related risk factors for the application."

Pendse advises BAs to understand how CIOs (chief information officers) of the organisation have arrived at a focus on a particular application.

"The BA could use the same frameworks to project the CIO's strategic concerns and be able to propose solutions in a manner which would tend to get the CIO's mindshare."

Useful Guidance

Talent investment

IBM fills key positions with top-notch people, and takes steps to hold these 'A players' to high standards through an explicit process to determine the factors that differentiate high and low performance in each position and then measure people against those criteria.

"The company last year developed a series of ten leadership attributes - such as the abilities to form partnerships with clients and to take strategic risks - each of which is measured on a four-point scale delineated with clear behavioural benchmarks," informs a December 2005 article included in Harvard Business Review on Talent Management (www.tatamcgrawhill.com).

Drawing on the strengths and weaknesses revealed in their evaluations and with the help of tools available on the company's intranet, people in IBM's A positions are required to put together a development programme for themselves in each of the ten leadership areas, the authors continue.

To support the disproportionate investment made in developing talent, the compensation system is also disproportionate. "Today, annual salary increases go to only about half the workforce, and the best-performing employees get raises roughly three times as high as those received by the simply strong performers."

Contd...

Notes

Compelling Presentation

Skills have a shelf Life

The debate on offshoring and migration needs to be considered within the context of how work itself is changing, says Mark Kobayashi-Hillary in Who Moved My Job? A tale for anyone with a job that has yet to move (www.vivagroupindia.com). There is always an alternative future, though it may not be obvious, he cheers.

"Remember when the career you trained for, and possibly went to university to study for, would remain unchanged for a lifetime? Now it can be only be taken as read that many skills learned in one decade are outdated in the next. A university degree is no longer a one-way ticket to career success."

The author exhorts those in mature societies to reconsider their expectations. "Our systems of education, social security, and income tax may need to be scrapped and rebuilt.

Our ideas that the state itself may entirely support pension age citizens will almost certainly need to be scrapped."

He also recommends that the concept of schooling and education be geared towards life-long pursuit, rather than something that we can opt out at sixteen.

2.2 Summary

- The word 'simulation' is frequently used alone in a variety of contexts.
- Simulation should be used when the penalties of a proposed action, plan or design cannot be directly and right away observed.
- Continuous System Simulation illustrates analytically and systematically how mathematical models of dynamic systems, typically described by sets of either ordinary or partial differential equations perhaps attached with algebraic equations, can be simulated on a digital computer.

2.3 Keywords

Pure Pursuit: It is a tracking algorithm that functions by scheming the curvature that will shift a vehicle from its existing position to some objective position.

Relationships: Relationships are the associations that occur between elements and attributes.

2.4 Self Assessment

Fill in the blanks:

1. Relationships are the ..... that occur between elements and attributes.
2. ..... is a system that can be intelligently manipulated by the action of humans.
3. ..... are conceptualized as purposeful systems whose parts have no purposes of their own.
4. ..... represents the behaviour of the system in response to external and internal events
5. Semantic Data Models are used to describe the ..... structure of data processed by the system

Notes

6. ..... are lists of all of the names used in the system models.
7. Selecting an ..... strategy for the simulation experiment needs an intimate knowledge of the particular nature of the business being examined.
8. ..... simulation permits you deal with this vagueness in a quantifiable manner.
9. Numerical integration methods like Range Kutta, or ..... are used to elucidate the system of ODEs.
10. Continuous System Simulation illustrates analytically and systematically how mathematical models of …… systems.
11. The first reaction is very rapid, which means that the …… rate is restricted by the extent of mixing.
12. …….. simulation languages are a dominant tool for solving simulation problems.
13. The calculating time for a …….. formula is considerably larger and for this reason multistep formulas, which involve the predictor- corrector scheme, are however appealing.
14. Derivative terms mentioned explicitly from known inputs and other ……. outputs.

2.5 Review Questions

1. Examine how chemical reactor unite different reactants to form the extremely popular chemical products.
2. Scrutinize the different types of systems.
3. Make distinctions between Numerical Integration and Continuous system Simulation. What are the methods used in numerical integration?
4. "Modern modelling and simulation environments alleviate the occasional user from having to understand how simulation actually works." Comment.
5. "Continuous System Simulation is written by engineers for engineers." Elaborate.
6. Do you think in continuous simulation, the continuous time reply of a physical system is modeled using ODEs? Give Reasons.
7. Find out the difference between Numerical Integration and Continuous System Simulation. Explain with examples.
8. Newton's 2nd law, F = ma, is a superior example of a single ODE continuous system. Explain.
9. Discuss the selection of Integration Formulas.
10. "Continuous System Simulation is a extremely software-oriented text, depending on MATLAB." Elaborate.

Answers: Self Assessment

1.	Associations	2.	Control System
3.	Animate Systems	4.	State Machine Models
5.	Logical	6.	Data dictionaries

Notes

7. inventory control
8. Probabilistic
9. Bulirsch-Stoer
10. dynamic
11. real reaction
12. Block-oriented
13. Runge-Kutta
14. ODE

2.6 Further Readings

Books

Balci, O.,(1994), Validation, verification, and testing techniques throughout the life cycle of a simulation study, Annals of Operations Research, 53: 121–173.

Balci, O., (2001), *A methodology for certification of modelling and simulation applications, ACM Transactions on Modelling and Computer Simulation, 11*: 352–377.

Birta, L.G. and Ozmizrak, N.F., (1996), *A knowledge-based approach for the validation of simulation models: The foundation, ACM Transactions on Modelling and Computer Simulation*, 6: 67–98.

Boehm, B.W., (1979), *Software engineering: R&D trends and defense needs, in: P.* Wegner (Ed.), Research Directions in Software Technology, MIT Press, and Cambridge, MA.

Cellier, F.E., (1986), Combined discrete/continuous system simulation – application, techniques and tools, in Proceedings of the 1986 Winter Simulation Conference.

Department of Defense (DoD) Recommended Practices Guide (RPG) for Modelling and Simulation VV&A, Millennium Edition (available at http://vva.dmso.mil).

General Accounting Office, (1976), Report to the Congress: Ways to improve management of federally funded computerized models, report LCD-75-111, U.S. General Accounting Office, Washington, DC.

General Accounting Office, (1979), Guidelines for model evaluation, report PAD-79-17, U.S. General Accounting Office, Washington, DC.

General Accounting Office, (1987), DOD simulations: Improved assessment procedures would increase the credibility of results, report GAO/PEMD-88-3, U.S. General Accounting Office, and Washington, DC.

Ören, T.I., (1981), Concepts and criteria to access acceptability of simulation studies, Communications of the ACM, 24: 180–189.

Online links

http://www.bakker.org/cfm/webdoc9.htm

http://www.inf.ethz.ch/personal/cellier/Pubs/Sim/springer_2.html

http://pubs.rsc.org/en/content/articlelanding/2001/cp/b106075a

Unit 3: Simulation of Continuous System (II)

Notes

CONTENTS

Objectives

Introduction

3.1 Simulation of a Servo System

3.2 Simulation of Water Reservoir System

3.3 Analog and Digital Simulation

3.4 Summary

3.5 Keywords

3.6 Self Assessment

3.7 Review Questions

3.8 Further Readings

Objectives

After studying this unit, you will be able to:

- Understand the Simulation of servo systems
- Describe the simulation of water reservoir system

Introduction

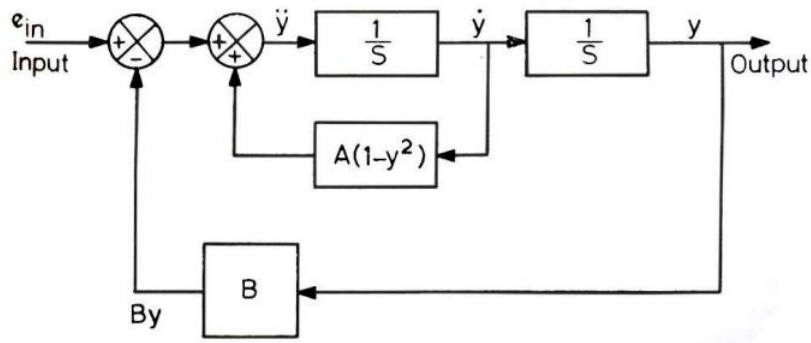
Servos & Simulation is the premiere manufacturer of Control Loading and Motion Systems for all applications. These applications include servo systems, simulation, flight simulators, control loaders, control loading, motion systems, motion bases, motion seats, night vision systems, stable platforms, E/O systems, electro-optics, virtual reality, entertainment systems and gimbal systems. Reservoir simulation is an area of reservoir engineering in which computer models are used to predict the flow of fluids (typically, oil, water, and gas) through porous media.

3.1 Simulation of a Servo System

A very important application of continuous system simulation is in design and analysis of control systems. Let us study the behaviour of a secondorder nonlinear feedback system represented by the following block diagram.

Notes

Figure 3.1: A Non-linear second-order Servo System



(Those of you not familiar with the control theory symbols may ignore the diagram and simply start from the differential equation.)

This block diagram represents many natural as well as man-made servo systems. Some examples are: beating of the heart, periodic opening and closing of flowers in response to the sunlight, rate of variation of prices, squeaking of door with rusty hinges, dripping of a leaky tap, a neon-lamp oscillator, to name a few.

The system of Figure 3.1 can also be described by the following differential equation:

$$\ddot{y} = A(1 - y^2)\dot{y} - By + e_{in}$$

where A and B are positive constants.

In the case of zero input signal, the equation becomes

$$\ddot{y} = A(1 - y^2)\dot{y} - By \qquad ...(1)$$

This is the well-known Van der Pol non-linear equation. (It can be seen that when the amplitude y is small, the damping term $A(l - y^2)$ is negative, but when y becomes large the damping becomes positive. Thus small-amplitude oscillations will tend to build up, while large-amplitude oscillations will be damped out.)

Let us simulate this system. The second-order differential equation can be written as a set of two simultaneous equations of first order. We replace Eq. (1) with (using the variable y_1 in place of y)

$$\dot{y}_1 = y_2$$

$$\dot{y}_2 = A(l - Y_1^2)Y_2 - BY_1$$

To be more specific let constants A = 0.1, B = l.0 and let the initial conditions be

$$y_1(0) = l.0$$

$$y_2(0) = 0.$$

Our equations therefore become

$$\dot{y}_1 = y_2$$

and

$$\dot{y}_2 = 0.1(l - Y_1^2)Y_2 - Y_1 \qquad ... (2)$$

Notes

An instantaneous description of the state of the system is given by the outputs of the two integrators, i.e., by variables y_1 and y_2. We will use, once again, the fourth-order Runge-Kutta method to obtain the values of y_1 and y_2 as a function of time. We will choose the step-size $\Delta t = H = .001$ second and simulate the system for 5 seconds. Thus the number of steps will be N = 5,000. This is too large a number of outputs to be plotted or examined. We will, therefore, print out the values of y_1 and y_2 only once every 100 integration steps. This can be implemented by keeping a counter K which is decremented by I for each integration step. Every time K equals zero, y_1 and y_2 are printed and K is reset to 100. The following FORTRAN program performs the simulation.

```
C       H IS TIME STEP, N IS NO. OF STEPS, Yl, Y2 INITIAL VALUES
        T = O.
        YI = 1.
        Y2 = 0.
        H = .001
        N = 5000
        K= 1
        DO 120 I – I, N
        K = K – I
        IF (K.NE.O) GO TO 110
C       PRINT ONCE IN 100 STEPS
        PRINT, T, YI, Y2
        K = 100
C       CALCULATE THE RUNGE-KUTTA TERMS
  110   Ull = H*Y2
        UI2 = H*(.1*(1. – Y1*Y1)*Y2 – Y1)
        U21 = H*(Y2*+ .5*UI2)
        U22 = H*(.1 *(l. – (YI +.5*U11)*(Y1 +.5*U11)*(Y2+.5*U12) – (Y1 +.5*U11)
        U31 = H*(Y2 + .5*U22)
        U32 = H*(.1 *(1. – (Y1 +.5*U2l)*(Y1 + .5*U21))*(Y2+ .5*U22) – (Y1 + .5*U21)
        U41 = H*(Y2 + U32)
        U42 = H*(.I*(1. – (Y1 + U31)*(Y1 + U3l)*(Y2 + U32) – (Yl + U31)
C       CALCULATE Y1 AND Y2
        Yl = Y1 + (U11 + 2.*U21 + 2.*U31 + U4l)/6.
        Y2 = Y2 + (U12 + 2.*U22 + 2.*U32 + U42)/6.
        T = T + H
120 CONTINUE
        PRINT, T, YI, Y2
        STOP
        END
```

By studying the three examples in this unit so far, you may have acquired the incorrect impression that (i) every dynamic continuous system must first be expressed as a set of simultaneous differential equations before being simulated and, that (ii) such a system is always deterministic. The next example is meant precisely to dispel this notion. Moreover, in this example we are simulating a system which is too large and expensive to ex · periment with and where an incorrect design could become very costly.

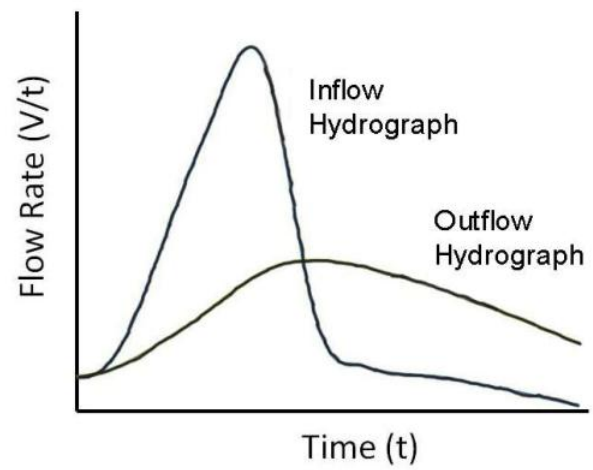
Notes

3.2 Simulation of Water Reservoir System

Introduction

Reservoir storage is essential to regulate highly variable water flows for more steady uses like municipal and industrial water supply, irrigation, hydroelectric power generation, and navigation. Usually, the water drawn from a reservoir is used at a much slower (and constant) rate than the rate and constancy of the water flowing into the reservoir (see Figure 3.2). Reservoir modelling has naturally been employed to assist size reservoir storage capacities, launching operating policies, assessing operating plans, administering water allocations, developing management strategies, and real-time operations.

Figure 3.2: Inflow and Outflow Hydrograph



The basic prerequisite for adequate illustration of a reservoir is employment of the continuity equation, or conservation of volume over a period of time. This is a function that communicates dynamically with the existing state of the reservoir. The foundational equation for preservation of volume is:

$$h\frac{S_{t+\Delta t} - S_t}{\Delta t} = \frac{I_t + I_{t+\Delta t}}{2} - \frac{O_t + O_{t+\Delta t}}{2}$$

The term "Reservoir System Operations" points to the practice of preserving and supervising a reservoir for multiple purposes, under dynamic conditions. The word "system" is used due to the complexity inherent in the operations of a usual reservoir or network of reservoirs. The state of the reservoir system is frequently in flux, requiring dynamic methods of simulation to estimate and model them. The term "Reservoir System Operations Model" refers to a computer program used for simulating and optimizing changes in storage, water deliveries, and flood control for one or numerous reservoirs.

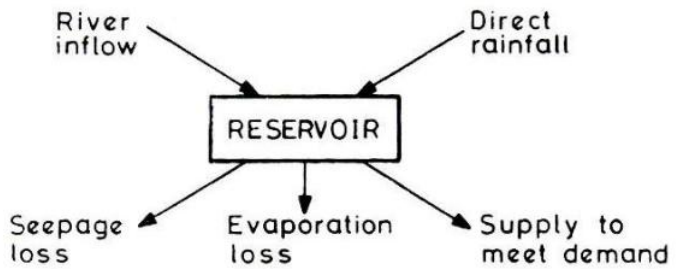
Time and again, the purpose of the reservoir operation is to balance the control of flood storage and preserve reliable water supply. Operational procedures are diverse for flood events than what are employed under water scarce conditions and thus, the model must be tailored for these changing conditions.

Notes

Caution To better administer potential changes to reservoir operations given worries or changes in circumstances, it is helpful to build up a calibrated simulation model of the reservoir

Let us consider the following proposal for constructing a dam across a river to create a reservoir. The reservoir is to be constructed at a specified site. The curve of the projected demand for the water from the reservoir has been determined (from the expected growth pattern and the seasonal fluctuations). The input to the reservoir is from the river inflow and from the rainfall directly over the reservoir. The output consists of the seepage and evaporation losses, in addition to the water supplied to meet the projected demand. This system (called a simple run-of-river storage demand system) is represented symbolically in Figure 3.3.

Figure 3.3: A Simple run-of-river Storage Demand System



The amount of seepage loss is not a constant but depends on the volume of the water stored. We have been given a curve (converted into a table) showing the seepage loss as a function of volume for the proposed reservoir. Likewise, the evaporation loss depends on the area of the exposed surface and the coefficient of evaporation. We are given another curve showing the surface area as a function of volume as well as the seasonal variation of the coefficient of evaporation. Therefore, for a given volume of water in the reservoir at a particular time of the year we can calculate the two losses.

In reality no reasonable finite-sized reservoir can provide an absolute guarantee of meeting the demand 100 % of the time because the river inflow, the rainfall, the losses, the demand are all random variables. To build such a large dam which will never fail (to meet the demand) through its entire life will generally be uneconomical. Therefore, in practice one determines the reservoir size which will meet the demand with a specified risk of failure (of water shortage). For example, a 2 % failure means that once in 50 years the reservoir would become empty before meeting the demand for water. The objective of the study is to determine the size of the reservoir with a specified risk of failure.

There is a single state variable in this system, namely, the volume of water in the reservoir. Since the volume varies continuously with time, we are dealing with a continuous system. It is reasonable to take one month as the basic time interval for the simulation study. Thus, for example, if we wish to simulate the system for 100 years, the simulation run length will be 1200. The simulation will be repeated assuming several different capacities of the reservoir. The output will be in series of ranked shortages for each capacity.

The basic procedure, to be repeated for each time step, may be expressed in terms of the following steps:

(l) For the current month M of the current year IY determine the total amount of river inflow and the total rainfall directly over the reservoir. Let the sum of two inputs be denoted by VIN (= RAIN + RFLOW).

Notes

(2) Add the input volume VIN to the volume left over in the reservoir at the end of the last month, VOL (m – 1). This gives us the gross volume, GROSSV = VIN + VOL(m – 1).

(3) On the basis of the last month's volume VOL (m – 1) calculate this month's seepage and evaporation losses and add them as total loss TLOSS = SEEP + EVAP.

(4) From the demand curve (stored as a table in the computer memory) determine the demand of water for the current month DEM.

(5) If the TLOSS ≥ GROSSY, then the reservoir runs dry without supplying any water and therefore shortage, SHORT = DEM. The volume of water left at the end of the current month VOL (m) = 0. Spillage SPILL = 0 and go co Step 8; else if TLOSS < GROSSY then the net water volume available to satisfy the demand is YNET = GROSSV – TLOSS.

(6) If DEM ≥ VNET the reservoir runs dry and the shortage is given by SHORT = DEM – VNET, and SPILL = 0. Go to Step 8; else if DEM < VNET, then the difference DIFF = VNET – DEM is the water left over.

(7) If this leftover water exceeds the capacity CAP of the reservoir there will be a spill over, i.e., if DIFF > CAP then SPILL = DIFF – CAP and VOL (m) = CAP; else if DIFF ≤ CAP then SPILL = 0 and VOL (m) = DIFF.

(8) Print out SPILL and SHORT for this month, and move to the next month. If the period exceeds the intended simulation length stop, else go to Step I.

The following format-free FORTRAN program implements these steps:

```
      READ, N, VOL, CAP
      1Y = 1
      M = 1
      DO 30 IY = 1, N
      DO 30 M =1, 12
      SPILL = 0
      SHORT = 0.
      CALL RIVFLO (IY,M,RFLOW)
      CALL RA1NF (IY,M,RAIN)
      VIN = RFLOW + RAIN
      GROSSY = VOL + VIN
      CALL SEPEJ (VOL,SEEP)
      CALL EVPRSN (M,YOL, EVAP)
      TLOSS = SEEP + EVAP
      CALL DEMAND (IY,M,DEM)
      VNET = GROSSV – TLOSS
      IF (VNET .LE. O.) VNET = 0.
      DIFF = VNET - DEM
      IF (DIFF .GE. O.) GO TO 10
      SHORT = –DIFF
      VOL = 0.
      GO TO 20
10    VOL = DIFF
      IF (CAP .GT. DIFF) GO TO 20
      SPILL – DIFF – CAP
      VOL = CAP
```

Notes

```
20      PRINT, FLOW, RAIN, TLOSS, SHORT, SPILL, VOL
30      CONTINUE
        STOP
        END
```

Subroutines

The foregoing program contains five subroutines requiring data about the riverflow, rainfall and the demand as a function of time; the seepage loss as a function of water stored in the reservoir; and the evaporation loss as a function of the volume (and hence the exposed surface) and the particular month of the year. The long sequence of data for river inflow and the rainfall could either be obtained from historical records or generated using suitable pseudorandom number generators. Similarly we can design the other subroutines, from an intimate knowledge of the system. As an example, we will write down the subroutine EVPRSN, for computing the evaporation loss.

Let the unit of measuring the volume be a million cubic meters. Suppose the highest possible dam at this site will create a reservoir of capacity 1000 units. Also suppose that we have a curve that gives the exposed surface area as a function of volume from 0 to 1000 units. Let the x-axis be divided into 100 equispaced ranges; and the data be stored in the form of a table (SURTBL) with 100 columns giving the surface area at volume 10, 20, ... , 1000 units. The surface area SAREA for any intermediate value of VOL can be computed using an appropriate interpolation formula. Let us assume that a linear interpolation will suffice. We are also given 12 values for the coefficient of evaporation COEF – one for each month of the year. Then the following subroutine will yield the evaporation loss.

```
SUBROUTINE EVPRSN (M, VOL, EVAP)
REAL SURTBL (100), COEF(12)
DATA SURTBL /...,        ...,        .../
DATA COEF /...,   ...,        .../
IVOL = VOL/10.
RVOL = IVOL
FRAC = VOL/10. – R VOL
SAREA = SURTBL (IVOL) + FRAC*(SURTBL(IVOL+l) – SURTBL(IVOL))
EVAP = SAREA * COEF (M)
RETURN
END
```

Note that the third statement requires 100 values and the fourth statement requires 12 values. Other subroutines can be written down similarly.

Output: The output will be a series of monthly shortages and spills.

The shortages could be combined into total annual shortages. These annual shortages can be ranked according to the amount of shortage involved. From this ranked series of shortages for each capacity of the reservoir we would determine the acceptable reservoir size.

In the foregoing model no distinction was made regarding how the shortage is distributed month-wise within a particular year. For example, the total failure to meet any demand for one month may be more serious than a 10 per cent shortage for ten consecutive months. Such refinement can be easily incorporated by a procedure of assigning weights to different types of failures within a particular year.

Notes

There is another refinement which should be made in our computation. The losses for the entire current month were computed on the basis of the volume in storage at the end of last month. If a large change in volume takes place between two consecutive time steps, this would lead to errors. This could be corrected by first finding a temporary value of VOL (m) on the basis of given VOL (m – I) (as usual) and then recalculating the losses on the basis of the average value of the two volumes.

$$\frac{VOL(m) + VOL(m - l)}{2}$$

3.3 Analog and Digital Simulation

The simulations will be performed using the commercial software, "Electronics Workbench", which is mounted on the laboratory computers. With some research before the lab, you should be able to do both an analog and a digital simulation. The simulations execute the circuits in software and all voltages, currents, and waveforms can be scrutinized and printed out.

The circuits are entered by graphic capture, i.e. you draw the circuit on the computer screen by putting the parts and linking them with wires. Once the circuit is entered, an investigation is run. There are numerous kinds of analyses including DC conditions, small signal transfer functions, and digital logic states for the case of digital circuits.

Analog Simulation

The analog simulation circuit will be the difference amplifier from the difference amplifier lab.

Before the lab organize a table of of the measured and computed DC node voltages, the sum and difference gains, CMRR, and the amplifier high and low roll-off frequencies. You should have approximately all this information in your laboratory notebook. The simulated circuit should have the same constituent values used in the lab circuit. The consequences of the simulation should then be entered as a separate column in your table.

1. Simulate the difference amplifier circuit by means of the circuit you built for the Difference Amplifier lab. The constituent values should be indistinguishable to those used in the lab. We use the Electronics Workbench application.
2. The first step is to make a schematic diagram.
3. Perform a DC analysis. Check the results against the expected values.
4. Execute a Transient analysis for 5 cycles and compute the difference and common mode gains. Use the same frequency as you used in the lab. Use the investigation Graphs window to print out the desired waveforms.
5. Execute a frequency analysis and establish the low and high rolloff frequencies. Print out the Bode plot using the Analysis Graphs window.
6. Present the consequences as a three column table: theory, measurement, and simulation for the requested quantities. Most people did not calculate the frequency dependence in lab, but you should be able to evaluate the roll-off frequencies.
7. The final consequences will include the table, a copy of the circuit diagram, waveforms, and your explanations as to why the simulation might be providing different results from theory and measurement.

Digital Simulation

Notes

You will simulate a 4 bit Successive Approximation ADC prepared up of a DAC, comparator, D flip flops, J-K flip flops, and NAND gates. You will enter a DC input voltage, run the simulation, and read out the digital state of the circuit under the control of CLK pulses. The circuit also involves START and CLEAR lines.

1. Enter the circuit via schematic capture.
2. The initial conditions for the various lines are entered using the Word Generator tool, which permits you to enter the CLK, START and CLEAR pulses and execute them as a sequence either step by step or as an automatic series at a specified frequency.
3. Use the Logic Analyzer tool to exhibit the circuit output. This permits you to display the waveform on up to 16 channels concurrently. The analog output can be viewed on the Oscilloscope instrument. The DAC ouput is Vout = Vref × D/256 where Vref = 5 V. The binary outputs can also be viewed with LEDs which light up on a logical "1".
4. Make a plot of the DAC output as a function of consecutive clock pulses. You should observe the DAC output congregate to the analog input voltage with an accuracy reliable with the 4bits of the circuit, i.e. one part in fifteen. The digital result exists in the four D flip flops.

   Try numerous different DC input voltages.

Case Study

Indiasoft Tech to Market Simulation Tool

The Pune-based Indiasoft Technologies and the UK-based software developer Lanner launched Witness 2006, a modelling-cum-simulation software targeted mainly at manufacturing companies in Chennai.

The product has been developed by Lanner and Indiasoft Technologies will distribute and maintain the product in India.

The software will focus on optimising workflow in a manufacturing organisation. Using simulation tools, the software will arrive at the most efficient workflow or process for the company, said Mr R.K. Irani, Director- Operations, Indiasoft Technologies. The company is aiming to address supply chain logistics process that is inefficient in most organisations, he said.

Six modules of the software are available. The basic one "Witness 2006", is a modelling and simulation tool, the next one Optimiser helps optimise results derived from Witness 2006. "We expect to see maximum demand for these two modules," said Mr Irani.

The other modules are Virtual Reality (VR) that helps create a 3D virtual plant complete from inventory to finished product, Documentor (that stores the logic of processes and other results), Witness Miner (for data mining) and Witness SDX (a communicative tool that links Witness with Factory CAD).

Witness 2006 has a standalone and a Network (Enterprise) edition and is priced between ıo 13 lakh and ıo 23 lakh, depending on the modules incorporated.

The company is looking at discrete component manufacturers, specifically in the automotive, mechanical and aerospace division.

Indiasoft's clients include Bajaj Auto, Varroc Group and SKF. The company recently distributed Witness-2006 to eight educational institutes.

Source: http://www.thehindubusinessline.in/2006/07/07/stories/2006070703320400.htm

Notes

3.4 Summary

- Numerical integration methods like Runge Kutta, or Bulirsch-Stoer are used to elucidate the system of ODEs.
- A servo control system is one of the most significant and extensively used forms of control system. Any machine or part of equipment that has rotating parts will enclose one or more servo control systems.
- The main use of a reservoir is to control a determined quantity of water during some phase of time.

3.5 Keywords

Simul8: Standard is an incorporated environment for functioning with simulation models.

System: A system is an assemblage of interrelated parts that work together by way of some driving process.

3.6 Self Assessment

Fill in the blanks:

1. The use of real equipment like the …… Servo Trainer aids enormously to locate the special features of the different techniques.
2. A continuous time technique or a …… method based on Z transforms.
3. The water drawn from a reservoir is used at a much slower (and constant) rate than the rate and constancy of the water flowing into the ……
4. The term "Reservoir System Operations" points to the practice of …… and supervising a reservoir for multiple purposes, under dynamic conditions.
5. Reservoirs are operated depending on policies that engross …… pools that are defined to be used for dissimilar purposes.
6. The …… control zone is to stay empty except during the times following a flood event upstream of the reservoir.
7. The flood zone is usually drained in a controlled way through use of an outlet works with an operated gate or ……
8. The conservation pool is used to stock up water temporarily for ……
9. The top of the conservation pool performs as a guide, target, or …… for operators.

10 Water supply allocation is the execution of a permit system and adjudication of ……

11. …… flow programming might also be suitable as this is an competent form of LP which can be used to symbolize a network of nodes and links similar to a reservoir/water demand network.
12. It is easy to discharge …… from an upstream reservoir to a downstream reservoir but not the other way around.
13. The purpose for reservoirs in parallel should be to …… storage depletions among the two.

Notes

14. The …… simulation circuit will be the difference amplifier from the difference amplifier lab.
15. The simulated circuit should have the same …… values used in the lab circuit.

3.7 Review Questions

1. Study the use of Servo System Controllers in organizing a servo mechanism.
2. What is the difference between analog and digital simulation in your opinion?
3. "A servo control system is one of the most significant and extensively used forms of control system." Comment.
4. "Manufacturing industry would come to an end without servo systems." Explain
5. What are Servo System Controllers?
6. Do you think Reservoir storage is essential to regulate highly variable water flows for more steady uses? Give reasons.
7. The main use of a reservoir is to control a determined quantity of water during some phase of time. Why?
8. Explain the interaction with groundwater in bank storage.
9. Discuss Multiple Reservoirs in Series.
10. Make the analysis and find out the difference between analog and digital simulation.

Answers: Self Assessment

1.	CE110	2.	Digital
3.	Reservoir	4.	Preserving
5.	Multiple	6.	Flood
7.	Regulator	8.	Downstream
9.	rule curve	10.	water rights
11.	Network	12.	Water
13.	Balance	14.	Analog
15.	constituent		

3.8 Further Readings

Books

Balci, O.,(1994), Validation, verification, and testing techniques throughout the life cycle of a simulation study, Annals of Operations Research, 53: 121–173.

Balci, O., (2001), *A methodology for certification of modelling and simulation applications, ACM Transactions on Modelling and Computer Simulation, 11*: 352–377.

Birta, L.G. and Ozmizrak, N.F., (1996), *A knowledge-based approach for the validation of simulation models: The foundation, ACM Transactions on Modelling and Computer Simulation*, 6: 67–98.

Notes

Boehm, B.W., (1979), *Software engineering: R&D trends and defense needs, in: P.* Wegner (Ed.), Research Directions in Software Technology, MIT Press, and Cambridge, MA.

Cellier, F.E., (1986), Combined discrete/continuous system simulation – application, techniques and tools, in Proceedings of the 1986 Winter Simulation Conference.

Department of Defense (DoD) Recommended Practices Guide (RPG) for Modelling and Simulation VV&A, Millennium Edition (available at http://vva.dmso.mil).

General Accounting Office, (1976), Report to the Congress: Ways to improve management of federally funded computerized models, report LCD-75-111, U.S. General Accounting Office, Washington, DC.

General Accounting Office, (1979), Guidelines for model evaluation, report PAD-79-17, U.S. General Accounting Office, Washington, DC.

General Accounting Office, (1987), DOD simulations: Improved assessment procedures would increase the credibility of results, report GAO/PEMD-88-3, U.S. General Accounting Office, and Washington, DC.

Ören, T.I., (1981), Concepts and criteria to access acceptability of simulation studies, Communications of the ACM, 24: 180–189.

Online links

http://www.bakker.org/cfm/webdoc9.htm

http://www.sidman.com/hdd.htm

http://en.wikipedia.org/wiki/Reservoir_simulation

Notes

Unit 4: Discrete System Simulation (I)

CONTENTS

Objectives

Introduction

4.1 Discrete System Simulation

4.1.1 Components of a Discrete-event Simulation

4.1.2 Application Areas/Common Uses

4.2 Fixed Time Step vs Event-to-Event Model

4.3 Summary

4.4 Keywords

4.5 Self Assessment

4.6 Review Questions

4.7 Further Readings

Objectives

After studying this unit, you will be able to:

- Understand fixed time step
- Describe event to event model

Introduction

Discrete Event Simulation (**DES**) concerns the of a system as it evolves over time by representing the changes as separate events. This is the opposite of Continuous Simulation where the system evolves as a continuous function (differential). Fixed time step and even to event are the models for moving a system during time. The simulation needs to generate random variables of various kinds, depending on the system model. References to Monte Carlo simulation are often encountered in the and simulation literature.

4.1 Discrete System Simulation

In discrete-event simulation, the operation of a system is represented as a chronological sequence of events. Each event occurs at an instant in time and marks a change of state in the system.

Example: If an elevator is simulated, an event could be "level 6 button pressed", with the resulting system state of "lift moving" and eventually (unless one chooses to simulate the failure of the lift) "lift at level 6".

A common exercise in learning how to build discrete-event simulations is to model a queue, such as customers arriving at a bank to be served by a teller. In this example, the system entities are CUSTOMER-QUEUE and TELLERS. The system events are CUSTOMER-ARRIVAL and CUSTOMER-DEPARTURE. (The event of TELLER-BEGINS-SERVICE can be part of the logic of the arrival and departure events.) The system states, which are changed by these events, are

Notes

NUMBER-OF-CUSTOMERS-IN-THE-QUEUE (an integer from 0 to n) and TELLER-STATUS (busy or idle). The random variables that need to be characterized to model this system stochastically are CUSTOMER-INTERARRIVAL-TIME and TELLER-SERVICE-TIME.

A number of mechanisms have been proposed for carrying out discrete-event simulation; among them are the event-based, activity-based, process-based and three-phase approaches (Pidd, 1998). The three-phase approach is used by a number of commercial simulation software packages, but from the user's point of view, the specifics of the underlying simulation method are generally hidden.

4.1.1 Components of a Discrete-event Simulation

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

Clock

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

Events List

The simulation maintains at least one list of simulation events. This is sometimes called the pending event set because it lists events that are pending as a result of previously simulated event but have yet to be simulated themselves. An event is described by the time at which it occurs and a type, indicating the code that will be used to simulate that event. It is common for the event code to be parameterised, in which case, the event description also contains parameters to the event code.

When events are instantaneous, activities that extend over time are modeled as sequences of events. Some simulation frameworks allow the time of an event to be specified as an interval, giving the start time and the end time of each event.

Single-threaded simulation engines based on instantaneous events have just one current event. In contrast, multi-threaded simulation engines and simulation engines supporting an interval-based event model may have multiple current events. In both cases, there are significant problems with synchronization between current events.

The pending event set is typically organized as a priority queue, sorted by event time. That is, regardless of the order in which events are added to the event set, they are removed in strictly chronological order. Several general-purpose priority queue algorithms have proven effective for discrete-event simulation, most notably, the splay tree. More recent alternatives include skip lists and calendar queues.

Typically, events are scheduled dynamically as the simulation proceeds.

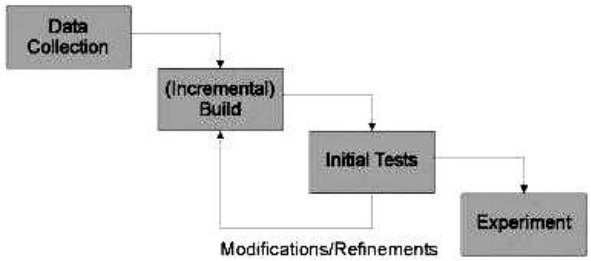
Example: In the bank example noted above, the event CUSTOMER-ARRIVAL at time t would, if the CUSTOMER_QUEUE was empty and TELLER was idle, include the creation of the subsequent event CUSTOMER-DEPARTURE to occur at time t+s, where s is a number generated from the SERVICE-TIME distribution.

Notes

4.1.2 Application Areas/Common Uses

Well-known examples of Simulation are Flight Simulators, Fleet Management and Business games. However, there are a large number of potential areas for Discrete Event Simulation. One of the main areas currently being explored is in designing new manufacturing areas, especially where high capital investment is involved. For example, if a company wishes to build a new production line, then the line can be first simulated to assess feasibility and efficiency. The diagram below shows the key stages in using Discrete Event Simulation. It can be noted that this bears a strong resemblance to other simulation techniques and other analysis program development methodologies (prototype method) [Somerville, 1992].

Figure 4.1: Stages used in Discrete event Simulation



Key Principles

Although, discrete event simulation could conceivably be carried out by hand it can be computationally intensive, therefore will invariably involve computers and software. The software could be a high level programming language such as Pascal or a specialised event/data driven application, such as iBright Ltd's 'baseSim' (Monte Carlo Simulation). The five key features found in the software simulation model are:

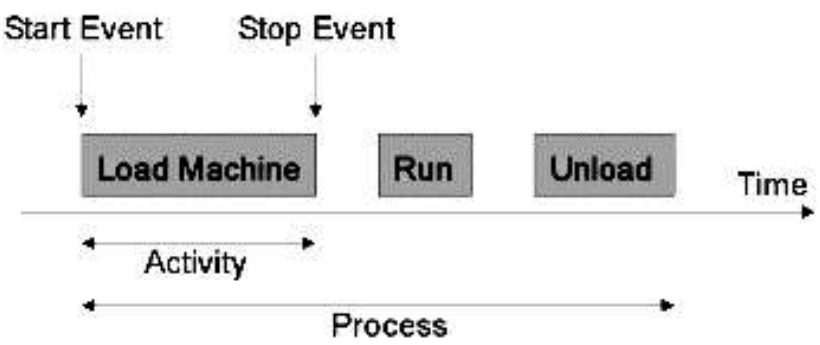
1. ***Entities:*** Representations of real-life elements e.g. in manufacturing these could be parts or machines.
2. ***Relationships:*** Link entities together e.g. a part may be processed by a machine.
3. ***Simulation Executive:*** Responsible for controlling the time advance and executing discrete events.
4. ***Random Number Generator:*** Helps to simulate different data coming into the simulation model. Important that the random data can be reproduced in different simulation runs.
5. ***Results & Statistics:*** Important in validating the model and for providing performance measures.

The simulation executive may operate in one of two manners [Ball, 1996]:

1. ***Time Slicing:*** Advances the model by a fixed amount each time, regardless of the absence of any events to carry out.
2. ***Next Event:*** Advances the model to the next event to be executed, regardless of the time interval. This method is more efficient than Time Slicing, especially where events are infrequent, but can be confusing when being represented graphically (processes that take different times will appear to happen in the same time frame if the stop event is the next event after the start event).

Notes

Figure 4.2: Approaches to Discrete Event Simulation



There are also three approaches to describing the discrete simulation, see the Diagram above [Pidd, 1992].

1. ***Event:*** This approach describes an instantaneous change, usually from a stop event to a start event. This is the most common one used, easy to understand and efficient and is acceptable to implement.
2. ***Activities:*** Represents duration. Essentially groups a number of events in order to describe an activity carried out by an entity e.g. a machine loading. This approach is easy to understand and to implement but is not efficient.
3. ***Process:*** These approach groups activities to describe the life cycle of an entity e.g. a machine. This is less common and more difficult to plan and implement, but is generally thought to be the most efficient.

Tasks

Define the most common and easy to understand approach used in describing the discrete simulation.

More Common uses of DES

Diagnosing Process Issues

Simulation approaches are particularly well equipped to help users diagnose issues in complex environments. The Goal (Theory of Constraints) illustrates the importance of understanding bottlenecks in a system. Only process 'improvements' at the bottlenecks will actually improve the overall system. In many organizations bottlenecks become hidden by excess inventory, overproduction, variability in processes and variability in routing or sequencing. By accurately documenting the system inside a simulation model it is possible to gain a bird's eye view of the entire system.

A working model of a system allows management to understand performance drivers. A simulation can be built to include any number of performance KPIs such as: worker utilization, on-time delivery rate, scrap rate, cash cycles, and so on.

Notes

Hospital Applications

An operating theater is generally shared between several surgical disciplines. Through better understanding the nature of these procedures it may be possible to increase the patient throughput. Example: If a heart surgery takes on average four hours, changing an operating room schedule from eight available hours to nine will not increase patient throughput. On the other hand, if a hernia procedure takes on average twenty minutes providing an extra hour may also not yield any increased throughput if the capacity and average time spent in the recovery room is not considered.

Custom Order Environments

Many systems show very different characteristics from day to day depending on the order mix. Many small orders may cause bottle-necks due to excess changeovers. Large custom orders may require extra processing at a point where the system has particularly low capacity. Simulation allows management to understand what changes 'on average' would have the largest impact and greatest return-on-investment.

Lab Test Performance Improvement Ideas

Many systems improvement ideas are build on sound principles, proven methodologies (Lean, Six Sigma, TQM, etc.) yet fail to improve the overall system. A simulation model allows the user to understand and test a performance improvement idea in the context of the overall system.

Evaluating Capital Investment Decisions

Simulation is commonly used to model potential investments. Through investments decision-makers can make informed decisions and evaluate potential alternatives.

Often these decisions look at altering existing operations. Typically, a model of the current state is constructed. This 'current state' model is tested and validated against historical data. Once the model is operating correctly, the simulation is altered to reflect the proposed capital investments. This 'future state' model is then stress-tested to ensure the alterations perform as desired.

Occasionally, organizations take on entirely new operations processes. These could be new Lean facilities, designed around new products or using new technology. In these cases only a 'future state' model is constructed. The testing and validation may require more analysis. There are companies and experts that specialize in simulation building who may be brought in to help.

Stress Test a System

Models can be used to understand how a system will be able to weather extraordinary conditions. A simulation can help management understand: large increases in orders, significant swings in product mix, new client delivery demands (i.e. 1 week lead times), and economic events (i.e. a multinational with operations in South America and Asia sees significant swings in currencies).

Did u know? Full form of DES

DES: Discrete Event Simulation

Notes

Visualisation

Visual Interactive Simulation (VIS) has been available since the late 1970's. Before this simulation models were simply 'black boxes' - data going in and results coming out. In such a scenario establishing credibility and confidence in the simulation model would not have been easy.

Using on-screen animations in a simulation model enables the status of the model to be viewed as it progresses e.g. a machine that breaks down may change its color to red. This enables visual cues to be passed back to the operator of the simulation model, so action could be taken. Additionally, visualisation is useful in convincing management of the model's credibility. For example, in manufacturing if the Directors can see a visualisation of the production line with widgets travelling down a conveyor belt, it would do more to sell the concept of the model than a 'black box', churning out data.

With VIS the prime motivation is not only portrayal of the running simulation model but also the interaction with it. For example, in using the above scenario, if the User wanted to see how the production line would run with an extra machine then he could simply 'plug in' a machine, at the appropriate position, and monitor the effect that this would have on the model.

Visual Interactive (VIM) takes this concept one stage further by allowing the model to be created interactively. This allows a model to be constructed by dragging (with a mouse) 'Entities' (machines, parts etc.) from a library onto a frame. The entities could then be connected in the desired order. Many of the advanced VIM simulation tools allow program code to be attached to the entities and events, therefore making the model potentially more sophisticated and flexible.

Visualisation and simulation are extensively used in the training of operational staff, especially where the training cannot be carried out in real life

Example: Shutting down the reactor of a nuclear power station after an earthquake.

Object Oriented Simulation

Object Oriented techniques have been developed since the early 1960's as a result of simulation development (SIMULA). Until recently, the two were not coupled despite their original tandem development. There are currently only a handful of Object Oriented simulation applications available on a commercial basis; one of the most prominent of these is iBright Ltd's 'baseSim'.

The main difference between traditional program development and Object Oriented techniques is the way in which the data and the program code are stored and manipulated. In traditional software, the data and the program code are intermingled throughout the program, making data security and integrity difficult to achieve (it is sometimes possible for one procedure to cause knock-on effects as global data is changed). However, in Object Oriented simulation software all data and procedures relating to a single entity (object) are encapsulated within an object, with the object controlling its own interaction and data integrity permissions with other objects. Clearly, the methods inside the object could cause similar knock-on effects, if poorly implemented.

Object Oriented simulation tools, in particular iBright Ltd's 'baseSim', are very powerful as they make use of Object Oriented techniques such as modularity, class structure, inheritance, hierarchy and polymorphism.

Notes

Notes *Object Oriented Simulation* will succeed as a precious resource to students and skillful professionals and researchers alike, as it offers an extensive, however intelligible introduction to the fundamental principles of object-oriented , design and execution of simulation models.

Statement from a Company about Discrete-system Simulation

Our Discrete-system Simulator (DSS) is used by INTRACOM, the largest Greek telecom company. We shall expand DSS (in connection with the DSP library) to allow for the specification, simulation and performance testing of distributed protocols for almost any form of environment, including networks that allow hierarchical description, very fast networks and heterogeneous networks with mobile units. The visualisation facility of DSS will be enhanced so the user can see the protocol execution in very large networks in a meaningful way through, e.g., a hierarchical description and graph-drawing techniques. INTRACOM, and also INTRASOFT (the largest Greek software company), are interested in the new DSS and DSP library for (1) simulating and testing distributed banking applications, and (2) designing hardware for fast networks (routers and switches).

During the first year of ALCOM - IT, we established strong interactions with the industrial link INTRAKOM/INTRASOFT a major Telecom and Systems House conglomerate of Companies. After analysing the users' needs, we identified a large algorithmic area which is a subset of the area of on-line algorithms, namely the area of Call Admission Control (CAC) algorithms. Our group and, in general, the ALCOM team has had already shown significant research on the issue. Since the problem of Admission Control is currently addressing ATM network technology, we decided to explore the transfer of algorithmic engineering technology in that area by specializing and focusing the ALCOM-IT produced DSS tool in the above direction. The new DSS tool aims to provide:

1. An abstract, clean and semantically correct high level model of any ATM network and of the on-line calls for connections.
2. A library of Call Admission Control algorithms in a form that allows them to him easily used by non - experts or to be used and tested by algorithms and networks designers.
3. The capability for the user Industry to design new Call Control protocols and/or test such designs.

The ATM technology is considered as the state of the art network technology that is expected to play an important role in the future networks. The ATM networks are fast packet switching networks achieving their speed by avoiding flow control and error checking at the intermediate nodes in a transmission. ATM operates in a connected mode, but a connection can only be set up and serviced if sufficient resources are available in order to preserve the quality of service to the previous accepted connections. This function is controlled by the Call Admission Control algorithms running in the ATM switches.

DSS provides an abstract model for the description of any ATM network which is independent of details of the underlying technology. It simulates the basic functionality of an ATM network which IS that in each time unit cells are produced from traffic generators or forwarded from the switches to their destination.

Under the scope of DSS an ATM network topology consists of links, ATM switches, terminals (workstation) and call/traffic generators (network applications). The critical characteristics of the topology such as the size of the buffers of the ATM switch, the bandwidth of the links, the

Notes

virtual paths and virtual circuits over the links and the traffic parameters can be defined by the user to approach the behavior of the today's and the future network components. Each ATM switch is modelled as a Communicating Finite State Machine. A receipt of a CAC cell combined with its current state activates an action routine of the CAC algorithm.

Emphasis is given in the abstract of traffic generation. Adversarial traffic leads the on-line algorithms to their worst case competitive ratio of performance (measured against ideal off-line algorithms that know the future). The DSS cannot simulate worst-case adversaries but can approximate their behaviour by exploiting certain distribution of call request of high Kolmogorov complexity. The DSS can of course use externally (pragmatic) generate call sequences.

Tasks Analyze the objectives provided by DSS tool.

4.2 Fixed Time Step vs Event-to-Event Model

In simulating any dynamic system-continuous or discrete-there must be a mechanism for the flow of time. For we must advance time, keep track of the total elapsed time, determine the state of the system at the new point in time, and terminate the simulation when the total elapsed time equals or exceeds the simulation period

In a fixed time-step model a timer is simulated by the computer, this clock is updated by a fixed time interval ?, and the system is examined to see if any event has taken place during this time interval, all events that take place during that interval are treated as if they occurred simultaneously at the end of this interval.

In a next-event or event-to-event model the computer advances time to the occurrence of the next event, thus it shifts from one event to the next the system state does not change in between. When something of interest happens to the system, the current time is kept track of.

The flowcharts for both models are shown.

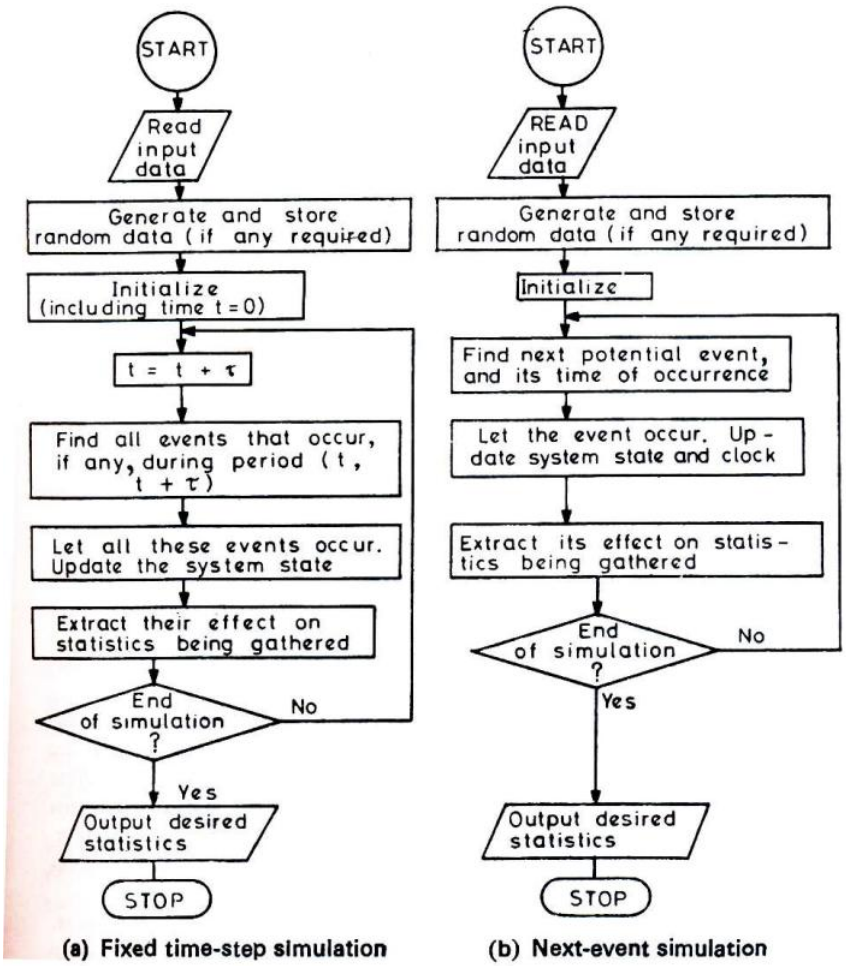
To exemplify the difference between the two models, let us presuppose that we are simulating the dynamics of the population in a fish bowl, starting with, say, 10 fish. If we used the fixed time-step model with, say, or = 1 day, then we would scan the fish-bowl once every 24 hours, and any births and deaths that take place are presumed to be during the last moment of this period. Alternatively, if we use a next-event model then we will first find out when the next-event is to take place and then advance the clock exactly to that time.

In general, the next-event model is preferred, excepting when you may be forced to use the fixed time-step model because you do not waste any computer time in scanning those points in time when nothing takes place.

This waste is bound to occur if we pick a reasonably small value for or. On the other hand, if or is so large that one or more events must take place during each interval then our model becomes unrealistic and may not yield meaningful results. Therefore in most simulations of discrete systems the next `event model is used. The only drawback of the next-event model is that usually its implementation turns out to be more complicated than the fixed time-step model.

Notes

Figure 4.3: (a) Fixed time-step simulation and (b) Next-event simulation



Caselet

War Gaming

It's not animation, it's not gaming. It is war gaming, which is becoming big business for simulation tool vendors.

Companies such as Dassault, UGS, PTC, MultiGen-Paradigm and even local ones such as the Bangalore-based EDS Technologies and Cades work on the exciting domain of simulation and virtual product development.

These companies offer software tools that bring interactive and high fidelity, real-time 3D applications to the computer. With them, designers in automotive and aircraft manufacturing can now get a very specific and detailed idea about their creation.

From how each component works/reacts to exact space and thermal computations to learning the various factors involved in a real-world scenario, these complex tools offer

Contd...

Notes

functions that help make faster flights, safer cars and even a designer uninterruptible power supply system.

Serious business

War gaming is now serious business, says S. Senthil, President of EDS Technologies India, a Rs 65-crore firm that is investing largely in the simulation business. Tactical gunnery trainers and battlefield simulators, night-time warfare training and battle tank training simulation can simplify rigorous processes while planning a coup.

Terms such as rapid prototyping, dynamic texture mapping and terrain forming (taken from satellite imagery) were discussed at Multigen Paradigm User Forum - 06 held in Bangalore recently.

Representatives of defence organisations also attended the event, which showcased some of the most advanced state-of-the-art simulation systems that can be played on a laptop.

These PC-based tactical military trainers can easily be made abreast of the ever-changing simulation requirements of our armed forces for upgrading them in asymmetric warfare and urban operations.

Down to the last detail

Virtual product development and simulation techniques have evolved over the years to become more realistic, and match physical world constraints by allowing the designer to input multiple factors. Now, you can get a scenario where the end product you are designing will be detailed down to its shadows.

The shadows are modelled depending on the time of day, by taking into account the position of the sun (which can be programmed into the software before it `realises' the object. Such attention to detail can be observed now even down to weather effects such as snow or rain.

Using technology such as Blueberry's multi-texture feature, even the dust sprayed when a helicopter lands can be replicated. Such realistic video is now combined with suitable audio effects to give a feel of the war ground to the designer.

Such software is even being used in new verticals such as infrastructure - in town planning and hotel design.

Imagine being able to see a video animation of you walking through your yet-to-be constructed home.

Not only could you choose finer details such as what colour to paint the walls, but also how the plumbing would be fixed and lighting details. Such is the power of today's design software.

Other non-warfare simulation uses are in disaster relief, airport traffic controlling, simulation for various high security installations, 3D sport simulators and driving.

Road-track simulation for driving isn't new, with several driving schools offering learners the opportunity to test their skills at driving before even stepping foot in a car. However, many users are denied its advantage due to its high cost, says Senthil.

Tasks Analyze whether fixed time step is better for games when developing game loops.

Notes

4.3 Summary

- In discrete-event simulation, the operation of a system is represented as a chronological sequence of events. Each event occurs at an instant in time and marks a change of state in the system.
- Although, discrete event simulation could conceivably be carried out by hand it can be computationally intensive, therefore will invariably involve computers and software.
- Using on-screen animations in a simulation model enables the status of the model to be viewed as it progresses e.g. a machine that breaks down may change its color to red. This enables visual cues to be passed back to the operator of the simulation model, so action could be taken.
- The main difference between traditional program development and Object Oriented techniques is the way in which the data and the program code are stored and manipulated.
- In a fixed time step model, a timer or clock is simulated by computer. This clock is modernized by a fixed time interval (say, tau) and the system is observed to view if any event has taken place during this unit time interval.
- In next-event simulation model the computer advances time to the incidence of the subsequent event. So, shift is from event to event.

4.4 Keywords

Discrete-event Simulation: In discrete-event simulation, the operation of a system is represented as a chronological sequence of events.

Entities: Representations of real-life elements e.g. in manufacturing these could be parts or machines.

Fixed Time step Model: In a fixed time step model, a timer or clock is simulated by computer.

4.5 Self Assessment

Fill in the blanks:

1. In ..... simulation, the operation of a system is represented as a chronological sequence of events.
2. The simulation maintains at least one list of simulation events. This is sometimes called the .....
3. ..... is responsible for controlling the time advance and executing discrete events.
4. With ..... the prime motivation is not only portrayal of the running simulation model but also the interaction with it.
5. Our Discrete-system Simulator (DSS) is used by ....., the largest Greek telecom company.
6. DSS provides an abstract model for the description of any ..... network which is independent of details of the underlying technology.
7. The full infection timeline in our malaria models corresponds to the "....." in Macdonald's model.

Notes

8. In discrete-event simulations, as opposed to real time simulations, time 'hops' because events are .....
9. Well-known examples of Simulation are Flight Simulators, Fleet Management and .....
10. ..... could conceivably be carried out by hand it can be computationally intensive, therefore will invariably involve computers and software.
11. Results & Statistics are important in ..... the model and for providing performance measures.
12. Simulation approaches are particularly well equipped to help users ..... issues in complex environments.
13. A working model of a system allows management to understand ..... drivers.
14. Many small orders may cause ..... due to excess changeovers.
15. ..... takes this concept one stage further by allowing the model to be created interactively.

4.6 Review Questions

1. What is Discrete System Simulation? Examine the various components of discrete event simulation.
2. What do you think are the common uses of discrete event simulation?
3. What is Visualization? Explain how does it enables the status of the model to be viewed as it progresses.
4. Make distinctions between traditional program development and Object Oriented techniques.
5. Scrutinize the implementation of Plasmodium Falciparum Malaria in discrete event simulation.
6. Analyze the distinctive points between Fixed Time Step vs. Event-to-Event Model.
7. The simulation must keep track of the current simulation time. Elaborate.
8. Do you think the simulation maintains at least one list of simulation events?
9. Single-threaded simulation engines based on instantaneous events have just one current event. Comment.
10. Explain the key features found in the software simulation model.

Answers: Self Assessment

1.	discrete-event	2.	pending event set
3.	Simulation Executive	4.	visualization interactive simulation(VIS)
5.	INTRACOM	6.	ATM
7.	incubation interval	8.	instantaneous

Notes

9. Business games
10. Discrete event simulation
11. validating
12. diagnose
13. performance
14. bottle-necks
15. Visual Interactive (VIM)

4.7 Further Readings

Books

Balci, O.,(1994), Validation, verification, and testing techniques throughout the life cycle of a simulation study, Annals of Operations Research, 53: 121–173.

Balci, O., (2001), A methodology for certification of and simulation applications, ACM Transactions on and Computer Simulation, 11: 352–377.

Birta, L.G. and Ozmizrak, N.F., (1996), A knowledge-based approach for the validation of simulation models: The foundation, ACM Transactions on and Computer Simulation, 6: 67–98.

Boehm, B.W., (1979), Software engineering: R&D trends and defense needs, in: P. Wegner (Ed.), Research Directions in Software Technology, MIT Press, and Cambridge, MA.

Cellier, F.E., (1986), Combined discrete/continuous system simulation – application, techniques and tools, in Proceedings of the 1986 Winter Simulation Conference.

Department of Defense (DoD) Recommended Practices Guide (RPG) for and Simulation VV&A, Millennium Edition (available at http://vva.dmso.mil).

General Accounting Office, (1976), Report to the Congress: Ways to improve management of federally funded computerized models, report LCD-75-111, U.S. General Accounting Office, Washington, DC.

General Accounting Office, (1979), Guidelines for model evaluation, report PAD-79-17, U.S. General Accounting Office, Washington, DC.

General Accounting Office, (1987), DOD simulations: Improved assessment procedures would increase the credibility of results, report GAO/PEMD-88-3, U.S. General Accounting Office, and Washington, DC.

Ören, T.I., (1981), Concepts and criteria to access acceptability of simulation studies, Communications of the ACM, 24: 180–189.

Online links

http://oxford-mphil.com/images/f/fc/Simulation-econometrics.pdf

http://gafferongames.com/game-physics/fix-your-timestep/

http://e2e.ti.com/support/development_tools/analog_elab_and_tools/f/234/t/74630.aspx

Notes

Unit 5: Discrete System Simulation (II)

CONTENTS

Objectives

Introduction

5.1 Simulating Randomness

5.2 Random-number Generators

5.3 Generation of Random Numbers

5.4 Summary

5.5 Keywords

5.6 Self Assessment

5.7 Review Questions

5.8 Further Readings

Objectives

After studying this unit, you will be able to:

- Describe Simulating Randomness
- Explain Generation of Random Numbers

Introduction

To generate truly random numbers requires precise, accurate, and repeatable system measurements of absolutely non-deterministic processes. In comparison with PRNGs, TRNGs extract randomness from physical phenomena and introduce it into a computer. You can imagine this as a die connected to a computer, but typically people use a physical phenomenon that is easier to connect to a computer than a die.

5.1 Simulating Randomness

1. ***Stochastic Systems*:** Systems with intrinsic randomness or changeability in their behaviour.
   (a) These systems with chance can be normal or man-made.
   (b) Instances of systems where randomness is simulated are:
      (i) Inventory System
      (ii) Water Reservoir
      (iii) Advent of customers in a store
      (iv) Demand for telephone line at exchange
      (v) Births and deaths in a population
      (vi) Particle collision in a reactor
      (vii) Appearance of an elevator on a given floor

Notes

2. ***Discrete dynamic systems*** could be categorized as:

   (a) *Deterministic*: For specified input, output is set. So, its just a conversion and that too computationally not as much of demanding

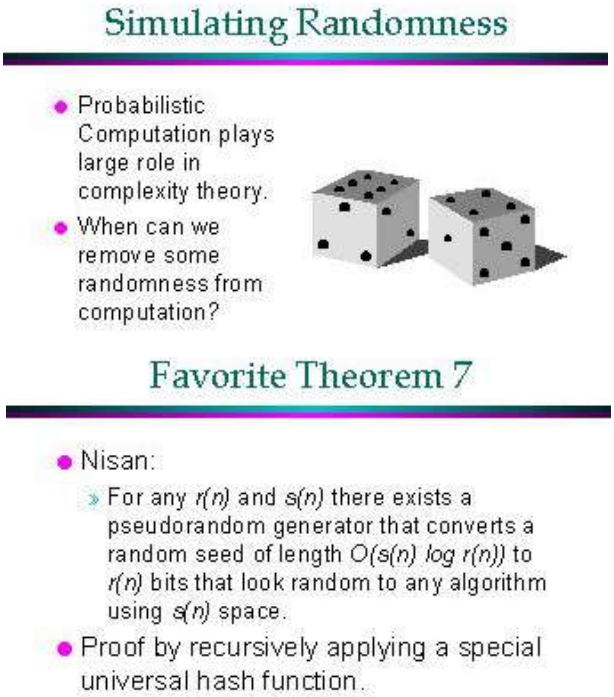
   (b) *Stochastic*: For specified input, there can be more than one output. It is called a system in which at least one of the variables is specified by a probability function.

Deterministic Randomness

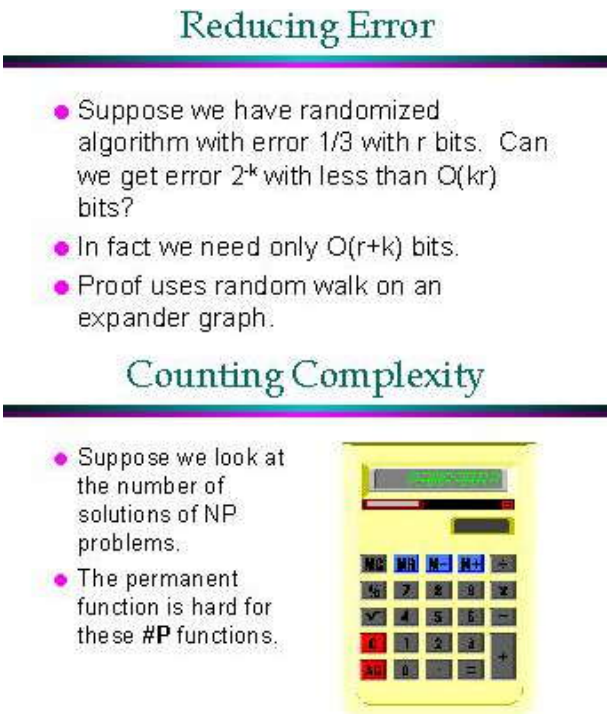
1. Computers are deterministic; no chance involved
2. Always same output for same input; unless error
3. Generate pseudo-random numbers
4. Monte Carlo calculations: simulate random events

Examples

1. Thermal motion
2. Games of chance (meaning?)
3. Radioactive decay
4. Solve equations statistically
5. Solve intractable problems



Notes



Notes Complex discrete, dynamic, stochastic systems frequently disobey an analytic solution and are as a result studied in the course of simulation.

5.2 Random-number Generators

The simulation needs to generate random variables of various kinds, depending on the system model. This is accomplished by one or more pseudorandom number generators. The use of pseudorandom numbers as opposed to true random numbers is a benefit should a simulation need a rerun with exactly the same behaviour.

One of the problems with the random number distributions used in discrete-event simulation is that the steady-state distributions of event times may not be known in advance. As a result, the initial set of events placed into the pending event set will not have arrival times representative of the steady-state distribution. This problem is typically solved by bootstrapping the simulation model. Only a limited effort is made to assign realistic times to the initial set of pending events. These events, however, schedule additional events, and with time, the distribution of event times approaches its steady state. This is called bootstrapping the simulation model. In gathering statistics from the running model, it is important to either disregard events that occur before the steady state is reached or to run the simulation for long enough that the bootstrapping behavior is overwhelmed by steady-state behavior. (This use of the term bootstrapping can be contrasted with its use in both statistics and computing.)

Tasks Analyze the problems that take place in random number distributions.

Notes

Statistics

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

Ending Condition

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

Simulation Engine Logic

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

Start

1. Initialize Ending Condition to FALSE.
2. Initialize system state variables.
3. Initialize Clock (usually starts at simulation time zero).
4. Schedule an initial event (i.e., put some initial event into the Events List).

"Do loop" or "While loop"

While (Ending Condition is FALSE) then do the following:

1. Set clock to next event time.
2. Do next event and remove from the Events List.
3. Update statistics.

End

Generate statistical report.

5.3 Generation of Random Numbers

A sequence of pseudo random numbers can be generated by a computer algorithm, such as the Linear Congruential Method. Such sequences are not random, since each number is completely determined from a set of numerical seeds. The sequences "appear" to be random, however, and are useful in calculations and simulations. The following Java applet demonstrates random number generation with the Linear Congruential Method. You can change the multiplier (a), modulus (m), and the initial seed (I). The algorithm produces random numbers between 0 and 1. You can see the distribution of 5000 random numbers develop as they are produced, by pressing the Run button.

You may also choose from three ways to show the random number distributions:

1. ***1D:*** shows the frequency distribution of all random numbers as a histogram divided into 20 bins from 0 to 1.

Notes

2. ***2D:*** pairs of random numbers are used to define the coordinates of points in a plane.
3. ***3D:*** groups of three random numbers are used to specify the coordinates of points in 3 dimensional space. By dragging the mouse on the picture you can rotate your viewpoint. Dragging the mouse while holding down the shift key allows you to zoom in and out.

The initial choices of multiplier and modulus are that of the RANDU generator, distributed by IBM in the 1960's. Only after it was widely distributed was it noticed to have a serious problem when generating 3 dimensional points. See if you can see the Marsaglia effect in the 3D view, by rotating your viewpoint. Better behaviour results with a different multiplier, such as 69069.

A random number generator (often abbreviated as RNG) is a computational or physical device designed to generate a sequence of numbers or symbols that lack any pattern, i.e. appear random. Hardware-based systems for random number generation are widely used, but often fall short of this goal, though they may meet some of the statistical tests for randomness intended to ensure that they do not have any easily discernible patterns. Methods for generating random results have existed since ancient times, including dice, coin flipping, the shuffling of playing cards, the use of yarrow stalks (by divination) in the I Ching, and many other techniques.

The many applications of randomness have led to many different methods for generating random data. These methods may vary as to how unpredictable or statistically random they are, and how quickly they can generate random numbers.

Before the advent of computational random number generators, generating large amounts of sufficiently random numbers (important in statistics) required a lot of work. Results would sometimes be collected and distributed as random number tables.

A growing number of government-run lotteries, and lottery games, are using RNGs instead of more traditional drawing methods, such as using ping-pong or rubber balls.

Caution Linear Congruential Method is used only for uniform random numbers

Physical Methods

The earliest methods for generating random numbers — dice, coin flipping, roulette wheels — are still used today, mainly in games and gambling as they tend to be too slow for applications in statistics and cryptography.

Some physical phenomena, such as thermal noise in Zener diodes appear to be truly random and can be used as the basis for hardware random number generators. However, many mechanical phenomena feature asymmetries and systematic biases that make their outcomes not truly random. The many successful attempts to exploit such phenomena by gamblers, especially in roulette and blackjack are testimony to these effects.

There are several imaginative sources of random numbers online. A common technique is to run a hash function against a frame of a video stream from an unpredictable source. This technique was used by Lava rand which used images of a number of lava lamps. Lithium Technologies uses a camera pointed at the sky on a windy and cloudy day. Random.org uses variations in the amplitude of atmospheric noise. Details about how they turn their input into random numbers can be found on their respective sites.

Completely randomized design falls within the category of true random number generation. The generation of true random numbers outside the computer environment is based on the theory of entropy. Sources of entropy include nuclear decay and atmospheric conditions. HotBits uses radioactive decay, while Random.org uses radio noise to generate randomness.

Notes

Another common entropy source is the behavior of human users of the system, if such users exist. While humans are not considered good randomness generators upon request, they generate random behavior quite well in the context of playing mixed strategy games. The utilization of human game play entropy for randomness generation was studied by Ran Halprin and Moni Naor, see "Games for Extracting Randomness".

Computational Methods

Pseudo-random Number Generators (PRNGs) are algorithms that can automatically create long runs (for example, millions of numbers long) with good random properties but eventually the sequence repeats (or the memory usage grows without bound). One of the most common PRNG is the linear congruential generator, which uses the recurrence $X_{n+1} = (aX_n + b) \bmod m$ to generate numbers. The maximum number of numbers the formula can produce is the modulus, m. To avoid certain non-random properties of a single linear congruential generator, several such random number generators with slightly different values of the multiplier confident and are typically used in parallel, with a "master" random number generator that selects from among the several different generators.

Most computer programming languages include functions or library routines that purport to be random number generators. They are often designed to provide a random byte or word, or a floating point number uniformly distributed between 0 and 1.

Such library functions often have poor statistical properties and some will repeat patterns after only tens of thousands of trials. They are often initialized using a computer's real time clock as the seed. These functions may provide enough randomness for certain tasks (for example video games) but are unsuitable where high-quality randomness is required, such as in cryptographic applications, statistics or numerical analysis. Much higher quality random number sources are available on most operating systems; for example /dev/random on various BSD flavors, Linux, Mac OS X, IRIX, and Solaris, or CryptGenRandom for Microsoft Windows.

Notes A simple pen-and-paper method for generating random numbers is the so-called middle square method suggested by John Von Neumann. While simple to implement, its output is of poor quality.

Practical Applications and Uses of Random Numbers

Random number generators have applications in gambling, statistical sampling, computer simulation, cryptography, and other areas where a random number is useful in producing an unpredictable result.

Note that, in general, where unpredictability is paramount–such as in security applications–hardware generators are generally preferred, where feasible, over pseudo-random algorithms.

Random number generators are very useful in developing Monte Carlo method simulations as debugging is facilitated by the ability to run the same sequence of random numbers again by starting from the same random seed. They are also used in cryptography so long as the seed is secret. Sender and receiver can generate the same set of numbers automatically to use as keys.

The generation of pseudo-random numbers is an important and common task in computer programming. While cryptography and certain numerical algorithms require a very high degree of apparent randomness, many other operations only need a modest amount of unpredictability. Some simple examples might be presenting a user with a "Random Quote of the Day", or determining which way a computer-controlled adversary might move in a computer game. Weaker forms of randomness are also closely associated with hash algorithms and in creating amortized searching and sorting algorithms.

Notes

Some applications which appear at first sight to be suitable for randomization are in fact not quite so simple. For instance, a system that 'randomly' selects music tracks for a background music system must only appear to be random; a true random system would have no restriction on the same item appearing two or three times in succession.

Did u know? **The Meaning of Cryptography**

The word is obtained from the Greek *kryptos*, meaning hidden. Cryptography comprises techniques like microdots, merging words with images, and other methods to hide information in storage or transfer.

Activities and Demonstrations

The SOCR resource pages contain a number of hands-on interactive activities and demonstrations of random number generation using Java applets.

"True" Random Numbers vs. Pseudorandom Numbers

There are two principal methods used to generate random numbers. One measures some physical phenomenon that is expected to be random and then compensates for possible biases in the measurement process. The other uses computational algorithms that produce long sequences of apparently random results, which are in fact completely determined by a shorter initial value, known as a seed or key. The latter types are often called pseudorandom number generators.

A "random number generator" based solely on deterministic computation cannot be regarded as a "true" random number generator, since its output is inherently predictable. John von Neumann famously said "Anyone who uses arithmetic methods to produce random numbers is in a state of sin." How to distinguish a "true" random number from the output of a pseudo-random number generator is a very difficult problem. However, carefully chosen pseudo-random number generators can be used instead of true random numbers in many applications. Rigorous statistical analysis of the output is often needed to have confidence in the algorithm.

Generating Random Numbers from Physical Processes

There is general agreement that, if there are such things as "true" random numbers, they are most likely to be found by looking at physical processes which are, as far as is known, unpredictable.

A physical random number generator can be based on an essentially random atomic or subatomic physical phenomenon whose unpredictability can be traced to the laws of quantum mechanics.

Example of this is the Atari 8-bit computers, which used electronic noise from an analog circuit to generate true random numbers. Other examples include radioactive decay, thermal noise, shot noise and clock drift. Even lava lamps have been used by the Lava rand generator.

To provide a degree of randomness intermediate between specialized hardware on the one hand and algorithmic generation on the other, some security related computer software requires the user to input a lengthy string of mouse movements, or keyboard input.

Post-processing and Statistical Checks

Even given a source of plausible random numbers (perhaps from a quantum mechanically based hardware generator), obtaining numbers which are completely unbiased takes care. In

Notes

addition, behavior of these generators often changes with temperature, power supply voltage, the age of the device, or other outside interference. And a software bug in a pseudo-random number routine, or a hardware bug in the hardware it runs on, may be similarly difficult to detect.

Generated random numbers are sometimes subjected to statistical tests before use to ensure that the underlying source is still working, and then post-processed to improve their statistical properties.

5.4 Summary

- The simulation needs to generate random variables of various kinds, depending on the system model. This is accomplished by one or more pseudorandom number generators.
- A sequence of pseudo random numbers can be generated by a computer algorithm, such as the Linear Congruential Method.

5.5 Keywords

Monte Carlo Simulation: References to Monte Carlo simulation are often encountered in the and simulation literature.

Next Event: Advances the model to the next event to be executed, regardless of the time interval.

Next-event Simulation Model : In next-event simulation model the computer advances time to the incidence of the subsequent event.

Object-oriented Simulation: Object Oriented techniques have been developed since the early 1960's as a result of simulation development (SIMULA).

Random Number Generator: Helps to simulate different data coming into the simulation model.

5.6 Self Assessment

Fill in the blanks:

1. An event handler is code, typically a function or routine written in a ..... language that receives control when the corresponding event occurs.
2. ..... are the systems with intrinsic randomness or changeability in their behaviour.
3. A "random number generator" is based solely on ..... computation.
4. ..... methods are a class of computational algorithms that rely on repeated random sampling to compute their results.
5. The simulation needs to generate random variables of various kinds, depending on the ……...
6. The initial set of events placed into the pending event set will not have arrival times representative of the ……. distribution.
7. The simulation typically keeps track of the system's …….
8. The sequences "appear" to be random, however, and are useful in …… and simulations.
9. The initial choices of multiplier and modulus are that of the …… generator.

Notes

10. A growing number of government-run lotteries, and lottery games, are using …… instead of more traditional drawing methods.
11. A common technique is to run a …….. against a frame of a video stream from an unpredictable source.
12. …… Technologies uses a camera pointed at the sky on a windy and cloudy day.
13. One of the most common …….. is the linear congruential generator.
14. Most computer programming languages include functions or ……. routines that purport to be random number generators.
15. A physical random number generator can be based on an essentially random atomic or subatomic physical phenomenon whose unpredictability can be traced to the laws of……...

5.7 Review Questions

1. Do you think the use of pseudorandom numbers as opposed to true random numbers is a benefit?
2. Explain the main loop of a discrete-event simulation.
3. "A sequence of pseudo random numbers can be generated by a computer algorithm, such as the Linear Congruential Method." Comment
4. "A random number generator (often abbreviated as RNG) is a computational or physical device." Discuss.
5. What are the earliest methods for generating random numbers?
6. Explain why the Thermal noise in Zener diodes appear to be truly random?
7. Discus imaginative sources of random numbers online.
8. Analyze completely randomized design falls within the category of true random number generation. If yes then why?
9. "Pseudo-random Number Generators (PRNGs) are algorithms that can automatically create long runs." Elaborate
10. Give the practical applications and uses of random numbers.

Answers: Self Assessment

1. scripting
2. Stochastic Systems
3. Deterministic
4. Monte Carlo
5. system model
6. steady-state
7. statistics
8. calculations
9. RANDU
10. RNGs
11. hash function
12. Lithium
13. PRNG
14. library
15. quantum mechanics

5.8 Further Readings

Notes

Books

Balci, O.,(1994), Validation, verification, and testing techniques throughout the life cycle of a simulation study, Annals of Operations Research, 53: 121–173.

Balci, O., (2001), A methodology for certification of and simulation applications, ACM Transactions on and Computer Simulation, 11: 352–377.

Birta, L.G. and Ozmizrak, N.F., (1996), A knowledge-based approach for the validation of simulation models: The foundation, ACM Transactions on and Computer Simulation, 6: 67–98.

Boehm, B.W., (1979), Software engineering: R&D trends and defense needs, in: P. Wegner (Ed.), Research Directions in Software Technology, MIT Press, and Cambridge, MA.

Cellier, F.E., (1986), Combined discrete/continuous system simulation – application, techniques and tools, in Proceedings of the 1986 Winter Simulation Conference.

Department of Defense (DoD) Recommended Practices Guide (RPG) for and Simulation VV&A, Millennium Edition (available at http://vva.dmso.mil).

General Accounting Office, (1976), Report to the Congress: Ways to improve management of federally funded computerized models, report LCD-75-111, U.S. General Accounting Office, Washington, DC.

General Accounting Office, (1979), Guidelines for model evaluation, report PAD-79-17, U.S. General Accounting Office, Washington, DC.

General Accounting Office, (1987), DOD simulations: Improved assessment procedures would increase the credibility of results, report GAO/PEMD-88-3, U.S. General Accounting Office, and Washington, DC.

Ören, T.I., (1981), Concepts and criteria to access acceptability of simulation studies, Communications of the ACM, 24: 180–189.

Online links

http://oxford-mphil.com/images/f/fc/Simulation-econometrics.pdf

http://www.experiencefestival.com/applications_of_randomness_-_simulation

http://stackoverflow.com/questions/712785/simulating-randomness

Notes

Unit 6: Discrete System Simulation (III)

CONTENTS

Objectives

Introduction

6.1 Generation of Non-uniformly Distributed Random Numbers

6.2 Monte Carlo Computation vs Stochastic Simulation

6.3 Summary

6.4 Keywords

6.5 Self Assessment

6.6 Review Questions

6.7 Further Readings

Objectives

After studying this unit, you will be able to:

- Understand generation of non-uniformly distributed random number
- Describe Monte Carlo Computation vs stochastic Simulation

Introduction

Even though the RAND function can be useful for generating Uniform random numbers, most of the time you will need to model various non-uniform distributions, such as the Normal, Lognormal, Exponential, Gamma, and others. Monte Carlo simulation is a computerized mathematical technique that allows people to account for risk in quantitative analysis and decision making.

6.1 Generation of Non-uniformly Distributed Random Numbers

Principles

The uniform distribution is hardly ever used straightforwardly in any econometric model. Other distributions, especially the normal or Gaussian distribution, are much more established. Various methods survive to convert a uniform RNG into a non-uniform counterpart.

When it is practicable, the most suitable method to derive random number generators for a non-uniform distribution is the inversion principle. This develops from the property:

$$\Pr(U_i \le u) = P(u) = u \quad \text{for} \quad U_i \sim U(0,1) \text{ and } u \in [0,1].$$

As a result, P(u) and u can be interchanged in any source. Let $F(\cdot)$ indicate the cumulative $F^{-1}(u_i)$ implies that $u_i = F(e_i)$ and therefore:

$$\Pr(e_i \le u) = \Pr(F(e_i) \le F(u)) = \Pr(U_i \le F(u)) = P(F(u)) = F(u). \text{ as necessary.}$$

Notes

Using the exponential distribution as an instance: $X \sim \exp(\lambda)$, with CDF $F(x) = 1 - \exp(-\lambda x)$ where $\lambda > 0$, the draws can be achieved from uniform random numbers u_i as:

$$x_i = -\lambda^{-1} \log (1 - u_i) .$$

Apart from zero and one from the uniform random numbers make sure that –1and zero are averted for the exponential distribution.

If a cumulative distribution function and density are obtainable, but no analytical formula for the inverse, it is easy to execute the quantile function (i.e., the inverse function) using the Newton-Raphson technique. This could then offer a generator for random numbers. Though, while this can be a expedient approach, it is typically very much slower than direct methods.

The rejection method, which can be traced back to Von Neumann, often provides an efficient alternative when inversion is slow. Suppose we wish to sample from a density $f(\cdot)$, and that f is bounded by another density g: $f(\cdot) \le cg(\cdot)$. Sampling from $f(\cdot)$ can then be employed as:

repeat

generate a random examination vi from $g(\cdot)$,

until $u_i cg(v_i) \le f(v_i)$, (1)

where u_i is independently U(0, 1) distributed. The potential profit comes when it is hard to sample from f, but simple for g. The closer cg is to f, the better the bound, and the more competent the rejection method. An instance for the normal distribution is specified below.

The most significant non-uniform distribution is the standard normal, which, unluckily, does not have a logical inverse. Precise, non-iterative, rough calculations to the normal quantiles exist, but would lead to a comparatively slow process.

A coarse method for generating $e_i \sim IN[0, 1]$ depends on the estimated central limit result:

$$\left(\sum_{j=1}^{12} u_j - 6\right) = e_i \qquad (2)$$

Although this is simple, it is twelve times slower than the uniform RNG, and of restricted correctness.

The method of Box and Muller (1958) converts two uniform random numbers into two independent standard normals:

$$(e_i, e_{i+1}) = h_i (\cos 2\pi u_{i+1}, \sin 2\pi u_{i+1}) \text{ where } h_i = (-2 \log u_i)^{\frac{1}{2}} \qquad (3)$$

A well-tested and empirically acceptable generator must be used for input to the Box–Muller method when (u_i, u_{i+1}) are consecutively generated. Particularly, an LCG with poor lattice structure could be difficult. The most well-liked method is the polar–Marsaglia method (Marsaglia and Bray, 1964), which depends on Box–Muller as below:

repeat

$v_1 = 2u_1 - 1$, $v_2 = 2u_2 - 1$

$d = v_1^2 + v_2^2$

until $d < 1$

$e_1 = [“2 \log(d)/d]^{1/2} v_1$, $e_2 = [-2 \log(d)/d]^{1/2} v_2$ (4)

This needs, on average, 1.27 as many uniforms as (3) but averts the trigonometric function.

Notes

As an instance of the rejection method, consider the logistic distribution as a hurdle for the standard normal. From discrete choice methods, it is identified that binary probit and binary logit are very comparable , so the logistic may be a fine candidate that can be reversed logically. The logistic distribution with scale zero and variance $\beta^2\pi^2/3$ has cdf:

$$G(x) = [1 + \exp(-x/\beta)]^{-1} .$$

The matching thickness is $g(x) = \beta^{-1}G(x)[1 - G(x)]$ and the quantiles are $G^{-1}(u) = -\beta \log[(1 - u)/u]$. Execution needs a pick for the scaling parameter c in (1) and the discrepancy of the logistic distribution during β, such that the rejection probability is (approximately) diminished. Devroye demonstrates the principle of how these parameters can be calculated. Following this, we maximize $f(x)/g_\beta(x)$, or more expediently, $d(x) = \log f(x) - \log g_\beta(x)$ to locate one solution at zero for $\beta \geq 0.71$. For lesser β's, zero is a local minimum. Solving numerically, it is found that $\min_\beta \max_x d(x)$ is at $x_m = 0.98226$ when $\beta_m = 0.65$. The value of c is then $f(x_m)/g_{\beta m}(x_m) = 1.081$. This shows a rejection algorithm:

repeat

$w_1 = \log(u_1)$, $w_2 = \log(1 - u_1)$

$v = 0.65 _ (w_1 - w_2)$;

until $\log(u_2) + w_1 + w_2 + \log(1.081/0.65) < -v^2/2$ " $\log(2\pi)/2$ return v. (5)

It will be slower than (4), as it needs at least two uniforms and three logarithms for each standard normal.

There is a large journalism on the generation of non-uniform random numbers.

Efficient Methods for the Standard Normal Distribution

The abovementioned polar technique is simple to execute, but not chiefly fast. Due to the significance of the normal distribution, many more well-organized methods have been anticipated. The method of Wallace (1996) is attractive, as it generates normal random numbers directly, in addition to being very well-organized. Though, due to this it is also rather harder to understand the properties of this technique.

Perhaps the fastest obtainable rejection method at the moment is the so called 'ziggurat method,' commenced by Marsaglia and Tsang (1984) and consequently refined (Marsaglia and Tsang, 2000). This technique has some scarcities, which were corrected by Doornik (2005*b*) at the cost of some of its speed.

The ziggurat partitions the standard normal thickness into horizontal blocks of equal area. The standardization can be mislaid, using $f(x) = \exp(-x^2/2)$ as an alternative. All blocks are rectangular boxes, apart from the bottom one, which includes a box joined with the remainder of the density.

This is demonstrated in Figure 6.1, using four boxes, labeled from bottom to top as B_0, B_1, B_2, B_3.

Equal areas of size V entails:

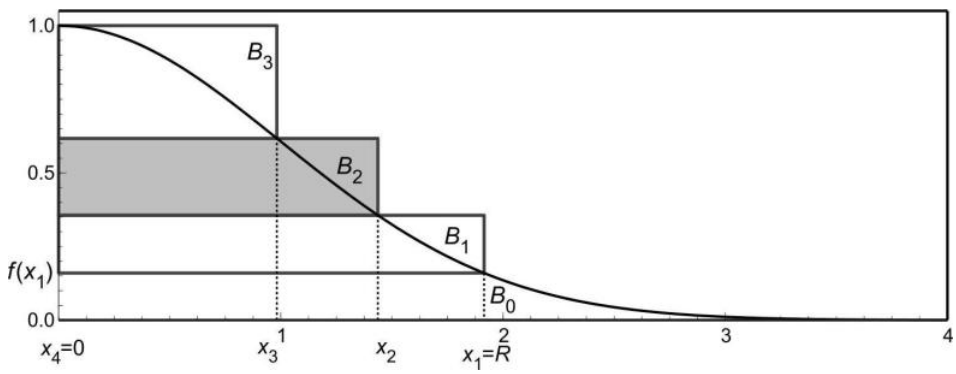
$$x_3 [f(x_4) - f(x_3)] = x_2 [f(x_3) - f(x_2)] = x_1 [f(x_2) - f(x_1)] = x_1 f(x_1) + \int_{x_1}^{\infty} f(x)dx = V .$$

Note that, working backward along the X-axis, earlier values of x can be easily obtained. For Instance, if x_1 is known:

$$x_2 = f^{-1} (f(x_1) + V/x_1) .$$

Notes

Figure 6.1: Example of a Four-way Ziggurat Partitioning of the Standard Normal Density



Tasks Analyze the methods used for non-uniform distributed random numbers.

Testing Non-uniform Random Numbers

Testing the normal RNG is possibly more significant than the fundamental uniform, as it offers the basis leading econometric simulation experiments. A simple technique is to convert the normal random numbers e_i back to consistency using the normal cdf Φ: $\tilde{u}_i = \Phi(e_i)$. The $\tilde{u}_i$ can then be used in the obtainable testing software, in our case the crush test suite of L'Ecuyer and Simard (2005). When inversion is used, testing is unnecessary: $\tilde{u}_i = \Phi(\Phi^{-1}(u_i)) = u_i$. Though, for the other methods: $\tilde{u}_i \neq u_i$.

Table 6.1 offers the failure count on the crush test for two techniques: the sum of 12 uniforms (2), the polar method (4), in addition to the ziggurat method (but only for MWC8222). The first is shown to be problematic for all uniform RNGs. The consequences for the polar method are interesting, as it shows that the new uniform random numbers $\tilde{u}_i$ are significantly more uniform than the underlying u_i for LCG31 and MWC60, but also to some extent for LFSR113 and WELL1024.

This is mainly the outcome of using the rejection method, whereas there is no added problem from using consecutive numbers. So, though LCG31 and MWC60 were rejected as uniform RNGs, they are more adequate for standard normal RNGs based on the polar method.

Lastly, consider the Student-t(n) distribution. This can be produced as

$$\phi\left(\frac{n}{x}\right)^{1/2}, \phi \sim N[0,1], x \sim x^2(n)$$

The χ^2 distribution in turn can be obtained from a gamma distribution, for which algorithms 3.19 and 3.20 from Ripley (1987) can be used. Table 6.2 offers the failure count on the crush test for n = 6 and n = 12 after mapping the Student-t numbers back to uniformity. Yet again, LCG31 and MWC60 do not act as well as the others, but not devastatingly so.

Notes

Did u know? The full form of RNG

RNG: Random Number Generator

Table 6.1: Number of Failures in the Crush Battery of Tests (94 tests in total) for some Standard Normal Random Number Generators

	Sum of 12 uniforms			polar method			ziggurat	
	10^{200}	10^{15}	10^{5}	10^{200}	10^{15}	10^{5}	10^{200}	10
LCG31	13	7	4	1	5	0		
MWC60	10	3	2	3	3	0		
LFSR113	10	3	2	0	0	0		
WELL1024	10	4	0	0	0	0		
MWC8222	10	4	1	0	0	0	0	
MT19937	10	4	0	0	0	0		

Table 6.2: Number of Failures in the Crush Battery of Tests (94 tests in total) for Student-t Random Number Generators

	Student-$\ell(6)$			Student-$\ell(12)$		
	10^{-300}	10^{-15}	10^{-5}	10^{-300}	10^{-15}	10^{-5}
LCG31	4	0	1	3	0	1
MWC60	1	0	1	1	0	1
LFSR113	0	0	0	0	0	0
WELL1024	0	0	0	0	0	0
MWC8222	0	0	0	0	0	0
MT19937	0	0	0	0	0	0

6.2 Monte Carlo Computation vs Stochastic Simulation

References to Monte Carlo simulation are often encountered in the and simulation literature. This somewhat fanciful label refers to a problem-solving methodology that is loosely related to, but is very different from, the topic that we explore in this textbook. The term refers to a family of techniques that are used to find solutions to numerical problems. The distinctive feature of these techniques is that they proceed by constructing an artificial stochastic (probabilistic) system whose properties contain the solution of the underlying problem. The origins of the approach can be traced back to Lord Raleigh who used it to develop approximate solutions to simple partial differential equations. The power of the methodology was exploited by von Neumann and colleagues in solving complex problems relating to their work in developing a nuclear arsenal in the latter years of the Second World War. The Monte Carlo label for the methodology is, in fact, attributed to this group.

Notes

Example: Perhaps the simplest example of the method is its application to the evaluation of the definite integral:

$$I = \int_a^b f(x)dx \tag{1}$$

for the special case where f(x) >= 0. The value of I is the area under f(x) between x = a and x = b. Consider now a horizontal line drawn at y = K such that f(x) <= K for a <= x<= b .The rectangle R, enclosed by x = a, x = b, y = 0, and y = K has the area K(b– a) and furthermore I <= K(b – a). Suppose a sequence of points (x_i, y_i) is chosen at random within the rectangle R such that all points within R are equally likely to be chosen (e.g., by choosing from two uniform distributions oriented along the length and width of R). It can then be easily appreciated that the ratio of the number of points that fall either on the curve or under it (say, n) to the total number of points chosen (say, N) is an approximation of the ratio of I to the area of the rectangle R. In other words,

$$n/N \approx I/[K(b - a)] \tag{2}$$

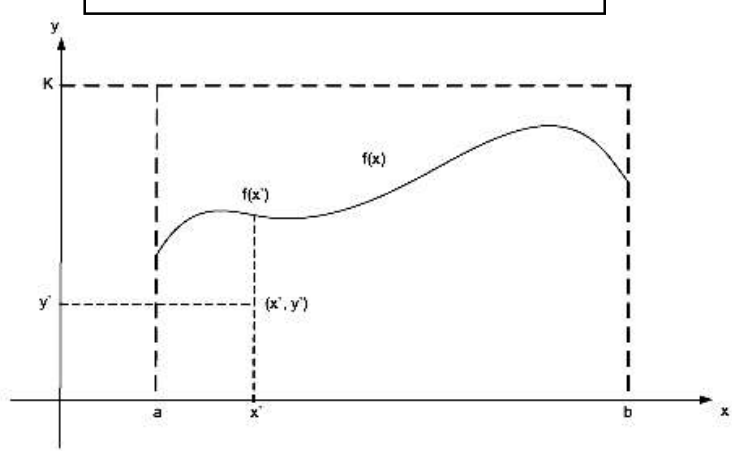
Or

$$I \approx nK(b - a)/N \tag{3}$$

In the procedure, a point (x_i,y_i) is included in the count n, if y_i <= f(x_i). The accuracy of the approximation improves as N increases.

The interesting feature in this example is that the original problem is entirely deterministic and yet the introduction of probabilistic notions can yield an approximation to its solution.

Figure 6.2: Monte Carlo Simulation



The class of problems that can be effectively investigated by Monte Carlo simulation generally falls within the domain of numerical analysis. The approach provides an alternate, and often very effective, solution option for these problems. However, these problems do not fall within the scope of the and simulation methodology because they lack the prerequisite of 'behaviour,' that is, an evolution over time. The reference to 'simulation' in the label for the approach could be regarded as a reflection of the dissimilarity between the solution mechanism and the inherent nature of the problem.

Monte Carlo methods are a class of computational algorithms that rely on repeated random sampling to compute their results. Monte Carlo methods are often used when simulating physical and mathematical systems. Because of their reliance on repeated computation and random or

Notes

pseudo-random numbers, Monte Carlo methods are most suited to calculation by a computer. Monte Carlo methods tend to be used when it is unfeasible or impossible to compute an exact result with a deterministic algorithm.

Monte Carlo simulation methods are especially useful in studying systems with a large number of coupled degrees of freedom, such as fluids, disordered materials, strongly coupled solids, and cellular structures. More broadly, Monte Carlo methods are useful for phenomena with significant uncertainty in inputs, such as the calculation of risk in business. These methods are also widely used in mathematics: a classic use is for the evaluation of definite integrals, particularly multidimensional integrals with complicated boundary conditions. It is a widely successful method in risk analysis when compared to alternative methods or human intuition. When Monte Carlo simulations have been applied in space exploration and oil exploration, actual observations of failures, cost overruns and schedule overruns are routinely better predicted by the simulations than by human intuition or alternative "soft" methods.

The term Monte Carlo method was coined in the 1940s by physicists working on nuclear weapon projects in the Los Alamos National.

Overview

The Monte Carlo method can be illustrated as a game of battleship. First a player makes some random shots. Next the player applies algorithms (i.e. a battleship is four dots in the vertical or horizontal direction). Finally based on the outcome of the random sampling and the algorithm the player can determine the likely locations of the other player's ships.

There is no single Monte Carlo method; instead, the term describes a large and widely-used class of approaches. However, these approaches tend to follow a particular pattern:

1. Define a domain of possible inputs.
2. Generate inputs randomly from the domain.
3. Perform a deterministic computation using the inputs.
4. Aggregate the results of the individual computations into the final result.

Example: The value of π can be approximated using a Monte Carlo method:

1. Draw a square on the ground, then inscribe a circle within it. From plane geometry, the ratio of the area of an inscribed circle to that of the surrounding square is $\pi/4$.
2. Uniformly scatter some objects of uniform size throughout the square. For example, grains of rice or sand.
3. Since the two areas are in the ratio $\pi/4$, the objects should fall in the areas in approximately the same ratio. Thus, counting the number of objects in the circle and dividing by the total number of objects in the square will yield an approximation for $\pi/4$. Multiplying the result by 4 will then yield an approximation for π itself.

Notice how the approximation follows the general pattern of Monte Carlo algorithms. First, we define a domain of inputs: in this case, it's the square which circumscribes our circle. Next, we generate inputs randomly (scatter individual grains within the square), then perform a computation on each input (test whether it falls within the circle). At the end, we aggregate the results into our final result, the approximation of π. Note, also, two other common properties of Monte Carlo methods: the computation's reliance on good random numbers, and its slow convergence to a better approximation as more data points are sampled. If grains are purposefully dropped into only, for example, the center of the circle, they will not be uniformly distributed,

Notes

and so our approximation will be poor. An approximation will also be poor if only a few grains are randomly dropped into the whole square. Thus, the approximation of ð will become more accurate both as the grains are dropped more uniformly and as more are dropped.

History

The name "Monte Carlo" was popularized by physics researchers Stanislaw Ulam, Enrich Fermi, John von Neumann, and Nicholas Metropolis, among others; the name is a reference to the Monte Carlo Casino in Monaco where Ulam's uncle would borrow money to gamble. The use of randomness and the repetitive nature of the process are analogous to the activities conducted at a casino.

Random methods of computation and experimentation (generally considered forms of stochastic simulation) can be arguably traced back to the earliest pioneers of probability theory (see, e.g., Buffon's needle, and the work on small samples by William Gosset), but are more specifically traced to the pre-electronic computing era. The general difference usually described about a Monte Carlo form of simulation is that it systematically "inverts" the typical mode of simulation, treating deterministic problems by first finding a probabilistic analog. Previous methods of simulation and statistical sampling generally did the opposite: using simulation to test a previously understood deterministic problem. Though examples of an "inverted" approach do exist historically, they were not considered a general method until the popularity of the Monte Carlo method spread.

Perhaps the most famous early use was by Enrico Fermi in 1930, when he used a random method to calculate the properties of the newly-discovered neutron. In the 1950s they were used at Los Alamos for early work relating to the development of the hydrogen bomb, and became popularized in the fields of physics, physical chemistry, and operations research. The Rand Corporation and the U.S. Air Force were two of the major organizations responsible for funding and disseminating information on Monte Carlo methods during this time, and they began to find a wide application in many different fields.

Uses of Monte Carlo methods require large amounts of random numbers, and it was their use that spurred the development of pseudorandom number generators, which were far quicker to use than the tables of random numbers which had been previously used for statistical sampling.

Notes Monte Carlo methods were central to the simulations required for the Manhattan Project, though were severely limited by the computational tools at the time. Therefore, it was only after electronic computers were first built (from 1945 on) that Monte Carlo methods began to be studied in depth.

Applications

As mentioned, Monte Carlo simulation methods are especially useful for phenomena with significant uncertainty in inputs and in studying systems with a large number of coupled degrees of freedom. Specific areas of application include:

Physical Sciences

Monte Carlo methods are very important in computational physics, physical chemistry, and related applied fields, and have diverse applications from complicated quantum chromo dynamics calculations to designing heat shields and aerodynamic forms. The Monte Carlo method is widely used in statistical physics, in particular, Monte Carlo molecular as an alternative for

Notes

computational molecular dynamics; see Monte Carlo method in statistical physics. In experimental particle physics, these methods are used for designing detectors, understanding their behavior and comparing experimental data to theory.

Design and Visuals

Monte Carlo methods have also proven efficient in solving coupled integral differential equations of radiation fields and energy transport, and thus these methods have been used in global illumination computations which produce photorealistic images of virtual 3D models, with applications in video games, architecture, design, computer generated films, special effects in cinema.

Finance and Business

Monte Carlo methods in finance are often used to calculate the value of companies, to evaluate investments in projects at corporate level or to evaluate financial derivatives. The Monte Carlo method is intended for financial analysts who want to construct stochastic or probabilistic financial models as opposed to the traditional static and deterministic models for its use in the insurance industry.

Telecommunications

When planning a wireless network, design must be proved to work for a wide variety of scenarios that depend mainly on the number of users, their locations and the services they want to use. Monte Carlo methods are typically used to generate these users and their states. The network performance is then evaluated and, if results are not satisfactory, the network design goes through an optimization process.

Games

Monte Carlo methods have recently been applied in game playing related artificial intelligence theory. Most notably the game of Go has seen remarkably successful Monte Carlo algorithm based computer players. One of the main problems that this approach has in game playing is that it sometimes misses an isolated, very good move. These approaches are often strong strategically but weak tactically, as tactical decisions tend to rely on a small number of crucial moves which are easily missed by the randomly searching Monte Carlo algorithm.

Monte Carlo Simulation versus "What If" Scenarios

The opposite of Monte Carlo simulation might be considered deterministic using single-point estimates. Each uncertain variable within a model is assigned a "best guess" estimate. Various combinations of each input variable are manually chosen (such as best case, worst case, and most likely case), and the results recorded for each so-called "what if" scenario.

By contrast, Monte Carlo simulation considers random sampling of probability distribution functions as model inputs to produce hundreds or thousands of possible outcomes instead of a few discrete scenarios. The results provide probabilities of different outcomes occurring.

Example: A comparison of a spreadsheet cost construction model run using traditional "what if" scenarios, and then run again with Monte Carlo simulation and Triangular probability distributions shows that the Monte Carlo analysis has a narrower range than the "what if" analysis. This is because the "what if" analysis gives equal weight to all scenarios.

Notes

Use in Mathematics

In general, Monte Carlo methods are used in mathematics to solve various problems by generating suitable random numbers and observing that fraction of the numbers obeying some property or properties. The method is useful for obtaining numerical solutions to problems which are too complicated to solve analytically. The most common application of the Monte Carlo method is Monte Carlo integration.

Task Analyze the different areas where you can use Monte Carlo methods.

Integration

Deterministic methods of numerical integration operate by taking a number of evenly spaced samples from a function. In general, this works very well for functions of one variable. However, for functions of vectors, deterministic quadrate methods can be very inefficient. To numerically integrate a function of a two-dimensional vector, equally spaced grid points over a two-dimensional surface are required. For instance a 10x10 grid requires 100 points. If the vector has 100 dimensions, the same spacing on the grid would require 10^{100} points – far too many to be computed. 100 dimensions are by no means unreasonable, since in many physical problems, a "dimension" is equivalent to a degree of freedom.

Monte Carlo methods provide a way out of this exponential time-increase. As long as the function in question is reasonably well-behaved, it can be estimated by randomly selecting points in 100-dimensional space, and taking some kind of average of the function values at these points. By the law of large numbers, this method will display convergence – i.e., quadrupling the number of sampled points will halve the error, regardless of the number of dimensions.

A refinement of this method is to somehow make the points random, but more likely to come from regions of high contribution to the integral than from regions of low contribution. In other words, the points should be drawn from a distribution similar in form to the integrand. Understandably, doing this precisely is just as difficult as solving the integral in the first place, but there are approximate methods available: from simply making up an integrable function thought to be similar, to one of the adaptive routines discussed in the topics listed below.

A similar approach involves using low-discrepancy sequences instead – the quasi-Monte Carlo method. Quasi-Monte Carlo methods can often be more efficient at numerical integration because the sequence "fills" the area better in a sense and samples more of the most important points that can make the simulation converge to the desired solution more quickly.

Integration Methods

1. Direct sampling methods
   - (a) Importance sampling
   - (b) Stratified sampling
   - (c) Recursive stratified sampling
   - (d) VEGAS algorithm
2. Random walk Monte Carlo including Markov chains
   - (a) Metropolis-Hastings algorithm
3. Gibbs sampling

Notes

Optimization

Another powerful and very popular application for random numbers in numerical simulation is in numerical optimization. These problems use functions of some often large-dimensional vector that are to be minimized (or maximized). Many problems can be phrased in this way: for example a computer chess program could be seen as trying to find the optimal set of, say, 10 moves which produces the best evaluation function at the end. The traveling salesman problem is another optimization problem. There are also applications to engineering design, such as multidisciplinary design optimization.

Most Monte Carlo optimization methods are based on random walks. Essentially, the program will move around a marker in multi-dimensional space, tending to move in directions which lead to a lower function, but sometimes moving against the gradient.

Optimization Methods

1. Evolution strategy
2. Genetic algorithms
3. Parallel tempering
4. Simulated annealing
5. Stochastic optimization
6. Stochastic tunneling

Inverse Problems

Probabilistic formulation of inverse problems leads to the definition of a probability distribution in the model space. This probability distribution combines a priori information with new information obtained by measuring some observable parameters (data). As, in the general case, the theory linking data with model parameters is nonlinear, the a posteriori probability in the model space may not be easy to describe (it may be multimodal, some moments may not be defined, etc.).

When analyzing an inverse problem, obtaining a maximum likelihood model is usually not sufficient, as we normally also wish to have information on the resolution power of the data. In the general case we may have a large number of model parameters, and an inspection of the marginal probability densities of interest may be impractical, or even useless. But it is possible to pseudorandomly generate a large collection of models according to the posterior probability distribution and to analyze and display the models in such a way that information on the relative likelihoods of model properties is conveyed to the spectator. This can be accomplished by means of an efficient Monte Carlo method, even in cases where no explicit formula for the a priori distribution is available.

The best-known importance sampling method, the Metropolis algorithm, can be generalized, and this gives a method that allows analysis of (possibly highly nonlinear) inverse problems with complex a priori information and data with an arbitrary noise distribution.

Computational Mathematics

Monte Carlo methods are useful in many areas of computational mathematics, where a lucky choice can find the correct result.

Notes

A classic example is Rabin's algorithm for primality testing: for any n which is not prime, a random x has at least a 75% chance of proving that n is not prime. Hence, if n is not prime, but x says that it might be, we have observed at most a 1-in-4 event. If 10 different random x say that "n is probably prime" when it is not, we have observed a one-in-a-million event.

In general a Monte Carlo algorithm of this kind produces one correct answer with a guarantee n is composite, and x proves it so, but another one without, but with a guarantee of not getting this answer when it is wrong too often — in this case at most 25% of the time.

Monte Carlo and Random Numbers

Interestingly, Monte Carlo simulation methods do not always require truly random numbers to be useful — while for some applications, such as primality testing, unpredictability is vital. Many of the most useful techniques use deterministic, pseudo-random sequences, making it easy to test and re-run simulations. The only quality usually necessary to make good simulations is for the pseudo-random sequence to appear "random enough" in a certain sense.

What this means depends on the application, but typically they should pass a series of statistical tests. Testing that the numbers are uniformly distributed or follow another desired distribution when a large enough number of elements of the sequence are considered is one of the simplest and most common ones.

General

Statistics Portal

1. Auxiliary field Monte Carlo
2. Bootstrapping (statistics)
3. Demon algorithm
4. Evolutionary Computation
5. Las Vegas algorithm
6. Markov chain
7. Molecular dynamics
8. Monte Carlo option model
9. Monte Carlo integration
10. Quasi-Monte Carlo method
11. Random number generator
12. Randomness
13. Resampling (statistics)

Application Areas

1. Graphics, particularly for ray tracing; a version of the Metropolis-Hastings algorithm is also used for ray tracing where it is known as Metropolis light transport
2. light transport in biological tissue
3. Monte Carlo methods in finance

Notes

4. Reliability engineering
5. In simulated annealing for protein structure prediction
6. In semiconductor device research, to model the transport of current carriers
7. Environmental science, dealing with contaminant behavior
8. Search and Rescue and Counter-Pollution. Models used to predict the drift of a life raft or movement of an oil slick at sea
9. In probabilistic design for simulating and understanding the effects of variability
10. In physical chemistry, particularly for simulations involving atomic clusters
11. In biomolecular simulations
12. In polymer physics
    (a) Bond fluctuation model
13. In computer science
    (a) Las Vegas algorithm
    (b) LURCH
    (c) Computer go
    (d) General Game Playing
14. The movement of impurity atoms (or ions) in plasmas in existing and tokamaks (e.g.: DIVIMP)
15. Nuclear and particle physics codes using the Monte Carlo method:
    (a) GEANT — CERN's simulation of high energy particles interacting with a detector
    (b) CompHEP, PYTHIA — Monte-Carlo generators of particle collisions
    (c) MCNP(X) — LANL's radiation transport codes
    (d) MCU — universal computer code for simulation of particle transport (neutrons, photons, electrons) in three-dimensional systems by means of the Monte Carlo method
    (e) EGS — Stanford's simulation code for coupled transport of electrons and photons
    (f) PEREGRINE — LLNL's Monte Carlo tool for radiation therapy dose calculations
    (g) BEAMnrc — Monte Carlo code system for radiotherapy sources (LINAC's)
    (h) PENELOPE — Monte Carlo for coupled transport of photons and electrons, with applications in radiotherapy
    (i) MONK — Serco Assurance's code for the calculation of k-effective of nuclear systems of foam and cellular structures of tissue morphogenesis
16. Computation of holograms
17. Phylogenetic analysis, i.e., Bayesian inference, Markov chain Monte Carlo

Other Methods Employing Monte Carlo

Notes

1. Assorted random models, e.g. self-organised criticality
2. Direct simulation Monte Carlo
3. Dynamic Monte Carlo method
4. Kinetic Monte Carlo
5. Quantum Monte Carlo
6. Quasi-Monte Carlo method using low-discrepancy sequences and self avoiding walks
7. Semiconductor charge transport and the like
8. Electron microscopy beam-sample interactions
9. Stochastic optimization
10. Cellular Potts model
11. Markov chain Monte Carlo
12. Cross-entropy method
13. Applied information economics
14. Monte Carlo localization

Monte Carlo as a Procedure

Monte Carlo is an estimation procedure. The basic idea is as follows. You want to know the average value of some random variable. You can't work out what its distribution is, exactly, or you don't want to do integrals numerically, but you can take samples from that distribution. (The random variable may; for instance, be some complicated function of variables with simple distributions, or they distribution may have a hard-to-compute normalizing factor ["partition function" in statistical mechanics].) To estimate it, you simply take samples, independently, and average them. If you take enough samples, then the law of large numbers says your average must be close to the true value. The central limit theorem says that your average has a Gaussian distribution around the true value.

Here's one of the canonical examples. Say you want to measure the area of a shape with a complicated, irregular outline. The Monte Carlo approach is to draw a square around the shape and measure the square. Now you throw darts into the square, as uniformly as possible. The fraction of darts falling on the shape gives the ratio of the area of the shape to the area of the square. Now, in fact, you can cast almost any integral problem, or any averaging problem, into this form. So you need a good way to tell if you're inside the outline, and you need a good way to figure out how many darts you should throw. Last but not least, you need a good way to throw darts uniformly, i.e., a good random number generator. That's a whole separate art I shan't attempt to describe.

Now, in fact, you don't strictly need to sample independently. You can have dependence, so long as you end up visiting each point just as many times as you would with independent samples. This is useful, since it gives a way to exploit properties of Markov chains in designing your sampling strategy, and even of speeding up the convergence of your estimates to the true averages. (The classic instance of this is the Metropolis-Hastings algorithm, which gives you a way of sampling from a distribution where all you have to know is the ratio of the probability densities at any two points. This is extremely useful when, as in many problems in statistics and statistical mechanics, the density itself contains a complicated normalizing factor; it drops out of the ratio.)

Notes

Monte Carlo methods originated in physics, where the integrals desired involved hydrodynamics in complicated geometries with internal heating, i.e., designing nukes. The statisticians were surprisingly slow to pick up on it, though by now they have, especially as "Markov chain Monte Carlo," abbreviated "MC Monte Carlo" (suggesting an gambling rapper) or just "MCMC". Along the way they picked up the odd idea that Monte Carlo had something to do with Bayesians. In fact it's a general technique for estimating sample distributions and related quantities, and as such it's entirely legitimate for frequentists. Physicists now sometimes use the term for any kind of stochastic estimation or simulation procedure, though I think it's properly reserved for estimating integrals and averages.

Notes Monte Carlo simulation is a technique for *iteratively* estimating a deterministic model using sets of random numbers as inputs. This method is frequently used when the model is complex, nonlinear, or includes more than just a couple unsure parameters. A simulation can normally involve *over 10,000 evaluations* of the model, a task which in the past was only realistic using super computers.

Case Study

IT Adoption in Manufacturing

Amidst all the sectors, barring probably the construction industry which is still nascent in IT adoption, manufacturing is probably in the evolving side of the curve, feels Uma Balakrishnan, CEO, Axcend Automation & Software Solutions Pvt Ltd, Bangalore (http://bit.ly/F4TAxcend).

While retail and BFSI adopted early on, mainly because of size/scale of operations, diverse geographies, and direct consumer interface with the using population, manufacturing has shown a mixed trend, she observes, during the course of a recent e-mail interaction with eWorld. Indian manufacturing enterprises that acquired global companies and turned multinational have been early in the increased scale of adoption cycle, while smaller entrepreneurial operations are still very nascent in IT adoption, explains Uma.

Three primary factors drive IT adoption, she elaborates: (a) Sheer size and scale that necessitate moving into IT automated systems, as manually it becomes challenging to run business; (b) external world in the form of clients or statutory regulatory bodies that make the manufacturers align to some best business practices using IT systems; and (c) internal management maturity to leverage IT and gain competitive advantage in business through quality, cost, operational excellence, or shortened time to market with product.

Excerpts from the interview.

What are the prevalent technologies in manufacturing — especially in discrete industries — considering that India is fast becoming a global manufacturing hub in some sectors?

Manufacturing organisations have multiple functional needs and technologies can be broadly mapped under horizontal and vertical dimensions. Business IT enablers such as ERP (enterprise resource planning) systems have reached a higher level of adoption as general business IT tools of the manufacturing.

The prevalent technologies vary across functions in the organisations. At the shop floor level, CNC (computer numerically controlled), PLC (programmable logic controls), and HMI (human machine interface) provide the industrial automation technologies that serve in the automated control of the physical manufacturing.

Contd...

Notes

More advanced users have matured into centralised plant operations control with SCADA technologies. In terms of product design and manufacturing process design, 3D design, PLM (product lifecycle management), and digital manufacturing are evolving in adoption cycle from the nascent stage of 2D design platform.

Very advanced users, though a few, have evolved to use complete plant and simulation, with virtual commissioning and advanced tools such as process simulate, factory flow, factory CAD and so on. Here, practically before a single rupee is invested in physical plant infrastructure, the entire model and operations can be visualised on different what-if business scenarios.

In the horizontal functions, ERP (enterprise resource planning) has a mature adoption curve, while SCM (supply chain management) systems are in mid stage of maturity in adoption. Very progressive companies have looked at vertical application integrations, between the shop floor and business systems, to provide competitive advantage and have adopted manufacturing execution systems (MES) and manufacturing intelligence solutions.

Fundamentally, we can look at the users from two perspectives.

First, the manufacturers who are part of the Indian manufacturing sourcing hub which serves the global OEMs. They are more aligned to technologies prevalent in sourcing geography and are early adopters of the same, especially in the area of product design and quality management which has direct impact on the end product being shipped by the global client. So, 3D product design, PLM, online inspection and quality management system adoptions are quite concurrent with the western users.

Question

Are there any differences in these technology adoptions between global users and Indian manufacturing users?

Source: http://www.thehindubusinessline.in/ew/2010/04/19/stories/2010041950140400 .htm

6.3 Summary

- The uniform distribution is hardly ever used straightforwardly in any econometric model. Other distributions, especially the normal or Gaussian distribution, are much more established. Various methods survive to convert a uniform RNG into a non-uniform counterpart.
- Monte Carlo simulation methods are especially useful in studying systems with a large number of coupled degrees of freedom, such as fluids, disordered materials, strongly coupled solids, and cellular structures
- Interestingly, Monte Carlo simulation methods do not always require truly random numbers to be useful — while for some applications, such as primality testing, unpredictability is vital.

6.4 Keywords

Relationships: Link entities together e.g. a part may be processed by a machine.

Simulation Executive: Responsible for controlling the time advance and executing discrete events.

Time Slicing: Advances the model by a fixed amount each time, regardless of the absence of any events to carry out.

Notes

6.5 Self Assessment

Fill in the blanks:

1. Monte Carlo methods are often used when simulating physical and ..... systems.
2. An ..... is a notification that occurs in response to an action
3. ..... Method is used for uniform random numbers.
4. The ..... distribution is hardly ever used straightforwardly in any econometric model.
5. If a ..... distribution function and density are obtainable, but no analytical formula for the inverse.
6. The most significant non-uniform distribution is the standard normal, which, unluckily, does not have a .....
7. The ..... partitions the standard normal thickness into horizontal blocks of equal area.
8. Testing the normal RNG is possibly more significant than the fundamental uniform, as it offers the basis leading econometric simulation .....
9. The origins of the approach can be traced back to Lord Raleigh who used it to develop approximate solutions to simple ..... differential equations.
10. Monte Carlo methods are often used when simulating ..... and mathematical systems.
11. Monte Carlo methods are useful for phenomena with significant uncertainty in inputs, such as the calculation of ..... in business.
12. Uses of Monte Carlo methods require large amounts of .....
13. Monte Carlo methods have recently been applied in game playing related ..... theory.
14. The opposite of Monte Carlo simulation might be considered deterministic using ..... estimates.
15. A refinement of this method is to somehow make the points random, but more likely to come from regions of high contribution to the integral than from regions of .....

6.6 Review Questions

1. Explain why the polar technique is simple to execute, but not chiefly fast?
2. "Monte Carlo simulation are often encountered in the and simulation literature". Comment.
3. "The Monte Carlo label for the methodology is, in fact, attributed to this group". Elaborate.
4. Do you think Random methods of computation and experimentation can be arguably traced back to the earliest pioneers of probability theory?
5. Discuss the Applications of Monte Carlo.
6. Describe the Integration of Monte Carlo and its methods.
7. Describe the optimization of Monte Carlo and its methods.
8. Probabilistic formulation of inverse problems leads to the definition of a probability distribution in the model space. Give reasons

Notes

9. Make an analysis on Monte Carlo as a Procedure.
10. Give the various methods employing Monte Carlo.

Answers: Self Assessment

1.	Mathematical	2.	Event	3.	Linear Congruential
4.	uniform	5.	cumulative	6.	logical inverse
7.	ziggurat	8.	experiments	9.	partial
10.	physical	11.	risk	12.	random numbers
13.	artificial intelligence	14.	single-point	15.	low contribution

6.7 Further Readings

Books

Balci, O.,(1994), Validation, verification, and testing techniques throughout the life cycle of a simulation study, Annals of Operations Research, 53: 121 –173.

Balci, O., (2001), A methodology for certification of and simulation applications, ACM Transactions on and Computer Simulation, 11: 352–377.

Birta, L.G. and Ozmizrak, N.F., (1996), A knowledge-based approach for the validation of simulation models: The foundation, ACM Transactions on and Computer Simulation, 6: 67–98.

Boehm, B.W., (1979), Software engineering: R&D trends and defense needs, in: P. Wegner (Ed.), Research Directions in Software Technology, MIT Press, and Cambridge, MA.

Cellier, F.E., (1986), Combined discrete/continuous system simulation – application, techniques and tools, in Proceedings of the 1986 Winter Simulation Conference.

Department of Defense (DoD) Recommended Practices Guide (RPG) for and Simulation VV&A, Millennium Edition (available at http://vva.dmso.mil).

General Accounting Office, (1976), Report to the Congress: Ways to improve management of federally funded computerized models, report LCD-75-111, U.S. General Accounting Office, Washington, DC.

General Accounting Office, (1979), Guidelines for model evaluation, report PAD-79-17, U.S. General Accounting Office, Washington, DC.

General Accounting Office, (1987), DOD simulations: Improved assessment procedures would increase the credibility of results, report GAO/PEMD-88-3, U.S. General Accounting Office, and Washington, DC.

Ören, T.I., (1981), Concepts and criteria to access acceptability of simulation studies, Communications of the ACM, 24: 180–189.

Online link

http://oxford-mphil.com/images/f/fc/Simulation-econometrics.pdf

Notes

Unit 7: Simulation of Queuing System (I)

CONTENTS

Objectives

Introduction

7.1 Simulation of Queuing System

7.1.1 Single-channel Queuing System

7.1.2 Multiple-channels Queuing System

7.2 Rudiments of Queuing Theory

7.3 Computer Simulation for Queuing System

7.4 Summary

7.5 Keywords

7.6 Self Assessment

7.7 Review Questions

7.8 Further Readings

Objectives

After studying this unit, you will be able to:

- Understand simulation of queuing system
- Discuss rudiments of queuing theory

Introduction

The type of queuing system a business uses is an important factor in determining how efficient the business is run. In this project, we examine two types of queuing systems: the single-channel and the multiple-channels queues which are commonly seen in banks and fast food restaurants respectively. We use computer programs to simulate the queues and predict the queue length, waiting time and wait probability. The input to the simulation program is based on the statistics collected over a span of a week. The discrete–event simulation approach is used to model the queuing systems and to analyze the side effects when one system is changed to the other.

As the size of the world's population increases so do the number of queues and their queue length. In the business world, more customers mean more business transactions. Out of the many ways to attract customers, an efficient queuing system plays an important role as it reduces a customer's waiting time. As a result, the shorter waiting time makes customers happy, and one thing for sure is that a happy customer will come back for business again.

This unit begins with a description of the single-channel and multiple-channels queuing systems and their influence on a customer's waiting time and wait probability. We use discrete-event simulation program to verify the live data, and predict the performance if the configuration of the existing queue is changed.

Notes

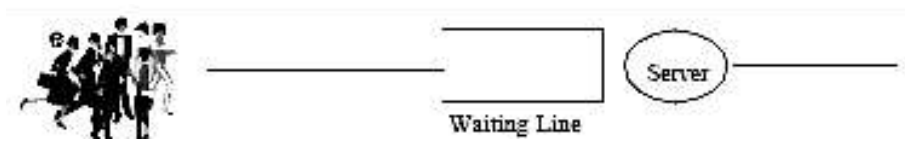
7.1 Simulation of Queuing System

General Queue

In a queuing system, the calling population is assumed to be infinite that is, if a unit leaves the calling population and joins the waiting line or enters service; there is no change in the arrival rate. Figure 7.1 shows the model used to analyze a general queue.

The arrivals occur one at a time in a random order and once the customer joins the queuing system he will eventually receive the service.

Figure 7.1: A Simple Queue



The arrival rate and services are modeled as variables which follow statistical distributions. If the arrival rate is greater than the service rate, the waiting line will grow without bound.

Did u know? **Arrival pattern**

The arrival pattern of consumers to the service system is categorized into two groups: **static** and **dynamic**. These two are further classified depending on the nature of arrival rate and the control that can be work out on the arrival pattern.

In **static arrival process**, the control is based on the nature of arrival rate (random or stable). Random arrivals are either at a stable rate or unstable with time. Therefore to examine the queuing system, it is essential to endeavor to portray the possibility distribution of arrivals.

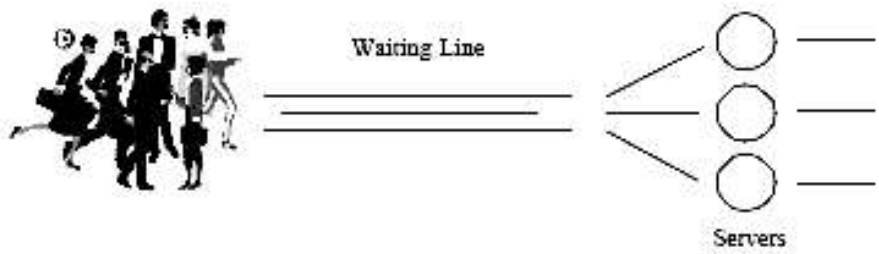
The **dynamic arrival process** is restricted by both service facility and consumers. The service facility regulates its capacity to match alterations in the demand intensity, by either changing the staffing levels at dissimilar timings of service, modifying service charges (like telephone call charges at unusual hours of the day or week) at unusual timings, or permitting entry with schedules.

7.1.1 Single-channel Queuing System

The single-channel queuing system can be seen in places such as banks and post offices, where one single queue will diverge into a few counters. The moment a customer leaves a service station, the customer at the head of the queue will go to the server. The disadvantage of a single-channel queue is that the queue length seems to be very long, thus it can discourage customers from joining the queue.

Notes

Figure 7.2: Single-channel Queuing System

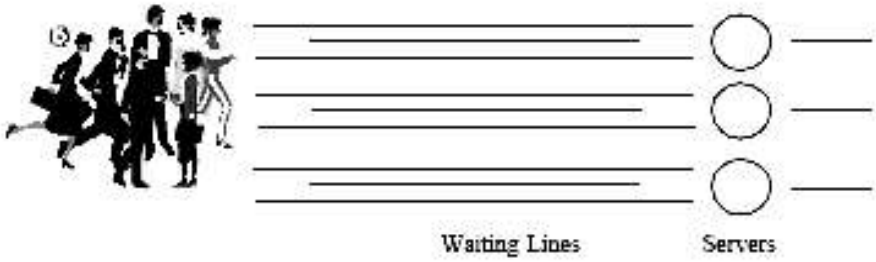


Example: POSB is a banking service entity that uses the single queue system. Customers join one queue and when they reach the front they will go to a server that is idle. The main characteristic of a single-channel queue is the first-come- first-serve feature.

7.1.2 Multiple-channels Queuing System

The multiple-channels queuing system is commonly observed in fast food restaurants like KFC, Burger King, McDonalds, etc. It is a system whereby the customers line up in rows directly in front of each server. They are arranged in relatively straight lines that do not converge. Generally, the customers in a multiple-channels queue feel happier because the queue length is shortened as they are distributed to different counters.

Figure 7.3: Multiple-channels Queuing System



Example: McDonalds is a chain of fast food restaurants that uses the multiple-channels queue. The restaurants are centrally located in areas where many people pass through daily like in Town Centres and around Orchard Road. In this type of queuing system the customers can feel a sense of "speed" when they see less people standing in front of them as compared to that in POSB.

Materials and Methods

This unit evaluates the performance of single-channel and multiple-channels queuing systems, specifically that of fast food restaurants like McDonalds and banks like POSB. To do this, we first went to both McDonalds and POSB to collect the statistical data. We spent a week recording inter-arrival times and service times at different locations and times.

Notes

Using the data collected as the input to the discrete-event simulation program, we set up two simulators for predicting the behaviour of a single-channel queue and a multiple-channels queue respectively.

The purpose of using simulation technique to analyze the collected data is to avoid costly design errors, and to analyze the behaviours of the existing systems. More importantly, simulation can be used to predict the performance of the existing system when the input parameters such as the arrival rate and service rate are changed.

Simulation technique can also be applied to analyze the behaviours of system which has not even been created yet.

Materials Used

The first phase of this project is data collection. We used stopwatches to time the inter-arrival time and service time, and calculate the average timings as shown in Tables 7.1 and 7.2.

Table 7.1: Data Collected at Choa Chu Kang and Clementi MacDonalds from 15:30 to 16:30 and 10:30 to 11:30

	Inter-Arrival Time (sec)	Service Time (sec)
Average	48	70

Table 7.2: Data Collected at Taman Warna and Clementi POSB from 15:30 to 16:30 and 15:10 to 16:10 respectively

	Inter-Arrival Time/s	Service Time/s
Average	30	130

Methods

Discrete-event Approach

Discrete event is a technique used to model the real-world scenarios. In the queuing model two types of events are used, namely arrival and departure. The arrival corresponds to the real-world event when a customer reaches a service station, and the departure corresponds to the event when the customer leaves. Due to the causality constraints, the arrival event for a customer must be executed before its departure event.

Each event has a timestamp corresponding to the wall-clock time when it occurs. Discrete-event technique has been widely used in the simulation of communication and transportation systems, such as telephone networks, seaport and airport operations, etc.

What if POSB changes its queue to multiple-channels and McDonalds changes its queue to single-channel?

We first validate the data collected and compare the deviation of the observed and predicated queue length, waiting time and wait probability. As the simulation results follow closely the observed values, we brought up the question of whether McDonalds and POSB will benefit if they switched their queuing systems. The simulator was used to find out the answers.

The changes to the input data are as follows: To convert from a single-channel queue to a multiple-channels queue for n servers we will have to divide the arrival rate by n because the

Notes

arriving customers are distributed to the n channels. Therefore, the arrival rate to each channel is scaled down by a factor of n. As for converting a multiple channels queue to a single-channel queue, the arrival rate is multiplied by n due to the merging effect. The service rate remains unchanged for the conversions.

Tasks Analyze different examples for arrival and departure.

Results

Tables 7.3 and 7.4 show the simulation results. As observed in both tables, the server utilization remains fairly constant as the total workload does not changed. As for converting from a multiple-channels queue to a single-channel queue, Table 7.3 shows that the queue length is reduced by 48%. More significantly, the wait probability is reduced from 0.68 to 0.35, and the average queue time is reduced from 106 seconds to 11 seconds. This implies that the probability that a customer coming to the McDonalds will have to wait for service is significantly reduced by half, and even if the customer is not immediately served the waiting time is reduced by 9 folds. As such, the change from a multiple-channels queue to single-channel will make a substantial improvement in the queuing performance.

Table 7.3: Comparison of Queuing Performance for McDonalds

Queue Type	Average Queue Length	Ave. Queue Time (sec)	Wait Probability	Server Utilization
Multiple-channels (existing)	1.52	106.36	0.68	0.69
Single-channel	0.78	10.86	0.35	0.68

On the other hand, Table 7.4 shows that the average queue length, average queue time and the wait probability all become worse if the POSB is to convert its single channel queue to multiple-channels.

Table 7.4: Comparison of Queuing Performance for POSB

Queue Type	Average Queue Length	Ave. Queue Time (sec)	Wait Probability	Server Utilization
Single-channels (existing)	4.25	127.34	0.68	0.86
Multiple-channels	5.91	883.63	0.87	0.84

7.2 Rudiments of Queuing Theory

As you know that a queue is a waiting for service. In general form of model of a queue, you assume that customers arrive randomly. There is only one service counter, which can serve only one customer at a time. The amount of time it takes to serve a customer also varies randomly. If the service counter is busy, arriving customers join the queue. From the queue the customers are served on a FCFS means first-come-first-served basis. As soon as a customer is served he gets out of the system like first-in-first-out system.

Notes

Poisson arrival pattern: To facilitate the analysis, so assume that the interarrival times of the customers are such that there is a fixed long-term average time gap between two arrivals. Let this time be α. Let us suppose that the customers arrive independently, and that the probability of and arrival during any period depends only on the length of that period.

Therefore, the probability of a customer arriving during a very small slice of time h is h/ α. Hence the probability of a customer not arriving during time h is (1 – h/ α). Now let us define that

f{t) = Probability that the next customer does not arrive during the interval t given that the previous customer arrived at t = 0, and likewise;

f(t + h) = Probability that the next customer does not arrive during the interval (t + h) given that the previous customer arrived at t = 0.

Since the arrivals of customers in different periods are independent events (i.e., the queue has no memory), we can write

$$f(t + h) = f(t).\left(1 - \frac{h}{\alpha}\right)$$

$$\frac{f(t + h) - f(t)}{h} = \frac{-f(t)}{\alpha}$$

Taking limits on both sides as h tends to zero, we get

$$\frac{df(t)}{dt} = \frac{-f(1)}{\alpha}$$

The solution of this differential equation is

$$f(t) = ce^{-t/\propto} \qquad ...(1)$$

Since at time t = 0 an arrival has just taken place, the probability of a nonlinear at t = 0 is 1. That is, f(0) = 1, and therefore the constant c in Eq. (1) is unity. Thus

$$f(t) = ce^{-t/\alpha} \qquad ...(2)$$

Here is two very simple assumptions, first censtancy of a long-term average and second statistical independence of arrivals have led to Eq. (2) which gives the probability that the next customer does not arrive before time t has elapsed since the arrival of the last customer.

The probability that a customer arrives during an infinitesimal interval between t and t + δt is given by the product of the probability that no customer arrives before time t and the probability that exactly one customer arrives during δt. This product is

$$\left(e^{-t/\alpha}\right).\left(\frac{\delta t}{\alpha}\right)$$

We can say, the probability density function of the interarrival time is 1

$$\frac{1}{\alpha}.e^{-1/\alpha} \qquad ...(3)$$

The integral

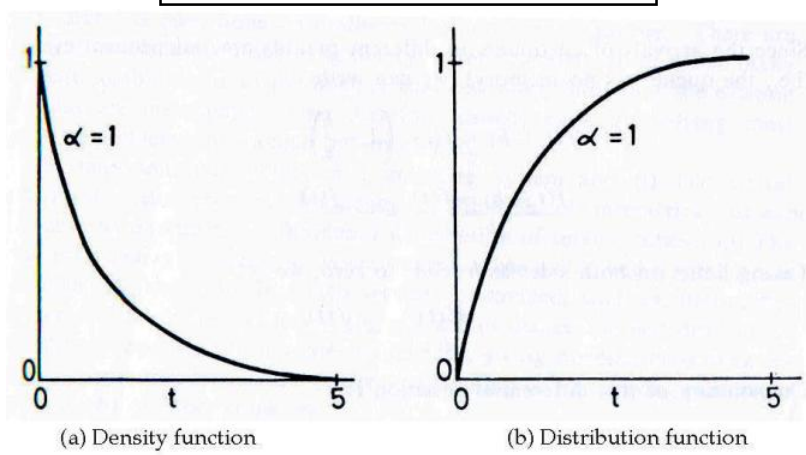
$$\frac{1}{\alpha}\int_u^t e^{-t/x}\, dt = 1 - e^{-t/\alpha} \qquad ...(4)$$

is the probability distribution function.

Notes

The curves for Eqs. (3) and (4) are shown in Figure 7.4. These are the curves for the exponential distribution. The curve in Figure 7.4(b) gives the probability that the next customer arrives by time t, given that the preceding customer arrived at time zero. Customarily the inverse of the average interarrival time is denoted by λ, which is nothing but the average number of customers arriving at the system per unit time.

Figure 7.4 : Interarrival Time of Customers



Having found that the interarrival time is governed by exponential distribution, we will now find the probability that exactly k customers have arrived during the time interval t (assuming that no customer had arrived by time t = 0).

Let $q_k(t)$ be the probability that exactly k arrivals take place between the start time zero and t, for k = I, 2, 3, ... Then we can write

$q_l(t + h)$ = (Probability that no arrivals take place between time zero and t). (probability that one arrival takes place during time h) + (probability that one arrival takes place between time zero and t) . (probability that no arrivals take place during time h)

$$= f(t).\frac{h}{\alpha} + q_1(t).\left(1 - \frac{h}{\alpha}\right)$$

where f(t), as previously determined, is $e^{-t/\alpha}$.

Therefore,

$$\frac{q_1(t+h) - q_1(t)}{h} = \frac{1}{\alpha}[f(t) - q_1(t)]$$

as $h \to 0$ this expression becomes

$$\frac{dq_1(t)}{dt} = \frac{1}{\alpha}[f(t) - q_1(t)]$$

The solution of this differential equation can easily be seen to be

$$q_1(t) = \frac{1}{\alpha}e^{-t/a} = \frac{1}{\alpha}f(t)$$

Extending the same argument (since at most one arrival can take place in the interval h) we get

$$q_2(t) = \left(\frac{t}{\alpha}\right)^2 \frac{1}{2!}.f(t)$$

Notes

It can be seen that, in general

$$\frac{dq_k(t)}{dt} = \frac{q_{k-1}(t) - q_k(t)}{\alpha}$$

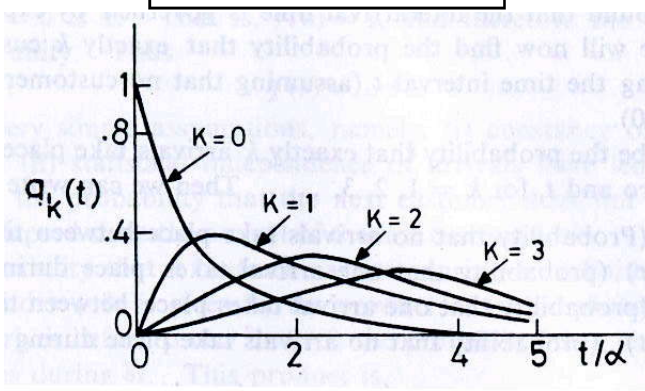
We have already solved this differential equation for k = 1 and 2 (for k = 0, $q_o(t) = f(t)$). Solving it successively for k = 3, 4, ... , we will get

$$q_k(t) = \left(\frac{t}{\alpha}\right)^k \frac{1}{k!} e^{-t/\alpha} \qquad ...(5)$$

Expression (5) is known as the Poisson distribution formula. It is one of the most important and widely encountered distributions. Curves for $q_k(t)$ for several values of k are shown in Figure 7.5. Note that for k = 0, we get the negative exponential curve,

$$q_0(t) = e^{-t/a} = f(t)$$

Figure 7.5: Poisson Distribution



We have thus seen that if the interarrival time is distributed exponentially, the number of arrivals is given by Poisson distribution. Therefore, the two terms negative exponential arrival or Poisson arrival are often used interchangeably. It should be emphasized here that Poisson arrival pattern is just one of many possible arrival patterns in a queueing situation. It results from the three assumptions that (i) the successive arrivals are statistically independent of each other, that (ii) there is a long-term inter-arrival time constant ex, and that (iii) the probability of an arrival taking place during a time interval h is directly proportional to h.

Exponential service time: Let us make similar assumptions about the servicing process, namely, (i) the statistical independence of successive servicings, (ii) the long-term constancy of service time, and (iii) the probability of completing the .service for a customer during a time interval h is proportional to h. Therefore, as in the case of interarrival time, we will get

$$g(t) = e–t/\beta \qquad ...(6)$$

where g(t) is the probability that a customer's service could not be completed in time t, (given that the previous customer's service was completed at time zero) and β is the long-term average service time.

Operating characteristics: Now we have completely defined the four parameters of the queueing system. These are (i) Poisson arrival pattern, (ii) negative exponential service times, (iii) a single server, and (iv) the first-come-first-served queue discipline. For this particular queueing system we will now derive some interesting and useful statistics about the system:

Notes

Clearly, at any given time t the probability of the service counter being busy is

$$\frac{\text{average service time}}{\text{average arrival gap}} = \frac{\beta}{\alpha} = \rho \qquad \text{... (7)}$$

ρ is an important ratio. It is called 'the utilization factor of the service facility.

It immediately follows from Eq. (7) that the probability of finding the service counter free is

$$1 - \rho \qquad \text{...(8)}$$

Let $P_n(t)$ be the probability of exactly n customers being in the system at time I. Let h > 0 be a very small slice of time. The probability of one customer arriving and no customer departing during interval h is

$$\frac{h}{\alpha}.\left(1-\frac{h}{\beta}\right)$$

Likewise, the probability of one customer arriving and one customer departing during interval h is

$$\frac{h}{\alpha}.\frac{h}{\beta}$$

The probability of no customer arriving and one customer departing is

$$\left(1-\frac{h}{\alpha}\right).\frac{h}{\beta}$$

and finally of no customer arriving and no customer departing is

$$\left(1-\frac{h}{\alpha}\right).\left(1-\frac{h}{\beta}\right)$$

Since h is so small that no more than one departure and no more than one arrival can take place, these are the only possible changes that could occur during an interval h.

For the queueing system to have n customers at time (t + h), it must have had either n or (n + 1) or (n – 1) customers at time t. The probability that there are n customers in the system at time (t + h) can therefore be expressed as the sum of these three probabilities. Thus for any n > 0 we can write

$$P_n(t+h) = P_n(t).\left(1-\frac{h}{\alpha}\right).\left(1-\frac{h}{\beta}\right) + P_n(t).\frac{h}{\alpha}.\frac{h}{\beta} + P_{n+1}(1)\left(1-\frac{h}{\alpha}\right).\frac{h}{\beta} + P_{n-1}(t).\frac{h}{\alpha}.\left(1-\frac{h}{\beta}\right)$$

From this we get

$$\frac{P_n(t+h) = P_n(t)}{h} = \frac{1}{\beta}P_{n+1}(t) + \frac{1}{\alpha}P_{n+1}(t) - \left(\frac{1}{\alpha}+\frac{1}{\beta}\right)P_n(t) + \frac{h}{\alpha\beta}[2P_n(t) - P_{n+1}(t) - P_{n-1}(t)]$$

Taking the limits of both sides of this equation when h tends to zero

$$\lim_{h\to 0}\frac{P_n(t+h) - P_n(t)}{h} = \frac{1}{\beta}P_{n+1}(t) - \left(\frac{1}{\alpha}+\frac{1}{\beta}\right)P_n(t) + \frac{1}{\alpha}P_{n-1}$$

Notes

that is

$$\frac{dPn(t)}{dt} = \frac{1}{\beta}P_{n+1}(t) - \left(\frac{1}{\alpha} + \frac{1}{\beta}\right)P_n(t) + \frac{1}{\alpha}P_{n-1} \quad ...(9)$$

Equation (9) holds for all n > 0. When n = 0, the contributions made by the P_{n-1} terms would be zero. Therefore

$$\frac{dP_0(t)}{dt} = \frac{1}{\beta}P_1(t) - \frac{1}{\alpha}P_0(t) \quad ...(10)$$

$P_n(t)$ can be obtained by solving the two differential Equations (9) and (10). We are most interested in a steady state solution. If $\rho < 1$, after the passage of a sufficiently long time the queue would reach an equilibrium, and $P_n(t)$ would converge to a constant. Thus we can set

$$\frac{dP_n}{dt} = 0$$

Applying this to Eq. (10) and (9) gives, respectively

$$P_1 = \frac{\beta}{\alpha}P_0(t) = \rho P_0 \quad ...(11)$$

and

$$P_{n+1} = (1+\rho).P_n - \rho.P_{n-1} \quad ...(12)$$

Repeated substituion of Eq. (11) into (12) gives

$Pn = \rho^n P_0$,

for all n > 0 and $\rho < 1$. ...(13)

Clearly

$$\sum_{n=0}^{\infty} P_n = 1$$

Therefore

$$P_0 \sum_{n=0}^{\infty} \rho^n = 1$$

or $P_0 \frac{1}{1-\rho} = 1$, since $\rho < 1$

Therefore,

$P_0 = (1 - \rho)$,

the same result as in Eq. (8).

The average number of customers in the system is given by

$$\sum_{n=0}^{\infty} n.P_n = P_0 \sum n.\rho^n = \frac{\rho}{1-\rho} \quad ...(14)$$

The average number of customers in the system but not being served (i.e., the average queue length) is the difference of the average number of customers in the system and the probability that at least one customer is in the system (which is nothing but ρ). Thus

Notes

average queue length = $\frac{\rho}{1-\rho} - \rho$

$$= \frac{\rho^2}{1-\rho} \quad ...(15)$$

Substituting Eq. (14) into (13), the probability of n customers being in the system can also be expressed as

$$P_n = \rho^n(1 - \rho) \quad ...(16)$$

Probability of n customers being in the queue is the same as the probability of (n + 1) customers being in the system which is

$$\rho^{n+1}(1 - \rho), \text{ for } n > 0 \quad ...(17)$$

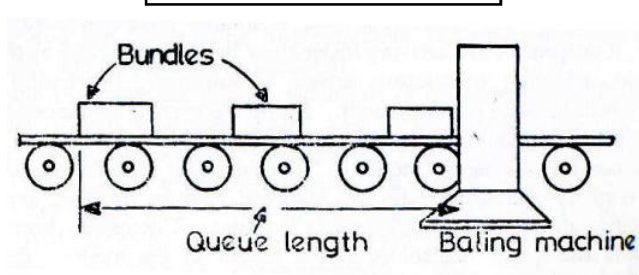
Probability of more than n customers being in the system

$$\begin{aligned} &= 1\sum_{t=0}^{n}\rho^i(1-\rho) \\ &= 1-[(1-\rho)+\rho(1-\rho)+\rho^2(1-\rho)+...+\rho^{n-1}(1-\rho)+\rho^n(1-\rho)] \\ &= 1-[1-\rho^{n+1}] \\ &= \rho^{n+1} \end{aligned} \quad ...(18)$$

These and similar other statistics about the queue are called the operating characteristics of the queueing system. Derivation of some of the order characteristics are left as exercises. Usually one is interested in only some of these quantities. As an illustration of how a particular formula from queueing theory can be put to use, let us consider the following design problem.

Problem: Suppose you are to design a roller conveyor system for baling and strapping large jute bundles, as shown in Figure 7.6. The number of

Figure 7.6: Baling of Bubdles



bundles arriving per unit time is nQt fixed, but it obeys the Poisson distribution law with average interarrival time oc of 3 minutes. The time taken by the baling machine is also not fixed but obeys the negative exponential distribution with average service time β of 2.5 minutes. How long should the conveyor be built so that it is sufficient to hold the bundles waiting to be baled, if each bundle is 1.5 meters long?

Solution: The utilization factor in this case is

$$\rho = \frac{\beta}{\alpha} = \frac{2.5}{3.0} = \frac{5}{6},$$

and therefore

Average number of bundles on the conveyor $= \frac{\rho}{1-\rho} = 5.$

Notes

A conveyor just long enough to hold only 5 bundles would not do, because this is only the average number of bundles. On the other hand, we ask the question : What is the maximum number of bundles that can build up on the conveyor? Theoretically the answer is an infinite number. For design purposes, however, we can use formula (4-18), and then by setting a reasonable limit on the probability figure we can evaluate n as follows:

Suppose we build the conveyor so that 99 per cent of the time it is sufficient to hold the bundles waiting to be baled. Thus the probability of having more than n bundles being in the system is

$$(5/6)^{n+1} = .01,$$

and yields

$$n = 24.26$$

Thus if the conveyor is made 24 × 1.5 = 36 meters long, with 99 per cent assurance we can say that it is long enough. This would be an acceptable risk.

Beginning from the time of birth (generally concerning an approximately 9-month period from the instant of commencement) until death (an entire life-time – whether concise, wide-ranging or in between) and at many moments along the way human beings frequently discover themselves waiting for things, events, circumstances, etc. A foremost topic of Applied Mathematics that deals with this event of waiting is known as Queuing Theory. Using the word "Queue", which is more ordinary in British than American English and means "a line up" or "to form a line", a closely consistent body of mathematical theory has been developed to explain this ordinary human activity – theory applicable to normal economic activity. Practical applications can be made to the event of customers in anticipation of the delivery of goods/services, in addition to goods-in-process approaching to be finished goods.

Queuing Theory occurs from the use of powerful mathematical analysis to notionally explain production processes together with statistical/probabilistic methods to report for changeable dynamic outlines within the stages of a creative process. The problem to be met - that occasioned the growth of such theory - is simply entitled "congestion", what happens when a system does not function easily or proficiently.

Here the reader is invited to summon up his/her own multiple common examples for the application of the hypothetical concepts that pursue. The claim for solutions to congestion problems occurs all across the board of the worldwide economy, in addition to the course of every day living. The origination of the official study of Queuing Theory is accredited to A. K. Erlang, a Danish telephone engineer who in the 1920's was endeavoring to predict telephone call service.

As discussed above Queuing Theory observes the progress of consumers (people) pursuing obtainable services, in addition to goods-in- process (things) obtaining the status of completed goods (whether capital or consumer goods). There are three regions of focus: Arrivals, Queue, Service Facility - each of which is further subdivided by a multiplicity of logical detail. A single process may include more than one stage/station, if the person/product passes through a series of Service Facilities; and it may also have more than one actual Queue/channel/ waiting-line leading to the following Service Facility(ies).

The first region of focal point considers Arrivals - entries into a productive organization. (Please note down that the theory pertains equally to customers and goods-in-process). The number of Arrivals is important (limited - few/many; or unlimited - probably infinite). The outline of arrival may confess of a tight/loose agenda (so many per time period) or completely randomly Arrivals. The behavior of the Arrivals may include quiet, patient people, stable/unstable objects, argumentative children/adults, etc. Therefore the first region of worry is the nature and number of the object(s) entering the process.

Secondly, concentration is drawn to the Queue/Waiting-Line itself, which once recognized could have a limited (few or many) or unlimited inhabitants. Of greater significance is the

Notes

Queue Discipline - the process by which a new Arrival advances to truly begin receiving service. The most ordinary technique is FIFO (first in, first out). But there are other methods: LIFO (last in, first out); pre-assigned precedence (advance to service separately determined before arrival); priority by types (categories established before arrival for a variety of reasons, e.g., length of service time <shorter required service time advances earlier> and size of economic price for waiting <greater cost advances earlier>); and preemptive (new Arrival displaces another person/ object in the Service Facility and the former returns to the Queue before enduring to receive the service). We also state here that some consumers may remove themselves from an lively waiting line and so be "lost" to the system - if they are discontented (upset with waiting) or if they discover that the good/service is out-of-stock/unavailable.

While it may or may not be really calculable, there is at least an financial cost attributable to each person or good-in-process while remaining in the Queue. This is a realistic deliberation for a manager who is concerned with diminishing cost in the productive system.

Lastly we come to the Service Facility as treated in Queuing Theory. Availability, whether the facility is free or already in- service, is of major concern. Service Time is another serious issue, identifying that this can differ - as a single server or numerous may offer the same service to individual customers/goods-in-service or as the individual requires of exacting consumers/goods-in-service vary as they come to the server. This could be the most unbalanced variable in non-production line processes. Capacity of the Service Facility, whether of one or more stations (number of consumers/goods-in-process that can be serviced concurrently), is another measurement of the analysis. By extension we observe here that the total system - Queue(s) and Service Facility(ies) shared - may have a certain restricted capacity, which would in turn limit the number of new Arrivals to precisely matching consumers/goods-in-service exiting the Service Facility.

The practical intention of Queuing Theory is to offer examination tools for systems consisting of Queues leading to Service Facilities, in order that the systems may be prepared more competent. Queuing Theory deals mathematically with both the regularities and indiscretions of such systems - ultimately identifying incidents of congestion (resulting from indiscretions) and offering avenues for enhancing efficiency, in addition to producing particular numerical data for further application. The dimensions of congestion offered are as below:

1. The mean and distribution of time used up in a Queue;
2. The mean and distribution of consumers/goods-in-service in both the Queue itself and the whole system;
3. The mean and distribution of the Service Facility's "busy periods" (utilization/ unavailability). Each of the three distributions above can be articulated in terms of mathematical possibilities.

Queuing Theory with its fine-tuned analysis offers a base for a to some extent simplified and simpler to use set of tools called Model Building and Simulation. The practical production/ operations manager identifies that there is a transaction between queuing costs and service costs and that jointly they make-up a total cost shared by the producer and the consumer. With an eye toward reducing overall costs he/she will require an optimum mix of queuing and service costs to obtain this goal.

Simulation from adequate Models is the method that bridges the gap involving the theoretical plane of Queuing Theory and the practical tool of a Decision Support System. Simulation is the experimental laboratory for testing modifications in a productive system through the use of mathematical models - practically integrating the elements and outlines of a productive system while displaying key variables subject to experimental manipulation. Using a calculator or a computer, a investigator can insert suitable values for these key variables into the model, run the simulation and obtain achievable values for the genuine system without superseding in the system itself. With a computer the model can be run rapidly and frequently with various alterations.

Notes

Under the caption of "Characteristics of a Waiting-Line System", they spell out Arrival, Waiting-Line, and Service Facility in a pattern analogous to what has been specified above. Now I will describe four specific models as valuable POM applications:

1. ***Single Channel Queuing Model:*** which includes a single Queue and a single Service Facility and presumes that Arrivals are limitless and can be officially treated in a Poisson Distribution. Service Times are understood to take on the form of an Exponential Distribution (reflecting the normal uncertainty of service times).
2. ***Multiple-Channel Queuing Model:*** which adds a 2nd Queue intended for a 2nd Service Facility.
3. ***Constant Service Time Model:*** This shows a single Queue leading to a single Service Facility which, in turn, handles each consumer in the precise same quantity of time. Here the Exponential Distribution does not relate to Service Time.
4. ***Limited Population Model -*** which concerns with a inadequate number of Arrivals and a single server and relates to circumstances where a single operator (the Service Facility) deals with a restricted number of machines (the Queue).

We have seen that as a system gets congested, the service delay in the system increases. A good perceptive of the relationship between congestion and delay is essential for designing effective congestion control algorithms. Queuing Theory provides all the tools needed for this analysis. This article will focus on understanding the basics of this topic.

Communication Delays

Before we proceed further, lets understand the diverse components of delay in a messaging system. The total delay experienced by messages can be classified into the following categories:

Processing Delay	• This is the delay between the time of receipt of a packet for transmission to the point of putting it into the transmission queue. • On the receive end, it is the delay between the time of reception of a packet in the receive queue to the point of actual processing of the message. • This delay depends on the CPU speed and CPU load in the system.
Queuing Delay	• This is the delay between the point of entry of a packet in the transmit queue to the actual point of transmission of the message. • This delay depends on the load on the communication link.
Transmission Delay	• This is the delay between the transmission of first bit of the packet to the transmission of the last bit. • This delay depends on the speed of the communication link.
Propagation Delay	• This is the delay between the point of transmission of the last bit of the packet to the point of reception of last bit of the packet at the other end. • This delay depends on the physical characteristics of the communication link.
Retransmission Delay	• This is the delay that results when a packet is lost and has to be retransmitted. • This delay depends on the error rate on the link and the protocol used for retransmissions.

In this article we will be dealing primarily with queueing delay.

Notes

Little's Theorem

We start our analysis of queueing systems by understanding Little's Theorem. Little's theorem states that:

The average number of customers (N) can be determined from the following equation:

$$N = \lambda T$$

Here lambda is the average customer arrival rate and T is the average service time for a customer.

Proof of this theorem can be obtained from any standard textbook on queueing theory. Here we will focus on an intuitive understanding of the result. Consider the example of a restaurant where the customer arrival rate (lambda) doubles but the customers still spend the same amount of time in the restaurant (T). This will double the number of customers in the restaurant (N). By the same logic if the customer arrival rate remains the same but the customers service time doubles, this will also double the total number of customers in the restaurant.

Queueing System Classification

With Little's Theorem, we have developed some basic understanding of a queueing system. To further our understanding we will have to dig deeper into characteristics of a queueing system that impact its performance. For example, queueing requirements of a restaurant will depend upon factors like:

1. How do customers arrive in the restaurant? Are customer arrivals more during lunch and dinner time (a regular restaurant)? Or is the customer traffic more uniformly distributed (a cafe)?
2. How much time do customers spend in the restaurant? Do customers typically leave the restaurant in a fixed amount of time? Does the customer service time vary with the type of customer?
3. How many tables does the restaurant have for servicing customers?

The above three points correspond to the most important characteristics of a queueing system. They are explained below:

Arrival Process	• The probability density distribution that determines the customer arrivals in the system. • In a messaging system, this refers to the message arrival probability distribution.
Service Process	• The probability density distribution that determines the customer service times in the system. • In a messaging system, this refers to the message transmission time distribution. Since message transmission is directly proportional to the length of the message, this parameter indirectly refers to the message length distribution.
Number of Servers	• Number of servers available to service the customers. • In a messaging system, this refers to the number of links between the source and destination nodes.

Notes

Notes Queuing Theory and Simulation functions hand-in-glove to uncover and smooth out some of the uneven spots in a productive process - whether this includes delivering a service or a fabricated item to the instant consumer. When in use a gloved hand discloses only the glove; but without the hand within the glove is just an empty shell. Simulation Models only work due to the logical power of Queuing Theory which underlies and facilitates them

7.3 Computer Simulation for Queuing System

The computer simulation is a method that demonstrates dynamically the structure and the behaviors of a system with computer in order to evaluate and predict the effect of the behaviors of some system and provide information for decision. It is an effective way to solve complicated practical problems. The queuing system is the most typical problem in the discrete event system, and computer system, communication system and transportation system are all typical tangible or intangible queuing system. As a result of the widely used queuing system, the queuing character, the queuing regulation, the service organization become more and more complex so that the parsing method nearly can't be obtain. The computer simulation is a quite effective way for solve the queuing problem and analyzing the performances of the queuing system, which construct a real system model with computer program, and attain the performances and the characters changing with time through computation. With the computer simulation, the cost of the development of the system can be reduced, and the safety of the experiment and the debugging, thus it will bring great society effort and economic effort. In the queuing model, data units are often considered as customs and CPUs, transmission lines, channels and terminals are thought as queue. This is just a queuing model. Because input, computation, transmission, storage and output are discrete in time field, it is also called discrete queuing model. In real life, waiting in queue is a common phenomenon, which makes people inconvenient .This paper takes the payment of minitype supermarket as an example to discuss the computer simulation for the single-channel and multi-channel queuing system.

Task Analyze an instance of Computer Simulation for Queuing System

Some Conceptions in the Queuing System

The queuing problem is a problem about a balance between average waiting time and idle time of server, i.e. how to queue to be both good for entity and server.

The queuing theory is a science to solve the problem mentioned as above, and it is also named random serve theory since the arrival time of entities and the time of serve acceptance is usually random variable obeying some probability distribution. In the simulation of the queuing system, there are some conceptions usually being used as below:

1. ***Entity Arrival Mode:*** The entity is limited or limitless, and the arrival of the entity is in individual or in batch. Entity arrival mode is often described with arrival interval. The random arrival mode applied in the system appears very complex, and different probability distributions have to be adopted for different systems. Index distribution, normal

Notes

distribution, Poisson distribution and etc are quite common. Poisson distribution arrival is provided as below:

In (t, t+s), the probability of entity number k is:

$$P\{N(t+s)-N(t)=k\}=\frac{e^{-s}(s)^k}{k!}$$

In the formula, N(t) is the number of entity arrival in (0,t). t >=0, s>=0, k=0, 1, 2, λ is the arrival velocity. If the entity arrival satisfies steady Poisson distribution, arrival interval will obey Index distribution and density function is:

$$f(t)=\lambda e^{-\lambda t}=\frac{1}{\beta}e^{-1/\beta}, t\geq 0\cdot\beta=\frac{1}{\lambda}$$

average value of arrival interval.

2. ***Service Mode:*** Its character is that its server may be single or multiple, and service time distribution is nothing about time or something about time, and server's service time is certain or random. Random service time is described with probability distribution, for instance Normal distribution,

$$f(t)=\frac{1}{\sqrt{2\pi\sigma}}e-\frac{(x-\mu)^2}{2\sigma^2}, -\infty<\mu<\infty, \sigma>0$$

In the equation above, t is the time of server for each custom, which is obeying Normal distribution, and the average μ, the variance is σ^2.

3. ***Queuing Rule and the Criteria of the Queuing System:*** There are some queuing rules such as FCFS, Random served, priority served and SCFS, etc. With studying the performances of the queuing system, some criteria usually used are as below:

(a) Steady-state means delaying time d:

$$d=\lim_{n\to\infty}\sum_{i=1}^{n}D_i/n$$

In the equation, D_i means NO.i entity's delaying time, i.e. waiting time in the queue; n is the number of the accepted entities; d is the mean time of waiting time of the n entities; D_i the staying time of the entity in the system w:

$$w=\lim_{n\to\infty}\sum_{i=1}^{n}w_i/n=\lim_{n\to\infty}\sum_{i=1}^{n}(D_i+S_i)/n$$

In the equation as above, W_i is the staying time of No.i entity in the system, and equals to the sum of waiting time in the queue D_i and accepting service time S_i.

(b) Steady-state means step-length Q:

$$Q=\lim_{T\to\infty}\int_0^T Q(t)dt/T$$

In the formula, Q(t) is the length of the queue at t, and T is the simulate time of the system.

(c) Steady-state entity mean number L:

$$L=\lim_{T\to\infty}\int_0^T L(t)dt/T=\lim_{T\to\infty}\int_0^T Q(t)+S(t)dt/T$$

In the formula, L(t) is the number of the entity in the system at t, and equals to Q(t) and S(t).

Notes

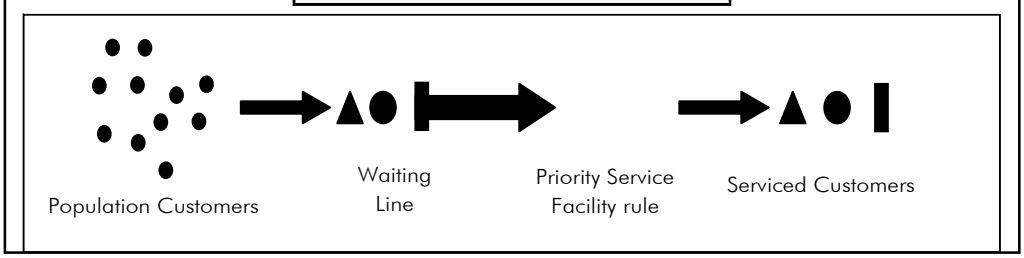
Task Differentiate between waiting time and idle time of the server.

Single-server M/M/1 Model

Construction of Model

In the minitype super-market, there is a cash desk, and customers reaches the desk in random. Provided that the customer reaches as the cashier is idle, the customer will pay off immediately and leave. If the cashier is busy as the customer reaches, the customer will have to wait in the line, nay, no person leaves without waiting. Once the customer enters in the queue, he will receive service according to FCFS rule. The customer departs after receiving once service. The interval of customers arriving desk obeys negative index distribution with average value equaling to 5, and service time of each customer complies with normal distribution with average value being 1.6 and standard deviation being 0.6. Time calculates at minute, and service time must be positive.

Figure 7.7: Single-server M/M/1 Model



Simulation of Model

1. The creation of random number. It is desired to describe random factors in the objective process in nearly all of the simulation process like arrival process and service process in actual system. Random number comes from collectivity in random. In this model, there are the interval of customers' arrival and the service time of each customer. The former obeys negative index distribution with average number being 2.5, and the latter obeys normal distribution with average value being 1.6 and standard deviation being 0.6. The symmetrical-distribution random number U(0,1) must create ahead of the creation of specific-distribution random number.

   (a) Creating the algorithm of obeying negative index distribution with Inversion of Transforms method the density function of negative index distribution:

   $$f(x) = \lambda e^{-\lambda x}, x \geq 0 \times E(X) = 1/\lambda$$

   Its distribution

   Function:

   $$F(X) = \int_0^x \lambda e^{-\lambda t} dt = 1 - e^{-\lambda x}, x \geq 0, \text{i.e.},$$

   $$R = F(X) = 1 - e^{-\lambda x}$$

   Through inversion transform, we can obtain:

   $X = -1/\lambda \ \ln(1-R)$ Let $u=1-R$, thus u is a random number in (0,1) (R obeys symmetrical distribution in (0,1))

Notes

Therefore $x=-1/\lambda \ln u$. x is a random number obeying negative index distribution.

With this method, as long as producing a random number obeying even distribution, we could get a random number x obeying the negative index number distribution with parameter λ.

(b) Creating the algorithm of obeying normal distribution with discarding-selecting method

(i) Create two random number R_1 and R_2 in (0, 1)

(ii) With R_1, create a random number in [a,b]: $x=a+(b-a)R_1$, and calculate the value of f(x)

(iii) when $R_2<=f(x)/M$, select x as, and M is the max value of f(X)

Simulating the model with event step-length method. In some degree, the process of the simulation is considered as a series of event point. In terms of the sequence of event, the system arranges the sequence of event executing with a table called "event table". Event table has three properties, ID, event style and time.

The simulation algorithm as below:

Step 1: Initiation (clear simulation clock

Configuring the state of the system

Clear accumulation statistic

creating initiating event table)

Step 2: Do

```
{
Find most recent event from event table
Simulation clock increase
If (Customers arrive)
 Add the time of next customer arriving into event table
If (cashier is busy/ S=1)
 LQ++ //queue increment
Else
 S=1 //cashier turn to busy
 Add the time of the customer finishing into event
 Table
Else
 Delete the customer from event table //service ends
If (the queue is not empty/ LQ>0)
 LQ— //the length of queue decrease
 Add the time of the customer finishing into event table
Else
 S=0 // cashier is idle
} While (event table is not null)
Output result
```

Notes

2. Ascertaining the times of repeating simulation for reaching the precision β

   (a) Repeat simulation R_0 ($R_0 \geq 2$) times independently, and set $R = R_0$

   (b) Compute $\overline{Y}(R), S^2(R)$ and absolute procession θ (Ora). $\overline{Y}(R), S^2(R)$ Are sample average value and variance after R times running, and θ (Ora), is half length of the Confidence Interval of R times running under significance degree.

Did u know? **Event step-length method**

Event step-length method takes the time of event as increment, and simulates the behaviors of the system according to the process of time until the scheduled time ends.

Caselet

The Value of Queuing Theory

Professor Jack Byrd [Byrd, J. 1978. The value of queueing theory. Interfaces 8 (3) 22-26.], argues that queueing theory has little value to the practicing professional. Professor Byrd bases his argument on a number of examples in which queueing theory, as it exists today, appears to be inadequate to provide answers to the questions being posed. The purpose of this article is to offer a rebuttal to the conclusions of Byrd's paper. This rebuttal is in the form of a case study dealing with a command and control problem. This case study is of general interest because there is a fundamental difference between the question addressed in the case study discussed below and the questions addressed by the examples in Byrd's paper.

7.4 Summary

- This unit has evaluated the performance of single-channel and multiple-channels queues using the discrete-event simulation technique.
- In a queuing system, the calling population is assumed to be infinite that is, if a unit leaves the calling population and joins the waiting line or enters service; there is no change in the arrival rate.
- The single-channel queuing system can be seen in places such as banks and post offices, where one single queue will diverge into a few counters.
- The multiple-channels queuing system is commonly observed in fast food restaurants like KFC, Burger King, McDonalds, etc. It is a system whereby the customers line up in rows directly in front of each server.
- Queuing Theory arises from the use of powerful mathematical analysis to theoretically describe production processes along with statistical/probabilistic techniques to account for varying dynamic patterns within the stages of a productive process.
- The computer simulation is a method that demonstrates dynamically the structure and the behaviors of a system with computer in order to evaluate and predict the effect of the behaviors of some system and provide information for decision.

Notes

7.5 Keywords

Computer Simulation: The computer simulation is a method that demonstrates dynamically the structure and the behaviors of a system.

Discrete Event: Discrete event is a technique used to model the real-world scenarios.

7.6 Self Assessment

Fill in the blanks:

1. The arrivals occur one at a time in a random order and once the customer joins the …… he will eventually receive the service.
2. The arrival pattern of consumers to the service system is categorized into two groups: …… and dynamic.
3. The …… queuing system can be seen in places such as banks and post offices, where one single queue will diverge into a few counters.
4. The purpose of using simulation technique to analyze the collected data is to avoid costly design errors, and to analyze the behaviors of the …… systems.
5. Each event has a …… corresponding to the wall-clock time when it occurs.
6. A good perceptive of the relationship between …… and delay is essential for designing effective congestion control algorithms.
7. Queuing Theory occurs from the use of powerful …… analysis.
8. Queuing Theory with its fine-tuned analysis offers a base for a to some extent simplified and simpler to use set of tools called …… and Simulation.
9. The behaviors of a system with computer in order to …… and predict the effect of the behaviors of some system and provide information for decision.
10. The computer simulation is a quite effective way for solve the …… and analyzing the performances of the queuing system.
11. In the …… super-market, there is a cash desk, and customers reaches the desk in random.
12. The creation of random number it is desired to describe …… factors in the objective process in nearly all of the simulation process.
13. The entity is limited or limitless, and the arrival of the entity is in individual or in ……
14. The queuing problem is a problem about a balance between average …… and idle time of server.
15. The queuing character, the …… the service organization become more and more complex so that the parsing method nearly can't be obtain.

7.7 Review Questions

1. Find out the difference between single-channel and multiple-channels queuing systems.
2. “In the queuing model two types of events are used, namely arrival and departure.” Elaborate.

Notes

3. "Discrete-event technique has been widely used in the simulation of communication and transportation systems." Comment.
4. Do you think Queuing Theory provides all the tools needed for this analysis?
5. Simulation is the experimental laboratory for testing modifications in a productive system through the use of mathematical models. Give reasons.
6. Describe four specific models as valuable POM applications.
7. "The queuing system is the most typical problem in the discrete event system." Explain.
8. Discuss the main conceptions in the Queuing System.
9. Simulating the model with event step-length method. Discuss.
10. Why queuing theory is a science to solve the problem?

Answers: Self Assessment

1.	queuing system	2.	static
3.	single-channel	4.	existing
5.	timestamp	6.	congestion
7.	mathematical	8.	Model Building
9.	Evaluate	10.	queuing problem
11.	minitype	12.	random
13.	batch	14.	waiting time
15.	queuing regulation		

7.8 Further Readings

Books

Balci, O.,(1994), Validation, verification, and testing techniques throughout the life cycle of a simulation study, Annals of Operations Research, 53: 121–173.

Balci, O., (2001), A methodology for certification of modelling and simulation applications, ACM Transactions on Modelling and Computer Simulation, 11: 352–377.

Birta, L.G. and Ozmizrak, N.F., (1996), A knowledge-based approach for the validation of simulation models: The foundation, ACM Transactions on Modelling and Computer Simulation, 6: 67–98.

Boehm, B.W., (1979), Software engineering: R&D trends and defense needs, in: P. Wegner (Ed.), Research Directions in Software Technology, MIT Press, and Cambridge, MA.

Cellier, F.E., (1986), Combined discrete/continuous system simulation – application, techniques and tools, in Proceedings of the 1986 Winter Simulation Conference.

Department of Defense (DoD) Recommended Practices Guide (RPG) for Modelling and Simulation VV&A, Millennium Edition (available at http://vva.dmso.mil).

Notes

General Accounting Office, (1976), Report to the Congress: Ways to improve management of federally funded computerized models, report LCD-75-111, U.S. General Accounting Office, Washington, DC.

General Accounting Office, (1979), Guidelines for model evaluation, report PAD-79-17, U.S. General Accounting Office, Washington, DC.

General Accounting Office, (1987), DOD simulations: Improved assessment procedures would increase the credibility of results, report GAO/PEMD-88-3, U.S. General Accounting Office, and Washington, DC.

Ören, T.I., (1981), Concepts and criteria to access acceptability of simulation studies, Communications of the ACM, 24: 180–189.

Online links

http://www.math.wsu.edu/faculty/genz/416/lect/l06-34.pdf

http://home.iitk.ac.in/~skb/ee679/ee679.html

http://www.slideshare.net/avtarsingh/queuing-theory-2129896

Unit 8: Simulation of Queuing System (II)

Notes

CONTENTS

Objectives

Introduction

8.1 Simulation of Single and Two Server Queue

8.1.1 Single-server Queue

8.1.2 Two-server Queue Simulation

8.2 Summary

8.3 Keywords

8.4 Self Assessment

8.5 Review Questions

8.6 Further Readings

Objectives

After studying this unit, you will be able to:

- Understand simulation of single server queue
- Discuss simulation of two server queue

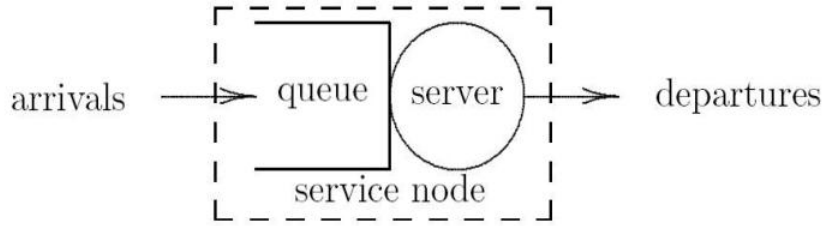
Introduction

Single-server queues are, perhaps, the most commonly encountered queuing situation in real life. One encounters a queue with a single server in many situations, including business (e.g. sales clerk). Multiple (identical)-servers queue situations are frequently encountered in telecommunications or a customer service environment.

8.1 Simulation of Single and Two Server Queue

8.1.1 Single-server Queue

Figure 8.1: Single-server Queue



Notes

A Single-sever service node consists of a server plus its queue.

Queue discipline: The algorithm used when a job is selected from the queue to enter service

FIFO – first in, first out

LIFO – last in, first out

SIRO – serve in random order

Priority – typically shortest job first (SJF)

FIFO is also known as first come, first serve (FCFS)

1. The order of arrival and departure are the same
2. This observation can be used to simplify the simulation

Service is non-preemptive

Once initiated, service of a job will continue until completion

Service is conservative

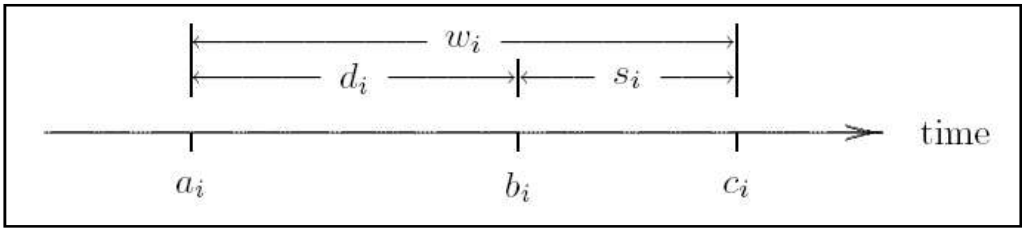
Server will never remain idle if there is one or more jobs in the service node

Caution Unless otherwise specified, assume FIFO with infinite queue capacity.

Specification Model

For a job i:

1. The arrival time is a_i
2. The delay in the queue is d_i
3. The time that service begins is $b_i = a_i + d_i$
4. The service time is s_i
5. The wait in the node is $w_i = d_i + s_i$
6. The departure time is $c_i = a_i + w_i$



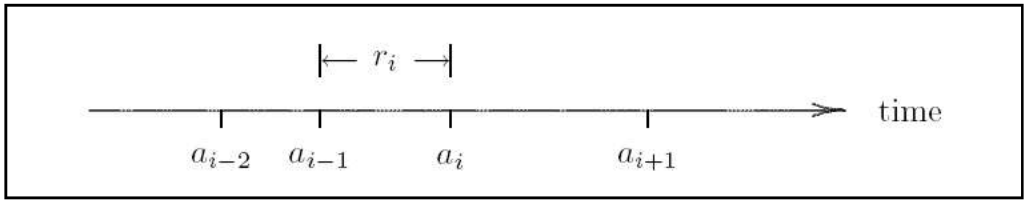
Arrivals

1. The interarrival time between jobs i "– 1 and i is

$$r_i = a_i - a_{i-1}$$

where, by definition, $a_0 = 0$

Notes



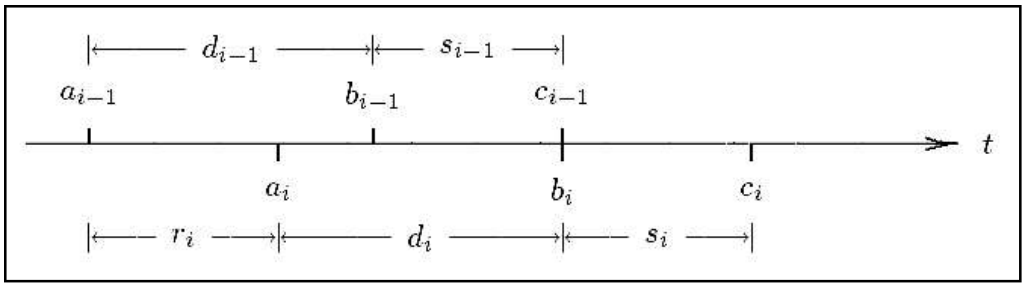
Note that $a_i = a_{i-1} + r_i$ and so (by induction)

$a_i = r_1 + r_2 + ... + r_i$ $\quad$ i = 1, 2, 3, ...

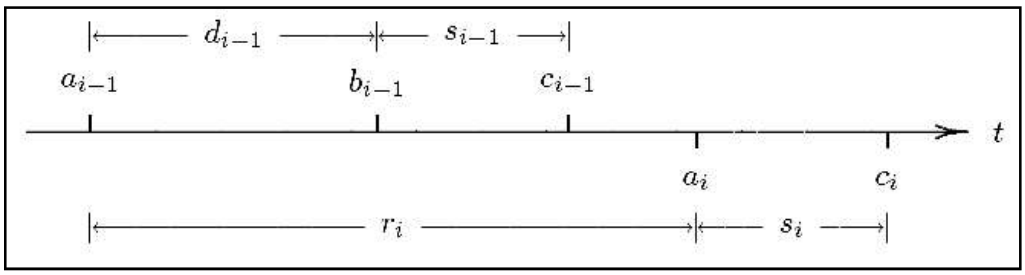
2. Given the arrival times and service times, can the delay times be computed?
3. For some queue disciplines, this question is difficult to answer
4. If the queue discipline is FIFO,
   a. d_i is determined by when a_i occurs relative to c_{i-1}.

Cases

If $a_i < c_{i-1}$, job i arrives before job i–1 completes:



If $a_i < c_{i-1}$, job i arrives after job i –1 completes:



Calculating Delay for Each Job

Algorithm 1.2.1

```
c0 = 0.0; /* assumes that a0 = 0.0 */
i = 0;
While (more jobs to process)
 {
 i++;
 ai = GetArrival();
```

Notes

```
 if (a_i < c_i-1)
 d_i = c_i-1-a_i;
 else
 d_i = 0.0;
s_i = GetService();
c_i = a_i + d_i + s_i;
}
n = i ;
return d_1, d_2, . . . , d_n;
```

Example: Algorithm 1.2.1 used to process n = 10 jobs

	i	1	2	3	4	5	6	7	8	9	10
read from file	a_i	15	47	71	111	123	152	166	226	310	320
from algorithm	d_i	0	11	23	17	35	44	70	41	0	26
read from file	s_i	43	36	34	30	38	40	31	29	36	30

For future reference, note that for the last job

$a_i = 320$

cn = an + dn + sn = 320 + 26 + 30 = 376

Output Statistics

1. The purpose of simulation is insight — gained by looking at statistics
2. The importance of various statistics varies on perspective:
   (a) Job perspective: wait time is most important
   (b) Manager perspective: utilization is critical
3. Statistics are broken down into two categories
   (a) Job-averaged statistics
   (b) Time-averaged statistics

Job-averaged Statistics

Job-averaged statistics: computed via typical arithmetic mean

Average interarrival time:

$$\bar{r} = \frac{1}{n}\sum_{i=1}^{n} r_i = \frac{a_n}{n}$$

Where $1/\bar{r}$ is the arrival rate.

Average service time:

$$\bar{s} = \frac{1}{n}\sum_{i=1}^{n} s_i$$

Where $1/\bar{s}$ is the service rate.

Notes

Example: For the 10 jobs in Example 1

1. Average interarrival time is $\bar{r} = a_n/n = 320/10 = 32.0$ seconds per job
2. Average service is s = 34.7 seconds per job
3. Arrival rate is 1/r 1/4 0.031 jobs per second
4. Service rate is 1/s 1/4¼ 0.029 jobs per second

   The server is not quite able to process jobs at the rate they arrive on average.

The average delay and average wait are defined as:

$$\bar{d} = \frac{1}{n}\sum_{i=1}^{n} d_i \qquad \bar{w} = \frac{1}{n}\sum_{i=1}^{n} w_i$$

Recall $w_i = d_i + s_i$ for all i

$$\bar{w} = \frac{1}{n}\sum_{i=1}^{n} w_i = \frac{1}{n}\sum_{i=1}^{n}(d_i + s_i) = \frac{1}{n}\sum_{i=1}^{n} d_i + \frac{1}{n}\sum_{i=1}^{n} s_i = \bar{d} + \bar{s}$$

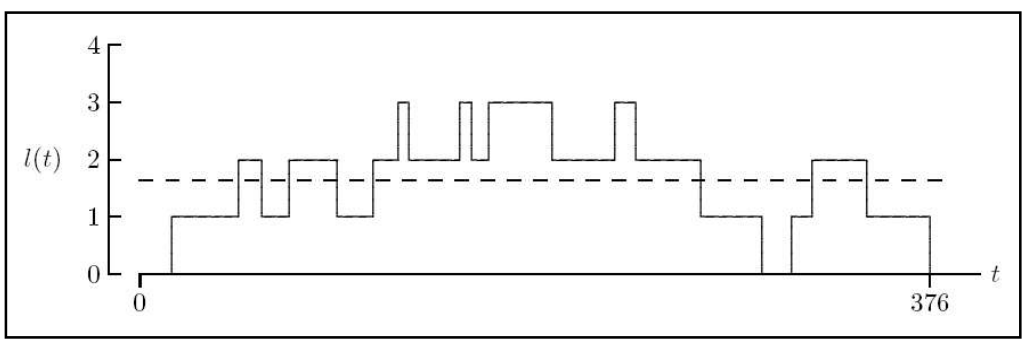
Sufficient to compute any two of w, d, s.

Time-averaged Statistics

1. Time-averaged statistics: defined by area under a curve (integration)
2. For SSQ, need three additional functions
   (a) l(t): number of jobs in the service node at time t
   (b) q(t): number of jobs in the queue at time t
   (c) x(t): number of jobs in service at time t
3. By definition, l(t) = q(t) + x(t).
4. l(t) = 0, 1, 2, . . .

   q(t) = 0, 1, 2, . . .

   x(t) = 0, 1
5. All three functions are piece-wise constant



Notes

Figures for q(•) and x(•) can be deduced

q(t) = 0 and x(t) = 0 if and only if l(t) = 0

Over the time interval (0, τ):

Time-averaged number in the node:

$$\overline{l} = \frac{1}{\tau}\int_0^{\tau} l(t)dt$$

Time-averaged number in the queue:

$$\overline{q} = \frac{1}{\tau}\int_0^{\tau} q(t)dt$$

Time-averaged number in service:

$$\overline{x} = \frac{1}{\tau}\int_0^{\tau} x(t)dt$$

Since l(t) = q(t) + x(t) for all t > 0

$$\overline{x} \approx 0.28$$

Sufficient to calculate any two of $\overline{l}, \overline{q}, \overline{x}$.

Example

1. From Example 1 (with τ = c10 = 376),

   l = 1.633 q = 0.710 x = 0.923

2. The average of numerous random observations (samples) of the number in the service node should be close to l.

   (a) Same holds for q and x

3. Server utilization: time-averaged number in service (x)

4. x also represents the probability the server is busy

Little's Theorem

How are job-averaged and time-average statistics related?

If (a) queue discipline is FIFO,

(b) service node capacity is infinite, and

(c) server is idle both at t = 0 and t = c_n

Then

$$\int_0^{c_n} l(t)dt = \sum_{i=1}^{n} w_i \quad \text{and}$$

$$\int_0^{c_n} q(t)dt = \sum_{i=1}^{n} d_i \quad \text{and}$$

$$\int_0^{c_n} x(t)dt = \sum_{i=1}^{n} s_i$$

Proof

Notes

For each job i = 1, 2, . . ., define an indicator function

$$\psi_i(t) = \begin{cases} 1 & a_i < t < c_i \\ 0 & \text{otherwise} \end{cases}$$

Then

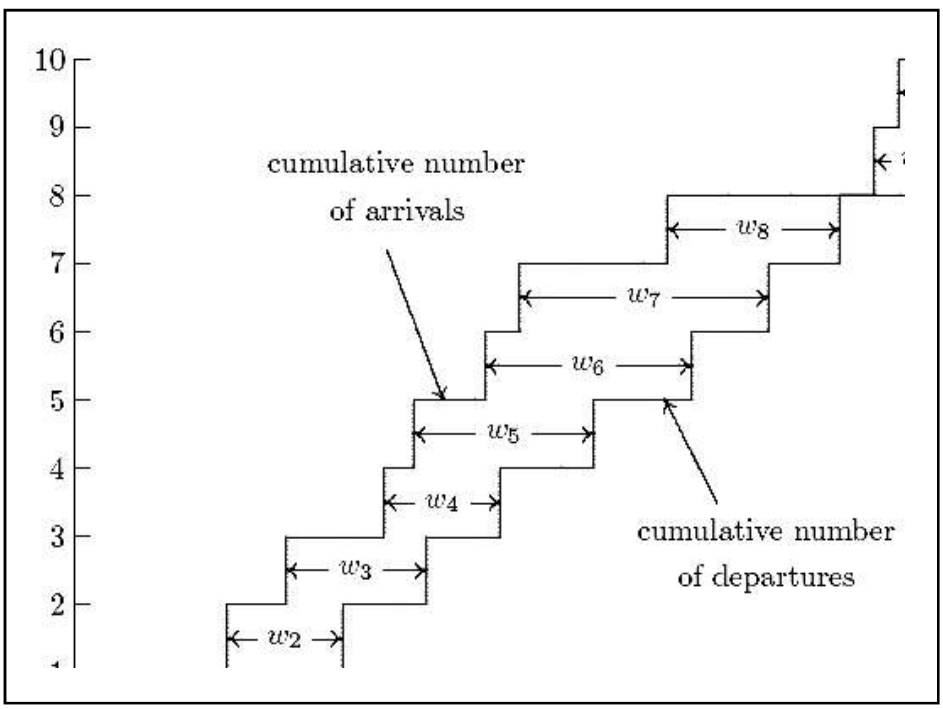
$$l(t)\sum_{i=1}^{n}\psi_i(t) \qquad 0 < t < c_n$$

and so

$$\int_0^{c_n} l(t)dt = \int_0^{c_n}\sum_{i=1}^{n}\psi_i(t)dt = \sum_{i=1}^{n}\int_0^{c_n}\psi_i(t)dt = \sum_{i=1}^{n}(c_i - a_i) = \sum_{i=1}^{n} w_i$$

The other two equations can be derived similarly.

Example



$$\int_0^{376} l(t)dt = \sum_{i=1}^{10} w_i = 614$$

Notes

Little's Equation

Using $\tau = c_n$ in the definition of the time-averaged statistics, along with Little's Theorem, we have

$$c_n \bar{l} \int_0^{c_n} l(t)dt = \sum_{i=1}^{n} w_i = n\bar{w}$$

We can perform similar operations and ultimately have

$$\bar{l} = \left(\frac{n}{c_n}\right)\bar{w} \text{ and } \bar{q} = \left(\frac{n}{c_n}\right)\bar{d} \text{ and } \bar{x} = \left(\frac{n}{c_n}\right)\bar{s}$$

Computational Model

1. The ANSI C program ssq1 implements Algorithm 1.2.1
2. Data is read from the file ssq1.dat consisting of arrival times and service times in the format

a_1	S_1
a_2	S_2
.	.
.	.
.	.
a_n	S_n

Since queue discipline is FIFO, no need for a queue data structure

Example

Running program ssq1 with ssq1.dat

$$1/\bar{r} \approx 0.10 \text{ and } 1/\bar{s} \approx 0.14$$

If you modify program ssq1 to compute l , q, and x

$$\bar{x} \approx 0.28$$

Despite the significant idle time, q is nearly 2.

8.1.2 Two-server Queue Simulation

Two Servers in Series

1. ***Hypothesis:*** Nonhomogeneous $\lambda(t)$ Poisson arrivals; service at server 1, then by server 2 service for each consumer; service times are RVs with distribution G_1 and G_2; no consumers after final arrival time T.
2. ***Instances:*** Airline checkin, doctor's office, restaurant.
3. ***Variables:*** Time t; counters N_A, N_B; system state SS = (n_1, n_2) = customer #s at server 1,2; output $(A_1(i),A_2(i),D(i))$ customer *i* arrival (for each server) and departure times; event list EL = (t_A, t_1, t_2) next arrival and finishing point times.

Notes

1. ***Two Servers in Series Simulation Algorithm:***

   Initialize: $t = N_A = N_D = 0$, $SS = (n_1, n_2) = (0, 0)$; generate T_0, *set* $t_A = T_0$, $t_1 = t_2 = \infty$.

   Update the system state by means of the subsequent cases:

   (a) $t_A = \min(t_A, t_1, t_2)$ (a new arrival at time t_A)
   reset $t = t_A$, $N_A = N_A + 1$, $n_1 = n_1 + 1$;
   generate T_t and reset $t_A = T_t$ (next arrival time);
   if $n_1 = 1$ generate $Y_1 \sim G_1$ and reset $t_1 = t + Y_1$;
   collect output data $A_1(N_A) = t$.

   (b) $t_1 < t_A$, $t_1 \le t_2$ (a departure from 1 at time t_1)
   reset $t = t_1$, $n_1 = n_1 - 1$, ; $n_2 = n_2 + 1$;
   if $n_1 > 0$ generate $Y_1 \sim G_1$ and reset $t_1 = t + Y_1$;
   otherwise set $t_1 = \infty$ (queue 1 is empty);
   if $n_2 = 1$ generate $Y_2 \sim G_2$ and reset $t_2 = t + Y_2$;
   collect output data $A_2(N_A - n_1) = t$.

   (c) $t_2 < t_A$, $t_2 < t_1$ (a departure from 2 at time t_2)
   reset $t = t_2$, $n_2 = n_2 - 1$, $N_D = N_D + 1$;
   if $n_2 > 0$ generate $Y_2 \sim G_2$ and reset $t_2 = t + Y_2$;
   otherwise set $t_2 = \infty$ (queue 2 is empty);
   collect output data $D(N_D) = t$.
   Cases 1, 2, and 3 are used until $t_A > T$.
   Then cases 2 and 3 are used until $n_1 = 0$.
   Then case 3 is used until $n_2 = 0$.
   Then $T_p = \max(t - T, 0)$.

2. End of "run" results

   (a) Times $(A_1(1),A_2(1),D(1)), \ldots, (A_1(N_A),A_2(N_A),D(N_A))$ provide $(A_2 - A_1)$, $(D - A_2)$, $(D - A_1)$ averages.

   (b) Time T_p is server overtime.

   (c) (Event,Time) data $(n_1(t_i), n_2(t_i), t_i)$ offers history.

3. Averages over many runs provide expected service time, server overtime, and other statistics.

4. Generalizations to numerous servers in series.

5. Example Matlab Function for Two Servers in Series function $[A_1\ A_2\ D\ T_p\ E_v]$ = sriesv(T,lam, l_1, l_2)

```
Sample Runs
for i =1:100
[A1, A2, D, Tp, Ev] = sriesv(9,6,4,3);
DA(i) = mean(D-A1);
end, disp(mean(DA))
6.1033
```

```
[A1, A2, D, Tp, Ev] = sriesv(.5,6,4,3); disp(Ev)

1     0     0.006042
0     1     0.082711
0     0      0.14119
1     0     0.15819
2     0     0.23596
1     1     0.26024
```

Notes

1	0	0.33996
0	1	1.0999
0	0	1.9058

```
function [A1 A2 D Tp Ev] = sriesv( T, lam, l1, l2 )
% Two Servers in Series Q Simulation
t = 0; na = 0; nd = 0; n1 = 0; n2 = 0; j = 0;
ta = -log(rand)/lam; t1 = inf; t2 = inf;
while ta <= T % time left
if ta <= min( t1, t2 ) % new arrival
t = ta; n1 = n1 + 1; na = na + 1;
ta = t - log(rand)/lam; A1(na) = t;
if n1 == 1, t1 = t + G(l1); end
elseif t1 <= t2 % departure from Q1
t = t1; n1 = n1 - 1; n2 = n2 + 1; A2(na-n1) = t;
if n1 > 0, t1 = t + G(l1); else, t1 = inf; end
if n2 == 1, t2 = t + G(l2); end
else % departure from Q2
t = t2; n2 = n2 - 1; nd = nd + 1; D(nd) = t;
if n2 > 0, t2 = t + G(l2); else, t2 = inf; end
end, j = j + 1; Ev(j,:) = [ n1 n2 t ];
end % no more arrivals
while n1 > 0 % empty Q1
if t1 <= t2, t = t1; n1 = n1 - 1; n2 = n2 + 1;
if n1 > 0, t1 = t + G(l1); else, t1 = inf; end
if n2 == 1, t2 = t + G(l2); end, A2(na-n1) = t;
else, t = t2; n2 = n2 - 1; nd = nd + 1; D(nd) = t;
if n2 > 0, t2 = t + G(l2); else, t2 = inf; end
end, j = j + 1; Ev(j,:) = [ n1 n2 t ];
end % Q1 is empty
while n2 > 0 % empty Q2
t = t2; n2 = n2 - 1; nd = nd + 1; D(nd) = t;
if n2 > 0, t2 = t + G(l2); else, t2 = inf; end
j = j + 1; Ev(j,:) = [ n1 n2 t ];
end, Tp = max(t-T,0); % Q2 is empty, find Tp
% end sriesv
function Y = G(a), Y = -log(rand)/a; % Exponential(a) RV
```

Two Servers in Parallel

1. *Hypothesis:* nonhomogeneous $\lambda(t)$ Poisson arrivals;
   arrivals form single queue, service at server 1 or 2 as obtainable;
   service times are RVs with distribution G_1 and G_2;
   no consumers after final arrival time T.
2. *Instances:* airline checkin, doctor's office, restaurant.
3. *Variables:* time t;
   counters N_A, (C_1,C_2) #s of customers served;
   system state SS = (n, i_1, i_2) = customers in system, with customer # i_j at server j;
   output (A(i),D(i)) customer i arrival and departure times;
   event list EL = (t_A, t_1, t_2) next arrival and completion times.

Notes

Notes If n customers are in the system, and j = max(i_1, i_2), the customer at the head of the queue has # j + 1.

4. ***Two Servers in Parallel Simulation Algorithm:***
   Initialize: $t = N_A = N_D = 0$, SS = (n, i_1, i_2) = (0, 0, 0);
   generate T_0, set $t_A = T_0$, $t_1 = t_2 = \infty$.
   Update the system state using the following cases:

   (a) $t_A = \min(t_A, t_1, t_2)$ (a new arrival at time t_A)
   reset $t = t_A$, $N_A = N_A + 1$;
   generate T_t and reset $t_A = T_t$ (next arrival time);
   collect output data $A(N_A) = t$.
   if SS = (0, 0, 0), reset SS = (1,N_A, 0) (use server 1),
   generate $Y_1 \sim G_1$ and reset $t_1 = t + Y_1$;
   else if SS = (1, j, 0), reset SS = (2, j,N_A) (use server 2),
   generate $Y_2 \sim G_2$ and reset $t_2 = t + Y_2$;
   else if SS = (1, 0, j), reset SS = (2,N_A, j) (use server 1),
   generate $Y_1 \sim G_1$ and reset $t_1 = t + Y_1$;
   else reset SS = (n + 1, i_1, i_2).
   Two Servers Parallel Simulation Algorithm Cases a and b

   (b) $t_1 \le t_A$, $t_1 \le t_2$ (a departure from 1 at time t_1) reset $t = t_1$, $C_1 = C_1 + 1$; collect output data $D(i_1) = t$;
   if n = 1 reset SS = (0, 0, 0), $t_1 = \infty$;
   else if n = 2 reset SS = (1, 0, i_2), $t_1 = \infty$;
   else reset SS = (n — 1, max(i_1, i_2) + 1, i_2), and generate $Y_1 \sim G_1$ and reset $t_1 = t + Y_1$.

   (c) $t_2 < t_A$, $t_2 < t_1$ (a departure from 2 at time t_2)
   reset $t = t_2$, $C_2 = C_2 + 1$; collect output data $D(i_2) = t$;
   if n = 1 reset SS = (0, 0, 0), $t_2 = \infty$;
   else if n = 2 reset SS = (1, i_1, 0), $t_2 = \infty$;
   else reset SS = (n — 1, i_1, max(i_1, i_2) + 1), and generate $Y_2 \sim G_2$ and reset $t_2 = t + Y_2$.
   Cases 1, 2, and 3 are used until $t_A > T$.
   Then cases 2 and 3 are used until n = 0,
   Then $T_p = \max(t - T, 0)$.

5. End of "run" results

   (a) Times (A(1),D(1)), . . . , ($A(N_A)$,$D(N_A)$); (D — A) average provides average service time.

   (b) Server overtime T_p, customers served C_1, C_2.

   (c) (Event,Time) data ($n(t_j)$, $i_1(t_j)$, $i_2(t_j)$, t_j) provides history.

6. Averages over numerous runs provide expected service time, server overtime, and other statistics.

7. Simplification to numerous servers in parallel and grouping with series servers.

Tasks Differentiate between two servers in series and two servers in parallel.

Notes

Caselet

LP Model in a Paint Company

FENTON Paints Ltd manufactures three grades of paints — Venus, Diana and Aurota. The plant operates on a three-shift basis and the data presented in Table 1 are made available from the production records.

Table 1

Requirement of resource	Grade			Availability (capacity per month)
	Venus	Diana	Aurota	
Special Additive (kgs per litre)	0.30	0.25	0.75	650 tonnes
Milling (kilolitres per machine shift)	2.50	3.50	5.00	110 machine shifts
Packing (kilo litres per shift)	12.00	12.00	12.00	100 shifts

There are no limitations on other resources. The particulars of sale forecasts and estimated contribution to overheads and profits are given in Table 2.

Table 2

	Venus	Diana	Aurota
Maximum possible sales per month (kilo litres)	120	450	600
Contribution (Rs/kl)	4,000	3,600	2,500

Due to commitments already made, a minimum of 200 kilolitres per month of Aurota has to be necessarily supplied the next year. Just as the company was able to finalise the monthly production programme for the next 12 months, an offer was received from a nearby competitor for hiring 50 machine shifts per month of milling capacity for grinding Diana paint, that can be spared for at least a year.

However, due to additional handling and the profit margin of the competitor involved, by using this facility, the contribution from Diana will get reduced by ₹ 1.50 per litre.

Formulate this problem as a linear programming (LP) model for determining the monthly production programme to maximise contribution. You are not required to solve the LP model.

Let x litres of Venus grade paint, y litres of Diana grade paint and

Table 3

	Venus	Diana	Aurota
Sale quantity (kl)	120	450	600
Contribution (Rs/kl)	4,000	3,600	2,500

z litres of Aurota grade paint be manufactured as shown in Table 3.

Maximisation of contribution:

Maximise z = 4x + (3.6 - 1.5)y + 2.5z

z = 4x + 2.1y + 2.5z

Contd...

Notes

Subject to constraints

Special additive: 0.30 x + 0.25 y + 0.75 z less than or equal to 6,50,000

Milling x/2500 + y/ 3500 + z/5000 less than or equal to 110 - 55

x/2500 x 35000 + y/3500 x 35000 + z/5000 x 35000 less than or equal to 55 x 35,000

14x + 10 y + 7z less than or equal to 19,25,000

Packing: x/12000 + y/12000 + z/12000 less than or equal to 100

x + y + z =100 x 12,000 = 12,00,000

0.30x + 0.25y + 0.75z less than or equal to 6,50,000

14x +10y + 7z less than or equal to 19,25,000

x + y + z less than or equal to 12,00,000

Subject to x being less than or equal to 1,20,000

Y being less than or equal to 4,50,000

2,00,000 being less than or equal to and z being less than or equal to 6,00,000. (2,00,000 is prior commitment)

Right Option

IF THE profit-volume (PV) ratio is 40 per cent and sales value ₹ 10,000, the variable cost will be: a) ₹ 4,000; b) ₹ 6,000; c) ₹ 5,000; d) none of the above.

Variable cost ratio is a complement of PV ratio. When PV ratio is 40 per cent, variable cost ratio is 60 per cent of sales. Sixty per cent of ₹ 10,000 will be ₹ 6,000. Hence option (b) is correct.

In a service department manned by one server, on an average one customer arrives every five minutes, the service time is four minutes per customer. The probability of the server being idle is: a) 40 per cent; b) 20 per cent; c) 15 per cent; d) none of the above.

S = 60/4 = 15 per hour.

R = 12/15, that is, 4/5

A = 60/5 =12 per hour.

The probability of server being idle is 1 - R, that is, 1 - 4/5 = 1/5, that is, 20 per cent. Hence, option (b) is correct.

A company budgets for fixed overhead of ₹ 24,000 and production of 4,800 units. Actual production is 4,200 units and fixed overhead cost incurred is ₹ 22,000. The fixed overhead volume variance is: a) ₹ 3,000 (adverse); ₹ 1,000 (adverse); ₹ 2,000 (favourable); ₹ 3,000(favourable)

SR = BFO/BQ, that is, ₹ 24,000 / 4,800 units = ₹ 5 per unit.

Fixed overhead volume variance = SR (AQ - BQ)

5 (4200 - 4800) = ₹ 3000 A. Hence, option (a) is correct.

Flexible Budgeting

VIKAS Textiles Ltd has just completed the first year of operation on March 31, 2003, and the summarised result of the operating as as follows:

Installed capacity: 20,000 kg of yarn

Production and sales: 14,000 kg of yarn

Contd...

Notes

Table 4

	Rs	Rs
Income		27,30,000
Expenditure		
Variable cost:		
Materials	3,50,000	
Labour	4,20,000	
Oveheads:		
Factory	2,80,000	
Marketing	2,10,000	12,60,000
Contribution		14,70,000
Fixed cost		9,80,000
Profit		4,90,000

The income and expenditure details are given in Table 4.

The managing director wishes to expand the operation for the year 2003-04 and has asked you to prepare Flexible Budgets on capacity utilisation levels of 80 per cent, 90 per cent and 100 per cent based on the following estimate:

1. Price (₹/kg of yarn) at 80 per cent level — ₹ 210; at 90 per cent level — ₹ 200; at 100 per cent — ₹ 195

   Whatever produced during the year is expected to be sold within the year.

2. Increase in variable cost components: materials at 12 per cent; labour at 10 per cent; factory overheads at 15 per cent; and marketing overheads at 20 per cent.

3. Inflation rate applicable to fixed cost is 15 per cent. Additionally, if the capacity utilisation exceeds 80 per cent, fixed cost is expected to increase by 10 per cent up to 100 per cent capacity utilisation.

Comment on this plan of sub-contracting with a view to maximising the profit of the company.

Table 5

Capacity	80%	90%	100%
Production & sales (kg)	16000	18000	20000
Selling price (Rs)	210	200	195
Sales value (Rs lakh)	33.60	36.00	39.00
Variable cost:(WN1)			
Materials at Rs.28	4.48	5.04	5.60
Labour at Rs. 33	5.28	5.94	6.60
OH: Factory at Rs. 23	3.68	4.14	4.60
OH: Marketing at Rs.18	2.88	3.24	3.60
Total variable cost at Rs.102 =	16.32	18.36	20.40
Contribution 33.60 – 16.32 =	17.28	17.64	18.60
Less: Fixed cost	11.27	11.27	11.27
Increase 10%	--	1.127	1.127
Total fixed cost	11.27	12.397	12.397
Profit	6.01	5.243	6.203

Contd...

Notes

The flexible budget (100 per cent capacity 20,000 kg) is shown in Table 5.

Sub-contracting:

Cost (4,000 x 105) = ₹ 4,20,000

Variable cost (mfg) 4,000 x 102 = ₹ 4,08,000

Incremental cost on sub-contracting — ₹ 12,000

Increase in fixed cost saved — ₹ 1,12,700

Increase in fixed cost saved — ₹ 1,12,700

Net saving — ₹ 1,00,700

Table 6

Sales at 100% capacity		3900000
Less: Variable cost 16000 x 102 =	1632000	
Sub-contracting 4000 x 105	420000	2052000
Contribution		1848000
Less: Fixed cost		1127000
Profit		721000
i.e. 6,20,300 + 1,00,700 = 7,21,000		

The profit is worked out in Table 6.

The production manager's plan of sub-contracting the production of 4,000 kg will result into the incremental profit of ₹ 1,00,700. Then the profit of the company will maximise at ₹ 7,21,000.

Table 7

Last yearr rates	Increase	Current year rates (Rs)
Materials	350000/14000 = 25 + 12% =	28
Labour	420000/14000 = 30 + 10% =	33
OH: Factory	280000/14000 = 20 + 15% =	23
OH: Marketing	210000/14000 = 15 + 20% =	18
Fixed cost	980000 + 15% =	1127000

The working note is presented in Table 7.

Queue System

A FOOD storage agency receives stocks of foodgrains at an average rate of eight trucks per hour. A crew of three operatives can unload on an average 10 trucks per hour.

Operatives are paid a wage rate of ₹ 20 per hour. If the crew size is doubled, the unloading rate can be increased to 18 trucks per hour.

When a truck is kept waiting idle an hourly demurrage charge at the rate of ₹ 60 has to be paid.

Determine whether it would be worthwhile to employ a second crew. You may assume that the conditions of a (M/M/1) queue system as applicable.

Arrival rate (lambda) = 8 per hour

Service rate (mu)= 10 per hour

Contd...

Notes

Present cost of unloading [for three operatives] = 3 x 20 per hour = ₹ 60 per hour.

Traffic intensity = P = lambda/mu = 8/10 = 0.80

Average queue length = P2/1 - P = (0.80){+2} = 0.80 x 0.80 / 1 - 0.800.20 = 3.2

Opportunity cost per unit time = 60 x 3.2 = ₹ 192

Actual cost of current year = 60 x 3.2 = ₹ 192

Total cost of current year = opportunity cost + actual cost = 192 + 192 = ₹ 384

Proposal: If crew is doubled, service rate = mu = 18 per hour

Cost of unloading [for six operatives] = 6 x 20 = ₹ 120 per hour

Traffic intensity = P lambda/mu = 8/18 = 4/9 = 0.44

Average queue length = P2/1-P = (0.44){+2} /1-0.44 = 0.44 x 0.44/0.56 = 0.35

Opportunity cost = 60 x 0.35 = ₹ 21

Actual cost = 120 x 0.35 = ₹ 42

Total cost = 21 + 42 = ₹ 63

Thus the total cost of by employing second crew is less than the single crew. Hence, it is worthwhile employing the second crew.

Note: Decision to employ second crew will not change even if we consider total salary for crew is taken as ₹ 20 per hour.

Simulation

SHREE Lakshmi Finance Corporation is an investment company.

The management of the company wants to study the investment in a project based on the following three factors: a) market demand, b) profitability, and c) amount of investment required.

Table 8

Annual demand		Profit per unit		Investment required	
Units	Probability	Rs	Probability	Rs	Probability
20,000	0.05	3.00	0.10	17,50,000	0.25
25,000	0.10	5.00	0.20	20,00,000	0.50
30,000	0.20	7.00	0.40	25,00,000	0.25
35,000	0.30	9.00	0.20		
40,000	0.20	10.00	0.10		
45,000	0.10				
50,000	0.05				

Table 9

Units	Probability	Cumulative probability	Random number
20,000	0.05	0.05	1 – 5
25,000	0.10	0.15	6 – 15
30,000	0.20	0.35	16 – 35
35,000	0.30	0.65	36 – 65
40,000	0.20	0.85	66 – 85
45,000	0.10	0.95	86 – 95
50,000	0.05	1.00	96 – 100

Contd...

Notes

Table 10

Rs.	Probability	Cumulative probability	Random number
3.00	0.10	0.10	1 – 10
5.00	0.20	0.30	11 – 30
7.00	0.40	0.70	31 – 70
9.00	0.20	0.90	71 – 90
10.00	0.10	1.00	91 – 100

Table 11

Rs.	Probability	Cumulative Probability	Random number
17,50,000	0.25	0.25	1 – 25
20,00,000	0.50	0.75	26 – 75
25,00,000	0.25	1.00	76 – 100

In analysing a new consumer product, the corporation estimates the probability distribution shown in Table 8.

The following random numbers are to be used:

1. For demand: 28, 57, 60, 17, 64, 20, 27, 58, 61, 30.
2. For profit: 19, 07, 90, 02, 57, 28, 29, 83, 58, 41.
3. For investment: 18, 67, 16, 71, 43, 68, 47, 24, 19, 97.

Required: Using simulation technique, repeat the trial ten times, compute the return on investment for each trial considering these three factors into account. Approximately,what is the highest likely return?

Table 12

Random number	Demand units	Random number	Profit Unit	Total profit	Random number	Investment (Rs)
1	2	3	4	5	6	7
28	30,000	19	5	150,000	18	17,50,000
57	35,000	07	3	105,000	67	20,00,000
60	35,000	90	9	315,000	16	17,50,000
17	30,000	02	3	90,000	71	20,00,000
64	35,000	57	7	245,000	43	20,00,000
20	30,000	28	5	150,000	68	20,00,000
27	30,000	29	5	150,000	47	20,00,000
58	35,000	83	9	315,000	24	17,50,000
61	35,000	58	7	245,000	19	17,50,000
30	30,000	41	7	210,000	97	25,00,000

The annual demand, profit per unit and investment required are presented in Tables 9, 10, and 11 respectively. And from Table 12 it can be said that the highest return is ₹ 3,15,000 on the sale of 35,000 units.

Source: http://www.thehindubusinessline.in/mentor/2004/03/15/stories/200403150019 1000.htm

8.2 Summary

- A Single-sever service node consists of a server plus its queue.
- The input to the simulators is based on live data collected at McDonalds Fast Food Restaurant and POSB.
- Our simulation results show that a single-channel queue is more efficient than a multiple-channels queue. If so, why does McDonalds still want to use the multiple-channels queuing system?

Notes

- From our observations, one reason is that in a multiple-channels queue the customers can hop from one queue to another.
- Another reason is the impression of shorter waiting time but our simulation result has proven it wrong. While hopping queues may seem to allow customers to get to the server faster, it is not always true.
- A customer can hop to a shorter queue but the service time needed by the customers in the queue may be longer thus resulting in an even longer waiting time. Another side effect is that it can lead to disorder during the crowded hours and the most recent incident is the fight at McDonalds during their Hello Kitty promotion.

8.3 Keywords

Feature: The object which is being mapped for use in a GIS system. Features may be points, lines or areas.

Geographic Information System (GIS): A mapping system which combines positional data with descriptive information to form a layered map.

Residual: A quality indicator for a GPS position that is determined during the differential correction process indicates uncorrectable error. High residuals are not desirable.

Value: Descriptive information about a Feature. Values can be thought of as the answers to the questions posed by Attributes.

8.4 Self Assessment

Fill in the blanks:

1. In the queuing model two types of events are used, namely arrival and .....
2. Constant Service Time Model shows a ..... Queue leading to a single Service Facility which, in turn, handles each consumer in the precise same quantity of time.
3. A typical plot of ..... number of arrivals and departures on an over-saturated, uniform arrival rate approach to a signalized intersection.
4. The primary purpose of doing a queuing analysis at a signalized intersection is to be able to determine the ..... of queues that form.
5. The ..... is a method that demonstrates dynamically the structure and the behaviors of a system with computer in order to evaluate and predict the effect of the behaviors of some system and provide information for decision.
6. Time-averaged statistics is defined by area under a ..... (integration)
7. In terms of the sequence of event, the system arranges the sequence of event executing with a table called "..... ".
8. The arrival corresponds to the ..... event when a customer reaches a service station
9. The abscissa is time and the ..... is the cumulative number of arrivals as well as the cumulative number of departures for a given stream (or approach) at an intersection.
10. In the queuing model, data units are often considered as ..... and CPUs, transmission lines, channels and terminals are thought as .....
11. In FIFO, the ..... of arrival and departure are the same

Notes

12. Two server queue simulation is classified into servers in ..... and server in parallel.
13. The average of numerous ..... observations (samples) of the number in the service node should be close to l.
14. The purpose of simulation is insight - gained by looking at .....
15. A Single-sever service node consists of a server plus its .....

8.5 Review Questions

1. Enlighten the simulation of single and two server queue.
2. Explore the classification of two server simulation. Discuss in detail.
3. Which one is better to implement in your opinion, job-averaged Statistics or Time-averaged Statistics? Evaluate.
4. Make distinction between single-channel and multi-channel queuing system? Give examples.
5. Explain single-channel queuing system. Analyze it with example.
6. Examine the disadvantages of a single-channel queue.
7. Analyze the impact of Delay and Queue Analysis with example.
8. Explain the arrivals in the single server queue.
9. Find out the difference between the output statistics and job average statistics.
10. How are job-averaged and time-average statistics related in Little's theorem?

Answers: Self Assessment

1.	departure	2.	single
3.	cumulative	4.	probability distribution
5.	computer simulation	6.	curve
7.	event table	8.	real-world
9.	ordinate	10.	customs, queue
11.	Order	12.	series
13.	random	14.	statistics
15.	queue		

8.6 Further Readings

Books

Balci, O.,(1994), Validation, verification, and testing techniques throughout the life cycle of a simulation study, Annals of Operations Research, 53: 121–173.

Balci, O., (2001), A methodology for certification of modelling and simulation applications, ACM Transactions on Modelling and Computer Simulation, 11: 352–377.

Notes

Birta, L.G. and Ozmizrak, N.F., (1996), A knowledge-based approach for the validation of simulation models: The foundation, ACM Transactions on Modelling and Computer Simulation, 6: 67–98.

Boehm, B.W., (1979), Software engineering: R&D trends and defense needs, in: P. Wegner (Ed.), Research Directions in Software Technology, MIT Press, and Cambridge, MA.

Cellier, F.E., (1986), Combined discrete/continuous system simulation – application, techniques and tools, in Proceedings of the 1986 Winter Simulation Conference.

Department of Defense (DoD) Recommended Practices Guide (RPG) for Modelling and Simulation VV&A, Millennium Edition (available at http://vva.dmso.mil).

General Accounting Office, (1976), Report to the Congress: Ways to improve management of federally funded computerized models, report LCD-75-111, U.S. General Accounting Office, Washington, DC.

General Accounting Office, (1979), Guidelines for model evaluation, report PAD-79-17, U.S. General Accounting Office, Washington, DC.

General Accounting Office, (1987), DOD simulations: Improved assessment procedures would increase the credibility of results, report GAO/PEMD-88-3, U.S. General Accounting Office, and Washington, DC.

Ören, T.I., (1981), Concepts and criteria to access acceptability of simulation studies, Communications of the ACM, 24: 180–189.

Online links

http://www.math.wsu.edu/faculty/genz/416/lect/l06-34.pdf

http://www.comp.nus.edu.sg/~teoym/cs6205/9900/L2/sld012.htm

http://en.wikipedia.org/wiki/Queueing_model

Unit 9: Simulation of a PERT Network (I)

Notes

CONTENTS

Objectives
Introduction
9.1 Network Model of a Project
9.2 PERT/CPM for Project Scheduling & Management
9.2.1 Brief History of CPM/PERT
9.2.2 Planning, Scheduling & Control
9.2.3 The Framework for PERT and CPM
9.2.4 Drawing the CPM/PERT Network
9.2.5 Expected Length of a Project
9.2.6 Probability of Project Completion by Due Date
9.3 Program Evaluation Review Technique (PERT)
9.3.1 Principle
9.3.2 Achievement
9.3.3 Uncertainty in Project Scheduling
9.4 Critical Path Method (CPM)
9.5 Analysis on Activity Network
9.6 Summary
9.7 Keywords
9.8 Self Assessment
9.9 Review Questions
9.10 Further Readings

Objectives

After studying this unit, you will be able to:

- Describe network model of a project
- Discuss the analysis of pert network

Introduction

Wherever the hierarchical database model structures data as a tree of records, with each record having one parent record and many children, the network model allows each record to have multiple parent and child records, forming a comprehensive graph structure. This property applies at two levels: The schema is a generalized graph of record types connected by relationship types (called "set types" in CODASYL), and the database itself is a generalized graph of record occurrences connected by relationships (CODASYL "sets"). Cycles are permitted at both levels.

Notes

The chief dispute in favour of the network model, in comparison to the hierarchic model, was that it allowed a more natural modeling of relationships between entities. Although the model was widely implemented and used, it botched to become dominant for two main reasons. Firstly, IBM chose to stick to the hierarchical model with semi-network extensions in their established products such as IMS and DL/I. Secondly, it was ultimately displaced by the relational model, which offered a higher-level, more declarative interface. Until the early 1980s the performance benefits of the low-level navigational interfaces offered by hierarchical and network databases were persuasive for many large-scale applications, but as hardware became faster, the extra productivity and flexibility of the relational model led to the gradual obsolescence of the network model in corporate enterprise usage.

9.1 Network Model of a Project

Input and Output Controllers

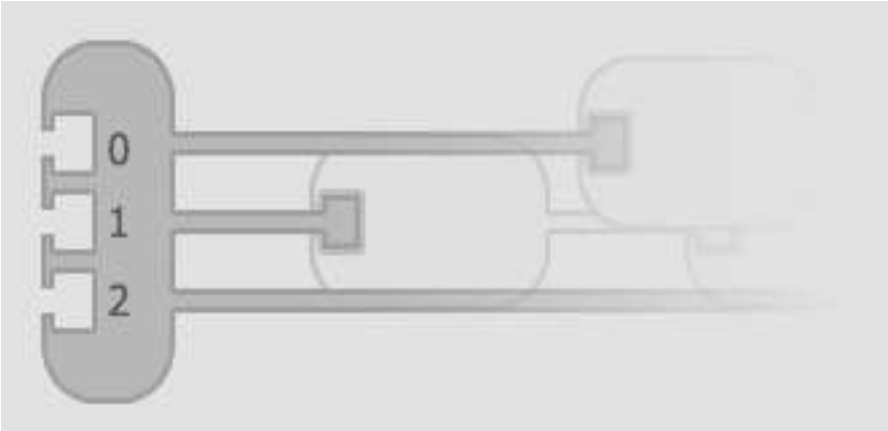
Network input controllers are responsible for relaying data to the proper nodes for processing.

Since it does not process the data, only relays it, there are only two requirements for an input controller.

1. One or more input socket
2. An equal number of output paths

Output controllers function exactly the same way as input controllers, but in reverse. They both are designed to make networks more portable. A network can have more than one input controller and more than one output controller.

Figure 9.1: Input Output Controllers



How to Chat with Networks?

Ultimately the network user will want to communicate using characters and symbols. To accomplish this, some kind of character set must be used. One could use ASCII, however all 8 bits are not necessary for this purpose. ISO developed a fairly unknown 6 bit character set. It provides English communication using the lowest number of bits possible.

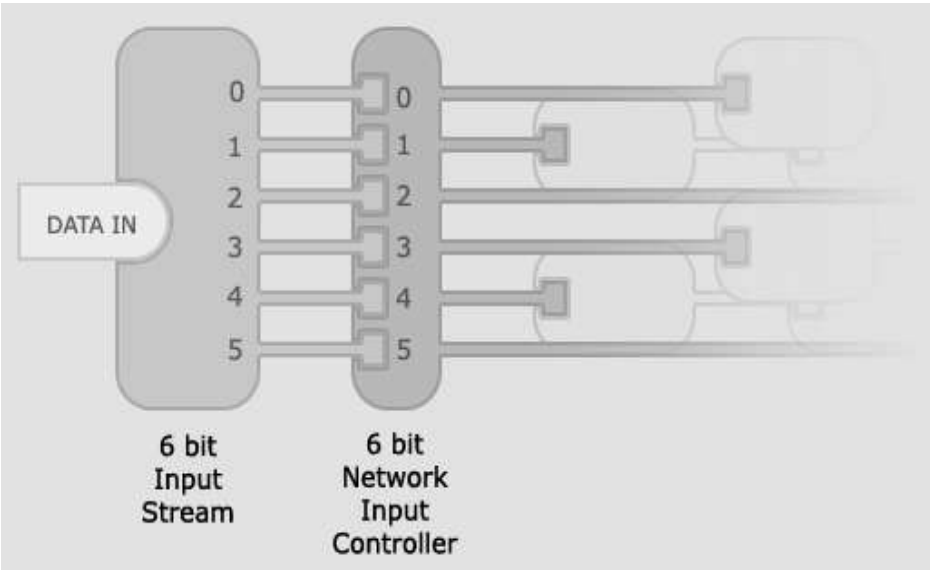
Notes

Notes This is ideal because the fewer input bits you have, the easier it is to engineer the network. Below is a modified version of ISO's 6 bit character set.

	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17
000	NUL	HT	LF	VT	FF	CR	!	?	(	)	*	+	,	-	.	/
020	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	'
040	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	
060	P	Q	R	S	T	U	V	W	X	Y	Z	[	%	]	ESC	DEL

The system now needs an object responsible for interpreting character data and distributing it to the input controller. The diagram below illustrates how the input stream feeds data to the input controller by converting the character data to binary channels.

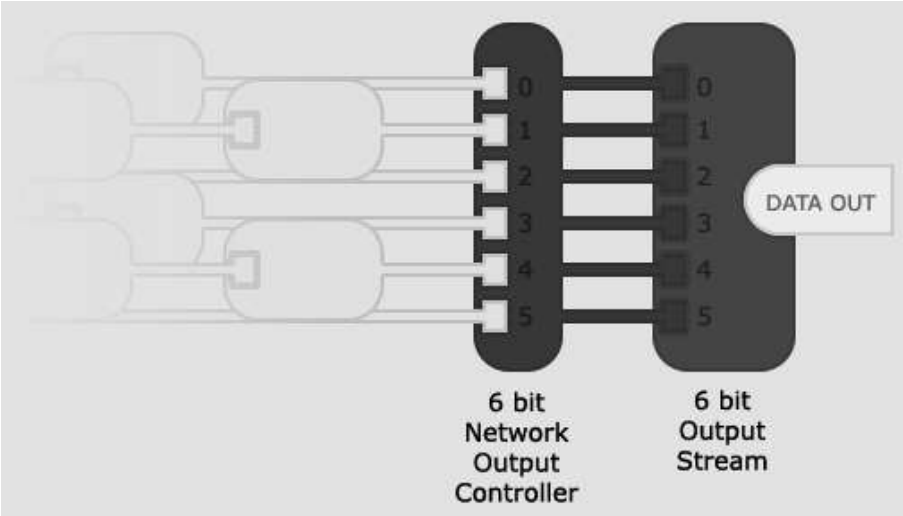
Figure 9.2: Input Stream Working with the Network Input Controller



The output process works exactly the same way, but in reverse. The output stream converts the binary input to character data.

Notes

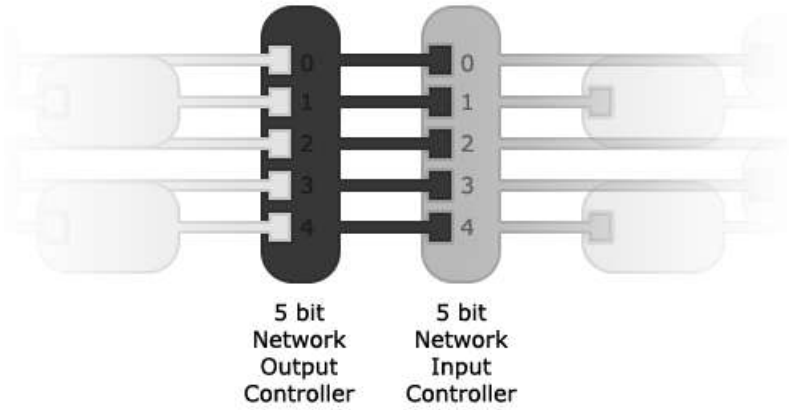
Figure 9.3: Network Output Controller feeding Signals to the Output Stream



Joining Networks

Networks can be connected to other networks to work together in completing a task. This encourages neural network engineers to develop modular networks that can be integrated into much larger neural network systems.

Figure 9.4: A Five Bit Output Controller Connects to a Five Bit Input Controller, Successfully Joining two Networks Together



The **network model** is a database model imagine as a flexible way of representing objects and their relationships. Its distinguishing feature is that the schema, viewed as a graph in which object types are nodes and relationship types are arcs, is not restricted to being a hierarchy or lattice.

Notes

Did u know? **Charles Bachman**

The network model's original inventor was Charles Bachman, and it was developed into a standard specification published in 1969 by the CODASYL Consortium.

9.2 PERT/CPM for Project Scheduling & Management

Essentially, CPM (Critical Path Method) and PERT (Programme Evaluation Review Technique) are project management techniques, which have been created out of the need of Western industrial and military establishments to plan, schedule and control complex projects.

9.2.1 Brief History of CPM/PERT

CPM/PERT or Network Analysis as the technique is occasionally called, developed along two parallel streams, one industrial and the other military.

CPM was the innovation of M.R.Walker of E.I.Du Pont de Nemours & Co. and J.E.Kelly of Remington Rand, circa 1957. The computation was designed for the UNIVAC-I computer. The first test was made in 1958, when CPM was applied to the construction of a new chemical plant. In March 1959, the method was applied to a maintenance shut-down at the Du Pont works in Louisville, Kentucky. Unproductive time was reduced from 125 to 93 hours.

PERT was planned in 1958 for the POLARIS missile program by the Program Evaluation Branch of the Special Projects office of the U.S.Navy, helped by the Lockheed Missile Systems division and the Consultant firm of Booz-Allen & Hamilton. The calculations were so approved so that they could be carried out on the IBM NORC at Dahlgren, Virginia.

Did u know? **Full of NORC**

Naval Ordinance Research Computer

9.2.2 Planning, Scheduling & Control

Planning, Scheduling (or organising) and Control are measured to be basic Managerial functions, and CPM/PERT has been rightfully accorded due significance in the literature on Operations Research and Quantitative Analysis.

Far more than the practical benefits, it was found that PERT/CPM provided a focus around which managers could brain-storm and put their ideas together. It proved to be a great communication medium by which thinkers and planners at one level could converse their ideas, their doubts and fears to another level. Most important, it became a useful tool for evaluating the performance of individuals and teams.

Task There are many variations of CPM/PERT which have been useful in planning costs, scheduling manpower and machine time. CPM/PERT can answer the following important questions:

1. How long will the entire project take to be completed? What are the risks involved?

Contd...

Notes

2. Which are the critical activities or tasks in the project which could delay the entire project if they were not completed on time?
3. Is the project on schedule, behind schedule or ahead of schedule?
4. If the project has to be finished earlier than planned, what is the best way to do this at the least cost?

9.2.3 The Framework for PERT and CPM

Essentially, there are six steps which are common to both the techniques. The procedure is listed below:

1. Define the Project and all of it's important activities or tasks. The Project (made up of several tasks) should have only a single start activity and a single finish activity.
2. Develop the relationships amongst the activities. Decide which activities must precede and which must follow others.
3. Draw the "Network" connecting all the activities. Each Activity should have single event numbers. Dummy arrows are used where required to avoid giving the same numbering to two activities.
4. Allot time and/or cost estimates to each activity
5. Compute the longest time path during the network. This is called the critical path.
6. Use the Network to help plan, schedule, monitor and control the project.

The Key Concept used by CPM/PERT is that a petite set of activities, which make up the longest path through the activity network control the entire project. If these "critical" activities could be identified and assigned to dependable persons, management resources could be optimally used by concentrating on the few activities which determine the fate of the entire project.

Non-critical activities can be replanned, rescheduled and resources for them can be reallocated flexibly, without affecting the whole project.

Task Five useful questions to ask when preparing an activity network are:

1. Is this a Start Activity?
2. Is this a Finish Activity?
3. What Activity Precedes this?
4. What Activity Follows this?
5. What Activity is Concurrent with this?

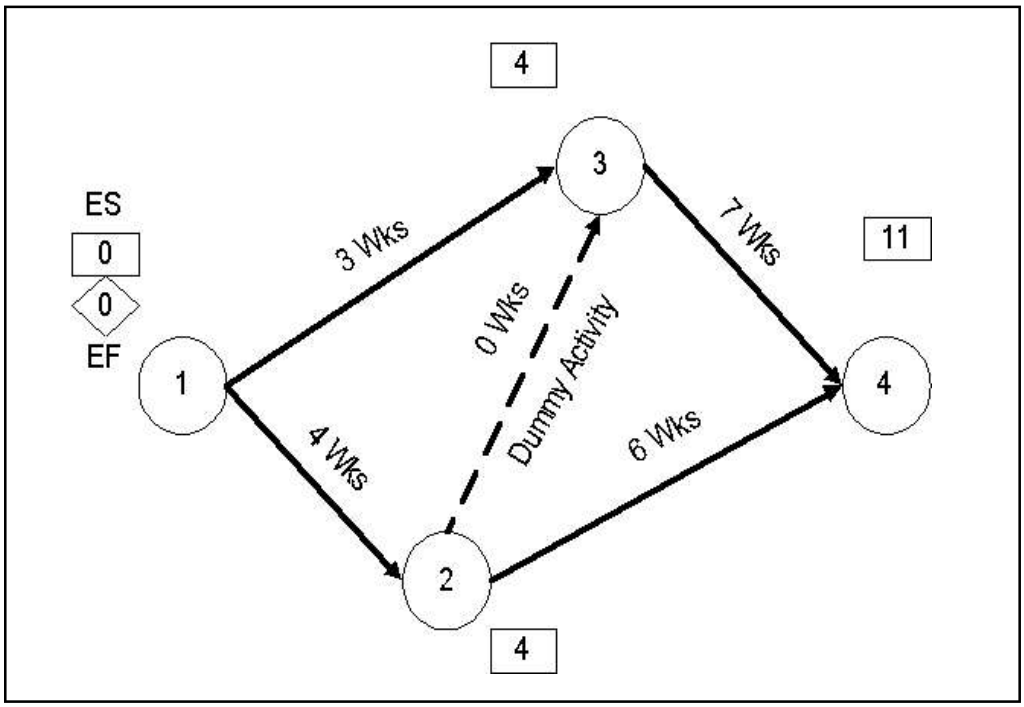
Some activities are successively linked. The second activity can begin only after the first activity is completed. In certain cases, the activities are concurrent, because they are independent of each other and can start simultaneously. This is particularly the case in organisations which have supervisory resources so that work can be delegated to various departments which will be responsible for the activities and their completion as planned.

When work is delegated like this, the need for stable feedback and co-ordination becomes an important senior management pre-occupation.

Notes

9.2.4 Drawing the CPM/PERT Network

Each activity (or sub-project) in a PERT/CPM Network is represented by an arrow symbol. Each activity is preceded and succeeded by an event, represented as a circle and numbered.



At Event 3, we have to evaluate two predecessor activities - Activity 1-3 and Activity 2-3, both of which are predecessor activities. Activity 1-3 gives us an Earliest Start of 3 weeks at Event 3. However, Activity 2-3 also has to be completed before Event 3 can begin. Along this route, the Earliest Start would be 4+0=4. The rule is to take the longer (bigger) of the two Earliest Starts. So the Earliest Start at event 3 is 4.

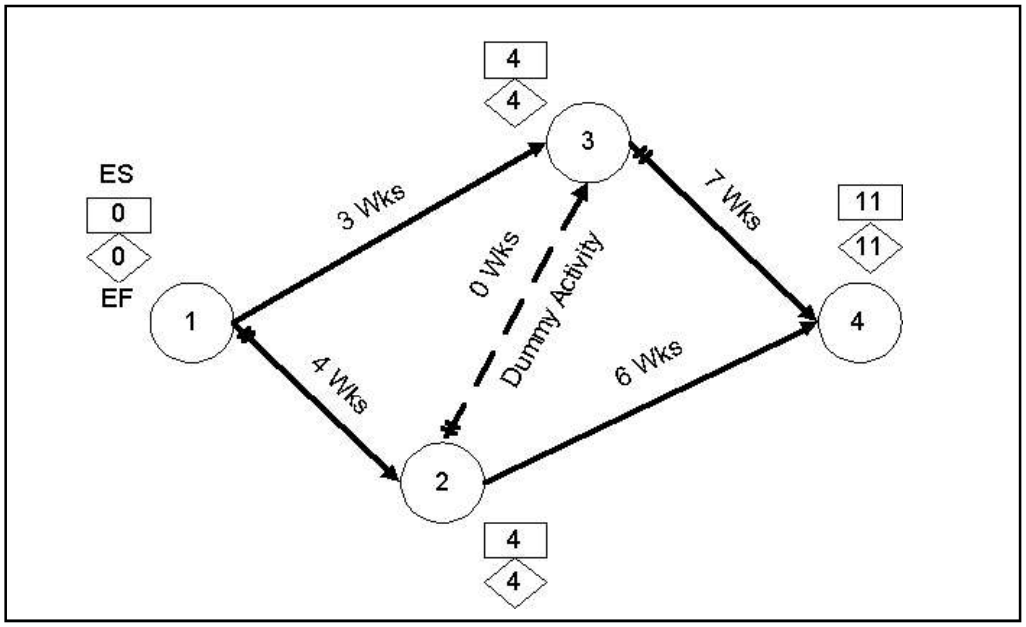
Similarly, at Event 4, we find we have to evaluate two predecessor activities - Activity 2-4 and Activity 3-4. Along Activity 2-4, the Earliest Start at Event 4 would be 10 wks, but along Activity 3-4, the Earliest Start at Event 4 would be 11 wks. Since 11 wks is larger than 10 wks, we select it as the Earliest Start at Event 4.

Notes We have now found the longest path through the network. It will take 11 weeks along activities 1-2, 2-3 and 3-4. This is the Critical Path.

The Backward Pass - Latest Finish Time Rule

To make the Backward Pass, we begin at the sink or the final event and work backwards to the first event.

Notes



At Event 3 there is only one activity, Activity 3-4 in the backward pass, and we find that the value is 11-7 = 4 weeks. However at Event 2 we have to evaluate 2 activities, 2-3 and 2-4. We find that the backward pass through 2-4 gives us a value of 11-6 = 5 while 2-3 gives us 4-0 = 4. We take the **smaller value** of 4 on the backward pass.

Tabulation & Analysis of Activities

We are now ready to tabulate the diverse events and calculate the Earliest and Latest Start and Finish times. We are also now ready to compute the SLACK or TOTAL FLOAT, which is defined as the difference between the Latest Start and Earliest Start.

Event	Duration(Weeks)	Earliest Start	Earliest Finish	Latest Start	Latest Finish	Total Float
1-2	4	0	4	0	4	0
2-3	0	4	4	4	4	0
3-4	7	4	11	4	11	0
1-3	3	0	3	1	4	1
2-4	6	4	10	5	11	1

1. The Earliest Start is the value in the rectangle near the tail of each activity
2. The Earliest Finish is = Earliest Start + Duration

Notes

3. The Latest Finish is the value in the diamond at the head of each activity

4. The Latest Start is = Latest Finish - Duration

There are two significant types of Float or Slack. These are Total Float and Free Float.

TOTAL FLOAT is the spare time available when all earlier activities occur at the **earliest** possible times and all succeeding activities occur at the **latest** possible times.

Total Float = Latest Start - Earliest Start

Activities with zero Total float are on the Critical Path

FREE FLOAT is the spare time available when all previous activities occur at the **earliest** possible times and all succeeding activities occur at the **earliest** possible times.

When an activity has zero Total float, Free float will also be zero.

There are a variety of other types of float (Independent, Early Free, Early Interfering, Late Free, Late Interfering), and float can also be negative. We shall not go into these situations at present for the sake of simplicity and be concerned only with Total Float for the time being.

Having calculated the various parameters of each activity, we are now ready to go into the scheduling phase, using a type of bar chart known as the Gantt Chart.

There are various extra types of float (Independent, Early Free, Early Interfering, Late Free, Late Interfering), and float can also be negative. We shall not go into these situations at present for the sake of plainness and be concerned only with Total Float for the time being. Having computed the various parameters of each activity, we are now ready to go into the scheduling phase, using a type of bar chart known as the Gantt Chart.

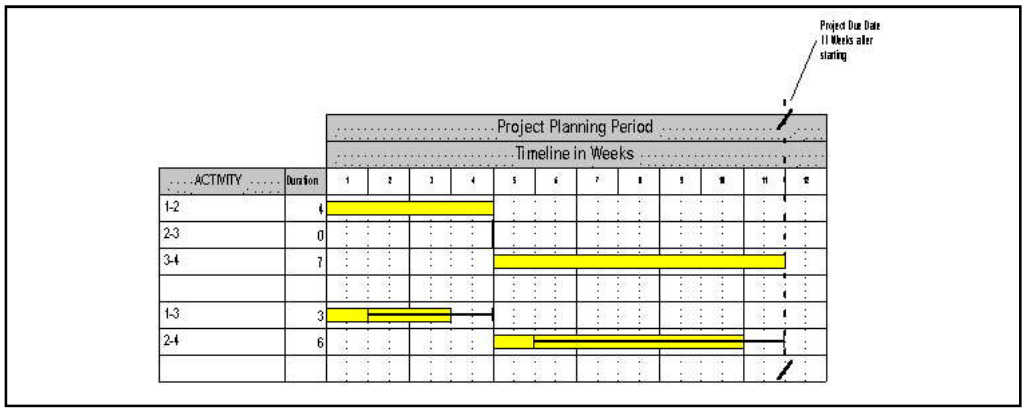
Did u know? **What is the difference between Total Float and Free Float?**

Total Float is the spare time available when all preceding activities occur at the earliest possible times and all succeeding activities occur at the latest possible times.

Free Float is the spare time available when all preceding activities occur at the earliest possible times and all succeeding activities occur at the earliest possible times.

Scheduling of Activities using a Gantt Chart

Once the activities are laid out along a Gantt Chart (Please see chart below), the concepts of Earliest Start & Finish, Latest Start & Finish and Float will become very obvious.



Notes

Activities 1-3 and 2-4 have total float of 1 week each, represented by the solid timeline which begins at the latest start and ends at the latest finish. The difference is the float, which gives us the flexibility to schedule the activity.

For example, we might send the staff on leave during that one week or give them some other work to do. Or we may choose to start the activity slightly later than planned, knowing that we have a week's float in hand. We might even break the activity in the middle (if this is permitted) for a week and divert the staff for some other work, or declare a National or Festival holiday as required under the National and Festival Holidays Act.

These are some of the examples of the use of float to schedule an activity. Once all the activities that can be scheduled are scheduled to the convenience of the project, normally reflecting resource optimisation measures, we can say that the project has been scheduled.

Lab Exercise

A Social Project manager is faced with a project with the following activities:

Activity-id	Activity - Description	Duration
1-2	Social Work Team to live in Village	5 Weeks
1-3	Social Research Team to do survey	12 Weeks
3-4	Analyse results of survey	5 Weeks
2-4	Establish Mother & Child Health Program	14 Weeks
3-5	Establish Rural Credit Programme	15 Weeks
4-5	Carry out Immunisation of Under Fives	4 Weeks

1. Draw the arrow diagram, using the helpful numbering of the activities, which suggests the following logic:
2. Unless the Social Work team lives in the village, the Mother and Child Health Programme cannot be started due to ignorance and superstition of the villagers
3. The Analysis of the survey can obviously be done only after the survey is complete.
4. Until rural survey is done, the Rural Credit Programme cannot be started
5. Unless Mother and Child Programme is established, the Immunisation of Under Fives cannot be started

   Calculate the Earliest and Latest Event Times

   Tabulate and Analyse the Activities

   Schedule the Project using a Gantt Chart

The PERT (Probabilistic) Approach

So far we have talked about projects, where there is high assurance about the outcomes of activities. In other words, the cause-effect logic is well known. This is particularly the case in Engineering projects.

However, in Research & Development projects, or in Social Projects which are defined as "Process Projects", where learning is an significant outcome, the cause-effect relationship is not so well established.

Notes

In such situations, the PERT approach is useful, because it can contain the variation in event completion times, based on an expert's or an expert committee's estimates.

For each activity, three time estimates are taken

1. The Most Optimistic
2. The Most Likely
3. The Most Pessimistic

The Duration of an activity is calculated using the following formula:

$t_e = \frac{t_o + 4^t{}_m + t_p}{6}$ Where t_e is the Expected time, t_o is the Optimistic time, t_m is the most credible activity time and t_p is the Pessimistic time.

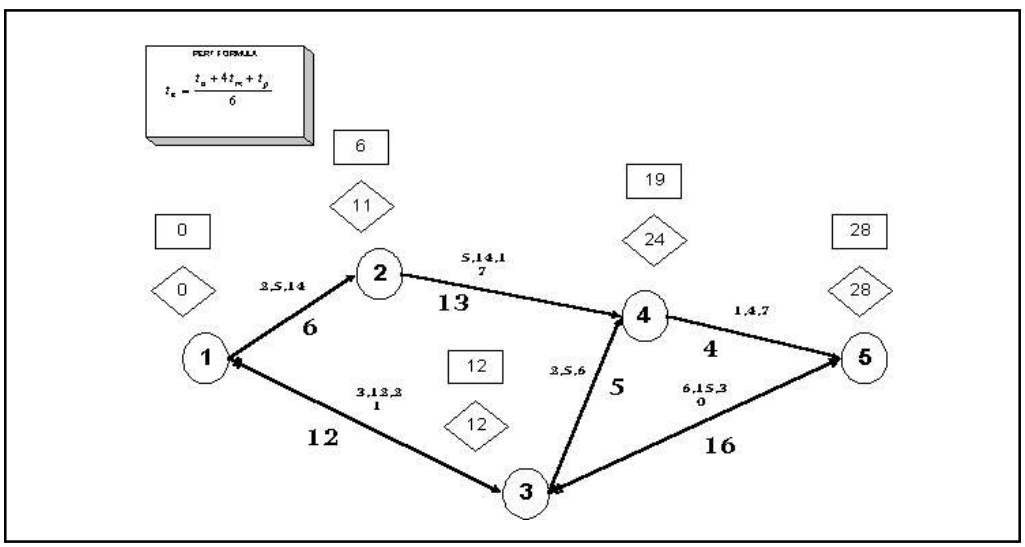
It is not necessary to go into the theory behind the formula. It is sufficient to know that the weights are based on an approximation of the Beta distribution.

The Standard Deviation, which is a good measure of the variability of each activity is calculated by the rather simplified formula:

$s_1 = \frac{t_p - t_o}{6}$ The Variance is the Square of the Standard Deviation.

PERT Calculations for the Social Project

In our Social Project, the Project Manager is now not so certain that each activity will be completed on the basis of the single estimate he gave. There are many assumptions involved in each estimate, and these assumptions are illustrated in the three-time estimate he would prefer to give to each activity.



Notes

In Activity 1-3, the time estimates are 3,12 and 21. Using our PERT formula, we get:

$$t_e = \frac{3 + (4 \times 12) + 21}{6} = \frac{72}{6} = 12$$

$$s_1 = \frac{(21-3)}{6} = \frac{18}{6} = 3$$

The Standard Deviation (s.d.) for this activity is also calculated using the PERT formula

We calculate the PERT event times and other details as below for each activity:

Event	t_o	t_m	t_p	t_e	ES	EF	LS	LF	TF	s.d.	Var.
1-3	3	12	21	12	0	12	0	12	0	3	9
3-5	6	15	30	16	12	28	12	28	0	4	16
1-2	2	5	14	6	0	6	5	11	5	2	4
2-4	5	14	17	13	6	19	11	24	5	2	4
3-4	2	5	8	5	12	17	19	24	7	1	1
4-5	1	4	7	4	19	23	24	28	5	1	1

Estimating Risk

Having calculated the s.d. and the Variance, we are prepared to do some risk analysis. Before that we should be aware of two of the most important assumptions made by PERT.

1. The Beta distribution is suitable for calculation of activity durations.
2. Activities are independent, and the time necessary to complete one activity has no bearing on the completion times of it's successor activities in the network. The validity of this assumption is questionable when we consider that in practice, many activities have dependencies.

9.2.5 Expected Length of a Project

PERT assumes that the expected length of a project (or a sequence of independent activities) is purely the sum of their separate expected lengths.

Therefore the summation of all the t_e's along the critical path gives us the length of the project.

Likewise the variance of a sum of independent activity times is equal to the sum of their individual variances.

In our instance, the sum of the variance of the activity times along the critical path, VT is found to be equal to (9+16) = 25.

Notes

The square root VT gives us the standard deviation of the project length. Thus, ST=Ö 25=5. The higher the standard deviation, the greater the ambiguity that the project will be completed on the due date.

Though the t_e's are randomly distributed, the average or expected project length Te approximately follows a Normal Distribution.

As we have a lot of information about a Normal Distribution, we can make several statistically significant conclusions from these calculations.

A random variable drawn from a Normal Distribution has 0.68 probability of falling within one standard deviation of the distribution average. Consequently, there is a 68% chance that the actual project duration will be within one standard deviation, ST of the estimated average length of the project, t_e.

In our case, the t_e = (12+16) = 28 weeks and the ST = 5 weeks. Assuming t_e to be normally distributed, we can state that there is a probability of 0.68 that the project will be completed within 28 ± 5 weeks, which is to say, between 23 and 33 weeks.

Notes It is known that just over 95% (.954) of the area under a Normal Distribution falls within two standard deviations, we can state that the probability that the project will be completed within 28 ± 10 is very high at 0.95.

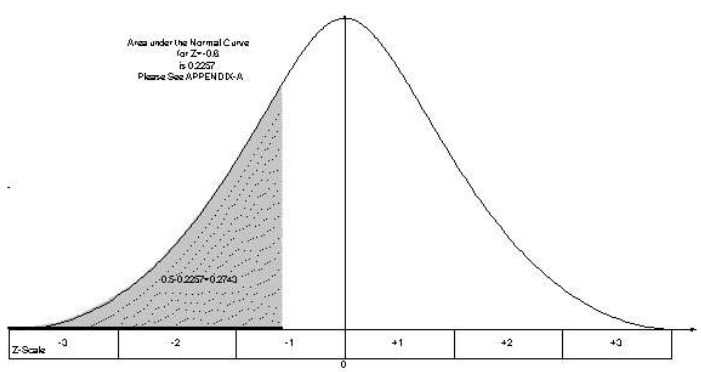
9.2.6 Probability of Project Completion by Due Date

Now, even though the project is estimated to be completed within 28 weeks (t_e=28) our Project Director would like to know what is the probability that the project might be completed within 25 weeks (i.e. Due Date or D=25).

For this calculation, we use the formula for calculating Z, the number of standard deviations that D is away from t_e.

$$Z = \frac{D - t_e}{S_t} = \frac{25 - 28}{5} = \frac{-3}{5} = -0.6$$

By looking at the following extract from a standard normal table, we see that the probability associated with a Z of -0.6 is 0.274. This means that the chance of the project being completed within 25 weeks, instead of the expected 28 weeks is about 2 out of 7. Not very encouraging.

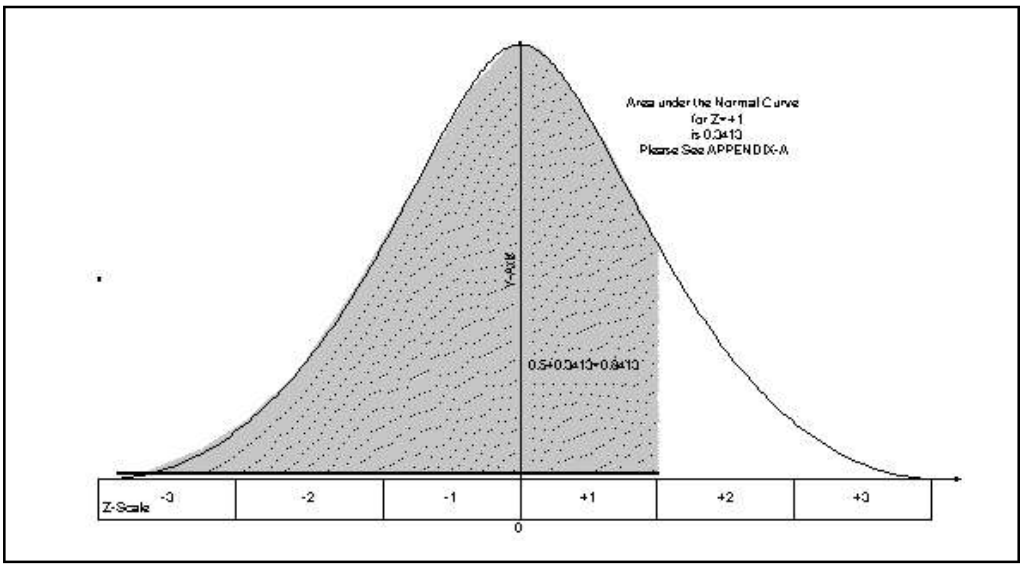


Notes

On the other hand, the probability that the project will be completed within 33 weeks is calculated as follows:

$$Z = \frac{D - t_e}{S_t} = \frac{33 - 28}{5} = \frac{5}{5} = 1$$

The probability associated with Z= +1 is 0.84134. This is a strong probability, and indicates that the odds are 16 to 3 that the project will be completed by the due date.



If the probability of an event is p, the odds for its occurrence are a to b, where:

$$\frac{a}{b} = \frac{p}{1-p} = \frac{0.84134}{0.15866} = \frac{16}{3}$$

9.3 Program Evaluation Review Technique (PERT)

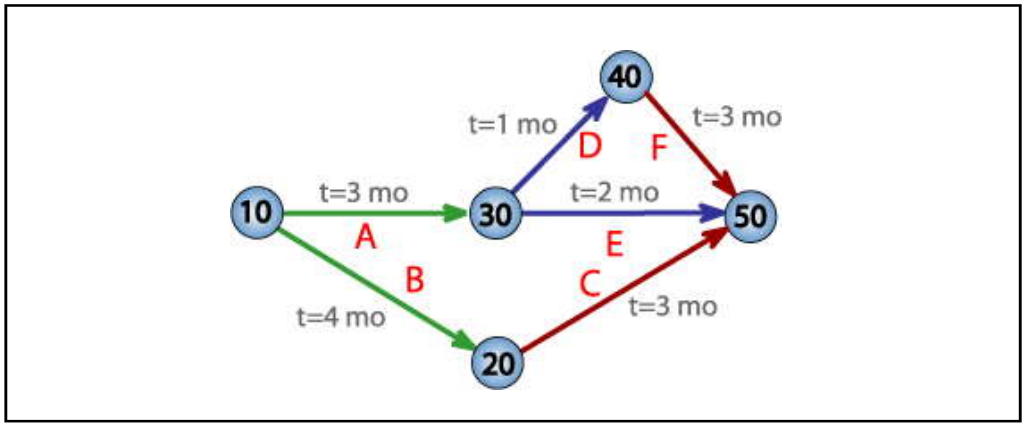
The **Program** (or **Project**) **Evaluation and Review Technique**, usually abbreviated **PERT**, is a model for project management designed to analyze and represent the tasks involved in completing a given project. It is usually used in conjunction with the critical path method or **CPM.**

PERT is a method to examine the involved tasks in completing a given project, especially the time needed to complete each task, and recognize the minimum time needed to complete the total project.

PERT was developed mainly to simplify the planning and scheduling of large and complex projects. It was developed for the U.S. Navy Special Projects Office in 1957 to support the U.S. Navy's Polaris nuclear submarine project. It was able to include uncertainty by making it possible to schedule a project while not knowing precisely the details and durations of all the activities. It is more of an event-oriented technique slightly than start- and completion-oriented, and is used more in projects where time, rather than cost, is the major factor. It is applied to very large-scale, one-time, complex, non-routine infrastructure and Research and Development projects. An example of this was for the 1968 Winter Olympics in Grenoble which applied PERT from 1965 until the opening of the 1968 Games.

Notes

This project model was the first of its kind, a renewal for scientific management, founded by Frederick Taylor (Taylorism) and later refined by Henry Ford (Fordism). DuPont corporation's critical path method was invented at approximately the same time as PERT.



9.3.1 Principle

1. A PERT chart is a tool that ease decision making. The first draft of a PERT chart will number its events sequentially in 10s (10, 20, 30, etc.) to allow the later insertion of additional events.
2. Two successive events in a PERT chart are linked by activities, which are conventionally represented as arrows (see the diagram above).
3. The events are presented in a logical series and no activity can commence until its immediately preceding event is completed.
4. The planner decides which milestones should be PERT events and also choose their "proper" sequence.
5. A PERT chart may have numerous pages with many sub-tasks.

Task "PERT is valuable to manage where multiple tasks are occurring simultaneously to reduce redundancy." Comment

Basic Terms

1. ***PERT event*:** A point that marks the start or completion of one or more activities. It put away no time and uses no resources. When it marks the completion of one or more tasks, it is not "reached" (does not occur) until *all* of the activities leading to that event have been completed.
2. ***Predecessor event*:** An event that instantly precedes some other event without any other events intervening. An event can have multiple predecessor events and can be the predecessor of multiple events.
3. ***Successor event*:** An event that immediately follows some other event without any other intervening events. An event can have multiple successor events and can be the successor of multiple events.

Notes

4. ***PERT activity*:** The actual presentation of a task which consumes time and requires resources (such as labor, materials, space, machinery). It can be understood as representing the time, effort, and resources required to move from one event to another. A PERT activity cannot be performed until the predecessor event has occurred.

5. ***Optimistic time* (O):** The minimum possible time required to achieve a task, assuming everything proceeds better than is normally expected

6. ***Pessimistic time* (P):** The maximum probable time required to accomplish a task, assuming everything goes wrong (but excluding major catastrophes).

7. ***Most likely time* (M):** The best estimate of the time necessary to accomplish a task, assuming everything proceeds as normal.

8. ***Expected time* (T_E):** The best estimate of the time required to accomplish a task, accounting for the fact that things don't always continue as normal (the implication being that the expected time is the average time the task would require if the task were repeated on a number of occasions over an extended period of time).

   $T_E = (O + 4M + P) \div 6$

9. ***Float or Slack*** is the quantity of time that a task in a project network can be delayed without causing a delay - Subsequent tasks – (free float) or Project Completion – (total float)

10. ***Critical Path*:** The longest likely continuous pathway taken from the initial event to the terminal event. It determines the total calendar time required for the project; and, therefore, any time delays along the critical path will delay the reaching of the terminal event by at least the same amount.

11. ***Critical Activity*:** An activity that has total float equal to zero. Activity with zero float does not mean it is on the critical path.

12. ***Lead time*:** The time by which a *predecessor event* must be completed in order to allow sufficient time for the activities that must elapse before a specific PERT event reaches completion.

13. ***Lag time*:** The earliest time by which a *successor event* can follow a specific PERT event.

14. ***Slack*:** The **slack** of an event is a measure of the excess time and resources available in achieving this event. Positive slack would indicate *ahead of schedule*; negative slack would indicate *behind schedule*; and zero slack would indicate *on schedule*.

15. ***Fast tracking*:** Performing more critical activities in parallel

Did u know? **What is Crashing Critical Path?**

Shortening duration of critical activities

9.3.2 Achievement

The first step to scheduling the project is to conclude the tasks that the project requires and the order in which they must be completed. The order may be easy to record for some tasks even as difficult for others (There are two areas that need to be graded, but there are only sufficient bulldozers to do one). Additionally, the time estimates usually reflect the normal, non-rushed time. Many times, the time required to perform the task can be reduced for an additional cost or a reduction in the quality.

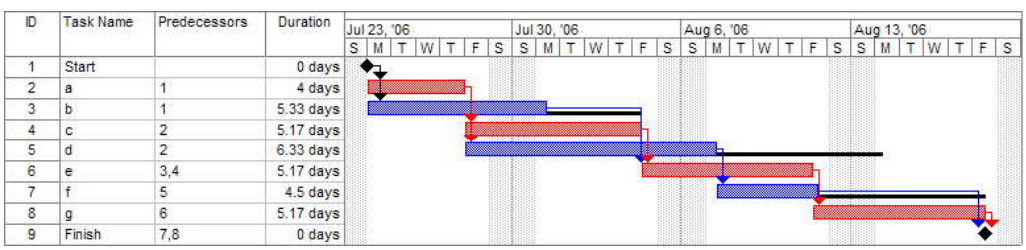
Notes

Example: When building a house, the land must be graded before the foundation can be laid)

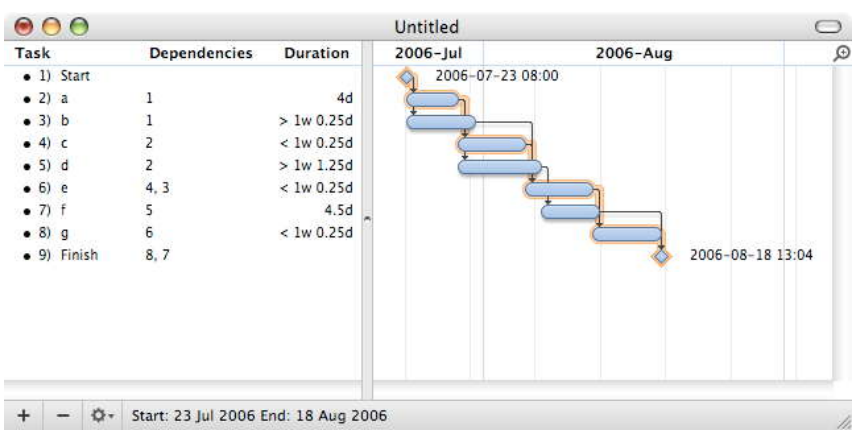
Example: In the following example there are seven tasks, labeled *A* through *G*. Some tasks can be done concurrently (*A* and *B*) while others cannot be done until their predecessor task is complete (*C* cannot begin until *A* is complete). Additionally, each task has three time estimates: the optimistic time estimate (*O*), the most likely or normal time estimate (*M*), and the pessimistic time estimate (*P*). The expected time (T_E) is computed using the formula $(O + 4M + P) \div 6$.

Activity	Predecessor	Time estimates			Expected time
		Opt. (*O*)	Normal (*M*)	Pess. (*P*)	
A	–	2	4	6	4.00
B	–	3	5	9	5.33
C	*A*	4	5	7	5.17
D	*A*	4	6	10	6.33
E	*B*, *C*	4	5	7	5.17
F	*D*	3	4	8	4.50
G	*E*	3	5	8	5.17

Once this step is complete, one can draw a Gantt chart or a network diagram.



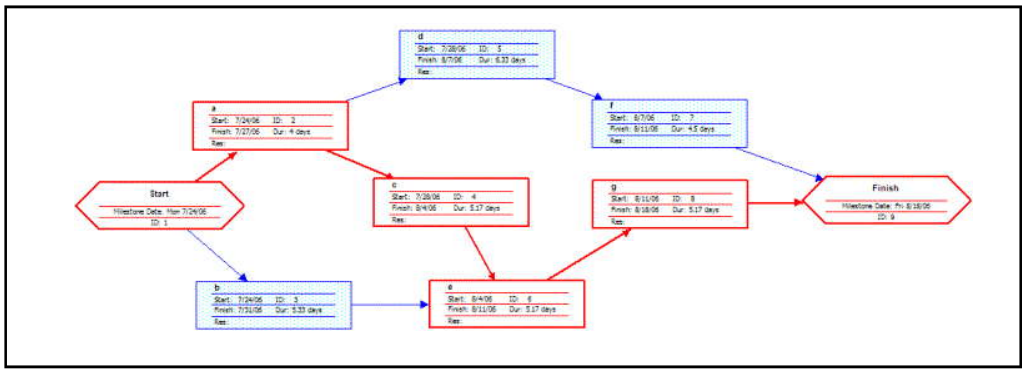
A Gantt chart created using Microsoft Project (MSP). Note (1) the critical path is in red, (2) the slack is the black lines connected to non-critical activities, (3) since Saturday and Sunday are not work days and are thus excluded from the schedule, some bars on the Gantt chart are longer if they cut through a weekend.



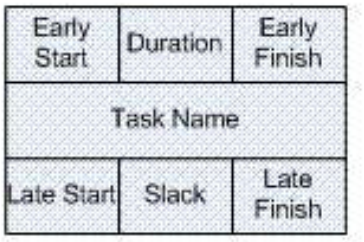
Notes

A Gantt chart created using OmniPlan. Note (1) the critical path is highlighted, (2) the slack is not specifically indicated on task 5 (d), though it can be observed on tasks 3 and 7 (b and f), (3) since weekends are indicated by a thin vertical line, and take up no additional space on the work calendar, bars on the Gantt chart are not longer or shorter when they do or don't carry over a weekend.

A network diagram can be created by hand or by using diagram software. There are two types of network diagrams, activity on arrow and activity on node . Activity on node diagrams are generally easier to create and interpret. To create an AON diagram, it is recommended (but not required) to start with a node named *start*. This "activity" has a duration of zero (0). Then you draw each activity that does not have a predecessor activity (*a* and *b* in this example) and connect them with an arrow from start to each node. Next, since both *c* and *d* list *a* as a predecessor activity, their nodes are drawn with arrows coming from *a*. Activity *e* is listed with *b* and *c* as predecessor activities, so node *e* is drawn with arrows coming from both *b* and *c*, signifying that *e* cannot begin until both *b* and *c* have been completed. Activity *f* has *d* as a predecessor activity, so an arrow is drawn connecting the activities. Likewise, an arrow is drawn from *e* to *g*. Since there are no activities that come after *f* or *g*, it is recommended (but again not required) to connect them to a node labeled *finish*.



A network diagram created using Microsoft Project (MSP). Note the critical path is in red.



A node like this one (from Microsoft Visio) can be used to display the activity name, duration, ES, EF, LS, LF, and slack.

By itself, the network diagram pictured above does not give much more information than a Gantt chart; however, it can be expanded to display more information. The most common information shown is:

1. The activity name
2. The normal duration time
3. The early start time (ES)

Notes

4. The early finish time (EF)
5. The late start time (LS)
6. The late finish time (LF)
7. The slack

In order to determine this information it is assumed that the activities and normal duration times are given. The first step is to determine the ES and EF. The ES is defined as the maximum EF of all predecessor activities, unless the activity in question is the first activity, for which the ES is zero (0). The EF is the ES plus the task duration (EF = ES + duration).

1. The ES for *start* is zero since it is the first activity. Since the duration is zero, the EF is also zero. This EF is used as the ES for *a* and *b*.
2. The ES for *a* is zero. The duration (4 work days) is added to the ES to get an EF of four. This EF is used as the ES for *c* and *d*.
3. The ES for *b* is zero. The duration (5.33 work days) is added to the ES to get an EF of 5.33.
4. The ES for *c* is four. The duration (5.17 work days) is added to the ES to get an EF of 9.17.
5. The ES for *d* is four. The duration (6.33 work days) is added to the ES to get an EF of 10.33. This EF is used as the ES for *f*.
6. The ES for *e* is the greatest EF of its predecessor activities (*b* and *c*). Since *b* has an EF of 5.33 and *c* has an EF of 9.17, the ES of *e* is 9.17. The duration (5.17 work days) is added to the ES to get an EF of 14.34. This EF is used as the ES for *g*.
7. The ES for *f* is 10.33. The duration (4.5 work days) is added to the ES to get an EF of 14.83.
8. The ES for *g* is 14.34. The duration (5.17 work days) is added to the ES to get an EF of 19.51.
9. The ES for *finish* is the greatest EF of its predecessor activities (*f* and *g*). Since *f* has an EF of 14.83 and *g* has an EF of 19.51, the ES of *finish* is 19.51. *Finish* is a milestone (and therefore has a duration of zero), so the EF is also 19.51.

Barring any unforeseen events, the project should take 19.51 work days to complete. The next step is to determine the late start (LS) and late finish (LF) of each activity. This will eventually show if there are activities that have slack. The LF is defined as the minimum LS of all successor activities, unless the activity is the last activity, for which the LF equals the EF. The LS is the LF minus the task duration (LS = LF - duration).

1. The LF for *finish* is equal to the EF (19.51 work days) since it is the last activity in the project. Since the duration is zero, the LS is also 19.51 work days. This will be used as the LF for *f* and *g*.
2. The LF for *g* is 19.51 work days. The duration (5.17 work days) is subtracted from the LF to get an LS of 14.34 work days. This will be used as the LF for *e*.
3. The LF for *f* is 19.51 work days. The duration (4.5 work days) is subtracted from the LF to get an LS of 15.01 work days. This will be used as the LF for *d*.
4. The LF for *e* is 14.34 work days. The duration (5.17 work days) is subtracted from the LF to get an LS of 9.17 work days. This will be used as the LF for *b* and *c*.
5. The LF for *d* is 15.01 work days. The duration (6.33 work days) is subtracted from the LF to get an LS of 8.68 work days.
6. The LF for *c* is 9.17 work days. The duration (5.17 work days) is subtracted from the LF to get an LS of 4 work days.

Notes

7. The LF for *b* is 9.17 work days. The duration (5.33 work days) is subtracted from the LF to get an LS of 3.84 work days.

8. The LF for *a* is the minimum LS of its successor activities. Since *c* has an LS of 4 work days and *d* has an LS of 8.68 work days, the LF for *a* is 4 work days. The duration (4 work days) is subtracted from the LF to get an LS of 0 work days.

9. The LF for *start* is the minimum LS of its successor activities. Since *a* has an LS of 0 work days and *b* has an LS of 3.84 work days, the LS is 0 work days.

The next step is to determine the critical path and if any activities have slack. The critical path is the path that takes the **longest** to complete. To determine the path times, add the task durations for all available paths. Activities that have slack can be delayed without changing the overall time of the project. Slack is computed in one of two ways, slack = LF - EF *or* slack = LS - ES. Activities that are on the critical path have a slack of zero (0).

1. The duration of path *adf* is 14.83 work days.

2. The duration of path *aceg* is 19.51 work days.

3. The duration of path *beg* is 15.67 work days.

The critical path is *aceg* and the critical time is 19.51 work days. It is important to note that there can be more than one critical path (in a project more complex than this example) or that the critical path can change. For example, let's say that activities *d* and *f* take their pessimistic (b) times to complete instead of their expected (T_E) times. The critical path is now *adf* and the critical time is 22 work days. On the other hand, if activity *c* can be reduced to one work day, the path time for *aceg* is reduced to 15.34 work days, which is slightly less than the time of the new critical path, *beg* (15.67 work days).

Assuming these scenarios do not happen, the slack for each activity can now be determined.

1. *Start* and *finish* are milestones and by definition have no duration, therefore they can have no slack (0 work days).

2. The activities on the critical path by definition have a slack of zero; however, it is always a good idea to check the math anyway when drawing by hand.

   a. $LF_a - EF_a = 4 - 4 = 0$

   b. $LF_c - EF_c = 9.17 - 9.17 = 0$

   c. $LF_e - EF_e = 14.34 - 14.34 = 0$

   d. $LF_g - EF_g = 19.51 - 19.51 = 0$

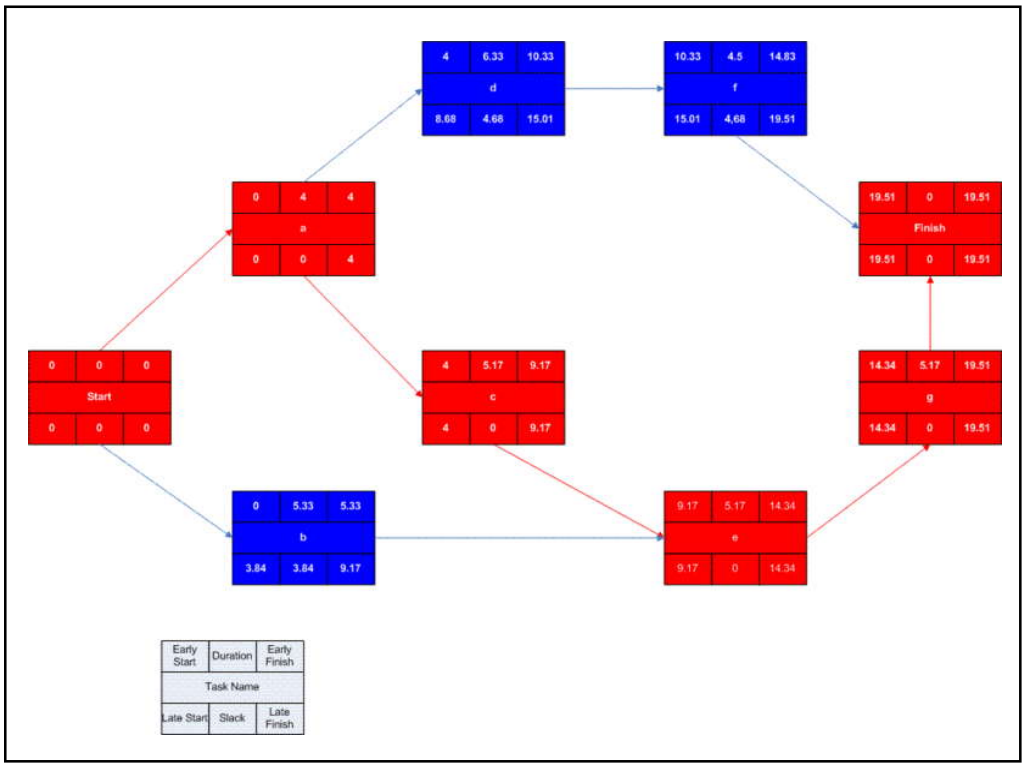
3. Activity *b* has an LF of 9.17 and an EF of 5.33, so the slack is 3.84 work days.

4. Activity *d* has an LF of 15.01 and an EF of 10.33, so the slack is 4.68 work days.

5. Activity *f* has an LF of 19.51 and an EF of 14.83, so the slack is 4.68 work days.

Therefore, activity *b* can be delayed almost 4 work days without delaying the project. Likewise, activity *d* **or** activity *f* can be delayed 4.68 work days without delaying the project (alternatively, *d* and *f* can be delayed 2.34 work days each).

Notes



A completed network diagram created using Microsoft Visio. Note the critical path is in red.

Advantages of PERT

1. PERT chart explicitly defines and makes noticeable dependencies (precedence relationships) between the WBS elements
2. PERT facilitates identification of the critical path and makes this visible
3. PERT facilitates identification of early start, late start, and slack for each activity,
4. PERT offers for potentially reduced project duration due to better understanding of dependencies leading to improved overlapping of activities and tasks where feasible.
5. The large amount of project data can be prepared & presented in diagram for use in decision making.

Did u know? **The full form of AOA and AON**

Activity on Arrow and Activity on Node

Disadvantages of PERT

1. There can be potentially hundreds or thousands of activities and individual dependency relationships
2. The network charts tend to be large and clumsy requiring several pages to print and requiring special size paper
3. The lack of a timeframe on most PERT/CPM charts makes it harder to show status although colours can help (e.g., specific colour for completed nodes)

Notes

4. When the PERT/CPM charts become unwieldy, they are no longer used to supervise the project.

9.3.3 Uncertainty in Project Scheduling

During project execution, however, a real-life project will never execute exactly as it was planned due to uncertainty. It can be ambiguity resulting from subjective estimates that are prone to human errors or it can be variability arising from unexpected events or risks. The main reason that the Project Evaluation and Review Technique (PERT) may provide inaccurate information about the project completion time is due to this schedule uncertainty. This inaccuracy is large enough to render such estimates as not helpful.

One possibility to maximize solution sturdiness is to include safety in the baseline schedule in order to absorb the anticipated disruptions. This is called proactive scheduling. A pure proactive scheduling is a utopia, incorporating security in a baseline schedule that allows to cope with every possible disruption would lead to a baseline schedule with a very large make-span.

Notes A second approach, reactive scheduling, consists of defining a procedure to react to disruptions that cannot be absorbed by the baseline schedule.

9.4 Critical Path Method (CPM)

The **critical path method** is an algorithm for scheduling a set of project activities. It is an important tool for effectual project management.

The Critical Path Method is a project modeling method developed in the late 1950s by Morgan R. Walker of DuPont and James E. Kelley, Jr. of Remington Rand. Kelley and Walker related their memories of the growth of CPM in 1989. Kelley attributed the term "critical path" to the developers of the Program Evaluation and Review Technique which was developed at about the same time by Booz Allen Hamilton and the US Navy. The precursors of what came to be known as Critical Path were urbanized and put into practice by DuPont between 1940 and 1943 and contributed to the success of the Manhattan Project.

CPM is usually used with all forms of projects, including construction, aerospace and defense, software development, research projects, product development, engineering, and plant maintenance, among others. Any project with mutually dependent activities can apply this method of mathematical analysis. Although the original CPM program and approach is no longer used, the term is usually applied to any approach used to analyze a project network logic diagram.

Basic System

The essential technique for using CPM is to construct a model of the project that includes the following:

1. A list of all activities required to complete the project (typically categorized within a work breakdown structure),
2. The time (duration) that each activity will take to completion, and
3. The dependencies between the activities

Notes

Using these values, CPM calculates the longest path of planned activities to the end of the project, and the earliest and latest that each activity can start and finish without making the project longer. This process determines which activities are "critical" (i.e., on the longest path) and which have "total float". A project can have several, parallel, near critical paths. An additional parallel path through the network with the total durations shorter than the critical path is called a sub-critical or non-critical path.

Notes It can be delayed without making the project longer). In project management, a critical path is the sequence of project network activities which add up to the longest overall duration. This determines the shortest time possible to complete the project. Any delay of an activity on the critical path directly impacts the planned project completion date (i.e. there is no float on the critical path.

These results allow managers to prioritize activities for the effective management of project completion, and to shorten the planned critical path of a project by pruning critical path activities, by "**fast tracking**" (i.e., performing more activities in parallel), and/or by "**crashing the critical path**" (i.e., shortening the durations of critical path activities by adding resources).

Originally, the critical path method considered only logical dependencies between terminal elements. Since then, it has been expanded to allow for the inclusion of resources related to each activity, through processes called activity-based resource assignments and resource leveling. A resource-leveled schedule may include delays due to resource bottlenecks (i.e., unavailability of a resource at the required time), and may cause a previously shorter path to become the longest or most "resource critical" path. A related concept is called the critical chain, which attempts to protect activity and project durations from unforeseen delays due to resource constraints.

Since project schedules change on a regular basis, CPM allows continuous monitoring of the schedule, allows the project manager to track the critical activities, and alerts the project manager to the possibility that non-critical activities may be delayed beyond their total float, thus creating a new critical path and delaying project completion. In addition, the method can easily incorporate the concepts of stochastic predictions, using the Program Evaluation and Review Technique (PERT) and event chain methodology.

Currently, there are several software solutions available in industry that use the CPM method of scheduling, see list of project management software. The method currently used by most project management software is based on a manual calculation approach developed by Fondahl of Stanford University.

Flexibility

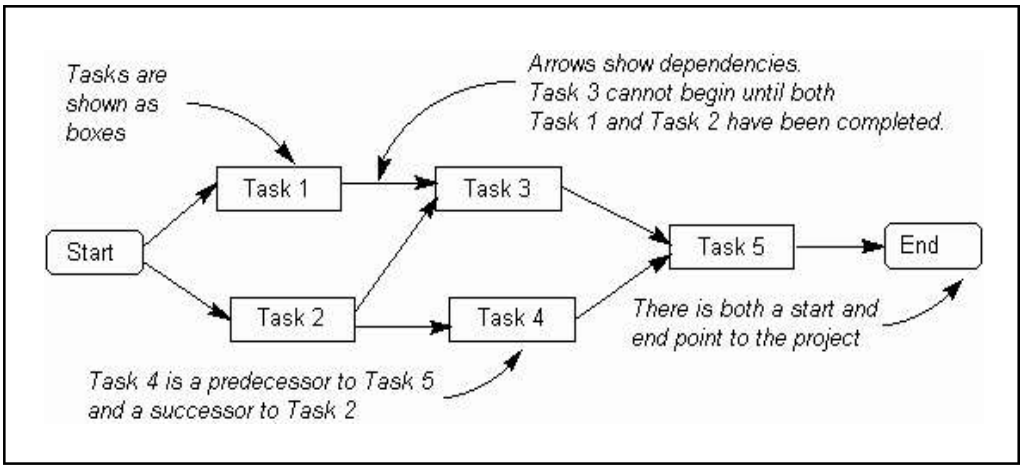
A schedule generated using critical path techniques often is not realised precisely, as estimations are used to calculate times: if one mistake is made, the results of the analysis may change. This could cause an upset in the implementation of a project if the estimates are blindly believed, and if changes are not addressed promptly. However, the structure of critical path analysis is such that the variance from the original schedule caused by any change can be measured, and its impact either ameliorated or adjusted for. Indeed, an important element of project postmortem analysis is the As Built Critical Path (ABCP), which analyzes the specific causes and impacts of changes between the planned schedule and eventual schedule as actually implemented.

Notes

9.5 Analysis on Activity Network

One of the 'second seven tools', this is a typical project planning tool for piecing together a multi-task complex set of interdependent activities. It allows you to calculate what task starts when, what float there is in the project, etc.

Also called an 'Activity-on-Node Diagram', the option (but potentially more troublesome) method of doing this is with an 'Activity-on-Arrow Diagram', or simply 'Arrow Diagram'.



An Activity Network Diagram (AND) is also called an Arrow Diagram (because the pictorial display has arrows in it) or a PERT (Program Evaluation Review Technique) Diagram, and it is used for recognize time sequences of events which are pivotal to objectives. In Critical Path Analysis this helps the teams to comprehend specific event sequences driving time requirements for objective achievement. Activity Network Diagrams are also very useful when a project has multiple activities which need simultaneous management.

Activity Network Diagrams ongoing out as an engineering and construction project management tool. Critical Path Analysis draws on this methodology to identify and standardize medical management activities.

An Activity Network Diagram helps to find out the most competent sequence of events needed to complete any project. It enables you to create a realistic project schedule by graphically showing

1. The total amount of time needed to total the project
2. The sequence in which tasks must be carried out
3. Which tasks can be carried out at the same time
4. Which are the critical tasks that you need to keep an eye on.

It was urbanized by the U.S. Department of Defense. It was first used as a management tool for military projects. It was Adapted as an educational tool for business managers.

Notes

Basic Stipulations

Basic Vocabulary of the PERT Diagram:

1. Activity – part of the project signify by an arrow or line
2. Best Estimate(B) – earliest completion time
3. Critical path(CP) – the most time consuming path through the diagram with no slack time
4. Earliest Start Time(EST) – earliest probable time for an activity to begin
5. Network – the project shown graphically
6. Most likely estimate(M) – length of time probably needed
7. Expected Time(ET) – the average duration time
8. Event – represents the start or finish of one or more events (shown as a circle, square, or other symbol)
9. Latest Start Time(LST) – latest time an activity can begin and still be completed before the next activity
10. Slack Time – latest start time minus earliest start time for an activity
11. Worst estimate(W) – negative time estimate

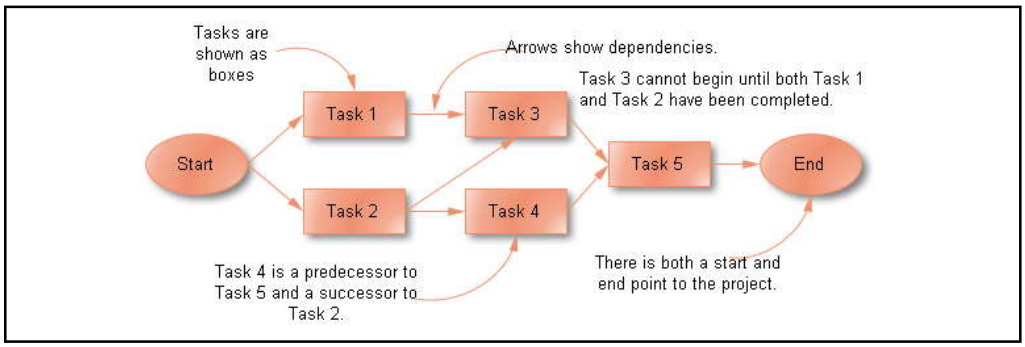
Basic Rule

1. All the preceding activities must be completed previous to the project can begin.
2. The arrows represent the logical precedence of the project.

Procedure for Development

1. Recognize all activities and relationships among them.
2. Sketch the diagram.
3. Estimate the times for each activity, or node, in the diagram .
4. Conclude the critical path.
5. Evaluate the diagram for milestones and target dates in the overall project.

Example of Activity Network Diagram: A project is composed of a set of actions or tasks which typically have some kind of interdependency. For example, before an axle can be turned, it must first be designed, the metal must be purchased, etc. This type of complex system is much easier to appreciate through the use of diagrams than through textual description, as actual interconnections between tasks can be shown. You can draw the activity network diagram easily with Edraw software.



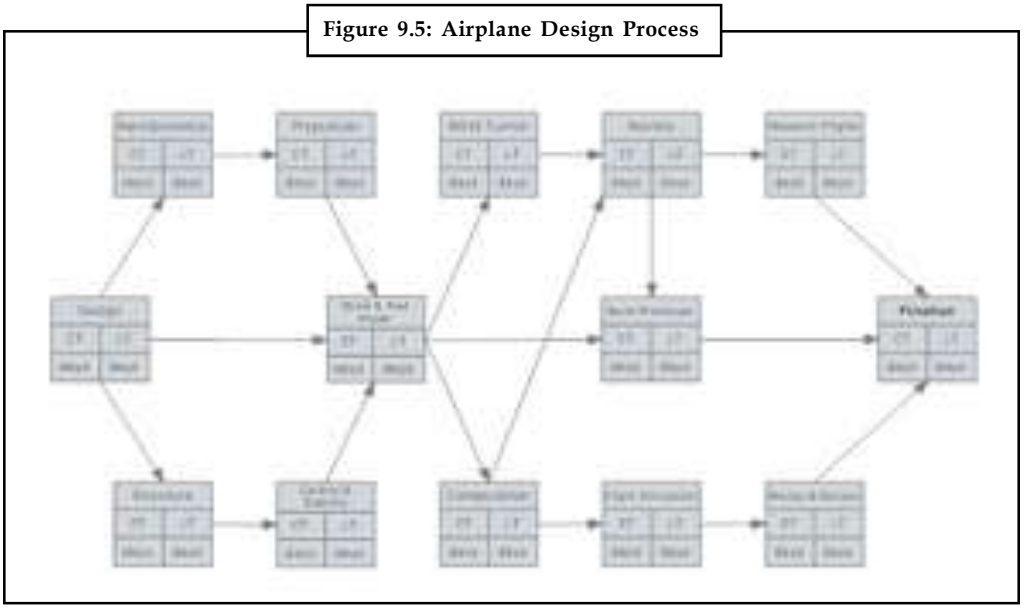
Notes

The Activity Network diagram displays interdependencies between tasks during the use of boxes and arrows. Arrows pointing into a task box come from its predecessor tasks, which must be completed before the task can start. Arrows pointing out of a task box go to its successor tasks, which cannot start until at least this task is complete.

Activity Network (PERT) Chart

Activity Network and Project Evaluation and Review Technique, or PERT, charts are a way of documenting and examine the tasks in a project. They consist of a series of boxes representing tasks with arrows connecting them to show the flow from one task to another. Each box may include the name of the task, the names of those answerable for carrying the task out, and the time estimated to complete each task. This estimate may include a best time, a worst time, and the average of the two.

Figure 9.5: Airplane Design Process



Typical Uses

Activity Networks and PERT Charts are typically used to document complex projects in a visual manner. Elements such as expected time and actual time help viewers make adjustments, and reduce the occurrence of errors during the project. They are also used to establish the critical path of a project.

Best Practices

1. ***Determine the tasks:*** The first step in making an Activity Network (PERT) chart is to establish what tasks are involved in the project.
2. ***Establish order:*** Using one of the conventional Activity Network (PERT) Chart objects, layout each task in the order which it will be completed during the project. Some find using a Gantt Chart helpful when doing this. Also, creating the chart can be made quick and easy using the Activity Network (PERT) Chart template offered by SmartDraw.
3. ***Establish the critical path:*** Draw arrows between the tasks to symbolize the critical path for the project. Any deviation from the critical path can cause delays or errors in the project.

Notes

4. ***Time:*** For each task, input the expected time, elapsed time (if known) and the lead time. Expected time is the amount of time the task is expected to take. Elapsed time is the amount of time the task actually took. Lead time is used to document the time needed before a task can be started. Alternatively, input the best (shortest) time, worst (longest) time, and the average of the two.

5. ***Review:*** Once the Activity (PERT) chart has been completed, the last step is to review and analyze the chart.

Caselet

HLL Plans to enter Network Marketing

HINDUSTAN Lever Network is the name of the FMCG major's network marketing initiative, through which the company hopes to deliver a range of mirror image products in its existing categories.

"We need to be present in all channels. And we see network marketing as a bigger opportunity than a threat," Mr Dalip Sehgal, Executive Director - New Ventures and Marketing Services, Hindustan Lever Ltd (HLL), said at a news conference on Tuesday.

The company, which is already present in the direct selling channel with its beauty products under the brand, Aviance, has introduced another range of products in this channel under the brand name of Lever Home.

The FMCG company, with a large distributive channel, believes there is a huge opportunity in the direct marketing channel. According to Mr K.K. Rajesh, Business Head, Hindustan Lever Network, the Indian industry in direct marketing estimated at ₹ 1,700 crore is growing at 20 per cent. Worldwide, direct selling is estimated at $83 billion.

HLL, an Indian company, is well poised to enter this initiative primarily because of its understanding of the Indian consumer products market, Mr Rajesh said.

Apart from personal care and home care, HLL plans to introduce categories including kitchen care, laundry care, male grooming and foods into the direct marketing channel. "We will look at mirror image products of our existing range," Mr Sehgal said.

HLL had made its foray into direct marketing in 1999 with the launch of Aviance which offers a range of personal care products including skin care, hair care, colour cosmetics, fragrances and anti-perspirants.

Aviance has a base of 75,000 consultants. Home and laundry care will account for 30-40 per cent of the business in this channel.

The products launched under the brand Lever Home include liquid fabric cleanser, dishwashing liquid, laundry concentrate and fabric freshener spray. These products have been developed in-house at HLL and Unilever's global innovation centres.

The company said it is aiming to be the most preferred direct selling company in India by "partnering our consultants to success".

However, the network model at HLL is different from other models. The compensation plan provides multiple payouts, Mr Rajesh said.

Source: http://www.thehindubusinessline.in/2003/01/29/stories/2003012902420100.htm

Notes

9.6 Summary

- Ultimately the network user will want to communicate using characters and symbols.
- To accomplish this, some kind of character set must be used. One could use ASCII, however all 8 bits are not necessary for this purpose. Essentially, CPM (Critical Path Method) and PERT (Programme Evaluation Review Technique) are project management techniques, which have been created out of the need of Western industrial and military establishments to plan, schedule and control complex projects.
- The Program (or Project) Evaluation and Review Technique, usually abbreviated PERT, is a model for project management designed to analyze and represent the tasks involved in completing a given project.
- It is usually used in conjunction with the critical path method or CPM. The Critical Path Method (CPM) is an algorithm for scheduling a set of project activities. It is an important tool for effectual project management.
- One of the 'second seven tools', this is a typical project planning tool for piecing together a multi-task complex set of interdependent activities. It allows you to calculate what task starts when, what float there is in the project, etc.

9.7 Keywords

AND: Activity Network Diagram

CPM: Critical Path Method

Earliest Start Time(EST): Earliest probable time for an activity to begin

Event: It represents the start or finish of one or more events (shown as a circle, square, or other symbol).

Expected Time(ET): The average duration time

Latest Start Time(LST): The latest time an activity can begin and still be completed before the next activity

Most Likely Estimate(M): Length of time probably needed

PERT: Programme Evaluation Review Technique

9.8 Self Assessment

Fill in the blanks:

1. Computing the longest time path during the network is called the …….
2. …… is the spare time available when all previous activities occur at the earliest possible times and all succeeding activities occur at the earliest possible times.
3. A …… may have numerous pages with many sub-tasks.
4. One possibility to maximize solution sturdiness is to include safety in the baseline schedule in order to absorb the anticipated ……...
5. The critical path method (CPM) is an algorithm for ……. a set of project activities.
6. An additional parallel path through the network with the total ……. shorter than the critical path is called a sub-critical or non-critical path.
7. The structure of critical path analysis is such that the …… from the original schedule caused by any change can be measured, and its impact either ameliorated or adjusted for.

Notes

8. Activity Networks and PERT Charts are typically used to …… complex projects in a visual manner.
9. Once the Activity (PERT) chart has been completed, the last step is to review and …… the chart.
10. Networks can be connected to other …… to work together in completing a task.
11. The …… is a database model imagine as a flexible way of representing objects and their relationships.
12. The first test was made in 1958, when CPM was applied to the construction of a new ……
13. The Key Concept used by …… is that a petite set of activities.
14. Activities with …… Total float are on the Critical Path.
15. The PERT approach is useful, because it can contain the …… in event completion times.

9.9 Review Questions

1. CPM/PERT has been rightfully accorded due significance in the literature on Operations Research and Quantitative Analysis.
2. Analyze what the major step for drawing the CPM/ PERT network?
3. "Performing more critical activities in parallel". What do you mean't by this statement?
4. Do you think the critical path method considered only logical dependencies between terminal elements?
5. All the preceding activities must be completed previous to the project can begin. Is this the rule for analyzing the activity network? Why or why not?
6. Activity Network and Project Evaluation and Review Technique, or PERT, charts are a way of documenting and examine the tasks in a project.
7. Is it possible to decrease the critical path of a project? How?
8. "PERT assumes that the expected length of a project is purely the sum of their separate expected lengths." Explain.
9. Do you think a PERT chart is a tool that ease decision making? If yes then why?
10. A network diagram can be created by hand or by using diagram software. So which software is used in network diagram?

Answers: Self Assessment

1.	critical path	2.	Free Float
3.	PERT chart	4.	Disruptions
5.	Scheduling	6.	Durations
7.	Variance	8.	Document
9.	Analyze	10.	networks
11.	network model	12.	chemical plant
13.	CPM/PERT	14.	zero
15.	variation		

Notes

9.10 Further Readings

Books

Balci, O.,(1994), Validation, verification, and testing techniques throughout the life cycle of a simulation study, Annals of Operations Research, 53: 121–173.

Balci, O., (2001), A methodology for certification of modeling and simulation applications, ACM Transactions on Modeling and Computer Simulation, 11: 352–377.

Birta, L.G. and Ozmizrak, N.F., (1996), A knowledge-based approach for the validation of simulation models: The foundation, ACM Transactions on Modeling and Computer Simulation, 6: 67–98.

Boehm, B.W., (1979), Software engineering: R&D trends and defense needs, in: P. Wegner (Ed.), Research Directions in Software Technology, MIT Press, and Cambridge, MA.

Cellier, F.E., (1986), Combined discrete/continuous system simulation – application, techniques and tools, in Proceedings of the 1986 Winter Simulation Conference.

Department of Defense (DoD) Recommended Practices Guide (RPG) for Modeling and Simulation VV&A, Millennium Edition (available at http://vva.dmso.mil).

General Accounting Office, (1976), Report to the Congress: Ways to improve management of federally funded computerized models, report LCD-75-111, U.S. General Accounting Office, Washington, DC.

General Accounting Office, (1979), Guidelines for model evaluation, report PAD-79-17, U.S. General Accounting Office, Washington, DC.

General Accounting Office, (1987), DOD simulations: Improved assessment procedures would increase the credibility of results, report GAO/PEMD-88-3, U.S. General Accounting Office, and Washington, DC.

Ören, T.I., (1981), Concepts and criteria to access acceptability of simulation studies, Communications of the ACM, 24: 180–189.

Online links

http://www.netmba.com/operations/project/pert/

http://syque.com/improvement/Activity%20Network.htm

Notes

Unit 10: Simulation of a PERT Network (II)

CONTENTS

Objectives
Introduction
10.1 Critical Path Computation
10.2 Simulation of an Activity Network
10.3 Computer Program for Simulation
10.4 Summary
10.5 Keywords
10.6 Self Assessment
10.7 Review Questions
10.8 Further Readings

Objectives

After studying this unit, you will be able to:

- Understand Critical Path Computation
- Discuss the simulation of a activity network
- Explain resource allocation and cost consideration

Introduction

CPM is commonly used with all forms of projects, including construction, aerospace and defense, software development, research projects, product development, engineering, and plant maintenance, among others. An activity network Diagram is also called: arrow diagram, network diagram, activity chart, node diagram, CPM (Critical Path Method) chart or PERT (Program Evaluation and Review Technique) chart.

10.1 Critical Path Computation

In the Activity Diagram described in the before, a network of tasks were set up to show the dependent sequence of activities within a project. The Critical Path Method can be applied to such as network to answer the most ordinary question asked of project managers: How long will the overall project take?

Some 'project management' computer programs, such as Microsoft Project, will compute the critical path for you. You can also do it by hand or build a spreadsheet to calculate it, using the method described below.

Notes

How does it work?

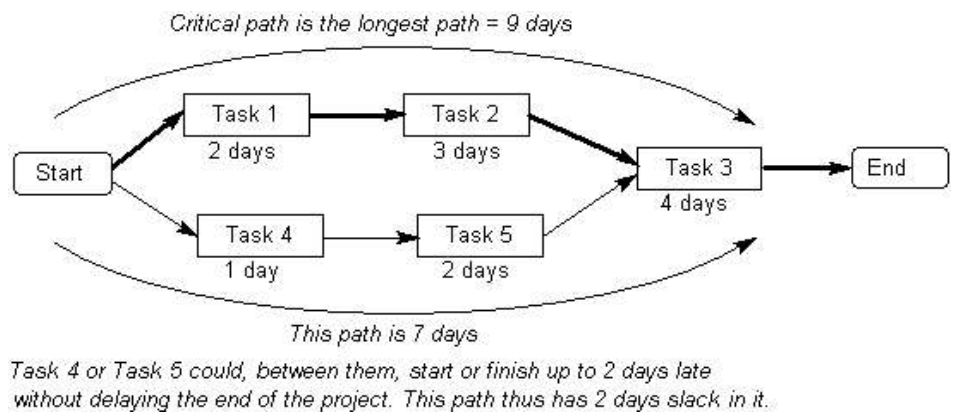
In the Activity Diagram explained before, a network of tasks was set up to show which tasks needed completion before other tasks could be started. A very common next step is to add timings to show how long each task will take and then to recognize the critical path, which is the route through the network that will take the longest amount of time.

Tasks on the critical path have no slack and this feature may be used to actually identify the critical path. It is also quite common to have more than one critical path: indeed, the perfectly balanced project is *all* critical path.

Did u know? **What is Slack?**

Tasks which are not on the critical path have more scope, and may be slipped without affecting the end date of the project. This is called *slack* or *float*.

Figure 10.1: A Critical Path



It may be possible to decrease the critical path of a project (and consequently pull in the completion date) by rearranging some tasks which have a possible sequence or by moving key people onto tasks in the critical path so you can reduce the time for these tasks.

How do you do it?

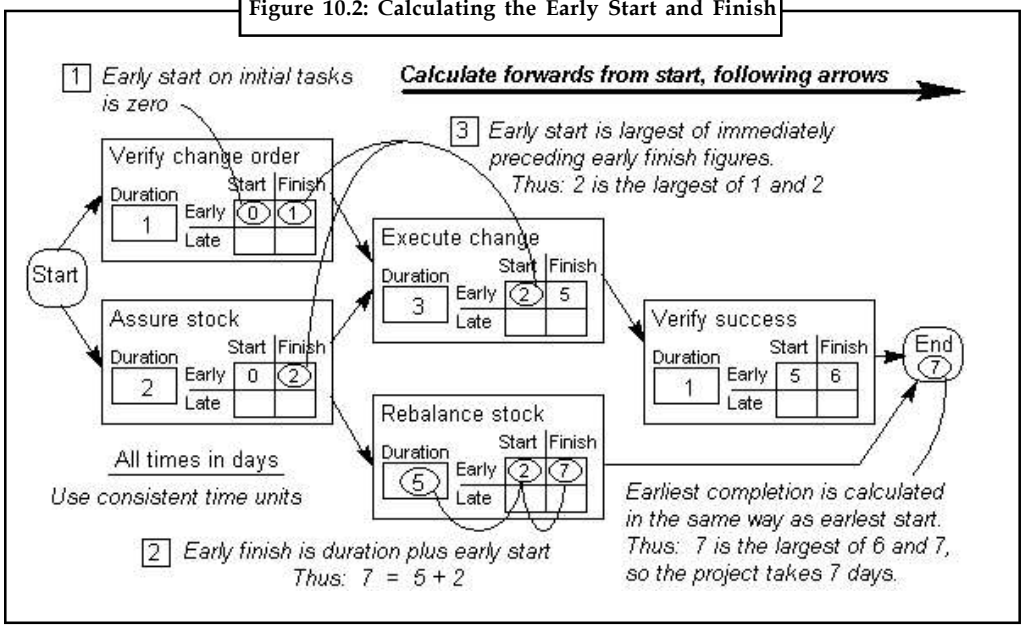
1. Build an Activity Diagram, as described before, as well as estimating the time required (or *duration*) for all tasks. Include a space on each task card for early and late start and finish dates or times (times, rather than dates, is required for tasks where hours or minutes are significant).

   The *early start* and *early finish* are only the earliest times that a task can start or finish. The *late start* and *late finish* are the latest times that a task can start or finish.

2. Starting with the tasks at the beginning of the diagram, absolute the early start and early finish for each task in turn, following the arrows to the next task, as in the figure below. The early start of a task is the same as the early finish of the preceding task. If there is more than one predecessor task, then there are several possible early start figures. Select the *largest* of these. The early finish for each task is equal to the early start plus the duration of the task. The final calculation is for the earliest completion time for the project. This is calculated in the same way as the early start date.

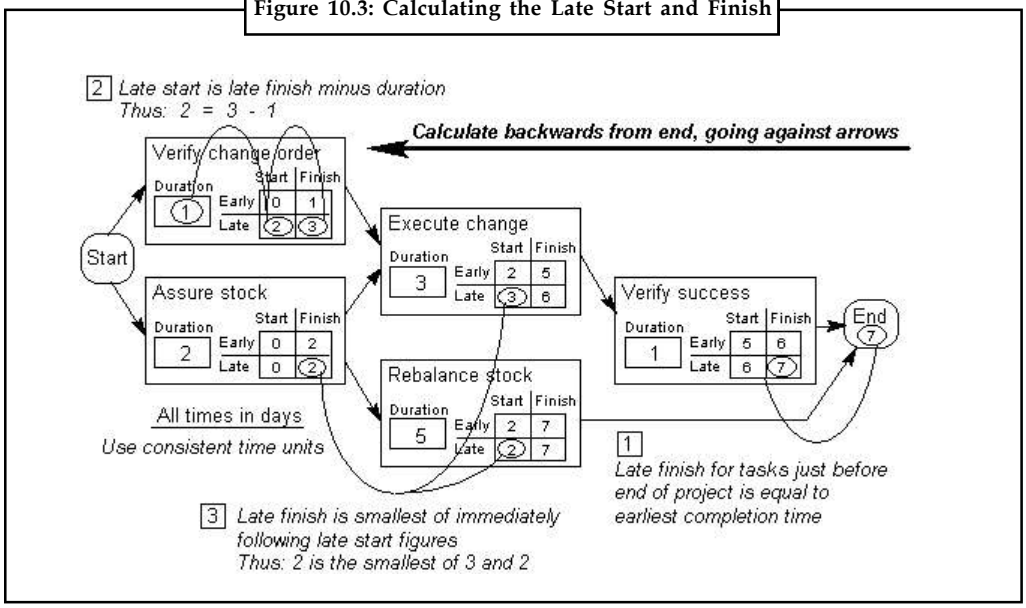
Notes

Figure 10.2: Calculating the Early Start and Finish



3. Preliminary with the tasks at the end of the diagram, calculate the late start and late finish for each task in turn, following the arrows in the reverse direction to the previous task, as in the diagram below. The late finish is the same as the late start of the following task (for the final tasks in the project, this is equal to the earliest completion date). If there is more than one successor task, then there are several possible late figures. Select the *smallest* of these. The late start for each task is the late finish minus the duration of the task. The final calculation is for the earliest completion time for the project. This is calculated in the same way as the early start date.

Figure 10.3: Calculating the Late Start and Finish

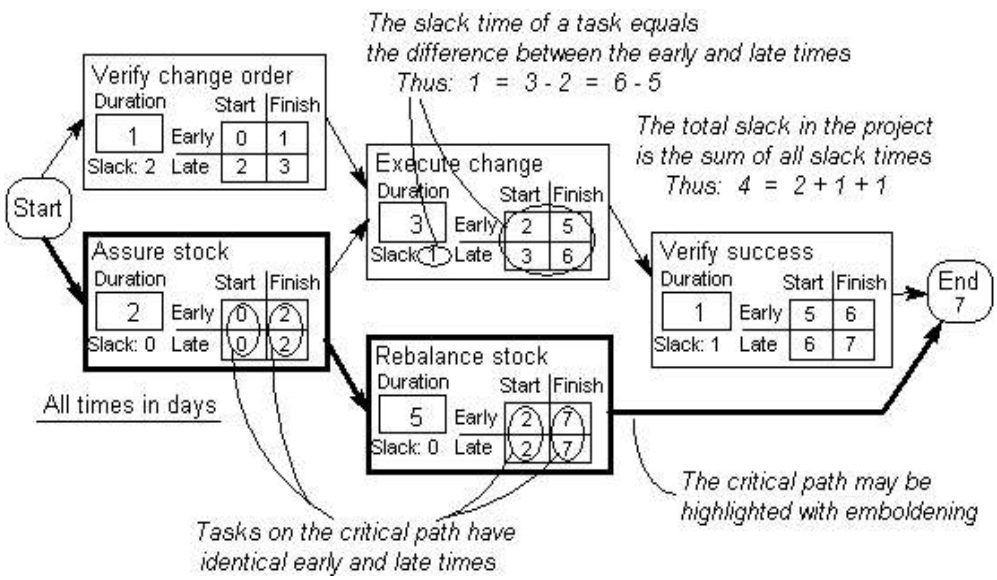


4. You now have, for each task, the initial and latest times that it can start and finish. Now find the slack time (or 'float') for each task by subtracting the early start from the late start.

Notes

Notes The slack time is the amount of time the task can be slipped by without moving the end date of the process. The critical path can now be identified as all paths through the network where the slack time is zero.

Figure 10.4: Calculating the Slack and finding the Critical Path



10.2 Simulation of an Activity Network

An object-orientated simulation approach towards an integrated planning of production systems. The main obstacle for an integrated use of simulation over different planning areas and stages are the dissimilar views on a production system. Therefore, an object model is developed, which enables the co-existence of different views and levels of detail in the same simulation model while maintaining its consistency. This is attained by combining object-orientated technology with a network based simulation approach. The prevailing idea is to offer the opportunity to re-use existing models for the investigation of different aspects of a production system. The approach is theoretically described as a conceptual object model and is thus, independent from a concrete simulation language, tool or environment. The last part of this paper introduces the simulation tool OSim, that implements this object model and demonstrates its usage through an example

Simulation is an application with high degree of intrinsic parallelism. This parallelism can be beneficially exploited in a multiprocessor or multicomputer system to drastically reduce the simulation time. Distributed simulation on slackly coupled systems involves problem partitioning into components, allocation of components to computing systems, communication and co-ordination between components. This paper addresses itself to these problems for distributed simulation of activity networks used for planning and scheduling of physical projects. We discuss conditions under which project network can be decomposed into sub-project networks so that the sub-projects can be simulated independently. The integration is achieved through the

Notes

network of sub-projects. Two alternative strategies under central control and decentralized control for the simulation of time analysis have been presented.

In the simulation of a stochastic activity network (SAN), the usual purpose is to obtain point and confidence-interval estimators of the mean completion time for the network. A new procedure for using path control variants to improve the efficiency of such estimators. Because each path control is the duration of an associated path in the network, the vector of selected path controls has both a known mean and a known covariance matrix. All of this information is incorporated into point- and interval-estimation procedures for both normal and non normal responses. To evaluate the performance of these procedures experimentally, we compare actual versus predicted reductions in point-estimator variance and confidence-interval half-length for a set of SANs in which the following characteristics are systematically varied: (a) the size of the network (number of nodes and activities); (b) the topology of the network; (c) the relative dominance (criticality index) of the critical path; and (d) the percentage of activities with exponentially distributed durations. The experimental results indicate that large variance reductions can be achieved with these estimation procedures in a wide variety of networks.

Task Analyze how the size of network affects the simulation of an activity network?

10.3 Computer Program for Simulation

A computer simulation, a computer model, or a computational model is a computer program, or network of computers, that challenge to simulate an abstract model of a particular system. Computer simulations have become a useful part of mathematical modeling of many natural systems in physics (computational physics), astrophysics, chemistry and biology, human systems in economics, psychology, social science, and engineering. Simulations can be used to explore and gain new insights into new technology, and to estimate the performance of systems too complex for analytical solutions.

Computer simulations vary from computer programs that run a few minutes, to network-based groups of computers running for hours, to ongoing simulations that run for days. The scale of events being simulated by computer simulations has far exceeded anything possible (or perhaps even imaginable) using the traditional paper-and-pencil mathematical modeling. Over 10 years ago, a desert-battle simulation, of one force invading another, involved the modeling of 66,239 tanks, trucks and other vehicles on simulated terrain around Kuwait, using multiple supercomputers in the DoD High Performance Computer Modernization Program; a 1-billion-atom model of material deformation (2002); a 2.64-million-atom model of the complex maker of protein in all organisms, a ribosome, in 2005; and the Blue Brain project at EPFL (Switzerland), began in May 2005, to create the first computer simulation of the entire human brain, right down to the molecular level.

Simulation versus Modeling

Traditionally, shaping large models of systems has been via a mathematical model, which attempts to find analytical solutions to problems and thereby enable the prediction of the behavior of the system from a set of parameters and initial conditions.

While computer simulations might use some algorithms from merely mathematical models, computers can combine simulations with reality or actual events, such as generating input responses, to simulate test subjects who are no longer present. Whereas the missing test subjects are being modeled/simulated, the system they use could be the actual equipment, revealing performance limits or defects in long-term use by these simulated users.

Notes

Note that the term *computer simulation* is broader than *computer modeling*, which entails that all aspects are being modeled in the computer representation. However, computer simulation also includes generating inputs from simulated users to run actual computer software or equipment, with only part of the system being modeled: an example would be flight simulators which can run machines as well as actual flight software.

Computer simulations are used in many fields, counting science, technology, entertainment, health care, and business planning and scheduling.

Computer simulation was developed hand-in-hand with the rapid growth of the computer, following its first large-scale deployment during the Manhattan Project in World War II to model the process of nuclear detonation. It was a simulation of 12 hard spheres using a Monte Carlo algorithm. There are many different types of computer simulations; the common feature they all share is the attempt to generate a sample of representative scenarios for a model in which a complete enumeration of all possible states of the model would be prohibitive or impossible. Computer models were initially used as a supplement for other arguments, but their use later became rather widespread.

Notes Computer simulation is often used as an adjunct to, or substitution for, modeling systems for which simple closed form analytic solutions are not possible.

Data Preparation

The external data requirements of simulations and models vary extensively. For some, the input might be just a few numbers (for example, simulation of a waveform of AC electricity on a wire), while others might require terabytes of information (such as weather and climate models).

Input sources also vary widely:

1. Sensors and other physical devices connected to the model;
2. Control surfaces used to direct the progress of the simulation in some way;
3. Current or Historical data entered by hand;
4. Values extracted as by-product from other processes;
5. Values output for the purpose by other simulations, models, or processes.

Lastly, the time at which data is available varies:

1. "invariant" data is often built into the model code, either because the value is truly invariant (e.g. the value of π) or because the designers consider the value to be invariant for all cases of interest;
2. data can entered into the simulation when it starts up, for example by reading one or more files, or by reading data from a preprocessor;
3. data can be provided during the simulation run, for example by a sensor network;

Because of this variety, and that a lot of common elements exist between diverse simulation systems, there are a large number of specialized simulation languages. The best-known of these must be Simula (sometimes Simula-67, after the year 1967 when it was proposed). There are now many others.

Systems accepting data from external sources must be very careful in significant what they are receiving. While it is easy for computers to read in values from text or binary files, what is much

Notes

harder is knowing what the accuracy (compared to measurement resolution and precision) of the values is. Often it is expressed as "error bars", a minimum and maximum deviation from the value seen within which the true value (is expected to) lie. Because digital computer mathematics is not perfect, rounding and truncation errors will multiply this error up, and it is therefore useful to perform an "error analysis" to check that values output by the simulation are still usefully accurate.

Even small errors in the original data can accumulate into considerable error later in the simulation. While all computer analysis is subject to the "GIGO" (garbage in, garbage out) restriction, this is especially true of digital simulation. Indeed, it was the observation of this inherent, cumulative error, for digital systems that is the origin of chaos theory.

Types

Computer models can be classified according to several independent pairs of attributes, including:

1. Stochastic or deterministic (and as a special case of deterministic, chaotic) - see External links below for examples of stochastic vs. deterministic simulations
2. Steady-state or dynamic
3. Continuous or discrete (and as an important special case of discrete, discrete event or DE models)
4. Local or distributed.

Equations define the relationships between elements of the modeled system and attempt to find a state in which the system is in equilibrium. Such models are often used in simulating physical systems, as a simpler modeling case before dynamic simulation is attempted.

1. Dynamic simulations model changes in a system in response to (usually changing) input signals.
2. *Stochastic* models use *random number generators to* model chance or random events;
3. A *discrete event simulation* (DES) manages events in time. Most computer, logic-test and fault-tree simulations are of this type. In this type of simulation, the simulator maintains a queue of events sorted by the simulated time they should occur. The simulator reads the queue and triggers new events as each event is processed. It is not important to execute the simulation in real time. It's often more important to be able to access the data produced by the simulation, to discover logic defects in the design, or the sequence of events.
4. A *continuous dynamic simulation* performs numerical solution of differential-algebraic equations or differential equations (either partial or ordinary). Periodically, the simulation program solves all the equations, and uses the numbers to change the state and output of the simulation. Applications include flight simulators, construction and management simulation games, chemical process modeling, and simulations of electrical circuits. Originally, these kinds of simulations were actually implemented on analog computers, where the differential equations could be represented directly by various electrical components such as op-amps. By the late 1980s, however, most "analog" simulations were run on conventional digital computers that emulate the behavior of an analog computer.
5. A special type of discrete simulation which does not rely on a model with an underlying equation, but can nonetheless be represented formally, is *agent-based simulation*. In agent-based simulation, the individual entities (such as molecules, cells, trees or consumers) in the model are represented directly (rather than by their density or concentration) and possess an internal *state* and set of behaviors or *rules* which determine how the agent's state is updated from one time-step to the next.

Notes

6. Distributed models run on a network of interconnected computers, possibly through the Internet. Simulations dispersed across multiple host computers like this are often referred to as "distributed simulations". There are several standards for distributed simulation, including Aggregate Level Simulation Protocol (ALSP), Distributed Interactive Simulation (DIS), the High Level Architecture (simulation) (HLA) and the Test and Training Enabling Architecture (TENA).

CGI Computer Simulation

Formerly, the output data from a computer simulation was occasionally presented in a table, or a matrix, showing how data was affected by numerous changes in the simulation parameters. The use of the matrix format was related to traditional use of the matrix concept in mathematical models; however, psychologists and others noted that humans could quickly perceive trends by looking at graphs or even moving-images or motion-pictures generated from the data, as displayed by computer-generated-imagery (CGI) animation. Although observers couldn't necessarily read out numbers, or spout math formulas, from observing a moving weather chart, they might be able to predict events (and "see that rain was headed their way"), much faster than scanning tables of rain-cloud coordinates. Such intense graphical displays, which transcended the world of numbers and formulae, sometimes also led to output that lacked a coordinate grid or omitted timestamps, as if straying too far from numeric data displays. Today, weather forecasting models tend to balance the view of moving rain/snow clouds against a map that uses numeric coordinates and numeric timestamps of events.

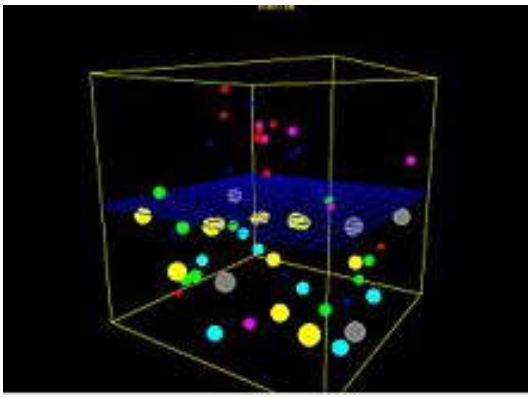
Likewise, CGI computer simulations of CAT scans can simulate how a tumor might shrink or change, during an extended period of medical treatment, presenting the passage of time as a spinning view of the visible human head, as the tumor changes.

Did u know? **One more application of CGI computer simulations.**

One more applications of CGI computer simulations are being developed to graphically display large amounts of data, in motion, as changes occur during a simulation run.

Computer Simulation in Science

Figure 10.5: Computer Simulation of the Process of Osmosis



Notes

Generic examples of types of computer simulations in science, which are derived from an underlying mathematical description:

1. A numerical simulation of differential equations which cannot be solved analytically, theories which involve continuous systems such as phenomena in physical cosmology, fluid dynamics (e.g. climate models, roadway noise models, roadway air dispersion models), continuum mechanics and chemical kinetics fall into this category.
2. A stochastic simulation, typically used for discrete systems where events occur probabilistically, and which cannot be described directly with differential equations (this is a discrete simulation in the above sense). Phenomena in this category include genetic drift, biochemical or gene regulatory networks with small numbers of molecules.

Specific examples of computer simulations follow:

1. Statistical simulations based upon an agglomeration of a large number of input profiles, such as the forecasting of equilibrium temperature of receiving waters, allowing the gamut of meteorological data to be input for a specific locale. This technique was developed for thermal pollution forecasting .
2. Agent based simulation has been used effectively in ecology, where it is often called *individual based modeling* and has been used in situations for which individual variability in the agents cannot be neglected, such as population dynamics of salmon and trout (most purely mathematical models assume all trout behave identically).
3. Time stepped dynamic model. In hydrology there are several such hydrology transport models such as the SWMM and DSSAM Models developed by the U.S. Environmental Protection Agency for river water quality forecasting.
4. Computer simulations have also been used to formally model theories of human cognition and performance, e.g. ACT-R
7. Computer simulation using molecular modeling for drug discovery
8. Computer simulation for studying the selective sensitivity of bonds by mechanochemistry during grinding of organic molecules.
9. Computational fluid dynamics simulations are used to simulate the behaviour of flowing air, water and other fluids. There are one-, two- and three- dimensional models used. A one dimensional model might simulate the effects of water hammer in a pipe. A two-dimensional model might be used to simulate the drag forces on the cross-section of an aeroplane wing. A three-dimensional simulation might estimate the heating and cooling requirements of a large building.
10. An understanding of statistical thermodynamic molecular theory is fundamental to the appreciation of molecular solutions. Development of the Potential Distribution Theorem (PDT) allows one to simplify this complex subject to down-to-earth presentations of molecular theory.

Notes

Computer Simulation in Practical Contexts

Figure 10.6: Smog around Karl Marx Stadt (Chemnitz)

Computer simulations are used in a wide variety of practical contexts, such as:

1. Analysis of air pollutant dispersion using atmospheric dispersion modeling
2. Design of complex systems such as aircraft and also logistics systems.
3. Design of Noise barriers to effect roadway noise mitigation
4. Flight simulators to train pilots
5. Weather forecasting
6. Simulation of other computers is emulation.
7. Forecasting of prices on financial markets (for example Adaptive Modeler)
8. Behavior of structures (such as buildings and industrial parts) under stress and other conditions
9. Design of industrial processes, such as chemical processing plants
10. Strategic Management and Organizational Studies
11. Reservoir simulation for the petroleum engineering to model the subsurface reservoir
12. Process Engineering Simulation tools.
13. Robot simulators for the design of robots and robot control algorithms
14. Urban Simulation Models that simulate dynamic patterns of urban development and responses to urban land use and transportation policies. See a more detailed article on Urban Environment Simulation.
15. Traffic engineering to plan or redesign parts of the street network from single junctions over cities to a national highway network, for transportation system planning, design and operations.
16. Modeling car crashes to test safety mechanisms in new vehicle models

The reliability and the trust people put in computer simulations depends on the validity of the simulation model, therefore verification and validation are of crucial importance in the development of computer simulations. Another important aspect of computer simulations is that of reproducibility of the results, meaning that a simulation model should not provide a

Notes

different answer for each execution. Although this might seem obvious, this is a special point of attention in stochastic simulations, where random numbers should actually be semi-random numbers. An exception to reproducibility are human in the loop simulations such as flight simulations and computer games. Here a human is part of the simulation and thus influences the outcome in a way that is hard if not impossible to reproduce exactly.

Vehicle manufacturers make use of computer simulation to test safety features in new designs. By building a copy of the car in a physics simulation environment, they can save the hundreds of thousands of dollars that would otherwise be required to build a unique prototype and test it. Engineers can step through the simulation milliseconds at a time to determine the exact stresses being put upon each section of the prototype.

Computer graphics can be used to display the results of a computer simulation. Animations can be used to experience a simulation in real-time e.g. in training simulations. In some cases animations may also be useful in faster than real-time or even slower than real-time modes. For example, faster than real-time animations can be useful in visualizing the buildup of queues in the simulation of humans evacuating a building. Furthermore, simulation results are often aggregated into static images using various ways of scientific visualization.

In debugging, simulating a program execution under test (rather than executing natively) can detect far more errors than the hardware itself can detect and, at the same time, log useful debugging information such as instruction trace, memory alterations and instruction counts. This technique can also detect buffer overflow and similar "hard to detect" errors as well as produce performance information and tuning data.

Drawback

Although sometimes ignored in computer simulations, it is very important to perform sensitivity analysis to ensure that the accuracy of the results are properly understood. For example, the probabilistic risk analysis of factors determining the success of an oilfield exploration program involves combining samples from a variety of statistical distributions using the Monte Carlo method. If, for instance, one of the key parameters (i.e. the net ratio of oil-bearing strata) is known to only one significant figure, then the result of the simulation might not be more precise than one significant figure, although it might (misleadingly) be presented as having four significant figures.

Model Calibration Techniques

The following three steps should be used to produce accurate simulation models: calibration, verification, and validation. Computer simulations are good at portraying and comparing theoretical scenarios but in order to accurately model actual case studies, it has to match what is actually happening today. A base model should be created and calibrated so that it matches the area being studied. The calibrated model should then be verified to ensure that the model is operating as expected based on the inputs. Once the model has been verified, the final step is to validate the model by comparing the outputs to historical data from the study area. This can be done by using statistical techniques and ensuring an adequate R-squared value. Unless these techniques are employed, the simulation model created will produce inaccurate results and not be a useful prediction tool.

Model calibration is achieved by adjusting any available parameters in order to adjust how the model operates and simulates the process. For example in traffic simulation, typical parameters include look-ahead distance, car-following sensitivity, discharge headway, and start-up lost

Notes

time. These parameters influence driver behaviors such as when and how long it takes a driver to change lanes, how much distance a driver leaves between itself and the car in front of it, and how quickly it starts to accelerate through an intersection. Adjusting these parameters has a direct effect on the amount of traffic volume that can traverse through the modeled roadway network by making the drivers more or less aggressive. These are examples of calibration parameters that can be fine-tuned to match up with characteristics observed in the field at the study location. Most traffic models will have typical default values but they may need to be adjusted to better match the driver behavior at the location being studied.

Model verification is achieved by obtaining output data from the model and comparing it to what is expected from the input data. For example in traffic simulation, traffic volume can be verified to ensure that actual volume throughput in the model is reasonably close to traffic volumes input into the model. Ten percent is a typical threshold used in traffic simulation to determine if output volumes are reasonably close to input volumes. Simulation models handle model inputs in different ways so traffic that enters the network, for example, may or may not reach its desired destination. Additionally, traffic that wants to enter the network may not be able to if any congestion exists. This is why model verification is a very important part of the modeling process.

The final step is to validate the model by comparing the results with what's expected based on historical data from the study area. Ideally, the model should produce similar results to what has happened historically. This is typically verified by nothing more than quoting the R2 statistic from the fit. This statistic measures the fraction of variability that is accounted for by the model. A high R2 value does not necessarily mean the model fits the data well. Another tool used to validate models is graphical residual analysis. If model output values are drastically different than historical values, it probably means there's an error in the model. This is an important step to verify before using the model as a base to produce additional models for different scenarios to ensure each one is accurate. If the outputs do not reasonably match historic values during the validation process, the model should be reviewed and updated to produce results more in line with expectations. It is an iterative process that helps to produce more realistic models.

Validating traffic simulation models requires comparing traffic estimated by the model to observed traffic on the roadway and transit systems. Initial comparisons are for trip interchanges between quadrants, sectors, or other large areas of interest. The next step is to compare traffic estimated by the models to traffic counts, including transit ridership, crossing contrived barriers in the study area. These are typically called screenlines, cutlines, and cordon lines and may be imaginary or actual physical barriers. Cordon lines surround particular areas such as the central business district or other major activity centers. Transit ridership estimates are commonly validated by comparing them to actual patronage crossing cordon lines around the central business district.

Three sources of error can cause weak correlation during calibration: input error, model error, and parameter error. In general, input error and parameter error can be adjusted easily by the user. Model error however is caused by the methodology used in the model and may not be as easy to fix. Simulation models are typically built using several different modeling theories that can produce conflicting results. Some models are more generalized while others are more detailed. If model error occurs as a result of this, in may be necessary to adjust the model methodology to make results more consistent.

In order to produce good models that can be used to produce realistic results, these are the necessary steps that need to be taken in order to ensure that simulation models are functioning properly. Simulation models can be used as a tool to verify engineering theories but are only valid if calibrated properly. Once satisfactory estimates of the parameters for all models have been obtained, the models must be checked to assure that they adequately perform the functions for which they are intended. The validation process establishes the credibility of the model by

Notes

demonstrating its ability to replicate actual traffic patterns. The importance of model validation underscores the need for careful planning, thoroughness and accuracy of the input data collection program that has this purpose. Efforts should be made to ensure collected data is consistent with expected values.

Example: In traffic analysis, it is typically common for a traffic engineer to perform a site visit to verify traffic counts and become familiar with traffic patterns in the area. The resulting models and forecasts will be no better than the data used for model estimation and validation.

Resource Allocation and Cost Considerations

PERT to include both random activity durations and project cost considerations. Project costs include both planned activity costs and penalties for activities exceeding their allowed durations. Several problem formulations are mentioned, and the determination of a minimum cost schedule satisfying a predetermined project deadline is discussed in detail. This latter problem is formulated as a separable programming problem which can be solved by the computer algorithm documented in the appendices.

PERT/CPM is effectual tool for project planning. PERT stands for critical path method. Even though these techniques were develop for defence applications they are useful for many maintenance projects. The conventional "Bat Chart" as given below is adequate for a small straight forward jobs having around 200 activities. But for larger maintenance projects such as plant shut down and overhaul jobs where activities are many PERT/CPM techniques help us to ascertain the logistics of different resources.

Caselet

Intel 'TEN' to ride Piggyback on Akshaya Network

The 'Akshaya' model of e-services delivery will now provide Intel's latest curriculum module 'Technology and Entrepreneurship' (TEN) through its 2,200 customer service centres (CSCs) across Kerala.

Mr Anjan Ghosh, Regional Director - Corporate Affairs, Asia-Pacific, Intel, launched the curriculum at a grassroots stakeholder roundtable held here on Tuesday.

Basic Concepts

The roundtable on 'ICT as enabler for sustained socio-economic transformation in Kerala' was jointly organised by Akshaya and Intel.

The roundtable highlighted the diverse sustainability pillars and supporting systems leading to sustainability of the Akshaya centres.

The Technology and Entrepreneurship course introduces learners to basic concepts and process of entrepreneurship, and demonstrates how technology can be used to advance a business idea.

Using internet tools and office applications, learners research and formulate a business idea, and create and present a business plan for the same.

Earlier Programme

Intel had launched an e-literacy programme in 2005 through Akshaya and the multinational giant has empowered 86,000 students in Kerala in the age group of 8-16 yrs.

Contd...

Notes

This programme was aimed at reaching out to the children and adolescents in communities with limited or no access to technology.

It was designed to encourage and develop technology skills, critical thinking and collaboration among learners through effective community based education, propagated by successful models of learning outside the formal school setting.

"Our association with Kerala IT has reaped huge benefits not just for the students in Kerala but also to Intel's learning initiative," Mr Ghosh said.

Learning Experience

The whole programme and its execution through various Akshaya centres have been a learning experience on how non-formal education system has come to have a better and farther reach through the right platform.

"This is something that we can take back and try to implement in other Asia Pacific countries as well," Mr Ghosh said. Intel would now seek to replicate this success in the entrepreneurship space.

"Having launched this new course, we are eager to associate with Kerala IT more and see how we can empower the women of the State."

According to Ms Ishita Roy, Director, Kerala State IT Mission, having empowered a section of the populace through the Intel Learn Programme, the next step is to take this association with Intel to encourage entrepreneurship amongst youngsters between 18-25 years.

Next Step

The new curriculum being introduced through Akshaya is a step in this direction. "We are also working on getting this course certified by the Indira Gandhi National Open University, Ms Roy said.

The Intel Learn programme has been a major trigger in the growth and sustainability of Akshaya, according to Mr Korath V. Mathew, Director, Akshaya.

"This has also inspired us into looking at various effective learning methodologies and work towards empowering the citizens of the State through the Akshaya network", he added.

The roundtable had various presentations on how information and communication technology can enable major growth in various sectors in Kerala.

Source: http://www.thehindubusinessline.in/2011/01/20/stories/2011012052372300.htm

10.4 Summary

- In the Activity Diagram described in the before, a network of tasks were set up to show the dependent sequence of activities within a project. An object-orientated simulation approach towards an integrated planning of production systems.
- The main obstacle for an integrated use of simulation over different planning areas and stages are the dissimilar views on a production system.
- A computer simulation, a computer model, or a computational model is a computer program, or network of computers, that challenge to simulate an abstract model of a particular system.

Notes

10.5 Keywords

ALSP: Aggregate Level Simulation Protocol

DIS: Distributed Interactive Simulation

GIGO: Garbage In, Garbage Out

HLA: High Level Architecture

SAN: Stochastic Activity Network

TENA: Test and Training Enabling Architecture

10.6 Self Assessment

Fill in the blanks:

1. Tasks on the critical path have no …….. and this feature may be used to actually identify the critical path.
2. The opportunity to re-use existing models for the …….. of different aspects of a production system.
3. …….. models use random number generators to model chance or random events.
4. Computer simulations vary from computer programs that run a few minutes, to …….. of computers running for hours, to ongoing simulations that run for days.
5. The …….. and the trust people put in computer simulations depends on the validity of the simulation model.
6. …….. run on a network of interconnected computers.
7. The late start for each task is the late finish …….. the duration of the task.
8. The …….. calculation is for the earliest completion time for the project.
9. The slack time for each task by …….. the early start from the late start.
10. An object-orientated simulation approach towards an …….. planning of production systems.
11. Simulation is an application with high degree of intrinsic ……..
12. The vector of selected path controls has both a known mean and a known …… matrix.
13. Simulations can be used to explore and gain new insights into new technology, and to estimate the …….. of systems too complex for analytical solutions.
14. Computer models were initially used as a …….. for other arguments, but their use later became rather widespread.
15. Equations define the relationships between elements of the modeled system and attempt to find a state in which the system is in ……..

10.7 Review Questions

1. Compare actual versus predicted reductions in point-estimator variance and confidence-interval half-length for a set of SANs.
2. Explain how a continuous dynamic simulation performs numerical solution of differential-algebraic equations.

Notes

3. "PERT to include both random activity durations and project cost considerations." Comment.
4. Did you think Computer simulation was developed hand-in-hand with the rapid growth of the computer? Explain.
5. Explain how Systems accepting data from external sources must be very careful in significant what they are receiving?
6. Computer models can be classified according to several independent pairs of attributes. Elaborate.
7. How the output data from a computer simulation was occasionally presented in a table?
8. Explain the computer simulation in science and practical contents.
9. What are model calibration techniques?
10. "PERT/CPM is effectual tool for project planning." Comment.

Answers: Self Assessment

1.	Slack	2.	Investigation
3.	Stochastic	4.	network-based groups
5.	reliability	6.	Distributed models
7.	minus	8.	final
9.	subtracting	10.	integrated
11.	parallelism	12.	covariance
13.	performance	14.	supplement
15.	equilibrium		

10.8 Further Readings

Books

Balci, O.,(1994), Validation, verification, and testing techniques throughout the life cycle of a simulation study, Annals of Operations Research, 53: 121–173.

Balci, O., (2001), A methodology for certification of modeling and simulation applications, ACM Transactions on Modeling and Computer Simulation, 11: 352–377.

Birta, L.G. and Ozmizrak, N.F., (1996), A knowledge-based approach for the validation of simulation models: The foundation, ACM Transactions on Modeling and Computer Simulation, 6: 67–98.

Boehm, B.W., (1979), Software engineering: R&D trends and defense needs, in: P. Wegner (Ed.), Research Directions in Software Technology, MIT Press, and Cambridge, MA.

Cellier, F.E., (1986), Combined discrete/continuous system simulation – application, techniques and tools, in Proceedings of the 1986 Winter Simulation Conference.

Department of Defense (DoD) Recommended Practices Guide (RPG) for Modeling and Simulation VV&A, Millennium Edition (available at http://vva.dmso.mil).

Notes

General Accounting Office, (1976), Report to the Congress: Ways to improve management of federally funded computerized models, report LCD-75-111, U.S. General Accounting Office, Washington, DC.

General Accounting Office, (1979), Guidelines for model evaluation, report PAD-79-17, U.S. General Accounting Office, Washington, DC.

General Accounting Office, (1987), DOD simulations: Improved assessment procedures would increase the credibility of results, report GAO/PEMD-88-3, U.S. General Accounting Office, and Washington, DC.

Ören, T.I., (1981), Concepts and criteria to access acceptability of simulation studies, Communications of the ACM, 24: 180–189.

Online links

http://www.netmba.com/operations/project/pert/

http://syque.com/improvement/Activity%20Network.htm

http://chestjournal.chestpubs.org/content/129/1/182.full

http://en.wikipedia.org/wiki/Computer_simulation

Notes

Unit 11: Design and Evaluation of Simulation Experiments (I)

CONTENTS

Objectives
Introduction
11.1 Simulation Experiment
11.1.1 Advanced Properties
11.1.2 Model Time Properties
11.1.3 Presentation Properties
11.1.4 Windows Properties
11.1.5 Parameters Properties
11.2 Length of Simulation Run
11.3 Variance Reduction Techniques
11.3.1 Common Random Numbers
11.3.2 Antithetic Variables
11.3.3 Control Variates
11.3.4 Conditioning
11.4 Summary
11.5 Keywords
11.6 Self Assessment
11.7 Review Questions
11.8 Further Readings

Objectives

After studying this unit, you will be able to:

- Understand length of simulation runs
- Discuss variance reduction techniques

Introduction

Simulation models provide relatively fast and economical estimates of the performance of alternative system configurations and/or alternative operating procedures. This tutorial provides some techniques for planning a set of simulation model runs, in order to gain insight on system behavior. There is an emphasis on graphical methods for planning the experiment and displaying the results.

11.1 Simulation Experiment

When a new project is formed, one simulation experiment, called Simulation, is automatically created. You can create your own experiments and set up any simulation settings you like.

Notes

To create a Simulation Experiment

1. In the Projects view, right-click the model you are working with and choose New|Experiment from the popup menu. The New Experiment dialog box is displayed.
2. Choose Simulation in the Experiment Type list.
3. Type the name of the experiment in the Name edit box.
4. Select the main object for the experiment from the Main active object class (root) drop-down list.
5. If you want to apply model time settings from another experiment, leave the Copy model time settings from check box selected and choose the experiment in the drop-down list to the right.
6. Click Finish.

General Properties

1. ***Name:*** The name of the experiment.

   Caution Since AnyLogic generates Java class for each experiment, please follow Java naming guidelines and start the name with an uppercase letter.

2. ***Main active object class (root)***: Using the drop-down list, choose the major active object for the experiment. The instance of this class will play a role of a root hierarchical tree of active objects in your model.
3. ***Ignore***: If selected, the experiment is excluded from the model.
4. ***Random number generator***: Here you identify, whether you want to initialize random number generator for this model randomly or with some fixed seed. This makes sense for stochastic models. Stochastic models require a random seed value for the pseudorandom number generator. In this case model runs cannot be reproduced since the model random number generator is initialized with different values for each model run. Specifying the fixed seed value, you initialize the model random number generator with the same value for each model run, thus the model runs are reproducible. Moreover, here you can substitute AnyLogic default RNG with your own RNG.
5. ***Random seed (unique simulation runs)***: If chosen, the seed value of the random number generator is random. In this case random number generator is initialized with the same value for each model run, and the model runs are reproducible.
6. ***Fixed seed (reproducible simulation runs)***: If selected, the seed value of the random number generator is fixed (specify it in the Seed value field). In this case random number generator is initialized with the same value for each model run, and the model runs are reproducible.
7. ***Custom generator (subclass of Random)***: If for any reason you are not content with the quality of the default random number generator Random, you can substitute it with your own one. Just prepare your custom RNG (it should be a subclass of the Java class Random, e.g. MyRandom), choose this particular option and type the expression returning an instance of your RNG in the field on the right, for example: new MyRandom() or new MyRandom(1234)

Notes

By default, all probability distribution functions in AnyLogic, the Enterprise Library objects, the random transitions and events, the random layouts and networks and the AnyLogic simulation engine itself – in other words, all randomness in AnyLogic, is based on the default random number generator. The default random number generator is an instance of the Java class Random, which is a Linear Congruental Generator (LCG).

If for any reason you are not satisfied with the quality of Random, you can:

1. Substitute AnyLogic default RNG with your own RNG.
2. Have multiple RNGs and explicitly specify which RNG should be used when calling a probability distribution function.

To substitute the default RNG with your own RNG

1. Prepare your custom RNG. It should be a subclass of the Java class Random, e.g. MyRandom.
2. Select the experiment and open the General page of its properties.
3. Select the radio button Custom generator (subclass of Random) and in the field on the right write the expression returning an instance of your RNG, for example: New MyRandom() or New MyRandom(1234)

Setting a Custom Random Number Generator as default RNG

The initialization of the default RNG (provided by AnyLogic or by you) occurs during the initialization of the experiment and then before each simulation run.

In addition you can substitute the default RNG at any time by calling:

setDefaultRandomGenerator(Random r)

However you should be aware that before each simulation run the generator will be set up again according to the settings on the General page of the experiment properties.

To use a Custom RNG in a Particular Call of a Probability Distribution Function

1. Create and initialize an instance of your custom RNG. For example, it may be a plain variable myRNG of class Random or its subclass.
2. When calling a probability distribution function, provide myRNG as the last parameter, for example:

   uniform(myRNG) or

   triangular(5, 10, 25, myRNG)

Notes If a probability distribution function has several forms with different parameters, some of them may not have a variant with a custom RNG, but the one with the most complete parameter set always has it.

11.1.1 Advanced Properties

1. ***Maximum available memory***: [*Application option, not applied when model runs as applet*] The maximum size of Java heap allocated for the model.

Notes

2. ***Java machine arguments***: [*Application option, not applied when model runs as applet*] Specify here Java machine arguments you want to apply on launching your model.
3. ***Command-line arguments***: [*Application option, not applied when model runs as applet*] Here you can specify command line arguments you want to pass to your model. You can get the values of passed argument values in the experiment's Additional class code using the method String[] getCommandLineArguments()
4. ***Load root object from snapshot***: If selected, the experiment will load the model state from the snapshot file specified in the control to the right. The experiment will be started from the time when the model state was saved.
5. ***Imports section***: Import statements needed for correct compilation of the experiment class' code. When Java code is generated, these statements are inserted before definition of the Java class.
6. ***Additional class code***: Arbitrary member variables, nested classes, constants and methods are defined here. This code will be inserted into the experiment class definition. You can access these class data members anywhere within this experiment.
7. ***Initial experiment setup***: Experiment initialization code. It is executed when the experiment is created (and its UI is created).
8. ***Before each experiment run***: The code executed before each simulation run. This code is run on setup of the model. At this moment the root object of the model is already created, but the model is not started yet. You may perform here some actions with elements of the root active object,

   Example: assign actual parameter values here.

9. ***Before simulation run***: The code executed before simulation run. This code is executed when simulation engine finishes the model execution (Engine.finished() method is called). This code is not executed when you stop your model by clicking the **Terminate execution** button.
10. ***After simulation run***: The code executed after simulation run.
11. ***Random selection mode for simultaneous events***: If selected, in case several events are available at the same time, AnyLogic can make nondeterministic choice with equal probability for each event. Otherwise, events will be selected in the defined order.
12. ***Differential equations***: Method used to solve ordinary differential equations.
13. ***Algebraic equations***: Method used to solve algebraic equations.
14. ***Mixed equations***: Method used to solve algebraic-differential equations.
15. ***Absolute accuracy***: The desired absolute value accuracy for solving equations. Absolute accuracy is used when it is impossible to use relative accuracy – e.g., when the value is close to zero.
16. ***Time accuracy***: The desired time accuracy for finding change events (switch points) when solving equations.
17. ***Relative accuracy***: The desired relative value accuracy for solving equations with methods that change the integration step (e.g. Newton). Used by default.
18. ***Fixed time step***: Fixed time step for methods using the fixed integration step (e.g. RK4).
19. ***Persistent Top-level Presentation Group***: If selected, the presentation of the experiment is accessible from the code as presentation.

Notes

11.1.2 Model Time Properties

1. ***Use calendar***: If selected, start and stop times are specified as calendar dates.
2. ***Stop***: Defines, whether the model will **Stop at specified time**, **Stop at specified date**, or it will **Never** stop. In the first two cases, the stop time is specified using the **Stop time/Stop date** controls.
3. ***Start time***: [Enabled if Use calendar is not selected] The initial time for the simulation time horizon.
4. ***Start date***: [Enabled if Use calendar is selected] The initial calendar date for the simulation time horizon.
5. ***Stop time***: [Enabled if Use calendar is not selected] The final time for the simulation time horizon (the number of model time units for model to run before it will be stopped).
6. ***Stop date*** : [Enabled if Use calendar is selected] The initial calendar date for the simulation time horizon.

11.1.3 Presentation Properties

1. ***Enable anti-aliasing***: If selected, anti-aliasing is performed. Anti-aliasing is a process of smoothing the drawing of points or lines that would otherwise appear jagged. However, note that more time is spent on rendering the presentation with the anti-aliasing set.
2. ***Enable adaptive frame management***: Choose, whether you want to specify adaptive frame rate, or explicitly defined fixed update rate in frames per second. Adaptive update rate will be recalculated during the model simulation to fix up the specified ratio between simulation speed and presentation smoothness.
3. ***CPU ratio (Presentation:Simulation)***: [Enabled if **Enable adaptive frame management** is set] If the adaptive frame management is turned on, specify here the ratio between simulation speed and presentation smoothness. However, presentation rendering takes longer, and frequent presentation update will slow the model simulation. So, choose between smooth presentation and fast simulation and set up the presentation update rate according to your needs.
4. ***Frames per second***: [Enabled if **Enable adaptive frame management** is not set] If the adaptive frame management is turned off, specify here the presentation update rate (in frames per second). The greater update rate you specify, the smoother presentation will appear. However, presentation rendering takes longer, and frequent presentation update will slow the model simulation.So, choose between smooth presentation and fast simulation and set up the presentation update rate according to your needs.
5. ***Execution mode***: Choose the execution mode here:
6. ***Virtual time (as fast as possible)***: The model runs at its maximum speed and no mapping is made between model time and real time. This time mode is useful when you need to simulate your model for a long period of time.
7. ***Real time with scale...***: In this mode you specify how many model time units one second takes. It is frequently needed when you want your presentation to appear as in real life.

Task Analyze the difference between model time and presentation properties

Notes

11.1.4 Windows Properties

1. ***Window properties*** define the appearance of the presentation window, that will be shown, when the user starts the experiment.
2. ***Title***: The title of the presentation window.
3. ***Enable panning***: If selected, the user will be allowed to pan the presentation window.
4. ***Enable zoom***: If selected, the user will be allowed to zoom the presentation window.
5. ***Width***: [Enabled if **Maximized size** is not selected] The width of the presentation window. If **Maximized size** option is selected, this setting makes no sense, since the window will be maximized.
6. ***Height***: [Enabled if **Maximized size** is not selected] The height of the presentation window. If **Maximized size** option is selected, this setting makes no sense, since the window will be maximized.
7. ***Maximized size***: If selected, the presentation window will be maximized on model launch.
8. ***Show Toolbar sections*** properties section defines, what sections of the toolbar of the presentation window are visible. To make some toolbar section visible, just select the corresponding checkbox.
9. ***Show Statusbar sections*** properties section defines, what sections of the status bar of the presentation window are visible. To make some status bar section visible, just select the corresponding checkbox.

11.1.5 Parameters Properties

Parameters properties are available only when the root active object class of the experiment has any parameters. Here you can define actual values for these parameters using expressions. However, if you want just to initialize a parameter with a static value, we recommend you to use controls on the General property page of the experiment.

11.2 Length of Simulation Run

To design a stochastic simulation experiment, it is caring to have an estimate of the simulation run lengths required to achieve desired statistical precision. Preliminary estimates of required run lengths can be obtained by approximating the stochastic model of interest by a more elementary Markov model that can be analyzed analytically. When steady-state quantities are to be estimated by sample means, we often can estimate required run lengths by calculating the asymptotic variance and the asymptotic bias of the sample mean in the Markov model.

To design a stochastic simulation experiment, it is helpful to have an estimate of the simulation run lengths required to achieve desired statistical precision. Preliminary estimates of required run lengths can be obtained by approximating the stochastic model of interest by a more elementary Markov model that can be analyzed analytically. When steady-state quantities are to be estimated by sample means, we often can estimate required run lengths by calculating the asymptotic variance and the asymptotic bias of the sample mean in the Markov model.

Simulation experiments are like exploring trips. We usually have initial goals, but the interesting discoveries often come from the unexpected. We typically do not know in advance precisely how we will proceed and we cannot predict all the benefits. In reality, most simulation experiments are sequences of experiments, with new goals being based on successive discoveries; see Albin (1984). Thus, there obviously is a limit to what can be planned; nevertheless simulation

Notes

planning is our concern. One reason is that we have to expect something before we can recognize the unexpected. The purpose of this paper is not so much to suggest that we look before we leap, but to suggest a few things that we might look at. The context is a simulation of a stochastic model to estimate steady-state quantities of interest. Our idea is to develop some expectations and design the initial experiment by doing some preliminary mathematical analysis. We focus on simulation run lengths. Of course, since we are going to simulate, the stochastic model of interest presumably is relatively complicated, so that it is not easy to calculate the quantities of interest analytically. Thus, we suggest approximating the stochastic model of interest by a more elementary stochastic model that can be analyzed analytically. For the approximating model, in addition to the steady-state quantity of interest, we calculate the asymptotic variance and the asymptotic bias of the sample mean used to estimate the steady-state quantity of interest (assuming that a sample mean will be used). Then we apply these quantities to estimate the simulation run lengths required in the original model to obtain desired statistical precision. The estimated simulation run lengths can then be used, before any data have been collected, to design the experiment, i.e., to determine what cases to consider, what statistical precision to aim for, what experimental budget is appropriate, or even whether to conduct the experiment at all. There are two steps: First, we must find a suitable approximation for the given stochastic model and, second, we must calculate the asymptotic quantities of interest for the approximating model. Of course, we do not want the preliminary analysis to be harder than doing the experiment itself. The preliminary analysis better be easy or we wouldn't bother with it. Fortunately, there is substantial literature supporting these two steps. In Whitt (1989a) we show how this program can be carried out for a large class of stochastic models. The models considered are those for which the stochastic process of interest can be approximated by reflecting Brownian motion (RBM). Through much previous work on heavytraffic limit theorems and diffusion approximations, it is known that many queueing processes can be approximated by RBM, at least roughly, so that the class of models covered is relatively large. The class includes the standard GI/G/1 queue, as was shown previously in related work by Blomqvist (1969), Moeller and Kobayashi (1974) and Woodside, Pagurek and Newell (1980), but also applies to many other models. In Whitt (1989a) we also show how to calculate the asymptotic quantities of interest for RBM to obtain very simple approximate formulas. Moreover, we show that the scaling of time in the heavy-traffic limit theorem plays an essential role in determining the form of the final formulas.

Of course, RBM is by no means the only stochastic process that can be used as an approximation. For example, the Ornstein-Uhlenbeck diffusion process is a natural candidate for infinite-server queues, which also yields very simple formulas. In Whitt (1989b) we apply recurrent potential theory for Markov processes to obtain asymptotic formulas for a large class of Markov processes, including general birth and death processes and diffusion processes. In Heidelberger and Whitt (1989) we compare the asymptotic formulas, and thus the simulation efficiency, for several different queueing models. There we show that it usually is easier to obtain reliable estimates for an infiniteserver queue than for a single-server queue. It usually is even easier for small closed queueing networks. Roughly speaking, simulation efficiency increases (estimation becomes easier) as the relaxation time (the time required to reach steady state; see Keilson (1979)) decreases. Our purpose here is to provide a brief overview. We review the standard statistical theory leading up to the large-sample formula for the required simulation run length with the relative width criterion; i.e., the run length such that the width of the confidence interval divided by the quantity being estimated (which is presumed to be positive) takes some prescribed value. In Section 3 we review some of the RBM-type examples from Whitt (1989a), including the M/M/1 queue, RBM, the GI/G/m queue and a packet queue model from Fendick, Saksena and Whitt (1989). The packet queue model is relatively complicated, so that an exact analysis is evidently not possible with current methodology. However, simple formulas to determine appropriate simulation run lengths can be obtained from an RBM approximation.

Notes

These examples show that it can be very misleading to do an exploratory simulation with one set of parameter values to determine appropriate simulation run lengths, because the appropriate run lengths are very different for different traffic intensities.

Did u know? **Full form of RBM**

Reflecting Brownian Motion

11.3 Variance Reduction Techniques

In a simulation study, we are interested in one or more performance measures for some stochastic model. For example, we want to determine the long-run average waiting time, E(W), of customers in a G/G/1 queue. To estimate E(W) we can do a number of, say K , independent runs, the ith one yielding the output random variable Wi with E(Wi) = E(W). After K runs have been performed, an estimator of E(W) is given by $\overline{W} = \sum_{i=1}^{K} W_i / K$. However, if we were able to obtain a different unbiased estimator of E(W) having a smaller variance than W , we would obtain an estimator with a smaller confidence interval. In this section we will present a number of different methods that one can use to reduce the variance of the estimator $\overline{W}$. We will successively describe the following techniques:

1. Common Random Numbers
2. Antithetic Variables
3. Control Variates
4. Conditioning

The first method is typically used in a simulation study in which we want to compare performance measures of two different systems. All other methods are also useful in the case that we want to simulate a performance measure of only a single system.

11.3.1 Common Random Numbers

The idea of the method of common random numbers is that if we compare two different systems with some random components it is in general better to evaluate both systems with the same realizations of the random components. Key idea in the method is that if X and Y are two random variables, then

$$Var(X - Y) = Var(X) + Var(Y) - 2Cov(X, Y).$$

Hence, in the case that X and Y are positively correlated, i.e. Cov(X, Y) > 0, the variance of X – Y will be smaller than in the case that X and Y are independent. In general, the use of common random numbers leads to positive correlation of the outcomes of a simulation of two systems. As a consequence, it is better to use common random numbers instead of independent random numbers when we compare two different systems.

Let us illustrate the method using the following scheduling example. Suppose that a finite number of N jobs has to be processed on two identical machines. The processing times of the jobs are random variables with some common distribution function F. We want to compare the completion time of the last job, C_{max}, under two different policies. In the LPTF policy, we always choose the remaining job with the longest processing time first, in the SPTF policy we choose the remaining job with the shortest processing time first.

Notes

In Table 11.1 and Table 11.2 we compare for N = 10 and F(x) = 1 – e^{-x} the estimators and confidence intervals for $E(C_{max}^{SPTF} - C_{max}^{LPTF})$ when we do not, resp. do, use common random numbers.

We conclude that in this example the use of common random numbers reduces the standard deviation of the estimator and hence also the length of the confidence interval with a factor 5.

Table 11.1: Estimation of $E(C_{max}^{SPTF} - C_{max}^{LPTF})$ without using common random numbers

# of runs	mean	st. dev.	95% conf. int.
1000	0.8138	2.5745	[0.645, 0.973]
10000	0.8293	2.4976	[0.780, 0.878]
100000	0.8487	2.4990	[0.833, 0.864]
1000000	0.8398	2.4951	[0.835, 0.845]

Table 11.2: Estimation of E(CSPTF) using common random numbers

# of runs	mean	st. dev.	95% conf. int.
1000	0.8559	0.5416	[0.822, 0.889]
10000	0.8415	0.5230	[0.831, 0.852]
100000	0.8394	0.5164	[0.836, 0.843]
1000000	0.8391	0.5168	[0.838, 0.840]

When we want to use common random numbers, the problem of synchronization can arise: How can we achieve that the same random numbers are used for the generation of the same random variables in the two systems?

In the previous example, this synchronization problem did not arise. However, to illustrate this problem, consider the following situation. In a G/G/1 queueing system the server can work at two different speeds, v_1 and v_2. Aim of the simulation is to obtain an estimator for the difference of the waiting times in the two situations. We want to use the same realizations of the interarrival times and the sizes of the service requests in both systems (the service time is then given by the sizes of the service request divided by the speed of the server). If we use the program of the discrete event simulation of Section 3 of the G/G/1 queue, then we get the synchronization problem because the order in which departure and arrival events take place depends on the speed of the server. Hence, also the order in which interarrival times and sizes of service requests are generated depend on the speed of the server.

The synchronization problem can be solved by one of the following two approaches:

1. Use separate random number streams for the different sequences of random variables needed in the simulation.
2. Assure that the random variables are generated in exactly the same order in the two systems.

Example: The G/G/1 queue, the first approach can be realized by using a separate random number stream for the interarrival times and for the service requests. The second approach can be realized by generating the service request of a customer already at the arrival instant of the customer.

11.3.2 Antithetic Variables

The method of antithetic variables makes use of the fact that if U is uniformly distributed on (0, 1) then so is 1 – U and furthermore U and 1 – U are negatively correlated.

Notes

The key idea is that, if

W_1 and W_2 are the outcomes of two successive simulation runs, then

$$\mathrm{Var}\left(\frac{W_1+W_2}{2}\right)=\frac{1}{4}\mathrm{Var}(W_1)+\frac{1}{4}\mathrm{Var}(W_2)+\frac{1}{2}\mathrm{Cov}(W_1,W_2)].$$

Hence, in the case that W_1 and W_2 are negatively correlated the variance of $(W_1 + W_2)/2$ will be smaller than in the case that W_1 and W_2 are independent.

The question remains how we can achieve that the outcome of two successive simulation runs will be negatively correlated. From the fact that U and 1 – U are negatively correlated, we may expect that, if we use the random variables $U_1,\ldots,U_m$ to compute W_1, the outcome of the first simulation run, and after that $1 - U_1,\ldots,1 - U_m$ to compute W_2, the outcome of the second simulation run, then also W_1 and W_2 are negatively correlated. Intuition here is that, e.g., in the simulation of the G/G/1 queue large realizations of the U_i 's corresponding to large service times lead to large waiting times in the first run. Using the antithetic variables, this gives small realizations of the U_i' s corresponding to small service times and hence leading to small waiting times in the second run.

We illustrate the method of antithetic variables using the scheduling example of the previous subsection.

Table 11.3: Estimation of E(CLPTF)without using antithetic variables

# of runs	mean	st. dev.	95% conf. int.
1000	5.0457	1.6201	[4.945, 5.146]
10000	5.0400	1.6020	[5.009, 5.071]
100000	5.0487	1.5997	[5.039, 5.059]
1000000	5.0559	1.5980	[5.053, 5.059]

Table 11.4: Estimation of E(CLPTF)using antithetic variables

# of pairs	mean	st. dev.	95% conf. int.
500	5.0711	0.7216	[5.008, 5.134]
5000	5.0497	0.6916	[5.030, 5.069]
50000	5.0546	0.6858	[5.049, 5.061]
500000	5.0546	0.6844	[5.053, 5.056]

In Table 11.3 and Table 11.4 we compare, again for N = 10 and $F(x) = 1 - e^{-x}$, the estimators and confidence intervals for $E(C_{max}^{LPTF})$ when we do not, resp. do, use antithetic variables. So, for example, we compare the results for 1000 independent runs with the results for 1000 runs consisting of 500 pairs of 2 runs where the second run of each pair uses antithetic variables. We conclude that in this example the use of antithetic variables reduces the length of the confidence interval with a factor 1.5.

Finally, remark that, like in the method of common random numbers, the synchronization problem can arise. Furthermore, it should be noted that the method is easier to implement if all random variables are generated using the inversion transform technique (only one uniform random number is needed for the realization of one random variable) than if we use, e.g., the rejection method to generate random variables

Notes A random number of uniform random numbers are needed for the realization of one random number.

Notes

11.3.3 Control Variates

The method of control variates is based on the following idea. Suppose that we want to estimate some unknown performance measure E(X) by doing K independent simulation runs, the ith one yielding the output random variable X_i with $E(X_i) = E(X)$. An unbiased estimator for E(X) is given by $\overline{X} = \left(\sum_{i=1}^{K} X_i\right)/K$. However, assume that at the same time we are able to simulate a related output variable Y_i, with $E(Y_i) = E(Y)$, where E(Y) is known. If we denote by $\overline{Y} = \left(\sum_{i=1}^{K} Y_i\right)/K$, then, for any constant c, the quantity $\overline{X} = c\left(\overline{Y} - E(Y)\right)$ is also an unbiased estimator of E(X). Furthermore, from the formula

$$\mathrm{Var}(\overline{X} + c(\overline{Y} - E(Y)) = \mathrm{Var}(\overline{X}) + c^2\mathrm{Var}(\overline{Y}) + 2c\mathrm{Cov}(\overline{X}, \overline{Y}).$$

it is easy to deduce that $\mathrm{Var}\left(\overline{X} + c\left(\overline{Y} - E(Y)\right)\right)$ is minimized if we take c =c*, where

$$c^* = -\frac{\mathrm{Cov}(\overline{X}, \overline{Y})}{\mathrm{Var}(\overline{Y})}.$$

Unfortunately, the quantities $\mathrm{Cov}(\overline{X}, \overline{Y})$ and $\mathrm{Var}(\overline{Y})$ are usually not known beforehand and must be estimated from the simulated data. The quantity $\overline{Y}$ is called a control variate for the estimator $\overline{X}$. To see why the method works, suppose that $\overline{X}$ and $\overline{Y}$ are positively correlated. If a simulation results in a large value of $\overline{Y}$ (i.e. $\overline{Y}$ larger than its mean E(Y)) then probably also $\overline{X}$ is larger than its mean E(X) and so we correct for this by lowering the value of the estimator $\overline{X}$. A similar argument holds when $\overline{X}$ and Y are negatively correlated.

Caution In the simulation of the production line of section 2, a natural control variate would be the long-term average production rate for the line with zero buffers.

11.3.4 Conditioning

The method of conditioning is based on the following two formulas. If X and Y are two arbitrary random variables, then E(X) = E(E(X | Y)) and Var(X) = E(Var(X | Y)) +Var(E(X | Y)) ≥ Var(E(X | Y)). Hence, we conclude that the random variable E(X | Y) has the same mean as and a smaller variance than the random variable X.

How can we use these results to reduce the variance in a simulation? Let E(X) be the performance measure that we want to estimate. If Y is a random variable such that E(X | Y = y) is known, then the above formulas tell us that we can better simulate Y and use E(X | Y) than that we directly simulate X.

The method is illustrated using the example of an Mλ/Mμ/1/N queueing model in which customers who find upon arrival N other customers in the system are lost. The performance measure that we want to simulate is E(X), the expected number of lost customers at some fixed time t. A direct simulation would consist of K simulation runs until time t. Denoting by Xi the number of lost customers in run i,then $\overline{X} = (\sum_{i=1}^{K} X_i)/K$ is an unbiased estimator of E(X). However, we can reduce the variance of the estimator in the following way. Let Y_i be the total

Notes

amount of time in the interval (0, t) that there are N customers in the system in the ith simulation run. Since customers arrive according to a Poisson process with rate λ, it follows that $E(X|Yi) = \lambda Y_i$. Hence an improved estimator would be $\lambda \overline{Y}$, where $\overline{Y} = \left(\sum_{i=1}^{K} Y_i\right)/K$.

In Table 11.5 and Table 11.6 we compare, for $\lambda = 0.5$, $\mu = 1$, N = 3 and t = 1000, the estimators and con?dence intervals for E(X) when we do not, resp. do, use conditioning. We conclude that

Table 11.5: Estimation of E(X) without using conditioning

# of runs	mean	st. dev.	95% conf. int.
10	33.10	10.06	[26.86, 39.33]
100	34.09	9.21	[32.28, 35.90]
1000	33.60	8.88	[33.05, 34.15]

Table 11.6: Estimation of E(X) using conditioning

# of runs	mean	st. dev.	95% conf. int.
10	33.60	7.16	[29.17, 38.05]
100	33.82	6.91	[32.46, 35.18]
1000	33.36	6.57	[32.95, 33.76]

in this example the use of conditioning reduces the standard deviation of the estimator and hence also the length of the con?dence interval with a factor 1.3.

Caselet

KMA to Hold Business Environment Simulation Exercise

Kerala Management Association is organising a stimulating and enriching Business Environment Simulation Exercise exclusively for B School Students on Saturday at Management House in Panampilly Nagar.

Budding mangers and entrepreneurs are required to research and prepare a sound business plan.

The contestants should take the role of an entrepreneur who wants to venture into the business to increase the investment culture in stocks and trading in Kerala.

Mr Jibu Paul, Chairman, KMA Students Forum said that a team could consist of maximum 5 members. The plan includes a role-play model presentation, giving proper designations and roles to each team member.

Each team is expected to make a power point presentation for five minutes first and if selected then for 20 minutes finally.

For more details contact the office of Kerala Management Association @ 0484 2317917 or info@kma.org.in.

The winner will get ₹ 10,000 cash award and certificates; First Runner up will get ₹ 7,500 cash award and certificates; second runner up will get ₹ 5,000 cash award and certificates.

The competition is sponsored by ACUMEN.

The awards will be distributed at the KMA Annual Convention inaugural day by the Chief Guest Dr Sam Pitroda on February 3 at Hotel Le Meridian.

Notes

11.4 Summary

- When a new project is formed, one simulation experiment, called Simulation, is automatically created. You can create your own experiments and set up any simulation settings you like.
- To design a stochastic simulation experiment, it is caring to have an estimate of the simulation run lengths required to achieve desired statistical precision.
- In mathematics, more purposely in the theory of Monte Carlo methods, variance reduction is a procedure used to increase the precision of the estimates that can be obtained for a given number of iterations.

11.5 Keywords

Fixed Seed (Reproducible Simulation Runs): The seed value of the random number generator is fixed (specify it in the Seed value field).

Random Seed (Unique Simulation Runs): The seed value of the random number generator is random.

Relative Accuracy: The desired relative value accuracy for solving equations with methods that change the integration step.

Time Accuracy: The desired time accuracy for finding change events (switch points) when solving equations.

11.6 Self Assessment

Fill in the blanks:

1. Specifying the fixed seed value, you initialize the model …….. with the same value for each model run, thus the model runs are reproducible.
2. The desired relative value …….. for solving equations with methods that change the integration step.
3. The greater update rate you specify, the …….. presentation will appear.
4. Simulation experiments are like …….. trips.
5. The …….. queue model is relatively complicated, so that an exact analysis is evidently not possible with current methodology.
6. Common random numbers, antithetic variates, …….., importance sampling and stratified sampling.
7. Stochastic models require a random seed value for the …….. number generator.
8. In random seed value of the random number generator is ……..
9. The default random number generator is an instance of the Java class Random, which is a ……..
10. The initialization of the default …….. occurs during the initialization of the experiment and then before each simulation run.
11. …….. is a process of smoothing the drawing of points or lines that would otherwise appear jagged.
12. If the …….. frame management is turned on, specify here the ratio between simulation speed and presentation smoothness.

Notes

13. The greater update rate you specify, the smoother …….. will appear.
14. The model runs at its maximum speed and no mapping is made between model time and ……..
15. If …….. is selected, the presentation window will be maximized on model launch.

11.7 Review Questions

1. Do you think parameters properties are available only when the root active objects class of the experiment has any parameters?
2. "A variance reduction technique is presented that is effective when the system characteristic to be estimated is strongly influenced by rare events." Comment.
3. What are the steps required to make the simulation experiment?
4. If for any reason you are not satisfied with the quality of Random then what you can do? Give reasons.
5. Explain the general properties of simulation experiment.
6. Do you think the Maximum available memory is the maximum size of Java heap allocated for the model?
7. "The code is executed before simulation run." Elaborate.
8. Absolute accuracy is used when it is impossible to use relative accuracy. Give reasons
9. Find out the difference between the model time properties and presentational properties of simulation experiment.
10. "Parameters properties are available only when the root active object class of the experiment has any parameters." Explain.

Answers: Self Assessment

1.	random number generator	2.	accuracy
3.	smoother	4.	exploring
5.	packet	6.	control variates
7.	pseudorandom	8.	random
9.	Linear Congruental Generator (LCG)	10.	RNG
11.	Anti-aliasing	12.	adaptive
13.	presentation	14.	real time
15.	maximized size		

Notes

11.8 Further Readings

Books

Balci, O.,(1994), Validation, verification, and testing techniques throughout the life cycle of a simulation study, Annals of Operations Research, 53: 121–173.

Balci, O., (2001), A methodology for certification of modeling and simulation applications, ACM Transactions on Modeling and Computer Simulation, 11: 352–377.

Birta, L.G. and Ozmizrak, N.F., (1996), A knowledge-based approach for the validation of simulation models: The foundation, ACM Transactions on Modeling and Computer Simulation, 6: 67–98.

Boehm, B.W., (1979), Software engineering: R&D trends and defense needs, in: P. Wegner (Ed.), Research Directions in Software Technology, MIT Press, and Cambridge, MA.

Cellier, F.E., (1986), Combined discrete/continuous system simulation – application, techniques and tools, in Proceedings of the 1986 Winter Simulation Conference.

Department of Defense (DoD) Recommended Practices Guide (RPG) for Modeling and Simulation VV&A, Millennium Edition (available at http://vva.dmso.mil).

General Accounting Office, (1976), Report to the Congress: Ways to improve management of federally funded computerized models, report LCD-75-111, U.S. General Accounting Office, Washington, DC.

General Accounting Office, (1979), Guidelines for model evaluation, report PAD-79-17, U.S. General Accounting Office, Washington, DC.

General Accounting Office, (1987), DOD simulations: Improved assessment procedures would increase the credibility of results, report GAO/PEMD-88-3, U.S. General Accounting Office, and Washington, DC.

Ören, T.I., (1981), Concepts and criteria to access acceptability of simulation studies, Communications of the ACM, 24: 180–189.

Online links

http://portal.acm.org/citation.cfm?id=76751

http://en.wikipedia.org/wiki/Variance_reduction

Notes

Unit 12: Design and Evaluation of Simulation Experiments (II)

CONTENTS

Objectives
Introduction
12.1 Experimental Layout
12.2 Validation
12.2.1 Qualitative and Quantitative Simulation
12.2.2 Explanation and Prediction
12.2.3 Types of Validity
12.3 Analysis for the Design of Simulation Experiments
12.3.1 The Standard Statistical Framework
12.3.2 The Asymptotic Parameters for a Function of a Markov Chain
12.3.3 Birth-and-Death Examples
12.3.4 Diffusion Processes
12.3.5 Stochastic-Process Limits
12.4 RBM Approximations
12.5 Summary
12.6 Keywords
12.7 Self Assessment
12.8 Review Questions
12.9 Further Readings

Objectives

After studying this unit, you will be able to:

- Describe Experimental Layout
- Discuss simulation validation

Introduction

Careful planning, or designing, of simulation experiments is generally a great help, saving time and effort by providing efficient ways to estimate the effects of changes in the model's inputs on its outputs. Validation is concerned with determining whether the conceptual simulation model itself is accurate representation of the system under study.

Notes

12.1 Experimental Layout

Validation of design rules taking into account fine details such as line-edge roughness, and full chip layout simulation for design inconsistencies, before actual fabrication, are among the main objectives of current software assisted metrology tools. Line-edge roughness quantification should accompany Critical Dimension (CD) measurements since it could be a large fraction of the total CD budget. A detailed simulation and metrology approach of line-edge roughness quantification versus the length scales in a layout are presented in this work using a combination of electron beam simulation for the exposure part, and stochastic simulations for the modeling of resist film, postexposure bake, and resist dissolution. The method is applied also on a test layout with critical dimension of 200 nm and the resulted simulation and scanning electron microscopy images are compared with the aid of a pattern matching algorithm which enables the identification of the desired layout for metrology on a complex layout containing many printed features.

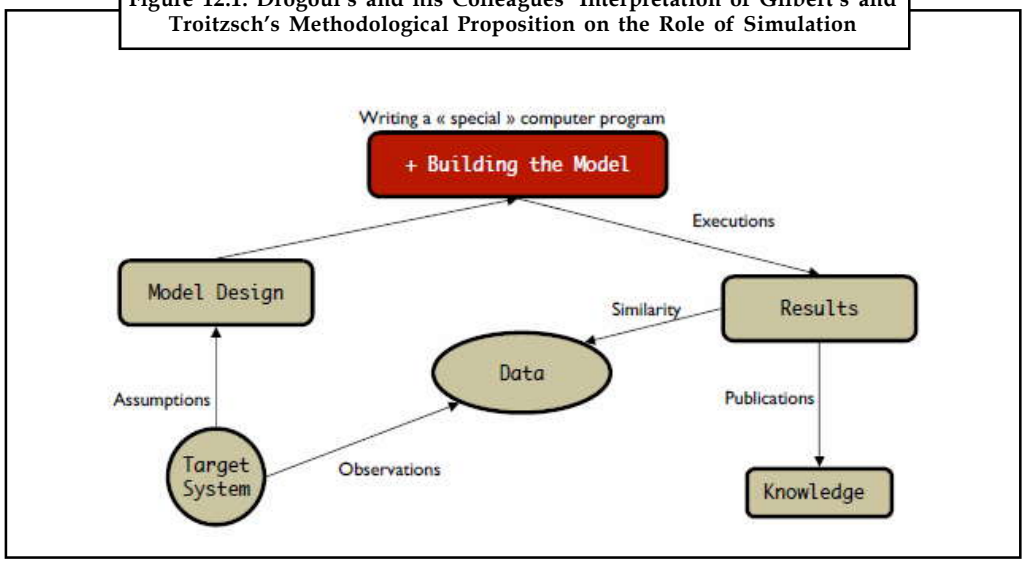
12.2 Validation

It starts with a short review of arguments used in the simsoc mailing list discussion on theory, simulation and explanation a few months ago, deals with the use of quantitative and qualitative computational models to make quantitative and qualitative predictions or rather to draw conclusions from complex antecedents, and then discusses different types of explanation and prediction (and the relation between these two), It closes with an overview of topics in validity and validation from the point of view of the structuralist programme in the philosophy of science.

A few months ago, the simsoc mailing list experienced a longish discussion1 which originated from Thomas Kron's question "about the relation of computer simulation and explanation, especially sociological explanation". More than fifty contributions to this discussion followed within less than three weeks, and contributors discussed the role of simulation in theory building (mostly, but not only) in the social, economic and management sciences—as well as the relation between observation on one hand and computer-assisted theory building (Hanneman 1988) on the other. Scott Moss came back to his presidential address at the 1st conference of the European Social Simulation Association, Groningen, September 2003, in which he said "that if social simulation with agents is to be anything other than another in the long line of failed approaches to social science, it will be a positive departure only because it facilitates the dominance of observation over theory" and continued that the great successful scientists (outside the social sciences) built their generalisations around observation, developing new theoretical structures based on and validated by new evidence (quoted from his contribution to the simsoc mailing list as of November 14, 2003). Well in the line of this trait of thinking is the role of simulation or computational modeling which can be found in Gilbert and Troitzsch 1999 which was recently extended by Alexis Drogoul (Drogoul et al. 2003: 5) and can be seen in Figure 12.1.

Notes

Figure 12.1: Drogoul's and his Colleagues' Interpretation of Gilbert's and Troitzsch's Methodological Proposition on the Role of Simulation



This diagram does not even contain the word 'simulation', but in its centre we find 'data' which are taken from observation and compared with results from simulation runs, for their similarity. Gilbert's and Troitzsch's original diagram describing "the logic of simulation as a method" (Gilbert and Troitzsch 1999: 16, 54, see also Troitzsch 1990: 2) is much the same: A model is built by abstraction from a target system, it is translated into a computer programme which can then be run and delivers results in the form of simulated data which can, and have to, be compared to data gathered from the same kind of target systems in the real world from which the model was abstracted.

Being aware that observation (as contrasted to just looking around in the world) presupposes at least some primitive form of theory (which tells us which entities and which of its properties to observe and which relations between them to register to find out whether there are some regularities), we should admit that our assumptions and our observation are not independent from each other (although Figure 5.1 insinuates this). And we should admit that in most cases computational (and other) models do not directly start from observation data but from a theory which in turn should build on, but often does not refer explicitly to observation data. Instead, we often start from a verbal theory which expresses our (or other authors') belief in how reality works, comparing simulation results with stylised facts instead of observation data. A good example of this strategy is Sugarscape where the question "can you explain it?" is interpreted as "can you grow it?", and where "a given macrostructure [is] 'explained' by a given microspecification when the latter's generative sufficiency has been established." (Ep- stein and Axtell 1996: 177)

At the other extreme, we might have microanalytical simulation which starts from a large collection of observational data on persons and households and the population as a whole. The model is initialised with empirical estimates of transition probabilities, age-specific birth and death rates and so on. Tens of thousands of software agents are created with data from real world people. And all this aims at predicting something like the age structure or kinship networks of this empirical poulation in the far future (see for instance Harding 1996).

Notes

In what follows we want to discuss the use of quantitative and qualitative computational models to make quantitative and qualitative predictions or rather to draw conclusions from complex antecedents and discuss different types of explanation and prediction (and the relation between these two) and close with an overview of topics in validity and validation.

Task Analyze the difference between validity and validation.

12.2.1 Qualitative and Quantitative Simulation

Although most simulation uses quantitative procedures — doing calculations with numbers, often real valued, which make believe that the properties of the target system are quantitative, metric properties —, most of our mental models and verbal theories which are the predecessors of most of our simulation programmes do not talk about numbers and numerical values, but rather of properties which are categorical or, at best, ordinal. "However we claim that the use of numbers in this way is often simply a result of laziness — we often use numbers as a stand-in for qualitative aspects that we do not know how to program or have not the time to program." (Edmonds and Hales 2003: 3)

Example: Gender desegregation among staffs of schools

The following example — which is taken from (Gilbert and Troitzsch 1999: 108–114) and earlier papers — tries to "explain" how the process of overcoming gender segregation in German schools went on in the 1950s and 1960s. The modeling process started from a large collection of empirical data showing the proportion of male and female teachers in all grammar schools in the federal state of Rhineland-Palatinate (approximately 150 in number) from 1950 to 1990.

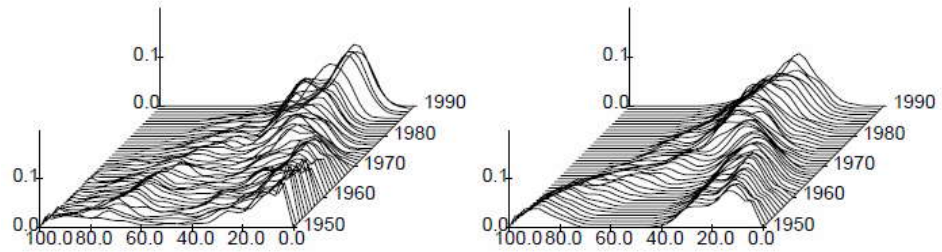
The model reproducing the empirical distribution of this proportion over time quite well was designed as parsimonious as possible, just assuming three hypotheses:

1. That all teachers leaving their jobs are replaced by men and women with equal overall probability (Article 2 linea 2 of the German Basic Law),
2. That men stay in their jobs approximately twice as long as women (an empirical observation), and
3. That new women are assigned to an individual school with probability $P(W|\xi) = \vee(t)\exp(\kappa\xi)$ according to the percentage ξ of women among its teachers (a theoretical assumption); ??is 0.5 in this simulation run, and ?(t) is such that at all times men and women have the same overall probability of replacing retired teachers, to comply with hypothesis 1.

The simulation is initialized with a gender distribution close to the empirical distribution of 1950.

Notes

Figure 12.2: Distribution of percentages of women among teachers at 150 secondary schools in Rhineland- Palatinate from 1950 to 1990; left: empirical data, right: simulation



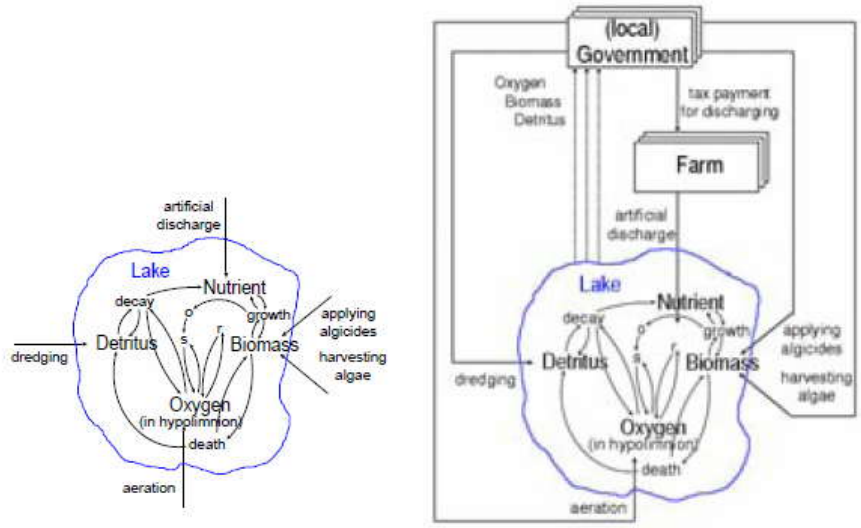
The simulation model reproduced the qualitative result that in the early 1970s the staff of all these 150 schools became mixed after twenty years of segregation where there were schools with high proportions of either male or female teachers but nearly no schools with between 40 and 60 per cent female teachers. And this reproduction / retrodiction was effected with the help of quantitative simulation, calculating probabilities of assigning teachers to schools. But did the model explain how and why this happened? Obviously not—since it is clear that the school authority, in fact officers in the ministry of education, did not cast dice or draw random numbers to select candidates for particular schools. Perhaps these officers saw to it that the overall proportion of men and women in school staffs was sufficiently equal to give women an equal chance, but even this has not been observed — instead we know that the process of desegregation of school staffs had entirely different origins: it was only the consequence of desegregation among girls and boys which in turn was due to the fact that most small towns could not afford separate schools for boys and girls (the percentage of girls in grammar schools rose steeply in the 1950s and 1960s). To summarise: a nice prediction (or at least retrodiction), but a poor explanation.

Example: Artificial eutrophication of a lake

Another example which is at the borderline between quantitative and qualitative simulation is the following. It was derived from a purely quantitative System Dynamics simulation in the tradition of Meadows and Forrester (Anderson 1973) which was used to quantitatively predict the consequences of bringing fertiliser into the soil in the neighbourhood of a lake and of actions taken to avoid these consequences by, for instance, harvesting algae or dredging the ground of the lake. This was, as it were, a simulation machine to predict the outcomes of real-world experiments or perhaps to replace such experiments. Anderson's model was not designed to predict how farmers, fishers, tourist offices, local authorities around the lake would act when they realised that dead fish was swimming on the surface of the lake or when its water reeked of decay: this was only introduced in a revised model where local authorities — modelled as software agents — could decide which action to take when they were informedabout the state of the lake, and where local farmers — also modelled as software agents — could decide whether it was more profitable for them to pay taxes for using artificial fertiliser on their fields and to grow more crop or to waive fertilising, not to pay fertiliser taxes and to be satisfied with lower yield (M¨ohring and Troitzsch 2001; see Figure 12.3).

Notes

Figure 12.3: Two versions of Anderson' model of the eutrophication of a lake (Anderson 1973); left: a model of a lake subject to simulation experiments, right: a model of the lake and its (agri-) cultural environment



The difference between the two approaches is twofold:

1. In the original approach Anderson took a formal model of the physical, chemical and biological processes running in a lake to simulate what could happen if these processes were disturbed by external influences imposed on the model by the simulating experimenter — whereas the extended model embeds the model of a lake into its social, political and economic environment and models influences external to the lake as internal to the model, thus taking into account that the lake and its socioeconomic environment interact and are interdependent.

2. The original approach starts from physical, chemical and biological theory providing the equations between the main variables describing the lake and generates quantitative simulation results (predictions) which might be compared to further observation data and help improve (fine tune) the theory of the biochemical processes occurring in a lake — whereas the extended model takes the behaviour of the lake for granted and adds a number of assumptions about the behaviour and actions of a number of human actors which are (or at least could have been) based on everyday observation and discussions with stakeholders, and these assumptions are not aimed at generating quantitative predictions about the effect of the tax rate imposed on artificial fertilisers on the oxygen concentration in the lake, but only in qualitative predictions answering the question under which tax regime and under which additional measures taken by government and other neighbours of the lake its eutrophication could or could not be avoided.

Both approaches would serve as explanatory models: the restricted model would explain how and why under certain external or internal circumstances a lake would be eutrophicated and suffocate and what was the physical and biochemical mechanism behind the processes leading to total consumption of oxygen at the ground of the lake —and it would at the same time recommend countermeasures and allow for an estimation for their potential success; and the extended model would explain under which conditions such countermeasures would be taken by the population living around the lake and what incentives one part of the population would have to promise another part of the population to take the necessary countermeasures.

Notes

12.2.2 Explanation and Prediction

There is a long discussion about the question whether explanation and prediction are equivalent, or, to put it in other words, whether a theory which predicts empirical observations correctly at the same time explains what it predicts. Gr¨unbaum (1962) pleaded for the equivalence while Scriven (1969) pleaded that both were "nonsymmetrical".

If we consider prediction and explanation equivalent then our first example above would have explained the gender desegregation in German schools observed in the second half of the 20th century (although this was only retrodiction, but in principle, the three assumptions could have been stated in 1950), but this explanation is of the same quality as the explanation

Mesopotamian priests could give 2,500 years ago for their (mostly correct) predictions of solar eclipses. In both cases, some scepticism in in order: from our research into the history of school staffs we know that desegregation had different causes than those stated in the assumptions, and the Mesopotamian theories of planetary movements were superseded 400 years ago by new theories which are substantially more valid.

The controversy between Gr¨unbaum and Scriven, however, was different: Scriven had argued the other way round: "Satisfactory explanation of the past is possible even when prediction of the future is impossible." (Scriven 1969: 117; Gr¨unbaum 1962: 126) while we argued above that even when prediction of the future is possible with the help of a theory, this does not mean that this theory satisfactorily explains what happened (another theory could yield the same prediction and deliver a better explanation).

Without going into the details of this old controversy we should instead discuss what explanation and prediction could mean in the context of (social) simulation. Epstein and Axtell argued that explanation of a phenomenon is achieved once the phenomenon could be reconstructed or generated ("grown"). From this point of view, the development of the distribution of percentages of female teachers in German grammar schools is explained by the three assumptions mentioned above, since this time-dependent frequency distribution as a macrostructure could be reconstructed quite well from the microstructure defined in the three assumptions. Of course, this reconstruction is by no means quantitatively precise: the two graphs are similar, but not identical (perhaps due to some simplifications in the assumptions, perhaps due to the fact that the random number generator in the simulation run which generated the time-dependent frequency distribution was not perfect, or for any other reasons) — and, of course, Sugarscape explanations are of the same, nonquantitative type.

What simulation models like these are designed to predict is only how a target system might behave in the future qualitatively; what we want to know is whether any macrostructures might be observed and what these macrostructures might look like, given that on a micro level some specific rules are applied or some specific laws hold. This is what we should call a qualitative prediction which at best would tell us that a small number of categorical outcomes can be expected with their respective probabilities. But this is not the type of prediction as the objective of simulation "which most people think of when they consider simulation as a scientific technique" (Axelrod 1997: 24)—"most people think of" attempts at simulating planetary formation (Casti 1996: 14) instead of "simulating the movement of workers or armies". But if we use prediction in a non-quantitative sense, predictions delivered by simulations might still be useful "for the discovery of new relationships and principles" which Axelrod finds "at least as important as proof or prediction". They might answer questions like "Which kinds of macro behaviour can be expected from a given micro structure under arbitrarily given parameter combinations and initial conditions?" The definition of this micro structure will typically be derived from observations on the micro level, and the simulated macro structures will typically be compared to macro structures in the target systems (which perhaps have not even be discovered).

Notes

And a simulation model which generates a macro structure which resembles real-world macro structures from simulated micro structures which resemble micro structures observable in the real world might be accepted as a provisional explanation of real-world macro structures.

In a second step we might apply simulation to proceed to a second stage of qualitative prediction, where we are not interested in the general behaviour of a certain *class* of target systems, but in the future behavior of a particular *instance* of this class of target systems — say, the future market shares of a number of competing products in a market, trying to answer the question whether most trademarks will survive with reasonable market shares or whether most of them will survive only in small niches whereas one product will gain an overwhelming share of the whole market; this would still be a qualitative answer: we might not be interested in which trademark will be the winner, and we might not be interested in how many per cent of the market it will win (this would be only the third use of simulation, namely to predict quantitatively and numerically, as in microanalytical simulation and, perhaps, also in the simulation of climatic changes where we would not be content with the outcome that mean temperatures will rise but wanted to know when, where and how fast this process would have effects on nature and society). Or, to return to the example of the lake, its eutrophication and the countermeasures taken by its neighbours, we would

1. First apply simulation to the very general question whether an artificial society "living" around an artificial lake which functions much like an empirical lake could ever learn to avoid eutrophication (something like a tragedy-of-the-commons simulation),
2. Then apply simulation to an empirical setting (describing and modelling an existing lake and its surroundings) to find out whether in this specific setting the existing lake can be rescued, and
3. Eventually to apply simulation to the question which political measures have to be taken to make the lake neighbours organise their economy in away that the best possible use is made of the lake — and obviously this would be a discursive model in which stakeholders should be involved to negotiate and find out what "best possible use" actually means for them.

Notes Note that to involve stakeholders in the development of a simulation model like this it will be necessary to validate the model— otherwise stakeholders would not believe it was worthwhile to work with the simulation model.

12.2.3 Types of Validity

With Zeigler we should distinguish between three types of validity:

1. ***Replicative Validity:*** The model matches data *already acquired* from the real system (retrodiction).
2. ***Predictive Validity:*** The model matches data *before* data are acquired from the real system.
3. ***Structural Validity:*** The model "not only reproduces the observed real system behaviour, but truly reflects the way in which the real system operates to produce this behaviour." (Zeigler 1985: 5).

Zeigler here addresses three different stages of model validation (and development). Social science simulation does not seem to have followed this path in all cases:

Since often data are very poor in the social sciences, early models, too, tried to be structurally valid and did not bother much about replicative or predictive validity.

Notes

"Data already acquired from the real system" were not available in a form that could be matched to the bulk of data simulation models had generated. There are several reasons for this difference between natural and social sciences:

Data collection is a very expensive task in the latter, and in most cases it is even impossible to generate long time series for individual or group behaviour — individual attitudes, e.g., may be changed by the very measurement process, and groups may have changed in their composition before they were able to generate a time series which would have been long enough to allow for parameter estimation. On the other hand, the different kinds of influences non-living things exact upon each other are very much limited in their number, such that a structurally valid model can much more easily be found for the target systems natural sciences deal with than for social systems.

When talking about structural validity, a digression on structuralism might be in order: Structuralism as defined by Sneed (1979) and Balzer *et al.* (1987) sees both simulation models and observations as models of a theory which in turn — for them — is a mathematical structure consisting of (among others) three sets of such models.

And these models — full models, potential models, and partial potential models — are defined as lists of terms and functions and (in the case of full models) invariants.

Observations in this structuralist programme in the philiosophy of science are intended applications of a theory, they are a subset of the set of its partial potential models in a sense that we can talk about them in terms which are non-theoretical with respect to a theory **T** in question ("T-non-theoretical terms", for short). Elsewhere it was shown that a simulation model *"of a theory"* is "analogous to a structuralist reconstruction of this theory", and that such reconstructions can easily be translated into simulation models and vice versa (Troitzsch 1994), provided the simulation language is object-oriented and functional (in other simulation languages the translation might be less straightforward). Simulation models would then be translated into full models in so far as they contain both T-non-theoretical terms (those we can use for talking about the target system irrespective of whether the theory is validated or not) and its T-theoretical terms — those which are only introduced by the theory, "in the sense that their meaning depends on **T**", (Balzer *et al.* 1987: 40) — and, thirdly, the axioms or invariants the theory postulates — whereas observations (or rather: intended applications, to keep to the terminology of structuralism) are only partial potential models listing just the terms which are non-theoretical with respect to this theory.

Thus, simulation is "richer" than observation.

Validation of simulation models is thus the same (or at least analogous) to validation of theories. In the sense of structuralism, we can interpret validation as the attempt at finding whether there exist intended applications of a theory (observations to which the theory refers) which belong to the content of the theory—which means that it should be possible to make an observation (in Tnon Tnon-theoretical terms) which complies with the axioms of the theory (which in turn might be expressed in Ttheoretical terms, but then these must be linked to T-nontheoretical terms).

What does this mean for agent-based simulations in the range defined in the introduction? Sugarscape agents and plants correspond to T-theoretical terms, and the rules which the agents obey correspond to the axioms of this theory. But is there any empirical claim of the theory behind Sugarscape? If this theory predicts that — with a given parameterisation and initialisation — macrostructures emerge from the microstructures programmed into "its" models, and the emerging macrostructures sufficiently resemble observable macrostructures, we could admit that this observable macrostructure together with its microstructure (provided it resembles the model's microstructure) is an intended application of the theory behind Sugarscape and that it complies with its axioms.

Notes

In the case of the empirical examples sketched above, the case is even simpler. Our model of a lake and its socioeconomic environment was based on observation, but it would still contain a number of terms which can only be used within a theory of, say, ecological consciousness:

There would be some link between the state of the lake (its smell or colour) and the state of ecological consciousness of a particular person living near the lake (something like "the worse the water smells, the more am I willing to protect the lake from sewage") and the action this person takes, and we could only observe the direct link between the observable smell of the lake and the observable actions taken, so the two "internal" links (as functions with their numerical coefficients, or as fuzzy rules with their membership functions) would remain theroetical with respect to such a theory — but the computer programme used for this simulation would still be a full model of this theory, because it would contain a function or rule representing this link, and that part of the simulation output which could be compared to empirical observational data would be the partial potential model of the theory.

Stakeholders, however, might find that the T-theoretical links between the observable state of the lake and the observable actions on one hand and the T-theoretical state of ecological consciousness comply with what they think how ecological consciousness (if ever such a thing exists) works. And this could be the special value simulation could have in participatory modelling approaches (cf. the last few paragraphs of El hadouaj *et al.* 2001).

12.3 Analysis for the Design of Simulation Experiments

The traditional (important) methods to design statistical experiments, but rather techniques that can be used, before a simulation is conducted, to estimate the computational effort required to obtaindesired statistical precision for contemplated simulation estimators. In doing so, we represent computational effort by simulation time, and that in turn by either the number of replicationsor the run length within a single simulation run. We assume that the quantities of interest will be estimated by sample means. In great generality, the required length of a single simulation run can be determined by computing the asymptotic variance and the asymptotic bias of thesample means. Existing theory supports this step for a sample mean of a function of a Markovprocess. We would prefer to do the calculations directly for the intended simulation model, but that usually is prevented by model complexity. Thus, as a first step, we usually approximatethe original model by a related Markovian model that is easier to analyze. For example,relatively simple diffusion-process approximations to estimate required simulation run lengthsfor queueing models can often be obtained by heavy-traffic stochastic-process limits.

Simulations are *controlled experiments*. Before we can run a simulation program and analyze the output, we need to choose a simulation model and decide what output to collect; i.e., we need to *design* the simulation experiment. Since (stochastic) simulations require statistical analysis of the output, it is often appropriate to consider the perspective of *experimental design*.

Example: As in Cochran and Cox (1992), Montgomery (2000) and Wu and Hamada (2000).

Simulations are also *explorations*. We usually conduct simulations because we want to learn more about a complex system we inadequately understand. To head in the right direction, we should have some well-defined goals and questions when we start, but we should expect to develop new goals and questions as we go along. When we think about experimental design, we should observe that the time scale for computer simulation experiments tends to be much shorter than the time scale for the agricultural and medical experiments that led to the theory of experimental design. With the steadily increasing power of computers, computer simulation has become a relatively rapid process. After doing one simulation, we can quickly revise it and conduct others. Therefore, it is almost always best to think of simulation as an *iterative process*:

Notes

We conduct a simulation experiment, look at the results and find as many new questions as answers to our original questions. Each simulation experiment suggests subsequent simulation experiments. Through a succession of these experiments, we gradually gain the better understanding we originally sought. To a large extent, it is fruitful to approach simulation in the spirit of *exploratory data analysis*, e.g., as in Tukey (1977), Velleman and Hoaglin (1981).

Successful simulation studies usually involve an artful mix of both experimental design and exploration. We would emphasize the spirit of exploration, but we feel that some experimental design can be a big help. When we plan to hike in the mountains, in addition to knowing what peak we want to ascend, it is also good to have a rough idea how long it will take to get there: Should the hike take two hours, two days or two weeks?

That is just the kind of rough information we need for simulations. A major purpose of simulation experiments, often as a means to other ends, is to estimate unknown quantities of interest. When we plan to conduct a simulation experiment, in addition to knowing what quantities we want to estimate, it is also good to have a rough idea how long it will take to obtain a reliable estimate: Should the experiment take two seconds, two hours or two years?

As in Whitt (1989), in this chapter we discuss techniques that can be used, before a simulation is conducted, to estimate the computational effort required to obtain desired statistical precision for contemplated simulation estimators. Given information about the required computational effort, we can decide what cases to consider and how much computational effort to devote to each. We can even decide whether to conduct the experiment at all.

The theoretical analysis we discuss should complement the experience we gain from con- ducting many simulation experiments. Through experience, we learn about the amount of computational error required to obtain desired statistical precision for simulation estimators in various settings. The analysis and computational experience should reinforce each other, giving us better judgment. The methods in this chapter are intended to help develop more reliable expectations about statistical precision. We can use this knowledge, not only to design better simulation experiments, but also to evaluate simulation output analysis, done by others or ourselves.

The experimental design problem may not seem very difficult. First, we might think, given the amazing growth in computer power, that the computational effort rarely needs to be that great, but that is not the case: Many simulation estimation goals remain out of reach, just like many other computational goals; e.g., see Papadimitriou (1994). Second, we might think that we can always get a rough idea about how long the runs should be by doing one *pilot run* to estimate the required simulation run lengths. However, there are serious difficulties with that approach. First, such a preliminary experiment requires that we set up the entire simulation before we decide whether or not to conduct the experiment.

Nevertheless, if such a sampling procedure could be employed consistently with confidence, thenthe experimental design problem would indeed not be especially difficult. In typical simulation experiments, we want to estimate steady-state means for several different input parameters.

Did u know? **What are explorations?**

The act of exploring, penetrating, or ranging over for purposes of discovery.

Unfortunately, doing a pilot run for one set of parameters may be very misleading, because the required run length may change dramatically when the input parameters are changed. To illustrate how misleading one pilot run can be, consider a simulation of a *queueing model*. Indeed, we shall use queueing models as the context examples throughout the chapter. Now consider the simulation of a single-server queue with unlimited waiting, with the objective of estimating the mean steady-state (or long-run average) number of customers in the system, as a function of

Notes

basic model data such as the arrival stochastic process and the service-time distribution. This queueing experimental design problem is interesting and important primarily because a uniform allocation of data over all cases (parameter values) is not nearly appropriate. Experience indicates that, for given statistical precision, the required amount of data increases dramatically as the traffic intensity ρ (arrival rate divided by the service rate) increases toward the critical level for stability and as the arrival-and-service variability (appropriately quantified) increases.

For example, the required simulation run length to obtain 5% relative error (width of confidence interval divided by the estimated mean) at a high traffic intensity such as 0:95 tends to be 100 times greater than at a lower traffic intensity such as 0:50. (The operative formula underlying this rough estimate is $f(\rho) \equiv (1-\rho)^{-2}$; note that f(0:95)/f(0:50) = 400/4 = 100. If we consider the more extreme case ρ = 0:995, then the factor is 10,000. If we used a criterion of absolute error instead of relative error, then the operative formula becomes even more impressive: then $f(\rho) \equiv (1-\rho)^{-4}$.

In this queueing example, and throughout this paper, we use simulation time as our characterization of computational effort. (For a theoretical discussion of this issue, see Glynn and Whitt (1992).) Some computational experience or additional experiments on the selected computer are needed to convert simulation time into computational effort. Since there is a degree of freedom in choosing the measuring units for time, it is important to normalize these time units. For example, in a queueing model we might measure time in terms of the number of arrivals that enter the system or we might stipulate that a representative service-time distribution has mean 1. On the positive side, focusing on required simulation time has the advantage that it yields characterizations of computational effort that are independent of the specific computer used to conduct the simulation. It seems best to try to account for that important factor separately.

We assume that the quantities of interest will be estimated by sample means. With sample means, in great generality the required amount of simulation time can be determined by computing quantities called the asymptotic variance and the asymptotic bias of the sample means. Thus, we want to estimate these quantities before conducting the simulation. In general, that is not so easy to do, but existing theory supports this step for a sample mean of a function of a Markov process. However, the stochastic processes of interest in simulation models are rarely Markov processes.

It is important to approach this approximation step with the right attitude. Remember that we usually only want to obtain a rough estimate of the required simulation run length. Thus, we may well obtain the desired insight with only a very rough approximation. We do not want this analysis step to take longer than it takes to conduct the simulation itself. So we want to obtain the approximation quickly and we want to be able to do the analysis quickly. Fortunately, it is often possible to meet these goals.

For example, we might be interested in simulating a non-Markovian open network of single-server queues.

We might be interested in the queue-length distributions at the different queues. To obtain a rough estimate of the required simulation run length, we might first solve the traffic- rate equations to find the net arrival rate at each queue. That step is valid for non-Markovian queueing networks as well as Markovian queueing networks; e.g., see Chen and Yao (2001), Kelly (1979) or Walrand (1988). Given the net arrival rate at each queue, we can calculate the tra±c intensity at each queue by multiplying the arrival rate times the mean service time. Then we might focus on the *bottleneck queue*, i.e., the queue with the highest traffic intensity. We do that because the overall required run length is usually determined by the bottleneck queue. Then we analyze the bottleneck queue separately (necessarily approximately). We might approximate the bottleneck queue by the Markovian M=M=1 queue with the same traffic intensity, and apply the techniques

Notes

described in this paper to the Markovian queue-length process in order to estimate the required simulation run length. Alternatively, to capture the impact of the arrival and service processes beyond their means, we might use heavy-tra±c limit theorems to approximate the queue-length process of the bottleneck queue by a reflected Brownian motion (RBM); e.g., see Chen and Yao (2001) and Whitt (2002). We then apply the techniques described in this paper to the limiting RBM, which is also a Markov process.

Notes Methods described in these last two paragraphs, we can treat quite general queueing-network models, albeit roughly.

12.3.1 The Standard Statistical Framework

Probability Model of a Simulation

We base our discussion on a probability model of a (stochastic) simulation experiment: In the model, the simulation experiment generates an initial segment of a stochastic process, which may be a discrete-time stochastic process $[X_n : n \geq 1]$ 1g or a continuous-time stochastic process $[X(t) : t \geq 0]$. We form the relevant sample mean

$$\bar{X}_n \equiv n^{-1} \sum_{i=1}^{n} X_i \quad \text{or} \quad \bar{X}_t \equiv t^{-1} \int_0^t X(s)\, ds,$$

and use the sample mean to estimate the long-run average,

$$\mu = \lim_{n \to \infty} \bar{X}_n \quad \text{or} \quad \mu = \lim_{n \to \infty} \bar{X}_t,$$

which is assumed to exist as a proper limit with probability one (w.p.1). Under very general regularity conditions, the long-run average coincides with the expected value of the limiting steady-state distribution of the stochastic process.

These stochastic processes arise in both observations from a single run and from independent replications. For example, in observations from a single run, a discrete-time stochastic process $[X_n : n \geq 1]$ arises if we consider the waiting times of successive arrivals to a queue. The random variable X_n might be the waiting time of the nth arrival before beginning service; then μ is the long-run average waiting time of all arrivals, which usually coincides with the mean steady-state waiting time. On the other hand, X_n might take the value 1 if the nth arrival waits less than or equal to x minutes, and take the value 0 otherwise; then μ ≡ μ (x) is the long-run proportion of customers that wait less than or equal to x minutes, which usually corresponds to the probability that the steady-state waiting time is less than or equal to x minutes.

Alternatively, in observations from a single run, a continuous-time stochastic process [x(t) : t≥ 0] arises if we consider the queue length over time, beginning at time 0. The random vari- able X(t) might be the queue length at time t or X(t) might take the value 1 if the queue length at time t is less than or equal to k, and take the value 0 otherwise.

With independent replications (separate independent runs of the experiment), we obtain a discrete-time stochastic process $\{X_n : n \geq 1\}$. Then X_n represents a random observation obtained from the nth run. For example, X_n might be the queue length at time 7 in the nth replication or X_n might be the average queue length over the time interval [0, 7] in the nth replication. Then the limit μ represents the long-run average over many independent replications, which equals the expected value of the random variable in any single run. Such expected values describe the expected *transient* (or time-dependent) behavior of the system.

Notes

Bias, Mean Squared Error and Variance

By assuming that the limits exist, we are assuming that we would obtain the exact answer if we devoted unlimited computational effort to the simulation experiment. In statistical language, e.g., see Lehmann and Castella (1998), we are assuming that the estimators $\bar{X}_n$ and $\bar{X}_t$ are consistent estimators of the quantity to be estimated, μ. For finite sample size, we can describe the statistical precision by looking at the bias and the mean squared error. The bias, which we denote by $\bar{\beta}_n$ in the discrete-time case and $\bar{\beta}_t$ in the continuous-time case, indicates how much the expected value of the estimator differs from the quantity being estimated, and in what direction. For example, in the discrete-time case, the bias of $\bar{X}_n$ is

$$\bar{\beta}_n = E[\bar{X}_n] - \mu$$

The mean-squared error (MSE_n or MSE_t) is the expected squared error, e.g.,

$$MSE_n = E\left[\left|\bar{X}_n - \mu\right|^2\right]$$

If there is no bias, then the MSE coincides with the variance of $\bar{X}_n$, which we denote by $\bar{\sigma}_n^2$, i.e.,

$$\bar{\sigma}_n^2 \equiv Var(\bar{X}_n) \equiv E\left[\left|\bar{X}_n\right] - E[\bar{X}_n]\right|^2]].$$

Then we can write

$$\bar{\sigma}_n^2 \equiv Var(\bar{X}_n) = n^{-2}\sum_{i=1}^{n}\sum_{j=1}^{n} Cov(X_i, X_j),$$

where $Cov(X_i, X_j)$ is the covariance, i.e.,

$$Cov(X_i, X_j) \equiv E[X_i X_j] - E[X_i]E[X_j].$$

Analogous formulas hold in continuous time. For example, then the variance of the sample mean $\bar{X}_t$ is

$$\bar{\sigma}_t^2 \equiv Var(\bar{X}_t) = t^{-2}\int_0^t\int_0^t Cov(X(u), X(v))\,du\,dv.$$

Unfortunately, these general formulas usually are too complicated to be of much help when doing preliminary planning.

The Classical Case: Independent Replications

In statistics, the classical case arises when we have a discrete-time stochastic process $\{X_n : n \geq 1\}$. where the random variables X_n are mutually Independent and Identically Distributed (IID) with mean μ and finite variance σ^2, and we use the sample mean $\bar{X}_n$ to estimate the mean μ. Clearly, the classical case arises whenever we use independent replications to do estimation, which of course is the great appeal of independent replications.

In the classical case, the sample mean $\bar{X}_n$ is a consistent estimator of the mean μ by the Law of Large Numbers (LLN). Then there is no bias and the MSE coincides with the variance of the sample mean, $\bar{\sigma}_n^2$, which is a simple function of the variance of a single observation X_n:

Notes

$$\bar{\sigma}_n^2 = MSE(\bar{X}_n) = \frac{\sigma^2}{n}$$

Moreover, by the Central Limit Theorem (CLT), $\bar{X}_n$ is asymptotically normally distributed as the sample size n increases, i.e.,

$$n^{1/2}[\bar{X}_n - \mu] \Rightarrow N(0,\sigma^2) \, as \, n \to \infty,$$

where N(a, b) is a normal random variable with mean a and variance b, and $\Rightarrow$ denotes convergence in distribution.

We thus use this large-sample theory to justify the approximation

$$P(\bar{X}_n \le x) \approx P(N(\mu,\sigma^2/n) \le x) = P(N(0,1) \le (x-\mu)/\sqrt{\sigma^2/n}\ .$$

Based on this normal approximation, a (1 − α)100% confidence interval for μ based on the sample mean $\bar{X}_n$ is

$$[\bar{X}_n - z_{\alpha/2}(\sigma/\sqrt{n}), \bar{X}_n + z_{\alpha/2}(\sigma/\sqrt{n})]$$

Where

$$P(-z_{\alpha/2} \le N(0,1) \le +z_{\alpha/2}) = 1-\alpha\ .$$

A common choice is a 95% confidence interval, which corresponds to $\alpha = 0.05$; then $z_{\alpha/2} = 1.96 \approx 2$.

The statistical precision is typically described by either the absolute width or the relative width of the confidence interval, denoted by $w_a(\alpha)$ and $w_r(\alpha)$, respectively, which are

$$w_a(\alpha) = \frac{2z_\alpha/2^\sigma}{\sqrt{n}} \, and \, w_r(\alpha) = \frac{2z_\alpha/2^\sigma}{\mu\sqrt{n}}$$

There are circumstances where each measure is preferred. For specified absolute width or relative width of the confidence interval, ε, and for specified level of precision α, the required sample size $n_a(\alpha)$ or $n_r(\varepsilon, \alpha)$ is then

$$n_a(\varepsilon,\alpha) = \frac{4\sigma^2 z^2{}_{\alpha/2}}{\varepsilon^2} \, or \, n_r(\varepsilon,\alpha) = \frac{4\sigma^2 z^2{}_{\alpha/2}}{\mu^2\varepsilon^2} \qquad (1)$$

From these formulas, we see that $n_a(\varepsilon, \alpha)$ and $n_r(\varepsilon, \alpha)$ are both inversely proportional to ε^2 and directly proportional to σ^2 and $z^2_{\alpha/2.}$

Standard statistical theory describes how observations (data) can be used to estimate the unknown quantities μ and σ^2. We might use a two-stage sampling procedure, exploiting the first stage to estimate the required sample size. However, here we are concerned with what we can do without any data at all.

We propose applying additional information about the model to obtain rough preliminary estimates for these parameters without data. Following the general approach of this paper, we suggest trying to estimate μ and σ^2 before conducting the simulation by analyzing the probability distribution of the outcome of a single replication, X_n (using knowledge about the model). Admittedly, this preliminary estimation is often difficult to do; our approach is usually more useful in the context of one long run, which is discussed in the next section.

Notes

However, more can be done in this context than is often thought. Again, we must remember that we are only interested in making a rough estimate. Thus, we should be ready to make back-of-the-envelope calculations.

To illustrate what can be done, suppose that we focus on the the relative-width criterion. With the relative-width criterion, it suffices to estimate the squared coefficient of variation (SCV, variance divided by the square of the mean)

$$c^2 \equiv \frac{\sigma^2}{\mu^2}$$

instead of both μ and σ^2. With the relative-width criterion, the required sample size is:

$$n_r(\varepsilon, \alpha) = \frac{4c^2 z_{\alpha/2}^2}{\varepsilon^2}$$

From the analysis above, we see that we only need to estimate a single parameter, the SCV c^2, in order to carry out this preliminary analysis. In many cases, we can make reasonable estimates based on "engineering judgment". For that step, it helps to have experience with variability as quantified by the SCV. First, note that the SCV measures the level of variability independent of the mean: The SCV of a random variable is unchanged if we multiply the random variable by a constant. We are thus focusing on the variability independent of the mean. Clearly, it is important to realize that the mean itself plays no role with the relative- width criterion.

Once we learn to focus on the SCV, we quickly gain experience about what to expect. A common reference case for the SCV is an exponential distribution, which has $c^2 = 1$. A unit point mass (deterministic distribution) has $c^2 = 0$. Distributions relatively more (less) variable than exponential have $c^2 > (<) 1$. In many instances we actually have a rough idea about the SCV. We might be able to judge in advance that the SCV is one of: (i) less than 1, but probably not less than 0.1, (ii) near 1, (iii) bigger than 1, but probably not bigger than 10, or (iv) likely to be large, even bigger than 10. In other words, it is not unreasonable to be able to estimate the SCV to within an order of magnitude (within a factor of 10). And that may be good enough for the desired rough estimates we want to make.

In lieu of information about the SCV, to obtain a rough estimate we can just let $c^2 = 1$. To proceed, we can also let $\alpha = 0.05$, so that $z_{\alpha/2} \approx 2$. Then, if we set $\varepsilon = 10^{-k}$, the required simulation run length is

$$n_r(10^{-k}, 0.05) = 16 \times 10^{2k} :$$

Thus, when $c^2 = 1$, 10% relative precision requires about 1600 replications, while 1% relative precision requires 160,000 replications. If $c^2 \neq 1$, then we would multiply the number of required replications above by c^2. We thus can easily factor in the only unknown, the SCV c^2.

We have just reviewed the classical IID case, which is directly relevant when we use independent replications. However, in this chapter we concentrate on the more complicated case in which we consider an initial segment of a stochastic process from a single simulation run. It is good to have the classical IID case in mind, though, to understand the impact of bias and dependence upon the required computational effort.

An Initial Segment from a Single Run

Now suppose that we intend to estimate a long-run average within a single run by the sample mean from an initial segment of a stochastic process, which could evolve in either discrete time or continuous time. The situation is now much more complicated, because the random

Notes

observations need not be IID. Indeed, they need not be either independent or identically distributed. Unlike the case of independent replications, we now face the problems of bias and dependence among the random variables.

Fortunately, there are generalizations of the classical IID framework that enable us to estimate the bias and the mean squared error as a function of the sample size in terms of only two fundamental parameters: the asymptotic bias and the asymptotic variance; see Whitt (1992) and references therein. That theory tells us that, under regularity conditions, both the bias and the MSE are of order $1 \neq n$.

Within a single run, the stochastic processes tend to become stationary as time evolves. Indeed, now we assume that $X_n \Rightarrow X(\infty)$ as $n \to \infty$ (in the discrete-time case) and $X(t) \Rightarrow X(\infty)$ as $t \to \infty$ (in the continuous-time case). The stochastic processes fail to be stationary throughout all time primarily because it is necessary (or at least more convenient) to start the simulation in a special initial state. We thus can reduce the bias by choosing a good initial state or by deleting (not collecting statistics over) an initial portion of the simulation run. Choosing an appropriate initial state can be difficult if the stochastic process of interest is not nearly Markov. For example, even for the relatively simple $M/G/s/\infty$ queueing model, with s servers and non-exponential service times, it is necessary to specify the remaining service time of all customers initially in service.

The asymptotic bias helps us to determine if it is necessary to choose a special initial state or delete an initial portion of the run. The asymptotic bias also helps us estimate the final bias, whether or not we choose a special initial state or delete an initial portion of the run. It also helps us determine what proportion of the full run should be deleted if we follow that procedure.

Under regularity conditions, there is a parameter $\overline{\beta}$ called the *asymptotic bias* such that

$$\overline{\beta} = \lim_{n \to \infty} n\overline{\beta}_n \tag{2}$$

see Whitt (1992) and references therein. Given the definition of the bias $\overline{\beta}_n$, we see that the asymptotic bias must be

$$\overline{\beta} = \sum_{i=1}^{\infty} (E[X_i] - \mu);$$

the regularity conditions ensure that the sum is absolutely convergent. We thus approximate the bias of $\overline{X}_n$ for any sufficiently large n by

$$\overline{\beta} \approx \frac{\overline{\beta}}{n}$$

This approximation reduces the unknowns to be estimated from the function $\{\overline{\beta}_n : n \geq 1\}$ to the single parameter $\overline{\beta}$. Corresponding formulas hold in continuous time.

Given that we can ignore the bias, either because it is negligible or because it has been largely removed by choosing a good initial state or by deleting an initial portion of the run, we can use the asymptotic variance to estimate the width of confidence intervals and thus the required run length to yield desired statistical precision. Under regularity conditions, there is a parameter $\overline{\sigma}^2$ called the *asymptotic variance* such that

$$\overline{\sigma}^2 = \lim_{n \to \infty} n\overline{\sigma}_n^2, \tag{3}$$

Where (under the assumption that $\{X_n : n \geq 1\}$ is a stationary process)

$$\overline{\sigma}^2 = Var(X_1) + 2\sum_{i=1}^{\infty} Cov(X_1, X_{1+i}),$$

Notes

with σ^2 being the variance of X_n and Cov(X_1;X_{1+i}) being the lag-*i autocovariance*. Because of the dependence, $\bar{\sigma}^2$ often is much bigger than σ^2. We thus approximate $\bar{\sigma}_n^2$, the variance of $\bar{X}_n$ for any sufficiently large n by

$$\bar{\sigma}_n^2 \equiv Var(\bar{X}_n) \approx \frac{\bar{\sigma}^2}{n}.$$

Again, this approximation reduces the unknowns to be estimated from the function $\{\bar{\sigma}_n^2 : n \geq 1\}$ to the single parameter $\bar{\sigma}^2$.

In continuous time, we have the related asymptotic variance formula

$$\bar{\sigma}^2 = \lim_{t\to\infty} t\bar{\sigma}_t^2,$$

where (under the assumption that $\{X(t) : t \geq 0\}$ is a stationary process)

$$\bar{\sigma}^2 = 2\int_0^\infty Cov(X(0), X(t))dt.$$

In continuous or discrete time, a critical assumption here is that the asymptotic variance $\bar{\sigma}^2$ is actually finite. The asymptotic variance could be infinite for two reasons: (i) heavy-tailed distributions and (ii) long-range dependence. In our context, we say that X_n or $X(t)$ has a heavy-tailed distribution if its variance is infinite. In our context, we say that there is long-range dependence (without heavy-tailed distributions) if the variance Var(X_n) or Var($X(t)$) is finite, but nevertheless the asymptotic variance is infinite because the autocovariances Cov(X_j;X_{j+k}) or Cov($X(t)$; $X(t + k)$) do not decay quickly enough as $k \to \infty$; e.g., see Beran (1994), Samorodnitsky and Taqqu (1994) and Chapter of Whitt (2002).

Assuming that $\bar{\sigma}^2 < \infty$, we can apply CLT's for weakly dependent random variables (involving other regularity conditions, e.g., see Section 4 of Whitt (2002)) to deduce that $\bar{X}_n$ (as well as $\bar{X}_t$) is again asymptotically normally distributed as the sample size n increases, i.e.,

$$n^{1/2}[\bar{X}_n - \mu] \Rightarrow N(0, \bar{\sigma}^2) \, as \, n \to \infty,$$

so that the asymptotic variance σ^2 plays the role of the ordinary variance σ^2 in the classical IID setting.

We thus again can use the large-sample theory to justify a normal approximation. The new $(1 - \alpha)100\%$ confidence interval for μ based on the sample mean $\bar{X}_n$ is

$$[\bar{X}_n - z_{\alpha/2}(\bar{\sigma}/\sqrt{n}), \bar{X}_n + z_{\alpha/2}(\bar{\sigma}/\sqrt{n})],$$

which is the same as for independent replications except that the asymptotic variance $\bar{\sigma}^2$ replaces the ordinary variance σ^2.

The formulas for the confidence interval relative width, $w_r(\alpha)$, and the required run length, $n_r(\varepsilon, \alpha)$, are thus also the same as for independent replications in (1) except that the asymptotic

Notes

variance $\bar{\sigma}^2$ is substituted for the ordinary variance σ^2; e.g., the required simulation run length with a relative-width criterion is

$$n_r(\varepsilon,\alpha) = \frac{4\bar{\sigma}^2 z_{\alpha/2}^2}{\mu^2\varepsilon^2} \; and \; n_r(10^{-k}, 0.05) \approx \frac{\bar{\sigma}^2}{\mu^2} 16 \times (10)^{2k} \tag{4}$$

From (1) and (4), we immediately see that the required run length is approximately $\bar{\sigma}^2/\sigma^2$ times greater when sampling from one run than with independent sampling (assuming that we could directly observe independent samples from the steady-state distribution, which of course is typically not possible).

As with independent replications, established simulation methodology and statistical theory tells how to estimate the unknown quantities μ, $\bar{\beta}$ and $\bar{\sigma}^2$ from data; e.g., see Bratley *et al*. (1987) and Fishman (2001). Instead, we apply additional information about the model to obtain rough preliminary estimates for these parameters without data. For $\bar{\sigma}^2$, the representation of the asymptotic variance in terms of the autocovariances is usually too complicated to be of much help, but fortunately there is another approach, which we will describe in Section

12.3.2 The Asymptotic Parameters for a Function of a Markov Chain

From the previous section, it should be apparent that we can do the intended preliminary planning if we can estimate the asymptotic bias and the asymptotic variance. We now start to describe how we can calculate these important parameters. We first consider functions of a Markov chain. That illustrates available general results. However, fast back-of-the-envelope calculations usually depend on diffusion approximations, based on stochastic-process limits, after doing appropriate scaling. Indeed, the scaling is usually the key part, and that is so simple that back-of-the-envelope calculations are actually possible.

In this section, drawing on Whitt (1992), which itself is primarily a survey of known results (including Glynn (1984) and Grassman (1987a,b) among others), we observe that (again under regularity conditions) we can calculate the asymptotic bias and the asymptotic variance whenever the stochastic process of interest is a function of a (positive-recurrent irreducible) Markov chain, i.e., when $X_n = f(Y_n)$ for $n \geq 1$, where f is a real-valued function and $\{Y_n : n \geq 1\}$ is a Markov chain or when $X(t) = f(Y(t))$ for $t \geq 0$, where again f is a real-valued function and $\{Y(t) : t \geq 0\}$ is a Markov chain. As noted before, we usually obtain the required Markov structure by approximating the given stochastic process by a related one with the Markov property.

In fact, as in Whitt (1992), we only discuss the case in which the underlying Markov chain has a finite state space (by which we mean countably finite, i.e., $\{0, 1, ..., m\}$, not $[c, d]$), but the theory extends to more general state spaces under regularity conditions. For illustrations, see Glynn (1994) and Glynn and Meyn (1996). But the finite-state-space condition is very useful. Under the finite-state-space condition, we can compute the asymptotic parameters numerically, without relying on special model structure. However, when we do have special structure, we can sometimes go further to obtain relatively simple closed-form formulas. We also obtain relatively simple closed-form formulas when we establish diffusion-process approximations via stochastic-process limits.

Continuous-time Markov Chains

We will discuss the case of a Continuous-time Markov Chain (CTMC); similar results hold for discrete-time Markov chains. Suppose that the CTMC $\{Y(t) : t \geq 0\}$ is irreducible with finite state space $\{0, 1, ..., m\}$ (which implies that it is positive recurrent). Our sample-mean estimator is

Notes

$$\overline{X}_t \equiv t^{-1}\int_0^t X(s)ds,\ t \geq 0,$$

Where $X(t) = f(Y(t))$. (With the discrete state space, we write both $f(j)$ and f_j for the value of f at argument j.)

A finite-state CTMC is characterized by its *infinitesimal generator matrix* $Q \equiv (Q_{ij})$; Q_{ij} is understood to be the derivative (from above) of the *probability transition function*

$$P_{i,j}(t) \equiv P(Y(s+t) = j \mid Y(s) = i)$$

with respect to time t evaluated at $t = 0$. However, in applications the model is specified by defining the infinitesimal generator Q. Then the probability transition function can be obtained subsequently as the solution of the ordinary differential equations

$$P'(t) = P(t)Q = QP(t)\ ;$$

which takes the form of the matrix exponential

$$P(t) = e^{Qt} \equiv \sum_{k=0}^{\infty} \frac{Q^k t^k}{k!}.$$

Key asymptotic quantities associated with a CTMC are the stationary probability vector π and the fundamental matrix Z. (By convention, we let vectors be row vectors.) The *stationary probability vector* π can be found by solving the system of equations

$$\pi Q = 0\, with \sum_{i=0}^{m} \pi_i = 1.$$

The quantity we want to estimate is the expected value of the function f (represented as a row vector) with respect to the stationary probability (row) vector π, i.e.,

$$\mu = \pi f^T = \sum_{i=0}^{m} \pi_i f_i\ ,$$

where T is the *transpose*.

We would not need to conduct a simulation to estimate μ if indeed we can calculate it directly as indicated above. As noted before, in intended applications of this planning approach, the actual model of interest tends to be more complicated than a CTMC, so that we cannot calculate the desired quantity μ directly. We introduce a CTMC that is similar to the more complicated model of interest, and use the CTMC analysis to get rough estimates of both μ and the required computational effort in order to estimate μ by simulation. We will illustrate for queueing models later.

To continue, the fundamental matrix Z describes the time-dependent deviations from the stationary vector, in particular,

$$Z_{i,j} \equiv \int_0^{\infty} [P_{i,j}(t) - \pi_j]dt.$$

Given the stationary probability vector, π, the fundamental matrix Z can be found by first forming the square matrix II, all of whose rows are the vector π, and then calculating

$$Z = (II - Q)^{-1} - II,$$

with the inverse always existing; again see Whitt (1992) and references therein. We now consider how the desired asymptotic parameters can be expressed in terms of the stationary probability vector π and the fundamental matrix Z, including ways that are especially effective for computation.

Notes

Poisson's Equation

For that purpose, it is useful to introduce Poisson's equation. The stationary probability vector π and the fundamental matrix Z can be viewed as solutions x to *Poisson's equation*

$$xQ = y\,;$$

for appropriate (row) vectors y. It can be shown that Poisson's equation has a solution x if and only if $ye^T = 0$, where e is the row vector of 1's and e^T is its transpose. Then all solutions are of the form

$$x = -yZ + (xe^T)\pi.$$

For example, π is obtained by solving Poisson's equation when y is the zero vector (and normalizing by requiring that $xe^T = 1$). Then elements of Z can be obtained by choosing other vectors y, requiring that $xe^T = 0$.

In passing, we remark that there also is an alternate *column-vector form of Poisson's equation*, namely,

$$Qx^T = y^T,$$

which has a solution if and only if $\pi y^T = 0$. Then all solutions are of the form

$$x^T = -Zy^T + (\pi x^T)e^T.$$

It is significant that, for a CTMC, the asymptotic bias $\bar{\beta}$ defined in (2) and the asymptotic variance $\bar{\sigma}^2$ defined in (3) can be expressed directly in terms of π, Z, the function f and (for β) the initial probability vector, say ξ, i.e.,

$$\bar{\beta}(\xi) = \xi Zf \equiv \sum_{i=0}^{m}\sum_{j=0}^{m} \xi_i Z_{i,j} f_j$$

and

$$\bar{\sigma}^2 = 2(f\pi)Zf \equiv 2\sum_{i=0}^{m}\sum_{j=0}^{m} f_i \pi_i Z_{i,j} f_j.$$

Moreover, the asymptotic parameters $\bar{\beta}(\xi)$ and $\bar{\sigma}^2$ are themselves directly solutions to Poisson's equation. In particular,

$$\bar{\beta}(\xi) = xf^T,$$

where x is the unique solution to Poisson's equation for $y = -\xi + \pi$ with $xe^T = 0$. Similarly,

$$\bar{\sigma}^2 = 2xf^T,$$

where x is the unique solution to Poisson's equation for $y_i = -i(f_i - \mu)\pi_i$ with $xe^T = 0$.

Birth-and-Death Processes

Birth-and-death (BD) processes are special cases of CTMC's in which $Q_{i,j} = 0$ when $|i-j| > 1$; then we often use the notation $Q_{i,i+1} \equiv \lambda_i$ and $Q_{i,i}-1 \equiv \mu_i$, and refer to λ_i as the birth rates and μ_i as the death rates. For BD processes and skip-free CTMC's (which in one direction can go at most one step), Poisson's equation can be efficiently solved *recursively*.

Notes

To describe the rescursive algorithm for BD processes, we start by observing that for a BD process Poisson's equation $xQ = y$ is equivalent to the system of equations

$$x_{j-1}\lambda_{j-1} - x_j(\lambda_j + \mu_j) + x_{j+1}\mu_{j+1} = y_j, j \geq 0,$$

where $x_{-1} = x_{m+1} = 0$. Upon adding the first j +1 equations, we obtain the desired recursive algorithm,

$$x_{j+1} = (\lambda_j x_j + 8_j)/\mu_{j+1},$$

where

$$s_j = \sum_{i=0}^{j} y_i$$

Hence, Poisson's equation for BD processes can indeed be solved recursively.

For BD processes and their continuous-time relatives - diffusion processes - the asymptotic parameters can be expressed directly as sums and integrals, respectively. For BD processes,

$$\bar{\beta}(\xi) = \sum_{j=0}^{m-1} \frac{1}{\lambda_j \pi_j} \sum_{i=0}^{j} (f_i - \mu)\pi_i \sum_{k=0}^{j} (\xi_k - \pi_k)$$

and

$$\bar{\sigma}^2 = 2\sum_{j=0}^{m-1} \frac{1}{\lambda_j \pi_j} \left[\sum_{i=0}^{j} (f_i - \mu)\pi_i \right]^2,$$

where, as for CTMC's, π is the steady-state probability vector, while μ is the expected value of f with respect to π. However, for BD processes, it is usually easier to use the recursive algorithm for computation. Indeed, the recursive algorithm for the asymptotic bias and the asymptotic variance parallels the well known recursive algorithm for the steady-state probability vector π.

12.3.3 Birth-and-Death Examples

In this section we consider examples of BD processes, primarily of queueing models. These examples show that the model structure can make a big difference in the computational effort required for estimation by simulation.

Example: The M=M=1 queue.

Consider the queue-length (number in system, including the customer in service, if any) process $\{Q(t) : t \geq 0\}$ in the *M/M/*1 queue with unlimited waiting space. This model has a Poisson arrival process with constant rate and IID service times with an exponential distribution. The state space here is infinite, but the theory for the asymptotic parameters extends to this example. The queue-length process is a BD process with constant arrival (birth) rate and constant service (death) rate.

Let the service rate be 1 and let the arrival rate and traffic intensity be ρ. Fixing the service rate gives meaning to time in the simulation run length. Let $f(i) = i$ for all i, so that we are estimating the steady-state mean. The steady-state mean and variance are

Notes

$$\mu = \frac{\rho}{1-\rho} \, and \, \sigma^2 = \frac{p}{(1-\rho)^2};$$

e.g., see Cohen (1982).

To estimate the required simulation run length from a single long run, we use the asymptotic bias and the asymptotic variance. Let the system start out empty, so that the initial state is 0. As an argument of $\bar{\beta}(\xi)$, let 0 also denote the initial probability vector ξ that puts mass 1 on the state 0. Then

$$\bar{\beta}(0) = \frac{\rho}{(1-\rho)^3} \, and \, \bar{\sigma}^2 = \frac{2\rho(1+\rho)}{(1+\rho)^4}.$$

These formulas can be derived from the general BD formulas or directly; see Abate and Whitt (1987b, 1988 a,b).

Ignoring the initial transient (assuming that the queue-length process we observe is a stationary process), the required run length with a relative-width criterion, specified in general in (4), here is

$$t_r(\varepsilon,\alpha) = \frac{8(1+\rho)z^2_{\alpha/2}}{\rho(1-\rho)^2\varepsilon^2} \, and \, t_r(10^{-k}, 0.05) \approx \frac{32(1+\rho)(10)^{2k}}{\rho(1-\rho)^2}.$$

For 10% statistical precision (ε = 0.1) with 95% confidence intervals (α = 0.05), when the traffic intensity is ρ = 0.9, the required run length is about 675,000 (mean service times, which corresponds to an expected number of arrivals equal to 0.9×675,000 = 608,000); when the traffic intensity is ρ = 0.99, the required run length is 64,300,000 (mean service times, which corresponds to an expected number of arrivals equal to 0.9×64,300, 000 = 57,900,000). To summarize, for high traffic intensities, the required run length is of order 10^6 or more mean service times. We can anticipate great computational difficulty as the traffic intensity ρ increases toward the critical value for stability.

Compared to independent sampling of the steady-state queue length (which is typically not directly an option), the required run length is greater by a factor of

$$\frac{\bar{\sigma}^2}{\sigma^2} = \frac{2(1+\rho)}{\rho(1-\rho)^2}.$$

which equals 422 when ρ = 0.9 and 40,200 when ρ = 0.99. Clearly, the dependence can make a great difference.

Now let us consider the bias. The relative bias is

$$\frac{\bar{\beta}_t(0)}{\mu} \approx \frac{\bar{\beta}(0)}{t\mu} = \frac{1}{(1-\rho)^{2t}},$$

so that, for ρ = 0.9 the relative bias starting empty is about 100/*t*, where *t* is the run length. For a run length of 675,000, the relative bias is 1.5×10^{-4} or 0.015%, which is indeed negligible compared to the specified 10% relative width of the confidence interval. Hence the bias is in the noise; it can be ignored for high traffic intensities. The situation is even more extreme for higher traffic intensities such as ρ = 0.99.

Example. A small example with large asymptotic parameters

Notes

It is interesting to see that the asymptotic bias $\bar{\beta}(\xi)$ and the asymptotic variance $\bar{\sigma}^2$ can be arbitrarily large in a very small BD model with bounded rates. Suppose that $m = 2$, so that the BD process has only 3 states: 0, 1 and 2. Consider the symmetric model in which $\lambda_0 = \mu_2 = x$ and $\lambda_1 = \mu_1 = 1$, where $0 < x \leq 1$. Then the stationary distribution is:

$$\pi_0 = \pi_2 = \frac{1}{2+x} \, and \, \pi_1 = \frac{x}{2+x}.$$

Let $f_i = i$ for all i, so that we are estimating the mean μ. Then the mean is $\mu = 1$ and the asymptotic variance is

$$\bar{\sigma}^2 = \frac{4}{x(2+x)} \approx \frac{2}{x} \, for \, small \, x.$$

This model has a high asymptotic variance $\bar{\sigma}^2$ for small x because the model is *bistable*, tending to remain in the states 0 and 2 a long time before leaving. To see this, note that the mean first passage time from state 0 or state 2 to state 1 is $1/x$.

Note that the large asymptotic variance $\bar{\sigma}^2$ cannot be detected from the variance of the steady-state distribution, σ^2. As $x \downarrow 0$, σ^2, the variance of π, increases to 1. Thus, the ratio $\bar{\sigma}^2/\sigma^2$ is of order $O(1/x)$. The steady-state distribution has moderate variance, but the process has quite strong dependence (but not so strong that the asymptotic variance becomes infinite).

The asymptotic bias starting in state 0 (or state 2) is also large for small x. The asymptotic bias starting in state 0 is

$$\bar{\beta}(0) = \frac{-(x+1)^2}{x(x+2)^2} \approx \frac{-1}{4x} \, for \, small \, x.$$

As a function of the key model parameter x, the bias is much more important here than it was for the previous*M/M/*1 queue example. Here, *both* the asymptotic bias and the asymptotic variance are of order $O(1/x)$, so that as a function of x, for very small x, the width of the confidence interval is $O(1/\sqrt{x})$, while the bias is of order $O(1/x)$. Thus the bias tends to be much more important in this example. In particular, the run length required to make the bias suitably small is of the *same order* as the run length required to make the width of a confidence interval suitably small. For this example, using simulation to estimate the mean μ when the parameter x is very small would be difficult at best.

This model is clearly pathological. For very small x, a relatively long simulation run of this model starting in state 0 could yield a sample path that is identically zero. We might never experience even a single transition! This example demonstrates that it can be very helpful to know something about model structure when conducting a simulation.

Example: The M/M/ ∞ queue.

A queueing system with many servers tends to behave quite differently from a single-server queue. A queueing system with many servers can often be well approximated by an infinite-server queue. Thus we consider the number of busy servers at time t, also denoted by $Q(t)$, in an $M/M/\infty$ queue. As before, let the mean individual service time be 1, but now let the arrival rate be λ (since the previous notion of traffic intensity is no longer appropriate). Now the arrival rate λ can be arbitrarily large.

Notes

The first thing to notice is that as λ increases, the required computational effort for given simulation run length in the simulation increases, simply because the expected number of arrivals in the interval [0; *t*] is λt. Thus, with many servers, we need to do a further adjustment to properly account for computational effort. To describe the computational effort, it is appropriate to multiply the time by λ. Thus, for the $M/M/\infty$ model with mean individual service rate 1, we let $c_r = \lambda t_r$ represent the required computational effort associated with the required run length t_r.

It is well known that the steady-state number of busy servers in the $M/M/\infty$ model, say $Q(\infty)$, has a Poisson distribution with mean λ; e.g., see Cooper (1982). Thus, the mean and variance of the steady-state distribution are:

$$\mu \equiv E[Q(\infty)] = \lambda \; and \; \sigma^2 \equiv Var[Q(\infty)] = \lambda.$$

The asymptotic parameters also are relatively simple. As for the *M/M/1* queue, we assume that we start with an empty system. Then the asymptotic bias and asymptotic variance are:

$$\bar{\beta}(0) = \lambda \; and \; \bar{\sigma}^2 = 2\lambda$$

From the perspective of the asymptotic variance and relative error, we see that

$$\frac{\bar{\sigma}^2}{\mu^2} = \frac{2\lambda}{\lambda^2} = \frac{2}{\lambda},$$

so that simulation efficiency increases as λ increases. However, the required computational effort to achieve relative $(1 - \alpha)\%$ confidence interval width of ε is

$$c_r(\varepsilon,\alpha) \equiv \lambda t_r(\varepsilon,\alpha) = \frac{8z_\alpha^2}{\varepsilon^2},$$

which is *independent of* λ. Thus, from the perspective of the asymptotic variance, the required computational effort does not increase with the arrival rate, which is very different from the single-server queue.

Unfortunately, the situation is not so good for the relative bias. First, the key ratio is

$$\frac{\bar{\beta}(0)}{\mu} = \frac{\lambda}{\lambda} = 1.$$

Thus the required run length to make the bias less than ε is $1/\varepsilon$, and the required computational effort is λ/ε, which is *increasing in* λ. Unlike for the *M/M/*1 queue, as the arrival rate λ increases, the bias (starting empty) eventually becomes the dominant factor in the required computational effort.

For this $M/M/\infty$ model, it is natural to pay more attention to bias than we would with a single-server queue. A simple approach is to choose a different initial condition. The bias is substantially reduced if we start with a fixed number of busy servers not too different from the steady-state mean, λ. Indeed, if we start with exactly λ busy servers (assuming that λ is an integer), then the bias is asymptotically negligible as λ increases. Note, however, that this special initial condition does not directly put the system into steady state, because the steady-state distribution is Poisson, not deterministic.

If, instead, we were working with the $M/G/\infty$ model, then in addition we would need to specify the remaining service times of all the λ customers initially in service at time 0. Fortunately, for

Notes

the $M/G/\infty$ model, there is a natural way to do this: The steady-state distribution of the number of busy servers is again Poisson with mean λ, just as for the $M/M/\infty$ model. In addition, in steady-state, conditional upon the number of busy servers, the remaining service times of those customers in service are distributed as IID random variables with the *stationary-excess* (or equilibrium residual-life) Cumulative Distribution Function (CDF) G_e associated with the service-time CDF G, i.e.,

$$G_e(t) \equiv m^{-1}\int_0^t [1-G(u)]du, \tag{1}$$

where m is the mean of G (here $m = 1$); e.g., see Takács (1962).

It is natural to apply this insight to more general many-server queueing models. Even in more general $G/G/s$ models, it is natural to initialize the simulation by putting s customers in the system at time 0 and letting their remaining service times be distributed as s IID random variables with cdf G_e. For large s, that should be much better than starting the system empty.

For many-server queues, we may be interested in different congestion measures. By Little's law ($L = \lambda W$), we know that the expected steady-state number of busy servers in the $G/G/s/\infty$ model is exactly λ (provided that $\lambda < s$). Thus, in simulation experiments, we usually are more interested in estimating quantities such as $E[(Q(\infty) - s)^+]$, where $(x)^+ \equiv \max\{0, x\}$, or $P(Q(\infty) > s + k)$. Note that we can calculate the asymptotic bias and the asymptotic variance for these quantities in the $M/M/s$ model by applying the BD recursion with appropriate functions f. With large s, it often helps to start the recursion at s and move away in both directions. The initial value at s can be taken to be 1; afterwards the correct value is obtained by choosing the appropriate normalizing constant.

12.3.4 Diffusion Processes

Diffusion processes are continuous analogues of BD processes; e.g., see Karlin and Taylor (1981) and Browne and Whitt (1995). In this chapter we discuss diffusion processes because we are interested in them as approximations of other processes that we might naturally simulate using discrete-event simulation. We want to use the diffusion processes to approximate the asymptotic bias and the asymptotic variance of sample means in the original process.

Diffusion processes tend to be complicated to simulate directly because they have continuous, continuously fluctuating, sample paths. Nevertheless, there also is great interest in simulating diffusion processes and stochastic differential equations, e.g., for finance applications, and special techniques have been developed; see Kloeden, Platen and Schurz (1994) and Kloeden and Platen (1995). Hence the analysis in this section may have broader application.

For diffusion processes, there are integral representations of the asymptotic parameters, paralleling the sums exhibited for BD processes. Corresponding to the finite-state-space assumption for the BD processes, assume that the state space of the diffusion is the finite interval $[s_1, s_2]$ and let the diffusion be reflecting at the boundary points s_1 and s_2, but under regularity conditions the integral representations will be valid over unbounded intervals. Let $\{Y(t) : t \geq 0\}$ be the diffusion process and let $X(t) = f(Y(t))$ for a real-valued function f. The diffusion process is characterized by its drift function $\mu(x)$ and its diffusion function $\sigma^2(x)$.

Let π be the stationary probability density. The stationary probability density can be represented as

$$\pi(y) = \frac{m(y)}{M(s_2)}, s_1 \leq y \leq s_2.$$

Notes

where

$$m(y) \equiv \frac{2}{\sigma^2(y)s(y)}$$

is the *speed density*,

$$s(y) \equiv e^{-1\int_{s_1}^{y}[2\mu(x)/\sigma^2(x)]dx}$$

is the *scale density* and

$$M(y) = \int_{s_1}^{y} m(x)dx, s_1 \le y \le s_2.$$

provided that the integrals are finite.

Let $p(t, x, y)$ be the transition kernel. Then, paralleling the fundamental matrix of a CTMC, we can define the *fundamental function of a diffusion process*, $Z \equiv Z(x, y)$, by

$$Z(x,y) \equiv \int_0^{\infty}[p(t,x,y) - \pi(y)]dt.$$

As before, let μ be the average of f with respect to the stationary probability density π, i.e.,

$$\mu = \int_{s_1}^{s_2} \pi(x) f(x)dx.$$

Then the integral representations for the asymptotic bias $\bar{\beta}(\xi)$ starting with initial probability density ξ and the asymptotic variance $\bar{\sigma}^2$ are:

$$\bar{\beta}(\xi) = 2\int_{s_1}^{s_2} \frac{1}{\sigma^2(y)\pi(y)}\left[\int_{s_1}^{y}(f(x)-\mu)\pi(x)dx \int_{s_1}^{y}(\xi(z)-\pi(z))dz\right]dy$$

and

$$\bar{\sigma}^2 = 4\int_{s_1}^{s_2} \frac{1}{\sigma^2(y)\pi(y)}\left[\int_{s_1}^{y}(f(x)-\mu)\pi(x)dx\right]^2 dy.$$

We now discuss two examples of diffusion processes, which are especially important because they arise as limit processes for queueing models.

Example: RBM

Suppose that the diffusion process is reflected Brownian motion (RBM) on the interval $[0, \infty)$ with drift function $\mu(x) = a$ and diffusion function $\sigma^2(x) = b$, where $a < 0 < b$, which we refer to by RBM(a; b); see Harrison (1985), Whitt (2002) and references therein for more background. RBM is the continuous analog of the queue-length process for the $M/M/1$ queue (as we will explain in the next section). It is a relatively simple stochastic process with only the two parameters a and b.

In fact, we can analyze the RBM(a, b) processes by considering only the special case in which $a = -1$ and $b = 1$, which we call *canonical* RBM because there are *no free parameters*. We can analyze RBM(a, b) in terms of RBM(-1,1) because we can relate the two RBM's by appropriately scaling time and space. For that purpose, let $\{R(t; a, b, X) : t \ge 0\}$ denote RBM (a, b) with initial distribution

Notes

according to the random variable X. The key relation between the general RBM and canonical RBM is:

$$\{R(t;a,b,X):t\geq 0\}\overset{d}{=}\{c^{-1}R(d^{-1}t;-1,1,cX):t\geq 0\}$$

or, equivalently,

$$\{R(t;-1,1,X):t\geq 0\}\overset{d}{=}\{cR(dt;a,b,X/c):t\geq 0\}.$$

where

$$c=\frac{|a|}{b},d=\frac{b}{a^2},a=\frac{-1}{cd}and\,b=\frac{1}{c^2d},$$

where $\overset{d}{=}$ means equal in distribution (here as stochastic processes).

Hence it suffices to focus on canonical RBM. It has stationary density $\pi(x) = 2e^{-2x}$, $x \geq 0$. If we initialize RBM with its stationary distribution, then we obtain a stationary process. Let $R^* \equiv \{R^*(t; a, b) : t \geq 0\}$ denote *stationary* RBM, initialized by the stationary distribution.

If $f(x) = x$ for canonical RBM, then we would be estimating the steady-state mean $\mu = 1/2$. In this case, the asymptotic bias is $\bar{\beta} = 1/4$ (Theorem 1.3 of Abate and Whitt (1987a)) and the asymptotic variance (for R^*) is $\bar{\sigma}^2 = 1/2$ (Abate and Whitt (1988b)).

To describe the general RBM with parameters a and b, we apply the scaling relations in Subsection 6.1. As a consequence of those scaling properties, the mean and variance of the steady-state distribution of RBM(a, b) are:

$$\mu_{a,b}=\frac{b}{2|a|}and\,\sigma^2_{a,b}=\mu^2_{a,b}=\frac{b^2}{4a^2},$$

and the asymptotic parameters are:

$$\bar{\beta}_{a,b}=\frac{b^2}{4|a|^3}and\,\bar{\sigma}^2_{a,b}=\frac{b^3}{2a^4}.$$

For the relative-width criterion, the key ratios are:

$$\frac{\bar{\beta}_{a,b}}{\mu_{a,b}}=\frac{b}{2a^2}and\,\frac{\bar{\sigma}^2_{a,b}}{\mu^2_{a,b}}=\frac{2b}{a^2}.$$

Thus we see that the relative asymptotic bias is about the same as the relative asymptotic variance. Since the bias of the sample mean $\bar{X}_t$ is of order $O(1/t)$, while the square root of the variance of the sample mean $\bar{X}_t$ is of order $O(1/\sqrt{t})$, the bias tends to be negligible for large t.

Example: OU

Suppose that the diffusion process is the Ornstein-Uhlenbeck (OU) diffusion process on the interval $(-\infty, \infty)$ with drift function $\mu(x) = ax$ and diffusion function $\sigma^2(x) = b$, where $a < 0 < b$, which we refer to as OU(a; b). It is the continuous analog of the queue-length process in the $M/M/\infty$ queue when we center appropriately.

Notes

We also can analyze the OU (a, b) processes by considering only the special case in which $a = -1$ and $b = 1$, which we call *canonical* OU. We can analyze OU(a, b) in terms of OU(-1,1) because we can relate the two OU's by appropriately scaling time and space, just as we did for RBM. For that purpose, let $\{Z(t, a, b, X) : t \geq 0\}$ denote OU (a, b) with initial distribution according to the random variable X. The key relation between the general OU (a, b) and canonical OU(-1,1) is:

$$\{Z(t;a,b,X) : t \geq 0\} \stackrel{d}{=} \{c^{-1}Z(d^{-1}t;-1,1,cX) : t \geq 0\}$$

or, equivalently,

$$\{Z(t;-1,1,X) : t \geq 0\} \stackrel{d}{=} \{cZ(dt;a,b,X/c) : t \geq 0\}.$$

where

$$c = \frac{|a|}{b}, d = \frac{b}{a^2}, a = \frac{-1}{cd} and\, b = \frac{1}{c^2 d}.$$

Then the stationary density of canonical OU is normal with mean 0 and variance 1/2. The mean of canonical OU starting at x is

$$E[Z(t;-1,1,x)] = xe^{-t}, t \geq 0.$$

Paralleling our treatment of RBM, let $Z^* \equiv \{Z^*(t, a, b) : t \geq 0\}$ be *stationary* OU, obtained by initializing the OU(a,b) with the stationary normal distribution. For stationary canonical OU, the autocovariance function is

$$Cov(Z^*(0), Z^*(t)) = \frac{1}{2}e^{-t},\ t \geq 0.$$

Hence, the asymptotic parameters for canonical OU are

$$\bar{\beta}(\xi) = \xi\, and\, \bar{\sigma}^2 = \frac{1}{2}$$

Just as with RBM, we can apply Section to determine the effect of scaling. The mean and variance of the steady-state distribution of OU(a,b) are

$$\mu_{a,b} = 0\, and\, \sigma^2_{a,b} = \frac{b}{2|a|},$$

and the asymptotic parameters are

$$\bar{\beta}_{a,b,x} = x\frac{b^2}{|a|^3}\, and\, \bar{\sigma}^2_{a,b} = \frac{b^3}{2a^4}.$$

The relative-width criterion makes less sense here because the random variables are not non-negative.

12.3.5 Stochastic-Process Limits

In this section we discuss stochastic-process limits that make the RBM and OU diffusion processes serve as useful approximations for queueing models. We start by discussing the impact of scaling space and time. The scaling is often the key part.

Notes

Scaling of Time and Space

To obtain relatively simple approximate stochastic processes, we often consider stochastic-process limits, as in Whitt (2002). (We elaborate below.) To establish appropriate stochastic-process limits, we usually consider not just one stochastic process but a family of stochastic processes constructed by scaling time and space. It is thus important to know how the asymptotic parameters change under such scaling.

Suppose that we have a stochastic process $Z \equiv \{Z(t) : t \geq 0\}$ and we want to consider the scaled stochastic process $Z_{u,v} \equiv \{Z_{u,v}(t) : t \geq 0\}$, where

$$Z_{u,v}(t) \equiv uZ(vt);\ t \geq 0;$$

for positive real numbers u and v. Suppose that $Z(t) \Rightarrow Z(\infty)$ as $t \to \infty$. Then $Z_{u,v}(t) \Rightarrow Z_{u,v}(\infty)$ as $t \to \infty$, where

$$Z_{u,v}(\infty) = uZ(\infty).$$

Let μ be the mean and σ^2 the variance of $Z(\infty)$; let $\mu_{u,v}$ be the mean and $\sigma^2_{u,v}$ the variance of $Z_{u,v}(\infty)$. Then

$$\mu_{u,v} = u\mu \; and \; \sigma^2_{u,v} = u^2\sigma^2.$$

The relation is different for the asymptotic parameters: Observe that $EZ_{u,v}(t) = uEZ(vt)$ for $t \geq 0$ and, under the assumption that Z is a stationary process,

$$Cov(Z_{u,v}(0), Z_{u,v}(t)) = u^2 Cov(Z(0), Z(vt)), t \geq 0.$$

As a consequence, the asymptotic bias and the asymptotic variance are

$$\bar{\beta}_{u,v} = \frac{u}{v}\bar{\beta} \;\; and \;\; \bar{\sigma}^2_{u,v} = \frac{u^2}{v}\bar{\sigma}^2.$$

Thus, once we have determined the asymptotic parameters of a stochastic process of interest, it is easy to obtain the asymptotic parameters of associated stochastic processes constructed by scaling time and space. If the scaling parameters u and v are either very large or very small, then the scaling can have a great impact on the required run length. Indeed, as we show below, in standard queueing examples the scaling is dominant.

12.4 RBM Approximations

Consider the queue-length (number in system) stochastic process $\{Q_\rho(t) : t \geq 0\}$ in the $G/G/s/\infty$ with traffic intensity (rate in divided by maximum rate out) ρ, with time units fixed by letting the mean service time be 1, without the usual independence assumptions. As reviewed in Whitt (1989, 2002), in remarkable generality (under independence assumptions and beyond), there is a *heavy-traffic stochastic-process limit* for the scaled queue-length processes, obtained by dividing time t by $(1 - \rho)^2$ and multiplying space by $(1 - \rho)$, i.e.,

$$\{(1 - \rho)Q_\rho(t(1 - \rho)^{-2}) : t \geq 0\} \Rightarrow \{R(t, a, b) : t \geq 0\} \text{ as } \rho \uparrow 1$$

for appropriate parameters a and b, where $\{R(t, a, b) : t \geq 0\}$ is RBM(a, b) and again $\Rightarrow$ denotes convergence in distribution, but here in the function space D containing all sample paths.

The limit above is very helpful, because the number of relevant parameters has been greatly reduced. We see that the queue behavior for large ρ should primarily depend upon only ρ and the two parameters a and b. Moreover, it turns out the parameters a and b above can be conveniently characterized (in terms of scaling constants in central limit theorems for the arrival

Notes

and service processes). For example, in the standard $GI/GI/s/\infty$ model the heavy-traffic limit holds with

$$a = -s \text{ and } b = s(c_a^2 + c_s^2).$$

where c_a^2 and c_s^2 are the SCV's of an interarrival time and a service time, respectively (provided that the second moments are finite). Similar limits hold for workload processes, recording the amount of remaining unfinished work in service time in the system.

We thus can apply the stochastic-process limit with the scaling properties in Section and the properties of RBM to obtain approximations paralleling the exact results for the *M/M/*1 queue. We apply the stochastic-process limit to obtain the approximation

$$\{Q_\rho(t) : t \geq 0\} \approx \{(1-\rho)^{-1} R(t(1-\rho)^2; a, b) : t \geq 0\}.$$

The resulting approximations for the mean and variance of the steady-state distribution of the queue-length process are thus

$$E[Q_\rho(\infty)] \approx \frac{b}{2a(1-\rho)} \; and \; \sigma_\rho^2 \equiv Var(Q_\rho(\infty)) \approx \frac{b^2}{4a^2(1-\rho)^2};$$

the approximations for the asymptotic parameters are

$$\overline{\beta}_\rho \approx \frac{b^2}{4|a|^3(1-\rho)^3} \; and \; \sigma_\rho^2 \approx \frac{b^3}{2a^4(1-\rho)^4}.$$

In the $GI/GI/s/\infty$ case, we just substitute the specific parameters a and b above. The resulting approximate asymptotic variance is

$$\sigma_\rho^2 \equiv \sigma^2_{(s,\rho,c_a^2,c_s^2)} = \frac{(c_a^2 + c_s^2)^3}{2s(1-\rho)^4}.$$

Notes Note that these formulas agree with the limits of the *M/M/*1 formulas as $\rho \uparrow 1$.

Thus, we see that the *M/M/*1 formulas are remarkably descriptive more generally. But we also see the impact of s servers and the *GI* arrival and service processes. The asymptotic variance is directly proportional to $1/s$ and to the *third power* of the overall "variability parameter" $(c_a^2 + c_s^2)$ as well as to the *fourth* power of $(1-\rho)^{-1}$.

More generally, we see how the parameters s, ρ, a and b in more general $G/G/s/\infty$ models (with non-renewal arrival processes and non-IID service times) will affect the required simulation run length. Once we have established the corresponding heavy-traffic limit and identified the new values of a and b for these alternative models, we can apply the results above. For the relative-width criterion, the key ratios are

$$\frac{\overline{\beta}_{a,b}}{\mu_{a,b}} \approx \frac{b}{2a^2(1-\rho)^2} \; and \; \frac{\overline{\sigma}^2_{a,b}}{\mu^2_{a,b}} \approx \frac{2b}{a^2(1-\rho)^2}.$$

Values of the key parameters a and b in alternative models have been determined; e.g., see Sections of Whitt (1989) and Fendick, Saksena and Whitt (1989).

Notes

Did u know? **Markov Analysis**

Markov analysis provides a means of analysing the reliability and availability of systems whose components exhibit strong dependencies.

Caselet

Pokhran-II Revisited

In a recent interview with Karan Thapar, the former chief of the Atomic Energy Commission, Dr Anil Kakodkar, reiterated the Principal Scientific Advisor to the Government, Dr R. Chidambaram's stand on the success of Pokhran-II, but did not really address any of the concerns raised about the efficacy of the thermonuclear test. Most of the points he raised in a very general way have been dealt with before in a more detailed technical manner.

Take, for example, the issue about the lack of a crater for the thermonuclear explosion. It is true that if you bury such a device very deep, there will be a very small crater, or even none at all. But no one associated with Pokhran-II has come out with a number for that depth, though there is no secrecy needed here, as it reveals nothing about the design of the device. In fact, for Pokhran-I, we had immediately revealed that the device was buried 107m deep.

Only K. Santhanam, former DRDO scientist, has revealed that the thermonuclear device was buried at a depth of 130m, compared to the fission device's 100m deep location. If these numbers are correct, and no one has contradicted them, it is simply not credible to say that such a small difference in the depth (only 30m) made such a huge difference in the geology or in the crater size.

The repeated assertion that granite in the thermonuclear shaft was responsible for the small crater is also difficult to understand. Usually, shock waves couple better to hard rock and so the effect is expected to be larger. To muffle the explosion, one buries the device in soft material like sand or in an empty cavity. The reverse assertion seems to be a new advance in geology that the CTBT Organisation needs to take note of!

Puzzling Statements

Similarly, the statements on the simulations are puzzling. He brought out a new simulation experiment, perhaps done after Dr Santhanam's revelation. Using the borrowed data-base of an underground nuclear explosion in Nevada, they claim to have simulated what would have happened had the fission and fusion devices been interchanged between the two shafts S1 and S2.

He revealed that the fission device would have shown no crater, and the fusion device a much larger crater. This difference in the behaviour between the two sites, 1 km apart, and at almost similar depths, 100 m and 130 m, as revealed by Dr Santhanam, is inexplicable.

Simulations can be tweaked to predict anything you want. Also, there is a huge gap between simulating something and actually making it work in real life. Ultimately, there is no escape from detailed experiments. The computer and the word 'simulated' have been so extensively used by Dr Chidambaram and others that one wonders if there is any need at all for testing and experimental work in a wide variety of scientific investigations!

Contd...

Notes

I am also disappointed about the implications regarding statements from people not directly associated with Pokhran-II. This suggests that no experts, anywhere in the world, are competent to comment on Pokhran-II! This is unscientific.

One doesn't need to know every detail of the test to make intelligent estimates about the expected yield, or the crater size. In science, anyone can make a scientific observation, and it has to be refuted scientifically, and not by mere assertion. Those who are raising these questions have sufficient knowledge and experience to make those questions pertinent and relevant.

Searching Questions

Dr Kakodkar also revealed that we have more then one hydrogen bomb in our arsenal, and that we now have devices with yields ranging from sub-kt to 200 kt.

It must be reassuring to the military to know that the quantity issue has been addressed. But what is more important is to address the quality issue, about which the military should also ask searching questions. After all, deterrence is in the eyes of the enemy; it is not important what a few people assert, but what the whole world thinks.

Overall, I would say that the question mark over the yield and efficacy of the thermonuclear device, and hence our nuclear deterrent, still remains. Right from the beginning, the official response to very legitimate questions, raised by those who are knowledgeable and acting in the best interests of the nation, has been defensive and closed-minded. This does not augur well for the health of our strategic deterrent. Given the strong support from the new US administration for the NPT and the CTBT, and the international pressure that will certainly be put upon India in the near future, this is an issue the Government must address seriously.

Source: http://www.thehindubusinessline.com/todays-paper/tp-economy/article1071937.ece

12.5 Summary

- Validation of design rules taking into account fine details such as line-edge roughness, and full chip layout simulation for design inconsistencies, before actual fabrication, are among the main objectives of current software assisted metrology tools.
- The use of quantitative and qualitative computational models to make quantitative and qualitative predictions or rather to draw conclusions from complex antecedents, and then discusses different types of explanation and prediction.

12.6 Keywords

CD: Critical Dimension

IID: Independent and Identically Distributed

LLN: Law of Large Number

w.p.1: with probability one

12.7 Self Assessment

Fill in the blanks:

1. …… roughness quantification should accompany Critical Dimension (CD) measurements since it could be a large fraction of the total CD budget.

Notes

2. Biological processes running in a lake to simulate what could happen if these processes were disturbed by external influences imposed on the model by the ……
3. …… is a very expensive task in the latter, and in most cases it is even impossible to generate long time series for individual or group behavior.
4. Successful simulation studies usually involve an artful mix of both …… and exploration.
5. In a …… we might measure time in terms of the number of arrivals.
6. These …… arise in both observations from a single run and from independent replications.
7. A common reference case for the SCV is an …… distribution.
8. Unlike the case of…… we now face the problems of bias and dependence among the random variables.
9. The …… helps us to determine if it is necessary to choose a special initial state or delete an initial portion of the run.
10. …… of design rules taking into account fine details such as line-edge roughness.
11. The model is initialized with …… estimates of transition probabilities, age-specific birth and death rates and so on.
12. The simulation is initialized with a …… distribution close to the empirical distribution of 1950.
13. A simulation machine to predict the …… of real world experiments or perhaps to replace such experiments.
14. In …… the model matches data before data are acquired from the real system.
15. Validation of …… is thus the same (or at least analogous) to validation of theories.

12.8 Review Questions

1. In your view simulations are controlled experiments. Give reasons.
2. The experimental design problem may not seem very difficult. Explain with a example.
3. Under very general regularity conditions, the long-run average coincides with the expected value of the limiting steady-state distribution of the stochastic process. Comment.
4. What do you think as the difference between Bias, Mean Squared Error and Variance? Discuss.
5. We also obtain relatively simple closed-form formulas when we establish diffusion-process approximations via stochastic-process limits. Give examples.
6. Which is less time consuming Birth or Death Processes? Support your answer with proper reasoning.
7. Find out the difference between quantitative and qualitative predictions.
8. Diffusion processes are continuous analogues of BD processes. Justify your answer.
9. "The run length required to make the bias suitably small is of the same order as the run length required to make the width of a confidence interval suitably small." Comments.
10. Diffusion processes are continuous analogues of BD processes. Elaborate.

Notes

Answers: Self Assessment

1.	Line-edge	2.	simulating experimenter
3.	Data collection	4.	experimental design
5.	queuing model	6.	stochastic processes
7.	exponential	8.	independent replications
9.	asymptotic bias	10.	Validation
11.	empirical	12.	gender
13.	outcomes	14.	Predictive validity
15.	simulation models		

12.9 Further Readings

Books

Balci, O.,(1994), Validation, verification, and testing techniques throughout the life cycle of a simulation study, Annals of Operations Research, 53: 121–173.

Balci, O., (2001), A methodology for certification of modeling and simulation applications, ACM Transactions on Modeling and Computer Simulation, 11: 352–377.

Birta, L.G. and Ozmizrak, N.F., (1996), A knowledge-based approach for the validation of simulation models: The foundation, ACM Transactions on Modeling and Computer Simulation, 6: 67–98.

Boehm, B.W., (1979), Software engineering: R&D trends and defense needs, in: P. Wegner (Ed.), Research Directions in Software Technology, MIT Press, and Cambridge, MA.

Cellier, F.E., (1986), Combined discrete/continuous system simulation – application, techniques and tools, in Proceedings of the 1986 Winter Simulation Conference.

Department of Defense (DoD) Recommended Practices Guide (RPG) for Modeling and Simulation VV&A, Millennium Edition (available at http://vva.dmso.mil).

General Accounting Office, (1976), Report to the Congress: Ways to improve management of federally funded computerized models, report LCD-75-111, U.S. General Accounting Office, Washington, DC.

General Accounting Office, (1979), Guidelines for model evaluation, report PAD-79-17, U.S. General Accounting Office, Washington, DC.

General Accounting Office, (1987), DOD simulations: Improved assessment procedures would increase the credibility of results, report GAO/PEMD-88-3, U.S. General Accounting Office, and Washington, DC.

Ören, T.I., (1981), Concepts and criteria to access acceptability of simulation studies, Communications of the ACM, 24: 180–189.

Online links

http://portal.acm.org/citation.cfm?id=76751

http://en.wikipedia.org/wiki/Variance_reduction

Notes

Unit 13: Simulation Languages (I)

CONTENTS

Objectives

Introduction

13.1 Continues and Discrete Systems Simulation Languages

13.1.1 CODIS Framework

13.1.2 Discrete System Simulation

13.1.3 Example of Discrete-event Simulation (DES) – Models of Plasmodium Falciparum Malaria

13.2 Continuous Simulation Languages

13.3 Summary

13.4 Keywords

13.5 Self Assessment

13.6 Review Questions

13.7 Further Readings

Objectives

After studying this unit, you will be able to:

- Understand continuous and discrete simulation languages
- Discuss continuous simulation languages

Introduction

A computer simulation language describes the operation of a simulation on a computer. There are two major types of simulation: Continuous and discrete event though more modern languages can handle combinations. Most languages also have a graphical interface and at least simple statistical gathering capability for the analysis of the results. An important part of discrete-event languages is the ability to generate pseudo-random numbers and variates from different probability distributions.

13.1 Continues and Discrete Systems Simulation Languages

In models for discrete event dynamic systems (i.e., DEDS models) state changes occur at particular points in time whose values are not known a priori. As a direct consequence, (simulated) time advances in discrete 'jumps' that have unequal length.

In contrast, with models that emerge from the domain of continuous time dynamic systems (i.e., CTDS models), state changes occur continuously (at least in principle) as time advances in a continuous fashion over the length of the observation interval. It must, however, be realities introduced by the computational process. It is simply infeasible for any practical procedure to actually yield data at every value of time within the continuum of the observation interval.

Notes

Thus, from the perspective of the observer, state changes do apparently occur with discrete 'jumps' as the solution unfolds over the observation interval. Our presentation in this textbook may give the erroneous impression that models neatly separate into the two broad categories that we refer to as DEDS models and CTDS models. This is an oversimplification. There is, in fact a third category of models that are usually called combined models where the name reflects the combination of elements from both the discrete and continuous domains. As an illustration consider the parts in a manufacturing plant that move from one workstation to another on the way to assembly into a final product. At these workstations, queues form and the service function provided by the workstation may have random aspects (or may become inoperative at random points in time). Thus the basic elements of a DEDS model are present. At some workstations the operation may involve heating the part to a high temperature in a furnace. This heating operation and the control of it would best fall in the realm of a CTDS model. Hence the overall model that is needed has components from the two basic domains.

Work on the development of modelling formalisms and tools for handling this third category of combined models has a long history. The interested reader wishing to explore this topic in more detail will find relevant discussions in Cellier, Ören and Praehofer, the initial portion of a simulation experiment.

Modern systems that may be found in various domains like automotive, defense, medical and communications, integrate continuous and discrete models. In a recent ITRS study covering the domain of mixed continuous discrete systems, the conclusion is a "shortage of design skills and productivity arising from lack of training and poor automation with needs for basic design tools" as one of the most daunting challenges in this domain (ITRS, 2003). One of the main difficulties in the definition of CAD tools for Continuous/Discrete (C/D) systems is due to the heterogeneity of concepts manipulated by the discrete and the continuous components. Therefore, in the case of validation tools, several execution semantics have to be taken in consideration in order to perform global simulation:

1. In Discrete Models (DM), the time represents a global notion for the overall system and advances discretely when passing by time stamps of events, while in Continuous Models (CM), the time is a global variable involved in data computation and it advances by integration steps that may be variable.
2. In discrete models, processes are sensitive to events while in continuous models processes are executed at each integration step.

Currently, co-simulation is a popular validation technique for heterogeneous systems. This technique was successfully applied for discrete systems, but very few applied it for C/D systems. The cosimulation allows joint simulation of heterogeneous components. This requires the elaboration of a global execution model, where the different components communicate through a co-simulation bus via simulation interfaces performing adaptation. For C/D systems co-simulation, the simulation interfaces have to provide efficient synchronization models in order to cope with the heterogeneous.

This Unit presents CODIS (Continuous/Discrete Systems simulation), a co-simulation framework for C/D systems validation. This framework assists designers in building global simulation models.

Did u know? **What is Simulink?**

The supported simulators are Simulink for continuous models and OSCI SystemC simulator for discrete models.

Notes

13.1.1 CODIS Framework

Synchronisation and Generic Architecture for C/D Simulation in CODIS

For an accurate synchronisation, each simulator involved in a C/D co-simulation must consider the events coming from the external world and it must reach accurately the time stamps of these events. We refer to this as events detection. These time stamps are the synchronization and communication points between the simulators involved in the co-simulation. Therefore, the continuous simulator, Simulink, must detect the next discrete event (timed event) scheduled by the discrete simulator, once the latter has completed the processing corresponding to the current time. In case of SystemC, these events are: clock events, timed notified events, events due to the wait function. This detection requires the adjustment of integration steps in Simulink. The discrete simulator, SystemC, must detect the state events. A state event is an unpredictable event, generated by the continuous simulator, whose time stamp depends on the values of the state variables (ex: a zero-crossing event, a threshold overtaking event, etc.). This implies the control of the discrete simulator advancement in time: in stead of advancing with a normal simulation step, the simulator has to advance precisely until the time stamp of the state event (see Figure 13.1).

Figure 13.1: C/D Synchronisation in CODIS

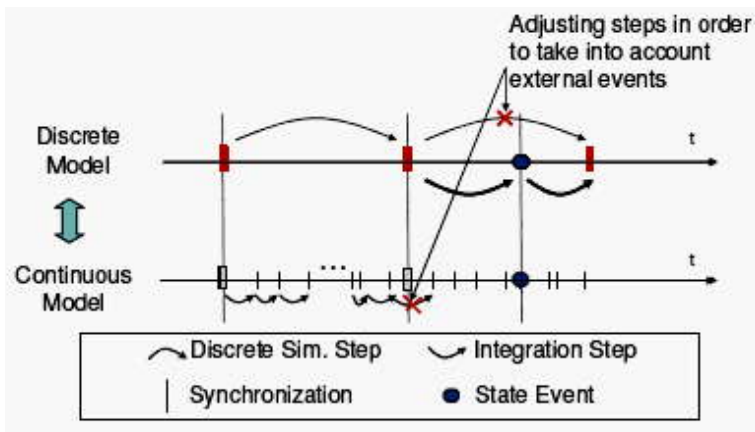
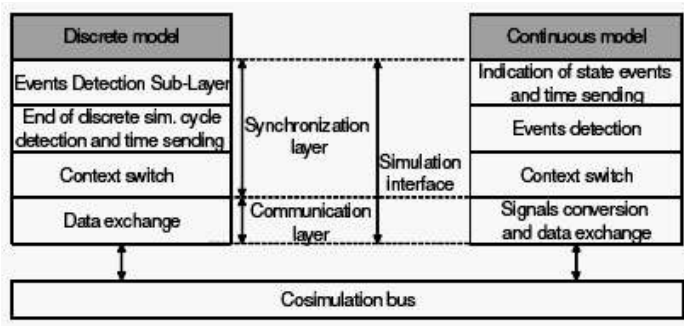


Figure 13.2 illustrates the generic architecture used in CODIS for the C/D simulation. CM and DM communicate through a co-simulation bus via simulation interfaces. Each simulation interface presents two main layers:

1. The synchronization layer provides the synchronisation requirements discussed above for both CM and DM; this layer is composed of three sub-layers, each of them achieving an elementary functionality for synchronisation.
2. The communication layer is in charge of sending/receiving data between CM and DM. More details on synchronization and CODIS simulation architecture may be found in Bouchimma et al., 2005.

Notes

Figure 13.2: Generic Architecture for Accurate C/D Simulation

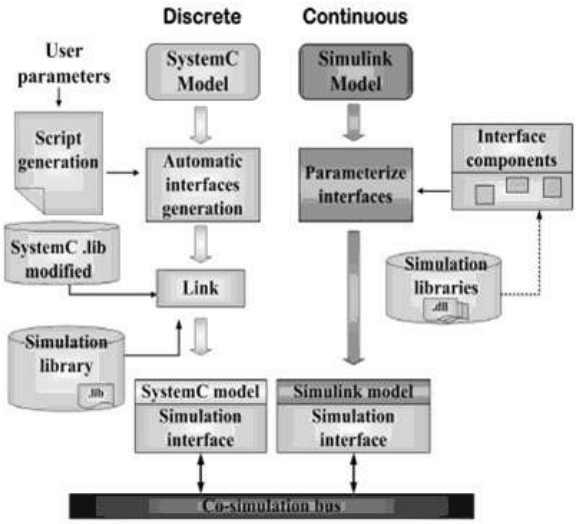


Simulation Model Generation in CODIS

Based on the presented synchronization model and the generic architecture, a flow for automatic generation of simulation models was implemented in CODIS (see Figure 13.3). The inputs of the flow are the CM in Simulink and the DM in SystemC. The output of the flow is the global simulation model, including the co-simulation bus and the simulation interfaces.

The interfaces are generated by composing elements from the CODIS library. These elements implement the layers in Figure. They may be customized in terms of number of ports and their data type. The interfaces for DM are automatically generated by a script generator that has as input user defined parameters. The model is compiled and the link editor calls the library from SystemC and a static library called "simulation library" (see Figure 13.3). The interfaces for Simulink are functional blocks programmed in C++ using S-Functions. These blocks are manipulated like all other components of the Simulink library. The user starts by dragging the interfaces from the library into the model's window, then parameterizes them and finally connects them to the model inputs and outputs. Before the simulation, the functionalities of these blocks are loaded by Simulink from the ".dll" dynamically linked libraries (see Figure 13.3).

Figure 13.3: Flow for Simulation Models Generation



Notes

Experimental Results

To analyze the capabilities of the proposed framework, we used two illustrative applications: a manipulator arm controller and a Δ/Σ_converter.

To evaluate the performances of simulation models generated in CODIS, we measured the overhead given by the simulation interfaces. The overhead caused by the Simulink integration step adjustment when detecting a SystemC event has been measured in a maximum of 10% of total simulation time. The overhead caused by IPC (Inter Process Communication) used for the context switch and the communication layers has been measured in order of maximum 20% of the total simulation time.

Notes The cost of the added synchronization functionality in the case of SystemC is negligible and does not exceed 0.02% of the total simulation time.

13.1.2 Discrete System Simulation

Discrete Event Simulation (**DES**) concerns the modelling of a system as it evolves over time by representing the changes as separate events. This is the opposite of Continuous Simulation where the system evolves as a continuous function (differential).

In discrete-event simulation, the operation of a system is represented as a chronological sequence of events. Each event occurs at an instant in time and marks a change of state in the system. For example, if an elevator is simulated, an event could be "level 6 button pressed", with the resulting system state of "lift moving" and eventually (unless one chooses to simulate the failure of the lift) "lift at level 6".

A common exercise in learning how to build discrete-event simulations is to model a queue, such as customers arriving at a bank to be served by a teller. In this example, the system entities are CUSTOMER-QUEUE and TELLERS. The system events are CUSTOMER-ARRIVAL and CUSTOMER-DEPARTURE. (The event of TELLER-BEGINS-SERVICE can be part of the logic of the arrival and departure events.) The system states, which are changed by these events, are NUMBER-OF-CUSTOMERS-IN-THE-QUEUE (an integer from 0 to n) and TELLER-STATUS (busy or idle). The random variables that need to be characterized to model this system stochastically are CUSTOMER-INTERARRIVAL-TIME and TELLER-SERVICE-TIME.

A number of mechanisms have been proposed for carrying out discrete-event simulation; among them are the event-based, activity-based, process-based and three-phase approaches (Pidd, 1998). The three-phase approach is used by a number of commercial simulation software packages, but from the user's point of view, the specifics of the underlying simulation method are generally hidden.

Task Analyze in discrete-event simulation, the operation of a system is represented as a chronological sequence of events?

Notes

Components of a Discrete-event Simulation

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

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

Events List: The simulation maintains at least one list of simulation events. This is sometimes called the pending event set because it lists events that are pending as a result of previously simulated event but have yet to be simulated themselves. An event is described by the time at which it occurs and a type, indicating the code that will be used to simulate that event. It is common for the event code to be parameterised, in which case, the event description also contains parameters to the event code.

When events are instantaneous, activities that extend over time are modeled as sequences of events. Some simulation frameworks allow the time of an event to be specified as an interval, giving the start time and the end time of each event.

Single-threaded simulation engines based on instantaneous events have just one current event. In contrast, multi-threaded simulation engines and simulation engines supporting an interval-based event model may have multiple current events. In both cases, there are significant problems with synchronization between current events.

The pending event set is typically organized as a priority queue, sorted by event time. That is, regardless of the order in which events are added to the event set, they are removed in strictly chronological order. Several general-purpose priority queue algorithms have proven effective for discrete-event simulation, most notably, the splay tree. More recent alternatives include skip lists and calendar queues.

Typically, events are scheduled dynamically as the simulation proceeds. For example, in the bank example noted above, the event CUSTOMER-ARRIVAL at time t would, if the CUSTOMER_QUEUE was empty and TELLER was idle, include the creation of the subsequent event CUSTOMER-DEPARTURE to occur at time t+s, where s is a number generated from the SERVICE-TIME distribution.

Random-number Generators

The simulation needs to generate random variables of various kinds, depending on the system model. This is accomplished by one or more pseudorandom number generators. The use of pseudorandom numbers as opposed to true random numbers is a benefit should a simulation need a rerun with exactly the same behaviour.

One of the problems with the random number distributions used in discrete-event simulation is that the steady-state distributions of event times may not be known in advance. As a result, the initial set of events placed into the pending event set will not have arrival times representative of the steady-state distribution. This problem is typically solved by bootstrapping the simulation model. Only a limited effort is made to assign realistic times to the initial set of pending events. These events, however, schedule additional events, and with time, the distribution of event times approaches its steady state. This is called bootstrapping the simulation model. In gathering statistics from the running model, it is important to either disregard events that occur before the steady state is reached or to run the simulation for long enough that the bootstrapping behavior is overwhelmed by steady-state behavior.

Notes

Caution The use of the term bootstrapping can be contrasted with its use in both statistics and computing.

Statistics

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

Ending Condition

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

Simulation Engine Logic

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

Start

1. Initialize Ending Condition to FALSE.
2. Initialize system state variables.
3. Initialize Clock (usually starts at simulation time zero).
4. Schedule an initial event (i.e., put some initial event into the Events List).

"Do loop" or "While loop"

While (Ending Condition is FALSE) then do the following:

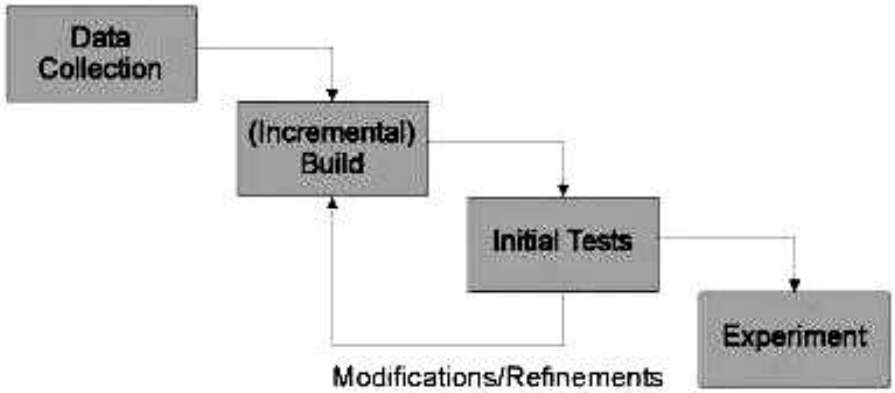
1. Set clock to next event time.
2. Do next event and remove from the Events List.
3. Update statistics.
4. End
5. Generate statistical report.

Application Areas/Common Uses

Well-known examples of Simulation are Flight Simulators, Fleet Management and Business games. However, there are a large number of potential areas for Discrete Event Simulation. One of the main areas currently being explored is in designing new manufacturing areas, especially where high capital investment is involved. For example, if a company wishes to build a new production line, then the line can be first simulated to assess feasibility and efficiency. The diagram below shows the key stages in using Discrete Event Simulation. It can be noted that this bears a strong resemblance to other simulation techniques and other analysis program development methodologies (prototype method) [Somerville, 1992].

Notes

Figure 13.4: Stages used in Discrete Event Simulation



Key Principles

Although, discrete event simulation could conceivably be carried out by hand it can be computationally intensive, therefore will invariably involve computers and software. The software could be a high level programming language such as Pascal or a specialised event/data driven application, such as iBright Ltd's 'baseSim' (Monte Carlo Simulation). The five key features found in the software simulation model are:

1. ***Entities:*** Representations of real-life elements e.g. in manufacturing these could be parts or machines.
2. ***Relationships:*** Link entities together e.g. a part may be processed by a machine.
3. ***Simulation Executive:*** Responsible for controlling the time advance and executing discrete events.
4. ***Random Number Generator:*** Helps to simulate different data coming into the simulation model. Important that the random data can be reproduced in different simulation runs.
5. ***Results & Statistics:*** Important in validating the model and for providing performance measures.

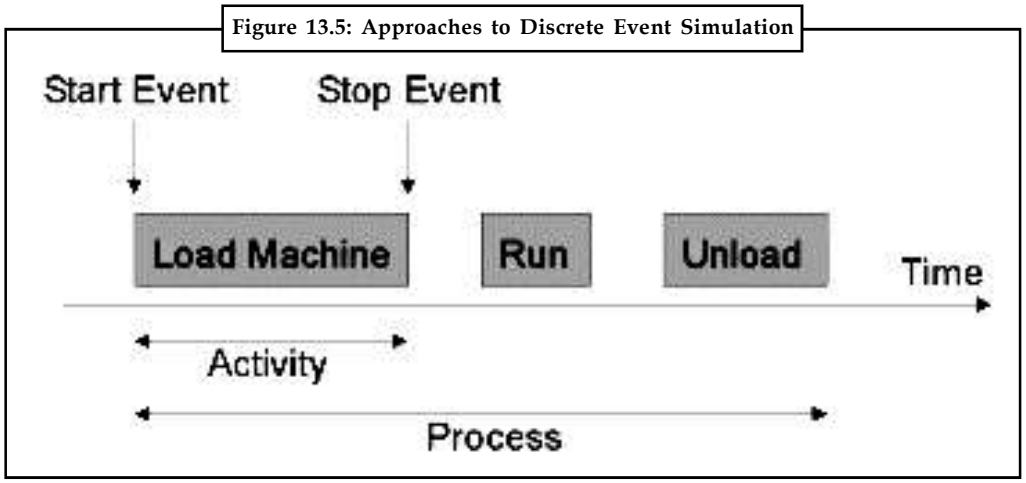
The simulation executive may operate in one of two manners [Ball, 1996]:

1. ***Time Slicing:*** Advances the model by a fixed amount each time, regardless of the absence of any events to carry out.
2. ***Next Event:*** Advances the model to the next event to be executed, regardless of the time interval. This method is more efficient than Time Slicing, especially where events are infrequent, but can be confusing when being represented graphically

Notes Processes that take different times will appear to happen in the same time frame if the stop event is the next event after the start event.

Notes

Figure 13.5: Approaches to Discrete Event Simulation



There are also three approaches to describing the discrete simulation, see the Diagram above [Pidd, 1992].

1. ***Event:*** This approach describes an instantaneous change, usually from a stop event to a start event. This is the most common one used, easy to understand and efficient and is acceptable to implement.
2. ***Activities:*** Represents duration. Essentially groups a number of events in order to describe an activity carried out by an entity e.g. a machine loading. This approach is easy to understand and to implement but is not efficient.
3. ***Process:*** These approach groups activities to describe the life cycle of an entity e.g. a machine. This is less common and more difficult to plan and implement, but is generally thought to be the most efficient.

More Common Uses of DES

1. ***Diagnosing Process Issues:*** Simulation approaches are particularly well equipped to help users diagnose issues in complex environments. The Goal (Theory of Constraints) illustrates the importance of understanding bottlenecks in a system. Only process 'improvements' at the bottlenecks will actually improve the overall system. In many organizations bottlenecks become hidden by excess inventory, overproduction, variability in processes and variability in routing or sequencing. By accurately documenting the system inside a simulation model it is possible to gain a bird's eye view of the entire system.

   A working model of a system allows management to understand performance drivers. A simulation can be built to include any number of performance KPIs such as: worker utilization, on-time delivery rate, scrap rate, cash cycles, and so on.

2. ***Hospital Applications:*** An operating theater is generally shared between several surgical disciplines. Through better understanding the nature of these procedures it may be possible to increase the patient throughput. Example: If a heart surgery takes on average four hours, changing an operating room schedule from eight available hours to nine will not increase patient throughput. On the other hand, if a hernia procedure takes on average twenty minutes providing an extra hour may also not yield any increased throughput if the capacity and average time spent in the recovery room is not considered.
3. ***Custom Order Environments:*** Many systems show very different characteristics from day to day depending on the order mix. Many small orders may cause bottle-necks due to

Notes

excess changeovers. Large custom orders may require extra processing at a point where the system has particularly low capacity. Simulation modeling allows management to understand what changes 'on average' would have the largest impact and greatest return-on-investment.

4. ***Lab Test Performance Improvement Ideas:*** Many systems improvement ideas are build on sound principles, proven methodologies (Lean, Six Sigma, TQM, etc.) yet fail to improve the overall system. A simulation model allows the user to understand and test a performance improvement idea in the context of the overall system.

5. ***Evaluating Capital Investment Decisions:*** Simulation modeling is commonly used to model potential investments. Through modeling investments decision-makers can make informed decisions and evaluate potential alternatives.

   Often these decisions look at altering existing operations. Typically, a model of the current state is constructed. This 'current state' model is tested and validated against historical data. Once the model is operating correctly, the simulation is altered to reflect the proposed capital investments. This 'future state' model is then stress-tested to ensure the alterations perform as desired.

   Occasionally, organizations take on entirely new operations processes. These could be new Lean facilities, designed around new products or using new technology. In these cases only a 'future state' model is constructed. The testing and validation may require more analysis. There are companies and experts that specialize in simulation building who may be brought in to help.

6. ***Stress Test a System:*** Models can be used to understand how a system will be able to weather extraordinary conditions. A simulation can help management understand: large increases in orders, significant swings in product mix, new client delivery demands (i.e. 1 week lead times), and economic events (i.e. a multinational with operations in South America and Asia sees significant swings in currencies).

Visualisation

Visual Interactive Simulation (VIS) has been available since the late 1970's. Before this simulation models were simply 'black boxes' - data going in and results coming out. In such a scenario establishing credibility and confidence in the simulation model would not have been easy.

Using on-screen animations in a simulation model enables the status of the model to be viewed as it progresses e.g. a machine that breaks down may change its color to red. This enables visual cues to be passed back to the operator of the simulation model, so action could be taken. Additionally, visualisation is useful in convincing management of the model's credibility. For example, in manufacturing if the Directors can see a visualisation of the production line with widgets travelling down a conveyor belt, it would do more to sell the concept of the model than a 'black box', churning out data.

With VIS the prime motivation is not only portrayal of the running simulation model but also the interaction with it. For example, in using the above scenario, if the User wanted to see how the production line would run with an extra machine then he could simply 'plug in' a machine, at the appropriate position, and monitor the effect that this would have on the model.

Visual Interactive Modelling (VIM) takes this concept one stage further by allowing the model to be created interactively. This allows a model to be constructed by dragging (with a mouse)

Notes

'Entities' (machines, parts etc.) from a library onto a frame. The entities could then be connected in the desired order. Many of the advanced VIM simulation tools allow program code to be attached to the entities and events, therefore making the model potentially more sophisticated and flexible.

Did u know? **Uses of Visualization and Simulation**

Visualization and simulation are extensively used in the training of operational staff, especially where the training cannot be carried out in real life e.g. shutting down the reactor of a nuclear power station after an earthquake.

Object Oriented Simulation

Object Oriented techniques have been developed since the early 1960's as a result of simulation development (SIMULA). Until recently, the two were not coupled despite their original tandem development. There are currently only a handful of Object Oriented simulation applications available on a commercial basis; one of the most prominent of these is iBright Ltd's 'baseSim'.

The main difference between traditional program development and Object Oriented techniques is the way in which the data and the program code are stored and manipulated. In traditional software, the data and the program code are intermingled throughout the program, making data security and integrity difficult to achieve (it is sometimes possible for one procedure to cause knock-on effects as global data is changed). However, in Object Oriented simulation software all data and procedures relating to a single entity (object) are encapsulated within an object, with the object controlling its own interaction and data integrity permissions with other objects. Clearly, the methods inside the object could cause similar knock-on effects, if poorly implemented.

Object Oriented simulation tools, in particular iBright Ltd's 'baseSim', are very powerful as they make use of Object Oriented techniques such as modularity, class structure, inheritance, hierarchy and polymorphism.

Statement from a Company about Discrete-system Simulation

Our Discrete-system Simulator (DSS) is used by INTRACOM, the largest Greek telecom company. We shall expand DSS (in connection with the DSP library) to allow for the specification, simulation and performance testing of distributed protocols for almost any form of environment, including networks that allow hierarchical description, very fast networks and heterogeneous networks with mobile units. The visualisation facility of DSS will be enhanced so the user can see the protocol execution in very large networks in a meaningful way through, e.g., a hierarchical description and graph-drawing techniques. INTRACOM, and also INTRASOFT (the largest Greek software company), are interested in the new DSS and DSP library for (1) simulating and testing distributed banking applications, and (2) designing hardware for fast networks (routers and switches).

During the first year of ALCOM - IT, we established strong interactions with the industrial link INTRAKOM/INTRASOFT a major Telecom and Systems House conglomerate of Companies. After analysing the users' needs, we identified a large algorithmic area which is a subset of the area of on-line algorithms, namely the area of Call Admission Control (CAC) algorithms. Our group and, in general, the ALCOM team has had already shown significant research on the issue. Since the problem of Admission Control is currently addressing ATM network technology, we

Notes

decided to explore the transfer of algorithmic engineering technology in that area by specializing and focusing the ALCOM-IT produced DSS tool in the above direction. The new DSS tool aims to provide:

1. An abstract, clean and semantically correct high level model of any ATM network and of the on-line calls for connections.
2. A library of Call Admission Control algorithms in a form that allows them to him easily used by non - experts or to be used and tested by algorithms and networks designers.
3. The capability for the user Industry to design new Call Control protocols and/or test such designs.

The ATM technology is considered as the state of the art network technology that is expected to play an important role in the future networks. The ATM networks are fast packet switching networks achieving their speed by avoiding flow control and error checking at the intermediate nodes in a transmission. ATM operates in a connected mode, but a connection can only be set up and serviced if sufficient resources are available in order to preserve the quality of service to the previous accepted connections. This function is controlled by the Call Admission Control algorithms running in the ATM switches.

DSS provides an abstract model for the description of any ATM network which is independent of details of the underlying technology. It simulates the basic functionality of an ATM network which IS that in each time unit cells are produced from traffic generators or forwarded from the switches to their destination.

Under the scope of DSS an ATM network topology consists of links, ATM switches, terminals (workstation) and call/traffic generators (network applications). The critical characteristics of the topology such as the size of the buffers of the ATM switch, the bandwidth of the links, the virtual paths and virtual circuits over the links and the traffic parameters can be defined by the user to approach the behavior of the today's and the future network components. Each ATM switch is modelled as a Communicating Finite State Machine. A receipt of a CAC cell combined with its current state activates an action routine of the CAC algorithm.

Emphasis is given in the abstract modelling of traffic generation. Adversarial traffic leads the on-line algorithms to their worst case competitive ratio of performance (measured against ideal off-line algorithms that know the future). The DSS cannot simulate worst-case adversaries but can approximate their behaviour by exploiting certain distribution of call request of high Kolmogorov complexity.

Notes The DSS can of course use externally (pragmatic) generate call sequences.

13.1.3 Example of Discrete-event Simulation (DES) – Models of Plasmodium Falciparum Malaria

We develop discrete-event simulation models using a single "timeline" variable to represent the Plasmodium falciparum lifecycle in individual hosts and vectors within interacting host and vector populations. Where they are comparable our conclusions regarding the relative importance of vector mortality and the durations of host immunity and parasite development are congruent with those of classic differential-equation models of malaria, epidemiology. However, our results also imply that in regions with intense perennial transmission, the influence of mosquito mortality on malaria prevalence in humans may be rivaled by that of the duration of host infectivity.

Notes

Introduction

The proximate cause of malaria in a human is the presence of Plasmodium parasites, which may appear after an infective bite by an Anopheles mosquito. The probability that parasites are transmitted from a mosquito to a human in any given interaction necessarily depends on the probability that parasites were transmitted from a human to that mosquito in some previous human-mosquito interaction, which in turn depends on earlier links in the chain of transmission, and thus on densities of infectious and susceptible humans and mosquitoes, on innate and acquired host immunity, strains and species of parasite and vector, sea and environmental factors, and so forth. That is, human malaria is characterized by hierarchies of dynamic processes, occurring on diverse time scales within and between heterogeneous populations.

Here we develop models of malaria epidemiology by depicting the biology of parasite development in individual humans and mosquitoes, and the corresponding interactions between humans and mosquitoes. We distinguish each host and vectorunit by its state within a characteristic repertoire of states, we define a set of probabilistic interactions that may effect state transitions in interacting units, i.e., parasite transmission, and we simulate multiple interactions among multiple entities. This scheme requires no special computing resources, but it provides realistic sampling processes by representing large, finite populations of individuals, and allows relatively realistic degrees of complexity in population-level dynamics to emerge from simple, transparent representations of individual-level malaria infections.

Such discrete-event models complement the differential-equation "compartment" models that have made such enormous contributions to our understanding of malaria transmission. The most influential of these models, Macdonald's refinement of the Ross archetype, focused a global malaria-eradication campaign on reducing adult mosquito survivorship. Though this campaign succeeded brilliantly in many temperate and subtropical regions, it often failed elsewhere, particularly in regions with intense perennial transmission. Macdonald published posthumously that "a powerful tool for the design of eradication and control programs, and for the analysis of difficulties in them, could be produced by the extension of dynamic studies using computer techniques." Our objectives here are to introduce a basic discrete-event simulation model, compare its results in conditions of intense perennial transmission to those of differential-equation models, and investigate circumstances in which the utility of such abstractions as average individuals and infinite populations might be challenged.

Task Explain the working of basic discrete-event simulation model.

Design

Plasmodium infection of a human begins with a small inoculum of sporozoites from the salivary glands of a blood-feeding female Anopheles mosquito. The sporozoites penetrate liver cells, and in hepatic schizogony transform and multiply to produce thousands of free merozoites. Each of these merozoites invades a red blood cell, completes another round of multiplication (in erythrocyte schizogony), then bursts the cell, releasing 8 to 32 more merozoites to invade more red blood cells. This asexual blood cycle may be repeated many times, in the course of which some invading merozoites may instead develop into the sexual, non-replicating transmissible stages known as gametocytes. If viable gametocytes of each sex are taken up by a feeding Anopheles, fertilization may produce the zygotes from which infective sporozoites arise within the mosquito, in sporogony.

Our discrete-event models identify the potential states of an individual host or vector with the sequential phases of a P. falciparum malaria infection in each such that a single variable that

Notes

tracks an individual's progress along an infection timeline also sketches the life cycle of the parasite. Because the fundamental question for two-party interactions is whether one participant is infectious and the other susceptible with respect to a parasite, the state of each host and vector thereby represents the presence or absence of a parasite life cycle stage appropriate for infectivity or susceptibility. Unit interactions simply mimic host-vector contact – i.e., the mosquito taking a blood meal – and each corresponding potential state transition the potential transmission of a parasite between host and vector. Actual transmission of a parasite in any given host-vector interaction may involve chance as well as additional biological factors; our representations here include a host immune component and a vector mortality function.

Infection timelines express the dynamic individual-level states that lead to the compartments of traditional population-level models; their single-valued state representations allow compact, efficient data structures that permit representations of very large interacting populations. Simple binning can translate the individual state descriptions into familiar population-level classes such as "infected" and "infectious," and allows ready calculation of prevalence and other epidemiological measures.

The full infection timeline in our malaria models corresponds to the "incubation interval" in Macdonald's model, i.e., "the complete period from the occurrence of infective gametocytes in one case to the development of infective gametocytes in the secondary cases derived from it," comprising "the period of extrinsic development of the parasite in the mosquito; the pre-patent period, or incubation period as it is normally known, in man; and any interval between the patency of asexual parasites and the development of fully infective gametocytes." Our malaria models represent this full interval, and the parasite life cycle, as a circuit from position "0" on the host timeline through position "0" on the vector timeline and back to the host "0," by way of two blood meals.

Three of the five temporal parameters of our basic model correspond directly to those in Macdonald's:

1. ***Vector Delay (DV)*** is the length of the interval between infection (gametocyte ingestion) and the onset of infectivity (sporozoite migration) in a vector, i.e., Macdonald's "period of extrinsic development," above;
2. ***Host Delay (DH)*** is the length of the interval between infection (sporozoite inoculation) and the onset of infectivity (gametocyte maturation) in a host, i.e., Macdonald's "pre-patent period" and subsequent "interval," above;
3. ***Vector Survivorship (VS)*** is the daily probability of a mosquito's survival, i.e., synonymous with Macdonald's expression for the "probability of survival through one day."
4. ***Host Window (WN)*** is the duration of a host's infectivity to vectors, from the first to the final presence of infective gametocytes. This factor was addressed by Macdonald at most indirectly, in terms of recovery rates and infective proportions; WN is closely analogous to the "loss rate" a_1 in the Dietz et al. extension of Macdonald's model.
5. ***Host Immunity (IM)*** defines a host's susceptibility to re-infection through the daily decay of a blocking immunity. This was addressed at most indirectly in Macdonald's model; IM closely resembles the "loss rate" ã in the Aron and May extension.

Implementation

Populations of hosts and vectors are represented by arrays that contain the state of each constituent host or vector with respect to its stage of infection, with these states represented on the corresponding infection timeline by the variables h and v, respectively. Thus a value $0 < h < D_H$ indicates that the host is infected but not yet infectious, $-WN < h <= 0$ that the host is (infected

Notes

and) infectious, and any other value of h that the host is not currently infected. For h < –WN, the host's immunity to re-infection is declining, as explained below. Similarly, $-D_V < v < 0$ indicates a vector that is infected but not infectious, $v <= -D_V$ a vector that is (infected and) infectious, and any other value of v a vector that is not currently infected. The basic model does not encompass the possibility of mosquito immunity. At each successive day of a simulation run, all h and v values are decremented by 1, advancing them through the appropriate delay, window or decay periods.

Parasite transmission is represented as a single state-altering interaction between an individual host and an individual vector. At each time step the model selects a mosquito at random and a human at random for an interaction. An initial parameter, V_B, the total number of daily host-vector interactions, sets the number of times this random pair selection is repeated each day. Only two pairs of host-vector states permit interactions that may directly induce state transitions:

1. If the mosquito is infectious, then with a probability based on the immune state of the human, that human changes state and becomes infected, setting h to D_H;
2. If the human is infectious, then with some probability, taken to be 1 in the basic model, that mosquito changes state and becomes infected, setting v to 0.

In the other possible state combinations there are no transitions: super infection of a human already infected with a given strain does not reset h, and super infection of a mosquito already infected with a given strain does not reset v. Hence, in humans, super infections with identical strains affect neither immune responses nor gametocyte production, and in mosquitoes they simply accumulate, neither accelerating nor delaying the onset of infectiousness. This follows Macdonald's conventions, but an advantage of our modeling approach is that virtually any super infection scheme can be implemented. Host and vector populations are sampled with replacement, so multiple feedings by a single mosquito as well as multiple bites on a single human may occur within a single day; both phenomena occur in nature.

Any host not previously infected is considered wholly susceptible. The decay of immunity in a previously-infected host is represented by a changing probability of re-infection of that host when bitten by an infectious mosquito. During an infection, i.e., for h greater than –WN, this probability is considered 0, consistent with the absence of state change noted above; after an infection is cleared, i.e., when the host state reaches –WN, this probability exponentially, asymptotically approaches 1. For h less than –WN, we model the probability of infection as $1-a^{im\,(hewn)}$; note that with h always more negative than –WN, the exponent is always negative. The immune half-life, (ln2)/IM, is the number of days required for this probability to reach 0.5. When an infectious mosquito is paired with any human, that human's probability of acquiring an infection is calculated, as above, and a uniform random number between 0 and 1 is compared with that probability. If the random number is less than the calculated probability, then the human becomes infected, setting the host state h to D_H; otherwise the host state remains unchanged.

At each time step for each vector, a uniform random number between 0 and 1 is compared with the probability of death, (1–VS). If the random number is less than that probability, then the mosquito is removed from the population and replaced with a new mosquito, maintaining the vector population at constant size. The vector half-life, (ln2)/ (1–VS), is the number of days over which mortality reduces any given cohort of new mosquitoes by half.

The initial parameters N_H and N_o fix the host and vector population sizes, respectively. Initially no mosquitoes are infected, and an initial fraction of the host population is infected by setting each h to a random value between D_H and 0. The initial state of each uninfected host is set to a large negative value, implying no initial immunity. We coded the models in C++ and ran them on an IBM-compatible PC under Windows 95.

Notes

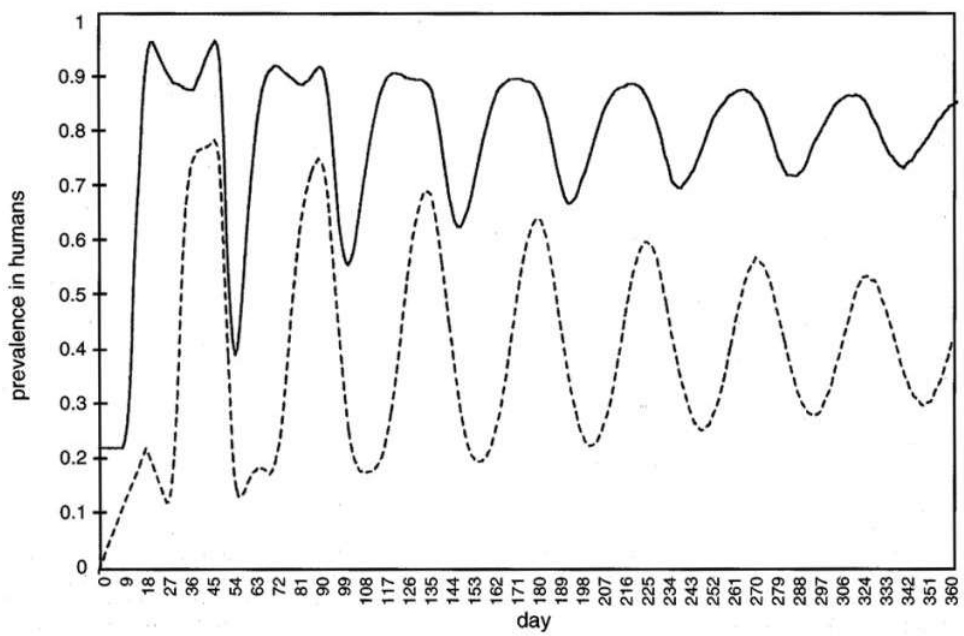
Did u know? **What is parasite transmission?**

Parasite transmission is represented as a single state-altering interaction between an individual host and an individual vector.

Parameter Values and Initial Conditions

Figure 13.6 shows the typical pattern of damped oscillations in prevalence for the wide ranges of parameter values and initial conditions considered here. The results presented in this paper are from data collected every tenth day during the last 100 days of 365-day runs. Because varying the delay parameters in this context leads to phase differences with little effect on such prevalence data, we generally fixed D_H = 20 and D_V = 10, then examined the relative influence of the remaining temporal parameters on prevalence by systematically varying these parameter values within a large plausible subspace. We most closely examined parameter values over a range from 50% to 150% of a plausible mean value for each, such that the values of the host window (WN) ranged from 10 to 30 days, vector half-life from 5 to 15 days (i.e., VS from 0.954 to 0.861), and host immunity half-life from 50 to 150 days (i.e., IM from 0.014 to 0.005). For each <VS, WN, IM> point, we performed 100 replicate runs; standard deviations in prevalence about each of the mean <VS, WN, IM> points used in our analyses were <0.01. We used Mathematical 2.0 (Wolfram Research, Champaign IL) to obtain best-fit planes to surfaces of 231 prevalence points for each pair-wise combination of parameters at five levels of the third parameter.

Figure 13.6



In Figure 13.6, an example of the time course of the overall malaria prevalence (solid line) and of the prevalence of infectious stages (dashed line) in a human population, demonstrating damped oscillations. The levels and rates at which prevalence stabilize depend.

We generally set N_V = 5000 and V_B = 2500, in accord with the usual idealized two-day geotropic cycle for Anopheles (i.e., N_V/V_B = 5000/2500), and set N_H = 500 (i.e., N_V/N_H = 10), with 25% of the hosts and none of the vectors initially infected. Varying the proportion(s) initially infected had

Notes

little effect on prevalence data in this context, though as one would expect, the further the initially-infected proportion(s) from the bounds within which the later prevalence stabilize, the greater the range(s) of earlier oscillations. As discussed below, the results are also valid for a range of vector population sizes and interaction frequencies.

In the context of perennial single-strain transmission, the parameter ranges considered here lead to human prevalence of 50 % to 85%, a typical range in several tropical regions. Recent results with PCR-based detection methods indicate that these high prevalence are even more common than had been assumed based on surveys using conventional microscopy. Our results show that, as expected under such conditions, prevalence in humans increases as mosquito survivorship increases (as in Macdonald's model).

Notes The prevalence in vectors increases as either the period of host infectivity or the rate of host immune decay increases.

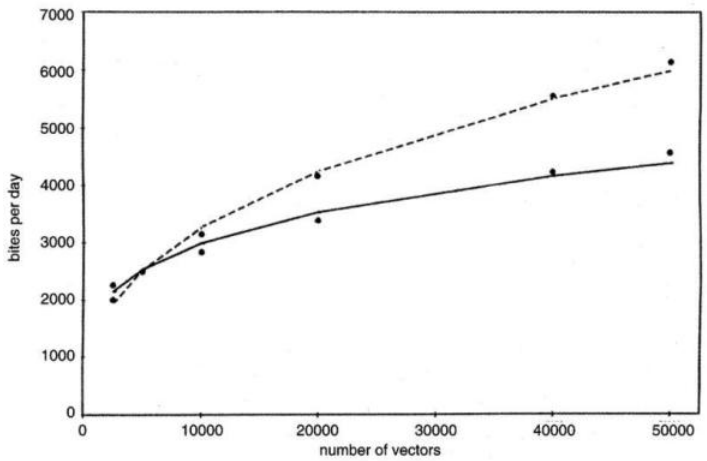
Results – Vector Population Size

It has long been recognized that the differing degrees of anthropophily among Anopheles species convolve with relationships between mosquito lifespan, parasite development cycles and other factors such that at any given moment a relatively small fraction of a mosquito population actually contributes to the transmission of malaria in a human population. Our models not only manifest similar behavior, but do so in a manner that allows quantification and scaling of the vector population required to maintain several key epidemiologic characteristics of infection in a given human population. All else being equal, maintaining a constant prevalence of infection in humans and a constant number of infectious mosquito bites per human per day requires a total number of mosquito bites per day, V_B, that scales less than linearly with the total mosquito population size, N_V. That is, some of the effects of enormous populations of vectors may be modeled without fully representing each constituent, such that it may be possible to consider "effective" vector population sizes in epidemiological as well as population-genetic terms.

Figure 13.6 plots prevalence isoclines for two extremes of mosquito survivorship, showing the joint values of V_B and N_V equivalent to a population of 5,000 mosquitoes feeding only on the given human population, with an idealized blood meal cycle (N_V/V_B) of two days. That is, in terms of the prevalence of infection in humans and the number of infectious mosquito bites per human per day in this context, 5,000 is the "effective" mosquito population size of each of these $<N_V, V_B>$ combinations. The six $<N_V, V_B>$ points shown for each parameter set closely follow an exponential (power-law) function, $V_B = aN_V^b$. Equivalently, for a given human population size and number of daily mosquito bites, the scaling formula $N_V = (V_B/a)^{(1/b)}$ approximates all population sizes of mosquitoes synonymous in key epidemiological terms with a given population size of mosquitoes biting only humans; other population sizes imply zooprophylaxis or different gonotrophic cycles. We estimated values for the constants a and b using Systat 4.0 (Systat Inc., Evanston, IL), and found that maintaining constant "effective" vector population sizes requires that the total number of daily host-vector interactions increase by less than the square root of the total vector population size.

Notes

Figure 13.7

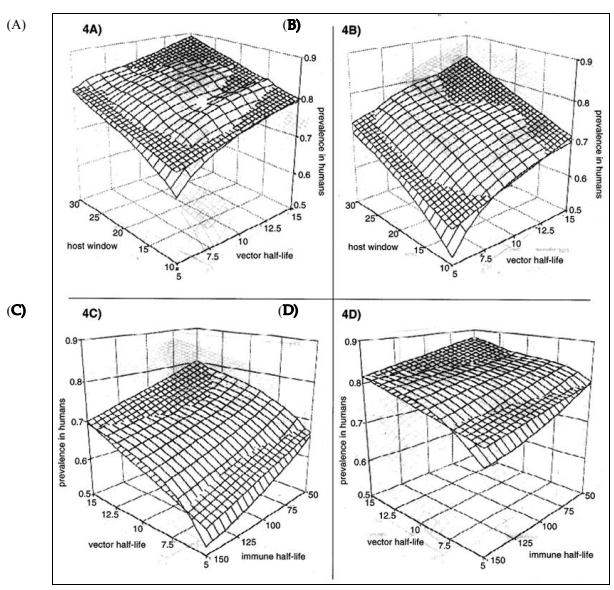


In Figure 13.7, an illustration of an "effective" mosquito population size (see text), with the parameter values N_H = 500, D_V = 10, D_H = 20, WN = 15, IM = 0.01, and VS = 0.95 or 0.86.

Results – Prevalence in Humans

Figure 13.8 shows the overall prevalence of infection in humans as a function of mosquito survivorship, the duration of human infectivity (host window) and the duration of human immunity to reinfection. Each response surface shows 231 prevalence points, each of which is an average of 100 simulation runs, resulting in a standard deviation of 0.01 or less. We have shown the best-fit planes not to suggest the existence of any underlying linear process, but simply to allow comparisons among the average effects of each parameter. The average effects on the overall prevalence of infection in humans of variations in vector mortality and variations in the duration of host infectivity are of similar magnitude, while the duration of host immunity is less influential.

Figure 13.8



Notes

In Figure 13.8, Plots of overall malaria prevalence in a human population, with parameter values $D_V = 10$ and $D_H = 20$, and initial conditions $N_H = 500$, $N_V = 5000$ and $V_B = 2500$.

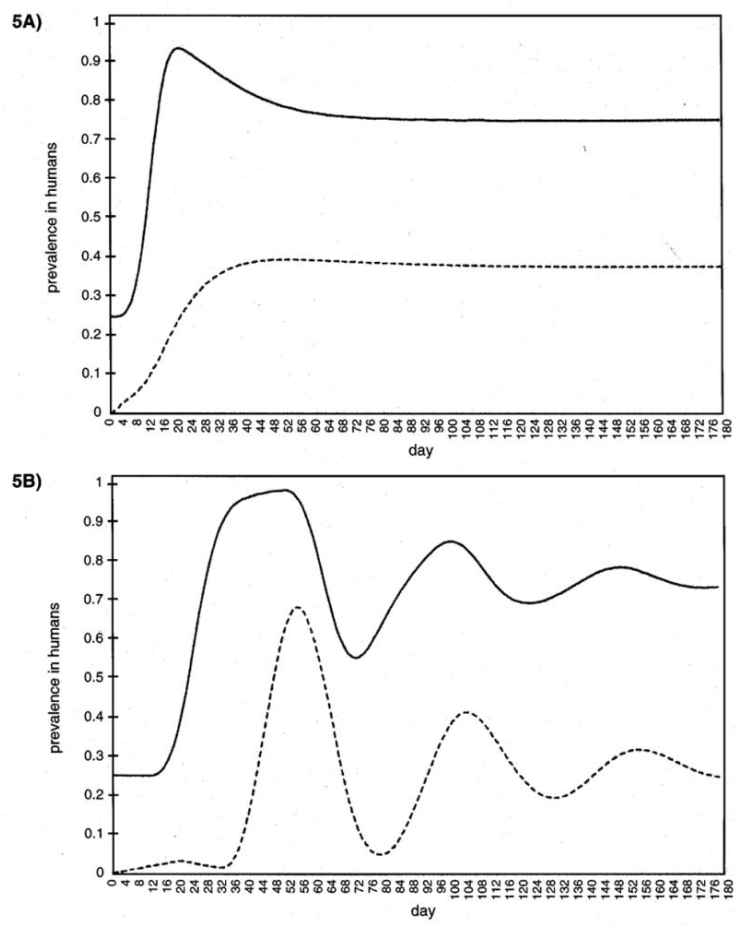
Did u know? **Did you know the range of values of the remaining temporal parameters of a plausible mean value?**

50% to 150%

Comparisons with Differential-equation Models

Figure 13.9 illustrates the dynamics of classic differential-equation and differential-delay-equation "compartment" models analogous to the discrete-event model; the models we have constructed are described in detail in the Appendix, as is Macdonald's model. Parameter values for Figure were calculated such that the prevalence curves are as closely comparable as possible to those shown in the Figure.

Figure 13.9



In Figure 13.9, an example of the time course of the overall malaria prevalence (solid lines) and the prevalence of infectious stages (dashed lines) in a human population, in (A) classic differential-equation and (B) differential-delay-equation "compartment".

Notes

The obvious differences are in the three rates of decay of oscillation. Given the parameter values and initial conditions of Figure 13.9, the classic differential-equation and differential-delay-equation models settle at equilibrium within four and 10 to 12 months, respectively, with an overall human prevalence of 75%. Continuing out 10 years with the parameter values used in Figure 13.9, an average over 25 runs shows overall human prevalence still oscillating slightly (within 1%) around an overall human prevalence of 82%.

In the discrete-event model, immunity is considered a continuous process of change in the probability that a previously exposed individual will block an infection when bitten by an infectious mosquito, i.e., the parameter IM represents the rate of decay of immune resistance to reinfection in an individual human. This operational view thereby incorporates the existence of partial immunity to reinfection, an aspect of epidemiology difficult to represent in terms of the discrete step transitions and flow rates of aggregate population-level models. That is, if adjacent discrete compartments are "totally immune" and "totally susceptible," and some quantity or proportion is lost from one to the other over each interval of time, conventional differential-equation models represent this loss as a rate of population flow between the two distinct states rather than as a process of decay within each individual in a population.

These two views of immunity cannot be reconciled, but the models that embody them can be compared by finding particular parameter values that relate an average duration of immunity in an aggregate in the simulation model in its stationary state, to an equilibrium flow rate in the differential-equation models. To do so requires the resolution of the full "age" distribution of intervals since infection over all "immune" individuals in the simulation, relative to each individual's probability of becoming infected. The dynamics of the three systems away from such fixed points of comparison may differ dramatically.

We developed two related forms of a differential-equation "compartment" model parallel to the discrete-event simulation model. The first, more traditional form is given by the equations:

$$dS/dt = qR - hFS,$$

$$dM/dt = hFS - kM,$$

$$dG/dt = kM - pG,$$

$$dR/dt = pG - qR,$$

$$dU/dt = b - hGU - dU,$$

$$dL/dt = hGU - cL - dL, \text{ and}$$

$$dF/dt = cL - dF,$$

where the dynamic variables S, M, G and R denote the proportions of susceptible, infected, infectious and immune humans, and U, L and F the proportions of susceptible, infected and infectious vectors, respectively. The parameters h, b and d represent daily rates of vector biting, natality and mortality, respectively; b/d gives the ratio of vectors to hosts (N_V/N_H). Flow rates between human compartments are represented by the parameters q, from immune to susceptible, k, from infected to infectious, and p, from infectious to immune; c represents the flow rate between the infected and infectious mosquito compartments. The equilibrium values for this model are:

$$S^* = dp(c + d)(d + hG^*)/(bch^2),$$

$$M^* = pG^*/k,$$

$$G = kq[bch^2 - d^2(c + d)^2]/\{h[bch(kp + kq + pq) + dkpq(c + d)^2]\},$$

$$R^* = pG^*/q,$$

Notes

$$U^* = b/(d + hG^*)$$

$$L^* = bhG^*/[(c + d)(d + hG^*)], \text{ and}$$

$$F^* = bchG^*/[d(c + d)(d + hG^*)].$$

The central problem in relating the population flow rates between compartments in the differential-equation models to the individual half-lives in each state in discrete-event models is that population transition rates in the latter depend upon infection histories, i.e., the "age" distribution of individual times since infection. It is possible to compare the models by equating average residence times in the "immune" state at an equilbrium, but their dynamics remain completely different.

Consider a "cohort" in the discrete-event model, i.e., all hosts infected on a single day, and let M = M(t) represent the fraction of that cohort still immune at time t, such that "dM/dt is the fraction losing immunity in dt, or, equivalently, is the probability density of the transition from the immune to the successor state. Then the average length of immunity is f_t(–dM/dt)dt, with the time integral taken from 0 to infinity.

The probability of an individual being infected on day t is $(1–e^{-qt})$, where q = IM, hence the number of cohort members being infected and thus moving from the immune to the successor state on day t is $M(1–e^{-qt})$. That is, M decays as dM/dt = "$M(1–e^{-qt})$, where q is the rate of decay of the probability of becoming infected once in the immune state (i.e., once the infection has been cleared). This cannot be substituted directly into the formula for the average length of immunity because this derivative includes the function M(t) itself. Therefore it is necessary to solve for M(t) first. By separation of variables:

1. $dM/M = – (1 – e^{-qt})\, dt$, and
2. $\text{In}(M) = – (t + e^{-qt})/q$

By substituting the initial condition M(0) = 1, $M = [e^{(1/q)}]/\exp[t+(e^{-qt}/q)]$ is the fraction of the cohort still immune, where exp(x) is synonymous with e^x, and hence:

$$\int t(-dM/dt)dt = \int t(1-e^{-qt})e^{(1/q)}\Big\}/\exp[t+(e^{-qt}/q)]dt$$

is the average time in the immune state.

Now let q_s represent this q(= IM), the rate of decay of immunity to reinfection in an individual, in the discrete-event model, and let q_c represent the flow rate q in the differential-equation model. Because in differential-equation models the immunity of any immune entity decays exponentially (i.e., if G = 0, $dR/dt = –q_cR$), the average time in the immune state in the discrete-event model, q_s, calculated by the integral above, is $(\ln 2)/q_c$. For example, to generate Figure 13.9 we evaluated the integral with q_s = IM = 0.01 (an individual host immunity half-life of 70 days in the discrete-event model), which yields an average time in the immune state of 12.55 days. Thus $q_c = \ln(2)/12.55 = 0.055$. To further illustrate the differences between an individual-level and a population-level time scale, note that a population average residence time of 70 days in the differential-equation models (i.e., $q_c = 0.01$) corresponds to a discrete-event-model immune half-life of 2,166 days.

Of course q_s and q_c can be equated in this manner only when the distribution of the individuals in the immune state over the times since they entered that state is uniform, which will usually be the case near an equilibrium (i.e., with hosts entering and leaving each state at a constant daily rate). As this is not likely to be true elsewhere, even with identical equilibria one would expect different system dynamics away from that point.

In this classic differential-equation model, translating the delays, D_V and D_H, and the host window of infectivity, WN, poses the problem of approximating a deterministic step function by a flow

Notes

rate in an exponential decay process. As above, assuming the system at equilbrium, we set p = k = (ln2)/WN = (ln2)/D_H, and c = (ln2)/D_V. With respect to vector mortality, because we assume that each mosquito has the same probability of dying each day we do not need to know the distribution, and we take this directly as the aggregate, exponential decay, i.e., d = (1–VS).

The second, differential-delay form replaces each of the parameters k, p and c with an explicit time lag corresponding to the host delay, the host window and the vector delay parameters in our simulation model, respectively, such that:

$$dS/dt = qR - hFS,$$

$$dM/dt = hFS - [hFS]_{t-k}$$

$$dG/dt = [hFS]_{t-k} - [hFS]_{t-(k+p)}$$

$$dR/dt = [hFS]_{t-(k+p)} - qR,$$

$$dU/dt = b - hGU - dU,$$

$$dL/dt = hGU - [1 - e^{-cd}][hGU]_{t-c} - dL, \text{ and}$$

$$dF/dt = [1 - e^{-cd}][hGU]_{t-c} - dF,$$

where the dynamic variables and other parameters are as above.

Here the system is underdetermined, i.e., no equilibrium can be calculated, and its dynamics depend on the initial conditions.

We obtained numerical approximations to solutions (Figure 13.9) by translating each of the models into a BASIC program, typically setting 25% of the hosts and none of the vectors initially infected, as in the discrete-event simulation model.

Macdonald's model can be expressed by the equations:

1. dX/dt = abm Y – X(abm Y + r), and
2. dY/dt = aX – Y(aX – Inp),

where the dynamic variable X represents, according to Macdonald, "the proportion of people affected," the dynamic variable Y its (implicit) counterpart in the vector population, and the parameters as follows, also quoted from Macdonald:

m	the anopheline density in relation to man,
a	the average number of men bitten by one mosquito in one day,
b	the proportion of those anophelines with sporozoites in their glands which are actually infective,
p	the probability of a mosquito surviving through one whole day, and
r	the proportion of affected people, who have received one infective inoculum only, who revert to the unaffected state in one day.

The crucial aspects of Macdonald's model are summarized in his formula for Z_0, the "basic reproduction rate" of malaria:

$$Z_0 = -(ma^2b)p^n/[r(\ln p)] = b/r)C,$$

where the parameter n represents "the time taken for completion of the extrinsic cycle," and C {= "$(ma^2)p^n/(\ln p)$} summarizes the "vectorial capacity" of malaria.

Macdonald derived Z_0 as an estimate of the average number of secondary cases arising in a very large population of completely susceptible humans following the introduction of a single primary case, and Z_0 = 1 as the transmission threshold, i.e., the value above which cases propagate and below which they recede. This formula for Z_0 holds that the influence of vector survivorship, p,

Notes

is greater than that of a or n, which are in turn greater than that of b, m, or r, and hence that vector survivorship is the single most important element in the basic reproduction rate of malaria. Macdonald's "affected" proportions do not distinguish between infected and infectious states, but his conclusion with respect to host infectivity was that: "Transmission can be altered by reduction of the mean period of infectivity of a case of malaria. The influence is, however, relatively small; the reproduction rate varies directly with the mean duration of infectivity, very great changes in which would be necessary to reduce the high' rates common in Africa and some other places below the critical level."

In terms of our model the "basic reproduction rate" is:

$$Z_0 = -k[V_B^2/(N_H N_V)][VS^{DV}/In(VS)] = kC$$

where "k" is an equivalent to the ratio b/r (see below).

Our parameters VS and D_V correspond directly to Macdonald's "p" and "n," and the ratios of our initial parameters N_V/N_H and V_B/N_V translate his "m" and "a," respectively (substituting "bites" for "men bitten"), such that for our model "ma^2" = $V_B^2/(N_H N_V)$. Macdonald's "b" is a measure of incidence (e.g. by its role in expressions for "inoculation rate" and "force of infection"), and "r" the reciprocal of the average duration of the "affected" state. Macdonald wrote that "in nature the value of the reproduction rate is greatly influenced by immunity altering the values of r and b," and in our model these proportions actually do vary dynamically with distributions of host immune states and infection histories, in a convolved, partly stochastic manner.

Therefore it is difficult to interpret "b" and "r" in terms of our model, particularly in terms of our parameters WN and IM. However, the C values considered here range from 3.7 to 29.2, in accord with field estimates, but see, so if we consider $1/(D_H+WN)$ roughly equivalent to r, then k ranges from 30b to 50b. Even if we consider the b-equivalent values in our model as ranging from 0.1 to 1 (k values from 3 to 50), field estimates span the resulting range of Z_0 values.

Did u know? **What is equilibrium?**

Equilibrium is the condition of a system in which competing influences are balanced.

Discussion

There are many useful approaches to modeling human malaria, and many differences among models constructed for different purposes, but some forms that are analogous are not equivalent: analogous classic differential-equation and differential-delay-equation models have different properties, and each has properties very different from those of the discrete-event models developed here, including different dynamics leading to the same equilibrium.

Among the factors in malaria epidemiology most difficult to represent in compartment models is the immunity of individuals. In the discrete-event simulation models, complex population-level dynamics emerge from a simple representation of individual-level malaria infections. Accordingly, we can readily represent probabilities that a human becomes infected if bitten by an infectious mosquito, even if those probabilities depend on that host's prior infection history and waning immunity to reinfection. We can represent the decay of immunity within an individual to one "strain," even if that decay depends on the interval since that individual cleared another "strain," and so forth. The decay of immunity within an individual may have a very complex relation to the decay of the immune component of a population, and in fact the time scales involved at the individual and the population levels may differ by orders of magnitude.

Any individual character that changes based on calculations with respect to that individual can be represented by aggregates in a priori population-level models only if the aggregates either

incorporate individual histories or make broad assumptions about their distribution. Therefore, if individuals in a population differ with respect to some character, and the behavior of the population critically depends upon these differences, models such as those developed here may provide a better approach. Discrete-event simulation models are thus worthy partners of differential-equation models of malaria epidemiology in the many situations in which their representations can more closely approximate the underlying biological processes and mechanisms.

Discrete-event models are often criticized because they provide no closed-form analytic solutions. Obviously we believe that in the appropriate context some advantages of discrete-event models are compensatory. One such advantage is the intrinsic occurrence of heterogeneous mixing: interactions may occur between rare variants rather than only between aggregates or averages within a given distribution. Recent mathematical and empirical studies that examine variously-defined subdivisions of parasite, host and vector populations suggest that the diversity and abundance of phenotypes and genotypes involved in malaria may have profound implications for vaccine, drug and other intervention strategies.

The models presented here have many other shortcomings. We consistently treat human populations as static, and we largely ignore mosquito population dynamics. There is not a certainty but some probability that a mosquito biting an infectious human becomes infected, and this probability should be represented by something other than an ad hoc tuning parameter. Our operational view of immunity incorporates the existence of partial immunities to reinfection, but it fails to encompass the possibility that immune responses may act to limit parasite densities upon reinfection. So little is known about human malaria immunology that the shortest immune half-life we consider may be too long, or the longest too short. We have not yet examined the many possibilities of immunologic ally cross-reactive or potentially recombinant strains of parasite.

Nonetheless, our results are strikingly congruent with those of the differential-equation models developed and tested during the past century, and the few seemingly anomalous results at this level of analysis concern factors our predecessors were unable to address in similar terms. Certainly the unsuspected importance of the duration of host infectivity merits some attention with respect to planned vaccine-based interventions, at least in regions with intense perennial transmission. Our preliminary results with a seasonal-transmission extension of the model suggest that vector mortality is in fact the dominant influence on prevalence in humans in short seasons, but that the influence of host window grows to near-parity with longer, ultimately perennial transmission seasons.

Notes The influence of host immunity appears to rise more rapidly with season length than that of either vector mortality or host window, but still falls short of parity in the perennial-transmission case.

13.2 Continuous Simulation Languages

Continuous Simulation refers to a computer model of a physical system that incessantly tracks system response over time according to a set of equations typically involving differential equations.

Continuous system simulation languages are very high level programming languages which assist modelling and simulation of systems characterized by ordinary and partial differential equations. Design principles and implementation techniques for continuous system simulation

Notes

languages. Following a brief introduction to very high level languages, design principles for continuous system simulation languages are presented. These principles are illustrated by examples from the Continuous System Modelling Program (CSMP) and the Partial Differential Equation Language (PDEL). A typical program in each language is included. Batch and interactive implementation techniques for continuous system simulation languages are discussed. The classical batch implementation technique is to provide a preprocessor which translates the simulation language into an algorithmic language such as FORTRAN or PL/1. The PL/1 preprocessor is described as a useful language for the implementation of very high level language translators. The final section of the paper presents an interactive implementation technique which interfaces a batch program processor to interactive graphics display and updating routines. In this manner, efficient simulation code is interfaced to flexible interaction routines.

Did u know? **What is batch processor?**

The batch processor is preserved intact, thus requiring only one implementation of the language for both batch and interactive applications.

Currently available Continuous System Simulation Languages (CSSLs) are sensibly effective in providing a more user-oriented interface to the computer. However, the fact that CSSLs represent a tool and not a panacea has not always been kept in perspective. Operational problems have hindered the use and consequent improvement of the languages. Designers and implementers must be willing to stay in the loop and smooth out the operational interface for the users. The future for CSSL is very bright. The increasing use and importance of remote, interactive terminals is based on the same user-problem approach that is the foundation of CSSL. It is in this area that CSSL will probably be most effective and will experience broad acceptance by new users.

It is distinguished as one of the first uses ever put to computers, dating back to the Eniac in 1946. Continuous simulation allows prediction of rocket trajectories, hydrogen bomb dynamics (N.B. this is the first use ever put to the Eniac), electric circuit simulation, and robotics. Established in 1952, The Society for Modeling & Simulation (SCS) is a nonprofit, volunteer-driven corporation dedicated to advancing the use of modeling & simulation to solve real-world problems. Their first publication strongly suggested that the Navy was wasting a lot of money through the inconclusive flight-testing of missiles, but that the *Simulation Council*'s analog computer could provide better information through the simulation of flights. Since that time continuous simulation has been proven invaluable in military and private endeavors with complex systems. No Apollo moon shot would have been possible without it.

Modern Applications

Continuous simulation is establish inside Wii stations, commercial flight simulators, jet plane auto pilots, and advanced engineering design tools. Indeed, much of modern technology that we enjoy today (along with technology that can destroy the planet) would not be possible without continuous simulation.

Mathematical Theory

In continuous simulation, the continuous time response of a physical system is modeled with ODEs.Newton's 2nd law, F = ma, is a good instance of a single ODE continuous system. Numerical integration methods such as Runge Kutta, or Bulirsch-Stoer are used to solve the system of ODEs. By coupling the ODE solver with other numerical operators and methods a continuous simulator can be used to model many different physical phenomena such as flight dynamics, robotics, automotive suspensions, hydraulics, electric power, electric motors, human respiration,

Notes

polar ice cap melting, steam power plants etc. There is virtually no limit to the kinds of physical phenomena that can be modeled by a system of ODE's. Some systems though can not have all derivative terms specified explicitly from known inputs and other ODE outputs. Those derivative terms are defined implicitly by other system constraints such as Kirchoff's law that the flow of charge into a junction must equal the flow out. To solve these implicit ODE systems a converging iterative scheme such as Newton-Raphson must be employed.

Advanced Continuous Simulation Language

The **Advanced Continuous Simulation Language**, or **ACSL** (pronounced "axle"), is a computer language designed for modelling and assessing the performance of continuous systems described by time-dependent, nonlinear differential equations. It is a dialect of the Continuous System Simulation Language (CSSL), originally designed by the Simulations Council Inc (SCI) in 1967 in an attempt to unify the continuous simulations field.

Task Analyze the difference between Continuous Simulation Language and Advanced Simulation Language

Language Highlights

ACSL is an equation-oriented language consisting of a set of arithmetic operators, standard functions, a set of special ACSL statements, and a MACRO capability which allows extension of the special ACSL statements.

ACSL is intended to provide a simple method of representing mathematical models on a digital computer. Working from an equation description of the problem or a block diagram, the user writes ACSL statements to describe the system under investigation.

An important feature of ACSL is its sorting of the continuous model equations, in contrast to general purpose programming languages such as Fortran where program execution depends critically on statement order.

Did u know? **Applications of ACSL.**

Applications of ACSL in new areas are being developed constantly. Typical areas in which ACSL is currently applied include control system design, aerospace simulation, chemical process dynamics, power plant dynamics, plant and animal growth, toxicology models, vehicle handling, microprocessor controllers, and robotics.

Caselet

T20 Scenarios in Finance

In today's turbulent business environment, the year-end finance figures are scorecards of the complete match, while breaking into events and analysing the same is like taking 20-20 snapshots.

As with every financial year closure, this is the time that reveals to managements an integrated view of financial outcomes resulting from a series of business decisions made

Contd...

Notes

during the year. But what would be of specific interest this year is for organisations to learn and understand the financial impact caused due to shortened time lines for business decision-making, says Uma Balakrishnan, CEO, Axcend Automation & Software Solutions Pvt Ltd, Bangalore (www.axcend.com).

For example, let's look at typical business decision-making cycles in the discrete manufacturing sector, she notes, in an email exchange with Business Line. "All major decisions - be they in enhancing existing capacity or building forward or backward integration business or entering complementary businesses - are made for the medium and long term, while operational decisions are made during the year."

The uncertainty in the economy and the sudden shift in market dynamics would have necessitated many interim course corrections in the business plan, with some assumptions being no longer valid, reasons Uma. "So, there is one aspect of shortened decision cycle triggered through this. The other is that all operating base lines on cost of raw material, operating cost, supplier business healthiness and availability internally, and demand forecast baseline from external world, also had huge swings…"

With the standard historical data based costing losing its relevance temporarily, what was needed was on-the-feet thinking from all facets of business and taking immediate actions, she explains.

Excerpts from the interview:

What are the lessons learnt?

The year closure provides a good trend of the past few quarters' financials and the same when mapped with time-based event mapping of key decisions taken would provide some insights, especially on business execution in turbulent times and its impact.

To cite an example, on the decision of using available credit lines in working capital in a particular week towards paying a local supplier who was critical to ensure completion of key customer delivery (as stated and highlighted by production then!) vis-à-vis having to import a long-lead item, which is still not in the radar as emergency, the finance department has to make a choice.

Given that there is a constraint on quantum of working capital availability, further a business strain with delayed collections cycles, the finance would take a decision based on business impact, rather than extinguishing escalations as they come from shop-floor.

The consequence of deferring local supplier payment might have impacted the revenue for the month, increased WIP to a defined extent, but based on the collection cycle expected for that customer shipment the cash flow could have some impact.

In the case of the imported supplier payment, there might be a firm defined collection with LC, export customer insisting on penalty for delayed delivery and WIP or raw-material holding.

This is like a T20 match scenario for finance, except that you don't have a dynamic scorecard as in the match with other team's net run rate inside that group - how you bat, who to bowl to win and what speed is relative to the real-time context of the situation with long-term impact of being in the game. The 5-day cricket strategy does not need ball-to-ball information, but innings to innings.

Today's business in a turbulent environment is akin to playing 5-day match, as a sum of many 20-20 sessions by the same team. So, the year-end finance figures are scorecards of the complete match, while breaking into events and analysing the same is like taking 20-20 snapshots. It helps to learn lessons and understand how to bat the next innings, that is, the next year, better.

Contd...

Notes

Looking back, what tools or information due you think would have helped the CFOs make a better and more flexible investment decision?

Every rupee investment either as capital expenditure or as working capital in today's world can be made with a decision taken in an informed environment. With the availability of tools and commonsensical and logical adaptation of the same - especially hybrid mix-and-match of different functional information - this can be done.

While businesses have traditionally looked at technology decisions by the CTO and CIO, the usage of applications has also been standalone. But just like how a senior management constituting of CEO, CIO, CTO, COO, and CFO have strategy discussions and align thoughts for business, it is also necessary to provide an information framework that allows all the key executives to delve into operational data in real time to take informed decisions in their functions.

Let's look at current assets. Availability of all current assets in real time provides a CFO a much more informed decision-making platform. While businesses are used to reconciling physical cash in hand and in bank, and ensuring that they are synced in real time with the IT records of cash component in current assets in the accounting systems, rarely do we see that in other components such as work in progress, inventory and finished goods.

There are valid records in the ERP, based on transactions, but there are no real-time records based on physical validations. The constant argument in case being the volume of physical quantity is so much, and that it is not practical. But there are tools which can be adapted akin to a bank statement or ATM records, to get records of physical transactions, to track from the shop floor manufacturing line and independently record the levels of inventory consumed, produced goods and asset usage in real time.

If the shop machine is intelligent to make the goods, with incremental intelligence and investments, it can certainly be made to share that intelligence on throughput, consumption availability to a real-time view of the finance team. Real-time finance dashboard with key operational data on hourly, shift, daily, weekly, and for that matter any defined time line resolution is feasible. The CFO should exercise the power to have it.

On the capital investments side, today in the discrete manufacturing sector, there are tools with use of which companies can make informed decisions. With a virtual modelling and discrete event simulation tool, you can build the entire facility, model the operations, and take investment decisions.

For example, take the case of a storage facility of x sq.mt size with steel housing. What optimum size that provides the right investment for the scale is an investment decision. Floor space usage in factory, size of storage, mix ratio of mechanised fork lift vs manual labourers are also investment decisions resulting from design stage decisions. Today there are design tools that provide what-if investment analysis for each option or choice made, and this can provide a CFO a clear insight into the impact of the choice of investment.

Going forward, are there IT tools which can help the CFO maintain a real-time dashboard as help in knowing the changes happening in the cost structure and take appropriate decisions before it is too late?

Typically most manufacturers have standard cost calculations based on base line cost of individual components of products, average operating costs validated year on year and then new product pricing is done based on product mix planned in the capacity for production.

Contd...

Notes

The challenge clearly has been to analyse the actual cost of production and know what contributed to the bottom line of the business. Was it the right product mix, was it the right cost of manufacturing, and was it the right selling price.

Many times, CFO grapples with why the business result was not as planned, despite revenue targets being met. Understanding each cost component in real time as it happens, gives senior management a powerful tool to analyse and know the contribution factor.

While most costs might be done on average weighted cost basis (say, for energy), it is quite possible that a particular size/quantity and category of product line draws more energy; consequently it is being subsidised, while another product line has lost its market share because of non-competitive selling price.

A real time costing analysis would assist businesses to know real time contributing factor for product mix, category, etc., and further also understand which customer is being profitable in terms of contribution.

Source: http://www.thehindubusinessline.com/todays-paper/tp-opinion/article989070.ece

13.3 Summary

- In models for discrete event dynamic systems (i.e., DEDS models) state changes occur at particular points in time whose values are not known a priori.
- As a direct consequence, (simulated) time advances in discrete 'jumps' that have unequal length.
- Continuous Simulation refers to a computer model of a physical system that incessantly tracks system response over time according to a set of equations typically involving differential equations.

13.4 Keywords

C/D: Continuous/Discrete (C/D)

CTDS: Continuous Time Dynamic Systems

DEDS:Discrete Event Dynamic Systems

DSS: Discrete-system Simulator

VIM: Visual Interactive Modelling

VIS: Visual Interactive Simulation

13.5 Self Assessment

1. One of the main difficulties in the definition of CAD tools for Continuous/Discrete (C/D) systems is due to the ……. of concepts manipulated by the discrete and the continuous components.
2. The interfaces are generated by …… elements from the CODIS library.
3. The communication layer is in charge of sending/receiving data between …….
4. There are a large number of ……. areas for Discrete Event Simulation.
5. …… approaches are particularly well equipped to help users diagnose issues in complex environments.

Notes

6. A simulation model allows the user to understand and test a ……. improvement idea in the context of the overall system.
7. Visual Interactive Modelling (VIM) takes this concept one stage further by allowing the model to be created ……...
8. The ATM technology is considered as the state of the art ……. technology that is expected to play an important role in the future networks.
9. Human malaria is characterized by hierarchies of …….. processes, occurring on diverse time scales within and between heterogeneous populations.
10. The parameters h, b and d represent daily rates of vector biting, ……. and mortality, respectively; b/d gives the ratio of vectors to hosts (N_V/N_H).
11. Continuous system simulation languages are very high level programming languages which assist …….. and simulation of systems characterized by ordinary and partial differential equations.
12. …… is intended to provide a simple method of representing mathematical models on a digital computer.
13. The continuous time response of a physical system is modeled with …….. , F = ma, is a good instance of a single ODE continuous system.
14. The fact that CSSLs represent a tool and not a …….. has not always been kept in perspective.
15. Discrete-event models are often criticized because they provide no …….. analytic solutions.

13.6 Review Questions

1. Do you think in the case of validation tools, several execution semantics have to be taken in consideration in order to perform global simulation?
2. Explain, how human malaria is characterized by hierarchies of dynamic processes?
3. What are the major components of a Discrete-event Simulation.
4. Describe the five key features found in the software simulation model.
5. Make an analysis and write a note on Discrete-Event Simulation (DES).Explain it with an example.
6. Can a designers and implementers must be willing to stay in the loop and smooth out the operational interface for the users? Why or why not?
7. Discrete Event Simulation (DES) concerns the modelling of a system. Explain why or why not?
8. Do you think the problems with the random number distributions used in discrete-event simulation are the steady-state distributions?
9. Well-known examples of Simulation are Flight Simulators. Give Reasons.
10. Object Oriented techniques are the way in which the data and the program code are stored and manipulated. Elaborate.

Notes

Answers: Self Assessment

1. heterogeneity
2. composing
3. CM and DM
4. potential
5. Simulation
6. performance
7. interactively
8. network
9. dynamic
10. natality
11. modeling
12. ACSL
13. ODEs.Newton's 2nd law
14. panacea
15. closed-form

13.7 Further Readings

Books

Balci, O.,(1994), Validation, verification, and testing techniques throughout the life cycle of a simulation study, Annals of Operations Research, 53: 121–173.

Balci, O., (2001), A methodology for certification of modeling and simulation applications, ACM Transactions on Modeling and Computer Simulation, 11: 352–377.

Birta, L.G. and Ozmizrak, N.F., (1996), A knowledge-based approach for the validation of simulation models: The foundation, ACM Transactions on Modeling and Computer Simulation, 6: 67–98.

Boehm, B.W., (1979), Software engineering: R&D trends and defense needs, in: P. Wegner (Ed.), Research Directions in Software Technology, MIT Press, and Cambridge, MA.

Cellier, F.E., (1986), Combined discrete/continuous system simulation – application, techniques and tools, in Proceedings of the 1986 Winter Simulation Conference.

Department of Defense (DoD) Recommended Practices Guide (RPG) for Modeling and Simulation VV&A, Millennium Edition (available at http://vva.dmso.mil).

General Accounting Office, (1976), Report to the Congress: Ways to improve management of federally funded computerized models, report LCD-75-111, U.S. General Accounting Office, Washington, DC.

General Accounting Office, (1979), Guidelines for model evaluation, report PAD-79-17, U.S. General Accounting Office, Washington, DC.

General Accounting Office, (1987), DOD simulations: Improved assessment procedures would increase the credibility of results, report GAO/PEMD-88-3, U.S. General Accounting Office, and Washington, DC.

Ören, T.I., (1981), Concepts and criteria to access acceptability of simulation studies, Communications of the ACM, 24: 180–189.

Online links

http://en.wikipedia.org/wiki/Advanced_Continuous_Simulation_Language

http://en.wikipedia.org/wiki/Simulation_language

Unit 14: Simulation Languages (II)

Notes

CONTENTS

Objectives

Introduction

14.1 Block-structured Continuous Simulation Languages

14.1.1 GPSS (General Purpose Simulation Languages)

14.1.2 General Purpose Simulation Packages

14.2 Expression based Languages

14.3 Discrete System Simulation Languages

14.4 Summary

14.5 Keywords

14.6 Self Assessment

14.7 Review Questions

14.8 Further Readings

Objectives

After studying this unit, you will be able to:

- Explain Block structured continuous simulation languages
- Discuss expression based languages
- Analyze discrete system simulation languages

Introduction

GSL (a combined continuous and discrete simulation language) is a FORTRAN-oriented language which merges the activity and process concepts of a discrete simula tion language with continuous simulation concepts, thereby permitting the simulation of systems which call for combining continuous and discrete simula tion techniques.

14.1 Block-structured Continuous Simulation Languages

Block oriented simulation languages are based on the method of analogue computers. The system must be expressed as a block diagram that defines the interconnection of functional units and their quantitative parameters. "Programming" means entering the interconnection of the blocks and their description. Then the user adds statements and/or directives that control the simulation. If the system is described as a set of equations, they must be converted to a block diagram. This conversion is a simple straightforward process. The typical blocks available in most continuous block oriented languages are integrators, limiters, delays, multipliers, hysteresis, constant values, adders, holders, gain (coefficient) and other.

Notes

Users of simulation languages have traditionally had to choose between one of two types of language. On the one hand, they could choose a block type language such as GPSS. Languages of this type have the advantage that the learning and programming of the language is both quick and relatively easy. Also, since the number of de-bugging runs is usually small, turn around time is likely to be quite rapid. However, the block structuring of the language may also create some difficulties. The fixed nature of the blocks, both as to the types of blocks available in the language and the manner in which the blocks simulate a particular activity, may render the language unsuitable for the simulation of certain types of systems. In addition, the need to pre-allocate much of the available memory space of the computer may make the use of a block structured language impossible on a small computer. The alternative to a block structured language is an algebraic language, such as SIMSCRIPT or FORTRAN. It would be desirable to have a language which combines the advantages of both types of language without some of the disadvantages. QUIKSIM represents an attempt to produce.

14.1.1 GPSS (General Purpose Simulation Languages)

General Purpose Simulation System (GPSS) (originally **Gordon's Programmable Simulation System** after creator Geoffrey Gordon. The name was changed when it was decided to release it as a product) is a discrete time simulation language, where a simulation clock advances in discrete steps. A system is modelled as transactions enter the system and are passed from one service (represented by blocs) to another. This is particularly well suited for problems such as a factory. It was popular in the late 1960s and early 1970s but is little used today. GPSS is less flexible than simulation languages such as Simula and SIMSCRIPT II.5.

This topic explains designing general purpose, parallel simulation languages. We discuss how parallel simulation languages differ from general purpose programming languages. Our thesis is that the issues of distribution, performance, unusual implementation mechanism, and the desire for determinism are the dominant considerations in designing a simulation language today. We then discuss the separate roles that special and general purpose simulation languages play. Next we use the two languages, Sim++ and CPS, to illustrate these issues. Then we discuss eight design considerations: process versus event oriented-view, basic program structure, I/O, making implementation cost explicit to the programmer, providing dynamic facilities, memory management, the semantics of false messages in time warp, and program development methodology considerations. A number of conclusions are drawn from our experiences in language design.

Modern simulation languages such as SIMSCRIPT II and SIMULA 67 are very powerful general purpose languages which contain facilities to handle lists and to schedule events in simulated system time (imperative sequencing statements). These languages do not include some of the useful but more specialized features of previous languages (GPSS, CSL, SOL) especially interrogative sequencing statements such as "SEIZE (facility)" or "WAIT UNTIL (Boolean expression)"; however, the definition capability of the new languages is powerful enough to permit their extension to include the interrogative features. The addition of some features of GPSS to SIMULA 67 was presented at a previous SIMULATION CONFERENCE. The present paper extends that work by describing an efficient algorithm which adds the "WAIT UNTIL" procedure to SIMULA.

Present day approaches to discrete event simulation can be generally classified as using either special purpose data-driven simula tion systems or general purpose simulation languages. Although the latter embrace a far wider range of applications and offer much greater flexibility in modeling for a particu lar application, their user base is restricted by the high level of expertise required. An Expert System interface to general purpose simulation languages, ISI (the Intelligent Simulation Interface), is pre sented here. Models are constructed and simulation runs

Notes

are performed in an integrated environment comprising interactive menu-driven graphics, a knowledge base system and a general purpose simulation language.

Notes The features of ISI include hierarchical model construc tion, interactive experiment specification, automatic code generation, run-time animation and interactive graphical post- processing of results.

Advantages of Special Purpose Languages

1. Simulation languages offer most (if not all) of the features needed in programming a simulation model resulting in a decrease in programming time which can often be significant.
   (a) Generating random numbers, that is U (0,1) random variables.
   (b) Generating random variables from a specified distribution.
   (c) Advancing simulated time.
   (d) Determining the next event from the event list
   (e) Adding records to, or deleting records from, a list.
   (f) Collecting and analyzing data.
   (g) Reporting the results.
   (h) Detecting error conditions.
2. Simulation models are usually easier to change when written in a simulation language.
3. They provide better error detection because many potential types of errors have been recognized and are checked for automatically. Since fewer lines of code have to be written, the chance of making an error will probably be smaller.

Caution Conversely, errors in a new version of a simulation language itself may be difficult for a user to find.

Advantages of General Purpose Languages

1. Most modelers already know a general purpose language, but this is often not the case with a simulation language.
2. General purpose languages are obtainable on virtual1y every computer, but a particular simulation language may not be accessible on the computer that the analyst wants to use.
3. An competently written general purpose program may require less execution time than the corresponding program written in a simulation language. This is because a simulation language is designed to model a wide variety of systems with one set of building blocks, whereas general purpose program can be tailored to the particular application.
4. General purpose languages allow greater programming flexibility than certain simulation languages.

Notes

Example: Complicated numerical calculations are not easy in GPSS.

Simulation Languages

Two options for coding computer simulations:

1. General purpose languages (C, C++, Fortran, Ada, etc.)
2. Simulation languages (SIMSCRIPT, GPSS, COMNET, OPNET, SimPack, etc.)

Advantages of using Simulation Languages

1. Simulation languages routinely provide most (if not all) of the features needed in programming a simulation model.
2. Simulation languages provide a natural framework for simulation modeling.
3. Simulations are generally easier to change when with in a simulation language.
4. Most simulation languages provide dynamic storage allocation during execution.
5. Most simulation languages provide better error detection.
6. Provide all of the statistical tools you need.

Advantages for General Purpose Languages

1. Most programmers already know a general purpose language.
2. General purpose languages are available on almost every computer.
3. An efficiently written general purpose language may require less execution time than the corresponding program written in a simulation language.
4. General purpose languages allow greater programming flexibility.

14.1.2 General Purpose Simulation Packages

AweSim

AweSim is a completely new general-purpose simulation system, released in June of 1996. AweSim takes advantage of the latest in Windows technology to integrate programs and provide componentware. AweSim includes the Visual SLAM simulation language to build network, discrete event, and continuous models. Network models require no programming yet allow user-coded inserts in Visual Basic or C. Discrete event and continuous models can be created using the object-oriented technology of Visual Basic, C or Visual C++ and can be combined with network models. This tutorial will demonstrate the process of using AweSim's componentware and provide examples of user interfaces that allow integration with other Windows applications both directly and through the AweSim database.

GPSS/SLX

GPSS/H is a very powerful simulation tool. It has a illustrious 20-year track record, and it has been used for a vast array of applications. We anticipate that it will be used for many more. While SLX is not a replacement for GPSS/H, it contains a lot of its spirit. For example, SLX's "zero tolerance" policy for errors was derived from a similar policy in GPSS/H. In SLX we tried

Notes

to build on the strengths of GPSS/H, eliminate cumbersome or rarely used features, combine related features into more general features, and package everything in an improved modeling environment. If you know and love GPSS/H, with a bit of gear-shifting, you can transfer your knowledge to SLX. We have implemented a subset of GPSS/H in SLX, so the familiar SEIZE, RELEASE, ENTER, LEAVE, QUEUE, DEPART, ADVANCE, and TERMINATE (among others) are available in SLX. GPSS/H's GENERATE has been recast as SLX's "arrivals" statement.

SIMPLE++

SimpSim is a simulator/assembler for the machine language as proposed in "Computer Science: An Overview" by J. Glenn Brookshear.

It features the following:

1. Windows 95/98/NT/XP/Vista, no installation required
2. Simulator with run, step and break functions
3. Built-in assembler editor with syntax highlighting
4. Built-in assembler
5. Loading and saving of files (assembler and machine coded (byte code))
6. Built-in assembler examples
7. Context sensitive helppages about virtually all items
8. Extra www infopage (this can be set by the teacher of a particular course to point to a page, used for that course, with the latest information, hints and so on)
9. Disassembler/trace window for stepping
10. Output window to output text from a machine language program (see examples)
11. No password or registration needed, so everyone can use it

SIMUL8

SIMUL8 simulation software is a creation of the SIMUL8 Corporation used for simulating systems that involve processing of discrete entities at discrete times. This program is a tool for planning, design, optimization and reengineering of real production, manufacturing, logistic or service provision systems. SIMUL8 allows its user to create a computer model, which takes into account real life constraints, capacities, failure rates, shift patterns, and other factors affecting the total performance and efficiency of production. Through this model it is possible to test real scenarios in a virtual environment, for example simulate planned function and load of the system, change parameters affecting system performance, carry out extreme-load tests, verify by experiments the proposed solutions and select the optimal solution. A common feature of problems solved in SIMUL8 is that they are concerned with cost, time and inventory.

SIMUL8 uses dynamic discrete simulation, which makes it possible to provide unambiguous and concrete results and proofs - information on how the designed or optimized production system will actually function. The outputs of SIMUL8 simulation are "hard data", values and statistics of performance parameters and metrics of the production system.

Task Compare SIMPLE++ and SIMUL8

Notes

Model Building

Construction of SIMUL8 models is classically not based on programming or statistical data, but rather on drawing organization schemes on the screen. However, SIMUL8 implements a two-way interface with Visual Basic, which leaves space for creation of advanced model features, which cannot be modeled using only the graphical interface. SIMUL8 also provides its own simulation language based on Visual Basic called Visual Logic, which allows the user to implement detailed logic of the simulation. The design of SIMUL8 also facilitates communication with other software packages such as Microsoft Access, Excel and Visio. The support of XML and OLE automation allows working with external sources of data and exporting internal data to other systems. SIMUL8 also supports communication with databases using SQL.

Basic Components of SIMUL8 Environment

A SIMUL8 simulation gyrates around processing **work items**. They enter the system via **work entry points**, pass through **work centers**, may temporarily reside in **storage areas** and leave via **work exit points**. In addition to this mechanism, work centers may need specific **resources** to process **work items**. A simulation consists of a number of these objects and of the routes between them, modeled as a directed graph.

Component	Description
Work item (element, entity)	models physical or logical objects moving through the system. Entities enter the system, induce different sorts of activities, use different kinds of resources and at the end leave the system. A customer, product or document can be a SIMUL8 model entity.
Work entry point (entrance)	objects that represent the entry of entities into the system (for example an arrival of customer or formation of a product
Work center (activity, action)	objects that model activities which the entities go through. Resources are typically used during execution of an activity
Storage bin (queue, stack)	objects that model cumulation of entities. The stack usually precedes activities for which the stacked entities wait because of lack of resources
Work exit point (exit)	a place through which the entities leave the modelled system (completion of an order, leaving of a customer)
Resource (source)	objects that are used for modelling capacity restraints of workers, material or means of production used in activities
Route	objects that connect all the other simulation objects. They represent sequences of activities and thus the movement of entities in the system.

Typical Inputs and Outputs

These are the most regular parameters of a SIMUL8 model, which are set by the user to influence the conditions of simulated environment:

1. Cycle times
2. Production rate
3. Capacity of production equipment
4. Arrival/order rates
5. Production rates of production equipment
6. Statistics of production equipment failures

Notes

The outputs of the simulation offer information about:

1. Utilization of production equipment
2. Identification of bottlenecks
3. Production system performance
4. Inventory levels

Areas of Use

SIMUL8 can be used to model any procedure where there is a flow of work, however the main areas of use are in manufacturing, health care, contact centers and supply chain.

SIMUL8 can be used to simulate different kinds of:

1. Manufacturing systems such as assembly line models or models of material flow during production.
2. Logic systems such as model of manipulation with material between storage, manufacturing and expedition, models of storage expeditionary systems or models of logistic services for distribution centers.
3. Administrative workflows such as model of received orders.
4. Client service systems or service delivery such as model of customer attendance at banks, models of call center customer attendance or models of customer attendance at hypermarket cash desks.

Taylor

Taylor II is a menu driven simulation package mainly used in manufacturing and logistics. It is developed for the analysis and quantitative evaluation of complex processes especially of those with a dynamic character. A lot of applications in different industries show that there is an increasing need for simulation tools. The paper demonstrates the process of building, analyzing and presenting models of real world systems with Taylor II.

Micro Saint

Built off the very successful Micro Saint engine, but totally redesigned to be faster, modular and more powerful! Micro Saint Sharp is a general purpose, discrete-event simulation software tool. Micro Saint Sharp's intuitive graphical user interface and flow chart approach to modeling make it a tool that can be used by generalists as well as simulation experts. Micro Saint Sharp has proven to be an invaluable asset in both small businesses and Fortune 500 companies and in many areas including the military, human factors, health care, manufacturing, and the service industry.

Micro Saint Sharp's power, suppleness and tools for optimization make it the simulation tool of choice for any organization. With a computer model of your process built in Micro Saint Sharp, you can begin to get the answers to your "what if" questions. What if I change the way humans work with the system? What if I change my resource mix? What if I rearrange the process? Find the answers with Micro Saint Sharp quickly and completely for systems of all sizes, shapes, and complexities.

Micro Saint Sharp Version 3.5 - now with floating licenses and a new Experiment capability! There are still 3 ways to view your models: network diagram view, 2D animation and 3D animation. Download a free demonstration version or contact us at

Notes

Features of Micro Saint Sharp

1. Microsoft Visio import/export
2. 2D and 3D Animation
3. More Power
4. Flexibility
5. Increased Execution Speed
6. Better Visualization
7. Customization
8. Interoperability
9. Optimization

New in Micro Saint Sharp Version 3.5

These new enhancements can be found in this release:

1. We have added many new ***Sample Models*** for version 3.5 that show how to use many of the features of Micro Saint Sharp. Each model has a description and detailed comments.
2. A new ***Start Page*** now shows when Micro Saint Sharp launches that gives you a quick way to open your recent files, open sample models, and view news and announcements.
3. New ***Experimenter*** ability allows you specify variables and ranges for a variety of inputs and run all of the possible permutations. You can now easily define multiple experimental runs and specify multiple iterations for each condition.
4. The ***Model Runtime Export*** has been improved to allow you to export models that use Animator, Animator3D, Object Designer, and/or the Communications capabilities. You can also now name your runtime application and define runtime execution settings (number of times to execute, random number seed, etc.)
5. You can now ***Quickly Add Variables*** to the variable catalog directly from text in an expression field
6. Quickly ***Insert Variables*** directly into your model expression code using the right-click menu.
7. New ***Export to HTML*** option permits you to view using an HTML tree view display with clickable task diagrams and display. You can send the HTML files to others that need to review the model structure but do not own Micro Saint Sharp.
8. New ***Histogram Viewer*** allows you to quickly generate a graphical chart of your snapshot data in histogram form.
9. A ***Pathfinding Capability*** has been added to Animator and Animator3D that allows you to define waypoints that are on a direct path with each other then use the new GetRoute and GetRouteDistance functions to have Micro Saint Sharp find the shortest route from one point to another.
10. Enhanced ***Notes*** feature now allows WYSIWYG text editing functionality.
11. You can now enable/disable ***Model Variance*** for any particular execution.
12. The ***Grid Editor*** has been enhanced to support editing reference tasks.
13. Use the new ***Console*** communication protocol to interact with command line applications.

Notes

14. ***Updated Floating License*** to better support users with 2 or more copies of Micro Saint Sharp.

Other useful recent enhancements include:

1. Import your existing Microsoft Visio business and technical drawings and make them into process simulation models
2. Create reusable modeling components with Referential tasks and paths which will reduce your overall model development time
3. Better visualize and add scenarios with the new Scenario Viewer
4. View new reports including tasks and scenario time-line reports
5. Find things quickly with the new "Go To" window
6. Turn off entity merging which greatly improves model run performance
7. Save time creating snapshots with a new automatic variable data collection option
8. Store output files where you want them for easier comparative model run analysis
9. Improved performance
10. Communication plug-in that allows users to export model data to spreadsheets easily

Here are some capabilities of Animator3D:

1. Create a virtual environment that represents your desired process with 3D moving images
2. Explore your virtual world with enhanced viewing capabilities that allow you to rotate objects and tour models from any angle
3. Use our built-in 3D objects library with models from multiple industries to get you started creating realistic, detailed scenes
4. Navigate through your virtual world by zooming in and out and rotating with ease
5. New 3D objects for military applications and landscapes
6. Integrated with your simulation model
7. Uses Microsoft's latest DirectX technology

System Requirements

1. 90-Megahertz Intel Pentium-class processor
2. 64 MB RAM (128 MB recommended)
3. 150 MB of free hard disk space
4. CD-ROM drive
5. Microsoft Windows Server 2003, Windows Server 2008, Windows XP, Windows Vista, Windows 7 (Operating systems must support the Microsoft .NET Framework 3.5)

Micro Saint Sharp Sale Package

Included with every sale of Micro Saint Sharp is:

1. Micro Saint Sharp Version 3.5 software
2. User Guide
3. One year of unlimited technical support
4. One year of software updates

Notes

14.2 Expression based Languages

Expression oriented continuous languages are based on writing expressions (equations) that represent the mathematical model. So the system simulated must be expressed by a set of equations. Then the user adds statements and/or directives that control the simulation. Some languages enable both block and expression based ways of system definition. Simulation control means selection of: the integration method (because some languages offer more), the integration step, the variables (outputs of blocks) that should be observed, the intervals for collecting data for printing and/or plotting, scaling of outputs (that may be also done automatically).

Notes Duration of the simulation runs, number of repetitions and the way certain values are changed in them.

14.3 Discrete System Simulation Languages

AutoMod

AutoMod simulation software meets the needs of equally the casual, first-time user and the full-time simulation model builder. You can easily and accurately simulate systems of any size or level of detail, from manual operations to fully automated facilities. **AutoMod is a** world-leading software for simulation of production and logistics systems. The software is designed for detailed analysis of operations and flows. Although mainly used in manufacturing and material handling systems analysis, AutoMod's flexible architecture allows it to be used in a wide range of application areas, from airports to semiconductor industry.

Application/Benefit

User-friendly, detailed 3D-animation, capable simulation and numerous analysis options are the keywords of AutoMod.

AutoMod offers a variety of probable applications due to its very efficient simulation core. These extend from modelling manufacturing processes through warehouse simulation and supply chain simulations to online link-up/emulation.

With AutoMod the emphasis is not only on processing power but also on the ever more important visualisation right through to virtual reality. Thus communication between management, production and staff can be improved significantly. Here modelling the simulation models is carried out interactively with the aid of graphic support. Complex controls and processes can be mapped with the aid of a simple simulation programming language.

Structure/Modules

In order to be able to adapt the software to the respective intended purpose, it has a modular structure. The basic system includes Process System, Simulator, Dtrace, ACE, IGES and SDX import option. The following modules are available as extension modules:

1. AS/ RS (warehouse technology)
2. Conveyor (conveyor technology)
3. Kinematics (robots)

Notes

4. Pathmover (floor conveyors)
5. Bridge Crane (bridge cranes)
6. Power and Free (trolley conveyors)
7. Tanks and Pipes (liquids etc.)
8. AutoTruck (transport vehicles)

Extension Programs

1. AutoView (animated films)
2. AutoStat (experiments, statistics)
3. Model Communications (data exchange)
4. ToolSim (quick simulator)

eM-Plant

To build a realistic simulation model is all very well - to add real value you must identify the major difficulties and generate better alternatives. Tecnomatix Technologies, developers of eM-Plant, the object oriented simulation tool for discrete event simulation, planning and optimization of production and logistics, are the world leaders of the e-Manufacturing market. eM-Plant is used across many industries from manufacturers like BMW and Daimler-Chrysler through shipyards to international finance. Experience from all these areas inspired the development of analysis and optimization tools as an integral part of the simulation objects that continuously monitor and evaluate the operation of each simulation object and its interactions at a local and global level. Thus, for example, bottlenecks in a material flow are automatically detected and Sankey diagrams generated. This object based approach fits closely with the object-oriented nature of eM-Plant that allows unprecedented accuracy and re-usability in simulation modeling. These analysis tools work with global evaluation wizards which, for example, make it simple to create specific Gantt-charts etc., and using Genetic Algorithms modules the simulation system can even propose better layouts or operating strategies.

Arena

Arena is a discrete event simulation software simulation and automation software developed by Systems Modeling and acquired by Rockwell Automation in 2000. It uses the SIMAN processor and simulation language. As of 2010, it is in version 13.0. It has been suggested that Arena may join other Rockwell software packages under the "FactoryTalk" brand.

In Arena, the user builds an experiment *model* by placing *modules* (boxes of different shapes) that represent processes or logic. Connector lines are used to join these modules together and specifies the flow of *entities*. While modules have specific actions relative to entities, flow, and timing, the precise representation of each module and entity relative to real-life objects is subject to the modeler. Statistical data, such as cycle time and WIP (work in process) levels, can be recorded and outputted as reports.

Arena integrates very well to Microsoft technologies. It includes Visual Basic for Applications so models can be further automated if specific algorithms are needed. It also supports importing Microsoft Visio flowcharts, as well as reading from or outputting to Excel spreadsheets and Access databases. Hosting ActiveX controls is also supported.

Notes

Uptake

Arena is used by many large companies engaged in simulating business processes. Some of these firms include General Motors, UPS, IBM, Nike, Xerox, Lufthansa, Ford Motor Company, and others. It has been noted that creating a simulation can require more time at the beginning of a project, but quicker installations and product optimizations can reduce overall project time. Arena can simulate diverse operation types, including call centers, for optimizing the use of agents and phone lines, the size and routing of pancake stacks in a food processing facility, and the design of a gold mine.

The original Arena Software was developed in Cambridge UK by completely different individuals as a Job Costing and Accounting system. It was originally programmed via Silicon Office, them moved to Modula-2 and finally to Microsoft Visual Basic. It was sold to BP, BAE systems and John Lewis as well as many architectural firms. Arena Software closed in 2000. The trademark was left to lapse at that time.

Commercial Software Editions

1. ***Professional Edition:*** The flagship product, provides the ultimate in functionality and flexibility to meet the needs of any simulation problem. Systems, regardless of complexity, can be represented and custom performance metrics may be measured and tracked. Included is the functionality of OptQuest, for optimizing systems, as well as the additional capability of object and template development.
2. ***Enterprise Suite:*** Offers the full power of Professional Edition with the added tools of 3D animation and the specialized templates that are effective for high-speed packaging operations (Packaging) as well as contact center operations (Contact Center).
3. ***Standard Edition:*** This mid-tier package has the versatility to solve simulation problems encountered in an array of industries and systems. This edition includes the base Arena templates
4. ***Basic Edition Plus:*** Includes the capability of the Basic Edition with the additional functionality of animation and material handling.
5. ***Basic Edition:*** Entry-level edition used for solving high-level business problems.

Academic Software Editions

1. ***Academic Lab Package:*** Academic version of the commercially available Enterprise Suite. This is 30-or more seat license is for academic, non-commercial usage. Universities that adopt the *Simulation with Arena* textbook are eligible for valuable offers and benefits.
2. ***Research Edition:*** This is the same edition as the Academic Lab Package, with this version for individual academic researchers. The same academic guidelines are specified for observance.
3. ***Student Edition:*** Free edition intended for students currently learning the software is included for download and/or included with many simulation textbooks. This version is perpetual, but limited in model size. This version is intended for academic, non-commercial usage. Universities that are using the software are eligible to make copies of the software to distribute to students for installation on their personal machines.

Notes

GASP

GASP IV is a FORTRAN-based simulation language which provides the framework for modeling systems involving both continuous and discrete phenomena. A general ization of the definition of "event" allows for time to be advanced in a next-event fashion while permit ting a step-wise evaluation of system state variables described by difference or differential equations. Subprograms are included in GASP IV to handle the details of state and event control (including state- variable integration when necessary), information storage and retrieval, collection and analysis of data on system performance, and generation of reports and plots. In this paper, the GASP IV philosophy and modeling approach are described. Descriptions of the subpro grams included in GASP IV and the required user- written subprograms are given. The types of applica tions that have utilized GASP IV are listed. A com panion paper (beginning on p. 71) presents a detailed example of the use of GASP IV for simulating a continuous reaction process involving discrete startups and shutdowns.

Plant Simulation

In times of increasing cost and time pressures in production, along with ongoing globalization, logistics has become a key factor in the success of a company. Money can be lost daily through inefficient schedules, local instead of global optimization, inefficient resource allocation, and poor productivity. The need to deliver just-in-time (JIT) / just-in-sequence (JIS), introduce Kanban, plan and build new production lines and manage global production networks requires objective decision criteria to help management evaluate and compare alternative approaches.

Tecnomatix Plant Simulation is a discrete event simulation tool that helps you to create digital models of logistic systems (e.g., production), so that you can explore the systems' characteristics and optimize its performance. These digital models allow you to run experiments and what-if scenarios without disturbing existing production systems or – when used in the planning process – long before the real production systems are installed. Extensive analysis tools, such as bottleneck analysis, statistics and charts let you evaluate different manufacturing scenarios. The results provide you with the information needed to make fast, reliable decisions in the early stages of production planning.

Using Plant Simulation, you can model and simulate production systems and their processes. In addition, you can optimize material flow, resource utilization and logistics for all levels of plant planning from global production facilities, through local plants, to specific lines.

Caselet

Mobiles with Five-language user Menu

Bangalore-based United Telecoms has launched radiation certified WIWO mobiles with user menu in five Indian languages — Tamil, Telugu, Kannada, Malayalam and Hindi.

"WIWO offers an 'Indian solution' by providing regional customers a choice of five Indian-language menus; it has entered the market with the tagline 'the new hello'; it is trendy, available in eight models, loaded with innovative features and applications. It offers differentiating features like noise-cancelling earplugs, dual battery, replaceable extra speaker, WIWO user interface and animation, motion sensor and power torch. Additional features like 3.5mm music jack in low-end phones, camera, bluetooth and expandable memory make the WIWO phones an exciting buy for the discerning consumer," Mr T.C. Sudhir, Executive Director, Teleworld United said.

Contd...

Notes

The company will initially launch the WIWO mobiles in South India and leverage on its sales and service network of 125 centres in the region. The national rollout is expected in three months.

Mr Sudhir said the WIWO mobiles target the country's discerning youth, as they aspire for differentiated yet relevant products.

United Telecom Group Chairman Dr C. Kasarbada Rao said the group has invested over ₹ 200 crore in telecom manufacturing infrastructure.

"With a projected turnover of ₹ 270 crore, ₹ 460 crore and ₹ 690 crore in 2011, 2012 and 2013 respectively, the future plans of WIWO Mobiles include an in-house product design and development lab at a cost of ₹ 5 crore, and a product testing and simulation lab at a cost of ₹ 5 crores by 2011. The company also plans to spend ₹ 125 crore on marketing over the next three years," he said.

Source: http://www.thehindubusinessline.com/todays-paper/tp-marketing/article1007340.ece

14.4 Summary

- Block oriented simulation languages are based on the method of analogue computers. The system must be expressed as a block diagram that defines the interconnection of functional units and their quantitative parameters.
- Expression oriented continuous languages are based on writing expressions (equations) that represent the mathematical model. So the system simulated must be expressed by a set of equations.

14.5 Keywords

C/D: Continuous/Discrete (C/D)

CTDS: Continuous Time Dynamic Systems

DEDS:Discrete Event Dynamic Systems

DSS: Discrete-system Simulator

GPSS: General Purpose Simulation Languages

ISI: Intelligent Simulation Interface

VIM: Visual Interactive Modelling

VIS: Visual Interactive Simulation

WIP: Work in Progress

14.6 Self Assessment

Fill in the blanks:

1. The alternative to a block structured language is an …….. language, such as SIMSCRIPT or FORTRAN.
2. General purpose languages allow greater programming ……..
3. ……. simulation software meets the needs of equally the casual, first-time user and the full-time simulation model builder.
4. The typical blocks available inmost continuous block oriented languages are ……, limiters, delays, multipliers, hysteresis, constant values, adders, holders, gain (coefficient) and other..

Notes

5. Users of simulation languages have traditionally had to choose between one of two types of …….
6. A system is modelled as transactions enter the system and are passed from one …….. to another.
7. The addition of some features of GPSS to SIMULA 67 was presented at a previous ……..
8. Simulation models are usually easier to ……. when written in a simulation language.
9. General purpose languages are obtainable on …… every computer.
10. Most simulation languages provide better ……..
11. An efficiently written general purpose language may require less ……. than the corresponding program written in a simulation language.
12. ……. includes the Visual SLAM simulation language to build network.
13. SLX's zero tolerance" policy for errors was derived from a similar policy in ……...
14. ……. is a simulator/assembler for the machine language.
15. ……. is a menu driven simulation package mainly used in manufacturing and logistics.

14.7 Review Questions

1. What are the major advantages of special purpose languages?
2. Modern simulation languages such as SIMSCRIPT II and SIMULA 67 are very powerful general purpose languages which contain facilities to handle lists and to schedule events in simulated system time. Explain.
3. Explain in detail General Purpose Simulation Packages.
4. Simulation languages provide a natural framework for simulation modeling. Comment.
5. How parallel simulation languages differ from general purpose programming languages?
6. Do you think SIMUL8 uses dynamic discrete simulation?
7. AweSim takes advantage of the latest in Windows technology to integrate programs and provide componentware. Give Reasons.
8. GPSS/H is a very powerful simulation tool. Elaborate.
9. Explain the features of Simpsim.
10. What are the basic Components of SIMUL8 ENVIRONMENT?

Answers: Self Assessment

1. algebraic	2. flexibility
3. autoMod	4. integrators
5. language	6. service
7. SIMULATION CONFERENCE	8. change
9. virtual1y	10. error detection
11. execution time	12. AweSim
13. GPSS/H	14. SimpSim
15. Taylor II	

LOVELY PROFESSIONAL UNIVERSITY
Jalandhar-Delhi G.T. Road (NH-1)
Phagwara, Punjab (India)-144411
For Enquiry: +91-1824-300360
Fax.: +91-1824-506111
Email: odl@lpu.co.in

Notes

14.8 Further Readings

Books

Balci, O.,(1994), Validation, verification, and testing techniques throughout the life cycle of a simulation study, Annals of Operations Research, 53: 121–173.

Balci, O., (2001), A methodology for certification of modeling and simulation applications, ACM Transactions on Modeling and Computer Simulation, 11: 352–377.

Birta, L.G. and Ozmizrak, N.F., (1996), A knowledge-based approach for the validation of simulation models: The foundation, ACM Transactions on Modeling and Computer Simulation, 6: 67–98.

Boehm, B.W., (1979), Software engineering: R&D trends and defense needs, in: P. Wegner (Ed.), Research Directions in Software Technology, MIT Press, and Cambridge, MA.

Cellier, F.E., (1986), Combined discrete/continuous system simulation – application, techniques and tools, in Proceedings of the 1986 Winter Simulation Conference.

Department of Defense (DoD) Recommended Practices Guide (RPG) for Modeling and Simulation VV&A, Millennium Edition (available at http://vva.dmso.mil).

General Accounting Office, (1976), Report to the Congress: Ways to improve management of federally funded computerized models, report LCD-75-111, U.S. General Accounting Office, Washington, DC.

General Accounting Office, (1979), Guidelines for model evaluation, report PAD-79-17, U.S. General Accounting Office, Washington, DC.

General Accounting Office, (1987), DOD simulations: Improved assessment procedures would increase the credibility of results, report GAO/PEMD-88-3, U.S. General Accounting Office, and Washington, DC.

Ören, T.I., (1981), Concepts and criteria to access acceptability of simulation studies, Communications of the ACM, 24: 180–189.

Online links

http://en.wikipedia.org/wiki/Advanced_Continuous_Simulation_Language

http://en.wikipedia.org/wiki/Simulation_language