Chapter 2

STOCHASTIC COMPUTING SYSTEMS

B. R. Gaines

Department of Electrical Engineering Science
University of Essex, Colchester, Essex, U.K.

1. INTRODUCTION

The invention of the steam engine in the late eighteenth century made it possible to replace the muscle-power of men and animals by the motive power of machines. The invention of the stored-program digital computer during the second world war made it possible to replace the lower-level mental processes of man, such as arithmetic computation and information storage, by electronic data-processing in machines. We are now coming to the stage where it is reasonable to contemplate replacing some of the higher mental processes of man, such as the ability to recognize patterns and to learn, with similar capabilities in machines. However, we lack the "steam engine" or "digital computer" which will provide the necessary technology for learning and pattern recognition by machines.

In the summer of 1965 research teams working on these topics in different parts of the world discovered, quite independently, a new form of computer which might provide the necessary technology. The "stochastic computer" ([1–7]), as this has come to be called, is unlikely to prove the ultimate answer to the technological problems of machine learning and pattern recognition. It is, however, a step in the direction of processing by parallel structures similar to those of the human brain, and is of great interest in its own right as a novel addition to the family of basic computing techniques, and as a system which utilizes what is generally regarded as a waste product—random noise.

This chapter reviews the data-processing requirements of pattern recognition and machine learning, and introduces the concept of stochastic computing through the representation of analog quantities by the probabilities of discrete events. The first part of the chapter gives a complete overview of stochastic computing and its relationship to other computational

techniques. Later sections treat various forms of stochastic computer in some detail, and outline the theoretical basis for computing with probabilistic devices. The final sections describe various systems for pattern recognition and control where the use of stochastic computing elements is advantageous.

1.1. Summaries of Contents of Main Sections

Section 2 analyzes the computational problems of learning machines and pattern recognizers, and discusses to what extent the conventional general-purpose digital computer solves them. Matrix operations are taken as an example of generally important computations for which a general-purpose machine is very slow compared with alternative approaches. It is concluded that in machine learning and pattern recognition one is inexorably driven toward a data-processing system which makes maximum use of every single gate for computational purposes, and has parallel rather than sequential processing.

Section 3 examines the nature and origins of stochastic computing. The representation of quantities by probabilities is first introduced and constrasted with the representation of numbers in other forms of computer. Examples are given of two previous computing systems where numbers have been represented as probabilities, and finally the logical development of this concept in work on learning machines is outlined.

Section 4 is an exhaustive account of three stochastic computing systems which utilize linear mappings from quantities to probabilities. The first representation is for unipolar quantities; the second maps bipolar quantities onto two lines; and the third representation maps bipolar quantities onto a single line. These last two representations give rise to families of computing elements which correspond closely to those of the analog computer. Stochastic computing elements for addition, subtraction, multiplication, squaring, integration, and interfacing are described for each representation. Implicit function generation for division and square-root extraction is outlined and, finally, the generation of discrete random variables using random noise, or pseudorandom feedback shift registers, is analyzed.

Section 5 gives an overview of some of the theoretical foundations of stochastic computing which lie in stochastic automata theory, and discusses the relationship between the economy of stochastic computing and equivalences between stochastic and deterministic automata.

Section 6 extends the results of Section 4 to include alternative stochastic representations of quantity, particularly those in which infinite quantities have a definite representation.

Section 7 gives a detailed analysis of the main sequential element of the stochastic computer, the generalized integrator or ADDIE. The behavior of the ADDIE as a smoothing element is compared with that of an equivalent deterministic process, and the optimality of the linear ADDIE as an estimator is demonstrated.

Section 8 gives the first example of a system where stochastic behavior is essential to the operation. Digital adaptive threshold logic in which the weights take a finite set of values is shown not necessarily to converge to a solution, even when one is within the range of the weights. An alternative, nondeterministic, adaptive algorithm is shown to converge whenever a solution exists, and an example is given of a speech and character recognition system which has been constructed using stochastic feedback.

Section 9 reviews the modeling of linear systems using steepest-descent techniques, and analyzes the effect of replacing analog multipliers with switching elements, such as relays. An experimental comparison between six identification techniques, including polarity-coincidence correlation and three stochastic techniques, is outlined, and the different approaches are contrasted in their speed of response, interaction between estimates, bias due to noise in the system being identified, and variance of the final estimates.

Section 10 gives a further example of system identification where the use of stochastic computing elements enables a computation to be carried out in a way which would be otherwise impossible. The technique is one of maximum likelihood prediction based on Bayes inversion of conditional probabilities, and a particular form of ADDIE is described which enables normalized likelihood ratios to be estimated directly, and in a form suitable for prediction.

Section 11 considers some functional networks of stochastic computing elements for coordinate transformations, and the solution of partial differential equation, and compares them with networks of artificial neurons.

2. COMPUTATIONAL PROBLEMS OF LEARNING MACHINES AND PATTERN RECOGNIZERS

2.1. Character of Computations Required in Machine Learning and Pattern Recognition [7a, 7b]

There are so many different approaches to machine learning [8,9] and pattern recognition [10,11] that it might seem impossible to draw any general conclusions about the nature of their computational requirements. However, whatever the algorithms used in the machines, there are certain character-

istics of their environments, and the problems they are required to solve, which seem to be of general occurrence.

The data input to the machines is generally very great and derived from a large number of coexistent sources, and the number of different inputs which the machine may receive is usually several orders of magnitude greater than the number of outputs from the machine: it has to perform a very substantial reduction of a large data stream. For example, an alphanumeric character recognizer with an input retina of 10×10 photocells has at least 2^{100} possible inputs and about 64 possible outputs—a data reduction of order 10^{28}. The human eye has some 10^8 photoreceptors, roughly corresponding to a $10^4 \times 10^4$ array, and it is not beyond the bounds of present technology to fabricate an electrooptical array with the same capacity. Having generated this massive data source, however, how one reduces the 10^8 binary inputs to obtain a meaningful classification of the optical input is another matter; it clearly does not solve the problem to send 10^8 signal wires elsewhere.

Secondly, the adaption, or learning, of the machines, although one of their most complex characteristics, may be regarded, in a computational context, as requiring that the data-reduction procedure be variable. Since this variation will be a function of the "experience" of the machine, it implies some feedback from the results of past data-processing to that at present. The machine must have internal parameters controlling the processing of information which can be adjusted as a result of the effects of previous processing.

Thirdly, it is also a general characteristic of machine learning and pattern recognition that the data input is, in some sense, redundant, and the machine does not have to make very fine discriminations based on small differences in the incoming signal. This effect is difficult to quantify, but it seems a universal, and very important, characteristic of the situations in which learning machines and pattern recognizers are expected to operate. Insofar as the computation is concerned, this means that the data processing is global, over whole regions, rather than local, and that any particular part of a computation does not have to be performed to high accuracy; a "law of large numbers" operates to give sensible overall results from a large number of rough local computations.

Up to this point learning machines and pattern classifiers have been grouped together, but it is worth noting their similarities and differences. They are similar in that both are devices with inputs and outputs, and a variable relationship between input and output contingent upon the results of previous input/output decisions. They differ in that a pattern recognizer's

decisions generally do not affect the sequence of inputs which it is receiving, while the outputs (actions) of a learning machine feed back into its environment and act to change it. Because of this, the decisions of a pattern recognizer may generally be evaluated one by one and its input/output relationship changed accordingly, whereas the decisions of a learning machine may be evaluated only over a period of time. For direct comparison, therefore, a complete *sequence* of inputs to a learning machine should be thought of as equivalent to a single input to a pattern recognizer. Hence, over eight successive steps a learning machine with a 10-bit binary input pattern has to cope with as much data as a pattern recognizer with 10^8 cells in its retina. Thus the problems of machine learning are similar, but generally more difficult, than those of pattern recognition.

Thus the computational problems of machine learning and pattern recognition may be summarized as: *the processing of a large amount of data with fairly low accuracy in a variable, experience-dependent way.*

In the following sections the defects of a conventional, stored-program, general-purpose digital computer for this type of problem are considered, in order to illustrate the necessity for new forms of data-processing hardware in the physical realization of learning machines and pattern recognizers.

2.2. Advantages and Disadvantages of Sequential Computation

No one can fail to be aware of the achievements of the general-purpose, stored-program digital computer, which, in the quarter of a century since von Neumann made the first step from the physical patching of ENIAC to the program control of EDVAC, has had an unequalled impact on scientific and commercial data processing. The size and financial performance of the computer industry indicate the degree of practical utilization of the digital computer, and it is clear that there is room for much expansion yet—in particular through multiaccess, interactive systems where the man and machine may become true partners. However, in the very strength of the conventional digital computer, its sequential, stored-program control, lies its greatest weakness insofar as pattern recognition and learning are concerned; this manifests itself as a tradeoff between the size of a computation and the speed of its solution.

The essential structure of a conventional digital computer is a comparatively simple arithmetic/control-unit coupled to a large, uniform store structure, generally based on magnetic cores. The arithmetic unit operates on small units of data stored as bit-patterns in one part of the store, and

the sequence of operations which it performs are determined by a program in another part of the store. Although the operations of the arithmetic unit are, in themselves, very simple, long sequences of these operations may be used to build up more complex transformations on the bit-patterns in the core store.

For example, the user may write a MULTIPLY subroutine, which combines SHIFT operations, ADD operations, conditional JUMPS, and store TRANSFERS, into a routine which takes two binary numbers in named store locations and replaces them with their product. He may then use this routine in many places in his program, calling each time on the single copy which constitutes the subroutine. Such a procedure is obviously very flexible, in that the user-defined routines can perform virtually any operation, and it is also very simple and economical in use, since the one routine may be called by a single JUMP-TO-SUBROUTINE instruction.

However, the breakdown of basic operations into a sequence of steps and the use of one operator (subroutine) many times in a program have important disadvantages, in that the times taken by each simple step take longer and longer to perform. In the following section matrix multiplication is taken to illustrate this effect, and both the disadvantages of the general-purpose computer in performing this operation, and the centrality of matrix operations in machine learning and pattern recognition, are demonstrated.

2.3. Computation of Matrix Operations

Suppose we wish to multiply an $N \times N$ square matrix A_{ij} by a vector X_j. The computation has the form that the product Y_i is given by

$$Y_i = \sum_{j=1}^{N} A_{ij} X_j \tag{1}$$

Thus to compute one element of the vector Y_i using a general-purpose computer requires N additions and multiplications (together with store transfers and step counting), and to compute all N elements requires N^2 additions and multiplications.

Now consider how the requirement to perform matrix multiplications might have arisen: in a typical engineering problem the matrix might be the voltage/current transfer equations for a network of resistors, and the multiplication of a vector by the matrix corresponds to the computation of currents in various legs of the network given the voltages at its inputs. Think now of the behavior of the original network when the same voltages are applied to it and the currents measured: it "computes" the output

currents, not through a sequence of steps, but in one single step; the factor of N^2 times some fairly long time interval has dropped to 1 times a very much shorter time interval. This reduction is of very great importance as soon as the matrix becomes large in size; for $N = 100$ it is a factor of some 100,000 times.

The speed advantage of a resistive network in performing matrix "computations" is very substantial. Karplus and Howard ([12]) have taken advantage of this in a hybrid computing system consisting of a general-purpose machine coupled through a digital–analog interface to a network of resistors. In use the general-purpose machine sets voltages into about 1000 transfluxor analog stores, and these drive the network of resistors representing the coefficients in the matrix. Analog-to-digital conversion channels enable the digital machine to read off voltages and currents in the network. Karplus ([13]) has used this system to investigate the dynamics of water flow in the Californian underground basins. The basic equations to be solved are two-dimensional diffusion partial differential equations, but the shapes of the basins, and hence the boundary conditions, are unknown; simulation results must be checked against field experiments and the boundaries adjusted for the best fit. If this computation were performed entirely by the general-purpose machine, 95% of its time would be allocated to the matrix inversion; when this latter operation is performed by the resistive network the time taken is decreased by a factor of at least ten.

The operation of matrix multiplication is not at all atypical, and in fact it is a very common one which underlies many algorithms for machine learning and pattern recognition. For example, in the STeLLA ([8,14]) learning system the behavior of the machine's environment is modeled by a set of transition probabilities among the possible states in the environment for a given action. Given an expected probability distribution over the set of states, the distribution which will follow an action is obtained by multiplication of the given distribution by the matrix of transition probabilities for the action. The environment of the simplest STeLLA machine, with a ten-bit input pattern, has about 1000 possible states, so that $N = 1000$ and the speed advantage of matrix multiplication through a resistive network over a digital computer is at least some ten million to one.

The importance of matrix operations in general "image transformation" has been discussed by Poppelbaum *et al.* ([2]); they point out that matrix operations on the points of a retina provide general linear transformations including: (1) translations, rotations, and magnifications, (2) conformal mappings, (3) convolutions, and (4) Fourier transforms. The defects of the general-purpose machine in performing these operations led

Poppelbaum ([15]), in an earlier work, to suggest an analog system based on a resistive net, and, in the later paper noted above, to describe means for performing general linear transformations using stochastic computing elements.

2.4. A Comparison of Sequential and Parallel Processing

The analog computing system based on a network of resistors is faster in performing matrix operations than the general-purpose digital machine because it is a "parallel" computing system, in which all the operations involved in computing the matrix multiplication are performed simultaneously. The general-purpose machine, on the other hand, performs the operations sequentially, and multiplexes its single processor to each part of the computation in turn. Figure 1 illustrates the effect of this multiplexing on the speed of computation as the size of problem grows.

The horizontal axis is the size of computation in terms of the number of equations to be solved. The vertical axis is the time taken to solve the problem. The general-purpose digital machine takes longer over the computation as the size of the problem grows, and the problems suitable for this machine lie above the sloping line. The analog computer, assuming that a simple resistive net is unsuitable and operational amplifiers have to be used, is limited in size rather then speed, and the computations suitable for this machine lie to the left of the vertical line.

This diagram originated in a similar one shown by Williams ([16]) in a paper on process control at the second congress of IFAC. He demonstrated that this decrease of speed with size of problem on general-purpose machines made it impossible to simulate even a simple, linearized model of a multiplate distillation column in real time on presently available computers.

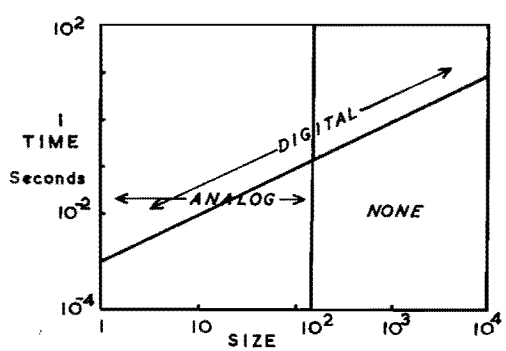

Fig. 1. Relationship between problem size and solution time for digital and analog computers.

Williams remarks that "It is truly unfortunate to note the number of chemical process systems which fall into the lower right-hand corner of Figure 1;" for the simulation of such systems neither present analog nor digital computers are suitable. If one considered an adaptive controller, or learning machine, attempting to control a multiplate simulation column, then simulation of the column itself, even if the parameters were completely known, would obviously be only a minor part of the computations required, and hence Williams's remark applies with even greater force to complex control systems, such as learning machines, than they do to simulators.

2.5. The Need for Low-Cost Parallel-Processing Hardware

The arguments of the previous sections have demonstrated that, despite its great versatility and variety of successful applications, the stored-program, general-purpose digital computer does not, by a very wide margin, provide the data-processing facilities required in machine learning and pattern recognition. Most experimental studies of these topics have used general-purpose machines, since these provide the most powerful and versatile tool available for data-processing at present, but at the same time no experimental study of these topics has so far produced a powerful or useful learning system, let alone one with facilities comparable to those of the human brain.

The parallel processing of the analog computer and associated resistive networks gives very great speed advantages over the general-purpose machine in certain types of computation, and it is clear that some form of parallel processing system will be required if learning machines and pattern recognizers are to be physically realized. The general-purpose machine achieves its relatively low cost through the use of a simple, homogeneous core-store structure for data and program storage. The core store is a passive system incapable of data processing in its own right, and cannot form the basis of a parallel computing system. However, it is reasonable to assume that any parallel machine must provide arithmetic units with associated data stores in the same proliferation as the magnetic cores in a general-purpose machine and at the same order of cost. Fortunately, device technology is coming to the stage where this requirement may be satisfied.

2.6. Sources of Low-Cost Computing Devices

There have been many forms of low-cost storage and computing devices investigated for application to learning machines and pattern recognizers.

The specification required of the elements follows from consideration of these machines as being essentially variable-parameter systems with input and output. This implies a minimum hardware complement of: stores to hold the parameters; multipliers to enable the parameters to weight other variable in the system; and parameter-adjustment logic to enable the weights to be changed as a result of experience.

The analog computer ([17]) provides stores, in the form of analog integrators, which are fairly easy to adjust incrementally. It also provides multipliers for analog variables which, although previously expensive, may nowadays be made cheaply using FET switches. At the same time, the logic for parameter-adjustment, function-selection, etc., has become available at low-cost through the advent of FET devices. Although the operational amplifiers are now available as low-cost integrated circuits and could be fabricated in large arrays, the associated storage capacitors are large and cannot be miniaturized or made at very low cost. Thus it is unlikely that analog computer devices can form the basis of practical learning machines and pattern recognizers. However, it is worth noting that analog systems have been proposed for adaptive-threshold-logic pattern recognizers ([18]), and that the use of hybrid analog–digital storage techniques may enable capacitors to be eliminated from the analog stores ([19]).

Other forms of analog store have also been proposed and fabricated, including those based on transfluxors ([20]); electroplating ([21]); and electrolysis ([22]). However, apart from individual defects in cost, size, and reliability, these have all been difficult to integrate into the overall system; the external circuitry necessary to adjust the stored value and use it to multiply other variables has exceeded the original device in complexity, size, and cost.

Optical data-processing systems offer the possibility of operating on very large data sources at low cost and high speed ([23]). At present, however, reversible storage elements for use in storing and emitting optical signals are not fully developed, although it is clear that photochromic systems ([24,25]) will some day be developed to perform these functions. For pattern recognition systems ([26,27]) it may well be that optical data-processing devices will eventually provide the best hardware. However, the position is far less clear in their application to general learning systems, and little effort has been devoted to this topic as yet.

2.7. Large-Scale Integration

There is one area of device technology where it is reasonable to expect that very large and complex systems may be fabricated at a low cost in the near future, and this is the large-scale integration (LSI) of silicon integrated

circuits ([28]). The implications of LSI go far beyond those of any previous technological advance. Whereas the transition from vacuum tubes to transistors brought about a tremendous increase in reliability and decrease in physical volume, any increase in system size was a byproduct of these rather than a main effect. What LSI offers is sheer quantity of devices at a low cost and in a small space—1000 logic gates on a single chip of silicon are currently being manufactured, and 10,000 gates or more are reliably forecast for 1973 ([29]).

Although LSI is expected to offer large numbers of devices at very low cost, a major objective of any machine learning, or pattern recognition, computer based on it must still be the economic use of hardware. For example, the processing unit of a general-purpose machine takes some 10^3 gates to provide its variety of functions; computational techniques for performing these functions with far fewer gates, even at greatly reduced speed, may be necessary to make emulation of human capabilities feasible.

The reason for this becomes clear when we consider the total number of devices available, and the amount of computation they will perform. Ten thousand devices at $1 apiece, each containing 10,000 gates, gives us an estimated learning machine selling in 1980 for $30,000 and containing 10^8 gates. If each of those gates could provide a computational function, such as ADD or MULTIPLY, then we have of order 10^8 computing elements; our learning machine has a "brain" two or three orders of magnitude down in size from that of a human being. If it takes one thousand gates to perform a useful computation, then we are five or six orders of magnitude down, and the machine looks pretty hopeless.

If the gain in computing power of one element entails a loss in speed, the power $\times$ speed may not change, or favor larger computing elements, as the number of gates per computation is changed. Given that it is global computing power rather than speed in computation which we require, however, the high-speed element has to be multiplexed to many data streams to achieve the same effect as many local low-speed elements, and the cost, in gates, of multiplexing may well exceed that of computing.

Thus one is inexorably driven toward a data-processing system which makes the maximum use of every single gate for computational purposes, and distributes its computations through space rather than time. There are a number of projects currently under way on the development of such computing systems which rely on the imminent coming of LSI to make them economically feasible. Many of these are multiprocessor developments of conventional computers, and attempt to give the general-purpose machine increased power by increasing processing capability rather than speed ([30,31]).

Other developments are aimed specifically at certain problems, with pattern recognition being one of the most prominent ([32,33]). The stochastic computer is one of these, and represents a deliberate attempt to utilize standard digital logic gates and flip-flops in a new mode particularly suited to the computations of machine learning and pattern recognition.

3. EMERGENCE OF STOCHASTIC COMPUTING

The arguments of the previous section suggest that in the development of hardware for learning machines and pattern recognizers which will begin to have comparable capabilities to the human brain one requires a non-multiplexed, parallel system, not necessarily of high speed, which makes the maximum use of every single element for computational purposes. These characteristics are not possessed by the general-purpose digital computer, and while they are possessed, to some extent, by the analog computer, it is in the stochastic computer that the required computing power is most completely distributed throughout arrays of low-cost elements.

The distributed computation of the stochastic computer is achieved through its peculiar representation of data by the *probability* that a logic level will be ON or OFF at a clock pulse (we will assume throughout this chapter that the logic elements of a stochastic computer operate in a synchronous, or "clocked" mode). In any other form of computer the logic levels representing data change deterministically from value to value as the computation procedes, and if the computation is repeated the same sequence of logic levels will occur. In the stochastic computer arithmetic operations are performed by virtue of the completely random and uncorrelated nature of the logic levels representing data, and only the probability that a logic level will be ON or OFF is determined; its actual value is a chance event which cannot be predicted, and repetition of a computation will give rise to a *different* sequence of logic levels.

Thus a quantity in the stochastic computer is represented by a clocked sequence of logic levels generated by a random process such that successive levels are statistically independent, and the probability of the logic level being ON is a measure of that quantity; such a sequence is called a *Bernoulli sequence.*

3.1. Comparison of Data Representations in Computers

The physical reality behind the mathematical concept of "probability" is, of course, that the generating probability of a sequence of logic levels

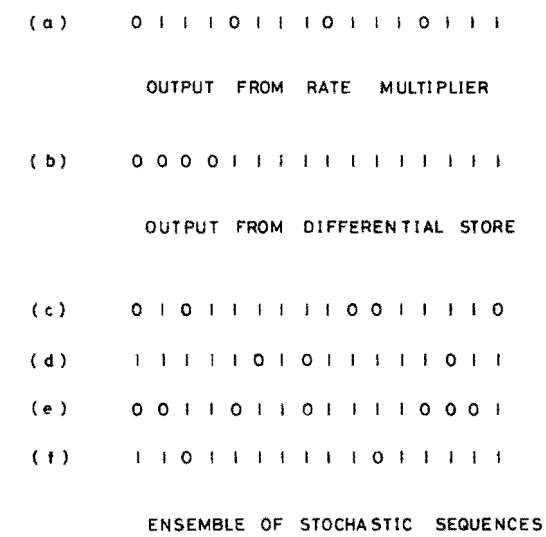

Fig. 2. Typical computing sequences in various incremental computers.

corresponds to the relative frequency of ON logic levels in a sufficiently long sequence. In its use of relative frequency to convey information the stochastic computer is similar to the other *incremental,* or *counting* computers, such as the digital differential analyzer (DDA) ([34,35]), operational digital computer ([36,37]), and phase computer ([38,39]). In all these computers, including the stochastic computer, quantities are represented as binary words for purposes of storage, and as the proportion of ON logic levels in a clocked sequence for purposes of computation. In previous machines, however, the sequences of logic levels are generated deterministically and are generally patterned or repetitious, whereas in the stochastic computer each logic level is generated by a statistically independent process.

This distinction is illustrated in Fig. 2, which shows typical sequences in the various forms of computer: (a) is the output from a synchronous rate multiplier, or from the "R" register of a DDA, corresponding to a store $\frac{3}{4}$ full—it will be noted that the ON and OFF logic levels are distributed as uniformly as possible; (b) is the output of a differential store in the phase computer—the OFF logic levels in a cycle all occur before the ON logic levels, giving the synchronous equivalent of a mark/space modulated signal; (c)–(f) are samples of stochastic sequences with a generating probability of $\frac{3}{4}$—any sequence [including (a) and (b)] may occur, but the proportion of ON levels in a large sample will have a binomial distribution with a mean of $\frac{3}{4}$.

One advantage of representing numerical data by a probability is that the latter is a continuous variable capable of representing analog data without the quantization necessary in other digital computers such as the general-purpose machine and DDA. However, a probability cannot be measured exactly but only estimated as the relative frequency of ON logic level in a sufficiently long sample, and hence, although quantization errors are absent, the stochastic computer introduces its own errors in the form of random variance. If we observe a sequence of N logic levels and k of them are ON, then the estimated generating probability is

$$\hat{p} = k/N \tag{2}$$

The sampling distribution of the variable k is binomial, and hence the standard deviation of the estimated probability $\hat{p}$ from the true probability p is

$$\sigma(\hat{p}) = [p(1 - p)/N]^{1/2} \tag{3}$$

Hence the accuracy in estimation of a generating probability increases as the square root of the length of the sequence examined, i.e., as the square root of the length, or time, of computation.

This result enables one to compare the various forms of computer in terms of their efficiency in representing data. It is reasonable to say that the analog computer is most efficient, since it utilizes a single, continuous signal to represent any value of an analog variable. The digital computer uses a coded representation of data as binary words, which is the most efficient possible if only binary levels are available. The DDA uses the relative frequency, or count, of ON logic levels, but this is determined completely by the data and is the most precise *count* possible. The stochastic computer uses a generating probability, so that the count only averages to the correct value over an extended period. Hence the number of levels required by various computers to carry analog data with a precision of one part in N is:

1. The analog computer requires one continuous level.
2. The digital computer requires $\log_2 kN$ ordered binary levels.
3. The DDA requires kN unordered binary levels.
4. The stochastic computer requires kN^2 unordered binary levels.

Here $k > 1$ is a constant representing the effects of round-off error or variance, $k = 10$, say. The N^2 term for the stochastic computer arises because the expected error in estimating a generating probability decreases as the square-root of the length of sequence sampled.

This progression $1:(\log_2 N):N:N^2$ shows the stochastic computer to be the least efficient in the representation of data, as might be expected, since random noise is being deliberately introduced into the data. This introduction of probabilistic processes, or "noise" sources, in data processing is unusual, to say the least, and before treating stochastic computation in general some specific examples will be given of the advantages to be gained through doing this. It will become apparent that the lack of coherency, or patterning, in stochastic sequences enables simple hardware to be used for complex calculations with data represented by probabilities.

The first two examples date from before the invention of stochastic computing, but may be regarded as particular forms of stochastic computer; the first is interesting because it forms the basis of a commercial instrument, and the second because it introduces several of the basic operations of the stochastic computer. The third example is a brief case history of the development of an adaptive element for learning machines which eventually give rise to the stochastic computer.

3.2. Round-off Error Elimination in Analog/Digital Convertors

A simple example of data-processing where the addition of a little noise can do a lot of good is in the avoidance of the cumulative effects of round-off error in analog-to-digital convertors. A successive-approximation digital voltmeter takes a sequence of decisions of the type: is the input voltage above half-scale range? If so, set the most significant digit and subtract half-scale voltage from the input; is the remainder above quarter-scale range? if so, etc. The least significant digit, the Nth say, is set in the same way by comparison of the $(N - 1)$th remainder with $2\text{–}N$ times the full-scale range, and the residual remainder is neglected. This residue or round-off error can have a maximum magnitude slightly less than $2^{-(N+1)}$ times the full-scale range (FSR).

Suppose now that a sequence of M readings of a fixed voltage are averaged to obtain the best estimate of its value. There is no reason to suppose that the round-off errors will cancel, and indeed if the voltage is fixed and the convertor is accurate, the errors will all be the same in magnitude and sign. Thus the mean error ε in the result is still bounded by

$$-\text{FSR}/2^{N+1} < \varepsilon < +\text{FSR}/2^{N+1} \tag{4}$$

and the averaging has made no reduction in the round-off error.

Consider now the effect of adding to the input voltage another voltage

V whose magnitude lies in the same range as the round-off error, and which is selected at random, uniformly in this range, each time a conversion is made. This added voltage is too small to affect any but the least significant digit, but the latter is now dependent on both the input voltage and the random voltage. The greater the round-off error in deterministic conversion, the less is the probability that the least significant digit will be set in random conversion. Thus there will be a tendency for errors in the least significant digit in random conversion to cancel out on averaging. That this tendency is exact may be shown by determining the expected state of the least significant digit (LSD) and hence its expected value. Let the remainder after determination of the $(N-1)$th digit be E, $0 \leq E < \mathrm{FSR}/2N$. The LSD is 1 if $E+V$ exceeds $\mathrm{FSR}/2N^{+1}$, and since V is evenly distributed over its range,

$$p(\mathrm{LSD}=1) = p(E+V > \mathrm{FSR}/2^{N+1}) = \frac{E}{\mathrm{FSR}/2^N} \tag{5}$$

Thus the expected value represented by the LSD is

$$p(\mathrm{LSD}=1)\mathrm{FSR}/2^N = E \tag{6}$$

and the average of a set of readings of the input voltage plus random noise has no bias or round-off error. It will of course have a variance, since it is based on a finite sample of a random process, and it may be shown to have an approximately normal distribution with variance:

$$\sigma^2 = \frac{|\varepsilon|[(\mathrm{FSR}/2^N) - |\varepsilon|]}{4M} \tag{7}$$

Thus the standard deviation of the random conversion is less than the round-off error of the deterministic conversion and goes to zero as the number of readings becomes large.

This technique has been used to good effect in the "Enhancetron," ([40]) a device for averaging evoked potentials to decrease the effects of noise, or for averaging any phase-locked voltage waveform. Averaging devices for this purpose must use digital stores, since analog integrators have too short a leakage time constant. The normal practice is to sample the waveform at regular intervals, convert it into a 12-bit digital form, and add this into a digital store corresponding to the particular sampling instant. The enhancentron is remarkable in that it converts to a single bit, using random conversion to prevent the tremendous round-off error which would otherwise accrue. The block diagram of Fig. 3 illustrates its operation: the incoming waveform is compared with a sawtooth (simulating a uniform random distribution) in a comparator; the output is commutated around a

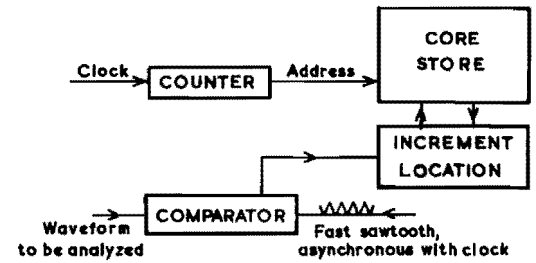

Fig. 3. Principle of the "Enhancetron."

ring of digital stores; at a clock pulse the appropriate store is incremented by one unit if the output of the comparator is ON and decremented otherwise; the ring of stores will eventually average out the noise at the input and the noise of random conversion, and contain a sampled representation of any repetitive signal whose phase is locked to their cycling rate. Thus the use of random conversion has replaced a 12-bit analog-to-digital convertor by a simple comparator, and replaced 12-bit binary addition by simple incrementing/decrementing.

3.3. Linearization of the Polarity-Coincidence Correlator

One of the most frequent computations on the analog computer is to cross-correlate two waveforms. Given two voltage waveforms with zero means, it is required to compute their covariance:

$$C_T = \frac{1}{T}\int_0^T U(t)V(t)\,dt \tag{8}$$

which is an awkward function because it involves integration over extended intervals and multiplication, both of which tax analog computing elements to their utmost. If the two waveforms have almost-Gaussian distributions, it may be shown that the correlation between their heavily-limited forms (in which only their signs are taken into account) is uniquely related to C_T ([41]). If

$$P_T = \frac{1}{T}\int_0^T \mathrm{sign}[U(t)]\,\mathrm{sign}[V(t)]\,dt \tag{9}$$

then as $T \to \infty$

$$C_T \to a\sin(\tfrac{1}{2}\pi P_T) \tag{10}$$

where a is a constant dependent on the variance of U and V.

This nonlinear relationship does not necessarily hold for non-Gaussian

distributions, and does not yield a simple additive effect if uncorrelated noise is added to the waveforms. Despite these limitations the polarity-coincidence correlator is very attractive because multiplication of two numbers whose modulus is unity can be carried out by a simple gate or relay, and if the waveforms are sampled at regular intervals the integration may be carried out digitally by a reversible counter; thus both analog multiplier and integrator may be replaced by economical and reliable digital devices.

The limitations can be completely removed if the polarity-coincidence correlator is converted to a simple stochastic computer by adding to the input voltages randomly-varying voltages with zero means and uniform distributions. Let $A(t)$ be a random waveform uniformly distributed in a range greater than that of $U(t)$ and having a δ-function autocorrelation, and let $B(t)$ be a similar uncorrelated waveform for $V(t)$. Let $P_{T,N}$ be the sampled polarity-coincidence correlation of $(U + A)$ against $(V + B)$:

$$P_{T,N} = \frac{1}{N} \sum_{i=0}^{N-1} \operatorname{sign}[U(iT/N) + A(iT/N)] \operatorname{sign}[V(iT/N) + B(iT/N)] \quad (11)$$

Then it may be shown ([41]) that as $T \to \infty$

$$C_T \to a \lim_{N \to \infty} P_{T,N} \quad (12)$$

where a is a contant.

Thus the polarity-coincidence correlator gives an unbiased estimate proportional to the covariance of the input signals no matter what their distribution, provided the signals are sampled rapidly enough over a sufficient period. The addition of random noise again introduces additional variance into the estimate, but this can be made negligible by taking a longer sample of the waveforms or sampling more often; thus the power of this correlator may be less than that of a normal cross-correlator, but its accuracy is the same.

A block diagram of one realization of a polarity coincidence correlator with added noise is shown in Fig. 4: the random waveforms are again approximated by very fast anharmonic sawtooths feeding inputs to comparators, on the other side of which are the waveforms to be correlated;

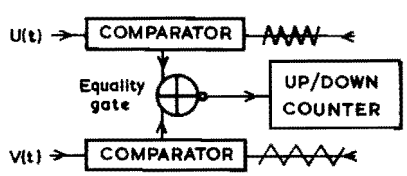

Fig. 4. Linearized polarity-coincidence correlator.

the logic levels out of the comparator are fed to an equality gate whose output is ON only when its inputs are equal; the output from this gate represents the product of the signs of the signals plus noise, and this is used to determine whether a binary counter shall increment or decrement at a clock pulse (sampling instant); the state of the counter eventually represents the covariance between the inputs. This is a digital circuit, more economical in its realization than an analog multiplier and integrator, and is the classic example of the advantages to be gained through the intentional introduction of noise into data processors.

3.4. An Adaptive Element for Learning Machines

At the University of Illinois ([1–3]) the stochastic computer developed out of a study of microplasmas in Zener diodes for noise generation, and was intended for use in general *image transformers* and pattern recognizers. In Britain a virtually identical computing system was developed as part of a program of research on the structure, realization, and application of advanced automatic controllers in the form of learning machines at the Standard Telecommunication Laboratories of ITT ([4–7]).

Although algorithms for search, identification, policy-formation, and the integration of these activities (in the STeLLA ([14]) learning machine), could be established and tested by computer simulation, there was no hardware available to make construction of the complex computing structure required in a learning machine feasible.

An element was required to hold the value of a weight representing some aspect of the accumulated experience of the machine. This weight had to be changed incrementally as the machine learned, and its value had to be multiplied by other variables in the machine. Analog and electrochemical elements proved unsatisfactory in this application and the decision was made to use standard digital flip-flops and gates. A binary up/down counter has the properties of an incremental store; however, while the weight was only required to a precision of a few bits, 1% say, the increments to the weight were required to be very much smaller. Rather than extend the counter to the precision required to accept small increments of variable size, it was decided to investigate the possibility of making "fractional increments" of less than the unit increment of the counter by incrementing a stochastic process. Thus if an increment of the value corresponding to one tenth of the least significant bit was required, the counter would be incremented by unity with a probability of one tenth; while individual increments would clearly not be "fractional," on average the required effect would be achieved.

This is the basic principle of stochastic computing, to represent analog quantities by the probabilities that discrete events will occur—an increment of one tenth of a unit is represented by a unit increment with a probability of one tenth that it will occur. The basic principle was immediately extended from its first application in the incrementing of stores, to the other computations which were required. For example, if the count in the counter was represented by a probability proportional to it that a logic line would be ON, then the value of the count could readily be made to multiply other variables as required if they also were represented as probabilities; the probability that the output of a two-input AND gate will be ON is the joint probability that its two inputs will be ON, and this is the product of the individual probabilities that each of its inputs is ON. Thus a simple AND gate acts as a multiplier for two variables represented as probabilities.

The adaptive element developed in this way was called an ADDIE, and is described in detail in later sections (Sections 4.9 and 7). Many variations of the ADDIE with different computational functions are now known, and it was the development of these which demonstrated that stochastic computing was, in some sense, "naturally" related to machine learning: computations which previously had been approximated in the digital machine because of the inordinate amount of computation required could be performed very simply by stochastic elements which had *no* analog computing equivalent (see the discussion of Bayesian prediction, Section 10); computing processes which were previously inefficient in their use of hardware now made maximum use of every bit of storage (see the discussion of adaptive threshold logic, Section 8). While a major part of this chapter, and of published work on the stochastic computer, is devoted to describing stochastic computing elements equivalent to analog computing elements, it is clear that the most important aspect of the stochastic computer is that it provides computational elements which have no direct equivalent in any other form of computer; these are the elements of main interest.

In the following section the present informal introduction to stochastic computing is systematized and a complete set of computing elements for *linear* representations of quantities as probabilities is described.

4. THE "LINEAR" STOCHASTIC COMPUTER

In some applications of the stochastic computer, such as the Bayesian and Markovian estimators described later, one is dealing with the estimation of probabilities of external events, and the (0, 1) range of generating probabilities in the computer itself has a natural interpretation. Generally, however, it is necessary to map some other variable into this range, in the same way that one maps quantities into the range of voltages of an analog computer. In this section we shall consider only linear transformations between a continuous variable and the (0, 1) range of generating probabilities; other transformations are considered in Section 6 and later sections. The linear transformations give rise to computing structures closely allied to those of the conventional analog computer, and a complete range of computing elements, similar in function to the invertors, adders, multipliers, and integrators of the analog computer, will be described.

4.1. Linear Mapping from Analog Variables to Probabilities

4.1.1. Representation I. Unipolar

Given a quantity E in the range $0 \leq E \leq V$, represent it by the binary variable A with generating probability p:

$$p = p(A = 1) = E/V \tag{13}$$

i.e., the magnitude of a positive bounded quantity is represented by a logic level which is always ON for maximum quantity, always OFF for zero quantity, and fluctuates randomly for intermediate quantities.

Suppose the value of A is noted at each of N clock intervals, and it takes the value A_i ($A_i = 0, 1$) at the ith clock pulse. Then the estimated value of the generating probability $\hat{p}$ is

$$\hat{p} = (1/N) \sum_{i=1}^{N} A_i \tag{14}$$

The expected value of $\hat{p}$, $\mathrm{Exp}(\hat{p})$, is

$$\mathrm{Exp}(\hat{p}) = p \tag{15}$$

Thus the expected value of the estimate is independent of the number of samples taken. The accuracy of the estimate is, however, determined by its variance, and this is a function of N.

The variance of $\hat{p}$, $\mathrm{Var}(\hat{p})$, is given by the following equation:

$$\mathrm{Var}(\hat{p}) = \mathrm{Exp}[(\hat{p} - p)^2] \tag{16}$$

It is shown in Section 5.7 that if A_i is a stationary Bernoulli sequence of random variables, then the expression for the variance of the estimate of

its mean may be simplified to the form

$$\mathrm{Var}(\hat{p}) = [\mathrm{Exp}(A_i^2) - p^2]/N \tag{17}$$

Since A_i is a binary variable taking the values zero and unity only, we have

$$A_i^2 = A_i \tag{18}$$

Hence:

$$\mathrm{Exp}(A_i^2) = \mathrm{Exp}(A_i) = p \tag{19}$$

so that

$$\mathrm{Var}(\hat{p}) = (p - p^2)/N = p(1 - p)/N \tag{20}$$

The standard deviation of the estimated value of p is

$$\sigma(\hat{p}) = [\mathrm{Var}(\hat{p})]^{1/2} = [p(1 - p)/N]^{1/2} \tag{21}$$

Thus the expected error in the estimate of p is zero when the probability is zero or unity, which both correspond to deterministic sequences, and takes its maximum value when $p = 0.5$; the error decreases as the square root of the length of the sequence sampled.

4.1.2. Representation II. Two-Line Bipolar

The above representation may be extended to bipolar quantities by representing the sign of the quantity as a logic level on one line, and the magnitude stochastically as above. This may be transformed to an equivalent but preferred arrangement in which positive sign and ON magnitude correspond to the UP line U being ON ($U = 1$), while negative sign and ON magnitude correspond to the DOWN line D being ON ($D = 1$). In this case for a quantity E such that $-V \leq E \leq V$ we have

$$E/V = p(U = 1) - p(D = 1) \tag{22}$$

so that maximum positive quantity is represented by the UP line always ON, and the down line always OFF; maximum negative quantity is represented by the UP line always OFF, and the DOWN line always ON; and zero quantity is represented either by both lines always OFF, or by them both fluctuating randomly with equal probabilities of being ON.

This representation does not give rise to a unique relationship between the probabilities and the quantities they represent. Because two signal lines are used to represent a quantity, there are four possible conditions of the lines, and hence three independent probabilities to be determined. It is

convenient to use a mnemonic notation for these probabilities, so that:

$$p(U = 0,\ D = 0) = v \tag{23a}$$

$$p(U = 1,\ D = 0) = u \tag{23b}$$

$$p(U = 0,\ D = 1) = d \tag{23c}$$

$$p(U = 1,\ D = 1) = c \tag{23d}$$

with the constraint that

$$c + d + u + v = 1 \tag{23e}$$

Then we have:

$$p(U = 1) = u + c \tag{23f}$$

$$p(U = 1) = d + c \tag{23g}$$

Hence, substituting from Eq. (22),

$$E/V = u - d \tag{24}$$

which shows that the conditions where the UP and DOWN lines are both ON, or both OFF, correspond to zero magnitude and do not contribute to the quantity represented.

The mean and variance of an estimate of the quantity represented may be obtained by considering a three-level, or ternary, random variable, B_i ($B_i = 1, 0, -1$) at the ith clock pulse, such that

$$B_i = U_i - D_i \tag{25}$$

where $U_i = 1$ if the UP line is ON, and $D_i = 1$ if the DOWN line is ON, at the ith clock pulse. The expected value of B_i over N clock pulses is

$$\hat{B} = v \cdot 0 + u \cdot 1 + d \cdot -1 + c \cdot 0 = u - d = E/V \tag{26}$$

as required. The variance of $\hat{B}$ is, as shown in Section 5.7,

$$\begin{aligned}\mathrm{Var}(\hat{B}) &= [\mathrm{Exp}(B_i^2) - \hat{B}^2]/N \\ &= [v \cdot 0 + u \cdot 1 + d \cdot 1 + c \cdot 0 - (u - d)^2]/N = [u + v - (u - v)^2]/N \\ &= [u(1 - u)/N] + [v(1 - v)/N] + [2ud/N] \end{aligned} \tag{27}$$

From Eq. (34) it can be seen that the variance of the estimate is minimized if either $d = 0$ ($u > d$) or $u = 0$ ($u < d$). Thus the minimum variance

representation of a bipolar quantity in this way is

$$\begin{aligned} p(U=1) &= E/V \quad \text{if} \quad E \geq 0 \\ &= 0 \quad \text{if} \quad E < 0 \end{aligned} \tag{28a}$$

$$\begin{aligned} p(D=1) &= -E/V \quad \text{if} \quad E \leq 0 \\ &= 0 \quad \text{if} \quad E > 0 \end{aligned} \tag{28b}$$

This still does not uniquely determine the probabilities c, d, u and v, because of the trivial equivalence between both lines being ON and OFF. This can be eliminated by assuming that both lines do not come ON together (a condition which may always be converted to both lines OFF by appropriate gating), so that $c = 0$. Equations (23*e*)–(23*g*), together with (28*a*) and (28*b*), then determine a unique, minimum variance representation, which is characterized by only the UP line, or only the DOWN line, ever being active, depending on whether the quantity represented, E, is positive or negative, respectively. The values of the generating probabilities for this representation are:

$$v = 1 - |E/V| \tag{29a}$$

$$\begin{aligned} u &= E/V \quad \text{if} \quad E \geq 0 \\ &= 0 \quad \text{if} \quad E < 0 \end{aligned} \tag{29b}$$

$$\begin{aligned} d &= -E/V \quad \text{if} \quad E \leq 0 \\ &= 0 \quad \text{if} \quad E > 0 \end{aligned} \tag{29c}$$

$$c = 0 \tag{29d}$$

It may be noted, and will be demonstrated later, that although the minimum variance representation may be established initially, it is not necessarily maintained in a computation.

A further constraint upon the four probabilities c, d, u, and v may be obtained if it is assumed that the probability of one line being ON is independent of the probability that the other line is ON. Then the probability of both lines being ON is equal to the product of the probabilities that each will be ON, i.e.,

$$p(U=1, D=1) = p(U=1)p(D=1) \tag{30}$$

So that, from Eqs. (23*d*), (23*f*), and (23*g*) we have

$$u = (u+c)(u+v) \tag{31}$$

which, by substitution from Eq. (23*e*) may be put in the form

$$ud = cv \tag{32}$$

the condition for statistical independence between UP and DOWN lines. It is clear that if $c = 0$, then either u or d must also equal zero if Equation (29*c*) is to be satisfied. Hence the minimum variance representation proposed above is the only one in which the lines are statistically independent and cannot both be ON together.

4.1.3. Representation III. Single-Line Bipolar

Alternatively, a bipolar quantity E ($-V \leq E \leq V$) may be represented by a binary variable C on a *single* line using the transformation

$$p = p(C=1) = (E/2V) + \tfrac{1}{2} \tag{33}$$

i.e., maximum positive quantity is represented by a logic level always ON, maximum negative quantity by it always OFF, and zero quantity by a logic level fluctuating randomly with equal probability of being ON or OFF.

The inverse transformation is clearly

$$E/V = 2p - 1 \tag{34}$$

This is a linear transformation of the expected value of the generating probability, and if $\hat{p}$ is an estimate of p obtained as in Section 4.1.1, then the best estimate of E/V, $\hat{E}/V$, is

$$\hat{E}/V = 2\hat{p} - 1 \tag{35}$$

The variance of this estimate may be obtained from Eq. (20) using the result of Section 5.7:

$$\mathrm{Var}(\hat{E}/V) = 4p(1-p)/N = [1 - (E/V)^2]/N \tag{36}$$

Hence the variance of the estimate of E is zero when E has its maximum positive or negative value, and the variance is maximum when E is zero.

This result may be contrasted with the corresponding results for the minimum variance version of representation II, where the variance is zero both for extremal values of E and for zero value of E. This difference becomes important when the zero value of a variable is critical to the solution of a problem; e.g., if E were the error in a control system. In this event representation II is to be preferred, while otherwise representation III requires less lines, and hence less computational hardware.

4.2. Stochastic Computing Elements

The computing elements of a stochastic computer are networks of gates and flip-flops, i.e. finite automata, whose inputs are Bernoulli sequences of logic levels representing numerical data. The outputs of the automata are generally also Bernoulli sequences, and it is the relationship between the generating probabilities of input and output sequences which, together with the transformation mapping analog variables into probabilities, determines the computation performed.

As an example of the use of logic gates for analog computations consider a two-input, single-output gate for binary variables. Such a gate can realize one of the sixteen different logical functions of two binary variables, and the particular function realized may be represented by a table of output values corresponding to each of the four different conditions of the input lines. If the input variables are A and B and the output is C, then the table of values of C for an AND gate is as shown in Table I. If now a second table is constructed of the probabilities of occurrence of each input configuration, so that for A and B independent Bernoulli sequences, the probabilities of the four input conditions are as shown in Table II, then cross-

TABLE I

Values of Output C Corresponding to Inputs A and B for an AND Gate

B	C	
	$A = 0$	$A = 1$
0	0	0
1	0	1

TABLE II

B	C	
	$A = 0$	$A = 1$
0	$[1 - p(A)][1 - p(B)]$	$p(A)[1 - p(B)]$
1	$[1 - p(A)](p(B)$	$p(A)p(B)$

TABLE III

Table of Values for an OR Gate

B	C	
	$A = 0$	$A = 1$
0	0	1
1	1	1

multiplication of the corresponding entries in the two tables and summation of the results gives the generating probability of the output.

Thus from Tables I and II we get for an AND gate

$$p(C) = p(A)p(B) \tag{37}$$

The table for an OR gate is given in Table III; from this the generating probability of the output of an OR gate is

$$\begin{aligned} p(C) &= p(A)[1 - p(B)] + [1 - p(A)]p(B) + p(A)p(B) \\ &= p(A) + p(B) - p(A)p(B) \end{aligned} \tag{38}$$

Thus, insofar as the probabilities themselves are concerned, an AND gate performs multiplication, and an OR gate performs imperfect addition [Eq. (38) is a good approximation to addition if $p(A)$ and $p(B)$ are small]. The overall calculation performed also depends on the transformation representing numerical quantities as probabilities. In the following sections standard analog computing elements for each of the three representations defined above are systematically described and analyzed.

4.3. Invertors

To multiply a quantity in representation II by -1 requires only the interchange of UP and DOWN lines (Fig. 5*a*). In representation III a simple logical invertor whose output is OFF when its input is ON performs arithmetic inversion (Fig. 5*b*). There is clearly no inverting element in representation I, since negative quantities cannot be represented.

The inverting action of the logical invertor of Fig. 5*b* may be confirmed by consideration of the relationship between the generating probability of

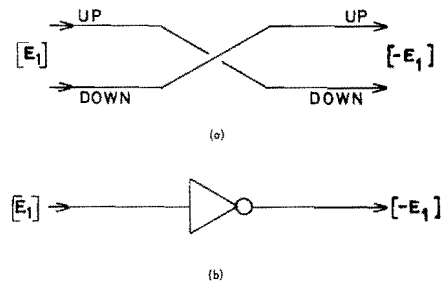

Fig. 5. Stochastic invertors.

its input sequence, p_I and the generating probability of its output sequence, p_0; since the output is OFF when the input is ON, we have

$$p_0 = 1 - p_1 \tag{39}$$

From Eq. (33) the relationship between these probabilities and the quantities they represent is

$$p_i = \tfrac{1}{2}E_i/V + \tfrac{1}{2} \tag{40}$$

and hence

$$E_0 = -E_1 \tag{41}$$

From the symmetry of Equations (27) and (36) it is clear that multiplication by -1 does not change the variance of the quantity represented.

4.4. Multipliers

4.4.1. For Representation I

To multiply together two quantities in representation I, a simple AND gate suffices (Fig. 6*a*). If the two inputs to the gate have generating probabilities p_1 and p_2, and the output has a generating probability p_0, then, since the output is ON only when both inputs are ON, we have

$$p_0 = p_1 p_2 \tag{42}$$

From Eq. (13) the relationship between these probabilities and the quantities they represent is

$$p_i = E_i/V \tag{43}$$

and hence

$$E_0 = E_1 E_2/V \tag{44}$$

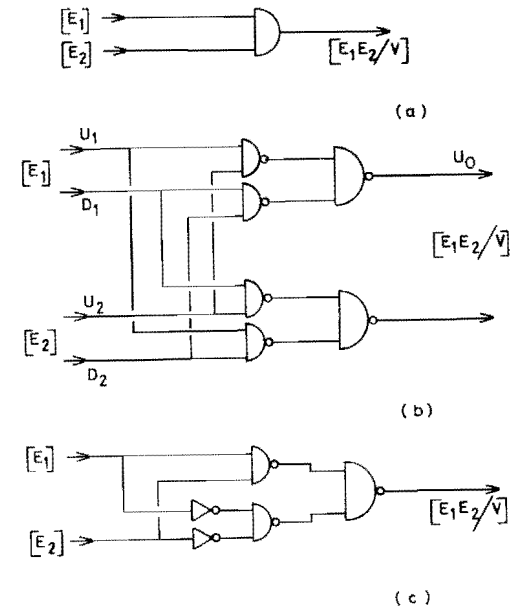

Fig. 6. Stochastic multipliers.

This is multiplication of E_1 by E_2, normalized with respect to the range of the variables, so that the result cannot lie outside the range of representation.

From Eq. (20) the variances of estimates of the p_i over N clock pulses are

$$\mathrm{Var}(\hat{p}_i) = p_i(1 - p_i)/N \tag{45}$$

Substituting for p_0 from Equation (42), we have

$$\mathrm{Var}(\hat{p}_0) = p_1 \mathrm{Var}(\hat{p}_2) + p_2 \mathrm{Var}(\hat{p}_1) - N \mathrm{Var}(\hat{p}_1) \mathrm{Var}(\hat{p}_2) \tag{46}$$

The variances of estimates of p_1 and p_2 are not themselves sufficient to determine that of p_0, because they do not uniquely determine the corresponding probabilities.

4.4.2. For Representation II

For multiplication of quantities in representation II, since they are bipolar and two positive quantities, or two negative quantities, must multiply together to give a positive quantity, it is required that the UP output be ON when both UP inputs are ON, or when both DOWN inputs are ON,

and that the DOWN output should be ON when one UP input and one DOWN input is ON. This may be realized by the gating shown in Fig. 6*b*.

If the four conditions of the input and output lines have probabilities as defined in Eqs. (23*a*)–(23*d*), indexed by 0 for the output, and 1 and 2 for the inputs (so that, e.g., the probability that both output lines will be ON is c_0), then the equations governing the output probabilities are

$$v_0 = v_1 + v_2 - v_1v_2 \tag{47}$$

$$u_0 = u_1u_2 + d_1d_2 \tag{48}$$

$$d_0 = u_1d_2 + d_1u_2 \tag{49}$$

$$c_0 = c_1(1 - v_2) + (1 - v_1)c_2 - c_1c_2 \tag{50}$$

Hence, from Eqs. (48) and (49) we have

$$u_0 - d_0 = (u_1 - d_1)(u_2 - d_2) \tag{51}$$

Since, from Eq. (24*b*) the relationship between these probabilities and the quantities they represent is

$$E_i/V = u_i - d_i \tag{52}$$

we have

$$E_0 = E_1E_2/V \tag{53}$$

which is again normalized multiplication of E_1 by E_2.

If both quantities represented on the input lines to the multiplier are in the minimum-variance form of Eqs. (29*a*)–(29*d*), then at most one of the four terms u_1u_2, d_1d_2, u_1d_2, and d_1u_2 is nonzero. Hence, from Eqs. (48) and (49) either u_0 or d_0 is zero, so that the output representation also has minimum variance. Thus the product of two quantities in the minimum-variance form of representation II is also represented with minimum variance.

The actual variance of an estimate of the quantity represented at the output may be obtained by consideration of Eq. (27) for the input and output variances:

$$\mathrm{Var}(\hat{E}_i/V) = [u_i + v_i - (u_i - v_i)^2]/N \tag{54}$$

Substituting for $(u_0 - d_0)$ from Eq. (51), we have

$$\mathrm{Var}(\hat{E}_0/V) = (u_1 + v_1)\,\mathrm{Var}(\hat{E}_2/V) + (u_2 + v_2)\,\mathrm{Var}(\hat{E}_1/V) - N\,\mathrm{Var}(\hat{E}_1/V)\,\mathrm{Var}(\hat{E}_2/V) \tag{55}$$

4.4.3. For Representation III

In representation III again two quantities of like sign must multiply together to give a positive quantity, so that a gate is required whose output is ON when its inputs are either both ON, or both OFF. Figure 6*c* shows such a gate whose output is ON only if its inputs are equal, and it can be seen to be an EXCLUSIVE-NOR function, or inverted half-adder.

If the two inputs to the gate have generating probabilities p_1 and p_2 and the output has a generating probability p_0, then the relationship between input and output probabilities is

$$p_0 = p_1p_2 + (1 - p_1)(1 - p_2) \tag{56}$$

From Eq. (33) the relationship between these probabilities and the quantities they represent is

$$p_i = [E_i/2V] + \tfrac{1}{2} \tag{57}$$

so that, substituting these values in Equation (56),

$$E_0 = E_1E_2/V \tag{58}$$

The variances of estimates of the input and output variables over N clock pulses in representation III are, from Eq. (36),

$$\mathrm{Var}(\hat{E}_i/V) = [1 - (E_i/V)^2]/N \tag{59}$$

so that, substituting from Eq. (58),

$$\mathrm{Var}(\hat{E}_0/V) = \mathrm{Var}(\hat{E}_1/V) + \mathrm{Var}(\hat{E}_2/V) - N\,\mathrm{Var}(\hat{E}_1/V)\,\mathrm{Var}(\hat{E}_2/V) \tag{60}$$

4.5. Squarers

It is convenient to represent any of the multipliers described in the previous section, and shown in Fig. 6, by the analog multiplier symbol of Fig. 7*a*. An important phenomenon is illustrated by the use of any of the multipliers as a squarer. If the two inputs to the multiplier are connected together, then: in representation I (Fig. 6*a*) the output is the same as the input; in the minimum variance form of representation II (Fig. 6*b*) the output is the modulus of the input; and in representation III (Figure 6*c*) the output is always ON.

These difficulties arise because we have assumed that the stochastic input sequences are statistically independent in obtaining the results of the previous section. Fortunately, a statistically independent replication of a

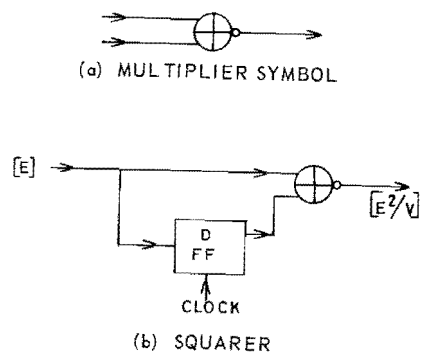

Fig. 7. Multiplier used as a squarer.

Bernoulli sequence may be obtained by delaying it through one clock pulse. The delay may be realized by placing a *D*-type flip-flop in one of the input lines to the multiplier (two flip-flops, for UP and DOWN lines, in representation III, to realize a stochastic squarer as shown in Fig. 7*b*.

Flip-flops used in this way act as stochastic *isolators*, performing no computation but statistically isolating two cross-correlated lines. The necessity for stochastic isolation in the squarer is an example of a general principle applying to all stochastic computation. It is assumed that whenever sequences of logic levels are brought together at the inputs of a computing element they are independent Bernoulli sequences; i.e., they are neither cross-correlated nor autocorrelated. Correlation effects may, if necessary, be removed by use of a delay line of random length. For example, if a stochastic sequence is fed to a shift register and the output is connected randomly, with equal probability, to any of the stages of the shift register, then the generating probability of the output will be equal to that of the input; however, a Markov chain of depth N (Section 5.4) at the input will be reduced to one of depth zero (a Bernoulli sequence) at the output, provided the shift register has more than N stages.

4.6. Summers

Having seen how readily inversion and multiplication may be performed by simple gates, one is tempted to assume that similar gates may be used to perform addition. However, this is not so, and stochastic logic elements must be introduced to sum the quantities represented by two stochastic sequences. For example, consider two stochastic sequences in representation III one representing maximum positive quantity and hence always ON,

the other representing maximum negative quantity and hence always OFF. The sum of these quantities is zero, and this is represented by a stochastic sequence with equal probabilities of being ON or OFF. A probabilistic output cannot be obtained from a deterministic gate with constant inputs, so that stochastic behavior must be built into the summing gates of a stochastic computer.

An alternative way of viewing the problem of summation in the stochastic computer is to consider whether a computation can generate a result which is outside the range of representation of the computer. Probabilities lie in the range [0, 1] but the sum of two probabilities, as is required in representation I, lies in the range [0, 2] and cannot be itself a probability. The weighted sum of two probabilities, $ap_1 + (1 - a)p_2$, where $0 \leq a \leq 1$, does lie in the range [0, 1], however, and such a weighted sum can be formed accurately, and without approximation.

The weighted sum of the generating probabilities of binary sequences on several different lines may be obtained by selecting one of the lines at random, with a certain probability of selecting each one, and connecting the selected line to the output for one clock interval. If there are k lines, each with an independent Bernoulli sequence of generating probability p_i $(0 \leq i \leq N)$, and the probability that the ith line will be connected to the output is a_i, then the generating probability of the output, p_0, is

$$P_0 = \sum_{i=1}^{N} a_i P_i \tag{61}$$

It is important to note that the selection probabilities (a_i) are not statistically independent, in that only one input line must be connected to the output at a time. In practice, they are obtained by gating together several lines having independent sequences—effectively converting a binary code to a "one-in-N-line," or linear, code.

Figure 8 shows a two-input weighted summer for the generating probabilities p_1 and p_2 of binary sequences on the two lines x_1 and x_2. The

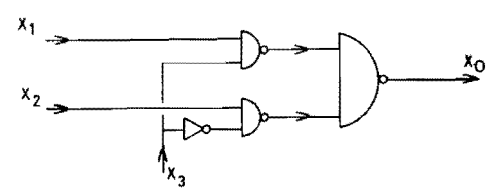

Fig. 8. Stochastic weighted summer.

state of a third line x_3 determines which of these two lines shall be switched to the output x_0. It will be noted that this gate is virtually identical to the multiplier for quantities in representation III (Figure 6*c*). The relationship between generating probabilities at the inputs and that at the output is

$$p_0 = p_3p_1 + (1 - p_3)p_2 \tag{62}$$

Equation (45) gives the variances of estimates of the p_i over N clock pulses in terms of the generating probabilities, and, substituting for p_0, we have

$$\mathrm{Var}(\hat{p}_0) = p_3 \mathrm{Var}(\hat{p}_1) + (1 + p_3) \mathrm{Var}(\hat{p}_2) + (p_1 - p_2)^2 \mathrm{Var}(\hat{p}_3) \tag{63}$$

The final term of this equation may be regarded as the noise variance introduced by the stochastic summing process.

4.6.1. For Representation I

The summer of Fig. 8 acts directly to sum quantities in representation I. Equation (43) gives the relationship between the input and output probabilities and the quantities they represent, and, substituting this in Equation (62) (except for p_3 which is regarded as a weight), we have

$$E_0 = p_3E_1 + (1 - p_3)E_2 \tag{64}$$

The variance of an estimate of the output may be readily obtained from Eq. (63).

4.6.2. For Representation II

Two summers of the form shown in Fig. 8 may be used—one to sum the UP lines of the input to provide the UP line of the output, and the other to sum the DOWN lines of the inputs to provide the DOWN line of the output. From Eq. (62) we have

$$u_0 = p_3u_1 + (1 - p_3)u_2 \tag{65}$$

$$d_0 = p_3d_1 + (1 - p_3)d_2 \tag{66}$$

so that

$$u_0 - d_0 = p_3(u_1 - d_1) + (1 - p_3)(u_2 - d_2) \tag{67}$$

and, substituting from Eq. (52) for the quantities represented by these probabilities, Eq. (64) is again obtained.

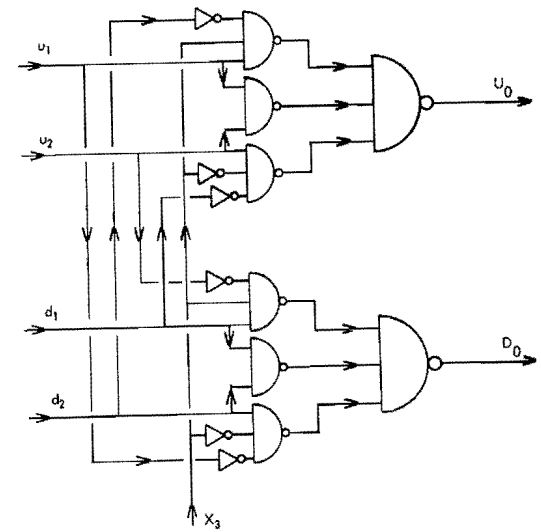

Fig. 9. Minimum variance summer for representation II.

Equation (54) gives the variances of estimates of the quantities represented at the input and output of the summer in terms of the generating probabilities. Substituting for u_0 and d_0 from Eqs. (65) and (66), we have

$$\mathrm{Var}(\hat{E}_0/V) = p_3 \mathrm{Var}(\hat{E}_1/V) + (1 - p_3) \mathrm{Var}(\hat{E}_2/V) + [(E_1/V) - (E_2/V)]^2 \mathrm{Var}(\hat{p}_3) \tag{68}$$

From Eqs. (65) and (66) it is clear that even if both E_1 and E_2 are represented in minimum variance form, then E_0 is not necessarily in minimum variance form. For example, if u_1 and d_2 are both zero, and u_2 and d_1 are nonzero (quantities of opposite signs), then both u_0 and d_0 will be nonzero. In the particular case of equal weighting, when $p_3 = \frac{1}{2}$, it is possible to reduce the variance of the output by the additional gating shown in Fig. 9, which allows positive signals on one set of inputs to cancel out negative signals on the other. The relationship between the generating probabilities of the output and input sequences for Fig. 9 is

$$u_0 = p_3(1 - d_2)u_1 + (1 - p_3)(1 - d_1)u_2 \tag{69}$$

$$d_0 = p_3(1 - u_2)d_1 + (1 - p_3)(1 - u_1)d_2 \tag{70}$$

so that

$$u_0 - d_0 = p_3(u_1 - d_1) + (1 - p_3)(u_2 - d_2) + (1 - 2p_3)(u_1d_2 - u_2d_1) \tag{71}$$

Hence if $p_3 = \frac{1}{2}$,

$$u_0 - d_0 = \tfrac{1}{2}(u_1 - d_1) + \tfrac{1}{2}(u_2 - d_2) \tag{72}$$

so that, substituting for the quantities represented from Eq. (52), we have

$$E_0 = (E_1 - E_2)/2 \tag{73}$$

However, adding Eqs. (69) and (70), we have

$$u_0 + d_0 = \tfrac{1}{2}(u_1 + d_1) + \tfrac{1}{2}(u_2 + d_2) - u_1 d_2 - u_2 d_1 \tag{74}$$

Substituting (72) and (74) in (54), we have

$$\operatorname{Var}(\hat{E}_0/V) = \tfrac{1}{2}\operatorname{Var}(\hat{E}_1/V) + \tfrac{1}{2}\operatorname{Var}(\hat{E}_2/V) + \{[(E_1/V) - (E_2/V)]^2/4N\} - [(u_1 d_2 - u_2 d_1)/N] \tag{75}$$

Comparison of Eq. (75) with Eq. (68) shows that the variance of the output is reduced by the term $(u_1 d_2 - u_2 d_1)/N$.

The effect of this is best seen by considering the two inputs to be equal in magnitude, but opposite in sign, and in minimum variance form: $u_1 = d_2 = a$, and $u_2 = d_1 = 0$, say. From Eq. (68) the variance of the output of a summer based on Fig. 8 is $a/4N$, while from Eq. (75) for the summer of Fig. 9 it is $a(1 - a)/4N$. Hence the larger the quantities to be cancelled out, the greater is the advantage of the more complex summer.

4.6.3. For Representation III

The summer of Fig. 8 may be used directly to add quantities in representation III; substitution of Eq. (57) in Eq. (62) leads immediately to Eq. (64). Substitution of Eq. (59) in (64) gives

$$\operatorname{Var}(\hat{E}_0/V) = p_3 \operatorname{Var}(\hat{E}_1/V) + (1 - p_3)\operatorname{Var}(\hat{E}_2/V) + [(E_1/V) - (E_2/V)]^2 \operatorname{Var}(\hat{p}_3) \tag{76}$$

4.7. Integrators

The elements so far considered have all involved only combinational, or memoryless, logic elements. In the three linear representations discussed, the operations of inversion, addition (and hence subtraction), multiplication, and operations derived from these may be performed with combinational elements alone. For other operations, such as division and square-rooting, implicit functions must be generated using integrators, and clearly integration itself involves storage, and hence the use of elements with memory.

Stochastic computing systems with nondigital storage elements have been described and constructed, but in this present exposition only digital storage elements, based on flip-flops, will be considered. The behavior of stochastic computing systems containing sequential circuits is far more complex than those constructed solely of combinational circuits, and the detailed discussion of this behavior is postponed to Section 7, which follows an outline of some of the necessary stochastic automata theory in Section 5. In the present section only the hardware form of suitable stochastic integrators and superficial demonstrations of their behavior will be discussed.

The basic integrator in any form of incremental computer, such as the DDA or operational digital machine, is a counter, usually a binary counter which may be incremented or decremented by unit count. A symbol for such a counter is shown in Fig. 10*a*; the count increases by unity at a clock pulse if the INCREMENT line is ON and DECREMENT line is OFF, whereas it decreases by unity at a clock pulse if the converse situation holds. If the lines are both OFF or both ON, then the count remains unchanged.

It is convenient to consider the counter as having $N + 1$ states, labeled $S_0, S_1, \ldots, S_N$, and to assign a numerical value to each state, s_i, where s_i is the output of the counter when it is in its ith state. For example, a linear mapping of the count in the counter into the range (0, 1) is

$$S_i = i/N \tag{77}$$

At any given clock pulse the counter will be in some state S, $S = S_i$ say, and its output will have some value s, $s = s_i$. In operation, if the counter is driven by stochastic sequences, its actual state and output may be un-

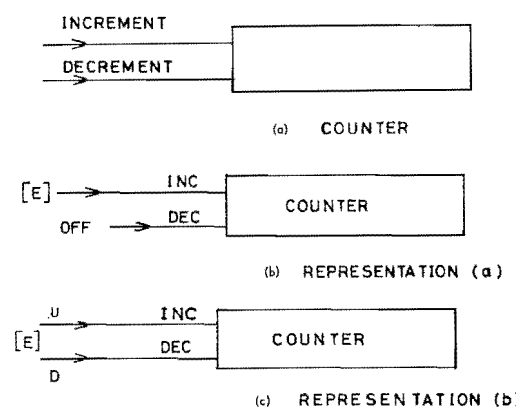

Fig. 10. Stochastic integrators.

predictable, and instead only the probability π_i that it is in the ith state may be known. In this event the output is a random variable, and its expected value $\bar{s}$ may be defined:

$$\bar{s} = \sum_{i=0}^{N} \pi_i s_i \tag{78}$$

If the input lines to the counter are driven from a pair of Bernoulli sequences such that the probability that the INCREMENT line will be ON and the DECREMENT line OFF is w, and the probability that the DECREMENT line will be ON and the INCREMENT line OFF is e, then the expected change in the output of the counter at a clock pulse is

$$\delta s = (w - e)/N \tag{79}$$

If the clock interval is T seconds, then the expected change in the output of the counter over a period of time may be written

$$\bar{s}(nT) - \bar{s}(0) = \sum_{m=0}^{n-1} \delta s(mT) = \sum_{m=0}^{n-1} [w(mT) - e(mT)]/N \tag{80}$$

This last expression may be seen as a simple, zero-order numerical integration formula for $W(t) - e(t)$, and the equation written in approximate form:

$$\bar{s}(t) \approx s(0) + \frac{1}{WT} \int_0^t w(\tau) - e(\tau)\, d\tau \tag{81}$$

Thus the counter may be regarded as an integrator with respect to time, whose gain is $1/NT$. More complex integration formulas than Eq. (80) are generally used for numerical integration, since this requires fine quantization of time to achieve a reasonable accuracy. However, in the stochastic computer time has to be finely quantized anyway because of the inefficiency of the stochastic representation of quantity, and a more complex formula gives no advantage. The exact relationship between analog integration and the stochastic behavior of the counter is discussed in more depth in Section 7.

4.7.1. For Representation I

The counter may be used as an integrator for quantities in all three linear representations previously described. In representation I only positive quantities are represented, and hence the counter need only increment. This is achieved by connecting the line representing the quantity to be integrated to the INCREMENT line of the counter and setting its DECREMENT line OFF (Fig. 10*b*). If E_1 is the quantity being integrated, and E_0 is the quantity represented by the counter in the counter, we have

$$w = E_1/V \tag{82}$$

$$e = 0 \tag{83}$$

$$s = E_0/V \tag{84}$$

Hence from Eq. (81)

$$E_0(t) \approx E_0(0) = \frac{1}{NT} \int_0^t E_1(\tau)\, d\tau \tag{85}$$

4.7.2. For Representation II

In representation II the UP and DOWN lines carrying the sequences representing the quantity to be integrated may be connected directly to the INCREMENT and DECREMENT lines, respectively, of the counter (Fig. 10*c*). Since the output of the counter now has to cover the range of bipolar quantities, it is convenient to use a transformation similar to that of representation III between the expected output of the counter and the quantity it represents:

$$E_0/V = (2\bar{s} - 1) \tag{86}$$

If the quantity represented on the input lines is E_1, represented by probabilities v_1, u_1, d_1, and c_1 [Eqs. (23*a*)–(23*d*)] then

$$w = u_1 \tag{87}$$

$$e = d_1 \tag{88}$$

$$E_1/V = u_1 - d_1 = w - e \tag{89}$$

Substituting these in Eq. (81), the integration equation is obtained:

$$E_0(t) = E_0(0) + \frac{2}{NT} \int_0^t E_1(\tau)\, d\tau \tag{90}$$

It will be noted that the effective gain of the integrator has been increased to $2/NT$.

In many applications a two-input summing integrator which integrates the sum of two variables is required. This may be realized, in any of the representations, by placing the appropriate summer (Fig. 8 or 9) before the

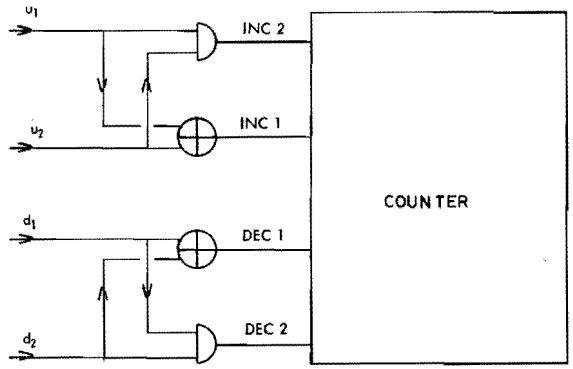

Fig. 11. Two-input summing integrator for representation II.

integrator. However, there is an alternative possibility for two-input, equally-weighted integration, using a counter which can increment or decrement by either one or two units. Figure 11 shows appropriate gating for two quantities in representation II to be summed and integrated in such a counter: the INCREMENT 2 line is ON only if both UP lines are ON; the INCREMENT 1 line is ON if either, but not both, UP lines are ON; similar logic applies to the DECREMENT and DOWN lines. Using the same nomenclature for input quantities and probabilities as before, we have that the expected change in s, δs, is

$$\begin{aligned}\delta s &= [2u_1u_2 + u_1(1 - u_2 - d_2) + u_2(1 - u_1 - d_1) - d_1(1 - u_2 - d_2) \\ &\quad - d_2(1 - u_1 - d_1) - (2d_1d_2)]/N \\ &= (u_1 - d_1 + u_2 - d_2)/N \end{aligned} \tag{91}$$

Hence, by the same argument that led to Eqs. (81) and (90)

$$E_0(t) = E_0(0) + \frac{2}{NT}\int_0^t E_1(\tau) + E_2(\tau)\, d\tau \tag{92}$$

This two-input integrator is readily realized in hardware, but suffers from one defect in that it is rather more difficult to analyze theoretically than the standard counter preceded by a summer. The source of this difficulty is the manner in which the counter may "jump" over a state in its chain of states when it increments, or decrements, by two units. The normal counter has to pass through every state in counting from one extremity to the other, and it will be found (Section 7) that this greatly simplifies the analysis of its behavior.

4.7.3. For Representation III

The counter may be used as an integrator for quantities in representation III by connecting the line representing the quantity to be integrated both to the INCREMENT line of the counter and also, inverted, to the DECREMENT line of the counter (Fig. 12*a*). If the generating probability of the input sequence is p_1, representing a quantity E_1 according to Eq. (33), then the increment and decrement probabilities are

$$w = p_1 \tag{93}$$

$$e = 1 - p_1 \tag{94}$$

so that from Eq. (35)

$$w - e = 2p_1 - 1 = E_1/V \tag{95}$$

Substituting in Eq. (81), we once again obtain Eq. (90).

The integrator of Fig. 12*a* only utilizes the two possibilities of the counter, either incrementing or decrementing. By taking advantage of the third possibility, that no change occurs, it is possible to realize a two-input, summing integrator without using the modified counter necessary for representation III. Figure 12*b* shows the required input gating; the counter increments by unity only if both input lines are ON; it decrements by unity only if they are both OFF; otherwise it does not change its count. Using the notation of Eq. (57) for quantities and probabilities, we have for the increment and decrement probabilities

$$w = p_1p_2 \tag{96}$$

$$e = (1 - p_1)(1 - p_2) \tag{97}$$

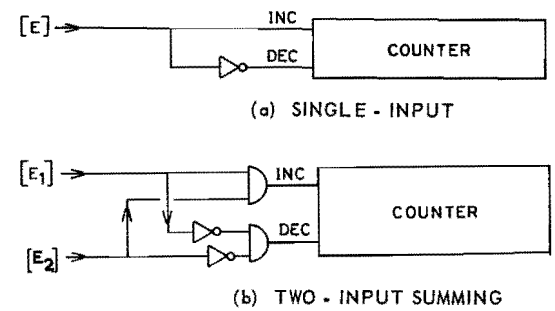

Fig. 12. Integrators for representation III.

so that

$$w - e = p_1 + p_2 - 1 = [(2p_1 - 1) + (2p_2 - 1)]/2$$
$$= (E_1 + E_2)2V \tag{98}$$

Hence, substituting in Eq. (81), we have

$$E_0(t) = E_0(0) + \frac{1}{NT}\int_0^t E_1(\tau) + E_2(\tau)\, d\tau \tag{99}$$

4.8. Stochastic Output from Integrators

The counters in the integrators described in the previous section represent the integral as a stored count in parallel binary form, and have no stochastic output which may be used for further stochastic computation. Such an output may be obtained by comparing the count in the counter with a random variable taking the integer values 0 through $N - 1$ with equal probability. If the output of the comparator is ON when the count in the counter is greater than the random variable, then the probability that the comparator output will be ON, p_0, when the counter is in its ith state is

$$p_0 = i/N = s_i \tag{100}$$

The generation of the random variable for comparison is not a trivial problem, and will be discussed later in Section 4.15 in the context of the general requirement for probability generators in the stochastic computer.

A counter with the necessary generator and comparator is shown in Fig. 13*a*, and it is convenient to subsume these within the counter symbol, presenting the comparator output as a stochastic output of Fig. 13*b*. This, together with the input arrangements of Figs. 10*b* and 12*b*, is a complete integrator for representations I and III, respectively.

In representation II the UP line of the output may be driven directly from the comparator output, and the DOWN line from the inverted output (Fig. 13*c*). We then have

$$u_0 = p_0 \tag{101}$$

$$d_0 = 1 - p_0 \tag{102}$$

so that

$$u_0 - d_0 = 2p_0 - 1 \tag{103}$$

which, from Eq. (86), is the required relationship. This is not a minimum variance output, however, and an alternative form of counter/comparator may be used in which the counter has $2N + 1$ states, labeled $-N \ldots 0 \ldots +N$

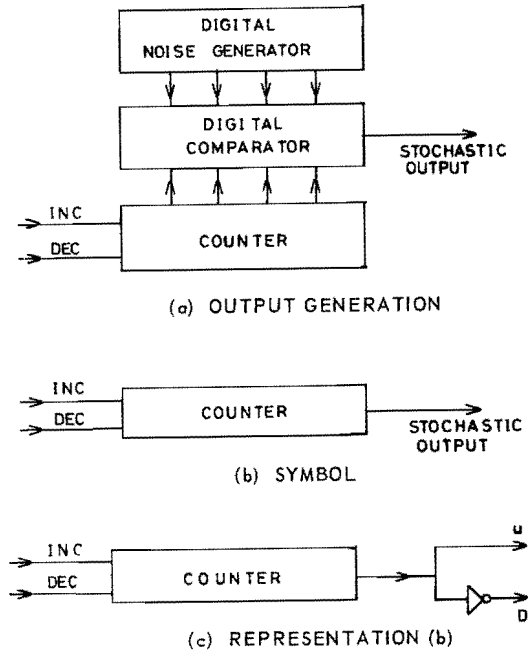

Fig. 13. Stochastic output from counters and integrators.

and the generator generates a random variable, as before, in the range 0 through $N - 1$. The magnitude of count in the counter is compared with the random variable to determine whether an output line shall be ON, and the sign of the count determines which line this shall be. If the random variable is R, then the relationship between count and output probabilities when the counter is in its ith state is

$$u_0 = (|\, i\,| + i)/2N \tag{104}$$

$$d_0 = (|\, i\,| - i)/2N \tag{105}$$

so that

$$u_0 - d_0 = i/N \tag{106}$$

which, since i now takes both positive and negative vaules, is the required result.

4.9. The ADDIE

It is convenient to use the standard analog integrator symbol (Fig. 14*a*) for the two-input, summing stochastic integrators with equal weighting shown in Fig. 12*b*. An important configuration is such an integrator with

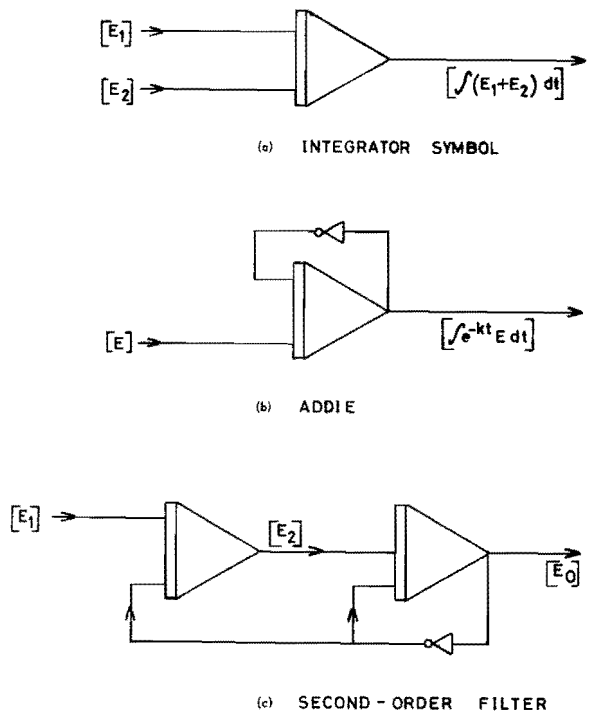

Fig. 14. Integrator configurations.

inverted feedback from its stochastic output to one of its inputs (Fig. 14*b*). The equivalent analog computing configuration is an operational amplifier with a feedback network consisting of a resistor and capacitor in parallel. This is a "leaky integrator," and takes an exponential, or running, average of the quantity at its input, and has a transfer function of the form $1/(as+b)$. A similar result may be proved for the stochastic integrator with negative feedback, which acts as an averaging, or smoothing element for the quantity represented at its input.

The true significance, and importance, of this configuration is, however, best appreciated by consideration of the relationship between the expected count in the counter and the generating probability of its input. In terms of the notation of Section 4.7, if $\bar{s}$ is the expected fractional count in the counter which has $N+1$ states, then, for a Bernoulli sequence of generating probability p at the input, it may be shown that $\bar{s}$ tends exponentially toward an unbiased estimator of p, with time constant NT:

$$\bar{s}(nT) = p + [\bar{s}(0) - p]e^{-n/N} \tag{107}$$

The final variance of s may be shown to be

$$\mathrm{Var}(s) = p(1-p)/N \tag{108}$$

Thus the integrator with feedback acts to form an estimate of the generating probability of a sequence of logic levels in the form of a count in a digital counter. This element was, as described in Section 3.3, the first element of the stochastic computer to be invented, primarily as a probability-estimating device for learning machines, and it was originally called an "ADDIE" (ADaptive DIgital Element). The ADDIE converts a probability to a parallel number and hence acts as a natural interface between the stochastic computer and general-purpose machines, or between stochastic representations and numerical representations of quantity.

The use of an ADDIE as a readout device for probabilities throws new light on the data representation in the stochastic computer. From Eq. (108) it may be seen that the variance of the count in the counter decreases with the number of possible states of the counter. However, from Eq. (107) we see that the time taken for the count in the counter to become an estimate of p increases with the number of counter states. Hence any quantity represented linearly in the stochastic computer may be read out to any required accuracy by using an ADDIE with a sufficient number of states, but the more states, the longer is the time constant of smoothing. If the quantities represented in the computer are static and do not vary with time, so that the corresponding Bernoulli sequences are stationary, then this increased time constant would only affect the speed of computation. If, however, as is required in practice, these quantities are time-varying, then the smoothing acts to reduce their higher harmonic content, and restricts the bandwidth of the stochastic computer.

It may be shown that the distribution of the count in the counter is binomial, and hence the source of variance in $\bar{s}$ may be regarded, for reasonably large N, as a Gaussian noise source added to the quantity represented by the generating probability p. The variance of $\bar{s}$, in representation III, is then the energy of this noise source. From Eqs. (107) and (108) we have the relationship

$$(\text{Noise energy}) \times (\text{Bandwidth}) :: I/T \tag{109}$$

Thus, regardless of the actual output arrangement, there is a fundamental constraint on any stochastic computer, that its noise/bandwidth product can only be improved by decreasing the clock period (i.e., increasing the clock frequency). For a given clock frequency, accuracy can be improved only at the expense of bandwidth, and *vice versa*. However, this tradeoff

may be manipulated as required by use of readout devices with an appropriate number of states, but without any other change in the computing elements.

The full theory of the ADDIE and similar sequential stochastic elements is developed in Section 7.

4.10. Integrators with Feedback

The ADDIE is the simplest example of an integrator with negative feedback used for smoothing. Other smoothing networks may be set up using classical linear network theory as applied to the design of active filters. Figure 14*c* shows a simple second-order filter with variable natural frequency and damping ratio, realized by the use of two stochastic integrators in cascade. The second integrator is connected as an ADDIE and receives the damping feedback, while the first integrator receives its feedback from the inverted output of the second integrator, and hence provides oscillatory negative feedback.

If the first integrator has $M + 1$ states and the second integrator has $N + 1$ states, then the relationship between the quantities represented at their inputs and outputs, as labeled in Figure 14*c* is, from Eq. (99)

$$\dot{E}_0 = (E_1 - E_0)/NT \tag{110}$$

$$\dot{E}_1 = (R_2 - E_0)/MT \tag{111}$$

Hence we have

$$(MNT^2)\ddot{E}_0 + (MT)\dot{E}_0 + E_0 = E_2 \tag{112}$$

Thus E_0 follows E_2 through a stable, second-order transfer function whose undamped natural frequency is

$$f_n = 1/2\pi T(MN)^{1/2} \tag{113}$$

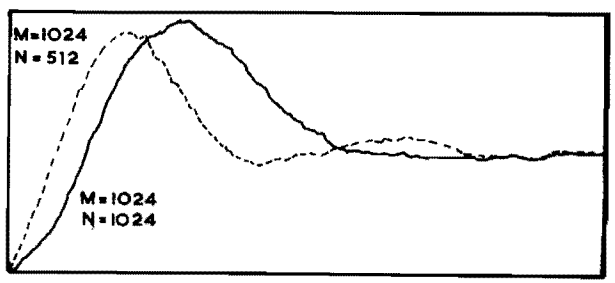

Fig. 15. Response of second-order stochastic filter.

and whose damping ratio is

$$k = (M/N)^{1/2}/2 \tag{114}$$

Typical step responses of the second-order stochastic transfer function are shown in Fig. 15 for the two conditions $M = 2^{10}$, $N = 2^{10}$ and $M = 2^{10}$, $N = 2^9$. These can be seen to be identical in form to those which would be produced by standard analog computing elements, with the addition of a small amount of random noise.

4.11. Dividers and Square-Root Extraction

The computing elements described so far make available a complete complement of computing elements for the normal arithmetic operations of the analog computer—addition, subtraction, multiplication, and integration. Many other operations, in particular, matrix arithmetic and the solution of linear differential equations, can be realized by combinations of these elements. However, there are other operations, such as division, square-root extraction, and switching functions, which cannot be set up directly in terms of these elementary operations. For many of these, approximate computational processes are available which can be used to attain any desired standard of accuracy, usually at the expense of computing speed or bandwidth. In Section 6 some alternative forms of stochastic data representation are discussed which enable a wider range of basic operations to be performed—in particular, division. In the present section some approximate procedures are outlined for the linear representations discussed so far.

Division is inherently difficult when the range of quantities represented is finite, because division by a small enough quantity leads to a result outside the range of representation, and hence a result that must be approximated. Addition of two probabilities p_1 and p_2 also leads to a result, $(p_1 + p_2)$ which is outside the range of probabilities. In Section 4.2 it was shown that an OR gate may be used for approximate addition; it forms the result $(p_1 + p_2 - p_1p_2)$, which is a reasonable approximation when either p_1 or p_2, or both, is near zero. In Section 4.6.1 it was shown that a weighted sum, $(p_1 + p_2)/2$, could be formed exactly. Clearly, no such exact result is possible for division, as the result of a computation may be infinity, and cannot be scaled to lie within the range of the computer. However, approximations of the first kind are readily obtained, particularly through the use of integrators in "steepest-descent" configurations.

A simple circuit which gives an approximate form of division is shown

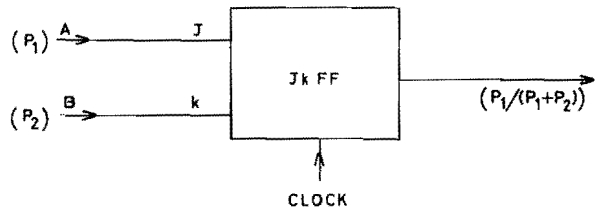

Fig. 16. Approximate divider.

in Fig. 16: a JK flip-flop is driven by two lines such that at a clock pulse it triggers into the state S_A if the line X_1 is ON alone; it triggers into S_B if the line X_2 is ON alone; it reverses its state if both lines are ON; and it remains in the same state if neither line is ON. The output of the circuit comes from one side of the flip-flop, and is ON when it is in state S_A, and OFF when it is in state S_B. If the generating probability of the sequence on X_1 is p_1, and that on X_2 is p_2, then the probability that when the flip-flop is in S_B it will change to S_A is p_1, and the probability of the reverse transition when it is in S_A is p_2. If p_0 is the generating probability of the output sequence, then the probability that the flip-flop is in S_A is also p_0, and the probability that it is in S_B is $(1 - p_0)$. The expected occurrence of transitions from S_A to S_B is $(1 - p_0)p_1$, and this must equal the expected occurrence of transitions in the reverse direction, p_0p_2; hence

$$p_0 = p_1/(p_1 + p_2) \tag{115}$$

If p_1 is very much smaller than p_2, then $p_0 \sim p_1/p_2$; clearly, for fixed p_1, as p_2 itself becomes smaller, the approximation becomes less exact.

A single JK flip-flop acts as an approximate divider for probabilities, and hence for quantities in representation I. A better approximation may be obtained by considering a dynamic error-reducing procedure. Suppose p_0 is an approximation to p_1/p_2, so that p_2p_0 should be equal to p_1. Define the error e to be the difference in these terms:

$$e = p_2p_0 - p_1 \tag{116}$$

Now e^2 is always positive and bounded below by zero, so that any procedure which changes p_0 to make the derivative of this term negative must eventually force e to zero. We have

$$d(e^2)/dt = 2ep_2\dot{p}_0 < 0 \tag{117}$$

Since p_2 is positive, this inequality clearly holds if e and p_0 have opposite

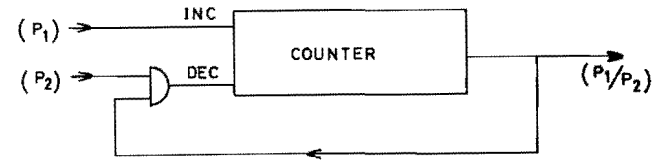

Fig. 17. Divider in representation I.

signs, and we may set

$$\dot{p}_0 = -\alpha e = -\alpha(p_2p_0 - p_1) \tag{118}$$

This equation may be readily realized using the counter with stochastic output of Fig. 13*b*. The stochastic output represents p_0 and is fed back through an AND gate, together with the line carrying p_2, into the DECREMENT line of the counter, to form the term p_2p_0. Similarly, as shown in Fig. 17, the line carrying p_1 is fed to the INCREMENT line of the counter. A simple explanation of how division occurs is to consider that in equilibrium the probability that the count will increase must equal the probability that it will decrease, so that $p_2p_0 = p_1$, and $p_0 = p_1/p_2$. However, this equation cannot hold exactly, since the probability of an increment when the counter is full must be zero, not p_1. The error introduced by this effect requires a deeper analysis of the behavior of the counter, which is given in Section 7; it clearly corresponds to the "limiting" of an analog integrator when the integral would exceed the range if linear operation.

A similar approach may be taken to the division of quantities in representation III. If E_0 is an approximation to E_1/E_2, we define the error E to be

$$E = E_2E_0 - E_1 \tag{119}$$

then the descent to a solution requires

$$d(E^2)/dt = 2EE_2\dot{E}_0 < 0 \tag{120}$$

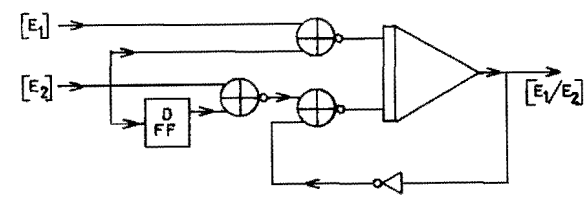

Fig. 18. Divider in representation III.

E_2 is now bipolar, so that this inequality holds if EE_2 and E_0 have opposite signs, and we may set

$$E_0 = -\alpha E_2 E = -\alpha E_2^2 E_0 + \alpha E_2 E_1 \tag{121}$$

A suitable circuit to realize this equation is shown in Fig. 18. It utilizes the multiplier of Fig. 7*a*, the squarer of Fig. 7*b*, and the two-input integrator of Fig. 14*a*. This circuit again only gives approximate division because of limiting in the integrator.

The computational approach to division described in Eqs. (119)–(121) is equally applicable in representation II, and the necessary computing elements, multipliers, squares, and summing integrators have been described in previous sections.

Square-root extraction may be performed using similar dynamic error-reduction, or steepest-descent, configurations. If, e.g.,

$$e = p_0^2 - p_1 \tag{122}$$

then

$$d(e^2)/dt = 4ep_0\dot{p}_0 < 0 \tag{123}$$

which is satisfied if

$$\dot{p}_0 = -\alpha(p_0^2 - p_1) \tag{124}$$

A circuit realizing this equation is shown in Fig. 19. It will be noted again that the equation cannot be realized exactly because this would imply a finite probability of the counter incrementing beyond its full state.

Similar configurations for square-root extraction may be realized in representations II and III. The steepest-descent equations have a wide range of application apart from the solution of implicit equations, and further examples of their use in system identification are given in Section 9.

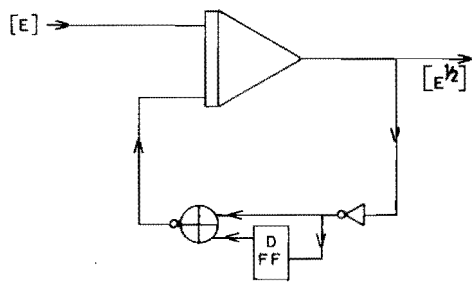

Fig. 19. Square-root extraction.

4.12. Discontinous Functions and Function Generators

A commonly required computational function is a switching, or threshold, element whose output changes stepwise as its input changes through certain values, e.g., an element whose output is ON if the quantity represented at the input is positive, and OFF if it is negative. A stepwise change in output for a small change in probability at the input is clearly impossible for a stochastic computing element, since a probability cannot be measured, but only estimated over a long sequence of inputs. However, as in the implicit computations described in the previous sections, an integrator may be used whose steady-state final condition satisfies the equation of the discontinuity; accuracy in computation may again be traded for a time delay.

The detection of the sign of a quantity in both representations II and III may be performed by a basic counter as shown in Fig. 10*a* without a stochastic output. The output line of the sign detector is connected to the most significant bit of the counter, so that it is ON when the counter is half full or more, and OFF otherwise. For quantities in representation II the UP lines of the input is connected to the INCREMENT line of the counter and the DOWN line to the DECREMENT, as shown in Fig. 10*c*. From Eq. (90) the quantity in the counter represents the integral of the quantity represented at the input, and hence, eventually, the counter will become full or empty depending on the sign of the input quantity. A similar result may be obtained for quantities in representation III by feeding the input line to the counter as in Fig. 12*a*; a comparator for two quantities in this representation may be made by using the two-input summing configuration of Figure 12*b*.

Arbitrary functional relationships between input and output probabilities are also readily realized with integrators connected as ADDIE's. The

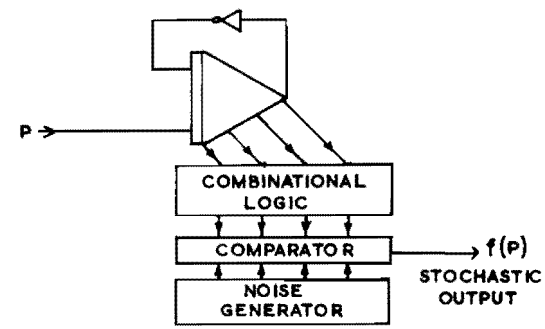

Fig. 20. ADDIE as a function generator.

counter in the integrator eventually contains an estimate of the probability that the input line will be ON, and networks of gates attached to the counter may be used to apply arbitrary transformations to the stored count. This transformed quantity can then be applied to the comparator, as before, to obtain a stochastic output; a diagram of this configuration is shown in Fig. 20. Because of the random variance on the count in the counter, any discontinuities in the function generated will be partially smoothed out. As discontinous a function as required may be realized, however, by using a counter with a sufficient number of states; the settling time of the function generator will, of course, increase proportionately.

4.13. The Outward Interface

The input and output of data to the stochastic computer can be performed in a number of ways, and there are no standard interface elements which would be generally present. A stochastic sequence of logic levels may be regarded as a rate-, or frequency-, modulated pulse stream, which may sometimes be fed directly to systems outside the computer. For example, in a process control system a stepping motor is a convenient actuator which may be regarded as the equivalent of a digital counter, and used as an integrator in the way described in Sections 4.6 and 4.7. If a voltage level is required out of the system, and the clock pulse rate of the stochastic computer is constant, then the sequence of logic levels representing a quantity to be output may be used to trigger a monostable flip-flop which provides constant-area pulses to a resistor–capacitor low-pass network.

The natural outward interface of the stochastic computer is, however, the digital counter used as an integrator. This provides a parallel binary representation of the quantity to be output, and this may be used directly, or converted to any other required form using standard general-purpose-computer interface elements. If no integrator containing the required quantity is already present in the system, then an integrator connected as an ADDIE may be used to convert the required probability to a parallel binary number.

4.14. The Inward Interface

Some transducers naturally produce a stochastic output which may be used almost directly as the input to a stochastic computer. A photomultiplier, e.g., converts individual photons to separate electrical impulses, provided the flux is not too great. Similarly, particles from radioactive sources may be individually detected by using a scintillation screen coupled to a photo-

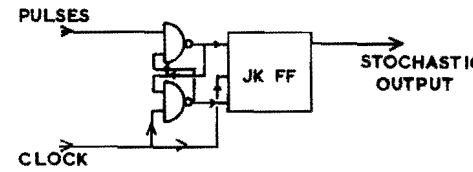

Fig. 21. Synchronizer for stochastic sequences.

cell. The stochastic sequences produced in this way are asynchronous, and must be converted to clock streams of logic levels before they may be used in the stochastic computing systems described in this chapter (asynchronous stochastic computing systems have been investigated, but they too are restricted in their maximum pulse rate).

Figure 21 shows a synchronizing element for converting asynchronous stochastic sequences into clocked Bernoulli sequences: the first pair of NOR gates act as an RS flip-flop, or memory element, recording that an impulse has occurred on the input line; at a clock pulse the state of this flip-flop is transferred to a clocked JK flip-flop driving the output line. Provided two or more pulses do not arrive on the input line during one clock pulse, it is clear that the pulse stream at the output replicates that at the input, except for superimposed time delays of less than one clock period to ensure syncronization. However, the input pulse train is assumed to be stochastic, and hence the number of pulses occurring in one clock interval is not known, or bounded. The distribution of the number of discrete, independent, random events occurring within a given time interval is a Poisson distribution, whose only parameter is the expected number of events in the given interval.

The probability that m pulses will occur in one clock interval when the expected number is k is:

$$p_m = e^{-k}k^m/m \tag{125}$$

The output will be ON if there is one or more pulse(s) during the preceding clock interval. Hence the generating probability of the output p is one minus the probability that there will be no pulses:

$$p = 1 - p_0 = 1 - e^{-k} \tag{126}$$

The difference δ between the output probability p and the expected rate of pulses at the input k is the error in representing the input at the output:

$$\delta = p - k = 1 - e^{-k} - k \tag{127}$$

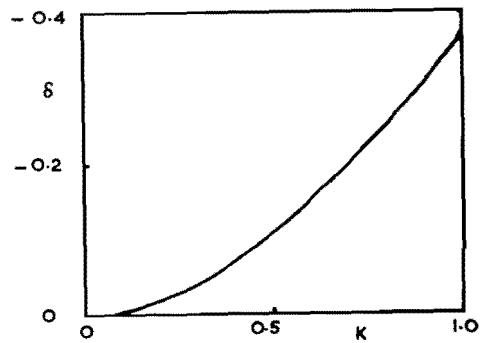

Fig. 22. Error in synchronizer.

A graph of δ against k is shown in Fig. 22; it may be seen that when the expected number of pulses is down to one in five clock intervals ($k = 0.2$), then the error is approximately 2% of full-scale; while when only one pulse in ten clock intervals is expected, the error is 0.5%.

The behavior of p for both small and large values of k is of interest. For small values of k, e^{-k} may be approximated by $(1 - k + (k^2/2) - (k^3/6)\ldots)$, so that Eq. (127) becomes

$$\delta \approx -(k^2/2) + (k^3/6) \tag{128}$$

and hence p is a good approximation to k up to second-order terms. For large values of k Eq. (126) shows that p is asymptotic to unity, but never reaches it, so that it always contains information about the magnitude of k. Hence, although the synchronizer is showing nonlinear behavior, it is not destroying information, and k may always be determined from p. This result holds because the input to the synchronizer is itself a stochastic sequence; with nonprobabilistic inputs, limiting would take place and the value of k would be lost. Thus the stochastic nature of the input enables it to be transformed without loss of information in a way which would otherwise be impossible. In practice, the transformation will not be perfect because no account has been taken of the finite width of a clock pulse, and other divergencies from the theoretical behavior of the input element.

The conversion of analog voltages to stochastic sequences may be performed by generating a Bernoulli sequence of random voltages and comparing the input voltage with a member of this sequence at each clock pulse; if the input voltage is greater than or equal to the random voltage, then the output is ON, otherwise it is OFF. If the probability density function of the random sequence is $\mu(x)$, so that $\mu(x)\,dx$ is the probability that the random

voltage will lie in the small interval $[x, x + dx]$, then the generating probability of the output, p, given an input voltage E is

$$p = \int_{-\infty}^{E} \mu(x)\,dx \tag{129}$$

The distribution (x) may be suitably determined to generate any mapping from quantity to probability. For the linear representations discussed so far $\mu(x)$ is a binary function taking the value zero outside the range of variation of the input quantities and a constant over the total range of variation within it [so that the integral of $\mu(x)$ over all x is unity].

The comparators in Figs. 3 and 4 are examples of this type of conversion, since they use a pseudorandom voltage generated by sampling a sawtooth waveform whose repetition period differs from that of the clock pulse. In these examples the sawtooth lies in the range $-V$ through $+V$, and there is equal probability that a sample from it will lie in any small interval forming part of this range. We have

$$\begin{aligned} \mu(x) &= 0 \quad && \text{for} \quad x < -V \\ &= 1/2V \quad && \text{for} \quad -V \leq x \leq V \\ &= 0 \quad && \text{for} \quad x > V \end{aligned} \tag{130}$$

Hence from Eq. (129)

$$p = \int_{-V}^{E} dx/2V = \tfrac{1}{2} + (E/2V) \tag{131}$$

and hence the input quantity is converted to representation III. Conversion to representation I may be achieved using a random voltage evenly distributed over the range 0–V. Conversion to representation II may be achieved by taking the magnitude of the input and converting to representation I, and determining from the sign of the input whether this shall be switched through to the UP or DOWN line.

Conversion using a comparator in this way involves the generation of analog random sequences, which is generally rather more difficult than the generation of equivalent digital sequences. In particular, the analog levels generated by sampling a sawtooth waveform will be correlated with one another unless the frequency of the sawtooth is very much greater than that of the clock and is anharmonic to it. Ratios of 1000:1 or greater have been found necessary in practice, and result in such a low clock rate that for a given accuracy the bandwidth of the stochastic computer is ridiculously low; e.g., from Section 3.1 a clock rate of 1 kHz and an accuracy requirement of 1% leads to a bandwidth of about 1/200 Hz!

A variation on this technique is to generate random numbers digitally, as described in the following section, and convert them to an analog voltage for comparison; this is expensive, as it requires a digital–analog convertor for each channel. Since the bandwidth of the input to the stochastic computer is restricted by accuracy requirements to be several orders of magnitude below that of the clock frequency, comparison of the input voltage at every clock pulse is unnecessary. The analog input could be sampled at a low rate, converted to digital form by a conventional analog–digital convertor, and the digital result could be loaded into a stochastic integrator whose output would then be the required sequence of logic levels. This is a reasonable system, since a single analog–digital convertor may serve many channels.

A direct and simple conversion from an analog voltage to a synchronous stochastic sequence may be performed if a random pulse source is available whose mean rate may be varied by an applied voltage. Several noise sources with this characteristic are known, including gas-discharge tubes and microplasmas in Zener diodes. The behavior of these latter phenomena have been extensively investigated by the Computer Science Department at the University of Illinois ([1]). Certain diodes have been shown to generate essentially band-limited Gaussian noise. If this is compared with an applied voltage in a comparator, then the switching of the output of the comparator is a random variable, and the probability that the output will be in one state varies monotonically with the applied voltage.

The output of the comparator may be fed to a synchronizing JK flip-flop and a stochastic sequence generated whose generating probability is a monotone function of the applied voltage. This function is, however, very nonlinear, and it is necessary to use this voltage-controlled generator in an analog feedback configuration to obtain accurate analog/stochastic conversion. Figure 23 shows such a converter: the synchronous stochastic output is normalized in voltage so that OFF corresponds to V_0 volts and ON to V_1 volts; this normalized output is applied to one input of an operational amplifier connected as a summing integrator; the other input of the summing integrator is connected to the voltage to be converted, E; the output of the integrator feeds back to the voltage-controlled noise source with negative sense. In equilibrium the output of the integrator will attain a value such that the mean voltages at the inputs sum to zero. Thus if p is the generating probability of the stochastic output,

$$(1 - p)V_0 + pV_1 + E = 0 \tag{132}$$

and hence

$$p = (E - V_0)/(V_1 - V_0) \tag{133}$$

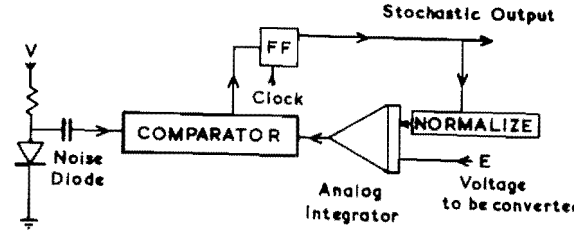

Fig. 23. Analog-to-stochastic convertor.

By taking $V_0 = 0$ and $V_1 = V$ representation I is obtained; by taking $V_0 = -V$, and $V_1 = V$ representation III is obtained; representation II may be obtained by converting magnitude as for I and using the sign to determine whether UP or DOWN lines should be activated.

The feedback loop of Fig. 23 is stable provided the time constant of the integrator is sufficiently dominant over the time constant of the Zener diode. Since the relationship between the generating probability of the output and the voltage controlling the noise source is nonlinear, the loop gain varies with the mean input voltage; however, this is not a major effect. The main defect of the circuit is that the stochastic output is only uncorrelated, and a Bernoulli sequence, if the bandwidth limiting the Gaussian noise is wide compared with the clock frequency; this restricts the clock frequency to a few hundred kHz at present.

The use of an analog integrator with stochastic pulse sequences and analog levels at its inputs has a wide range of potential application in stochastic computing systems. For example, analog integrators may be used to replace the stochastic integrators in the circuits for stochastic division and square-root extraction described in Section 4.11. Such circuits will not be detailed in the present chapter; they have the disadvantage of being subject to drift, etc., and of containing a capacitor not amenable to large-scale integration; they have the advantage of comparatively low cost at present.

The conversion of analog quantities to stochastic sequences is one of the less elegantly solved problems of the stochastic computer. However, in very many applications, particularly in machine learning, the requirement does not arise. The variables entering the computer are nonnumerical, logical inputs, denoting that certain events have occurred, and computations are performed with the probabilities of these events. Similarly, in pattern recognition the input may often be completely quantized as a binary (or N-ary, for small N) array, and analog variables need not be taken into

account. For simulation problems in process control, where all the variables may be essentially continuous, the data and parameters may all be entered in digital form, and no conversion is required.

4.15. Generation of Stochastic Sequences

It was assumed in Section 4.6 that a stochastic sequence with generating probability p_s was available as an input to the weighted summers. Similarly, it was assumed in Section 4.8 that a random variable taking the values 0 to $N-1$ with equal probability was available to convert the counts in counters to stochastic sequences. Throughout the stochastic computer there are required numbers of random variables forming Bernoulli sequences with known generating probabilities, and these random variables must be uncorrelated, one with another, if they ever come together in a computation.

The analog-to-stochastic convertor of Fig. 23 could clearly be used to derive sequences with any required generating probability, but the generation of stable sequences with precisely known distributions is best performed by using only sources whose generating probability is one half. These may be shown to be adequate for all purposes in the stochastic computer. Let A_i, $i = 1, 2, 3, \ldots$, be a set of independent binary random variables, each with a generating probability of $\frac{1}{2}$. Consider the Boolean functions of these variables: $B_1 = A_1$; $B_2 = \bar{A}_1 \cdot A_2$; $B_3 = \bar{A}_1 \cdot \bar{A}_2 \cdot A_3$; $B_4 = \bar{A}_1 \cdot \bar{A}_2 \cdot \bar{A}_3 \cdot A_4$; etc. These have the property that no more than one of the B_i may be ON in any given clock interval, i.e., $B_i \cdot B_j = 0$ for $i \neq f$. The generating probability of B_i is 2^{-i}, so that it is binary weighted, and since the set of B_i is independent, the generating probabilities sum if several sequences are fed to an OR gate. Hence if $x = X_1 X_2 X_3 \ldots$ is a fractional binary number ($X_1 = 0$ or 1) representing, e.g., the fractional count in a counter, then the binary variable X

$$X = X_1 \cdot B_1 + X_2 \cdot B_2 + X_3 \cdot B_3 + \ldots \tag{134}$$

has a generating probability equal to x.

A circuit using NAND gates to realize this logical function is shown in Fig. 24. It has a simple repetitive structure which is readily realized in an integrated circuit array. It may be noted that this circuit replaces the numerical noise generator and comparator of Fig. 13*a* with a set of binary noise generators and a simple logical array. The circuit of Fig. 24 can clearly be used to generate constants in the stochastic computer, either through wiring in the values of the X_i, or by setting them in a register.

Thus the only digital noise sources required in the stochastic computer

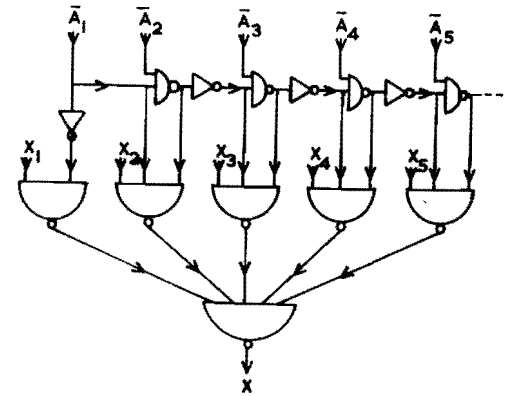

Fig. 24. Variable-probability generator.

are those which generate binary Bernoulli sequences with a probability of $\frac{1}{2}$. One such generator, based on the noise diodes described in the previous section, is shown in Fig. 25; the noise pulses now trigger a flip-flop between its two states, and the state of this flip-flop at a clock pulse is sampled by a JK flip-flop. The generating probability of the output must clearly be $\frac{1}{2}$, but successive logic levels will only be statistically independent if the first flip-flop is toggling at a far higher mean rate than the clock frequency. The error if this is not so may be determined from Eq. (125); it is required that the probability of the first flip-flop being in either of its two states is equal, no matter what its previous state, and hence that the probability of an even number of pulses in a clock interval is equal to the probability of an odd number of pulses:

$$p_0 + p_2 + p_4 + \cdots = p_1 + p_3 + p_5 + \cdots \tag{135}$$

For $k = 3$ the left-hand side sums to 0.5013 and the right-hand side to 0.4987. However, the probability of eight or more pulses occurring within

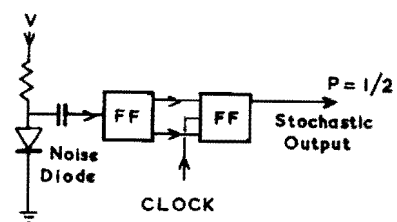

Fig. 25. System for generating output of probability $\frac{1}{2}$.

one clock interval is 0.0121, so that the first flip-flop must be capable of toggling at about ten times the clock rate. Hence a clock frequency of 1 MHz in the stochastic computer requires noise sources with a mean rate of 3 MHz, and logical elements capable of operating at 10 MHz.

The most difficult of these conditions to realize is to keep the mean noise rate at about the geometric mean of the clock pulse frequency and the upper limit of operation of the flip-flop. The Zener-based noise sources are very temperature-dependent, so that completely open-loop operation is not satisfactory. Instead, a similar feedback arrangement to that of Fig. 23 must be used, with the noise source maintained at the required mean frequency by feedback through the integrator. This makes the generation of a single binary noise source rather cumbersome and expensive, and the circuit of Fig. 25 only becomes attractive if an inherently stable noise source is available. A radioactive source with long half-life emitting alpha particles and painted on the surface of silicon detectors is one possibility. It should be fairly easily realized in integrated-circuit form with very many sources available on a single chip of silicon; however, no research has yet been carried out on the feasibility of such a system.

4.16. Pseudorandom Stochastic Generators

The random generators described so far have all relied on truly random phenomena generated by amplified quantum-mechanical effects in suitable physical systems. In recent years much research has taken place, initially in the context of coding theory, on the generation by normal, deterministic automata (i.e., arrays of flip-flop and gates) of digital sequences which have many of the properties of random sequences ([42–44]). For example, the so called "pseudorandom" sequences generated by linear feedback networks around shift registers have an autocorrelation function which is virtually a delta function but repeats after a very long interval. Since these pseudorandom sequences may be generated using the standard synchronous logic devices already found in the stochastic computer, they are an attractive source of "random" digital noise. In particular, they have the additional advantage that simple gate configurations may be used to obtain delayed replicas of the basic sequence, and hence multiple uncorrelated sources may be obtained from the same generator.

The source of pseudorandom digital noise investigated in most detail to date is the "maximal length null sequence" generated by a shift register with feedback to its input from a combination of the outputs of its various stages gated together in EXCLUSIVE-OR gates. The shift-register without

TABLE IV

Truth Table for Exclusive-Or Function

$\oplus$	0	1
0	0	1
1	1	0

feedback, with an output formed by gating together various states in a chain of EXCLUSIVE-OR gates, is a general-purpose linear digital filter. The theory of its behavior has been documented in great depth ([42–46]), and practical applications to noise generation have also been described ([47,48]).

The filter is "linear," and its behavior may be treated within the framework of linear systems theory because the EXCLUSIVE-OR gates act as an adder, modulo 2, and the shift register acts as a pure delay. The exclusive-or function between two variables A and B, written symbolically as $A \oplus B$, is:

$$A \oplus B = A \cdot \bar{B} + \bar{A} \cdot B \tag{136}$$

The truth table for this function is shown in Table IV. It may be seen that this is equivalent to adding the truth values of A and B, modulus 2; that is, so that $2 = 0$ (modulus 2). Modulus addition is a linear operation in that it obeys the superposition principle, provided superposition is evaluated by modulus 2 arithmetic. This makes the behavior of shift-register EXCLUSIVE-OR gate combinations particularly amenable to theoretical analysis. It will also be noted that the exclusive-or operation is *information-lossless* in that $A \oplus B$ and B may be used to regenerate A:

$$A = (A \oplus B) \oplus B \tag{137}$$

This is not true of the AND, OR, NAND, NOR operations.

Figure 26 shows a shift register consisting of three flip-flops acting as

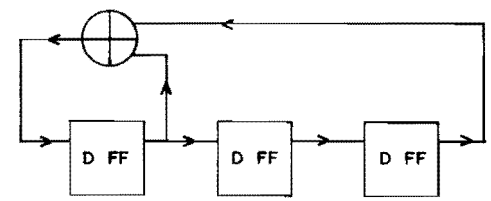

Fig. 26. Feedback shift register.

TABLE V

Time	FF_1	FF_2	FF_3
0	1	1	1
1	0	1	1
2	1	0	1
3	0	1	0
4	0	0	1
5	1	0	0
6	1	1	0
7	1	1	1
8	—	—	— and so on

delays in cascade, with feedback to the input from the output of the first and third flip-flops, gated together in an EXCLUSIVE-OR gate. If the three flip-flops all initially have their outputs ON (1), then the states of the flip-flops at successive clock-pulses are as shown in Table V. It will be noted that the flip-flops pass through every one of their 2^3 possible combinations, except 000, which is clearly a stable combination in its own right. They do not pass through these states in numerical order, as does a binary counter, and it is this property of arbitrary, or pseudorandom, passage through all combinations except zero which makes the feedback shift register a useful digital noise generator.

The behavior of feedback shift registers in general and the determination of feedback configurations having the maximal-length, pseudorandom, property is aided by use of a notation similar to that of ordinary algebra. Let the binary variable at the input of the shift-register be A, and let the previous, delayed value be written CA, and the value before that, D^2A, and so on; for convenience we write $A = 1A$. The equation for the shift register of Fig. 26 may be then written

$$A = DA \oplus D^3A \tag{138}$$

and hence, adding A to both sides, since $A \oplus A = 0$,

$$(D^3 + D + 1)A = 0 \tag{139}$$

The polynomial in D which multiplies A on the left-hand side of Eq. (139) is called the *characteristic polynomial* of the shift register with feedback, and completely defines its behavior. Peterson ([42]) has shown that for

the maximal-length sequence to be generated, the characteristic polynomial must be *primitive*; i.e., it must be irreducible and have no factors, and it must not itself be a factor of $(D^m \oplus 1)$ for any $m < 2^n - 1$, where n is the degree of the polynomial. If the polynomial is not primitive, then the feedback shift register does not have just two state cycles, the maximal-length sequence and the single state zero, but has a number of state trajectories, some forming cycles. For example, if feedback is taken from stages one and two, instead of one and three in Fig. 26, then the characteristic polynomial is

$$(D^2 \oplus D \oplus 1) = (D^3 \oplus 1)/(D \oplus 1)$$

Figure 27 shows the state trajectories for the maximal-length feedback of Fig. 26 compared with the alternative configuration where feedback is from the second flip-flop rather than the third; it will be seen that there are

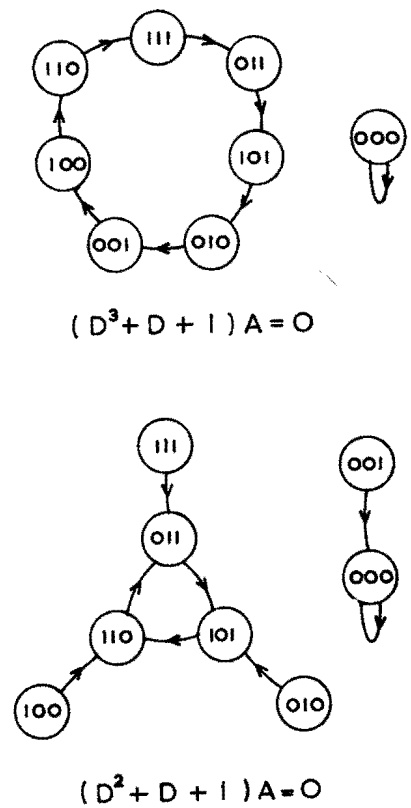

Fig. 27. State trajectories for feedback shift registers.

TABLE VI

Number of stages	Stage to be fed back	Length of sequence
6	1	63
7	1	127
9	4	511
10	3	1,023
11	2	2,047
15	1	32,767
18	7	262,143
22	1	4,194,303
28	3	268,435,455
33	13	8,589,934,591

still only two cycles, but the longest has decreased from seven states to three in length, and the remaining four states are not in a cycle.

Tables of irreducible polynomials enabling maximal-length generators to be designed have been given ([42,47]). In all cases one of the terms to be fed back is the output of the last stage, since otherwise the last stage would be without effect and the shift register could be shortened. In many cases it is necessary to feed back only one other stage, using a single EXCLUSIVE-OR gate. When this stage is the first one it is possible to remove even this gate, by replacing the first flip-flop by a JK flip-flop, both of whose inputs are connected to the output of the last stage. Examples of maximal-length feedback configurations which may be obtained by feeding back the outputs of the last stage and one other, combined in an EXCLUSIVE-OR gate, are shown in Table VI.

One of the remarkable features of maximal-length sequences is that delayed versions of the sequence may be generated by combining the outputs of the various stages in cascaded EXCLUSIVE-OR gates. Thus a delayed sequence may be obtained without using any more flip-flops. It may be shown that any combination of the outputs of various stages through EXCLUSIVE-OR gates is a delayed sequence, and that a sequence of any required delay may be obtained in this way. The computation of the combination required for any given delay is simple, and involves manipulation of the characteristic equation.

For example, the configuration shown in Fig. 26 has the characteristic equation (139). If D^5A is required, we may write

$$D^5A = D^5A + D^2(D^3 + D + 1)A = (D^3 + D^2)A \tag{140}$$

so that D^5A may be obtained by combining the output of the second and third stages in an EXCLUSIVE-OR gate. Similar expressions may be obtained for all other delays. For sequences with many states this procedure must be applied repeatedly, and automatic analysis on a computer is essential.

Since the length of a maximal-length sequence generated by a register of length N is $2N - 1$, the autocorrelation function is repetetive with period $2N - 1$. Its value is

$$\phi(m) = \sum_{i=0}^{2N-2} (D^iA)(D^{i+m}A) \tag{141}$$

This product can be evaluated by noting that for two binary variables A and B taking the values zero and unity

$$(2A - 1)(2B - 1) = 1 - 2(A \oplus B) \tag{142}$$

so that

$$2AB = A + B - A \oplus B \tag{143}$$

Now, $D^iA \oplus D^{i+m}A$ is some delayed replica of A, D^{i+d}, say, unless $m = 0$, when it is identically zero. Hence, combining this result with Eqs. (140) and (141), we have (for $m \neq 0$)

$$2\phi(m) = \sum_{i=0}^{2N-2} (D^iA + D^{i+m}A - D^{i+d}A) \tag{144}$$

Each of the terms being summed is the number of 1's in a complete cycle, which is $2T^{-1}$, and for $m = 0$ the last, negative term is not present. Hence

$$\begin{aligned} \phi(0) &= 2^{N-1} \\ \phi(m) &= 2^{N-2}, \qquad m \neq 0 \end{aligned} \tag{145}$$

Thus the discrete autocorrelation function assumes only two values, one at zero delays, and another at any other delay. It is this which gives the noise generated its "pseudorandom" properties.

The generating probability of the output of any one of the stages may be determined by noting that the maximal-length sequence from an N-state configuration contains 2^{N-1} ones and $2^{N-1} - 1$ zeros. Hence the generating probability p is

$$p = \frac{2^{N-1}}{2^N - 1} = \frac{1}{2 - (1/2^{N-1})} \tag{146}$$

For N greater than eight, p is within 1% of $\frac{1}{2}$, and for larger N it is a correspondingly better approximation to $\frac{1}{2}$. Since delayed replicas of the

output sequence may be obtained so easily, the maximal-length-sequence generator offers a number of Bernoulli sequences with generating probabilities of $\frac{1}{2}$ which are independent of one another up to long time delays. For example, a 33-stage generator has 8.6×10^9 states, and hence may be used to generate 1000 sequences which are completely independent of one another over a period of almost nine million clock periods. Equally, it could be used to generate a million sequences which are uncorrelated over a shorter period. This is more than adequate for any application of stochastic computing envisaged to date.

Some doubts have been raised in the literature as to the degree of randomness of "pseudorandom" numbers generated by shift register configuration ([49,50]). As Korn ([47]), has emphasized, while the first-order statistics and autocorrelation function of maximal-length sequences are very good approximations to those of a random telegraph waveform (binary Bernoulli sequence), the higher moments of the distribution are not the same, and this may have some effect on computations. For example, the subset of the maximal-length sequence regarded as binary numbers would have a binomial distribution if the noise were truly random. Experimentally, the agreement with a binomial distribution is found to be virtually perfect until the length of the sequence exceeds the number of flip-flops in the generating shift register; then the distribution takes on a distinctly skew form. Modifications to the generator to remove this effect have not been successful.

In the stochastic computer the longest sequence used as a binary number is that in the integrator, and one may hypothesize that, provided the number of bits in the shift register generating pseudorandom noise is greater than the number of bits in the integrator counters, no bias in computations will result. This hypothesis has not yet been tested exhaustively in practice, and although preliminary experiments with pseudorandom-noise-based stochastic computing configurations have not shown bias phenomena, the inherent difficulties in testing random processes for small deviations from their theoretical form means that they may yet exist.

One interesting feature of a stochastic computer using pseudorandom noise generators is that, although the theory of its operation is based on the behavior of stochastic automata, it is an entirely deterministic system, and hence subject to conventional automata theory. The relationship between the two possible approaches to the theoretical study of the behavior of such a pseudostochastic computer is an interesting topic for investigation. It may well throw light on the relationship between the behavior of sampled-data, nonlinear systems and that of Markov chains, an important topic to which little attention has been paid since Kalman's pioneering paper ([51]). Until

recently the relationship between the essentially continuum-based analysis of stochastic processes and the discrete, state-space analysis of automata was not amenable to theoretical study. Work by Arbib ([52]) and De Backer and Verbeek ([53]), on the topological properties of automata, and related developments in topological semigroups ([54,55]), suggest that fundamental advances in the analysis of nonlinear systems may take place in the next few years.

The pseudorandom stochastic computer has many advantages—in particular, the noise generation is simply, economically, and reliably performed by a single central element. However, this centralization is itself a disadvantage, as it works against the "neural-net" type of approach which is so attractive in the basic concept of a stochastic computer. To force all elements to run synchronously under the influence of a single central noise generator is unsatisfactory, and in this respect the true noise generators described in the previous section, particularly solid-state radioactive generators, offer the best approach. Noise generation is, however, definitely the weakest link in the stochastic computing structure at present. It may well be that noise-based computers will only really become useful when the data transmission is naturally stochastic, e.g., in an optical computer. For the present, in building working models of small stochastic computers pseudorandom generators are the best noise source.

4.17. Summary

This section has provided an extensive description and analysis of those elements of the stochastic computer which are most similar to conventional analog computing modules—invertors, adders, subtractors, multipliers, squarers, integraters, dividers, explicit and implicit function generation, and interface conversion units for connecting the computer to the outside world.

Configurations of these elements behave in a very similar way to their analog counterparts, with the essential difference that all variables in the stochastic computer have superimposed Gaussian noise. This noise is not constant in amplitude, but varies both with the value of the variable and the bandwidth which is required of the computer. In a given configuration, however, bounds may be placed upon the peak value of the noise, and computations of the associated noise variance have been given for all the elementary computing elements.

The analysis of more complex computing elements, especially those with memory, such as the integrator, is more difficult, and requires a knowl-

edge of stochastic automata theory and discrete Markov chains. A brief overview of these topics is given in the following section, and integrator-like elements are analyzed in some depth in Section 7.

The stochastic computer becomes of greatest interest, however, when the difference, rather than the similarity, between its operations and those of conventional computers are being examined. In Section 6 a nonlinear representation of data, particularly those with a representation of infinite quantities, are considered. Although these representations could be used with analog computing elements, even the most elementary operations, such as addition, involve a number of multiplications, and hence the simplicity of stochastic multipliers is particularly important. In Sections 8–10 application of the stochastic computer to particular computations in pattern recognition and machine learning are considered, with special emphasis on those computations where there is no equivalent computing system using conventional techniques.

5. THE THEORETICAL FOUNDATIONS OF STOCHASTIC COMPUTING [55a]

The necessary foundations for a rigorous theoretical analysis of stochastic computing lies in stochastic (or probabilistic) automata theory, which is a comparatively recent development ([56,57]). The relevant material on stochastic processes themselves, and particularly the analysis of discrete Markov chains ([58,59]), is of a somewhat earlier date (1906), but probability theory, and the notion of randomness, are themselves comparatively recent concepts.

The theory of stochastic automata is peculiar in that, unlike deterministic automata theory, it was not developed primarily for the purpose of synthesizing systems with the behavior studied. Instead, it has been regarded as far more relevant to the analysis of *unwanted* behavior in systems intended to be deterministic; the problems of unreliability and race hazards in sequential circuits are best treated as if they were caused by random disturbances to an otherwise deterministic system. Thus the problems tackled by stochastic automata theorists, and the results obtained to date, are not necessarily central to the requirements of stochastic computers (in which noise is used constructively).

It this section the relationship between stochastic automata and deterministic automata with stochastic inputs is examined and the notion of a Markov chain introduced. The centrality of the Rabin-Paz theorem on

equivalence between stochastic and deterministic automata is discussed, and finally some results in probability theory, relevant to the measurement of stochastic processes, are outlined.

5.1. Models of Imperfect Digital Computers

The basic mathematical object used to represent the behavior of a digital computer is the finite-state automation ([60–62]). The essential features of this object are a finite set of states, inputs and outputs, together with mappings from all state-input pairs to the subset of states and the subset of outputs. In a "noiseless," or perfectly reliable, computer the input sequence is uniquely specifiable by a control (program or data) tape, and the input sequences to the automaton can be treated as a representation of the actual "input" to the computer. If the digital computer is imperfect, then its inputs will not exert complete control over the behavior and a distinction must be made between the specified "input" and the actual "input." We shall call the former a control and reserve the term input for the latter; in a perfectly reliable computer there is a trivial isomorphism between controls and inputs.

The automaton taken to represent an unreliable computer is generally larger than that representing the equivalent reliable computer, for each of its controls gives rise to a set of possible inputs, and its state set must contain states which arise through component failure. In practice the finite-state automaton subsuming all possible faults is so large and unwieldy that a statistical approximation to its behavior is more useful. The mathematical object developed as a statistical representation of an unreliable computer is the stochastic automaton, and the theory of such objects treats transitions between hyperstates (state distributions) in much the same way that deterministic automata theory treats transitions between states. The theory of stochastic automata is best approached through the matrix representation of finite deterministic automata, and this is outlined below.

5.2. Matrix Representation of Finite Automata

Any finite automaton may be represented by a set of matrices, each matrix corresponding to an input and containing only ones and zeros, which define the transitions between its states and the relationships between its states and outputs. The element in the ith row and jth column of a transition matrix for a particular input will be 1 if that input would cause the automaton in its ith state to transit to its jth state (at the occurrence

of a clock pulse). Similarly, the ijth element of the output matrix for a particular input will be 1 if that input gives rise to the jth output when the automaton is in its ith state. Since a state and an input give rise to a unique output and next state, these matrices are characterized by having a single 1 in every row, and are therefore particular examples of probability matrices (whose rows sum to unity). This representation of a finite automaton reduces it to a semigroup of matrices, with its behavior determined by matrix multiplication.

Let the set of possible inputs to an automaton be (I_i), $0 < i \leq N$; the set of possible states be (S_i), $0 < i \leq M$; and the set of possible outputs be (O_i), $0 < i \leq P$. Let the state of the automaton be S, its output O, its input I, and the state after the next clock pulse S'. We have the transition mapping

$$S' = \sigma(S, I) \tag{147}$$

and the output mapping

$$O = \theta(S, I) \tag{148}$$

The $M \times M$ transition matrices (P^k) may now be defined. If $P^k \equiv (p_{ij}^k)$ is the transition matrix corresponding to an input I_k, then

$$\begin{aligned} p_{ij}^k &= 1 \quad \text{if} \quad S_j = \sigma(S_i, I_k) \\ &= 0 \quad \text{otherwise} \end{aligned} \tag{149}$$

Similarly, the $M \times P$ output matrices (O^k), where $O^k \equiv (o_{ij}^k)$ corresponds to the input I_k, may be defined

$$\begin{aligned} o_{ij}^k &= 1 \quad \text{if} \quad O_j = \theta(S_i, I_k) \\ &= 0 \quad \text{otherwise} \end{aligned} \tag{150}$$

The existence and uniqueness of the output and next state imply

$$\sum_{j=1}^{M} p_{ij}^k = 1 \tag{151}$$

$$\sum_{j=1}^{P} o_{ij}^k = 1 \tag{152}$$

Having defined the transition and output matrices, we may consider the behavior of the automaton to be determined entirely by the rules of matrix multiplication. For example, the sequence of inputs I_{λ_1} followed by I_{λ_2} and so on up to I_{λ_n}, applied when the automaton is in state S_i, leads

to state S_j if and only if

$$(P^{\lambda_1} P^{\lambda_2} P^{\lambda_3} \cdots P^{\lambda_n})_{ij} = 1 \tag{153}$$

and the output of the automaton will then be O_j if and only if

$$(P^{\lambda_1} P^{\lambda_2} \cdots P^{\lambda_n} O^{\lambda_n}) = 1 \tag{154}$$

These equations could be reduced to matrix/vector equations by defining the state of the automaton to be an M-vector whose ith element is unity if its state is S_i, and zero otherwise, and defining the output of the automaton to be a P-vector whose ith element is unity if its output is O_i, and zero otherwise. Transitions between states are determined by post-multiplying the present state vector by the transition matrix corresponding to the input, and the output is determined by post-multiplying it with the output matrix corresponding to the input. This matrix representation of finite automata is cumbersome in practice, but ideal for the conceptual transition from "noise" at the input to stochastic automata.

5.3. Stochastic Automata

Consider now a set of "controls" to the automaton which do not specify the input exactly, but rather give rise to probability distributions over the inputs. If the present state and control are known, then the probability of occurrence of each input may be used to calculate the probability that the next state will be a given state. Thus only the state distribution can be predicted. This distribution will be called a complete hyperstate, and the theory of stochastic automata concerns the matrix representation of transitions between complete hyperstates.

Let the set of controls to the automaton be (C_i), $0 < i \leq Q$, and let the application of the control C_i give rise to a probability ν_i^j that the input I_j will occur. Since some input must occur, we have

$$\sum_{j=1}^{N} \nu_i^j = 1 \tag{155}$$

If the automaton is in state S_i when the control is C_k, then the probability π_{ij}^k, that it will next be in state S_j is

$$\pi_{ij}^k = \sum_{r=1}^{N} \nu_k^r p_{ij}^r \tag{156}$$

Thus if the hyperstate of the automaton is the distribution (π_i), where π_i

is the probability that the automaton is the state S_i, then the hyperstate (π_i') after a control C_k is given by

$$\pi_j' = \sum_{i=1}^{M} \pi_i \pi_{ij}^k$$

$$= \sum_{i=1}^{M} \pi_i \sum_{r=0}^{N-1} v_k^r p_{ij}^r \tag{157}$$

The matrices $\pi^k \equiv (\pi_{ij}^k)$ are probability matrices, since their elements are nonnegative, and

$$\sum_{j=1}^{M} \pi_{ij}^k = 1 \tag{158}$$

they may be regarded as generalizations of the matrices P^k.

Probabilistic output matrices relating controls, hyperstates, and outputs may similarly be defined. This structure of controls, hyperstates, outputs, and probability matrices is a stochastic automaton. It is interesting to note that we may regard the stochastic automaton either as a special case of the finite automaton in which probability distributions are assigned to incompletely specified inputs, or as a generalization of the matrix representation of finite automata in which arbitrary probability matrices replace those previously containing only ones and zeros.

5.4. Markov Chains

The discrete transitions of a system from condition to condition are said to form a Markov chain ([58,59]) of the nth order if the probability of a transition to a new condition depends only on the previous n conditions. A Markov chain of the zeroth order is called a Bernoulli sequence, and is distinguished by the statistically independent generation of its elements. If the controls to a stochastic automaton are constant, or from a Bernoulli sequence, then the state sequences are Markov chains of the first order, and the output sequences are Markov chains generally of higher order (the output chains are first order if there is an inverse mapping from output–input pairs to states). If the controls are functions of previous outputs, or are generated in set sequence, then both the states and the outputs may form Markov chains of higher order. In stochastic computers processing Markov chains much of the simplicity of computation would be lost if the order of the chains increased at each stage, and stochastic computing elements are designed to receive and emit Bernoulli sequences.

The relationship between the average behavior of an ensemble of stochastic automata, which is the conceptual basis for their operation, and the average behavior over a number of transitions, which is the practical means for observing their operation, depends on the ergodicity of the Markov process generating the transitions. A hyperstate is said to be stationary for a transition if it does not change under that transition, and a sequence of transitions with one and only one stationary hyperstate is said to be ergodic. Stationary hyperstates in the stochastic computer correspond to steady states in the analog computer, having similar properties in that they are far more tractable theoretically than the transient, nonergodic behavior, and often act as "solutions" in a computation.

5.5. The Computations of Stochastic Automata

Deterministic automata theory investigates the set of input sequences which will take the automaton from a given initial state to one of a set of final states (these are usually included as part of the definition of the "automaton"); these sequences are called the input tapes "accepted" by the automaton and are said to define an "event." With stochastic automata one may consider only the probability that a control sequence will take the automaton from a given initial state to one of the given final states, and define the set of control tapes accepted by a stochastic automaton to be those for which this probability is greater than a threshold δ $(0 \leq \delta \leq 1)$.

To determine by experiment whether a given control tape is accepted by a stochastic automaton with threshold δ requires more and more instances as the actual probability of transition to one of the given final states approaches δ. It is only if this probability is bounded away from δ for all possible control tapes that an experiment of predetermined length may be used to decide whether a tape is accepted by the automaton with any required confidence; a threshold with this property is said to be isolated.

Rabin ([56]) and Paz ([57]) have shown that for every stochastic automaton with an isolated threshold there is a deterministic automaton which is equivalent in that it accepts the same set of control tapes. However, the deterministic automation may be less economical in storage, requiring more internal stages than the stochastic automaton. There is, of course, no real gain in storage, since an external store is required for the results of the series of experiments which determine whether the stochastic automaton "accepts" a particular tape. However, this result has practical applications, e.g., in the "Enhancetron," the waveform averager described in Section 3.2.

The first stage of the "Enhancetron" may be regarded as a two-state, stochastic automaton, receiving input tapes of unit length (in fact, analog voltages, but taken to be finely quantized for the sake of this example). The shift of storage burden is an advantage in this application because an external store is already needed for the averaging process.

5.6. Current Status of Stochastic Automata Theory

Stochastic automata theory is a comparatively new field of research, and although there has been steady progress since the pioneering work of Rabin and Paz, there are as yet few major studies or results. Much present effort is devoted to the determination of theorems equivalent to those already obtained for deterministic automata—e.g., on state minimization ([63,64]), decomposition ([65]), equivalences between machines ([66,67]), etc. Some of the previous studies concerned with computer reliability and Markov processes are also relevant to these developments.

The first application of the theory of random processes to computer design was that of von Neumann ([68]), who showed that under certain conditions arbitrary reliability could be obtained from a computer made of unreliable components through the use of parallel redundancy. This result bears a striking resemblance to Shannon's coding theorem for a discrete channel, and the work of Winograd and Cowan ([69]) shows that this resemblance is more than superficial. In practice, digital computers are still designed with little redundancy and with error-detection rather than error-correction, and work on computer reliability is more suggestive of the advantages of brainlike, homeostatic artifacts than of new developments in conventional computers ([70]).

5.7. The Variance of Bernoulli Sequences

One property of zero-order Markov chains, or Bernoulli sequences, that has been used many times in the section on the linear stochastic computer is the relationship between the variance of the estimated mean value of the sequence and the mean value of each member of the sequence squared. Since the properties of random variables can only be estimated with a certain expectation of error, it is important to have some advance knowledge of the error inherent in a computing configuration, and this result is the basis for such knowledge.

Suppose X_i is a stationary Bernoulli sequence of random variables taking real, finite values. The stationarity of the sequence implies that the

distribution function of the random variable is not a function of i. The fact that the sequence is "Bernoulli" implies that the distribution of X_i is independent of that of the value of X_j for $j \neq i$. In particular, it implies

$$\mathrm{Exp}(X_i X_j) = \mathrm{Exp}(X_i)\,\mathrm{Exp}(X_j) = X^2, \qquad j \neq i \tag{159}$$

where X is the expected value of X_i (and of X_j because of the stationarity of the sequence).

The mean value of X_i, from $i = 1$ to N, say, is an estimate of the expected value of X_i:

$$\hat{X} = \sum_{i=1}^{N} X_i/N \tag{160}$$

The variance of this estimate is defined to be the expected value of the square of the difference between the estimated value of the mean of X_i and the expected value of X_i:

$$\begin{aligned}
\mathrm{Var}(\hat{X}) &= \mathrm{Exp}[(\hat{X} - X)^2] = \mathrm{Exp}\left\{\left[\sum_{i=1}^{N} (X_i/N) - X\right]^2\right\} \\
&= \mathrm{Exp}\left[(1/N^2)\sum_{i=1}^{N} (X_i - X)^2\right] \\
&= (1/N^2)\,\mathrm{Exp}\left[\sum_{i=1}^{N} (X_i - X)^2 + 2\sum_{i=1}^{N}\sum_{j=i+1}^{N} (X_i - X)(X_j - X)\right]
\end{aligned} \tag{161}$$

Now

$$\begin{aligned}
\mathrm{Exp}[(X_i - X)(X_j - X)] &= \mathrm{Exp}(X_i X_j) - X\,\mathrm{Exp}(X_i) - X\,\mathrm{Exp}(X_j) = X^2 \\
&= 0
\end{aligned} \tag{162}$$

using Eq. (159). Hence from Eq. (161)

$$\begin{aligned}
\mathrm{Var}(\hat{X}) &= (1/N^2)\,\mathrm{Exp}\left[\sum_{i=1}^{N} (X_i - X)^2\right] \\
&= (1/N^2)[N\,\mathrm{Exp}(X_i^2) - NX^2] \\
&= [\mathrm{Exp}(X_i^2) - X^2]/N
\end{aligned} \tag{163}$$

This expresses the variance of an estimate of the mean of a Bernoulli sequence in terms of the expected value of the square of the sequence, and hence enables the expected error in estimation to be calculated from the distribution of the random variable.

6. ALTERNATIVE STOCHASTIC REPRESENTATIONS OF QUANTITY

The three representations of analog quantities by probabilities discussed in depth in Section 4 are basic examples of linear mappings between quantity and probability. There is an infinite variety of possible mappings, all of which lead to a certain range of computations and associated computing elements. The choice between these mappings must be based largely on the type of computation to be performed, both upon the range of variables required and the type, and relative numbers, of operations required. The representations II and III of Section 4.1 give rise to a family of stochastic computing elements similar to those of the conventional analog computer; the range of variables is bipolar and bounded, and the operations which may be performed include addition, subtraction, multiplication and integration. If quantities near zero are especially important, then representation II has the advantage of enabling more accurate representation of small quantities than III, but otherwise has the disadvantage of requiring two lines to represent one quantity.

The main difference between these stochastic computing elements and those of the analog computer is that stochastic multiplication is a very cheap and simple operation. This has already been taken advantage of in the steepest-descent implicit-function generators of Section 4.11. It is, however, an important factor in determining the relative attractiveness of other stochastic representations of quantity. All the mappings of quantity into probability proposed for the stochastic computer can also be clearly used with the conventional analog computer, but if the basic operations in these representations, whatever they are, demand the multiple use of expensive elements, such as multipliers, then the representation is not attractive.

The theoretical treatment of nonlinear representations of quantity is generally difficult because the expectation operator in the theory of random variables is essentially linear—for a scalar a and random variables A and B we have $\mathrm{Exp}(aA + B) = a\,\mathrm{Exp}(A) + \mathrm{Exp}(B)$, but in general, $\mathrm{Exp}(A/B) \neq \mathrm{Exp}(A)/\mathrm{Exp}(B)$, and $\mathrm{Exp}[f(A)] \neq f[\mathrm{Exp}(A)]$. In dealing with nonliner representations it is important to have a "feel" for stochastic variables, and in the following section some metric properties of probability spaces are examined to this end.

6.1. The Equal-Accuracy Metric on a Probability Space

When a quantity is mapped into a probability the evaluation of computations using that quantity is complicated because the accuracy of estimation of a probability varies with that probability. The variance of an estimate of a probability is greatest when the probability is one half and is zero when the probability itself is unity or zero. This induces a topology on the space of input quantities in that quantities which map near zero or unity probability are easily distinguished at the output, whereas quantities which map near $\frac{1}{2}$ probability are difficult to distinguish. An obvious measure of the "distance" between two quantities is the inverse of the time taken to distinguish between them with a given accuracy.

In Section 4.1.1 it is shown that the variance in the estimate of the generating probability p of a binary Bernoulli sequence is $\sigma^2 = p(1 - p)/N$, where the estimate is taken over N clock pulses. The standard deviation, which is the square root of this variance, is a measure of the difficulty of discriminating between a generating probability p and one of $p + \delta p$ a small amount away. The difference between the two probabilities may be measured to any accuracy dependent on the number of standard deviations between them, so that for constant accuracy of discrimination we have $\delta p/\sigma$ is constant.

If we define probabilities which may be discriminated in a given time with a given probability of error as being unit distance apart, then we have that the increment of distance δs corresponding to the increment of probability δp is

$$\delta s = \delta p/[p(1 - p)]^{1/2} = \delta p/(pq)^{1/2} \tag{164}$$

where $q = 1 - p$, and the scaling factor in s has been set at unity. This may be regarded as a first-order differential equation relating s and p, and it is clearly satisfied by

$$s = \sin^{-1}(2p - 1) \tag{165}$$

A graph of s against p is shown in Fig. 28, and it can be seen that for values of p around $\frac{1}{2}$, the rate of change of s with p is rather less than at the two extreme values of p.

From Eq. (165) it appears that for a bipolar quantity E such that $-V \leq E \leq V$ a mapping of the form

$$p = \sin[\pi(E + V)/4V] \tag{166}$$

leads to an equivalence between the data space and the probability space, in that equal changes in the data cause equally discriminable changes in the probabilities representing them. The computations performed by simple logic elements in a computer using this representation are interesting, and

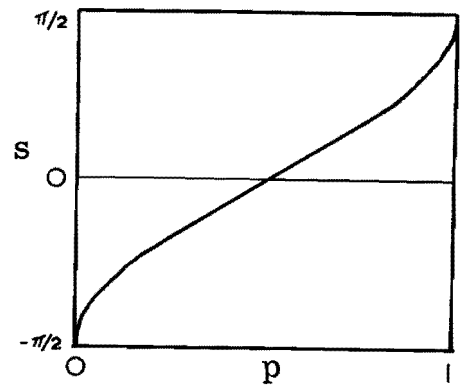

Fig. 28. Metric on the probability interval.

readily ascertained from the standard trigonometric relationships for sin A, sin B, and so on, but do not appear to be particularly useful.

In practice, of course, it is rare that uniform absolute accuracy is required over the range of data, and a requirement for something approaching uniform proportional accuracy, as on a slide rule, is more common. Truly proportional accuracy has the defect of requiring an infinite expansion of the scale about zero, and a compromise is necessary in any real system. This compromise is of the wrong form in the linear stochastic computing using representation III because it is the extremes of the range which are expanded to have greatest accuracy. In the following sections mappings are discussed which compress the scale of large quantities, even to the extent of enabling numerical data with an infinite range to be expressed in a finite probability interval.

6.2. Single-Line Representation of Unipolar Quantities with Infinite Range

Let $E \geq 0$ be a positive quantity, not necessarily bounded, and consider the transformation from E to a generating probability p of a binary sequence in a stochastic computer, such that

$$p = E/(e + E) \tag{167}$$

where $e > 0$ is a scaling factor. When E is zero, so is the generating probability; when $E = e$ the generating probability is $\frac{1}{2}$; and, for the probability representing E to become unity, E itself must be infinite in magnitude. This appears most clearly in the inverse transformation:

$$E = ep/q \tag{168}$$

Differentiating Eq. (68), we have

$$dE/dp = e/q^2 \tag{169}$$

and, substituting this in Eq. (161),

$$\delta s = e^{1/2}\, \delta E/[(e + E)E^{1/2}] \tag{170}$$

Hence the discrimination between different values of a quantity E, this representation varies as $E^{-1/2}$ for small values of E and as $E^{-3/2}$ for large values of E (compared with e). This may be compared with a truly proportional scale, in which the discrimination would vary as E^{-1}. Such a scale is logarithmic, and a range of quantities cannot be mapped into a finite range. The representation of Eq. (167) gives a scale which is similar to a proportional one for values of E near e, but varies more slowly below this value and more rapidly above it.

The standard gating functions forming p_1p_2, $(p_1 + p_2)/2$, etc., described in Section 4 also perform computations in this representation. For example, the EXCLUSIVE-OR gate of Fig. 6*c* such that the output probability p_0, is related to the input probabilities p_1 and p_2 by

$$p_0 = p_1p_2 + q_1q_2 \tag{171}$$

with a representation such that

$$p_i = E_i/(e + E_i) \tag{172}$$

performs the computation

$$E_0 = (e^2 + E_1E_2)/(E_1 + E_2) \tag{173}$$

which is reminiscent of the addition of hyperbolic tangents.

The standard arithmetic operations of addition, multiplication, and division may be performed in this representation by using a JK flip-flop in a similar configuration to that for approximate division (Section 4.11). First it may be noted that a logical invertor forms the reciprocal of the quantity represented on its input line; the relationship between the generating probability of the output p_0 and that of the input p_1 is

$$p_0 = 1 - p_1 \tag{174}$$

and hence from Eq. (172)

$$E_0 = e^2/E_1 \tag{175}$$

Thus any multiplier in this representation may also be used as a divider by inverting one of its inputs.

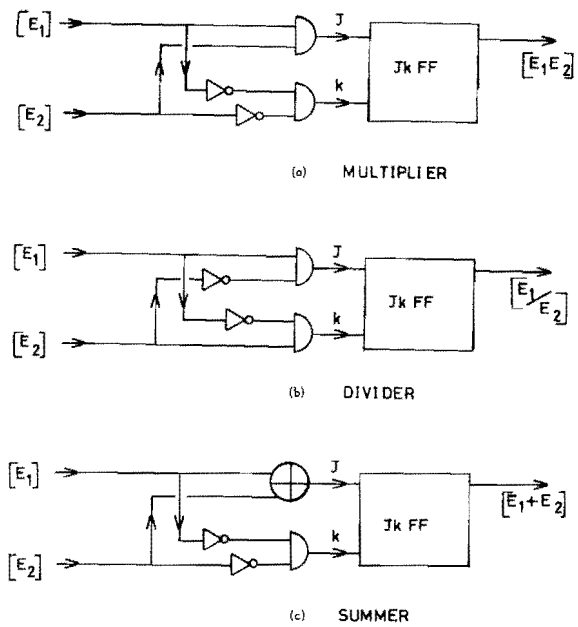

Fig. 29. Computing element in p/q representation.

A multiplier for the quantities on two lines may be formed by putting the two lines directly into an AND gate feeding the J input of a flip-flop, and both inverted into an AND gate feeding the K input of a flip-flop, as shown in Fig. 29*a* (a similar configuration to that of the two-input summing integrator of Figure 126. The output is taken from the side of the flip-flop corresponding to the J input, and its generating probability p_0 is the probability that the flip-flop will be in the corresponding state. The equilibrium equation for the flip-flop is:

$$jq_0 = kp_0 \tag{176}$$

where j is the generating probability of the sequence on the J input and k is that of the sequence on the K input. If p_1 and p_2 are the generating probabilities of the input sequences, than $j = p_1p_2$ and $k = q_1q_2$, so that

$$p_0/q_0 = p_1p_2/(q_1q_2) \tag{177}$$

and hence

$$E_0 = E_1E_2 \tag{178}$$

A divider formed from the same circuit by inverting one of the inputs is shown in Fig. 29*b*.

A two-input, equally-weighted summer may be constructed by replacing the AND gate on the J input with an EXCLUSIVE-OR gate, as shown in Fig. 29c. We now have $j = p_1q_2 + p_2q_1$, so that from Eq. (176)

$$p_0/q_0 = (p_1/q_1) + (p_2/q_2) \tag{179}$$

and hence

$$E_0 = E_1 + E_2 \tag{180}$$

Subtaction is clearly impossible in this representation, since there is no representation of negative quantities.

6.3. Single-Line Representation of Bipolar Quantities with Infinite Range

Unbounded quantities taking both positive and negative values may be represented on a single line using a modified version of representation III. Let the generating probability p of a binary sequence corresponding to an arbitrary real quantity E be defined by

$$E = e(p - q)/(2pq) \tag{181}$$

The inverse relationship is

$$p = [E - e + (E^2 + e^2)^{1/2}]/(2E) \tag{182}$$

so that p takes the value $\frac{1}{2}$ when E is zero, tends to zero as E tends to minus infinity, and tends to unity as E tends to plus infinity.

In this representation the invertor acts to form the negative of the quantity on its input line. Unfortunately, other computing elements are not so readily constructed. For example, forming a reciprocal involves mapping a probability just less than $\frac{1}{2}$ into one near zero, and one just greater than $\frac{1}{2}$ into one near unity. Clearly, an element with considerable memory would be required to form a good approximation to this computation. Despite these defects, this representation is one which enables completely arbitrary unbounded quantities to be represented by a digital sequence on a single line, and for that reason alone merits attention.

6.4. Two-Line Representation of Bipolar Quantities with Infinite Range

Examination of Eq. (181) shows that the representation of quantities of arbitrary sign and magnitude is obtained from a numerator in represen-

tation III giving sign and magnitude information, and a denominator which is always positive and acts as a scaling factor. The defects of the representation emanate from both numerator and denominator being functions of the generating probability on a single line. If two lines are used, then many of the problems may be overcome.

For example, suppose that the numerator is a quantity x in representation III such that $-e \leq x \leq e$, and it is represented on the X line by a binary sequence with generating probability p; and the denominator is a quantity y in representation I such that $0 \leq y \leq 1$, and it is represented on the Y line by a binary sequence with generating probability p'. Then for any real quantity E it is possible to choose p and p' such that

$$E = e(p - q)/p' = X/Y \tag{183}$$

The choice of p and p' is not unique—clearly, $0 \leq p' \leq \min(1, e/|E|)$, and minimum variance will be attained by letting $p' = \min(1, e/|E|)$.

An invertor on the X line changes the sign of quantities in this representation. Multiplication is readily carried out, since, if

$$E_i = x_i/y_i, \qquad \text{then} \qquad E_1E_2 = x_1x_2/y_1y_2 \tag{184}$$

and the numerator is a product in representation III which can be realized by an EXCLUSIVE-NOR gate on lines X_1 and X_2, while the denominator is a product in representation I which can be realized by an AND gate on lines Y_1 and Y_2. If the weighted sum of E_1 and E_2 is to be formed, we have

$$wE_1 + (1 - w)E_2 = [wy_2x_1 + (1 - w)y_1x_2]/(y_1y_2) \tag{185}$$

The denominator is formed by an AND gate on lines Y_1 and Y_2 as before. The numerator is a weighted sum of two quantities, x_1 and x_2, in representation III, and can be formed by the configuration of Fig. 8 (Section 4.6); the weighting probabilities will not be p_3 and $1 - p_3$, but, letting $w = p_3$, they will be $wy_2 = p_3p_2'$ and $(1 - w)y_1 = (1 - p_3)p_1'$. Hence a two more AND gates are required to form the weighting probabilities.

Division remains a difficult operation in this representation, since it requires conversion of the magnitude of a quantity in representation III to a quantity in representation I, and this involves converting, e.g., a stochastic sequence with a generating probability of $\frac{1}{2}$ to a deterministic sequence with a "generating probability" of zero. Clearly, an element with memory, such as an ADDIE, is required.

6.5. A Versatile Two-Line Representation of Arbitrary Quantities*

The problems in division with the representation of Eq. (183) may be removed by making it completely symmetrical and putting both numerator and denominator in the bipolar representation III. If the quantity x in the numerator is defined as before, but the quantity in the denominator y, is now in the range $-1 \leq y \leq 1$ and represented by a binary sequence with generating probability p' in representation III, we have

$$E = x/y = e(p - q)/(p' - q') \tag{186}$$

An invertor on either X or Y lines change the sign of a quantity in this representation. Interchange of X and Y lines gives the reciprocal of the quantity represented. Thus elements for addition may be used to perform subtraction, and elements for multiplication may be used to perform division. Multiplication follows from Eq. (184), as before, except that both multipliers are EXCLUSIVE-NOR gates. Weighted summation follows from Eq. (185), where the products y_2x_1 and y_1x_2 are first formed using EXCLUSIVE-NOR gates as multipliers, and then their weighted sum is formed by the configuration of Fig. 8.

Thus between two and four elements of the complexity of an EXCLUSIVE-NOR gate can, in this representation, perform the operations of addition, subtraction, multiplication, and division. Since the quantities in the computations are completely arbitrary, being bipolar and unbounded, the representation forms the basis of a very versatile, if not "universal," computing system. It has a defect, however, in that for large quantities, as the denominator approaches zero its sign becomes less and less determinate, and hence a result of large magnitude could be obtained with the wrong sign. This phenomenon can occur for quantities of any magnitude, since x and y may be scaled together to any small magnitude without affecting their ratio, but it is clearly of the greatest importance for large quantities, where it is an inherent defect. The defect does not arise in the representation discussed in the previous section because the denominator is then always positive.

An improvement in this representation may be obtained by expressing either the denominator, or both numerator and denominator, in the two-

* The versatility of this representation was first pointed out to the author by John Esch at the Department of Computer Science, University of Illinois, Urbana, who has been studying this representation and its applications in some depth.

line representation II, giving an overall representation on three lines or four lines, respectively. In either case, both change of sign and formation of the reciprocal reduce to appropriate interchange, or simple operations on the lines, and both addition and multiplication are readily performed. If the minimum variance representation is maintained in the denominator (which is always possible in addition and multiplication, since only multiplication of denominators is involved—Section 4.4.2, then the sign of the denominator is well-determined, and hence the sign of a large quantity is not indeterminate.

All the representations discussed in this section, even though they enable computations to be carried out with quantities of unbounded magnitude, are equally applicable to computations using standard analog computing elements. However, since analog multipliers have, to date, been expensive, unreliable elements of limited bandwidth, the fact that even addition in these representations requires a number of multipliers has acted against their use.

6.6. Use of Nonrandom Sequences with Stochastic Computing Elements

It was noted in Section 3 that the essential difference between the sequences of logic levels used in the stochastic computer to represent quantities and those used in other "counting" computers such as the DDA is that the sequences in the former are unpatterned—they are Bernoulli sequences in which successive logic levels are completely unpredictable from those which have occurred before. It is this feature of stochastic sequences which enables multiplication to be performed by a simple AND gate. However, this factor is not as necessary as it may first appear, and it is of interest to investigate further whether there are, e.g., other sequences for which an AND gate acts as a multiplier.

Clearly, if one sequence at the input of an AND gate is a Bernoulli sequence and the other is not, then multiplication of the generating probability of one by the relative proportion of ON logic levels in the other still occurs, since, if the generating probability of the stochastic sequence is p and the number of times the logic level of the other sequence is ON is k during N clock intervals, then the expected number of times that the output of the AND gate will be on is pk, and the relative frequency of ON logic levels at the output is $p(k/N)$.

The output of the AND gate will be a stochastic sequence, but not a Bernoulli sequence, and it will have gained a patterning, or nondelta-

function autocorrelation, from the nonstochastic sequence. The variance of the output function merits careful consideration; it is clearly not a function of the pattern in the nonstochastic sequence, if that is defined as having k out of N logic levels ON. If, however, this sequence is defined only by the mean number of ON logic levels in a sufficiently long sequence, then for any shorter sequence the proportion of ON logic levels may fluctuate widely. A form of sequence which minimizes these fluctuations is that shown in Fig. 2*a*, where ON logic levels are uniformly distributed.

The variance in the proportion of ON logic levels in the output for a nonstochastic sequence of this type with k ON levels in N clock intervals, multiplied by a Bernoulli sequence with generating probability p is the variance of the stochastic sequence over k clock intervals normalized over the whole interval; i.e.,

$$\text{Var} = [p(1-p)/k](k^2/N^2) = pk(1-p)/N^2 \tag{187}$$

The variance of a stochastic sequence with generating probability pk/N is

$$\text{Var} = pk[1-(pk/N)]/N^2 \tag{188}$$

Hence the variance is reduced by a factor of $1 + p(1-k/N)/(1-p)$.

It is interesting to consider whether the variance, which is a major factor limiting the accuracy and bandwidth of the stochastic computer, could be reduced still further by the avoidance of all stochastic sequences and the use of a second nonstochastic sequence. This would clearly have to be "uncorrelated" with the form of nonstochastic sequence so far discussed, and it is easy to see that over an arbitrary, or variable, interval there is no uniformly "uncorrelated" sequence. However, over any given interval containing a fixed number of clock pulses there is an orthogonal sequence —in fact, one of the form shown in Fig. 2*b*, where all the ON logic levels are grouped together at one end of the interval. A sequence of the form shown in Fig. 2*a* will multiply with one of the form shown in Fig. 2*b* in an AND gate in an equivalent manner to that of stochastic sequences. Moreover, it may be shown that the output has no variance, only a quantization error which is the minimum possible.

This phenomenon was first noted in studies of the application of the stochastic computer to automatic radar tracking ([38]), which is essentially a sampled-data problem where computations have to be carried out during prescribed intervals, and nonstochastic orthogonal sequences are possible. It has led to the development of a nonstochastic computer, the "phase computer," ([38,39]) which has some of the characteristics of the stochastic

computer, especially economy of hardware, but attains far greater speed in computation for a given accuracy of result.

The phase computer is interesting because it shows clearly the relationship between the stochastic computer and the DDA, and also because it has a programming section similar to that of a general-purpose machine, which is readily built in because of the all-digital computing elements used. Its main disadvantages over the stochastic computer is that, whereas two Bernoulli sequences in the stochastic computer interact together to give a further Bernoulli sequence which is immediately available for further computation, the two types of sequence in the phase computer interact together to give a third type of sequence, related to neither, which has to be converted back to a standard form using a counter, or "integrator."

Thus stochastic computing techniques are particularly amenable to the construction of *networks* of computing elements in which the output of any one element is in a form suitable for feeding to any other. When such a construction is not required and the data is not already in the form of random pulse sequence alternative nonstochastic computing techniques are advantageous. These are important considerations in determining the range of application of stochastic computing techniques.

7. SEQUENTIAL STOCHASTIC COMPUTING ELEMENTS

The general stochastic computing configuration is a stochastic automaton, and is amenable to theoretical analysis using the theory of Markov chains, etc. Often the system will be ergodic, and its stationary hyperstate is fairly easily determined by writing down the corresponding matrix equation. Solution of this equation may not be trivial, and cannot necessarily be thrashed out numerically using a digital computer, because of the vast dimension of the matrix involved. Low-order statistics of the stationary hyperstate may be calculated fairly readily by summation appropriate to the scalar product required, but manipulation of the entire matrix is generally impossible.

For example, the two-integrator configuration of Fig. 14*c*, with two ten-bit integrators, has over 10^6 possible states. From the transition equations for these states, which can be written in a reasonably closed form, since the two integrators may be separated and the transition probabilities are linear functions of the states, it is readily established that the final hyperstate is such that the expected generating probability of the output is

equal to that of the input, i.e., the steady-state solution is that determined from Eq. (112). Further than this, however, the analysis becomes more difficult, and proof that the transient behavior of the system is a damped sinusoid is not yet available.

There is one special form of stochastic automaton which is amenable to theoretical analysis, and fortunately this is one that is important in practice. An automaton with a *chain structure*, such that its states are completely ordered and transitions can only occur from one state to itself, or to adjacent states, has particularly simple equations for its stationary states because, in simple terms, it cannot jump over states in the chain, and in going from one state to another it must pass through all states in between. The importance of this structure in stochastic computing may also be stated in simple terms: since the variables in the stochastic computer are probabilities and the sequences which represent them fluctuate with random variance, the sequential elements of the computer are always required to have some "inertia," and not to jump from one state to another which is remote from it—state transitions are required to be local, and the chain-structured automata embodies this restriction in its strictest dynamic form. Since the ADDIE (Section 4.9) is the prime example of a chain-structured automaton, they have been given the generic name of *ADDIEs* and the particular one described earlier is called a *linear ADDIE.*

In the following sections the behavior of forms of the ADDIE are analyzed, and, in particular, both the steady-state and transient behavior of the linear ADDIE are treated in some detail. Finally, the optimality of the linear ADDIE as probability estimator is discussed.

7.1. Transition Equations for the Generalized ADDIE

The ADDIE is a chain-structured automaton with an ordered set of states, $S_0, S_1, \ldots, S_N$, and a single binary input $A = 0, 1$ and generating probability p (it is convenient to take $q = 1 - p$) whose transition probabilities are constrained so that when automaton is in state S_1, and $A = 0$, it can either remain in state S_i or move to state S_{i-1} (provided $i > 0$). When it is in S_i and $A = 1$, it can either remain in S_i or move to state S_{i+1} (provided $i < N$).

The matrices of transition probabilities for the ADDIE have a very simple structure, consisting of the main diagonal plus one adjacent diagonal —the one above if the input is ON, and the one below if the input is OFF. It is convenient to label these off-diagonal elements in a different way from the general form adopted in Section 5.3. Let the probability that, when the

automaton is in state S_i and the input is ON, its next state will be S_{i+1} be Q:

$$\pi_i{}^1{}_{i+1} = Q \tag{189}$$

Let the probability that, when the automaton is in state S_i and the input is OFF, its next state will be S_{i-1} be P_i:

$$\pi_i{}^0{}_{i-1} = P_i \tag{190}$$

The probability that it will remain in the same state when its input is ON is clearly $1 - Q_i$, and when its input is OFF this probability is $1 - P_i$; all other elements of the transition matrix are zero.

There are some further constraints on the transition probabilities. At the ends of the range we must have

$$P_0 = Q_N = 0 \tag{191}$$

To avoid trivial and nonergodic situations, in which the chain structure has become disconnected, we shall assume

$$P_i,\ Q_i \neq 0 \qquad 0 < i < N \tag{192}$$

It will also be true in general that

$$P_{i+1} > P_i, \qquad Q_{i+1} < Q_i \tag{193}$$

i.e., that forward transitions are rarer at the upper end of the chain and backward transitions are rarer at the lower end; this is the real basis for the use of the ADDIEs as estimators. A state diagram of a generalized ADDIE with these constraints is shown in Fig. 30.

The transition equation for the hyperstate of the generalized ADDIE may be written in terms of the P_i and Q_i. Let the probability that the ADDIE is in the ith state at time T be $\pi_i(T)$. Then

$$\begin{aligned}\pi_i(T+1) = {} & pQ_{i-1}\pi_{i-1}(T) + qP_{i+1}\pi_{i+1}(T) \\ & + p(1 - Q_i)\pi_i(T) + q(1 - P_i)\pi_i(T)\end{aligned} \tag{194}$$

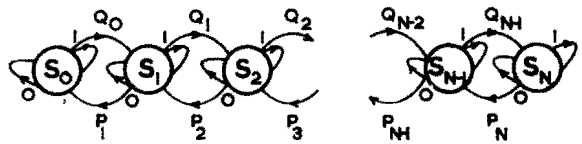

Fig. 30. State diagram of generalized ADDIE.

This is the fundamental equation for the transient behavior of the ADDIE, (the boundary conditions are satisfied if undefined terms are set to zero).

The behavior of the ADDIE is clearly ergodic [from inequalities (192)] and the single stationary hyperstate π_i may be determined by the above equation. However, this may be simplified either directly from the equation or by considering the equilibrium between two adjacent states. Under stationary conditions the probability of a transition from one state to another up the chain must equal the reverse probability down the chain. Hence

$$pQ_i\pi_i = qP_{i+1}\pi_{i+1} \qquad 0 \leq i < N \tag{195}$$

This is the fundamental equation for the stationary behavior of the ADDIE.

One may solve this set of equations for π_i; let

$$\begin{aligned}\gamma_i &= p^i q^{N-i}(i/P_i) \prod_{r=1}^{i-1} \frac{Q_r}{P_r} \qquad 0 < i \leq N \\ &= 1 \qquad\qquad\qquad\qquad\quad i = 0\end{aligned} \tag{196}$$

and let

$$\phi = \sum_{i=0}^{N} \gamma_i \tag{197}$$

then

$$\pi_i = \gamma_i/\phi \tag{198}$$

This is the solution for the stationary hyperstate in the general case. In the case of the linear ADDIE we have

$$P_i = 1 - Q_i = i/N \tag{199}$$

so that

$$\gamma_i = p^i q^{N-i} N!/[i!(N-i)!] \tag{200}$$

which are the terms, ${}^N C_i p^i q^{N-i}$, of the binomial expansion for $(p+q)^N$. Hence $\phi = 1$, and we may write

$$\pi_i = {}^N C_i p^i q^{N-i} \tag{201}$$

so that the hyperstate has a binomial distribution.

7.2. Output from the Generalized ADDIE

Since the ADDIE is intended as a computing and estimating element, it will have a numerical value, or output θ_i, when it is in its ith state. For most practical purposes we are less interested in the actual hyperstates and their behavior than in the expected output and other statistics of the output,

which are scalar functions of the hyperstates

$$\theta(T) = \sum_{i=0}^{N} \pi_i(T)\theta_i \tag{202}$$

the expected output at time T, and

$$\theta = \sum_{i=0}^{N} \pi_i \theta_i \tag{203}$$

the expected output in the steady state; similarly for the variances of the outputs, $\sigma^2(T)$, σ^2, e.g.,

$$\sigma^2 = \sum_{i=0}^{N} \pi_i \theta_i^2 - \theta^2 \tag{204}$$

These statistics depend on θ_i and on π_i, which is a function of all the P_i, Q_i, and of p. The θ_i is an arbitrary function of the ADDIE's state and its statistics depend both upon the transition probabilities and the input—the meaning of the expected output is thus dependent on establishing accurately all the output values and the transition probabilities—it is "openloop." However, Eq. (195) makes it feasible that with a suitable relationship between P_i and Q_i the expected forward transition probability will have p as a sufficient statistic. Thus if we put

$$\theta_i = P_i \tag{205}$$

the situation is now very much "closed-loop"—variations in P_i affect both output and transitions; it remains to show that cancellation occurs: From (195)

$$\sum_{i=0}^{N-1} pQ_i\pi_i = \sum_{i=0}^{N-1} qP_{i-1}\pi_{i+1} \tag{206}$$

Using (191),

$$p \sum_{i=0}^{N} Q_i\pi_i = q \sum_{i=0}^{N} p_i\pi_i \tag{207}$$

Substituting from Eqs. (205) and (203),

$$p \sum_{i=0}^{N} Q_i\pi_i = q\theta \tag{208}$$

The only relationship between Q_i and P_i which can ensure that p is a sufficient statistic of θ is a linear one, say

$$aP_i + bQ_i = 1 \tag{209}$$

Then

$$\theta = p/(ap + bq) \tag{210}$$

Thus if the output of the ADDIE is taken to have the same value as the backward transition probability (or, equally, some linear function of that probability) and the forward and backward transition probabilities are linearly related, then the generating probability of the input is sufficient to completely determine the expected output, no matter what the actual values of the transition probabilities. This is a very important practical result, since it implies that close control over the actual transition probabilities is not necessary in an ADDIE acting as a weight in a learning machine. It is easy to establish a linear relationship between P_i and Q_i by taking the line generating Q_i to be the inverse of the line generating P_i. We then have

$$P_i + Q_i = 1 \tag{211}$$

and the expected output is

$$\theta = p \tag{212}$$

Thus any device with a number of states, having a means of incrementing and decrementing to adjacent states, and a probabilistic output which varies from state to state, preferably monotonically, can be made to act as an estimator, or a *memory*, for holding the probability on its input line. The variance of the estimate and the time taken to attain it will obviously vary with the actual transition probabilities, but because of the inherent feedback the estimate itself is independent of them.

7.3. Behavior of the Linear ADDIE

The general solution for the motion of the hyperstate of a linear ADDIE has no simple closed form. However, the behavior of statistics of the hyperstate and the form of the terminal, or stationary hyperstate, are readily determined. In calculating the behavior of the ADDIE under increments and decrements (input line ON or OFF) it is worth noting that the ADDIE is self-dual; if

Increment	$\rightleftarrows$	Decrement	$(A \rightleftarrows \bar{A})$
p	$\rightleftarrows$	q	
P	$\rightleftarrows$	Q	
i	$\rightleftarrows$	$N - i$	

then none of the ADDIE equations, such as Eq. (194), are changed. Hence any result obtained for an increment may be dualized for a decrement and *vice versa*.

7.3.1. Change in Expected Output

The change in the expected output θ^+ after a single increment may be calculated from Eq. (194), with $p = 1$

$$\begin{aligned}\pi_i(T+1) &= P_i\pi_i(T) + Q_{i-1}\pi_{i-1}(T) \\ &= (I/N)\pi_i(T) + [(N-i+1)/N]\pi_{i-1}(T)\end{aligned} \tag{213}$$

for $0 < i \leq N$. Hence from Eq. (202)

$$\begin{aligned}\theta^+(T+1) &= \sum_{i=0}^{N} (i/N)\pi_i(T+1) \\ &= \sum_{i=0}^{N} (i^2/N^2)\pi_i(T) + \sum_{i=0}^{N} (i+1)(N-1)/(N^2)\pi_i(T) \\ &= (1-1/N)\theta(T) + 1/N\end{aligned} \tag{214}$$

The change in expected output θ^- after a single decrement may be obtained by dualizing this equation:

$$1 - \theta^-(T+1) = [1-(1/N)][1-\theta(T)] + (1/N) \tag{215}$$

so that:

$$\theta^-(T+1) = [1-(1/N)]\theta(T) \tag{216}$$

Equations (214) and (216) show that the expected output of a linear ADDIE is a sufficient statistic of its expected output following an increment or decrement.

7.3.2. Change in Variance of Expected Output

The change in the expected variance σ^{2+} after an increment may be obtained by substituting from Eq. (213) into (204):

$$\begin{aligned}\sigma^{2+}(T+1) &= \sum_{i=0}^{N} (i/N)^2\pi_i(T+1) - \theta^2(T+1) \\ &= (1/N^3)\sum_{i=0}^{N} [(N-2)i^2 + (2N-1)i + N]\pi_i(T) \\ &= -[1-(1/N)]^2\theta^2(T) \\ &= [(N-2)\sigma^2(T)/N] + \{\theta(T)[1-\theta(T)]\}/N^2\end{aligned} \tag{217}$$

This result is symmetrical, and dualizing gives the same equation, so that the change in variance is the same for an increment or decrement. Similar results may be obtained for the third and higher moments, and in general it may be shown that the moments up to order K of a linear ADDIE are sufficient statistics for all future moments up to order K, given the intervening input sequence.

7.3.3. Expected Number of Increments to Cause a Transition

In the state S_i an increment has a probability, $Q_i = \theta_i$ of causing a transition to the state S_{i+1}, and hence the expected number of increments needed to cause a transition is $1/\theta_i$.

Hence for a linear ADDIE the expected number of increments from the state S_i to the state S_j is

$$\begin{aligned}\tau^+_{i,j} &= \sum_{r=i}^{j-1} 1/\theta_r = \sum_{r=i}^{j-1} N/(N-r) \\ &\approx N\ln[(N-i)/(N-j)]\end{aligned} \tag{218}$$

7.3.4. Variance of the Stationary Hyperstate of a Linear ADDIE

The stationary distribution of the linear ADDIE has already been shown to be a binomial distribution [Eq. (201)] and the expected output is equal to the generating probability of the input [Eq. (212)]. The variance may be obtained either from the properties of the binomial distribution or by manipulating the equilibrium equation for the linear ADDIE. From Eq. (195) this is, for $0 \leq i < N$

$$p(N-i)\pi_i = q(i+1)\pi_{i+1} \tag{219}$$

Multiplying this equation by i/N^2 and summing from $i = 0$ to $N-1$, changing limits

$$p\sum_{i=0}^{N} (i/N)\pi_i - p\sum_{i=0}^{N} (i/N)^2\pi_i = q\sum_{i=0}^{N} (i-1)i\pi_i/N^2 \tag{220}$$

so that

$$\sigma^2 = \theta(1-\theta)/N = pq/N \tag{221}$$

The variance of the input sequence with generating probability p is equal to pq. Thus Linear ADDIE variance = (Input variance)/N, and the ratio $\varrho(p)$,

$$\varrho(p) = \frac{\text{Estimation variance}}{\text{Input variance}} = \frac{\sigma^2}{pq} \tag{222}$$

is a performance measure for the ADDIE as an estimator; it gives a measure of the accuracy to be expected in the estimate of the generating probability of the input sequence. It is not the only performance measure, and in practice must be weighed against response time, which generally goes in the opposite direction. It is a function of the generating probability itself, and an overall performance measure of this type can only be obtained when the distribution of possible inputs is known. If this distribution is $\mu(p)$ [i.e., the probability of the input being between p and $p + dp$ in its generating probability is $\mu(p)\,dp$], then one may define the overall performance measure of accuracy in estimation as

$$\varrho = \int_{p=0}^{1} \varrho(p)\,d\mu(p) \tag{223}$$

We have from Eq. (219)

$$\varrho(p\text{:linear ADDIE}) = 1/N \tag{224}$$

In the following section various performance measures for the linear ADDIE are compared with those of a nonstochastic, continuous estimation process, and in the final section the optimality of the linear ADDIE, in terms of the performance measure of Eq. (223) is analyzed.

7.4. Comparison of Linear ADDIE with Continuous Estimation Process

Equations (214) and (216), taken together, have a remarkable resemblance to the well-known exponential smoothing, or stochastic approximation, technique for measuring means and trends in random variables. If the input sequence to an ADDIE $A(T)$ is regarded as a random variable, and its mean $M(N-1)$ is computed over $N-1$ clock periods, then this is an unbiased estimator of the generating probability p. We have

$$M(N-1) = \sum_{i=1}^{N-1} A(i)/(N-1) \tag{225}$$

and

$$\text{Exp}[M(N-1)] = p \tag{226}$$

Now suppose that the mean is calculated over N clock periods as $M(N)$ and express this in terms of $M(N-1)$:

$$M(N) = \sum_{i=1}^{N} A(i)/N = [(N-1)/N]M(N-1) + [A(N)/N] \tag{227}$$

This gives a new estimate of the generating probability of the sequence $A(i)$ in terms of the previous best estimate and the latest value in the sequence. Suppose now that instead of increasing N with each new value that comes in we assume that our previous estimate was based on the $N-1$ values, and generate the next estimate using Eq. (227). This gives us an "exponential average" rather than a true moving average, and has the advantage that our averaging period does not increase with time (which would be a disadvantage if p was varying with time) and that we do not have to remember the N past values of $A(i)$ as we do with a moving average. If the value of the exponential average at time T is denoted by $E(T)$, we have from Eq. (227)

$$E(T+1) = [1 - (1/N)]E(T) + [A(T)/N] \tag{228}$$

Comparison with Eqs. (214) and (216) shows that the expected output of an ADDIE follows the same equation as an exponential average of the input sequence. A heuristic argument explaining this relationship is interesting because it bears out the original assumptions behind stochastic computing given in Section 3.3. Consider an ADDIE in its ith state, so that the value of its output is i/N, which receives an increment, and compare it with the analog weighting process of Eq. (228) with $E(T)$ also equal to i/N. The change in $E(T)$ is

$$E(T+1) - E(T) = (N-i)/N^2 \tag{229}$$

On the other hand, the probability that the ADDIE will change to state S_{i+1} and the value of its output will increase by $1/N$ is $Q_i = 1 - (i/N) = (N-i)/N$. Hence the *expected* change in value of the ADDIE is equal to the actual change in value of the equivalent analog weight. This is the advantage gained through stochastic computing: analog variables may be replaced by digital ones. A comparison of other performance criteria for the two processes indicates what has been lost by this replacement.

7.4.1. Expected Number of Increments for a Transition

Writing out Eq. (228) for $E(T)$, $E(T-1)$, etc., and substituting one in another, we have

$$\begin{aligned} E(T+1) &= (1/N)\{1 + [1 - (1/N)] + [1 - (1/N)]^2 + [1 - (1/N)]^{T-1}\} \\ &\quad + [1 - (1/N)]^T E(1) \\ &= 1 - [1 - (1/N)]^T [1 - E(1)] \end{aligned} \tag{230}$$

Thus if T increments are required to go from $E(1) = i/N$ to $E(T+1) = j/N$, we have

$$[1 - (1/N)]^T = [1 - (j/N)]/[1 - (i/N)] \tag{231}$$

so that

$$T \ln[1 - (1/N)] = \ln[(N - j)/(N - i)] \tag{232}$$

and, using the approximation that $\log(1 - x) \approx x$ for small x, we have:

$$T \approx N \ln[(N - i)/(N - j)] \tag{233}$$

This is identical to Eq. (218) for an ADDIE.

7.4.2. Variance of Estimate

If we square both sides of Eq. (228) and take expected values, we have

$$\begin{aligned}\text{Exp}(E^2) &= \left[1 - \frac{1}{N}\right]^2 \text{Exp}(E^2) + \frac{2[1 - (1/N)]\,\text{Exp}(A)\,\text{Exp}(E)}{N} + \frac{\text{Exp}(A^2)}{N^2} \\ &= \left[1 - \frac{1}{N}\right]^2 \text{Exp}(E^2) + \frac{2[1 - (1/N)]p^2}{N} + \frac{p}{N^2}\end{aligned} \tag{234}$$

Hence

$$\text{Var}(E) = pq/(2N - 1) \tag{235}$$

This may be compared with Eq. (221) for an ADDIE, from which it will be noted that

$$\frac{\varrho(p\text{:linear ADDIE})}{\varrho(p\text{:exponential smoothing})} = 2 - (1/N) \tag{236}$$

Hence exponential smoothing with the same time constant gives an estimate with only half the variance of that for the ADDIE. Or, in alternative terms, to attain the same accuracy with an ADDIE as with exponential smoothing, the risetime of response has to be $\sqrt{2}$ times as long. The extra contribution to the variance may be viewed as the internal noise of the stochastic processes in the ADDIE itself.

7.5. Sub-ADDIEs

One variation on the ADDIE which is of interest is the truncated ADDIE, or "sub-ADDIE," formed by setting P_j and Q_k of another ADDIE to zero, where $0 \leq j < k \leq N$. Since the states of the sub-ADDIE can only range between S_j and S_k, states outside of this range may be omitted

to give a (j, k) sub-ADDIE of the parent ADDIE. Rather than renumber the states from 0 to $k - j$ [the sub-ADDIE has $(k - j + 1)$ states], it is convenient to call the state of the sub-ADDIE corresponding to the ith state of the parent its ith *relative state*, and label it S_i'. Much of the behavior of a sub-ADDIE can be expressed simply in terms of that of its parent.

7.5.1. Stationary Hyperstate of a Sub-ADDIE

Let the stationary hyperstate for the relative states of a (j, k) sub-ADDIE of an $(N - 1)$ state ADDIE be π_i'. We have

$$\sum_j^k \pi_i' = 1 \tag{237}$$

and for stationary equilibrium, from Eq. (195), using the definition of a sub-ADDIE

$$pQ_i\pi_2' = qP_{i+1}'\pi_{i+1} \qquad j < i < k - 1 \tag{238}$$

$$p\pi_j' = qP_{j+1}\pi_{j+1}' \tag{239}$$

$$pQ_{k-1}\pi_{k-1}' = q\pi_k' \tag{240}$$

the original π_i [Eq. (198)] satisfies the first equation, and hence we define

$$u_i = \pi_i \qquad j < i < k \tag{241}$$

$$u_j = Q_j\pi_j \tag{242}$$

$$u_k = P_k\pi_k \tag{243}$$

Let

$$U = \sum_j^k u_i \tag{244}$$

Then

$$\pi_i' = u_i/U \qquad j \leq i \leq k \tag{245}$$

is the stationary hyperstate of the sub-ADDIE expressed in terms of that of the parent ADDIE.

7.5.2. Variance of the Stationary Hyperstate of a Linear Sub-ADDIE

To obtain the variance of the stationary hyperstate of a sub-ADDIE of a linear ADDIE, it is convenient to use Eqs. (238)–(240), substituting

for Q_i, etc., multiplying the first equation by i/N, the second by j/N, and the third by k/N. Adding all three and summing over i, we have

$$d(j/N)\pi_j' + p\sum_{i=j+1}^{k-1} i(N-i)\pi_i'/N^2 = q[(k-1)/N]\pi_k' + q\sum_{i=j}^{k-2} i(i+1)\pi_i'/N^2 \tag{246}$$

Hence the variance is

$$\sigma^2 = (pq/N) + p(J/N)\pi_j' + q[(N-k)/N]\pi_k' \tag{247}$$

This may be compared with the variance for the parent ADDIE, which is pq/N, and the variance for the ADDIE with the same number of states, which is $pq/(k-j)$. It can be seen that, provided the probability of the sub-ADDIE being in either of its end states is mall, the variance of the hyperstate of the sub-ADDIE is less than that of a complete ADDIE with the same number of states. This approximate result is given a more rigorous basis in the following section.

7.6. Optimality of the Linear ADDIE

The extent to which a linear ADDIE is a minimum variance estimator, in terms of Eq. (221), may be determined by considering the effect of small perturbations on its transition probabilities. Consider the $(N+1)$ state linear ADDIE which is almost linear, in that its kth transition probability has been slightly changed; let

$$\begin{aligned} P_i &= i/N && 0 \le i < k \\ &= k(1+e)/N && i = k \\ &= i/N && k < i \le N \end{aligned} \tag{248}$$

where e is small, so that terms which are $O(e)$ may be neglected.

The stationary hyperstate for this ADDIE satisfies Eq. (195), and this is satisfied for i above and below k by

$$\pi_i :: p^i q^{N-i}\, {}^N C_i = B_i, \quad \text{say} \tag{249}$$

Hence it is reasonable to put

$$\begin{aligned} \pi_i &= B_i(1+ae) && 0 \le i < k \\ &= B_i(1+be) && i = k \\ &= B_i(1+ce) && k < i \le N \end{aligned} \tag{250}$$

Substituting these in Eq. (195) we have, for $i = k-1$,

$$p(N-k+1)(1+ae)B_{k-1} = qk(1+e)(1+be)B_k \tag{251}$$

so that

$$a = 1 + b \tag{252}$$

Similarly, for $i = k$, we have

$$p[N-k(1+e)](1+be)B_k = q(N-k-1)(1+ce)B_{k+1} \tag{253}$$

Hence

$$c = b - [k/(N-k)] \tag{254}$$

Thus in terms of b, we have

$$\begin{aligned} \pi_i &= B_i(1+be+e) && 0 \le i < k \\ &= B_k(1+be) && i = k \\ &= B_i[(1+be-ke)/(N-k)] && k < i \le N \end{aligned} \tag{255}$$

Summing these equations for all i, we have

$$1 = \sum_{i=0}^{N} \pi_i = (1+be)\sum_{i=0}^{N} B_i + e\sum_{i=0}^{k-1} B_i - [ke/(N-k)]\sum_{i=k+1}^{N} B_i \tag{256}$$

Hence, making use of the fact that the binomial terms sum to unity,

$$b = [k/(N-k)] + [(B_k - N)/(N-k)]\sum_{i=0}^{k} B_i \tag{257}$$

To determine the variance of the stationary hyperstate, we sum $\pi_i p_i^2$ over all i:

$$(\sigma^2 + p^2) = (1+be)\sum_{i=0}^{N} (i/N)^2 B_i + e\sum_{i=0}^{k-1} (i/N)^2 B_i + 2e(k/N)^2 B_k - [ke/(N-k)]\sum_{i=k+1}^{N} (i/N)^2 B_i \tag{258}$$

Substituting for b and for the sum of binomial second moments, we have

$$\sigma^2 - (pq/N) = e[p^2 + (pq/N)][(B_k - N)/(N-k)]\sum_{i=0}^{k} B_i + [eN/(N-k)]\sum_{i=0}^{k} (i/N)^2 B_i + e(k/N)^2 B_k \tag{259}$$

This term can be simplified by using the properties of the partial

moments of the binomial distribution. It is readily shown that

$$\sum_{i=0}^{k} (i/N)^2 B_i = [p^2 + (pq/N)] \sum_{i=0}^{k-1} B_i - [pq(N-1)kB_k/N^2] + p(k/N)^2 B_k \tag{260}$$

Substituting this in Eq. (259), we have

$$\begin{aligned} \sigma^2 - \frac{pq}{N} &= -\frac{e[p^2 + (pq/N)]kB_k}{N-k} - \frac{epq(N-1)kB_k}{N-k} \\ &\quad + \frac{epk^2 B_k}{N-k} + \frac{ek^2 B_k}{N^2} \\ &= eB_k \frac{k}{N}\left[\frac{k}{N} - p\right] \end{aligned} \tag{261}$$

Thus the effect of a perturbation is to change the variance of the stationary hyperstate by an amount proportional to: the perturbation itself; the expectation of being in the perturbed state; and the difference between the output in the perturbed state and the generating probability of the input. It is this last term which is critical, since it determines the sign of the change. The variance will be reduced if outputs with values above the generating

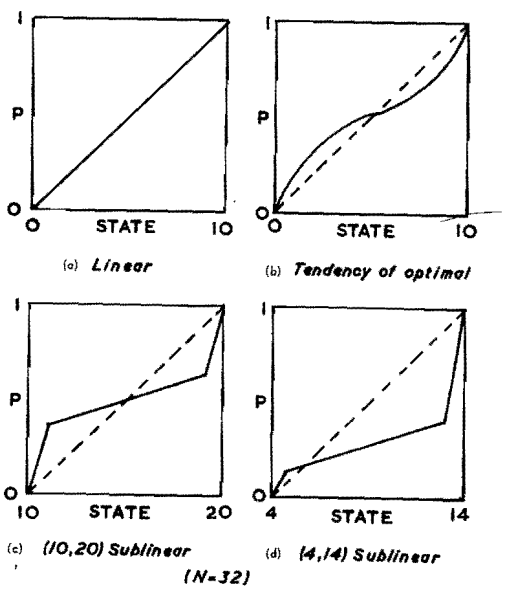

Fig. 31. Variation of output probability with state for various ADDIEs.

probability p are reduced and those below it are increased. A graph roughly indicating this form of perturbation is shown in Fig. 31*b* for $p = 0.5$. Graphs of the state versus output for various linear sub-ADDIEs are also shown to indicate their better approach to optimality.

Optimality as a Function of the Input Distribution

The results of the previous section apply only to the optimality of an ADDIE for a given generating probability, or to a small range of probabilities. In general, it is the mean performance measure of the ADDIE over its range of possible inputs which is to be optimized, and Eq. (261) must be substituted in Eq. (223) to obtain an overall performance measure:

$$\begin{aligned} \varrho &= \int_0^1 (\sigma^2/pq)\mu(p)\,dp = \int_0^1 \{(1/N) + eB_k[(k/N) - p]/pq\}\mu(p)\,dp \\ &= (1/N) + e(k/N)^N C_k \int_0^1 p^{k-1}q^{N-k-1}[(k/N) - p)]\mu(p)\,dp \end{aligned} \tag{262}$$

Thus for the linear ADDIE to be optimal for the input distribution $\mu(p)$ it is necessary and sufficient the last integral vanish for all k. Integrating this integral by parts, we have:

$$\varrho = (1/N) + e(k/N)^N C_k \int_0^1 p^k q^{N-k}(d\mu/dp)\,dp \tag{263}$$

The integral will be indentically zero for $d\mu/dp = 0$, in which case $\mu(p) = 1$, a uniform distribution.

Thus the linear ADDIE is optimal, in that it minimizes the mean expected reduction of variance for a uniform distribution of input generating probabilities. It will also be noted from Eq. (261) that the changes in linear ADDIE outputs required for input probabilities above and below the output are in opposite directions. Thus the linear ADDIE cannot be improved for all inputs simultaneously, and hence it is an *admissable* estimator when the input distribution is unconstrained.

8. ADAPTIVE THRESHOLD LOGIC

One of the most attractive forms of "learning module" for batch fabrication at low cost is the adaptive-threshold-logic element (ATLE) studied by many workers as a pattern classifier, and given the various names of Perceptron ([71]), Adaline ([72]), Learning Matrix ([73]), and so on ([74]). This device is simple enough to allow detailed analysis of convergence under

certain conditions, but it is also powerful enough to be utilized as the basic building block of complex, hierarchical control systems.

In this section the structure and analysis of convergence for a standard form of adaptive threshold element are first described, and then the problems encountered in utilizing a digital version of the element are outlined; these appear as lack of convergence under certain conditions, even when a solution is within the range of the element. Finally, the equivalent stochastic computing element is analyzed to demonstrate that convergence will always take place when a solution is available, and hence that the stochastic ATLE is more economical than a deterministic ATLE using the same components and with the same capabilities.

8.1. Convergence of Adaptive Threshold Logic Elements

The basic problem given an ATLE is to learn to classify two subsets of a set of input patterns into opposite classes. In the learning phase it is given a sequence of patterns together with information about the class to which they belong. In the classification phase it is given only the patterns, and itself has to classify them.

The input patterns to the ATLE are represented as binary n-vectors, of the form $Y_i = \pm 1$, for $1 \leq i \leq n$ (there are alternative forms of representation, but the end result, and theory of operation, are similar). Its internal classification procedure is based on the use of a variable weight vector W_i and a fixed threshold $\theta > 0$. The output of the ATLE is a ternary level, $Z = -1, 0, +1$, where -1 and $+1$ represent assignment of the input pattern to each of the two possible classes, and 0 represents inability to make a classification. The value of Z is determined within the ATLE by the rules. Let

$$S = \sum_{i=1}^{n} W_i Y_i \tag{264}$$

then

$$\begin{aligned} Z &= -1 \quad \text{if} \quad S < -\theta \\ &= 0 \quad \text{if} \quad -\theta \leq S \leq \theta \\ &= 1 \quad \text{if} \quad \theta < S \end{aligned} \tag{265}$$

This defines the "threshold logic" action of the ATLE—the discrete variable Z is a logical function of the binary variables, Y_i, and yet it is obtained by them partly through continuous, arithmetic operations. The "adaptive" nature of the element comes through procedures for adjusting the values of the W_i during the learning phase.

Let the input sequence to the ATLE during the learning phase be $Y(T)$, and let the correct assignment of $Y(T)$ to one of the two classes, labelled ± 1, be $U(Y(T))$. The weights of the ATLE are changed after each input according to the rules:

$$W_i(T+1) = \begin{cases} W_i(T) & \text{if } Z(T) = U(Y(T)) \\ W_i(T) + Y_i(T)U(Y(T)) & \text{if } Z(T) \neq U(Y(T)) \end{cases} \tag{266}$$

This is an error-correcting procedure which changes $W_i(T)$ only when the ATLE makes a wrong assignment, and in such a way as might reduce the difference between $Z(T)$ and $U(Y(T))$.

The analysis of this process is simplified if the input vectors are redefined to include their classification; for any input vector, Y_i, let:

$$X_i = U(Y)Y_i \tag{267}$$

The weight vector, W_i, now gives the correct assignment if:

$$S' = \sum_{i=1}^{n} W_i X_i > \theta \tag{268}$$

And the error-correcting procedure becomes:

$$W_i(T+1) = \begin{cases} W_i(T) & \text{if } S'(T) > \theta \\ W_i(T) + X_i(T) & \text{if } S'(T) \leq \theta \end{cases} \tag{269}$$

Not all assignments of input patterns to arbitrary classes can be realized by an ATLE—it is clear from Eq. (268) that Y_i and $-Y_i$ cannot both be assigned to the same class—and whenever the assignment can be realized by threshold logic, the two classes of input patterns are said to be *linearly separable*. Clearly the ATLE cannot converge to an error-free classifier of two sets of inputs if they are not linearly separable. It may be shown, however that the procedure of Eq. (268) always leads to a solution, if one is possible, in a finite time.

8.2. Novikoff's Proof of Convergence

Suppose that the input sequence to the ATLE frequently contains all those input patterns to be classified. For each input pattern, either the weights will be changed or they will retain their previous value. Consider the input sequence with the patterns which do not cause a change removed—let this sequence be $X_i(N)$; since it consists of patterns which the ATLE

misclassifies, it is sufficient to show that this sequence is bounded in number. For every member of this sequence we have

$$\sum_{i=1}^{n} W_i(N)X_i(N) \leq \theta \tag{270}$$

We also have that

$$W_i(N+1) = W_i(N) + X_i(N) \tag{271}$$

Since the classification is assumed to be realizable by an ATLE, there exists at least one set of weights, W_i say, which is a solution, so that for each N

$$\sum_{i=1}^{n} W_i X_i(N) > \theta \tag{272}$$

We may suppose, without loss of generality, that $W_i(0) = 0$. Multiplying Eq. (271) through by W_i, summing for $1 \leq i \leq n$, and summing all the equations for $0 \leq N < M$, we have

$$\sum_{i=1}^{n} W_i(M)W_i = \sum_{N=0}^{M-1} \sum_{i=0}^{n} W_i X_i \tag{273}$$

Using the Schwarz inequality on the left-hand side, and inequality (272) on the right-hand side, we have

$$\sum_{i=1}^{n} W_i^2(M) > M^2\theta^2 / \sum_{i=1}^{n} W_i^2 \tag{274}$$

Squaring both sides of Eq. (271) and again summing for $1 \leq i \leq n$, for we have, $0 \leq N < M$,

$$\begin{aligned} \sum_{i=1}^{n} W_i^2(M) &= \sum_{N=0}^{M-1} \sum_{i=1}^{n} [2W_i(N)X_i(N) + X_i(N)X_i(N)] \\ &\leq 2M\theta + Mn \end{aligned} \tag{275}$$

using inequality (270).

Inequalities (274) and (275) together imply that

$$M < (2\theta + n)\left(\sum_{i=1}^{n} W_i^2 \right) / \theta^2 \tag{276}$$

Hence M is bounded above and the sequence of misclassified inputs must always terminate. This is the standard convergence proof for ATLEs, first given by Novikoff ([75]).

8.3. Lack of Convergence if Weights Are Bounded

The weight-adjustment procedure of Eqs. (266) and (271) is interesting because, if the initial values of the weights are integers, then so are all future values. Hence the ATLE may be realized with digital elements and the weights, since they are changed only by ± 1, may be represented as the counts in binary counters. The original ATLEs used analog weights, such as motorized potentiometers, electrochemical cells, and transfluxor stores, but these are bulky, expensive, and unreliable, and it is very attractive to replace them with all-digital elements fabricated by large-scale integration.

The use of digital elements makes manifest a defect in the weight-adjustment procedures discussed above. The counts in digital counters can only take a finite number of values, and hence only a restricted set of weight values are possible for any particular size of counter. Since the number of possible dichotomous classifications of a set of binary n-vectors is finite, there will clearly be some size of counter with which all linearly seperable classifications are realizable. In addition, inequalities (275) and (276), taken together, give a bound on the magnitude of the weights necessary for convergence. We may take θ to be near unity, and obtain

$$W_{\max} \text{ (for convergence)} < n^{3/2} W_{\max} \text{ (for separability)} \tag{277}$$

This bound may be improved, but its main implication is that the range of weight values necessary for linear separability may be rather less than that necessary for the adaptive procedure to attain a solution.

Since the range of weights is a function of the storage of the counters, in bits, this raises the question as to whether the capacity for adaptation of an ATLE requires storage elements which are unnecessary to the implementation of a solution once it is found. Inequality (277) does not prove that this is so, but a simple example shows that lack of convergence can occur even when a solution is within the range of weights.

The set of binary 4-vectors: $A = (1, 1, 1, -1)$, $B = (1, -1, -1, 1)$, $C = (-1, 1, -1, 1)$, and $D = (-1, -1, 1, 1)$ may be discriminated from the opposite set, $-A$, $-B$, $-C$, $-D$, by the weight vector: $W = (1, 1, 1, 2)$, since $W \cdot A = W \cdot B = W \cdot C = W \cdot D = +1 > \theta = \frac{1}{2}$ and $W \cdot (-A) = W \times (-B) = W \cdot (-C) = W \cdot (-D) = -1 < -\theta$. Hence weights with the possible values $w_i = -2, -1, 0, +1, +2$ should be capable of converging to a solution. Given the repetitive training sequence of inputs, A, B, C, D, $-A$, $-B$, $-C$, $-D$, A, B, C, D, etc., however, the weight-adjustment procedure of Eq. (266) (with "limiting" of the weights) causes the weights

to oscillate cyclically, and a solution is never attained:

$W(0)$		0	0	0	0	
$W(1)$	A	1	1	1	1	
$W(2)$	B	2	0	0	0	
$W(3)$	C	1	1	−1	1	
$W(4)$	D	0	0	0	2	...
$W(5)$	$-A$	1	1	1	1	
$W(6)$	$-B$	2	0	0	2	
$W(7)$	$-C$	1	1	−1	2	
$W(8)$	$-D$	0	0	0	2	
$W(9)$	A	1	1	1	1	
$W(10)$	B	2	0	0	2	
$W(11)$	C	1	1	−1	2	
$W(12)$	D	0	0	0	2	...

Hence it appears that the range of the counters used to hold the weights must be greater than the minimum necessary for a solution to exist. In the following section it will be shown that this inefficiency in the use of storage is unnecessary if the weight changes are made stochastically.

8.4. Convergence of Stochastic Adaptive Threshold Logic

The adaptive procedure of Eq. (271) implies that when an error is made *all* the weights are adjusted so as to decrease the error. Suppose now that only some of the weights are changed, and that the decision to change each weight is made independently according to a probabilistic law. If the probability of change is zero, then no correction is made and the algorithm obviously does not converge. Equally, if the probability of change is unity, then the process reduces to the deterministic algorithm of Eq. (271), which has been shown not to converge. For intermediate values of the probability, however, it may be shown that convergence is guaranteed provided there is a solution within the range of weights.

Using the notation of Section 8.3, suppose $X(N)$ is a sequence of misclassified, normalized input patterns so that Eq. (270) still holds, and that there exists a solution W_i so that Eq. (272) still holds. The weight-adjustment procedure of Eq. (271) is, however, modified to

$$W_i(N+1) = W_i(N) + \phi_i(N)X_i(N) \tag{278}$$

where $\phi_i(N) = 0,1$ is a random variable from a family of independent random variables whose generating probability is p, such that $0 < p < 1$.

Consider now the change in the deviation of $W_i(n)$ from W_i. From Equation (278) we have

$$\sum_{i=0}^{n} [W_i(N+1) - W_i]^2 = \sum_{i=0}^{n} [W_i(N) - W_i]^2$$
$$= +2 \sum_{i=1}^{n} W_i(N)[W_i(n) - W_i + X_i(N)/2]X_i(N) \tag{279}$$

Combining Eqs. (270) and (272), we have

$$\sum_{i=1}^{n} [W_i(N) - W_i]X_i(N) < 0 \tag{280}$$

Hence there must be some value of i, f say, such that

$$W_j(N) - W_i \leq -1 \tag{281}$$

There is a finite probability that

$$\phi_i(N) = \delta_{ij} \tag{282}$$

(the Dirac δ-function) and in this event, combining Eqs. (279) and (282) and using inequality (281) we have

$$\sum_{i=1}^{n} [W_i(N+1) - W_i]^2 \leq \sum_{i=1}^{n} [W_i(N) - W_i]^2 - 1 \tag{283}$$

Thus if any of the decisions of the stochastic ATLE are incorrect, they give rise to weight adjustments which have a finite probability of decreasing the distance of the weight vector from an arbitrary solution vector. Since the weights are bounded, and can take only a finite set of values, there is a finite probability that over any sequence longer than a certain length [in fact, longer than $\Sigma_{i=1}^{n} (W_i^2 + W_{\max}^2)$] the weights will have converged to an arbitrary solution. Hence the probability that the stochastic ATLE will have converged goes to unity with the length of the learning period.

Thus, provided there is a solution within the range of its weights, the stochastic ATLE will always converge to a solution; it does not require the extended range of weights of the ATLE necessary only to ensure convergence.

8.5. An Application of Stochastic ATLEs

Gene Clapper of IBM has developed an adaptive recognizer for classifying speech, or handwritten characters, into one of sixteen categories ([76]). The characters are written with a metal stylus, on a 3×5 matrix of metal tabs, and thus creates a 15-bit input pattern. The speech input is separated by filters into four frequency bands, and the presence or absence of energy in each of these bands for each of the first three discrete segments of the word spoken is noted. A fifth channel records whether or not each segment lasted more than 100 msec. Hence speech inputs are also coded as 15-bit input patterns.

The decision-making part of recognizer is a set of four ATLEs, each with a binary output, so that, taken together, they are able to classify inputs into 16 categories. In order to ensure linear separability for the patterns of interest, Clapper incorporates a recoding network between the 15-bit input patterns and the ATLEs. This expands the original inputs into 35-bit patterns, by taking each three-bit row of the original 3×5 matrix and transforming it, through binary-to-linear conversion, to a 1-in-7 line code.

The ATLEs incorporates 35 weights each, which can each take one of the five values $-2, -1, 0, 1, 2$. These weights are adjusted after each decision of the recognizer according to an algorithm similar to that of Eq. (266). Thus the situation is similar to that described in the previous section, with ATLEs having discrete bounded weights being used in a deterministic adaptive loop. Since the input patterns have 35 bits, $n = 35$, and for subsets of input patterns which may be linearly separated with weights taking values up to two in magnitude, the upper bound on the weight values required for convergence is $2 \times 35^{3/2} \approx 400$. Even if this bound is very much higher than necessary, one might suspect that Clapper's recognizer with deterministic feedback would run into problems of nonconvergence as illustrated in Section 8.4.

Clapper did indeed find this nonconvergence with deterministic feedback. When a fixed set of input characters was presented cyclically during the learning phase, "After 12 runs the machine got into a vicious circle that prevented 100% learning. It had most of the 16 patterns down pat, but was continuously confused by a few similar pairs." Most interestingly, however, he noticed that this did not seem to occur with speech inputs, and hypothesized that these "are varied enough even when the same person does all the speaking." In order to get a variety of inputs from characters similar to those produced by speech, Clapper introduced a stochastic perturbation

which randomly turned off some of the ones in the 35-bit input patterns to the ATLEs.

He states that, "Without this technique, called statistical conditioning, the machine would have to learn the full set of input patterns without seeing anything less than complete characters, a difficult task. With this conditioning, however, it learns that each digit is represented by one of several sets of identifying features, each set smaller than the one describing the complete character." In the same situation where the machine had previously reached a "vicious circle" he found that when, "the random number generator was switched on, and the same patterns were presented again, in the same order, the machine learned all of them perfectly in six trials."

Clapper's "statistical conditioning" introduces exactly the required random variation in the weight-adjustment process which was shown in Section 8.5 to be necessary for the convergence of discrete ATLEs with bounded weights. His justification of it in terms of the extraction of a subset of features from the entire set corresponding to an input character is an interesting variation on the mathematical approach taken in that section, and throws some light on the nature of the learning process in ATLEs.

8.6. Inefficiency and Redundancy in ATLEs

The requirement of linear separability of the two input classes for there to exist a solution realizable by an ATLE is a very strong one. The proportion of dichotomies of binary vectors which are linearly separable is not known in analytical form, but it has been shown to be asymptotic to zero with the size of the vector. Ridgeway ([77]) has shown experimentally that the number of binary patterns which may be classified arbitrarily averages about twice the number of weights, and hence the proportion of patterns which may be classified *ad lib* goes down as $n/2^{n-1}$, a very rapid fall off.

However, this measure of the efficiency of the ATLE as a pattern classifier only determines what possible classifications out of an arbitrary set are possible for it. It does not indicate how efficient the ATLE is in the use of its own storage, i.e., how many different dichotomies it can realize compared with the number of different weight vectors it can implement. Two factors work against the utility of the proportion of linearly separable input patterns as a measure of the efficiency of the ATLE. First, linear separability is a topological property which may be essential to the existence of "convergence" phenomena in pattern classifiers; certainly there is no other adaptive pattern classification algorithm with the universality and power of the ATLE. Second, the number of dichotomies of binary n-vectors is 2^{2^n}, so that 2^n bits of storage are required to label every possible dichotomy.

This is very large indeed in practice; e.g., in Clapper's recognizer $2^{15} \approx 3 \times 10^4$ bits would be necessary; for a 10×10 input retina some 10^{30} bits are required to store one out of all possible classifications.

Hence one does not wish to have all possible classifications available, and a more reasonable question to ask is how many different classifications can an ATLE with a certain number of bits of storage make. This question does not become meaningful until the ATLEs have discrete weights, since continuous weights correspond, in theory, to an infinite range of possible values. Aleksander and Albrow [78] make this comparison for an ATLE and for their adaptive classifier used in a threshold logic mode, and point out the inefficiency of storage in the ATLE due to the requirements of the adaptive algorithm.

The stochastic ATLE has no redundancy, which is necessary only for convergence. However, there remains some inefficiency in the use of storage, since, e.g., the weight vectors W_i and $2W_i$ will realize the same dichotomy. At present the information necessary to determine the extent of this redundancy is not available, since two questions concerning discrete threshold logic elements remain unanswered despite considerable research effort [79,80]. First, what is the maximum weight, as a function of n, necessary to realize all linear separations of binary n-vectors using integer weights? Second, a more difficult question which includes the first, what is the number of different linear separations possible with weights taking a certain range of values?

Thus there are two levels at which the efficiency of an ATLE can be evaluated: (1) The proportion of all dichotomous classifications of binary n-vectors which it can realize; and (2) the ratio of the number of different linear separations which it can realize to the number of different configurations which its weight values may take. Neither of these functions is yet known, but the second is of greatest practical importance. This section has shown that the second measure of efficiency may be greatly improved through the use of stochastic ATLEs, but the upper limit of this improvement is not yet known.

9. GRADIENT TECHNIQUES FOR THE IDENTIFICATION OF LINEAR SYSTEMS

The problem of *identifying* a linear system, or measuring its internal parameters from its input–output behavior, is a basic one in adaptive control, and a wide variety of possible techniques for doing this have been suggested and tested by simulation or by experiment [81,82]. One of the commonest and most widely applicable approaches to linear system identification is to cross-correlate the input and delayed output of the system over a range of time delays in order to obtain the impulse response [83,84,48]; from this all other characterizations of a linear system may be obtained.

Direct correlation techniques for measuring the impulse response have a number of disadvantages—in particular, the computation is open-loop, in that the value of the measured cross-correlation is used directly and must be measured accurately; other disadvantages are the requirement for many time delays, the fact that the form of the results is an impulse response rather than a model of the system, and the uncertainty about the "meaning" of the results if the system identified is not linear.

All these disadvantages are overcome in what is essentially a null-point variation of the correlation technique, the gradient technique for minimizing the "satisfaction error" between the output of the system and the output of a linear model driven from the same input [85–89]. The "model" consists of a number of signals derived from the input, multiplied by weighting coefficients and summed to give the output. The satisfaction error between the model output and the system output is multiplied by the delayed terms and fed back through integrators to the corresponding weighting terms in such a sense that excessive value of a weight causes it to be reduced, and *vice versa*. The weights will clearly attain stable values only when the cross-correlation between the satisfaction error and the delayed inputs are all zero. Hence this is a null technique, and has been termed "error-decorrelation. [90]"

The model-reference gradient technique has the advantages that: the computation is closed-loop, and errors in the measured parameters are fed back in such a sense as to reduce themselves; the model used may be a simple and natural physical model of the system, and the parameters have a direct interpretation; only the zero crossings of the correlation functions have to be accurately measured; and, if the system is nonlinear, the match between it and the model may be interpreted in terms of minimum satisfaction-error.

In this section the theoretical basis for model-reference parameter estimation is introduced, and an experimental study is outlined of various hardware implementations of the technique, including three stochastic computing approaches.

9.1. Theoretical Analysis of Model-Reference Parameter Identification

Gradient techniques for implicit function generation have already been described in Section 4.11, in the context of stochastic division and square-

root extraction, and the theory behind system identification using the gradient technique is essentially the same as given in that section. However, because the signals to be matched are now essentially time-varying, a rigorous analysis of the behavior and convergence of the gradient technique is more difficult. In this section a reasonable heuristic analysis will be given to explain the model-reference parameter identification technique and the problems which can arise in its application.

Suppose $X_i \equiv X_i(t)$ $(0 \leq i \leq K)$ is a family of signals, or functions of time, and that the output $Y \equiv Y(t)$ of an unknown system is a linear combination of these signals:

$$Y = \sum_{i=0}^{K} a_i X_i \tag{284}$$

Let Z be the output of a "model" formed by taking some linear combination of the signals:

$$Z = \sum_{i=0}^{K} w_i X_i \tag{285}$$

It is required to make the model fit the system by minimizing the squared deviation of the model output from the system output over some interval; i.e., by minimizing

$$S \equiv S(T) = \int_0^T (Y - Z)^2 \, dt \tag{286}$$

This integral can be put into a convenient form by letting

$$A_{ij} \equiv A_{ij}(T) = \int_0^T X_i X_j \, dt \tag{287}$$

so that A_{ij} is the covariance of X_i and X_i over the given interval, assuming, without loss of generality, that all the signals, except X_0, have a mean of zero; X_0 is the dc component. If we also put

$$e_i = w_i - a_i \tag{288}$$

then Eq. (286) may be written

$$S = \sum_{i=0}^{K} \sum_{j=0}^{K} e_i A_{ij} e_j \tag{289}$$

assuming that both a_i and w_i are constant.

Equation (288) shows that in general the effect of an error e_i is dependent not only on the signal X_i, but also on the covariance of X_i with other

signals. This is reasonable, since any covariation between signals implies that they have a common component, which is multiplied by both their coefficients. These covariance terms make for difficulties in the convergence of any process which is trying to drive the e_i to zero (e.g., if $X_1 = -X_2$, then a term of the form $wX_1 + wX_2$ can be added into the output of the model without affecting S, and hence a_1 and a_2 cannot be individually determined), and it is usual to choose the X_i to be orthogonal signals, so that

$$A_{ij} = 0 \qquad i \neq j \tag{290}$$

Equation (289) also shows that the effect of an error e_i in the performance measure S is a function of A_{ii}, which itself corresponds to the energy present in the signal X_i. If X_i should be of low strength, then the error e_i will have little effect, and any procedure estimating a_i cannot be expected to derive an accurate estimate; i.e., the part of the system corresponding to a_i has not been "excited."

Assuming that the signals are orthogonal and present in reasonable strength, if we set

$$\dot{w}_i = -\alpha(A - Y)X_i \tag{291}$$

where $0 < \alpha \ll 1$, then the w_i may be shown to tend to the a_i at a rate proportional to A_{ii}, the energy present in the signal X_i, since rewriting Eq. (291),

$$\dot{e}_i = -\alpha \sum_{j=0}^{K} e_j X_j X_i \tag{292}$$

This informal analysis indicates the type of computation and assumptions involved in obtaining a model of a linear system using a gradient technique. In practice, the model used will probably not contain all the terms necessary to match the system, and the signals will not be completely orthogonal. The most important practical complication, however, is that the signals X_i will not themselves be exactly known for substitution in Eq. (291). Instead, there will be available estimates of the X_i contaminated with noise:

$$Y_i = X_i + N_i \tag{293}$$

where the N_i are noise sources, each with zero mean, assumed uncorrelated both among themselves and with the X_i.

The effect of this added noise is to introduce a bias in the estimates of the parameters a_i. For, substituting Y_i instead of X_i in Eqs. (285) and (291) [not in Eq. (284), because the noise is only on the signals measured

for purposes of modeling], we have

$$\dot{e}_i = -\alpha \sum_{j=0}^{K} (e_i X_i + w_i N_i)(X_j + N_j) \tag{294}$$

Under the same assumption as before, e_i will be zero on average when

$$A_{ii} e_i + w_i \int_0^T N_i^2 \, dt = 0 \tag{295}$$

i.e.,

$$w_i = a_i A_{ii} / \left(A_{ii} + \int_0^T N_i^2 \, dt \right) \tag{296}$$

and the estimate of a_i is reduced by an amount which depends on the relative energies of X_i and N_i; this effect is called the *noise bias* in the estimate of a_i.

9.2. Linearization of Polarity-Coincidence and Relay Correlators

Before comparing the realization of gradient techniques in the stochastic computer with alternative implementations of the same technique, it is interesting to make the same comparison for open-loop identification using cross-correlation to obtain the impulse response. When covariances are computed for many delays at the same time the cost of analog multipliers is substantial, and there have been many attempts to reduce the cost of correlators by replacing these with relays or gates, and using mixed analog/digital ([17,91]), or purely digital ([92,93]) computation. The relay correlator approximates to the product XY with the term $X[\text{sign}(Y)]$, and can be realized by a comparator fed from Y and driving a relay selecting either X or $-X$. The polarity-coincidence correlator takes the sign of both channels, forming $[\text{sign}(X)][\text{sign}(Y)]$, and hence may be considered as a logical operation on two binary variables, in fact, an EXCLUSIVE-OR operation. Under low-noise conditions with Gaussian signals these techniques may be shown to give results in a 1:1 correspondence with the true correlation function ([41]). However, in the presence of noise or with strongly asymmetrical distributions this correspondence no longer holds, and the structure of the true impulse response is lost.

Addition of appropriate random signals to the correlator inputs has been suggested as a means of overcoming these defects ([47,94]). and leads to a structure identical to that of a stochastic multiplier and integrator for quantities in the single-line bipolar representation III (Section 4). This has been described in one of the examples of prestochastic computing using noise signals (Section 3.3). The accuracy of the results for a given period of integration may be increased by utilizing the two-line representation II. This may be seen as an application of coarse quantization ([95,96]) together with stochastic interpolation. Similarly, the action of a stochastic representation in removing the effects of noise on polarity-coincidence correlation and emulating analog multiplication may be viewed as statistical linearization ([97,98]) of the relay switching function.

9.3. Comparison of Hardware for Gradient Techniques

The same considerations as to economy in hardware by replacement of analog multipliers apply in closed-loop parameter determination using gradient techniques. However, while the linearization inherent in the stochastic representation is essential in obtaining unbiased results from an open-loop determination of process parameters using polarity-coincidence correlation, it is by no means obvious that the same is true of closed-loop determinations.

It has been suggested that relay correlation may lead to interactions between coefficients which are otherwise orthogonal ([88]), and it might be expected that the noise bias would be greater for polarity-coincidence techniques. Since, however, the zero crossings of the polarity-coincidence function and the true correlation function are the same for Gaussian inputs ([41]) even in the presence of noise, the noise bias of null-seeking gradient techniques should be virtually independent of the method of correlation used.

Because of the practical importance of gradient techniques in both communication and control, a detailed comparison of the performance of various forms of hardware would be of great interest. However, no general comparison is available in the literature, except for a minor one by the present author ([5]), which included various stochastic techniques; this is outlined below.

The problem selected for study was the determination of the parameters of the first-order lag whose transfer-function is $1/(a_1 + a_2 s)$. In terms of the general equation (284) the input–output relationship is

$$Y = a_1 X + a_2 \dot{X} \tag{297}$$

The signals X and $\dot{X}$ (the time derivative of X) were chosen because they are orthogonal in the sense of Eq. (290).

Writing $E = Z - Y$ Eq. (291) and assuming $X_1 = X$, $X_2 = \dot{X}$, the equation for variation of the model weights may be written $W_i = -\alpha E X_i$.

Six different techniques for approximating this equation have been compared:

a. Steepest Descent Technique. Here

$$w_i = -\alpha E X_i \tag{298}$$

and analog multipliers are required for both the weights and their adaption.

b. Relay Correlation Technique. Here

$$w_i = -\alpha \operatorname{sign}(E)\, X_i \tag{299}$$

and analog multipliers (which may be simple motorized potentiometers) are required for the weights only.

c. Stochastic Relay Correlation Technique. Here

$$w_i = -\alpha \operatorname{sign}(E + v)\, X_i \tag{300}$$

where v is a random variable, uniformly and symmetrically distributed over the range of E; this effectively represents E as a stochastic sequence in representation III (Section 4), and linearizes the relay.

d. Polarity-Coincidence Technique. Here

$$w_i = -\alpha \operatorname{sign}(E) \operatorname{sign}(X_i) \tag{301}$$

where w_i now takes discrete values and the integration is performed by a counter, so that no analog multipliers are required.

e. Stochastic Two-Line Technique. Here

$$w_i = -\alpha \operatorname{sign}(E) \operatorname{sign}(X_i)[1+\operatorname{sign}(|E| - v)][1+\operatorname{sign}(|X_i| - \delta)]/4 \tag{302}$$

which may be regarded as a polarity-coincidence correlation with stochastic weighting according to the magnitude of the inputs (v and δ are random variables uniformly distributed in the range of magnitudes of E and X_i, respectively), or as stochastic multiplication in representation II.

f. Stochastic Single-Line Technique. Here

$$w_i = -\alpha \operatorname{sign}(E + v) \operatorname{sign}(X_i + \delta) \tag{303}$$

which may be regarded as polarity-coincidence correlation with statistical linearization of the comparators (v and δ are random variables symmetrically and uniformly distributed in the range of magnitudes of E and X_i, respectively), or as stochastic multiplication in representation III.

9.4. Experimental Comparison of Gradient Techniques

There are four figures of merit which have to be determined for the various techniques.

a. Speed of Response. How rapidly are the weights adjusted in response to a step change? This is not simple to define, since the equivalent gain of a relay is a function of the input amplitude. Hence when E becomes large following a change in the system parameters, the speed of response of methods a, c, e, and f is greater than that of methods b and d. As the estimates converge and E becomes small, however, the speed of the first group decreases, while that of the second pair remains constant. This crossover shows clearly in the graph of Fig. 32, which compares methods a and b in estimating a_0 with noise-free data.

b. Bias Due to Noise on Data. All the techniques will underestimate the magnitude of the weights when uncorrelated noise is added to Z and each of the (X_i). For the reason stated earlier it was expected that this bias would not be greater for methods c and d than for the others, but this was subject to check under realistic conditions.

c. Interaction between Weights. In order that a step-change in one parameter shall not cause transient changes in the estimates of the others, it is necessary to choose the (X_i) to be orthogonal functions. The a_1 and a_2 multiply orthogonal functions with respect to methods a, c, e, and f, but it is not obvious that they remain orthogonal for b and d (although the results of Veltman and van den Bos ([41]) would suggest that they do under reasonable conditions).

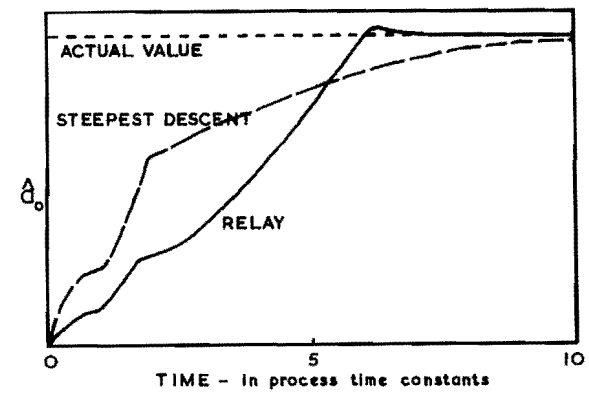

Fig. 32. Response of various parameter estimators.

d. Variance in the Final Estimates. Under noise-free conditions E tends to zero and the estimates reach fixed stable values (apart from the one-unit relay "chatter" in methods b and d, and the stochastic variance in methods c and f). Noise on the (X_i), however, induces variance in the (w_i) which increases with the gains α of the various methods. If the speeds of response are matched (in some sense), then the gain-varying methods (b and d) would be expected to have more variance in their estimates, as would also the stochastic binary techniques (c and e).

The gain of each method was adjusted so as to equalize their speeds of response (time taken to traverse 0.9 of the step) under high-noise conditions (uniformly-distributed noise, 0.2 of signal). The noise bias in the estimate was measured under these conditions together with the variance of the final estimate. The interaction between coefficients was measured under noise-free conditions by examining the effect of a step in one parameter on the estimate of the other. The overall time of convergence was set to be about five process time constants.

It was found that:

1. The noise-bias was the same in every case (estimate about 0.8 of true value).
2. The interaction was the same in every case.
3. The variances of the final estimates were approximately in the ratios:

steepest descent (1): relay correlation (4): stochastic relay correlation (2): polarity coincidence (10): stochastic two-line (1): stochastic single-line (2).

Thus the high value of α necessary to give the relay and polarity-coincidence techniques a sufficient speed of response leads to excessive gain in the converged condition and a lowered overall accuracy.

There are several features of interest in these results: (1) use of relay correlation and polarity-coincidence correlation do not lead to the artifacts in the results which would occur in the open-loop measurement of impulse response—this demonstrates the essential robustness of the null-point technique; (2) the stochastic representation (e) not only gives the same accuracy as the best deterministic technique (a), but also gives the same speed of response (the experimental response curves are virtually identical)—this equal efficiency derives from the fact that identification is itself an essentially slow process, and the time taken is a function of the process time constant, rather than the speed of computations; (3) these results are a very clear demonstration of the value of added noise ([99]) and statistical linearization, and emphasize the close links between these processes and the linear forms of the stochastic computer.

9.5. Identification of Low-Bandwidth Parameters in High-Bandwidth Systems

In the stochastic identification techniques discussed in the previous sections the entire computation has been carried out with the data represented as a stochastic sequence, and hence the bandwidth of the model has been severely limited. In many cases of interest, however, such as the identification of the characteristics of communication links, the system to be identified contains very-high-bandwidth signals, but its, parameters are varying very slowly. In this event the stochastic techniques may be applied only in the parameter-adjustment loop.

This is achieved by simultaneously sampling the signals in the system and its output, using a strobe pulse, or window, which is narrow compared with response time of the system. The samples taken at the same instant obey the system equations, and a set of them is sufficient to identify the system parameters ([100]). Information is clearly lost by infrequent sampling, but since the parameters of the system are assumed to be very slowly varying, there is an over-abundance of information about the system available, and inefficient use of it does not matter.

The configuration adopted will be very similar to that of Fig. 34, with strobed sample/hold circuits placed before the analog/stochastic convertors. In the forward path of the model, however, the input must be fed through directly, without conversion, to an analog, rather than a stochastic, summer. The weight values, formed in the counters of the stochastic integrators, must be used to multiply analog quantities. This may be done precisely by having the digital output of the counter control the switched summing resistors of an operational amplifier, but this leads to transients in the analog channel which decrease its available bandwidth, and is expensive. In fact, an exact linear relationship between the digital output and the analog coefficient is not essential, because the multiplier is within the adaptive loop; only a monotonic relationship without wide variation in slope is required. This may be achieved by using the digital output to control a photoemissive source driving a photoconductive input resistor to an amplifier; by this means low-bandwidth digital control of very-high-bandwidth analog channels is economically performed.

10. BAYES' PREDICTOR FOR BINARY INPUTS

Statistical inference based on Bayes' theorem is fundamental to modern probability theory ([101–103]), and devices to implement various predictors

based on the theorem have been put forward many times as basic components of learning machines ([104–107]). The estimation of the likelihood ratios involved in a form convenient for prediction is not very readily realized in hardware, however, and the techniques proposed in the literature have involved approximate estimation and prediction procedures. In this section the theory behind the Bayes predictor is first outlined, and then a stochastic computing system which simply and naturally performs the required computations without approximation is described.

10.1. Maximum Likelihood Prediction

Consider some event E whose occurrence [designated $e(n) = 1$] is dependent upon which members of a set of events $\{E_i\}$ have occurred (for $1 \leq i \leq N$). Let the binary vector $\{e_i\}$ be such that $e_i = 1$ if the event E_i has occurred and $e_i = 0$ otherwise. Then the probability that $e(n) = 1$, given $\{e_i(n)\}$, is to be used to make the best prediction of E. The simplest utilization is that of maximum liklihood prediction, in which E is predicted to occur if

$$p[e(n) = 1 \mid \{e_i(n)\}] > 0.5 \tag{304}$$

but more complex procedures may be used if further information is available, e.g., if the costs of predicting incorrectly one way or the other are known.

To estimate the conditional probability of E occurring for each possible combination of events from $\{E_i\}$ would not only require great storage capacity, but also much experience, and in practice it is necessary to make certain simplifications which are equivalent to having the machine generalize from its experience. An estimating and predicting scheme based on Bayes' theorem together with certain assumptions about the nature of the events E and $\{E_i\}$ is developed below.

10.2. Estimation and Prediction Based on Bayes' Theorem

Bayes' theorem is best regarded as a simple manipulation of conditional probabilities, in which the conditional probabilities of one event given another, $p(A|B)$, and its converse, $p(B|A)$, are both expressed in terms of the probability of their joint occurrence, $p(A \cdot B)$, and of their individual occurrences, $p(A)$, $p(B)$. By definition

$$p(A|B) = p(A \cdot B)/p(B) \tag{305}$$

$$p(B|A) = p(A \cdot B)/p(A) \tag{306}$$

and hence

$$p(A|B) = p(B|A)p(A)/p(B) \tag{307}$$

This apparently simple result is dignified with the name of a theorem because its foundations are far more complex than is implied here—the events A and B will not generally be simultaneous, and the effect on the probability of one event of the occurrence of a future event is not a simple concept—however, the debate on the meaningfulness of Bayes' theorem is of no concern in the present context.

In terms of the events $e(n)$ and $e_i(n)$ of the previous section, consider the likelihood ratio

$$L(n) = p[e(n) = 1|\{e_i(n)\}]/p[e(n) = 0|\{e_i(n)\}] \tag{308}$$

By Bayes' theorem this can be re-written

$$L(n) = \frac{p[\{e_i(n)\}|e(n) = 1]\,p[e(n) = 1]}{p[\{e_i(n)\}|e(n) = 0]\,p[e(n) = 0]} \tag{309}$$

Now assume that the components $e_j(n)$ are statistically independent under the occurrence of E, so that

$$p[\{e_i(n)\}|e(n) = 1] = \prod_{i=1}^{n} p[e_i(n)|e(n) = 1] \tag{310}$$

and apply Bayes' theorem to the expanded terms:

$$L(n) = \frac{p(e = 1)}{p(e = 0)} \prod_{i=1}^{N} \frac{p(e = 1|e_i)\,p(e = 0)}{p(e = 0|e_i)\,p(e = 1)} \tag{311}$$

which may be written

$$L(n) = L_0(n) \prod_{i=1}^{N} L_i(n) \tag{312}$$

where

$$L_0(n) = p(e = 1)/p(e = 0) \tag{313}$$

and

$$L_i(n) = \frac{p(e = 1|e_i)\,p(e = 0)}{p(e = 0|e_i)\,p(e = 1)} \tag{314}$$

where the notation is obvious. The quantities $L_i(n)$ are the normalized

likelihood ratios relating E and E_i, and it can be seen that $L_i(n)$ is unity if the occurrence of E_i does not affect the probability of occurrence of E.

Equation (314) gives the overall likelihood ratio $L(n)$, which is sufficient for maximum likelihood prediction, since, from Eq. (304) the condition for predicting the occurrence of E is now

$$L(n) > 1 \tag{315}$$

This quantity is expressed as the product of $L_0(n)$, the unconditional likelihood ratio, and the product of the normalized likelihood ratios for those events which have occurred. These likelihood ratios are difficult terms with which to compute, since they range in value from zero to infinity. It is customary to take the logarithm of Eq. (314), turning the product into a sum for easier computation; the quantity $\ln[L_i(n)]$ is in fact the information in the event E_i relative to the event E ([108]). The logarithms of the likelihood ratio take values in the range from minus infinity to plus infinity, however, and some truncation is necessary. Estimation of L_i, $\ln(L_i)$, or even the unormalized likelihood ratios is also difficult because they are nonlinear and unbounded functions. In practice, no compatible estimation and prediction scheme using simple computing elements has been developed for the Bayes predictor.

10.3. Stochastic Computing Realization of Bayes' Estimator and Predictor

Estimation of the $L_i(n)$ and prediction from them can be achieved very simply using a form of stochastic integrator, or ADDIE, with the likelihood ratios represented by the p/q mapping described in Section 6. To each event E_i there corresponds an ADDIE estimating a probability p_i such that

$$p_i/(1 - p_i) = L_i(n) \qquad 1 \leq i \leq N \tag{316}$$

A single integrator connected to the outputs of these estimators implements Eqs. (314) and (315) for maximum likelihood prediction.

The gating necessary for the fractional count p_i in the counter of an ADDIE to obey Eq. (316) is shown in Fig. 33*a*. The input lines corresponding to the events E and E_i are ON only when these events occur. The line representing E_i feeds both the input gates of the ADDIE so that no change occurs in the stored count if this event has not occurred. The line representing E goes to the gate on the INCREMENT line directly, and to that on the DECREMENT line inverted. The stochastic output of the integrator

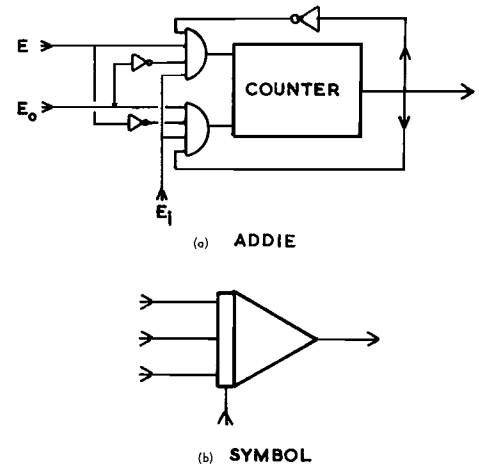

Fig. 33. ADDIE for Bayes estimation.

feeds back to the input gates in the normal way. The main difference between the configuration and that of a linear ADDIE is that another line, labeled E_0, feeds both inputs. This is the output of a linear ADDIE estimating the unconditional probability of occurrence E, and hence its generating probability is $p(E_0) = p(e = 1) = P_0$.

The equilibrium equation for the stochastic integrator with this gating at its input is

$$(1 - p_i)p(E \cdot E_i \cdot \bar{E}_0) = p_i p(\bar{E} \cdot E_i \cdot E_0) \tag{317}$$

Now E_0 is independent of the other two terms, so that this equation may be rewritten

$$(1 - p_i)p(e = 1|e_i)p(e = 0) = p_i p(e = 0|e_i)p(e = 1) \tag{318}$$

which, from Eqs. (314) and (316), is the required relationship. It is convenient to use a three-input integrator symbol for the ADDIE with this gating, as shown in Fig. 33*b*, even though it does not form the sum of its inputs, as does the two-input summing integrator previously described; the E_i input must be differentiated from the other three, and is shown as a track/hold input to the integrator.

This form of ADDIE performs the direct estimation of normalized likelihood ratios. To use these for maximum likelihood prediction, it is only necessary to compute whether inequality (315) holds, and this is

equivalent to

$$\prod_{i=0}^{N} p_i > \prod_{i=0}^{N} (1 - p_i) \tag{319}$$

where the product is taken over the events E_i which have occurred. Figure 34 shows a complete estimation and prediction system for performing this computation: the ADDIEs estimating the likelihood ratios change their state only when the ESTIMATE line is ON; the integrator at the output is an $(N + 1)$ input version of the same type of ADDIE; its counter increments only when all its inputs are ON and decrements only when they are all OFF—hence the output of the integrator, used as a switching function, is ON or OFF depending on whether inequality (319) is or is not satisfied.

Thus an estimator and predictor for binary events based on Bayes' theorem and maximum likelihood prediction is simply and economically realized using stochastic computing elements.

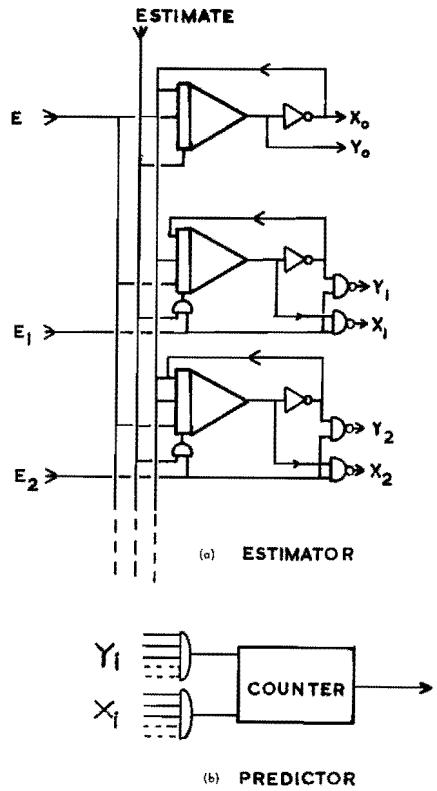

Fig. 34. Bayesian estimator and predictor.

11. NETWORKS OF STOCHASTIC COMPUTING ELEMENTS

It was noted in Section 6.6 that stochastic computing techniques are particularly amenable to the construction of *networks* of computing elements in which the output of any one element is in a form suitable for feeding to another. Examples of stochastic computing nets have already been given: the adaptive threshold logic element of Section 8 uses a number of ADDIEs and gates uniformly connected to give a general-purpose pattern classifier; the steepest-descent identifier of Section 9 uses ADDIEs and gates, or analog weighting elements driven from the ADDIEs, uniformly connected as a general-purpose identifier of linear systems; the Bayesian estimator and predictor of Section 10 uses special-purpose ADDIEs and gates, uniformly connected to form an identifier of discrete, probabilistic systems.

Further examples of uniform computing networks have been given in the literature on stochastic computing: Poppelbaum *et al.* [2] have described a stochastic array computer in which a network of general computing elements is interconnected to derive a particular function; they have also described a stochastic image transformer to perform general matrix operations on a retina, and hence realize translations, rotations, magnifications, Fourier transformations, and so on; Ribeiro [3] has described spatial integrators and matrix multipliers consisting of uniform networks of stochastic computing elements.

The following subsection gives a brief description of the application of stochastic computing techniques to the solution of partial differential equations, where uniform networks of computing elements again naturally arise.

11.1. Solution of Partial Differential Equations

Analog computers have one natural independent variable, and that is time. Partial differential equations involving several independent variables are usually solved iteratively on a hybrid computer, or through a discrete approximation on the digital computer. The use of stochastic computing elements makes it feasible to solve the equations with spatial arrays of special-purpose computing elements, with one spatial dimension for each independent variable. A stochastic solution of Laplace's equation will be used to illustrate this technique.

Laplace's equation in two dimensions,

$$u_{xx} + u_{yy} = 0 \tag{320}$$

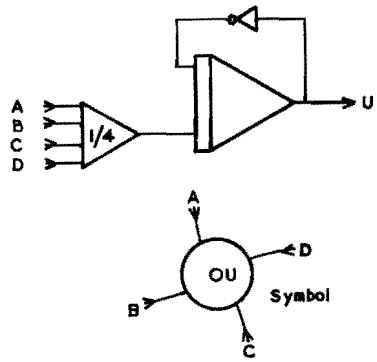

Fig. 35. "Laplacian" element.

with boundary conditions defined on a closed curve, may be expressed in a discrete form by taking intervals of e in each of the independent variables:

$$u(x-e,y)-2u(x,y)+u(x+e,y)+u(x,y-e)-2u(x,y)+u(x,y+e)=0 \tag{321}$$

so that

$$4u(x,y)=u(x-e,y)+u(x+e,y)+u(x,y-e)+u(x,y+e) \tag{322}$$

The function $u(x, y)$ is defined at each point of a rectangular grid in x and y, and Eq. (322) gives a relationship between the value of u at one point on the grid and its value at the four neighboring points. If $u(x, y)$ is represented by the output of a stochastic integrator at each point of the grid, then Eq. (322) may be enforced by feeding back to its input its inverted output together with the outputs of the four neighboring integrators. A suitable computing element, the "Laplacian," is shown in Fig. 35: a linear ADDIE receives the output of an adder, which sums the peripheral terms of the above equation together with the normal inverted feedback from its output; the value represented by the ADDIE can settle only when the above equation is satisfied. Networks of these units, interconnected as shown in Fig. 36, can be used to solve Laplace's equation in two-dimensions for arbitrary shapes and boundary conditions.

11.2. Relationship to "Neural Nets"

Spatial arrays of such stochastic computing elements as that shown in Fig. 37 and described elsewhere in this chapter have interesting resemblances

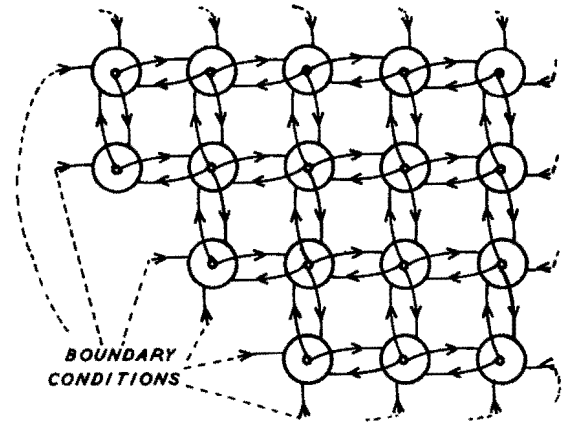

Fig. 36. Network of "Laplacians" for solving Laplace's equation.

to the "neural nets" simulated by Farley and Clark [109] and Beurle [110], and the stochastic computing elements themselves are reminiscent of the artificial "neurons" studied by Harmon [111] and other workers. Neurons in the brain are known to show stochastic behavior, and it is possible that stochastic computing may provide not only a new impetus to work on neural nets, but also a reasonable model of some cortical functions, e.g., the cross-correlational processes of visual disparity and auditory formant separation.

Other examples of stochastic computing elements whose behavior may be relevant to the functioning of the human nervous system have been given in preceding sections. The synchronizing element of Fig. 21, described in Section 4.14, is of interest not only because it synchronizes the input stream, but because the rate of pulses at the output is strictly limited to be less than the "clock" rate, and yet the element never overloads and loses information, no matter what the pulse rate at the input. The conversion of a stochastic pulse stream whose pulse rate may vary by several orders of magnitude to a

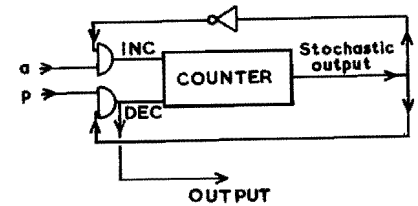

Fig. 37. Adaptive stochastic convertor.

stochastic pulse stream which is bounded in its rate is a problem faced by the photoreceptors and neurons in the retina of the eye.

The basic synchronizing element of Fig. 21 shows no memory of its past firing rate, and hence has no equivalent of the time constants in dark/light adaption. Consider a model of a sensory receptor in the eye in which an incoming photon causes a nervous impulse to be emitted if it is absorbed in a molecule of "pigment". In being absorbed it bleaches the pigment and hence reduces the probability that another photon will be absorbed; pigment regenerates at a slow rate. The overall effect will be that the rate of nervous firing is a monotone function of the rate of input of photons, but the function is clearly nonlinear, since the proportion of photons which is absorbed decreases with the mean rate of input.

In the stochastic computer this phenomenon may be simulated by and ADDIE, made up of a counter with stochastic output (representing the fraction of pigment unbleached), which is fed back in the normal way to AND gates on its INCREMENT and DECREMENT lines. The other inputs to the AND gates are not driven from the same input, as in the normal linear ADDIE, however; the gate on the DECREMENT line is fed from the input line, on which there is a stochastic sequence with generating probability p; the gate on the INCREMENT line is fed from a constant stochastic sequence of generating probability a (representing the rate of spontaneous regeneration of the pigment).

The output of the device is the DECREMENT line to the counter, on which there is a stochastic sequence with generating probability y. If the stochastic output of the integrator has a generating probability s, the integrator has $N + 1$ states, and the unit of time is taken to be one clock interval, then we have

$$y = ps \tag{323}$$

and

$$N\dot{s} = -ps + a(1 - s) \tag{324}$$

Hence, assuming p is constant after time zero,

$$s(t) = s(0)e^{-(a+p)t/N} + [a(1 - e^{-(a+p)t/N})/(a + p)] \tag{325}$$

so that

$$y(t) = y(0)e^{-(a+p)t/N} + \{(1 - e^{-(a+p)t/N})\, p/[1 + (p/a)]\} \tag{326}$$

The steady-state solution for this equation is such that the output probability tends to $p/[1 + (p/a)]$, and hence it is half the input probability when $p = a$. When p becomes unity the output probability goes to $1/[1 + (1/a)]$; if the counter were operated asynchronously, then p could go to infinity, but the output probability would still be asymptotic to unity. Hence this element again performs a normalizing function for a wide range of inputs. However, it is interesting to note that the behavior now has a time constant, and that this is different for increasing and decreasing p. The actual time constant is $N/(a + p)$. Hence, in "dark adaption," when p has suddenly gone to azero, the time constant is N/a, whereas in light adaption, when p has become large, Ka say, the time constant is $N/(K + 1)a$. This difference in time constants corresponds to a similar effect in the time response of the human eye.

At present these remarks on the similarities between stochastic computing processes and the activity of the human nervous system are purely speculative, and no rigorous model of neuronal functioning based on stochastic computation has been investigated. However, it is apparent that stochastic computing systems have a greater similarity to neural systems than any alternative form of computer, and there is clearly scope for research to investigate the potential value of stochastic computing concepts in brain modeling.

12. SUMMARY AND CONCLUSION

It is difficult to summarize the state of the art in stochastic computing at present. This review has ranged fairly comprehensively over topics which have already been investigated in reasonable depth. Other work has not been mentioned because its scope and implications are tenuous, or because it does not have an essential constituent of purely stochastic computing. The stochastic computer has as yet had no major practical impact on data-processing systems. Equally, the analogy between nervous processes and stochastic computing has not been carried to a stage where the stochastic computer can be justified solely as a model of the central nervous system. Indeed, present justification for interest in stochastic computing is of a scientific nature—it extends the range of known data-processing systems.

The original reason for investigating stochastic computation was to find a suitable data-processing technique for machine learning and pattern recognition. In Section 2.1 the computational problems of these topics were summarized as: the processing of a large amount of data with fairly low accuracy in a variable, experience-dependent way. In Section 2.7 it was shown that the solution of these problems required a data-processing system which makes the maximum use of every single gate for computational purposes, and distributes its computations through space rather than time.

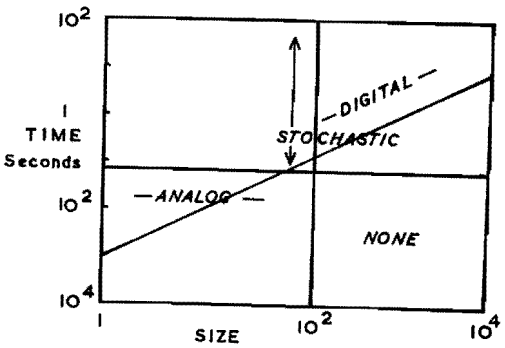

Fig. 38. Relationship between problem size and solution time for digital, analog, and stochastic computers.

In many ways the stochastic computer satisfies these requirements, and networks of stochastic computing elements offer low-cost, high-density parallel processing. In other ways the stochastic computing systems envisaged and made so far are not suitable for real computational problems. In particular, the generation of stochastic sequences by present techniques is inelegant and unsatisfactory. If true stochastic processes are utilized, e.g., from light photons or radioactive sources, then it would seem better to establish computing elements working directly with the original sources; photon/photon interactions brought about by certain electron systems would seem to offer a great potential for "natural" stochastic computation.

Although the practical application of the stochastic computer is not yet apparent, it is possible to see its place in terms of the problem-size/solution-time criteria advanced in Section 2.4. Figure 38 illustrates that since the stochastic computer is limited in speed rather than size, computations which it may perform lie above a horizontal line cutting the time axis at about 0.1 sec (for an accuracy of 1% and a clock frequency of 1 MHz). There remains an area in the lower right-hand corner which is intractable for all forms of computer, but this is now bounded by a horizontal rather than a sloping line, implying that the stochastic computer suffers no loss in speed as the size of the problem increases.

It is interesting to note that advances into the missing area of large, fast systems are largely independent of increases in the speed or size of conventional digital computers; only parallel processing can make inroads on this area. The diagram is anamalous in making it appear that analog and stochastic computers are everywhere alternatives to the digital computer. Figure 38 represents two of the constraints limiting all data-processing

systems—within these constraints many other criteria play a part in the selection of an appropriate computational technique, e.g., low cost, increased reliability, and increased flexibility give the advantage to the digital computer in many applications where an analog computer would suffice.

A combination of the three types of computer in a hybrid stochastic computer would prove very powerful in many control applications: the digital computer providing flexible sequencing and supervision of operations; the analog computer providing fast, but simple, models for iterative optimization; and the stochastic computer providing real-time identification of complex, nonlinear process dynamics. The application of complex adaptive controllers has been inhibited by lack of suitable hardware, and the ease of system identification, linear, nonlinear, and pattern classifying in the stochastic computer offers a data-processing capability which has not previously been available.

It would be wrong, however, to extrapolate from past to future system organization—even though the stochastic computer has similarities to both analog and digital computers, it is radically different from either. It is not the similarities, but the differences which are important—the computations which are most relevant to future development are those in which stochastic computing provides a simple and elegant solution to what would otherwise be a difficult problem, e.g., in the Bayesian predictor or adaptive threshold logic systems. System design is fashioned by many factors, not the least of which is ease of realization—there has been little time as yet to discover all that is simple and easy with stochastic computing elements, and virtually no time for it to influence system design. The design of learning machines and pattern recognizers which take full advantages of the properties peculiar to stochastic computers offers the greatest promise for the future development and exploitation of the systems surveyed in this chapter.

ACKNOWLEDGMENTS

Much of the work described in this chapter was supported by Standard Telecommunication Laboratories, ITT (Europe).

I would like to thank Dr. J. H. Andreae (now Professor of Electrical Engineering, University of Canterbury, New Zealand) for many fruitful discussions on machine learning and stochastic computers, and Mr. P. L. Joyce for building the stochastic computer Mk 1. I would also like to acknowledge, with much pleasure, my discussions with Professor W. J. Poppelbaum and Mr. J. W. Esch, University of Illinois, and thank them for access to their parallel research on the stochastic computer.

REFERENCES

1. W. J. Poppelbaum and C. Afuso, Noise Computer, University of Illinois, Dept. Computer Science, Quarterly Technical Progress Reports (April 1965–January 1966).
2. W. J. Poppelbaum, C. Afuso, and J. W. Esch, Stochastic Computing Elements and Systems, *in* "Proc. American Federation of Information Processing Societies, Fall Joint Computer Conference," Vol. 31, pp. 635–644, Books, Inc., New York (1967).
3. S. T. Ribeiro, Random Pulse Machines, *IEEE Trans. Electronic Computers* **EC-16** 261–276 (June 1967).
4. B. R. Gaines, Stochastic Computing, *in* "Proc. American Federation of Information Processing Societies, Spring Joint Computer Conference," Vol. 30, pp. 149–156, Books, Inc., New York (1967).
5. B. R. Gaines, Techniques of Identification with the Stochastic Computer, *in* "Proc. International Federation of Automatic Control Symposium on Identification, Prague, June 1967."
6. B. R. Gaines, Stochastic Computer Thrives on Noise, *Electronics* **40** (14), 72–79 (July 10 1967).
7. B. R. Gaines, Stochastic Computing, *in* "Encyclopaedia of Information, Linguistics and Control," pp. 766–781, Pergamon Press, New York and London (1968).

7a. J. T. Tou, Engineering Principles of Pattern Recognition, *in* "Advances in Information Systems Science," Vol. 1, J. T. Tou, ed., Plenum Press, New York (1969).

7b. K. S. Fu, Learning Control Systems, *in* "Advances in Information Systems Science," Vol. 1, J. T. Tou, ed., Plenum Press, New York (1969).

8. J. H. Andreae, Learning Machines—A Unified View, *in* "Encyclopaedia of Information, Linguistics and Control," Pergamon Press, New York and London (1968).
9. E. Feigenbaum and J. Feldman, "Computers and Thought," McGraw-Hill Book Co., New York (1964).
10. L. Uhr, "Pattern Recognition," John Wiley and Sons, New York (1966).
11. J. T. Tou and R. H. Wilcox, "Computer and Information Sciences," Spartan Books, Washington, D.C. (1964).
12. W. J. Karplus and J. A. Howard, A Transfluxor Analog Memory Using Frequency Modulation, *in* "American Federation of Information Processing Societies," Vol. 26, Part 1, pp. 673–683, Books, Inc., New York.
13. W. J. Karplus, A Hybrid Computer Technique for Treating Nonlinear Partial Differential Equations, *IEEE Trans. Electronic Computers* **EC-13** (5) 597–605 (1964).
14. B. R. Gaines and J. H. Andreae, A Learning Machine in the Context of the General Control Problem, *in* "Proc. 3rd International Congress International Federation of Automatic Control, London, 1966," Institution of Mechanical Engineers, London (1967).
15. W. J. Poppelbaum, Hybrid Graphical Processors, *in* "Computer Technology," *IEEE Conf. Publ.*, Vol. 32 (July 1967).
16. T. J. Williams, Process Dynamics, *in* "Proc. 2nd International Congress International Federation of Automatic Control, Basle, 1963."
17. G. A. Korn and T. M. Korn, "Electronic Analog and Hybrid Computers," McGraw-Hill Book Co., New York (1964).
18. J. R. Smith and C. O. Harbourt, An Adaptive Threshold Logic Gate Using Capacitive Analog Weights, *IEEE Trans. Electronic Computers* **EC-17** (1), 78–81 (1968).
19. H. Schmid, Sequential Analog–Digital Computer (SADC), *in* "American Federation of Information Processing Societies, Joint Computer Conference," Vol. 27, Part 1, pp. 915–928, Books, Inc., New York (1965).

20. D. L. Greer, Characterization of the Magnetic Second-Harmonic Memory, *IEEE Trans. Electronic Computers* **EC-17** (6), 551–558 (1968).
21. B. Widrow, An Adaptive ADALINE Neuron Using Chemical Memistors, ERL Tech. Rep. No. 1553-2, Stanford University (1960).
22. G. Nagy, A Survey of Analog Memory Devices, *IEEE Trans. Electronic Computers* **EC-12** 388–393 (August 1963).
23. S. Larach, "Photoelectronic Materials and Devices," D. van Nostrand, Englewood Cliffs, New Jersey (1965).
24. D. R. Bosomworth and H. J. Gerritsen, Thick Holograms in Photocromic Materials, *Appl. Optics* **7** (1), 95–98 (1968).
25. G. U. Kalman, Holographic Graphical Storage in Thick Alkali-Halide Crystals, *IEEE Int. Conv. Digest*, p. 35 (1968).
26. J. K. Hawkins and C. J. Munsey, Parallel Computer Organizations and Mechanizations, *IEEE Trans. Electronic Computers* **EC-12** (3), 251–262 (1963).
27. D. K. Pollock, C. J. Koester, and J. T. Tippett, "Optical Processing of Information," Spartan Books, Washington, D.C. (1963).
28. S. J. Mathis, R. E. Wiley, and L. M. Spandorfer, "Microelectronics and Large Systems," Spartan Books, Washington, D.C. (1965).
29. R. L. Petritz, Technological Foundations and Future Directions of Large-Scale Integrated Electronics, *in* "American Federation of Information Processing Societies, Fall Joint Computer Conference," Vol. 29, pp. 65–87, Books, Inc., New York (1966).
30. D. L. Slotnick, W. C. Borck, and R. C. McReynolds, The Solomon Computer—A Preliminary Report, *in* "Computer Organization," pp. 66–92, Spartan Books, Washington, DC, (1963).
31. W. A. Clark, S. M. Orstein, M. J. Stuki, A. S. Blum, T. J. Chaney, R. E. Olsen, R. A. Dammhoehler, W. E. Ball, C. E. Molnar, A. Antharvedi, Macromodular Computer Systems, *in* "American Federation of Information Processing Societies, Spring Joint Computer Conference," Vol. 30, pp. 355–401, Books, Inc., New York (1967).
32. R. H. Fuller and R. M. Bird, An Associative Parallel Processor with Application to Picture Processing, *in* "American Federation of Information Processing Societies, Fall Joint Computer Conference," Vol. 27, Part 1, pp. 105–116, Books, Inc., New York (1965).
33. J. C. Murtha, Highly Parallel Information Processing Systems, *in* "Advances in Computers," Vol. 7, pp. 1–116 Academic Press, New York (1966).
34. E. L. Braun, "Digital Computer Design," Chapter 8, Academic Press, New York (1963).
35. F. V. Mayorov and Y. Chu, "Digital Differential Analysers," Iliffe Books (1964).
36. W. J. Karplus, Analog and Digital Techniques Combined, *in* "Computer Control Systems Technology," pp. 148–155 McGraw-Hill Book Co., New York (1961).
37. Digital Operational Techniques, *Computer Design* **1963** (November), 12.
38. B. R. Gaines and P. L. Joyce, Phase Computers, *in* "Proc. 5th International Congress AICA, Lausanne, 1967."
39. B. R. Gaines, A Modular Programmed DDA for Real-Time Computation, *in* "Proc. IFIP 68, Edinburgh, 1968."
40. W. R. Schumann, Method and Apparatus for Averaging a Series of Transients, United States Patent 3, 182, 181 (May 5, 1965).
41. B. P. Th. Veltman and A. van den Bos, The Applicability of the Relay Correlator and Polarity Coincidence Correlator in Automatic Control, *in* "Proc. 2nd International Congress International Federation of Automatic Control," Basle, 1963.

42. W. W. Peterson, "Error Correcting Codes," John Wiley and Sons, New York (1961).
43. W. H. Kautz, "Linear Sequential Switching Circuits," Holden-Day, San Francisco (1965).
44. S. W. Golomb, "Shift Register Sequences," Holden-Day, San Francisco (1967).
45. A. Gill, "Finite-State Machines," McGraw-Hill Book Co., New York (1962).
46. A. Gill, "Linear Sequential Circuits," McGraw-Hill Book Co., New York (1967).
47. G. A. Korn, "Random-Process Simulation and Measurements," McGraw-Hill Book Co., New York (1966).
48. P. A. N. Briggs, P. H. Hammond, M. T. G. Hughes, and G. O. Plumb, Correlation Analysis of Process Dynamics Using Pseudo-Random Binary Test Perturbations, *in* "Advances in Automatic Control," pp. 37–51, Institution of Mechanical Engineers, London (1965).
49. R. P. Gilson, Some Results of Amplitude Distribution Experiments on Shift-Register Generated Pseudo-Random Noise, *IEEE Trans. Electronic Computers* **EC-15** (6), 926–927 (1966).
50. R. C. White, Experiments with Digital Computer Simulations of Pseudo-Random Noise Generators, *IEEE Trans. Electronic Computers* **EC-16** (3), 355–357 (1967).
51. R. E. Kalman, Nonlinear Aspects of Sampled-Data Control Systems, *in* "Proc. Symp. Nonlinear Circuit Analysis," pp. 273–313, Polytechnic Institute of Brooklyn, New York (1957).
52. M. Arbib, Tolerance Automata, *in* "Kybernetika Cislo 3," pp. 223–233 (1967).
53. W. De Backer and L. Verbeek, Study of Analog, Digital, and Hybrid Computers Using Automata Theory, *ICC Bulletin* **5**, 215–244 (1966).
54. M. Arbib, "Algebraic Theory of Machines, Languages and Semigroups," Academic Press, New York (1968).
55. K. H. Hofman and P. S. Mostert, "Elements of Compact Semigroups," Merrill Books, Columbus, Ohio (1966).
55a. J. T. Tou, ed., "Applied Automata Theory," Academic Press, New York (1968).
56. M. O. Rabin, Probabilistic Automata, *Information and Control* **6**, 230–245 (1963).
57. A. Paz, Some Aspects of Probabilistic Automata, *Information and Control* **9**, 26–60 (1966).
58. W. Feller, "An Introduction to Probability Theory and Its Applications," John Wiley and Sons, New York (1957).
59. L. Takacs, "Stochastic Processes," Methuen, London (1960).
60. R. McNaughton, The Theory of Automata—A Survey, *in* "Advances in Computers," Vol. 2, pp. 379–421, Academic Press, New York (1961).
61. J. Fox, "Mathematical Theory of Automata," Polytechnic Institute of Brooklyn Press, New York (1963).
62. T. L. Booth, "Sequential Machines and Automata Theory and Sons, New York" John Wiley (1967).
63. G. C. Bacon, Minimal-State Stochastic Finite State Systems, *IEEE Trans.* **CT-11**, 307–308 (1964).
64. G. H. Ott, Reconsider the Minimization Problem for Stochastic Finite State Systems, *in* "Proc. IEEE 7th Symp. Switching and Automata Theory," pp. 251–261 (1966).
65. G. C. Bacon, The Decomposition of Stochastic Automata, *Information and Control* **7**, 320–339 (1964).
66. C. V. Page, Equivalences between Probabilistic Machines, Tech. Rep. 03105-41-T, University of Michigan (1965).

67. J. W. Carlyle, State-Calculable Stochastic Sequential Machines, Equivalences and Events, *in* "Proc. IEEE Symp. Switching Circuit Theory and Logical Design," pp. 258–263 (1965).
68. J. von Neumann, Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components, *in* "Automata Studies," Princeton University Press, Princeton, New Jersey (1956).
69. S. Winograd and J. D. Cowan, "Reliable Computation in the Presence of Noise," MIT Press, Cambridge, Mass. (1963).
70. J. D. Cowan, The Problem of Organismic Reliability, *in* "Cybernetics of the Nervous System," Elsevier Amsterdam (1965).
71. F. Rosenblatt, A Model for Experiental Storage in Neural Networks, *in* "Computer and Information Sciences," (J. T. Tou and R. H. Wilcox, eds.), pp. 16–66, Spartan Books Washington, D.C. (1964).
72. B. Widrow and F. W. Smith, Pattern-Recognizing Control Systems, *in* "Computer and Information Sciences," (J. T. Tou and R. H. Wilcox, eds.), pp. 288–317, Spartan Books Washington, D.C. (1964).
73. K. Steinbuch and U. A. W. Piske, Learning Matrices and Their Applications, *IEEE Trans. Electronic Computers* **EC-12** (5), 846–862 (1963).
74. N. J. Nilsson, "Learning Machines," McGraw-Hill Book Co., New York (1965).
75. A. Novikoff, On Convergence Proofs for Perceptrons, *in* "Automata Theory," pp. 615–622, Polytechnic Institute of Brooklyn Press, New York (1963).
76. G. L. Clapper, Machine Looks, Listens, Learns, *Electronics* **1967** (October 30), 91–102.
77. W. C. Ridgeway, An Adaptive Logic System with Generalizing Properties, Report SEL-62-040, Stanford Electronics Laboratories (April 1962).
78. I. Aleksander and R. C. Albrow, Adaptive Logic Circuits, *Computer J.* **11** (1), 65–71 (1968).
79. S. Muroga, Lower Bounds on the Number of Threshold Functions and a Maximum Weight, *IEEE Trans. Electronic Computers* **EC-14** 136–148 (1965).
80. Sze-Tsen Hu, "Threshold Logic," University of California Press, Berkeley, Calif. (1965).
81. M. Cuenod and A. P. Sage, Comparison of Some Methods Used for Process Identification, *in* "Proc. International Federation of Automatic Control Symposium on Identification in Automatic Control Systems, Prague, June 1967."
82. P. Eykhoff, Process Parameter and State Estimation, *in* "Proc. International Federation of Automatic Control Symposium on Identification in Automatic Control Systems, Prague, June 1967."
83. M. J. Levin, Optimum Estimation of Impulse Response in the Presence of Noise, *IRE Natl. Conv. Record* **4**, 147–182 (1959).
84. W. W. Lichtenberger, A Technique of Linear System Identification Using Correlating Filters, *IRE Trans. Automatic Control* **AC-6** (2), 183–199 (1961).
85. Y. Kaya and S. Yamamura, A Self-Adaptive System with a Variable Parameter PID Control, *AIEE Trans. Appln. Ind.* **58**, 378–386 (1962).
86. M. Margolis and C. T. Leondes, A Model-Referenced Parameter Tracking Technique for Adaptive Control Systems, *IEEE Trans. Appln. Ind.* **68**, 241–261 (1963).
87. B. G. Madden, Simultaneous Determination of System Parameters from Transient Response, *IEEE Trans. Appln. Ind.* **69**, 327–331 (1963).
88. P. C. Young, The Determination of the Parameters of a Dynamic Process, *Radio and Electronic Engineer* **29** (6), 345–361 (1965).
89. D. D. Donalson and F. H. Kishi, Review of Adaptive Control System Theories and

Techniques, *in* "Modern Control Systems Theory," McGraw-Hill Book Co., New York (1965).
90. P. E. K. Donaldson, Error Decorrelation Studies on a Human Operator Performing a Balancing Task, *Med. Electron. Biol. Eng.* **2**, 393–410 (1964).
91. T. F. Potts, G. N. Ornstein, and A. B. Clymer, The Automatic Determination of Human and Other System Parameters, *in* "Proc. Western Joint Computer Conference, Los Angeles, pp. 645–660 (1961).
92. C. L. Becker and J. V. Wait, Two-Level Correlation on an Analog Computer, *IRE Trans. Electronic Computers* **EC-10** (4), 752–758 (1961).
93. Y. Lundh, A Digital Integrator for On-line Signal Processing, *IEEE Trans. Electronic Computers* **EC-12** (1), 26–28 (1963).
94. P. Jespers, P. T. Chu, and A. Fettweis, A New Method to Compute Correlation Functions, *in* "International Symp. Information Theory, Brussels, 1962."
95. B. Widrow, Statistical Analysis of Amplitude-Quantized Sampled Data Systems, *IRE Trans.* **CT-3** (1956).
96. D. G. Watts, "A General Theory of Amplitude Quantization with Applications to Correlation Determination," IEE Monograph No. 481 M (November 1961).
97. A. K. Nath and A. K. Mathalanabis, Method of Statistical Linearization, *Proc. IEE* **113** (12), 2081–2086 (1966).
98. A. A. Pervozanskii, "Random Processes in Nonlinear Control," Academic Press, New York (1965).
99. O. I. Elgard, High-Frequency Signal Injection: A Means of Changing the Transfer Characteristics of Nonlinear Elements, WESCON (1962).
100. G. R. Cooper, R. L. Gassner, and C. D. McGillem, *in* "Proc. 21st National Electronics Conference," pp. 656–661, National Electronics Conference, Chicago, Illinois (1965).
101. L. J. Savage, "The Foundations of Statistical Inference," Methuen, London (1962).
102. S. Watanabe, Information-Theoretical Aspects of Inductive and Deductive Inference, *IBM J. Res. Dev.* **14** (2), 208–231 (1960).
103. D. Middleton, "Topics in Communication Theory," McGraw-Hill Book Co., New York (1965).
104. M. E. Maron, Design Principles for an Intelligent Machine, *IRE Trans. Information Theory* **IT-8** (5), 179–185 (1962).
105. M. Minsky and O. G. Selfridge, Learning in Random Nets, *in* "Information Theory," pp. 335–347, Butterworths, London (1961).
106. K. S. Fu, A Learning Control System Using Statistical Decision Processes, *in* "Proc. International Federation of Automatic Control Symp. Self-Adaptive Control Systems, Teddington, England, 1965," Plenum Press, New York (1965).
107. D. R. Hill, An ESOTerIC Approach to Automatic Speech Recognition, *Int. J. Man–Machine Studies* **1** (1) (January 1969).
108. S. Kullback, "Information Theory and Statistics," John Wiley and Sons, New York (1959).
109. B. G. Farley and W. A. Clark, Simulation of Self-Organizing Systems by Digital Computer, *IRE Trans. Information Theory* **IT-4**, 76–84 (September 1954).
110. R. L. Beurle, Properties of a Mass of Cells Capable of Regenerating Pulses, *Trans. Roy. Soc. (London)* **B240**, 55–94 (August 1956).
111. L. D. Harmon, W. A. Bergeijk, and J. Levinson, Studies with Artificial Neurons, *Kybernetik* **1**, 89–117 (December 1961).