A HYBRID ANALOG-DIGITAL PARAMETER OPTIMIZER FOR ASTRAC II

by

Baker Adams Mitchell, Jr.

A Thesis Submitted to the Faculty of the
DEPARTMENT OF ELECTRICAL ENGINEERING
In Partial Fulfillment of the Requirements
For the Degree of

MASTER OF SCIENCE

In the Graduate College
THE UNIVERSITY OF ARIZONA

1 9 6 4

STATEMENT BY AUTHOR

This thesis has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: _____

Baker D. Mitchell

APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:

G. A. Korn

G. A. KORN

Professor of Electrical Engineering

4/9/64

Date

ACKNOWLEDGEMENT

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.

TABLE OF CONTENTS

	Page
LIST OF ILLUSTRATIONS	vi
LIST OF TABLES	vii
ABSTRACT	viii
INTRODUCTION	1
Motivation For Random Search	6
PRINCIPLES OF OPERATION	10
THE ASTRAC II SYSTEM	12
Optimizer Logic	14
Hybrid Analog-digital Noise Generator	14
Success-Failure Indicator	15
Master Clock	17
PARAMETER SETTING	19
Binary-counter Operation	19
U-L-D Digital Logic	19
U-L-D Memory	23
Correlation Circuit	25
OPTIMIZATION STRATEGIES	27
Pure Random Perturbations	27
Random Walk with Reflecting Barriers	27
Random Walk with Varied Step Size	28
Correlated or Biased Random Walk	28
TESTS	30
Functions Optimized	30
Strategies Employed	32
Results	32
Stopping Conditions	36
Conclusions	39

TABLE OF CONTENTS--Continued

	Page
LIST OF REFERENCES	40
APPENDIX I	41

LIST OF ILLUSTRATIONS

Figure	Page
1. Optimizer Block Diagram	3
2. Typical Optimization Paths	4
3. Example	7
4. Example	9
5. Flow Diagram For a Typical Optimization Routine	11
6. Photograph of Optimizer	13
7. Success-Failure Indicator	16
8. Pulse Sequence From Master Clock	18
9. Parameter Setting	20
10a. U-L-D Selector Circuit	22
10b. U-L-D Memory Circuit	24
11. Correlation Circuit	26
12a. Ridge	31
12b. Ridge Simulation	33
12c. Optimization Paths on Ridge	34
13. Ridge Optimization	37
14. Ridge Optimization	38
15. Gray Code Counter Circuit	42
16. D/A Multiplier and Switch	43
17. Binary Counter and Shift Register	44
18. Digital Logic Block Diagram	45

LIST OF TABLES

Table	Page
I U-L-D Probability Distributions	21

ABSTRACT

A new automatic multi-parameter 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

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

where

$$y(t) = y(t, \alpha_1, \dots, \alpha_n) \quad (2)$$

$$u(t) = u(t, \alpha_1, \dots, \alpha_n) \quad (3)$$

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

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

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 $F(\alpha_1, \dots, \alpha_n)$ 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 perturbations² on all parameters α_k and step to the perturbed point whenever the perturbation yields an improvement in the performance measure $F(\alpha_1, \dots, \alpha_n)$.

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 converters³ for simplified design and improved reliability.⁴ The principle of the optimizer is shown in the block diagram of Fig. 1.

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

$${}^r\alpha_i = {}^{r-1}\alpha_i + {}^r\Delta\alpha_i \quad (5)$$

which successively improve ${}^rF = F({}^r\alpha_1, \dots, {}^r\alpha_n)$. The most frequently used method employs n trial steps to compute approximate gradient components $\Delta F/\Delta\alpha$ in each parameter direction; these gradient components are then stored and used to compute the optimal correction $\Delta\alpha_i$ 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 $F(\alpha_1, \dots, \alpha_n)$, they may "hang up" on ridges or in canyons of the multi-dimensional 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 1 Optimizer Block Diagram

Figure 2 Typical Optimization Paths

(a)

(b)

(c)

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 Fig. 2c. Such a scheme causes future increments $\Delta\alpha_i$ 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.

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; and if the step size is too large, this information from the preceeding 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 to such a degree that convergence to the absolute optimum is excessively time-consuming. 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.

Figure 3

Example

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

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 (Fig. 3 & 4):

Figure 4 Example

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, $\alpha_1, \alpha_2, \alpha_3$, 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, $\alpha_1, \alpha_2, \alpha_3$, be minimum over the frequency range of interest. 5

PRINCIPLES OF OPERATION

Most of the optimization strategies proposed here can be based on the flow chart of Fig. 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, $\alpha_i + \Delta\alpha$, 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 generator⁷ 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 5 Flow Diagram for a Typical Optimization Routine

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 1Kc. 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 conventionally 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.

Figure 6 Photograph of Optimizer

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 (Fig. 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 will employ a new noise generator¹⁰ 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 (Fig. 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, (e.g., $r_F < L_F$ 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 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 Fig. 8. S' , as shown, occurs only after failures, i.e., $S = 1$. If $S = 0$, $S' = 1$ throughout the RESET period. A success or failure also causes n_S or n_F to occur (Fig. 15).

Figure 8 Pulse Sequence From Master Clock

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.

Binary-counter Operation

Referring to Fig. 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 Δ 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-DOWN-LOCK) Logic is composed of two sections: a central U-L-D Selector Circuit (Fig. 10a), and a U-L-D Memory associated with the binary counter for each parameter (Fig. 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 10a U-L-D Selector Circuit

The gate outputs, U_0 , L_0 , D_0 , determine the states of the dc 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 dc 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 $\Delta\alpha_k$, 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 $S^i = 1$. 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".

Figure 10b U-L-D Memory Circuit

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 Fig. 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 $\Delta\alpha_k$ is positive, the RC line changes the selector gating so as to increase the chance of a positive $\Delta\alpha_k$ 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, $\Delta\alpha_k$ 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.

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 preceeding 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 to signify that a parameter has reached a barrier - possibly from a comparator in the analog system - 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, e.g., 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 $P(N_1 N_2) = 1/4$, $P(N_1 \tilde{N}_2) = 1/4$, $P(\tilde{N}_1 N_2) = 1/4$, $P(\tilde{N}_1 \tilde{N}_2) = 1/4$. These can combine to give $P(X) = 1/2$, or $P(Y) = 3/4$, where $X = N_1 N_2 + N_1 \tilde{N}_2$, $Y = N_1 N_2 + N_1 \tilde{N}_2 + \tilde{N}_1 N_2$. There are three possible states for the U-L-D Memory: 1. Up; 2. Down; 3. Lock. If a failure occurred in the Up state, 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.

Also, the relatively 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

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

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 volts 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

Figure 12a Ridge

planes as shown in Fig. 12a. The simulation (Fig. 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 volts of the maximum.

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 Correlated and Biased Random Walk.)

The step size, Δ , 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

Figure 12b Ridge Simulation

Figure 12c Optimization Paths on Ridge

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.

For the case, $a = b = 1$, using strong positive correlation with successes and strong negative correlation after failures, $CC = 1$; the average trial converged in 16 runs.

The most favorable step-size variation strategy was to increase Δ after 2 successes and decrease Δ 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; Fig. 12a) to be discussed here is one expressly designed to present the greatest difficulty to the optimizer. Referring to Fig. 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 Fig. 13a (α_1 versus time, α_2 versus time); this corresponds to path A in Fig. 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 Fig. 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 Fig. 12c). This favorable possibility is not present in ordinary gradient techniques. Path A in Fig. 12c is most nearly like a path resulting from conventional gradient techniques.

Fig. 14a shows a correlated random search 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.

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.

Figure 13 Ridge Optimization

scale: horiz. 1 major division = 5 sec.
vert. 1 major division = 20 v.

(a)

(b)

Figure 14a Ridge Optimization

scale: horiz. 1 major division = 0.5 sec.
 vert. 1 major division = 20 v.

Figure 14b Ridge Optimization

scale: horiz. 1 major division = 1 sec.
 vert. 1 major division = 20 v.

Conclusions

While experience with this optimizer is still quite 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 the conventional deterministic schemes - permitting a direct comparison between random and deterministic methods optimizing the same function.

Presently, however, the random techniques seem well able to handle cases in which the performance measure is not well-behaved (Fig. 3 and 4).

REFERENCES

1. Korn, G. A., and T. M. Korn, Electronic Analog and Hybrid Computers, McGraw-Hill, N. Y., 1964 (in print).
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.
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.

APPENDIX I

The following units are contained in the optimizer digital 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. 3 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. (Fig. 15).
2. 1 8-bit right-left shift register. The DC 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. (Fig. 17b).
3. 4 8-bit up-down binary counters. The DC 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. (Fig. 17a).
4. 4 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. (Fig. 16).
5. 2 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 3×10^7 random bits. This can be divided into two noise generators, each having a 1.5×10^7 random bit sequence. The noise generator shifts out a random bit when a pulse is applied to the shift input.

Figure 15 Gray Code Counter Circuit

Figure 16 D/A Multiplier and Switch

Figure 17 Binary Counter and Shift Register

(a)

(b)

Figure 18 Optimizer Logic Block Diagram