[5] E. L. Lawler, "Minimal Boolean expression with more than two levels of sums and products," 1962 *Proc. 3rd Ann. Symp. on Switching Circuit Theory and Logical Design*, pp. 50–59.
[6] G. A. Maley and J. Earle, *The Logic Design of Transistor Digital Computers*. Englewood Cliffs, N. J.: Prentice-Hall, 1963, ch. 6.
[7] D. L. Dietmeyer and P. R. Schneider, "A Computer-oriented factoring algorithm for NOR logic design," *IEEE Trans. Electronic Computers*, vol. EC-14, pp. 868–874, December 1965.
[8] E. J. McCluskey, Jr., "Logical design theory of NOR gate networks with no complemented inputs," *1963 Proc. 4th Ann. Symp. on Switching Circuit Theory and Logical Design*, pp. 137–148.
[9] J. F. Gimpel, "The minimization of TANT networks," *IEEE Trans. Electronic Computers*, vol. EC-16, pp. 18–38, February 1967.
[10] D. T. Ellis, "A synthesis of combinational logic with NAND or NOR elements," *IEEE Trans. Electronic Computers*, vol. EC-14, pp. 701–705, October 1965.
[11] L. Hellerman, "A catalog of three-variable OR-INVERT and AND-INVERT logical circuits," *IEEE Trans. Electronic Computers*, vol. EC-12, pp. 198–223, June 1963.
[12] R. A. Smith, "Minimal three-variable NOR and NAND logic circuits," *IEEE Trans. Electronic Computers (Short Notes)*, vol. EC-14, pp. 79–81, February 1965.
[13] J. P. Roth, "Systematic designs of automata," *1965 Fall Joint Computer Conf., AFIPS Proc.*, vol. 27, pt. 1. Washington, D. C.: Spartan, 1965, pp. 1093–1100.
[14] R. E. Miller, "Combinational circuits," vol. 1 in *Switching Theory*. New York: Wiley, 1965.
[15] J. P. Roth, "Algebraic topological methods for the synthesis of switching systems: I," *Trans. Am. Math. Soc.*, vol. 88, pp. 301–326, July 1958.
[16] D. L. Dietmeyer and P. R. Schneider, "Identification of symmetry, redundancy, and equivalence of Boolean functions," *IEEE Trans. Electronic Computers*, vol. EC-16, pp. 804–817, December 1967.
[17] ——, "Combinational switching function sub-routine set," 9.4.062, COMMON library, IBM Corp. DP Program Information Dept., Hawthorne, N. Y., July 1966.
[18] Y. H. Su and D. L. Dietmeyer, "Computer reduction of two-level multiple-output switching circuits," this issue, pp. 58–63.

A Methodical Approach to Analyzing and Synthesizing a Self-Repairing Computer

DOUGLAS C. DORROUGH

Abstract—The literature on computer reliability is replete with very convincing arguments for the need and the use of self-repair techniques, as a viable approach to significantly enhancing the reliability of both maintainable and nonmaintainable computers. However, it would seem that no comprehensive and coherent program for the development and optimal employment of such techniques exists. This means that no method exists in the open literature for deciding the following: 1) what self-repair techniques, taken singularly or in combination, provide the greatest improvement in reliability; 2) what methods are optimum for initiating fault diagnosis and self-repair by redundancy and replacement; 3) what constitutes a closed set of self-repair techniques and what theory can be formulated to demonstrate the set's completeness; and 4) what is the effect of self-repair on the total system relative to design, maintenance, availability, and so forth.

Although several investigators have addressed themselves to some of the subproblems presupposed by items 1) through 4), no one seems to have considered the possibility of resolving such problems by the development and methodical employment of a comprehensive systems-effectiveness measure. Such an approach, together with some of its results, is described in this paper.

Index Terms—Configurational redundancy, fault diagnosis, functional redundancy, optimally reliable computing systems, reliability, self-repair, self-repairing systems, system-effectiveness measure.

Manuscript received July 20, 1967. This work was supported by Douglas Aircraft Company, Inc., under Program Account 80271-019.
The author was with McDonnel Douglas Corporation, Advanced Electronics Department, Santa Monica, Calif. He is now with the Department of Philosophy and Computer Science, Louisiana State University, Baton Rouge, La. 70803

1. Statement of the Problem

IN THE literature on system theory, the concept of self-repair is not new. It has been proposed and discussed by numerous individuals (see, for example, [1]–[16]). Research in this area is presently pursued by several groups within industrial and university environments. In general, these research efforts are along two distinct lines. One line regards self-repair as a regenerative process in which a failing substructure is partially or completely renewed. The other line regards self-repair as a process in which redundant structures within the system are employed by the system to effect either a partial or complete restoration of some subfunction. To some extent, the choice of approach is conditioned by a particular researcher's definition of "failure." For the purposes of this paper, the term "failure" is to be synonymous with the term "error." This definition implies that any dynamical system is to be treated informationally with respect to both failure and repair. While there is some controversy as to whether or not all dynamical systems can be so treated, there is mounting evidence [17] that most mechanical, electromechanical, and electronic systems are amenable to some extension of the Shannon–Weaver model.

1.1 *Definition of "Self-Repair" and Related Terms*

In the most general sense of the term, "self-repair" is applied to those (and only those) dynamical systems that, by means of redundancy, monitoring, and learning capacity, have the ability either to correct or to compensate for internal error. This definition embraces three key terms that are themselves in need of definition.

The first is the word "redundancy" which may be generically defined as the employment of two or more structures to accomplish the same task. Within the constraints of this definition, two subtypes of redundancy can be delineated: configurational redundancy and functional redundancy. The former is defined as the employment of two or more *like* structures (physical or symbolic) to accomplish the same task. The latter can be specified as the employment of two or more *unlike* structures to accomplish the same task. An example of the former might be a jet aircraft employing three identically structured engines concomitantly; an example of the latter might be the employment, at different times, of both oars and a sail on a small boat. It is clear that both types allow for parallel as well as standby redundancy.

The second key term of the definition is "monitoring." This word is intended to embrace that methodology and those techniques which allow for and enhance a system's ability to isolate error sources within itself.

The third and final key term is "learning capacity." This means the potential of a system for acquiring, prior to (off-line) or during (on-line) task accomplishment, knowledge of its own error sources as well as knowledge of its own control functions.

1.2 The Problem

In terms of the foregoing definitions and distinctions, the general problem of developing a self-repairing system becomes the following set of specific problems.

1) Accurately specifying a system's self-repair capability such as to ensure the following:
   a) the allocation of redundancy can be optimized;
   b) quantitative comparisons among self-repairing systems can be obtained.
2) Extending the mathematical theory and body of techniques for automatic fault location so that some resolution of the dichotomy between time-optimal and item-optimal solutions to automatic checkout can be obtained.
3) Developing and/or implementing learning techniques that will accomplish the following:
   a) optimize on-line learning (by a system) of the means and variances of error from sources within the system;
   b) show promise of allowing a system, off-line, to learn to specify its own controller.

Problem 1) entails the development of an adequate system effectiveness measure, that is, one that measures not only length of performance but also quality of performance. It also involves development of a technique, or set of techniques, for applying that measure to the actual design of a self-repairing system.

Problems 2) and 3) essentially reduce to the problem of specifying an informational network that will optimally check itself, optimally check the rest of the system, employ existing redundant (both configurational and functional) structures to effect repair, and, where necessary, modify its internal structure with respect to changes in its error-producing environment.

It is obvious that the ultimate objective in the solution of the entire self-repair problem (as stated above) is a very large and difficult one. This paper outlines a proposed approach and some of the specific techniques essential to solution of some of the problems. In a sense, the discussions contained are programmatic and reflect those ultimate necessary conditions for arriving at an acceptable self-repairing android. Many subdisciplines and methodologies that are relevant to the topic of self-repair are either only mentioned or entirely ignored. The controlling reason for this is to be found in the fact that, before a particular discipline can be considered relevant or irrelevant to the self-repairing problem, the problem must be carefully delineated and what would constitute an adequate solution defined.

2. Methodology

2.1 Evaluation and Synthesis of Self-Repairing Systems

The problem of self-repair, with respect to computer systems and subsystems as well as with respect to their networks, can be described in terms of two factors: detection of failure and correction of failure. This does not mean that these two factors are always physically separable, but only that they can be separated conceptually. Now, assuming that the terms "error" and "failure" are synonymous, it is virtually certain that the correction of error can be obtained only by some sort of redundancy. With respect to error detection, redundancy is not an absolute requirement; that is, in certain situations, something other than a redundant structure could be employed to detect an error. However, the extensive and effective employment by designers of redundant structures to detect errors in digital networks and systems in general indicates the importance of redundancy for facilitating that factor.

In view of these considerations, the significance of redundancy in any effort to enhance the self-repairing capability of a system becomes very large. It is, therefore, necessary to discuss briefly the needs for a method to optimize the employment of redundant structures.

2.1.1 The Need for a System Effectiveness Measure (SEM)

As indicated in Section 1, all redundancy, whether symbolic or physical, parallel or standby, reduces to two distinct types: configurational redundancy and functional redundancy. Besides the distinctions among types of redundancy, it is useful to distinguish between levels of redundancy. At least two levels may be articulated: digital network redundancy, that is, redundancy at the level of gating networks; and system and /or subsystem redundancy, that is, redundancy at the level of system organization.

Employment of redundancy at the first level leads almost certainly to a consideration of the powerful technique of interwoven networks, with the concomitant problem of maximizing the number of errors detected and corrected while minimizing the number and network levels of switching devices used. Employment of redundancy on the second level leads to the problems associated with double or triple subsystems, possibly involving decision logic, and to the optimization of software techniques for diagnosis and checkout.

All of these problems can be subsumed, however, under the single most significant design problem involved with the use of redundant structures at any level. This problem is one of accurate evaluation of the effects of different kinds of redundant structures and the allocations of such structures. This is to say that there is no adequate and uniform method, within most of the open literature, that may be employed so as to optimize the use of redundant structures either to preclude error, or to compensate for it, or both. It is true that several investigators have developed results that are impressive solutions to some of the more straightforward subproblems of system optimization with respect to redundancy. Thus, Dick [8], [9], Einhorn [11], and Kletsky [14] have addressed themselves with some success to deciding which configurationally redundant technique, taken singly, provides the greatest improvement in system reliability. Pierce [18], by deriving limit theorems for systems displaying exponential degradation, has determined the optimum configurational redundancy to achieve "on-line" self-repair. It would seem, however, that the methods proposed by these men apply only to a specific type of redundancy (configurational), assume statistical independence with respect to the occurrence of error, and assume that the underlying statistical distribution for system failure is identical to that for its respective classes of components.

2.1.2 Description of the SEM to be Employed—Performance Capability Measure (PCM)

As a result of these and other considerations, several new techniques for measuring the effects of different kinds and allocations of redundant structures have been developed by Dorrough [19], [20] and applied on the system level [10]. Of these techniques, one that indicates a large application potential has been further developed and is used to accomplish the following:

1) Imply, initially, both the type of redundancy on the system organization level and the type of redundancy on the logical network level for several types of gating devices.
2) Provide a means for evaluating the effects of both configurational and functional redundancy, at both levels, on system and/or network self-repair.

The technique indicated is best described in terms of a Performance Capability Measurement (PCM). Thus, if some nominal value for each critical system parameter (that is, some constant which bounds the performance of a dynamical system, and which if exceeded, would be a sufficient condition for task failure), as well as the maximum allowable error in that parameter, can be formulated, PCM can be defined in the following way:

$$\mathrm{PCM} = \sum_{i=0}^{n} P_i(t) E_i(\Delta p < \epsilon/\mu_i, \sigma_i, t, \epsilon) \tag{1}$$

where

$P_i(t)$ = probability that a system (or network) occupies the ith degraded state at time t

$E_i(\Delta p < \epsilon/\mu_i, \sigma_i, t, \epsilon)$
= probability that the error Δp in a critical system parameter is less than some maximum allowable ϵ, given that the system is in the ith state at time t and that μ_i and σ_i are the mean and standard deviations, respectively, of the error in that critical system parameter

n = number of noncatastrophic system states.

It should be noted that the term "system state" is not restricted in meaning to that of a single performance mode but may also indicate sequences and/or combinations of such modes. It is evident that if system states are restricted to single performance modes, the contribution of error from modes, that is, states, occupied previous to time t will be ignored. To account for alternate system histories and those error contributions caused by previously occupied modes, it would be necessary (in most cases, for computation reasons) to define system states as sequences and/or combinations of performance modes. In order to handle such sequences and combinations, it would be necessary to utilize the technique of partitioned stochastic matrixes first examined and proposed in 1964 by Dorrough ([19], pp. 25–28) and later ramified and rigorized by Pierce ([18], pp. 162–170).

In evaluating the PCM [Equation (1)] it is clear that the E_i function could be set to one in those cases where the quantity of error generated by any particular state is either unknown or is irrelevant. This, of course, would permit concentration upon the $P_i(t)$ function. Where system degeneration is assumed to be Markovian in nature, the method for evaluating $P_i(t)$ can be set forth in terms of the following three cases.

Case 1: Solution of k failures out of n identical components or subsystems, called single sequential flow (Fig. 1).

Case 2: Difference-equation formulation of a sequential state flow graph or multiple sequential flow (Fig. 2).

Case 3: Matrix formulation of arbitrary flow equations.

Fig. 1 shows Case 1, the simplest configuration for state transitions. The values of λ are the transition rates per some unit of time. When all of the λ's are equal (that

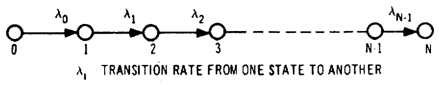

Fig. 1. Single sequential flow.

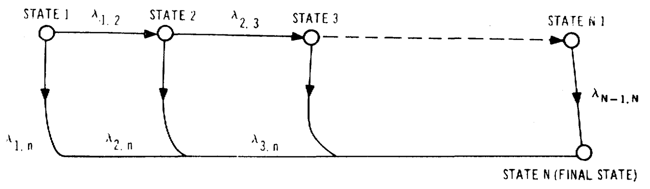

Fig. 2. Sequential state flow.

is, with respect to failure rate, all components or subsystems are identical) and the system is initially in state 0, the state occupancies expressed as a function of λ are as follows:

$$P_K(t;\lambda) = \frac{(\lambda t)^k e^{-\lambda t}}{\Gamma(k+1)} \tag{2}$$

$$= P \text{ (occupancy of } k\text{th state at time } t)$$

$$P_N(t;\lambda) = \sum_{k=N}^{\infty} \frac{(\lambda t)^k e^{-\lambda t}}{\Gamma(k+1)} = \int_0^{\lambda t} \frac{X^{N-1}e^{-X}dX}{\Gamma(N)} \tag{3}$$

$$Q_t(K;\lambda) = \frac{\lambda(\lambda t)^{K-1}e^{-\lambda t}}{\Gamma(k)} \tag{4}$$

$$= P \text{ (time } t \text{ required to reach } k\text{th state).}$$

Note that $P_K(t;\lambda)$ is the Poisson distribution and that $Q_t(k;\lambda)$ is the Poisson's dual, the gamma distribution.

When the values of λ are not equal, as in Case 2, a set of simultaneous equations must be solved. Under such conditions, let us assume that $\lambda_{i,\ i+1} \to \lambda_1$ and that $\lambda_{in} \to \mu_1$ in Fig. 2. The states are ordered according to their probable occurrence, and hence are sequential. Accordingly, if a short time interval Δt is considered, the occupancy probabilities may be written as

$$P_i(t+\Delta t) = (1 - \lambda_i\Delta t - \mu_i\Delta t)P_i(t) + \lambda_{i-1}\Delta t P_{i-1}(t) + \epsilon \tag{5}$$

where $1<i<N$.

$$P_1(t+\Delta t) = (1 - \lambda_1\Delta t - \mu_1\Delta t)P_1(t) + \epsilon \tag{6}$$

$$P_N(t+\Delta t) = P_N(t) + \sum_{i=1}^{N-2} \mu_i \Delta t P_i(t) + \lambda_{N-1}\Delta t P_{N-1}\Delta t + \epsilon \tag{7}$$

when

$$\lim_{\Delta t\to 0} \left\{\frac{\epsilon}{\Delta t}\right\} = 0.$$

Rearranging terms and letting $\Delta t \to 0$ results in the following differential equations:

$$\frac{dP_i(t)}{dt} = \lambda_{i-1}P_{i-1}(t) - (\lambda_i + \mu_i)P_i(t) \tag{8}$$

where $1<i<N$.

$$\frac{dP_1(t)}{dt} = -(\lambda_1 + \mu_1)P_1(t) \tag{9}$$

$$\frac{dP_N(t)}{dt} = \sum_{i=0}^{N-2} \mu_i P_i(t) + \lambda_{N-1}P_{N-1}(t) + P_N(t). \tag{10}$$

These are most easily solved by reformulating the equations as a Laplace transform:

$$P_i(S) = \frac{\lambda_{i-1}P_{i-1}(S)}{S\left|(\lambda_i + \mu_i) + S\right|} \tag{11}$$

where $1<i<N$.

The same reformulation can be effected for $P_1(S)$ and $P_N(S)$. The expressions for $P_i(t)$ are easily found by the initial values of state occupancy and the technique of partial fraction expansions. For (11), $P_i(t)$ is a sum of the terms of the form $\alpha_j \exp\{-(\lambda_j+\mu_j)t\}$ when they are the coefficients from the partial fraction expansion.

Case 3 has the following matrix formulation:

$$\begin{bmatrix} (1-\Sigma\lambda_{0i}\Delta t) & \lambda_{10}\Delta t & \cdots \lambda_{N0}\Delta t \\ \lambda_{01}\Delta t & (1-\Sigma\lambda_{1i}\Delta t) & \cdots \lambda_{N1}\Delta t \\ \lambda_{02}\Delta t & \lambda_{12}\Delta t & \cdots \lambda_{N2}\Delta t \\ \vdots & \vdots & \vdots \\ \lambda_{0N}\Delta t & \lambda_{iN}\Delta t & \cdots \quad 1 \end{bmatrix} \cdot \begin{bmatrix} P_0(t) \\ P_1{'}t) \\ P_2(t) \\ \vdots \\ P_N(t) \end{bmatrix} + \begin{bmatrix} \epsilon_0 \\ \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_N \end{bmatrix} = \begin{bmatrix} P_0(t+\Delta t) \\ P_1(t+\Delta t) \\ P_2(t+\Delta t) \\ \vdots \\ P_N(t+\Delta t) \end{bmatrix}. \tag{12}$$

This, in turn, gives the differential equation

$$\begin{bmatrix} -\Sigma\lambda_{0i} & \lambda_{10} & \cdots \lambda_{N0} \\ \lambda_{01} & -\Sigma\lambda_{1i} & \cdots \lambda_{N1} \\ \lambda_{02} & \lambda_{12} & \cdots \lambda_{N2} \\ \vdots & \vdots & \vdots \\ \lambda_{0N} & \lambda_{1N} & \cdots \quad 0 \end{bmatrix} \begin{bmatrix} P_0(t) \\ P_1(t) \\ P_2(t) \\ \vdots \\ P_N(t) \end{bmatrix} = \frac{d}{dt}\begin{bmatrix} P_0(t) \\ P_1(t) \\ P_2(t) \\ \vdots \\ P_N(t) \end{bmatrix} \tag{13}$$

which can be transferred into the Laplace matrix form:

$$\begin{bmatrix} S+\Sigma\lambda_{0i} & -\lambda_{10} & \cdots -\lambda_{N0} \\ -\lambda_{01} & S+\Sigma\lambda_{1i} & \cdots -\lambda_{N1} \\ -\lambda_{02} & -\lambda_{12} & \cdots -\lambda_{N2} \\ \vdots & \vdots & \vdots \\ -\lambda_{0N} & -\lambda_{1N} & \cdots \quad S \end{bmatrix} \begin{bmatrix} P_0(S) \\ P_1(S) \\ P_2(S) \\ \vdots \\ P_N(S) \end{bmatrix} = \begin{bmatrix} P_0(0+) \\ P_1(0+) \\ P_2(0+) \\ \vdots \\ P_N(0+) \end{bmatrix}. \tag{14}$$

These are solved by standard Laplace transform methods and inverted, with $P_i(0+)$ as the initial condition of state i.

In describing the kind of stochastic process discussed in this paper, it is necessary that something be said about hazard functions. The hazard function of a transition probability is always defined as the conditional probability $h(t)$ of a transition if one has not occurred at time t. In (12) through (14), λ corresponds to the hazard function. Constant hazard functions lead to exponential distributions. Other distributions, such as the Weibull or the extreme value, are given by nonconstant hazard functions. The form of (13) is unchanged, except that the λ factors are replaced by the new hazard function.

With respect to the evaluation of $E_i(\Delta p < \epsilon/\mu_i, \sigma_i, t, \epsilon)$ under conditions of Markov degeneration, one must first consider the probability of an ith state at time t, $P_i(t)$. The time derivative of this quantity is generally twofold: 1) the rate at which the given system or network enters the state $P_i^+(t)$, and 2) the rate at which it leaves the state $P_i^-(t)$. This is given by

$$\frac{d}{dt} P_i(t) = P_i^+(t) - P_i^-(t). \tag{15}$$

For example, if the states are Poisson distributed, then

$$P_i(t) = \frac{(\lambda t)^i}{i!} e^{-\lambda t} = II(\lambda, i; t), \tag{16}$$

and for a gamma distribution,

$$P_i^+(t) = \frac{\lambda(\lambda t)^{i-1}}{\Gamma(i)} e^{-\lambda t} = \gamma(\lambda, i; t) \tag{17}$$

$$P_i^-(t) = \frac{\lambda(\lambda t)^i}{\Gamma(i+1)} e^{-\lambda t} = \gamma(\lambda, i+1; t). \tag{18}$$

Assume that $f_i(\tau; t)$ is the probability density function of the time to transition from state i if the system entered state i at time t. Here τ is the time to transition from t. Assuming that all input terms to state i are removed, the expression $f_i(\tau; t)$ is found from the general occupancy probability of (13), where the initial conditions are $P_i(t) = 1$, and $P_j(t) = 0$ when $i \neq j$. If the above conditions are fulfilled, then $P_i(x)$, with $x > t$, is the required $f_i(x-t; t)$. The probability of the system entering state i at time τ and leaving at time t, $t > \tau$, is then given by

$$P_i^+(\tau) f_i(t - \tau; \tau) = \alpha_i(t - \tau; \tau). \tag{19}$$

The quantities $P_i(t)$ and $P_i^-(t)$, respectively, are given by

$$P_i(t) = \int_0^t \int_t^\infty P_i^+(x) f_i(y - x; x) dy dx \tag{20}$$

and

$$P_i^-(t) = \int_0^t P_i^+(x) f_i(t - x; x) dx. \tag{21}$$

$P_i^+(t)$ is derived from the initial conditions associated with the transition matrix (13) when outputs from state i are removed. Then $P_i(t)$ gives the required $P_i^+(t)$. If a path exists from state i through intermediate states to state j, removing the outputs of state i is not allowed, and the inputs to state i alone must be considered without removing the outputs.

Assume that $D_k(\tau; t)$ is the error in a critical system parameter when the system is in state k for τ seconds after entering at time t. Then the expected mean of this error is

$$\begin{aligned} \mu_k'(t) &= \int_0^t P_k(x) D_k(t - x; x) \int_t^\infty f_k(y - x; x) dy dx \\ &= \int_0^t \int_t^\infty \alpha_k(y - x; x) D_k(t - x; x) dy dx \end{aligned} \tag{22}$$

and the expected variance is found by:

$$\sigma_k'^2(t) = \int_0^t \int_t^\infty \alpha_k(y - x; x) D_k^2(t - x; x) dy dx - \mu_k'^2(t). \tag{23}$$

Note that these quantities are already weighted by the probability of the system occupying state k at time t; hence, the expected error and error variance, respectively, at time t are

$$\mu_i(t) = \frac{\mu_k'(t)}{P_k(t)} \tag{24}$$

and

$$\sigma_k^2(t) = \frac{\sigma_k'^2(t)}{P_k(t)} \cdot \tag{25}$$

Now, the probability that the error caused by the kth state is less than ϵ is found from

$$E_k = \int_{-\epsilon}^{\epsilon} g_k(x; \mu_k, \sigma_k) dx \tag{26}$$

where $g_k(x; \mu_k, \sigma_k)$ is the density function of error in the kth state whose mean is μ_k and whose standard deviation is σ_k. This may be approximated by a normal distribution or by a Gram–Charlier expansion of the density function in terms of its moments. Additional moments are found by

$$\begin{aligned} \mu_k{}^j = \frac{1}{P_k(t)} \int_0^t \int_t^\infty & \alpha_k(y - x; x) \\ & \cdot \{D_k(t - x; x) - \mu_k(t)\}^j dy dx. \end{aligned} \tag{27}$$

Normally, only a few of the moments are required because only a few items of the Gram–Charlier expansion need be used.

After $E_k(\Delta p < \epsilon/\mu_k, \sigma_k, t, \epsilon)$ is found, the PCM may be easily evaluated. Note that nothing has been said about whether the states are individual, or sequences, or combinations of performance modes. All that is necessary is that the expression of $D_k(\tau; t)$ be in some form suitable for integration. This may be done by simulation with curve fitting or by analytical methods. In practice, the function $D_k(\tau; t)$ is usually simulated, expressed as a low-order polynomial, and integrated. In the case of exponential distributions, evaluation of the necessary integrals is remarkably straightforward.

For a method of evaluating the PCM discussed herein under those conditions where a system's degradation is non-Markovian, that is, where consideration of system history during task accomplishment is necessary, the reader is referred to the Extended Performance Capability Measure discussed in [20].

2.1.3 Application of PCM to the Evaluation of a System

The application of the PCM to evaluating the system effectiveness of a large computer-based system is demonstrated in the Appendix of this paper. There a typical ground-mode ballistic mission is presupposed, and an inertial system for accomplishing that mission is described and simulated. Three different computers, each employing integrated circuitry, are involved in three respective simulations of the control loop: 1) a minimal computer composed entirely of irredundant subsystems, 2) a computer whose redundancy was allocated on a purely intuitive and pragmatic basis and whose self-repair capability is off-line, and 3) a computer (described in Section 3) whose redundancy allocation was decided in terms of PCM and the methods of Section 2.1.4 and whose self-repair potential is both on-line and off-line.

Among the many results of these simulations, the following should be mentioned: 1) the minimal computer's contribution to platform misalignment began to equal those of the other subsystems in the control loop after the elapse of 12 000 hours, 2) the pragmatic computer, whose self-repair function was accomplished off-line, took 34 000 hours before its contributions to misalignment equalled that of the other subsystems, and 3) the computer whose self-repairing capability was optimized by the techniques previously and subsequently discussed ran 124 000 hours before its contributions to misalignment were equivalent to those of the other subsystems. These results become even more significant when it is realized that the third computer gave the highest PCM value only among a total of three possible organizations and/or designs. With a higher number to consider and where networks with learning capability and multifunction logic ability are considered, even larger PCM values can be expected. Because all of these considerations involve applications of the PCM at the digital network level as well as at the system organization level, the error states may be quite numerous. Hence, some method of partitioning the Markov transition matrices must be devised and utilized in order to obtain some upper bound on the hazard rate and thereby determine $P_i(t)$. The general method for doing this is the next consideration.

Consider a network composed of two tiers of randomly connected redundancy modified gates (Fig. 3), that is, a configurationally redundant network. The function of the network is to use the redundant information on its n input channels so as to produce more reliable information on its n output channels. Further, $z(r)$ shall be defined as a random variable denoting the number of zero state inputs for the rth redundancy-modified gate, and $\phi_i(r) = p[z(r) = i]$. Now if an average is obtained over all possible redundancy-modified gates and over all possible errors within those gates, then the transition matrix T is given by

$$\begin{bmatrix} \phi_0(r+1) \\ \phi_1(r+1) \\ \vdots \\ \phi_n(r+1) \end{bmatrix} = T \begin{bmatrix} \phi_0(r) \\ \phi_1(r) \\ \vdots \\ \phi_n(r) \end{bmatrix}. \tag{28}$$

Under the conditions of relatively small n, where the number of performance states is not too large, (28) can be easily solved so as to evaluate $P_i(t)$. However, with the condition that n is relatively large, some technique for collecting and/or partitioning the various states into manageable sets must be devised. For the particular problem at hand, the computation could be reduced by trichotomizing the possible performance states into the following: D = the one state in which the number of zeros lies between 0 and Δ; E = the error state in which the number of zeros lies between Δ and $1-\Delta$; and F = the zero state in which the number of zeros lies between $1-\Delta$ and 1.

Combining a theorem developed by Pierce ([18], pp. 162–168) with the inequality information about the cumulative sums of the columns of the T matrix of (28), one can produce the U and S matrices of that theorem:

$$U = \begin{bmatrix} U_{DD} & U_{DE} & U_{DF} \\ U_{ED} & U_{EE} & U_{EF} \\ U_{FD} & U_{FE} & U_{FF} \end{bmatrix} \tag{29}$$

$$S = \begin{bmatrix} S_{DD} & S_{DE} & S_{DF} \\ S_{ED} & S_{EE} & S_{EF} \\ S_{FD} & S_{FE} & S_{FF} \end{bmatrix}. \tag{30}$$

The hazard rate λ is then the probability of transition from one of the nonerror states D or F to the error state E. In this case λ is not unique unless further specified. However, the combining of all possible performance states so as to have only three sets of such states provides an upper bound for λ:

$$\lambda \leq \max[(u_{ED} + u_{FD}), (s_{DF} + s_{EF})].$$

Within this upper bound it is possible to set λ to any one of several arbitrary values and then utilize such values in (12) through (14) to obtain the quantity $P_i(t)$ in (1), which is the PCM.

It is clear that the technique of combining and/or partitioning performance states indicated by (29) and (30) allows for an effective method for computing ϵ in the function $E_i(\Delta p<\epsilon/\mu_i, \sigma_i, t, \epsilon)$ of the PCM (1). This is done by first assuming that the network in question is of a certain size (where q denotes the number of redundancy modified gates within the network). The maximum number of multiple critical, subcritical, or critical/subcritical errors that a network, of size q, of such gates corrects is then computed by conventional techniques. (Critical error on the input of a redundancy modified gate can be defined as the minimum number of such errors sufficient to give an output error; subcritical error is any error which is not critical.) The value obtained from this computation is the value assigned to ϵ. This means that during the time of accomplishment of some task assigned to the network (such as conventional digital logic perhaps involving computation), the various single and multiple errors (that is, a single set or multiple sets of zeros between Δ and $1-\Delta$) cannot at any time t combine so as to exceed ϵ without the network being declared in an intolerable, and hence catastrophic, performance state. Thus, it is clear by definition that the Δp [Equation (1)] of the various n number of performance states are all less than ϵ, and hence the E_i function of the PCM can be set to the value of 1 with PCM being evaluated solely in terms of the evaluation of $P_i(t)$ by the method previously described. In such cases it is obvious that $\text{PCM}=1-P_i(t)$.

2.1.4 Application to PCM to System Synthesis

The next problem that must be considered is the development and employment of some procedure for deciding the amount and allocation of redundancy which will be best at each redesign step. Moreover, this procedure must permit the synthesizing of both optimal netwroks and optimal systems. All of this, of course, assumes that for every step some figure-of-merit, for a system and/or network, is available. This figure-of-merit would be used as a criterion for optimizing tradeoff between the improvement of a system's or network's PCM value and some penalty such as cost, where the latter term is construed generically in terms of weight, size, or power consumption, as well as in terms of monetary value.

One of the figures-of-merit that will be considered here is denoted by M, and for networks is given by the ratio

$$M = \lim_{R\to\infty} \ln \frac{[P_i(t)]}{R} \tag{31}$$

where R = number of redundant elements or subsystems.

For computer systems other than networks, M would be given by

$$M = \lim_{R\to\infty} \ln \frac{[P_i(t)E_i(\Delta p < \epsilon/\mu_i, \sigma_i, t, \epsilon)]}{R}\,. \tag{32}$$

On the network level, where $\text{PCM}=1-P_i(t)$, and where a simple log failure probability with equal exponential survival probabilities for, say, redundancy modified gating structures is assumed, the PCM for the unimproved network of such structures would be

$$\ln(\text{PCM}) = \sum_{i=1}^{n} \ln[1 - P_i(t)] \tag{33}$$

where i now denotes failure state i, for example, the production of a critical error at the output of the ith redundancy-modified gate.

Because the objective of the synthesis procedure is to increase ln (PCM) as cheaply as possible, it can be asserted that in adding the allocated redundancy, that step is optimal which gives the greatest ratio $\{\ln[1-P_i'(t)]-\ln[1-P_i(t)]\}/\{c_i'-c_i\}$, where c denotes cost.

Thus, if the redundancy R on some original network is increased by k elements (where element denotes channels or switching nodes), and the figure-of-merit M also changes to M', then the optimal ratio α is

$$\begin{aligned}\alpha &= \frac{\exp[R_iM_{i(R_i)}] - \exp[(R_i+k)M'_{i(R_i+k)}]}{(R_i+k)c_i - R_ic_i}\\ &\cong \frac{\exp[R_iM_{i(\infty)}] - \exp[(R_i+k)M'_{i(\infty)}]}{(R_i+k)c_i - R_ic_i}\,.\end{aligned} \tag{34}$$

This holds, of course, for a network of redundancy-modified gates in which only one input channel on one gate need be operational (free of error) in order for the network to be operational. This could easily be extended to those cases where more than one operational channel or gate is required for an operational network.

For the case in which R_i is changed to R_i+1 without any change in M_i, the approximation given in (33) becomes

$$\alpha = \frac{\exp[R_iM_{i(\infty)}]\{1-\exp[M_{i(\infty)}]\}}{c_i}\,. \tag{35}$$

To completely clarify the procedure generated by these considerations, the following steps can be articulated:

Step 1: Synthesize the least costly network to do the job.

Step 2: Generate n sets of redundancy improvements.

Step 3: Order the sets of improvements according to the decreasing ratio of $\{\ln\ 1-P_i'(t)-\ln\ 1-P_i(t)\}/\{c_i'-c_i\}$.

Step 4: Select the highest-order set for the first redesign step toward improving the original network.

Step 5: Return to Step 2, and start over for the next improvement.

It is evident that Steps 1 through 5 can be accomplished, at least in part, by means of a computer program in which the original network is simulated. The optimality of such a program can be gleaned by first considering a function $\phi(k)$ which denotes the rate of increase of $\ln[1-P_i(t)]$ per cost, evaluated at an incremental cost of k. Select any set of improvements at random and order it according to the decreasing ratio of $\{[\ln(1-P_i'(t)]-\ln[1-P_i(t)]\}/\{c_i'-c_i\}$, giving the function $\phi(k)$. The difference in incremental $\ln[1-P_i(t)]$ between this set of improvements and the optimum $\phi'(k)$ selected in order of decreasing ratio is the integral

$$-\int_0^{\Delta c}\{\phi'(k)-\phi(k)\}dk,$$

the latter being nonpositive since $\phi'(k)\geq\phi(k)$.

It is clear from the definitions of $P_i(t)$ and M that redesign Steps 1 through 5 can be applied on the network level to both functional and configurational redundancy. Hence, the synthesizing of networks using bias devices and microprogramming suggested later (Section 2.2.2) can be methodically and optimally accomplished.

This five-step procedure describes a "deterministic" search process which results in a system with the highest PCM evaluation. More efficient procedures, based on learning theory, have been developed in connection with research on learning control systems. These procedures are used to determine a vector which describes the optimal state of a learning controller [21]-[23]. The extension and application of these procedures will be referred to as "probabalistic" because they involve those stochastic variables associated with learning.

Extension of Matyas' simple adaptive random optimization algorithm [24] has been found to have the best convergence characteristics for high-dimension problems. A development of this algorithm is presented here in an application to the present synthesis problem.

Let Step 1 in the above five-step procedure represent the initial state of the system. Let this state be denoted by a vector $S(0)$ whose components $s_1, s_2, \cdots, s_n$ are associated with elementary logic functions. (An elementary logic function here is the lowest-order network for which redundancy will be considered; it may be a two-input AND gate or an entire storage register.) The problem may now be stated as one in which a search is made for a vector S for which

$$\text{PCM}(S_{\text{opt}}) \geqq \text{PCM}(S), \qquad \text{for all } S.$$

Let $S(k)$ denote the kth state of the system and $\text{PCM}[S(k)]$ an evaluation of the PCM function at $S(k)$. Let $\delta(k)$ be a stochastic vector process with variable mean value and variable correlation matrix; that is

$$\delta(k) = d(k) + T(k)\xi$$

where

$$E\{\delta(k)\} = d(k),$$

$T(k)$ is a transformation matrix, and ξ is an n-dimensional normal random vector with zero mean value and unit correlation matrix. Consider now perturbations in the state of the system of the form

$$S(k) + \delta^{(1)}(k)$$

where the notation $\delta^{(1)}(k)$ denotes a realization of the random process at the kth step. If

$$\text{PCM}[S(k) + \delta^{(1)}(k)] > \text{PCM}[S(k)]$$

the step is considered a success and is denoted by $Y(k)=1$; if, on the other hand,

$$\text{PCM}[S(k) + \delta^{(1)}(k)] \leq \text{PCM}[S(k)]$$

the step is considered a failure and is denoted by $Y(k)=0$. The next state of the system is determined by the decision rule

$$S(k+1) = \begin{cases} S(k) & \text{when } Y(k) = 0 \\ S(k) + \delta^{(1)}(k) & \text{when } Y(k) = 1 \end{cases}$$

and the parameters $d(k+1)$ and $T(k+1)$ are a function of their value at the kth step whether or not this step was a success.

A specific application of the above is now offered. Let the elements of S be odd integers, $s_i=1, 3, 5, \cdots$, $(i=1, 2, \cdots, n)$ which denote the number of inputs to a majority element. The starting configuration $S(0)$ will include no redundancy; that is, $s_i=1$ $(i=1, 2, \cdots, n)$. Adaptive random optimization may now be applied to determine the vector S which results in a maximum PCM evaluation. This vector specifies the amount of redundancy and its distribution (that is, the number of inputs to each majority device under consideration).

2.2 The Implementation of Self-Repairing Systems

In considering the reliability improvement of a digital computer system, one can assume that such a system consists of electronic components such as resistors and transistors; logical function networks such as decoders, counters, and registers; and subsystems such as memory units and adder units. Therefore, in order to significantly improve the reliability of the computer, all levels of the computer structure must be improved. Some research has been devoted to redundancy on the level of individual components. Most of the results, however, are seriously inadequate by reason of the cost, size, and power consumption constraints of the system. Moreover, achieving statistical independence for error occurrence is almost impossible (see [20], pp. 47-50). As the digital system becomes more complex, it demands components of unrealistically high reliability to yield a reliable system. Thus, the levels at which reliability improvement can be advantageously applied are on the network, subsystem, and system levels. Configurational and functional redundancies in the forms of multiprocessor and multicomputer systems are some of the approaches that have been taken to improve the computer on the subsystem and system levels. On the network levels, reliability improvement techniques have been discussed in various places [5], [18], [25], [26]. All of these techniques, however, are limited to configurational redundancy. Certain approaches such as parallel, stand-by, redundant networks require additional detecting and switching circuits which themselves contribute to the unreliability of the computer. The network reliability improvement techniques described in this paper are directed towards functional redundancy as well as configurational redundancy. In particular, for configurational redundancy, interwoven redundant logic will be considered, and for functional redundancy, the use of multifunction logic will be discussed. It should be stated at this point that except for parity checking, little attention is given to error codes because their value in systems involving computation is, at best, marginal (see [18], pp. 132–145). Finally, the set of techniques briefly discussed is not a closed one.

2.2.1 On-Line Monitoring

Interwoven redundant logic is a form of configurational redundancy that promises to be a valuable aid for a logic design engineer in executing a self-repairing network. An interwoven network is a logical network in which the redundant input signals are interconnected so that a maximum amount of statistical independence is obtained between the various redundant inputs. The error correction is achieved by inhibiting the propagation of the signal of the error-causing logical element. The corrected signal is obtained with the aid of redundant logical elements. Interwoven redundant logic can be divided into the following two types: 1) redundancy-modified gating, and 2) redundancy-using decision elements. The first type uses only regular logical gates. The second uses such decision elements as threshold devices and majority devices, either instead of or in addition to regular gates. Two examples of each type will be given. In all cases the monitoring and correction of error is done on-line while performing regular digital logic.

A simple logical network for achieving $Z=AB+C$, which illustrates the error-correcting properties of the first type of interwoven redundant logic, is shown in Fig. 3. The signals A, B, and C are quadrupled. The logical functions performed by the gates shown are such that $D1=A1\cdot A2\cdot B1\cdot B2$ and $Z1=D1+D2+C1+C2$. Each output labeled Z is correct for any single error in Ai or Bi, some multiple-input errors, or a $1\to 0$ error in $C1$. For example, if a $0\to 1$ error occurs in $A1$, $B1$, or both, $D1$ and $D3$ are still the correct value for AB. If a $1\to 0$ error occurs in $A1$ when the correct value of B is 1, the value of $D1$ and $D3$ will incorrectly be 0, but the correctness of $D2$ and $D4$ will still result in each Zi being the correct value of $AB+C$. The only time a Zi output is incorrect for a single error is when the correct value of AB is 0 and there is a $0\to 1$ error in one Ci.

Frequently, a logic designer wishes to use NAND and NOR gates rather than AND and OR gates. A network in which the redundancy-modified gates are NAND and NOR gates is also possible as, for example, in the network for $Z=AB\overline{C}$ shown in Fig. 4. Each output labeled Z is correct for any double error in Ai or Bi, any double

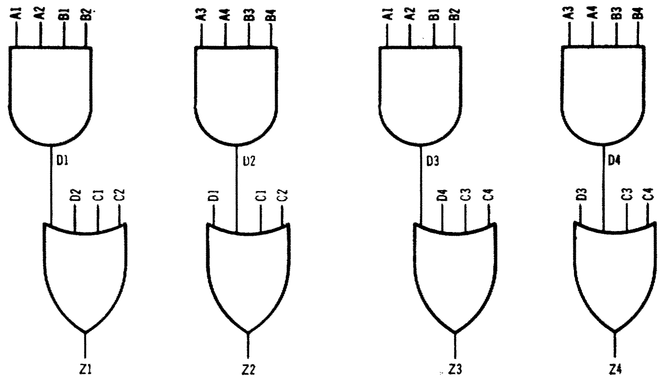

Fig. 3. AND/OR self-correcting network.

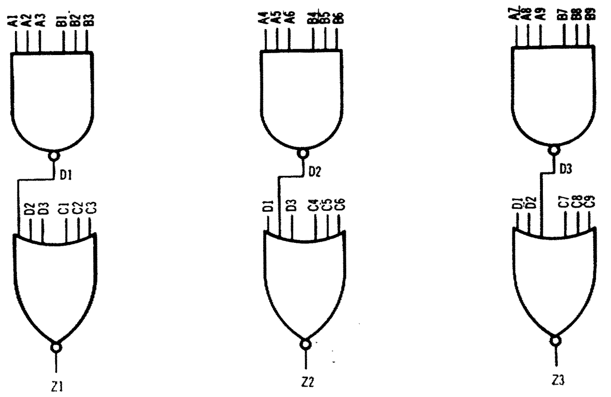

Fig. 4. Double error correction employing redundancy-modified NAND/NOR gates.

$1\to 0$ error in Ci, and some errors of greater multiplicity. For example, if a $0\to 1$ error occurs in $A1$ and $A2$, $D1$ is still correct. If a $1\to 0$ error occurs in $A1$ and $A4$ when the correct value of B is 1, $D1$ and $D2$ will incorrectly be 0, but the correctness of $D3$ will result in each Zi being correct. The only single or double error which will cause an erroneous Zi output is a $0\to 1$ error in one or two Ci when AB is 1.

A threshold logic element is shown in Fig. 5. The output Z is 1 if

$$\sum_{i=1}^{n} X_i \geqq \theta,$$

where the X_i are the inputs and θ is the threshold value. The X_i may have a value which produces the effect of $a-1$ in addition to the usual 0 and 1 inputs. In the following example there are no -1 inputs. The synthesis of the function $Z=AB+C$ is given in Fig. 6 to illustrate the second type of interwoven logic using threshold devices. The signal Di is correct for any single error in Ai or Bi; similarly, Zi is correct for any single error in Ci. The threshold logic is triplicated because the gate itself may be unreliable.

In each of the above examples it should be noted that an error in Ai, Bi, or Ci may represent an earlier error in a gate, and at the analysis of an error in Ci is the same as for an error in the first layer of gates.

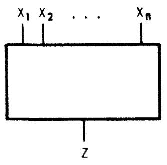

Fig. 5. Threshold logic element.

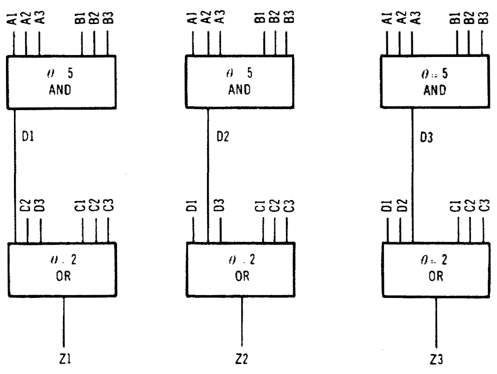

Fig. 6. Threshold element synthesis for $Z=AB+C$.

The majority vote taker is a special form of a threshold logic device which provides a means of deciding which signal to use if signals which should be the same are not. The synthesis of $Z=AB+C$ using three-input majority voters is shown in Fig. 7. The signal Zi is the correct value of $AB+C$ for any single error in Ai, Bi, Ci, Di, or E_i since the other two correct Ei will result in each Zi being correct. The two-out-of-three voters are triplicated to increase the reliability of the output Zi.

Rules for interconnecting error-correcting elements and the approximate reliability improvement obtained thereby were developed by Pierce [18]. In general, for a logic network to correct n errors, its redundancy-modified gates must have $(n+1)$ redundant inputs yielding $(n+1)^2$ such gates. For example, the cost of correcting two errors in a redundant network is nine times that of a nonredundant network. Because of the greatly increased cost for error correction using redundancy-modified gates, it appears that the use of threshold logic is the most economical means of network configurational redundancy. Use of threshold logic also offers the possibility of incorporating some of the work now being done on adaptive networks to increase reliability (see [27] and [28]).

2.2.2 *Off-Line Monitoring*

Besides on-line monitoring and/or correction, there are off-line techniques that comprise programmed check-routines, configurationally redundant structures in stand-by mode, and multifunction logic devices. All of these techniques are considered off-line since they require the interruption of normal system functioning in order to isolate and correct the errors due to subsystem failure.

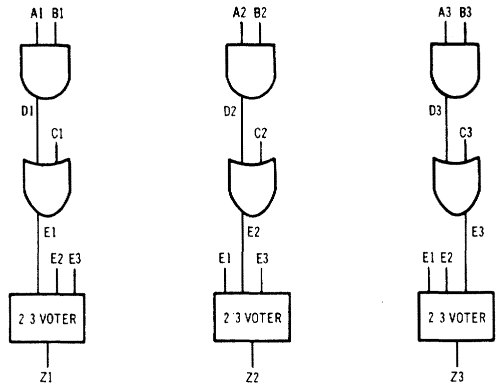

Fig. 7. Single error correction employing majority logic.

With respect to the use of any programmed automatic device for isolating faults that have occurred in a system, the task of developing a search program is considerable. Essentially, the problem reduces itself to two subproblems: 1) the estimation of the probabilities of error-generating failures lying in respective parts of the system, and 2) devising optimum search policies based upon these estimates. Although some work in this area has been done and reported [29], [30], it has been entirely confined to systems whose error-producing situations do not involve conditional probabilities. In this respect, some present [31] and projected research involves considerable effort in the direction of developing a set of estimators which can be used in arriving at adequate search policies for highly complex systems.

2.2.2.1 Teaming of Multifunction Logic: The traditional approach to building a logical function into a computer has been to write the function in terms of the logical operations AND, inclusive-OR, and NOT, and then build the function from gates which realize these three functions as well as from gates which realize the negation of the first two functions. Another approach is to incorporate multifunction devices which, with a slight adjustment, called biasing, can realize any one of a set of functions. For example, such a device might be one with three inputs which forms one function of two variables when the third input is 0, but a different function of two variables when the third input is 1. Using microprogramming to replace a failed logical network with a network of biased multifunction devices is a form of functional redundancy which seems to have promise as a means of self-repair for networks for which no methods for the correction of failures by interwoven logic are known, or for which known methods are too complicated or expensive to be worthwhile. The advantage of using multifunction devices rather than a traditionally built new network is that one multifunction device can be built into a computer as a possible replacement for several different logical networks.

Among the existing multifunction devices are adders, Rutz commutator transistors, integrated-circuit counter adapters, and integrated-circuit half adders. Dunham

[32] discusses the use of the first two of these, and Doctors [33] discusses the use of the others. In both of these works and in the second part of [34], the relative efficiency of different ways of forming logical functions is discussed. However, very few general conclusions are drawn and many factors which should be considered in developing a criterion of optimality are not, although many of them are mentioned. Hence, research needs to be done to decide the criterion for selecting devices and to find, if possible, an analytical means of determining what kind of multifunction devices meet this criterion. Investigation of this area is in progress [35].

As an example of biasing, the following shows how logical functions can be formed from adders. All logical functions can be formed from three basic functions: the AND ($\cdot$), inclusive-OR ($+$), and NOT ($-$). However, it is as easy to get two more—the exclusive-OR ($\forall$) and the "if, and only if" ($\equiv$)—from an adder as it is to get the three basic ones, and these two can be used to simplify some expressions. For example, $\overline{(p \cdot q)} \cdot (p+q)$ is the same as $p \forall q$, and $(p \cdot q)+\overline{(p+q)}$ is the same as $p \equiv q$. These five basic functions are achieved by biasing an adder. Biasing an adder means that one or two of the adder's inputs are always fixed to either a 0 or a 1. For the inputs p, q, and r, the adder produces the sum s and the carry c shown in the following table:

p	q	r	s	c
1	1	1	1	1
0	1	1	0	1
1	0	1	0	1
0	0	1	1	0
1	1	0	0	1
0	1	0	1	0
1	0	0	1	0
0	0	0	0	0

When r is biased to 0, $s = p \forall q$ and $c = p \cdot q$. When r is biased to 1, $s = (p \equiv q)$ and $c = p+q$. Hence, if r is biased to 1 and q to 0, $s = \bar{p}$ and $c = p$.

A logical network can be replaced by adders by making the following one-to-one replacements: replace each AND gate by an adder with r biased to 1, and replace the output of the gate by the carry output of the adder. Replace each OR gate by an adder with r biased to 1, and replace the output of the gate by the carry output of the adder. Replace each inverter with an adder with r biased to 1 and q to 0, and replace the output of the inverter with the sum output of the adder. Schematically, it can be said that Fig. 8 may be replaced by Fig. 9.

One feature of Fig. 9 immediately noticed is that the three outputs, $A \forall B$, C, and $(A \cdot B) \equiv \overline{C}$ are not used; it cannot be avoided in this case. However, in some cases these extra outputs can be used so that the number of adders in the new network is less than the number of

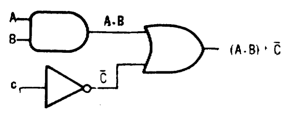

Fig. 8. Synthesis of $A \cdot B+\overline{C}$ with gates.

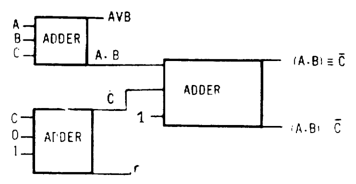

Fig. 9. Synthesis of $A \cdot B+\overline{C}$ with adders.

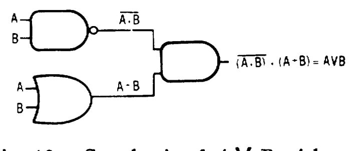

Fig. 10. Synthesis of $A \forall B$ with gates.

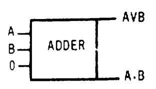

Fig. 11. Synthesis of $A \forall B$ with an adder.

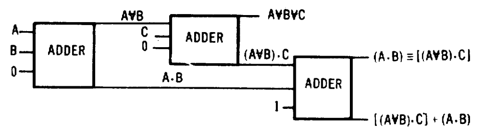

Fig. 12. Synthesis of $(A \forall B) \cdot C+(A \cdot B)$.

gates and inverters in the old network. As a trivial example, Figs. 10 and 11 show how the network for an exclusive-OR can be replaced by one adder. Less trivially, the logical function $(A \forall B)C+(A \cdot B)$ requires six gates or three adders (Fig. 12), while the equivalent function $(A \cdot C)+(B \cdot C)+(A \cdot B)$ requires five gates or five adders. Hence, the function is normally designed with five gates, but if this network fails it can be replaced by only three adders.

Normally, a check for failure will be made in each subsystem either constantly or at fixed intervals of time. When such a check shows a failure, a diagnostic program will be run automatically. If the diagnosis shows that a network in the subsystem, such as the network shown in Fig. 8, has failed, a microprogram would rechannel signals destined for that network through an equivalent system of adders. In the example of Figs. 8 and 9, the microprogram would rechannel A, B, and C from the locations shown in Fig. 8 to those shown in Fig. 9. The program would also bias the three adders in Fig. 9, would direct the outputs of two of the adders to the third, and would channel the $(A \cdot B)+\overline{C}$ output of the third adder to the location where the output of the network had been channeled. In order to replace logical

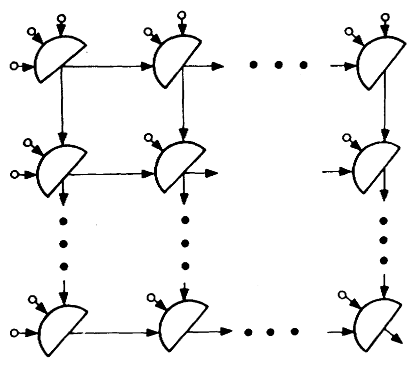
Fig. 13. Matrix of majority function elements.

networks with adders, the computer must be capable of being microprogrammed.

Many other schemes for building logical functions have been devised. Some of these are similar to biasing and have the advantages of biasing; that is, the same devices are possible replacements for similar devices and for several logical networks. Hence, these schemes can be considered for use in functionally redundant designs.

For example, one such scheme due to Canaday [6] uses a matrix of three-input and one- or two-output devices for which the output is the majority function; that is, the inputs A, B, and C result in the output $AB+AC+BC$. The matrix is always connected as shown in Fig. 13 with external inputs at the circles.

It is possible to build the matrix so that all functions of a certain number of variables can be realized. For example, all logical functions of three variables can be realized by a 3×3 matrix and all logical functions of four variables can be realized by a 4×6 matrix. However, if only one or a few functions are wanted from the matrix, the size of the matrix required can frequently be reduced. The inputs necessary to have the matrix act as a given function are discussed by Canaday, and others.

The problems for study in this area include but do not exhaust such things as how to check for failure, how much increase in performance capability is provided by various possible schemes in return for how much additional hardware, and what specific combinations of single-function and multifunction logic produce optimum self-repair.

3. Design of a Self-Repairing Computer

Various interrelated reliability improvement techniques have been presented and discussed. The problem now is how to apply these techniques to various equipments within the computer so that maximum improvement in reliability may be realized. In this respect, the first (deterministic) synthesis technique previously discussed (Section 2.2.4) is employed here to produce a highly reliable computer.

A digital computer may be classified into the following four major units: 1) memory, 2) control, 3) arithmetic, and 4) input-output. Each of these units has its own problems and peculiarities. In fact, no single reliability technique is best for all applications. Parity check, for example, is effective in checking memory error but is useless in improving the reliability of arithmetic operations. On the other hand, either an interwoven logic approach or residue checking has potential for error correction in arithmetic operations. The problem in the design of a reliable digital computer lies then in simulation and the use of those synthesis methods articulated earlier in Section 2 in order to know which reliability techniques to apply and where to apply them in the logical design. Paper design of a sample computer is undertaken here to illustrate the application of different types of redundancy and to give the reason for the choice of techniques employed. Furthermore, it is hoped the design will adequately demonstrate the feasibility of the self-repair program described earlier.

3.1 Design Philosophy

The fact that any degree of reliability may be obtained by a corresponding increase in logical hardware redundancy has been discussed earlier. However, the added cost, size, and so forth, normally do not justify the degree of reliability achieved. This is another way of saying that c in our previously discussed ratio, $[1-P_i'(t)]-[1-P_i(t)]/c_i'-c_i$, is unduly high in value. In certain applications, such as airborne control computers, the increase in weight, size, and power consumption is highly undesirable. It has been decided, therefore, to adopt the approach which combines hardware redundancy and computer software ability to achieve a more reliable system without the large penalty in hardware. The system design will be able to tolerate at least a single failure in most subsystems, and in certain critical areas two or more failures, without decrease in performance. Also, more failures can be tolerated with a certain amount of degradation in computing capabiity.

As stated in Section 1, a self-repairing computer should have the capability of reorganizing its own structure and should retain the ability to perform diagnosis. That is, the computer should be able to perform at least the following basic instructions upon any occurrence of malfunction:

1) Add
2) Subtract
3) Transfer
4) Test and jump
5) Store.

There is a portion of the machine in which failures cannot be tolerated. This hard-core portion must, of course, be kept as small as possible in order to attain a reasonaby high PCM value. The hard-core logics may be those of the memory address-selection circuits, the parity checker, the instruction-execution-control sequence circuits, or any other logics which do not lend themselves easily to reliability improvement techniques.

Fig. 14 (figure abbreviations are noted in Table I) is

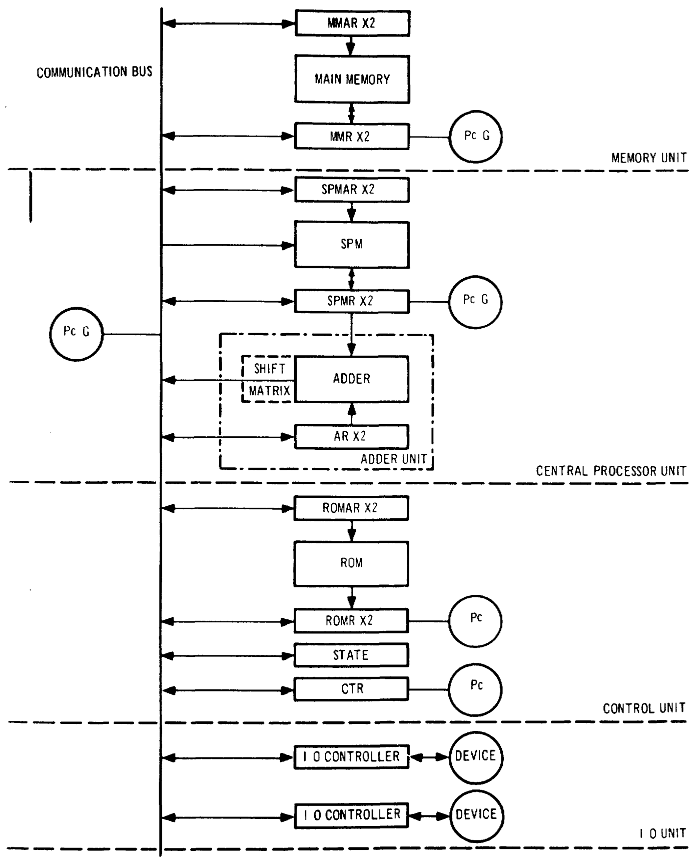

Fig. 14. Self-repair computer.

TABLE I
LEGEND FOR THE SELF-REPAIR COMPUTER

Abbreviation	Meaning
MMAR	main memory address register
MMR	main memory register
SPMAR	scratch-pad memory address register
SPM	scratch-pad memory
SPMR	scratch-pad memory register
AR	arithmetic register
ROMAR	READ-only memory address register
ROM	READ-only memory
ROMR	READ-only memory register
CTR	counter
Pc	parity checker
Pc/G	parity checker and/or generator

a schematic block diagram of a proposed self-repairing computer. The value of self-repair is highly dependent upon the machine structure, the amount of hardware required to implement the techniques, and the number of failures which the machine can tolerate in various subsystems. Thus, the computer must be simulated to evaluate the design techniques of self-repair computers —to determine whether the techniques employed do improve the reliability, or if the gain is offset by the extra logics needed to implement the techniques.

3.2 Computer Organization

The computer organization chosen for the demonstration of self-repair techniques has the following main characteristics:

1) random-access core main memory—primary data storage;
2) two's complement parallel arithmetic unit;
3) scratch-pad memory—intermediate data shortage;
4) READ-only memory—microprogram control.

The machine uses a data bus for all the data transfers in order to provide organizational flexibility. Thus, any spare register or memory location can replace the failed units. In addition, common error-detection logics can be employed for all data transmission.

To ensure high reliability, fast error detection, and diagnosis, the machine operation, such as register-to-register data transfer or register-to-register data modification, is controlled by simple microcommands. Each microstep of operation operates serially; virtually all data transfer must pass through the adder and be subjected to accuracy checking. Further, the sequence of each operation is performed by microprograms stored in the READ-only memory. The number of integrated-circuit hardware registers is kept to only the essential registers; all other registers used to store data and intermediate results, such as program counter, accumulators, index registers, and so forth, are stored in the scratch-pad memory.

3.2.1 Memory Unit

The memory is a high-speed random-access magnetic-core memory used for primary data and program storage. The following reliability techniques are incorporated in the memory system.

3.2.1.1 Parity Checking: The transfer of data between core memory and main memory register (MMR) is checked for odd parity. Parity is generated when information is written into memory and checked when it is read out to MMR.

3.2.1.2 Memory Protection: Each block of 128 memory words is assigned a code-lock number. The assignment is under program control. The block can be entered with new information only when the key fits the code lock.

The memory protection feature, therefore, provides identification for different programs and data. In addition, it helps detect program error, loss of program sequencing operation, and malfunction of memory addressing logic.

To facilitate the program execution of supervisory control and diagnostic routines, it is necessary to be able to have access to all the storage locations without having to reassign the code lock to the same number. For this, a master key code can be provided for entering any coded area.

3.2.1.3 Secondary Storage Location: In certain applications such as real-time data computation, critical data errors cannot be tolerated. Additional alternate storage area is provided in memory locations $(0000\text{–}0128)_{10}$. As the data are written into the desired memory location, the same data are also regenerated in the corresponding location in the $(0000\text{–}0128)_{10}$ memory area. For example, if the real-time data are being entered into the memory locations $(1250\text{–}1300)_{10}$, the same data are also written into memory locations $(0098\text{–}0020)_{10}$. This feature can be implemented with very little increase in hardware and with no requirement for additional computer time.

3.2.2 Central Processing Unit

The central processing unit (CPU) consists of two parts: a scratch-pad memory for local storage of intermediate results and data registers, and an arithmetic unit for parallel data manipulation.

3.2.2.1 Scratch-Pad Memory: The scratch-pad memory (SPM) is used as a local storage for internal data manipulation within the CPU. It is a destructive read-out, high-speed, random-access core storage. Its operation may be overlapped with the main storage.

The scratch-pad memory, with its relatively low cost and inherently better reliability (than the comparable transistorized flip-flops), is a highly desirable local storage for computer implementation. Also, the reliability of data in the SPM is improved by means of a centralized parity check unit.

The replacement of failed registers with spare registers is normally a costly and complicated process if conventional transistorized registers are used, because all the input and output signals of the failed registers must then be transferred to the replacement registers. By keeping as many as possible of the working and spare registers in the SPM, the problem of register replacement is reduced. Also, the spare registers form a pool of replacement registers, any one of which is able to replace any failed register rather than just a limited type of failed register. Thus, the reliability is maximized.

3.2.2.2 Arithmetic Unit: The arithmetic unit performs arithmetic and logical operations. It consists of a parallel binary adder, a shifter, and an arithmetic register (AR). Since the adder and the shifter are the center of all data manipulation in the computer system, it is desirable to optimize their reliability. Therefore, interwoven logic techniques employing redundancy-modified gates will be used to implement the arithmetic unit. With this technique, each logic gate in the nonredundant version may fail before the unit fails.

3.2.3 Control Unit

The control unit executes micro-operations of the microprograms. It provides the timing control to perform these operations in proper sequence. It also controls interrupt processing, error detection, and operations involving input, output, and storage. The control unit consists of a READ-only memory, control counters, a state register, and any other timing and control circuits.

3.2.3.1 READ-Only Memory (ROM): To achieve high reliability, flexibility, and simplicity, the control commands are implemented in the READ-only memory. The correctness of the command word is checked by a parity checker. The microcommand register (ROMR) can also accept the micro-operation from the main memory, thus providing the programmer with the ability to control the computer operation down to the hardware logic level. In this way, diagnosis checking is made easier and more accurate. Also, this technique provides additional flexibility. The programmer can flow-chart many error instructions, or can reflow-chart the existing instructions to bypass the malfunctioning units, thereby achieving functional redundancy.

Triple majority redundancy and interwoven logic will also be employed in major critical areas where uninterrupted operation is required. The areas of critical importance are those that are needed to sustain the computer operation for diagnostic routines. These critical networks may be timing, pulse generators, and counters.

3.2.3.2 State Register: The state register is used to define machine organization. It specifies which registers or redundant units are being used to form the computer organization, and provides the computer capability to perform self-repair.

3.2.4 Input–Output Unit

The buffer data registers for the input–output channels, together with the control words, are assigned to the scratch-pad storage. The data transfer between the main storage and input–output is controlled by the microcommand stored in the READ-only memory. In this manner, extra registers available in the spare pool in the SPM can be called upon to replace the malfunctioning buffer and control word registers.

3.3 Remarks

It is clear that the foregoing computer design does not represent the ultimate in self-repair. Although it has a very high PCM value among the set of three considered, no attempt was made to optimize the diagnostics employed nor was there any consideration given to other more modular types of organization. Moreover, the technique for synthesis was deterministic (Section 2.1.4). The probabilistic technique of random optimization is known to give better results. Finally, no attempt to incorporate a learning network or to use multifunction logic was made.

APPENDIX

PERFORMANCE ANALYSIS OF A VELOCITY-DAMPED SCHULER LOOP WITH MALFUNCTIONS

As an example of the application of the theory described earlier, a control loop problem, that is, a velocity-damped Schuler loop, is analyzed. Platform rotation of an otherwise inertially stabilized orientation through gyrotorquing, based on integration of relative acceleration which is contaminated by a component of g, can be considered a basic mission. This mission must be accomplished in every terrestrial inertial navigation system, for example, in airplane navigation or in the ground mode of a ballistic missile. These missions may take hours, days, or even weeks, and it would appear valuable to consider such a mission to obtain quantitative information about platform misalignment caused by a malfunction in the electronics of the loop (especially the digital subsystem). Also, the derivation of the error equations seem sufficiently complicated and nonuniform to offer general conclusions for the theory. To avoid unnecessary complications, the loop was considerably simplified, but only to the extent that the characteristic properties were still apparent.

A1. Assumptions for System

The vehicle motion is assumed to occur along a static great circle (no Earth rotation). The x–y plane is identical with the trajectory plane, the z axis is oriented to local vertical, and the y axis is perpendicular to the x–z plane.

The accelerometer input is mounted parallel to x; the mating gyro to be torqued (ω_y) has its case-input axis normal to that accelerometer input axis.

The position of the vehicle is determined on the great circle by an actual angle θ_G. The calculated value for this angle is θ_0. Assuming a spherical Earth, the misalignment in the trajectory plane is given by the difference

$$\epsilon_y = \theta_G - \theta_0$$

The output is velocity damped; the total magnitude of the velocity is measured by either an odometer or Doppler radar, and the velocity components (x_s) are computed in the digital computer by means of additional gimbal angle measurements.

A flow diagram, in Laplace transform representation, of the single-channel Schuler loop with velocity damping is shown in Fig. 15. Fig. 16 shows the loop after the malfunctions were introduced.

The accelerometer output (or better, interface input) contains a scale factor and a fictitious term

$$a = a_{gx}(1 + k_a) + a_2.$$

The digital subsystem contributes to the inaccuracy of $\dot{x}_s$, $\dot{x}_c$, and ω_y

$$\dot{x}_c = \dot{x}_{c1}(1 + k_i) + \dot{x}_{c2}$$

$$\dot{x}_s = V_{Gx}k_D + V_{Dx} \qquad \text{with } k_D = k_{D1}(1 + \Delta k_D) \quad (36)$$

and

$$\omega_y = \left(\dot{x}_c + k_{2D}(\dot{x}_c - \dot{x}_s)_t - T_1 \cdot \frac{1}{R} + \omega_{yd}\right).$$

Thus, the stabilization gyro output becomes

$$\theta_0 = \left(\int \omega_y dt\right)(1 + k_g) + \theta_{0b}.$$

All the scale factor errors are assumed to be caused by instrument anomalies or by truncation or roundoff in

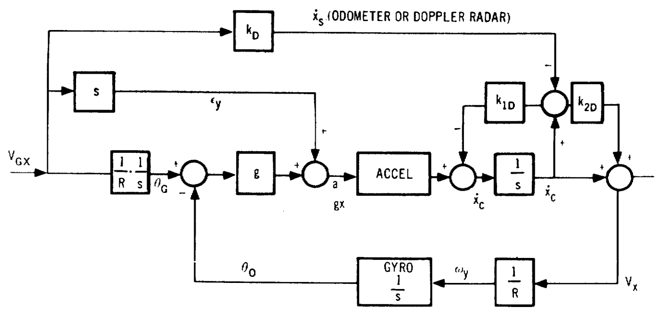

Fig. 15. Single-channel Schuler loop with damping.

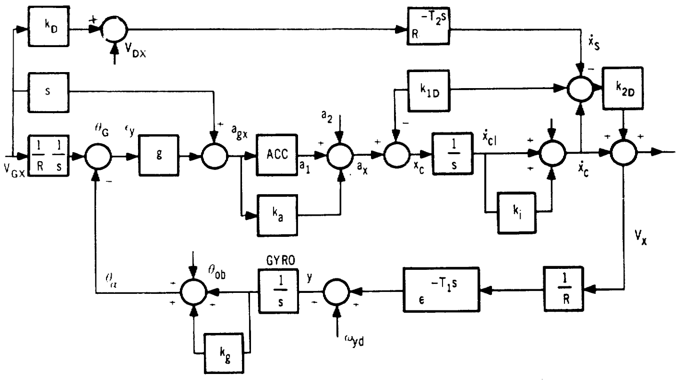

Fig. 16. Single-channel Schuler loop with malfunctions.

the digital computer during the failure-free mode. Only the fictitious terms (V_{Dx}, a_2, x_{c2}, ω_{yd}, θ_{0b}) and the time lags T_1 and T_2 are caused by failures. For dynamic considerations, the malfunctions are introduced as step functions. Their magnitude can be determined only in a statistical manner which is detailed in Section A5. The hardware responsible for this kind of failure is listed, together with the failure rates, in Table II.

The effect of the failures on the system parameter ϵ_y is studied within the following two problem areas.

1) *Stability and time response:* Its effect on misalignment angle is not cumulative, and is therefore interesting only at the time of failure occurrence.

2) *Steady-state behavior:* The steady-state misalignment angle is cumulative and especially important in long-range missions. Its probability density function will be derived in Section A5.

The component failure rates are as follows:

Component	
Transistor	$\lambda = 0.01/10^6$ hour
Diode	$\lambda = 0.002/10^6$ hour
Resistor	$\lambda = .005/10^6$ hour
Integrated Circuit	
Element	$\lambda = 0.025/10^6$ hour.

A2. Assumptions Concerning Failure Classes

The term failure is understood here to denote a permanent breakdown of hardware elements and, therefore, permanent sources of error. Failure states are usually determined by the muliplicity of failing ele-

TABLE II
HARDWARE FAILURE RATES

j	Malfunctions (Output Quantities)	Failure	λ_j for 10^6 hours	$\lambda_{j,n}$ for 10^6 hours	Class
1	ω_d	Y register	0.35	1.07×10^{-4}	3
		torque amplifier	0.12		
		digital-analog converter	0.30		
		total	0.77		
2	Δa	acceleration encoder	0.22	1.6×10^{-4}	3
		sampler	0.10		
		E, F register	0.53		
		adder with input and output gates (ADD$_1$)	0.30		
		total	1.15		
3	ΔV	odometer and gimbal angle encoder	0.50	1.85×10^{-4}	3
		G, H register	0.53		
		adder with input and output gates (ADD$_2$)	0.30		
		odometer	---		
		total	1.33		
4	T_1	every noncatastrophic failure in the digital subsystem which does not destroy any input quantity: arithmetic unit, control unit, and core memory	200.0	278×10^{-4}	2 (type d) or 3

ments in the various subsystems. However, every failure state can be related to a certain degradation in system performance, measured in terms of output errors (X_{ik}). This relationship suggests reduction of the extremely large number of failure states to malfunction classes, each characterized by the type of effect on system performance.

In the case of an inertial guidance system, there are five classes. They can be defined in the following manner.

Class 1: The failure causes a permanent decreased computation speed, but no error in any input quantity.

Example: Replacement of the multiplication control by a multiplication subroutine.

Class 2: The failure causes an intermittent decreased computation speed resulting in a single exceeding of the schedule time for the outputs, but no error in any input quantity.

Example: The failing part is replaced, and the program proceeds with the speed of the failure-free mode after partial repetition of wrong computations.

Class 3: The failure causes an intermittent malfunction in the computation of output quantities and results in a permanent error of at least one system parameter.

Example: Permanent failure in one of the buffers of the interface equipment. After replacing the faulty element, the interface is again functioning properly.

Class 4: The failure causes a permanent malfunction in the computation of at least one output quantity. The error will be accumulated from the time of occurrence until completion of the mission.

Example: Failure in one of the accelerometers.

Class 5: The effect of the failure is unpredictable and, therefore, not tolerable. This type is catastrophic.

Both Classes 1 and 2 may occur at any time in the mission; Class 2 may also occur with any multiplicity. Note that after the occurrence of Class 2 malfunctions, the computer is used to improve the reliability of the system. Permanent failures may have intermittent effect only and do not necessarily contribute to degraded performance. Transient malfunctions fall also into Classes 2 and 3. The effect of these malfunctions on performance will be treated with the same mathematical tools as those malfunctions derived in subsequent sections.

Classes 3, 4, and 5 are assumed to be mutually exclusive, that is, statistically independent from the standpoint of occurrence. In the system and computer program organization, provisions are made so that no failure state can contribute to more than one class of malfunction. If these provisions are not made, then the assumption is true, at least in a first approximation. The effect of off-schedule times on accuracy degradations is negligible if the same failure state, in addition, causes loss of input quantities.

A3. Frequency Response of the Misalignment Angle and Characteristic Equation

The frequency response for the misalignment angle is obtained from the block diagram in Laplace transform and is

$$y = \frac{s^2 c_3 + s^1 c_2 + c_1}{s^2 + sA_4 + A_5} \tag{37}$$

with the partial transcendent coefficients

$$c_3 = -\theta_{0b}$$

$$c_2 = \frac{V_{Gx}}{R} - (1 + k_g)\omega_{yd} - e^{-T_1 s}\cdot\frac{1 + k_g}{R}\cdot A_6 - \theta_{0b}A_4$$

$$c_1 = (1 + k_i)k_{1D}\left(\frac{V_{Gx}}{R} - (1 + k_g)yd\right)$$
$$-\frac{1}{R}(1 + k_i)(1 + k_g)(k_{1D}(V_{Dx} + k_D V_{Gx})e^{-(T_1+T_2)s}$$
$$+ (1 + k_{2D})a_2 e^{-T_1 s})$$

$$A_4 = k_{1D}(1 + k_i)$$

$$A_5 = \frac{g}{R}(1 + k_{2D})(1 + k_i)(1 + k_a)^2 e^{-T_1 s}.$$

The characteristic equation of the loop becomes

$$s^2 + sk_{1D}(1 + k_i) + \frac{1}{R}(1 + k_{2D})(1 + k_i)(1 + k_a)^2 g e^{-T_1 s} = 0. \tag{38}$$

It is apparent that only the scale factor errors of the off-schedule time T_1 of the feedback path appear in the equation, and not the fictitious terms and T_2, which appear only as perturbation functions in the denominator of (37). To simplify the stability considerations, the transcendental term $e^{-T_1 s}$ in (38) is approximated by a rational function, usually called the Padé approximation

$$e^{-T_1 s} \cong \frac{1 - 0.5\, T_1 s}{1 + T_1 s}. \tag{39}$$

The coefficients of the rational function are determined so that the first three terms of the series obtained by carrying out the division become equal to the first three terms of the series expansion of the e function.

Thus

$$s^3 T_1 + s^2(1 + T_1 A_4) + s(A_4 - 0.5\, T_1 D_0) + D_0 = 0 \tag{40}$$

with

$$A_4 = (1 + k_i)k_{1D}$$

$$D_0 = \frac{g}{R}(1 + k_{2D})(1 + k_i)(1 + k_a)^2.$$

A4. Stability Considerations

The simplest way to get some quantitative information from the third-order characteristic (40) seems to be by means of the Routh criterion. From the theorem that the number of sign changes in the first column of the Routh scheme determines the number of roots in the right half of the s plane, a condition can be derived for the limit case, where stable oscillations exist

$$k_{1d}(1 + (1 + k_i)k_{1D}T_1) = 0.5\frac{g}{R}(3 + T_1 k_{1D}(1 + k_i)) T_1(1 + k_{2D})(1 + k_a)(1 + k_g). \tag{41}$$

If the time lag T_1 is equal to 0, undamped oscillations are possible only for $k_{1D} = 0$. If $T_1 \neq 0$, the relationship between the critical value for T_1, the loop constants, and the scale factor errors is given by (41). In the special case where no scale factor error exists, the critical time lag is determined by

$$T_{1,C} = -\frac{M}{2} \pm \sqrt{\frac{M^2}{4} + N}$$

with

$$M = \frac{8.31 \cdot 10^{-3}(1 + k_{2D}) - k_{1D}^2}{2.78 \cdot 10^{-3} k_{1D}(1 + k_{2D})} \tag{42}$$

and

$$N = \frac{1}{2.78 \cdot 10^{-3}(1 + k_{2D})}.$$

From (42) it follows that the effect of Class 1 and Class 2 malfunctions can be at least partially compensated for by changing the sorted constants k_{1D} and k_{2D} (see also Section A5) for that computation cycle in which the malfunction has occurred. From the critical upper bound of T_1, certain conclusions can be drawn for the maximal tolerable time difference between check routines, or for the critical length of the operating program periods. The downtimes in this production line model consist of the following:

1) The operation of check routines and failure localization procedures. Their length is, in general, a random variable.
2) Partial program repetitions.
3) The time interval between failure occurrence and start of the check routines which in the worst case is equal to the total length of one operating period.

The maximum length of the bad period must be significantly smaller than the critical time lag T_1.

It is not claimed here that the previous considerations about the effect of malfunctions on stability are complete. Especially missing is a study about the effect of intermittent off-schedule times (only constant time lags are considered) and the relationship between the damping ratio and the magnitude of the various malfunctions.

A5. Time Response and Steady-State Behavior

To obtain the time response for the various malfunctions (output quantities), the inverse Laplace transform has to be conducted for (37). Because of the transcendental nature of the coefficients, the transformation becomes

$$z = e^{Ts} \tag{43}$$

where

T = sampling rate

and was introduced in (37) with the transformed quotient developed in a continued fraction in z^{-i} ($i = 0, 1, \cdots$). The coefficients of z^{-i} are the amplitudes of the time response at the sampling instants iT. This procedure is described in Section 4 of [10].

Some of the curves are shown in Figs. 17 and 18, but only to the extent that it is necessary to obtain information about the frequency of the damped oscillations and to confirm the results of the stability considerations. The frequency of the oscillations is approximately 15 minutes; damping out occurs after about 30 minutes. It seems justifiable for long-range missions to study mainly the cumulative steady-state error.

The steady-state values of the misalignment angle caused by malfunctions can easily be derived by means

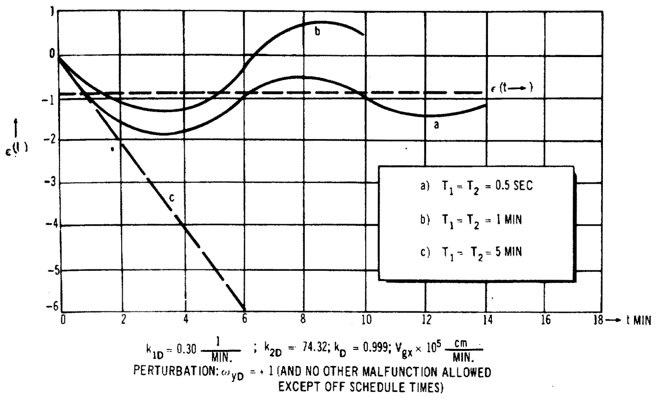

Fig. 17. Platform misalignment caused by malfunction in the digital subsystem.

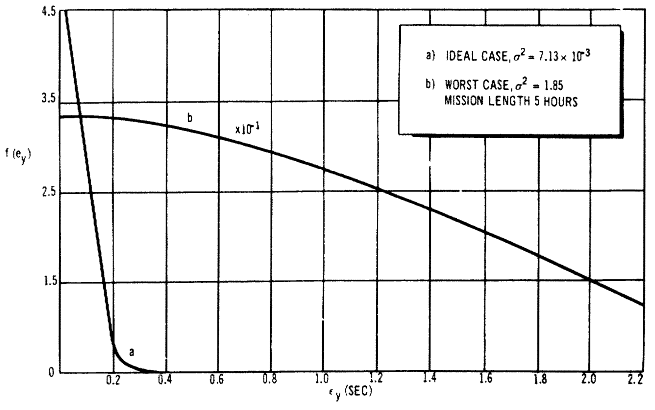

Fig. 18. Probability density of platform misalignment.

of the final value theorem

$$\lim_{t\rightarrow\infty} f(t) = \lim_{s\rightarrow\infty} sF(s)$$

if

$$F(s) = L(f(t)) \cdots \text{Laplace transform of } f(t).$$

These values are listed for the various perturbation functions in Table III. It is evident that neither T_1 nor T_2 appears in any of the formulas, which leads to the conclusion that off-schedule times do not decrease the system's value as long as the system remains stable and has sufficient damping. Only if T_1 becomes too large and has to be compensated for by changing the gain constants (k_{1D}, k_{2D}) will the steady-state error be automatically increased and the system's value affected.

The total error in the system parameter ϵ_y can be accumulated from the various expressions in Table III, assuming 1) all of the malfunctions can occur, 2) there is no limit on the multiplicity of malfunctions in the mission, and 3) two malfunctions cannot occur at the same time.

Thus

$$\lim_{t\rightarrow\infty} \epsilon_y(t) = k_1\omega_d + k_2\Delta a + k_3\Delta V = \Delta W_1 \qquad (44)$$

TABLE III
STEADY-STATE VALUES OF MISALIGNMENT ANGLE

Perturbation Function	Steady State
$V_{gx} = V_g \frac{1}{s}$	$V_g \frac{k_{1D}}{(1+k_2)(1+k_a)g}\left(\frac{1}{1+k_g} + k_d\right)$
$\omega_{yd} = \omega_d \frac{1}{s}$	$-\omega_d \frac{k_{1D}R}{1 + k_{2D}(1+k_a)g}$
$x_{c2} = \Delta v \frac{1}{s}$	0
$a_2 = \Delta a \frac{1}{s}$	$-\Delta a \frac{1}{(1+k_a)g}$
$V_{dx} = \Delta V \frac{1}{s}$	$-\Delta V \frac{k_{1D}}{(1+k_{2D})(1+k_a)g}$
$\theta_{0b} = \theta_0 \frac{1}{s}$	0

with

$$k_1 = \frac{k_{1D}}{1+k_{2D}} \cdot \frac{R}{g} = 43.1$$

$$k_2 = \frac{1}{g} = 0.0010$$

$$k_3 = \frac{k_{1D}}{1+k_{2D}} \cdot \frac{1}{g} = 0.676 \cdot 10^{-7}$$

for

$$k_{1D} = 0.005, \qquad k_{2D} = 74.32.$$

The semi-invariants for the variant ΔX_i,

$$\Delta X_1 = \omega_d, \quad \Delta X_2 = \Delta a, \quad \Delta X_3 = \Delta V$$

become

$$\begin{aligned} H_{1,i} &= \Delta X_i(1 - e^{-\lambda_i t}) \approx \Delta X_i \cdot \lambda_i t \\ H_{2,i} &= \Delta X_i^2(1 - e^{-\lambda_i t}) \approx \Delta X_i^2 \cdot \lambda_i t. \end{aligned} \qquad (45)$$

The semi-invariants for the final distribution become

$$\begin{aligned} H_1 &= k_1 \cdot H_{11} + k_2 H_{12} + k_3 H_{13} \\ H_2 &= k_1^2 \cdot H_{21} + k_2^2 H_{22} + k_3^2 H_{23} \end{aligned} \qquad (46)$$

with

$$\begin{aligned} &\Delta X_1 \cong 0.1''/\text{s} \\ &\Delta X_2 \cong 1.964 \times 10^3 \text{ cm/s}^2 \\ &\Delta X_3 \cong 5 \times 10^4 \text{ cm/s and } t = 5 \text{ hours.} \end{aligned} \qquad (47)$$

The mean and variance of $f(\epsilon_y)$ become

$$\begin{aligned} \alpha_1 &= H_1 = 0.0017 \\ \sigma^2 &= H_2 = 7.13 \times 10^{-3} \end{aligned} \qquad (48)$$

therefore

$$f(\epsilon_y) = 4.76 \cdot \exp \frac{\epsilon_y^2}{0.0143} \cdot$$

This curve (Fig. 18) represents the ideal case, namely, that all the failures in the digital computer are in Class 2 and only the interface contributes to the steady state ϵ_y.

It can be shown, on the same example, just how significantly reliability programming can influence the probability of mission success.

With less care in the program for redundant storing of input quantities, the malfunctions will cause not only an exceeding of the schedule time for the outputs, but also accuracy degradations in the outputs. The malfunctions will belong partially in Class 2 and partially in Class 3. In the worst case, all failures in the computer fall in Class 3 instead of Class 2. The semi-invariants in this worst case become

$$H_1 = 43.1 \cdot 10^{-3} \qquad H_2 = 1.85 \tag{49}$$

thus

$$f(\epsilon_y) = 0.294 \exp\left[\frac{(\epsilon_y - 0.043)^2}{3.70}\right].$$

In (48) and (49) an ideal replacement organization is assumed with an infinite supply of reserve elements. It should be noted that the large difference in the dispersion between (48) and (49) was caused only by different programming and different memory allocation.

The contribution of the self-correcting digital subsystem to platform misalignment seems quite small compared with the contribution of instrument anomalies. The variance for the latter is approximately 7 to 10 seconds and does not increase with the length of the mission. On the contrary, the variance caused by the digital subsystem is proportional to the length of the mission

$$\sigma_{\text{dig.s.}}^2 \propto \text{mission length.}$$

In the ideal case (48), the contribution of the digital subsystem to misalignment equals that of instrument anomalies only after $34 \cdot 10^3$ hours have elapsed. In the worst case (49), such contribution equals that of instrument anomalies by the time 133 hours have elapsed.

Acknowledgment

The author gratefully acknowledges his indebtedness to two of his colleagues, M. Arya and F. Veal. To the former goes the credit for the detailed logic design of the self-repairing computer organization presented in Section 3; to the latter goes the credit for saving the author from several mathematical inelegancies by a careful and critical reading of the original text.

References

[1] W. R. Ashby, "The self-reproducing system," *1962 Proc. 1st Internat'l Symp. on Biosimulation*, pp. 9–18.
[2] H. S. Balaban, "Some effects of system redundancy," *1960 Proc. 6th Nat'l Symp. Reliability and Quality Control*, pp. 388–402.
[3] ——, "Redundancy and switching failure," *Electronic Design*, vol. 10, pp. 72–75, 1962.
[4] ——, "Some effects of redundancy on system reliability," Research and Development Reliability, Electronic Division, American Society for Quality Control, pp. 119–141, 1961.
[5] W. G. Brown, J. Tierney, and R. Wasserman, "Improvement of electronic-computer reliability through the use of redundancy," *IRE Trans. Electronic Computers*, vol. EC-10, pp. 407–416, September 1961.
[6] R. H. Canaday, "Two-dimensional iterative logic," *1965 Fall Joint Computer Conf., AFIPS Proc.*, vol. 27, pt. 1. Washington, D.C.: Spartan, 1965, pp. 343–353.
[7] J. D. Cowan, "Many-valued logics and reliable automata," in *Principles of Self-Organization*, H. von-Foerster and G. W. Zopf, Eds. New York: Pergamon Press, 1962, pp. 135–180.
[8] R. S. Dick, "The reliability of repairable complex systems, pt. A: The similar machine case," *1961 Proc. 5th Mil-E-Con. Symp. on Military Electronics*, pp. 111–150.
[9] ——, "The reliability of repairable complex systems, pt. B: The dissimilar machine case," *IEEE Trans. Reliability*, vol. R-12, pp. 1–8, March 1963.
[10] D. C. Dorrough, "A reliability model for digital guidance systems," Douglas Aircraft Company, Inc., Santa Monica, Calif., SM-47657, June 1964.
[11] S. J. Einhorn, "Reliability prediction for repairable redundant systems," *Proc. IEEE*, vol. 51, pp. 312–317, February 1963.
[12] J. D. Esary, "The reliability of coherent systems," in *Redundancy Techniques for Computing Systems*, R. H. Wilcox and W. C. Mann, Eds. Washington, D. C.: Spartan, 1962, pp. 47–61.
[13] E. J. Kletsky, "Self-repairing machines," Syracuse University Research Institute, Syracuse, N. Y., Rept. RADC-TR-61-914, 1961.
[14] ——, "Upper bounds on mean life of self-repairing systems," *IRE Trans. Reliability and Quality Control*, vol. RQC-11, pp. 43–48, October 1962.
[15] R. R. Landers, "Achieving higher reliability through self-repair," *IEEE Trans. Aerospace*, vol. AS-1, pp. 735–747, August 1963.
[16] W. C. Mann, "Restorative processes for redundant computing system," in *Redundancy Techniques for Computing Systems*, R. H. Wilcox and W. C. Mann, Eds. Washington, D. C.: Spartan, 1962, pp. 267–284.
[17] I. G. Wilson and M. E. Wilson, *Information, Computers, and System Design*. New York: Wiley, 1965.
[18] W. H. Pierce, *Failure-Tolerant Computer Design*. New York: Academic Press, 1965.
[19] D. C. Dorrough, "Model theoretic approach to evaluating and enhancing the reliability of complex systems," Douglas Aircraft Company, Inc., SM-47767, December 1964.
[20] ——, "Redundancy and system reliability," Douglas Aircraft Company, Inc., Douglas Paper 3513, November 1965 (to be published in *Technometrics*).
[21] J. M. Mendel, "Self-organizing control systems, vol. 2: Open-loop time-optimal control of a stable maneuverable re-entry vehicle," Douglas Aircraft Company, Inc., SM-47904, June 1965.
[22] J. M. Mendel and J. J. Zapalac, "Self-organizing control systems, vol. 3: Off-line training of time-optimal, fuel-optimal and minimum-energy controllers," Douglas Aircraft Company, Inc., SM-51975, February 1966.
[23] J. J. Zapalac, "Self-organizing control systems, vol. 1: On adaptive computers," Douglas Aircraft Company, Inc., SM-47857, July 1965.
[24] J. Matyas, "Random optimization," *Automation and Remote Control*, vol. 26, February 1965.
[25] R. E. Barlow and F. Proschan, *Mathematical Theory of Reliability*. New York: Wiley, 1965.
[26] J. D. Esary and F. Proschan, "Coherent structures of non-identical components," *Technometrics*, vol. 5, no. 2, pp. 191–209.
[27] M. V. Wilkes, "Self-repairing computers," *IRE Trans. Electronic Computers* (*Correspondence*), vol. EC-10, pp. 93–94, March 1961.
[28] J. J. Zapalac, "Adaptive processes in decision making," Douglas Aircraft Company, Inc., SM-45921, April 1964.
[29] S. I. Firstman and B. Gluss, "Optimum search routines for automatic fault location," *Operations Research*, 1960.
[30] B. Gluss, "An optimum policy for detecting a fault in a complex system," *Operations Research*, 1959.
[31] D. C. Dorrough, "Minimizing diagnostic time by optimizing the sequence of testable items," Douglas Aircraft Company, Inc., Paper 4262 (in preparation).

[32] B. Dunham, "The multipurpose bias device, I: The commutator transistor," *IBM J. Research and Develop.*, vol. 1, pp. 116–129, April 1957.

[33] S. Doctors, "Multipurpose logic devices, pt. I," AC Spark Plug, El Segundo, Calif., Memo. 3249-LA-52, April 1961.

[34] B. Dunham, "The multipurpose bias device, II: The efficiency of logical elements," *IBM J. Research and Develop.*, vol. 3, January 1959.

[35] E. F. Veal, "On symmetry in Boolean functions and the sizes of interchange classes," Douglas Aircraft Company, Inc., Douglas Paper 4486, May 1967.

Bibliography

This bibliography is not exhaustive of the field but is merely a list of those works in self-repair and redundancy which are most relevant to this paper. For a more comprehensive listing of references on redundancy techniques, see Pierce, Failure-Tolerant Computer Design, or Barlow and Proschan, Mathematical Theory of Reliability.

[36] M. J. Abzug and W. G. O'Neil, "Application of adaptive control techniques to aero/space vehicles," Douglas Aircraft Company, Inc., Douglas Paper 947, 1960.

[37] M. C. Arya, "Reliability improvement techniques for digital computers," Douglas Aircraft Company, Inc., SM-49378, April 1966.

[38] W. R. Ashby, "The self-reproducing system," *1962 Proc. 1st Internat'l Symp. on Biosimulation*, pp. 9–18.

[39] H. S. Balaban, "Some effects of system redundancy," *1960 Proc. 6th Nat'l Symp. on Reliability and Quality Control*, pp. 388–402.

[40] ——, "Redundancy and switching failure," *Electronic Design*, vol. 10, p. 72–75, 1962.

[41] ——, "Some effects of redundancy on system reliability," Research and Development Reliability, Electronic Division, American Society for Quality Control, pp. 119–141, 1961.

[42] R. E. Barlow and L. C. Hunter, "Mathematical models for system reliability," Electronic Defense Lab., Mountain View, Calif., ASTIA Doc. AD-228-131, 1959.

[43] ——, "System efficiency and reliability," *IRE Nat'l Conv. Rec.*, pt. 6, pp. 104–110, 1959.

[44] ——, "Criteria for determining optimum redundancy," *IRE Trans. Reliability and Quality Control*, vol. RQC-9, pp. 73–77, April 1960.

[45] R. E. Barlow, L. C. Hunter, and F. Proschan, "Optimum redundancy when components are subject to open and short circuit failures," Boeing Scientific Research Lab., Seattle, Washington, Math Note 233, 1961.

[46] ——, "Optimum redundancy when computers are subject to two different kinds of failure," *SIAM J.*, vol. 2, March 1963.

[47] R. E. Barlow and F. Proschan, *Mathematical Theory of Reliability*. New York: Wiley, 1965.

[48] I. Bazovsky, *Reliability Theory and Practice*. Englewood Cliffs, N. J.: Prentice-Hall, 1961.

[49] R. Bellman and S. Dreyfus, "Dynamic programming and the reliability of multicomponent devices," *Operations Research*, vol. 6, no. 2, pp. 200–206, 1958.

[50] Z. W. Birnbaum, J. D. Esary, and S. C. Saunders, "Multicomponent systems and structures and their reliability," *Technometrics*, vol. 3, pp. 55–77, 1961.

[51] G. Black and F. Proschan, "Optimal redundancy," *Operations Research*, vol. 7, no. 4, pp. 581–588, 1959.

[52] W. G. Brown, J. Tierney, and R. Wasserman, "Improvement of electronic computer reliability through the use of redundancy," *IRE Trans. Electronic Computers*, vol. EC-10, pp. 407–416, September 1961.

[53] M. W. Burt and D. C. James, "How much does redundancy improve reliability?" *Control Engineering*, vol. 10, no. 6, pp. 71–76, 1963.

[54] R. H. Canady, "Two-dimensional iterative logic," *1965 Fall Joint Computer Conf., AFIPS Proc.*, vol. 27, pt. 1. Washington, D. C.: Spartan, 1965, pp. 343–353.

[55] R. R. Carhart, "A survey of the current status of the electronic reliability problem," Rand Corporation, Santa Monica, Calif., Corp. Memo. 1131, 1953.

[56] J. D. Cowan, "Many-valued logics and reliable automata," in *Principles of Self-Organization*, H. von-Foerster and G. W. Zopf, Eds. New York: Pergamon Press, 1962, pp. 135–180.

[57] ——, "Toward a proper logic for parallel computation in the presence of noise," Proceedings of Wright Air Development Division Bionics Symposium, WADD Tech. Rept. 60-600, pp. 93–152, 1960.

[58] J. D. Cowan and S. Winograd, "Minimally redundant reliable neural nets," in *Seminar on Computation in the Presence of Noise*. Cambridge, Mass.: M.I.T. Press, 1961.

[59] J. A. Daly, R. D. Joseph, and D. M. Ramsey, "An iterative design technique for pattern classification logic," Astropower, Inc., Newport Beach, Calif., EL-6320, July 1963.

[60] D. J. Davis, "An analysis of some failure data," *J. Am. Statist. Assoc.*, vol. 47, no. 258, pp. 113–150.

[61] R. S. Dick, "The reliability of repairable complex systems, pt. A: The similar machine case," *1961 Proc. 5th Mil-E-Con. Symp. on Military Electronics*, pp. 111–150.

[62] ——, "The reliability of repairable complex systems, pt. B: The dissimilar machine case," *IEEE Trans. Reliability*, vol. R-12, pp. 1–8, March 1963.

[63] S. Doctors, "Multipurpose logic devices, pt. I," AC Spark Plug, El Segundo, Calif., Memo. 3249-LA-52, April 1961.

[64] D. C. Dorrough, "A reliability model for digital guidance systems," Douglas Aircraft Company, Inc., SM-47657, June 1964.

[65] ——, "Model theoretic approach to evaluating and enhancing the reliability of complex systems," Douglas Aircraft Company, Inc., SM-47767, December 1964.

[66] ——, "Redundancy and system reliability," Douglas Aircraft Company, Inc., Douglas Paper 3513, presented at the Fall SIAM Meetings, Seattle, Washington, November 1965.

[67] ——, "The logic of reliability measurement," (to be published in *Am. J. of Math.*)

[68] ——, "Minimizing diagnostic time by optimizing the sequence of testable items," Douglas Aircraft Company, Inc., Douglas Paper 4262 (in preparation).

[69] B. Dunham, "The multipurpose bias device, I: The commutator transistor," *IBM J. Research and Develop.*, vol. 1, pp. 116–129, April 1957.

[70] ——, "The multipurpose bias device, II: The efficiency of logical elements," *IBM J. Research and Develop.*, vol. 3, January 1959.

[71] S. J. Einhorn, "Reliability prediction for repairable redundant systems," *Proc. IEEE*, vol. 51, pp. 312–317, February 1963.

[72] B. Epstein and J. Hosford, "Reliability of some two unit redundant systems," *1960 Proc. 6th Nat'l Symp. on Reliability and Quality Control*, pp. 469–477.

[73] B. Epstein and M. Sobel, "Life testing," *J. Am. Statist. Assoc.*, vol. 48, no. 263, pp. 486–502, 1953.

[74] J. D. Esary, "The reliability of coherent systems," in *Redundancy Techniques for Computing Systems*, R. H. Wilcox and W. C. Mann, Eds. Washington, D. C.: Spartan, 1962, pp. 46–61.

[75] J. D. Esary and F. Proschan, "Further results on the reliability of multicomponent structures," presented at the 2nd Conf. on Mathematical Statistics for Reliability, Boeing Scientific Research Laboratory, 1961.

[76] ——, "Relationship between system failure rate and component failure rate," *Technometrics*, vol. 5, no. 2, pp. 183–189.

[77] ——, "Coherent structures of non-identical components," *Technometrics*, vol. 5, no. 2, pp. 191–209.

[78] S. I. Firstman and B. Gluss, "Optimum search routines for automatic fault location," *Operations Research*, 1960.

[79] B. Gluss, "An optimum policy for detecting a fault in a complex system," *Operations Research*, 1959.

[80] J. Goldberg, R. C. Minnick, and W. H. Kautz, "Development of techniques for improving the reliability of digital systems through logical redundancy," Stanford Research Institute for Jet Propulsion Laboratory, Pasadena, Calif., Contract M-44501, 1963.

[81] W. Gore, "System redundancy and information theory," in *Redundancy Techniques for Computing Systems*, R. H. Wilcox and W. C. Mann, Eds. Washington, D. C.: Spartan, 1962, pp. 294–303.

[82] R. D. Joseph, P. M. Kelley, and S. S. Viglione, "An optical decision filter," *Proc. IEEE*, vol. 51, pp. 1098–1118, August 1963.

[83] E. J. Kletsky, "Self-repairing machines," Syracuse University Research Institute, Syracuse, N. Y., Rept. RADC-TR-61-914, 1961.

[84] ——, "Upper bounds on mean life of self-repairing systems," *IRE Trans. Reliability and Quality Control*, vol. RQC-11, pp. 43–48, October 1962.

[85] R. R. Landers, "Achieving higher reliability through self repair," *IEEE Trans. Aerospace*, vol. AS-1, pp. 735–747, August 1963.

[86] R. E. Lyons and W. Vanderkulk, "The use of triple-modular redundancy to improve computer reliability," *IBM J. Research and Develop.*, vol. 6, no. 2, pp. 200–209, 1962.

[87] A. Madansky, "Approximate confidence limits for the reliability of series and parallel systems," Rand Corporation, Santa Monica, Calif., Res. Memo RM-2552, 1960.

[88] W. C. Mann, "Systematically introduced redundancy in logical systems," *IRE Internat'l Conv. Rec.*, vol. 9, pt. 2, pp. 241–263, 1961.
[89] ——, "Restorative processes for redundant computing system," in *Redundancy Techniques for Computing Systems*, R. H. Wilcox and W. C. Mann, Eds. Washington, D. C.: Spartan, 1962, pp. 267–284.
[90] J. Matyas, "Random optimization," *Automation and Remote Control*, vol. 26, no. 2, pp. 244–251, February 1965.
[91] W. S. McCulloch, "The utility of anastomatic nets," in *Redundancy Techniques for Computing Systems*, R. H. Wilcox and W. C. Mann, Eds. Washington, D. C.: Spartan, 1962, pp. 62–65.
[92] J. McReynolds, "Evaluation of the majority principle as a technique for improving digital system reliability," Hycon Eastern, Inc. (now Hermes Electronic Company, a Division of Itek), Cambridge, Mass., HEI Publication M-577, 1958.
[93] J. M. Mendel, "On applications of biological principles of the design of feedback control systems," Douglas Aircraft Company, Inc., SM-47772, November 1964.
[94] ——, "Self-organizing control systems, vol. 2: Open-loop time-optimal control of a stable maneuverable re-entry vehicle," Douglas Aircraft Company, Inc., SM-47904, June 1965.
[95] J. M. Mendel and J. J. Zapalac, "Self-organizing control systems, vol. 3: Off-line training of time-optimal, fuel-optimal and minimum-energy controllers," Douglas Aircraft Company, Inc., SM-51975, February 1966.
[96] E. F. Moore and C. E. Shannon, "Reliable circuits using less reliable relays," *J. Franklin Inst.*, vol. 262, pp. 191–208 and 281–297, 1965.
[97] A. A. Mullin, "On the nature of the reliability of automata," in *Redundancy Techniques for Computing Systems*, R. H. Wilcox and W. C. Mann, Eds. Washington, D. C.: Spartan, 1962, pp. 196–204.
[98] W. H. Pierce, "A proposed system of redundancy to improve the reliability of digital computers," Stanford Electronic Laboratory, Palo Alto, Calif., Tech. Rept. 1552-1, 1960.
[99] ——, "Improving reliability of digital systems by redundancy and adaptation," Stanford Electronic Laboratory, Tech. Rept. 1552-3, 1961.
[100] ——, "Adaptive vote-takers improve the use of redundancy," in *Redundancy Techniques for Computing Systems*, R. H. Wilcox and W. C. Mann, Eds. Washington, D. C.: Spartan, 1962, pp. 229–250.
[101] ——, "Adaptive decision elements to improve the reliability of redundant systems," *IRE Internat'l Conv. Rec.*, pt. 4, pp. 124–131, 1962.
[102] ——, "Redundancy in computers," *Scientific American*, vol. 210, no. 2, 1964.
[103] ——, *Failure-Tolerant Computer Design.* New York: Academic Press, 1965.
[104] F. Proschan, "Redundancy for reliability," in *Statistical Theory of Reliability.* Madison, Wis.: University of Wisconsin Press, 1963, pp. 55–57.
[105] F. Rosenblatt, *Principles of Neurodynamics.* Washington, D. C.: Spartan, 1962.
[106] J. R. Rosenblatt, "On prediction of system performance from information on component performance," *Proc. 1957 Western Joint Computer Conf.*, pp. 85–94.
[107] S. Seshu, "Self-repairing machines, II," Syracuse University Research Institute, Syracuse, N. Y., AD-26396T, 1961.
[108] G. A. Thompson and W. K. Rapp, "A rapid reliability estimator for redundant standby configurations," *1964 Proc. 10th Nat'l Symp. on Reliability and Quality Control*, pp. 573–578, 1964.
[109] J. G. Tryon, "Redundant logic circuitry," U. S. Patent 2 942 193 to Bell Telephone Labs., Inc., 1958.
[110] ——, "Reliability of redundant logic circuits," Bell Telephone Laboratories, N. J., Tech. Memo 59-4114-25, 1959.
[111] ——, "A redundant logical circuitry of high reliability," Bell Telephone Laboratories, N. J., Tech. Memo 59-4114-24, 1959.
[112] ——, "Quadded logic," in *Redundancy Techniques for Computing Systems*, R. H. Wilcox and W. C. Mann, Eds. Washington, D. C.: Spartan, 1962, pp. 205–228.
[113] E. F. Veal, "On symmetry in Boolean functions and the sizes of interchange classes," Douglas Aircraft Company, Inc., Douglas Paper 4486, May 1967.
[114] J. von Neumann, "Probabilistic logics and the synthesis of reliable organisms from unreliable components," in *Automata Studies, Ann. of Math. Studies*, no. 34. Princeton, N. J.: Princeton University Press, 1956.
[115] B. Widrow and J. B. Angell, "Reliable, trainable networks for computing and control," *Aerospace Engineering*, vol. 21, no. 27, p. 78, 1962.
[116] B. Widrow and W. H. Pierce, "Improvements of reliability by redundancy and adaptation," Stanford Electronic Laboratories, Palo Alto, Calif., Consolidated Quarterly Status Rept. 6, 1960.
[117] B. Widrow, W. H. Pierce, and J. B. Angell, "Birth, life, and death in microelectronic systems," *IRE Trans. Military Electronics*, vol. MIL-5, pp. 191–201, July 1961.
[118] M. V. Wilkes, "Self-repairing computers," *IRE Trans. Electronic Computers*, vol. EC-10, pp. 93–94, March 1961.
[119] J. J. Zapalac, "Adaptive processes in decision making," Douglas Aircraft Company, Inc., SM-45921, April 1964.
[120] ——, "Self-organizing control systems, vol. 1: On adaptive computers," Douglas Aircraft Company, Inc., SM-47857, July 1965.