# GPSS

## Simulation Made Simple

**Thomas M. O'Donovan**
*Department of Statistics*
*University College, Cork*

*A Wiley—Interscience Publication*

JOHN WILEY & SONS
Chichester · New York · Brisbane · Toronto

***British Library Cataloguing in Publication Data:***

O'Donovan, Thomas M.
GPSS simulation made simple.
—(Wiley series in computing)
1. GPSS (Computer program language)
1. Title
001.4'24 QA76.73.G18 79–40520
ISBN 0 471 27614 6

Typeset by Preface Ltd., Salisbury, Wiltshire and Printed in Great Britain by Page Bros. (Norwich) Ltd., Norwich

# GPSS

## *WILEY SERIES IN COMPUTING*

*Consulting Editor*
**Professor D. W. Barron,** *Department of Mathematics, Southampton University*

**Numerical Control—Mathematics and Applications**

*P. Bezier*

**Communication Networks for Computers**

*D. W. Davies and D. L. A. Barber*

**Macro Processors and Techniques for Portable Software**

*P. J. Brown*

**A Practical Guide to Algol 68**

*Frank G. Pagan*

**Programs and Machines**

*Richard Bird*

**The Codasyl Approach to Data Base Management**

*T. William Olle*

**Computer Networks and their Protocols**

*D. W. Davies, D. L. A. Barber, W. L. Price, and C. M. Solomonides*

**Algorithms: Their Complexity and Efficiency**

*Lydia Kronsjö*

**Data Structures and Operating Systems**

*Teodor Rus*

**Writing Interactive Compilers and Interpreters**

*P. J. Brown*

**The Design of a Relational Data Base Management System**

*A. T. F. Hutt*

**GPSS**
**Simulation Made Simple**

*Thomas M. O'Donovan*

*To*
*Vera*

# *Preface*

The objective of this book is to introduce the concepts of discrete-event simulation using the programming language GPSS. A discrete system is one in which only discontinuous changes of state occur. The term event is used to mean such a change. In discrete-event simulation, the system is only studied at time points when changes of state occur. GPSS (which stands for *G*eneral *P*urpose *S*imulation *S*ystem) is a programming language which was specially designed for discrete-event simulation. Programming such simulations in GPSS is very simple, which makes GPSS an excellent medium for teaching the concepts of discrete-event simulation since students can spend more time studying the concepts instead of worrying about details of programming. No prior programming experience is needed to learn GPSS, though a knowledge of elementary probability is necessary to understand discrete-event simulation.

This book uses a new approach to introduce the concepts of discrete-event simulation. The stages in any simulation problem are first described. Hand simulation of a simple queue is used to illustrate how simulation works. The point is made that computer simulation performs exactly the same calculations but much more quickly. A unique step-by-step approach is used to introduce the features of GPSS. The emphasis is not on describing what GPSS is but on showing what it can do. However, enough explanatory detail is given to enable the student to make correct use of GPSS. A series of examples of standard queueing systems is considered, ranging from the simple to the complicated. As each new complication is introduced, the corresponding features of GPSS are described. In this way, students see what each feature is needed for and its precise function. In other textbook treatments of GPSS, the examples used introduce several new features of GPSS simultaneously and students may be confused as to the precise function of each. In this text, as soon as a student has studied several new features separately, his understanding of the use of these features is tested by numerous exercises to which the solutions are given.

The plan of the book is straightforward and should be clear from the table of contents. Finally, it should be mentioned that GPSS exists in many dialects. The most widely used is GPSS/360, which is the version discussed here.

I would like to thank Dr. P. D. Bourke of the Department of Statistics, University College, Cork, who read the manuscript and made many helpful

suggestions. I am also indebted to Mr. Michael Crowley, Statistical Computing Analyst, for help with the illustrations and to Ms. Norma Gallwey for her excellent typing.

THOMAS M. O'DONOVAN

# *Contents*

# *Chapter 1*

# *Simulating a single line, single server queue*

## 1.1 Simulation and its uses

Computer simulation is widely used today as a decision-making tool in business and industry. For the purposes of the present chapter, simulation may be taken to mean constructing a mathematical model of a physical system. While the idea of a mathematical model may seem strange to some readers, the example discussed in this chapter should make the concept clear. The basic requirement for this model is that it should 'behave' like the physical system it is supposed to be modelling. The advantages of having such a mathematical model will now be made clear. Suppose it is proposed to change an existing production line by installing new machinery or employing extra staff at various stages. Such changes will involve considerable expenditure and it would be very useful to estimate in advance the effect these changes will have on the production line. Simulation enables us to do this (see Figure 1.1). Suppose we have a mathematical model (model 1) for the existing production line and are confident that this model behaves like the present system. Then, by changing the model to reflect the proposed changes to the present system, we get a new model (model 2) which we hope will behave like the proposed production line. Thus, by studying the behaviour of our new model, we can forecast the behaviour of the proposed production line, without going to the expense of making the proposed changes. The only expense involved is that of building and running the simulation model. Once the simulation model has been built, the behaviour of the proposed system over a period of months or years can be simulated in a matter of seconds. Because of the speed of computer simulation, it is feasible to study a variety of proposed systems in turn and in this way to identify the changes necessary to optimize the performance of the existing system.

To avoid giving the impression that simulation is a cure-all, it should be said that building a simulation model for a complex system can be extremely costly in man-hours and computer time and thus simulation should only be used as a last resort when other techniques, such as queueing theory, have failed. The sort of simulation described here can only be used to model queueing systems. However, this is not as restrictive as it seems, because many systems can be regarded as queueing systems.

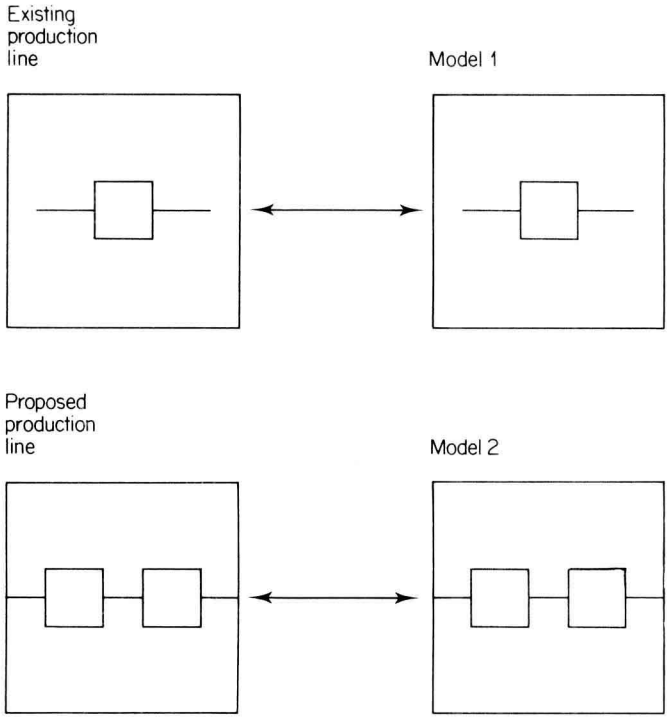


Figure 1.1

## 1.2 Stages in a simulation problem

The stages in any simulation problem may be briefly outlined as follows:

(a) Defining the problem
(b) Formulating a model
(c) Running the model
(d) Experimenting with the model

Later in this chapter it will be made clear what the first two stages consist of. Running the model involves checking to see that the simulation model behaves like the present system. If it fails to perform satisfactorily, we have to go back and formulate a different model and repeat the process until we find a model that does behave like the present system. Only then can we proceed to the last stage, which is to experiment with the model to reflect proposed changes in the existing system. In this way, we can forecast the performance of these proposed systems.

## 1.3 A single line, single server queue

To illustrate what a simulation model looks like and what use can be made of it, we consider a simple queueing system: a barbershop. Although this example is

very simple, it illustrates many of the elements in a general simulation problem and the stages in its solution.

Joe has a one-man barbershop. If a customer arrives and finds Joe free, he gets served immediately. If Joe is busy when a customer arrives, that customer joins a waiting line of customers seated on chairs. Waiting customers are served first-come-first-served, i.e. when Joe becomes free, he serves the customer who has been waiting longest.

Joe's problem is this. He has a good business but he is worried by the fact that some customers may have to wait too long. He knows that if waiting times are too long, some waiting customers may be inclined to leave without waiting to be served. It may also happen that customers may arrive and leave without waiting if they find too many customers before them. Joe has not noticed this happening yet, but he is worried about it. For this reason, Joe is considering getting one or more assistants, but he would like to know in advance the effect this would have in reducing average customer waiting time, i.e. the time a customer would have to wait on average. Simulation makes it possible to study this problem.

## 1.4 Defining the problem

The first stage in any simulation problem is to *define the problem*. This involves *identifying relevant variables* of various types. A *customer variable* is simply a customer characteristic that varies from one customer to the next. Examples of customer variables are interarrival times and haircut times. Suppose that the first three customers arrive at the barbershop 12, 30, and 53 minutes, respectively, after the shop is opened. Then the interarrival times for these three customers are 12, 18, and 23 minutes. Similarly, suppose that the times these three customers take to get a haircut are 18, 15, and 18 minutes, respectively. Another customer variable is customer waiting time. An example of a *system variable* is the number of customers waiting for service at a specified time point. However, these variables are not as *basic* as interarrival and haircut times because it can be shown that they are determined by interarrival and haircut times, as will be seen shortly. Interarrival and haircut times are known as *uncontrolled variables,* because it is not possible for Joe to influence the times between customer arrivals nor the time it takes to give a haircut. However, there is a variable that Joe can control: the number of barbers, i.e. he can decide to get one or more assistants. This variable is called a *decision variable.* To decide on the 'optimum value' of the decision variable in the light of the uncontrolled variables, we must have some criterion for measuring the performance of the system. Here the criterion that Joe is using is average customer waiting time. The problem can now be expressed as follows: how many barbers should there be to give an 'acceptable level' of average customer waiting time, for a given pattern of customer arrivals and haircut times? To summarize, defining the problem involves

(a) Identifying the basic uncontrolled variables

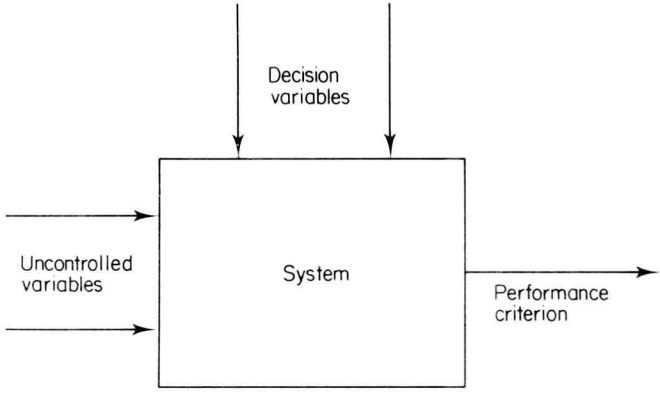


Figure 1.2

(b) Identifying the decision variable or variables
(c) Deciding on a performance criterion or criteria

The problem can then be expressed as follows: what value(s) of the decision variable(s) optimize system performance in the light of the basic uncontrolled variables? (See Figure 1.2.)

## 1.5 Formulating a model

Here the existing system is a barbershop with a single chair. The first step towards formulating a model for this is to draw a *flow diagram*. The purpose of the flow diagram is to show the sequence of events that occur as each customer passes through the shop. Recall that an *event* is just a change of state of the system. In this case, the sequence of events is easily written down (see Figure 1.3).

At each of the time points when such an event occurs, and only then, the state of the system changes. For example, when a customer arrives, the number of customers in the system increases by one; when a customer joins the waiting line, the number of waiting customers increases by one, etc. Of course, some of these events will occur simultaneously. For example, the first customer to enter the shop on a particular day will arrive, join the (empty) waiting line, leave the waiting line, and begin his haircut, all at the same time.

The flow diagram tells us the sequence in which events occur as each customer passes through the barbershop. In order to complete our model of the barbershop, we need to know the time points at which these events take place in the barbershop. As will be made clear in the next section, if we knew the interarrival times and the haircut times for each customer on a particular day in the barbershop, we could work out the times at which all the events occurred on that day and hence all statistics of interest, e.g., the average customer waiting time. However, the only way to find out about interarrival times and haircut times in the barbershop is to have records kept of the interarrival and haircut

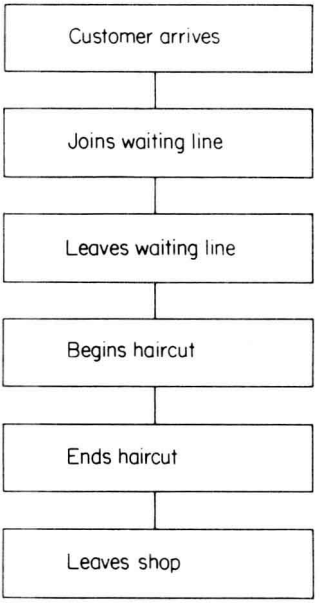


Figure 1.3

times for each customer over a period of days. Then it will be found that while interarrival and haircut times vary from day to day as well as from customer to customer on any particular day, when the data for several days is pooled, interarrival times follow a certain 'pattern of variability'; the same is true of haircut times. It will be shown later how to incorporate these 'the patterns of variability' into the model. At this stage, however, we adopt a simpler approach.

Suppose Joe tells us that the interarrival time between customers is 18 minutes 'on average' and that it takes 16 minutes to give a haircut, 'on average'. The simplest assumption we could make about interarrival and haircut times is that a customer arrives every 18 minutes and every haircut takes exactly 16 minutes, i.e., to allow no variability at all. This, however, is rather unrealistic. Suppose that Joe says that an interarrival time can be as short as 12 minutes or as long as 24 minutes and that a haircut time could be as short as 12 minutes or as long as 20 minutes. Then it would be reasonable to assume that the interarrival time for each customer is 'equally likely' to be any of the integers 12, 13, ..., 24. This could be achieved by having a thirteen-sided die, with one of the numbers 12 to 24 on each side. If the die is rolled and the number 15 comes up, then the next customer will arrive after 15 minutes. Alternatively, this could be achieved by using a table of random numbers. Those familiar with probability theory will realize that we are assuming that interarrival times are uniformly distributed in the range $18 \pm 6$ minutes. Similarly, we assume that haircut time for any customer is equally likely to be any of the intergers 12, 13, ..., 20, i.e. uniformly distributed in the range $16 \pm 4$ minutes. These assumptions may be highly unrealistic and bear no relationship to the actual 'patterns of