R63-39 The Solomon Computer—D. L. Slotnick, W. C. Borck, and R. C. McReynolds. (*Proc. AFIPS Fall Joint Computer Conf.*, Philadelphia, Pa., December 4–6, 1962, pp. 97–107.)

The Solomon (*S*imultaneous *O*peration *L*inked *O*rdinal *M*odular *N*etwork) is a problem-oriented computer system. The system is developed especially to solve problems involving sets of variables that permit simultaneous and identical operation on each individual variable within the set. To this purpose the system consists of many (*e.g.*, 32×32) identical "processing elements" under control of a central processor. Connections between the processing elements are arranged according to the problem that has to be solved. As an example the solution of the 2-dimensional Laplace equation is taken. Since the iteration formula is identical for each mesh point, the calculations can be done simultaneously for each mesh point by the corresponding processing elements under control of one and the same program.

Each processing element consists of a serial adder which has access to two 64-word core stores (32 bits each) as well as to the core stores of its four closest neighbors. All operations of the processing elements, such as addition, subtraction, multiplication and division are performed serially under supervision of the central processor on numbers located in equal addresses of the respective core stores. A mode control switches off those processing elements where the operation is not wanted. Undoubtedly, here a very interesting idea is announced, which shows a way to speed up the solution of a very important class of problems. However, it is hard to believe that the gain in speed in comparison with the available large scale digital systems should be of the order of a factor of 60 to 200 as the authors claim.

Supposing a cycle time of 2 μsec, addition of 32-bit words would seem to take 64 μsec and multiplication about 2000 μsec. If a typical program shows 6 additions upon every multiplication, such a set of 7 operations would take $6\times64+2000=2400$ μsec. Because of the 32×32 processing elements working simultaneously, in this period about 7000 operations would have been performed. This gives an average of $3\cdot10^6$ operations per second. Nowadays large scale computers will perform up to about $0.5\cdot10^6$ operations per second. Thus it would seem reasonable that the system described is 5 to 10 times as fast as the fastest now available computers for the class of problems considered.

Furthermore it seems impossible or at least very difficult to perform floating operations in the Solomon system because of the fact that all processing elements must do the same micro-instructions.

As far as the amount of components is concerned, the individual stores of the 1000 processing elements together would form a big store of $12{,}8000\times32$ bits. All 1000 serial adders together would also form 32 parallel arithmetic units. It is still a question whether such a parallel arrangement would not make the system more general.

The clearly written article, anyhow, signals a new and very interesting line of development which seems promising for certain time-consuming problems.

H. J. Heijn
Philips Research Laboratories
N. V. Philips' Gloeilampenfabrieken
Eindhoven, Netherlands

I. Bionics

R63-40 An Active Pulse Transmission Line Simulating Nerve Axon—J. Nagumo, S. Arimoto, and S. Yoshizawa. (Proc. IRE, vol. 50, pp. 2061–2070; October, 1962.)

The authors develop a simple electronic model of the nerve membrane, consisting of three branches in parallel: a capacitor, a tunnel diode and EMF in series, and an inductor and resistor in series. This model is an approximate realization of the reviewer's nonlinear 2-variable mathematical (BVP) model,[1] based on a cubic *N*-shaped current-voltage characteristic, which is here approximated by that of the tunnel diode. Unlike some other neuron models, the transmembrane current and potential are simulated directly as current and potential. A number of these models can therefore be connected together to form the rungs of a ladder network, joined by longitudinally placed resistors; the result is a discrete approximation to a continuous nerve fiber (axon). It exhibits the expected physiological properties of an axon: it can be stimulated by a current at one end; if the stimulus exceeds a certain threshold value, an electrical disturbance arises and propagates away, approaching a constant wave form and velocity. This is the stable traveling wave which is the unit of activity of the nervous system, the nerve impulse. If the stimulus is below threshold, the impulse dies away. Two colliding impulses mutually annihilate each other.

The same phenomenon is studied theoretically. The authors combine the BVP equations with the cable equation and obtain a partial differential equation. Assuming the solution to be a fixed wave form travelling with constant velocity the authors derive an ordinary differential equation. (This procedure is similar to that used by Hodgkin and Huxley.[2]) The conduction velocity enters into this equation as a parameter to be adjusted so as to make the solution approach zero at the two "ends" of the infinite cable. At incorrect velocity values, the solution approaches infinity. When this is done, not one but two values of velocity are found. A similar result was first found by Huxley,[3] who concluded that the larger value corresponded to the normal stable impulse, and the lower one to an unstable impulse which is never recorded experimentally. The unstable impulse seems to form a borderline state (analogous to a saddle point in a phase plane, for instance) between those functions of distance that are large enough (in some sense) to approach a propagated action potential, while those that are too small die away. This point is clearly made by the authors' ladder network experiments, and is a basic property of both real axons and their mathematical and electronic models. A full mathematical proof of the stability of these theoretical impulses has not been carried out, but is an interesting problem.

This paper provides 1) a good brief summary of the mathematical aspects of axon conduction, and 2) a simple and probably useful electronic analog of the axon. It should be noted that the action potentials of this electronic model do not necessarily imitate accurately the shapes of real action potentials, and it is not likely that it will show the phenomenon of adaptation (a relatively slow accommodative change in impulse frequency during a constant current stimulus). Further refinements in these directions might be desirable. However, the model does have many of the important qualitative properties of the axon, and may provide a more practical way to simulate complicated nerve or heart muscle structures than solving differential equations on a computer, trading, of course, accuracy for simplicity.

Richard FitzHugh
National Institutes of Health
Bethesda, Md.

[1] R. FitzHugh, "Impulses and physiological states in theoretical models of nerve membrane," *Biophysical J.*, vol. 1, pp. 445–466; July, 1961.

[2] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *J. Physiol.*, vol. 117, pp. 500–544; August, 1952.

[3] A. F. Huxley, "Can a nerve propagate a subthreshold disturbance?" *J. Physiol.*, vol. 148, pp. 80–81P; October, 1959.

J. Control Theory

R63-41 A Nonlinear Digital Optimizing Program for Process Control Systems—Raymond A. Mugele. (*Proc. AFIPS Spring Joint Computer Conf.*, San Francisco, Calif., May 1–2, 1962, pp. 15–32.)

This article represents a fairly valuable contribution to the literature of nonlinear programming for several reasons. First of all, it is a succinct review of the status and techniques of nonlinear programming. Secondly, it draws conclusions as to what are apparently the better ways to approach on-line nonlinear programming problems as are likely to occur in process control systems. This article would be valuable not only for novices in the field but also for those who are actively engaged in applications of nonlinear programming in practice. After stating necessary definitions the article proceeds with a review of the various nonlinear programming techniques available today. It then draws conclusions as to which are the two basic methods that appear most promising at the present time for systems optimization problems with nonlinear constraints as well as nonlinear objective functions. Following this programming flow charts are developed for these more promising methods and three examples of different degrees of complexity are considered.

Cornelius T. Leondes
Department of Engineering
University of California
Los Angeles, Calif.