AD-771 614

COMPUTER SIMULATION METHODS FOR MILITARY OPERATIONS RESEARCH

William K. McQuay

Air Force Avionics Laboratory
Wright-Patterson Air Force Base, Ohio

October 1973

DISTRIBUTED BY:

NTIS

National Technical Information Service
U. S. DEPARTMENT OF COMMERCE
5285 Port Royal Road, Springfield Va. 22151

AFAL TR 73-341

AD 771614

COMPUTER SIMULATION METHODS FOR MILITARY OPERATIONS RESEARCH

WILLIAM K. McQUAY, CAPTAIN, USAF

TECHNICAL REPORT AFAL-TR-73-341

OCTOBER 1973

Approved for public release; distribution unlimited

Reproduced by
NATIONAL TECHNICAL
INFORMATION SERVICE
U S Department of Commerce
Springfield VA 22151

AIR FORCE AVIONICS LABORATORY
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

ACCESSION for
NTIS White Section
DDC Buff Section
UNANNOUNCED
JUSTIFICATION
BY
DISTRIBUTION/AVAILABILITY CODES
Dist. AVAIL. and/or SPECIAL
A

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

AIR FORCE/56780/30 November 1973 — 200

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* Air Force Avionics Laboratory	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED 2b. GROUP

3. REPORT TITLE

Computer Simulation Methods for Military Operations Research

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

Technical Report

5. AUTHOR(S) *(First name, middle initial, last name)*

William K. McQuay, Captain, USAF

6. REPORT DATE October 1973	7a. TOTAL NO. OF PAGES 228	7b. NO. OF REFS 42
8a. CONTRACT OR GRANT NO. b. PROJECT NO. 691X c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) AFAL TR 73-341 9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*	

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY AFAL/WRA Wright-Patterson AFB, Oh 45433

13. ABSTRACT

Computer simulation is a major evaluation tool used by military decision makers. This report is a comprehensive reference on computer simulation techniques and the basic concepts employed in model building. It is suitable for use by both military planners and operations research analysts.

The report includes a general discussion of model building as a decision-making aid and the methodology of a simulation study. Since representing random behavior is a significant part of most models, generation of pseudorandom numbers and random variates are discussed in detail. Statistical analysis of model output is covered in depth and a sample simulation model is described and analyzed. Several typical military applications of simulation models are briefly discussed.

DD FORM 1 NOV 65 1473

UNCLASSIFIED

Security Classification

ii

UNCLASSIFIED
Security Classification

14. KEY WORDS	LINK A ROLE	LINK A WT	LINK B ROLE	LINK B WT	LINK C ROLE	LINK C WT
Computer Simulation						
Models						
Digital Models						
Mathematical Models						
Simulation						

*U.S.Government Printing Office: 1973 — 758-428/266

UNCLASSIFIED
Security Classification

COMPUTER SIMULATION METHODS FOR MILITARY OPERATIONS RESEARCH

WILLIAM K. McQUAY, CAPTAIN, USAF

Approved for public release; distribution unlimited

id

FOREWORD

The significance of this effort to the Air Force is that it provides military operations research analysts with a comprehensive reference on computer simulation techniques. This report summarizes the concepts and methods used in building simulation models for operations research analysts who have had limited experience in simulation and for veteran model builders who desire an easily available reference. Additionally, it provides background information for users of simulation analyses so that they may better understand the source of the data generated for their reports.

This report is an in-house effort conducted by the Air Force Avionics Laboratory, Electronic Warfare Division, Analysis and Evaluation Branch. The project engineer was Capt W. K. McQuay (AFAL/WRA).

This report has been reviewed and is approved for publication.

Ollie H. Edwards

OLLIE H. EDWARDS
Colonel, USAF
Chief, Electronic Warfare Division

ACKNOWLEDGEMENTS

The author gratefully acknowledges the many helpful suggestions from Maj Gene E. Perkins, Hq AFLC, Wright-Patterson AFB and his contribution of the subsection "Computer Simulation in Logistics." I also thank Messrs Leonard L. Brown and William H. Bond (AFAL/WRA) for reviewing the draft report.

Further gratitude is expressed to Mrs. Connie Hukle and Mrs. Donna Phelps (AFAL/WRA) for their patient and excellent typing of this report.

ABSTRACT

Computer simulation is a major evaluation tool used by military decision makers. This report is a comprehensive reference on computer simulation techniques and the basic concepts employed in model building. It is suitable for use by both military planners and operations research analysts.

The report includes a general discussion of model building as a decision-making aid and the methodology of a simulation study. Since representing random behavior is a significant part of most models, generation of pseudorandom numbers and random variates are discussed in detail. Statistical analysis of model output is covered in depth and a sample simulation model is described and analyzed. Several typical military applications of simulation models are briefly discussed.

TABLE OF CONTENTS

SECTION		PAGE
I	INTRODUCTION TO OPERATIONS RESEARCH. . . .	1
II	THE ESSENTIAL APPROACH TO COMPUTER SIMULATION	28
III	REPRESENTING RANDOM BEHAVIOR	37
IV	SYSTEM REPRESENTATION.	79
V	OUTPUT GENERATION AND STATISTICAL ANALYSIS	96
VI	A SIMULATION MODEL FOR A CLASSICAL PROBLEM.	131
VII	COMPUTER SIMULATION LANGUAGES.	157
VIII	MILITARY APPLICATIONS OF COMPUTER SIMULATION	167
	REFERENCES	183
	APPENDIX	187
	GLOSSARY	199
	INDEX	227

LIST OF ILLUSTRATIONS

FIGURE		PAGE
1	A Classification of Decision-Making Aids	5
2	The Structure of Simulation	12
3	A Simplified View of Three Types of Simulation	14
4	Open and Closed Loop Systems	17
5	The Objective of Simulation	18
6	Computer Components	21
7	Language Translation in Computer Programming	23
8	Flowchart Description of an Algorithm	27
9	Steps in a Simulation Study	31
10	FORTRAN Subroutine for Multiplicative Congruential Generator	44
11	FORTRAN Subroutine for Mixed Congruential Generator	46
12	FORTRAN Subroutine for Additive Random Number Generator	48
13	FORTRAN Subroutine for Pseudorandom Number Generator	68
14	A FORTRAN Function for Exponential Random Variates	70
15	A FORTRAN Function for Geometric Random Variates	71
16	A FORTRAN Function for Negative Binomial Random Variates	72
17	A FORTRAN Function for Uniform Random Variates on (a,b)	73

FIGURE		PAGE

18	A FORTRAN Function for Binomial Random Variates	74
19	A FORTRAN Function for Poisson Random Variates	75
20	A FORTRAN Function for Gamma Random Variates	7
21	A FORTRAN Function for Normal Random Variates	
22	Basic Structure in Discrete Event Simulations	83
23	A Comparison of Real World Events and Fixed Time Increment, and Next Event Increment Simulated Time	85
24	Flowchart for Fixed Time Increment Time Flow Mechanism	88
25	Flowchart for Next Event Time Flow Mechanism	89
26	Patrolling Repairman Problem with N=4 . . .	93
27	Measurement of Attributes Using Truncation and Stratified Sampling	107
28	Flowchart for Machine Interference Problem	133
29	Sample Printout for Centrally Located Repairman	149
30	Sample Printout for Patrolling Repairman ...	150
31	The Structure of Simulation in EW	170
32	The Interrelationship of Modeling Subtasks .	174
33	Typical Model Output Illustrating Penetrator Kill as a Function of Missile Miss Distance.	176
34	Effectiveness of AAA Tracking System Performance	178

FIGURE		PAGE
35	Penetrator Kill as a Function of Distance From Missile Site	179
36	Bubble Sort FORTRAN Subroutine	188
37	Virtual Sort FORTRAN Subroutine	189
38	Maximum Search FORTRAN Subroutine	191
39	Distribution of Sortie Lengths	192
40	A FORTRAN Function for Table Look-Up . . .	193
41	Cumulative Distribution of Sortie Lengths .	193
42	Table Look-Up with Interpolation	194

LIST OF TABLES

TABLE		PAGE
1	Computer Word Lengths	43
2	Chi Square Test on 5000 Numbers	51
3	Chi Square Tests of Local Behavior	52
4	KS Goodness of Fit Test	54
5	Runs Above and Below Mean	55
6	Correlation Test of 4985 Numbers	56
7	Runs Up	57
8	Poker Test	58
9	Coupon Collector Test	58
10	Permutation Test	59
11	Spectral Test	59
12	Serial Test	61
13	Gap Test	62
14	Initial Value Test	63
15	Mean Generation Time	64
16	Summary of Statistical Tests	66
17	Comparisons of Tests	67
18	Convergence to Steady State	147
19	Values of Autocorrelation Function	151
20	Covariances for Various Sample Sizes	152
21	Covariance Estimates	152
22	Summary of Efficiency Estimates	153

TABLE		PAGE
23	95% Confidence Intervals for Estimates of Total System Efficiency	154
24	Test of Difference of Means Total System Efficiency	156
25	The Availability of Simulation Programming Languages	160
26	A Comparison of Simulation with Real World .	172
27	Continuous Probability Distributions	195
28	Discrete Probability Distributions	196
29	ASCII Character Set	197
30	Powers of Two	198

SECTION I

AN INTRODUCTION TO OPERATIONS RESEARCH

Operations research (OR) is the application of mathematics and scientific methods to aid the decision making of management. The implications of this definition may be more apparent to the reader through a closer examination of the specifics of this statement. Aiding decision makers is an inherent characteristic of OR. No model or analysis study is ever so sufficient unto itself as to become independent of the judgment supplied by knowledgeble managers. Scientific suggests emphasis on objective methods of assessing a situation. Applied mathematics in a broad sense is a predominant part of most studies. First, mathematics is used as a tool to solve problems and, secondly, it plays a role in formulation of the problem typically as a mathematical model.

During World War II operational problems associated with radars, aircraft, submarines, and weapons allocation gave rise to highly skilled teams of mathematicians, physicists, and engineers which attempted to solve the problems. These quantitative methods evolved into operations research and later variations such as systems engineering, management science, cost-effectiveness analysis, and systems analysis. After the war these early operations researchers

transferred their experience throughout the military and industry. Increasing interest led to the development of theory in linear and dynamic programming, queuing, gaming, network analysis, inventory, scheduling, and simulation. These theories have been applied to manufacturing, transportation, communications, construction, health care, banking, and military operations.

There is often confusion concerning the names of fields which are variations of OR. If emphasis is given to planning and design of new industrial or military systems to increase performance of existing operations, then the term "systems research" or "systems engineering" is used. Dealing with the problems of efficient management or control of systems is called "management science." Attention to differences in costs or resource requirements among available alternatives is referred to as "cost effectiveness analysis." Finally, the term "systems analysis" is applied to any systematic approach to the comparison of alternatives. One should note that in the military establishment, systems analysis connotes long-range planning and is associated with problems where one decides what ought to be done, not just how to do it. The total analysis is more complex, usually qualitative, and seldom suited to quantitative optimization.

In any event, little distinction remains between operations research and its variations. The differences are a matter of emphasis and it's not worthwhile attempting to firmly distinguish among them. The term operations research will be used throughout this report and is considered synonymous with systems analysis.

Basic Themes in OR

Three pervasive and interrelated themes are found in OR literature. First, there is emphasis on optimization. Typically the optimization is constrained so that values of the decision variables which maximize the objective function are restricted so as to satisfy certain technological restraints. Secondly, the analyst seeks derivation of analytic properties of the model. These could include sensitivity of an optimal solution to model parameters, the structural form of an optimal solution such as an (s,S) policy in inventory theory, or operating characteristics of the solution, for example, the probability of no bombers reaching their targets. Thirdly, there is explicit recognition of system interaction. The results of an OR analysis cannot be determined or applied in isolation from the surrounding military or industrial environment, but are a part of and are affected by that environment. Since OR cuts across many fields---engineering science, economics, physical science, and biological science---it

must be regarded as a systems effort.

Emphasis on making decisions or taking actions is central to operations research applications. We will now consider the decision making process and then the availability of aids which the decision maker can employ.

The Decision Making Process

Since military decisions involve the security of the nation and its scarcest resources, decision makers must make the best possible decision every time.

The decision making process can be viewed as follows: (1) A review of the goal or ends to be served, (2) a study of proposed alternatives, (3) the ordering or ranking of the alternatives in some rational arrangement, and (4) the selection of one or some combination of the alternatives. Thus, decision making may be defined as selecting a course of action from among a number of alternatives according to some criterion. In selecting the best alternative, it is not always immediately apparent which alternative is most desirable. In any case it is necessary to develop a method of measuring the effectiveness of various solutions. This process alone will be an aid to the decision maker in clarifying the objective. It may also be necessary to work toward subgoals. For example, the main objective of a logistics system is to maximize the effectiveness of operational units. But a more tractable goal may be to

minimize the supply backorder rate, supply fill rate, or aircraft ready rate.

The statistician and mathematician sh considerable light on decision making through the application of statistically derived models and game theory to military management problems. Economics, with its well developed analytical tools, addresses the problem of resource allocation. Information science focuses attention on the vital ingredient of management decision making - the flow and organization of data. All of these fields contribute to the various techniques available to the operations research analyst.

Decision Making Aids

Several methods are available to aid the decision maker in determining and selecting among alternatives. Figure 1 shows the relationship of various decision making aids. Based on previous experience alone, the decision maker may use his intuition to decide which solution to adopt. A small business man such as the corner grocery store (if any remain) may have developed a rule of thumb or may simply "know" how many items to order to retain a certain level of inventory. After all, it's always worked in the past. The inefficiency of such trial and error approaches may never be apparent to the decision maker especially if the penalty, lost customers say for an

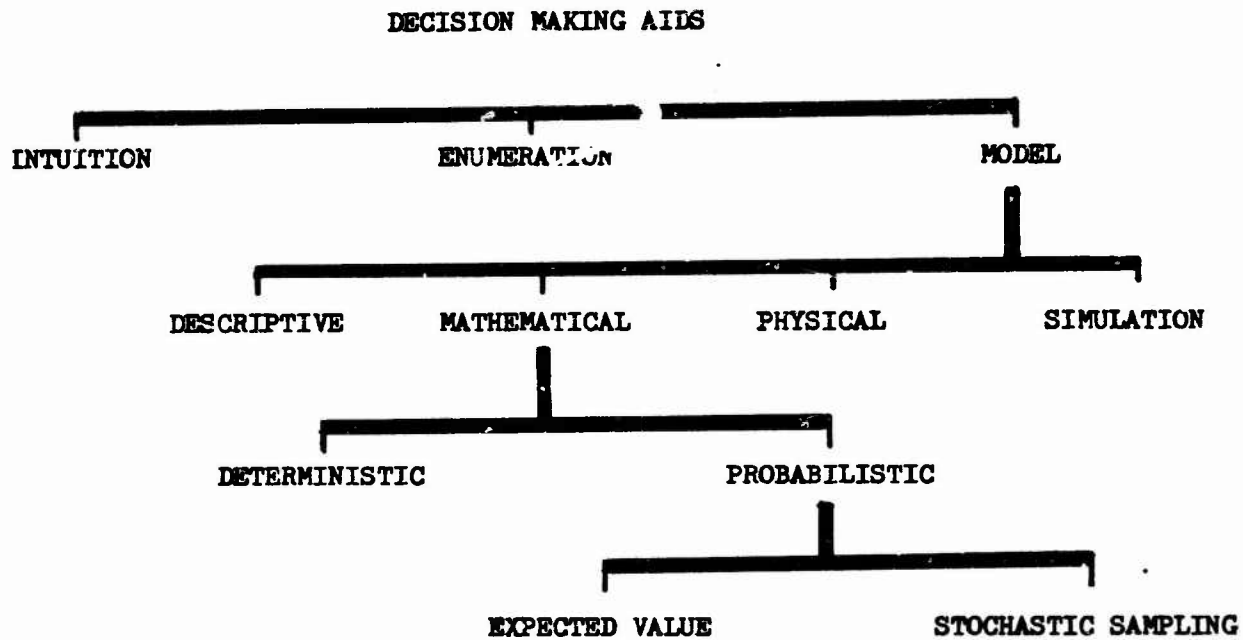


Figure 1. A Classification of Decision Making Aids

out of stock condition, is small. It may be the closest store so the customers always return.

A more rigorous approach is to enumerate all possible alternatives and select the best solution according to some criterion or objective function. Generally enumeration is too time consuming or physically impractical for numerous alternatives. The third possibility is to model the problem using a descriptive, mathematical, physical, or simulation model.

Descriptive models are expressed verbally in one's native language and generally are used in the humanities or social sciences. For example, once psychologists conceived of human beings as motivated solely by their need to reduce tension or discomfort. Man's behavior was explained in terms of the tension reduction model of motivation. With the model, man was described as an organism seeking to avoid discomfort.

Mathematical models use concise mathematical symbols to describe the status of variables in the system and the way in which they interact and change. Linear programming theory of operations research is a prime example of mathematical modeling. Another example is the classical transportation problem which can be generally formulated as follows: A product is available in known quantities at each of m origins. Given quantities of the product are

required to be shipped to each of n destinations. The minimum cost of shipping a unit of the product from any origin to any destination is known. The analyst must determine the shipping schedule which minimizes the total cost of shipment. The general mathematical model is:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

such that $\sum_{j=1}^{n} x_{ij} = S_i$ for $i = 1,2,\ldots, m$ (supply)

$$\sum_{i=1}^{m} x_{ij} = D_j \text{ for } j = 1,2,\ldots, n \text{ (demand)}$$

where x_{ij} = quantity of product to be shipped from origin i to destination j

c_{ij} = cost of shipping one unit of the product from source i to destination j

S_i = amount of the product available from source i

D_j = amount of the product desired at destination j

x_{ij} are non-negative integers where all S_i and D_j are positive integers such that

$$\sum_{i=1}^{m} S_i = \sum_{j=1}^{n} D_j$$

Incidentally, the solution is provided by the Transportation Simplex Algorithm.

A primary objective in developing a mathematical model is to provide a realistic representation of the behavior of the real system, whether certain or random. In those cases where the values of all parameters and data required by the model are known exactly, the model is called deterministic. However, decision making usually occurs in an uncertain environment, i.e., one does not always know for certain what will happen as a result of particular actions. By associating probabilities with the occurrence of a particular event, one can use the statistical results of the random process to form a probabilistic model.

There are basically two approaches in treating probabilistic phenomenon. One approach is to model the behavior in terms of the expected value of different states. The expected values as functions of time are determined from experimentation or observation and approximated with analytical functions. The expected value at a given time is calculated from the function and used as input to determine the probability of occurrence of the event. The advantage of the expected value approach is that only a single execution of the program is required to determine model results. For example, in engineering reliability studies, an equipment component often is found to have a constant mean time to failure throughout the

equipment lifetime. If $E(t) = 1/\lambda$; then it can be shown that $f(t) = \lambda e^{-\lambda t}$, the probability of failure at time t. By empirically determining, the above function can be used to determine the probability of the event "component fails" at time t.

The second approach employs stochastic sampling procedures and is often mistakenly called Monte Carlo. It is distinguished from the expected value probabilistic model by the use of statistical sampling and random numbers. A typical application would be determining the single scan probability of detection of a target by a radar. The probability of detection (P_d) can be mathematically described by an exponential distribution such as

$$P_d = Ae^{-k\frac{s}{n}}$$

where A is an attention factor for the radar operator

K is a factor accounting for the radar type

$\frac{s}{n}$ is the signal to noise ratio.

To determine on a particular scan if the target is detected, a uniform random number (the sample value) is generated and compared to P_d. If the random number is less than P_d, the target is detected; otherwise, there is no detection.

Physical models are scaled replicas of the real system. Included are floor plan layouts, one-eighth scale wind

tunnel aircraft models, or full scale mock-ups of newly designed weapon systems. In the most trivial sense, a street map can be considered a physical model of a city. While on the other extreme the petroleum and chemical industry often build a half scale fully operating version of a new refinery to field test a new design or procedure.

The last and, for our purposes, the most important model is simulation. Simulation is the representation of certain features of the behavior of a physical system by the behavior of another system. In many cases, simulation involves mathematical/logical models of real systems. In fact, the development of a simulation model usually starts with a mathematical model of the real world system. Simulation has received increased emphasis in recent years since it allows the examination of the dynamic interrelationships of variables and parameters especially under continuously changing situations.

The Structure of Simulation

Figure 2 shows the structure of simulation as it relates to method, approach, and objective. The simplest simulation model is a manual simulation in which model behavior, bookkeeping, and any other actions are accomplished by the person participating. To illustrate, consider a war game. A war game is a simulation of a military operation involving two or more opposing forces using

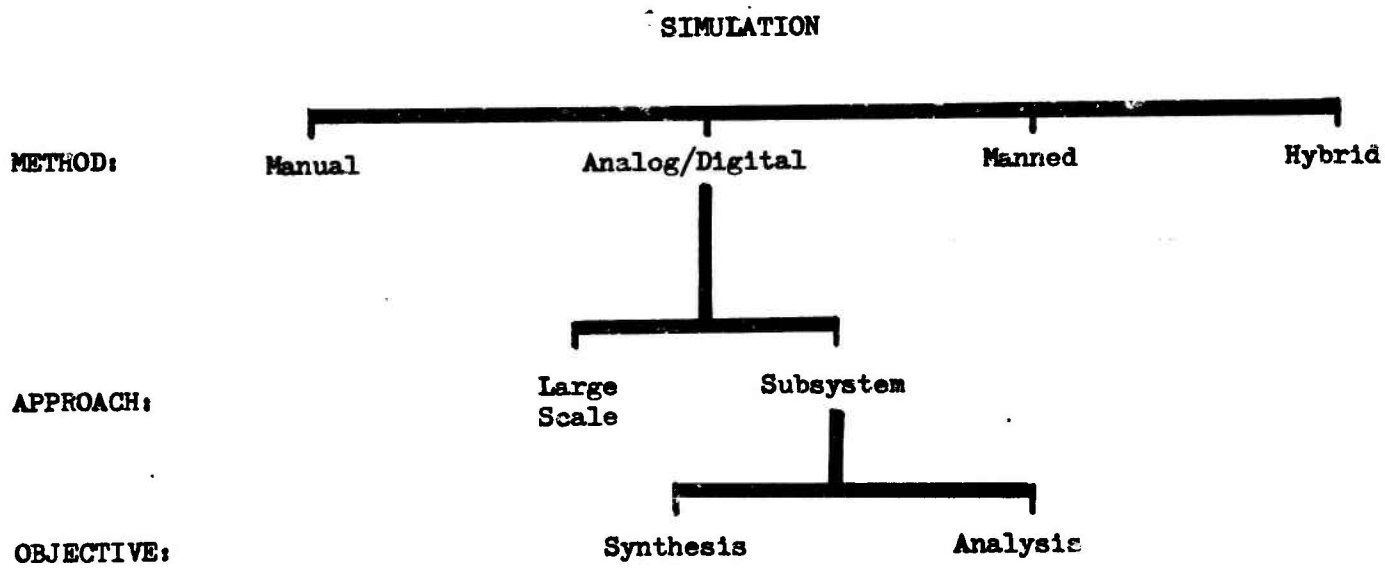


Figure 2. The Structure of Simulation

rules, data, and procedures designed to depict a real-life situation. An educational war game provides the participant with decision making experience while an analytical war game provides the participant or an observer with decision making information. In a manual war game all forces and equipment are represented artificially and participants play the game by providing the movement of the simulated forces and making game decisions. For instance, toy soldiers, artillery, and aircraft might be positioned on a map to represent a battle. The players could move the figures and equipment over the map to simulate different strategies and thereby gain insights into their relative success or failure.

The most frequently used method for simulation is an analog or digital computer. The digital computer simulation model is the subject of this report and will be discussed in depth throughout the remaining sections. Again consider the war game example. Nowadays war games are often executed on a digital computer according to some fixed logic incorporated into the program. In a pure digital model there are no real players and all physical world characteristics are pre-programmed or are input. The computer allows the war game to be rapidly repeated with different inputs and generates information for the comparison of various alternative actions.

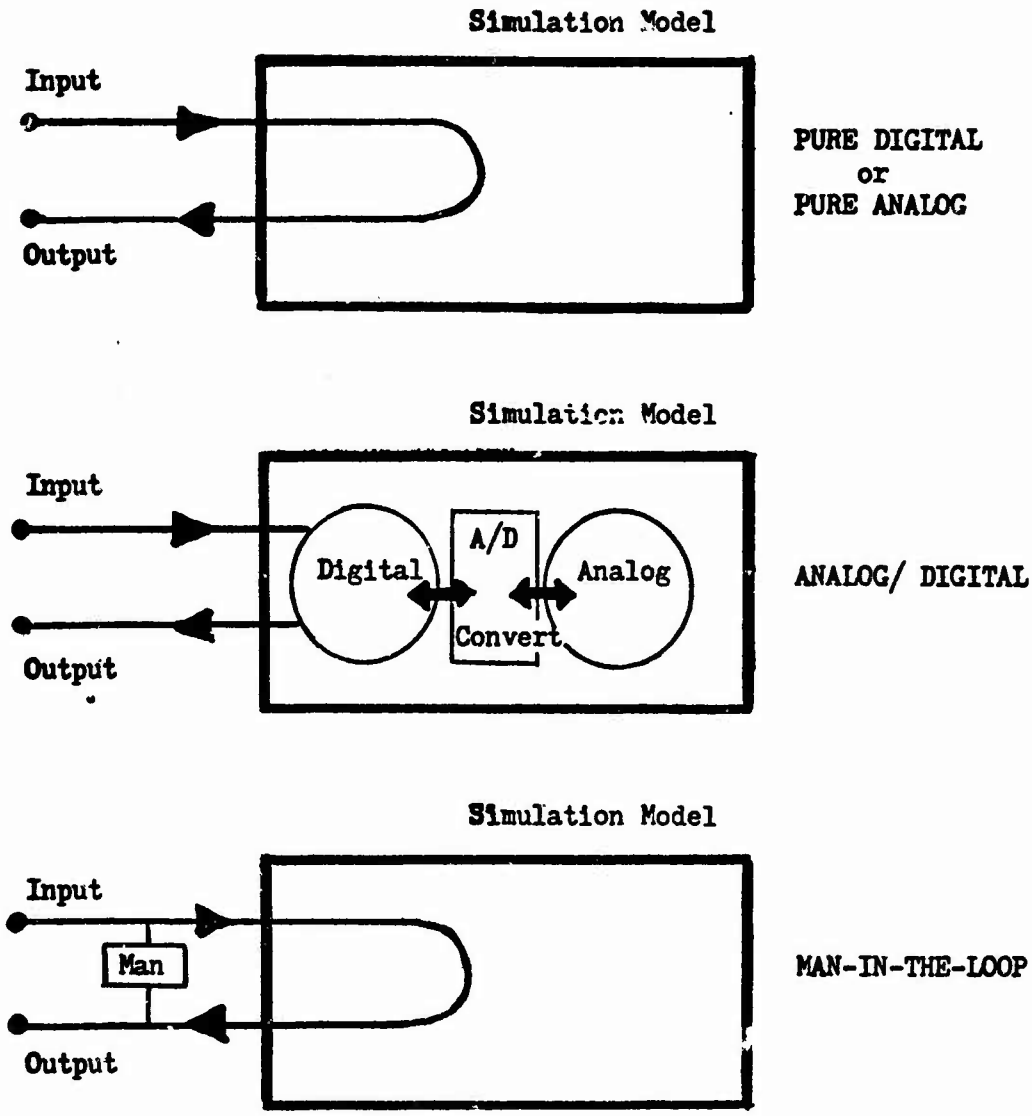


Figure 3. A Simplified View of Three Types of Simulation

If the simulation is complex but human interaction is still desired, the model could be manned with the bookkeeping function of what, where, and when as well as related numerical calculations relegated to the computer. These manned simulations are also called "man-in-the-loop" or man-machine games. Depending on the nature of the simulation, the human interaction may be continual decision making or only occasional inputs. A well-known manned simulation is the Link Trainer which simulates an aircraft's responses to the actions of a student pilot.

Finally, the simulation may be hybrid, in that, various combinations of the other methods and even physical models (replicas) are included. For example, a tactical air training exercise might involve real aircraft against real and simulated radars which control digital computer simulations of surface-to-air missiles or anti-aircraft artillery.

As an aside, one should note that the term "computer simulation" refers to simulations which use either analog or digital computers entirely or in part. Thus, the term applies to all types of simulation except manual.

The simulation approach may be to examine overall behavior, such as the large scale simulation of an air defense system, or subsystem behavior such as the flight dynamics of a surface-to-air missile. Directly related to the approach is the concept of open and closed loop

systems. Figure 4 illustrates the two basic structures. The difference between them is that the output of the closed loop system feeds back into the input.

In the open loop system, variations in the output are caused by variations in the input or system parameters. A parametric study could determine the appropriate cause and effect relationships. On the other hand, closed loop systems have their own dynamics and tend to be parameter insensitive. In a closed loop system the input depends on the output. The change in input due to output may oppose the original output (negative feedback) and cause the system to underrespond. Oppositely, the output may feed back in such a manner as to increase the output (positive feedback).

Large scale simulations tend to be closed loop systems and simulations of subsystem behavior, open loop. When applied to the management of large systems, closed loop systems theory is called systems dynamics. The original application to industry and social systems was accomplished by Jay W. Forrester at Massachusetts Institute of Technology [Reference 11 and 12].

Simulation may have one of two objectives: analysis or synthesis. In many cases one is able to describe in detail the characteristics of a system and desires only to know how that system reacts to a given input. Hence, in analysis

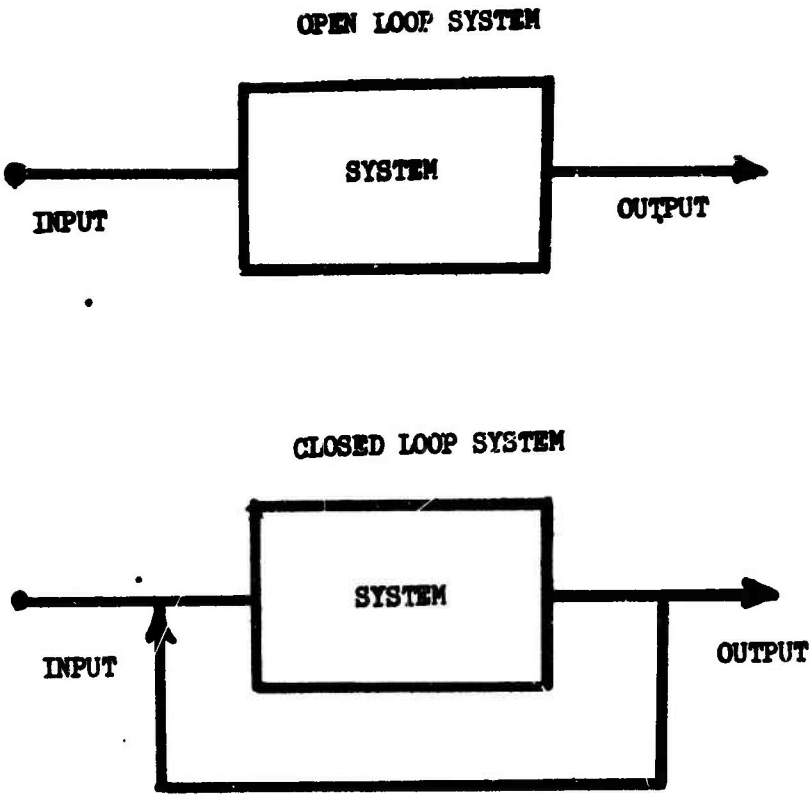


Figure 4. Open and Closed Loop Systems

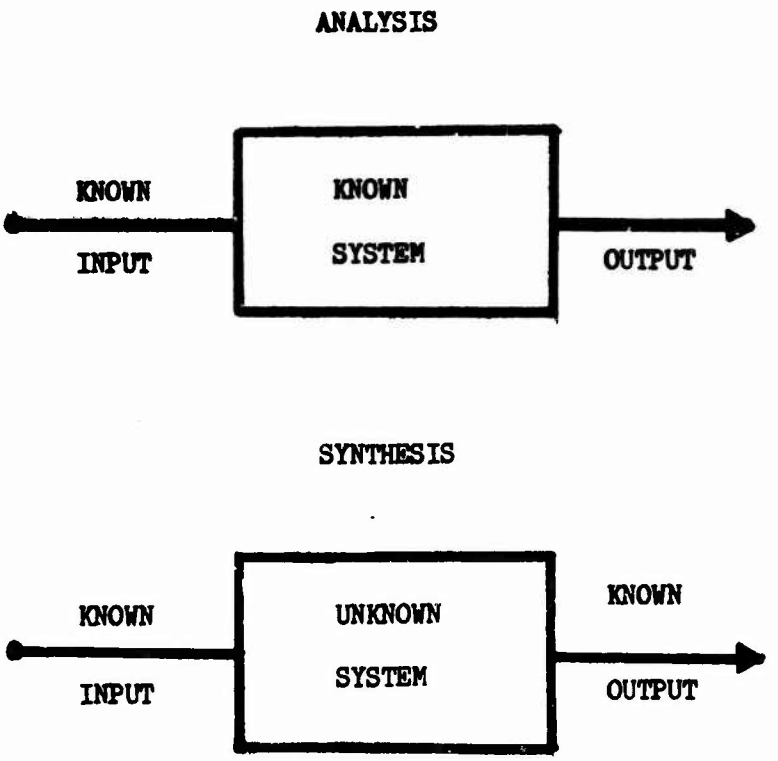


Figure 5. The Objective of Simulation

one inputs known values of parameters into a well-known system to determine a previously unknown output. As an example, suppose one models some ECM techniques. The unknown output is data on the survivability and expected cost of a mission for aircraft carrying the given ECM which penetrates the threat environment. But, in synthesis, one knows the input and output but desires a description of the model which produces that output. For example, a given amount of money is allotted for ECM (known input) and a given acceptable aircraft survivability is stipulated (known output), for which one must develop ECM concepts to provide that survivability for the given dollar cost.

Either analysis or synthesis can be equally well applied to an open loop simulation. But the situation is quite different for a closed loop system. For an open loop system, the relationship between input and output is essentially stable so that synthesis can determine a system description or analysis can determine cause and effect. In a closed loop, the output affects the input and one cannot be sure whether the output is determined externally by the input or internally from output feedback. In a closed loop system, synthesis usually fails so that the analyst must employ other approaches from feedback control theory.

Digital Computers and Programming

The complexity of mathematical models used in OR, the volume of data to be manipulated, and the magnitude of

computations necessitate the use of digital computers. The special significance of the computer lies in its ability to remember, process, and provide information at high speed. A brief discussion follows on the components of a computer system and the requirements for algorithmic representation of instructions to the computer.

Components of a Digital Computer

Any computer system is made up of hardware and software. The hardware is the physical equipment or associated devices and peripheral equipment in the computer system. The software is all programs and routines which control or extend the capabilities of the computer.

Hardware

The hardware components may be grouped into three major categories: (1) input unit, (2) output unit, (3) the central processing unit (or main frame). The central processing unit consists of three sections: control, arithmetic and logic, and primary storage (memory).

The input unit transcribes the input data from a source medium such as punched cards to an internal medium such as magnetic core storage. All such input data is converted by the computer to a binary form which can be stored in the computer's memory.

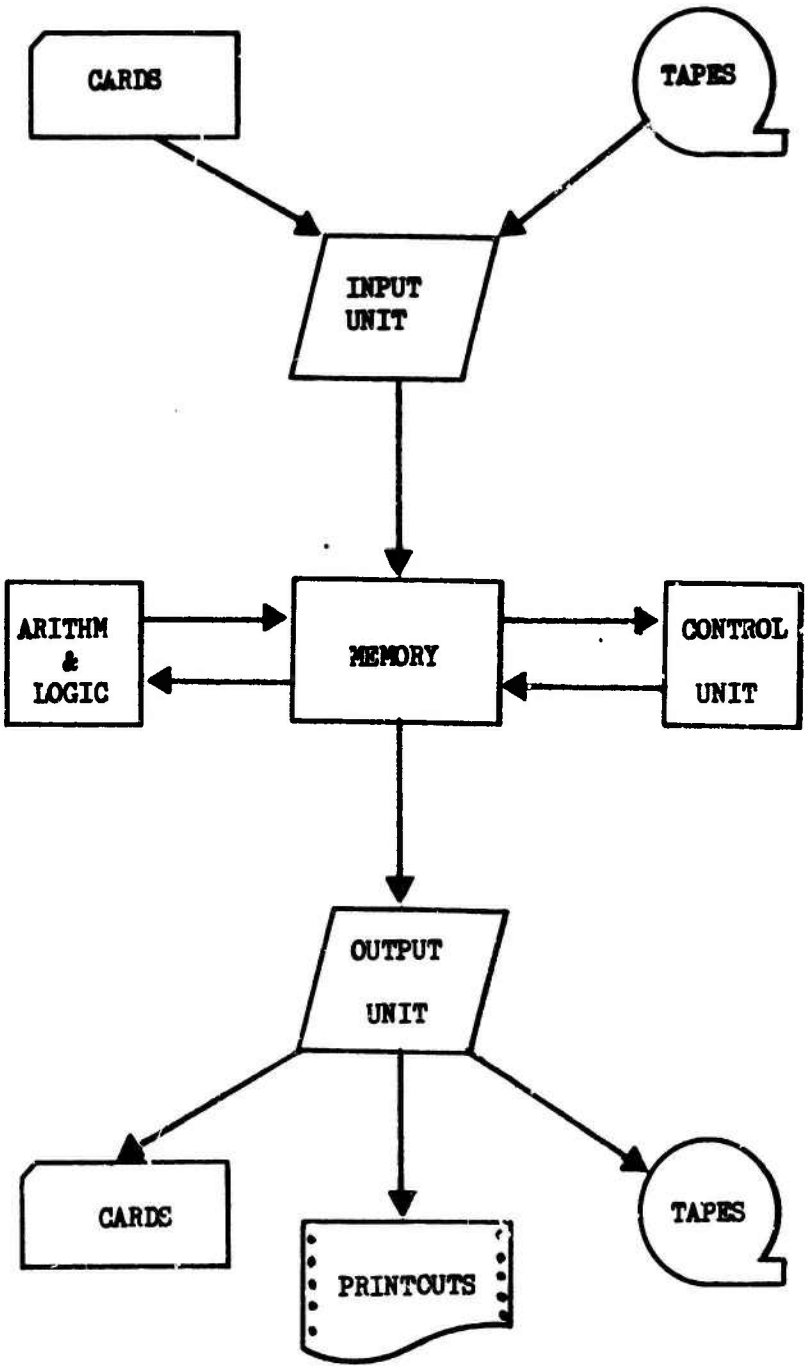


Figure 6. Computer Components

The output unit performs the reverse transcription process for information which has been derived from the data processed within the central processing unit. It transcribes the results of computation from the computer storage medium back to an external medium usable by the analyst.

The central processing unit is the heart of the computer. The control section decodes and interprets program instructions stored in memory and sends commands to the other computer components to execute those instructions. It also performs the timing and sequencing function to provide for proper manipulation of the problem data.

The arithmetic and logic section performs the basic operations: addition, subtraction, multiplication, division, and certain logical tests and branching. The execution of a single operation is measured in nanoseconds.

The primary storage or memory unit stores data and program instructions for instantaneous access by other computer sections. Memory is organized in a hierarchy according to speed and cost. Magnetic core is the fastest type of memory with magnetic drum, magnetic disc, magnetic tape, and punched cards or paper tape in descending order of cost and speed. A large system would include several memories of each kind.

Software

The software may be grouped into three major categories: (1) executive system, (2) command language, and (3) library.

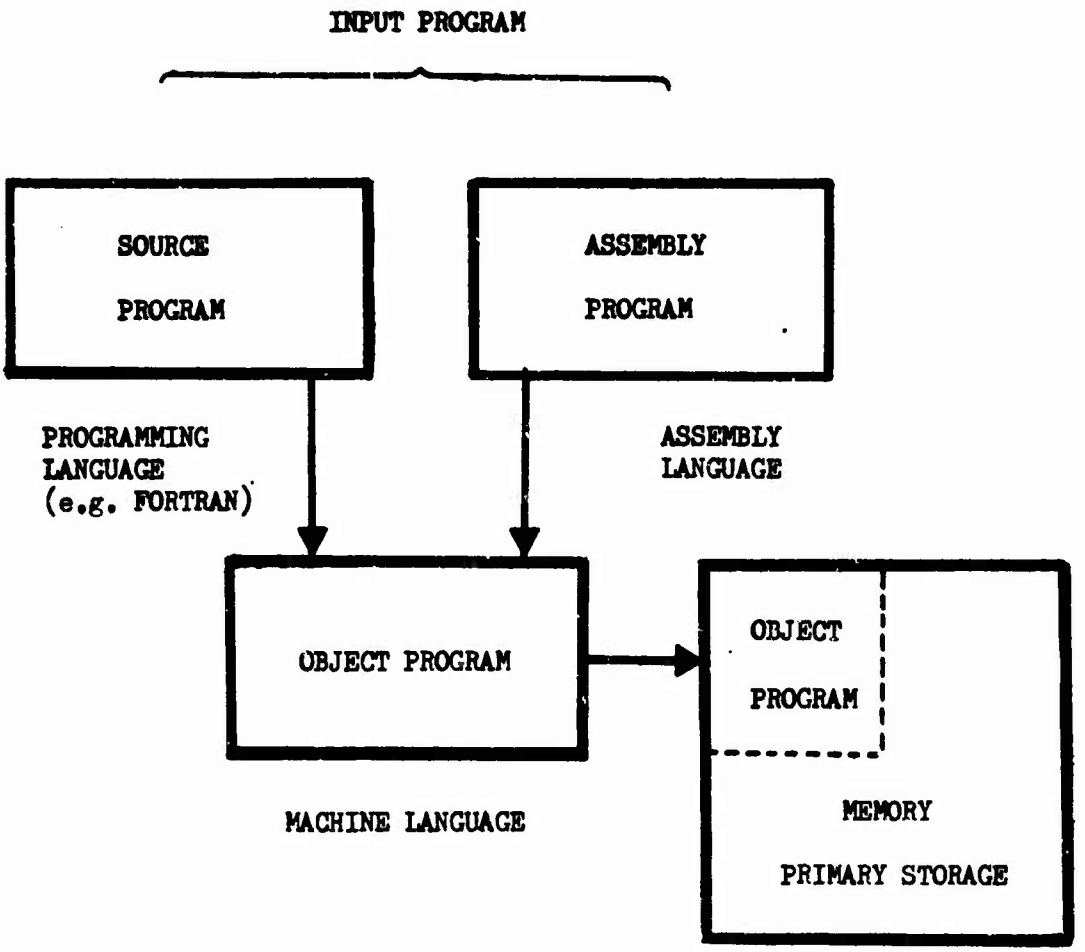


Figure 7. Language Translation in Computer Programming

The executive system is designed to organize and regulate the flow of work in the computer.

The software most directly affecting the user is the computer program. A program is a set of instructions written by a programmer which tells the computer what is to be done at each step in processing. The initial program or source is expressed in a computer programming language such as FORTRAN or ALGOL. The source program is translated or compiled into machine language for execution by the computer. In some cases, the analyst may write all or part of the source program in assembly language. The computer program in machine language is called the object program and is read into the memory unit. Machine language consists of a set of codes built from the binary digits used by the computer. Each procedure such as add or subtract, is coded by a combination of 0's and 1's.

Algorithms

A program is defined as a logical sequence of operations to be performed by a digital computer in solving a problem or in processing data. Since a computer program usually represents an algorithm, we need to examine the nature of algorithms.

An algorithm is a set of rules acting effectively according to a known objective on some input to produce an output in a finite time period. An algorithm has five

fundamental properties: finiteness, definiteness, input, output, and effectiveness. An algorithm can always be performed in a finite number of steps and is comprised of steps which are precisely and unambiguously specified. Clearly English with its numerous ambiguities is not suited to representing algorithms. The algorithm operates on a set of inputs to produce a set of outputs. In being effective, it should be possible to perform the individual steps of the algorithm and produce exact values.

English Language Description of an Algorithm

Suppose we wish to determine the square root X of a number A. Most digital computers use the Newton-Raphson method or some variation, i.e., repeatedly apply the formula:

$$X_i = 1/2\ (A/X_{i-1} + X_{i-1})$$

where X_o is initially some starting value from which we calculate an approximation to the real square root.
X_i is the approximation at iteration i (repetition of the process)
X_{i-1} is the previous value of X for i 2
X_o for i = 1

The first step is to set X_{i-1} to a starting value X_o. The starting value has a significant influence on the number of iterations required to converge to the square root. For convenience let's begin with $X_o = 1$. We must calculate a new value of X and test whether the absolute value of the difference between X_i and X_{i-1} is less than some pre-

determined tolerance. If they are within the tolerance we stop; otherwise, we do another iteration with X_i assigned as the value of X_{i-1}. The flowchart in Figure 8 concisely describes the algorithm in a clear and unambiguous fashion. It can then be implemented on any computer by programming in a language such as FORTRAN or ALGOL.

A FORTRAN Program of the Algorithm

The same algorithm as a FORTRAN subroutine would be as follows:

```
        SUBROUTINE SQRT  (A,X)
        TOLERN = 1.E-6
        X1 = 1
3       X = .5*(A/X1 + X1)
        IF (ABS(X-X1).LE.TOLERN) GO TO 7
        X1 = X
        GO TO 3
7       PRINT 8, A, X
8       FORMAT (1X, "THE SQUARE ROOT OF," F10.5," IS", F10.5)
        RETURN
        END
```

The subroutine is called from the main program, calculates the square root, writes the answer on the printer, and returns to the main program with the calculated value of X.

Algorithms are necessary since we must communicate with the computer in an unambiguous fashion. Therefore computer programming languages allow the analyst to describe a process or operation in algorithmic form.

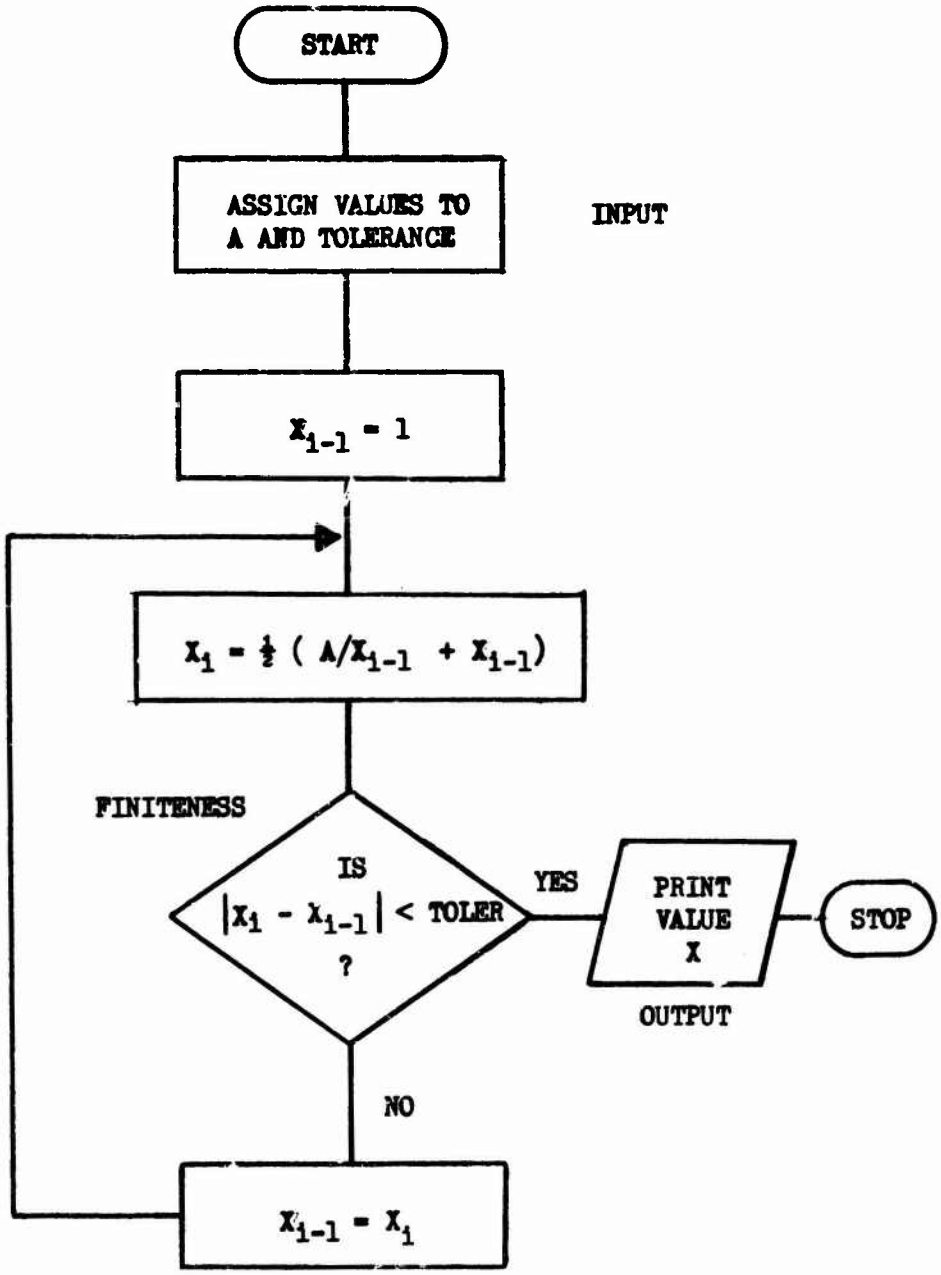


Figure 8. Flowchart Description of an Algorithm

SECTION II

THE ESSENTIALS OF COMPUTER SIMULATION

In Section I we sketched an outline of operations research and how simulation fits in as a decision making aid. The remainder of this report is concerned specifically with digital computer simulation models--- how to build them, use them, and interpret their outputs. In this section we will discuss when computer simulations are used, the advantages and disadvantages of simulation models, and general guidelines for a simulation study.

Why Simulate?

On his first exposure to computer simulation, an individual may validly pose the question - "Why should I use a simulation model?" One of the primary reasons for using simulation is that it is either impossible or very costly to observe detailed behavior of the real world system. For instance, without simulation it would be virtually impossible to observe the effects of electronic countermeasures employed onboard aircraft penetrating a large air defense system. Another reason may be that the real world system is so complex that it is impossible to describe it in terms of a set of mathematical equations suitable for analytic solution. Simulation makes it possible to experiment with the dynamics of the system

and study the complex interaction of subsystems. Through parameter variations, one can change the system's environment and observe the effects on the model's behavior. A simulation study can yield valuable insight into which variables and their interrelationships are most important. Dynamic Systems may be studied in real time, expanded time, or compressed time. Lastly, simulation models may be used as teaching devices, for experimentation in new situations which the real world system has not yet encountered, or for tests of new policies or strategies for operation of the system. In all cases, simulation is an effective means of generating data which would be otherwise difficult to obtain.

Advantages and Disadvantages of Simulation

Computer simulation possesses particular characteristics which provide both advantages and disadvantages for the user [8]. To begin on an optimistic note, let's look at the advantages. A simulation is completely repeatable since the analyst has complete control over model development, input data, and execution of the program. The model is an ideal system from the standpoint of collection and processing of data. Also, a simulation model is free from the physical limitations of the system being studied. For instance, a machine repair simulation can be run continuously for several hundred thousand seconds of

simulated time without worry about the repairman getting tired. Finally, the analyst is able to build realism into the model only constrained by computer size and cost.

Simulation also has its disadvantages. A simulation model is artificial since it expresses natural phenomena in purely symbolic terms. It can be inflexible since slight changes in the purpose of the model could cause drastic changes in the computer program. Lastly, a simulation study can be lengthy and the models quite complex and expensive.

The analyst must determine if simulation is the appropriate tool for his particular problem. It should be not only applicable but the lowest cost computational procedure for solution. Furthermore, the type of model used should have output which can be relatively easily interpreted by those who will use the results.

General Guidelines for a Simulation Study

The general approach used in a simulation study is basically the steps of the scientific method. In certain ways, a simulation study is no different from any other system study using some other technique. A flowchart of the steps is shown in Figure 9.

The approach begins with formulation of the problem. It may so happen that the initial statement of the problem differs considerably from the final version since re-

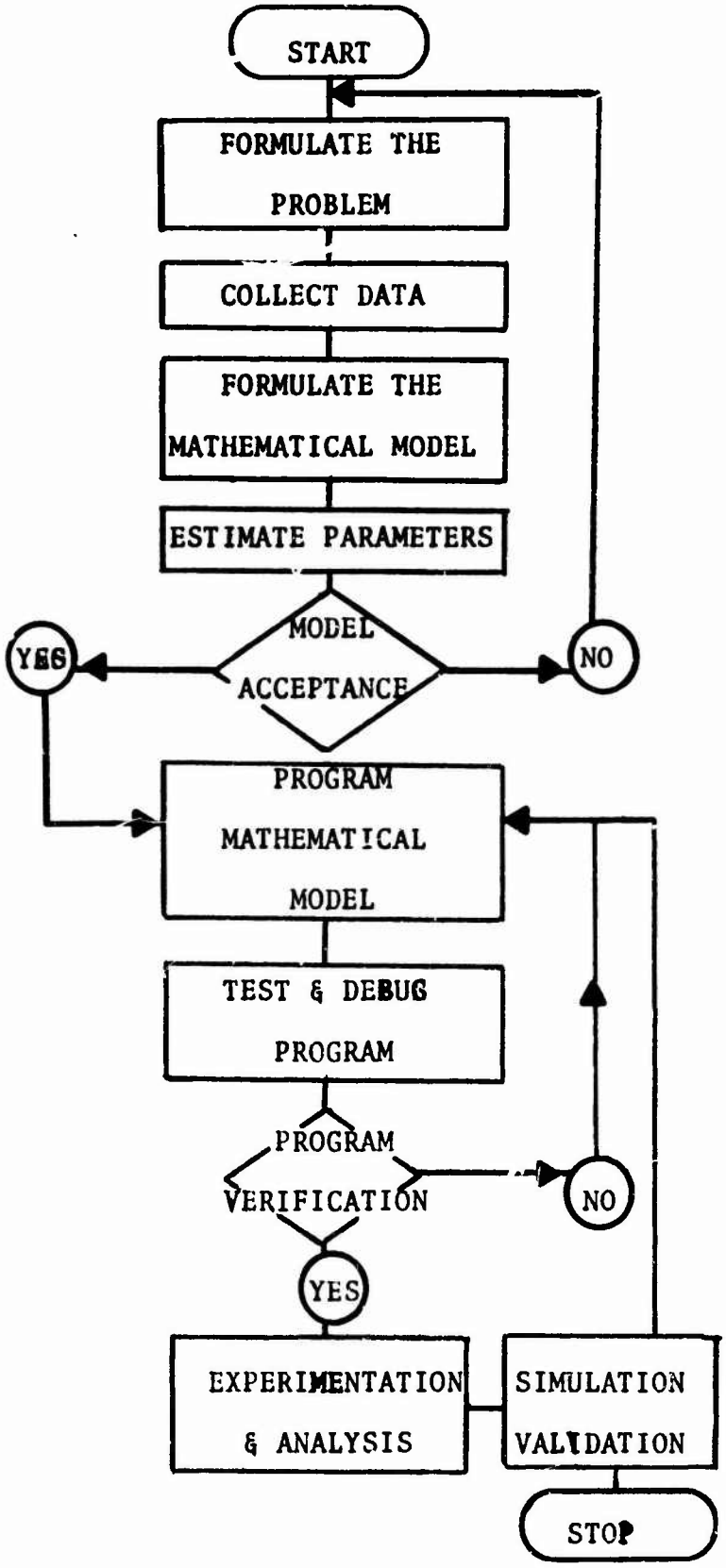


Figure 9. Steps in a Simulation Study

finement of the objectives usually occurs throughout the study. Additionally in this step, the analyst must decide on a set of criteria for evaluating the degree to which the objectives are fulfilled by simulation experiments conducted in the study.

The second step is to collect real world data. Some preliminary data was probably already collected in order to formulate the problem. Additional data is processed in some fashion so that it's in a suitable form to aid in formulating a mathematical model.

In formulating the mathematical model, the analyst must consider the number of variables to include, how complex the model should be, the computational efficiency of the equations or mathematical techniques employed, computer programming time required to implement the model, the degree of realism required or permissible, and the compatibility of the model with the objectives of the study. The greatest flexibility of model application and insight into the operation of the system results from models which are modularly constructed. The overall system is represented by a set of submodels of the individual components of the system. Through such an approach, the mind is able to grasp the basic relationships between subsystems, and the programmer's burden is somewhat lessened since he can independently check out the submodels.

The fourth step is estimation of parameters. The parameters are estimated by statistical inference from the real world system or from design specifications when the system does not yet exist. In many cases, the purpose of the study is parameter sensitivity analysis, i.e., determination of crucial parameters and the model's reaction over a particular range of values. For such studies the real world values are either unknown or not easily attainable and the objective is to determine which variables are important to accurately estimate.

The next step is model acceptance. It is important to examine the model structure, its underlying assumptions, and the estimates of parameters for accuracy and relevancy. The analyst must determine if all pertinent variables have been included and that none currently in use are superfluous, if relationships between variables are correctly formulated, and the level of detail is satisfactory for the study. If the analyst and decision maker are assured that the model meets their requirements, the study continues; otherwise, they must return to step 1.

When the mathematical model is accepted, it is programmed for the computer. The structure of the program is influenced by the availability of programming languages: either general purpose languages (GPL) or simulation programming languages (SPL). The choice of

languages is affected by the experience of the programmers, compatibility of the mathematical model and the language, and the type and form of output. In Section VII, we will discuss the choice between GPL and SPL in more depth.

Once the program is written, it must be tested and debugged. Depending on its complexity, this step could be the most frustrating and time consuming. At this time, the analyst must also define the initial state of the model and determine the number of samples needed for statistical analysis.

The eighth step is program verification. Does the program represent the model? Are there biases which influence the results? Is the output suitable? Does the model behave the way the analyst intended?

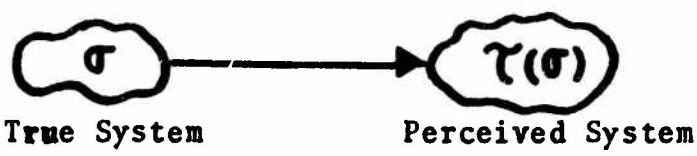
If the program is acceptable, the study can proceed with experimentation, analysis, and validation of model output. Validation, the process of testing the agreement between the behavior of the simulation model and the real world system, should be a continuing effort which is part of all simulation studies. Each study with the features unique to its particular objective may subject the model to a previously unencountered environment and test the model's validity under such conditions. The results of the analysis and validation can feedback vital information for model design.

The ten steps above outline the basic approach for a simulation study. They serve as a general guideline for the iterative procedures involved in building a computer simulation model.

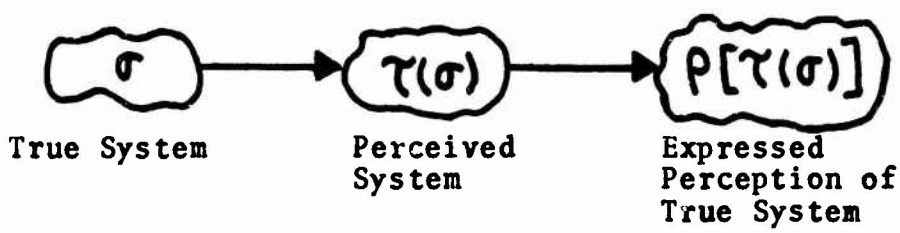
The Transformation Effect

Realism and fidelity are persistent problems for the analyst. The model must not only behave like the real world system but do it well. However one must be aware that throughout each step in the model building process, the true system has undergone a transformation.

The true system is perceived by the analyst and described initially as a natural language model. The mere description of the true system is a transformation from



the real world. Then the perception must be expressed as a symbolic or mathematical model.



Then finally the mathematical model is written as a computer program or code.

$$\sigma \rightarrow \tau(\sigma) \rightarrow \rho[\tau(\sigma)] \rightarrow \gamma\{\rho[\tau(\sigma)]\}$$

True System	Perceived System	Expressed Perception of True System	Coded Expression of True System

The true system has undergone three transformations from its initial state to the final representation as a computer program. Consequently, a simulation model is necessarily an approximate representation of reality - an abstraction of the real world.

Primary Areas of Concern

In building a probabilistic computer simulation model, the analyst has four primary areas of concern:

- Representing random behavior
- System representation, including event specification and the time flow mechanism
- Extract and communication of model results, and
- Statistical analysis of model output

The next three sections of this report discuss each area in considerable depth.

SECTION III

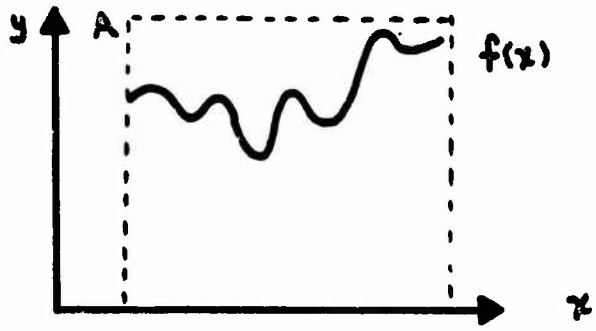
REPRESENTING RANDOM BEHAVIOR

Most applications of simulation models incorporate random behavior. For example, arrival and service times in a queuing model, target detection on a radarscope, or vehicle tracking error by a weapons controller require sampling from probability distributions.

In Section I, the terms Monte Carlo and stochastic sampling were introduced in the discussion of probabilistic models. Although some analysts use them interchangeably there is a clear distinction. Monte Carlo analysis is used to solve a deterministic analytic problem by converting it to a probabilistic problem having a similar mathematical formulation. Then random sampling techniques are used to estimate the solution to the original deterministic problem. On the other hand, stochastic sampling techniques in a computer simulation model are used on dynamic problems which have no closed form mathematical representation. The simulation model is required to represent the observed behavior of the real system and stochastic sampling is the means of providing certain aspects of that behavior.

The confusion between Monte Carlo methods and stochastic sampling in simulation arises because both

use random numbers. To illustrate the use of Monte Carlo methods, consider the area beneath the irregular curve below. The area is represented by the integral $\int_a^b f(X)$.



Suppose we construct the rectangle A around the curve and begin to generate random points within the rectangle. If the number of points (P_c) within the region bounded by the curve is compared to the total number of points generated (P), the probability of a point landing beneath the curve is found to be P_c / P. Then,

$$f(x) = Pc/P \times \text{Area of rectangle A.}$$

We have taken a deterministic problem which was difficult to evaluate and converted it to a probabilistic analog. The solution was then found using sampling techniques.

In simulation models, random variables are used to represent the behavior of those factors in the system whose real world counterpart fluctuates in an unpredictable but statistically describable way. For instance, the interarrival time (time between arrivals) of jobs at the base computer center could be exponentially distributed with a mean which can be estimated. Random variables

may also be used to represent a deterministic process when too detailed a model would be required. Adequate results could be obtained with a simpler model by aggregating the process into one statistically determined effect. A model of a man operating a radarscope would be too complex if all details of his functions and workload were incorporated. So in large scale campaign models an exponential distribution, which is a function of radar type, signal-to-noise ratio, and with a multiplier for operator attention, is used for determining target probability of detection.

Pseudorandom Number Generation

Random numbers are generated either by a device separate from the computer or by a deterministic calculation within the computer. Only the latter type of generation will be discussed here since it is the most frequently used. Pseudorandom numbers are defined as numbers produced by a non-random process which demonstrates sufficiently random behavior in a statistical sense.

For simulation purposes, random numbers generated by an acceptable method should be: uniformly distributed, stochastically independent, reproducible, non-repeating for any desired length, capable of high speed generation, and require little internal storage. Lehmer has been quoted as describing such numbers as "a vague notion

embodying the idea of a sequence in which each term is unpredictable to the uninitiated and whose digits pass a certain number of tests, traditional with statisticians and depends somewhat on the uses to which the sequence is to be put" [15,P.240].

Middle-of-the-Square-Method

The first deterministic procedure proposed to generate pseudorandom numbers was the middle-of-the-square method. Some initial value X_0 is specified along with a number of digits n in a base such as 7 or 10. The algorithm follows:

1. $X^1 \leftarrow X_0^2$
2. $X_1 \leftarrow n(X_1^1)$ Where n selects the central n digits of X_1^1
3. $X_0 \leftarrow X_1$
4. If more values are needed, go to 1; otherwise STOP.

The middle-of-the-square method can lead to very short sequences of non-repeating values. Let X_0 = 30 base 10 which is 42 base 7. The square $(42)^2$ is 2424 base 7 which is 42 base 7 [Reference 27].

STEP	ITERATION 1	ITERATION 2
1. $X_1^1 \leftarrow X_0^2$	2424	2424
2. $X_1 \leftarrow n(X_1^1)$	42	42
3. $X_0 \leftarrow X_1$	42	42

The sequence has length 1 thus totally unsuitable. However, with other choices of arguments very long sequences of values have been generated [Reference 15].

Linear Congruential Methods

In 1947, Lehmer proposed the linear congruential method and it swiftly replaced the middle-of-the-square. The linear congruential method is based on modular arithmetic from abstract algebra and number theory. The following discussion presumes a basic understanding of modular arithmetic. Three congruential techniques exist: multiplicative, mixed, and additive.

Multiplicative Congruential Method

The multiplicative congruential generator is of the form

$$X_{n+1} = a X_n \pmod{m} \quad n \geq 0$$

$$\text{where } X_0 > 0, \ a > 0.$$

The generator has a full period when the starting value X_0 (also frequently called the seed) is relatively prime to m and a is a primitive root modulo m. If m is prime, the period will be of length m-1. The above recursion means to multiply the last random number X_n by the constant a and take the result modulo m.

Any number sequence generated by a deterministic process with finite input will eventually repeat when the input is the same as at some previous stage. The number

of entries before such a repeat is called the period. In selecting a pseudorandom number generator, we desire a period as long as possible.

A natural choice for the modulus is the computer word length. If a number exceeds the word length, truncation results and the remaining digits are the residue modulo m. Table 1 lists the word lengths of various computers in bits where the sign bit is neglected. (Since overflow will probably cause the sign bit to change, some type of bit manipulation such as the FIELD function in UNIVAC's FORTRAN V should be used to zero the sign bit. The absolute value function IABS does not necessarily change the sign bit.)

Naylor et al [Reference 24] show that for a choice of modulus of the form 2^b, the maximum period attainable is 2^{b-2}. Jansson [Reference 17] demonstrates that the maximum period occurs when a=3 or 5 (mod 8) with X_0 odd. Finally, Greenberger's formula [Reference 15, p.238] for an approximation to serial correlation p is

$$p = \frac{1}{a} - \frac{6c}{am} (1 - c/m) + K$$

Values of a near $\sqrt{m}$ will yield small values of p. Suppose the computer word length is 2^{35} and we choose as the initial value X_0=1907. Since the square root of 2^{35} is near 2^{17}, a is chosen as $2^{17} + 3 = 131075$. The period will be 2^{33} which should be sufficiently large for most uses.

TABLE 1. COMPUTER WORD LENGTHS

COMPUTER MODEL	WORD LENGTH WITHOUT SIGN (Bits)
Burroughs B5500	2^{39}
CDC 1604, 3600, 3800	2^{47}
CDC 6000 series	2^{59}
PDP-6	2^{35}
GE 200 Series	2^{19}
GE 400 Series	2^{23}
GE 600 Series	2^{35}
Honeywell 800, 1800	2^{44}
IBM 7040	2^{35}
IBM 7090, 7094	2^{35}
IBM 360, 370	2^{31}
RCA Spectra 70	2^{31}
UNIVAC 1108	2^{35}

```
C     * * * MULTIPLICATIVE CONGRUENTIAL * * *
C     * * * RANDOM NUMBER GENERATOR     * * *
C
C     ISTART = THE INITIAL STARTING VALUE
C     IA     = THE MULTIPLIER
C     N      = NUMBER OF RANDOM NUMBERS TO BE GENERATED

      SUBROUTINE RMULT (ISTART, IA, N)
      COMMON RNUM(5000)
      I = ISTART
      DO 20 J = 1, N
      I = IA * I
      FLD(0,1,I) = 0
20    RNUM (J)   =  I/2.0 ** 35
      RETURN
      END
```

Figure 10. FORTRAN Subroutine for Multiplicative Congruential Generator.

Figure 10 shows the FORTRAN IV subroutine for a multiplicative congruential generator on a UNIVAC 1108 with word length 2^{35}.

Mixed Congruential Generator

The mixed congruential generator is of the form

$$x_{n+1} = a\, x_n + C \pmod{m}.$$

Jansson [17] shows that the generator for modulus 2^b has maximum period when $a=1 \pmod 4$ and $C = 1 \pmod 2$. Hull and Dobell [16] tested over a thousand different multipliers of the form $2^s + 1$ and $2^s + 3$. The generators were acceptable provided that the multiplier did not satisfy either $a = 1 \pmod{2^{13}}$ or $a < 30$ and $a \neq 2^{12} + 1$. The choice of C had small effect but more complicated values of C tended to improve performance.

Other empirical evidence has indicated that a equal to $2^7 + 1$ has been a satisfactory choice [17,24,30]. The initial value 1907 might be used with a equal to 129 and C equal to 1. The modulus can be conveniently chosen to be the computer word length. Figure 11 shows the FORTRAN IV subroutine.

Additive Congruential Generator

The additive generator is of the form

$$x_{n+1} = \sum_{j=0}^{k} x_{n-j} \pmod{m}$$

```
C  * * *  SUBROUTINE FOR MIXED CONGRUENTIAL
C  * * *  RANDOM NUMBER GENERATOR
C  ISTART - INITIAL STARTING VALUE
C  IA =   THE MULTIPLIER
C  IC =   THE CONSTANT TERM
C  N  =   THE NUMBER OF RANDOM NUMBERS TO BE GENERATED
          SUBROUTINE RMIX (ISTART, IA, IC, N)
          COMMON RNUM (5000)
          I = ISTART
          DO20 J = I, N
          I = IA * I + IC
          FLD (0,1,I) = 0
     20   RNUM (J) = I/2.0 ** 35
          RETURN
          END
```

Figure 11. FORTRAN Subroutine for Mixed Congruential Generator

A special case is the generalized Fibonacci generator which is defined as

$$x_{n+1} = x_n + x_{n-j} \pmod{m}, \quad j > 1.$$

Some j values are required initially before the generator can be used. Jansson [17] has examined the periods of Fibonacci generators for various moduli and values of j. Depending on the nature of the polynomial, i.e., primitive, irreducible or reducible, the period may be independent or dependent on initial values. Green, Smith, and Klem [14] also examined generators for j values between 2 and 16 and suggested j = 16 as the smallest value for acceptable pseudorandom numbers. They described the period as equal to $k_n 2^{b-1}$, where k_n is a constant (a table of k_n values was given). For j=16 and b=35, the period is 255×2^{34}.

Primitive Polynomials Modulo Two

R. C. Tausworthe [32] has developed a method for generating random binary bit patterns by the addition of primitive polynomials modulo 2. For example, using polynomials of degree 4, the recursion

$$A_{n+1} = A_{n-3} + A_{n-4} \pmod{2}$$

would be used. Suppose the starting values are 1000, 0110, 1101, and 0111. The sequence is

```
C * * * ADDITIVE RANDOM NUMBER GENERATOR ***
C
C N =    NUMBER OF RANDOM NUMBERS TO BE GENERATED
C K =    NUMBER OF INITIAL VALUES TO BE GENERATED BY THE
         MULTIPLICATIVE GENERATOR
C
C
         SUBROUTINE RADD (N,K)
         COMMON RNUM (5000)
         DIMENSION KORE (79)
         I1 = 0
         I2 = K-1
C        MULTIPLICATIVE GENERATOR FOR INITIAL K VALUES
         L = 1907
         DO 10 J =1, K
         L = 131075 * L
         FLD (0,1,L) = 0
   10    KORE (J) = L
         DO 20 J = 1, N
C        INCREMENT INDEXES OF VALUES TO BE ADDED
         I1 = I1 + 1
         I2 = I2 + 1
         IF (I1.GT.K) I1 = 1
         IF (I2.GT.K) I2 = 1
C        ADD TWO STORED VALUES
         LAST = KORE (I1) + KORE (I2)
         FLD (0,1,LAST) = 0
         KORE (I1) = LAST
C        TRANSFORM TO UNIFORM (0,1)
   20    RNUM (J) = LAST/2.0 **35
         RETURN
         END
```

Figure 12 FORTRAN Subroutine for Additive Random Number Generator

$$\left.\begin{array}{l}1000\\0110\\1001\\0111\end{array}\right\}\text{Starting Values}$$

0110
1011
1010
0001
1101

For primitive polynomials of degree m the period is 2^m-1.

Tests for Randomness

If the numbers generated by some pseudorandom technique are random, they must be uniformly distributed and stochastically independent. The following tests determine if there is statistical justification for rejection of a sequence of numbers as a random sample from a uniform distribution. A test may be based on either the property of independence, or uniformity, or possibly both. Each test is briefly explained and results of the test on each of the three types of congruential generators is discussed. The generators are used with the initial values suggested above for a 2^{35} bit word. The additive generator used 79 starting values which were initially produced by a multiplicative generator.

All of the following tests are also described in Knuth [19] and many are available in statistics packages associated with most computer library files. The particular programs used here are described in Reference 28.

Chi Square Test

The Chi Square Test is one of the earlier methods of statistical inference, originally proposed in 1900 by Karl Pearson.

Consider a finite number K of mutually disjoint sets (categories) $A, A_2, \ldots, A_k$. Let $P(A_i) = P_i$, the probability that the outcome of the random experiment is an element of A_i. The random experiment is to be repeated n independent times and e_i is the number of times the outcome is an element of set A_i, i.e., $e_i, e_2, \ldots, e_k = n-e_i-e_2\ldots-e_{k-1}$ are the frequencies with which the outcome is an element of $A_1, A_2, \ldots, A_k$. The joint pdf is multinomial with parameters $n, p_1, \ldots, P_{k-1}$.

If the null hypothesis is Ho: $P_i = P$, for all i

Ha: all alternatives

If H_o is true, the random variable $X^2 = \sum_{i=1}^{n} \frac{(ei-np)^2}{np}$

has approximately a Chi Square distribution with K-1 degrees of freedom with critical region $X^2 \geq X^2 d, K-1$. Pearson's Chi Square criterion is equivalent to the likelihood ratio test for large samples since -2 ln X (where X is the likelihood ratio) has an approximate Chi Square distribution with K-1 degrees freedom. The Chi Square test is a test of uniformity.

All generators were subjected to a Chi Square test for a sequence of 5000 numbers as a whole to test global behavior, then for groups of 1000 and 500 to test local behavior. The null hypothesis was taken as $H_0: X^2 < \chi^2_{\alpha, m-1}$ and the alternative $H_1: X^2_{m-1} \geq \chi^2_{\alpha, m-1}$. Table 2 below shows the Chi Square statistics for global behavior of all generators and for which the null hypothesis cannot be rejected at the .05 significance level.

TABLE 2 CHI SQUARE TEST ON 5000 NUMBERS

	Number of Intervals			
	n=10		n=25	
	X^2		X^2	
MULTIPLICATIVE	5.46	.5002	21.83	.5002
MIXED	10.03	.5012	27.70	.5012
ADDITIVE	14.61	.4987	35.03	.4987
TABLE Chi Sq	16.92		36.42	α=.05

Table 3 summarizes the results for the local behavior of each generator. The multiplicative and mixed generators pass all local tests but the additive generator, for a sample size of 500, does not pass one of the Chi Square tests. Such behavior should be noted but is not of itself due cause for rejection.

The means for the sample size of 500 were subjected to a Sign Test. Consider the means as a random sample from a binomial population with probability .5 of obtaining a value greater than the mean of a uniform distribution on

TABLE 3. CHI SQUARE TESTS OF LOCAL BEHAVIOR
(9 degrees of freedom)

MULTIPLICATIVE GENERATOR

SAMPLE SIZE:	500		1000	
TEST NO.	X^2	MEAN	X^2	MEAN
1	4.32	.5067	8.36	.5129
2	9.28	.5191	9.12	.5088
3	12.68	.5147	4.46	.5035
4	7.28	.5028	7.84	.4995
5	14.20	.5170	9.28	.4864
6	7.76	.4899		
7	6.64	.5009		
8	15.32	.4781		
9	8.88	.4856		
10	12.96	.4871		

MIXED GENERATOR

1	8.88	.4994	6.84	.5046
2	6.52	.5099	12.44	.4923
3	13.20	.4971	6.30	.4936
4	8.20	.4875	11.40	.5080
5	6.76	.4912	9.12	.5074
6	3.12	.4959		
7	13.80	.4869		
8	10.16	.5292		
9	10.48	.5093		
10	4.48	.5054		

ADDITIVE GENERATOR

1	4.76	.4928	10.64	.4977
2	17.40	.5026	10.32	.4997
3	7.08	.4997	11.02	.4964
4	9.52	.4997	8.00	.5097
5	13.52	.4860	8.20	.4901
6	8.52	.5069		
7	5.16	.5129		
8	8.52	.5065		
9	6.16	.4922		
10	5.36	.4880		

- -

TABLED CHI SQ 16.919 $\alpha = .05$

(0,1). The null hypothesis was taken as H_0: $p = .5$ and the alternative H_1: $p \neq .5$. The critical region is for values more than 8 or less than 2 above the mean. The null hypothesis cannot be rejected for any generator. The means could be from a uniform population.

Kolmogorov-Smirnov Test

The second test of uniformity is the Kolmogorov-Smirnov Goodness of Fit Test (KS test). The test is based on the difference between the theoretical cumulative distribution function and the empirical distribution function which is derived from the sampled values. The KS test may be used when the theoretical cumulative distribution function is continuous. For a detailed discussion the reader is referred to Knuth [19,p.41] or a statistics text.

The test was first applied to 5000 numbers in 100 blocks of 50 each and then again applied to the 100 succeeding results. The null hypothesis is that the numbers are uniformly distributed. The critical region is $K > K_{n,\alpha/2}$ and $K < K_{n,1-\alpha/2}$, where K_n, $n = 100$ are tabled values [19,p.44]. Global behavior is described by the block statistics given in Table 4. The null hypothesis cannot be rejected for any generator. For local behavior in samples of size 50, all generators failed some tests.

The multiplicative failed thirteen out of 100, and the mixed and additive nine each.

TABLE 4 KS GOODNESS OF FIT TEST
on 5000 numbers in 100 blocks of 50 each

GENERATOR:

MULTIPLICATIVE	1.0651	.5020	.7287	.1948
MIXED	.7899	.3513	.7683	.1999
ADDITIVE	.6054	.1591	1.0791	.8621

TABLED K VALUE	97.5%	2.5%
	.0961	1.3879

Runs Above and Below the Mean

A run is a succession of identical symbols which is followed and preceded by different symbols. For a sequence of random numbers $r_1, r_2, \ldots, r_n$, there corresponds a sequence $s_1, s_2, \ldots, s_n$ where s_i is the letter "a" if r_1 is greater than .5 and "b" if less than .5. The resulting sequence might look like a a b a b b a...etc.

Assuming independence between the random numbers, the probability and expected numbers of runs of various lengths can be calculated and a Chi Square statistic formed. The null hypothesis is that the numbers are random and the critical region is $X^2 \geq \chi^2_{.05,r}$.

TABLE 5 RUNS ABOVE AND BELOW MEAN

GENERATOR	CHI SQUARE STATISTIC (8 deg. freedom)	
MULTIPLICATIVE	8.423	
MIXED	6.900	
ADDITIVE	11.208	
TABLED CHI SQ	15.507	$\alpha = .05$

The null hypothesis cannot be rejected for any generator.

Correlation Coefficient

If a set of numbers is random, a given number should not depend on its predecessors. The correlation for any given lag L should be zero. Since we do not have truly random numbers, the magnitude of any correlation should be small and have equally distributed positive and negative signs.

A correlation test for lags from 1 to 15 was made on a set of 4985 numbers from each generator. The results are displayed in Table 6. Of special interest is the correlation of lag 1 (serial correlation). Anderson [1] has shown that the serial correlation for large samples ($N > 75$) is approximately normally distributed with mean $-1/(N-1)$ and variance $(N-2)/(N-1)^2$. The single tail significance points are $\frac{-1 \pm 1.645\sqrt{N-2}}{N-1}$, i.e., for $N=5000$, $-0.0235 < p_1 < +0.0231$. No generator is rejected on the basis of the serial correlation.

TABLE 6

CORRELATION TEST OF 4985 NUMBERS

LAG	MULTIPLICATIVE	MIXED	ADDITIVE	RANGE
1	-.0101	.0218	.0024	-.0176
2	.0053	.0043	-.0008	-.0113
3	.0028	.0012	-.0021	.0063
4	-.0027	.0076	.0130	.0155
5	-.0198	.0143	.0044	.0016
6	-.0127	-.00004	-.0226	.0065
7	.0089	-.0030	.0053	-.0079
8	-.0003	-.0177	-.0105	-.0162
9	.0067	.0280	-.0040	.0042
10	.0018	.0191	-.0023	.0007
11	-.0044	.0091	-.0073	-.0083
12	.0033	-.0054	.0023	-.0004
13	.0149	.0116	-.0055	-.0030
14	.0001	-.0061	.0130	-.0161
15	.0093	-.0011	-.0114	.0295

A sign test can be applied to the other correlations of lag L. Consider the signs of the correlations as a random sample from a binomial population with probability .5 of obtaining a positive correlation The null hypothesis is H_0: $p = .5$ and the alternative H_1: $p \neq .5$. For sample size N=15 and $\alpha = .05$, the null hypothesis is rejected if the number of successes (positive signs) exceeds 12 or less than 3. The null hypothesis cannot be rejected for any generator.

Runs Up

A sample of 5000 numbers from each generator was subjected to Knuth's Runs Up test [19 P.60]. The sample is examined for lengths of sequences which are increasing and a statistic computed which has a Chi Square distribution. The critical region is $X^2 > \chi^2_{.05,\gamma}$.

TABLE 7. RUNS UP

GENERATOR	CHI SQUARE STATISTIC (6 deg. freedom)	
MULTIPLICATIVE	6.678	
MIXED	2.998	
ADDITIVE	4.210	
TABLED CHI SQ	12.59	α = .05

The null hypothesis cannot be rejected for any generator.

Poker Test

The Poker Test is a frequency test for combinations of distinct values in a set of five. The categories are:

5 different	=	all different
4 different	=	one pair
3 different	=	two pairs or three of a kind
2 different	=	full house or four of a kind
1 different	=	five of a kind

Counts in each category are compared to the expected numbers and a Chi Square test is made.

The Poker Test was applied to 1000 groups of five numbers to test independence. Table 8 below shows the Chi Square statistics. The null hypothesis is that the numbers are independently distributed against the alternative that they are not. The critical region is $X^2 \geq \chi^2_{.05,\gamma}$.

TABLE 8. POKER TEST

GENERATOR	CHI SQUARE STATISTIC (3 deg. freedom)	
MULTIPLICATIVE	3.439	
MIXED	.937	
ADDITIVE	.432	
TABLED CHI SQ	7.815	$\alpha = .05$

The null hypothesis cannot be rejected for any generator.

Coupon Collector Test

The Coupon Collector Test was applied to 5000 numbers for a set of integers 0 to 4. The lengths of sequences to complete the set of integers is observed and a Chi Square statistic calculated. The null hypothesis is that the numbers are independently distributed. The critical region is $X^2 \geq \chi^2_{.05,\ 11}$.

TABLE 9 COUPON COLLECTOR TEST

GENERATOR	CHI SQUARE STATISTIC (11 deg. freedom)	
MULTIPLICATIVE	10.123	
MIXED	20.509	
ADDITIVE	9.656	
TABLED CHI SQ	19.675	$\alpha = .05$

The mixed generator failed the test, but the null hypothesis was not rejected for the other generators.

Permutation Test

If the numbers are truly random, there should be an equal probability of any permutation of digits. The re-

sults of the test are given in Table 10. The null hypothesis is that the numbers are independently distributed. The critical region is $X^2 \geq \chi^2_{.05, \nu}$.

TABLE 10. PERMUTATION TEST

GENERATOR	CHI SQUARE STATISTIC (119 deg. freedom)
MULTIPLICATIVE	122.48
MIXED	120.08
ADDITIVE	103.28
TABLED CHI SQ	157.7

The null hypothesis is not rejected for any generator.

Spectral Test

The spectral test [19] was applied to the values produced by the multiplicative and mixed congruential generators. It is a test of independence among n-tuples of numbers produced by linear congruential generators and successful performance depends on the choice of the multiplier and modulus. The results for values of n = 2,3,4 are given in Table 11.

TABLE 11 SPECTRAL TEST

GENERATOR	C(2)	C(3)	C(4)
MULTIPLICATIVE	1.57	$.125 \times 10^{-5}$	$.193 \times 10^{-5}$
MIXED	$.152 \times 10^{-5}$	$.262 \times 10^{-3}$	$.398 \times 10^{-1}$
PASSING	.1	.1	.1
UPPER BOUND	3.63	5.90	9.86

Knuth [19] states that a generator passes the spectral test if $C(n) > 0.1$ for $n = 2$, 3 and 4, and $C(n) > 1$ is considered very good. Based on the criteria neither generator passes the spectral test. The multiplicative generator has a good value of C(2) indicating independence of consecutive pairs, but the values of C(3) and C(4) are too low as are all the values for the mixed generator. The test results suggest that to improve independence a larger multiplier should be used. Knuth [19] has found that a multiplicative congruential generator with a multiplier of 5^{15} has very good results on the spectral test.

Serial Test

Pairs of successive random numbers should be distributed uniformly and independently. In the serial test, the number of times the pair

$$(r_{2j}, r_{2j+1}) = (a, b) \text{ for } \begin{array}{l} 0 < j < n \\ 0 < a, b < d \end{array}$$

occurs is counted. A Chi Square test is applied to these d^2 categories with a $1/d^2$ probability for each category. The expected number in each category is n/d^2. The serial test was applied to a sequence of 5000 numbers for pairs whose digits are between 0 and 9, and a Chi Square statistic is computed. The null hypothesis is that $X^2 < \chi^2_{.05, \nu}$. The results are displayed in Table 12.

TABLE 12
SERIAL TEST

GENERATOR	CHI SQUARE STATISTIC (99 deg. freedom)
MULTIPLICATIVE	91.46
MIXED	125.86
ADDITIVE	83.30
RANGE	120.98
TABLED CHI SQ	134.51

The null hypothesis cannot be rejected for any generator.

Gap Test

The Gap Test incorporates both independence and uniformity in examining the lengths of subsequences (gaps) separating values in a specific range. Let α, β be such that $0 \leq \alpha < \beta \leq 1$. Consider lengths of subsequences separating specific values X_{j-1}, X_j, X_{j+1}, ..., X_{j+r}, so that X_{j-1}, X_{j+r} belong to

$$\{\alpha \leq X_{j-1}, X_{j+r} \leq \beta\} \cap [\{X_{j+k} < \alpha\} \cup \{X_{j+k} > \beta\}]$$

where $k=0, 1..., r-1$

The r+2 numbers give a gap of length r. Suppose the interval is .4 to .5. Then the subsequence

X_{-1}	X_0	X_1	X_2	X_3	X_4	X_5
.41	.32	.22	.78	.61	.10	.48

$\underbrace{\qquad\qquad\qquad\qquad\qquad}_{\text{not in the interval}}$

represents a gap of length 5.

The distribution of gap lengths is geometric

$$h(j) = \begin{cases} p(1-p)^{j-1} & 0 < j \leq n \\ 0 & \text{otherwise} \end{cases}$$

where $p = \beta - \alpha$, the probability that

$$\alpha \leq x_i \leq \beta$$

A Chi Square test is applied to the t+1 values of counts of length r, $0 \leq r \leq t$. The gap test has aspects of uniformity and independence: i.e.,"uniformity" (or equally likely) because the same probability p is assigned to the occurrence of X_i in (α, β), and "independence" because independent geometric trials are assumed. For each generator from among 5000 numbers, 500 gaps were requested for the intervals .4 to .5 and .5 to .6. Table 13 presents the results. The null hypothesis is that $X^2 < \chi^2_{.05,\gamma}$ The mixed generator failed both gap tests while the null hypothesis cannot be rejected for the other generators.

TABLE 13
GAP TESTS

500 gaps on the intervals

	.4 to .5		.5 to .6	
GENERATOR	DEG FREEDOM	X^2	DEG FREEDOM	
MULTIPLICATIVE	20	25.505	20	29.506
MIXED	19	35.969	[illegible]	33.321
ADDITIVE	20	21.036	20	28.438
TABLED CHI SQ (19 Deg f)		30.144		$\alpha = .05$
TABLED CHI SQ (20 Deg f)		31.410		

Sensitivity to Initial Values

The multiplicative and mixed congruential generators were tested with three different starting values with respect to the Chi Square, Kolomogorov-Smirnov, and Runs Above and Below the Mean tests. The Chi Square was a

test of global behavior on all 5000 numbers. The number corresponding to KS(local) is a count of the number of failures of the KS test in 100 repetitions of sample size 50. Table 14 summarizes the results.

TABLE 14 INITIAL VALUE TEST

	starting value		
MULTIPLICATIVE	1	203	1907
MEAN	.4976	.5016	.5002
CHI SQUARE	N	N	N
KS (global)	N	R	N
KS (local)	12	11	9
RUNS	N	N	N

** NOTE: N = not reject null hypothesis; R = reject

The initial value does have some effect on the properties of the sequence of numbers generated. A very small prime, say one, produces detrimental effects, such as failure to pass the global KS test for a uniform distribution. Larger initial values moved the mean closer to .50.

The additive generator was tested with initial values generated by a multiplicative with its initial value equal to one. The additive generator failed the KS test when applied to the block statistics. The additive type of all generators should be most sensitive to starting values.

Generation Time and Storage

All pseudorandom number generator subprograms were timed on a UNIVAC 1108 for the generation of 5000 numbers.

The mean times for five runs each are displayed in Table 15. The fastest are the multiplicative and mixed while the additive is substantially slower.

TABLE 15. MEAN GENERATION TIME

GENERATOR	SECONDS
MULTIPLICATIVE	.07
MIXED	.07
ADDITIVE (79)	.23
ADDITIVE (16)	.22

There is no significant difference between storage requirements of the multiplicative and mixed generators. Both depend only on the last number generated and have no requirement for additional values; also the programs themselves are the same length.

However, the additive generator requires at least 16 initial values to start, which also must be updated. For example, the additive congruential generator used in this report had 79 initial values. Such requirements could be a hindrance if storage is a severe restriction as in a mini-computer.

Conclusion Based on the Statistical Results

The multiplicative congruential generator produces the most favorable statistical results, has the fastest generation time, and has the least storage requirements of all generators considered. The multiplicative did

have a higher number of failures of the Kolmogorov-Smirnov test for local uniform distribution behavior than the other generators. The difference did not appear significant. Unsatisfactory local behavior is possible even in a sequence with good global behavior [15]. The generator did fail the spectral test overall, but independence for consecutive pairs was good.

The mixed congruential generator has a theoretical advantage of longer period and an easier basic theorem to satisfy. However, its statistical behavior is not as good. In particular, the mixed generator did not pass the spectral, coupon, and gap tests. Time and storage requirements are equivalent to those of the multiplicative. The mixed generator as formulated for this report is not recommended for general use. The failure of the spectral test indicates a conspicuous lack of independence for consecutive pairs, triples, and quadruples.

The additive type is markedly slower but has favorable statistical behavior. It appears that these generators are sensitive to initial starting values. Storage requirements are the major disadvantage for additive generators with acceptable random numbers.

A summary of results for selected tests on all generators is given in Table 16. Table 17 also presents the probabilities $P(\chi^2 \leq x^2)$ for a better

comparison within a given set of test results. The serial correlation is the actual computed value.

Among the three congruential generators, the multiplicative has the advantage in uniformly distributed numbers. For independence the results are close between the additive and multiplicative. Serial correlation is lowest for the additive, possibly giving it the edge in considerations of independence.

TABLE 16 SUMMARY OF STATISTICAL TESTS

	MULTIPLICATIVE	MIXED	ADDITIVE
Uniformly Distrib.			
CHI SQUARE	N	N	N
KS	N	N	N
Independence			
POKER	N	N	N
COUPON	N	R	N
PERMUTATION	N	N	N
RUNS ABOVE & BELOW MEAN	N	N	N
SERIAL CORRELATION	N	N	N
RUNS UP	N	N	N
SPECTRAL	R	R	-
SERIAL	N	N	N
GAP	N	R	N
TESTS	-	-	-
REJECTED	1	3	0

Note: N = Null hypothesis not rejected; R= rejected.

TABLE 17 COMPARISONS OF TESTS

	MULTIPLICATIVE	MIXED	ADDITIVE	RANGE
Uniformly Distrib				
CHI SQUARE	20	70	90*	86
POKER	67*	20	8	89
COUPON	50*	96x	43	33
Independence				
RUNS ABOVE & BELOW MEAN	60	45	80*	65
RUNS UP	63*	20	37	49
SERIAL CORRELATION	-.0101	.0218	.0024*	-.0176
GAP (.4 to .5)	80*	98x	60	50

Note: * The highest non-failing probability for that test (disregarding RANGE)
x Failed at .05 significance level

A Recommended Pseudorandom Number Generator

The above results indicate that a multiplicative congruential generator produces the most favorable results. For computers having a word length of 2^{35}, the generator with initial value 1907, and multiplier 5^{15} demonstrates very good statistical behavior. The FORTRAN IV subroutine is Figure 13.

Random Variates

The pseudorandom number generators produced numbers between zero and m-1 in modulo m. A keen observer would have noticed that in the FORTRAN subroutines the random number was divided by the word length 2^{35} to produce a

```
C * * * PSEUDORANDOM NUMBER GENERATOR * * *
C
C       FUNCTION RANG(DUMMY)
        I = IA * I
        FLD ( 0,1,I) = 0
        RANG = I/2.0 ** 35
        RETURN
C
C       INITIALIZATION OF MULTIPLIER AND INITIAL VALUE
C
        ENTRY RAN (INITAL)
        I = INITAL
        IA = 5 ** 15
        RETURN
        END
```

Figure 13. FORTRAN Subroutine for Pseudorandom Number Generator

uniform random variable on (0,1). To generate other random variates we must consider some transformation techniques from the uniform to other distributions.

Transformations are broadly categorized as table look up routines or mathematical techniques. Many table look ups are based on an inverse transformation. Suppose F(X) is the cumulative distribution function corresponding to a desired probability density function f(X). From probability theory we know that the probability distribution for the cumulative distribution function F(X) is uniform on (0,1). Therefore we can use the uniformly distributed random numbers to generate the distribution of F(X) regardless of the functional form of the distribution. It follows then that for r a random variable on (0,1)

$$r = F(X)$$

and

$$X = F^{-1}(r)$$

$F^{-1}(r)$ is the inverse of the original cumulative distribution. In the computer, an array would represent the functional values of F(X) and a subroutine would interpolate to determine the value of X corresponding to the F(X) produced by the pseudorandom number generator.

Mathematical Techniques

The inverse transformation can also be applied mathematically to simple distributions such as the exponential.

Exponential

The probability density function for the exponential with mean θ is

$$f(x) = \begin{cases} \frac{1}{\theta} e^{-\frac{x}{\theta}} & x > 0,\ \theta > 0 \\ 0 & \text{otherwise} \end{cases}$$

Then,

$$F(Z) = \int_0^z f(x)\,dx = \int_0^z \frac{1}{\theta} e^{-x/\theta}\,dx$$

$$F(Z) = 1 - e^{-z/\theta}$$

$$-\frac{z}{\theta} = \ln[1-F(Z)]$$

$$Z = -\theta \ln(1-r) \quad \text{let } r = F(z)$$

or using symmetry,

$Z = -\theta \ln r$, where r is the uniform random number.

```
C   * * *  FORTRAN FUNCTION FOR EXPONENTIAL * * *
C   * * *  RANDOM VARIATES
C
           FUNCTION EXPRV (XMEAN)
           EXPRV = - XMEAN * ALOG (RANG(RANUM))
           RETURN
           END
```

Figure 14. A FORTRAN Function for Exponential Random Variates.

Geometric

The probability density function for the geometric distribution is $f(x) = pq^{x-1}$, $X = 1, 2, \ldots$

where $p = 1-q$

Then $F(X) = \sum_{y=1}^{x} pq^{y-1}$

$$1-F(X) = \sum_{y=X+1}^{\infty} (pq)^{y-1} = \sum_{y=X+1}^{\infty} (1-q)\, q^{y-1}$$

$$= \sum_{y=X+1}^{\infty} (q^{y-1} - q^{y}) = q^{X}$$

$$\frac{1-F(X)}{q} = q^{X-1}$$

$$r = q^{X-1}$$

$$X-1 = \log_q r$$

$$X-1 = \frac{\ln r}{\ln q}$$

$$X = \frac{\ln r}{\ln q} + 1$$

where r is the uniform random number and q is the probability of failure on an individual trial.

```
C * * * FORTRAN FUNCTION FOR * * *
C * * * GEOMETRIC RANDOM VARIATES * * *
C
      FUNCTION GEOM (Q)
      GEOM = ALOG (RANG(RNUM))/ALOG(Q)+1
      RETURN
      END
```

Figure 15 A FORTRAN Function for Geometric Random Variates

Negative Binomial

A negative binomial distribution can be described as a series of Bernoulli trials which are repeated until K successes have occurred. The pdf is

$$f(x) = \binom{k+x-1}{x} p^k q^x \qquad X=0, 1, 2...$$

(where probability of success is p) which is the sum of K geometric variates.

$$X = \sum_{i=1}^{k} \ln r_i$$

$$X = \frac{\ln \left(\prod^{k} ri\right)}{\ln q}$$

ri is random number i=1,..., k
q is probability of failure

```
C  * * *  FORTRAN FUNCTION FOR NEGATIVE  * * *
C  * * *  BINOMIAL RANDOM VARIATES * * *
C

          FUNCTION NEGBN (K, Q)
          PROD = 1.0
          DO 10  I = 1, K
     10   PROD = PROD * RANG (RANUM)
          NEGBN = ALOG (PROD)/ALOG (Q)
          RETURN
          END
```

Figure 16 . A FORTRAN function for Negative Binomial random variates.

Uniform on (a, b)

The pseudorandom number generator produces uniform random variates on (0, 1). But frequently uniform random variates are required on other intervals, say (a, b). The uniform density function is

$$f(x) = \begin{cases} \frac{1}{b-a} & a < x < b \\ 0 & \text{Otherwise} \end{cases}$$

The cumulative distribution function is

$$F(x) = \int_a^x \frac{1}{b-a} \, dt = \frac{x-a}{b-a}$$

Let $F(x) = r$

so $\frac{x-a}{b-a} = r$

and $x = a + (b - a) r$

where r is a uniform random variate m (0, 1)

```
C  * * *  FORTRAN FUNCTION FOR UNIFORM  * * *
C  * * *  RANDOM VARIATES ON (A, B)  * * *
C
          FUNCTION UNIFM (A, B)
          UNIFM  = A + (B - A) * RANG (RANUM)
          RETURN
          END
```

Figure 17 A FORTRAN Function for Uniform Random Variates on (a, b)

Rejection Method

The rejection method can be applied to any probability distribution for which upper and lower bounds can be placed on the range of values. For example, consider the binomial distribution.

Binomial

The binomial can be described as X successes in n trials with probability P of success:

$$f(x) = \binom{n}{x} p^x q^{n-x} \quad X = 0, 1, 2, \ldots n$$

The following algorithm effectively reproduces the n bernoulli trials and rejects, i.e., does not count, the failures.

1. I = 0

   KOUNT = 0

2. I = I + 1

3. IF I > n , STOP

4. GENERATE UNIFORM RANDOM NUMBER RI

5. IF RI ≤ P, KOUNT = KOUNT + 1

6. GO TO 2

The value of KOUNT is the number of successes in n trials.

```
C  * * *  FORTRAN FUNCTION FOR BINOMIAL * * *
C  * * *  RANDOM VARIATES               * * *
C
          FUNCTION IBINOM (N, P)
          I = 0
          IBINOM = 0
          I = I = 1
          DO 10 I = 1, N
          R = RANG (RNUM)
          IF (R. LE. P)  I BINOM = IBINOM + 1
     10   CONTINUE
          RETURN
          END
```

Figure 18 A FORTRAN Function for Binomial Random Variates.

Poisson

Generating Poisson random variates depends on the known relationship between the exponential and Poisson

distribution. The Pdf for the Poisson is

$$f(x) = \frac{e^{-\lambda} \lambda^{x}}{x!} \qquad \begin{array}{l} x = 0, 1, 2 \ldots \\ \lambda > 0 \end{array}$$

If the time interval between events has an exponential distribution with mean $1/\lambda$, then the number of events x occurring during a unit time interval has a Poisson distribution with mean λ. Thus, the method is to generate exponential random variates $t_1, t_2, \ldots$ where $t_i = -\ln r_i$ and accumulates them until $\sum_{i=1}^{x} t_i \leq \lambda < \sum_{i=1}^{x+1} t_i$ the sum exceeds λ. Then X, the number of exponential variates is the desired Poisson random variate.

```
C   * * *  FORTRAN FUNCTION FOR POISSON * * *
C   * * *  RANDOM VARIATES              * * *
C
          FUNCTION  IPOISSN (MEAN)
          REAL MEAN
          SUM = 0.
          IPOISSN = 0
        3 T = - ALOG (RANG(RANUM))
          SUM = SUM + T
          IF(SUM - MEAN) 10, 6, 6
        6 IPOISSN = IPOSSN + 1
          GO TO 3
       10 RETURN
          END
```

Figure 19 A FORTRAN Function for Poisson Random Variates.

Gamma

The Gamma distribution is also related to the exponential. The pdf for the gamma is

$$f(X) = \begin{cases} \dfrac{1}{B^{\alpha}\Gamma(\alpha)} X^{\alpha-1} e^{-X/B} & X > 0,\ \alpha > 0,\ B > 0 \\ 0 & \text{otherwise} \end{cases}$$

The special case where α is an integer will be considered. For non-integral values of α see reference 41. The sum of n exponentials with mean θ is distributed as a gamma with $\alpha = n$ and $B = 0$. Thus, $X = -\sum_{i=1}^{n} \theta \ln r_i$

which is equivalent to (and possibly faster for computer execution)

$$X = -\theta \ln \prod_{i=1}^{n} r_i .$$

```
C * * * FORTRAN FUNCTION FOR GAMMA * * *
C * * * RANDOM VARIATES              * * *
C
      FUNCTION GAMMA (IALPHA, BETA)
      PROD= 0.
      DO 10 I = 1, IALPHA
   10 PROD = PROD * RANG (RANUM)
      GAMMA = - BETA * ALOG (PROD)
      RETURN
      END
```

Figure 20 A FORTRAN Function for Gamma Random Variates.

Normal

Since the normal distribution is so frequently used, several techniques have been developed to generate normal random variates. Muller [22] discusses six approaches in current use. Three methods will be discussed below.

The most common approach is table look up. In GPSS language, for example, values of the cumulative distribution

```
C  * * *  FORTRAN FUNCTION FOR NORMAL * * *
C  * * *  RANDOM VARIATES              * * *
C
         FUNCTION XNORMAL (MEAN, STD)
         REAL MEAN
         SUM = 0.
         DO 10 I = 1, 12
   10    SUM = SUM + RANG (RANUM)
         XNORMAL = STD * (SUM - 6.) + MEAN
         RETURN
         END
```

Figure 21 A FORTRAN Function for Normal Random Variates

function for the particular normal distribution are stipulated as a table which is then searched to determine the random variate.

Box and Muller [4] have developed a direct approach to generate a pair of random variates from the same normal distribution. Let r_1 and r_2 be uniformly distributed independent random variables on (0, 1). Then,

$$X_1 = (-2 \ln r_1)^{1/2} \cos 2 \quad r_2$$
$$X_2 = (-2 \ln r_1)^{1/2} \sin 2 \quad r_2$$

Where (X_1, X_2) will be a pair of independent random variables whose joint pdf is a standard normal distribution.

The last technique is based on the Central Limit Theorem which states that the mean of identical uniform independent random variables is asymptotically normally distributed. Consider a set r_i of K uniformly distributed independent variables on (0, 1). Then

$$X = \frac{\sum r_i - \frac{k}{2}}{\sqrt{k/12}}$$

is normally distributed with mean 0 and variance 1. Thus, to generate a normal variate with mean μ and standard deviation σ

$$X = \sigma\left(\frac{12}{K}\right)^{1/2} \left(\sum_{i=1}^{k} r_i - \frac{k}{2} \right) + \mu$$

The values of X are reliable within three standard deviations. The value of K should be no less than 10, and 12 is frequently used.

SECTION IV

SYSTEM REPRESENTATION

A model is a representation of a real world system which the analyst uses for prediction or control. The model should reasonably approximate the real world system and incorporate the important features of the real system but not become so complex that it is impossible to understand or manipulate.

In this section we will look in more detail at model building and the components of the model which represent the real system. It is important to realize that model building is as much an art as a science and that success depends on the analyst's experience as well as technique and considerable luck.

Model Structure

A basic problem is how to describe the system using a simulation model. For our purposes, we will define four elements which comprise the model structure: entities, attributes, functional relationships, and a time flow mechanism.

Entities are the objects by which a system can be defined, i.e., the components of the system represented in the model. They are further divided into permanent and temporary entities. Permanent entities are entities

whose presence in the model are unaffected by time. Temporary entities are entities whose presence in the model is affected by time.

Using set notation, the entities are described as

$$E = \left\{ p_1, p_2, \ldots, p_n, t_1, t_2, \ldots t_m \right\}.$$

The permanent entities form a subset $P = \left\{ p_1, p_2, \ldots, p_n \right\}$ $P \subseteq E$ and similarly, the temporary entities form a subset $T = \left\{ t_1, t_2, \ldots, t_m \right\}$, $T \subseteq E$. As a simple example consider a queuing system consisting of a single server and single

waiting line. The permanent entity is the single server--he is always there whether customers are there or not. The temporary entities are the customers which arrive, wait, are served, and then leave the system.

The next element of the model is the set of attributes. Attributes are parameters or variables associated with the entities. Attributes may be further classified as exogenous, endogenous, or status variables. Exogenous variables are the independent or input variables which are assumed to be predetermined and given independent of the system modeled. They have values which affect, but are unaffected by the system. Endogenous variables are the dependent variables of the system. Their values are

determined by the other variables in the system.

The attribute set is

$$A = \left\{ v(e_j) : e_j \in E \right\}$$

where e_j is some permanent or temporary entity in the model.

Functional relationships are the expressed relationships among permanent and temporary entities. They describe the interaction of the variables and attributes of the model.

The functional relationship set is

$$F = \left\{ f_i(p_j, t_k) : p_j \in P, t_k \in T \right\}$$

for some i.

Returning to the queuing example, the exogenous variables are the interarrival time of the customers and the interservice time of the server. The status variables include the amount of time a customer spends waiting and the amount of time the server is idle. An endogenous variable is the total time a customer is in the system. Parameters include the expected interarrival time and expected interservice time. The probability density functions for the interarrival time and interservice times express the functional relationship between the entities.

The first three sets -- entities, attributes, and functional relationships -- comprise the static

description of the system. The static model (M) is then the union of the three sets

$$M = E \cup A \cup F$$

The state of the model at time τ (S^{τ}) is the value assigned at that time to the variables associated with all entities in the model i.e.,

$$S^{\tau} = \left\{ s^{\tau}(e_j) : \text{for all } j,\ e_j \text{ E at time } \tau \right\}$$

The state of entity e_j may also be written s_j when the time t is obvious from the context or usage. The state space is the set of all states of the model

$$S = \left\{ S^{\tau} : \quad \tau \in \mathcal{T} \right\}$$

where $\mathcal{T}$ is the time horizon of the simulation.

The principle type of simulation discussed in this report is discrete event simulation, i.e. , simulation of a system which has discrete units such as customers or machine failures flowing through a sequence of stations or locations. The basic simulation structure is illustrated in Figure 22.

A primary task in simulation is the method of representing processes, which are activities that proceed over time. The initiation, alteration, or conclusion of an activity is called an <u>event</u>. Since events are associated with the system's entities, the state of the system changes if and only if an event occurs. The significance of all this is that processes are not modeled

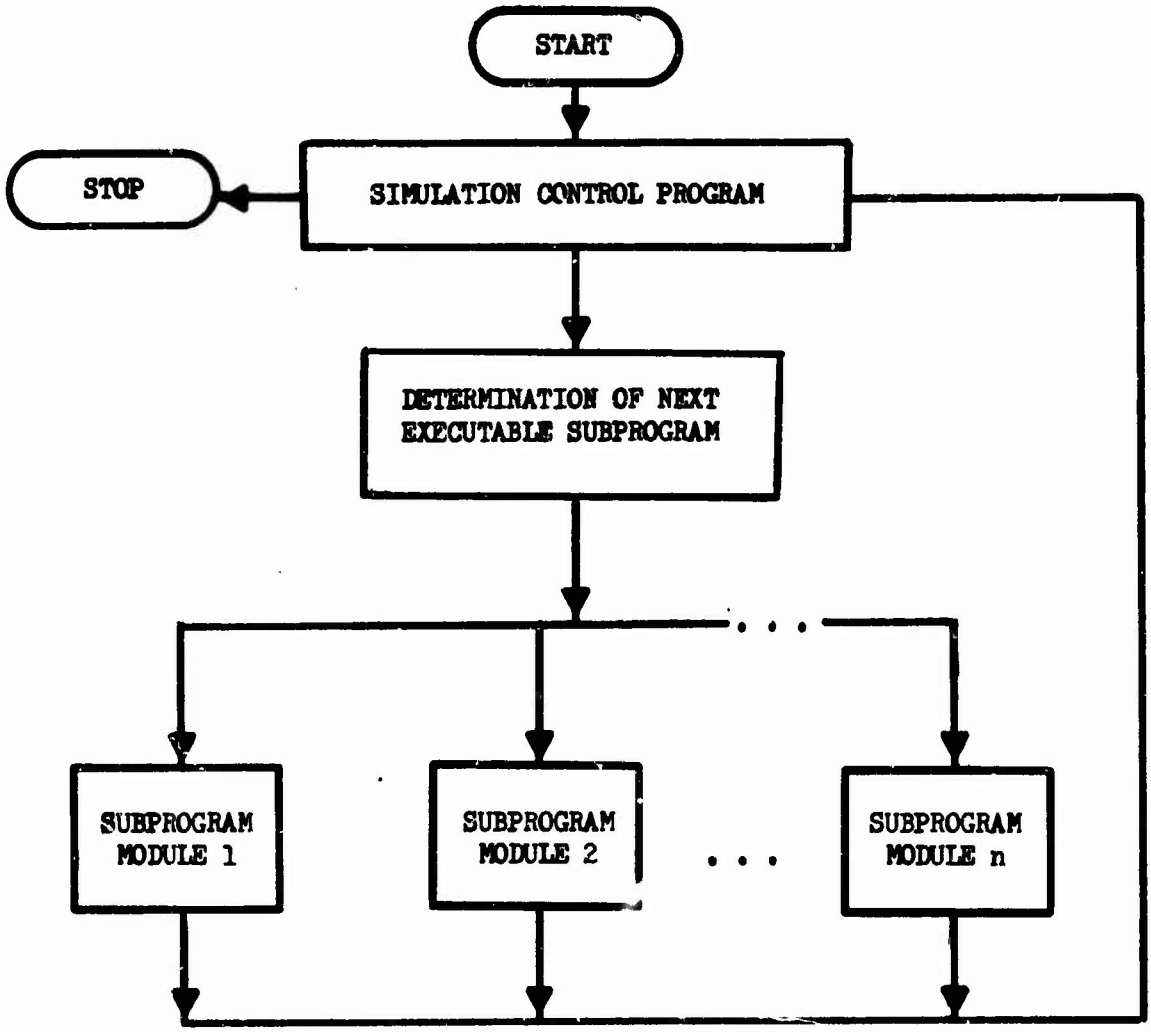


Figure 22. Basic Structure in Discrete Event Simulations (After Reference 18)

explicitly but rather are represented by the modeling of the events which affect the status of the process.

In building a simulation model the analyst must construct an event list which contains descriptions of events computed to occur at some future time in the simulation. The list is updated as other events are added and as the system state changes. The event time prediction routine or time flow mechanism is the heart of the simulation model since it determines the system's dynamics.

Time Flow Mechanisms

Time flow mechanisms, timing routines, event scheduling procedures, or simulation executive routines are widely used terms which will be treated as synonymous. Regardless of the name, its function is to advance simulation time and select a subprogram for execution that performs a specified activity. Historically, time flow mechanisms have been categorized as belonging to one of two general methods: fixed time increment or next event increment. We will now compare the logic involved in each formulation.

Fixed Time Incrementing

Consider a sequence of events $E_1, E_2, \ldots, E_n$ as they actually occurred with real world times $r_1, r_2, \ldots, r_n$.

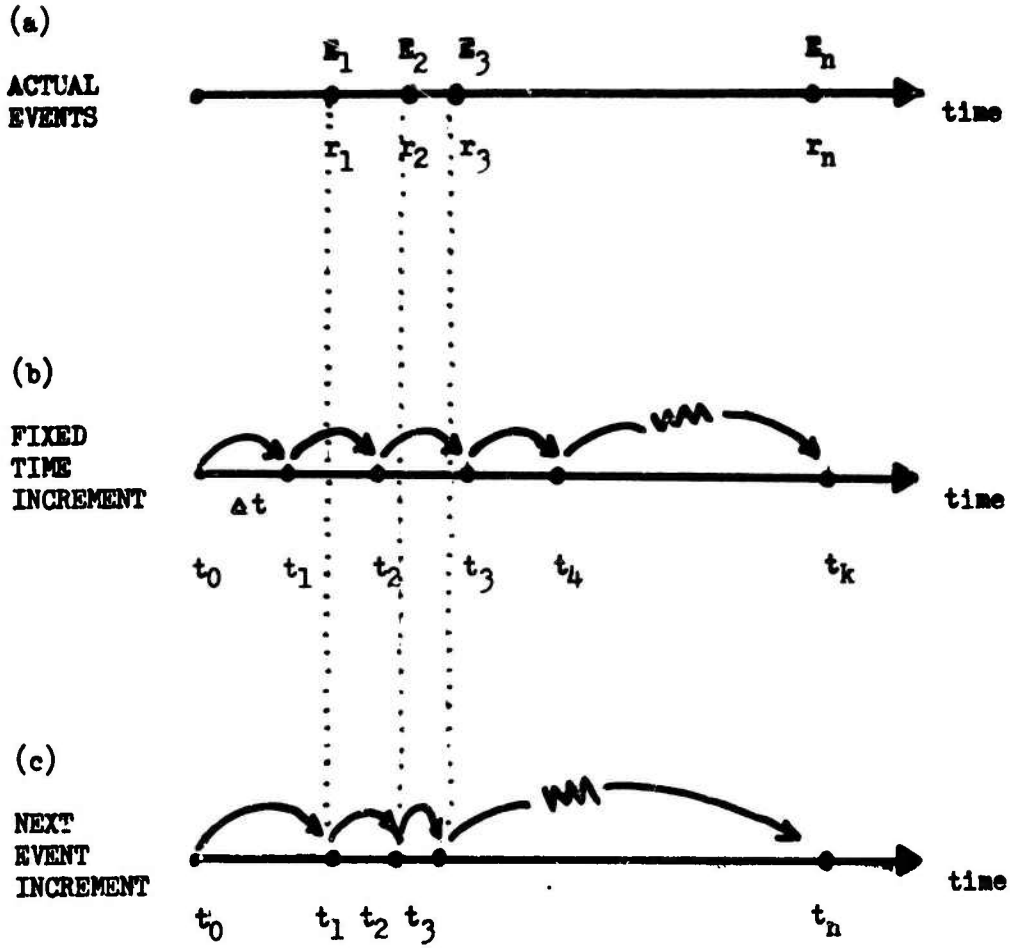


Figure 23. A Comparison of Real World Events and Fixed Time increment, and Next Event Increment Simulated Time

They are depicted as (a) in Figure 23. In fixed time incrementing the analyst defines an incremental time value Δt and the model progresses through simulated time by Δt increments:

for $t_o = o$, $t_1 = \Delta t$, $t_2 = 2\Delta t$, $t_3 = 3\Delta t$, $t_4 = 4\Delta t, \ldots, t_k = K\Delta t$

Line (b) of Figure 23 illustrates the fixed time increments. Time advances uniformly from t_o to t_1, t_1 to t_2, and continues advancing until t_k, which is the end of the simulation. Since time is simulated only at the time points t_o, t_1, $t_2, \ldots, t_k$, the events E_1, $E_2, \ldots$, E_n do not occur at their "real world" times. Event E_1 is shifted or translated in time to occur at t_2, events E_2 and E_3 are shifted to occur at t_3, and the shifting continues until event E_n occurs at t_k. Note that events E_2 and E_3, which really occurred at different times appear to occur simultaneously at the end of the time interval. Thus, all events having end times r_i, r_{i+1}, $r_{i+2}, \ldots, r_{i+n}$ such that

$$t_{k-1} < r_i, r_{i+1}, r_{i+2}, \ldots, r_{i+n} \leq t_k$$

are treated as if they occurred at t_k. The analyst must be careful in choosing Δt so that the interrelationship of event occurrences are not radically altered solely because of the time increment size. In practice, this is not a difficult problem since Δt can be chosen small

enough such that at most only one event is likely to occur. Some experienced researchers use the rule of thumb that a reasonable time interval is one tenth of the shortest expected inter-event time [7,P. 162].

Next Event Incrementing

In next event incrementing, time points are defined only at the occurrence of an event or events. Line (c) of Figure 23 illustrates the next event increments. Simulated time started at time t_o and advanced immediately to t_1 (the time of event E_1) then to t_2 (the time of event E_2) and continued to t_n (the time of event E_n). Thus, $r_1=t_1$, $r_2=t_2$, ... $r_n=t_n$. The simulation model updates the status of those variables associated with event E_j, processes E_j, determines the time and subsequent events which occur as a result of E_j, and then advances simulated time to the time of the next event E_{j+1}. Two events are not processed as simultaneous events unless they bear identical occurrence times.

A Comparison of Fixed Time and Next Event

Figures 24 and 25 depict the general structure of next event and fixed time increment time flow mechanisms. In next event incrementing the simulation master clock is advanced by the amount necessary to cause the next most imminent event to take place. When a particular

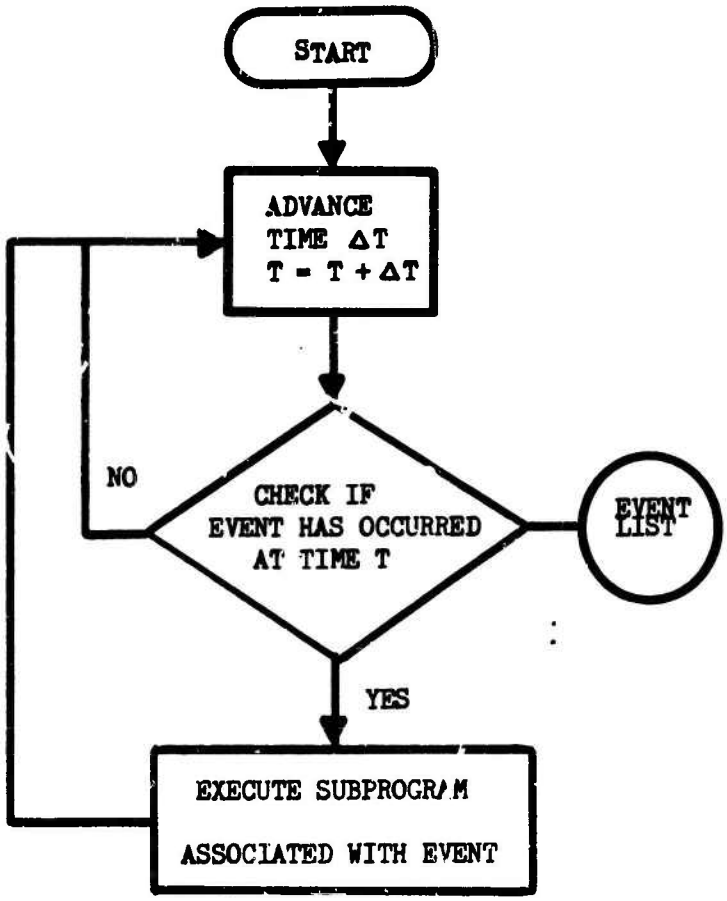


Figure 24. Flowchart For Fixed Time Increment Time Flow Mechanism

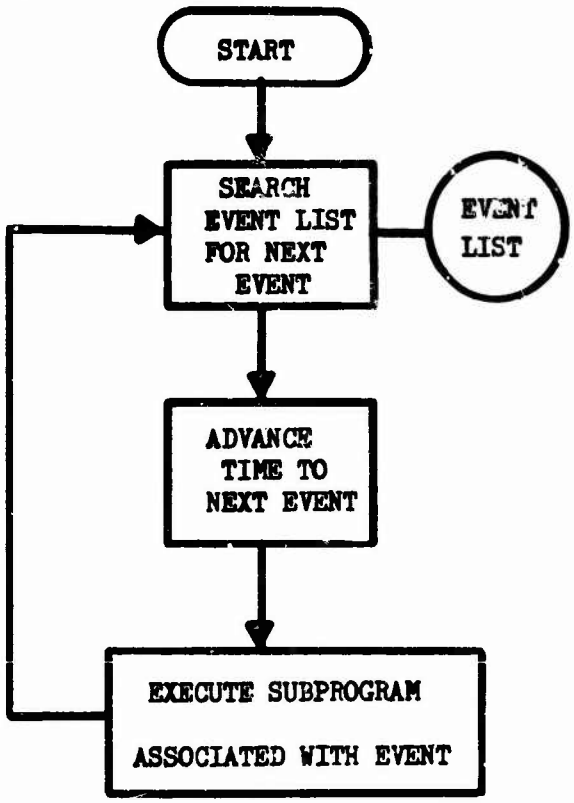


Figure 25. Flowchart For Next Event Time Flow Mechanism

event has been executed, clock time is advanced to the time at which the next significant event is to occur, whether seconds or hours away. The intervening periods, when no changes occur in the system, are skipped over. Thus, the passage of time is dependent on event occurrences. In fixed time incrementing, the master clock is updated in uniform discrete intervals of time. The system is scanned every unit of clock time to determine whether any events are due to occur at that particular clock time. Consequently, the occurrence of events is dependent on the passage of time.

The casual observer might foolishly conclude that a next event formulation is always preferred over the fixed time increment method. After all with next event method one does not need to choose an arbitrary and artificial time increment and events are not processed simultaneously unless they have identical occurrence times. The above argument is overly simplistic and a hasty generalization. Let's look at the tradeoffs entailed in the use of each method. With fixed time increments, some information is always lost since translation to the end of the time interval always results upon the occurrence of an event. False simultaneity can be induced due to translation of

events within the same interval. Furthermore, accuracy can be lost if Δt is too large and execution can be slowed if Δt is too small. Lastly, the model must always search for the next event even during intervals when no event occurs. With the next event method, execution time could be longer than fixed time incrementing for interdependent events due to increased computations. In addition, next event is cumbersome when one or more events change in a continuous fashion. Conway et al [6] has concluded that the desirability of the fixed time increment method increases with the increase in number of entities and the desirability of the next event method increases with the mean length of event times. Naylor [24] suggests using fixed time incrementing when events occur in a regular manner and next event when they occur unevenly in time. For some models, one method will be more effective than the other, and one must choose depending on which is best for the given situation.

Further Classification of Time Flow-Mechanisms

Earlier in this section we stated that historically time flow mechanisms were categorized as fixed time or next event incrementing. Kiviat [18] extended the classification to three methods: event scheduling, activities scan, and process interaction, and Nance [23] proposed a concept of a continuum of algorithms for time

flow in digital simulation. Since typically one is not able to conclusively separate time flow mechanisms into just two methods, we will examine the concept of a continuum in more depth.

The continuum has fixed time incrementing at one pole and next event at the other. A particular algorithm would lie somewhere along the continuum depending on the degree to which it possesses characteristics of each pole. Considering the innumerable discrete event simulations possible, there are no doubt an infinite number of time flow mechanisms. For any specific simulation application, the most efficient algorithm will be somewhere along the continuum.

Nance [23] examined several time flow mechanisms for the patrolling repairman problem to determine their relative effectiveness. The patrolling repairman problem is a classical machine interference problem from queuing theory. A single repairman is assigned to service a group of N semiautomatic machines which fail intermittently. We shall assume that the failure rates of the machines are identical, the event of failure for any machine is independent of the state of any other machine, and the time between failures of a single machine and the repair time are distributed negative exponentially with respective means λ and μ . The machine layout is a rectangular

pattern with two rows of the same number of machines. The repairman requires time T to walk between any two adjacent machines. The repairman unidirectionally patrols the perimeter of the machines repairing any failed machine. An event is defined as the failure of a machine.

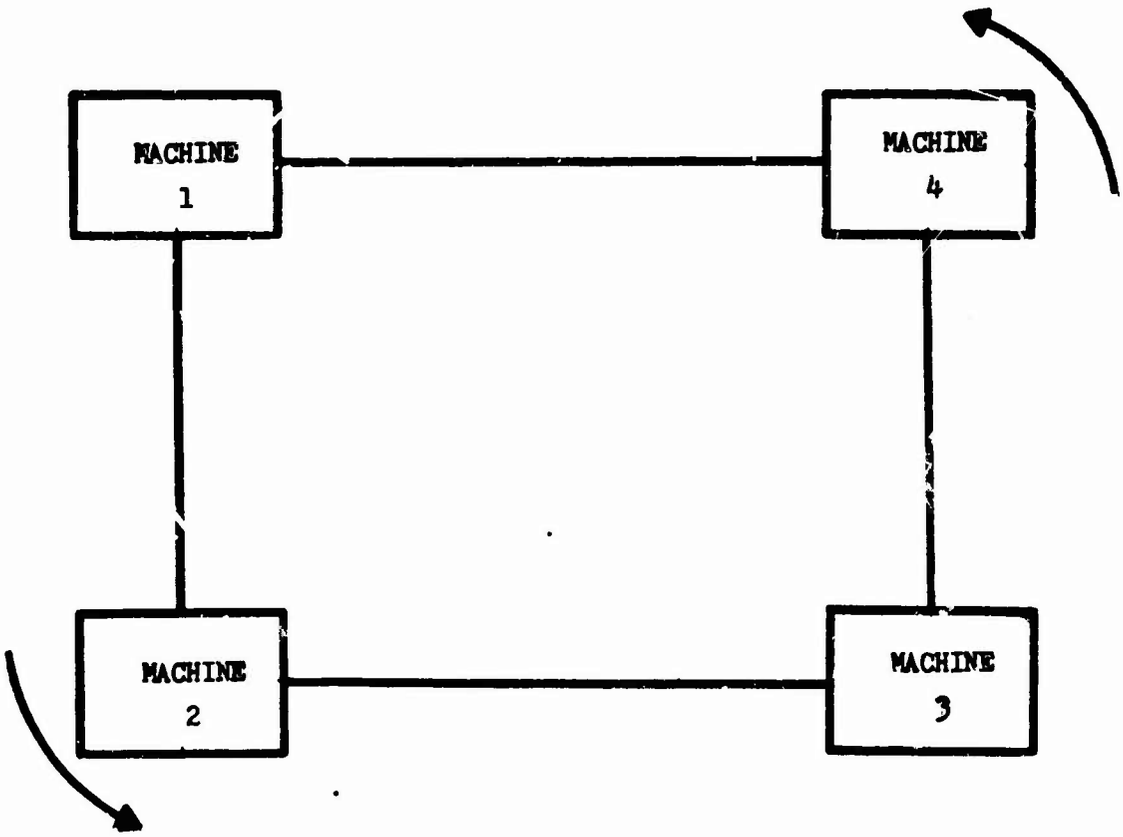


Figure 26. Patrolling Repairman Problem with N=4.

The time flow mechanisms which were examined included a pure fixed time increment, a largely next event type called MICA, and a hybrid called Constant Increment (CI). The CI algorithm is straightforward as follows:

STEP	COMMENT
1. J = J+1	Advance to next facility in path
2. IF(J>N)? J=1	Check for correct facility number so N is not exceeded. If true, set j=1; otherwise, go to 3
3. Clock = Clock + T	Increment time by Constant value
4. IF(CLOCK≥f_j) next event at j	Test if j-th machine has failed. Let f_j- failure time of machine j. If true the next imminent event occurs at j; otherwise, go to 1.
5 Go to 1.	

The algorithm records which machine is being inspected(j), increments CLOCK by T, and compares T with the failure time of machine j (fj). If the master clock time is greater than or equal to fj, then the jth machine has failed and the next imminent event occurs at machine j. If CLOCK is less than fj, then the machine is operational and j is incremented by one. On the continuum the time flow algorithm can be depicted as follows:

Execution time as a function of parameters λ and μ, the number of machines, and number of repairs was studied for each algorithm. As the number of machines increased the differences among the algorithms became pronounced. The conclusions were that as more

efficient than MICA as λ/μ increased, MICA was more sensitive and less efficient than CI as intermachine transit time (T) increased, and regardless of the number of repairs generated (within the test values of N, T, and λ/μ), CI was more efficient than MICA. The study indicated that MICA suffered from a requirement for too much computation time per test and an increasing number of machines tested due to an increase in machines or intermachine transient time.

The most important result from this study is that the procedure by which the passage of time is structured most advantageously depends on the system being modeled. It may display in various degrees characteristics of the two poles, i.e., fixed time or next event.

SECTION V

OUTPUT GENERATION AND STATISTICAL ANALYSIS

Frequently the areas of output generation and statistical analysis of the model results are given little emphasis by the model builder. But the analyst should keep in mind that decisions about structural characteristics of the model must be made in parallel with decisions about how the model will be used. This implies that decisions about how detailed the simulation should be are affected by what output is desired. The output, in turn, should satisfy the requirements of the decision maker, who, therefore, should be involved in output design.

Output Requirements

The analyst must address some fundamental questions before devising the output routines. Consistent with the objectives of the study, he must decide:

- What information is required,
- What statistics provide this information,
- What analysis techniques provide sufficient statistics on model behavior, and
- What format best communicates the information derived from model results and analysis.

The analyst must also consider whether to combine the data analysis routines in the simulation model program or provide information in some form to be used by a

separate analysis program. In addition, the output itself could be punch cards, tape or disc files, or printouts. Lastly, diagnostic information about the model should be incorporated with output information derived from the model. In this way data, say from the random number generator, can be compared to theoretical values of the distribution assumed.

If the simulation is deterministic, there are no problems of statistical inference. The output of the simulation at the end of the computer run is the required measure of system performance. Thus, we will henceforth treat probabilistic models.

Basic Terminology

Before beginning the discussion of statistical analysis of the model results, it would be wise to define some basic terminology. A simulation run is an uninterrupted exercise of the model for a specified combination of controllable variables or parameters. A replication (for stochastic models) of a run is an exercise for the same combination but with different random variations. An observation from the model is a segment of a run sufficient for estimating the value of some measure or statistic.

Preliminaries to Analysis

Once the general outline of the output requirement has been established, certain statistical considerations must be recognized. A simulation model is concerned with producing a stochastic process, i.e., a family of random variables $\{X(t),\ t \geq 0\}$ indexed by a time parameter t. The analyst is interested in the behavior of some variable X as a function of time. This time dependent behavior produces results which are not independent but, in fact, interdependent to some degree. For example, in a queuing model for any single simulation run, the waiting times of successive customers will be autocorrelated since there is a greater likelihood that the (j+1)st customer will be delayed if the jth customer waits, than if the jth customer were served immediately. To determine a statistically valid measure of system performance, the analyst must replicate experiments with varying sequences of pseudorandom numbers. Since the behavior of the time dependent process may be irregular, extended run lengths may be required.

After the model is constructed, the analyst is faced with designing the simulation experiment -- the set of runs for the model. He must determine the

- initial state definition, i.e., starting conditions for the model
- criteria for recognition of steady state,

- parameter settings to expose different system responses
- length of each run, and
- sampling procedures including sample size and method of selection.

Initial State Definition

Unlike the real world system, the simulation model is not in continual use. The analyst is faced with the problem of how to start the model and obtain measurements which are not biased by the method of starting or stopping.

Simulation is frequently used to study the performance of a system which operates under steady-state conditions. The analyst attempts to determine the limiting distribution of the state of the system or more precisely empirical estimates of the distribution's moments such as mean and variance.

A system whose behavior does not satisfy steady-state conditions is considered to be in a transient state. When starting conditions are not near steady-state, a transient period exists until the state of the model approaches the steady-state condition. Implied in the above is the assumption that a steady-state exists. For some real world systems, a transient phenomenon may actually occur prior to the steady state and the transient period may be pertinent to the study. For other systems, no steady-state exists and it is the transient phase itself

which is to be studied. In a strategic air defense model, it is the transient state, where aircraft initially attempt to penetrate the air defense system and destroy their assigned targets, that is of prime importance. Either no steady-state exists because of the system's dynamic and reactive capabilities, or it is not reached because the air battle does not last sufficient time or because there is not an infinite supply of aircraft or other resources.

Let us consider some strategies for setting initial conditions. An appealing method is, to use queuing terminology, empty and idle, i.e., the system has no activity. The simulation is then run until the transient effects are apparently insignificant. However, Conway [5] suggested that empty and idle is a poor state to consider since almost anything else is better. In fact, unless empty and idle is a typical state of the system we should choose another condition. Ideally, we would like to select starting conditions corresponding to the steady-state condition. But an analyst who has already determined what the equilibrium state of the system is either does not need simulation or may bias the results to his preconceived conclusions if data is collected before the transient effects are removed. Thus, we can only conclude that the analyst probably knows something about the equilibrium behavior of the system and should use this

information to select a starting value. Realistic initial conditions will reduce the cost to reach steady state and will affect the precision of statistical estimates.

Simulation models are frequently used to compare two alternative system configurations or disciplines. To avoid biasing the results and retain a basis for comparison, the initial state definition for both configurations should be the same, such as a compromise or average of the expected steady-state levels for each.

Two methods are commonly used to remove transient effects in simulations. The first is to use long simulation runs so the data from the transient period is insignificant compared to the amount of steady-state data. Although this method is simple to employ, it can be costly in terms of total running time and computer charges.

The second method is to run the simulation until steady-state conditions are achieved and then truncate all sampled data up to that point. The conditions at the end of the transient state become, in effect, the initial state for the data collection phase of the simulation. Theoretically, one could start the model anywhere and merely run the simulation for a sufficient period of time until transient effects are removed and truncate data. However, Fishman [10] has shown that if mean square error (MSE) is used as a criterion, where

$$\text{MSE} = \text{variance} + (\text{bias})^2$$

then truncation decreases the sample size needed to reduce bias but can inflate variance so that mean square error is not reduced. We cannot say categorically that the truncation strategy reduces mean square error. Later in this section we will discuss variance reduction techniques which, when used together with truncation, provide an acceptable treatment of transient effects.

Recognition of Steady-State

Whether we are interested in transient behavior or steady-state conditions, we require a criterion for recognition of steady-state. Equilibrium is a limiting condition which may be approached but never actually attained. There is no single point in the simulation beyond which the system is at steady-state. We will assume that the difference between the temporal and limiting distribution decreases with time and that beyond a given point one is willing to neglect the error made by considering the system to be at equilibrium. Thus, we assume that $L_k(t)$, $k=1,2,\ldots,n$ is the limiting distribution for an n state process and $S_k(t)$ is the simulated temporal distribution. Then

$$\left| L_k(t) - S_k(t) \right| = E_k(t)$$

where

$$\lim_{t \to \infty} E_k(t) = 0 \quad \text{for } k = 1,2,\ldots, n$$

Although there are no fixed rules for determining

when transient behavior ends, several methods are in current use. A simple method is to examine a sequence of observations and assume steady-state begins at the first point which is neither the maximum nor the minimum of the preceding sequence. A second method is to select the point which is the first value to repeat itself after the start. The third, and probably the most frequently used method, is to select the time when three consecutive points differ by no more than E, say E=.01. Each technique will differ in the point selected for steady-state and all may be good values. The analyst must choose which method is most appropriate for his simulation.

Sampling Procedures

It is not difficult to estimate the means of attributes associated with permanent and temporary entities. But is is much more difficult to state a level of confidence that the mean is the true mean. This requires the variance of the data. In the following discussion we will first examine the measurement of simulation results for permanent and then for temporary entities.

Permanent Entities

Permanent entities exist throughout the simulation and their attributes take on different values at each

point in time. For attribute X(t) over the time interval (T_r, T_t), an estimate of the mean is

$$E[X(t)] = \frac{1}{T_t - T_r} \int_{T_r}^{T_t} X(t)dt$$

where T_t = time simulation terminated
T_r = time statistical accumulation begins.

As we previously mentioned problems arise with the variance estimate due to autocorrelation introduced in calculating values associated with permanent entities.

Independent Replicates

One obvious method of eliminating problems of autocorrelation is to replicate each run several times. The mean from each run is treated as one observation in the total sample used to calculate the appropriate statistic. Of course, on each run the transient data prior to steady-state is lost leading to higher computer run costs. For sample size n, the estimate of the mean is $\bar{X} = \sum_{i=1}^{n} X_i/n$

and the estimate of the variance of the sample mean is

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

or equivalently,

$$Var(\bar{X}) = \frac{1}{n} (\sum X_i^2 - n\,\bar{X}^2).$$

The reader should note that the latter formulation of the

sample variance estimate is easier to implement in a computer program since only the sums of X_i^2 and X_i need be accumulated and no individual X_i values stored.

Stratified Sampling

To avoid the loss of data over the interval (T_r, T_t), consider replications by dividing a single simulation run into n parts over the interval (T_r, T_t). This prevents the waste of (n-1) periods of $(0, T_r)$ but succeeding sample values are not independent.

Let's define a measurement process in terms of measurement periods where an estimate $E[X(t)]$ is determined and nonmeasurement periods in which no statistical data is initially gathered [5]. Suppose we have somehow determined an interval of length so the measurement periods are independent. Then there are n/2 measurement periods for which attribute values $X_2, X_4, \ldots, X_n$ can be determined. We will assume that these values are n/2 independent identically distributed random variables with mean μ and variance σ^2. We have then for the measurement periods

$$X = \sum_{k=1}^{n/2} \frac{X_{2k}}{n/2} = \mu$$

$$\mathrm{Var}\ (X) = \frac{\sigma^2}{n/2} = 2\,\frac{\sigma^2}{n}$$

and for the nonmeasurement periods

$$Y = \sum_{k=1}^{n/2} \frac{X_{2K+1}}{n/2} = \mu$$

$$\mathrm{Var}\ (\bar{y}) = \frac{2\sigma^2}{n}$$

We can pool the means so that

$$\bar{Z} = \frac{\bar{X}+\bar{Y}}{2} = \sum_{i=2}^{n+1} \frac{X_i}{n}$$

and

$$E[\bar{Z}] = E[\frac{\bar{X}+\bar{Y}}{2}] = \frac{E[\bar{X}] + E[\bar{Y}]}{2}$$

$$= \frac{2\mu}{2} = \mu$$

The lack of independence prevents a directly additive form for the pooled variance.

$$\mathrm{Var}\ (\frac{X+Y}{2}) = 1/4\ [\mathrm{Var}\ \bar{X} + \mathrm{Var}\ \bar{Y} + \mathrm{Cov}\ (\bar{X},\bar{Y})]$$

$$= 1/4\ [\mathrm{Var}\ \bar{X} + \bar{Y}] + 1/4\ \mathrm{Cov}\ (\bar{X},\bar{Y})$$

$$= 1/4\ [\mathrm{Var}\ \bar{X} + \mathrm{Var}\ \bar{Y}] + \frac{1}{n^2} \sum_{i<j}\sum (X_i - \bar{X})(Y_j - \bar{Y})$$

$$= 1/4\ [\mathrm{Var}\ \bar{X} + \mathrm{Var}\ \bar{Y}] + \frac{2(n-1)c}{n^2}$$

where C is the covariance term $(X-\bar{X})(Y-\bar{Y})$ between adjacent periods. The covariance between nonadjacent periods is assumed to be zero. In fact the length of the period is selected so that this is true

The importance of this technique lies in the fact that the pooled variance is smaller than the variance from either set of periods. Since

$$\mathrm{Var}\ (\bar{Z}) = 1/4(\mathrm{Var}\ \bar{X} + \mathrm{Var}\ \bar{Y}) + \frac{2(n-1)c}{n^2}$$

$$= 1/4\ (\frac{2\sigma^2}{n} + \frac{2\sigma^2}{n}) + \frac{2(n-1)c}{n^2}$$

$$= \frac{\sigma^2}{n} + \frac{2(n-1)c}{n^2}$$

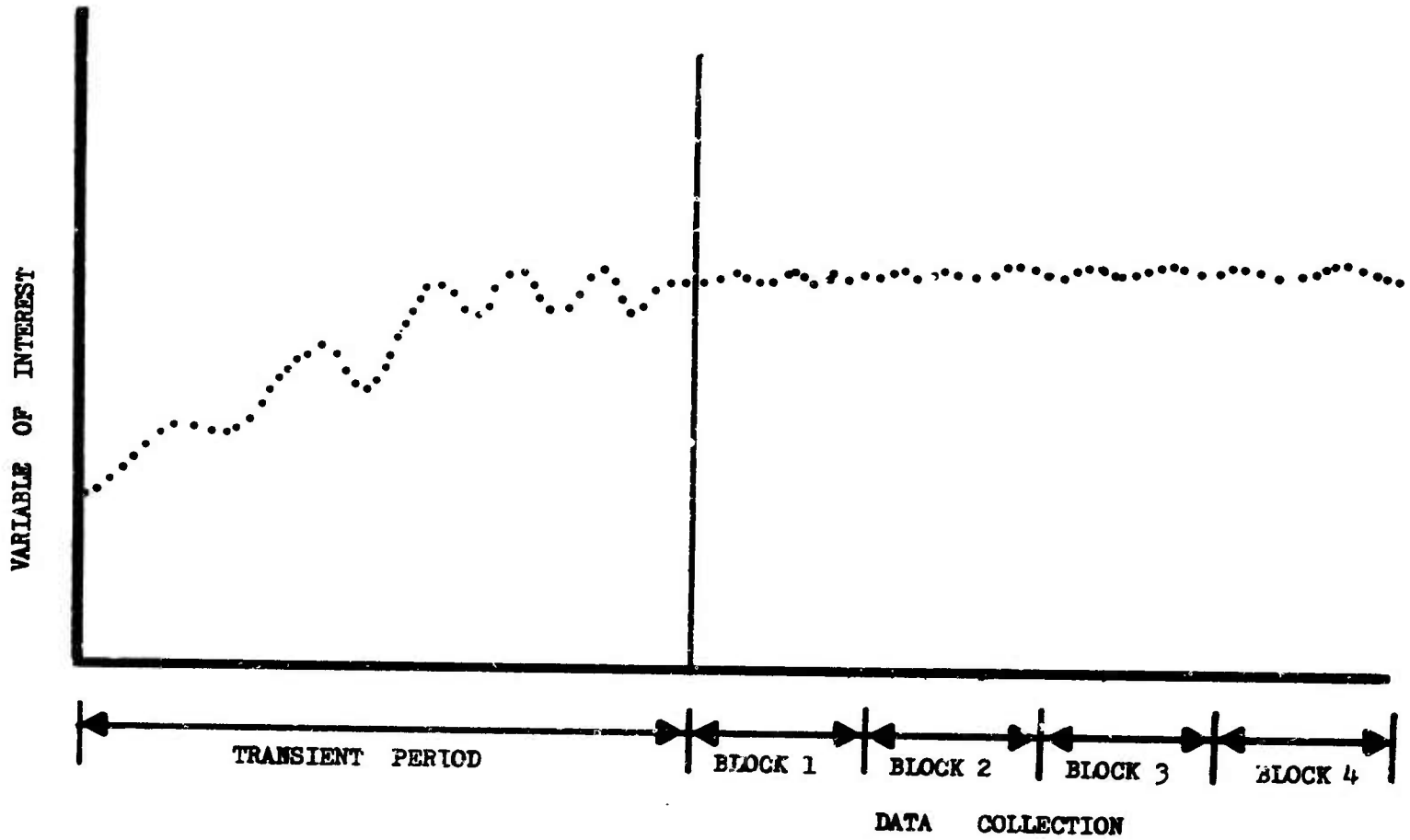


Figure 27. Measurement of Attributes Using Truncation and Stratified Sampling

We have $\mathrm{Var}\ (\bar{\bar{Z}}) = \frac{n\sigma^2+2(n-1)c}{n^2}$

It can be shown that $c \leq \frac{\sigma^2}{2}$

so $\mathrm{Var}\ (\bar{\bar{Z}}) = \frac{n\sigma^2+2(n-1)C}{N^2} \leq \frac{\sigma^2}{n}\ 1+\frac{(n-1)}{n}$

$$< \frac{2\sigma^2}{n} = \mathrm{Var}\ (\bar{X})$$

This variance reduction technique is called <u>stratified sampling</u>. It is also sometimes known as blocking since the data is essentially divided into odd and even blocks from which sample means are calculated.

<u>Estimating Autocorrelation</u>

In the above version of stratified sampling the length of each measurement period was selected so that consecutive measurement periods were independent, i.e., the correlation was zero. In some cases nonadjacent periods may have a positive covariance that is significant. Then the autocorrelation function can be estimated and included in estimates of the variance.

The autocorrelation function between an observation at simulated time t and an observation at t+s is

$$R^{(s)} = \frac{1}{q-s} \sum_{t=1}^{q-s} (X_t-\bar{X})\ (X_{t+s}-\bar{X})$$

for $s=0,1,2,\ldots, n-1$

where X_t = observation at time t

$\bar{X}$ = sample mean
q = total time of simulation
s = lag

The variance is then

$$Var(\bar{z}) = \frac{1}{n^2}\left[\sigma^2 n + 2\sum_{s=1}^{n-1}(n-s)\,R(s)\right]$$

Antithetic Variates

Another variance reducing method is Antithetic variates. The aim is to introduce negative correlation between two separate replications of the simulation, so the variance of the combined averages is less than if the replications were independent. The variance of the mean of two replications is

$$var\ (\bar{k}) = \frac{\sigma^2}{n}(1+P)$$

where σ^2 = population variance

p = correlation between pairs of observations.

If the observations are independent, P=0. But if there is negative correlation (P<0), then the variance of the sum of observations will be reduced.

The procedure most commonly followed to generate negatively correlated variates is to use uniform pseudorandom numbers [Vj] for probabilistic events on one run and to use [1-Vj] for the equivalent event on the second run. It is important to maintain event equivalence between runs since otherwise the processes will be out of phase and proper negative correlation will not result.

Sample Size Determination

The determination of an appropriate sample size for simulation is no different from sample size determination in ordinary statistical problems. The added complication is that two sizes must be determined: the run length and number of replications.

If stratified sampling is used, the length of each block or measurement period can be determined by examining the autocorrelation function. The number of replications or number of blocks is dependent on what the analyst believes is a valid sample size. Since computer time is not free and execution times for complex simulations can be great, sample sizes tend to be modest. The clear preference is for large sample sizes since large sample statistics ($n \geq 30$) allow the Central Limit Theorem to be invoked and normal approximations made for hypothesis testing.

Fishman's work with the spectral density function has led to procedures for determining the sample size for equivalent independent observations [9]. The crucial factor in deciding to use sophisticated techniques is whether the benefits are sufficient to warrant the extra computations required.

Temporary Entities

All of the above procedures were discussed in the context of measuring attributes associated with permanent

entities. The problems relating to temporary entities are somewhat different.

Temporary entities are active for only a portion of the total simulation and usually exist in such numerous quantities as to prevent maintaining records on each individual entity. Typically, only the final value of an attribute is measured for a temporary entity before it becomes inactive. Thus each entity contributes a single value and the sample size is the number of entities. Temporary entities existing at the same time are subject to the same system conditions so that the attributes tend to be correlated. Another difficulty involves which entities to include since temporary entities often are not created and destroyed in the same order. For example, in a simulation of customers shopping in a commissary the people usually do not leave in the same order in which they entered.

Conway [5] suggested three strategies for collecting a sample of temporary entities. The first was to designate an interval of time and include in the sample the final value of the relevant attribute for those entities whose active history terminated during the interval. In comparing say two alternatives, the sample size N may vary but the time interval is fixed. The second strategy was to specify a beginning point and a sample size and include the required number for consecutive

terminations after this point. Here the sample size is fixed but the duration of the run may vary. Furthermore, in both of the above strategies entities which were created during the transient period may be included in the sample. The third strategy is to specify a beginning point and sample size and include all attribute values for temporary entities created consecutively after this point. Although no transient effects are included, a problem of termination arises since the run must continue until all temporary entities in the sample are no longer active.

Confidence Intervals

In the statistical development thus far, we have been concerned with reducing the variance of the sample mean since it expresses in some fashion the reliability or variability of our estimate of the mean. To give better definition to our statistical estimates, we need to examine the idea of confidence intervals.

Confidence intervals are based on statistical properties of the sample statistic and express the probability $1-\alpha$ that the random variable will take on values within the constructed interval. The preassigned probability $1-\alpha$ is called the degree of confidence.

Below we will consider confidence intervals for some standard distributions. Let $X_1, X_2, \ldots, X_n$ be a random sample with the sample mean denoted by $\bar{X}$.

Mean of a Normal Distribution with Known Variance

Suppose we are sampling from a normal population with mean μ and known variance σ^2. The random variable

$$R = \frac{\bar{X} - \mu}{\sigma/\sqrt{n}}$$

is distributed normally with mean 0 and variance 1. Let $z_{\alpha/2}$ be such that the integral of the standard normal density from $z_{\alpha/2}$ to infinity equals $\alpha/2$. Then the random variable R will take on a value between $-z_{\alpha/2}$ and $z_{\alpha/2}$ with probability $1-\alpha$. We then have that

$$-Z_{\alpha/2} < \frac{\bar{X}-\mu}{\sigma/\sqrt{n}} < Z_{\alpha/2}$$

or $$\bar{X} - Z_{\alpha/2}\frac{\sigma}{\sqrt{n}} < \mu < \bar{X} + Z_{\alpha/2}\frac{\sigma}{\sqrt{n}}$$

where $\bar{X}$ is the sample mean. For given α, the value $Z_{\alpha/2}$ can be obtained from a standard normal cumulative probability table. The last expression above is the confidence interval for the mean when the variance is known.

Mean of a Normal Distribution with Unknown Variance

In some instances we may know that we are sampling from a normal population but do not explicitly know the variance. It can be shown [13, p.222] that for random samples of size n from normal populations the random variable

$$\frac{\bar{X}-\mu}{S/\sqrt{n}}$$

has a t distribution with n-1 degrees of freedom. The random variable will take on a value between $-t_{\alpha/2,\,n-1}$ and $t_{\alpha/2,\,n-1}$ with probability $1-\alpha$. Let the (unbiased) sample variance be defined as

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

Then for a sample with mean $\bar{X}$ and standard deviation S.

$$-t_{\alpha/2,\,n-1} < \frac{\bar{X}-\mu}{S/\sqrt{n}} < t_{\alpha/2,\,n-1}$$

or

$$\bar{X} - t_{\alpha/2,n-1}\frac{S}{\sqrt{n}} < \mu < \bar{X} + t_{\alpha/2,n-1}\frac{S}{\sqrt{n}}$$

The last expression is the $1-\alpha$ confidence interval

for μ when σ^2 is unknown for normal populations.

Large Sample Confidence Intervals For the Mean

Often we are able to deal with samples sufficiently large ($n \geq 30$) to justify the use of the Central Limit Theorem. If the variance of the population is known, then the $1-\alpha$ confidence interval for μ is

$$\bar{X} - Z_{\alpha/2} \cdot \frac{\sigma}{\sqrt{n}} < \mu < \bar{X} + Z_{\alpha/2} \frac{\sigma}{\sqrt{n}}$$

If the variance is unknown, then an approximate large sample confidence interval for μ is

$$\bar{X} - Z_{\alpha/2} \frac{S}{\sqrt{n}} < \mu < \bar{X} + Z_{\alpha/2} \frac{S}{\sqrt{n}}$$

where S is the sample standard deviation.

Variance of a Normal Distribution

Suppose we have a random sample of size n from a normal population and we desire a $1-\alpha$ confidence interval for σ^2. It can be shown [13, p.214] that the random variable

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

has a Chi Square distribution with n-1 degrees of freedom. The random variable will take on a value between $\chi^2_{1-\alpha/2,\, n-1}$ and $\chi^2_{\alpha/2,\, n-1}$ with probability $1-\alpha$. Then we have

$$\chi^2_{1-\alpha/2,\,n-1} < \frac{\sum (x_i - \bar{x})^2}{\sigma^2} < \chi^2_{\alpha/2,\,n-1}$$

or

$$\frac{\sum (x_i - \bar{x})^2}{\chi^2_{1-\alpha/2,\,n-1}} < \sigma^2 < \frac{\sum (x_i - \bar{x})^2}{\chi^2_{\alpha/2,\,n-1}}$$

The last expression is the $1-\alpha$ confidence interval for the variance of a normal population.

Hypothesis Testing

A simulation study is often employed to compare two or more strategies or alternatives. One means of comparison is to use statistical tests of hypotheses where some assumption is made about the probability distribution of the population. It is the nature of statistical testing that rejection of a hypothesis can be more positively stated than acceptance. The hypothesis formulated for testing is called the null hypothesis and is denoted H_0. The hypothesis which contradicts the null hypothesis is called the alternative hypothesis and is denoted H_1 or H_a.

Two types of errors may be committed while testing an hypothesis. If the null hypothesis is true and the analyst rejects it, he is said to have committed a Type I Error. If the null hypothesis is not rejected when in fact it is false, he is said to have committed a Type II Error.

In order to test an hypothesis the analyst must construct regions of acceptance and rejection for sample

values. The set of values for which the null hypothesis is rejected is called the <u>critical region</u>. The size of critical region is the probability of committing a Type I Error and is usually denoted as α. The size of the Type II Error is the probability that a sample value will be outside the critical region (therefore in the region of acceptance) when the null hypothesis is false. As long as the sample size is fixed, an inverse relationship exists between Type I and Type II errors: if the probability of one error is reduced, the probability of the other is increased. The only way to reduce the probabilities of both errors is to increase the sample size.

Tests are generally classified as one-sided or two-sided. In a one-sided test, one tail of the distribution of the test statistic is the critical region. In a two-sided test, both tails are used. Selection of one or two-sided tests depends on the nature of the alternative hypothesis. Usually two-sided alternatives lead to two-tailed tests and one-sided alternatives lead to one-tail tests. For instance, for the null hypothesis H_0; $\mu = \mu_0$ and the alternative H_1: $\mu \neq \mu_0$, we would have a two-sided test. If the alternative were H_1: $\mu > \mu_0$, the test would have been one-sided.

Often in hypothesis testing, the alternative to rejecting the null hypothesis is reserving judgment. Since the null

hypothesis is not accepted, we cannot commit a Type II error. Such tests are called tests of significance and the probability of a Type I error (α) is called the level of significance. The most frequently used values of α are .05 and .01.

Sampling from Normal Populations

Let $X_1, X_2, \ldots, X_n$ be a sample of size n from a normal population. Let $\bar{X}$ be the sample mean and α be the size of the critical region.

The Mean of Normal Population with Known Variance

H_0 : $\mu = \mu_0$

H_1 : $\mu > \mu_0$;or $\mu < \mu_0$; or $\mu \neq \mu_0$

The test statistic is

$$Z = \frac{\bar{X} - \mu_0}{\sigma/\sqrt{n}}$$

and the critical regions are

$Z \geq Z_\alpha$ for $\mu > \mu_0$

$Z \leq Z_\alpha$ for $\mu < \mu_0$

$|Z| \geq Z_{\alpha/2}$ for $\mu \neq \mu_0$

The Mean of a Normal Population with Unknown Variance

H_0 : $\mu = \mu_0$

H_1 : $\mu > \mu_0$; or $\mu < \mu_0$; or $\mu \neq \mu_0$

The test statistic is

$$t = \frac{\bar{X} - \mu_0}{S/\sqrt{n}}$$

where S^2 is the sample variance.

The critical regions are

$t \geq t_{\alpha, n-1}$ for $\mu > \mu_0$

$t \leq -t_{\alpha, n-1}$ for $\mu < \mu_0$

$|t| \geq t_{\alpha/2, n-1}$ for $\mu \neq \mu_0$

where $t_{\alpha, n-1}$ is such that the integral of the t distribution of n-1 degrees of freedom from negative infinity to $t_{\alpha, n-1}$ equals $1-\alpha$.

<u>Difference Between Means with Known Variances</u>

Let $X_1, X_2, \ldots, X_n$ and $Y_1, Y_2, \ldots, Y_m$ be random samples of size n and m from two normal populations with variances σ_1^2, and σ_2^2 respectively.

H_0: $\mu_1 - \mu_2 = \delta$

H_1: $\mu_1 - \mu_2 > \delta$; or $\mu_1 - \mu_2 < \delta$; or $\mu_1 - \mu_2 \neq \delta$

The test statistic is

$$Z = \frac{\bar{X} - \bar{Y} - \delta}{\sqrt{(\sigma_1^2/n) + (\sigma_2^2/m)}}$$

The critical regions are

$$Z \geq Z_{\alpha} \quad \text{for} \quad \mu_1 - \mu_2 > \delta$$

$$Z \leq -Z_{\alpha} \quad \text{for} \quad \mu_1 - \mu_2 < \delta$$

$$|Z| \geq Z_{\alpha/2} \quad \text{for} \quad \mu_1 - \mu_2 \neq \delta$$

<u>Difference Between Means for Unknown Variances</u>

The hypotheses are the same as in the previous case. The test statistic is

$$t = \frac{\bar{X} - \bar{Y} - \delta}{\sqrt{[(1/n)+(1/m)][(n-1)S_1^2 + (m-1)S_2^2]/(n+m-2)}}$$

where S^2, and S_2^2 are the sample variances.

The critical regions are

$$t \geq t_{\alpha, n-m-2} \quad \text{for} \quad \mu_1 - \mu_2 > \delta$$

$$t \leq -t_{\alpha, n-m-2} \quad \text{for} \quad \mu_1 - \mu_2 < \delta$$

$$|t| \geq t_{\alpha/2, n-m-2} \quad \text{for} \quad \mu_1 - \mu_2 \neq \delta$$

<u>Variance for Unknown mean</u>

Ho: $\sigma^2 = \sigma_1^2$

H_1: $\sigma^2 > \sigma_1^2$; or $\sigma^2 < \sigma_1^2$; or $\sigma^2 \neq \sigma_1^2$

The test statistic is

$$X^2 = \frac{\sum (x_i - \bar{x})^2}{\sigma_1^2}$$

The critical regions are

$$X^2 \geq \chi^2_{1-\alpha, n-1} \quad \text{for} \quad \sigma^2 > \sigma_1^2$$

$$X^2 \leq \chi^2_{\alpha, n-1} \quad \text{for} \quad \sigma^2 < \sigma_1^2$$

$$X^2 \leq \chi^2_{1-(\alpha/2), n-1}$$

or $X^2 \geq \chi^2_{\alpha/2,\, n-1}$ for $\sigma^2 \neq \sigma_1^2$

Where χ^2 is from a Chi Square distribution with N-1 degrees of freedom.

Large Sample Tests

When sample sizes are large enough ($n \geq 30$) for the Central Limit Theorem to be applied, the following tests may be used.

Mean When Variance Is Known

H_0: $\mu = \mu_0$

H_1: $\mu > \mu_0$;or $\mu < \mu_0$;or $\mu \neq \mu_0$

The test statistic is

$$Z = \frac{\bar{X} - \mu_0}{\sigma / \sqrt{n}}$$

the critical regions are

$Z \geq Z_\alpha$ for $\mu > \mu_0$

$Z \leq -Z_\alpha$ for $\mu < \mu_0$

$|Z| \geq Z_{\alpha/2}$ for $\mu \neq \mu_0$

where $Z_{\alpha/2}$ is from a normal table.

Mean When Variance Is Unknown

H_0: $\mu = \mu_0$

H_1: $\mu > \mu_0$;or $\mu < \mu_0$;or $\mu \neq \mu_0$

The test statistic is

$$Z = \frac{\bar{X} - \mu_0}{S/\sqrt{n}}$$

where S^2 is the sample variance.

The critical regions are

$$Z \geq Z_\alpha \quad \text{for} \quad \mu > \mu_0$$

$$Z \leq -Z_\alpha \quad \text{for} \quad \mu < \mu_0$$

$$|Z| \geq Z_{\alpha/2} \quad \text{for} \quad \mu \neq \mu_0$$

<u>Differences in Mean When Variances are Unknown</u>

Ho: $\mu_1 - \mu_2 = \delta$

H_1: $\mu_1 - \mu_2 > \delta$; or $\mu_1 - \mu_2 < \delta$; or $\mu_1 - \mu_2 \neq \delta$

The test statistic is

$$Z = \frac{\bar{X}_1 - \bar{X}_2 - \delta}{\sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}}$$

where S_1^2, and S_2^2 are the sample variances. The critical regions are

$$Z \geq Z_\alpha \quad \text{for} \quad \mu_1 - \mu_2 > \delta$$

$$Z \leq -Z_\alpha \quad \text{for} \quad \mu_1 - \mu_2 < \delta$$

$$|Z| \geq Z_{\alpha/2} \quad \text{for} \quad \mu_1 - \mu_2 \neq \delta$$

Nonparametric Methods

When a normal assumption cannot be met or when large sample testing is not feasible, the analyst may adopt nonparametric methods. One such nonparametric test, the sign test, was introduced in the discussion of pseudorandom numbers.

Sign Test

Suppose we have a symmetrical distribution so that the probability that a value is greater than the mean equals the probability that it's less than the mean. We will take as our hypothesis

H_0: $\mu = \mu_1$

H_0: $\mu > \mu_1$; or $\mu < \mu_1$; or $\mu \neq \mu_1$

Then, we will test the null hypothesis $\mu = \mu_1$, by replacing each sample value exceeding μ_1, by a plus sign(+) and each value less than μ_1, by a minus (-) and testing a new null hypothesis that the plus and minus signs represent a random sample from a binomial population

H_0: $p = .5$

H_1: $p > .5$ (or $p < .5$ or $p \neq .5$)

The test statistic is S, the number of plus signs. The critical regions are

$S \geq k_d$ for $p > .5$

$S \leq k_d$ for $p < .5$

$S \geq k_{d/2}$ for $p \neq .5$

or $s \leq K'_{d/2}$

where k_d is the smallest integer such that

$$\sum_{y=k}^{n} b(y;n,.5) \leq d$$

and

$b(y;n,.5)$ is the binomial probability of y successes in n trials.

Similarly, k^1_d is the largest integer for which

$$\sum_{y=0}^{k'_d} b(y;n,.5) \leq d$$

the values k^1_d and k_d can be quickly determined from a table of binomial probabilities for the given sample size n by respectively moving down the table of Y values from y=0 until the cumulative probability is d and by moving up from y=n until the cumulative probability is d .

One should note that if symmetry is not a valid assumption, the population median can be used in lieu of the mean. Also, if $n > 100$ the normal approximation to the binomial distribution can be used and the test statistic becomes

$$Z = \frac{y - np}{\sqrt{np(1-p)}}$$

which for p=.5,

$$Z = \frac{2Y-n}{\sqrt{n}}$$

The critical regions are

$$Z \geq Z_{\alpha} \quad \text{for} \quad p > .5$$

$$Z \leq -Z_{\alpha} \quad \text{for} \quad p < .5$$

$$|Z| \geq Z_{\alpha/2} \quad \text{for} \quad p \neq .5$$

Tables of binomial probabilities from n=2 to n=49 can be found in reference 34 and from n=50 to n=100 in reference 29.

Wilcoxon Test

The Wilcoxon test also known as Mann-Whitney or the U test) is used to test the null hypothesis that two samples come from identical populations. The two sets of sample values of sizes n_1 and n_2 are taken as if they were one sample of size n_1+n_2 and the values are arranged in order of increasing magnitude. The values are assigned ranks $1,2,3,\ldots, n_1+n_2$. We observe which ranks are occupied by each sample. If there are ties, each is assigned the rank which is the mean of the ranks they occupy, i.e., if ranks sixteen and seventeen are the same, each is assigned the value 16.5.

The rank information used to determine the value of U,

$$U = N_1N_2 + \frac{N_1 (N_1+1)}{2} - R_1$$

where n_1, n_2 are the sample sizes, R_1 is the sum of the ranks assigned to one of the samples, say the first.

For sample sizes n_1 and n_2 of less than eight, special tables of the U distribution are required as in reference 26 . For n_1 and n_2 values both greater than eight, a normal approximation to U can be used. Then,

H_0: $\mu_1 = \mu_2$

H_1: $\mu_1 > \mu_2$; or $\mu_1 < \mu_2$;or $\mu_1 \neq \mu_2$

The test statistic is

$$Z = \frac{U - E(U)}{\sqrt{Var\ (U)}}$$

where $E\ (U) = \frac{(n_1\ n_2\ )/2}{2}$

$$Var(U) = \frac{n_1n_2(n_1+n_2+1)}{12}$$

The critical regions are

$Z \leq -Z_\alpha$ for $\mu_1 > \mu_2$

$Z \geq Z_\alpha$ for $\mu_1 < \mu_2$

$|Z| \geq Z_{\alpha/2}$ for $\mu_1 \neq \mu_2$

Verification and Validation

In the discussion of model building in Section II, we concluded that a simulation model is an approximate representation of reality and, in fact, an abstraction of the real world. The analyst still must determine how accurately the model portrays reality. He hears the recurring questions "Is the model valid?" and "Is your model telling me the truth?". Now let's examine the verification and validation process which has been called "the most elusive of all the unresolved problems associated with computer simulation" [24, p.310].

First, we will define some basic terminology. Verification is ensuring that a simulation model behaves as the analyst intends. Validation is testing the agreement between the behavior of the simulation model and the real world system [2].

Verification is not a predetermined number of steps but a continuing effort throughout the model building process to ensure that the model behaves as intended. In Figure 9 verification of the mathematical model and the computer program are discretely represented in the flowchart. For stochastic models, one suggested method of verification is to replace all probabilistic effects of the model by a deterministic sequence (constants) so that hand computations of expected results can be compared to model output [2]. Also, using the

statistical tests discussed in this section, the stochastic model output can be compared to theoretical or historical data to determine the existence of significant differences. Throughout the verification process the analyst is checking, rechecking, and debugging the simulation model until he is assured that it performs the way he intends it.

Once the internal consistency of the model is established, the analyst can begin the complex and everlasting process of validation. Face validity is the surface impression of a simulation's realism and is obtained by asking people who know the real system [7]. In actuality it is an examination of the credibility or reasonableness of the model. The "expert" on the real system could compare real world and simulation model output to see if he can differentiate between them. However, the analyst must be cautious since Turing has shown that one can never prove two finite state systems "are identical simply by comparing a finite sample of input-output transformations." [2] Some one knowledgeable about the real world system could also examine the model in detail and review its structure and parameters.

Parameter Validity is the agreement between the values of variables and parameters in the model and their real world counterparts. Sensitivity analysis, where variations

are made in the values of parameters to determine the effect on output, is one test of parameter validity. Factors which are insensitive to variation do not need to be closely estimated. As part of verification, the parameters should have been accurately estimated from existing data. Various statistical tests, such as Chi Square, Kolmogorov-Smirnov, and analysis of variance, can be used to evaluate how well the parameters fit the real world process.

Validity of functional relationships can be partly examined when face validity is checked. But more detailed investigation is required to determine if the subsystem models are valid. Simple empirical tests on means and variances or more complex statistical tests may be required. If a real world system is available, some special data collection may be arranged to acquire the necessary data base for the submodels.

As we have emphasized on several occasions, the decision maker should be actively involved throughout the model building process. In the validation process, he is able to explore the interactions in the simulation model and determine if there is sufficient agreement between the model output and actual data.

The following quotation indicates the elusiveness of validation alluded to in the opening remarks. "It is never

possible to completely validate a decision-aiding model since there is never real data about alternatives not implemented; this problem is common to any decision-aiding procedure, not only to simulation. If the decision makers believe the model is useful and use it, the analyst has done his job"[7, p. 206]. Well, he has almost done his job. One significant point not mentioned in the above quotation is the analyst's inherent obligation to advise the decision maker of the assumptions, limitations, or simplifications in the model. These should be clearly documented for the decision maker as well as future users of the model.

SECTION VI

A SIMULATION MODEL FOR A CLASSICAL PROBLEM

In this section we will develop a simulation model to compare two different strategies for the repair of a system of machines. In doing so, we will utilize the concepts of computer simulation advanced in this report and apply some techniques of statistical analysis.

Consider a single system where a repairman is assigned to service a group of N semi-automatic machines which fail intermittently. We shall assume that the failure rates of the machines are identical, the event of failure for any machine is independent of the state of any other machine, and the time between failures of a single machine and the repair time are distributed negative exponentially with respective means λ and μ. The machine layout is a rectangular pattern with two rows of the same number of machines. The repairman requires time T to walk between any two adjacent machines.

Two strategies for tending the machines will be considered:

(1) Unidirectionally patrol the perimeter of the machines repairing any failed machine, or

(2) Repair the machines in the order of their failure with the repairman located at a central point when no

machines are failed. In the latter strategy, we will assume that if a machine fails while the repairman is enroute to the idle location, the repairman continues to the central location before starting to the machine. In the patrolling strategy, if a machine he has just passed fails before he reached the next machine, the repairman cannot turn around and return to the failed machine.

Development of the Model

The general structure of the simulation is shown in the block diagram of Figure 28 . The primary difference between the models for the two strategies is the time flow mechanism. For the patrolling repairman, the Constant Increment Algorithm from reference [23] was used as a basis for the time flow mechanism. The CI algorithm was shown to be very efficient for this particular problem. For the centrally located repairman, a next event method was devised.

The following variables are defined for use in the simulation:

TDT - total time down for all machines

EFF - total system efficiency

DWAIT(J) - total time machine J is down awaiting repair

DMT(J) - individual time down for machine J

Total system efficiency is further defined as the total

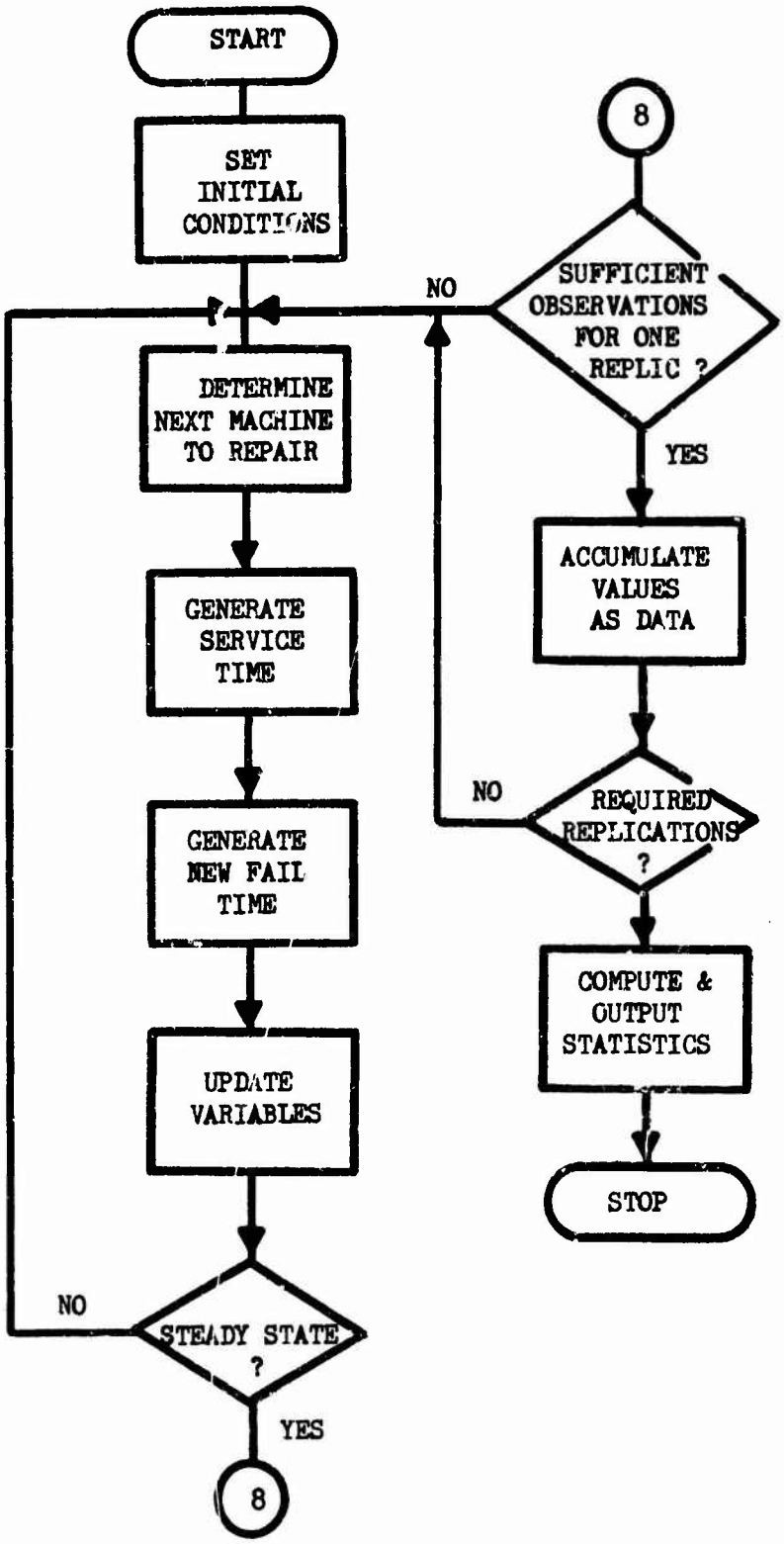


Figure 28. Flowchart For Machine Interference Problem

operating time for all machines divided by total possible operating time. The efficiency is then expressed as a percentage.

The simulation program consisted of the following subprograms:

(1) MAIN program read the input parameters lambda, mu, number of machines, model to be used, initial value for the random number generator, number of repairs to be used in each block of data (stratified sampling is used), number of blocks, indicators for determining if steady state is to be found and the epsilon to be used in comparing consecutive blocks. MAIN also calls the initialization subroutine and selects the time flow mechanism to be used.

(2) PATROL- is the time flow mechanism for the patrolling repairman which determines the next machine to be repaired, calls the event processing subroutine, and checks if the necessary number of replications have been completed.

(3) FIFO- is the time flow mechanism for the centrally located repairman, and its functions are similar to PATROL. If ties for failure times exist, they are broken by choosing the closest machine, or, if they remain tied, by lowest machine number.

(4) MATH- is the event processing subroutine which determines a service time and new failure time, calculates wait time and down time, and updates the system variables.

If steady state has not yet been reached, SSTEST is called to test the values of efficiency. If the number of repairs in the block equals the number required, the STAT subroutine is called to store the values of the variables.

(5) SET- is the initialization subroutine which sets the necessary variables to zero, initially assigns failure times to all machines, and calls the TMATRX subroutine for the centrally located repairman for the distances between machines. At the end of each block SET is called again and entered at the multiple entry point ZERO to initialize the appropriate variables.

(6)= SSTEST- is the subroutine to determine when steady state is reached. The last three efficiencies are stored and th their differences compared according to the steady state criterion discussed later. The steady state clock time, last three efficiencies, and number of repairs are printed out. The appropriate variables are initialized and data collection for simulation can begin.

(7) TMATRX - is the subroutine used for the centrally located repairman to calculate the distances between machines and between machines and the central point.

(8) RANG- is the random number generator function. The multiplicative congruential generator with multiplier 5^{15} and was also tested with an initial value of 1907 using

the Pseudorandom Number Generator Test Package described by Overstreet, et al [25]. It passed all tests satisfactorily including very good results on the spectral test.

(9) EXTIME - is the function to determine negative exponentially distributed random variates with a given mean failure time.

(10) ACOVAR- is the subroutine to calculate covariance term C between consecutive blocks of data.

(11) STAT- is the statistics subroutine which accumulates values of the variables at the end of each block, and calls ACOVAR if the necessary number of blocks have been reached. At end of the required number of replications, STAT is entered at the multiple entry point FINISH to compute the final statistics using stratified sampling.

Modular development of the simulation program was achieved without difficulty. Virtually all subroutines are common to the simulation of each strategy. The major difference is in the time flow mechanisms which were expressly chosen and tailored for the particular strategy. The other difference is in the initialization subroutine where the distances between various locations must be generated for the centrally located repairman.

SIMULATION ANALYSIS IN FORTRAN IV

MACHINE INTERFERENCE PROBLEM

INPUT INSTRUCTIONS: THE VALUES OF THE FOLLOWING PARAMETERS SHOULD BE TYPED IN ORDER ON A DATA CARD ACCORDING TO FORMAT 1.

```
C     LAMBIA - (REAL) MEAN FAILURE TIME
C     MU     - (REAL) MEAN SERVICE TIME
C     N      - (INTEGER) NUMBER OF MACHINES
C     T      - (REAL) TIME REQUIRED TO WALK BETWEEN MACHINES
C     MODEL  - (INTEGER) 1 - PATROLLING REPAIRMAN
                         2 - CENTRALLY LOCATED REPAIRMEN
C     INITIAL- (INTEGER) INITIAL VALUE TO BE USED IN RANDOM NUMBER
               GENERATOR
C     ITEST  - (INTEGER) 0 - IF NO TEST FOR STEADY STATE REQUIRED
                         1 - IF A TEST FOR STEADY STATE DESIRED
C     EPSLON - (REAL) VALUE TO BE USED IN DETERMINING STEADY STATE
               REACHED WHEN 3 CONSECUTIVE VALUES DIFFER BY NO MORE
               THAN EPSLON
C     NBLOCK - (INTEGER) NUMBER OF OBSERVATIONS PER BLOCK
C     IAUTO  - (INTEGER) 0 - IF NO CALCULATIONS OF COVARIANCE REQUIRED
                         N - SOME NONZERO VALUE OF THE NUMBER OF BLOCKS
                             TO BE USED IN CALCULATION OF COVARIANCE
                             BETWEEN CONSECUTIVE BLOCKS, FOR EXAMPLE, 50.
C     NREPET - (INTEGER) NUMBER OF BLOCKS (REPLICATIONS) OF SIZE NBLOCK
               TO BE USED IN OVERALL SIMULATION ANALYSIS
             - (REAL) VALUE OF COVARIANCE BETWEEN CONSECUTIVE BLOCKS

C     DEFINITION OF VARIABLES:

C     CLOCK - MASTER CLOCK TIME FROM BEGINNING OF SIMULATION RUN
C     TSTART- TIME WHEN DATA COLLECTION BEGAN IN CURRENT BLOCK OF EVENTS

C     FOR EACH BLOCK:

C     IBLOK - BLOCK COUNTER
C     TDT   - TOTAL TIME DOWN FOR ALL MACHINES
C     EFF   - TOTAL SYSTEM EFFICIENCY
C     DWAIT(J) - TOTAL TIME MACHINE J IS DOWN AWAITING REPAIR
C     DMT(J)- INDIVIDUAL TIME DOWN FOR MACHINE J
C     DIST(I,J) - TIME REQUIRED TO WALK FROM MACHINE I TO MACHINE J
```

```
C       *****  MAIN PROGRAM  *****
C
C
        COMMON/C1/N,FAIL(18),T,CLOCK,TSTART,NBLOCK,NREPET,IBLOK
       1      /C2/TDT,EFF,DWAIT(18),DMT(18),EF(3),KOUNT,IAUTO,C
       2      /C3/DIST(19,19)
       4      /C4/ LAMBDA,MU,ITEST,MODEL,EPSLON
        REAL LAMBDA,MU
C       READ INPUT PARAMETERS
  100   READ(5,1,END=99) LAMBDA,MU,N,T,MODEL,INITAL,ITEST,EPSLON,NBLOCK,
      1   IAUTO,NREPET C
      1 FORMAT(2F5.1,I5,F5.1,3I5,F5.3,3I5,F5.3,3I5,F5.3)
        WRITE (6,2) LAMBDA,MU,N,T,MODEL,INITAL,ITEST,EPSLON,NBLOCK,
       1   IAUTO,NREPET,C
      2 FORMAT ('INPUT:',F5.1,5X,F5,1,5X,I2,5X,F4,0,5X,I8,5X,
       1  I2,5X,F3,2,5X,I2,5X,I2,5X,I3,5X,F5.3)
C       INITIALIZE RANDOM NUMBER GENERATOR AND VARIABLES
        CALL RAN(INITIAL)
        CALL SET
C       BRANCH TO APPROPRIATE TIME FLOW MECHANISM
        IF(MODEL - 1) 10,10,20
   10   CALL PATROL
        GO TO 100
   20   CALL FIFO
        GO TO 100
   99   STOP
        END


C           *****     TIME FLOW MECHANISM FOR      *****
C           *****   PATROLLING REPAIRMAN STRATEGY  *****

C       SUBROUTINE PATROL
        COMMON/C1/N,FAIL(18),T,CLOCK,TSTART,NBLOCK,NREPET,IBLOK
       1      /C2/TDT,EFF,DWAIT(18),DMT(18),EF(3),KOUNT,IAUTO,C
        J = 0
        TEST FOR END OF REPLICATIONS
       1 IF(BLOK.GT.NREPET) GO TO 99
       2 J = J + 1
        IF(J.GT.N) J = 1
        CLOCK = CLOCK + T
        IF(CLOCK.LT.FAIL(J)) GO TO 2
C       K = J
        CALL MATH SUBROUTINE TO UPDATE VALUES OF VARIABLES
        CALL MATH(K)
        GO TO 1
C       WHEN ALL REPLICATIONS COMPLETED CALL SUBROUTINE FOR CALCULATIONS
        OF FINAL STATISTICS
```

```
   99 CALL FINISH
      RETURN
      END

C                  *****  TIME FLOW MECHANISM FOR  *****
C                *****  CENTRALLY LOCATED REPAIRMAN  *****
C
C
      SUBROUTINE FIFO
      COMMON /C1/ N,FAIL(18),T,CLCOK,TSTART,NBLOCK,NREPET,IBLOK
     1       /C2/ TDT,EFF,DWAIT(18),DMT(18),EF(3), KOUNT,IAUTO,C
     2       /C3/ DIST(19,19)
      N1 = N + 1
      K = N1
C     TEST FOR END OF REPLICATIONS
     1 IF(IBLOK.GT.NREPET) GO TO 99
       DETERMINE MINIMUM FAILURE TIME, IF A TIE, USE CLOSEST ONE,
     2 FMIN = 2.**30
       DO 10 J = 1,N
       IF(FAIL(J) -EMIN) 5.3.10
     3 IF(DIST(K,JMIN).LE.DIST(K,J)) GO TO 10
     5 FMIN = FAIL(J)
       JMIN = J
    10 CONTINUE
C
C      IF FAILED ALREADY, GO TO THAT MACHINE:  OTHERWISE GO TO CENTRAL
C      LOCATION.
       IF(CLOCK.GE.FMIN) GO TO 20
       CLOCK = CLOCK + DIST (K,N1)
       K = N1
       IF(CLOCK.GE.FMIN) GO TO 20
       CLOCK = FMIN
C
C      UPDATE CLOCK
    20 CLOCK = CLOCK + DIST(K,JMIN)
       K = JMIN
C
C    UPDATE VARIABLES
     CALL MATH (K)
     GO TO 1
  99 CALL FINISH
     RETURN
     END
```

```
C               *****  EVENT PROCESSING ROUTINE  *****
C
C
      SUBROUTINE MATH(K)
      COMMON/C1/N,FAIL(18),T,CLOCK,TSTART,NBLOCK,NREPET,IBLOK
     1      /C2/TDT,EFF,DWAIT(18),EF(3),KOUNT,IAUTO,C
     4      /C4/LAMBDA,MU,ITEST,MODEL,EPSLON
      REAL,LAMBDA, MU
C     DETERMINE SERVICE TIME
      ST = EXTIME(MU)
C
C     DETERMINE TIME MACHINE WAITED FOR REPAIRMAN
      WAIT = CLOCK - FAIL(K)
C     DETERMINE DOWNTIME AND UPDATE TOTAL DOWNTIME FOR ALL MACHINES
      DOWN = WAIT + ST
      TDT = TDT + DOWN
C
C     UPDATE CLOCK AND VARIABLES
      CLOCK = CLOCK + ST
      EFF = (N*(CLOCK - TSTART) - TDT) / (N*(CLOCK - TSTART)
     1    )*100.
      DWAIT(K) = DWAIT(K) + WAIT
      DMT(K) = DMT(K) + DOWN
C
C     CALCULATE NEW FAILURE TIME
      FT = EXTIME(LAMBDA)
      FAIL(K) = CLOCK + FT
C     INCREMENT COUNTER FOR NUMBER OF REPAIRS
      KOUNT = KOUNT + 1
C
C     IF REQUIRED, CALL STEADY STATE TEST SUBROUTINE
      IF (ITEST.LE.0) GO TO 99
C     IF STEADY STATE HAS BEEN REACHED, CALL STATISTICS SUBROUTINE
      CALL SSTEST
   99 IF(ITEST.LE.0. AND. KOUNT.GE.NBLOCK)  CALL STAT
      RETURN
      END

C               ******  VARIABLE INTIALIZATION  ******
C               ******        SUBROUTINE        ******
C
C
      SUBROUTINE SET
      COMMON/C1/N,FAIL(18),T,CLOCK,TSTART,NBLOCK,NREPET,IBLOK
     1      /C2/TDT,EFF,DWAIT(18),DMT(18),EF(3),KOUNT,IAUTO,C
     4      /C4/ LAMBDA,MU,ITEST,MODEL,EPSLON
      REAL LAMBDA, MU
```

```
C     INITIAL ENTRY POINT TO INITIALIZE VARIABLES
      CLOCK = 0
      IBLOK = 1
      DO 10 K = 1 N
   10 FAIL(K) = EXTIME (LAMBDA)
C     IF MODEL 2 USED  CALL SUBROUTINE TO CALCULATE VARIOUS DISTANCES
C     BETWEEN MACHINES
      IF(MODEL.LE.1)  GO TO 11
      CALL TMATRX(N.T)
C
C
C     SUBSEQUENT ENTRY POINT TO ZERO VARIABLES IN EACH BLOCK
      ENTRY ZERO
C
   11 TSTART = CLOCK
      EF(1)  = 199.
      EF(2)  = 299.
      EF(3)  = 399.
      KOUNT  = 0
      TDT    = 0.
      EFF    = 0.
      DO 15 K= 1,N
      DWAIT(K)=0.
   15 DMT(K) = 0.
      RETURN
      END

C             *****  STEADY STATE TEST  *****
C
C
      SUBROUTINE SSTEST
      COMMON/C1/N,FAIL(18),T,CLOCK,TSTART,NBLOCK,NREPET, IBLOK
     1      /C2/TDT,EFF,BWAIT(18),DMT(18), EF(3), KOUNT,IAUTO,C
     4      /C4/LAMBDA,MU ITEST MODEL EPSLON
      DATA I/0/
C     STORE LAST THREE EFFICIENCIES
      I = I + 1
      IF(I.GT. 3) I = 1
      EF(I) = EFF
C
C     IF ALL OF LAST THREE EFFICIENCIES DIFFER BY EPSLON OR LESS,
C     STEADY STATE REACHED.
      IF (ABS(EF(1)-EF(2)).LE.EPSLON. AND. ABS(EF(2) - EF(3))
     1  .LE.EPSLON.AND.ABS(EF(1) - EF(3)).LE.EPSLON)
     2  GO TO 90
       GO TO 99
   90 ITEST = -1
```

```
C
C       PRINT INFORMATION ON STEADY STATE
        WRITE(6,91)-CLOCK, EF(1), EF(2), EF(3), KOUNT
   91   FORMAT(1HO'STEADY STATE AT ',F15.3,'CLOCK TIME',5X,'EFFICIENCIES
      1 :',3(F8,3,3X),'AND',I10,'REPAIRS')
        CALL ZERO
   99   RETURN
        END


        SUBROUTINE TMATRX(N,T)
C       ***** SUBROUTINE FOR CALCULATION OF *****
C       ***** DISTANCES BETWEEN MACHINES    *****
C
C
        COMMON/C3/DIST(19,19)
C       CENTRAL LOCATION IS POSITION N+1, WHERE N IS NUMBER OF MACHINES
        N1 = N + 1
        DO 10 I = 1, N
        DO 10 J = I, N
        K = N/2
C       TEST FOR MACHINES IN SAME ROW: OTHERWISE USE DISTANCE FORMULA
        IF(I.LE.K.AND.J.GT.K) GO TO 5
        C = J - I
        GO TO 6
      5 C = SQRT((N = 1 - J - I)**2 + 1)
C       MULTIPLY DISTANCE BY TIME BETWEEN MACHINES
      6 DIST(I,J) = C * T
     10 DIST(J,I) = DIST(I,J)
C
C       CALCULATE DISTANCE TO CENTRAL LOCATION
        DO 15 J = 1,K
        DIST(N1,J) = T*SQRT((J-N/4-1)**2+,25)
        M = J + K
   15   DIST(N1,M) = DIST(N1,J)
        D) 20 J = 1,N
   20   DIST(J,N1) = DIST(N1,J)
        RETURN
        END


C               ***** SUBROUTINE FOR COVARIANCE *****
C               ***** BETWEEN CONSECUTIVE BLOCKS *****
C
C
      SUBROUTINE ACOVAR
      COMMON/C2/TDT, EFF, DWAIT(18), DMT(18), EF(3), KOUNT, IAUTO,C
```

```
     1     /C5/BTDT(100), BEFF(100), BDWAIT(100,18), BDMT(100,18)
      DIMENSION X(100),COVAR(100)
      XBAR = 0.
C     COPY VALUES OF VARIABLE IN TO X ARRAY.
      DO 5 I=1,IAUTO
  10  XBAR = SBAR + X(I)
      AVG = 0.
      SBAR = SBAR/IAUTO
      N = IAUTO - 1
C
C     CALCULATE COVARIANCE TERM C BETWEEN CONSECUTIVE BLOCKS
      DO 15 I = 1,N
      COVAR(I) = (X(I) - XBAR)*(X(I-1)-XBAR)
 15   AVG = AVG + COVAR(I)
      AVG = AVG/N
C     PRINT CALCULATIONS
      WRITE(6,20)  (COVAR(I), I=1,N), AVG
  20  FORMAT((1HO,10(1X,F11.8)))
      RETURN
     END

C
C        ***** STATISTICAL CALCULATIONS SUBROUTINE *****
C
C
C
C
C
C
      SUBROUTINE STAT
      COMMON  /C1/N,FAIL(18),T,CLOCK,TSTART,NBLOCK,NREPET,IBLOK
     1        /C2/TDT,EFF,DWAIT(18),DMT(18),EF(3),KOUNT,IAUTO,C
     2        /C3/DIST(19,19)
     4        /C4/LAMBDA,MU,ITEST,MODEL,EPSLON
     5        /C5/BTDT(100),BEFF(100),BDWAIT(100,18),BDMT(100,18)
      REAL LAMBDA MU
      DIMENSION AWAITL(18),ADMT1(18),AWAIT2(18),ADMT2(18),AWAIT(18),
     1    VWAIT(18),SWAIT(18),VWAIT1(18),VWAIT2(18), VDMT(18),
     2    VDMT (18), VDMT2(18), SDMT(18), ADMT(18)
C
C     ASSIGN VALUES TO VARIABLES FROM EACH BLOCK
      BTDT(IBLOK) = TDT
      DEFF(IBLOK) = EFF
      DO 10 I=1,N
      BDWAIT(IBLOK I) = DWAIT(I)
  10  BDMT (IBLOK, I) = DMT(I)
```

```
C
C     CALL COVARIANCE SUBROUTINE IF APPROPRIATE
      IF (IAUTO.GT.0.AND.IBLOK.GE.IAUTO)  CALL ACOVAR
C     INCREMENT BLOCK COUNTER
      IBLOK = IBLOK + 1
C
C     ZERO VARIABLES FOR NEXT BLOCK
      CALL ZERO
      RETURN
C
C     ENTRY POINT FOR FINAL CALCULATIONS
      ENTRY FINISH
      ATDT1 = 0.
      ATDT2 = 0.
      AEFF1 = 0.
      AEFF2 = 0.
      VTDT1 = 0.
      VTDT2 = 0.
      VEFF1 = 0.
      VEFF2 = 0.
      IBLOK = IBLOK - 1
      IF(IAUTO.GT.0) RETURN
      IHALF = IBLOK/2
C
C     CALCULATE MEAN TOTAL TIME DOWN AND EFFICIENCY FOR EVEN AND ODD
C     BLOCKS AND OVERALL MEAN
      DO 20 I=1, IBLOK,2
      ATDT1 = ATDT1 + BTDT(I)
      ATDT2 = ATDT2 + BTDT (I+1)
      AEFF1 = AEFF1 + BEFF (I + 1)
   20 AEFF2 = AEFF2 + BEFF (I+1)
      ATDT1 = ATDT1/IHALF
      ATDT2 = ATDT2/IHALF
      AEFF1 = AEFF1/IHALF
      AEFF2 = AEFF2/IHALF
      ATDT = (ATDT1 + ATDT2)/2,
      AEFF = (AEFF1 = AEFF2)/2.
C
C     COMPUTE VARIANCES FOR EVEN AND ODD BLOCKS AND OVERALL VARIANCES
      DO 30 I=1,IBLOK,2
      VTDT1 = VTDT1 + (BTDT(I) - ATDT1)**2
      VTDT2 = VTDT2 + (BTDT(+1) - ATDT2)**2
      VEFF1 = VEFF1 + (BEFF(I) - AEFF1)**2
   30 VEFF2 = VEFF2 + (BEFF(I+1) - AEFF1)**2
      VTDT2 = VTDT2/IHALF
      VTDT1 = VTDT1/IHALF
      VEFF1 = VEFF1/IHALF
      VEFF2 = VEFF2/IHALF
      VEFF = .25*(VEFF1 + VEFF2) +C*2.*(IBLOK-1)/IBLOK**2
```

```
      VTDT =. 25*(VTDT.+VTDT2) +C*2. *(IBLOK-1)/IBLOK**2
C     DETERMINE STANDARD DEVIATIONS
      SEFF = SQRT(VEFF)
      STDT = SQRT(VTDT)
C
C     CALCULATE MEAN WAITING TIME AND TOTAL TIME DOWN FOR EACH MACHINE
C     FOR EVEN AND ODD BLOCKS AND OVERALL
      DO 40 J = 1,N
      AWAIT1(J) = 0.
      AWAIT2(J) = 0.
      ADMT1(J) = 0.
      ADMT2(J) = 0.
      DO 40 I = 1, IBLOK,2
      AWAIT1(J) = AWAIT1(J) + BDWAIT(I,J)
      AWAIT2(J) = AWAIT2(J) + BDWAIT(I+1,J)
      ADMT1(J) = ADMT1(J) + BDMT(I,' J)
   40 ADMT2(J) = ADMT2(J) + BDMT(I+1,J)
      DO 50 J=1,N
      AWAIT1(J) = AWAIT1(J)/IHALF
      AWAIT2(J) = AWAIT2(J)/IHALF
      ADMT1(J) = ADMT1(J)/IHALF
      ADMT2(J) = ADMT2(J)/IHALF
      AWAIT(J) = (AWAIT1(J)/IHALF
   50 ADMT(J) = (ADMT1(J) + ADMT2(J))/2
C
C     COMPUTE VARIANCES AND STANDARD DEVIATIONS
      DO 60 J=1,N
      DO 60 I=1,IBLOK,2
           VWAIT1(J) = VWAIT1(J) +(BDWAIT(I,J) - AWAIT1(J))**2
           VWAIT2(J) = VWAIT2(J) +(BDWAIT(I+1,J) - AWAIT2(J))**2
      VDMT1(J) = VDMT1(J) + (BDMT(I,J)-ADMT1(J))**2
   60 VDMT2(J) = VDMT2(J) + (BDMT(I+1,J)-ADMT2(J))**2
      DO 70 J=1N
      VWAIT1(J) = VWAIT1(J)/IHALF
      VWAIT2(J) = VWAIT2(J)/IHALF
      VDMT1(J) = VDMT1(J)/IHALF
      VDMT2(J) = VDMT2(J)/IHALF
      VWAIT(J) = .25*(VWAIT1(J) + VWAIT2(J))+C*2,*(IBLOK-1)/IBLOK**2
      SWAIT(J) = SQRT(VWAIT(J))
      VDMT(J) = .25*(VDMT1(J) + VDMT2(J))+C*2(IBLOK-1)/IBLOK**2
   70 SDMT(J) = SQRT(VDMT(J))
C
C     PRINT OUTPUT DATA AND STATISTICS
      WRITE (6,80) MODEL,LAMBDA MU, ATDT,STDT,AEFF,SEFF,
     1    CLOCK, IBLOK,NBLOCK
   80  FORMAT(1H)/1HO,38X,'OUTPUT VALUES FOR MODEL',I2,'WITH LAMBDA',
     1  '=',FY,1,'AND MU = ',F5.1///1HO,45X,'MEAN',15X,
     2                 'STANDARD DEVIATION'/1HO,'TOTAL TIME DOWN',
     3  'FOR ALL MACHINES',8X,F15,3,13X,F11.3/1HO,'TOTAL',
```

```
     4  'SYSTEM EFFICIENCY',17X,F15.3,6X,F15.3,13X/1HO,/1HO,
     5  'FINAL CLOCK TIME', 13X,F25.3/ 1HO, NUMBER OF REPLICATIONS',
     6  18X,I10/ 1HO,'NUMBER OF REPAIRS PER REPLICATION',7X, I10)
      PRINT 90
   90 FORMAT(1HO,'MACHINE', 15X,'TIME DOWN',23X,'TIME DOWN AWAITING',
     1   'REPAIR'/18X,"MEAN',14X,'STD DEV',15X,'MEAN',13X,'STD DEV'/)
      DO 91 J = 1,N
   91 WRITE(6,92) J, ADMT(J),SDMT(J),AWAIT(J),SWAIT(J)
   92 FORMAT(3X,I3,2(10X,F10.3,9X,F10,3))
      RETURN
      END

C     *****  RANDOM NUMBER GENERATOR  *****
C
C
      FUNCTION RAND(DUMMY)
      I = IA * I
      FLD(0,1 I) = 0
      RANG = I /2.0**35
      RETURN
C
C     INITIALIZATION OF MULTIPLIER AND INITIAL VALUE
      ENTRY RAN( INITAL)
      I = INITAL
      IA = 5 ** 15
      RETURN
      END

C       *****   EXPONENTIAL INTEROCCURRENCE  *****
C       *****           TIME FUNCTION        *****
C
C
      FUNCTION EXTIME(MEAN)
      REAL MEAN
      EXTIME + - MEAN* ALOG(RANG(RANUM))
   10 RETURN
      END
```

Convergence to Steady State

The simulation for both strategies began with all machines working. The repairman began at machine one in the patrolling case and at the idle position for the centrally located strategy.

The total system efficiency was chosen as a variable which would be indicative of the model's convergence to steady state. The steady state criterion used was: steady state was reached when three consecutive values differed by no more than .01. Table 18 summarizes the steady state times and number of repairs for each set of parameters.

TABLE 18 CONVERGENCE TO STEADY STATE
$\mu = 30$

N	λ	T	PATROLLING			CENTRALLY LOCATED		
			CLOCK	REP	EFF	CLOCK	REP	EFF
6	600	10	17822	177	87.22	6579	62	91.58
	300	10	17332	285	75.05	11138	195	77.79
	600	100	50738	306	55.96	32921	194	53.32
	300	100	56784	411	36.47	64932	372	28.72
18	600	10	20019	404	63.38	32068	518	54.49
	300	10	9785	240	39.16	36497	572	27.34
	600	100	36665	280	26.15	262554	698	10.00
	300	100	31145	240	14.80	244965	397	6.58

Statistical Measurements

Statistical Measurements were obtained on total system efficiency, total time down for all machines, individual time down for each machine, and total time each machine was down awaiting repair. Total system efficiency was considered the variable of primary interest and the estimates for each strategy are given in Table 22. Sample model printouts are shown in Figures 29 and 30.

The first concern in data collection was the sample size, i.e., the number of repairs contained in each replication. Several techniques are available for determining length of the sample needed for an adequate statistical analysis. Fishman [9] proposed a method for finding the number of equivalent independent observations using the spectral density function. A detailed evaluation of the mathematics involved, length of computer program required, and run time of the program led to the conclusion that for the length of the simulation required for this study, the spectral density approach was not appropriate. The determination of a sample size would involve calculations for lags up to 500 or more which alone would be several times the total run time of the entire simulation for all sets of parameters. For simulation models where each replication is very lengthy and costly, the Fishman technique would be valuable.

INPUT: 300.0 30.0 10 10. 2 1907 1 .01 600 0 50 .356

STEADY STATE AT 36497.601 CLOCK TIME EFFICIENCIES : 27.344 27.339 27.337 AND 972 REPAIRS

OUTPUT VALUES FOR MODEL 2 WITH LAMBDA = 300.0 AND MU = 30.0

	MEAN	STANDARD DEVIATION
TOTAL TIME DOWN FOR ALL MACHINES	499068.760	11962.313
TOTAL SYSTEM EFFICIENCY	26.516	.962

FINAL CLOCK TIME 1922618.900

NUMBER OF REPLICATIONS 50

NUMBER OF REPAIRS PER REPLICATION 600

MACHINE	TIME DOWN MEAN	TIME DOWN STD DEV	TIME DOWN AWAITING REPAIR MEAN	TIME DOWN AWAITING REPAIR STD DEV
1	28153.197	1271.642	27157.300	1243.236
2	27526.644	1061.468	26559.636	1045.986
3	27814.404	1090.059	26834.307	1058.974
4	27636.343	1157.792	26622.578	1079.449
5	27822.502	1189.140	26846.135	1151.453
6	27779.429	1261.715	26784.030	1225.039
7	27935.379	1046.556	26940.338	1033.910
8	27565.324	1311.452	26557.672	1291.120
9	27692.809	1114.726	26528.737	1049.447
10	27914.535	1304.924	26956.845	1274.363
11	27891.706	1006.917	26872.372	970.553
12	27667.834	1051.213	26817.322	1021.167
13	27471.529	1050.775	26512.159	1039.170
14	27541.617	1135.039	26521.853	1083.240
15	27613.378	1092.046	26612.955	1041.190
16	27622.740	1121.422	26596.286	1101.502
17	27822.921	1149.942	26807.775	1137.101
18	27739.556	1053.022	26774.533	1033.270

Figure 29. Sample Printout For Centrally Located Repairman

Reproduced from best available copy.

INPUT: 300,0 30,0 16 10, 1 1907 1 ,01 600 0 50 ,360

STEADY STATE AT 9785,147 CLOCK TIME EFFICIENCIES : 39,163 39,159 39,158 AND 240 REPAIRS

OUTPUT VALUES FOR MODEL 1 WITH LAMBDA = 300,0 AND MU = 30.0

	MEAN	STANDARD DEVIATION
TOTAL TIME DOWN FOR ALL MACHINES	269330,270	14083,946
TOTAL SYSTEM EFFICIENCY	40,088	1,562

FINAL CLOCK TIME 1257965,800

NUMBER OF REPLICATIONS 50

NUMBER OF REPAIRS PER REPLICATION 600

MACHINE	TIME DOWN MEAN	TIME DOWN STD DEV	TIME DOWN AWAITING REPAIR MEAN	TIME DOWN AWAITING REPAIR STD DEV
1	15186,575	925,744	14195,668	907,800
2	14928,552	998,333	13908,969	951,730
3	14795,940	1010,801	14030,445	985,819
4	14674,326	1140,821	13682,703	1086,172
5	15040,923	932,185	14034,323	893,633
6	14960,096	866,173	13949,356	810,517
7	15202,458	1002,096	14227,884	985,913
8	15044,753	993,318	14067,224	973,114
9	14719,350	1110,320	13732,780	1044,424
10	15000,742	1140,486	13971,772	1076,101
11	14957,498	933,344	13917,574	914,616
12	14827,517	1198,650	13846,189	1185,568
13	14996,757	1060,293	14021,137	1061,035
14	15113,539	924,649	14093,268	912,929
15	15002,376	902,923	13963,560	876,083
16	14906,938	1019,442	13933,955	972,351
17	14834,519	896,837	13880,007	859,686
18	14934,938	972,629	13892,718	970,077

Figure 30. Sample Printout For Patrolling Repairman

Reproduced from best available copy.

As a first step in designing a method of sample size selection, the values of the autocorrelation function were examined for lags of 100 to 600 for the patrolling strategy with $\lambda = 600$, $\mu = 30$, N=6, and T=10. Some values are displayed in Table 19 below.

TABLE 19. VALUES OF AUTOCORRELATION FUNCTION

LAG	AUTOCORRELATION
100	.868
150	-.062
200	-.047
250	-.098
300	-.024
350	-.017
400	.057
450	.055
500	.048
550	-.028
600	-.045

It is clear that the autocorrelation function does not asymptotically approach zero but exhibits some cyclic behavior. The autocorrelation closest to zero was -.0002 which occurred for a lag of 533. Since in the simulation model, we desire independence between replications (blocks of observations of some fixed size) or at least a small covariance, the ACOVAR subroutine was used to determine covariance between consecutive blocks. From a preliminary check of various values, the covariance between nonadjacent blocks was always several orders of magnitude less than the covariance between adjacent blocks. Obviously, if the covariance between adjacent blocks was made sufficiently small, the covariance for nonadjacent blocks could be

considered zero. Some values of covariance for various sample sizes for λ =600, μ=30, n=6, T=10 are presented in Table 20.

TABLE 20 COVARIANCE FOR VARIOUS SAMPLE SIZES
50 BLOCKS

SAMPLE SIZE EACH BLOCK	PATROLLING	CENTRALLY LOCATED
100	-.105	-.191
200	.041	.197
300	-.027	.062
400	.042	.021
500	.042	-.038
600	-.088	-.015
700	-.021	-.033
800	-.005	-.028

After a comparison of various combinations of sample sizes for all the parameter sets, a size of 600 repairs was chosen as producing the lowest covariances for all sets. By using the same sample size for all simulation runs, one is able to make a more valid comparison. The covariances of adjacent blocks for a sample size of 600 and six machines for various parameters are displayed in Table 21.

TABLE 21 COVARIANCE ESTIMATES
N=6, μ=30, 50 BLOCKS

λ	T	PATROLLING	CENTRALLY LOCATED
600	10	-.089	-.015
300	10	-.026	-.113
600	100	.095	.255
300	100	.196	-.028

After steady state was reached, the simulation run was broken into replications (blocks). Each block contained 600 repairs and the entire run was comprised of 50 blocks.

Fifty was chosen so that large sample statistics could be later applied in a comparison of the results of each strategy. The blocks were classified as odd or even and a value of each variable stored for every block. A mean and variance was then calculated for the odd blocks and for the even blocks. A pooled mean and variance was determined. The pooled variance was calculated from

$$\mathrm{Var}\left(\frac{X+Y}{2}\right) = 1/4\ (\mathrm{Var}\ X + \mathrm{Var}\ Y + \mathrm{Cov}(X,Y))$$

$$= 1/4\ (\mathrm{Var}\ X + \mathrm{Var}\ Y) + 1/4\ \mathrm{Cov}(X,Y)$$

$$= 1/4\ (\mathrm{Var}\ X + \mathrm{Var}\ Y) + \frac{2}{n^2} \sum_i \sum_j (X_i - \bar{X})\ (Y_j - \bar{Y})$$

$$= 1/4\ (\mathrm{Var}\ X + \mathrm{Var}\ Y) + \frac{2(n-1)}{n^2}\ C$$

where C is the covariance term between adjacent blocks previously determined. The covariance between nonadjacent blocks was assumed to be zero.

The estimates of the total system efficiency for each set of parameters and strategy are shown in Table 22 below.

TABLE 22 SUMMARY OF EFFICIENCY ESTIMATES

$\mu = 30$

N	λ	T	PATROLLING MEAN	PATROLLING STD DEV	CENTRALLY LOCATED MEAN	CENTRALLY LOCATED STD DEV
6	600	10	88.58	.54	91.68	.45
	300	10	75.76	.98	79.71	1.12
	600	100	57.05	.98	56.40	1.71
	300	100	35.74	.85	29.34	.94
18	600	10	67.08	1.33	52.95	1.71
	300	10	40.09	1.56	26.52	.96
	600	100	25.20	.77	9.25	.44
	300	100	12.82	.47	4.59	.32

Reliability of Estimates

Since we have purposely chosen the number of replications to be sufficiently large to use large sample (normal) statistics, we invoke the central limit theorem and determine confidence intervals for the estimates of total system efficiency. Thus, from the normal distribution we have

$$-Z_{\alpha/2} < \frac{\bar{X} - \mu}{S/\sqrt{n}} < Z_{\alpha/2}$$

and $$P\left(\bar{X} - Z_{\alpha/2}\frac{S}{\sqrt{n}} < \mu < \bar{X} + Z_{\alpha/2}\frac{S}{\sqrt{n}}\right) = 1-\alpha$$

where S is sample standard deviation

X is the estimate of total system efficiency

n is the sample size, in our case, 50

Assuming a significance level of .05, we have the confidence intervals shown in Table 23.

TABLE 23. 95% CONFIDENCE INTERVALS FOR ESTIMATES OF TOTAL SYSTEM EFFICIENCY

$\mu = 30$

			Patrolling		Centrally Located	
N		T	Lower Bound	Upper Bound	Lower Bound	Upper Bound
6	600	10	88.43	88.73	91.56	91.81
	300	10	75.49	76.03	79.39	80.02
	600	100	56.78	57.32	55.93	56.88
	300	100	35.50	35.97	29.08	29.60
18	600	10	66.71	67.45	52.47	53.42
	300	10	39.65	40.52	26.25	25.78
	600	100	24.99	25.42	9.13	9.37
	300	100	12.69	12.95	4.50	4.68

Comparison of Strategies

Since the samples were large enough for the central limit theorem to be invoked, one can test the difference between the estimates of total system efficiency for the two strategies for a given set of parameters and number of machines. Assume the samples are from two normal populations where the patrolling strategy has mean u_1 and the central location, mean u_2 with variances unknown. The statistic

$$Z = \frac{X_1 - X_2}{\sqrt{\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}}}$$

where X_1 is the estimate from the patrolling strategy,

X_2 is the estimate from the centrally located strategy,

S_1 and S_2 are the respective sample standard deviations,

$n_1 = n_2 = 50$.

has a standard normal distribution. The null hypothesis that $u_1 - u_2 = 0$ is tested against the alternative that $u_1 - u_2 \neq 0$. The critical region is $|Z| \geq Z_{\alpha/2}$. For $\alpha = .05$, one must have $|Z| \geq 1.96$. The statistic for each case is presented in Table 24. The null hypothesis is rejected in all cases. There is a significant difference in the means for the two strategies in all cases.

In the first two cases with six machines, one can see that the centrally located strategy produced a higher total system efficiency. For the remaining cases, the patrolling strategy had a higher total system efficiency. For the remaining cases, the patrolling strategy had a higher total system efficiency.

TABLE 24. TEST OF DIFFERENCE OF MEANS
TOTAL SYSTEM EFFICIENCY
$\mu = 30$

N	λ	T	Z
6	600	10	-31.2
	300	10	-18.7
	600	100	2.3
	300	100	30.2
18	600	10	46.0
	300	10	53.4
	600	100	402.
	300	100	321.

In a qualitative sense one can observe from Table 24 that: (1) As the walk time between machines increased with other parameters constant, the total system efficiency for both strategies decreased and the centrally located strategy had a larger decrease.

(2) As the failure rate increased with other parameters constant, the total system efficiency decreased.

(3) For given λ and T, as the number of machines increased, the total system efficiency decreased.

SECTION VII

COMPUTER SIMULATION LANGUAGES

The discussion of simulation techniques throughout the major part of this report is essentially independent of the programming language used. However, at some point the analyst must decide what language to use: either a general purpose language (GPL) or a simulation programming language (SPL).

The purpose of this section is to highlight the differences between GPL and SPL, and examine the impact of using a particular language on the model. This section is not a detailed discussion of programming languages. The reader is directed to the appropriate reference cited in the discussion for an in depth treatment of features of each language.

Regardless of the language used, certain basic functions are required:

- Pseudorandom number generation
- Creation of random variates
- A time-flow mechanism
- Data storage
- Statistical analysis
- Data output
- Diagnostic information on logic and other errors.

In the following discussion, we will comment on each of these features.

General Purpose Languages

An analyst with access to a computer system would have some general purpose language available for use in simulation. The language may be FORTRAN, ALGOL, COBOL, or PL/1. Let's examine the advantages and disadvantages of using a GPL for simulation rather than an SPL.

The chief advantage in using a GPL is flexibility. The analyst is free to write any subroutine he believes is required for the simulation. An SPL is usually rigidly structured so that the analyst must use what is built into the language. He may or may not be able to conveniently add his own subroutines. Also, an SPL may imply that the problem be expressed in a particular form, such as flow through a block diagram as in GPSS.

A second advantage of a GPL is universal availability. FORTRAN compilers, for example, exist for almost every make and size computer. This permits greater compatibility between machines for simulations which may be transferred between computer installations.

In terms of the basic requirements, the analyst must provide a pseudorandom number generator and a means of creating random variates. This is not really a disadvantage since by providing the random number generator the analyst <u>knows</u> exactly what he has. He probably has already evaluated the generator's behavior and has demon-

strated its acceptability. By using an SPL pseudorandom number generator, the analyst is probably assuming that the generator has been tested and demonstrates randomness in a statistical sense. This unfortunately could be an invalid assumption leading to cataclysmic results. Often the random number generator in an SPL cannot be directly accessed for statistical testing and the user is left unsure of its behavior.

In a GPL, the analyst is free to choose the time flow mechanism which he believes is most appropriate for that particular simulation. An SPL may force the analyst into using a particular time flow mechanism, usually next event.

The analyst must also provide a means of storing and manipulating data for the output report. But he is free to use graphical or tabular formats in whatever representation is suitable. This flexibility is not always present in SPLs.

The chief disadvantage of a GPL is in diagnostic information on the program. The GPL compiler will detect syntax errors in language usage but does not have the elaborate logic checks present in SPL.

Simulation Programming Languages

The most popular Simulation Programming Languages are GASP, GPSS, SIMSCRIPT, and SIMULA. Table 25 displays the current availability of these languages.

Table 25. The Availability of Simulation Programming Languages

Computer Manufacturer	GPSS	SIMSCRIPT II	SIMULA
Burroughs	X	X	X
CDC	X	X	
Honeywell	X	X	
IBM	X	X	
UNIVAC	X	X	X

The discussion of each language will include its advantages and disadvantages as well as how it views the world.

GASP

The General Activity Simulation Program (GASP) is unlike other SPLs since it is written in FORTRAN and can be used on any machine with a FORTRAN compiler. GASP is a set of FORTRAN subroutines and functions which are linked by a main program called the GASP EXECUTIVE. The main program starts the simulation, enables sequencing of time, monitors intermediate results, and initiates printout of results.

The chief advantages of GASP are that it is not necessary to learn a new language or obtain a new compiler, it employs the modular approach used in simulation, and it has machine independence.

The disadvantages relate to GASP's basis in FORTRAN. Problems are encountered in input/output formating and in debugging the program.

GASP employs a next event time flow mechanism. An event selection function is used to sequence the execution of event routines. The simulation switches from event to event according to the times events are scheduled to occur. The analyst must write event routines to specify system state changes that occur at event times and the future events that are generated by event occurrences. An event file is used to store attributes, such as occurrence times, and events are inserted in the file and removed from it in chronological order.

Basically GASP views the world as consisting of elements, attributes, events, decision rules, processes, states, and values. Elements may be permanent or temporary and they interact through events. Events are FORTRAN subroutines written by the user which change one or more elements.

The primary reference for GASP is 37.

GPSS

The General Purpose Simulation System (GPSS) develops the simulation model in terms of a block diagram. The structure and action of a system is described using blocks which represent a step in the movement of entities through the system.

One advantage of GPSS is that no knowledge of computer operation is assumed. The analyst needs only a block diagram using the standard GPSS blocks to directly transcribe these into the proper punch card formats. The simulation is then ready to be run. A second advantage is that GPSS is easy to learn. Thirdly, it is a powerful tool for problems that fit a block structure.

The disadvantages include limited arithmetic statements, requirement for a separate compiler, and slow execution time.

The GPSS time flow mechanism has a next event orientation. Temporary entities are called transactions and may encounter time delays in any block. When that happens, the transaction is placed in a Future Events Chain in ascending order of departure from the block. Other transactions are in a Current Events Chain. The Current Events Chain is scanned at each clock time to attempt to move transactions into the next block. If movement is possible, the transaction moves through as many blocks

as possible until (1) it encounters a time delay and is placed into the Future Events Chain, (2) is blocked from entering a next block and remains in the Current Events Chain, or (3) is destroyed. After the current events chain is completely scanned without being able to move any transaction, the clock is updated to the block departure of the first transaction in the Future Events Chain.

GPSS views the world as consisting of blocks, transactions, and equipment. The simulation model is developed in terms of a block diagram of the flow of transactions through the system. A specific set of block types exists with which the model can be built. Program flow is achieved by linking elemental blocks together.

The primary references for GPSS are 39 and 40.

SIMSCRIPT II

SIMSCRIPT II is a powerful simulation language which can be used as a GPL in its own right. SIMSCRIPT is defined on several levels; this allows the analyst to adopt as many features as required for the particular problem at hand.

The advantages of SIMSCRIPT include more flexibility and power for modifying system states than other languages, emphasis on manipulation of entity attributes, explicit

use of temporary entities, and use of set operations. In addition, the language is English-like. For example, some statements are:

```
ADD E TO V

SUBTRACT E FROM V

DEFINE A AND B AS REAL VARIABLES

LET S = X * Y

SCHEDULE A NEW. ARRIVAL IN 6 MINUTES
```

The diagnostic capabilities of SIMSCRIPT are quite good and some syntax errors are corrected by the compiler.

The chief disadvantage is the complexity of the language and the necessity for another compiler.

The time flow mechanism is "next event." The system mechanism observes data cards and previously scheduled events, orders them by event time, and causes them to occur when the time arrives. The events are called as subprograms, as in GASP. The basic unit of action is an activity and an event is an instant in time which starts or stops an activity.

SIMSCRIPT's view of the world is based on the notion that the state of a system can be described in terms of entities, attributes, and sets (groups of entities). The analyst must explicitly specify all permanent and temporary entities with a complete list of attributes and set memberships.

SIMULA

SIMULA is not as widely used as the other languages. It has a block structure based on that of the parent language ALGOL, where both static and dynamic aspects are found in the block. Each block describes a component of the system. SIMULA is mentioned here only to acquaint the reader with its existence. The primary reference is 38.

Considerations in Selecting a Language

Some considerations in selecting a particular programming language are:

- Availability
- Programming costs
- Execution costs
- Validity of results

Programming costs are affected by

- Experience of the programmer
- Ease of programming in the language
- Ease of debugging
- Transferability between machines
- Availability of standard routines and functions
- Flexibility in using time flow mechanisms

Execution costs are determined by

- Efficiency of the compiler
- Time flow mechanism suitability to the problem

- Storage requirements

Validity of the results are influenced by

- Modularity of the resultant program
- "Naturalness" of the language with respect to the problem simulated
- Tolerance by the compiler of non-standard constructs
- Diagnostic features and error checking routines
- Quality of the random number generators

Final Comments on Languages

The general advantage of an SPL over a GPL is that it helps the analyst crystallize his thinking and reduce model formulation time. The SPL can provide a way of thinking about the problem which can be a great asset if the world view is appropriate to the problem. Secondarily, an SPL eases the burden of obtaining data and statistics, and provides extensive error checking to prevent misuse of the language and to locate logic errors.

SECTION VIII

MILITARY APPLICATIONS OF COMPUTER SIMULATION

The early history of operations research is deeply rooted in military applications. Likewise, computer simulation has been adopted as a major evaluation tool by military decision makers. Although many problems such as logistics, inventory, and maintenance are common to both the military and industrial environment, several uses are uniquely military. In this section we will examine several areas of military application of simulation.

An Overview

Let's briefly examine three areas where simulation is used as a significant analysis tool. One important area is aircraft design. Although computer models are used throughout all design phases, they are gaining increasing importance in the preliminary or conceptual phase. The use of models allows engineers to establish performance specifications, evaluate design proposals, and reconfigure subsystems to lessen vulnerability of components.

Another area of significant application is nuclear survivability/vulnerability analysis. Survivability in a nuclear environment is extremely important for

military aircraft and weapon systems. Digital models are used to calculate the effects due to a nuclear burst and evaluate the survivability/vulnerability of systems located in the vicinity. Effects usually include those due to neutrons, gammas, induced gammas, blast, thermal, and X-rays.

A third case is simulation of the Air Force logistics system. In an era of complex and sophisticated weapons systems, extensive maintenance and inventory facilities are required. Digital models are used to determine the resource requirements for current and projected Air Force equipment and weapon systems.

There are many other fields which use simulation extensively but the intention here was to merely highlight several widespread areas. References 31 and 35 are excellent bibliographies of current simulation literature and indicate the wideranging application of modeling.

Computer Simulation in Electronic Warfare

Another area of frequent application of simulation to military operations research is in electronic warfare analysis. Electronic warfare (EW) is defined as military action involving the use of electromagnetic energy to determine, exploit, reduce, or prevent hostile use of the electromagnetic spectrum and action which retains

friendly use of that spectrum. We will examine simulation in the division of EW known as electronic countermeasures (ECM), i. e., actions taken to prevent or reduce an enemy's effective use of the electromagnetic spectrum.

The decision maker is faced with the problem of determining which basic concept the attacking force should employ: ignore the defense and accept any consequent losses; avoid the defense; deny information to the defender's sensors, such as through the use of radar absorbing materials, reactive or evasive tactics, or exploitation of terrain masking and clutter; degrade the operation of the defense system with noise jamming, deception techniques, or confusion; or destroy the defense sensors and weapons. Simulation is used to aid the decision maker in selecting a concept to employ.

Three types of simulation are used in ECM analysis --- digital models, simulators, and flight tests. A simulator refers to a man-in-the-loop simulation and, within an EW context, is defined as a man-machine tool which combines operators, analog/digital computers, and actual hardware, such as real radar displays or operational ECM components, to simulate the real world system. Flight tests are tests involving real aircraft and simulated threat radars with associated operators which are performed in an instrumented or controlled environment.

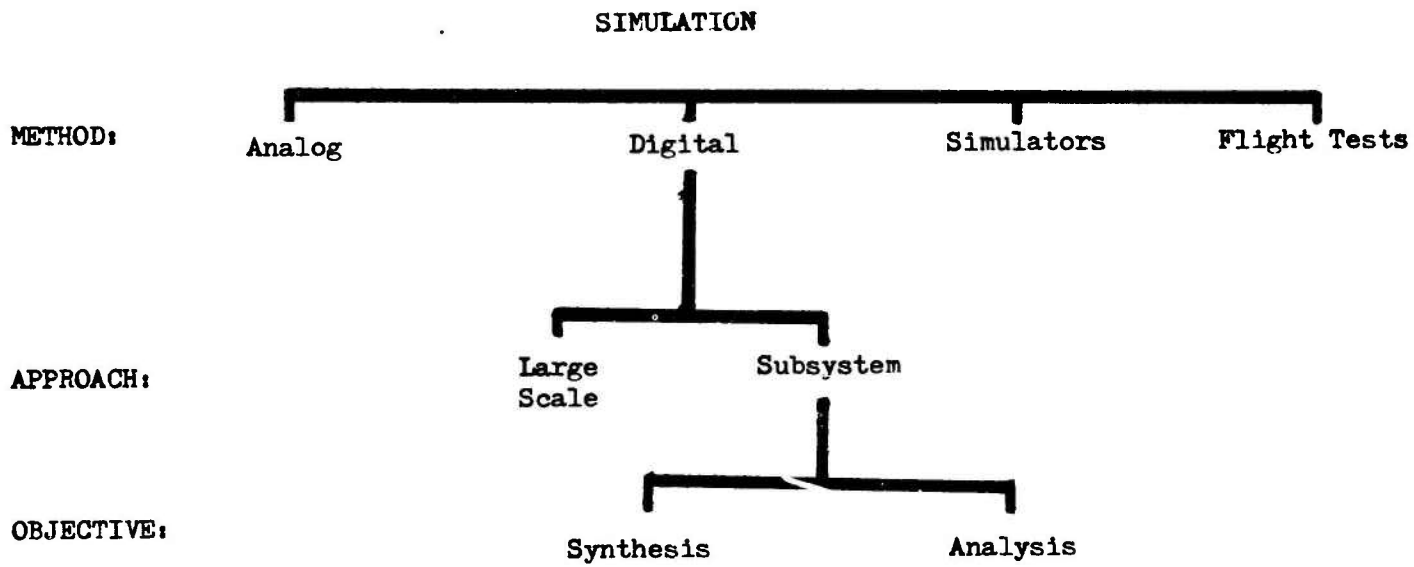


Figure 31. The Structure of Simulation in EW

Therefore, in considering the structure of simulation for ECM analysis we have Figure 31, a special case of Figure 2 in Section I.

The Advantages and Limitations of Modeling

In choosing among digital models, simulators, and flight tests, the decision maker is faced with distinctive advantages and limitations. Let's look at some specific uses of digital models and simulators. Both can be used to verify analytical results, obtain results not amenable to purely analytical solution, and conduct parametric and sensitivity studies. They also can provide a base for flight test design, verify instrumentation requirements for flight tests, evaluate equipment specifications, optimize equipment usage, and evaluate tactics. The advantages of digital models and simulators include rigid test control, ease in making configuration changes, lower cost than flight tests, and the limited instrumentation required. Simulators enhance realism over pure digital models by including human interactions and actual ECM hardware or radar equipment. The limitations on both models include restrictions on numbers of aircraft and ECM, less realism than flight tests, and lack of many real world interactions.

Flight tests may be used in the same fashion as above for the other models but also can evaluate real e-

Table 26. A Comparison of Simulations with Real World

	DIGITAL MODELS	SIMULATORS	FLIGHT TESTS	REAL WORLD
REALISM	Medium/low	Good	High	Highest
CONTROL	Easy	Easy	Difficult	Not possible
INSTRUMENTATION COSTS	Low	Moderate	High	High
DATA REDUCTION COSTS	Low	Moderate	High	High
OVERALL COSTS	Low	Moderate	High	Very High
TIME REQUIREMENTS	Days	Weeks	Months	Year
VERIFICATION SOURCE	Simulators Flight tests Real world	Flight tests Real world	Real world	

quipment, confirm tactics, and evaluate operational readiness. The advantages of flight testing include the capability to use multiple aircraft and multiple penetration aids, incorporation of real world phenomena and interactions, and high realism. The limitations include extensive ground instrumentation, limited on-board instrumentation, limited test control, fixed geography of test range, and restrictions to several penetrators against only one radar.

Table 26 is a comparison of the three simulations according to several measures. It is clear that tradeoffs in time, cost, and realism are involved in the selection of a particular simulation model.

A Hypothetical EW Study

To demonstrate how simulation models are used in an EW analysis, consider the following hypothetical study task. The overall objective is to identify areas of highest payoff for employment of ECM onboard manned aircraft penetrating an air defense system at high, medium, and low altitudes. In particular, the analyst would examine the effectiveness of surface-to-air missiles and antiaircraft artillery employed against the penetrators and those factors which are significant in limiting the performance of the defense system.

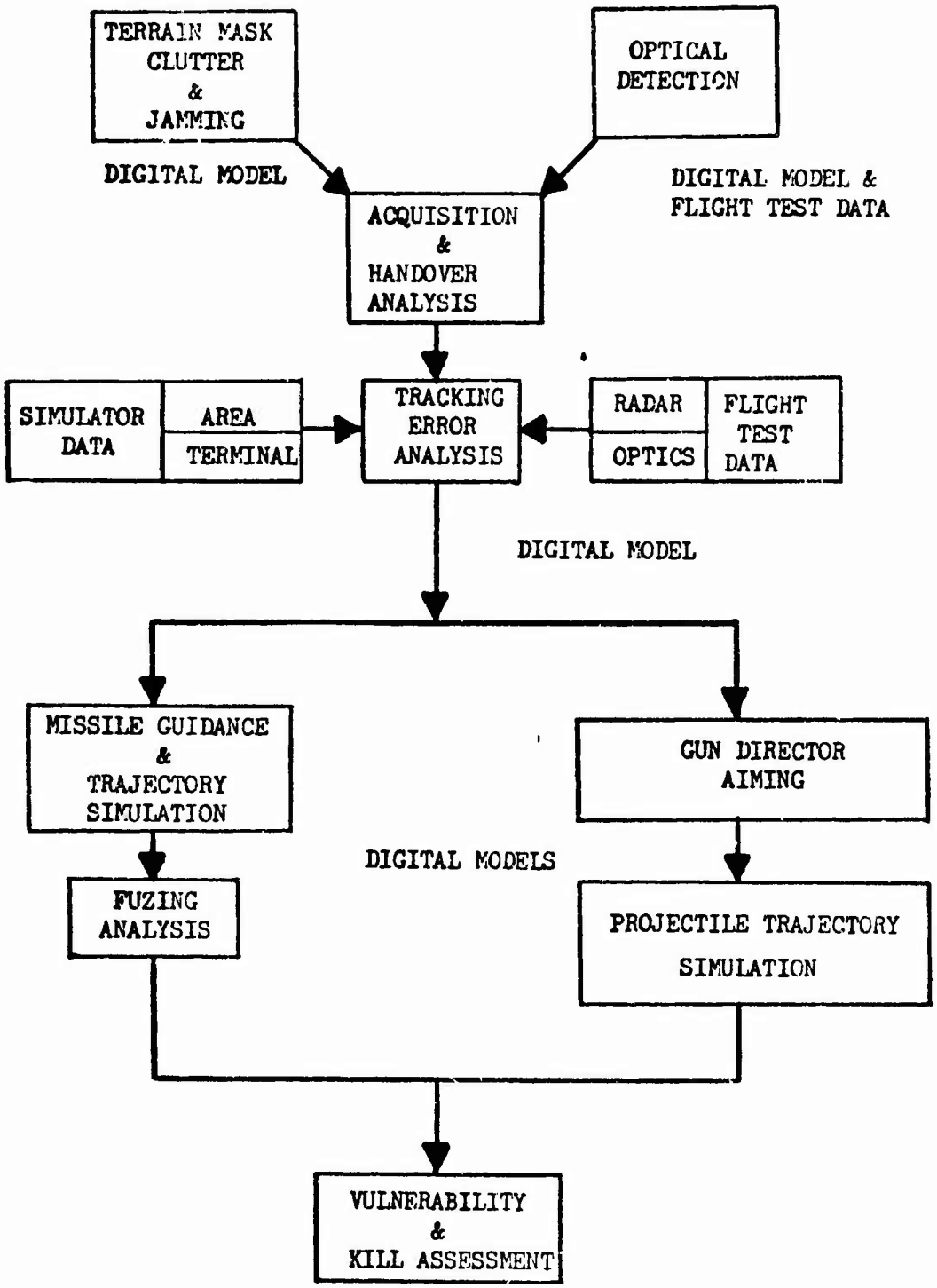


Figure 32. The Interrelationship of Modeling Subtasks

Several important features should be incorporated into the analysis study. The analysis should include the penetration force's use of tactics and ECM, penetrator acquisition by enemy weapons systems, enemy tracking system performance in both clear and jamming environments, weapon trajectory and fuzing errors, and vehicle vulnerability due to warhead fragmentation.

A composite modeling approach could be employed in the study. Target acquisition can be modeled by digital computer techniques as can simulation of missile guidance, missile and shell trajectories, missile fuzing, and vehicle vulnerability. However, human tracking performance should be based on simulator and flight test data. Figure 32 depicts the interrelationship of various subtasks and modeling techniques utilized within the study.

In order to make accurate judgments on the basis of aircraft survivability, the effects of target tracking errors must be transformed to weapon miss distances. Then by modeling the vulnerable components of the aircraft and blast fragmentation of the warhead, the probability of killing the penetrator can be described as a function of weapon miss distance. Figure 33 shows a graph of illustrative output from such digital model. By systematically evaluating the same defense configuration with and without various types of electronic countermeasures,

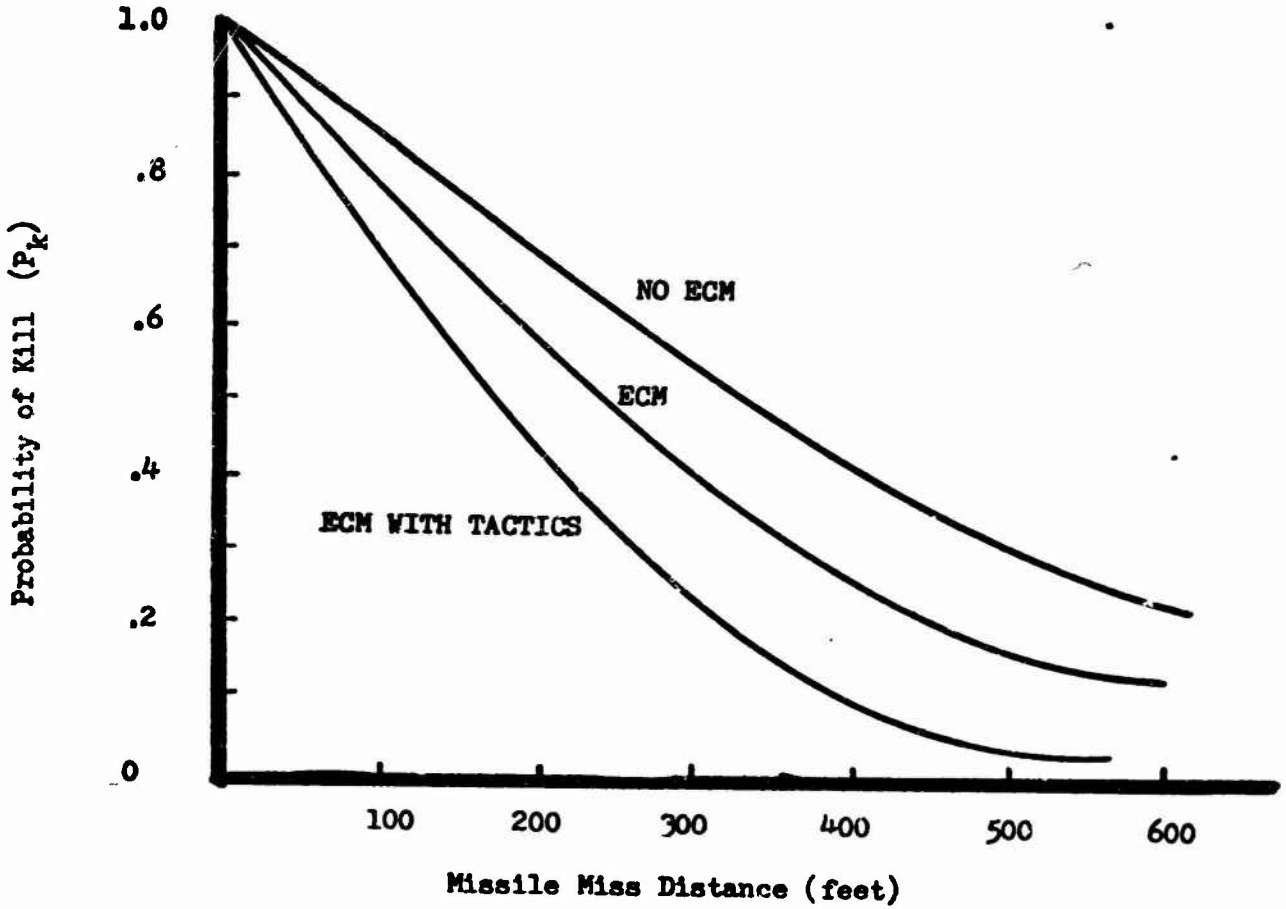


Figure 33 . Typical Model Output Illustrating Penetrator Kill as a Function of Missile Miss Distance

a family of probability curves could be generated depicting the effectiveness of ECM against terminal weapons systems. The probability of penetrator kill can also be described as a function of missile miss distance and downrange distance from the surface-to-air missile site. Figure 34 illustrates the variations in vehicle kill probability as a function of warhead detonation position.

The data resulting from an evaluation of tracking system performance, say for antiaircraft artillery could be summarized by a cumulative probability that the shell missed by less than a given distance. Figure 35 shows such a graph as a function of tracking mode.

The three outputs above typify the probabilistic relationships which can be derived from data generated by digital models. A systematic variation of various parameters, ECM techniques, force sizes, or tactics would result in a thorough investigation for the proposed task. As a result, an assessment can be made of the vulnerable areas of the defense system and those functions against which ECM can be most effectively employed.

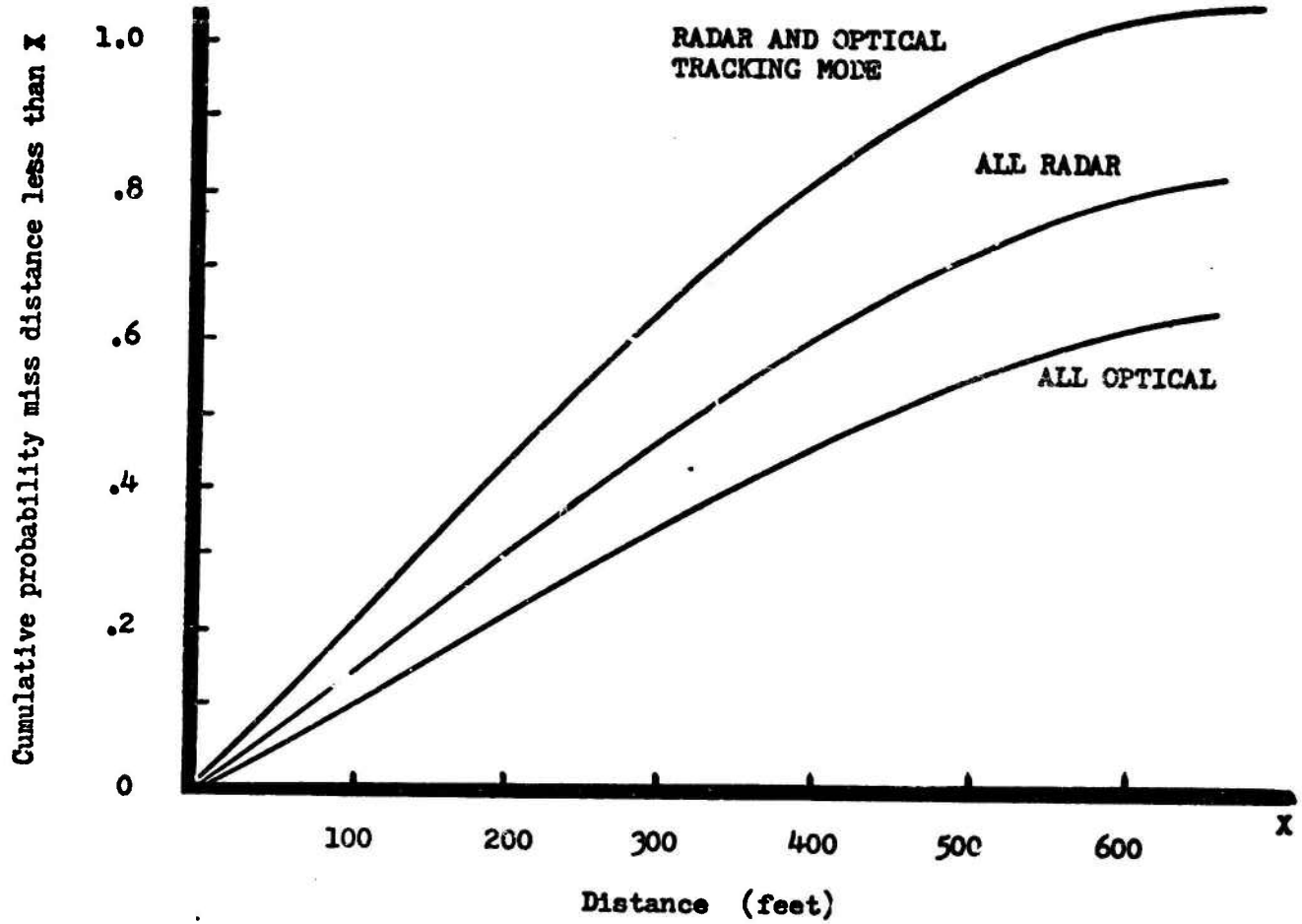


Figure 34. Effectiveness of AAA Tracking System Performance

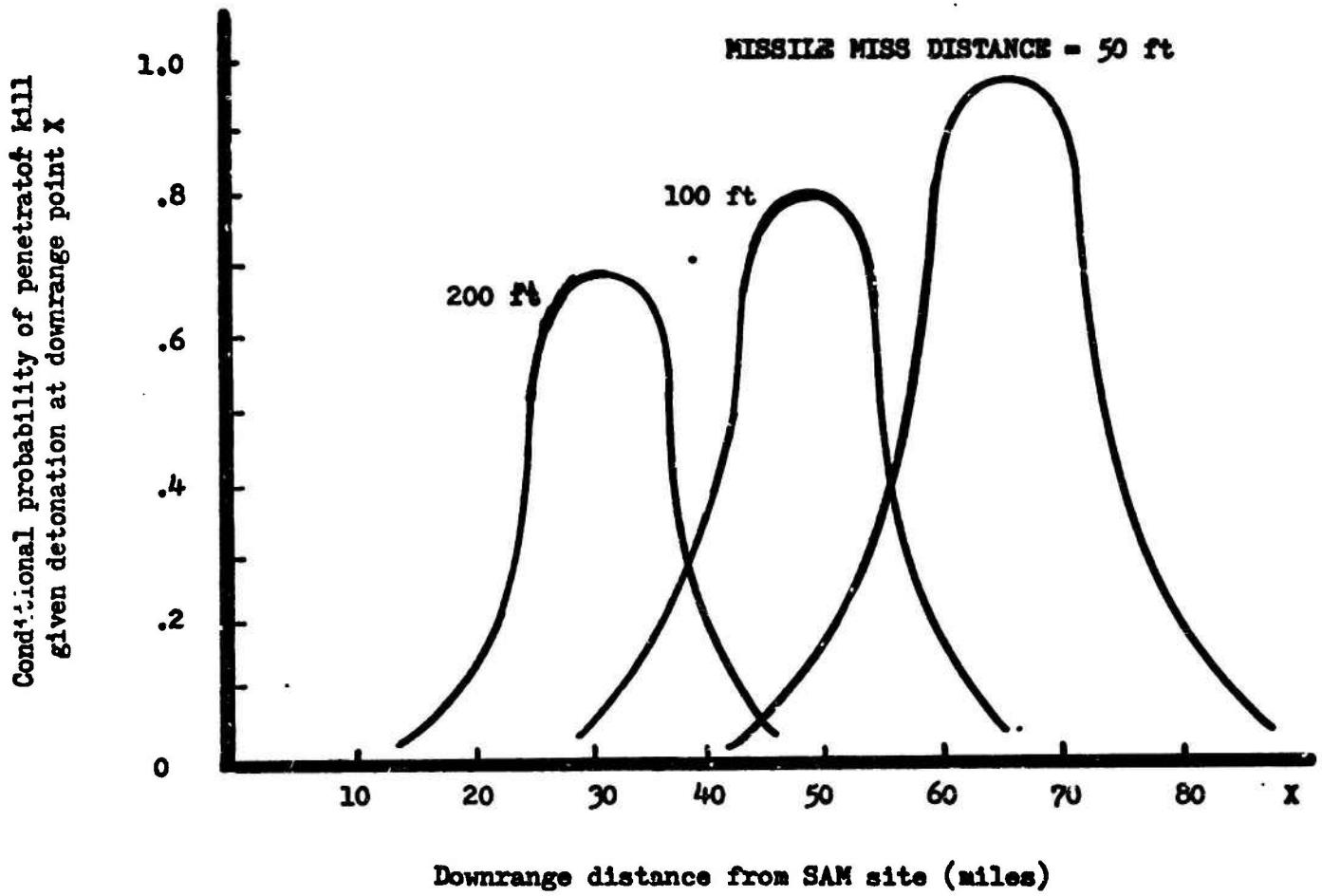


Figure 35. Penetrator Kill as a Function of Distance From Missile Site

Computer Simulation in Logistics

The effects on operations of supply distribution policies, spares stockage levels, and maintenance capabilities is usually difficult or impossible to determine using analytical techniques. Demands for supplies, spare parts, and repair work occur at random intervals and frequently the demand rates cannot be correlated to other activity levels. For example, the removal rates of aircraft engines may decrease for a period of time when flying activity increases. In these cases, the interactions of various logistics functions can best be studied using simulation.

A goal of the supply function is to preposition the limited available assets where they can fill the greatest need with the least delay. A large portion of the military's resources is invested in a class of items known as recoverable spares. These items may be removed from an aircraft when they fail, replaced by a like serviceable unit, and the failed unit is then sent to a repair facility. Some of the units may cost up to $500,000 each. For example, if an Inertial Navigation Unit (INU) fails on an aircraft, it must be sent to a depot for repair. When the depot completes the repair of the INU, a decision must be made as to where to locate the unit until it is needed. It may require several days to ship the unit to a base.

The depot may decide to:

a. Ship the unit to any one of several bases where the unit may be needed, or

b. Retain the unit at the depot until it is known which base needs it most.

At the time the decision is made, none of the bases may have a "desperate" need for one more unit. If base A has the lowest stock on hand and the unit is shipped to base A, it may develop that base B uses up its on-hand stock first and becomes "desperate" for a replacement. In this case, a "bad decision" has been made. If, on the other hand, the depot decides to retain the unit and base A actually develops a need for the unit, then the delay of shipping time occurs before base A receives the unit and it may appear that the depot made a "bad decision" in retaining the unit. Clearly the only way to make the "right decision" every time is to have complete knowledge of the future. Lacking this, one must consider the known probabilities of error for various alternatives and select the cours of highest probable payoff.

Several decision making rules for this redistribution problem are possible. The current policy in the Air Force is to establish an authorized stock level for each type of unit for each base and when a base sends a unit to the depot for repair, the depot sends the base a like unit as replacement.

An alternative policy proposed by RAND Corporation [42] is called Real Time METRIC. Under this policy a "need" based on on-hand stock and expected demand rate is computed for each base. A depot "reluctance" is also computed. If the "need" at one or more bases exceeds the "reluctance" then the unit is shipped to the base with the greatest "need". Otherwise, the depot retains the unit until the next decision time.

Would this policy give better results? Simulation techniques were used to test the policy and results indicated that a 30% reduction in base backorders could occur. This decision policy is being included in the Advanced Logistics System. Other redistribution policies are also being developed and tested through simulation.

REFERENCES

1. Anderson, R.L., "Distribution of Serial Correlation Coefficients," Annals of Mathematical Statistics, 13 (1), March 1942.

2. Berman, M.B. Notes on Validating/Verifying Computer Simulation Models, RAND Corporation, P-4891, August 1972.

3. Bhat, U. N. Elements of Applied Stochastic Processes, Wiley & Sons, New York, 1972.

4. Box, G.E.P. and M.E. Muller, "A Note on the Generation of Normal Deviates," Annals of Mathematical Statistics, 28, 1958.

5. Conway, R.W., "Some Tactical Problems in Digital Simulation," Management Science, 10 (1), October 1963.

6. Conway, R.W., B. M. Johnson, and W. L. Maxwell, "Some Problems of Digital Systems Simulation," Management Science, October 1959.

7. Emshoff, J.R. and R.L. Sisson, Design and Use of Computer Simulation Models, Macmillan, New York, 1970.

8. Evans, G.W., G.F. Wallace, and G.L. Sutherland, Simulation Using Digital Computers, Prentice-Hall Englewood Cliffs, N.J., 1967.

9. Fishman, G.S. "Problems in the Statistical Analysis of Simulation Experiments: The Comparison of Means and the Length of Sample Records," Communications of the ACM, 10, February 1967.

10. Fishman, G.S. "Bias Consideration in Simulation Experiments," Department of Administrative Sciences, Yale University, April 1971.

11. Forrester, J.W., Industrial Dynamics, MIT Press, Cambridge, Mass., 1961.

12. Forrester, J.W., Principles of Systems, Wright-Allen Press, 1968.

13. Freund, J.E., Mathematical Statistics, Prentice-Hall Englewood Cliffs, N.J., 2nd Edition, 1971.

14. Green, B.F., J.E.K. Smith, and L. Klem, "Empirical Tests of an Additive Random Number Generator," J. ACM, 6, December 1959.

15. Hull, T.E., and A.R. Dobell, "Random Number Generators," SIAM Review, 4 (3), 1962.

16. Hull, T.E., and A.R. Dobell, "Mixed Congruential Random Number Generators for Binary Machines," J. ACM, 11, January 1964.

17. Jansson, Birger, Random Number Generators, Victor Pettersons, Stockholm, 1966.

18. Kiviat, P.J., "Digital Computer Simulation: Computer Programming Languages," RAND Report RM-5883-PR, January 1969.

19. Knuth, Donald E., The Art of Computer Programming: Volume 2/Seminumerical Algorithms, Addison-Wesley Reading, Massachusetts, 1964.

20. McQuay, W.K., The Nature and Application of Computer Simulation Models in Electronic Warfare, AF Avionics Laboratory WRA Technical Memo, March 1973.

21. Mood, A.M., and F.A. Graybill, Introduction to Theory of Statistics, McGraw-Hill, New York, 2nd Edition, 1963.

22. Muller, M.E., "A Comparison of Methods for Generating Normal Deviates on Digital Computers," J. ACM. 6 (3), July 1959.

23. Nance, Richard E., "On Time Flow Mechanisms for Discrete System Simulation," Management Science, 18, September 1971.

24. Naylor, T.H., J.L. Balintify, D.S. Burdick, and K.Chu, Computer Simulation Techniques, John Wiley & Sons, New York, 1966.

25. Overstreet, Claude, et al , "A FORTRAN V Package for Testing and Analysis of Pseudorandom Number Generators," Technical Report No. CP-700011, CS/OR Center, Southern Methodist University, Dallas, Texas, 1970.

26. Owen, D.B., Handbook of Statistical Tables, Addison Wesley, Reading, Mass, 1962.

27. Page, E.S. "The Generation of Pseudo-Random Numbers," Digital Simulation in Operational Research, S.H. Hollingsdale (ed), American Elsevier, 1967.

28. Quade, E.S. (ed), Analysis for Military Decisions, RAND Report, R-387-PR, November 1964.

29. Romig, H.G., 50-100 Binomial Tables, Wiley & Sons, New York, 1953.

30. Rotenberg, A., "A New Pseudo-Random Number Generator," J. ACM, 7 (1960), pp. 75-77.

31. Shubik, M., and G. D. Brewer, Reviews of Selected Books and Articles on Gaming and Simulation, RAND Report R-732-ARPA, June 1972.

32. Tausworthe, R.C., "Random Numbers Generated by Linear Recurrence Modulo Two," Math Comp, 19, April 1965.

33. Zehna, Peter W. (ed), Selected Methods and Models in Military Operations Research, Naval Postgraduate School, US Government Printing Office, 1971.

34. Tables of the Binomial Probability Distribution, National Bureau of Standards Applied Mathematics Series No. 6, US Government Printing Office, 1950.

35. Shubik, M., G. D. Brewer, and E. Savage, The Literature of Gaming, Simulation, and Model-Building: Index and Critical Abstracts, RAND Report R-620-ARPA June 1972.

36. Kiviat, P. J., R. Villaneuva, H. M. Markowitz, The Simscript II Programming Language, Prentice-Hall, Englewood Cliffs, N. J., 1969.

37. Pritsker, A., and P. J. Kiviat, Simulation with GASP II, Prentice-Hall, Englewood Cliffs, N.J., 1969.

38. Dahl, O., and K. Nygaard, SIMULA, Introduction and User's Manual, Norwegian Computing Center, Oslo, 1965.

39. IBM Corp, General Purpose Simulation System/360 Introductory User's Manual, H20-0304-4,1969.

40. IBM Corp, General Purpose Simulation System/360 User's Manual, H20-0326-3, 1970.

41. Berman; M. B., Generating Gamma Distributed Variates for Computer Simulation Models, RAND Corporation R-641-PR, February 1971.

42. Miller, B. L., A Real Time METRIC for the Distribution of Serviceable Assets, RAND Corporation, RM-5687-PR, October 1968.

APPENDIX

TRICKS OF THE TRADE

Certain computer programming techniques are used in many different types of simulation models. They provide fundamental capabilities and are, for the most part, simple but enlightening uses of elementary concepts. For the uninitiated the method of implementation may not be obvious; therefore, some "tricks of the trade" are documented in this section.

Sorts

The programmer may be required to sort an array of values, for example, into ascending order of magnitude. There are at least two techniques -- a bubble sort and a virtual sort.

Bubble Sort

In a bubble sort an array of values is checked element by element and compared to the succeeding value. If the objective is to order the elements by increasing magnitude, then when a succeeding value is smaller than its predecessor the elements are reversed. The FORTRAN subroutine is shown in Figure 36 . Note that if the list of values is already completely sorted, only one pass will be required.

```
C  * * *                                  * * *
C  * * *   BUBBLE SORT                    * * *
C
C
          SUBROUTINE SORT (A, ISTART, ISTOP)
          DIMENSION A (100)
          M = ISTART + 1
          N = ISTOP
1         ISW = 1
          DO 2 I = M, N
          IF (A(I-1).LE.A(I))  GO TO 2
          TEMP = A(I-1)
          A(I-1) = A(I)
          A(I) = TEMP
          ISW = 0
2         CONTINUE
          N = N-1
          IF (ISW.EQ.0)  GO TO 1
          RETURN
          END
```

Figure 36. Bubble Sort FORTRAN Subroutine

Virtual Sort

In a bubble sort, the original ordering of the values is lost since the ultimate effect is to reorder the elements according to increasing size. In a virtual sort, the end result is the same but the original array is not changed. A new indexing array is used as a pointer to establish the new ordering. The FORTRAN subroutine is shown in Figure 37 .

```
C  * * *                                  * * *
C  * * *   VIRTUAL SORT                   * * *
C  * * *                                  * * *
C
          SUBROUTINE VIRSORT(A,M,ISTART,ISTOP)
          DIMENSION A(100), M(100)
          DO  10 I = I = ISTART, ISTOP
     10   M(J) = I
          IMAX = ISTOP - 1
          DO 20 I = ISTART, IMAX
          K = I + 1
          DO 10 J = K, ISTOP
          IF (A(M(I).LE.A(M(J)) GO TO 20
          ITEMP = M(I)
          M(I) = M(J)
          M(J) = ITEMP
     20   CONTINUE
          RETURN
          END
```

Figure 37. Virtual Sort FORTRAN Subroutine

Search for Extrema

One of the most elementary exercises in computer programming is to determine the maximum or minimum from among a set of values. A simple FORTRAN DO Loop and a greater or less than test will suffice. Figure 38 is the FORTRAN subroutine for finding the maximum value in an array and returning the value of the array index. The only point of significance is the test IF (A(I).LT.XMAX). Note that if two elements in the array have the same value XMAX, the index of the latter one will be chosen as IMAX. Had the test been IF(A(I).LE.XMAX), then the first value encountered would be retained as IMAX. The programmer should be aware of this, especially if the index has some effect on results, such as alternative solutions.

```
C  * * *  SEARCH FOR MAXIMUM          * * *
C
         SUBROUTINE FINDMAX (A,IMAX)
         DIMENSION A(100)
         XMAX = -1.E 10
         DO 10 I = 1, 100
         IF(A(I).LT.XMAX) GO TO 10
         XMAX = A(I)
         IMAX = I
      10 CONTINUE
         RETURN
         END
```

Figure 38. Maximum Search FORTRAN Subroutine

TABLE LOOK-UP

If a random variate is to be selected from one of the standard distributions, it is usually best to use the tranformation methods given in Section III. However, random variates sometimes are needed from a distribution which is difficult or impossible to describe using the known probability density functions. For example, suppose one desires to generate the length (in-hours) of an aircraft sortie where 5% of all sorties are 1 hour long, 45% are 7 hours long, 40% are 9 hours long, and 10% are 24 hours long.

See Figure 39.

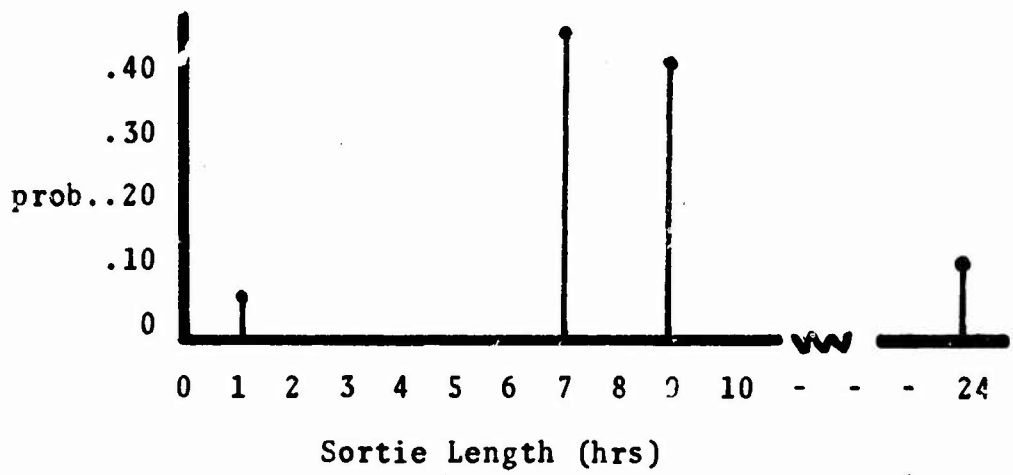


Figure 39. Distribution of sortie lengths

In this case a table look-up procedure should be used. First a table must be established thus:

F(X)	X
.05	1
.50	7
.90	9
1.00	24

To use this table to generate random sortie lengths, the FORTRAN function in Figure 40 is called.

```
        FUNCTION SORLEN (DUMMY)
        DIMENSION F (4), HOURS (4)
        DATA F/.05,.50,.90,1.00/,HOURS/1.0,7.0,9.0,24.0/
        R = RANG(RNUM)
        DO  _   I = 1,4
        IF (F(I).GE.R) GO TO 2
1       CONTINUE
2       SORLEN = HOURS (I)
        RETURN
        END
```

Figure 40. A FORTRAN function for table look-up.

Thus, if the random number generated is 0.62, the random variate 9.0 will be returned.

A variation of this technique involves interpolation. If the length of sorties is described by the graph in Figure 41, the function in Figure 42 would be used.

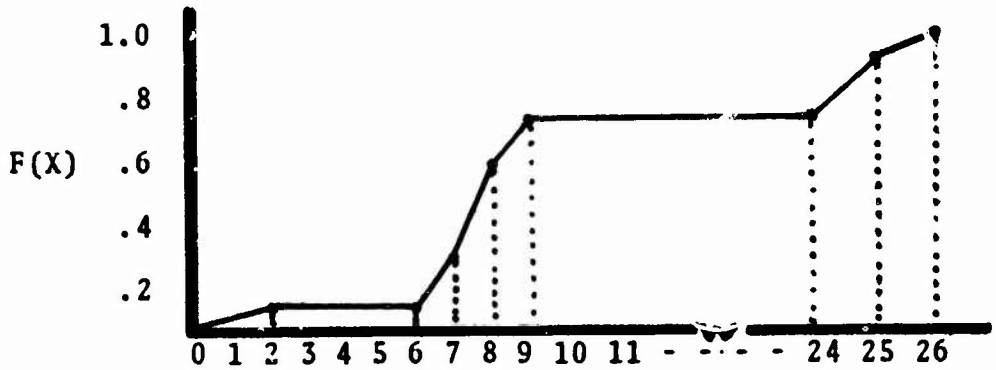


Figure 41. Cumulative distribution of sortie lengths

```
      FUNCTION SORLET (DUMMY)
      DIMENSION F(9),H(9)
      DATA F/.0,.1,.1,.3,.6,.75,.75,.95,1.0/
      DATA H/0.,2.,6.,7.,8.,9.,24.,25.,26./
      R = RANG (RANUM)
      DO 1  I  = 1,9
      IF(F(I).GE.R) GO TO 2
1     CONTINUE
2     SORLET = H(I-1) + (H(I) - H(I-1))*(R-F(I-1))
     1   /(F(I)-F(I-1))
      RETURN
      END
```

Figure 42. Table look-up with interpolation.

Thus, if the generated random number is 0.20, the returned sortie length will be 6.5 hours.

Table 27. Continuous Probability Distributions

DISTRIB	PDF	MEAN	VARIANCE
UNIFORM CONTINUOUS	$f(x) = \frac{1}{\beta - \alpha}$ $\alpha < x < \beta$	$\frac{\alpha+\beta}{2}$	$\frac{(\alpha-\beta)^2}{12}$
NORMAL	$f(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$ $\sigma^2>0$, $-\infty<x<\infty$, $-\infty<\mu<\infty$	μ	σ^2
GAMMA	$f(x) = \frac{1}{\Gamma n} e^{-x} x^{n-1}$ $x>0$	n	n
	$f(x;\alpha,\beta) = \frac{1}{\alpha!\,\beta^{\alpha+1}} x^{\alpha} e^{-\frac{x}{\beta}}$	$(\alpha+1)\,\beta$	$(\alpha+1)\,\beta^2$
BETA	$f(x;\alpha,\beta) = \frac{(\alpha+\beta+1)!}{\alpha!\,\beta!} x^{\alpha}(1-x)^{\beta}$ $0<x<1$, $\alpha,\beta>-1$	$\frac{\alpha+1}{\alpha+\beta+2}$	$\frac{(\alpha+1)(\beta+1)}{(\alpha+\beta+3)(\alpha+\beta+2)^2}$
CHI SQUARE	$f(x) = \frac{1}{2^{n/2}\,\Gamma n/2} x^{\frac{n}{2}-1} e^{-\frac{x}{2}}$	n	$2n$
EXPONENTIAL	$f(x) = \frac{1}{\beta} e^{-\frac{x}{\beta}}$ $x>0$	β	β^2

Table 28. Discrete Probability Distributions

DISTRIB	PDF	MEAN	VARIANCE
DISCRETE UNIFORM	$f(x) = \frac{1}{n}$ $x = 1, 2, \ldots, n$	$\frac{n+1}{2}$	$\frac{n^2-1}{12}$
POINT BINOMIAL	$f(x) = p^x (1-p)^{1-x}$ $x = 0, 1$	p	0
BINOMIAL	$b(x; n, p) = \binom{n}{x} p^x q^{n-x}$ $x = 0, 1, \ldots, n$	np	npq
POISSON	$f(x) = \frac{e^{-\lambda} \lambda^x}{x!}$ $\lambda > 0$ $x = 0, 1, 2, \ldots$	λ	λ
HYPER-GEOMETRIC	$f(x) = \frac{\binom{a}{x}\binom{b}{k-x}}{\binom{a+b}{k}}$	$\frac{ak}{a+b}$	$\frac{abk(a+b-k)}{(a+b)^2(a+b-1)}$
GEOMETRIC	$f(x) = p\, q^{x-1}$ $x = 1, 2, \ldots$	$\frac{1}{p}$	$\frac{q}{p}$
NEGATIVE BINOMIAL	$f(x) = \binom{x-1}{k-1} p^k q^{x-k}$ $x = k, k+1, \ldots$	$\frac{k}{p}$	$\frac{kq}{p^2}$

Table 29. ASCII Character Set

Character	Hollerith (026)	Hollerith (029)	ASCII Code
: †	8-2	8-2	072
A	12-1	12-1	101
8	12-2	12-2	102
C	12-3	12-3	103
D	12-4	12-4	104
E	12-5	12-5	105
F	12-6	12-6	106
G	12-7	12-7	107
H	12-8	12-8	110
I	12-9	12-9	111
J	11-1	11-1	112
K	11-2	11-2	113
L	11-3	11-3	114
M	11-4	11-4	115
N	11-5	11-5	116
O	11-6	11-6	117
P	11-7	11-7	120
Q	11-8	11-8	121
R	11-9	11-9	122
S	0-2	0-2	123
T	0-3	0-3	124
U	0-4	0-4	125
V	0-5	0-5	126
W	0-6	0-6	127
X	0-7	0-7	130
Y	0-8	0-8	131
Z	0-9	0-9	132
0	0	0	060
1	1	1	061
2	2	2	062
3	3	3	063
4	4	4	064

Character	Hollerith (026)	Hollerith (029)	ASCII Code
5	5	5	065
6	6	6	066
7	7	7	067
8	8	8	070
9	9	9	071
+	12	12-8-6	053
-	11	11	055
*	11-8-4	11-8-4	052
/	0-1	0-1	057
(	0-8-4	12-8-5	050
)	12-8-4	11-8-5	051
$	11-8-3	11-8-3	044
=	8-3	8-6	075
blank	no punch	no punch	040
(comma)	0-8-3	0-8-3	054
(period)	12-8-3	12-8-3	056
#	0-8-6	8-3	043
(apostrophe)	8-7	8-5	047
!	0-8-2	12-8-7	041
%	8-6	0-8-4	045
" (quote)	8-4	8-7	042
_ (underline)	0-8-5	0-8-5	137
]	11-0 or 11-8-2	11-0 or 11-8-2	[illegible]
&	0-8-7	12	[illegible]
@	11-8-5	8-4	100
?	11-8-6	0-8-7	077
[	12-0 or 12-8-2	12-0 or 12-8-2	173
>	11-8-7	0-8-6	076
<	8-5	12-8-4	074
\	12-8-5	0-8-2	134
^ (circumflex)	12-8-6	11-8-7	136
; (semicolon)	12-8-7	11-8-6	073

Table 30. Powers of Two

n	2^n	n	2^n
1	2	31	21474 83648
2	4	32	42949 67296
3	8	33	85899 34592
4	16	34	17179 86918 4
5	32	35	34359 73836 8
6	64	36	68719 47673 6
7	128	37	13743 89534 72
8	256	38	27487 79069 44
9	512	39	54975 58138 88
10	1024	40	10995 11627 776
11	2048	41	21990 23255 552
12	4096	42	43980 46511 104
13	8192	43	87960 93022 208
14	16384	44	17592 18604 4416
15	32768	45	35184 37208 8832
16	65536	46	70368 74417 7664
17	13107 2	47	14073 74883 55328
18	26214 4	48	28147 49767 10656
19	52428 8	49	56294 99534 21312
20	10485 76	50	11258 99906 84262 4
21	20971 52	51	22517 99813 68524 8
22	41943 04	52	45035 99627 37049 6
23	83886 08	53	90071 99254 74099 2
24	16777 216	54	18014 39850 94819 84
25	33554 432	55	36028 79701 89639 68
26	67108 864	56	72057 59403 79279 36
27	13421 7728	57	14411 51880 75855 872
28	26843 5456	58	28823 03761 51711 744
29	53687 0912	59	57646 07523 03423 488
30	10737 41824	60	11529 21504 60684 6976

GLOSSARY

The following glossary is a selection of frequently occurring simulation terms whose definitions are essential to understand or even converse in the military operations research world.

The computer terminology is based on the ADP Glossary, NAVSO P-3097, Department of the Navy, December 1970.

absolute address
(1) An address that is permanently assigned by the machine designer to a storage location.
(2) A pattern of characters that identifies a unique storage location without further modification.

access time
(1) The time interval between the instant at which data are called for from a storage device and the instant delivery begins.
(2) The time interval between the instant at which data are requested to be stored and the instant at which storage is started.

accumulator
A register in which the result of an arithmetic or logic operation is formed.

accuracy
The degree of freedom from error, i.e., the degree of conformity to truth or to a rule. Accuracy is contrasted with precision. For example, four place numerals are less precise than six place numerals, nevertheless a properly computed four place numeral might be more accurate than an improperly computed six place numeral.

address
An identification as represented by a name, label, or number for a register, location in storage, or any other data source.

ADP
Automatic Data Processing

air defense
All defensive measures designed to destroy attacking enemy aircraft or missiles in the earth's atmosphere.

ALGOL
ALGOrithmic Language. A language primarily used to express computer programs.

algorithm
A prescribed set of all the letters in a language, including letters with diacritical signs where appropriate; punctuation marks are not part of an alphabet.

alphameric
same as alphanumeric

alphanumeric
Pertaining to a character set that contains letters, digits, and usually other characters such as punctuation marks. Synonymous with alphameric.

American Standard Code for Information Interchange
A standard seven-bit coded character set developed by the American National Standards Institute (ANSI) to be used for information interchange among information processing systems, communications systems, and associated equipment. Abbreviated ASCII. Same as USASCII.

analog
Pertaining to representation by means of continuously variable physical quantities. Contrast with digital.

analog computer
(1) A computer in which analog representation of data is mainly used.
(2) A computer that operates on analog data by performing physical processes on these data. Contrast with digital computer.

analysis
The methodical investigation of a problem, and the separation of the problem into smaller related units for further detailed study.

analyst
A person who defines problems and develops algorithms and procedures for their solution.

arithmetic unit
The unit of a computing system that contains the circuits that perform arithmetic operations.

array
An arrangement of elements in one or more dimensions.

artificial intelligence
The capability of a device to perform functions that are normally associated with human intelligence, such as reasoning, learning, and self-improvement. Related to machine learning.

artificial language
A language based on a set of prescribed rules that are established prior to its usage. Contrast with natural language.

artillery
Complete projectile-firing weapons consisting of cannon or missile launchers on suitable carriages or mounts. Field artillery cannons are classified according to caliber as: light-120mm and less medium-121-160mm, heavy-161-210mm, very

heavy-greater than 210mm.

assemble
To prepare a machine language program from a symbolic language program by substituting absolute operation codes for symbolic operation codes and absolute or relocatable addresses for symbolic addresses.

assembler
A computer program that assembles.

assembly language.
(1) A machine-oriented programming language which belongs to an assembly program or system.
(2) In writing instructions using an assembly language, the programmer is primarily concerned with a label field, an operation field, and an operand field. It is possible to relate the symbolic coding to its associated flowchart if desired, by appending comments to each instruction line or program segment.

attenuation
Decrease in intensity of a signal, beam, or wave as a result of absorption of energy and of scattering out of the path of a detector, but not including the reduction due to geometric spreading, i.e., the inverse square of distance effect.

attributer
Descriptive parameters associated with entities.

attrition
The reduction of the effectiveness of a force caused by loss of personnel and material.

batch processing
Is processing, without unscheduled interruption, of a group of items prepared or required for one or more related operations.

BCD
Binary-coded decimal notation.

bias
The amount by which the average of a set of values departs from a reference value.

binary
(1) Pertaining to a characteristic or property involving a selection, choice, or condition in which there are two possibilities.

(2) Pertaining to the number representation system with a radix of two.

binary code
A code that makes use of exactly two distinct characters, usually 0 and 1.

binary coded decimal notation
Positional notation in which the individual decimal digits expressing a number in decimal notation are each represented by a binary numeral; e.g., the number twenty-three is represented 0010 0011 in the 8-4-2-1 type of binary-coded decimal notation and by 1011 in binary notation. Abbreviated BCD.

binary search
A dichotomizing search in which the number of items of the set is divided into two equal parts at each step of the process. Appropriate adjustments are usually made for dividing an odd number of items.

bit
A binary digit. Same as shannon. See check bit, information bits, parity bit, sign bit.

block
(1) A set of things, such as words, characters, or digits handled as a unit.
(2) A collection of contiguous records recorded as a unit. Blocks are separated by block gaps and each block may contain one or more records.

block diagram
A diagram of a system, instrument, or computer in which the principal parts are represented by suitably associated geometrical figures to show both the basic functions and the functional relatioships among the parts. Contrast with flowcharts.

buffer
A routine or storage used to compensate for a difference in rate of flow of data, or time of occurrence of events when transmitting data, from one device to another.

bug
A mistake or malfunction.

byte
A sequence of adjacent binary digits used to represent a character (usually 6 or 8 bits long).

card column
(1) A single line of punch positions parallel to the short edge of a 3 1/4 by 7 3/8 inch punched card.
(2) Positions in a line parallel to the vertical edge of the card.

card punch
A device to record information in cards by punching holes in the cards to represen letters, digits, and special characters.

card reader
A device which senses and translates into internal form the holes in punched cards.

cathode ray tube display
A device that presents data in visual form by means of controlled electron beams. Abbreviated CRT display.

chad
The piece of material removed when forming a hole or notch in a storage medium such as punched tape or punched cards. Synonymous with chip.

character
A letter, digit, or other symbol that is used as part of the organization, control, or representation of data. A character is often in the form of a spatial arrangement of adjacent or connected strokes.

characteristic
The integral part of a logarithm. For example, in the expression, log 643=2.808, the .808 is the mantissa and the 2 is the characteristic.

character recognition
The identification of graphic, phonic, or other characters by automatic means.

closed loop
A loop from which there is no exit other than by intervention from outside the program.

closed shop
Pertaining to the operation of a computer facility in which most productive problem programming is performed by a group of programming specialists rather than the problem originators. The use of the computer itself may also be described as closed=shop if full time trained operators, rather than user/programmers serve as the operators. Contrast with open shop.

code
A set of unambiguous rules specifying the way in which data may be represented, e.g., the set of correspondences in the standard Synonymous with coding scheme.

Command and control system
The facilities, equipment, communications, procedures, and personnel essential to a commander for planning, directing, and controlling operations of assigned forces pursuant to the missions assigned.

command language
A source language consisting primarily of procedural operators, each capable of invoking a function to be executed.

common business oriented language
A specific language by which business data processing procedures may be precisely described in a standard form. The language is intended not only as a means for directly presenting any business program to any suitable computer, for which a compiler exists, but also as a means of communicating such procedures among individuals. Commonly referred to as COBOL.

common field
A field that can be accessed by two or more independent routines.

compile
To prepare a machine language program from a computer program written in another programming language by making use of the overall logic structure of the program, or generating more than one machine instruction for each symbolic statement, or both, as well as performing the function of an assembler.

compiler
A program that compiles.

complement
A number that can be derived from a specified number by subtracting it from a second specified number. For example, in radix notation, the second specified number may be a given power of the radix or one less than a given power of the radix. The negative of a number is often represented by its complement.

computer
A data processor that can perform substantial computation, including numerous arithmetic or logic operations, without intervention by a human operator during the run.

computer program
A series of instructions or statements, in a form acceptable to a computer, prepared in order to achieve a certain result.

computer word
A sequence of bits or characters treated as a unit and capable of being stored in one computer location. Synonymous with machine word.

conditioned event
Events used to trigger events caused by a new system state.

congruence
The congruence x≡y (mod m) is read X is congruent to y modulo m and means that (x-y) is divisible by m.

console
That part of a computer used for communication between the operator or maintenance engineer and the computer.

control card
A punched card containing input data or parameters for initializing or modifying a program.

control program
A collective or general term for all routines in the operating system that contribute to the management of resources, implement the data organization or communications conventions of the operating system, or contain privileged operations.

copy
To reproduce data in a new location or other destination, leaving the source data unchanged, although the physical form of the result may differ from that of the source. For example, to copy a deck of cards onto a magnetic tape. Contrast with duplicate.

counter
A device such as a register or storage location used to represent the number of occurrences of an event.

CPU
Central Processing Unit.

data
A representation of facts, concepts, or instructions in a formalized manner suitable for communication, interpretation, or processing by humans or automatic means.

debug
To detect, locate, and remove mistakes from a routine or malfunctions from a computer.

deck
A collection of punched cards. Synonymous with card deck.

diagnostic
Pertaining to the detection and isolation of a malfunction of mistake.

digit
(1) A symbol that represents one of the non-negative integers smaller than the radix. For example, in decimal notation, a digit is one of the characters from 0 to 9.
(2) Synonymous with numeric character.

digital computer
(1) A computer in which discrete representation of data is mainly used.
(2) A computer that operates on discrete data by performing arithmetic and logic processes on these data. Contrast with analog computer.

documentation
(1) The creating, collecting, organizing, storing, citing, and disseminating of documents or the information recorded in documents.
(2) A collection of documents or information on a given subject.

double precision
Pertaining to the use of two computer words to represent a number.

dummy
Pertaining to the characteristic of having the appearance of a specified thing but not having the capacity to function as such. For example, a dummy character, dummy plug, or a dummy statement.

dump
To copy the contents of all or part of a storage, usually from an internal storage into an external storage.

dynamic programming
In operations research, a procedure for optimization of a multistage problem solution wherein a number of decisions are

available at each stage of the process.

electromagnetic radiation
Radiation made up of oscillating electric and magnetic fields and propagated with the speed of light. Includes gamma radiation, X-rays, ultraviolet, visible and infrared radiation, and radar and radio waves.

electronic counter-countermeasures
That major subdivision of electronic warfare involving actions taken to insure our own effective use of electromagnetic radiations despite the enemy's use of countermeasures.

electronic countermeasures
That major subdivision of electronic warfare involving actions taken to prevent or reduce the effectiveness of enemy equipment and tactics employing or affected by electromagnetic radiations and to exploit the enemy's use of such radiations.

electronic deception
The deliberate radiation, reradiation, alteration, absorption, or reflection of electromagnetic radiations in a manner intended to mislead an enemy in the interpreation of data received by his electronic equipment or to present false indications to electronic systems.

electronic jamming
The deliberate radiation, reradiation, or reflection of electromagnetic signals with the object of impairing the use of electronic devices by the enemy.

electronic warfare
Military action involving the use of electromagnetic energy to determine, exploit, reduce or prevent hostile use of the electromagnetic spectrum and action which retains friendly use of the electromagnetic spectrum. Also called EW. There are three divisions within electronic warfare:

1. electronic warfare support measures--That division of electronic warfare involving actions taken to search for, intercept, locate, and identify immediately radiated electromagnetic energy for the purpose of immediate threat recognition. Thus, electronic warfare support measures provide a source of information required for immediate action involving electronic countermeasures, electronic counter-countermeasures, avoidance, targeting, and other tactical employment of forces. Also called ESM.

2. electronic countermeasures--That division of electronic warfare involving actions taken to prevent or reduce an enemy's effective use of the electromagnetic spectrum. Also called ECM. Electronic countermeasures include:
   a. electronic jamming--The deliberate radiation, re-radiation, or reflection of electromagnetic energy devices, equipment, or systems being used by an enemy.
   b. electronic deception--The deliberate radiation, re-radiation, alteration, absorption, or reflection of electromagnetic energy in a manner intended to mislead an enemy in the interpretation or use of information received by his electronic systems. There are two categories of electronic deception: (1) manipulative deception--The alteration or simulation of friendly electromagnetic radiations to accomplish deception. (2) imitative deception--The introduction of radiations into enemy channels which imitate his own emissions.
3. electronic counter-countermeasures--That division of electronic warfare involving actions taken to insure friendly effective use of the electromagnetic spectrum despite the enemy's use of electronic warfare. Also called ECCM.

emulate
To imitate one system with another such that the imitating system accepts the same data, executes the same programs, and achieves the same results as the imitated system. Contrast with simulate.

entity
An object by which the system can be defined, i.e., a component of the system represented in the model.

error correcting code
A code in which each acceptable expression conforms to specific rules of construction that also define one or more equivalent nonacceptable expressions, so that if certain errors occur in an acceptable expression the result will be one of its equivalents and thus the error can be corrected.

error detecting code
A code in which each expression conforms to specific rules of construction, so that if certain errors occur in an expression, the resulting expression will not conform to the rules of construction and, thus, the presence of the errors is detected. Synonymous with self-checking code.

Euler's function
In modular arithmetic the number of positive integers less than m and relatively prime to m. Denoted $\phi(m)$. For m prime, $\phi(m) = m-1$.

event
Points of initiation, alteration, or conclusion of activities.

Fibonacci series
A series of integers in which each integer is equal to the sum of the two preceding integers in the series. The series is formulated mathematically by $x_i=x_{i-1}+x_{i-2}$, where $X_0=0$, $x_1=1$, i.e., 0,1,1,2,3,5,8,13,21...

field
In a record, a specified area used for a particular category of data, e.g., a group of card columns used to represent a wage rate set of bit locations in a computer word used to express the address of the operand.

FIFO
First-in-First-out priority test.

first generation computer
A computer utilizing vacuum tube components.

fixed point arithmetic
A method of calculation in which operations take place in an invariant manner, and in which the computer does not consider the location of the radix point. This is illustrated by desk calculators or slide rules, with which the operator must keep track of the decimal point. Similarly with many automatic computers, in which the location of the radix point is the programmer's responsibility.

flag
(1) Any of various types of indicators used for identification, e.g., a wordmark.
(2) A character that signals the occurrence of some condition, such as the end of a word.
(3) Synonymous with mark, sentinel, tag.

Flight Simulator
An electronic device to simulate the entire characteristics of the flight and operation of military aircraft. It is used to test and check out pilots in the operation of aircraft and the use of electronic equipment employed in actual flight tactics and combat operation.

floating point arithmetic
A method of calculation which automatically accounts for the location of the radix point. This usually is accomplished by handling the number as a signed mantissa times the radix raised to a integral exponent, i.e., the decimal number to an integral exponent, i.e., the decimal number to an integral exponent, i.e., the decimal number +88.3 might be written .883x102; the binary number +.0011 as +.11x22.

flowchart
A graphical representation for the definition, analysis, or solution of a problem, in which symbols are used to represent operations, data, flow, equipment, etc. Contrast with block diagram.

FORTRAN
FORmula TRANslating system. A language primarily used to express computer programs by arithmetic formulas.

general purpose computer
A computer that is designed to handle a wide variety of problems.

ground controlled interception
A radar technique which permits control of friendly aircraft or guided missiles for the purpose of effecting interception.

guided missile
An unmanned vehicle moving above the surface of the earth, whose trajectory of flight path is capable of being altered by an external or internal mechanism.

guided missile (air-to-air)
An air-launched guided missile for use against air targets.

guided missile (air-to-surface)
An air-launched guided missile for use against surface targets.

guided missile (surface-to-surface)
A surface-launched guided missile for use against surface targets.

half-word
A contiguous sequence of bits or characters which comprises half a computer word and is capable of being addressed as a unit.

hard copy
A printed copy of machine output in a visually readable form, for example, printed reports, listings, documents, etc.

hardware
Physical equipment, as opposed to the computer program or method of use, e.g., mechanical, magnetic, electrical, or electronic devices. Contrast with software.

header card
(1) A card that contains information related to the data in cards that follow.
(2) A card that contains supplemental information related to the data on the succeeding card(s). Contrast with trailer card.

heuristic
Pertaining to exploratory methods of problem solving in which solutions are discovered by evaluation of the progress made toward the final result. Contrast with algorithm.

hexadecimal
Same as sexadecimal.

high order position
Pertaining to the weight or significance assigned to the digits of a number, i.e., in the number 267349 the highest order digit is 2, the lowest order digit is 9.

Hollerith
Pertaining to a particular type of code or punched card utilizing 12 rows per column and usually 80 columns per card.

initialize
To set counters, switches, and addresses to zero or other starting values at the beginning of, or at prescribed points in, a computer routine.

input
Pertaining to a device, process, or channel involved in the insertion of data or states, or to the data or states involved.

instruction
A statement that specifies an operation and the values or locations of its operands. In this context, the term instruction is preferable to the terms command or order which are sometimes used synonymously.

integer programming
In operations research, a class of procedures for locating the maximum or minimum of a function subject to constraints, where some or all variables must have integer values.

interceptor
A manned aircraft utilized for identification and/or engagement of airborne objects.

interface
A shared boundary. An interface might be a hardware component to link two devices or it might be a portion of storage or registers accessed by two or more computer programs.

interpreter
(1) A computer program that translates and executes each source language statement before translating and executing the next one.

(2) A device that prints on a punched card the data already punched in the card.

I/O
An abbreviation for input/output.

job control statement
A statement in a job that is used in identifying the job or describing its requirements to the operating system.

keypunch
A keyboard actuated device that punches holes in a card to represent data.

language
A set of representations, conventions, and rules used to convey information.

left-justify
(1) To adjust the printing positions of characters on a page so that the left margin of the page is regular.
(2) By extension, to shift the contents of a register so that the most significant digit is at some specified position of the register. Contrast with normal size.
(3) To align characters horizontally so thatthe left-most character of a string is in a specified position.

Life cycle cost
The total cost of an item or system over its full life. It includes the cost of development, acquisition, ownership (operation, maintenance, support, etc.) and, where applicable, disposal.

LIFO
Last-in-First-out priority basis.

line printer
A device that prints all characters of a line as a unit. Contrast with character printer.

linear programming
In operations research, a procedure for locating the maximum or minimum of a linear function of variables that are subject to linear constraints. Synonymous with linear optimization. Abbreviated LP.

load
In programming, to enter data into storage or working registers.

load-and-go
An operating technique in which there are no stops between the loading and execution phases of a program, and which may include assembling or compiling.

logical record
A collection of items independent of their physical environment. Portions of the same logical record may be located in different physical records.

loop
A sequence of instructions that is executed repeatedly until a terminal condition prevails.

low order position
Pertaining to the weight of significance assigned to the digits of a number, i.e., in the number 396148, the low order digit is 8.

machine instruction
An instruction that a machine can recognize and execute.

machine language
(1) A language that is used directly by a machine.
(2) The set of instructions expressed in the number system basic to a computer, together with symbolic operation codes with absolute addresses, relative addresses, or symbolic addresses.

macro instruction
An instruction in a source language that is equivalent to a specified sequence of machine instructions.

magnetic core
A configuration of magnetic material that is, or is intended to be, placed in a spatial relationship to current-carrying conductors and whose magnetic properties are essential to its use. It may be used to concentrate an induced magnetic field as in a transformer induction coil, or armature, to retain a magnetic polarization for the purpose of storing data, or for its nonlinear properties as in a logic element. It may be made of such material as iron, iron oxide, or ferrite and in such shapes as wires, tapes, toroids, rods, or thin film.

magnetic disc
A flat circular plate with a magnetic surface on which data can be stored by selective magnetization of portions of the flat surface.

magnetic drum
A right circular cylinder with a magnetic surface on which data can be stored by selective magnetization of portions of the curved surface.

magnetic storage
A storage device that utilizes the magnetic properties of materials to store data, e.g., magnetic cores, tapes, and films.

magnetic tape
(1) A tape with a magnetic surface on which data can be stored by selective polarization of portions of the surface.
(2) A tape of magnetic material used as the constituent in some forms of magnetic cores.

main frame
Same as central processing unit.

management
A process of establishing and attaining objectives to carry out responsibilities. Management consists of those continuing actions of planning, organizing, directing, coordinating, controlling, and evaluating the use of men, money, materials, and facilities to accomplish missions and tasks.

mantissa
The fractional part of a logarithm. In the expression, log 643=2.808, the .808 is the mantissa and the 2 is the characteristic.

Markov chain
A probabilistic model of events in which the probability of an event is dependent only on the event that precedes it.

mask
(1) A pattern of characters that is used to control the retention or elimination of portions of another pattern of characters.
(2) A filter.

mass storage device
A device having a large storage capacity, e.g., magnetic disc, magnetic drum.

mathematical model
A mathematical representation of a process, device, or concept.

matrix
(1) In mathematics, a two-dimensional rectangular array of quantities. Matrices are manipulated in accordance with the rules of matrix algebra.

(2) In computers, an array of any number of dimensions.

median
An average of a series of quantities or values; specifically, the quantity or value of that item which is so positioned in the series, when arranged in order of numerical quantity or value, that there are an equal number of items of greater magnitude and lesser magnitude.

memory
Same as storage.

mini computer
Generally it is a small, general-purpose digital computer with a central processor and core memory (approximately 4096 words), and weighs about 85 pounds. The cabinet holding these two components would be about the size of an electric typewriter. It has a small word-size, 12 bit or less, and is economically priced, usually from $4,000 to $13,000. Mostly they are used in the on-line, real-time environment and are built into larger systems as special-purpose data reducers and controllers. Current machines vary in word length , input/output facilities, instruction sets, software, and performance.

mnemonic symbol
A symbol chosen to assist the human memory, e.g., an abbreviation such as "mpy: for multiply.

model
Any representation of a real world system.

Monte Carlo method
A method of obtaining an approximate solution to a numerical problem by the use of random numbers, e.g., the random walk method.

nanosecond
A billionth of a second.

natural language
A language whose rules reflect and describe current usage rather than prescribe usage. Contrast with artificial language.

noise
(1) Random variations of one or more characteristics of any entity such as voltage, current, or data.

(2) A random signal of known statistical properties of amplitude, distribution, and spectral density.
(3) Loosely, any disturbance tending to interfere with the normal operation of a device or system.

object code
Output from a compiler or assembler which is itself executable machine code or is suitable for processing to produce executable machine code.

object program
A fully compiled or assembled program that is ready to be loaded into the computer. Synonymous with target program. Contrast with source program.

OCR
Optical character recognition.

octal
(1) Pertaining to a characteristic or property involving a selection, choice or condition in which there are eight possibilities.
(2) Pertaining to the number representation system with a radix of eight.

offline
(1) Pertaining to equipment or devices not under control of the central processing unit.
(2) Descriptive of a system and of the peripheral equipment of devices in a system in which the operation of peripheral equipment is not under the control of the central processing unit.

online
(1) Pertaining to equipment or devices under control of the central processing unit.
(2) Pertaining to a user's ability to interest with a computer.

open shop
Pertaining to the operation of a computer facility in which most productive problem programming is performed by the problem originator rather than by a group of programming specialists. The use of the computer itself may also be described as open shop if the program user/programmer also serves as the operator, rather than a full time trained operator. Contrast with closed shop.

operating system
Software which controls the execution of computer programs and which may provide scheduling, debugging, input/output control, accounting, compilation, storage assignment, data management, and related services.

operations research
The use of the scientific method to provide criteria for decisions concerning the actions of people, machines, and other resources in a system involving repeatable operations. Synonymous with operations analysis.

optical character recognition
The machine identification of printed characters through use of light-sensitive devices. Contrast with magnetic ink character recognition. Abbreviated OCR.

order
In modular arithmetic, the order of X modulo m is the least positive exponent n with $x^n \equiv 1 \pmod{M}$ where x and m are relatively prime.

output
Pertaining to a device, process, or channel involved in an output process, or to the data or states involved.

overflow
(1) That portion of the result of an operation that exceeds the capacity of the intended unit of storage.
(2) Pertaining to the generation of overflow as in (1).
(3) Contrast with underflow.

overlay
The technique of repeatedly using the same blocks of internal storage during different stages of a program. When one routine is no longer needed in storage, another routine can replace all or part of it.

pack
To compress data in a storage medium by taking advantage of known characteristics of the data, in such a way that the original data can be recovered, e.g.,.to compress data in a storage medium by making use of bit or byte locations that would otherwise go unused.

page
A segment of a computer program which has a virtual address and can be located in main storage or in auxiliary storage.

paging
The scheme used to locate pages to move them between main storage and auxiliary storage or to exchange them with pages of the same or other computer programs.

parameter
A variable that is given a constant value for a specific purpose or process.

pebetration
A form of offensive maneuver which seeks to break through the enemy's defensive position, widen the gap created, and destroy the continuity of his postions.

penetration aids
Techniques and/or devices employed by aerospace systems to increase the probability of weapon system penetration of an enemy defense. Examples are: low altitude flight profiles, trajectory adjustments, reduced radar cross-sections of attack vehicles, improved vehicle hardness to effects of defense engagements, terrain avoidance radar, bomber defense missiles, decoys, chaff, electronic countermeasures, etc. Penetration aids are used by an offensive system to penetrate more effectively enemy defenses.

peripheral and auxiliary equipment
Card and tape input-output devices, printers for hard copy output drums and discs for auxiliary memory storage, magnetic tape to microfilm devices, optical character recognition equipment (which may also be in the computer category), cathode ray tube displays, transaction recorders, magnetic ink character recognition equipment, sophisticated plotter-display output in a photographic or pseudophotographic form, and photographic and magnetic sheet, strip, or ship memory storage devices. Automatic data processing data transmission facilities also come within this broader definition when connected to an on-base computer.

Polish notation
Same as prefix notation.

possible
A term used to qualify a statement made under conditions wherein some evidence exists to support the statement. This evidence is sufficient to warrant mention, but insufficient to warrant assumption as true.

prefix notation
A method of forming mathematical expressions in which each operator precedes its operands. For example, in prefix notation, the expression "(a plus b) multiplied by c" could be represented by +abc. Synonymous with Lukasiewicz notation, parentheses-free notation, Polish notation.

primary event
Events which may cause the creation of a new future event regardless of the system state.

primative root
In modular arithmetic, the primative root of m is a number whose order is equal to $\emptyset$ (m), [Euler's function]

probability
A mapping of the real line into the interval [0,1] on the real line.

probable
A term used to qualify a statement made under conditions' wherein the available evidence indicates that the statement is factual until there is further evidence in confirmation or denial.

problem oriented language
A programming language designed for the convenient expression of procedures used in the solution of a wide class of problems.

process
An activity that requires time.

program
(1) A series of actions proposed in order to achieve a certain result.
(2) Loosely, a routine.
(3) To design, write, and test a program as in (1).
(4) Loosely, to write a routine.

programmer
A person mainly involved in designing, writing, and testing computer programs.

programming flowchart
A flowchart representing the sequence of operations in a program.

programming language
A language used to prepare computer programs.

pseudorandom number sequence
A sequence of numbers, determined by some defined arithmetic process, that is satisfactorily random for a given purpose, such as by satisfying one or more of the standard statistical tests for randomness. Such a sequence may approximate any one of several statistical distributions, such as uniform, normal, or gaussian distribution.

punch card
A card suitable for punching in a pattern of holes to represent data.

radar
Radio detection and ranging equipment that determines the distance and usually the direction of objects by transmission and return of electromagnetic energy.

radar clutter
Unwanted signals, echoes, or images on the face of the display tube which interfere with observation of desired signals.

radar coverage
The limits within which objects can be detected by one or more radar stations.

radar echo
The signal indication of an object which has reflected energy transmitted by a radar.

radix
In positional representation, the integer, if it exists, by which the significance of the digit place must be multiplied to give the significance of the next higher digit place. For example, in decimal notation, the radix of each place is ten; of the fives place is two. Synonymous with base.

random access
Same as direct access.

random variable
A function mapping the sample space into the real line.

range
The difference between the highest and lowest value that a quantity or function may assume.

real time
Pertaining to the performance of a computation during the actual time that the related physical process transpires, in order that results of the computation can be used in guiding the physical process.

register
A device capable of storing a specified amount of data such as one word.

replication
repeating a run of a program for the same combination of parameters but with different random variations.

roundoff
To delete the least significant digit or digits of a numeral, and to adjust the part retained in accordance with some rule.

routine
An ordered set of instructions that may have some general or frequent use.

run
A single, continuous performance of a computer program for a specified combination of parameters.

scan
To examine sequentially, part by part.

search
To examine a set of items for one or more having a desired property.

second generation computer
A computer using solid state components.

semantics
The relationships between symbols and their meanings.

simulate
To represent certain features of the behavior of a physical or abstract system by the behavior of another system.

simulation
The representation of certain features of the behavior of a physical or abstract system by the behavior of another system, e.g., the representation of physical phenomena by means of operations performed by a computer or the representation of operations of a computer or the representation of operations of a computer by those of another computer.

simulator
In electronic warfare, a man-machine tool which combines operators, analog/digital computers, and actual hardware to simulate a real world system.

skew
The angular displacement of a symbol or data medium from the intended or ideal placement.

small arms
All arms, including automatic weapons, up to and including .60 caliber and shotguns.

smooth

To apply procedures that decrease or eliminate rapid fluctuations in data.

software

The programs and routines used to extend the capability of automatic data processing equipment. The types of software are as follows:

a. Basic Software comprises those routines and programs designed to extend or facilitate the use of particular automatic data processing equipment, the requirement for which takes into account the design characteristics of such equipment. This software is usually provided by the original equipment manufacturer and is normally essential to and a part of the system configuration furnished by him. Examples of basic software are executive and operating programs; diagnostic programs; compilers; assemblers, utility routines, such as sort-merge and input/output conversion routines; file management programs and data management programs. Data management programs are commonly linked to, and/or under the control of, the executive or operating programs.

b. Application Software consists of those routines and programs designed by or for automatic data processing equipment users to accomplish specific, mission-oriented tasks, jobs or functions using the automatic data processing equipment and basic software available. Applications software may be either general purpose packages, such as demand-deposit accounting, payroll, machine tool control, etc.; or specific application programs tailored to accomplish a single or limited number of users' functions, such as base level personnel, depot maintenance, missile or satellite tracking, etc. Except for general purpose packages which are acquired directly from software vendors or from the original equipment manufacturers, this type of software is normally developed by the user, either with in-house resources or through contract services.

sort

To segregate items into groups according to some definite rules. Same as order.

source language

The language from which a statement is translated.

source program

A computer program written in a source language. Contrast with objective program.

special purpose computer
A computer that is designed to handle a restricted class of problems.

statistic
A random variable whose values are determined by sample data.

storage
Pertaining to a device into which data can be entered, in which they can be held, and from which they can be retrieved at a later time. Synonymous with memory.

string
A linear sequence of entities such as characters of entities such as characters of physical element.

syntax
(1) The structure of expressions in a language.
(2) The rules governing the structure of a language.

system
A collection of identifiable parts capable of interacting in such a way that the entire collection functions together.

table look-up
A procedure for obtaining the function value corresponding to an argument from a table of function values.

third generation computer
A computer utilizing solid logic technology components.

time sharing
(1) Pertaining to the interland use of the time of a device.
(2) A computer operation, under control of an executive routine incorporating a scheduling priority algorithm, which effectively enables computer availability to a multitude of users virtually simultaneously.

translate
To transform statements from one language to another without significantly changing the meaning.

translator
A program which translates from one programming language into another programming language.

transliterate
To convert the characters of one alphabet to the corresponding characters of another alphabet.

truncate
To terminate a computational process in accordance with some rule, e.g., to end the evaluation of a power series at a specified term.

underflow
Pertaining to the condition that arises when a machine computation yields a nonzero result smaller than the smallest nonzero quantity that the intended unit of storage is capable of storing. Contrast with overflow.

user
Anyone who requires the use of services of a computing system or its products.

validation
Testing the agreement between the behavior of a simulation model and the real world system.

variable
A quantity that can assume any of a given set of values.

verification
Ensuring that a simulation model behaves as the analyst intends.

verify
(1) To determine whether a transcription of data or other operation has been accomplished accurately.
(2) To check the results of keypunching.

virtual address
A symbol that can be used as a valid address part but does not necessarily designate an actual location.

vulnerability
(1) The susceptibility of a nation or military force to any action by any means through which its war potential or combat effectiveness may be reduced or its will to fight diminished.
(2) The characteristics of a system which causes it to suffer a definite degradation (incapability to perform the designated mission) as a result of having been subjected to a certain level of effects in unnatural (manmade) hostile environment.

war game
A simulation, by whatever means, of a military operation involving two or more opposing forces, using rules, data, and procedures designated to depict an actual or assumed real life situation.

word

A character string or a bit string considered as an entity; normally the smallest entity which can be fetched from, stored in, or addressed in memory.

word length

A measure of the size of a word, usually specified in units such as characters or binary digits.

write

To record data in a storage device or a data medium. The recording need not be permanent, such as the writing on a cathode ray tube display device.

INDEX

Algorithms, 24
  English Language, 25
  Flowchart, 26
  FORTRAN, 25

Analysis, 16

Antithetic variates, 109
ASCII characters, 197
Attributes, 80
Autocorrelation, 108

Binomial variates, 73

Chi square test, 50
Closed loop simulation, 15
Confidence intervals, 113
Correlation coefficient, 55
Cost effectiveness analysis, 2
Coupon collector test, 58

Decision making:
  Classification of aids, 5
  Process, 4
Descriptive models, 7
Digital computer components, 20

Entities, 79
  Permanent, 79, 105
  Temporary, 180
Enumeration, 7
Event, 82
EW applications, 168
Exponential variate, 70

Flowchart, 26
Functional relationships, 81

Gamma variates, 75
Gap test, 61
Geometric variates, 70

Hardware, digital computer, 20
Hypothesis testing, 116

Initial state definition, 99

Initial values:
  Model state, 99
  Pseudorandom no. gen., 62
Intuition, 5
Inverse transformation, 69

Kolmogorov-Smirnov test, 53

Logistics applications, 180

Management science, 2
Man-in-the-loop, 15
Mathematical models, 7
Middle of the square, 40
Mixed congruential, 45
Monte carlo, 10, 37
Multiplicative congruential, 41

Negative binomial variate, 71
Nonparametric methods:
  Sign test, 56, 123
  Wilcoxon test, 125
Normal variates, 76

Open loop simulation, 15
Operations research:
  Basic themes, 3
  Definition, 1
Output generation, 96

Patrolling repairman problem, 92, 131
Permutation test, 58
Physical models, 10
Poisson variates, 74
Poker test, 57
Polynomials modulo 2, 47
Powers of two, 198
Probabilistic models:
  Expected value, 9
  Stochastic sampling, 10, 37
Probability distributions:
  Continuous, 195
  Discrete, 196
Pseudorandom numbers:
  Definition, 39
  Generation, 39

Randomness tests:
Chi square, 50
Correlation coefficient, 55
Coupon collector, 58
Gap, 61
Kolmogorov-Smirnov, 53
Permutation, 58
Poker, 57
Runs above & below the mean, 54
Runs up, 57
Serial, 60
Sign, 51, 123
Spectral, 59

Random variates:
Binomial, 73
Exponential, 70
Gamma, 75
Geometric, 70
Negative binomial, 71
Normal, 76
Poisson, 74
Uniform, 72
Rejection method, 73
Replication, 97
Runs above & below the mean, 54
Runs up, 57

Sample size determination, 110
Sampling procedures:
Permanent entities, 103
Temporary entities, 110
Search for maximum, 190
Serial test, 60
Sign test, 51, 123
Simulation:
Advantages, 29
Closed loop, 15
In EW, 168
In Logistics, 180
Military applications, 167
Model structure, 11, 169
Study guidelines, 30
Simulation languages, 157
GASP, 160
General Purpose, 158
GPSS, 160
SIMSCRIPT II, 163
SIMULA, 165

Software, digital computer, 22
Sorts:
Bubble, 187
Virtual, 189
Spectral test, 59
Steady state, 99, 102
Stratified sampling, 105
Synthesis, 16
System representation, 79
Systems analysis, 2
Systems engineering, 2

Table look-up, 192
Time:
Generation times for PRN, 63
State of the model, 82
Time flow mechanisms, 84
Fixed time increment, 84, 87
Next event, 87
Transformation effect, 35
Truncation, 101

Uniform variates, 72

Validation, 127
Verification, 127

War game, 11