Reprinted from "Encyclopaedia of Linguistics, Information and Control"
PERGAMON PRESS
OXFORD · LONDON · EDINBURGH · NEW YORK
TORONTO · SYDNEY · PARIS · BRAUNSCHWEIG

COMPUTERS, STOCHASTIC (Stochastic Automata—Their Application to Computing).

1. Introduction

There is a large class of problems which are intractable for present analog and digital computers. They arise in the control of chemical plant, in the control of aircraft, missiles and spacecraft, in the simulation of economic systems, and generally in the real-time simulation and control of large, complex systems. Analog computers are not suitable for these problems because of the large number of operational amplifiers required (a reasonable upper limit at present is 400 amplifiers, sufficient to simulate about 30 non-linear, fourth-order differential equations), and particularly because of the defects, size and expense of analog multipliers and integrators. Digital computers are not suitable for these problems because their bandwidth is limited by the use of a central processor to compute sequential solutions of differential equations. Williams (1965) has reviewed these difficulties in the context of process control and demonstrated the impossibility of simulating even a simple, linearized model of a multi-plate distillation column in real time with presently available computers. His diagram illustrating the applicability of digital and analog computers to processes of different orders (number of first-order differential equations) and time-constants is reproduced in Fig. 1. As Williams remarks, 'It is truly unfortunate to note the number of chemical process systems which fall into the lower right hand corner of Fig. 1', and a similar remark may be applied to many other systems of great practical importance.

There have been many attempts to design computers which combine the parallel operation of analog computers with the simplicity and accuracy of digital computers. Parallel digital differential analysers (Leondes 1959) using incremental digital stores to simulate the analog integrator (serial DDA's offer only economic advantages over the general purpose digital computer), and hybrid computers (Connelly 1962; Truitt 1964) in which a digital computer controls the operation of an analog computer have been the most successful to date, but their computing elements are as complex as those of conventional computers and it remains economically infeasible to simulate large, complex systems.

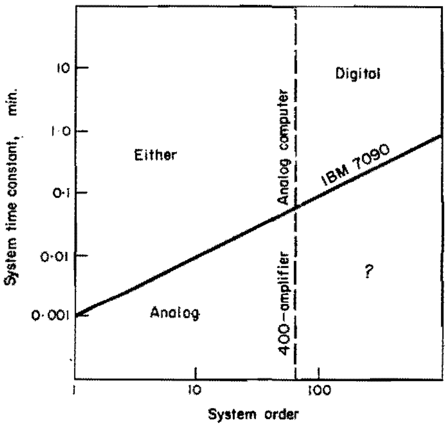


FIG. 1. *Applicability of analog and digital computers to process simulation.*

The main performance measures of a computer are size and range of possible problems, speed and accuracy of solution, and the physical size, reliability and cost of the computer. There are strong interactions between these measures and it is unlikely that any one form of computer will ever be optimal on all counts. The simulation of complex plant and multivariable control systems requires large numbers of computing elements such as multipliers, summers and integrators, working simultaneously and costing little. However, these elements do not have to compute a solution quickly or accurately, for a bandwidth of 10 c/s and an overall accuracy of 1 per cent is adequate in the simulation of economic and chemical processes, and in control systems where feedback operates a computational accuracy of 10 per cent may be ample. In these situations it should be possible to trade the accuracy of the digital computer and the speed of the analog computer for computing size and economy, and the stochastic computer is the most promising attempt to do this so far.

The stochastic computer was developed for the type of problem where the availability of large numbers of low-cost computing elements is more important than the speed and accuracy of computation. Information within the computer is carried through modulation of the statistics of digital 'noise', and the theory underlying its computational processes is that of Markov chains and stochastic automata. Research on the latter has been oriented towards their applications in the design of reliable digital computers and the literure is heavily biased to this point of view. The next section follows the evolution of stochastic automata as a natural derivative of state-determined machines, and outlines the main theoretical results. A brief indication is given of how the theory may be applied to problems of computer reliability, but the main section of this review describes the positive, rather than preventive, applications of stochastic automata to computing.

2. Digital Computer Theory

The basic mathematical object used to represent the behaviour of a digital computer is the finite-state automaton (McNaughton 1961; Rabin and Scott 1959). 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 sub-set of states and the sub-set 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 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 behaviour 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 (statistics of 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 in the next section.

2.1. 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 one 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 a one if that input gives rise to the jth output when the automaton is in its ith state. Since a state and input give rise to a unique output and next state, these matrices are characterized by having a single one 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 behaviour determined by matrix multiplication.

Let the set of possible inputs to an automaton be:

$$(I_i), \quad 0 < i \leqslant N;$$

the set of possible states be:

$$(S_i), \quad 0 < i \leqslant M;$$

and the set of possible outputs be:

$$(O_i), \quad 0 < i \leqslant P.$$

Let the state of the automaton be S, its ouput O, its input I, and the state after the next clock pulse S'. We have the transition mapping:

$$S' = \sigma(S, I);$$

and the output mapping:

$$O = \theta(S, I).$$

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

$$p^k_{ij} = \begin{cases} 1 \Leftrightarrow S_j = \sigma(S_i, I_k) \\ 0 \qquad \text{otherwise.} \end{cases}$$

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

$$o^k_{ij} = \begin{cases} 1 \Leftrightarrow O_j = \theta(S_i, I_k) \\ 0 \quad \text{otherwise.} \end{cases}$$

The existence and uniqueness of the output and next state imply:

$$\sum_{j=1}^{M} p^k_{ij} = 1$$

$$\sum_{j=1}^{P} o^k_{ij} = 1.$$

Having defined the transition and output matrices, we may consider the behaviour 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:

$$(P^{\lambda_1}P^{\lambda_2}P^{\lambda_3} \ldots P^{\lambda_n})_{ij} = 1,$$

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

$$(P^{\lambda_1}P^{\lambda_2} \ldots P^{\lambda_n}O^{\lambda_n})_{ij} = 1.$$

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.

2.2. 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, or expectation of a given state, 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:

$$(C_i) \quad 0 < i \leqslant Q,$$

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:

$$\sum_{j=1}^{N} \nu_i^j = 1.$$

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:

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

Thus if the hyperstate of the automaton is the distribution (Π_i), where Π_i is the probability that the automaton is in the state S_i, then the hyperstate (Π_i') after a control C_k is given by:

$$\Pi_j' = \sum_{i=1}^{M} \Pi_i \pi_{ij}^k = \sum_{i=1}^{M} \Pi_i \sum_{r=1}^{N} \nu_k^r p_{ij}^r.$$

The matrices $\Pi^k \equiv (\pi_{ij}^k)$ are probability matrices since their elements are non-negative and:

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

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.

2.3. Markov chains. The discrete transitions of a system from condition to condition are said to form a Markov chain (Takács 1960; Kemény and Snell 1960) 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 form 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 states and 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 of processing, and stochastic computing elements are designed to receive and emit Bernoulli sequences.

The relationship between the average behaviour of an ensemble of stochastic automata, which is the conceptual basis for their operation, and the average behaviour 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, non-ergodic behaviour, and often act as 'solutions' in a computation.

2.4. Computer reliability. This section is a digression from the main theme but the literature on stochastic automata is concerned mainly with computer reliability and the results obtained must be considered relative to this. 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 pre-determined 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 (1963) and Paz (1966) 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 automaton may be less economical in storage, requiring more internal states 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 its practical applications, for example, in the 'Enhancetron', a waveform averager described in Section 3.1. 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.

Whilst the statistical approximation to the behaviour of an unreliable computer through stochastic automata reduces the large input set to a smaller control set, it leaves the state set unchanged. This is a serious defect, since consideration of the states arising through faults in a computer, especially one with much redundancy, leads to a very large state set and an unwieldy automaton. One would like to group the states of the automaton in some way, but so doing generally leads to a Markov chain which is not of first order and hence equally unwieldy. Keméný and Snell (1960) call a transition matrix 'lumpable' if states can be grouped without raising the order of the chain, but the conditions for lumpability are highly restrictive. Pierce (1965) has considerably generalized their results by obtaining probability bounds on the hyperstates after grouping even when the matrix is not lumpable.

The first application of the theory of stochastic automata to computer design was that of von Neumann (Shannon and McCarthy 1956) 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 (1963) 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 brain-like, homeostatic artifacts, than of new developments in conventional computers (Cowan 1965).

3. The Use of Noise in Data-Processing

The application of the theory of stochastic automata to the design of reliable computers has been purely negative—the stochastic properties are a defect to be overcome rather than an essential feature of the computer. The emphasis of the remainder of this review is on the converse situation where probabilistic behaviour is used constructively and randomness is essential to the performance of the computer. Before introducing the general notion of stochastic representations of quantity, two simple examples are given of the advantageous introduction of noise into computers.

3.1 Round off error 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^{-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:

$$-\mathrm{FSR}/2^{N+1} < \varepsilon < +\mathrm{FSR}/2^{N+1},$$

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

Consider now the effect of adding to the input voltage another, 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 the probability that the least significant digit will be set in random conversion. Thus there will 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, $0 \leq E < \mathrm{FSR}/2^N$. The LSD is one if $E+V$ exceeds $\mathrm{FSR}/2^{N+1}$, and since V is evenly distributed over its range:

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

Thus the expected value represented by the LSD is:

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

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 a variance:

$$\sigma^2 = \frac{|\varepsilon|\ (\mathrm{FSR}/2^N - |\varepsilon|)}{4M}\,.$$

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', (Schumann 1965) 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 twelve-bit digital form, and add this into a digital store corresponding to the particular sampling instant. The Enhancetron 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. 2 illustrates its operation: the incoming wave-

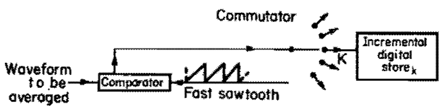


FIG. 2. *Principle of 'Enhancetron'.*

form is compared with a sawtooth (simulating a uniform random distribution) in a comparator; the output is commutated around a 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 repetetive signal whose phase is locked to their cycling rate. Thus the use of random conversion has replaced a twelve-bit analog to digital convertor by a simple comparator, and replaced twelve-bit binary addition by simple incrementing/decrementing; both considerable economies.

3.2. The polarity coincidence correlator (Veltman and Van den Bos 1965). 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 co-variance:

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

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 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. If:

$$P_T = \frac{1}{T}\int_0^T \operatorname{sgn}(U(t))\operatorname{sgn}(V(t))\,dt,$$

then as $T \to \infty$ $C_T \to a\sin\left(\frac{\pi}{2}P_T\right)$, where a is a constant dependent on the variance of U and V.

This non-linear 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 $(V+B)$:

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

Then it may be shown that as $T \to \infty$:

$$C_T \to a\lim_{N\to\infty} P_{T,N}, \qquad \text{where } a \text{ is a constant.}$$

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. 3: the random waveforms are again approximated by very

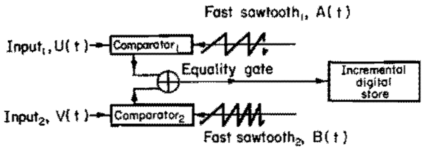

Fast sawtooth$_1$, A(t)
Input$_1$, U(t)
Comparator$_1$
Equality gate
Incremental digital store
Input$_2$, V(t)
Comparator$_2$
Fast sawtooth$_2$, B(t)

FIG. 3. *Polarity coincidence correlator.*

fast anharmonic sawtooths feeding inputs to comparators, on the other side of which are the waveforms to be correlated; 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. The full exploitation of these advantages in the stochastic computer depends on the use of general stochastic representations of quantity, and these are described in the next section.

4. *Stochastic Representation of Quantity*

The basic principle of the stochastic computer is to use the generating probability of Bernoulli sequences of logic levels to carry information. In the first example above, the probability that the least significant digit would be set was proportional to the voltage to be represented by the LSD, and hence the LSD's in a sequence of readings form a stochastic representation of this voltage. This is called the 'asymmetric binary representation' and its most general form is:

given any quantity E in the range $0 \leqslant E \leqslant V$, represent it by a Bernoulli sequence of binary logic levels with generating probability:

$$p(\text{ON}) = E/V.$$

Thus the maximum voltage is represented by a logic level always ON, zero voltage by it being always OFF, and values in between by some probability that it will be ON.

A second representation was exemplified by the polarity coincidence correlator where both positive and negative voltages had to be represented. The representation used was an example of the 'symmetric binary representation' whose most general form is:

given any quantity in the range $-V \leqslant E \leqslant +V$, represent it by a Bernoulli sequence of binary logic levels with generating probability:

$$p(\text{ON}) = \frac{E+V}{2V} = \tfrac{1}{2}E/V+\tfrac{1}{2}.$$

Thus the maximum positive voltage is represented by a logic level always ON, maximum negative voltage by it being always OFF, and zero voltage by a random sequence with equal probability of ON or OFF.

Many other forms of stochastic representation are possible, and those which have definite representations of infinite quantities are especially interesting. A binary representation may be regarded as a mapping from a line, the range of an analog variable, to the interval [0, 1], the range of probabilities. There is a natural 'distance' between two probabilities, defined by the reciprocal of the length of experiment necessary to distinguish between them. This gives the natural topology of the interval to the range of probabilities, so that it is usual for the mapping of an analog variable into this range to be monotonic, if not continuous. The computing elements described in the remainder of this paper operate in the binary symmetric representation.

4.1. Comparison of stochastic and other forms of representation. Stochastic representations of quantity may be compared with those which characterize other forms of computer. The most direct representation is that of the analog computer where continuous voltage levels replace analog quantities. In the digital computer a quantity is represented by an ordered set of binary logic levels or binary word. The DDA and pulse-counting computers represent a quantity by the number of ON logic levels occurring during an interval. In the stochastic computer only the probability of occurrence of a logic-level carries information.

If one compares the efficiency of these representations, in terms of the number of levels required to carry information to a precision of one part in N, then:

the analog computer requires one continuous level;
the digital computer requires $\log_2 kN$ ordered binary levels;
the pulse-counting computer or DDA requires kN unordered binary levels;
and the stochastic computer requires kN^2 unordered binary levels—where $k > 1$ is a constant representing the effects of round-off error or variance, $k = 10$ say. This progression from $1 : \log_2 N : N : N^2$ shows the stochastic computer to be by far the least efficient in its representation of quantity, but it is through this inefficiency that it gains in simplicity and economy. In the polarity coincidence correlator, for example, a simple equality gate performs multiplication, whereas multiplication in the analog computers requires several operational amplifiers—in the digital computer a complex central processor and sub-routine—and in the DDA two integrators. The next sections describe stochastic computing elements to perform the complete range of analog computing functions, inversion, multiplication, addition, integration and so on.

5. *Stochastic Computing Elements*

The elements of a stochastic computer consist of logic units, gates, and storage devices, together with random generators whose outputs are Bernoulli sequences of logic levels. For convenience in exposition the stochastic computer will be assumed to run synchronously, so that the inputs, outputs and states of elements within the computer change only at the occurrence of a 'clock pulse'. The properties of the most common binary elements are shown in Fig. 4, together with their graphic symbols. The logical computations performed by the computing elements induce arithmetical computations in the quantities represented by their input lines, and these depend on the particular representation in use.

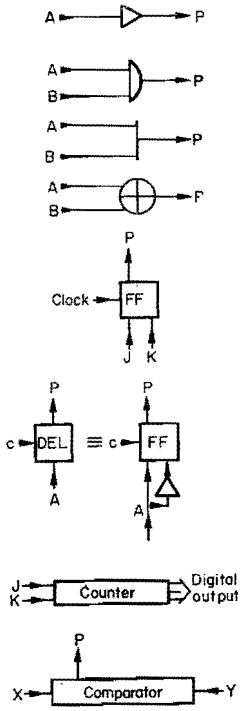
5.1. Stochastic invertors, multipliers and isolators. Consider the simple logical invertor (Fig. 4a) whose output is the complement of its input. The relationship between the probability that its input will be ON and the probability that its output will be ON is:

$$p_{\text{out}}(\text{ON}) = 1-p_{\text{in}}(\text{ON}).$$

In the binary symmetric representation the relationship between these probabilities and the quantities they represent is:

$$p_{\text{in}}(\text{ON}) = \tfrac{1}{2}+\tfrac{1}{2}E_{\text{in}}/V$$
$$p_{\text{out}}(\text{ON}) = \tfrac{1}{2}+\tfrac{1}{2}E_{\text{out}}/V,$$

hence $$E_{\text{out}} = -E_{\text{in}},$$



(a) Logical Invertor $P = \bar{A}$, the output is the complement of the input.

(b) AND Gate $P = A.B$, the output is ON $\Longleftrightarrow$ both inputs are ON.

(c) OR Gate $P = A \wedge B$, the output is ON $\Longleftrightarrow$ either or both inputs are ON.

(d) Equality Gate $P = A.B \wedge \bar{A}.\bar{B}$, the output is ON if the inputs are the same.

(e) JK Flip-Flop The output is the same as the internal state of the flip-flop. At a clock pulse the state becomes ON if the J line alone is ON, and OFF if the K line alone is ON. It complements if they are both on, and remains the same if they are both OFF.

(f) Unit Delay $P(t) = A(t-1)$, the JK flip-flop is connected so that its output reproduces its input delayed through one clock pulse.

(g) Counter A store with states $S_0 \ldots\ldots S_N$; in state S_k at a clock pulse, it goes to state S_{k+1} if its J line is on alone, and to state S_{k-1} if its K line is on alone; otherwise it remains in the same state.

(h) Comparator $P \equiv (X \geqslant Y)$, the output is determined by the relative magnitude of the (analog or digital) quantities at the inputs.

FIG. 4. *Properties of binary elements used in the stochastic computer.*

so that the logical invertor acts to give the negative of a number (in other representations it may subtract the number from a constant or compute its reciprocal).

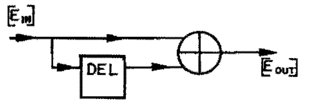
The equality gate (Fig. 4d) is used to carry out multiplication in the binary symmetric representation. Consider the two input streams to represent quantities E_{in}, E'_{in}, by probabilities p_{in}, p'_{in}. The output is ON when the inputs are both ON or both OFF, and hence:

$$p_{out}(ON) = p_{in}(ON)p'_{in}(ON) + [1-p_{in}(ON)][1-p'_{in}(ON)]$$

so that

$$E_{out} = E_{in}E'_{in}/V.$$

An important phenomenom is illustrated by the use of a stochastic multiplier as a squarer. It is not sufficient to short-circuit the inputs of the equality gate together, for its output will then always be ON. This difficulty arises because the multiplier inputs must be statistically independent if the probability of their conjunction is to be the product of their generating probabilities. Fortunately an independent replication of a Bernoulli sequence may be obtained by delaying it through one event, and Fig. 5 illustrates a squarer utilizing a flip-flop as a delay, (Fig. 4f), and an equality gate as a multiplier. The flip-flop used in this way is a stochastic 'isolator', performing no computation but statistically isolating two cross-correlated lines.



$$E_{OUT} = E_{IN}^2/V$$

FIG. 5. *Stochastic squarer.*

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; that is they are neither cross-correlated nor autocorrelated. Both these conditions may be neglected to advantage in pseudo-random realizations of the computer, but isolation must normally be used to maintain statistical independence.

5.2. Stochastic addition. Having seen how readily inversion and multiplication are effected by simple gates, one is tempted to assume that a similar gate may be used to perform addition. However, this is not so and a stochastic logic element must be introduced to effect addition in the symmetric binary representation. Consider the situation when one input is always ON, representing the maximum positive quantity, and the other is always OFF, representing the maximum negative quantity. The sum of the quantities represented by the inputs is zero, and hence the output must be random, with equal chances of being ON or OFF. A probabilistic output cannot be obtained from a deterministic gate with constant inputs, so that stochastic behaviour must be built into the gate itself.

One realization of a stochastic adder in the binary symmetric representation is illustrated in Fig. 6: flip-flop$_1$ is in a random state which is transferred to flip-

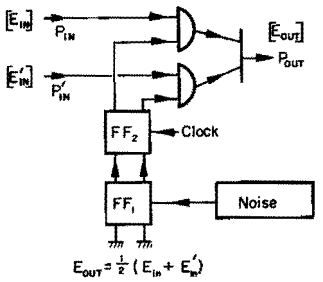


FIG. 6. *Stochastic adder.*

flop$_2$ at a clock pulse; dependent on the state of flip-flop$_2$, one or the other of the input lines is reproduced at the output. Thus the probability that the output will be ON is half the sum of the generating probabilities of the inputs:

$$p_{\text{out}}(\text{ON}) = \tfrac{1}{2}p_{\text{in}}(\text{ON}) + \tfrac{1}{2}p'_{\text{in}}(\text{ON}),$$

and hence:

$$E_{\text{out}} = \tfrac{1}{2}(E_{\text{in}} + E'_{\text{in}}).$$

A multi-input adder may be realized by reproducing any one of its inputs at random as the output, and a weighted adder may be realized by biasing the probabilities that given inputs will be reproduced at the output.

5.3. Stochastic integrators, the ADDIE, and Interface. Integration in the Polarity Coincidence Correlator (Fig. 3) is realized by a counter which increments its count by unity when its input is ON, and decrements it by unity when its input is OFF. However, the counter itself has the third possibility of not changing its count, and this is used to advantage in the two-input summing integrator illustrated in Fig. 7: the counter changes its state at a clock pulse only if the 'hold' line is ON; it

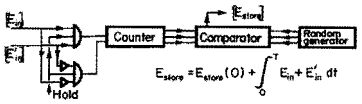


FIG. 7. *Stochastic two-input summing integrator.*

increments by unity if both its inputs are ON, and decrements by unity if they are both OFF; otherwise it remains in the same state. With both inputs joined together the two-input integrator behaves as the incremental digital store previously described.

A stochastic output from the integrator, representing the stored quantity, is obtained by comparing its count with a randomly varying digital quantity, uniformly distributed over the range of the store. If the store has $N+1$ possible states, $S_0 \ldots . S_N$, then when it is in state S_k the probability that its output will be ON at th next clock pulse is:

$$p_{\text{out}}(\text{ON}) = k/N.$$

The integrator with unity feedback illustrated in Fig. 8b is called an ADDIE, and performs the important

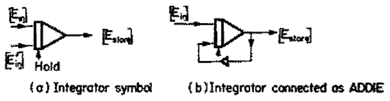


FIG. 8.

function of exponentially averaging the quantity represented by its input line. The count in its store is an unbiased estimator of the generating probability of its input, and it may be read out in analog or digital form as the natural outward interface of the stochastic computer. The count in an $N+1$ state ADDIE, with an input of constant generating probability, p, is approximately normally distributed with a variance:

$$\sigma^2 = p(1-p)N,$$

and hence the standard deviation of the quantity represented, $\sigma(E)$, expressed as a fraction of the range is:

$$\frac{\sigma(E)}{|V|} = \frac{1}{N^{\frac{1}{2}}}\left[1-\left(\frac{E}{V}\right)^2\right]^{\frac{1}{2}} \leq \frac{1}{N^{\frac{1}{2}}}.$$

Thus quantities within the stochastic computer may be read out to any required accuracy by using ADDIE's with many states, but the more states the longer the time-constant of smoothing and the lower the bandwidth of the computer. Hence variables within the computer may be regarded as degraded by Gaussian noise, whose power increases with the bandwidth required from the computer.

The integrator or ADDIE may be used to realize arbitrary functions by imposing suitable non-linear relationships between the count in its store and stochastic output. For example if all counts above the mid-level give rise to an ON output, and all those below it give rise to an OFF output, then the input/output relationship approximates to a discontinuous relay or switching-function. Integrators with their hold lines OFF may be used to store a constant during integration, and thus act as a 'potentiometer' if coupled to a multiplier.

The inward interface of the stochastic computer consists of comparators, one side of which accepts the analog or digital input, and the other side of which is driven by a random analog or digital waveform; these have been described for both the 'Enhancetron' and the polarity coincidence correlator. In the binary symmetric representation the random waveform will have a uniform distribution over the range of the inputs, symmetrical about zero; other representations may be obtained by transforming the input or biasing the random distribution.

6. *Applications of Stochastic Computing Elements*

The stochastic computing elements described form a complete set in the binary symmetric representation: invertor, adder, multiplier, integrator, function generator, inward and outward interfaces. These may be made available through a patch-board identical to that in conventional analog computers, so that the elements may be interconnected at will to simulate specific systems. Stochastic computing elements are constructed of such simple digital circuits, however, that many elements can be fabricated on a single chip of an integrated circuit, and to take full advantage of the compactness and economy offered it is desirable for interconnexions between elements to should be fabricated on the same chip. Thus practical computing units tend to be fairly large complexes of the basic elements interconnected to perform a specific computation. Some simple examples of such complexes are given in the concluding sections.

6.1. A second-order transfer function. A stable second-order transfer function with variable natural frequency and damping-ratio is shown in Fig. 9: two integrators

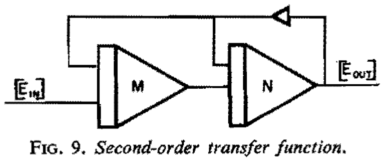


FIG. 9. *Second-order transfer function.*

are connected in series, and inverted feedback from the final output is taken to both their inputs. If the first integrator has $(M+1)$ states and the second has $(N+1)$ states, then the transformation realized is:

$$\frac{MN}{f^2}\ddot{E}_{\text{out}}+\frac{M}{f}\dot{E}_{\text{out}}+E_{\text{out}}=E_{\text{in}},\qquad \text{where } f \text{ is the clock frequency.}$$

This is a stable second-order transfer function with:

$$\text{undamped natural frequency} = \frac{f}{2\pi(MN)^{\frac{1}{2}}},$$

$$\text{damping ratio} = \tfrac{1}{2}(M/N)^{\frac{1}{2}}.$$

A simulated response of this stochastic computing unit to a step in position and velocity is shown in Fig. 10, for $M = N = 2000$, corresponding to a damping

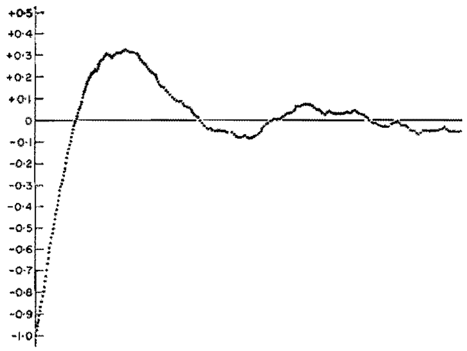


FIG. 10. *Response of stochastic second-order transfer function.*

ratio of $\frac{1}{2}$. Gaussian noise, superimposed on the normal overshoot and oscillatory decay, can clearly be seen.

6.2. Steepest descent computer. The most ubiquitous of computing configurations is that of 'steepest descent', (Donaldson and Kishi 1965; Tou and Wilcox 1964) and its uses range from implicit four-quadrant division to parameter identification and pattern recognition. Its usage is normally restricted by the number of multipliers and storage elements required for its implementation, but these requirements make it ideal for stochastic realization.

It is required to find weights, $w_1 \ldots . w_N$, for inputs, $x_1 \ldots . x_N$, which give the best approximation (in terms of r.m.s. error) to the required output, y, by a weighted sum of the inputs, z. That is to find $\{w_i\}$ such that:

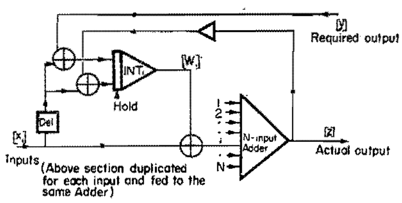
$$(\overline{e^2}) = \frac{1}{T}\int_0^T (z-y)^2\,\mathrm{d}t \qquad \text{is minimized,}$$

$$\text{where}\quad z = w_1x_1+w_2x_2+\ldots\ldots+w_Nx_N.$$

It may be shown that the best values of the weights can be established on-line through the computation (assuming no secondary minima):

$$\dot{w}_i = \alpha x_i(y-z),\quad \text{where}\quad \alpha > 0 \text{ is the slope of descent.}$$

A stochastic computer realizing this computation is shown in Fig. 11: the weights are represented by the outputs of integrators (only one section is shown), whose inputs receive feedback from the error between actual and required output which is multiplied by the



[Y] Required output
[W]
INT.
Hold
Del
[X]
Inputs (Above section duplicated for each input and fed to the same Adder)
N-input Adder
1
2
N
[Z] Actual output

FIG. 11. *Steepest descent computer.*

appropriate input. When the 'hold' lines of the integrators are ON the computer makes a steepest descent approach to the best linear relationship between inputs and required output, and when the 'hold' lines are OFF the 'actual output' simulates the 'required output' by using this relationship. Note the need for isolation of the input feedback, and the use of three multipliers for each weight. Stochastic computing elements allow by far the most economical realization of the steepest descent computer.

6.3. Partial differential equations and neural nets. 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 simulate the equations by spatial arrays of special-purpose computing elements, with one spatial dimension for each independent variable, and these arrays bear an interesting resemblance to 'neural nets'. A stochastic solution of Laplace's equation will be used to illustrate this technique.

Laplace's equation in two dimensions:

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2} = 0, \qquad \text{with boundary}$$

conditions on a closed curve, may be simulated by expressing it in discrete form:

$$u(x-\varepsilon, y)-2u(x, y)+u(x+\varepsilon, y)+u(x, y-\varepsilon)-2u(x, y)+u(x, y+\varepsilon) = 0.$$

If $u(x, y)$ is represented as the output of a stochastic integrator, then the above relationship can be enforced by feedback to its input, assuming other integrators are representing neighbouring grid-points. A suitable computing element, the 'Laplacian', is shown in Fig. 12: an 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. 13, can be used to solve Laplace's equation in two-dimensions for arbitrary shapes and boundary conditions.

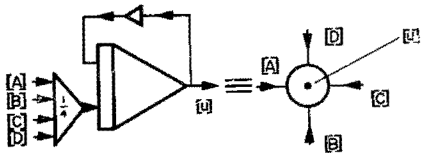


FIG. 12. *Two-dimensional Laplacian element.*

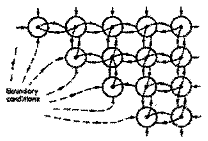


FIG. 13. *Net of Laplacians.*

The properties of these spatial arrays have interesting resemblances to those of the 'neural nets' simulated by Farley and Clark (1954) and Beurle (1956), and the computing elements themselves are reminiscent of the 'neurons' of Harmon (1961) and other workers. Neurons in the brain are known to show stochastic behaviour, 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; for example, the cross-correlational processes of visual disparity and auditory formant separation.

7. *Summary and Conclusions*

The place for the stochastic computer in the diagram of Fig. 1. should now be clear, and is shown in Fig. 14.

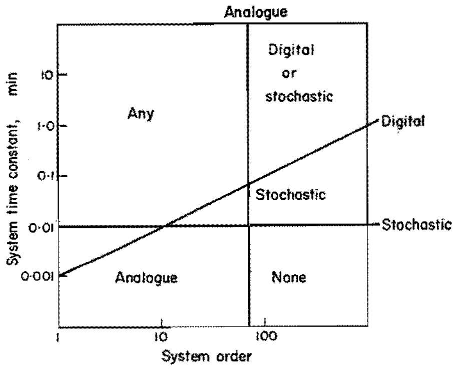


FIG. 14. *Applicability of analog, digital and stochastic computers to process simulation.*

The main limitation is its speed, and systems suitable for simulation by a stochastic computer lie above a horizontal line cutting the time axis at approximately 0·01 minutes. 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 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 digital computers; sequential processing is a grave disadvantage in the simulation of systems with many interacting components.

The diagram is anomalous in making it appear that analog and stochastic computers are everywhere alternatives to the digital computer. Figure 14 represents the physical constraints in real-time process simulation, and neglects the many applications where the digital computer proves cheaper, more flexible and easier to use. A combination of the three types of computer in a hybrid stochastic computer would prove very powerful in 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, realistic simulation for identification of process states and dynamics.

The application of complex adaptive controllers has been inhibited by the lack of suitable hardware, and the ease of multiplication and storage in the stochastic computer offers opportunities for parameter optimization which have not previously been available. The development of learning machines which take full advantage of the properties peculiar to stochastic hardware, and the development of stochastic computing elements to fulfil the particular requirements of learning machines are potentially the richest fields of research in both computation and control.

Acknowledgement. Part of this review is based on unpublished work by Dr. J. H. Andreae and the staff of the Learning Machines Department at Standard Telecommunication Laboratories, and the author would like to thank them for their co-operation, and STL for permission to publish this material.

See also: Automata, finite-state. Control by hybrid computers. Control, identification techniques for. Neuronal nets. Reliable commutation with unreliable elements.

Bibliography

BEURLE (1956) *Properties of a mass of cells capable of regenerating pulses, Trans. Roy. Soc.,* **B240,** Aug.

BLAKELOCK (1965) *Automatic Control of Aircraft and Missiles,* New York: Wiley.

CONNELLY (1962) *Real-time analog-digital computation, Trans. I.R.E, EC-***11,** (1) Feb., 31.

COWAN J. (1965) *The problem of organismic reliability,* in *Progress in Brain Research,* Vol. 17, Amsterdam: Elsevier.

DONALDSON and KISHI (1965) *Review of adaptive control systems theories and techniques,* in *Modern Control Systems Theory,* New York: McGraw-Hill.

FARLEY and CLARK (1954) *Simulation of self-organizing systems by digital computer, Trans. I.R.E.,* **IT-4,** Sept., 76.

HARMON (1961) *Studies with artificial neurons, Kybernetik,* **1,** Dec., 89.

KEMENY and SNELL (1960) *Finite Markov Chains,* New York: Van Nostrand.

KORN and KORN (1964) *Electronic Analog and Hybrid Computers,* New York: Van Nostrand.

LEONDES (1959) *Digital techniques in analog computation,* in *Handbook of Automation, Computation and Control,* New York: Wiley.

McNAUGHTON (1961) *The theory of automata—a survey,* in *Advances in Computers,* Vol. 2, New York: Academic Press.

MOORE (1964) *Sequential Machines,* New York: Addison-Wesley.

PAZ (1966) *Some aspects of probabilistic automata, Inf. and Control,* 9, (1), Jan., 26.

PIERCE (1965) *Failure-tolerant Computer Design,* New York: Academic Press.

RABIN (1953) *Probabilistic automata, Inf. and Control,* 6 (3), Sept. 230.

RABIN and SCOTT (1959) *Finite automata and their decision problems, I.B.M.J. Res. and Dev.,* 3 (2), April, 114.

SHANNON and McCARTHY (1956) *Automata studies, Annals of Mathematics,* **34,** Princeton: The University Press.

SCHUMANN (1965) *Method and apparatus for averaging a series of electrical transients,* U.S. Patent 3, 182, 181.

TAKACS (1960) *Stochastic Processes,* London: Methuen.

TOU and WILCOX (1964) *Computer and Information Sciences,* New York: Spartan Press.

TRUITT (1964) Hybrid Computation . . . What is it? Who needs it? *Proc. Spring Joint Computer Conf.,* 249.

VELTMAN and VAN DEN BOS (1965) *The applicability of the relay-correlator and polarity coincidence correlator in automatic control, Proc. 2nd Congress. Intern. Fed. Automatic Control,* London: Butterworths.

WILLIAMS (1965) *Process dynamics and its application to industrial process design and process control, Proc. 2nd Congr. Intern. Fed. Automatic Control,* London: Butterworths.

WINOGRAD and COWAN (1963) *Reliable Computation in the Presence of Noise,* Cambridge, Mass.: M.I.T. Press.

B. R. GAINES