Computers and Their Applications[1]

B. C. MOORE, Engineering Experiment Station, Texas A&M College, College Station, Texas

IN THIS PAPER we propose to give a brief history of the meteoric growth of the digital computer in the past 15 years and to impress readers with the tremendous utility of this new tool of the mathematician, scientist, and engineer; and we shall also indicate some of the ways in which the machine has not been successful, some of its weak points and some of its failings. A tremendous revolution in mathematical thought and engineering and scientific practices and procedures is occurring at what seems to be an ever-accelerating pace. We shall show some of the more prosaic results of this growing revolution and shall try, though rather hesitatingly, to indicate how some of these results are concerned with the work of oil chemists. We hope to show that not only the partial differential equations of involved reactions (which a very few years ago were considered not solvable and hence not very important) are now well on their way to being solved, but also that other problems, perhaps of a much more practical nature, are now being given a mathematical expression and are being solved with the aid of the digital computer.

Digital computers are much more than over-sized "souped-up" desk calculators though they perform the the arithmetical functions of a desk calculator at an incredible speed. They are distinguished from desk calculators in that they can be programmed to solve complete problems. Their essential characteristic is their ability to make a decision and to proceed with the solution of the problem on the basis of that decision. These decisions are, of course, very elementary, being of the nature of deciding whether the result of a certain operation is positive or zero or negative and proceeding to differing sequences of operations, based on that decision.

McCormick (1) writes that computers are not new or mysterious and are not used only by supermen. He states that they do not think, but he then adds the disquieting statement that they can, in a sense, learn. It is hard to know where learning stops and thinking begins. We do seem able to program a machine so that it will outguess us, at least most of us.

Bellman (2) lists the five principal useful attributes of a computer as: it performs computations with extreme rapidity, it has large memory units, it is extremely reliable, it is capable of performing binary choices, and it is capable of automatic sequencing of operations. To this might be added a sixth useful attribute, it doesn't get confused!

The pioneer of the computer field was a British mathematician, apparently a very colorful and outspoken individual, by the name of Charles Babbage. He lived in the 19th century and attempted to build a computing engine running on mechanical principles. His theories were sound and remarkably like those of present-day computer designers, but he had no recourse to the fantastic speeds of present-day electronics and his attempts failed. Until about the start of World War II not much more was done in the field of computer design.

Saul Gorn of the University of Pennsylvania tells a fascinating story of neighboring defense activities at the University of Pennsylvania during World War II. A group of scientists and mathematicians were busy with the numerical calculations of ballistics problems, and a large fleet of desk calculators was kept busy night and day grinding out the interminable numerical calculations of the problem. In the next room another group of defense scientists were working with a new development called "radar" and were studying and measuring reactions of microseconds in duration. Perhaps it was inevitable that the need for speed so acutely felt in one section and the capacity for speed being generated in the next section should meet and produce a high-speed computer. This giant step forward was achieved in the latter years of the war.

Because of military secrecy, nothing was published on digital computers until about 1946. Apparently the first published report of a modern type of computer described the "ENIAC," which was developed by the Moore School of Engineering of the University of Pennsylvania.

THUS THE great race for speed of calculation was started. It still continues. In 1955 the following picture of the accelerating speed of calculating time was presented in the book "Faster, Faster" (3).

To form a thousand products by each time multiplying a ten-digit number by a ten-digit number to get a twenty-digit product would take the following time:

by hand with pencil and paper	1 week
by key control mechanical desk calculator	1 day
by electro-mechanical calculator	1 hour
by small electronic calculator	1 minute
by electronic "Supercalculator"	1 second

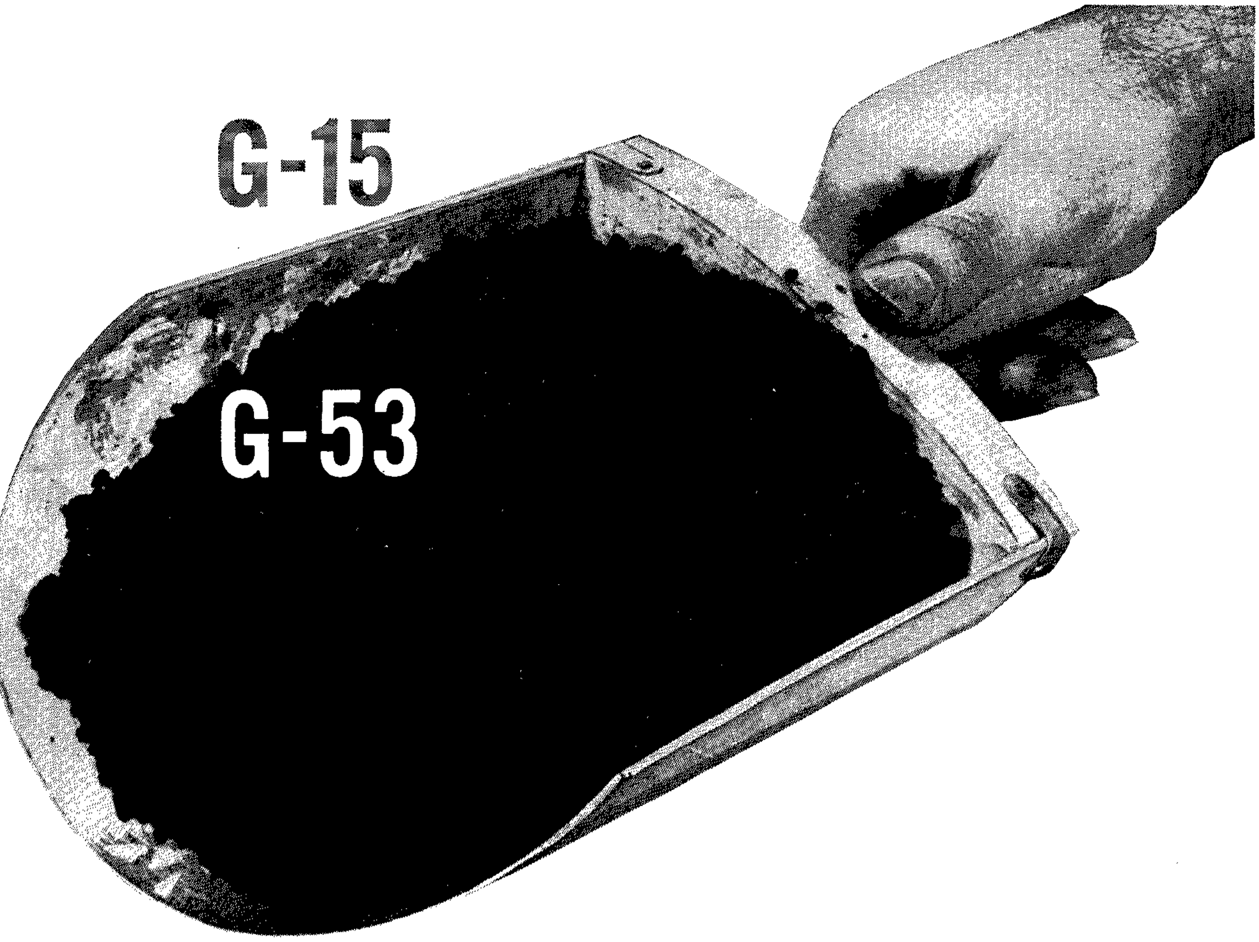
But the Supercalculator of 1955 (specifically, the IBM "NORC") has become slow in comparison with the newer machines. At Texas A. and M. we have an IBM 704 which will perform the thousand multiplications in about one-fifth of a second. This was considered pretty fast several years ago. Now it is a bit on the slow side. The IBM 709 will do the job in one-twenty-fifth of a second! Just operational on the scene is the IBM 7090, which will perform the thousand operations in one-one-hundredth of a second. The Philco Transac is about this speed and is operational. Under construction and due to arrive in a year or sooner are IBM's "STRETCH" and Remington Rand's "LARC." These machines are reported to be 60 to 100 times faster than the 704.

One final example of the speed of these machines. Suppose we have 96 linear equations in 96 unknowns. To solve these equations by an efficient method would take approximately 603,630 arithmetic operations. This would mean two years time using a desk calculator; the NORC would find the answers in about one minute. The 704 would take about 15 seconds, and "STRETCH" will have the answers in less than one-fourth of a second!

These fantastic calculation speeds cannot compensate for sloppy mathematical techniques. Bellman and Brook (2) point this out convincingly with the following example: 20 men are to be placed in 20 jobs; each man has a certain effectiveness in each of the 20 jobs; and the problem is to find that particular assignment of the 20 men to the 20 jobs that would give the maximum total effectiveness. Now, given a computer which can evaluate any particular arrangement of 20 men in 20 jobs at the rate of 1,000 per second, it would require more than 500,000 years to examine all possible arrangements of the 20 men in the 20 jobs.

There is a much quicker scheme for solving this problem, based on a group of mathematical concepts which come under the broad name of linear programming. Linear programming is a post-World War II development, utilizing, of course, a lot of theoretical and highly "useless" mathematics developed in the early years of the 20th century. It will be of interest to oil chemists, and in the years to come much more will be heard about it. Farmers will be much interested in linear programming because it tells how to distribute farm activities (on the basis of the best available data) to acquire a maximum profit. Manufacturers will learn the most profitable distribution

[1] Presented at spring meeting, American Oil Chemists' Society, Dallas, Tex., April 4-6, 1960.



NICKEL HYDROGENATION CATALYSTS

- High activity
- High selectivity
- Easy handling
- Outstanding filterability
- Excellent reuse properties
- Superior product color

G-15 and **G-53** are nickel-on-kieselguhr catalysts, dry-reduced and suspended in hardened vegetable oil or tallow. Their superior filterability has long been recognized.

These catalysts contain approximately 25 weight per cent nickel, 10 weight per cent support material and 65 weight per cent suspending medium. They are available as granules, 100 per cent passing through 6 mesh.

G-15 and **G-53** are prepared by different methods resulting in somewhat different selectivity characteristics.

G-53 is an extremely *rugged* catalyst characterized by excellent re-use properties. It has been demonstrated to be especially advantageous in difficult hydrogenations such as marine oils, fatty acid manufacture, etc.

Samples and more detailed information will be sent on request.

BRANCH OFFICES: 79 Wall Street, New York 5, New York; Phone: HAnover 2-3549 • 840 N. Michigan Avenue, Chicago 11, Illinois; Phone: WHitehall 4-3100 • 4007 Bellaire Blvd., Houston 26, Texas; Phone: MOhawk 4-3987

IN EUROPE, AFRICA, MIDDLE EAST: Girdler-Südchemie Katalysator G.m.b.H., Lenbachplatz 6, Munich 3, Germany

GIRDLER CATALYSTS

CHEMICAL PRODUCTS DIVISION • CHEMETRON CORPORATION

Louisville 1, Kentucky • Phone: SPring 8-4421

CHEMETRON

Catalysts for: Hydrogenation • Synthesis Gases • Hydrogen Generation • Desulfurization • New Catalytic Processes • Custom Applications of activities. Linear programming determines the most economical schedule for shipping materials from various warehouses to outlet stores. It determines the most economical balance between storage charges and the cost of fluctuating production in the manufacturing of such seasonal items as anti-freeze. (The Air Force applied this problem to the training and placing of air crews.) It will determine for the refinery engineer the most economical way to blend his gasoline so as to satisfy certain minimal and maximal specifications. We shall return to this problem in a minute.

Normally these problems are long, very long, and their solution is not practical without the aid of the high-speed digital computers. It was not until 1952 that the first linear programming problem was worked on a digital computer, the Bureau of Standards' SEAC. This problem, by the way, was an Air Force problem dealing with the deployment and support of aircraft to meet stipulated requirements. Now, eight years later, linear programming has become a major activity of many computing centers. In the next few years its growth will be phenomenal.

Since the mixing, or diet, problem should be of interest to many, it can be elaborated on it a little at this point. Given are the nutrient content of a number of different foods, the foods valued at varying prices, and the minimal amount of the nutrients that must be present in, say, 100 lbs. of the mix. The problem is to determine the amount of each food in the 100-lb. mixture so that the minimal requirements are satisfied and also so that it is a minimal-cost mixture. This is a simple straightforward problem that can be solved quickly and painlessly by a digital computer, provided the number of foods considered is not too large. Establishing maximal restrictions (of bulk, protein, or food components, for example) does not complicate the problem. As the number of variables in the problem increases however, the number of necessary storage units in the computer increases quite rapidly, and this brings us to the second criterion of modern computers, the amount of storage units, or memory available.

Here too we have seen tremendous advances in the past several years. The IBM 650 has 2,000 units of storage, the 704 up to 32,000, the 709 about the same, and STRETCH will have considerably more, at least up to 98,000.

However, just as we have seen that no matter how fast the computer, there are always problems that take too long to be practical, we know that no matter how big the storage we install in our computer one day, the next day a problem will come along that requires more storage than we have! And here is where the mathematician must use all of his knowledge, experience, and research ability to make the problem practical for the modern computer.

As a precise definition of Linear Programming I give the mathematical statement of a problem:

Given a system of M linear equations in n variables x_i:

$$(1)\qquad \sum_{i=1}^{n} a_{ij}\, x_i = b_j\,, \qquad j = 1, 2, \cdots M$$

find a solution of (1) such that $x_i \geqq 0$, and such that the value of a linear function L

$$(2)\qquad L = \sum_{i=1}^{n} c_i\, x_i \text{ is a maximum (or a minimum).}$$

In indicating the great scope of linear programming, it would be rather unfair not to point out also that many maximization or minimization problems are not linear programming problems because the restrictions cannot be put in the form (1), or the function L cannot be put in the form (2). Much research is going on in this area, and slow progress in solving these more general problems is being made.

We come to a third criterion of computers, which is concerned with minimizing a cancerous-like growth in computer calculations, called round-off error. In general, computers cannot carry or store fractions except as finite decimal fractions. (This is overlooking the fact that most computers use a binary scale rather than the decimal scale; but let's not get into that.) Because of this we find the digital computer, the "super brain" of the mathematician, telling us that two times one-third is not equal to two-thirds! $(2 \times 1/3) = (2) \times (.33) = .66$; while $2/3 = 2 \div 3 = .67$ since the computer is smart enough to round off to the next integer if the remainder is greater than $1/2$).

This sort of error is not very large if the number of digits that the computer carries in a number is large enough and if the computer doesn't propagate the error by making too many calculations. But what about the 603,630 operations mentioned before? The possibility of a large accumulated round off error must surely give us pause. The best way we know to minimize round-off error is to have the numbers the machine uses contain as many digits as possible. The number of digits a computer can store as a number is called the word-length, and the longer the word-length, the more calculations we can make without the round-off error seriously affecting our answers.

This need for longer word-length has apparently been seriously felt only in the last two or three years. The 650 has 10 decimal digits, and the 704 has 36 binary digits, which gives about the same word-length. STRETCH however will have 64 or more binary digits, and the LARC and TRANSAC word-lengths are about in this range.

A new sort of mathematics called Monte Carlo techniques, which are a sort of backward solving of probability problems, should be mentioned. If we wish to determine the probability of drawing four aces in a hand of five cards, we can either figure out the probability mathematically or we can deal ourselves 20 or 30 thousand hands of five cards and count the number of times we get four aces. To save time in this latter method, we shall simply program the computer to deal and count the hands at the rate of a thousand or so a second.

This, by the way, is probably the way a good card player becomes good at playing cards. He probably dosen't use the theories of mathematical probabilities, but by constant practice he develops a "feel" for the probable distribution of cards. In a sense then, the computer is learning to become a good card player. There is a large school of computer researchers that are using roughly the same techniques to program the computer to translate one language into another. (There are also other more methodical approaches to this problem.) This line of reasoning leaves us a little uneasy in thinking of the future of computers even if we are not professional card players.

To return to our four aces, the obvious question is how does the computer shuffle the cards? This is done by having the machine generate a sequence of pseudo-random numbers; pseudo here means that the sequence of numbers is not really random but that it is hard (we hope) to distinguish it from a truly random sequence.

Probability is not restricted to cards. We can show that solutions of partial differential equations are at the same time solutions to probability problems and hence can be solved, computerwise, by Monte Carlo techniques.

These methods also apply to the evaluation of definite integrals and many other interesting and (some day, inevitably) practical problems. Computers will not take the place of mathematicians but they will be constantly prodding mathematicians to greater and greater mathematical heights.

REFERENCES

1. McCormick, E. M., "Digital Computer Primer," New York, McGraw-Hill Company (1959).
2. Bellman, Richard, and Brock, Paul, American Mathematics Monthly, 67, 2, 119–134 (1960).
3. Eckert, W. J., and Jones, R., "Faster, Faster," McGraw-Hill Company, New York (1955).

[Received April 5, 1960]