EES SERIES REPORT NO. 9

NsG-646

HYBRID COMPUTERS AND MONTE-CARLO TECHNIQUES

1964-1965

ENGINEERING EXPERIMENT STATION
COLLEGE OF ENGINEERING
THE UNIVERSITY OF ARIZONA, TUCSON

EES SERIES REPORT NO. 9

HYBRID COMPUTERS AND MONTE-CARLO TECHNIQUES

1964 - 1965

Engineering Experiment Station
College of Engineering
The University of Arizona, Tucson

Granino Korn was born in Berlin, Germany. He graduated from Brown University in 1942, obtained his M.A. in physics from Columbia in 1943, and his Ph.D. from Brown in 1948.

Like many others Dr. Korn received his introduction to computers in the U. S. Navy (Special Devices Division). Next came the development of early analog computers and guidance work at Sperry and Curtiss-Wright/Columbus, with seven months on loan to Boeing.

With his attractive and talented wife, Terry, Granino wrote the first text on electronic analog computers in 1952. After a period as staff engineer with Lockheed, Granino again collaborated with Terry, this time as G. A. and T. M. Korn, Consultants, and produced the *Mathematical Handbook for Scientists and Engineers, Electronic Analog Computers* (second edition), and two children, Anne and John.

Since 1957 Dr. Korn has been teaching computers and statistical communications theory at the University of Arizona in Tucson. He has also taught in Mexico, gone to Chile as a consultant to the National Academy of Sciences, and to Europe to participate in AsICA activities.

In addition to his university interests, he has worked with Dr. Harry Huskey on the *Computer Handbook* and with Terry on *Electronic Analog and Hybrid Computers,* and on *Random-Process Simulation and Measurements.*

Dr. Korn is a member of SCi and Chairman of the Editorial Review Board of SIMULATION.

This article was the basis for an oral presentation by the same title at the IFIP Congress '65.

Hybrid computer Monte Carlo techniques

GRANINO A. KORN
University of Arizona
Tucson, Arizona

ABSTRACT

This report introduces hybrid analog-digital computer systems suitable for high-speed Monte-Carlo studies and suggests methods for further reducing the computing time, such as sequential estimation and a number of variance-reducing techniques. Shift-register-generated pseudo-random-noise is especially convenient for many hybrid analog-digital computations. Applications to the solution of partial differential equations and to random-search optimization are outlined.

INTRODUCTION

1. *Survey*

The repetitive-analog-computer Monte-Carlo method, originally developed by Hall, Van der Velde, and others at MIT, has come of age with the advent of fast hybrid analog-digital computer systems which permit very convenient digital accumulation of statistics taken over thousands of fast analog-computer runs (figure 1). The resultant *hybrid-computer Monte-Carlo method* is, for practical purposes, the only method for studies of really complex nonlinear dynamic systems with random inputs, initial conditions, or parameters. It is interesting to note that hybrid-computer averaging operations usually require relatively few bits in the analog-to-digital converters, since quantization errors tend to average out.[40,41] Another interesting development is the use of hybrid analog-digital pseudo-random-noise generators (shift-register sequence generators) instead

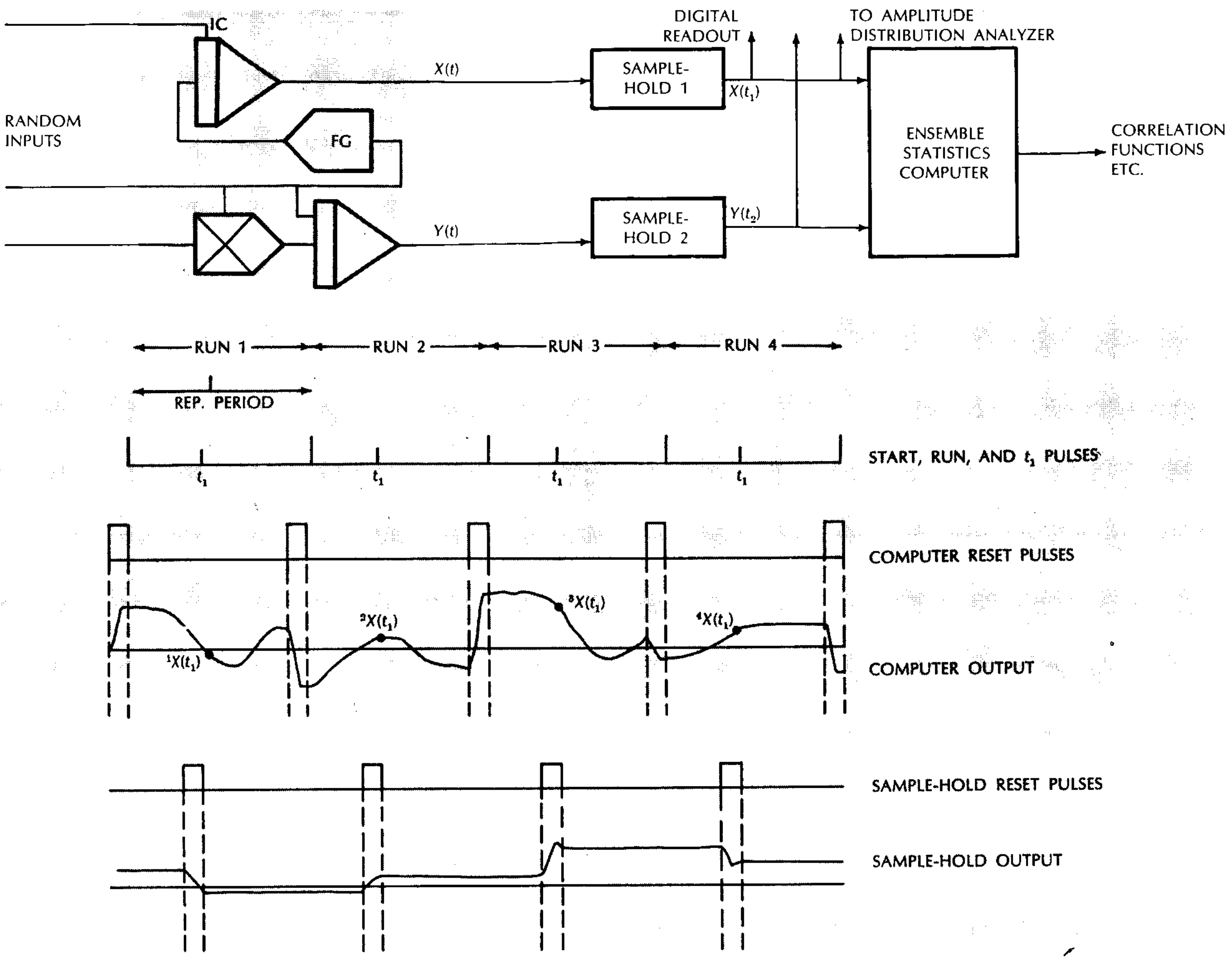


Figure 1 — The hybrid-computer Monte-Carlo technique. Successive analog-computer solutions $x(t)$, $y(t)$ are sampled, and statistics are computed with hybrid analog-digital computing devices.

of random-noise generators.[8,12,41] It is the purpose of this report to scan special computing methods and applications destined to increase the importance of hybrid-computer random-process simulation. Hybrid-computer circuits really suitable for fast statistical computation are still relatively new (section 2), promising techniques are as yet untried, and many existing results are derived from feasibility studies rather than from actual long-term computing experience. A number of interesting problems invite further theoretical and experimental research.

The 50-cps to 1,000-cps iteration rates of modern iterative differential analyzers can generate large statistical samples quickly, but intuitive interpretation by a human operator, as well as automatic cross-plotting and optimization of statistics, places a premium on still faster computation. Hence, we should like to make sample sizes as small as practical for acceptable statistical errors and confidence levels. Sample-size reduction is, of course, doubly important if computing accuracy requires slower analog and/or digital computation. Theoretical prediction of estimate variances, and thus of sample-size requirements, is practically impossible in most applications, so that we recommend following the example of classical statisticians in estimating sample variances or other measures of dispersion concurrently with estimate computation. We can, then, form an idea of the confidence levels corresponding to given accuracy requirements; we may, in fact, terminate data accumulation when a certain confidence level is reached (*sequential estimation*, sections 3 and 4). An additional possibility, also derived from classical experiment-design methods, is the reduction of estimate variances through the use of "doctored" samples designed to represent the ideal theoretical ensemble more accurately than a sample picked truly at random (section 5).

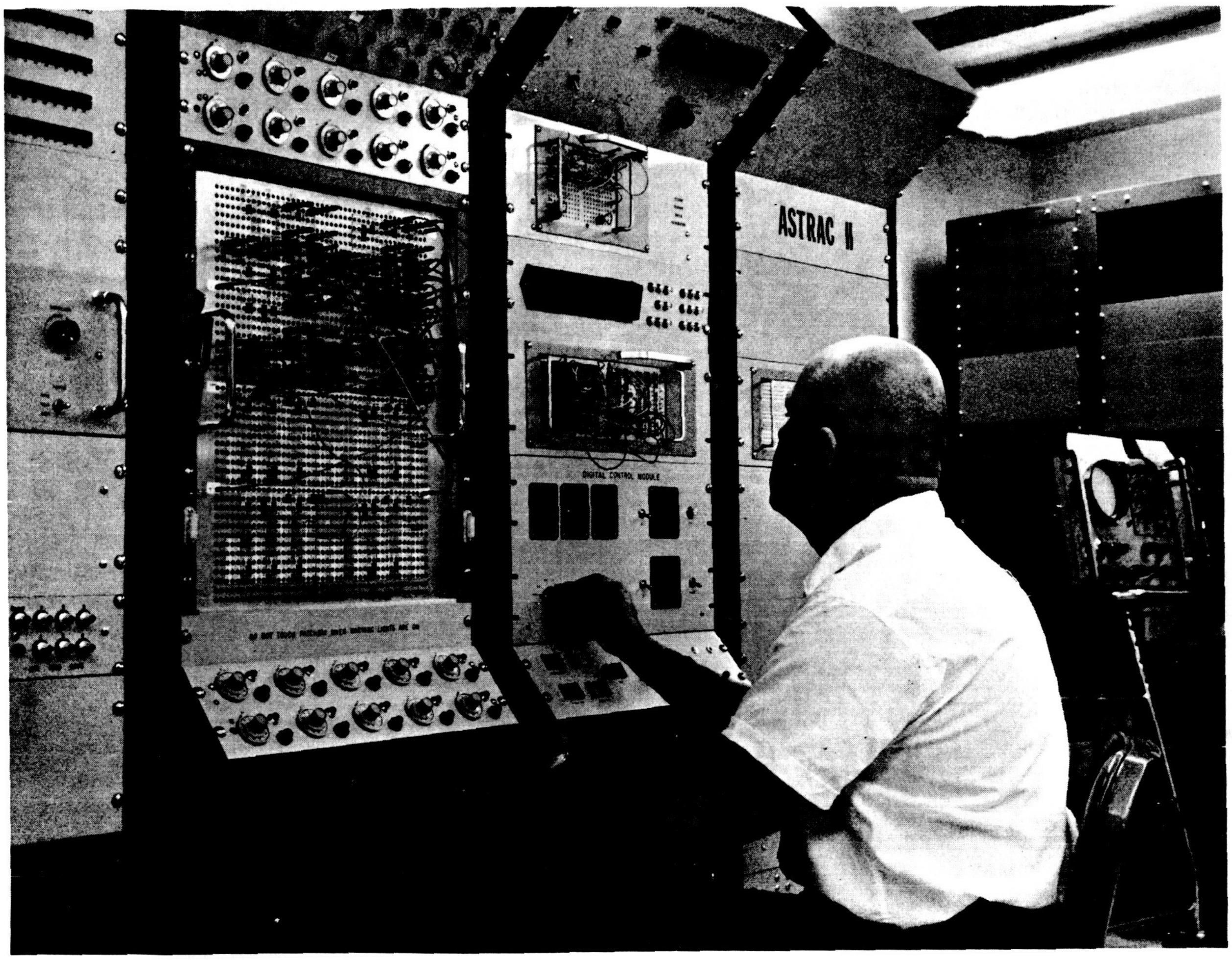


Figure 2 — ASTRAC II console. All analog computing elements other than coefficient potentiometers plug directly into the rear of the shielded analog patchbay.

2. *Fast analog-hybrid computation.* The ASTRAC II System.[1-20,39]

Every analog computer capable of repetitive operation can be adapted to statistical computations. But very high computing speed is desirable. The essential reason for ultrafast computation is not just the sheer mass of statistical data required, but the intuitive insight we gain by realizing the effects of system and parameter changes *on statistics* almost instantaneously. To simulate a dynamical system, say a control system, one thousand times per second, we must solve one to 20 linear or nonlinear differential equations once every millisecond and then reset the analog computer for the following run within 10 to 100 μsec. We require not only analog computing elements (summers, integrators, multipliers, etc.) capable of operating on 1- to 100-Kc signals with acceptable errors, but also integrator, track-hold, and analog-comparator timing within 20 to 100 nsec.

To meet such specifications, the University of Arizona's ASTRAC II iterative differential analyzer (figure 2)* employs $\pm$10v transistor amplifiers with a unity-gain bandwidth beyond 20 Mc and extremely low computing impedances. Summing-resistor values are 5K and 1K, as compared to the 1M and 100K resistors employed in "slow" analog computers. In addition, all amplifiers, multipliers, etc. are plugged directly into the rear of a shielded patchbay, while diode-function-generator networks plug into the storable problem boards. This system eliminates signal-wiring capacitances and crosstalk. Table 1 summarizes ASTRAC II performance.

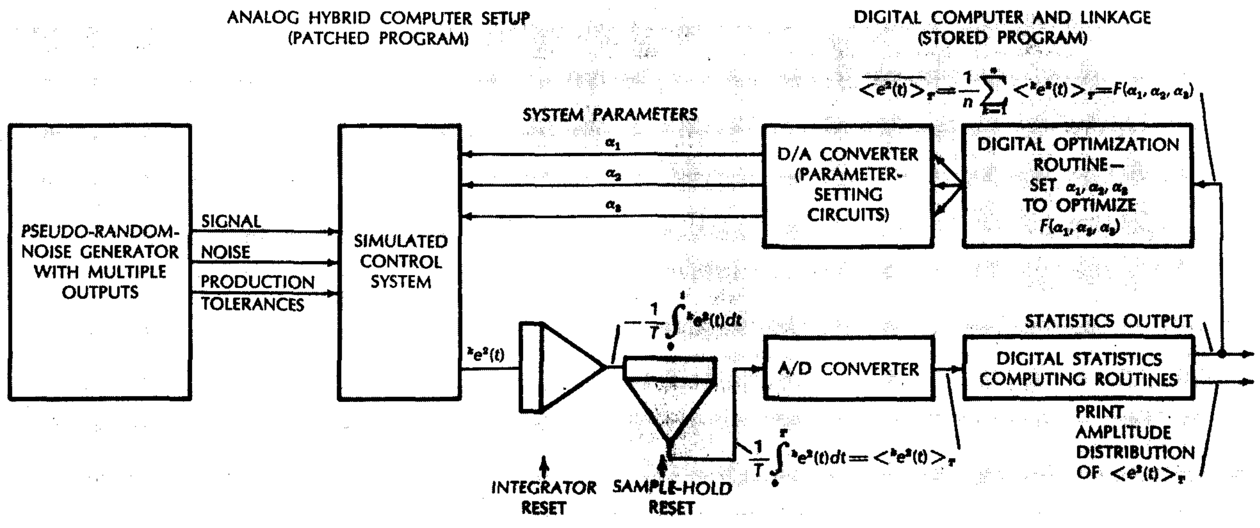
*ASTRAC stands for Arizona Statistical Repetitive Analog Computer. ASTRAC I was a vacuum-tube predecessor[1,11] of the all-solid-state ASTRAC II.

Table 1 – ASTRAC II performance data[15,18-20]

Amplifiers:	$\pm$10v, 50ma at dc; $\pm$10v, 30ma at 1 Mc
	Dc gain, $>10^6$ (120 db) with chopper stabilization
	Drift, 10 μv/°C, 1 na/°C with chopper stabilization
	0-db bandwidth, >20 Mc, 6 db/octave rolloff
	Unity-inverter phase error, 0.16% (0.09°) at 16 Kc
	Integrator phase error, 0.08% (0.05°) at 16 Kc
	Resistance ratios specified within 0.2%
Integrator/track-hold circuits	($C=1\mu$f, 0.1μf, or 0.01μf; data for $C=0.01\mu$f):
	Phase error in TRACK, <0.5% (0.3°) at 10 Kc
	Holding error, $\leq$0.1% of half-scale in 1 msec
	Switching time into HOLD or COMPUTE, <80 nsec; two circuits switch within 20 nsec of one another.
Diode quarter-square multipliers:	Static error <0.2% of half-scale
	Drift, <0.7 mv/°C
	Dynamic error, 0.5% of half-scale at 10 Kc (0.3° phase error)
Analog comparators:	Chopper-stabilized; response time, <120 nsec
	Static hysteresis, $\pm$15 mv
	Drift, <25 μv/°C

ASTRAC II employs digital circuits for iteration control and statistics accumulation. Integrator and track-hold mode control, comparator outputs, analog switches, and free logic elements are terminated in a *control patchbay*. A *digital control panel* permits the operator to control repetition rate, computer-run length, readout timing, and sample sizes for statistical computations.[10,39] A *resettable 25-bit shift-register pseudo-random-noise generator*[12] can produce up to four digital or analog outputs and has its own small patchbay with removable patchboards. Another small patchbay controls a packaged *parameter optimizer* with a variety of optimization strategies (see also section 7).

ASTRAC II also has built-in *analog/digital circuits for computing averages, mean squares and probability estimates,* but the ideal way to process statistical data obtained from fast analog-computer experiments is with a small stored-program digital computer in the \$10-20,000-class (e.g., Digital Equipment Corporation PDP 8 or Computer Control Corporation DDP116), as shown in figure 3. Digital programs are conveniently stored on punched tape. Digital data processing not only yields several estimates (e.g., mean value, mean square, and amplitude distribution) from each sample, but also permits more sophisticated methods of sequential estimation (section 3). *Since sampling, data conversion, and digital operations repeat only once or twice per analog-computer run, there are no stringent speed requirements on linkage and digital computer.* The resulting hybrid system is a much happier (and less expensive) combination than those now commonly employed for combined analog-digital simulation.



ANALOG HYBRID COMPUTER SETUP (PATCHED PROGRAM)
DIGITAL COMPUTER AND LINKAGE (STORED PROGRAM)
$\overline{<e^2(t)>_T} = \frac{1}{n}\sum_{k=1}^{n} <^k e^2(t)>_T = F(\alpha_1, \alpha_2, \alpha_3)$
SYSTEM PARAMETERS
α_1
α_2
α_3
D/A CONVERTER (PARAMETER-SETTING CIRCUITS)
DIGITAL OPTIMIZATION ROUTINE— SET $\alpha_1, \alpha_2, \alpha_3$ TO OPTIMIZE $F(\alpha_1, \alpha_2, \alpha_3)$
PSEUDO-RANDOM-NOISE GENERATOR WITH MULTIPLE OUTPUTS
SIGNAL
NOISE
PRODUCTION TOLERANCES
SIMULATED CONTROL SYSTEM
$^k e^2(t)$
$-\frac{1}{T}\int_0^t {}^k e^2(t)dt$
STATISTICS OUTPUT
A/D CONVERTER
DIGITAL STATISTICS COMPUTING ROUTINES
PRINT AMPLITUDE DISTRIBUTION OF $<e^2(t)>_T$
$\frac{1}{T}\int_0^T {}^k e^2(t)dt = <^k e^2(t)>_T$
INTEGRATOR RESET
SAMPLE-HOLD RESET

Figure 3 – This example illustrates the possibilities of combining an ASTRAC II-type iterative analog computer with a small stored-program digital data processor. Analog computing elements produce successive samples of the squared-error time average $<^k e^2(t)>_T = (1/T)\int_0^T {}^k e^2(t)\,dt$ for a simulated control system with random input and noise. The digital computer finds successive values of the sample average $\overline{<^k e^2(t)>_T} = (1/n)\sum_{k=1}^{n} <^k e^2(t)>_T$ for $n = 100$ to 10,000 computer runs and changes control-system parameters so as optimize the sample average. Note that each analog/digital and digital/analog conversion and digital-computation sequence is required only once per analog-computer run.

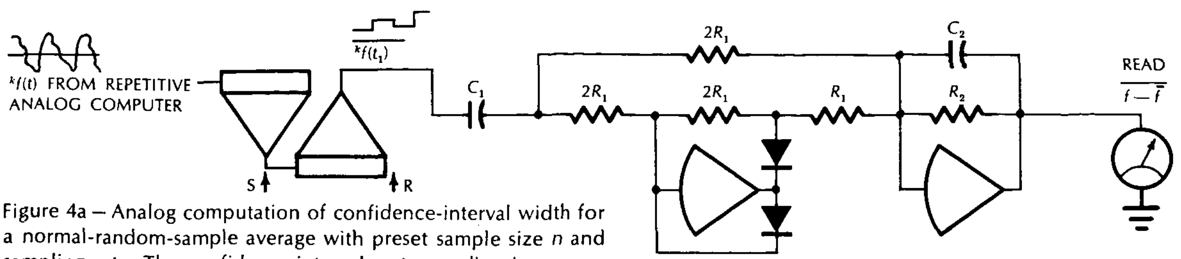


Figure 4a – Analog computation of confidence-interval width for a normal-random-sample average with preset sample size n and sampling rate. The confidence-interval-meter reading is proportional to an EWP (exponentially-weighted-past) average approximation $\overline{|f-\bar{f}|}$. The time constants R_1C_1, R_2C_2 must be matched to repetition rate and sample size.

SOME SPECIAL MEASUREMENT AND SAMPLING TECHNIQUES FOR STATISTICAL SIMULATION

3. *Confidence-interval measurement and sequential estimation*

(a) *Random-sample averages from approximately Gaussian data*

Consider the estimation of an ensemble average Ef by the sample average $\bar{f}=\frac{1}{n}\sum_{k=1}^{n} {}^kf$ over a random sample $({}^1f, {}^2f, \ldots, {}^nf)$ obtained in n computer runs. To gauge the sample size n required for a given accuracy and confidence level,[38] let us assume that the random variable f is at least approximately Gaussian; if necessary, we will further approximate this condition by preaveraging (section 4). For a normal (Gaussian) random sample, symmetrical confidence limits $\bar{f} \pm d$ for $E\{f\}$ are defined by[38]

$$d=t_{1-\alpha/2}\sqrt{\frac{\overline{(f-\bar{f})^2}}{n-1}} \tag{1}$$

where $t_{1-\alpha/2}$ is found from a t-distribution table with $n-1$ degrees of freedom for each given confidence level $1-\alpha$. To obtain either the half-width d of a confidence interval for given α or the confidence level $1-\alpha$ for a given acceptable error d, we must measure the sample variance $\overline{(f-\bar{f})^2}$, the sample standard deviation $\sqrt{\overline{(f-\bar{f})^2}}$, or, if this is easier, the sample mean absolute deviation $\overline{|f-\bar{f}|}$ concurrently with the sample average $\bar{f}$; note that

$$\sqrt{E\{[f-E\{f\}]^2\}}=\sqrt{\frac{\pi}{2}}\,E\{\,|f-E\{f\}|\,\} \tag{2}$$

for normal random samples, so that it may be permissible to employ the approximate relation

$$\sqrt{\overline{(f-\bar{f})^2}}\approx\sqrt{\frac{\pi}{2}}\,\overline{|f-\bar{f}|} \tag{3}$$

for $n > 30$. Confidence-interval estimation need not be accurate, since d is usually at most a few percent of the sample range.

For a preset sample size n, equation (1) permits us to compute and display either d or α from measured values of $\overline{(f-\bar{f})^2}$ or $\overline{|f-\bar{f}|}$. Figure 4a shows the circuit of a simple "confidence-interval meter," which may be considered as an accessory for a sample-averaging computer. A more sophisticated approach is to compute the sample variance $\overline{(f-\bar{f})^2}$ for each successive value of n (or, say, for every 10th n), and to terminate data-taking as soon as

$$\overline{(f-\bar{f})^2} < (n-1)\left(\frac{d}{t_{1-\alpha/2}}\right)^2 \tag{4}$$

for preset values of d and α (*sequential estimation*).

Sequential estimation is most convenient with a digital statistics computer. Computation of

$$^n\bar{f}=\frac{1}{n}\sum_{k=1}^{n} {}^kf \qquad {}^ns^2={}^n\overline{(f-\bar{f})^2}=\frac{1}{n}\sum_{k=1}^{n}({}^kf-{}^n\bar{f})^2 \tag{5}$$

can utilize the recurrence relations

$$^n\bar{f}={}^{n-1}\bar{f}+\frac{1}{n}({}^nf-{}^{n-1}\bar{f})$$

$$^ns^2={}^{n-1}s^2+\frac{1}{n}[({}^nf-{}^n\bar{f})^2-{}^{n-1}s^2]$$

$$={}^{n-1}s^2+\frac{1}{n}\left[({}^nf-{}^{n-1}\bar{f})-\frac{1}{n}({}^nf-{}^{n-1}\bar{f})\right]^2-{}^{n-1}s^2 \tag{6}$$

We note that fixed-point computation of $\bar{f}=1/R\sum_{k=1}^{n} {}^kf$ or with small 12- or 16-bit digital computers will, in any case, require n divisions by n (or rescaling) to prevent accumulator overloads, so that the formula (6) does not add much extra cost or computing time.

Continuous analog computation of the finite-time averages

$$<f(t)>_t=\frac{1}{t}\int_0^t f(t)dt \quad s^2(t)=<[f(t)-<f(t)>_t]^2>_t$$

$$=\frac{1}{t}\int_0^t [f(t)-<f(t)>_t]^2\,dt \tag{7}$$

is similarly obtainable through solution of the differential equations

$$\frac{d}{dt}<f(t)>_t=\frac{1}{t}[f(t)-<f(t)>_t]$$

$$\frac{d}{dt}s^2(t)=\frac{1}{t}\{[f(t)-<f(t)>_t]^2-s^2(t)\} \tag{8}$$

for $<f(t)>_t$ and $s^2(t)$. Division by $t=0$ can be implemented by a steepest-descent division loop,[39] or computation can be started at $t=t_1$ with the initial conditions $<f(t)>_{t_1}=f(t_1)$, $s^2(t_1)=0$; the resulting error will be small for $t_1 << t$.

(b) *Probability measurements*

Confidence intervals for probability measurements from random samples are easily derived from the binomial distribution. For given sample size n and confidence level $1-\alpha$, the confidence-interval width for the unknown probability is a function of the estimate (statistical relative frequency) itself.[42] Sequential estimation of probabilities is, again, conveniently implemented with a small digital computer; figure 4b shows a simple analog confidence-interval meter suitable for use with hybrid-computer amplitude-distribution analyzers.

4. *Pre-averaging methods*

Confidence-interval measurements and sequential estimation based on the assumption of approximately Gaussian data become much more generally applicable if averages over a random sample ${}^1f, {}^2f, \ldots, {}^nf$ are computed as samples over "pre-averages," *i.e.*

$$\bar{f}=\frac{1}{n}({}^1f+{}^2f+\ldots+{}^nf)$$

$$=\frac{m}{n}\left[\frac{1}{m}({}^1f+{}^2f+\ldots+{}^mf)\right. \qquad (9)$$

$$\left.+\frac{1}{m}({}^{m+1}f+{}^{m+2}f+\ldots+{}^{2m}f)+\ldots\right]$$

The set of pre-averages $\frac{1}{m}({}^1f+{}^2f+\ldots+{}^mf)$,

$\frac{1}{m}({}^{m+1}f+{}^{m+2}f+\ldots+{}^{2m}f),\ldots$

very often becomes a useful approximation to a normal sample of size n/m; $m=8$ has proved to be a practical choice.[21] Note that pre-averaging in accordance with equation (9) will not affect the value of our sample average $\bar{f}$, but will change the sample size together with the sample variance or other dispersion measures used for confidence-level estimation (see also the note at the end of this article).

Computation of pre-averages and dispersion measures is, again, most conveniently accomplished if a stored-program digital computer is available for data processing. As an alternative, the track-hold circuits used for random-process sampling in hybrid computers can implement pre-averaging (track-hold accumulator, reference 39).

5. *Variance reduction by special sampling techniques*

In principle, every Monte-Carlo computation may be considered as the estimation of a suitable integral

$$I=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty} f(\lambda_1,\lambda_2,\ldots,\lambda_N)\,dP(\lambda_1,\lambda_2,\ldots,\lambda_N) \qquad (10)$$

by a random-sample average

$$\overline{f(x_1,x_2,\ldots,x_N)}=\frac{1}{n}\sum_{k=1}^{n} f({}^kx_1,{}^kx_2,\ldots,{}^kx_N) \qquad (11)$$

where $(x_1,x_2,\ldots,x_N)$ is a (generally multidimensional, $N>1$) random variable with known distribution function $P(x_1,x_2,\ldots,x_N)$; for random-process studies involving, say a flat-spectrum noise input of bandwidth B for T seconds, N can be of the order of $2BT$. For simplicity, we shall base the following discussion on Monte-Carlo estimation of the *one-dimensional* integral

$$I=\int_{-\infty}^{\infty} f(\lambda)dP(\lambda) \qquad (12)$$

by

$$\overline{f(x)}=\frac{1}{n}\left[f({}^1x)+f({}^2x)+\ldots+f({}^nx)\right] \qquad (13)$$

although we would, most probably, employ Monte-Carlo computation in the one-dimensional case only if we required direct simulation for a partial system test, or to gain intuitive insight. Note that each calculation of $f({}^kx)$ may involve solution of differential equations.

The variance of our estimate (13) of (12) on the basis of a random sample, $({}^1x,{}^2x,\ldots,{}^nx)$ is

$$\mathrm{Var}\{\overline{f(x)}\}=\frac{1}{n}\mathrm{Var}\{f(x)\} \qquad (14)$$

so that the rms fluctuation decreases only as $1/\sqrt{n}$ with increasing n. The estimate variance is due to the random fluctuation in the distribution of different samples $({}^1x,{}^2x,\ldots,{}^nx)$. We will now attempt to "doctor" the sample $({}^1x,{}^2x,\ldots,{}^nx)$ so as to reduce these fluctuations, while still preserving the relation

$$E\{\overline{f(x)}\}=E\{f(x)\}=I \qquad (15)$$

i.e., without biasing our estimate.

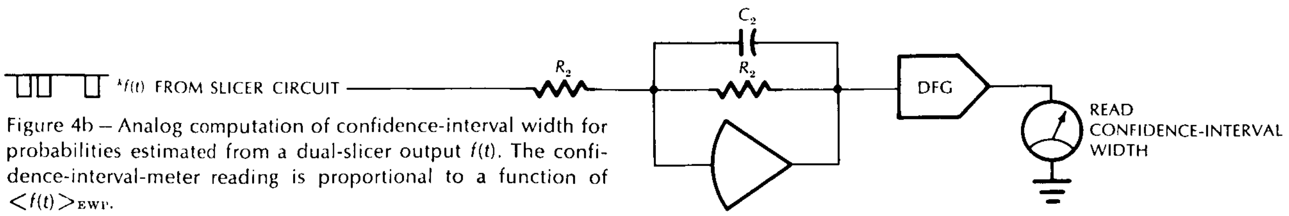


Figure 4b – Analog computation of confidence-interval width for probabilities estimated from a dual-slicer output $f(t)$. The confidence-interval-meter reading is proportional to a function of $\langle f(t)\rangle_{\mathrm{EWP}}$.

(a) *Stratified sampling*[27]

We divide the range of the random variable x into a number of suitably chosen class intervals $\xi_{j-1} < x \leq \xi_j$ and agree to fix the number n_j of otherwise independent sample values ${}^kx = {}^ix_j$ $(i = 1, 2, \ldots, n_j)$ falling into the j^{th} class interval. Assuming *a priori* knowledge of the probabilities

$$P_j = \text{Prob}\,[\xi_{j-1} < x \leq \xi_j] = P(\xi_j) - P(\xi_{j-1}) \qquad (16)$$

associated with our class intervals (*e.g.*, on the basis of symmetry, uniform distribution, etc.), we can employ the *stratified-sample* average

$$f(x)_{\text{STRAT}} = \sum_j P_j \frac{1}{n_j} \sum_{i=1}^{n_j} f({}^ix_j) \qquad (17)$$

as an unbiased estimate of I, with

$$\text{Var}\,\{\overline{f(x)}_{\text{STRAT}}\} = \sum_j \frac{P_j^2}{n_j} \text{Var}\,\{f({}^ix_j)\} \qquad (18)$$

Note that repeated stratified samples will differ only within class intervals. The variance (18) can be smaller than the random-sample variance $\text{Var}\{\overline{f(x)}\}/n$ with $n = \sum_j n_j$ if *a priori* information permits a favorable choice of the ξ_j and n_j. In principle, it would be best to choose class intervals for equal variances

$$\text{Var}\,\{f({}^ix_j)\} = \frac{1}{P_j} \int_{\xi_{j-1}}^{\xi_j} f^2(\lambda)dP(\lambda) - \left[\frac{1}{P_j} \int_{\xi_{j-1}}^{\xi_j} f(\lambda)dP(\lambda)\right]^2 \qquad (19a)$$

and then to assign the theoretically correct number of samples to each class interval, *i.e.*,

$$n_j = nP_j \qquad (19b)$$

In this ideal case, we should have the relatively small estimate variance

$$\text{Var}\,\{f(x)_{\text{STRAT}}\} = \frac{1}{n} \text{Var}\,\{f({}^ix_j)\} \qquad (20)$$

As the class intervals are decreased, the stratified-sampling techniques will produce results analogous to that of an integration formula, but ordinarily the class intervals are larger; practical applications are usually multidimensional, so that simple symmetry relations may yield favorable class intervals.

(b) *Use of correlated samples*[26,27]

If individual sample values ${}^kx$ are not statistically independent (as they would be in a true random sample), the expression (14) for our estimate variance is replaced by

$$\text{Var}\{\overline{f(x)}_{\text{CORREL}}\} = \frac{1}{n} \text{Var}\,\{x\} + \frac{2}{n^2} \sum\sum_{i<k} \text{Cov}\,\{{}^ix, {}^kx\} \qquad (21)$$

Judiciously introduced *negative correlation* between selected sample-value pairs ${}^ix$, ${}^kx$ will produce negative covariance terms in equation (21) and may reduce the variance well below the random-sample variance $\text{Var}\,\{f(x)\}/n$ without biasing the estimate.

As a simple example,[27] let x be uniformly distributed between $x = 0$ and $x = 1$, and let $f(x)$ be the monotonic function $(e^x - 1)/(e - 1)$. We design our sample so that n is even, and ${}^2x = 1 - {}^1x$, ${}^4x = 1 - {}^3x$, $\ldots$, ${}^nx = 1 - {}^{n-1}x$, with sample values otherwise independent. Since $f(x)$ and $f(1 - x)$ are negatively correlated, we find

$$\text{Var}\,\{\overline{f(x)}_{\text{CORREL}}\} \approx \frac{1}{31} \text{Var}\,\{\overline{f(x)}\}$$

so that the rms fluctuation is reduced by a factor of about 5·6. In addition, the correlated sample requires us to generate fewer random numbers. More interesting applications are, again, to multidimensional problems.[27] Note that stratified sampling, in effect, also introduces negative correlation between sample values: ${}^{k+1}x$ can no longer fall into a given class interval if ${}^kx$ has filled the latter.

(c) *Use of pseudo-random samples*

Instead of constructing and possibly recording stratified and/or correlated samples for Monte-Carlo computations, we may utilize special properties of pseudo-random-noise sequences. In particular, the shift-register states of a maximal-length shift-register generator[12] with r stages correspond to the r-digit binary numbers between (and not including) 0 and $2^r - 1$, and one shift-register-sequence period produces each of these numbers exactly once. At least for low-dimensional random inputs, shift-register pseudo-random-noise generators can thus supply uniformly distributed stratified (and correlated) samples if we sample over an integral number of shift-register periods; more general distributions can be obtained with the aid of function generators.[41]

(d) *Use of a priori information: Importance sampling*

As a matter of principle, Monte-Carlo computations often can and should be simplified through judicious application of partial *a priori* knowledge of results.[27] As a case in point, *importance-sampling* techniques attempt to estimate an integral (12) by a sample average $f(y)/g(y)$, where y is a random variable with probability density

$$P_y(y) = g(y) \frac{dP(y)}{dy} \qquad (22)$$

The estimate is easily seen to be unbiased. The function $g(y)$ is chosen so that

$$\text{Var}\left\{\frac{f(v)}{g(y)}\right\} = E\left\{\left[\frac{f(v)}{g(y)} - I\right]^2\right\} \qquad (23)$$

is small, subject to the constraint $\int_{-\infty}^{\infty} p_y(y)\,dy = 1$. In particular, $g(y) = f(y)/I$ would reduce the variance (23) to zero, but this would require knowledge of the unknown quantity I. Importance sampling permits us to "concentrate" sampling near values of y of special interest, e.g., where $f(y)$ varies rapidly.

6. *Hybrid-computer Monte-Carlo solution of partial differential equations*

Some of the earliest investigators of the Monte-Carlo method[21,24] have suggested its application to generalized Dirichlet problems requiring the solution of a quasi-linear partial differential equation

$$a_1\frac{\partial^2 u}{\partial x^2}+a_2\frac{\partial^2 u}{\partial y^2}=K_1(x,y)\frac{\partial u}{\partial x}+K_2(x,y)\frac{\partial u}{\partial y} \qquad (a_1,a_2\geq 0) \tag{24}$$

for the unknown function $u(x,y)$ inside a simple closed contour C of the xy plane, where $u(x,y)$ is given, bounded, single-valued, and piecewise continuous on the boundary C. A special case is the Dirichlet boundary-value problem for the familiar Laplace equation

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=0 \qquad [u=u(x,y)\text{ on }C] \tag{25}$$

In this case, $a_1=a_2=1$, $K_1\equiv K_2\equiv 0$.

A derivation rather too elaborate for inclusion here[21,24,28] shows that *the random walk generated by the solutions x(t), y(t) of the stochastic equations of motion*

$$\frac{dx}{dt}=-K_1(x,y)+X(t) \qquad \frac{dy}{dt}=-K_2(x,y)+Y(t) \tag{26}$$

with starting values $x(0)=x_0$, $y(0)=y_0$ *and independent white-Gaussian-noise forcing functions* $X(t)$, $Y(t)$ *with zero mean and power spectral densities respectively proportional to* a_1, a_2 *will cross the boundary C at random points* (x_c, y_c) *such that*

$$E\{u(x_c,y_c)\}=u(x_0,y_0) \tag{27}$$

Hence, the sample average $u(x_c,y_c)$ *over a suitable number n of random walks is an unbiased estimate for the desired solution* $u(x_0,y_0)$ *at* (x_0,y_0). Simple convergence conditions were derived by Petrowsky.[28]

Chuang, Kazda, and Windeknecht[28] were the first to employ this theorem for analog/hybrid-computer solution of partial differential equations. They solved the ordinary differential equations (26) on a conventional "slow" analog computer with tape-recorded random-noise inputs and applied $x(t)$, $y(t)$ to the horizontal and vertical plates of an oscilloscope. To demonstrate the feasibility of the Monte-Carlo method, they restricted themselves to boundary functions $u(x_c,y_c)$ constant and equal to 100 on a continuous portion C_1 of the boundary C and equal to 0 on the remaining boundary. Boundary crossings of the oscilloscope beam marking the point (x,y) were detected by an arrangement of masks and photocells associated with C and C_1, and $E\{u(x_c,y_c)\}$ could be estimated simply by 100 times the fraction of boundary crossings taking place across C_1, as determined by a decimal counter. Computer and noise source limited computing speed to about one random walk per second; solution errors for samples of 300 to 2,200 runs were of the order of a few percent and were ascribed mainly to statistical fluctuations.

In modern analog/hybrid computers, accurate and convenient combinations of function generators, analog comparators, and digital logic replace the cumbersome photocell circuits used to detect boundary crossings, and complicated boundary functions can be generated and averaged digitally (figure 5). More significantly, ASTRAC II-type iterative differential analyzers can perform the 500 to 2,000 complete

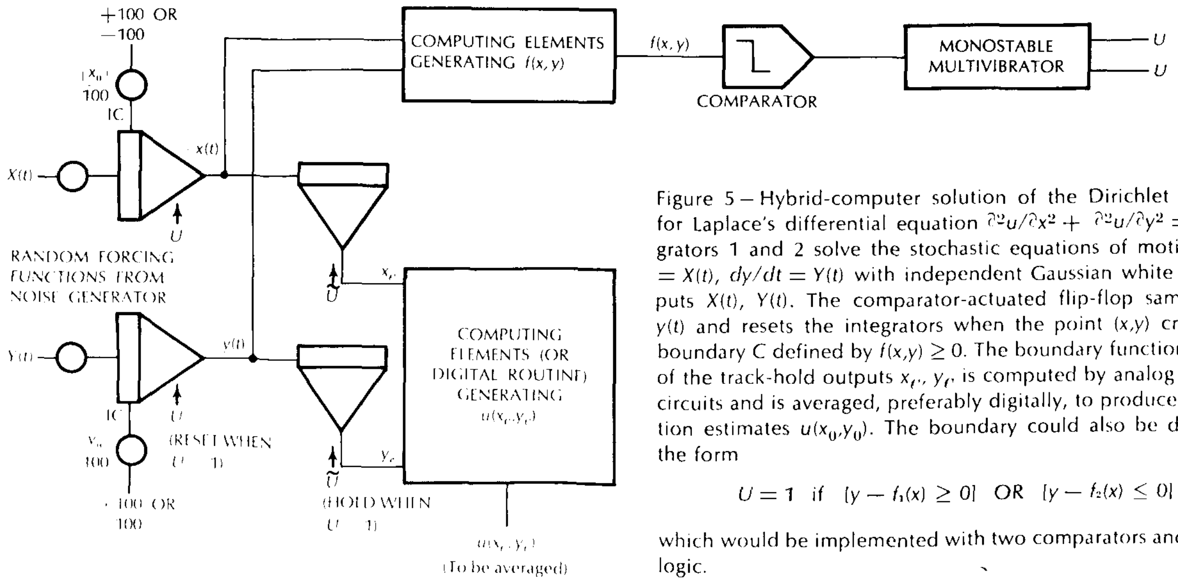

+100 OR −100
x_0 100
IC
X(t)
x(t)
U
RANDOM FORCING FUNCTIONS FROM NOISE GENERATOR
Y(t)
y(t)
IC
U
y_0 100
(RESET WHEN U = 1)
+100 OR −100
COMPUTING ELEMENTS GENERATING f(x, y)
f(x, y)
COMPARATOR
MONOSTABLE MULTIVIBRATOR
U
U
x_c
$\tilde{U}$
y_c
$\tilde{U}$
(HOLD WHEN U = 1)
COMPUTING ELEMENTS (OR DIGITAL ROUTINE) GENERATING $u(x_c, y_c)$
$u(x_c, y_c)$ (To be averaged)

Figure 5 — Hybrid-computer solution of the Dirichlet problem, for Laplace's differential equation $\partial^2u/\partial x^2+\partial^2u/\partial y^2=0$. Integrators 1 and 2 solve the stochastic equations of motion $dx/dt=X(t)$, $dy/dt=Y(t)$ with independent Gaussian white noise inputs $X(t)$, $Y(t)$. The comparator-actuated flip-flop samples $x(t)$, $y(t)$ and resets the integrators when the point (x,y) crosses the boundary C defined by $f(x,y)\geq 0$. The boundary function $u(x_c,y_c)$ of the track-hold outputs x_c, y_c is computed by analog or digital circuits and is averaged, preferably digitally, to produce the solution estimates $u(x_0,y_0)$. The boundary could also be defined in the form

$$U=1 \text{ if } [y-f_1(x)\geq 0] \text{ OR } [y-f_2(x)\leq 0]$$

which would be implemented with two comparators and patched logic.

random walks required for each solution point within one to three seconds; it is this fact which may make the hybrid-computer Monte-Carlo method competitive with other methods of solving partial differential equations. The Monte-Carlo method, still largely unexplored because of insufficient computing speeds with earlier equipment, offers a number of intriguing possibilities:

1. Unlike other solution methods, the Monte-Carlo method permits us to compute the solution $u(x, y)$ *only at specific desired points* (x, y) *of interest.*
2. Computer setups such as those in figure 5 are easily generalized to apply to *three-dimensional problems.* Relatively little additional equipment is required, while conventional methods of solving partial differential equations would become radically more complicated.
3. After the solution is computed for a number of points, it is often possible to simplify computation of the solution at a point (x, y) by averaging over solution values $u(x, y)$ computed earlier for surrounding points. In particular, the solution $u(x, y)$ of Laplace's equation equals the average of solution values over any circle centered at x, y.[25]
4. Solution time can be reduced through sequential estimation (section 3) and by the variance-reducing techniques of section 5. In particular, shift-register noise generators can generate successive negatively correlated random walks for variance reduction. The shift-register can, for instance, be reset to produce each pseudo-random-noise sequence twice[12,41] and we can invert one sequence of each pair to introduce negative correlation between them.

Finally, the application of Monte-Carlo methods to more general classes of partial differential equations[25,28] and more general random forcing functions presents a fascinating field for future research.

At least in principle, one sample of random walks starting at (x_0, y_0) defines the solution of the partial differential equation (24) at (x_0, y_0) for *all* admissible boundary functions $u(x_C, y_C)$. If we divide the boundary C into small arcs centered at the boundary points $(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)$ and record the fractions h_k of the total number of boundary crossings falling into the kth arc, then

$$E\{u(x_0, y_0)\} \approx \sum_{k=1}^{m} h_k u(x_k, y_k) \tag{28}$$

7. *Random-search methods for parameter optimization*

The most commonly used computer methods for determining the parameter values $\alpha_1, \alpha_2, \ldots$ which will optimize (minimize) a differentiable criterion function $F(\alpha_1, \alpha_2, \ldots)$ involve computation or measurement of the gradient components

$$\frac{\partial F}{\partial \alpha_1} \approx \frac{\Delta F}{\Delta \alpha_1}, \quad \frac{\partial F}{\partial \alpha_2} \approx \frac{\Delta F}{\Delta \alpha_2}, \ldots$$

corresponding to each parameter at a trial point $(\alpha_1, \alpha_2, \ldots)$. Each trial parameter value is then incremented by an amount proportional to the corresponding gradient component, so that the parameter point progresses in the direction of steepest descent (figure 6a); we usually decrease the step size as the minimum is approached. Gradient methods may fail to converge, or converge too slowly, if the criterion-function "hill" has ridges, winding canyons, etc., or if the function $F(\alpha_1, \alpha_2, \ldots)$ is only piecewise differentiable or continuous. In such situations, one may turn to random-search methods which can, in addition, simplify the computation routines. A *pure random search* would simply compute the criterion function at a number of randomly chosen points $(\alpha_1, \alpha_2, \ldots)$ in parameter space and select the parameter point yielding the smallest value of $F(\alpha_1, \alpha_2, \ldots)$. This optimization technique, which does not utilize any known continuity properties of the criterion function, is sometimes employed to find starting values for other optimization methods, but it is essentially impractical by itself. Assume that we have N parameters $\alpha_1, \alpha_2, \ldots, \alpha_N$, each capable of varying between zero and 100 percent, and that we wish to locate a single minimum with only 10 percent accuracy. In this case, the probability of finding the desired minimum in a single trial is 10^{-N}, and the probability of finding the minimum at least once in m independent trials is

$$P = 1 - (1 - 10^{-N})^m \approx 10^{-N} m \tag{29}$$

We are truly looking for a needle in an N-dimensional haystack; with the number N of parameters only as large as five or six, any realistic optimization method must utilize known properties of the criterion function $F(\alpha_1, \alpha_2, \ldots, \alpha_N)$, such as continuity and differentiability. *Optimization by sequential random perturbations* (creeping random search) permits multiparameter optimization with a minimum of control logic.[30-34] Referring for simplicity to the two-parameter example of figure 6b, we start with a trial point $({}^0\alpha_1, {}^0\alpha_2)$ and vary α_1, α_2 simultaneously by independent *random* positive or negative increments $\Delta\alpha_1$, $\Delta\alpha_2$ obtained from a noise generator. If $F(\alpha_1, \alpha_2)$ is not improved, we try new random increments $\Delta\alpha_1$,

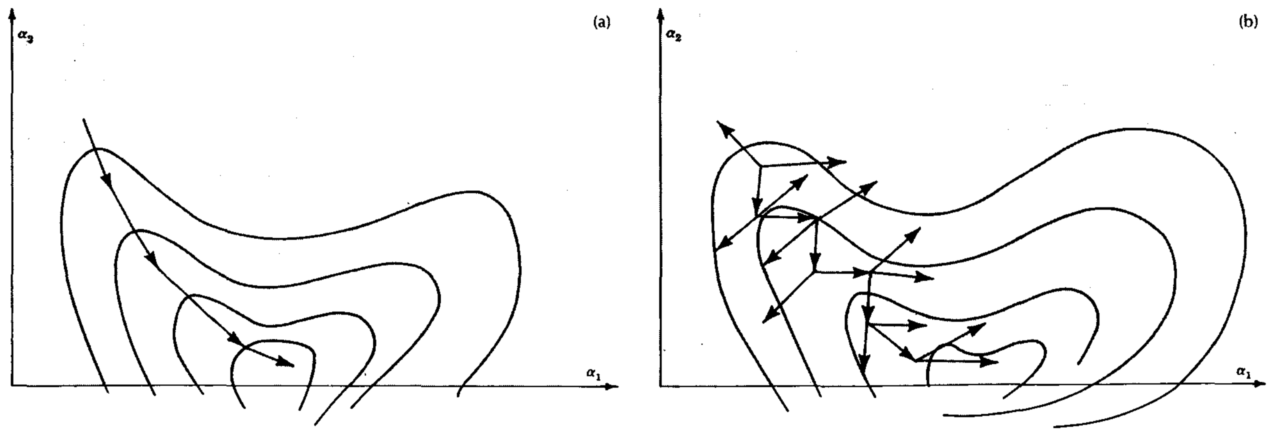


Figure 6 — Two-dimensional parameter optimization ("hill climbing") by a gradient method (a), and by sequential random perturbations (b).

$\Delta\alpha_2$ until an improvement is obtained; then we use $({}^0\alpha_1 + \Delta\alpha_1, {}^0\alpha_2 + \Delta\alpha_2)$ as the next trial point $({}^1\alpha_1, {}^1\alpha_2)$. With random perturbations distributed about zero with a small standard deviation (small step sizes), the iteration will surely converge whenever the gradient method does. Although convergence is slower, our random-perturbation scheme involves no exploration steps, varies all parameters simultaneously, and is not affected by ugly parameter-space terrain features, such as ridges and canyons.

Since unused perturbations $\Delta\alpha_1$, $\Delta\alpha_2$ must be stored and subtracted out, it appears best to restrict perturbation values to h, $-h$ or to h, $-h$, and 0, where h is a suitably chosen step size; increment values can then be stored digitally, say, in flip-flops. A shift-register noise generator is, once again, an especially convenient source of binary random perturbations.

Random perturbation optimization is especially suitable for very fast iterative differential analyzers like ASTRAC II (section 2), where it is conveniently implemented with hybrid analog-digital coefficient-setting circuits (simple D/A converters).[37] The following relatively simple improvements have been shown to speed convergence at the expense of relatively little added digital logic:

1. A hybrid analog-digital parameter-setting circuit makes it easy to change the step size h as a function of past failures or successes, or as a function of F.[37]
2. We can make, say, every 10th or 20th step a large one to detect secondary maxima or minima, or saddle points.[34,37]
3. We can *correlate* successive random perturbations, *i.e.*, we can make perturbations in the direction of the last success more likely than perturbations in the directions of past failures.[37]

Rastrigin[36] has shown that even with randomly-directed perturbations in an N-dimensional parameter space, the expected rate of progress *in the gradient direction* exceeds that of a simple gradient method employing N gradient-determining steps followed by a working step, provided that $N \geq 4$. Rastrigin's result is not conclusive, because both gradient and random-perturbation methods are usually modified by various step-size changing strategies and other maneuvers.[39] His result is, however, suggestive if one considers the relative simplicity of the sequential-perturbation method.

Note on Pre-Averaging

W. Giloi* has suggested that, in view of

$$\frac{E\{\overline{(z-\bar{z})^2}\}}{\frac{n}{m}-1} = \frac{m}{n}\,\mathrm{Var}\,\{z\} = \frac{1}{n}\,\mathrm{Var}\,\{f\} = \frac{E\{\overline{(f-\bar{f})^2}\}}{n-1}$$

confidence-interval estimation on the basis of equation (1) does not require actual implementation of the pre-averaging operation; one merely employs $n/m - 1$ degrees of freedom instead of $n-1$ degrees of freedom for the t distribution. This seems indeed justified, provided that n/m is large enough. The above relation, together with equations (27.7.1) and (27.7.2) of Cramér's *Mathematical Methods* of *Statistics* (Princeton University Press, 1951) shows that the expected values of $[(z-\bar{z})^2/(n/m-1)]^{1/2}$ and $[f-\bar{f})^2/(n-1)]^{1/2}$ differ by an error of the order of $(m/n)^{3/2}$, while their variances differ by a term of the order of $(m/n)^2$. The exact error can be obtained through a rather formidable calculation but depends on the (usually unknown) fourth-order central moment of f.

*Private communication

ACKNOWLEDGMENT

The writer is very grateful to the Information Sciences Directorate, Office of Aerospace Research, Air Force Office of Scientific Research and to the Office of Space Sciences, National Aeronautics and Space Administration for their support of this study under grants AF-AFOSR 89-64 and NASA NSG/03-002-024.

REFERENCES AND BIBLIOGRAPHY

The ASTRAC II system

1 G A KORN
New high-speed analog and analog-digital computing techniques: The ASTRAC *system*
Proceedings 3rd AsICA Conference Opatija Yugoslavia 1961
Presses Académiques Européennes Brussels 1962

2 ——
The impact of hybrid analog-digital techniques on the analog-computer art
Proceedings IRE May 1962

3 ——
ASTRAC *offers new computing methods*
Electronic Industries July 1962

4 ——
Parameter-perturbation generator for analog-computer error and optimization studies
Annales AsICA April 1963

5 ——
Performance of operational amplifiers with electronic mode switching
IEEETEC June 1963

6 ——
Analog/hybrid storage and pulse modulation
IEEETEC August 1963

7 ——
1962 progress and ASTRAC II *project*
ACL memo no 59 University of Arizona December 1962

8 R HAMPTON G A KORN B A MITCHELL
Hybrid analog-digital noise generator
IEEETEC August 1963

9 H R ECKES
Digital expansion system for ASTRAC I
Annales AsICA January 1964

10 —— G A KORN
Digital program control for iterative differential analyzers
SIMULATION February 1964

11 T A BRUBAKER
ASTRAC I *design, performance, and accuracy studies*
Annales AsICA April 1964

12 R L HAMPTON
A hybrid analog-digital pseudo-random noise generator
Proceedings SJCC 1964 and SIMULATION March 1965

13 H HANDLER R H MANGELS
A delta-sigma modulation system for time delay and analog function storage
Proceedings SJCC 1964

14 B A MITCHELL
A hybrid analog-digital parameter optimizer for ASTRAC II
Proceedings SJCC 1964 and SIMULATION June 1965

15 G A KORN
Fast analog-hybrid computation with digital control: the ASTRAC II *system*
Proceedings 4th AsICA Conference Brighton England 1964
Presses Académiques Européennes Brussels 1965

16 R L MAYBACH
Hybrid analog-digital measurement of sample averages and correlation functions
ACL memo no 85 University of Arizona 1965

17 ——
New techniques for measuring probability and probability density
ACL memo no 97 University of Arizona 1965

18 R WHIGHAM
A fast quarter-square multiplier
ACL memo no 88 University of Arizona 1964 and SIMULATION August 1965

19 ——
A fast and accurate analog comparator with digital output
ACL memo no 95 University of Arizona 1965

20 H R ECKES
A fast mode-control switch for ASTRAC II
ACL memo no 107 University of Arizona 1965

Special Monte Carlo techniques

21 N METROPOLIS S ULAM
The Monte Carlo method
Journal American Statistical Association 44 335 1949

22
Monte Carlo method
NBS Applied Mathematics Series vol 12 1951

23 H KAHN A W MARSHALL
Methods of reducing sample size in Monte Carlo computations
Journal Operations Research Society of America 263 1953

24 G W KING
Monte Carlo method for solving diffusion problems
Industrial Engineering Chemistry 43 2475 1951
Note also the papers by
G W KING L H THOMAS E C YOWELL
Proceedings Seminar on Scientific Computation December 1949
IBM Corporation New York 1951

25 G W BROWN
Monte Carlo methods in E F Breckenbach modern mathematics for the engineer
1st series McGraw-Hill New York 1956 (note bibliography)

26 J M HAMMERSLEY et al
A new Monte Carlo technique: Antithetic variates
Proceedings Cambridge Philosophical Society
449 and 476 1956

27 —— D C HANDSCOMB
Monte Carlo methods
Methuen London 1964

28 K CHUANG et al
A stochastic method for solving partial differential equations using an electronic analog computer
Project Michigan Report 2900-91-T Willow Run Laboratories
University of Michigan Ann Arbor 1960

29 D F DAWSON
Continuous measurement and display of time averages
ACL memo no 108 University of Arizona 1965

Optimization by random perturbations

30 S H BROOKS
A discussion of random methods for seeking maxima
Operations Research March-April 1958

31 —— M R MICKEY
Optimum estimation of gradient directions in steepest-descent experiments
Biometrics March 1961

32 R R FAVREAU R FRANKS
Random optimization by analog techniques
Proceedings 2nd AsICA Conference Strassbourg France 1958
Presses Académiques Européennes Brussels 1959

33 J K MUNSON A I RUBIN
Optimization by random search on the analog computer
IRETEC June 1959

34 D C KARNOPP
Search theory applied to parameter-scan optimization
PhD thesis Massachusetts Institute of Technology 1963

35 L A RASTRIGIN
External control by the random search method
Automatika i Telemekhanika September 1963

36 ——
The convergence of the random search method
Automatika i Telemekhanika November 1963

37 B A MITCHELL
Hybrid analog-digital parameter optimizer for ASTRAC II
MS thesis University of Arizona 1964
Proceedings SJCC 1964 and *SIMULATION* June 1965

Miscellaneous

38 G A KORN T M KORN
Mathematical handbook for scientists and engineers
McGraw-Hill New York 1961

39 —— ——
Electronic analog and hybrid computers
McGraw-Hill New York 1964

40 G A KORN
Statistical measurements with quantized data
SIMULATION April 1965

41 ——
Random-process simulation and measurements
McGraw-Hill New York (in print)

42 L LEVINE
Methods for solving engineering problems using analog computers
McGraw-Hill New York 1964

ASTRAC-I STUDY OF AN ORTHOGONAL-FUNCTION COMMUNICATIONS SYSTEM *

by Brian K. CONANT **

ABSTRACT

This paper describes the use of a fast iterative analog computer to study a sophisticated communications system. This system uses amplitude modulated orthogonal polynomial waveforms as carrier signals.

The Arizona Statistical Repetitive Analog Computer (ASTRAC-I) setup includes the polynomial waveform generator and a matched filter receiver. Accuracy and crosstalk are measured using a sample-hold-memory-pair module. The system is also tested with additive Gaussian noise with the aid of the ASTRAC-I hybrid analog-digital statistics computer (sample averaging unit and distribution analyzer).

Brian K. Conant was born in Yakima Washington, on September 30, 1937. He received his B.S. degree with honors in electrical engineering from Washington State University in 1960, and the M.S. degree from the University of Arizona in 1963.

From 1960 through 1961, and the summers of 1962 and 1963, he worked as an engineer in telemetry development for Sandia Corporation, Livermore, California. He now has a General Electric Fellowship and is working toward the PhD. degree at the University of Arizona.

Mr. Conant is a member of Tau Beta Pi, Phi Kappa Phi, IEEE, and Simulation Councils Inc.

INTRODUCTION

A proposed method of multiplex transmission employs repetitively generated Legendre polynomials as carrier waveforms for a set of signals [1].

For this study the polynomials are generated by digitally controlled analog computing elements in an iterative differential analyzer, the Arizona Statistical Repetitive Analog Computer (ASTRAC-I) [2]. Figure 1 indicates how system-performance measures (signal-to-noise ratios and detection probabilities) are measured by statistical sampling using a sample averaging unit [3] and an amplitude distribution analyzer [4].

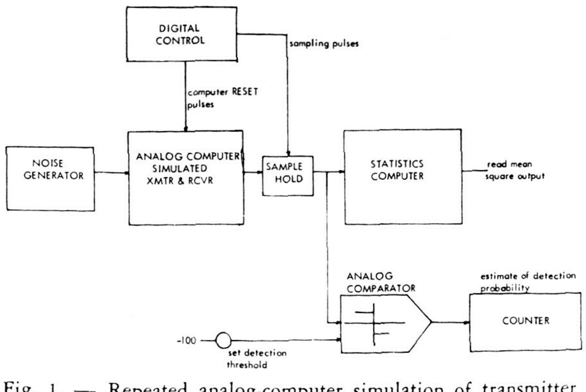


Fig. 1. — Repeated analog-computer simulation of transmitter and receiver operation with random-noise input permits measurement of mean-square signal and noise and of detection probabilities.

Since the theory for this communications system is well developed, the simulation serves as a check on the accuracy of ASTRAC-I and demonstrates the power of a digitally controlled iterative analog computer for studying a sophisticated communications system.

THE COMMUNICATION SYSTEM

The system to be simulated uses Legendre polynomials as carrier signals. The first five polynomials illustrated in figure 2a are

$$\begin{aligned} Q_0(x) &= 1 \\ Q_1(x) &= x \\ Q_2(x) &= \tfrac{1}{2}(3x^2 - 1) \\ Q_3(x) &= \tfrac{1}{2}(5x^3 - 3x) \\ Q_4(x) &= \tfrac{1}{8}(35x^4 - 30x^2 + 3) \end{aligned} \qquad (1)$$

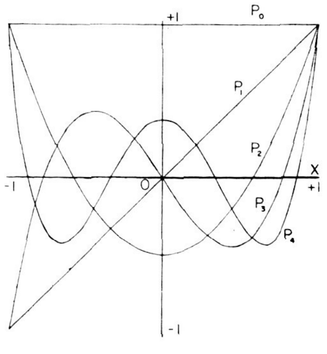


Fig. 2 a. — Legendre Polynomials.

* Manuscript received June 3, 1964.

** Department of Electrical Engineering, University of Arizona, Tucson, Arizona.

These polynomials satisfy the orthogonality condition

$$\int_{-1}^{1} Q_m(x)\, Q_n(x)\, dx \quad \begin{matrix} = 0 & (m \neq n) \\ \neq 0 & (m = n) \end{matrix} \qquad (2)$$

To generate analogous polynomials orthogonal over the interval $0 \leqslant t \leqslant T$ on an iterative analog computer, a shift to the right and a change of variable is necessary

$$x = (2/T)\,(t - T/2) \qquad (3)$$

The five polynomials then become

$$\begin{aligned} P_0(t) &= 1 \\ P_1(t) &= 2\,(t/T) - 1 \\ P_2(t) &= 6\,(t^2/T^2) - 6\,(t/T) + 1 \qquad (4) \\ P_3(t) &= 20\,(t^3/T^3) - 30\,(t^2/T^2) + 12\,(t/T) - 1 \\ P_4(t) &= 70\,(t^4/T^4) - 140\,(t^3/T^3) + 90\,(t^2/T^2) \\ &\quad - 20\,(t/T) + 1 \end{aligned}$$

Figure 2 illustrates how these waveforms may be used in a multiplex system. The $P_k(t)$ are generated by means of a polynomial waveform generator. The coefficients a_k represent the modulating information for each channel.

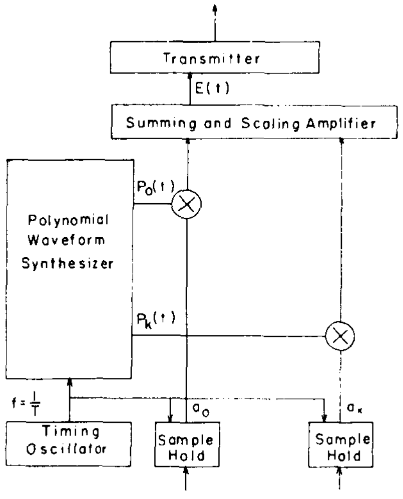


Fig. 2 b. — Transmitter.

The modulation process consists of four-quadrant multiplication. The a_k must remain constant throughout the interval T to preserve orthogonality of the waveforms. Therefore the modulating waveform must be sampled at a rate such that a_k remains essentially constant in the interval T, or the modulating voltage may be sampled and held constant in the interval by a sample-hold circuit.

The output of the waveform generator has the form

$$E(t) = \sum_{k=0}^{n} a_k\, P_k(t) \qquad 0 \leqslant t \leqslant T \qquad (5)$$

GENERATOR SIMULATION

This system especially lends itself well to study on a repetitive analog computer since, in fact, the actual system would be implemented using similar analog computer techniques.

The computer setup for generating the Legendre polynomials is illustrated in figure 3. This setup makes the four-quadrant multipliers unnecessary.

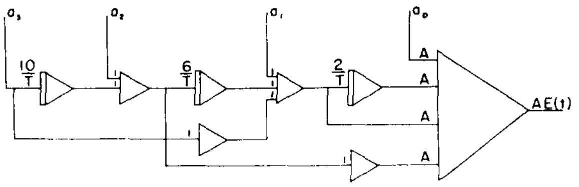


Fig. 3. — Computer Setup-Waveform Generator.

ASTRAC-I is operated at 10 runs per second. The period T of the waveforms was made equal to 80 milliseconds, shorter than the COMPUTE period (90 milliseconds) so that an accurate sampling time at T could be set into the computer using the digitally adjusted sampling time control. Gain « A » is adjusted so that the output voltage $E(t)$ is a convenient amplitude.

THE RECEIVER

A basic receiver is illustrated in figure 4. The output modulation coefficients a_k for each channel are determined by correlating the composite signal $AE(t)$ with the corresponding carrier waveform for that channel, i.e.

$$a_k = \frac{1}{T} \int_0^T E(t)\, P_k(t)\, dt \qquad (6)$$

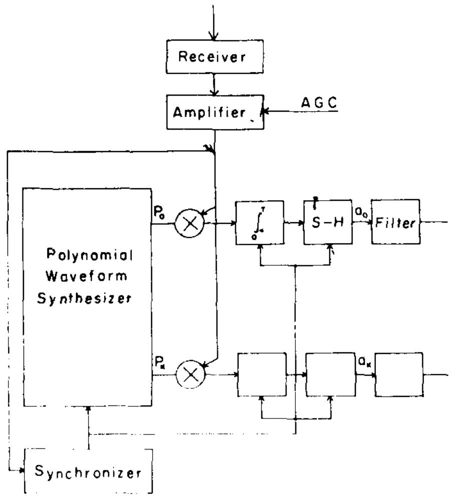


Fig. 4. — Receiver.

This equation rollows immediately from the orthogonality condition (2) together with the definition of the composite signal (5).

At the start of the RESET period, at time T, the correlator output voltages are sampled, and the integrators are reset in preparation for the next transmission interval. The sequence of sampled voltages taken from the sample-hold circuit for each channel represents the original modulation in sampled form.

The polynomial waveforms may also be separated at the receiver by matched filters, which does away with the need for explicit multiplication. A derivation of the equations for the matched filters for the first five polynomial waveforms is given in Appendix I.

RECEIVER SIMULATION

The computer setup for the matched filters is illustrated in figure 5. The integrators are reset at a 10 cps. rate in synchronism with the waveform generator-transmitter. Gain « B » is adjusted to give a convenient output voltage amplitude. S_0, S_1, S_2 and S_3 are the outputs of the filters for the inputs P_0, P_1, P_2 and P_3 respectively.

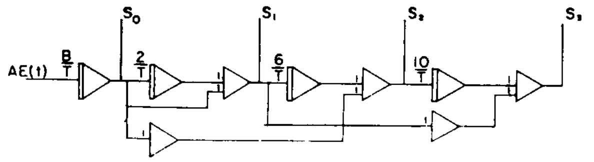


Fig. 5. — Computer Setup-Matched Filter.

ACCURACY

The output of the polynomial waveform generator was observed on an oscilloscope and appeared as shown in figures 6, 7, 8, 9, and 10. $a_4 P_4$ was generated merely to demonstrate the inaccuracy of higher order waveforms. The output summing amplifier takes the difference of large voltages and because of this, error and noise are magnified. The output magnitude or higher-order polynomials becomes very small because of the scaling that is necessary. To improve this scaling situation each polynomial would have to be generated separately at much greater equipment expense.

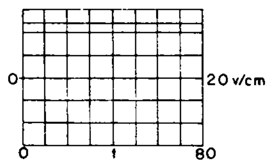


Fig. 6. — P_0.

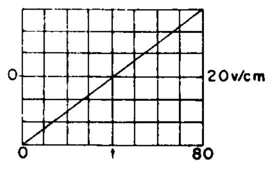


Fig. 7. — P_1.

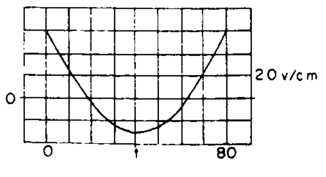


Fig. 8. — P_2.

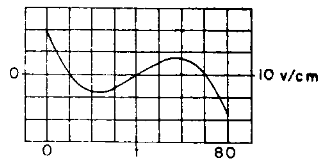


Fig. 9. — P_3.

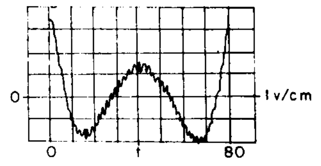


Fig. 10. — P_4.

Figure 11 is an example of the ouput sum of P_0, P_1, P_2, P_3 and P_4.

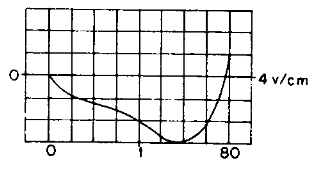


Fig. 11.

$\sum_{k=1}^{4} a_k P_k(t)$

Accuracy measurements were made on the four polynomial waveforms. This was done by sampling the starting voltage of each waveform $a_k P_k(0)$ and the voltage at time T, $a_k P_k(T)$ and comparing them with the input a. Each waveform was sampled using the ASTRAC-I sample-hold-memory-pair module and the sampled value read on a digital voltmeter. The sampling times $t = 0$ and $t = T$ are set by pushbuttons on the digital control unit of the computer. The results are sohwn in table I.

TABLE I

Generator Accuracy

Gain « A » = 10.0

	a_k	$a_k P_k(0)$	$a_k P_k(T)$
a_0	4.0	40.0	40.0
a_1	4.0	40.5	39.9
a_2	4.0	40.2	38.0
a_3	2.0	16.9	18.0

The output of the matched filters is illustrated in figures 13 and 14. The input polynomial sum is shown in figure 12 with a_3 equal to zero and the corresponding filter output in figure 13. It can be seen that $S_3(T)$ is approximately equal to zero in agreement with the theory of matched filters. Figure 14 shows the output $S_3(t)$ with a_3 unequal to zero.

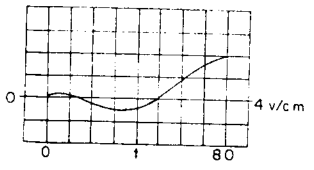


Fig. 12. — $P_0 + P_1 + P_2 + P_3$.

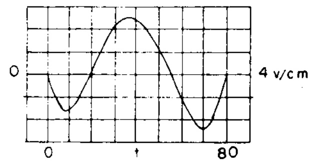


Fig. 13. — $S_3(t)$, $a_3 = 0$.

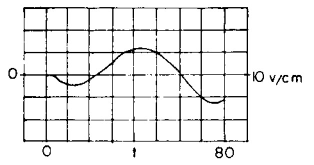


Fig. 14. — $S_3(t)$, $a_3 \neq 0$.

Measurements were made at the output of each matched filter and compared to its theoretical output calculated using the input a_k shown in table I. The results are shown in table II.

TABLE II

Filter Accuracy

	S_k(T) meas.	S_k(T) theo.	Gain « B »
S_0(T)	63.17	64.0	1.60
S_1(T)	21.2	21.4	1.60
S_2(T)	61.7	64.0	8.00
S_3(T)	43.0	45.7	16.00

The effects of computer inaccuray show up in the higher-order polynomials, which require computer set-ups with more integrators connected in series. Any error will be successively integrated. Error in the integrators and summing amplifiers is due mainly to the tolerances of the summing resistors and feedback resistors and capacitors which in this case were all one percent. Noise, particularly chopper noise, also contributes to the error. For the entire simulated system, from input to output, the greatest error

$$\frac{S_k(T)\text{ theo.} - S_k(T)\text{ meas.}}{S_k(T)\text{ theo.}} \times 100\%$$

is about 6 percent for the third order polynomial. These measurements were taken one channel at a time so that crosstalk did not enter into the measurements.

CROSSTALK

Crosstalk is due to inaccurate polynomial waveforms and to inaccuracies in the matched filter.

A slowly varying modulating signal was sampled by a sample-hold unit and fed intno one channel, the other channels having no input. The output of the other channels was observed. The ratio of the magnitude of the undesirable output signal to the input signal was used as a measure of crosstalk. The results are shown in table III.

TABLE III

Crosstalk

		S_0	S_1	S_2	S_3
a_0	$\frac{S_k}{a_k} \times 100\%$		0.8%	0.8%	0.88%
a_1		0.5%		0.5%	0.3%
a_2		0.1%	0.3%		0.7%
a_3		1.1%	0.4%	0.27%	

The crosstalk data are normalized so that Gain « A » and Gain « B » both equal unity. In each case the crosstalk is less than 1.1 percent.

STATISTICAL MEASUREMENTS

Noise response.

The noise source used for making statistical measurements was the ASTRAC-I Noise Generator, which filters a random telegraph wave generated by a radioactive source to give Gaussian noise [5].

The mean square output at time T of the matched filters $h_0(\lambda)$ and $h_1(\lambda)$ was measured (estimated) with the ASTRAC-I hybrid analog-digital statistics computer [3]. The results compared with the theoretical results (see Appendix II) are shown in table IV.

TABLE IV

Noise Output of Matched Filters h_0 and h_1

Input Noise = 313 v^2 RC = 1000 Gain « B » = 9.60

Filter	*Mean Square Output at Time T* measured	theoretical
h_0	704 v^2	712 v^2
h_1	239 v^2	231 v^2

The experimental results are seen to agree well with the calculated results.

Detection.

Detection probability measurements were made at the output of $h_0(\lambda)$ and $h_1(\lambda)$. Detection probability was estimated directly with the aid of the ASTRAC-I Amplitude Distribution Analyzer [4]. At the same time, the noise power output of the filter was measured with the statistics computer. Knowing the output noise power and using a table of the normal distribution, one can calculate the theoretical detection probability for a given critical region (see Appendix III). This was compared to the measured detection probability and the results shown in table V agree very well.

TABLE V

Detection Probability

Gain « B » = 8.0

	Input Noise	*Output Noise*	*Det. Prob.* meas.	calc.	$S_k(T)$	y_c
h_0	$300\ v^2$	$477\ v^2$	0.889	0.888	$31.7\ v$	$5\ v$
h_1	$316\ v^2$	$153\ v^2$	0.797	0.808	$21\ v$	$10\ v$

CONCLUSION

The digitally controlled repetitive analog computer has proven particularly useful in studying this communications system. Such a machine makes it easy to study the effects of noise on signal-to-noise ratios and detection probability by actual statistical measurements on a large sample of computer runs.

APPENDIX I

For a given signal $P(t)$ a matched filter is a network whose impulse response is [6]

$$h(\lambda) = P(T - t) \tag{6}$$

For the Legendre polynomial waveforms, the impulse responses of the matched filters are

$$\begin{aligned} h_0(\lambda) &= 1 \\ h_1(\lambda) &= -2(\lambda^2/T^2) + (1/T) \\ h_2(\lambda) &= 6(\lambda^3/T^3) - 6(\lambda^2/T^2) + (1/T) \\ h_3(\lambda) &= -20(\lambda^4/T^4) + 30(\lambda^3/T^3) \\ &\quad - 12(\lambda^2/T^2) + (1/T) \\ h_4(\lambda) &= 70(\lambda^5/T^5) - 140(\lambda^4/T^4) + 90(\lambda^3/T^3) \\ &\quad - 20(\lambda^2/T^2) + (1/T) \end{aligned} \tag{7}$$

Note that each impulse response has been multiplied by $1/T$, so that the expression for the filter output at time T will be independent of T.

Taking the Laplace transform of the impulse response, one finds the transfer function of the filters

$$\begin{aligned} H_0(s) &= 1/Ts \\ H_1(s) &= (-2/T^2 s^2) + (1/Ts) \\ H_2(s) &= (12/T^3 s^3) - (6/T^2 s^2) + (1/Ts) \\ H_3(s) &= (-120/T^4 s^4) + (60/T^3 s^3) \\ &\quad - (12/T^2 s^2) + (1/Ts) \\ H_4(s) &= (1680/T^5 s^5) - (840/T^4 s^4) \\ &\quad + (180/T^3 s^3) - (20/T^2 s^2) + (1/Ts) \end{aligned} \tag{8}$$

Each filter output is given by

$$\begin{aligned} S_k(t) &= \mathcal{L}^{-1}[a_k P_k(s) H_k(s)] \\ &= \frac{1}{2\pi j}\int_{-j\infty}^{j\infty} a_k P_k(s) H_k(s) e^{st}\, ds \end{aligned} \tag{9}$$

so that for $t = T$

$$S_k(T) = \frac{(-1)^k a_k}{2k+1} \tag{10}$$

APPENDIX II

If noise is added into the system at the input of the matched filters to simulate interference, the noise output of the filters may be expressed in the following manner [7]. We define the « autocorrelation function of the filter » as

$$R_{h_{kT}h_{kT}}(\tau, T) = \int_0^{T-\lambda} h_k(\lambda, T)\, h_k(\lambda + \tau, T)\, d\lambda \tag{11}$$

For example, for $h_0(\lambda, T)$ and $h_1(\lambda, T)$ we have

$$R_{h_{0T}} R_{h_{0T}}(\tau, T) = \frac{1}{T}\left(1 - \frac{|\tau|}{T}\right) \tag{12}$$

$$R_{h_{1T}} R_{h_{1T}}(\tau, T) = \frac{T^3 + 2|\tau^3| - 3|\tau| T^2}{3T^4} \tag{13}$$

The noise power at the filter output for $t = T$ with no signal present is

$$R_{SS}(T) = \int_0^T R_{h_{kT}h_{kT}}(\tau, T)\, R_{nn}(\tau)\, d\tau \tag{14}$$

where $R_{nn}(\tau)$ is the autocorrelation function of the input noise.

For white Gaussian noise passed through a simple low-pass RC filter as is done in the noise generator, the autocorrelation function of the generator output is

$$R_{nn}(\tau) = R_{nn}(0)\, e^{-|\tau|/RC} \tag{15}$$

where $R_{nn}(0)$ is the mean square value of the noise, or noise power.

Therefore the noise power output of the two example filters is

$$R_{s_0 s_0}(T) = 2R_{nn}(0)\frac{R^2C^2}{T^2}\left[e^{-T/RC} + \frac{T}{RC} + 1\right] \tag{16}$$

$$\begin{aligned} R_{s_1 s_1}(T) &= 2R_{nn}(0)\frac{R^2C^2}{T^2}\left[\frac{4R^2C^2}{T^2} + \frac{T}{3RC} - 1\right] \\ &\quad - 2R_{nn}(0)\, e^{-T/RC}\left[1 + \frac{2RC}{T}\right]^2 \end{aligned} \tag{17}$$

APPENDIX III

Detection.

Detection theory indicates that a test statistic useful for detection of a known signal in white Gaussian noise is generated by passing the signal corrupted by the noise through a matched filter [10].

Figure 15 illustrates the simple detection problem for the detection of a d-c signal which has been corrupted be additive independent Gausian noise. $S_k(T)$ is the expected value of the output of the matched filter when a signal is present. h_1 is the hypothesis that a signal is present; and h_0 is the hypothesis that there is no signal. $\phi\,[N_k(T)\,|h_1]$ is the probability density *function of the noise and the signal and* $\phi\,[N_k(T)\,|h_0]$ is the probability density function of the noise with no signal. The input noise has zero mean. The noise creates a region of uncertainty as illustrated.

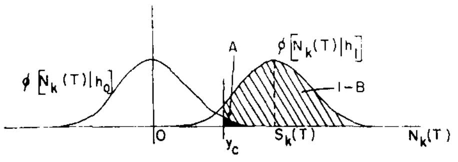


Fig. 15. — Detection of a Known Signal in Gaussian Noise.

For the purposes of this study a critical level y_c is determined by a Neyman-Pearson test, that is, the critical level is chosen so that the detection probability indicated by the area 1 — B, is maximized for a given false alarm probability, indicated by area A. For the purpose of the simulation, the value of y_c was chosen arbitrarily.

Acknowledgements.

The author wishes to express his appreciation to Professor G.A. Korn for his encouragement and advice.

The author performed much of this work as a graduate assistant in the Department of Electrical Engineering, University of Arizona. This financial support is gratefully acknowledged.

REFERENCES

[1] A.H. BALLARD, « A New Multiplex Technique for Telemetry », Proc. National Telemetering Conference, Vol. I, Session 6-2, 1962.

[2] T.A. BRUBAKER, « The Design, Development, and Applications of the ASTRAC Computer », Doctoral Dissertation, Department of Electrical Engineering, University of Arizona, Tucson, Arizona, 1963.

[3] B.K. CONANT, « A Hybrid Analog-Digital Statistics Computer », Analog-Hybrid Computer Laboratory Memorandum No. 45, Department of Electrical Engineering, University of Arizona, Tucson, Arizona, December, 1962.

[4] T.A. BRUBAKER and G.A. KORN, « Accurate Amplitude Distribution Analyzer Combines Analog and Digital Logic », Review of Scientific Instruments, Vol. 32, No. 3, pp. 317-322 ; March, 1961.

[5] J.R. MANELIS, « Generating Random Noise With Radioactive Sources », Electronics, pp. 66-69 ; September 8, 1961.

[6] C.W. HELSTROM, « Statistical Theory of Signal Detection », Pergamon Press, pp. 84-87 ; 1960.

[7] D. MIDDLETON, « An Introduction To Statistical Communication Theory », McGraw-Hill Book Co., Inc., New York, N.Y., pp. 161-164 ; 1960.

[8] C.W. HELSTROM, « Statistical Theory of Signal Detection », Pergamon Press, pp. 91-95 ; 1960.

A fast 10v diode quarter square multiplier

ROBERT H. WHIGHAM
University of Arizona
Tucson, Arizona

ROBERT HERSCHEL WHIGHAM was born in 1939 at McAllen, Texas. He attended Trinity and Baylor Universities as a pre-medical student and then, discovering that his interests lay elsewhere, changed majors and earned his BA from Baylor in 1960 with a major in mathematics. He next entered Texas Technological College and acquired a BS in electrical engineering. Feeling that his education was still rather thin, he thereupon entered graduate school at the University of Arizona in Tucson where he is now working as a half-time graduate assistant in the University's Analog-Hybrid Computer Laboratory. He has contributed to the University's Hybrid-Code Computer and has taught in the undergraduate computer laboratory. He has specialized in hybrid computer circuits involving exceptionally fast diode function generators, including the ASTRAC II quarter square multiplier, sine-cosine generator, diode function generator, and the new 100-nanosecond comparator for ASTRAC II.

Mr. Whigham recently received his MS degree from the University of Arizona and is well into a doctoral program which is scheduled for completion in 1967. He is a member of Tau Beta Pi and Eta Kappa Nu.

Outside his vocational field, Mr. Whigham's main interest is spelunking, primarily in Arizona, New Mexico, and Texas; he is a member of the Southwest Speleological Association.

ABSTRACT

This report describes the design and performance of a fast wideband quarter-square diode multiplier designed to work with the $\pm$10-v low-impedance computing circuits of a high-speed iterative differential analyzer (ASTRAC II), or in other hybrid analog-digital computer systems. To reduce cost, the new multiplier employs absolute-value squaring circuits and does not require committed computer amplifiers. Improved combination shunt-series switching circuits and low resistance values ensure wide bandwidth ($\pm$0.5% of half-scale dynamic error at 10 Kc, <1 degree phase shift below 70 Kc). Temperature-compensating diodes in the bias networks reduce thermal drift below 0.7 mv/°C, so that the multiplier static accuracy of $\pm$0.20% of half-scale is maintained from 15 to 40° C. A number of useful design hints are listed.

INTRODUCTION

This report describes the design of a new analog multiplier developed for a high-speed iterative differential analyzer, ASTRAC II (Arizona Statistical Repetitive Analog Computer II). ASTRAC II is a fast iterative analog computer with sophisticated digital controls and special components for statistical and optimization problems.[1] The large bandwidth of its analog computing components permits up to 1000 repetitions per second as well as real-time simulation. A large sample of solutions can be obtained quickly enough so that simulated system statistics taken over 1000 to 10,000 computer runs can be optimized manually or automatically in a very short time.

Because of the low computing impedances used with ASTRAC II (1K and 5K summing resistors), its speed is basically determined by the bandwidth of the operational amplifiers, rather than by the effects of external capacitances. ASTRAC II's operational amplifiers (similar to Burr-Brown type 1607A) have their unity-gain (0 db) frequency at a minimum of 20 Mc. Their small physical size permits them to be plugged directly into the rear of the computer patchbay without any signal wiring whatsoever, and their high current capability (30 ma at $\pm$10 v up to 1 Mc) enables them to drive the remaining stray capacitance to relatively high frequencies. Other analog computing components — such as quarter-square multipliers — also plug directly into the rear of the patchbay. This location is practical because frequent adjustments are not necessary.

The standard plug-in module used for all of ASTRAC II's analog computing elements is shown in figure 1. Figure 2 shows one of the multiplier's two absolute-value squaring circuits; figure 3 illustrates the principle of quarter-square multiplication with two absolute-value squaring circuits. This figure also shows special squaring inputs for squaring with reduced current requirement. Figure 4 shows the equivalent loading circuit used to calculate the required input current.

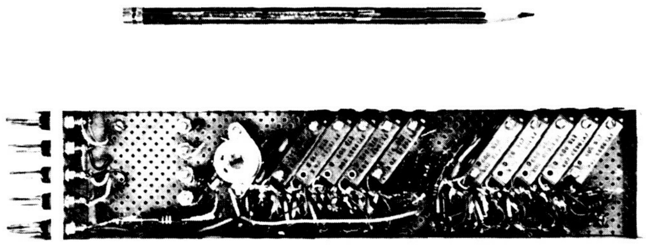


Figure 1 – ASTRAC II multiplier plug-in module

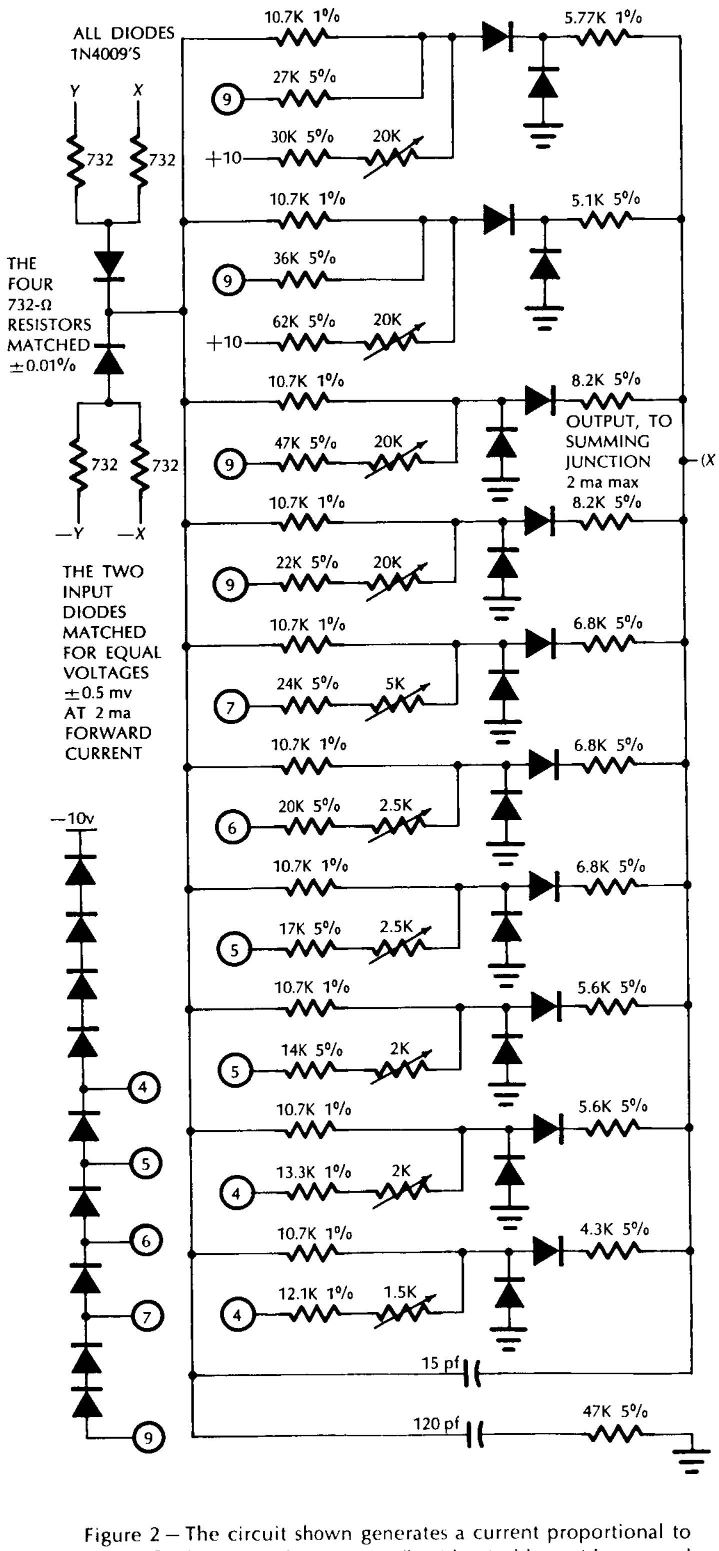


Figure 2 – The circuit shown generates a current proportional to $(X + Y)^2$. The circuit for $-(X - Y)^2$ is identical but with reversed diodes and bias. Temperature-stabilizing diodes in the bias line reduced thermal-drift error from ±7 mv/°C to ±0.7 mv/°C for squaring. A 15-pf capacitor and series resistor and capacitor to ground for phase-shift compensation extended the 0.5% of half-scale error frequency by about one octave.

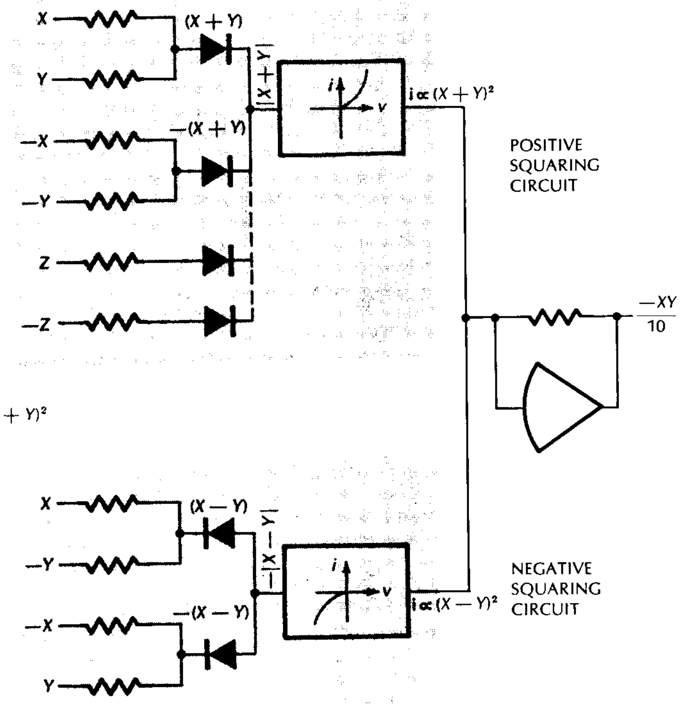


Figure 3 – Two absolute-value squaring circuits implement quarter-square multiplication. Special "Z" inputs permit squaring with reduced input current.

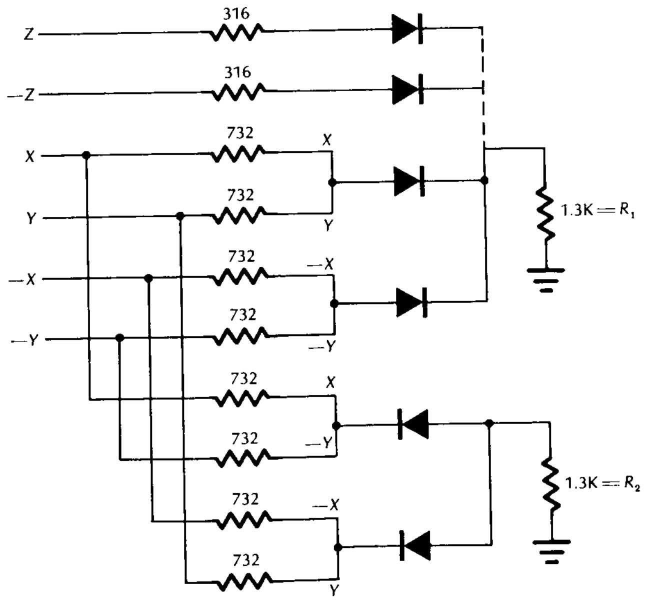


Figure 4 – An equivalent loading circuit enables one to calculate the current requirements of the multiplier in any application.

ANALYSIS OF CIRCUIT PERFORMANCE

Basic principles

Quarter-square multipliers implement the relation

$$XY/V = \frac{1}{4V}[(X+Y)^2 - (X-Y)^2]. \qquad (1)$$

Maximum errors from each of the two squaring circuits required can add. Consequently, $\pm 0.10\%$ of half-scale multipliers require $\pm 0.05\%$ of half-scale squaring devices.

Absolute-value squaring

Diode squaring circuits can use separate diode networks to form the positive and negative halves of the parabolic transfer characteristics $Y^2 = aX^2$, or a single network preceded by an absolute-value circuit may be employed as in figure 3. The latter alternative was chosen because it is more compact, less expensive, and easier to adjust. Absolute-value squarers are, on the other hand, more difficult to design because of the complications inherent in cascaded diode limiters.

Error sources in the absolute-value circuit are the forward bias required to turn the diodes on, their nonlinearity after they are on, their finite recovery or turnoff time, their capacitance, and mismatch in their characteristics. These problems were overcome by slightly forward biasing the two diodes for zero input, by designing the square-law network for an accurate overall transfer characteristic which takes into consideration nonlinearities in the absolute-value circuit as well as nonlinear loading of the summing resistors (as diodes switch off and on), by using fast-recovery low-capacitance diodes, and by matching the two absolute-value circuit diodes for equal forward-voltage drops (± 0.5 mv at 2 ma of forward current) to make the output of the absolute-value circuit for positive and negative ($X \pm Y$) very nearly the same.

The summing resistors were also matched to $\pm 0.01\%$. The primary source of static error, then, is the piecewise-linear approximation to a squaring curve.

The diode squaring circuits

Figure 5 illustrates the operation of the basic diode-limiter circuit. The transfer characteristic shows zero output current for input voltages smaller than the "breakpoint" voltage of V_{bp}. For input voltages greater than V_{bp}, the channel switches "on" and produces positive output current; this is a "positive" channel. The rate of increase (slope or transfer admittance) can be specified independently of the breakpoint. When several positive channels are driven by the same input and their output currents are summed at the summing junction of an operational amplifier, their composite transfer characteristic can be made to approximate the first quadrant of a square-law characteristic.

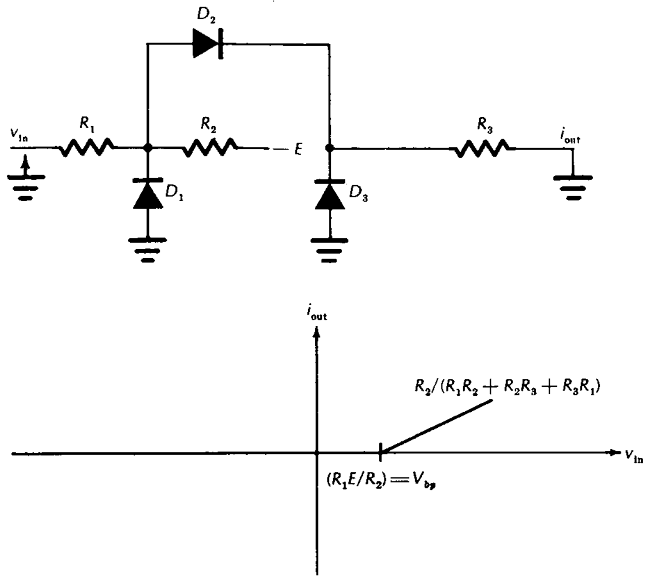


Figure 5 – A "positive" diode limiter channel and its transfer characteristic. A combination of shunt and series limiting can greatly reduce effects of diode capacitance. D_1 ("catching" diode) and D_3 perform shunt limiting, and D_2 performs series limiting. The improvement obtained with all three diodes did not justify the added expense, but a single diode was markedly inferior to a pair of diodes. R_1 could be chosen for the proper slope, and then R_2 chosen for the proper breakpoint, so that R_3 is not required; but R_3 permits the designer to adjust the slope without affecting the breakpoint.

Figure 6 illustrates the error due to polygonal approximation of a square-law characteristic. It can be shown that the maximum absolute error will be smallest if the breakpoints are equally spaced, the slope increases by an equal amount at each breakpoint except the origin, and the slope of the first segment at

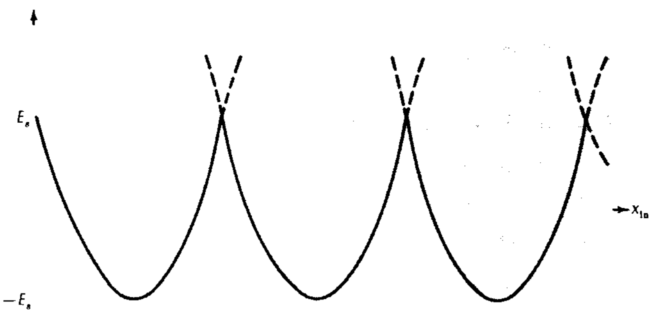


Figure 6 – The error in an exactly piecewise-linear approximation to a square-law characteristic

the origin equals one-half the increment in slope at subsequent breakpoints. The maximum absolute error is, then

$$E_s = (12.5/n^2)\ \%\ \text{of half-scale} \quad (2)$$

where n is the number of approximating segments per quadrant. The multiplication error will be twice this or

$$E_{\text{mult}} = (25/n^2)\ \%\ \text{of half-scale} \quad (3)$$

In low-impedance ±10-v circuits the variation in diode conductance is more significant than in high-impedance ±100-v circuits. Although this effect makes analytical design more difficult, it also permits "diode rounding" to approximately halve the measured maximum absolute error given by equation (3).

Increasing the number of the segments per quadrant in the piecewise-linear approximation to a square-law characteristic will not increase the accuracy past a certain point determined by the component tolerances. Trimming potentiometers were used to vary the bias resistors by about 10%, which permitted ±0.05% of half-scale squaring accuracy (at 25°C), using 1% and 5% resistors with unmatched diodes in the squaring network. Breakpoint adjustment is superior to slope adjustment, for breakpoint adjustments permit diode variations to be "adjusted out." Variations in the slopes, on the other hand, are caused by resistance variation, and the tolerance of the resistors can readily be made as small as necessary. It was found that with adjustable breakpoints the tolerance of the slopes could be rather large. The combination of 1% and 5% resistors used (stock values) gave a net slope tolerance of about 3%. On several of the fourteen squaring circuits which have been built, one or two of the resistors in series with trim pots had to be changed to a higher or lower value to change the range of adjustment slightly. A smaller tolerance on the "slope-adjusting" resistor R_3 would avoid this difficulty; but the change required was obvious and easily made when the circuit was adjusted. The slope-adjusting resistor, R_3, is shown in figures 5 and 7.

Use of catching diodes

As shown in figure 7, with $R_1 > R_3$, as in the multiplier, the catching diode D_1 with D_2 is superior to D_3 and D_2 in improving the frequency response. Besides improving the frequency response, the catching diodes D_1 also reduce the mutual interaction of the breakpoint adjustments, and this results in a simpler adjustment procedure. In addition, the catching diodes present a more constant load to the absolute-value circuits as limiter diodes switch off and on.

It is interesting to note that added shunt diodes did not seem to improve the operation of the absolute-value circuits.

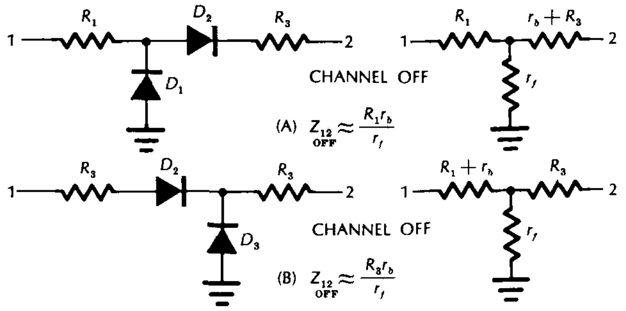


Figure 7 – The effectiveness of a catching diode D_1 with a series diode D_2 as compared to a shunt diode D_3 with D_2 in reducing the effects of diode capacitance depends on the relative size of the resistors R_1 and R_3. The back impedance (or impedance of a reverse-biased diode) is symbolized by r_b, while the forward impedance is symbolized by r_f. The equivalent circuits for the channel-off state are shown in (a) for D_1 and D_2 and in (b) for D_3 and D_2. The transfer impedance of the two "T" networks may be written immediately by inspection using the "Y" to "Δ" transformation and taking the 1-2 side. In (a), $Z_{12} = [R_1 r_f + R_1(r_b + R_3) + r_f(r_b + R_3)]/r_f \approx R_1 r_b/r_f$ when $r_b >> \max(R_1, R_3)$ and $r_f << \min(R_1, R_3)$. Similarly, in (b), $Z_{12} \approx R_3 r_b/r_f$. Thus, the transfer impedances are approximately equal for $R_1 = R_3$; but for $R_1 > R_3$, as in the multiplier, the catching diode in (a) gives higher transfer impedance. In each channel of the multiplier $R_1 \approx 2R_3$, and it was found experimentally that approximately one octave higher frequency for a given error was possible in the multiplier with catching diodes than with the other shunt diodes. This shows that r_b acts like a capacitance.

Improvement of accuracy near zero

Figure 8 shows that two diodes in series (one in the absolute-value circuit and one in the first limiter channel) turning on together at the origin were required to improve the accuracy by the same amount as one diode turning on at other breakpoints.

Slight forward biasing of the absolute-value circuit diodes for zero input was required for high accuracy near zero. Optimum forward bias was about 0.100 v. In addition, the first two limiter channels were slightly forward biased for zero input. This resulted in canceling ±3 mv d-c offsets from the two squaring circuits.

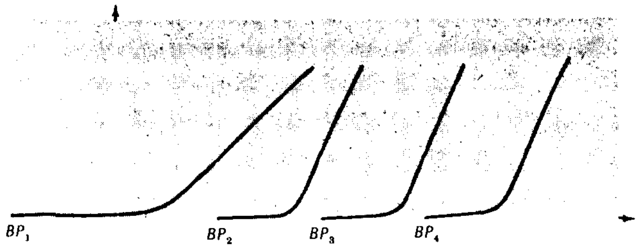


Figure 8 – Twice the curvature (caused by "diode rounding" as the diodes turn on) at the origin, where the increment in slope is one-half the increment in slope at later breakpoints, improves the accuracy by the same amount as the diode rounding at the later breakpoints. This is a bonus provided by the cascading of a diode in the absolute-value circuit and one in the first limiter channel, which turn on together.

Choice of diode type

Selection of diode type was based on observation of the rectification of a 300 Kc sine wave, and on a survey of diode specifications versus cost. To multiply accurately at 100 Kc the squaring circuits would have to generate accurate 200 Kc sine waves. Inexpensive low-conductance diodes and expensive high-conductance diodes were found to have similar characteristics near the origin (forward currents of less than 0.2 ma) so that inexpensive low-conductance diodes perform nearly as well as high-conductance ones. Substitution of different diode types, including germanium diodes, for the catching diode did not change the circuit performance.

The type of diode chosen was the 1N4009, which is an inexpensive high-quality silicon junction diode intended for the Skybolt program. *DATA Semiconductor and SCR Tabulation* gives its recovery time as 4 nsec and its capacitance as 4 pf. The limiting factor in its performance is its capacitance. Diodes with lower capacitance are available, but smaller capacitance is obtained by a smaller junction crossection, which results in greater thermal drift. The 1N4009's are particularly resistant to heat damage.* 1N4009's are approximately 33 cents each in lots of 100.

Temperature-drift compensation

The forward voltage required to turn a junction diode on decreases with increasing temperature. This results in severe drift in the composite characteristic. In the uncompensated circuit, for a temperature increase of 30°C, each limiter channel turned on 20 mv too soon; after the tenth channel had turned on, the total drift error was 200 mv. Temperature stabilization was achieved by placing diodes in the bias lines. When the temperature increases, these diodes produce less drop in the bias line, so the bias increases and compensates for the tendency of the series diodes to turn on sooner. The temperature compensation was designed empirically for one squaring circuit and found to work equally well on a second squaring network without special matching. Since the temperature compensation required diodes in the reverse bias lines, and since the first two diode-limiter channels were slightly forward biased to obtain the required accuracy near zero, offsetting positive and negative bias voltages were applied to these two channels to produce both temperature stabilization and net forward bias.

*The writer held a 25-w soldering iron against the glass at one end of the diode while observing the voltage drop at 2 ma forward current. After several minutes the voltage drop reached a new steady-state value. Both the anode and the cathode were so treated in an attempt to alter the room-temperature characteristic slightly. The diodes tested were not damaged, and their room-temperature characteristic was not altered appreciably (±0.5 mv).

Frequency-response equalization

A 15 pf lead capacitor from the output of the absolute-value circuit to the network output (to a summing junction) reduced the multiplier phase error, but caused amplitude error with increasing frequency. A roll-off network consisting of another capacitor and a resistor in series connected from the absolute-value circuit output to ground corrected this amplitude error to about 100 Kc. Capacitors across resistors in the individual diode limiters were tried, but were not as effective. Phase shift measured with different amplifiers was not exactly the same, so a compromise phase compensation was used.

An alternate and possibly better method of compensation would be to split the 5K feedback resistor into two 2.5K resistors with a 100 pf shunt capacitor to ground between them as shown in figure 9.

ADJUSTMENT AND TESTS

Breakpoint adjustment

Four methods for accurate breakpoint adjustment were investigated. Iterative d-c adjustment was used for adjusting the first squaring circuit, but it was too laborious for repeated use. A matching method was used to adjust all the other 14 squaring circuits required for seven multipliers. In this method, the circuit being adjusted is matched to another accurate square-law characteristic by subtracting the outputs of both circuits. The two remaining methods use a "slow" analog computer and a repetitive analog computer, respectively. A ramp $Y = kt$ is generated to drive the squaring network and a square $Y^2 = (kt)^2/10$ is generated and compared with the output of the squaring network to produce an error display. It was found difficult to obtain the desired accuracy by this method.[3]

As a fifth method, an attempt was made to adjust an entire multiplier so that it multiplied $10 \sin \omega t$ accurately by both 10 and -10v. It was found that this usually results in a badly misadjusted multiplier, which multiplies accurately only by 10 and -10.

Adjustments which are sometimes included in analog multipliers (a d-c adjustment, and *X* adjustment,

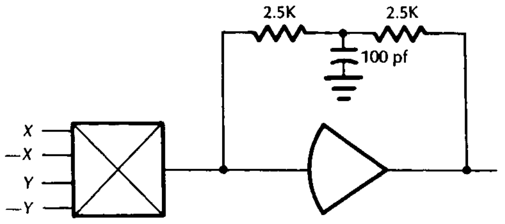


Figure 9 — The 100-pf capacitor indicated is optimum for the circuit shown for extending the 0.5% error frequency of an ASTRAC II inverter from approximately 30 Kc to 300 Kc.

a Y adjustment, and a gain adjustment) were not found necessary, although a gain adjustment was included which trims the 5K feedback resistor of the output amplifier.

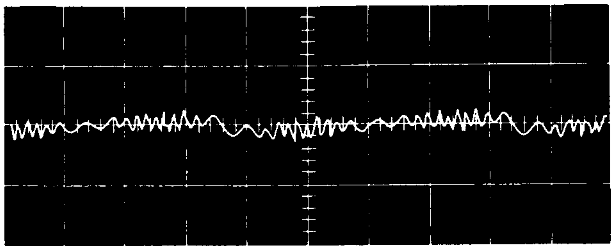
In the iterative d-c adjustment method, the output of a squaring network is conducted to the summing junction of an operational amplifier which has an accurate 5K feedback resistor. The input voltage to this squaring network is then set equal to $X = Y =$ 1.000, 2.000, 3.000, . . . , 10.000 v consecutively, and the network's breakpoints are adjusted consecutively for —0.100, —0.400, —0.900, . . . , —10.000 v out of the amplifier. One breakpoint is adjusted for each input. For example, with 5.000 v in, the fifth breakpoint is adjusted for —2.500 v out. Interaction between the breakpoints requires this sequential adjustment to be made many times before the characteristic converges for ± 5 mv accuracy. Also, about —3 mv of d-c offset was cancelled by another source (through a 5K summing resistor). Because of the large number of iterations required, this method should be avoided if possible.

The most satisfactory method is to match a new multiplier to one that is already accurately adjusted. A complete multiplier can be adjusted in this manner in about 15 minutes. Setting the input to the new multiplier for $X = Y = 10 \sin \omega t$ and $-X = -Y = -10 \sin \omega t$ (for $f \approx 40$ cps), then the output, $-10 \sin^2 \omega t$, will be generated entirely by its "positive" circuit. (In the "negative" circuit, the inputs will sum to zero at the summing resistors.) Setting the input to the reference multiplier equal to $X = -Y = 10 \sin \omega t$ and $-X = Y = -10 \sin \omega t$, its output, $10^2 \sin \omega t$, will be generated by its negative circuit. The outputs of the two multiplier networks can be summed at the same summing junction and should cancel. The breakpoints of the positive circuit of the new multiplier are then adjusted for minimum error. The negative network is next adjusted by interchanging the Y and $-Y$ inputs to the two multiplier networks. In practice, it was possible to match the squaring networks within ± 5 mv and typically to ± 2 mv.

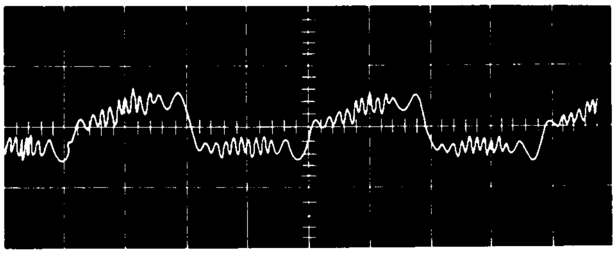
Tests and results

The multiplier was tested for static accuracy, thermal drift, and frequency response. Full-scale sine waves of various frequencies were multiplied by positive and negative constants and also zero. The multiplier output was then summed with an appropriate input to produce an error display. It was found that the static accuracy was approximately ± 10 mv at 25° C. Also, multiplier error caused by misadjustment of the ± 10-v reference supplies was approximately equal to this adjustment error. The photographs in figure 10 show the error at various frequencies (1 Kc, 5 Kc, and 10 Kc) for $XY/10 = (10)(10 \sin \omega t)/10$. The phase compensation approximately doubled the 0.5% of half-scale error frequency, extending it to over 10 Kc. Table 1 shows the dynamic error at various frequencies. This error is caused by a combination of phase shift in the amplifiers and nonlinear error in the squaring networks.

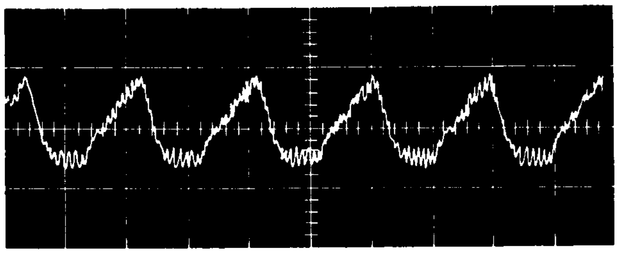
(a) 1 Kc



(b) 5 Kc



(c) 10 Kc



(d) 10 Kc

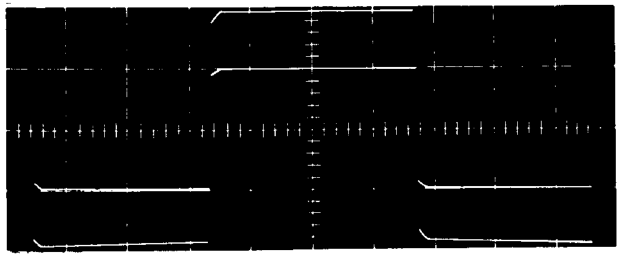


Figure 10 — In (a), (b), and (c) error photographs show the multiplier error in multiplying 10 times 10 sin $2\pi ft$ for $f = 1$ Kc, 5 Kc, and 10 Kc. The scale is 50 mv/cm or 0.50%-of-half-scale/cm. (c) Clearly shows dynamic error is less than ±0.50%-of-half-scale at 10 Kc. In (d) a dual trace photograph shows the input and output of the multiplier for a ±10-v 10-Kc square wave multiplied by $Y = -10$. Different scale factors were used so that the two wave forms could be distinguished (the input is shown at 10 v/cm and the output at 5 v/cm).

Discussion

Measured specifications are presented in table 1. The temperature stabilization by the diodes in the bias networks makes it incorrect to vary the reference voltages to obtain a variable denominator in XY/V. The multiplier can be built and aligned by a reasonably experienced technician (undergraduate engineering student) in about 20 working hours. The cost for components is about \$90 per multiplier, the major part of this cost being \$60 for trimmer potentiometers. The adjustment of the multiplier is easy, once one properly adjusted squaring circuit is available for matching. With a convenient setup, it takes about 15 minutes per multiplier.

As to possible improvements, as a price of the low-impedance wideband design, the multiplier can draw up to 17 ma from each driving amplifier because the summing resistors are so small. 2K summing resistors would probably be satisfactory and would reduce the load on the driving amplifiers to about 7.5 ma maximum. The use of 2K summing resistors would give approximately a 5-v swing out of the absolute-value circuit. (The maximum voltage out of the absolute-value circuit at present is approximately 7.5 v.) With a 5-v swing, the breakpoint spacing would be 0.5 v. This 0.5-v spacing would result in an increased overlapping of the breakpoint adjustments. In the writer's opinion, such overlapping might be about optimum for accurate curve fitting, although it would make the circuit more sensitive to thermal drift and more difficult to adjust.

A second disadvantage is the use of 20 trimmer potentiometers. These are the most expensive and bulky components in the circuit. If the circuit were constructed with all 1% resistors, perhaps adjustment of fewer breakpoints would be possible. Since only seven multipliers were built, however, it was not deemed worthwhile ordering special resistors for this purpose.

Finally, the method useo for phase compensation was probably not as good as splitting the feedback resistor with a capacitor to ground. (Figure 9).

If the ultimate in high-frequency performance were desired, however, four separate squaring circuits would be better than the absolute-value squaring circuits, which have cascaded phase shifts from summing resistors, absolute-value circuits, and limiter channels. In any case, the frequency response of the squaring circuits was comparable to that of the amplifiers used.

Table 1 – ASTRAC II multiplier specifications

Static accuracy ±0.20% of half-scale from 15 to 40° C, ±0.10% at 25° C

Thermal drift ....Less than 0.7 mv/°C

Dynamic error

$XY/10 =$ (0) (10 sin $2\pi ft$)		$XY/10 =$ (10) (10 sin $2\pi ft$)/10	
f(Kc)	E(±% of half-scale)	f(Kc)	E(±% of half-scale)
0.1	0.10	0.1	0.10
1.0	0.10	1.0	0.15
10.0	0.20	10.0	0.50
50.0	0.80	50.0	1.50
100.	1.30	100.	2.50

Driving current required.............17 ma from each driving amplifier
(See figure 4 for special low-current squaring inputs)

Maximum current to summing junction of output amplifier........2 ma

Input voltage range..±10 v

Reference supplies ....±10.000 v at 4 ma each

Table 2 – Requirements for high-frequency performance

1. Effects of stray capacitance were minimized by using low impedance levels.
2. Diodes with low capacitance and fast turn-off time were used (1N4009).
3. Catching diodes to reduce the effects of diode capacitance were used.
4. The frequency response of the completed multiplier was improved by adding phase lead to compensate for phase lag in the output amplifier.
5. Completely parallel limiter channels were used. These are less likely to cause phase shift than a series-parallel scheme.

Table 3 – Requirements for high static accuracy

1. The four summing resistors for each squaring circuit were matched to within 0.01%.
2. The absolute-value circuit diodes were matched for equal forward voltage drops ±0.5 mv at 2 ma of forward current.
3. The absolute-value circuit diodes were slightly forward biased for zero input.
4. Ten segments were used for each squaring circuit, *i.e.*, ten parallel limiter channels were used.
5. Potentiometer adjustments of the breakpoints to compensate for component tolerances were used.
6. Diodes were placed in the bias lines for temperature stabilization.
7. For drift-free zero error at zero it is best to have all channels with an "off" diode in the series path at zero. This prevents slight errors in the reference voltages from causing error at zero.
8. A diode was placed in the first limiter channel so that the rounding of its characteristic improved the accuracy near zero.
9. The squaring circuits were designed and adjusted as a unit, so the overall transfer characteristic takes into consideration nonlinearities in the absolute-value circuit and nonlinear loading of the summing resistors.

ACKNOWLEDGMENT

The project described in this report is part of a hybrid analog-digital computer study directed by Professor G. A. Korn. The writer is very grateful to the Office of Aerospace Research, Information Research Division, Air Force Office of Scientific Research and to the Office of Space Sciences, NASA for their support of this study under grants AF-AFOSR 89-64 and NASA NSG/03-002-024.

REFERENCES

1 G A KORN
Fast analog-hybrid computation with digital control: the ASTRAC II system
Proceedings 4th AICA Conference Brighton England 1964
Presses Académiques Européennes Brussels 1965

2 J V WAIT
A hybrid analog-digital differential analyser
Proceedings FJCC 1963

3 G A KORN T M KORN
Electronic analog and hybrid computers
McGraw-Hill New York 1964

A Hybrid Analog-Digital Parameter Optimizer for ASTRAC II

Reprinted with minor revisions by the author from the Proceedings of the 1964 Spring Joint Computer Conference.

by BAKER ADAMS MITCHELL, Jr. • *Electronic Associates, Inc.* • Princeton, New Jersey

BAKER ADAMS MITCHELL was born in Columbia, Tennessee and received a B. S. at Duke University in Durham, North Carolina. Upon leaving Durham the leisurely pace of the South was replaced by the extreme rapidity with which things are done in Tucson . . . particularly analog computation.

After two years of seeing that repetition is not boring when done at 1 Kc rates, he returned to the East where he is presently working with the Advanced Study Group of EAI at Princeton, New Jersey.

The dimension of size has been added to his perspective, and one frequently hears him mumbling about 90 parameter optimizers!

ABSTRACT

A new automatic multiparameter optimizer for iterative differential analyzers employs sequential random parameter perturbation. The nominal parameter point changes whenever the random perturbations improve the system performance measure. Binary counters operate simple digital-to-analog converters to implement parameter storage, multiplication, and step-size changes. All-digital logic yields different types of random perturbations, viz., simple random walk, random walk with reflecting or absorbing barriers, and various types of correlation over successive perturbations.

INTRODUCTION

This paper describes an optimizer designed to find system parameter combinations which optimize a functional, F, such as

$$F(\alpha_1,\ldots,\alpha_n) = \int_0^T [y^2(t) + u^2(t)]dt \qquad (1)$$

where

$$y(t) = y(t,\alpha_1,\ldots,\alpha_n) \qquad (2)$$

$$u(t) = u(t,\alpha_1,\ldots,\alpha_n) \qquad (3)$$

are state and control variables depending on the unknown parameters $\alpha_1,\ldots,\alpha_n$ in accordance with the system equations

$$\dot{y}_i = f_i(y_1,\ldots,y_k;\, u_1,\ldots,u_m;\, t) \qquad (4)$$

The new optimizer is designed to work with a fast all-solid-state iterative differential analyzer (ASTRAC II) which is capable of producing complete solutions $y_i(t)$ and the corresponding values of the performance measure $F(\alpha_1,\ldots,\alpha_n)$ for up to 1000 new parameter combinations per second.[1]

To simplify optimizer logic and memory requirements in problems involving many parameters, we simultaneously implement random perturbations[2] on all parameters α_k and step to the perturbed point whenever the perturbation yields an improvement in the performance measure $F(\alpha_1,\ldots,\alpha_n)$.

Simple all-digital logic permits implementation of different sequential optimization strategies, including correlation between random-perturbation vectors and step-size changes depending upon past successes and failures. The analog integrator/multipliers commonly used to set system parameters have been replaced by simple, reversible binary counters driving D/A converters[3] for simplified design and improved reliability.[4] The principle of the optimizer is shown in the block diagram of Figure 1.

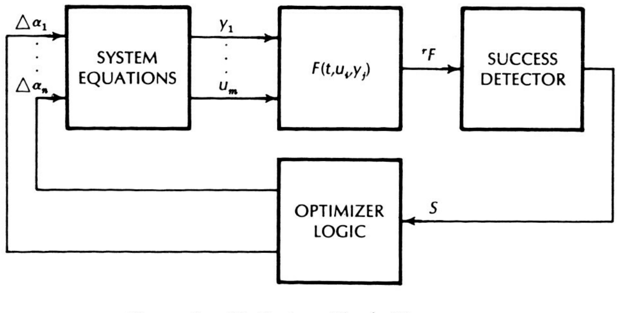


Figure 1 — Optimizer Block Diagram

Figure 2a shows the parameter-space path over which a conventional deterministic system would optimize a simple two-parameter system. Starting with a trial set of parameters ${}^0\alpha_i$, the conventional optimization logic employs the results of successive differential analyzer runs to obtain succeeding parameter values

$$ {}^r\alpha_i = {}^{r-1}\alpha_i + {}^r\triangle\alpha_i \qquad (5) $$

which successively improve ${}^rF = F({}^r\alpha_i, \ldots, {}^r\alpha_n)$. The most frequently used method employs *n trial steps* to compute approximate gradient components $\triangle F/\triangle\alpha$ in each parameter direction; these gradient components are then stored and used to compute the optimal correction $\triangle\alpha_i$ for a *working step*, or for a series of working steps in the same direction.[8]

Such deterministic methods require complex logic and storage. Although they may converge well for favorable performance functions $F(\alpha_1, \ldots, \alpha_n)$, they may "hang up" on ridges or in canyons of the multidimensional landscape of the performance measure domain.[5] Furthermore, if the performance measure contains discontinuities, nonlinearities, or large higher-order derivatives with respect to the parameters, our information of past performances will be of little value in determining succeeding steps; thus the step-size may have to be reduced to such a degree that convergence to the optimum is extremely time-consuming.

Figure 2b shows how a pure random-perturbation scheme might optimize the same function. Here, $\Delta\alpha_i$ may be positive, zero, or negative with equal probability, and the nominal parameter point is moved as soon as the first improvement in the performance function occurs. No attempt is made to affect future perturbations by past results or gradient methods.

On the other hand, if perturbations are to be correlated with past successes or failures, then the optimization path might appear as shown in Figure 2c. Such a scheme causes future increments α_i to favor the direction in which past improvements in the performance function were made. Notice, however, that we still do not require computation of individual gradient components, as in deterministic gradient optimization schemes. Hence, logic and memory requirements are reduced.

A feature of schemes b and c is that, if the result of adding the last set of increments ($\Delta\alpha_1$ and $\Delta\alpha_2$ in the cases shown in Figure 2) is unfavorable, these increments are subtracted before the next set is added. Thus each trial starts from the last "successful" point.

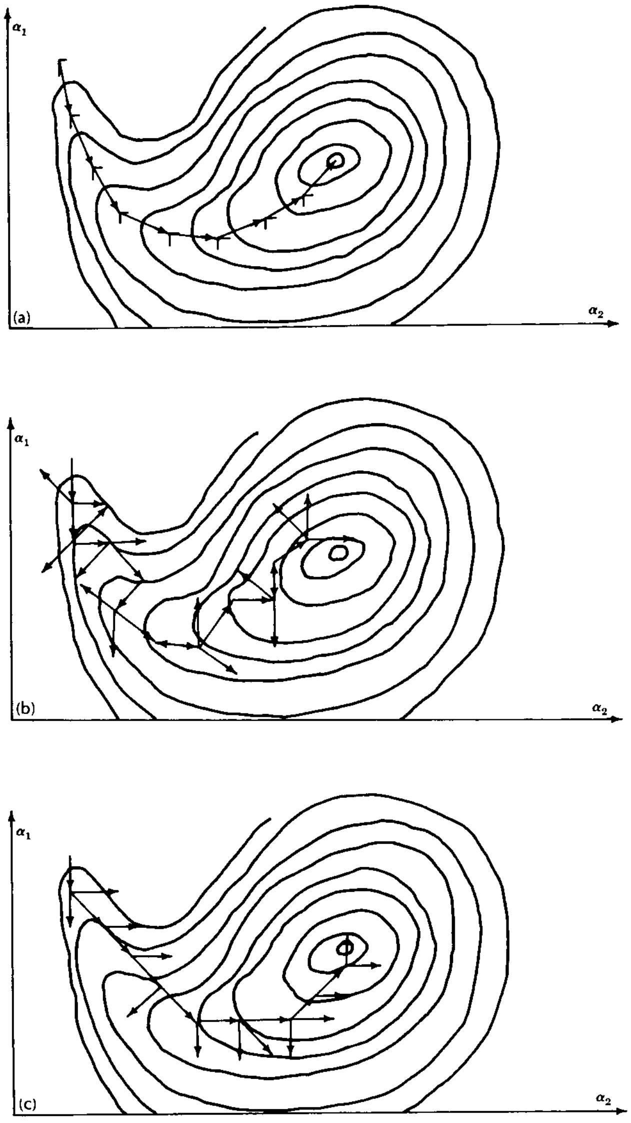


Figure 2 — Typical Optimization Paths

Motivation for Random Search

The basis for all direct computer methods of parameter optimization is the same: Using a mathematical model or simulated system, we set the parameters to some trial values and compute the performance criterion. Then according to some *rule*, we reset the parameters to new values and again compute the performance criterion. This procedure is repeated until some desired degree of improvement is obtained. Naturally it is hoped that the rule for adjusting the parameters will take *maximum* advantage of the knowledge gained from observing previous trials, and by so doing achieve the optimum set of values for the parameters in the shortest possible time. Usually, however, this rule depends *solely* upon the knowledge gained from recent past trials and this is thought to be equivalent to using this knowledge to maximum advantage.

If, however, the performance criterion contains discontinuities, nonlinearities, or large higher-order partial derivatives with respect to the parameters, our information of recent past behavior (actually a total or partial *first* derivative) may be of little value for the determination of successive steps; if the step size is too large, this information from the preceding step (or from a short forward trial step) will be totally misleading. Thus, with conventional, deterministic perturbation such as the gradient method, one may be forced to reduce the step size so much that convergence is excessively slow. For this reason it has been suggested that randomness be introduced into the search rule—perhaps in proportion to the expected severity of the discontinuities, nonlinearities, etc., present in the cost function domain.

In reality, it may be difficult to make any reliable prediction concerning the behavior of the performance function. Even with reasonably well-behaved performance functions, it can be quite difficult to foresee "ridges," "temporary plateaus," "saddle-points," and other features which render deterministic rules far from foolproof.

Reference 5 goes further into such motivation for random-search methods. Figures 3 and 4 illustrate situations in which prediction is likely to be difficult.

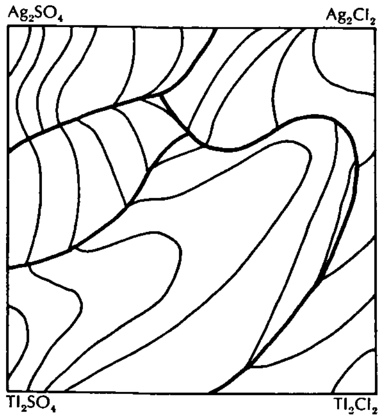
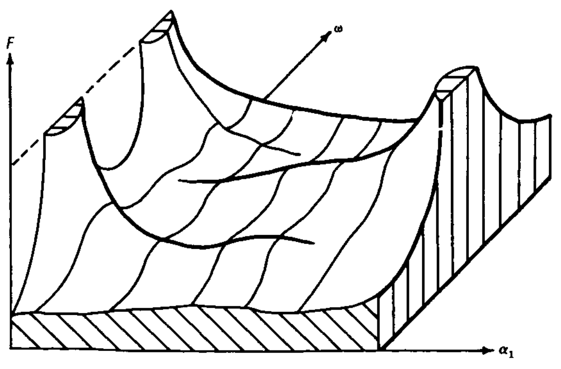
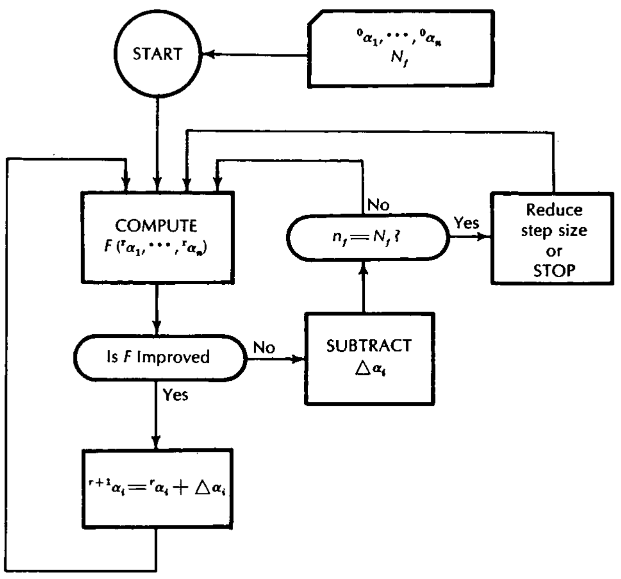


Figure 3—Function with Discontinuities
The function defined represents a system composed by particular percentages of each of the four compounds shown at the corners. Contour lines are drawn in order to indicate values of a property on the system. The heavy lines are lines of discontinuity in slope.[5]





START
${}^0\alpha_1, \cdots, {}^0\alpha_n$
N_f
COMPUTE
$F({}^r\alpha_1, \cdots, {}^r\alpha_n)$
No
$n_f = N_f$?
Yes
Reduce step size or STOP
Is F Improved
No
SUBTRACT
$\Delta\alpha_i$
Yes
${}^{r+1}\alpha_i = {}^r\alpha_i + \Delta\alpha_i$

Figure 5 — Flow Diagram for a Typical Optimization Routine

PRINCIPLES OF OPERATION

Most of the optimization strategies proposed here can be based on the flow chart of Figure 5. The operation common to all the various strategies consists of incrementing all parameters simultaneously by individual random increments $+\Delta\alpha$, $-\Delta\alpha$, or zero; the common magnitude $\Delta\alpha$ (step size) of these increments is subject to a separate decision. The term *success* will be used to indicate that the incremented set, $\alpha_1 + \Delta\alpha_i$, has yielded a more favorable value for the performance measure than was obtained with the unincremented set, α_i. A *failure* will mean that the incremented set yielded a less favorable value for the performance measure, in which case the failing increment is subtracted from the parameter before a new increment is added. Successive failures and successes can be counted and used to decide when an increase or decrease in the step size might be advantageous. A binary noise generator[7] together with digital correlation logic decides what the sign of each new increment will be. Thus, the overall effect of the perturbation scheme is an n-dimensional random walk.

← Figure 4 — Function with Saddle Points
This sketch of a dynamic vibration absorber shows the amplitude of vibration F plotted over the frequency range ω for values of the parameter α_1. The effects of only one of the three parameters, α_1, α_2, α_3, of the actual system could be drawn. A possible criterion for an optimum absorber is to require that the maximum amplitude of vibration yielded by a particular set of parameter values, α_1, α_2, α_3, be minimum over the frequency range of interest.[5]

THE ASTRAC II SYSTEM

The dynamic system and the performance criterion are to be simulated on the Arizona Statistical Repetitive Computer, ASTRAC II, although ASTRAC I served for preliminary studies. ASTRAC II is a ± 10-volt, all-solid-state iterative differential analyzer capable of iteration rates of 1 Kc as well as real-time computation. The first 20-amplifier section of ASTRAC II is to be completed in the fall of 1964.

The *analog section* has a large conventional-appearing patchbay. 20-Mc transistorized amplifiers mounted in shielded cans plug directly into the rear of the patchbay without any intervening wiring. The analog section will comprise sample-hold memory pairs, comparators, analog switches, switched integrators, diode quarter-square multipliers, and diode function generators.

Timing and logical control is furnished by the *digital section,* which provides timing pulses, integrator RESET pulses, and sampling pulses. The digital section has its own patchbay with removable patchboards for implementing various logic functions. Patchable gates and flip-flops are used in conjunction with the prewired timing and RESET circuits. In view of the amount of logic involved in the optimizer, however, it was thought best to build it as a separate digital section with its own removable patchboards, devoted to this purpose. The complete ASTRAC II optimizer will not only implement the sequential random search optimization described here, but will permit comparison with deterministic optimization schemes.

Optimizer Logic

The digital logic of the optimizer is subdivided into basic functional units whose inputs, outputs, and control points are wired to its patchbay (Figure 6). With a different prepatch panel, these components are also available for other uses besides optimization. In particular, the parameter-setting circuits will also serve for experiments with deterministic optimization schemes.

The functional units are built of commercial plug-in logic cards interconnected on racks with wire-wrap terminations for ease of modification and expansion.

The resistor networks and switches comprising the D/A multipliers are mounted in shielded plug-in cans adjacent to the operational amplifiers behind the analog patchbay. Shielded digital control lines connect each D/A multiplier to the optimizer patchbay.

Hybrid Analog-digital Noise Generator

ASTRAC II employs a new noise generator[10] producing pseudo-random maximum-length shift-register sequences at any desired clock rate up to 4 Mc. We may obtain either a single pseudo-random sequence repeating after 33 million bits, or four uncorrelated sequences one-fourth as long.

Success-failure Indicator

Essentially, the function of this circuit (Figure 7) is to compare the value of the best performance measure, ^{L}F, obtained to date, with the performance measure just yielded by the last computer run, ^{r}F.

The flip-flop output U_3 controls the operation of the second sample-hold and at the same time sets flip-flop 7 whose output, S, indicates to the digital optimization logic whether or not a success was obtained with the last set of incremented parameters. If the last run was a *failure,* (*i.e.,* $^{r}F < {}^{L}F$ for maximization), then U_3 and S remain zero, and ^{L}F is still held as being the best value. If, however, the last computer run was a *success* ($^{r}F > {}^{L}F$), then U_3 becomes "1," which causes the second sample-hold to take on the value just obtained.

If it is known that F will always be monotonically increasing during the latter part of the COMPUTE period, Switch 3 and U_2 need not be used.

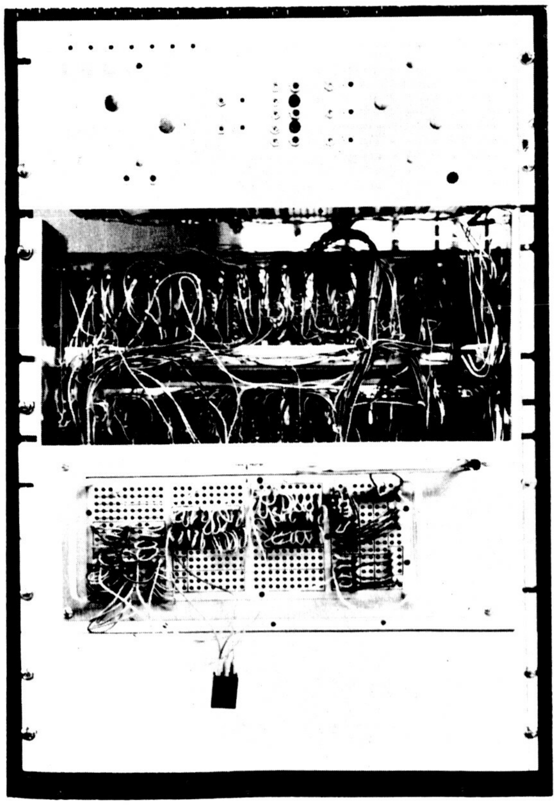


Figure 6 — Photograph of Optimizer

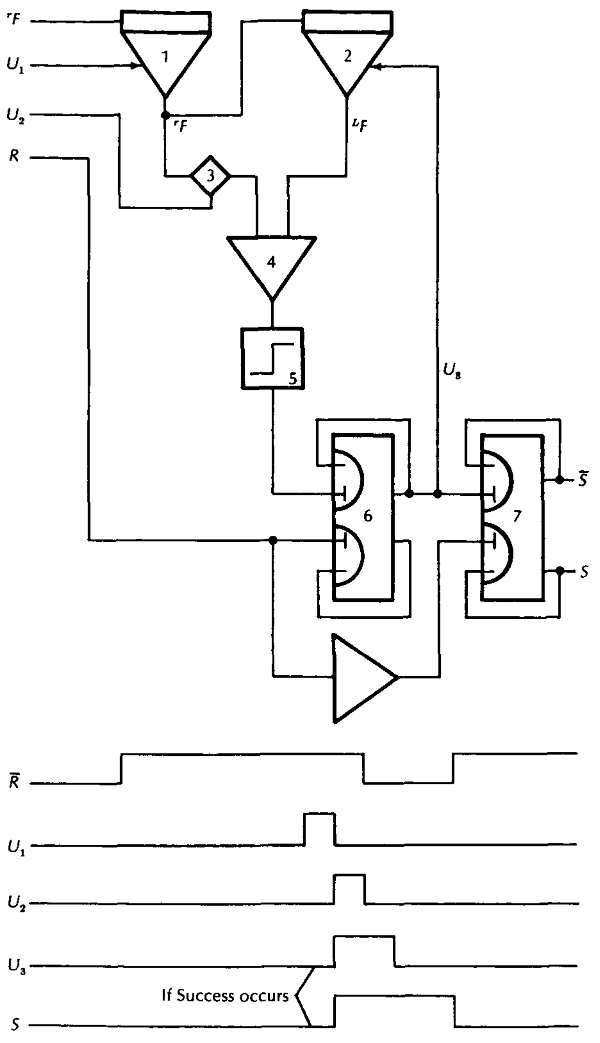


Figure 7 – Success-Failure Indicator

Master Clock

The Master Clock provides all timing pulses needed for sequencing logic operations throughout the digital optimization routine. It consists of a four-bit Gray-code counter driven by a 1-Mc pulse, C_l, from the differential-analyzer digital control unit. C_l is gated to the counter during the differential-analyzer RESET period. With the exception of the Success-Failure Indicator circuit, where timing is under control of the differential-analyzer, no optimization *logic* is performed during the COMPUTE period of the differential-analyzer.

The sequence of the timing pulses from the Master Clock is shown in Figure 8. $\bar{S}$, as shown, occurs only after failures, *i.e.*, $S=1$. If $S=0$, $\bar{S}=1$ throughout the RESET period. A success or failure also causes n_S or n_F to occur (Figure 15).

PARAMETER SETTING

The flexibility needed to implement a variety of different optimization-logic schemes while maintaining simplicity, reliability, and low cost is achieved by using a unique method of parameter setting.

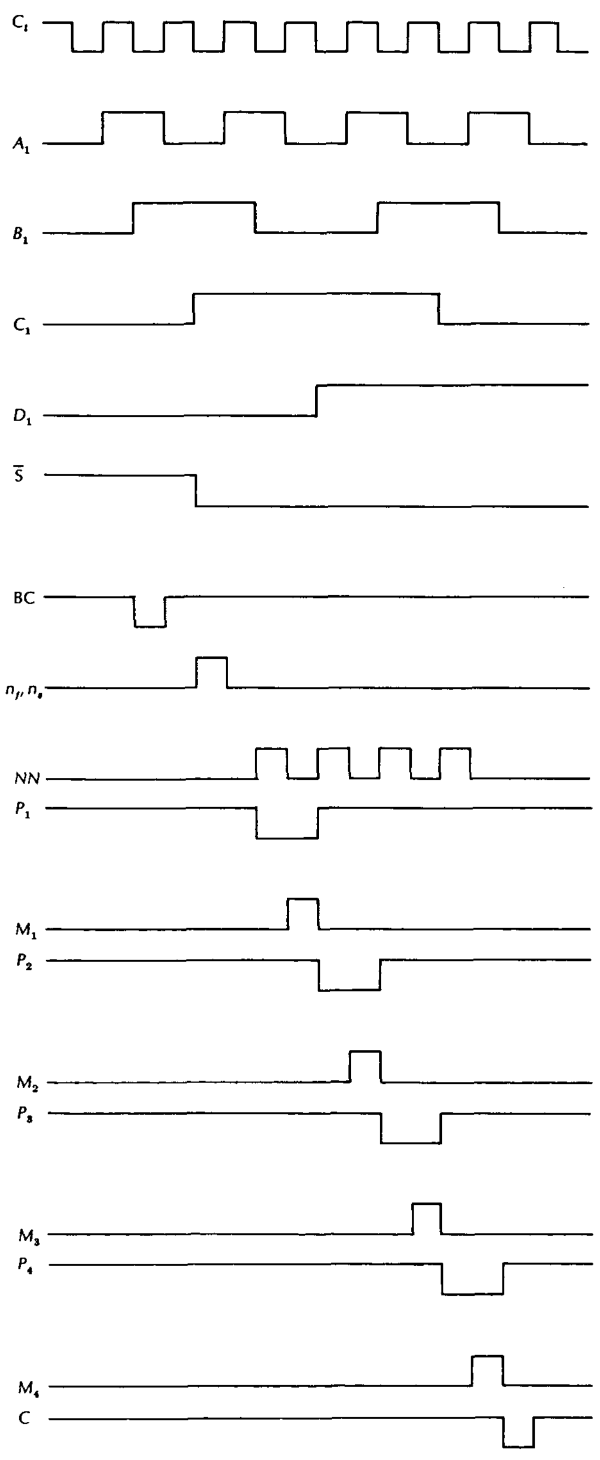


Figure 8 – Pulse Sequence from Master Clock

Binary-counter Operation

Referring to Figure 9, the binary up-down counter increments whenever a pulse appears on the "I" line. The right-left shift register contains zeros in all except one of its stages, and the position of this "1" selects the stage of the binary counter which is to receive the "I" pulse. By controlling the D/A multipliers, the counter has then increased or decreased α_i by an increment $\Delta\alpha_i$; the magnitude $\Delta\alpha_i$ is determined by the position of the "1" in the shift register, and the sign (+ or —) is determined by the logic level on the UD line (1 = count up, 0 = count down).

U-L-D Digital Logic

The U-L-D (UP-LOCK-DOWN) Logic is composed of two sections: a central *U-L-D Selector Circuit* (Figure 10a) and a *U-L-D Memory* associated with the binary counter for each parameter (Figure 10b).

The selector circuit accepts two uncorrelated random bits ${}^1N_1$, ${}^1N_2$ obtained from the ASTRAC II noise generator. Depending on the interconnections of the selector-circuit gates G_1, G_2, G_3, G_4 to gates U, L, D, the gate outputs U_0, L_0, D_0 will have different joint probability distributions as shown in Table I.

Table 1 – U-L-D Probability Distributions

P(U = 0)	P(L = 0)	P(D = 0)	Connect	to Gates
1	0	0	G_1, G_2, G_3, G_4	U
3/4	1/4	0	G_1, G_2, G_3 G_4	U L
3/4	0	1/4	G_1, G_2, G_3 G_4	U D
1/2	1/2	0	G_1, G_2 G_3, G_4	U L
1/4	3/4	0	G_1 G_2, G_3, G_4	U L
1/2	1/4	1/4	G_1, G_2 G_3 G_4	U L D
1/4	1/2	1/4	G_1 G_2, G_3 G_4	U L D
1/2	0	1/2	G_1, G_2 G_3, G_4	U D
0	1	0	G_1, G_2, G_3, G_4	L
1/4	1/4	1/2		
0	3/4	1/4		
0	1/2	1/2		
1/4	0	3/4		
0	1/4	3/4		
0	0	1		

Effectively, these distributions can be obtained by placing a logical "1" on the RC line.

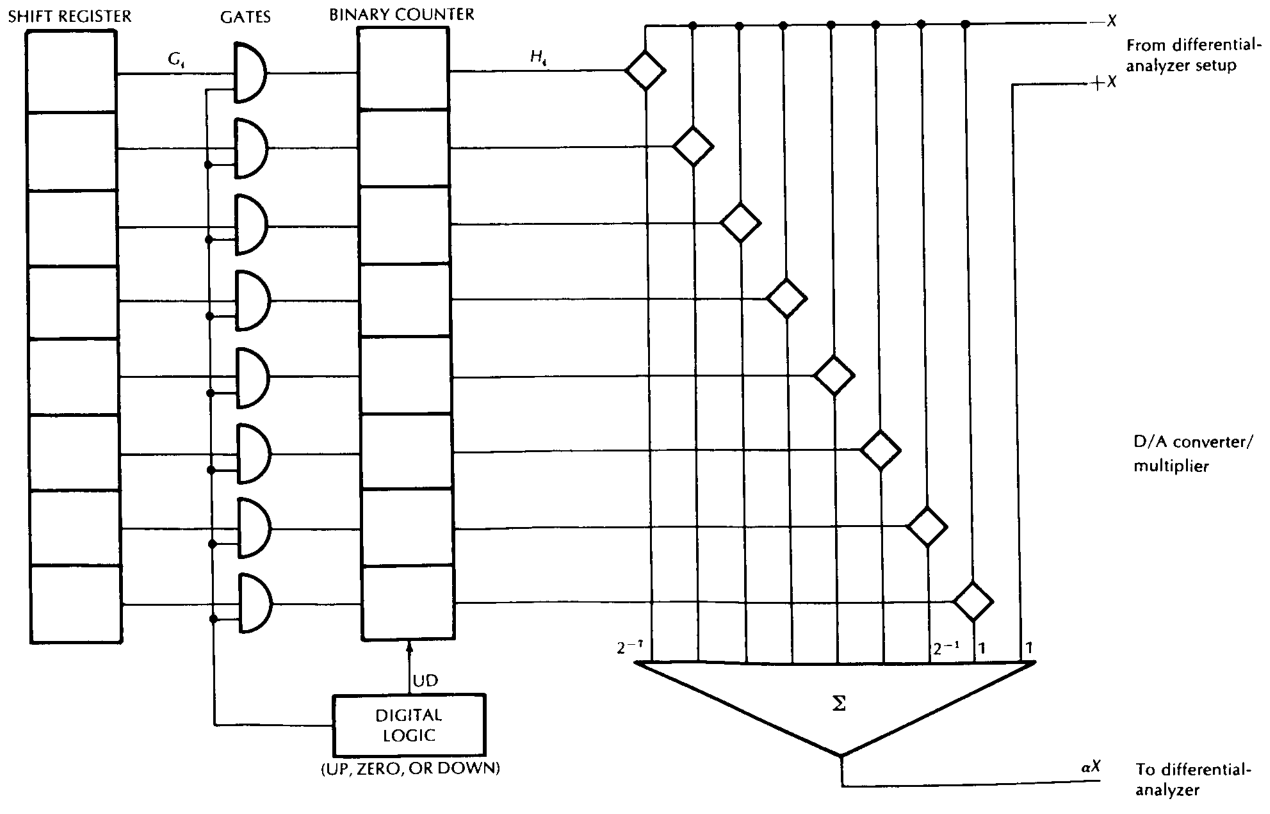


Figure 9 – Parameter Setting

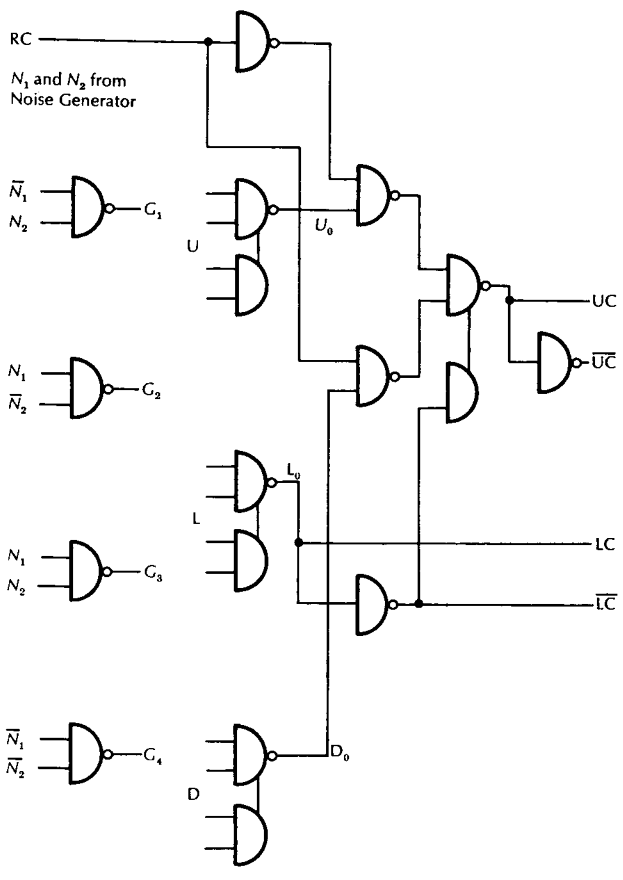


Figure 10a – U-L-D Selector Circuit

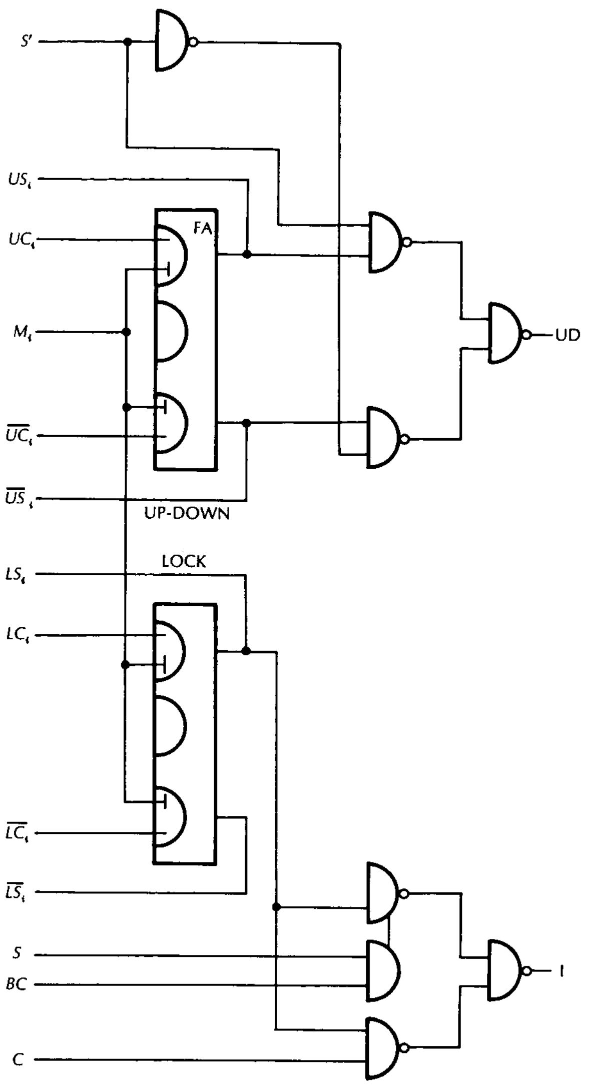


Figure 10b – U-L-D Memory Circuit

The gate outputs, U_0, L_0, D_0, determine the states of the d-c set and reset level controls on the UP-DOWN flip-flops and the LOCK flip-flops of all U-L-D Memories (one for each parameter) simultaneously. The pulse M_1 from the Master Clock now first sets the U-L-D flip-flops of the *first* parameter to their proper states as determined by the first set of random bits ${}^1N_1$, ${}^1N_2$.

Next, the noise generators are pulsed again by the Master Clock (pulses NN) producing two new random bits, ${}^2N_1$, ${}^2N_2$, which determine a new set of states for the d-c set and reset level controls of the U-L-D Memory flip-flops. Now, the pulse M_2 to the U-L-D Memory of the *second* parameter sets *its* UP-DOWN and LOCK flip-flops in accordance with ${}^2N_1$, ${}^2N_2$. The process repeats for the remaining parameters.

U-L-D Memory

Note that we shall require two decisions for each parameter. The LOCK circuitry decides whether α_k is to be incremented or not incremented ("locked"). The UP-DOWN circuits decide the direction (up or down) of the increments $\triangle\alpha_k$, if any. We shall consider the UP-DOWN circuits first.

If an UP-DOWN flip-flop is set (reset), a logical 1 (0) will appear on the "UD" line to the binary counter, *if* $S' = 1$. We now come to the LOCK decision. If the LOCK flip-flop is set or reset the incrementing commands "C" and "BC" from the Master Clock will or will not carry through the gates on "I" to increment the counters in the direction dictated by "UD."

Correlation Circuit

This circuit can be patched so as to introduce correlation between successive random perturbations according to some strategy selected to speed optimization. Suppose that the last set of increments succeeded in improving the performance function. Assuming that the performance function is fairly well-behaved, the greatest chance for another success lies in weighting each parameter so that it will probably increment in the same direction as in the previous trial (strong positive correlation). By the same reasoning, after a failure a strong negative correlation might be introduced. After either a failure or a success, if a particular parameter had been *locked (i.e.,* its last increment was zero), then a probability function giving equal weight to all three states might be desirable for the next trial.

Referring to Figure 11, the correlation circuit is patched as desired and senses the value of the last increment in each U-L-D Memory, starting with the first parameter. If $\triangle\alpha_k$ is positive, the RC line changes the selector gating so as to increase the chance of a positive $\triangle\alpha_k$ for the next run if the last run was successful. The reverse can take place if the last run was a failure, depending upon the correlation control "CC."

If, however, $\triangle\alpha_k$ was zero, as indicated by the state of the LOCK flip-flop, then the correlation line RC is turned on or off with equal probability by a random-noise input. This entire process is repeated for each parameter in turn, as they are sequenced by the pulses P_i from the Master Clock.

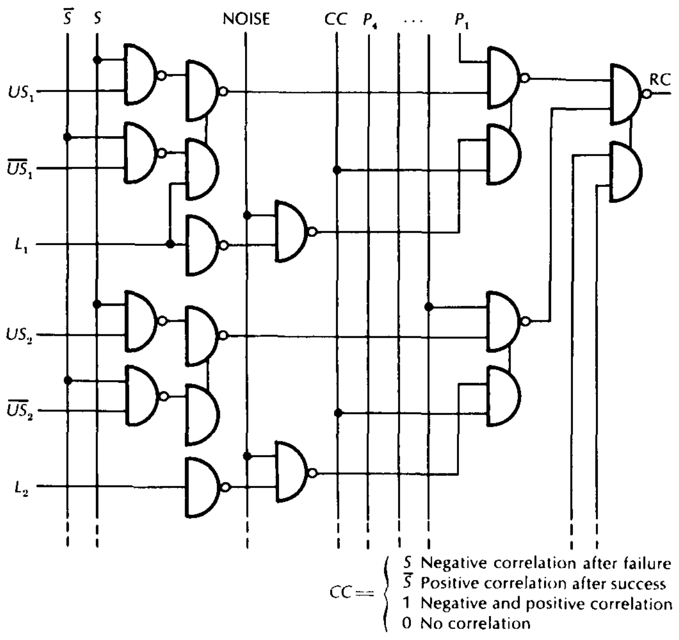

$\bar{S}$
S
NOISE
CC
P_4
$\cdots$
P_1
RC
US_1
$\overline{US}_1$
L_1
US_2
$\overline{US}_2$
L_2
CC =
S Negative correlation after failure
$\bar{S}$ Positive correlation after success
1 Negative and positive correlation
0 No correlation

Figure 11 — Correlation Circuit

OPTIMIZATION STRATEGIES

1. *Pure Random Perturbations*

Pure random perturbations are achieved if we preset the "1" in the shift register to gate the count pulse into the binary counter stage yielding the desired step distance, *e.g.,* 1/2, 1/4, . . ., 1/128. The noise generator is connected *directly* to the SET and RESET level controls of the UP-DOWN flip-flop of each parameter. During the analog RESET period, the noise generator is pulsed once for each parameter shortly before the sequenced pulse to the proper UP-DOWN flip-flop arrives. Thus, the UP-DOWN flip-flops sequentially assume the successive states of the noise source. The count pulse, C, is then fed to all parameters simultaneously. Hence, all parameters increment by the same magnitude, but with random signs during each analog RESET period, executing a random walk in the perturbation scheme.

With pulses *S*, *S'*, and BC added to the U-L-D Memory, the set of increments used in the preceding run can be subtracted from the counters prior to the addition of increments for the next run. Thus, after a failure in an optimization run, the counters can be returned to the state which yielded the last success before making another search.

2. *Random Walk with Reflecting Barriers*

This scheme is exactly the same as the pure random walk with one additional operation. Prior to assigning a new set of signs to the UP-DOWN flip-flops, a pulse—possibly from a comparator in the analog system—to signify that a parameter has reached a barrier, can be gated to the U-L-D logic of the parameter to be reflected, causing its UP-DOWN flip-flop to complement and, later, to ignore the sign that is assigned to it by the noise source. In this manner, the parameter will have been *reflected* to its old position held two analog COMPUTE periods previously.

3. *Random Walk with Varied Step Size*

This scheme also contains only one addition to the pure random walk. When it is desired to increase or decrease the step size—possibly after a certain number of successive failures have been counted—one has only to insert the increase or decrease command (*i.e.,* the counter output) into the shift-right or shift-left input of the shift register. This gates the count pulse either to the next higher or next lower parameter counter stage. The noise generator then assigns polarities to the UP-DOWN flip-flops, and the parameters are incremented by the new step size.

For this purpose, the optimizer contains two success-failure counters.

4. *Correlated or Biased Random Walk*

Since two independent noise sources are available, one can arrange $P(N_1N_2)=1/4$, $P(N_1\bar{N}_2)=1/4$, $P(N_1 \times \bar{N}_2)=1/4$. These can combine to give $P(X)=1/2$, or $P(Y)=3/4$, where $X=N_1N_2+N_1\bar{N}_2$, $Y=N_1N_2 + N_1\bar{N}_2+\bar{N}_1N_2$. There are three possible states for the U-L-D Memory: UP, DOWN, and LOCK. If a failure occurred in the Up state, then after subtraction of the failing set of increments, it would be desirable to assign weighted probabilities which would be more likely to result in the next state being Down. The probability distributions which can be prepatched are listed in Table 1. Likewise, if a success occurs in the Down state, the next assignment of states could be weighted on the same basis.

This strategy can be combined with step size variations implemented with the aid of the success/failure counters.

TESTS

The following tests were carried out using the optimizer in conjunction with ASTRAC I, a ± 100 volt, 100-run/sec iterative differential analyzer.

In order to generate performance measures with the precise characteristics desired, only algebraic functions $F(\alpha_1,\alpha_2)$ were used for quantitative evaluation of the various optimization strategies. For practical use in optimizing dynamic systems, the optimizer setup would be entirely unchanged, except that the function $F(\alpha_1,\alpha_2)$ would be generated as samples at the end of a differential-analyzer run.

The slow repetition rate of 10 runs/sec was employed only for recording purposes.

Functions Optimized

Initial tests were made with two different types of performance functions.

Minimization of functions of the general type

$$F(\alpha_1,\alpha_2)=\frac{(\alpha_1-k_1)^2}{a}+\frac{(\alpha_2-k_2)^2}{b} \qquad (6)$$

was carried out. The parameters α_1 and α_2 were allowed to vary over ± 100 volts, and convergence of α_1, α_2 was considered to be satisfactory when they were within 0.78 volt of their values which optimized F.

The next experiment involved maximization of a function $F(\alpha_1,\alpha_2)$ exhibiting sharp ridges formed by the intersection of three planes as shown in Figure 12a. The simulation (Figure 12b) uses three amplifiers to form the three planes: $F_1=-\alpha_1+\alpha_2$, $F_2=2\alpha_1-\alpha_2$, $F_3=-1.5\alpha_1-\alpha_2+155.5$. Extremely sharp intersections between the planes were formed by using high-speed "half-comparators." Thus, $F=F_i$, where $F_i \leq F_j$ and $i \neq j$ with $i,j=1,2,3$. The parameters were allowed to vary over ± 100 volts, and the optimum point is $\alpha_1=44.4$ volts and $\alpha_2=66.6$ volts. Convergence was considered to be satisfactory when F was within 0.5 volt of the maximum.

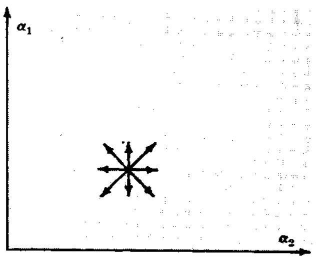


Figure 12a – Optimization Problems on the Sharp Ridge of a Triangular Pyramid

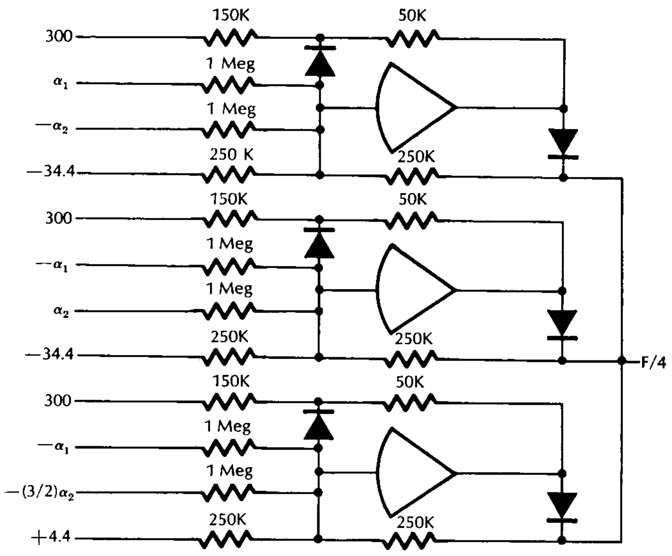


Figure 12b – Ridge Simulation

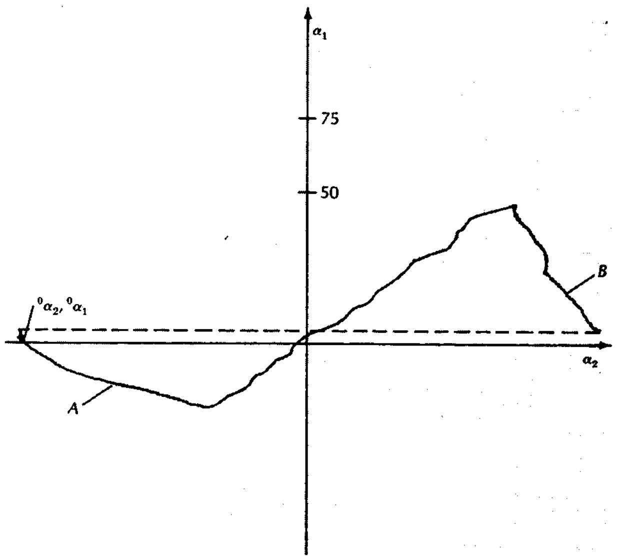


Figure 12c – Optimization Paths on the Ridge of a Pyramid

Strategies Employed

Parameters were allowed to step only after successful runs, *i.e.*, unsuccessful parameter changes were subtracted out.

Various types and degrees of correlation between successive parameter runs were tried. (See section on Correlated or Biased Random Walk, above.)

The step size, $\triangle\alpha$, was increased by a factor of 2 after N_S consecutive successes and decreased after N_F consecutive failures; N_S and N_F were varied.

Results

For the paraboloids (Equation 6), several thousand optimization trials were recorded using many variations within the general class of strategies outlined above.

As the eccentricity of the contours of constant F was increased, convergence was slowed somewhat, but not radically. In no case were more than 70 runs required for convergence.

For the case, $a = b = 1$, using no correlation in the perturbation scheme resulted in an average of 28 runs required for convergence with a standard deviation of 9 runs.

The most favorable step-size variation strategy was to increase $\triangle\alpha$ after 2 successes and decrease $\triangle\alpha$ after 2 failures.

The initial points of ${}^0\alpha_1$, ${}^0\alpha_2$ were always kept at the extreme of their ranges and only a slight decrease in convergence-time was noted as the initial point was placed closer to the optimum.

The particular ridge (Equation 7; Figure 12a) to be discussed here is one expressly designed to present the greatest difficulty to the optimizer. Referring to Figure 12a, it is seen that improvement is quite difficult if the parameter point falls close to the ridge. Had the sides not sloped so steeply, or had the ridge been oriented differently with respect to the parameter axis, the optimizer would have found convergence more natural.

The optimization of this function without step size changes is shown in Figure 13a (α_1 versus time, α_2 versus time); this corresponds to path A in Figure 12c (α_1 versus α_2). Note that the parameter point followed the direction of greatest improvement until it reached the ridge; then both parameters were forced to zigzag up the ridge to the peak. The step size was 1.56 volts, and convergence required 1800 runs. In Figure 13b, the α_2 counter complemented at the beginning of both trials No. 1 and No. 2. Thus, it was not necessary to climb the longest ridge. The paths converged more quickly up the shorter ridge (path B in Figure 12c). This favorable possibility is not present in ordinary gradient techniques. Path A in Figure 12c is most nearly like a path resulting from conventional gradient techniques.

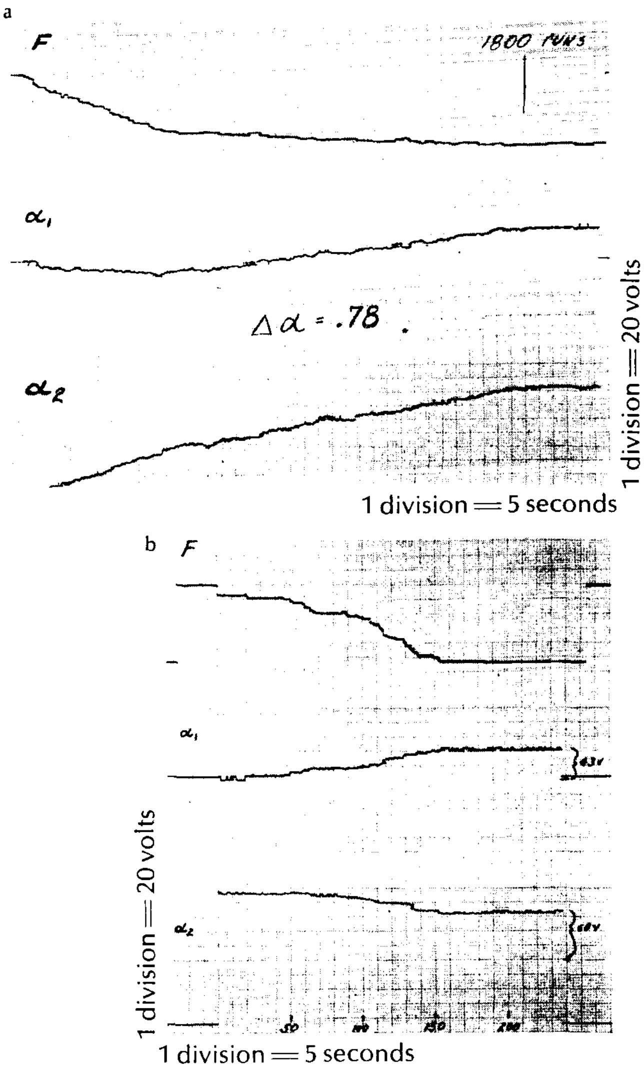
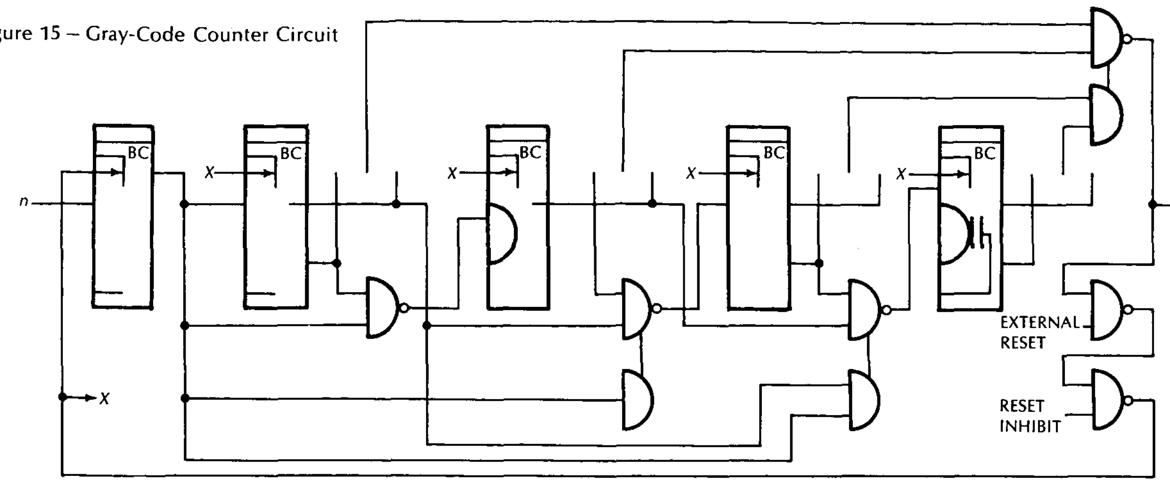


Figure 13 – Ridge Optimization

Figures 14a and 14b show a series of correlated random searches with step-size variations employed in the strategies. The optimizer is not confined to merely zigzagging slowly up the ridge, but can traverse great distances while searching for improvements. Correlation improved convergence time somewhat, but only when weak positive correlation after successes was used. For 45 trials using this correlation, the average time to converge was 73 runs if the step-size was decreased after 2 failures. Other types of correlation slowed convergence by 10-20 per cent.

For comparison note that an optimizer going directly in a straight line to the optimum point would require 106 steps if a fixed step size of 1.56 volts were used.

Figure 15 – Gray-Code Counter Circuit



APPENDIX I

The following units are contained in the optimizer digital section and/or are wired to the optimizer patchbay. On the drawings, a small circle on the end of a wire indicates a patchbay termination.

1. Three 4-bit Gray-code counters. One of these counters has a free-running multivibrator for the input. This counter is used exclusively for sequencing operations throughout the logic scheme subroutine, e.g., the analog RESET period. The remaining two Gray-code counters have their flip-flop outputs adjacent to two gates on the patchboard. These may be patched to yield outputs after any preset number of input pulses. These counters may reset themselves or may be reset externally (Figure 15).
2. One 8-bit right-left shift register. The d-c set and reset lines, along with the set and reset outputs, appear on the patchboard for parallel drop-in on command. Also the set and reset level controls for the first and last flip-flops are on the patchboard, thus permitting operation as a ring counter. Four inputs are provided for shifting in either direction (Figure 17b).
3. Four 8-bit up-down binary counters. The d-c set and reset lines, along with the set and reset outputs appear on the patchboard for parallel loading. A gated complement input for each flip-flop is brought out, allowing the counter to be stepped at any stage. A flip-flop with associated gates permits both pulse and level control of the up-down lines (Figure 17a).
4. Four 8-bit D/A multipliers. The digital control lines for the D/A multipliers (mounted in shielded cans behind the analog patchbay) are wired to the digital patchbay. Each set of lines is associated with one of the up-down binary counters (Figure 16).
5. Two pseudo-random discrete-interval binary noise generators. The noise generator for ASTRAC II is a 25-stage shift register with modulo-2 adders in feedback, generating a maximum length of 3 $\times$ 10^7 random bits. This can be divided into two noise generators, each having a 1.5 $\times$ 10^7-random-bit sequence. The noise generator shifts out a random bit when a pulse is applied to the shift input.

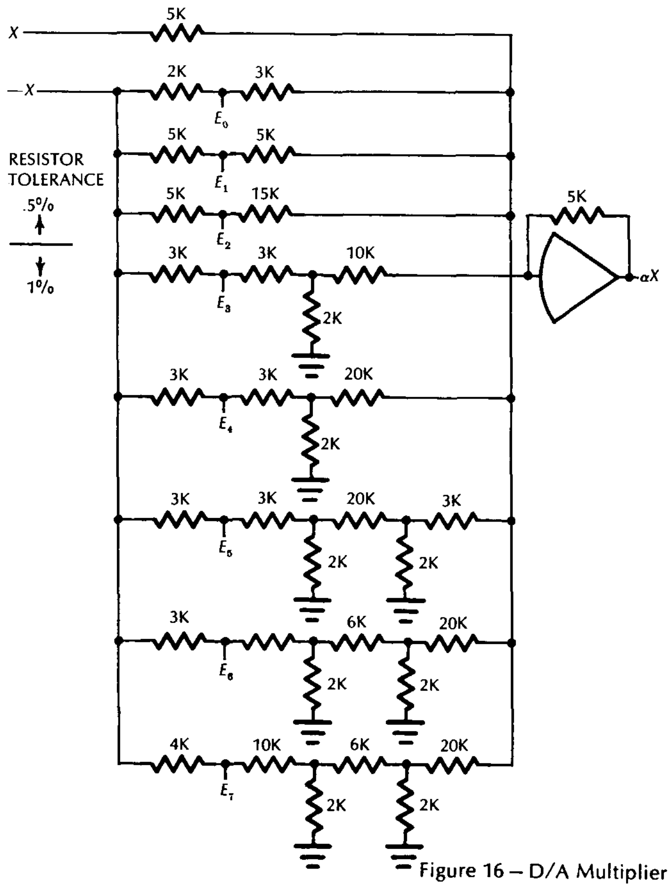
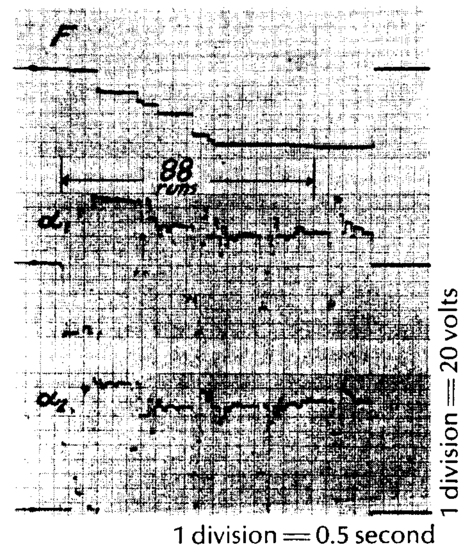


Figure 16 – D/A Multiplier



b

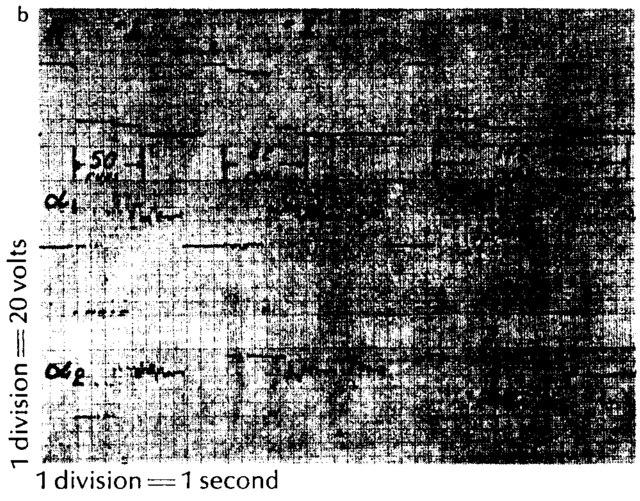


Figure 14 – Ridge Optimization

Stopping Conditions

When the optimum value of *F* is unknown, defining a stopping condition is subject to several factors. One such factor arises when we are not certain that only one peak exists in the domain of *F*. In this case, we would want to cycle the step size through its decrease-increase scheme several times before stopping. Then a rescaling of the simulation might be desirable in case the initial range of the parameters was chosen too large.

CONCLUSIONS

While experience with this optimizer is still limited, it appears that its performance can compare quite favorably with conventional techniques. Better conclusions can be made when this system is expanded to accommodate four parameters and the logic is enlarged to permit implementation of conventional deterministic schemes, thus permitting a direct comparison between random and deterministic methods for optimizing the same function.

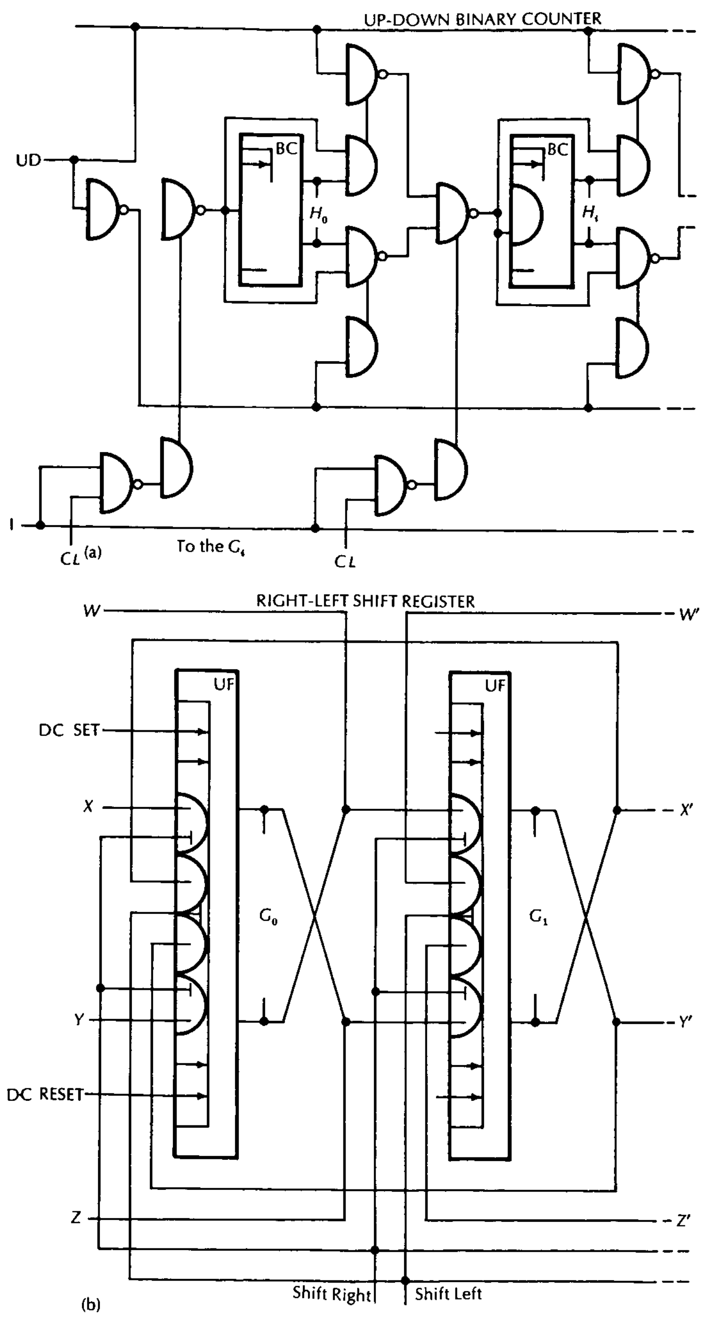
At present, however, the random techniques seem well able to handle cases in which the performance measure is not well-behaved (*e.g.*, the situations illustrated in Figures 3 and 4).

ACKNOWLEDGMENT

The project described in this report is part of a hybrid analog-digital computer study directed by Professor G. A. Korn. The writer is grateful to the Office of Aerospace Research, Information Research Division, Air Force Office of Scientific Research and to the Office of Space Sciences, National Aeronautics and Space Administration for their continuing support of this study under joint grant AF-AFOSR-89-63; and to Professors L. W. Matsch, Dean of Engineering, and H. E. Stewart, Head, Department of Electrical Engineering, for their encouragement and contribution of University facilities.

REFERENCES

1. Korn, G. A., and T. M. Korn, *Electronic Analog and Hybrid Computers,* McGraw-Hill, New York, 1964.
2. Munson, J. K., and A. I. Rubin, *Optimization by Random Search on the Analog Computer,* IRE Trans. PGEC, June, 1959.
3. Wait, J. V., and B. A. Mitchell, *A Simple Solid-State Digital-to-Analog Converter for Hybrid Systems,* ACL Memorandum No. 61, University of Arizona, 1963.
4. Mitchell, B. A., *A Hybrid Analog-Digital One Parameter Optimizer,* Ann. AICA, January, 1964.
5. Karnopp, R., *Ph.D. Thesis,* Massachusetts Inst. of Technology, 1963.
6. Brooks, S. H., *A Comparison of Maximum-seeking Methods,* Operations Research, July-August, 1959.
7. Hampton, R., G. A. Korn, and B. A. Mitchell, *Hybrid Analog-Digital Random-Noise Generation,* IEEE Trans. PGEC, August, 1963.
8. Witsenhausen, H. S., *Hybrid Techniques Applied to Optimization Problems,* Proc. SJCC, 1962 (reprinted in *SIMULATION*, Fall 1963).
9. Howell, M., *Automatic Parameter Optimization as Applied to Transducer* Design, Proc. SJCC, 1963.
10. Hampton, R., *Hybrid Analog-digital Pseudo-random Noise Generator,* Proc. SJCC, 1964 (reprinted in *SIMULATION*, April 1965).



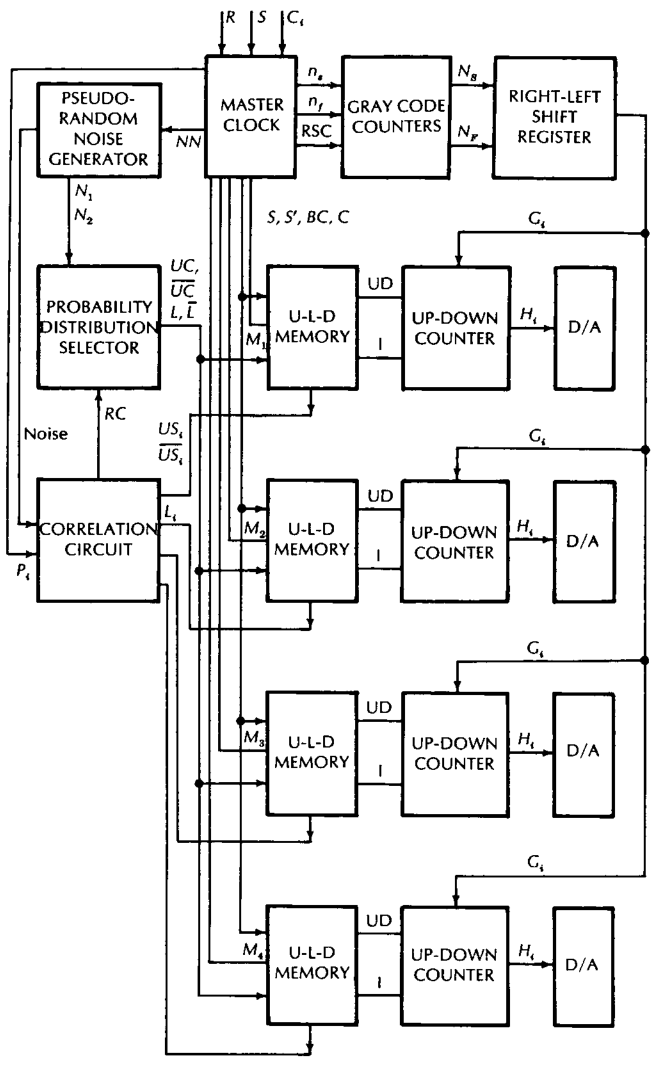
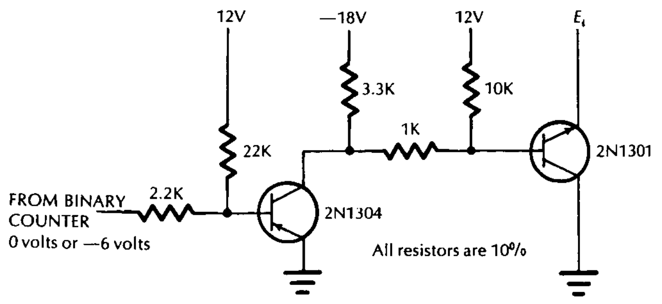


Figure 18 – Optimizer Logic Block Diagram

Figure 17 – Binary Counter and Shift Register

←



and Switch

<u>ASTRAC II DEMONSTRATION</u>

(Simulation Council Meeting, Tucson, May 1965)

The ASTRAC II demonstration problem is shown in figure 1. It is interesting to note that the CRT display of the correlator output curve (which looks like a continuous curve of the kind that might be generated by a repetitive rep-op computer solution of a single set of equations with constant coefficients) is, in fact, a plot of 2407 individual computations of the impulse response cross correlation, each with a different τ, displayed at the rate of one set per second. How's that for speed?

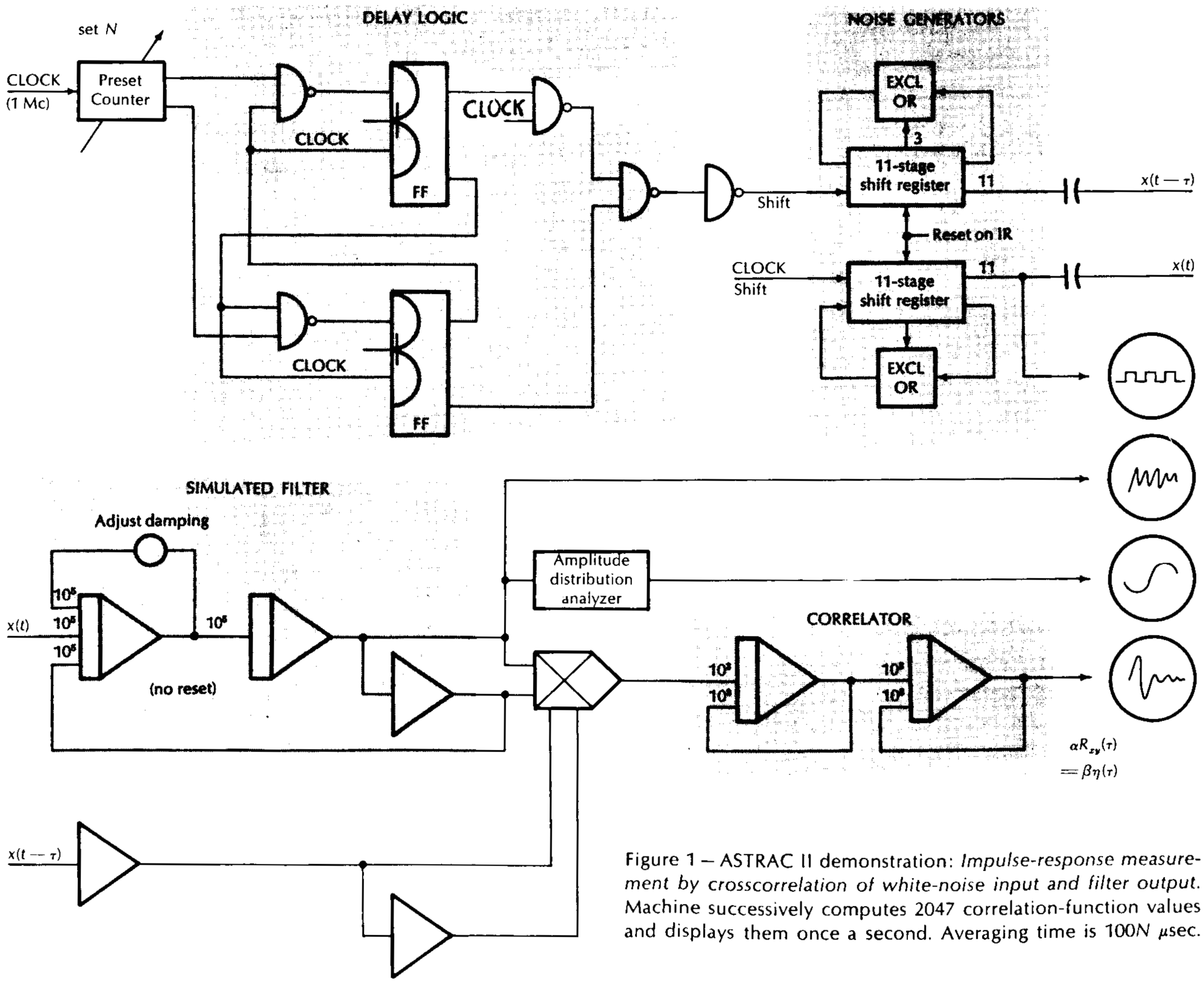


Figure 1 — ASTRAC II demonstration: *Impulse-response measurement by crosscorrelation of white-noise input and filter output.* Machine successively computes 2047 correlation-function values and displays them once a second. Averaging time is 100*N* μsec.