A Monte Carlo Simulation of a Production Planning Problem

by F. I. Musk

Summary: A computer program simulates the allocation of orders to a set of processing machines. In this setting, two aspects of the use of the Monte Carlo method are illustrated. Some details of the program are given, together with a note on computer facilities useful to work of this nature.

1 INTRODUCTION

How Monte Carlo methods are to be defined is dependent upon the field of reference. A definition acceptable to an operational research worker is unlikely to be accepted by a mathematical physicist. The simulation to be described here lies in the field of operational research, in which Monte Carlo methods are usually taken to mean obtaining data by sampling from models. The examples to be given of the use of the Monte Carlo method will refer to those aspects of the simulation concerned with model sampling.

The behaviour of a system is governed by certain laws, and in the course of its operation, it generates data, which are observations of particular states of the system taken at different times. A statement of the laws governing the system constitutes the "model," which may be hypothetical or real. If this statement includes laws of a statistical nature, a Monte Carlo method can be used to generate data, by "sampling" from the model. By "sampling" is meant applying random or pseudo-random numbers in such a way that they are transformed by the model to quite different numbers which represent data.

It may be postulated that data arising from the operation of a real system indicate the existence of a particular set of laws or restraints. This can be tested by generating data from this set and comparing them with data arising from the real system. On the other hand, we may be concerned with a real system operating under known conditions. For example, a group of machines is loaded in accordance with prescribed rules, and there is a large element of chance in the availability and form of the load. These rules can be compared with amended rules, by studying the data generated by sampling from models representing the behaviour of the system under both sets of rules. The program to be described is such an example.

Monte Carlo methods use random or pseudo-random sampling numbers; the requirements of such sampling numbers are outlined in Kendall and Babington Smith (1938). For automatic computation it is convenient to generate pseudo-random sampling numbers deterministically.

The multiplicative congruential method used was

$$U_{n+1} \equiv 23\ U_n \pmod{2^{35} + 1};$$

$$U_0 = 10{,}987{,}654{,}321.$$

This method was originated by Lehmer (1949).

It should not be supposed that use of the Monte Carlo method necessarily requires access to a computer, and much useful work has been carried out with no tools other than paper, pencil and tables of random numbers. The literature on the subject is not extensive, but some references to works describing practical applications of Monte Carlo methods are given at the end of this paper, for the benefit of readers who may wish to pursue the subject further.

2 THE SIMULATION PROGRAM

The intention was to simulate closely the normal production planning logic of a factory department in which a variety of product types, occurring as "customers' orders," are subjected to a particular kind of processing on a set of machines. A change in production policy is analogous to an alteration to the program, so that by running the program first as it is, and then in its amended form, using data representing orders entering the department, it is possible to assess, in units such as machine hours used, the results to be expected from a proposed change in production policy, without actually making the change. Such a program has great value where policy decisions carry elements of risk. The program can, in the same way, show the effect on the department of a change in the pattern of orders.

A detailed simulation of a plant operation, such as this, can be justified for a single major problem, or for a series of related problems, each of which may be investigated by modification of the main computer program. It can at the same time be used on a routine basis to perform the production planning of the department.

The problem presented two aspects:

(1) To represent the loading and running of the machines (the main program).
(2) To simulate the flow of customers' orders entering the department, to be used as data by the main program (data generation program).

The latter will be considered first.

2.1 *Data Generation*

Let us suppose that each order entering the processing department consists of a number of identical "packages." Each machine in the department has 100 positions. A package can be processed in each position, and the kind of processing given to the package is governed by the machine setting and certain physical characteristics of the package. The machine setting and package type also determine the average amount of time required to

Downloaded from https://academic.oup.com/comjnl/article/2/2/90/443688 by guest on 20 March 2024

process the package, and this average can be anything from 20 to 100 hours. Loading of an empty machine is staggered in such a way that processing of the first package is due to end just after the last package is placed in its position, and thereafter packages are replaced as they are processed. When a production run starts, therefore, and when it ends, there is a considerable loss of machine time, so that runs must be as long as possible.

The order data is available for a period of 50 weeks, but we require for the exercise a much longer period, say 500 weeks, and this has to be generated, using the Monte Carlo method. The orders can vary in three respects. There is variation in

(1) the total volume entering the department each week (in terms of machine capacity);
(2) the order types (physical characteristics of a package and kind of processing required);
(3) the number of packages in each order type.

With regard to the first source of variation, our available data enables us to build a step function, and hence by smoothing and substituting percentage probability for cumulative frequency, a distribution function as shown in Fig. 1.

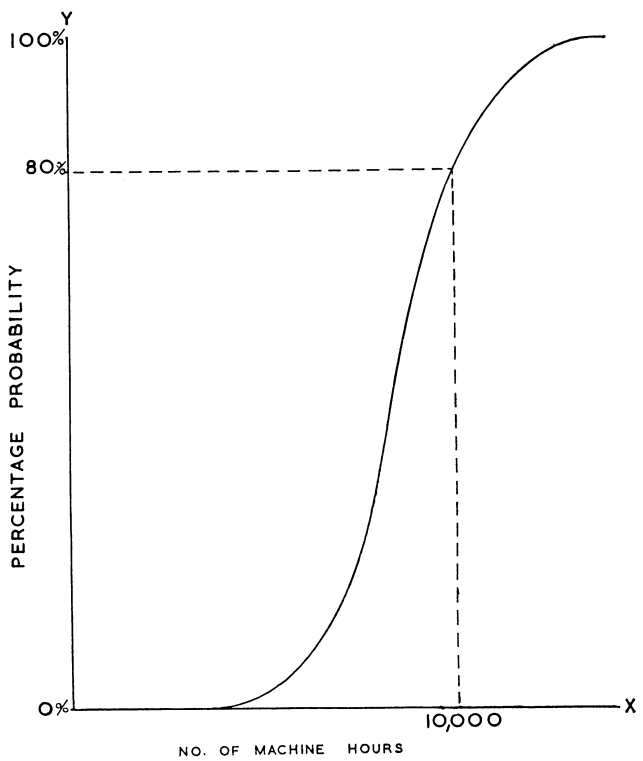

FIG. 1.—Machine capacity distribution curve.

This means, for example, that the probability of attaining 10,000 machine hours or less is 80%, but it can be used in another way. It will be noted that, as the curve is constantly increasing, there is a one-to-one correspondence through it between points on the x- and y-axes. If on the y-axis, the range of probability is divided into, say, 100 equal sections, and these sections correspond to the numbers 0 to 99, then if these numbers are drawn *at random*, the probability of drawing one particular section is exactly the same as the probability of drawing any of the others. Suppose section 80 is drawn—this will correspond, through the curve, with 10,000 machine hours. Thus, the constant drawing of random numbers can generate a set of machine hours, but in *this* set, the frequency of occurrence differs from member to member, for there is a greater probability of drawing from the centre of the machine hours range than from the ends. In fact, the set of machine hours obeys the frequency laws of the original data, and can be considered an extension of these data. This model sampling is a simple case of the use of the Monte Carlo method.

Suppose that 500 "total volumes" (described as T_1 to T_{500}) have been generated in the manner indicated. There are a number of different types of order, and, of course, some order types will occur more frequently than others, so the order types form a distribution. Within each order type there will be a distribution of packages. That is, the number of packages in an order of a particular type may vary each time that order type occurs. As we know for how many machine hours *on average* one package of each order type will run, the distribution of packages within order types can be specified in machine hours.

The types of order occurring in successive weeks are generally not unrelated, for there is a tendency for order types to occur in runs. These trends can be measured by auto-regression techniques, and due allowance can be made for them in the model. In the case under discussion this was not done, for the relationships were slight, probably because of the confounding introduced by the presence of many customers handling a variety of textile selling lines. Generally, to ignore such tendencies will introduce more frequent machine change-over than would occur in practice. If, however, the problem is to compare the ratio of change-over frequencies between one production policy and another, such a simplification will have but minor relevance. We can proceed as follows.

Using model sampling as before, and taking random numbers, the first number is related to an order type. The next random number is related, through the distribution curve for this order type, to a number of machine hours. This is h_i, say. Is $T_1 - h_i \leqslant 0$? If not, we return to the distribution of order types, take another random number, and repeat until $T_1 - \sum h_i \leqslant 0$. We have now generated a set of orders for the first week. Thus, sets of incoming orders can be built up week by week, and these are of the same pattern as actual orders.

When using a computer for this work, two modifications are necessary. Firstly, instead of dividing the y-axis into 100 (or multiples of 100) equal sections, it is more convenient, for a binary machine, to divide it into sections to a power of 2, say 128, or 1,024, for

Downloaded from https://academic.oup.com/comjnl/article/2/2/90/443688 by guest on 20 March 2024

obvious reasons. Secondly, to place a table of random numbers in the main store of the computer is a waste of storage space, and since most computers cannot generate random numbers, pseudo-random numbers, which they can generate, have to be used.

Assume that 128 divisions have been taken, and that the data on total machine hours are to be obtained. Corresponding to each of the 128 divisions there is a value for total machine hours, and these values are stored sequentially in the main store, the first at position 50, say. The first pseudo-random number, which is bigger than required, is found by a recurrence-relation method. The number is taken modulo 128, which means (as it is in binary) that all but the last seven bits are discarded. The total machine-hours value in main store address 50, modified by our pseudo-random number (modulo 128), is then chosen, and the process continues, thus building a set of weekly order capacities.

The point is worth making here that storage of a distribution is not essential. The curve of the distribution function is strictly increasing, and y is a known function of x, so equally x is a (different) one-valued function of y, the random number. Thus, as each random number is obtained, the variate x can be calculated from it. This requires that the original step function corresponds closely to a known and algebraically manageable distribution, or that the step function can be split into portions, and each portion corresponds closely to a portion of a known curve. The curve-fitting method is desirable when storage space is limited and where the distribution has some permanence. Model sampling for data should, however, be conducted whenever possible before input of the main program.

2.2 *The Main Program*

There are four aspects of the main program.

(*a*) Input of orders and adjustment of data carried over from the previous week in the main computer store.
(*b*) The means of deciding where each order is to be placed (the allocation logic—see 2.3).
(*c*) Simulation of machine loading and the resultant behaviour of machines (see 2.4).
(*d*) The output of results.

2.3 *The Allocation Logic* is a set of rules, not mutually exclusive, of which the following are examples.

(*a*) An order for processing this week is to be preferred to one for processing next week.
(*b*) An order requiring one machine setting cannot be placed on a machine with a different setting, without ending the run.
(*c*) Two orders of different types which look the same cannot be run together, even if they require the same machine setting.
(*d*) A machine must continue to run (fully loaded) as long as reasonably possible.

Given any particular machine, the attributes of a particular order which determine whether or not it shall go on that machine are five in number. They are

Priority (*P*). Orders are split into three categories of priority:

p orders: those which must be processed this week.
q orders: those which should be processed this week if possible, but which must be processed by the end of next week.
r orders: those which may be processed this week, if convenient, and must be processed by the end of the week after next.

Appearance (*A*). The bulk of orders are plain. These are called *W* orders (*W* for white). The rest (coloured orders) distinguishable from white, are called *C* orders.

Time (*T*). This is the average running time for one package of the particular order type.

Size (*S*). The size of an order is the number of initial packages on the order.

Machine Compatibility (*Y*). This attribute refers to the machine setting required.

As regards appearance, two plain orders of different types cannot be run consecutively, for they are indistinguishable from one another. During the change-over period, when machine positions vacated by packages of the first order are being filled by packages of the second order, orders of both types are sharing the machine, and there is a danger of confusion. During a machine run, plain orders must be separated in time by a coloured order. If they cannot be so separated, the machine must begin running-off (more and more positions become empty).

Each machine is stored as a word in the computer, and there are three sections of order locations (one section for each priority). At the end of each week, the remaining orders are moved up in priority, and the next set of weekly orders is fed in.

At the beginning of the week all machines are in one of four states:

State 1. Continuing to run into this week.
2. Running off, but not yet empty.
3. Empty, being used this week.
4. Empty, not being used this week.

A machine in state 4 has zeros throughout its word. A machine in state 3 shows $-1{\cdot}0$. States 1 and 2 show

Details of last order on	*RT*	*T* (*M*)

where *RT* is a signal indicating whether it is running off or not, and $T(M)$ is the time occupied on the machine during this week.

Downloaded from https://academic.oup.com/comjnl/article/2/2/90/443688 by guest on 20 March 2024

The words which represent orders are similar, showing

Details of order	S

where S is the number of packages in the order.

2.4 *The Simulation of Machine Loading* illustrates a further use of the Monte Carlo method. It will be supposed that the allocation logic has determined that a particular order is to be placed on a certain machine. For simplicity, it will be assumed that loading is not staggered, and that at the start of a machine run all packages will be placed at once.

The first position of the machine will be occupied at time t, say (and so will all the others). The *average* time taken to process a package of a particular order is known. Let us say this is H hours, but if N packages are to be processed, they will not all take exactly H hours. Their times will be distributed in some known manner.

Suppose the order to be placed has N packages, where N is greater than 100. As there are only 100 positions, the whole order will not be placed at once.

The distribution of processing times is known, so we can find, by Monte Carlo, a processing time for the package placed in each position. If for the first position this is t_{11}, the run will finish at $t + t_{11}$ hours. The run on the second machine position will finish at $t + t_{21}$ hours, and so on for all positions. There are now $N - 100$ packages left of the order, and we continue to place the packages until the order is exhausted. The first position may now be occupied for $t + t_{11} + t_{12} + t_{13}$ hours, say. If this takes it into next week we can forget it. If the allocation logic finds another order which can be put on, this will probably have a different distribution of *running* times. The process continues until the machine is running into next week, or there are no further orders which can appropriately be put on. The machine begins to empty (running off). When does it finish? There is a rule which requires machines to stop running when no more than 10 packages, say, are engaged, and we have a pattern as shown in Fig. 2.

The position of the lop-off (vertical) line in the time scale indicates when the machine is empty, and this is found from the eleventh longest time of running.

A point of comparison between this and the previous example of the use of the Monte Carlo method may be mentioned here. The first example generated data *before* the main program was placed in the computer. It is best to do this wherever possible. The present example is an integral part of the main program.

2.5 *The Program Structure*

At the beginning of a week, attention is given first to the machines loaded and running into this week. They are of two types for they are running plain (W) or coloured (C) orders. If the machine is W, the program inspects it and looks (first in priority p, then in q, then in

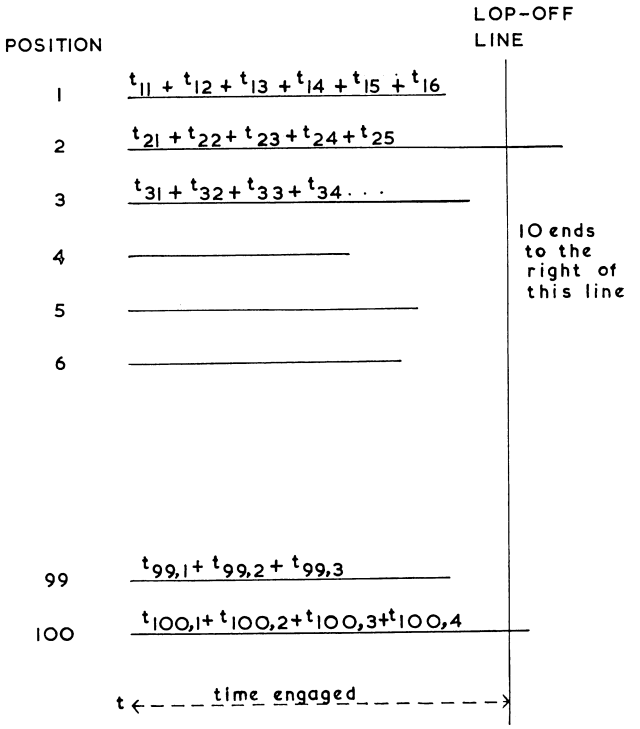

FIG. 2.—Machine position run lengths.

r) for another order, which is *identical*. If the identical order is a p order, it is placed, the program having first ensured that it will be completed in this week. Let there be no identical order. Another plain order cannot be sought, for it would not be possible to distinguish between the two, when they were running off. As previously mentioned, two plain orders must be separated by a coloured order. Because of this, and as there are many more plain than coloured orders, the program bestows coloured orders to machines grudgingly. If the machine has a coloured order on, the program prefers next, of course, a white to a coloured order.

Once the program begins to work on a particular machine, it never leaves it until the machine is empty or is running into next week. A state is reached where all machines, which have time left available this week, are empty at some time during the week.

The next stage of the program deals with machine states 2 and 3. This is an interesting sub-program, for it looks for chains of orders which will provide good runs on the machines. The chain of orders is now of prime importance, not the machine, and the program selects, tests and discards or keeps orders according to the rules of order priority and dependent upon the greatest amount of time available on a machine. The program is eager to find a chain which will cause a machine to run over into next week, for by doing so, it keeps runs of orders alive. It is playing a game with a steadily diminishing number of orders, until finally there are no orders left, or all machines are running into next week.

Downloaded from https://academic.oup.com/comjnl/article/2/2/90/443688 by guest on 20 March 2024

The final stage prints results, promotes in priority the unused orders, and takes in the next week's orders. The machine words are retimed to the beginning of the next week as origin, and machines with negative times are reduced to state 4. Machine time requirements are calculated and this generally entails transfer of state 4 machines to state 3. There is considerable comparison and adjustment of orders, after which the program returns to the first allocation stage.

Finally, the results obtained from a run on the "standard" program are compared with results derived from the "policy adjusted" program, and analysed by the use of routine statistical methods.

3 COMPUTER FACILITIES FOR SIMULATION WORK

What attributes of a computer are of special significance in programming this kind of simulation? An obvious one is a generous storage capacity, in both main and working stores. Another is a word length which allows for packing of a detailed classification. Further useful attributes are as follows:

(*a*) Extensive modify and count facilities.
(*b*) Fast logical shifts.
(*c*) Versatile arithmetical functions.
(*d*) Single word transfers between stores.
(*e*) Variety of jump instructions.

The last attribute is of value, if only to avoid inelegant programming routines, but particularly to avoid long exit routines where transfers from the main store deal only with small sections of program. There is generally no problem of entering a subroutine, but the provision of suitable exits, if the subroutine is to be used for more than one purpose, can be difficult. In this particular case, the necessary preparation, before entering a subroutine, involved the placing in the working store of all pieces of program to which the sub-program could possibly transfer, and the sub-program itself carried a tail of exit routines. In these circumstances, general-purpose subroutines have sizes bounded above and below. The lower limit is determined by the size of the exit tail, the upper by the degree of generality required.

Finally, a program of this nature is almost bound to be lengthy. Quite apart from detailed testing of sub-sections, there is a requirement for extensive testing of the full program when assembled. It is at this stage that the need for a facility to explore the route taken by a test through the program emerges. This can be done, for example, by embedding in the program instructions to print the contents of a working-store program location at frequent intervals, but it is convenient if an appropriate device is provided on the computer. Such a device was the optional "punch on block transfer" facility, which served as a guide through the network of loops for all tests.

4 ACKNOWLEDGEMENTS

My thanks are due to Miss G. M. Overton who co-operated in the work, to Mrs. W. F. M. Payne of Ferranti Limited and Mr. W. N. Jessop for their helpful advice and assistance, and to the Directors of Courtaulds Limited for permission to publish.

REFERENCES

CRANE, R. R., BROWN, F. B., and BLANCHARD, R. O. (1955). "An Analysis of a Railroad Classification Yard," *J. Operat. Res. Soc. Amer.*, Vol. 3, p. 262.

JESSOP, W. N. (1956). "Monte Carlo Methods and Industrial Problems," *Applied Statistics*, Vol. 5, p. 158.

JONES, H. G., and LEE, A. M. (1955). "Monte Carlo Methods in Heavy Industry," *Operat. Res. Quart.*, Vol. 6, p. 108.

KENDALL, M. G., and BABINGTON SMITH, B. (1938). "Randomness and Random Sampling Numbers," *J. R. Statist. Soc.*, Vol. 101, p. 147.

LEHMER, D. H. (1949). "Mathematical Methods in Large-Scale Computing Units" (Proceedings of a Second Symposium on Large-Scale Digital Calculating Machinery), *Annals of the Computation Laboratory*, Harvard University, Vol. 26, p. 141 (published 1951).

Downloaded from https://academic.oup.com/comjnl/article/2/2/90/443688 by guest on 20 March 2024