Random Event Generator
Qualification, Calibration, and Analysis

R. D. Nelson, G. J. Bradish, Y. H. Dobyns

Princeton Engineering Anomalies Research

School of Engineering/Applied Science
Princeton University, Princeton, NJ 08544

Technical Note PEAR 89001
April 1989

ABSTRACT

Statistical anomalies in the interaction of human operators with microelectronic random event generators (REGs) can be credibly established only after the equipment, protocols, and analytical methods have been demonstrated to be free of any consequential irregularities, biases, or artifacts. Extensive qualification and calibration procedures, designed to provide a complete background of information about machine performance in the absence of operator interactions, are an integral part of the Princeton Engineering Anomalies Research program. Initial qualification tests ensure that the REGs perform as specified by the design criteria and are insensitive to environmental factors such as ambient temperature and electromagnetic fields. Ongoing calibration tests, using samples of the size generated in the formal experiments, confirm that the undisturbed REG outputs constitute normally distributed random variates with parameters indistinguishable from theoretical predictions of the Gaussian approximations to the appropriate binomial combinatorials. Various features of the standard operating and contingency protocols further preclude any unforeseen biases or intermittent aberrations. A battery of statistical tests, including time-series analyses, goodness-of-fit tests, Fourier power spectra, and arcsine comparisons, show no evidence of non-randomicity at any level of concatenation of the calibration data. The same statistical tests applied to the active experimental data are thus appropriate for identifying anomalies that may be attributed to operator-machine interactions.

Table of Contents

List of Figures

List of Tables

I. INTRODUCTION

Research on anomalies in human-machine interactions involves numerous physical parameters that need to be controlled in the experimental design. Given the controversial nature of the claimed phenomena, their profound pragmatic implications, and the low signal to noise ratio that is typical of such experiments, exceptional care is required to guarantee that any observed anomalies cannot be attributed to environmental or statistical artifacts in these parameters. Experimental control begins with carefully constructed and qualified equipment that reliably produces appropriately random calibration data. Experimental procedures and protocols must then preclude spurious influences and possible misinterpretations of the data, and provide for multiple replications over long periods of time. The purpose of this report is to provide a detailed account of the design, qualification, and calibration procedures for one of the most basic classes of experiments conducted at the Princeton Engineering Anomalies Research laboratory -- those involving microelectronic Random Event Generators (REGs), which have demonstrated, over nearly a decade of operation, an extensive and persuasive body of correlations between machine performance and operator intention.

II. EXPERIMENTAL DESIGN

A. Equipment

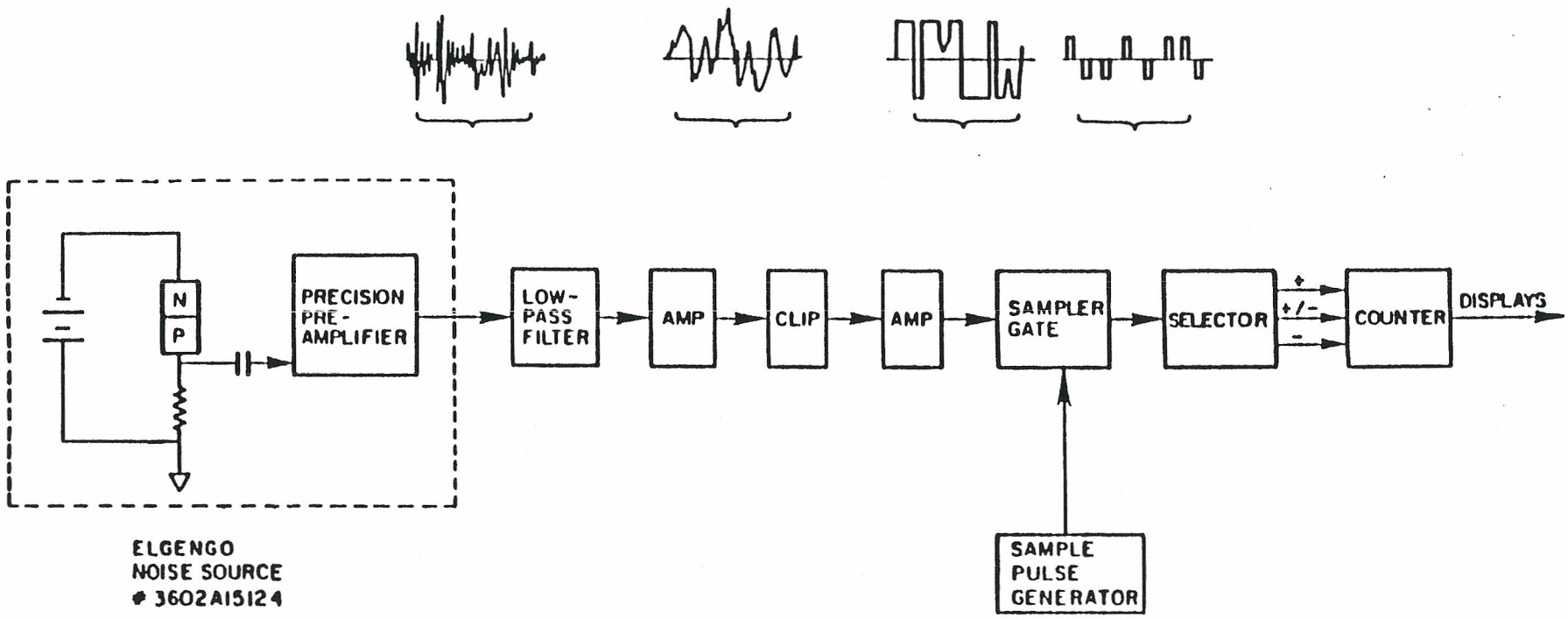
The benchmark REG currently in use is a third generation model of the same basic design. Following substantial studies with earlier breadboard and prototype machines, the present sophisticated version was designed to incorporate previous operational experience and to attend to a variety of critical concerns. The specifications were developed in consultation with members of the Princeton University Department of Electrical Engineering, and implemented in the Mechanical and Aerospace Engineering microelectronics laboratory using com-

ponents of the highest technical quality. Detailed descriptions of all three REG machines are presented in several technical reports,[1-3] and complete circuit diagrams are available for examination by interested investigators. Briefly, the primary random source is a commercial noise unit (Elgenco, Inc., Model 3602A-15124), that is based on a reverse-biased solid state junction. The fundamental random process is thermal electron penetration through this junction, generating high frequency white noise that is conditioned by subsequent electronics into a random sequence of positive and negative pulses, as sketched in Figure 1. The binary events in this sequence are sampled and counted according to preset experimental parameters. For example, the machine offers various sampling rates, and can be set to count only positive pulses, only negative pulses, or only those pulses that match the alternating sequence, +, –, +, –, etc. This alternating counting mode, which cancels to first order any intrinsic electronic bias, is used in all formal experiments. Figure 2 shows the front of the contemporary benchmark machine, displaying the count for a single 200-sample trial (middle LED), the cumulative average count of the foregoing run of 50 such trials (top LED), and the current trial number (bottom LED). Table I lists the range of sampling parameters that may be explored, with the most common parameters indicated by bold print.

Table I

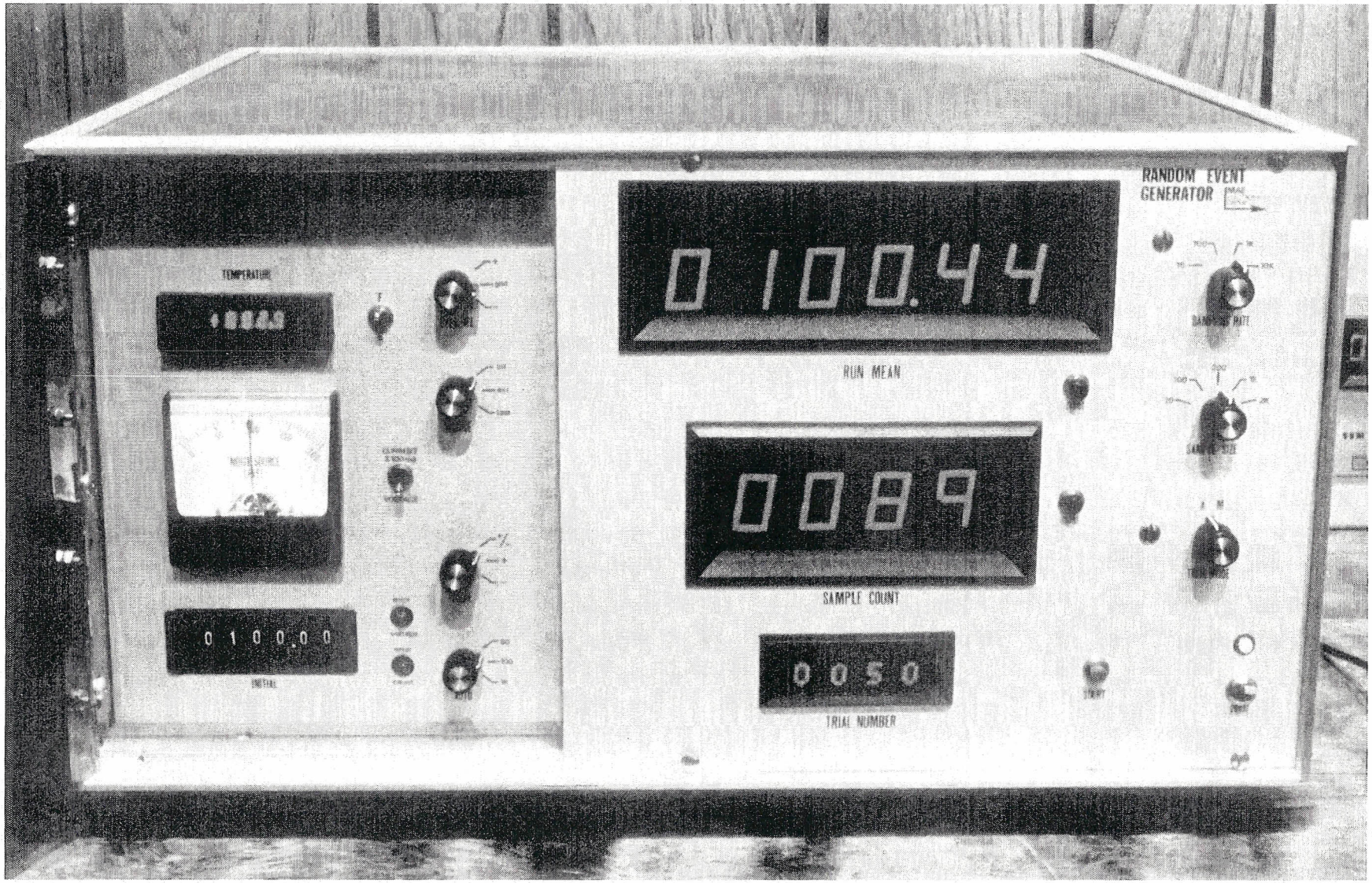
REG Front Panel Settings

Sample size	20	100	**200**	1000	2000
Sampling rate	10	100	**1000**	10000	
Trials per run	**50**	100	1000		
Initiation mode	**Automatic**			Manual	
Counting mode	+		–		+/–



Functional Schematic of REG

Figure 1



Random Event Generator

Figure 2

For this report, the typical combination employing 50-trial runs, with 200 binaries per trial, sampled in the alternating mode at 1000 per second, will be addressed, but similar qualification and calibration tests are made for all combinations employed in the experiments.

A number of electrical and mechanical guards and failsafes protect the integrity of the machine operation. For example, the initiation switch used by the operator is interlocked with the trial and run counters, and uses a monostable multivibrator to prevent any electrical contact other than the single button press that starts the intended trial or run sequence. An isolation transformer provides protection against line voltage surges, and voltage and current flow are monitored by a failsafe circuit that shuts off the machine if design values are not met. Another failsafe circuit is activated if the specified number of samples is not counted, as might occur, for example, in the rare case where a gated sample pulse occurs precisely at a zero crossing and integrates to zero. Although machine performance is robust over a wide range of internal and environmental temperatures, primary component temperature is monitored and maintained within a conservative window. Contingency protocols specify actions to be taken in case of technical malfunction or procedural error, thus avoiding any *ad hoc* decisions that might compromise data. While a complete record of all errors and malfunctions is maintained, these are too infrequent to allow formal correlations with experimental parameters or operator performance.

B. Experimental Protocol

A standard protocol established after extensive pilot testing has been used consistently for all formal basic REG experiments. Variants exploring different series lengths, the effect of multiple operators, operators at remote locations, and other random and pseudo-random sources all use parallel protocols and all are qualified and calibrated in essentially the same manner.

The experimental procedures embody several features to ensure data integrity during generation and analysis, and to provide multiple, cross-checked records. Data are acquired using a

computer program that controls the number and type of runs in accordance with prerecorded parameters set for the experimental series. Data recording is redundant, with on-line computer files and paper tape hardcopy created concurrently and independently, supplemented by hand recorded summary statistics in logbooks. The computer data are archived in a data base management system that provides access for a variety of analyses and creates yet another level of redundancy in data storage.

Beyond these technical safeguards, the experimental protocols are invariably "tripolar" in format: data are generated in runs of 50, 100, or 1000 trials under three interspersed, prerecorded directional intentions with all other conditions held constant. Thus, each experimental series has three independent data streams, differentiated solely by the primary parameter of intention: Baseline (BL), High (HI) and Low (LO). This design provides the ultimate control against physical artifact, since any such spurious influence would have to correlate with the operator's prestated intentions in order to produce any systematic bias in the output data.

Each series comprises a large number (5000, 3000, 2500, or 1000) of trials under each of the three intentions, and hence constitutes an independent, statistically self-sufficient experiment, performed by a single operator. The individual operator data bases are open-ended accumulations of replications of such tripolar series, wherein the numbers of trials for each run and for the full series are prespecified, as are all secondary experimental parameters, including the instructed/volitional mode of assignment for the intentions, and the manual/automatic sequencing of trials. In the instructed mode, the total number of trials and runs is set, but since the instruction is random, the relative numbers of HI and LO intention trials vary. The experimental protocol is further described and the data fully presented in tabular and graphic form in reference 3; other data summaries and discussions are available in references 4-8.

C. Statistical Design

In the absence of an operator influence, the REG is expected to produce a distribution of numbers conforming to the theoretical binomial combinatorial with mean $\mu = np$ and standard deviation $\sigma = (npq)^{1/2}$, where n is the number of binary events of equal probability $p = q = 0.5$. Given the large number of binaries accumulated in a trial, run, or series, such data may be modeled as normally distributed random variates using the Gaussian approximation to the binomial combinatorial.

For any body of experimental data, the primary test for significant deviations from chance expectation is the z-score, using theoretical mean and variance, or the Student *t*-test based on empirical estimates of those quantities:

$$z = N^{1/2}(M-\mu)/\sigma$$

$$t = N^{1/2}(M-\mu)/s$$

where N is the number of trials, M is the empirical mean, and s is the standard deviation of the mean. Since the empirical standard deviation of trial scores typically shows no significant difference from the theoretically expected value, and since the theoretical *t* distribution approximates the normal or z distribution for the large values of N involved, either test may be used for comparisons of the accumulated experimental deviations with the theoretically expected mean. Direction of effort is a specified parameter, hence a one-tailed criterion for significance is appropriate for each directional data string. Data may also be presented as differences between high and low efforts, i. e., as a combination of two independent efforts to achieve scores in the prestated directions of intention.

Beyond this formal hypothesis testing, a number of other statistical and graphical analyses are applied to the experimental data. The same tests are applied to the calibration data to certify their random character by these criteria. These are described in the following section; further general description of the analysis strategy for experimental data is given in section IV.

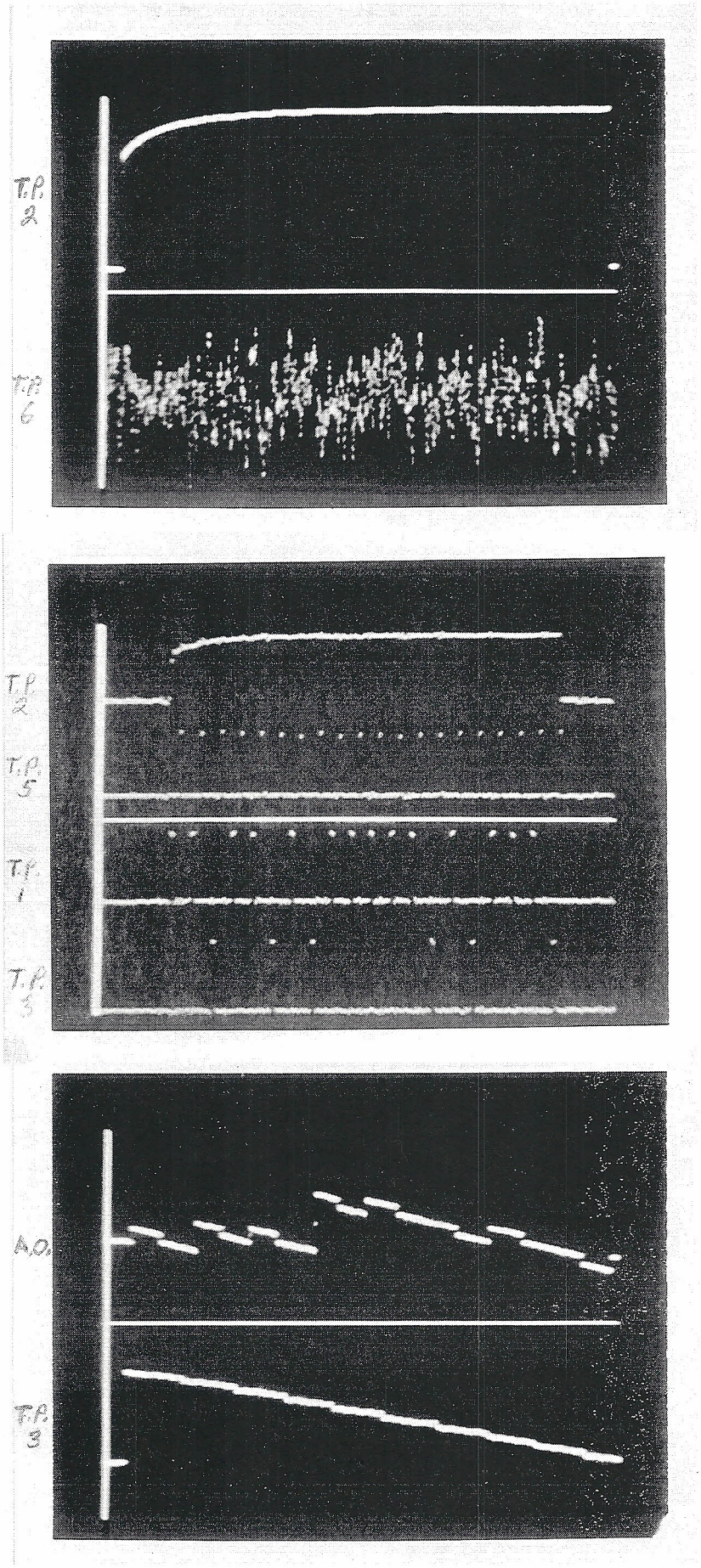
III. QUALIFICATION AND CALIBRATION

A. Qualification Tests

A sequence of qualification procedures establish the initial performance characteristics of the machines and ensure that their design specifications have been met; all subsequent modifications of the basic equipment are similarly tested:

1) Selected circuit testpoints display waveform and bias values for comparison with design specifications. Figure 3 shows samples of oscilloscope traces for the raw white noise and several of the downstream derivatives. In the upper photograph, two of the testpoints (TP 2 and TP 6) show the output of the test duration timer and the concurrent raw white noise at the diode output. In the center photograph, the duration trace (TP 2) for a 20-sample test is shown, together with the sampling signals (TP 5) and the separate trains of 14 positive and six negative pulses (TP 7 and TP 8). The bottom photograph shows the analog output voltage (A. O.) representing an accumulating run mean, with amplitude discontinuities corresponding to differences from the expected value of the individual trial scores constituting that run. Another testpoint (TP 3) shows an analog display of the progressively accumulating sample count.

2) Built in test sequences of known values are processed through the pulse and counting circuits to determine that these are functioning correctly. Independent external counters register separately counts of positive pulses, negative pulses, and their sum, to confirm the accuracy of the internal counters. These are also used to establish the primary input voltage bias setting that yields statistically equal numbers of plus and minus counts in sequences of 2×10^6 binaries. The final setting is iteratively determined by repeated runs of 10 million binaries each to achieve a grand mean that shows no bias. Although the data acquired in the +/– alternating mode cannot be affected to first order by any drift in this setting, this balance is nevertheless regularly rechecked.



Duration of test

Output of noise diode

Duration of test

Sampling signal

Positive sample pulses

Negative sample pulses

Voltage analog of run mean

Voltage analog of sample count

Oscilloscope Traces at Selected Testpoints

Figure 3

3) Various *ad hoc* tests for vulnerability to spurious environmental effects are made. For example, the effect of temperature changes near the noise source is found to be negligible within the normal operating range; in any case, this class of possible bias is also effectively eliminated by the alternating counting mode. Similarly, no effect has been found, even in plus only or minus only counting modes, of ambient electromagnetic fields or of static magnetic fields.

B. Calibration Procedures

Beyond the initial qualification tests, a battery of distribution and randomness tests, including both standard parametric statistical analyses and specialized tests, are regularly applied to calibration data:

1) Runs of 1000 trials at experiment sample sizes, but independent from the experimental data base, are characterized individually and in concatenations, using a variety of tests for deviations from expected values, goodness-of-fit, and time series characteristics. A comprehensive statistical analysis program (STAT), generates descriptive and comparative statistics including mean, maximum, minimum, counts above, below and at the expected value; standard deviation relative to theoretical and empirical means; *t*-and z-scores; standardized measures of skew and kurtosis; χ-squares with 8 and 16 degrees of freedom; and complete tables of integer and percentage counts at each possible data value. All of these tests employ canonical statistics, drawn from standard references.[9-13] The C language source for the STAT program, including all computational algorithms, is provided in Appendix A. Table II is a copy of the logbook summary of this standard analysis for a typical sequence of calibrations, and Table III shows overall statistics for an arbitrarily chosen set of 50 calibration runs.

Table II

Calibration Logbook Entries

DATE	TIME	OPR	FILE	TEMP	$\bar{x}$	S.D.	n+/n-	S.D. Ratio F	t	P_t	n-LOW	n $\bar{x}$	n-HIGH	Z^2_8	P_8	Z^2_{16}	P_{16}	SKEW 3R	KURTOSIS 4R
1/28/86	10:00	841	013	82.1	100.183	6.995	0143/9857	.988	.828	.204	463(82)	65	472(123)	10.34	.246	15.83	.468	0.523 0.412	-.539 -.561
1/28/86	10:16	841	014	83.8	99.968	7.052	1402/8598	.997	-0.143	.443	480(78)	58	462(122)	9.40	.311	14.66	.550	0.320 0.360	.647 .654
1/28/86	10:46	841	015	85.3	99.752	7.115	9887/0112	1.006	-1.102	.135	497(77)	53	450(122)	6.04	.644	20.57	.197	-0.404 0.541	.682 .692
1/29/86	16:21	870	016	88.5	99.864	7.007	1453/8647	.991	-0.614	0.270	480(76)	68	452(124)	14.13	.082	20.65	.194	0.382 0.497	0.193 0.358
1/29/86	16:40	870	017	88.5	99.961	7.029	0437/9563	.994	-0.175	.430	471(73)	56	473(120)	8.97	.349	15.70	.477	-.447 -.421	-.580 -.574
1/29/86	17/	870	018	88.1	99.997	7.030	0247/9753	.994	-0.013	.495	462(77)	63	475(120)	2.11	.977	10.79	.819	-.392 -.382	-.462 -.447
2/3/86	11/19	851	019	87.7	100.154	7.139	0449/9551	1.010	0.462	.248	474(79)	46	480(128)	3.31	.912	8.38	.937	.500 .385	.641 .642
2/3/86	14/50	851	020	88.0	99.562	7.309	9868/0132	1.034	-1.896	.029*	491(78)	49	460(122)	7.09	.528	18.14	.318	-.676 -.465	-.586 -.608
2/3/86	15/10	851	021	88.3	100.080	6.982	9855/0145	.987	0.045	.482	480(79)	59	461(124)	11.49	.180	20.59	.196	.173 .104	.683 .688
2/4/86	10/52	851	022	82.8	99.983	7.054	0364/9636	.998	-0.076	.470	481(82)	63	456(130)	4.69	.789	9.02	.911	.607 .615	.665 .672
2/4/86	11/18	851	023	85.4	99.835	6.769	0377/9623	.957	-0.771	.220	483(76)	49	468(123)	6.47	.596	14.59	.555	-.390 .174	.583 .592
2/4/86	11/35	851	024	86.9	99.904	6.962	0221/9779	.985	-.436	.331	481(79)	61	458(120)	11.14	.195	12.42	.714	.358 .447	-.704 -.702
2/4/86	11/53	851	025	88.9	100.106	7.085	0186/9814	1.002	.473	.318	474(77)	59	467(126)	16.17	.042*	27.10	.035*	.556 .502	-.602 -.605
2/4/86	12/12	851	026	90.1	99.896	7.090	0228/9772	1.003	-.464	.321	485(79)	59	456(126)	5.23	.732	18.41	.300	.399 .475	.431 .465
2/4/86	12/26	851	027	89.7	100.016	6.985	0060/9940	.988	.074	.471	474(76)	48	478(123)	4.22	.834	15.05	.521	.373 .361	.568 .575
2/5/86	10/25	810	028	74.6	100.243	7.159	0320/9680	1.012	1.073	.142	464(80)	65	471(126)	1.94	.982	8.01	.948	.595 .466	-.487 -.503
2/5/86	10/55	810	029	81.1	100.102	6.958	0344/9656	.984	0.464	.322	453(74)	51	496(121)	11.01	.202	16.99	.393	-0.513 -0.571	-.555 -.529
2/5/86	11/11	810	030	83.3	99.738	7.019	0710/9290	.993	-1.180	.119	487(73)	53	460(121)	5.16	.739	8.41	.933	-.487 -.187	.401 .420
2/5/86	14/40	851	031	88.3	100.178	7.174	0628/9372	1.015	0.785	.216	446(78)	71	483(125)	6.68	.573	11.72	.762	-.167 -.429	-.576 -.558
2/5/86	15/10	851	032	88.7	100.024	7.015	0208/9792	.992	0.108	.457	471(77)	52	477(125)	4.34	.823	14.40	.569	-.200 -.269	.499 .511
2/5/86	15/42	851	033	89.0	99.799	6.953	7800/0134	.983	-0.914	.180	459(79)	63	478(120)	8.19	.422	21.66	.161	-.584 -.498	-.382 -.415
2/5/86	16/03	851	034	89.2	99.802	7.026	0212/9788	.994	-0.891	.187	464(76)	55	481(122)	9.36	.315	20.08	.219	-.572 -.475	.303 .221
2/5/86	16/52	851	035	89.4	100.260	6.998	9970/0030	0.990	1.475	.121	466(83)	53	481(122)	4.17	.839	13.17	.660	.646 .551	-.669 -.685

Table III

REG Calibrations, Statistical Summary

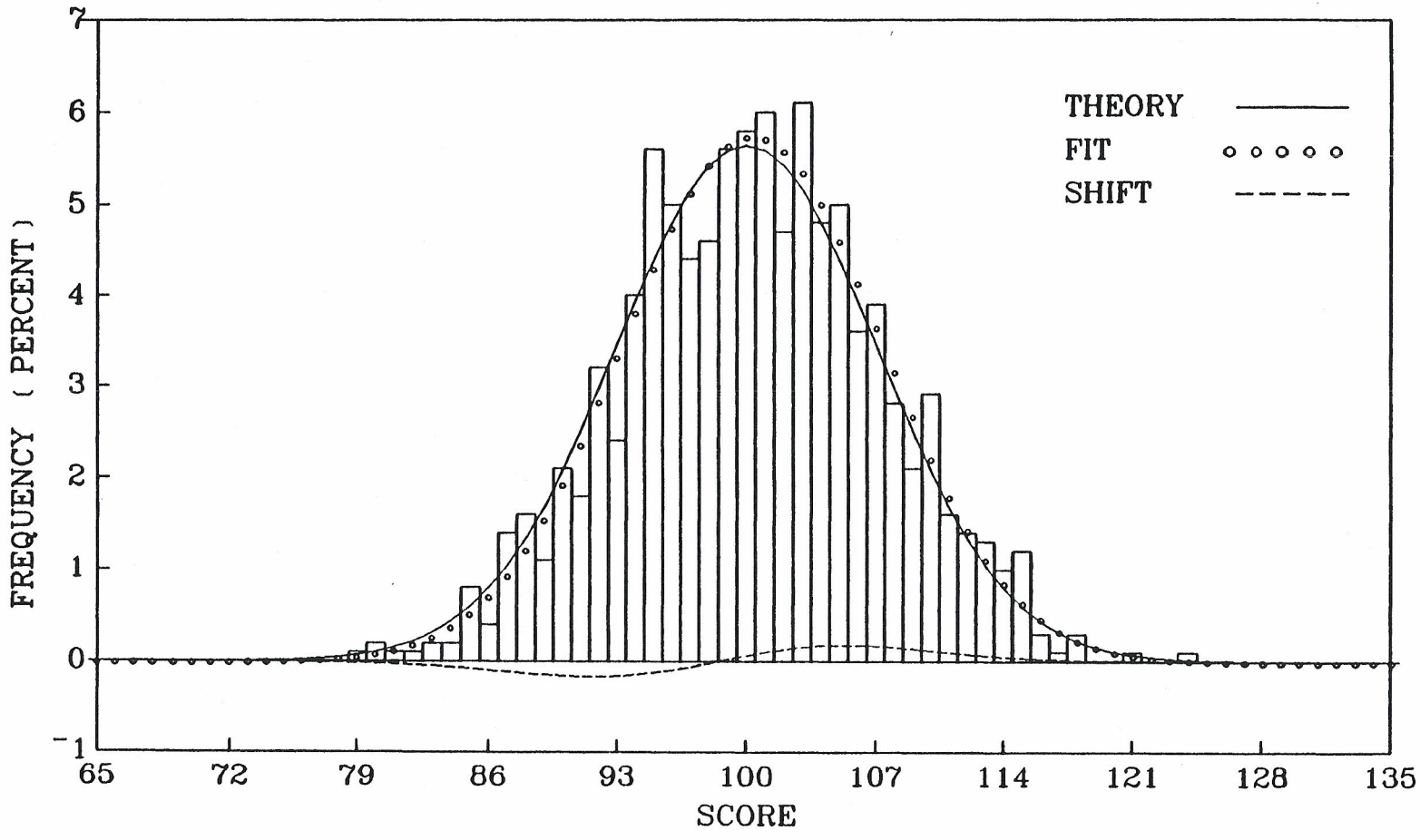
Statistic	Expected	Empirical	Standardized*
Number of trials		50000	
Bits per trial		200	
Maximum count		130	z = 4.243
Minimum count		72	z = -3.960
Mean	100	100.011	z = 0.348
Variance	50	50.039	F = 1.001
Standard deviation	7.071	7.074	p = .483
t-score	0	0.348	p = .360
z-score	0	0.348	p = .360
Skew	0	0.0105	z = 0.955
Kurtosis	3	3.004	z = 0.184
Chi-square	51	60.705	p = .173
Kolmogorov	0	.002	p = .671

*Definition of symbols: z is the standard z-score, F is the F-ratio for variance tests, p is chance probability.

2) Distributions of calibration data are graphically compared with appropriate standards. Figure 4 shows the frequency distribution of counts for a single run of 1000 calibration trials (equivalent to one intention in an experimental series) superimposed on the expected theoretical distribution, and Figure 5 shows a distribution for 50000 trials (equivalent to a concatenation across a very large operator data base). Despite their differing granularity, a χ-square test for goodness-of-fit confirms that both of these empirical frequency distributions are not significantly different from the theoretical Gaussian. Similarly, distributions of calibration data from even larger concatenations, expressed as z-scores, are found to be normal with mean and standard deviation statistically indistinguishable from zero and one, respectively.

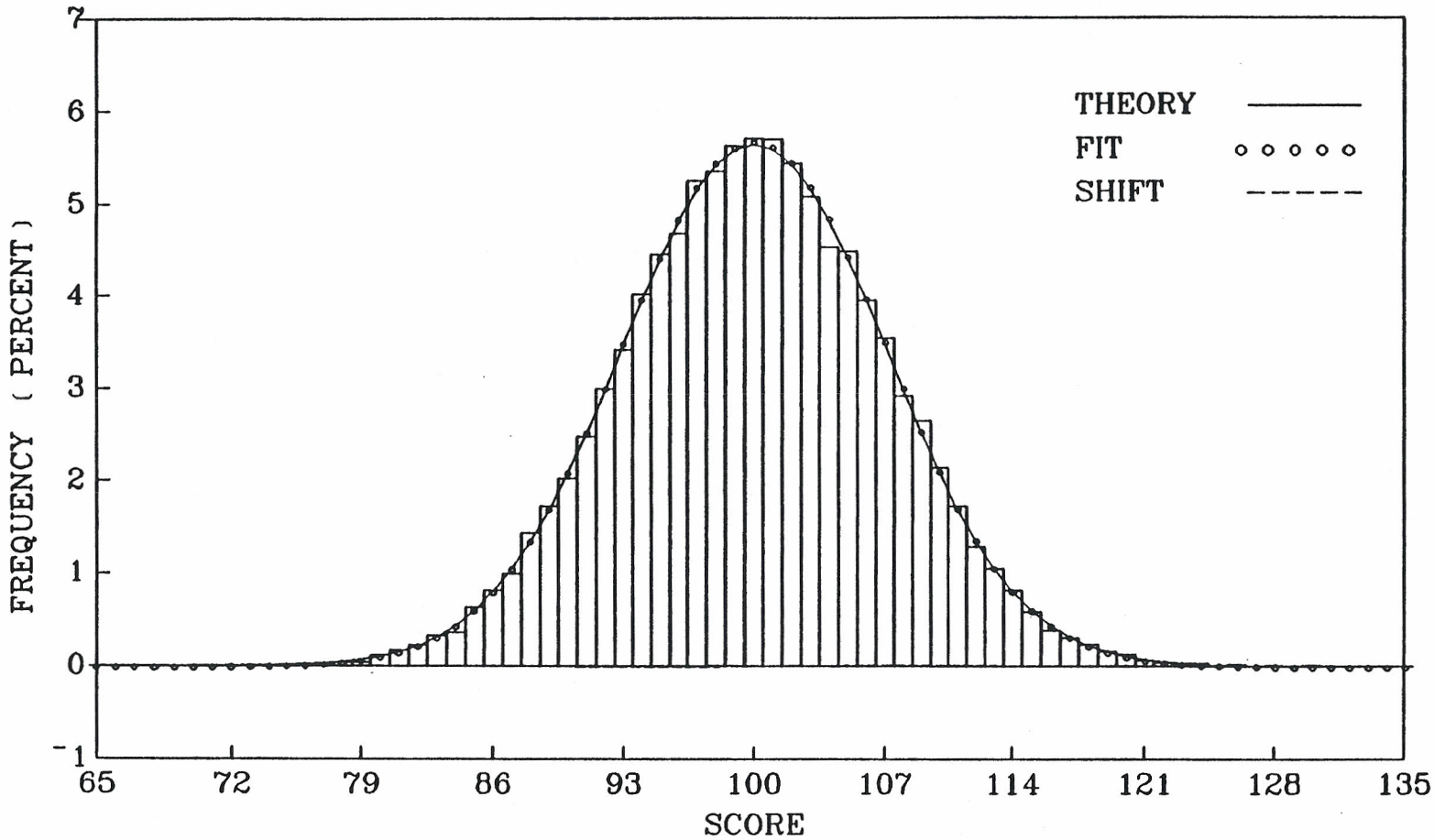
3) When these same data are arrayed as a cumulative distribution function, the resulting pattern can be compared with the theoretical cumulative distribution function, as shown in Figure 6. In this case, the Kolmogorov-Smirnov goodness-of-fit test calculates the maximum absolute distance between the curves and tests for significant positive or negative divergence.[13] Again the calibration data show no significant departures from theoretical expectations.

4) The calibration data may also be characterized as a one dimensional random walk by graphing their cumulative deviation from expected value. Such cumulative deviation of the same 50000 calibration trials is shown in Figure 7, along with an envelope indicating the loci in either direction of the one-tailed .05 chance probability for so large an absolute deviation at any prespecified number of trials. In this example, the random walk penetrates the envelope at one point, but the overall trend exhibits approximately zero slope; both features are consistent with a random process.



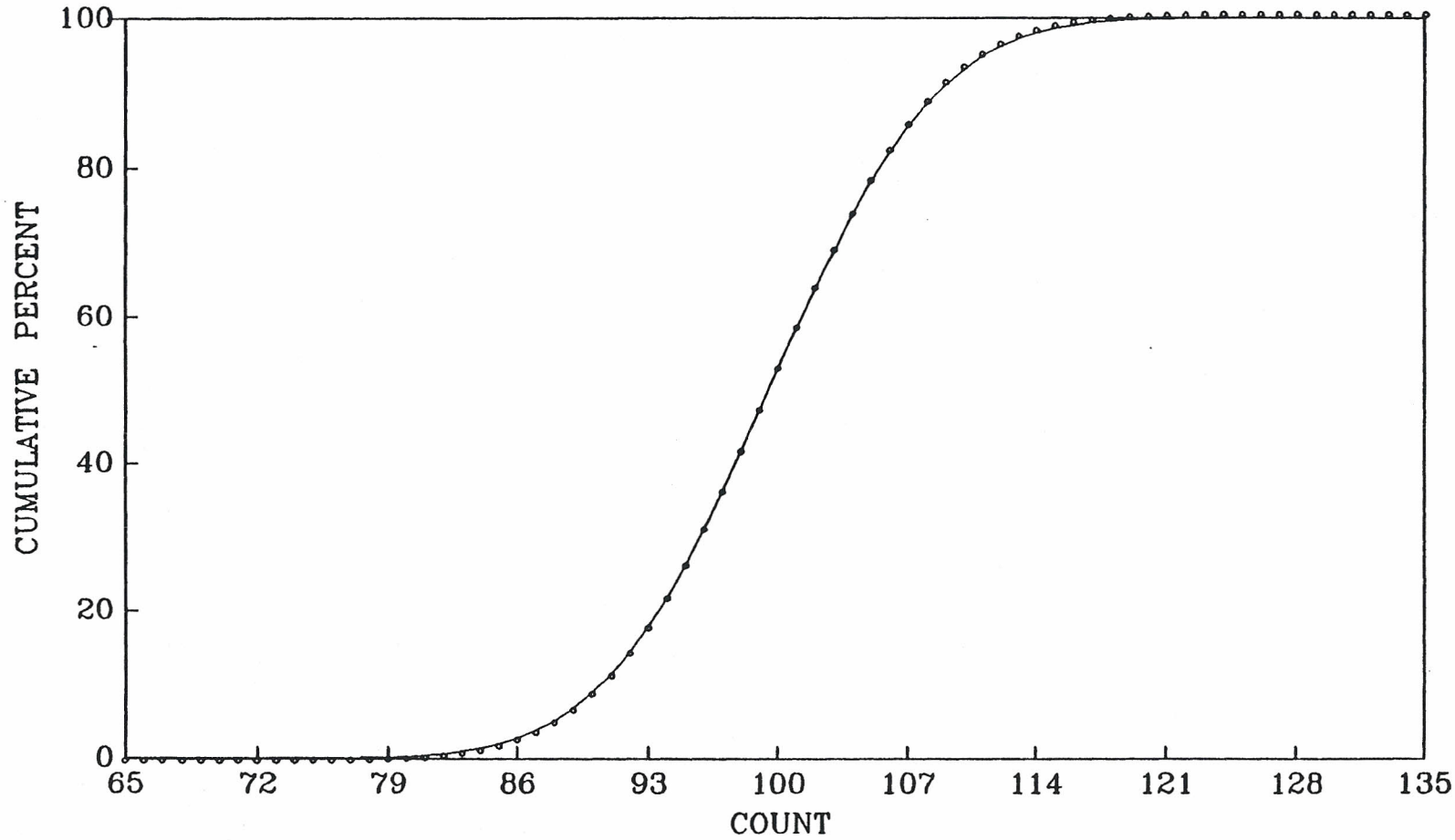
Frequency of Counts for 1000 Calibration Trials, on Theory

FIGURE 4



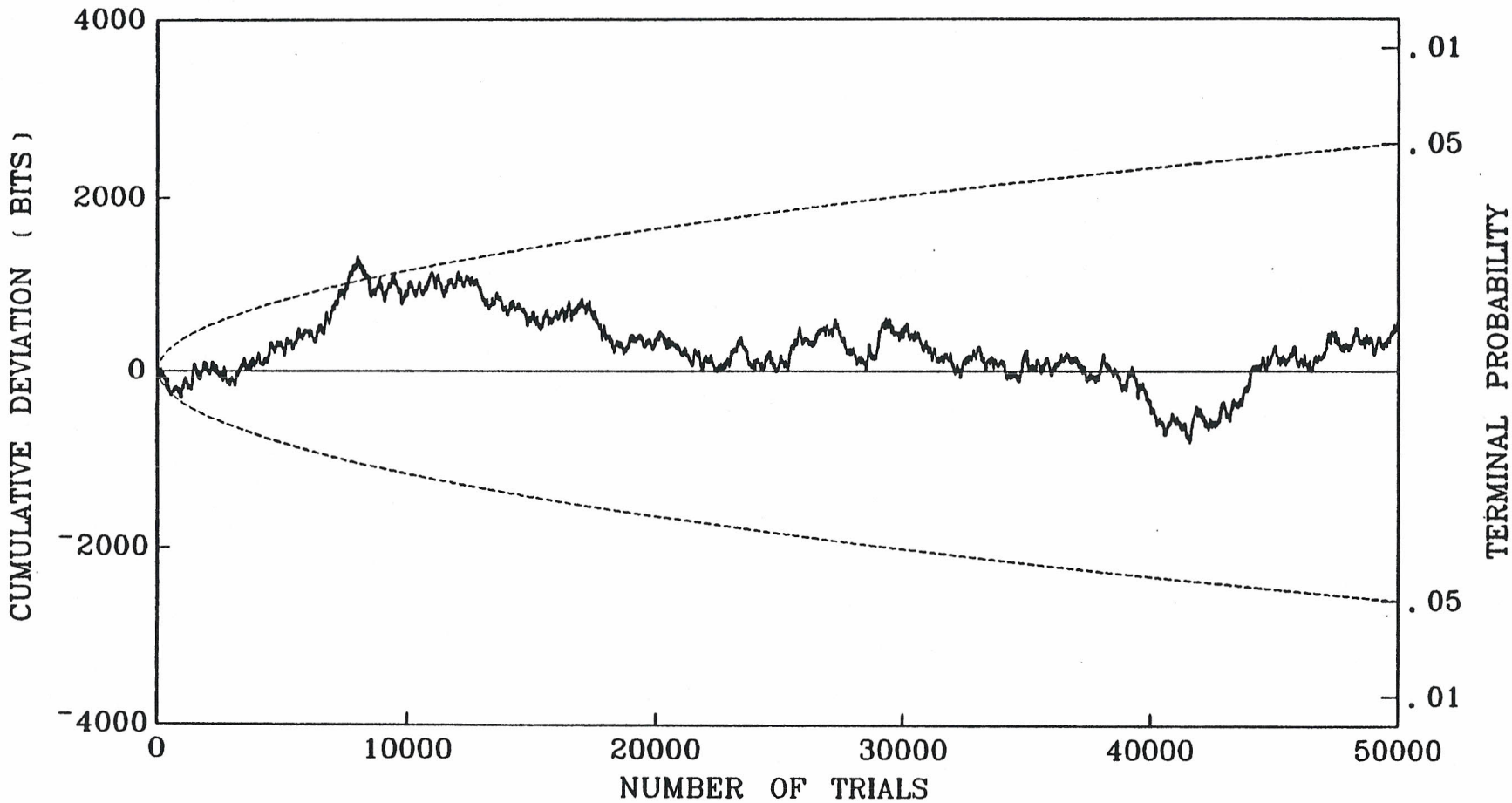
Frequency of Counts for 50000 Calibration Trials, on Theory

FIGURE 5



Cumulative Distribution Function for 50000 Calibrations, on Theory

FIGURE 6



Cumulative Deviation of 50000 Calibration Trials

FIGURE 7

5) Probability theory can be applied to estimate the maximum deviations attributable to chance anywhere in a sequence of random data. The law of the iterated logarithm,

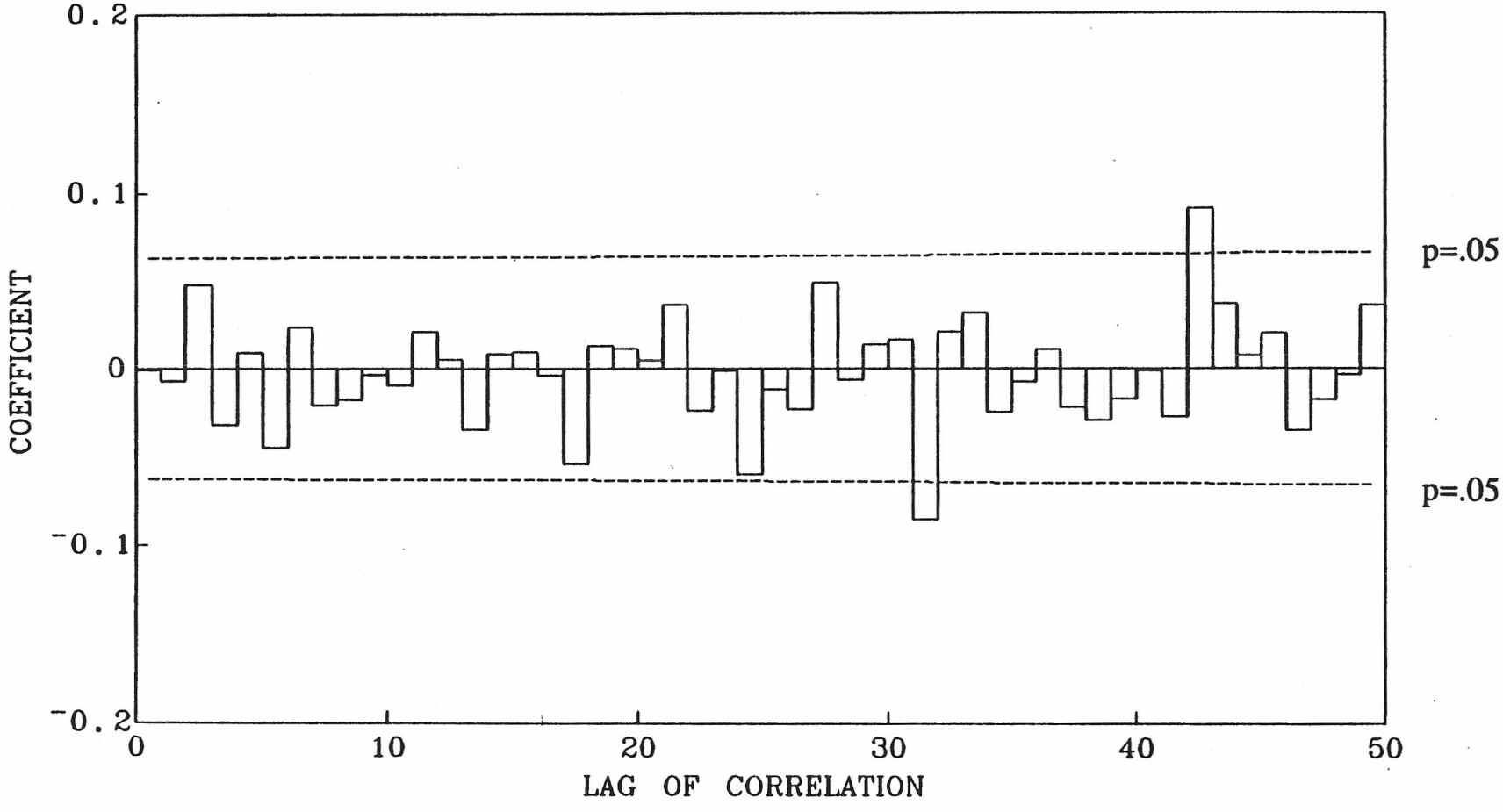
$$z_{max}(N) \equiv \{2 \ln(\ln N)\}^{1/2}$$

where N is the number of binaries, predicts increasing excursions for the z-score of a continued random walk that can be calculated by threshold crossing analysis.[14, 15] Although this computation is more appropriate to cases where "optional stopping" of the data accumulation is allowed, and hence is not directly relevant either to the calibrations or the experimental data since the number of trials is always prespecified, it nonetheless shows that for 50000 trials a single penetration of the .05 reference envelope has a probability of about 0.4 of occurring by chance at some time during the data accumulation. The terminal z-score for the entire prespecified sequence remains z-distributed.

6) Another standard test for randomness examines data for unusual repetition of any given value or sequence of scores (often called a "runs" test), for comparison against appropriate theoretical probabilities.[13] These tests show no unusual proportion of repeating sequences in the REG data.

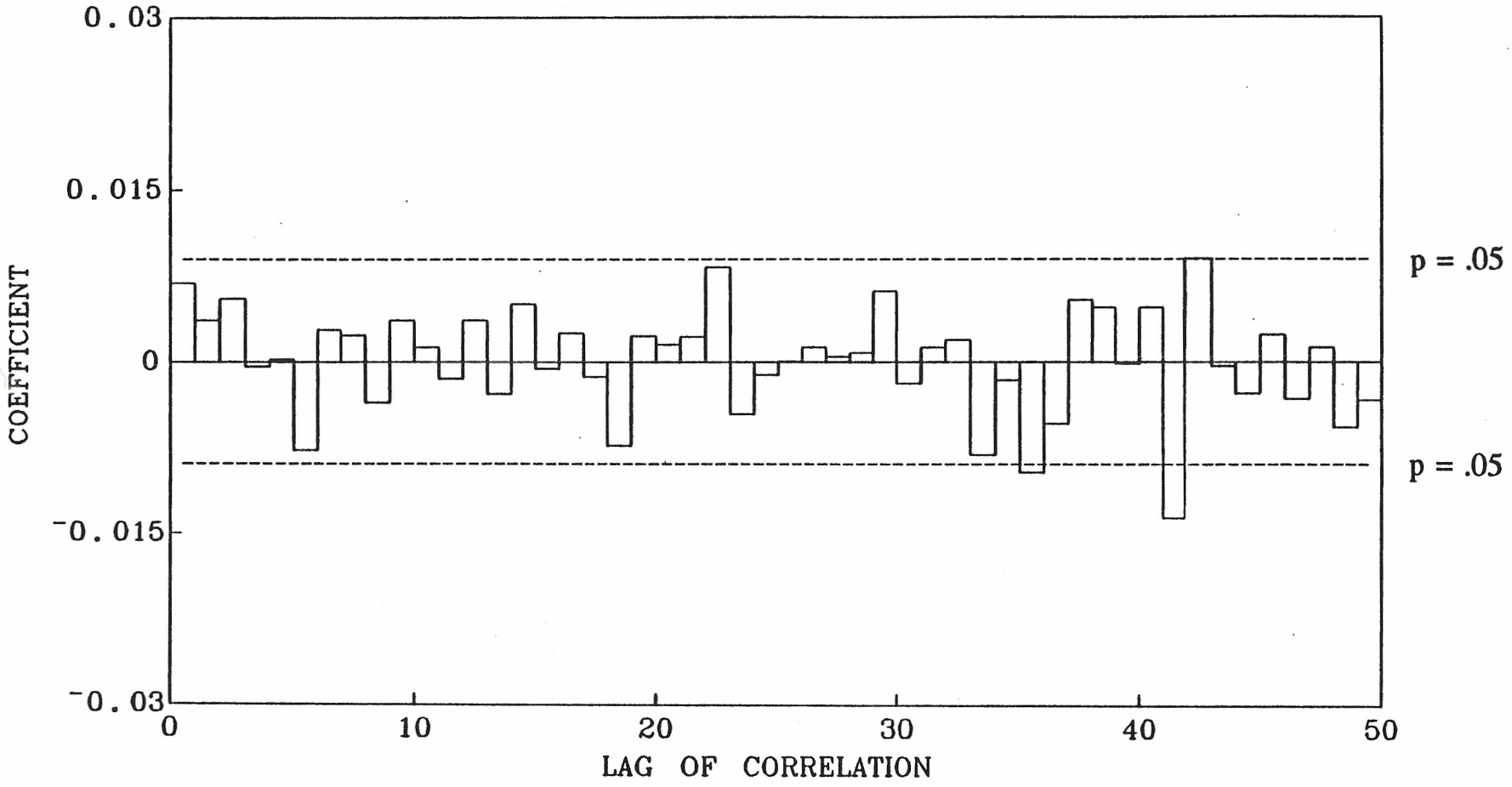
7) Time-series autocorrelation tests show only chance levels of correlation for trials with their near or distant neighbors, and there is no replication of autocorrelation patterns in the calibration data. Figure 8 displays a typical autocorrelation function with 95% confidence bounds for a single 1000 trial sequence and Figure 9 similarly displays the autocorrelations in a set of 50000 trials. In general, there are no indications beyond that expected by chance of autocorrelation at any lag up to 50 in these data, and the overall mean autocorrelation function is indistinguishable from that of white noise.

8) Similarly, Fourier analyses of single or concatenated calibration data sets display no trends or spikes at any spectral value. Figure 10 shows the raw Fourier spectrum amplitudes for a typical set of 1000 trials, and Figure 11 presents the integrated periodogram



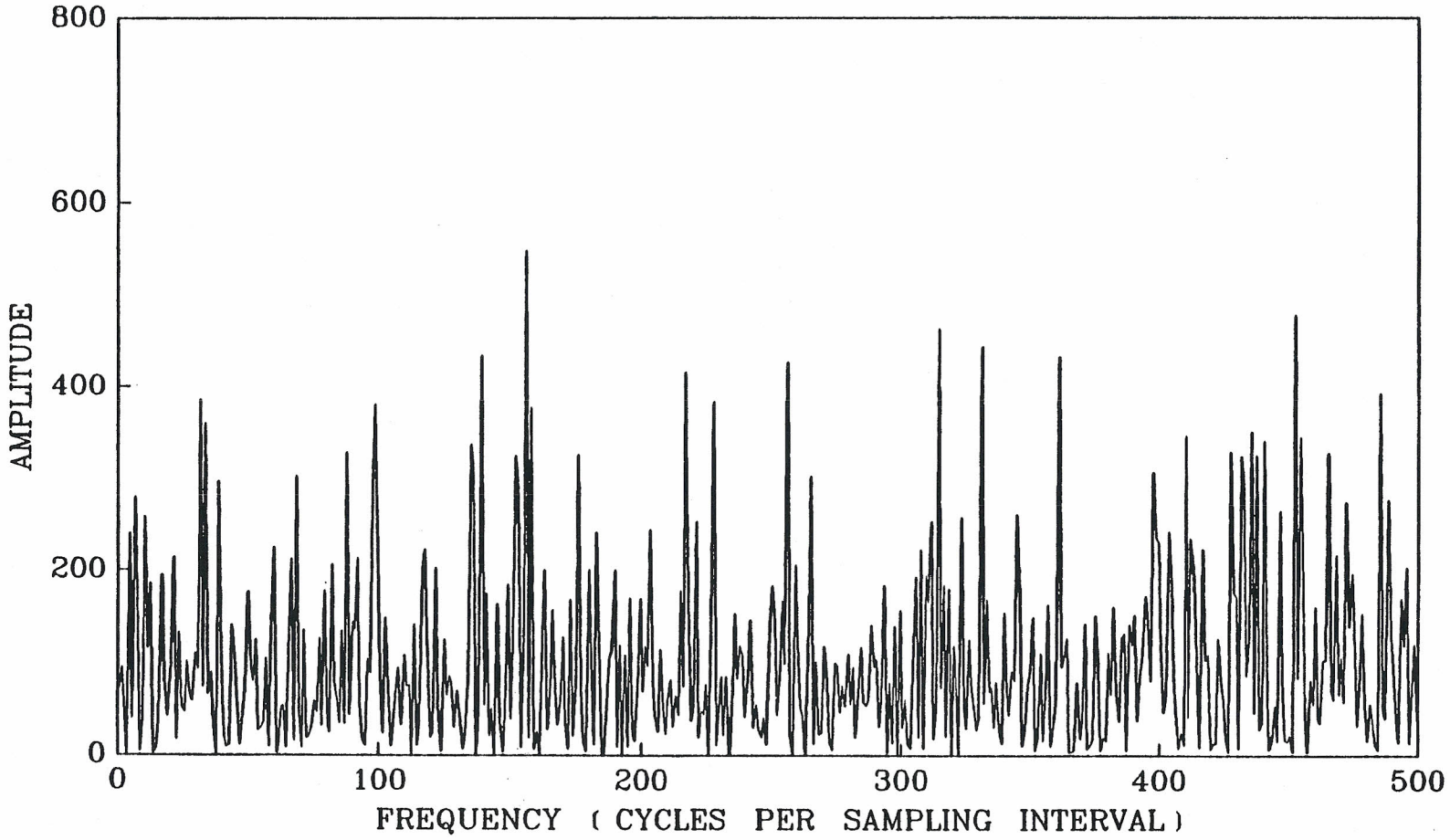
Autocorrelation Function for 1000 Calibration Trials

FIGURE 8



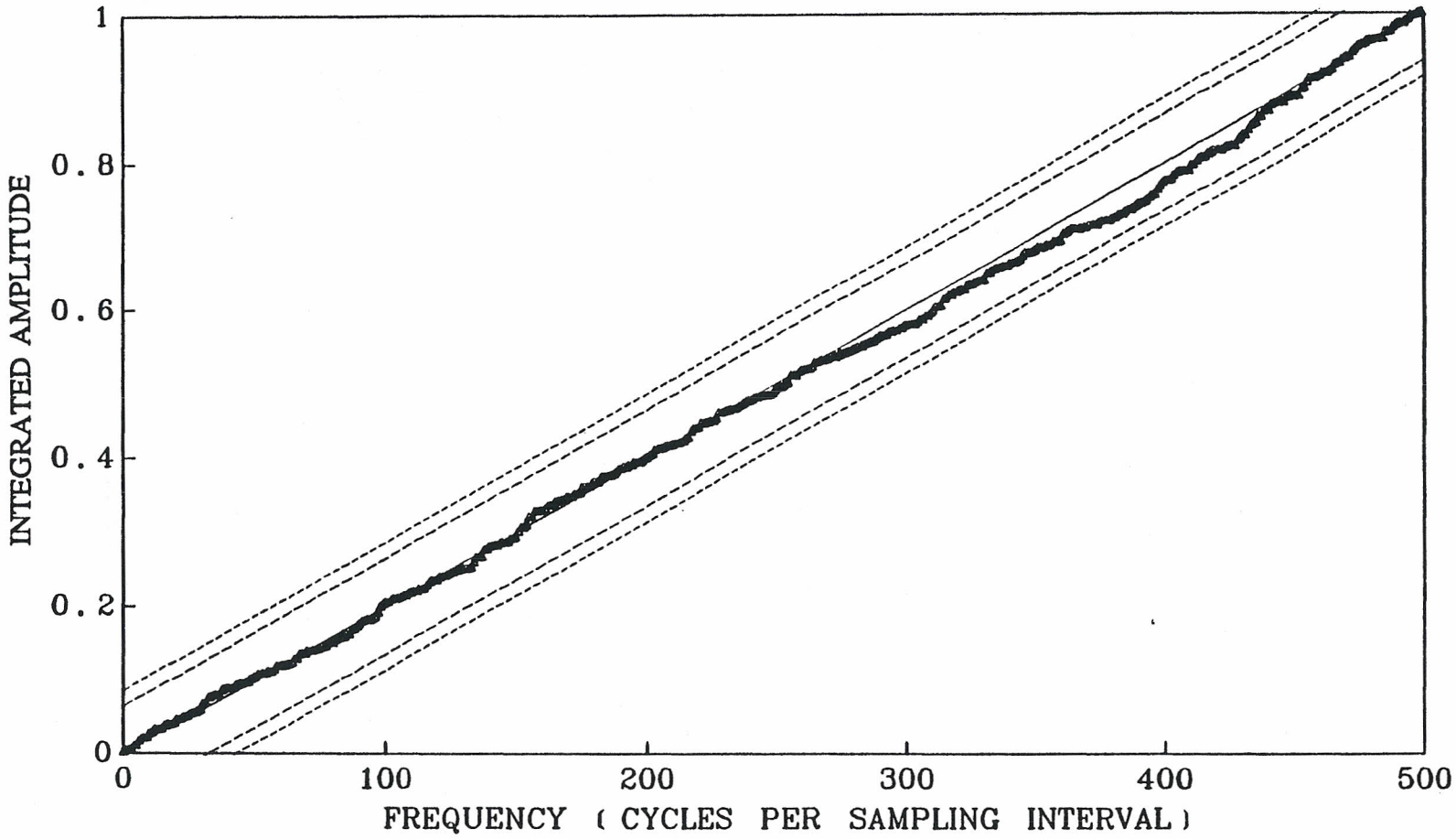
Autocorrelation Function for 50000 Calibration Trials

FIGURE 9



Fourier Spectrum Amplitudes for 1000 Calibration Trials

FIGURE 10



Integrated Periodogram for 1000 Calibration Trials

FIGURE 11

for the same data, with 75% and 95% Kolmogorov-Smirnov confidence intervals. These tests are also repeated in multiple samples, and again show no indication of periodicity in the calibration data.

9) Other useful and independent tests of the degree to which these data are representative of a normally distributed population of random variates may be drawn from further applications of probability theory.[14] For example, the proportion of time spent on one side of the origin in a random walk is predicted by the "arcsine law", and the data from calibrations should fall close to these predictions:

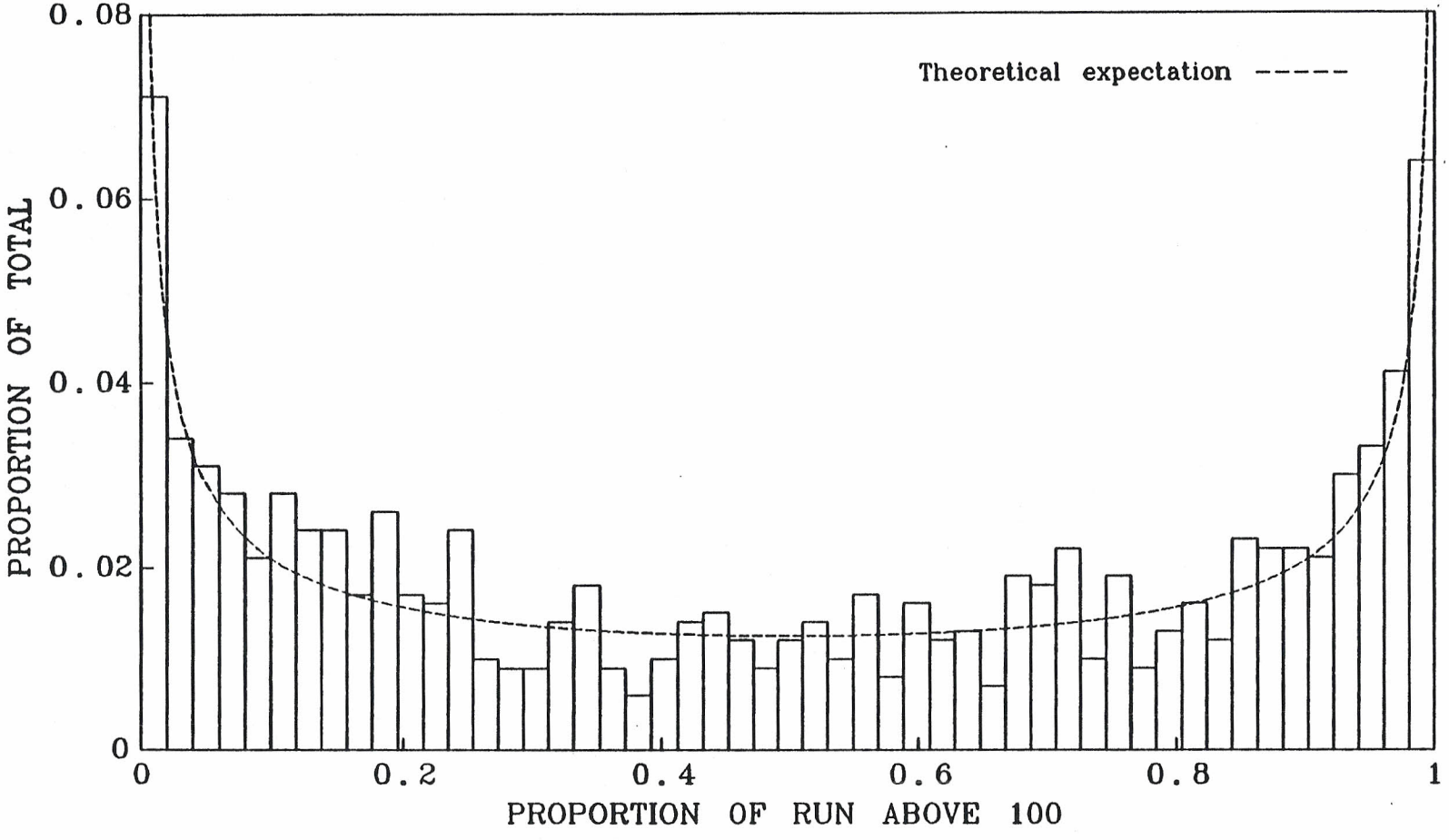
$$p(\alpha) = (1/\pi) \int_0^{\alpha} dx/\{x(1-x)\}^{1/2} = (2/\pi) \arcsin \alpha^{1/2}$$

where α is the fraction of the total trials in the series over which the cumulative deviation of the mean is positive (or negative), $p(\alpha)$ is the probability that this fraction is α or less, and x is a dummy variable of the integration. Figure 12 shows the frequency distribution for 1000 50-trial runs of the proportion of the cumulative deviation that is greater than its expected value, and bears out the counterintuitive prediction that the most probable number of zero-crossings in a random walk is zero, so that a relatively large proportion of runs remains exclusively on one side of the origin. Figure 13 shows the close match of these calibration data with the integrated theoretical arcsine curve.

10) Regression modeling may be used to examine calibrations for significant trends.[16] Analyses of full series concatenations indicate that the general linear model

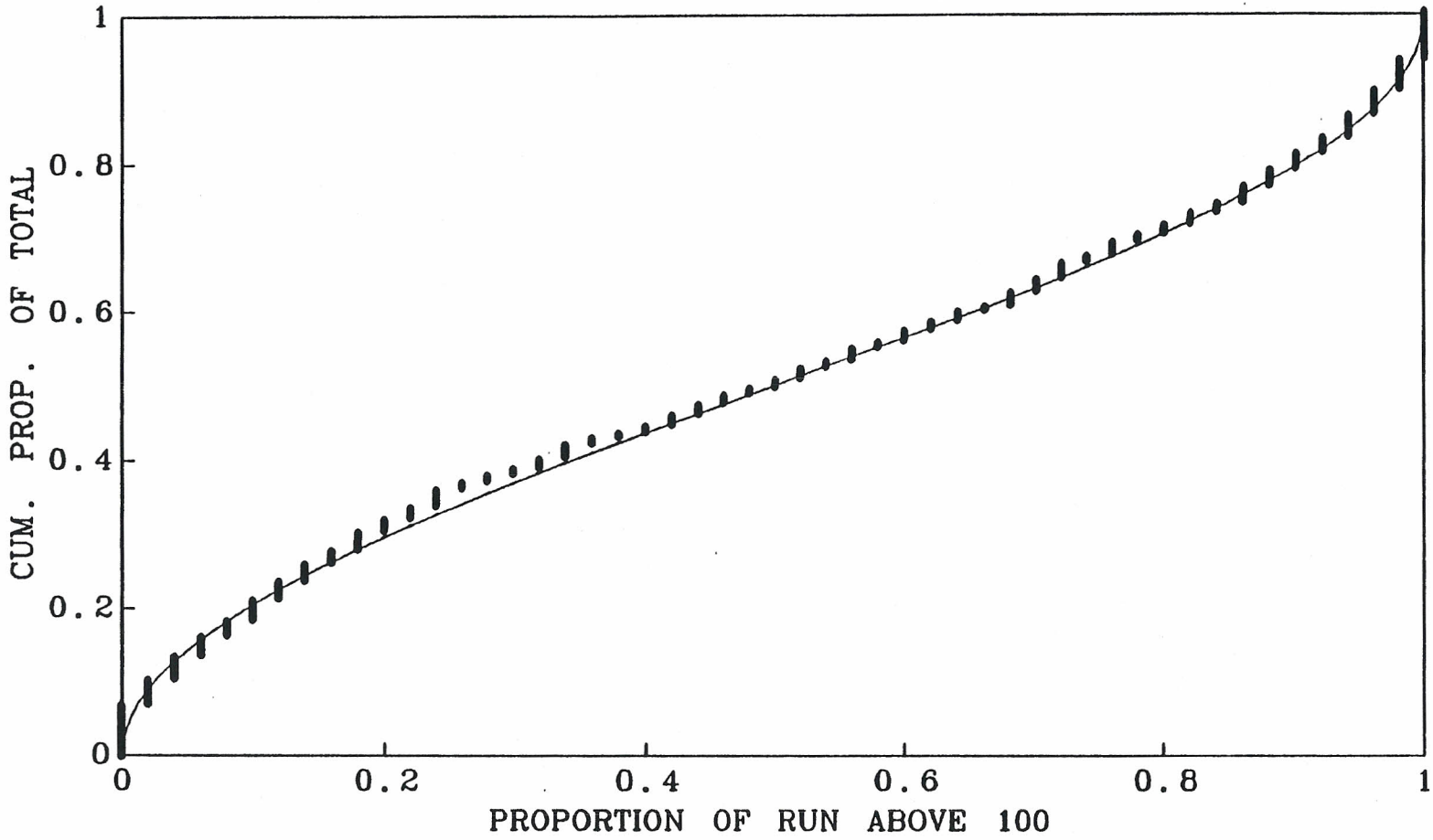
$$Y_i = \alpha X_i + \beta X_i^2 + \ldots + \varepsilon_i$$

is appropriate. In typical calibration data, both the constant term and the slope coefficient are statistically indistinguishable from the expected value of zero. In a sample of cases where they have been checked, the cubic and quartic coefficients are also statistically negligible.



Proportion of Random Walk on One Side of Origin: 1000 Runs of 50

FIGURE 12



Proportion of Random Walk Positive, 1000 Runs of 50, on Arcsine Law

FIGURE 13

C. Pseudorandom Devices

In addition to the benchmark REGs utilizing microelectronic noise diodes for their random sources, the laboratory has employed a number of pseudorandom devices in parallel experimental programs. Some of these are hardwired from microelectronic shift registers; others are programmed by algorithm onto personal computers.

Calibration of these hard-wired and algorithmic pseudorandom sources proceeds in a similar fashion, with allowance for their particular characters. The contemporary hard-wired system, for example, has a test mode in which the first N elements of the sequence are repeatedly drawn; they should, and do, always produce the same result for a given sampling rate and sample size. The system is otherwise calibrated in the manner described above, with results indicating an appropriately random source for use in the formal experiments.

The present algorithmic source uses the Borland Turbo-Basic RND and RANDOMIZE functions, and is seeded from a combination of current values of the system clock and microsecond timer. The program also has a calibration mode, in which seed values are entered for each run. A data base of 200 runs of 1000 trials generated from sequential random numbers as seeds exhibits good fit to theoretical parameters, and another large body of calibrations (6000 run means representing six million trials) generated from seed numbers in the range of experimental values shows no significant departures from normal distribution parameters.

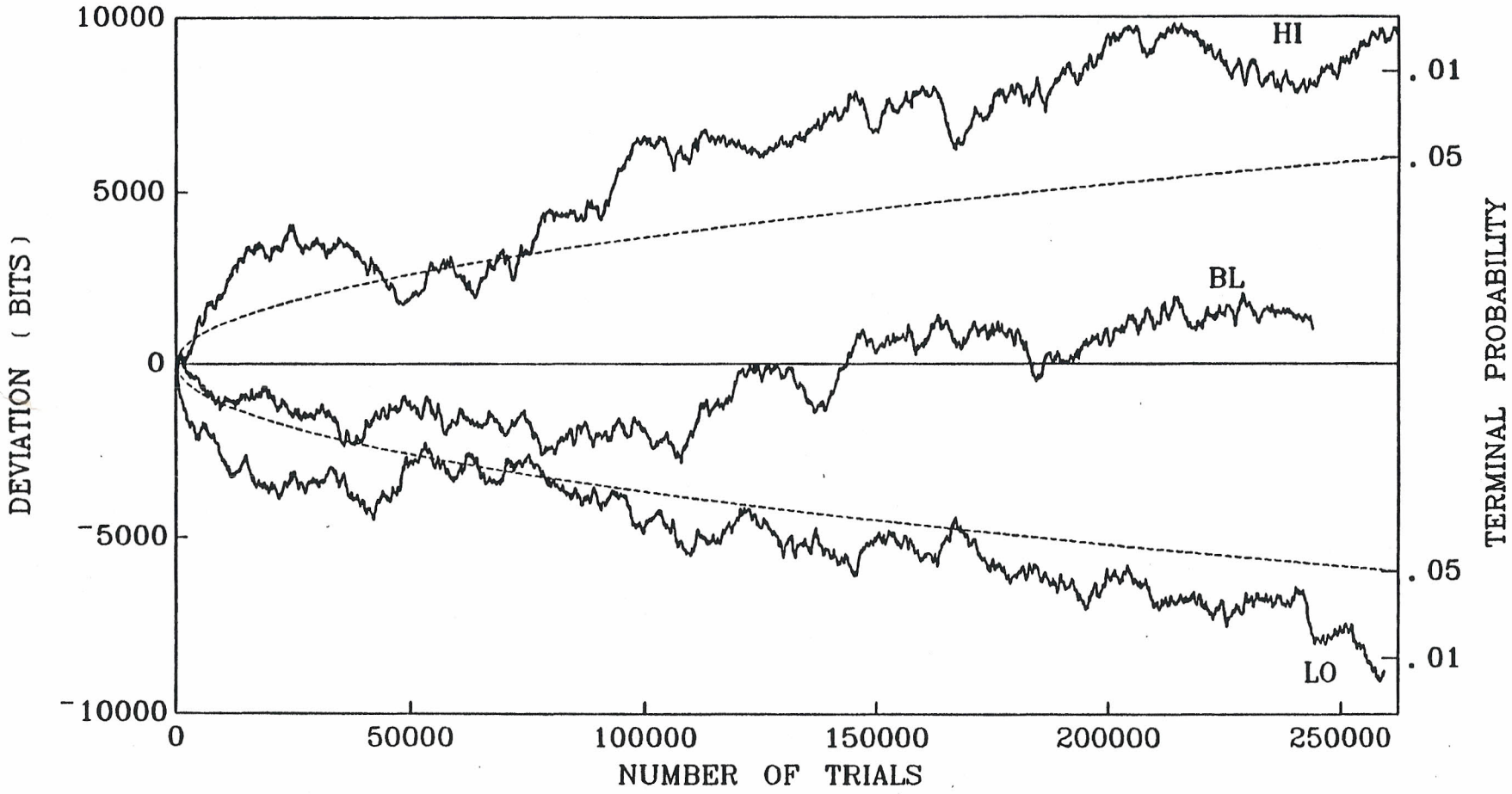
IV. DATA ANALYSIS

As described in the statistical design section, the primary assessment of experimental effect is based on simple and robust parametric analyses. Data assigned under each intention are separately compared with chance expectations in individual series and concatenations, using the appropriate z- or *t*-tests, and it is on the results of these tests that most discussions of anomalous effects are based. The preplanned standard analysis for each series includes all of

the tests in the STAT program described in Section III. For concatenations across operators and secondary parameters, a factorial analysis of variance may also be used, along with a number of standard and special-purpose analytical strategies employed for explorations, given a demonstration of effect in the primary analysis.[2-7]

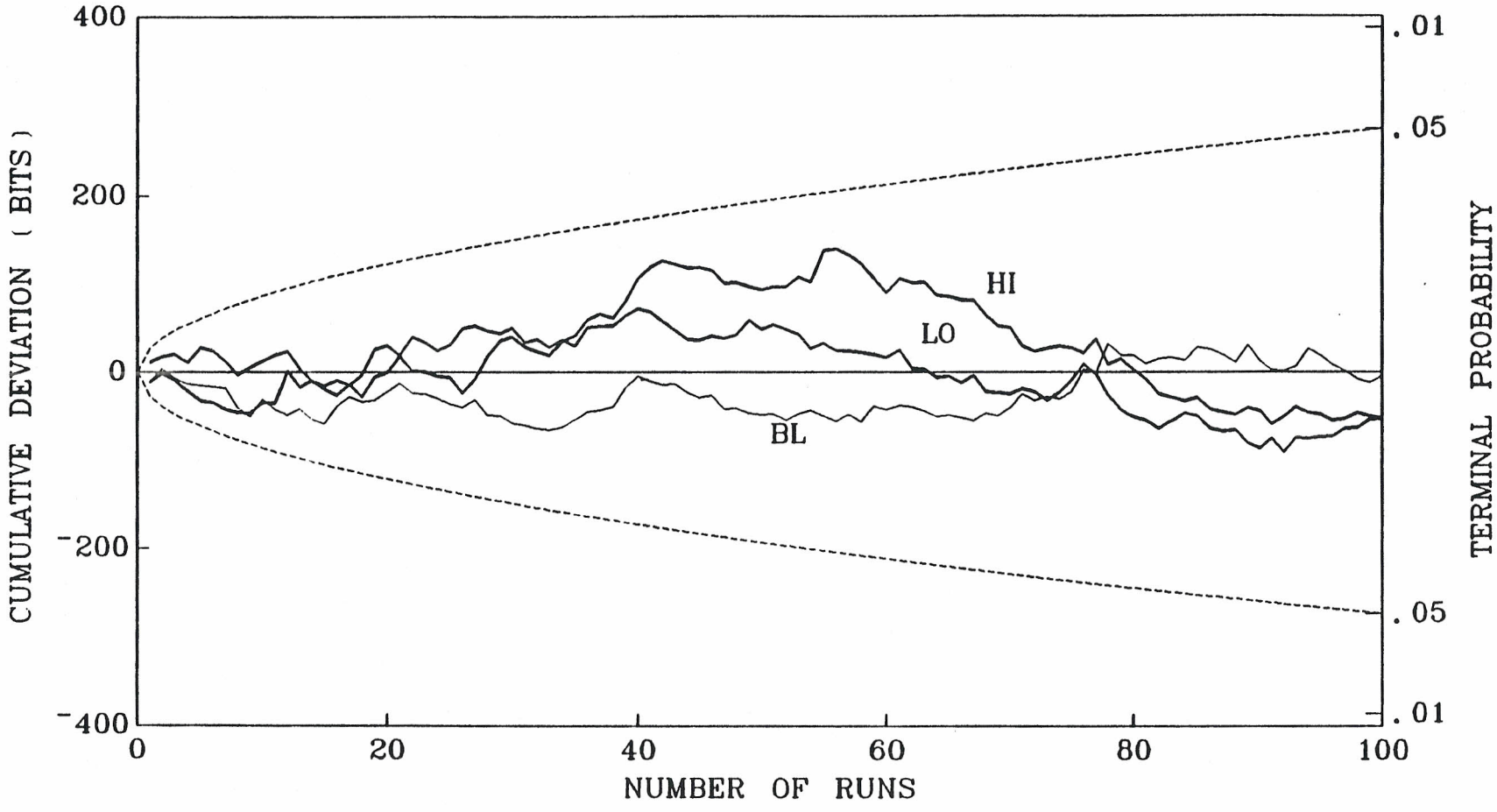
The formal statistical tests are complemented by various graphical displays of data, perhaps most instructive of which are cumulative deviation figures that trace the sequential development of results under each of the three operator intentions, in the form of corresponding random walks. These cumulative deviation traces are superimposed on an envelope indicating the significance of divergences from chance values, and show by inspection the magnitude and regularity of effect sizes, the influence of secondary parameters, and the degree of replicability of operator patterns. Figure 14 shows cumulative deviations of the complete 200-sample data base of 87 series, generated over the period January 1980 to February 1987 by 33 operators. Comparison of this figure with the calibration data similarly displayed in Figure 7 gives a good indication of the signal-to-noise situation for this type of experiment.

A number of other features of the experimental data can be illuminated by alternative graphical displays. For example, the possibility that the variance of trial score distributions also may deviate from chance expectation in correlation with operator intention may, like the mean shifts, be displayed in cumulative deviation graphs superimposed on chance probability envelopes (Figure 15). To pursue the infrastructure of the experimental data yet further, the patterns of frequency of individual trial scores can be displayed as histograms superimposed upon an envelope of Gaussian expectations to reveal whether the anomalous mean shifts are attributable to an excess or deficiency of particular score values, or follow a more general pattern of distortion (Figure 16). The results of such data dissections have proven extraordinarily instructive and are still being actively pursued.



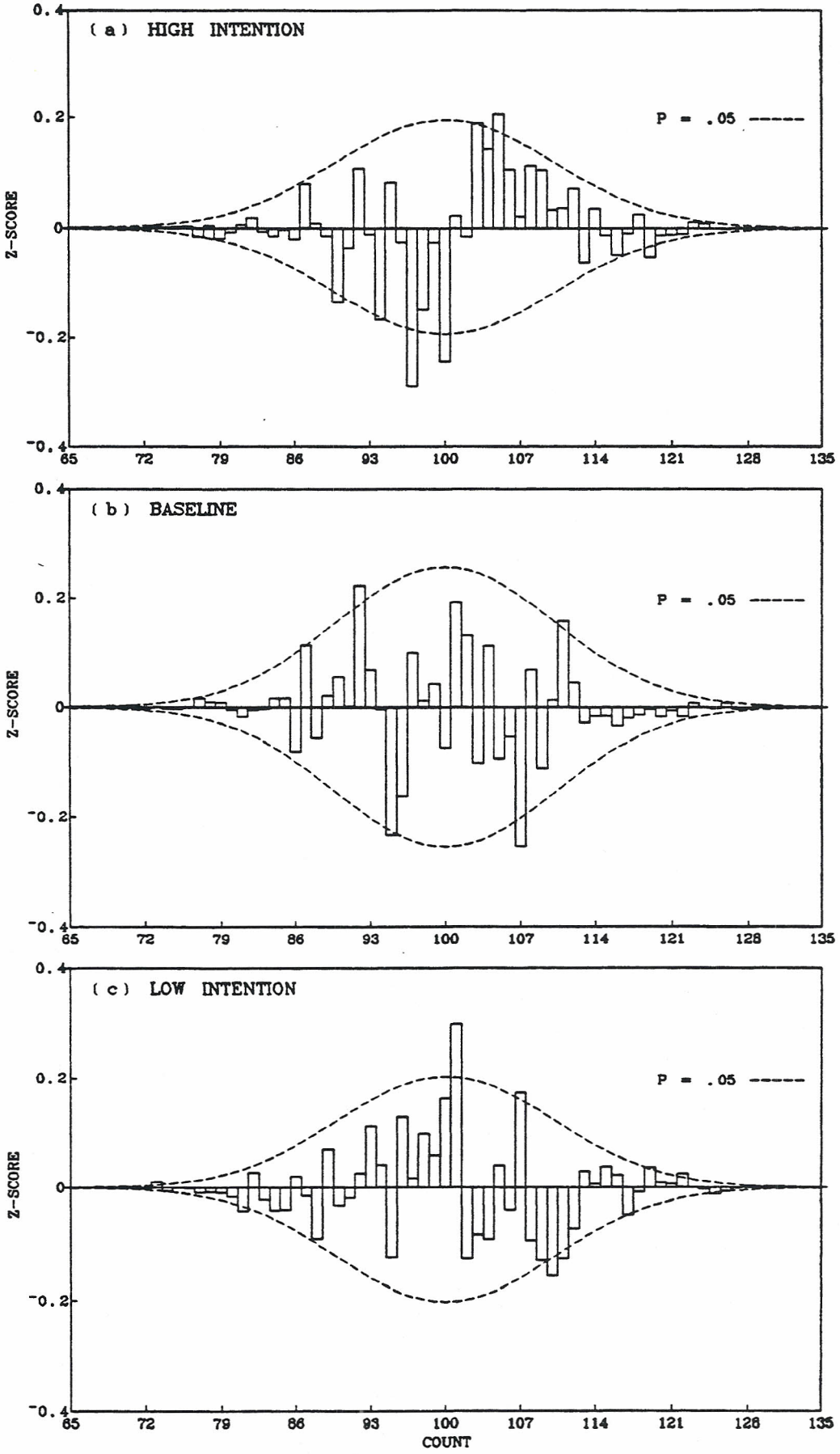
REG 200-sample: All Data, 87 Series, 33 Operators

FIGURE 14



Cumulative Deviation of Variance, Series 1, Operator 10

FIGURE 15



REG Excess Frequency of Counts, 35100 Trials, Manual

FIGURE 16

V. SUMMARY

Stringent qualification and calibration procedures ensure that the REG machines perform according to specifications. Various mechanical and operational failsafes protect against incidental aberrations during experimental applications, while tripolar experimental protocols further preclude distortion of data by artifacts. The calibration data, regularly generated in protocols that parallel those specified for experiments, are subjected to a standard battery of prespecified and well defined statistical procedures and graphical comparisons to confirm that the unattended REG output conforms closely to theoretical expectation. Thus, an appropriate background has been established for the demonstration and assessment of statistical anomalies in experimental data acquired under comparable conditions and analyzed by identical methods.

ACKNOWLEDGEMENTS

We wish to acknowledge the generous contributions of our colleagues in the PEAR laboratory, and the assistance and suggestions of colleagues within and outside the university. This work was supported by grants from the McDonnell Foundation, the John E. Fetzer Foundation, Inc., the Ohrstrom Foundation, and Helix Investments, Ltd.

REFERENCES

1. B. J. Dunne, R. G. Jahn, and R. D. Nelson, "An REG experiment with large data base capability." Technical Note PEAR 81001, Princeton Engineering Anomalies Research, Princeton University, School of Engineering/Applied Science, 1981.

2. B. J. Dunne, R. G. Jahn, and R. D. Nelson, "An REG experiment with large data base capability, II: Effects of sample size and various operators." Technical Note PEAR 82001, Princeton Engineering Anomalies Research, Princeton University, School of Engineering/Applied Science, 1982.

3. R. D. Nelson, B. J. Dunne, and R. G. Jahn, "An REG experiment with large data base capability, III: Operator related anomalies." Technical Note PEAR 84003, Princeton Engineering Anomalies Research, Princeton University, School of Engineering/Applied Science, 1984.

4. R. D. Nelson, R. G. Jahn, and B. J. Dunne, "Operator related anomalies in physical systems and information processes." *Journal of the Society for Psychical Research, 53,* 261-286, April 1986.

5. R. G. Jahn, B. J. Dunne, and R. D. Nelson, "Engineering anomalies research." *Journal of Scientific Exploration, 1,* 21-50, 1987.

6. S. Babu, "Analysis of variance of REG data." *Proceedings of the 29th Annual Convention of the Parapsychological Association,* Sonoma State University, Rohnert Park, CA, August, 1986.

7. R. D. Nelson and Y. H. Dobyns, "Individual operator contributions in large data base anomalies research." *Proceedings of the 31st Annual Convention of the Parapsychological Association,* Montreal, Quebec, August, 1988.

8. D. I. Radin and R. D. Nelson, "Evidence for consciousness-related anomalies in random

physical systems." *Foundations of Physics, Vol. 20, No. 1,* in press.

9. G. W. Snedecor, and W. G. Cochran, *Statistical Methods,* Seventh edition. Ames, Iowa: The Iowa State University Press, 1980.

10. G. E. P. Box, W. G. Hunter, and J. S. Hunter, *Statistics for Experimenters: An Introduction to Design, Data Analysis, and Model Building.* New York: Wiley, 1978.

11. A. L. Edwards, *Experimental Design in Psychological Research,* Fifth edition. New York: Harper and Row, 1985.

12. G. E. P. Box and G. M. Jenkins, *Time Series Analysis, Forecasting, and Control,* second edition. San Francisco: Holden-Day, 1976.

13. W. W. Daniel, *Applied Nonparametric Statistics.* Boston: Houghton Mifflin Company, 1978.

14. W. Feller, *An Introduction to Probability Theory and Its Applications,* Vol. 1, Second edition. New York: Wiley, 1957.

15. E. Vanmarcke, *Random Fields: Analysis and Synthesis.* Cambridge, MA: The MIT Press, 1983.

16. J. Neter, and W. Wasserman, *Applied Linear Statistical Models.* Homewood, IL: Richard D. Irwin, Inc., 1974.

APPENDIX A: C Language Source Code for REG Data Analysis

The original statistical tests used in routine analyses of early calibration and experimental data were embodied in a custom computer program called STAT, written in C language. This program has recently been supplanted by NEWSTAT, which provides a more comprehensive battery of analytical and graphical tools for data assesment. In addition, an extensive array of APL functions are routinely applied in assessing calibration data. The algorithms for the primary statistical parameters are all similar to those in the original STAT program, the source code for which is given here for examination or application by interested readers.

APPENDIX A: STAT

```
/*
 *   STATISTICAL ANALYSIS PROGRAM        April, 1980
 *     revised Feb 81, Sept 81.
 *
 * Statistical analysis program for a binary random event generator
 *
 * NOTE: Number of events per trial should be even ( restriction to
 *     simplify the analysis program ).  Number of trials ( data
 *     points ) in sample cannot be greater than 15,000 unless the
 *     dimension of the data array is increased.
 *
 * The following files must be available :
 *   (1) ../filenum       Current number of data storage file.
 *   (2) ../index         Index file for experiment identification.
 *
 * The following files will be modified or created :
 *   (1) ../filenum       The stored number, kkkk, will be incremented.
 *   (2) ../data.kkkk     Data file ( kkkk = 1 + current number )
 *   (3) ../stan.kkkk     Results of Statistical Analysis.
 *
 * To read ( print or display on CRT terminal ) any of these files,
 * type " t space filename ". The UNIX system generates a $ sign on
 * the terminal when it is ready to receive a command,  -  so the
 * instruction to print a listing of stan.0003 would look like this:
 *
 *            $ t stan.0003
 *
 * The data file can easily be edited to correct mistakes ( without
 * re-entering the entire set of data values ) by using the UNIX
 * editor.
 *
 *     Note: "Raw" data files are much more compact, but cannot
 * be edited as easily.  The raw option can be used when concatinating
 * files to save file space ( editing is no problem in this case ).
 *
 * Data input files should contain only strings of numbers separated
 * by one or more blanks, and should always be terminated by an EOF.
 * The End-of-File symbol is -1 ( 377 in octal, or 11111111 in binary ),
 * and is usually inserted automatically by UNIX programs.
 *
 * Compile the program as follows:
 *       cc -O -s stat.c -lm -o STAT        (as of sept 81)
 */
#include <stdio.h>
#include <math.h>
char nfile[] = "0000";
char *name[] = { "./data.0000",
                 "./stan.0000",
                 "./dist.0000"  };   /* dist added sept 81  */
char *form[] = { "File  Subject Exper  n   N  Rate   Date  Time",
                 "          E/H                          ",
                 "   c    m A E/L M                        ",
                 "   s xxx f a B/0 A xxxx/yyyy/zzzzz mm/dd/yy hh/mm" };
int data[15001], sflag, dflag = 0;   /* dflag and initialization 9/81 */
double PC[17] = { 0.999, 0.995, 0.990, 0.975, 0.950, 0.900, 0.800,
                  0.700, 0.500, 0.300, 0.200, 0.100, 0.050, 0.025,
                  0.010, 0.005, 0.001  };
double C8[17] = { 0.500, 1.344, 1.646, 2.180, 2.733, 3.490, 4.594,
                  5.527, 7.344, 9.524, 11.03, 13.36, 15.51, 17.53,
                  20.09, 21.96, 26.10  };
double C16[17]= { 1.000, 5.142, 5.812, 6.908, 7.962, 9.312, 11.15,
                  12.62, 15.34, 18.42, 20.47, 23.54, 26.30, 28.85,
                  32.00, 34.27, 39.30  };
main() {
    FILE *fopen(), *fp;
    extern int data[], sflag;
    char c, cc[2], buf[512], line[133];
    int max, min, num, i, j, k, m, n, nc, flag, qflag = 0;
    double fnum, avg, sdev;
    double sqrt();
    int stan();

start:

printf("\n\n              STATISTICAL ANALYSIS PROGRAM");
printf("\n              FOR  RANDOM EVENT GENERATOR");

/*  Open files for data storage, get file number, and type identification */

   sflag = 0;
   printf("\n\n Save data ? Type yes ( y ) or no ( n ) and RETURN:  ");
   fgets(line,133,stdin);
   if( line[0] == 'y' ) sflag = 1;
   if( sflag == 0 ) goto begin;

printf("\n Specify format of data storage file: s for Standard, r for Raw: ");
   fgets(line,133,stdin);
   if( line[0] == 'r' ) sflag = 2;

fp = fopen("./filenum", "r");
   if( fp == NULL ) { printf(" ***  Unable to read from filenum\n");
                goto quit; }
   num = fscanf(fp,"%4d",&nc);
   if( num == EOF ) { printf(" ***  Error in filenum file\n");
                goto quit;  }
   fclose(fp);
fp = fopen("./filenum","w");
   if( fp == NULL ) { printf(" ***  Unable to write to filenum\n");
                goto quit; }
   nc = nc + 1; if( nc <= 0 ) nc = 1;
            if( nc > 9999 ) nc = 1;
```

```
    k = nc;
    fprintf(fp,"%4d\n",nc);
    fclose( fp );
i = 10; do { name[0][i] = nc%10 + '0';
             name[1][i] = name[0][i];
             name[2][i] = name[0][i];          /* sept 81 */
             nfile[i-7] = name[0][i];
        i = i - 1;          }
     while( (nc/=10 ) > 0 );

fp = fopen("./index", "a" );
    if( fp == NULL ) { printf(" *** Unable to open file index\n");
                goto quit; }
printf("\n Storage file number for data and results = %3d", k);
printf("\n Type test identification data using the format shown:");
printf("\n\n%s\n%s\n%s\n%s\n%s ",form[0],form[1],form[2],form[3],nfile);

     fgets(line,133,stdin);
     fprintf(fp,"%4s ",nfile);
     fputs(line,fp );
     fclose(fp);

/*   Display instructions at the beginning of the test run  */

begin:
     m = 200;

printf("\n Type the number of random events, N , per trial.\n");
printf(" Default value is 200. Type value of N or RETURN: ");
printf("\n\n                N = ");
          fgets(line,133,stdin);
     sscanf(line,"%d",&m);
     if( m <= 0 ) m = 200;
     if( 2*(m/2) != m ) { printf("\n *** Note: N should be even\n\n");
                          goto begin; }
     avg = (m/2);
     sdev = sqrt(avg/2.0);
     n = 0; num = 0; flag = 0; fnum = 0.0; max = 0; min = m;

printf("\n For N = %4d , Expected Result = %5.1f with Std Dev = %7.3f\n\n",
          m,              avg,          sdev);

printf(" Maximum number of data entries limited to 15,000 per file\n");
repeat:
printf("\n Type t to take input data from the terminal\n");
printf(" Type f to take input from a standard ASCII UNIX file\n");
printf(" Type d to take input from a raw data file\n");
printf(" Type q to quit\n\n");
printf(" Select source of input data: ");
     fgets(line,133,stdin); c = line[0];
     if( c == 'f' ) goto file;
          if( c == 'd' ) { flag = 1; goto file; }
     if( c == 'q' ) goto quit;
     if( c != 't' ) { printf(" *** Unable to identify choice.\n");
                goto repeat; }

printf("\n Enter test data, separating numbers by spaces or newlines\n");
printf(" Type e to indicate end of data file and continue\n");
printf(" Type q to indicate end of last data file\n\n");

enter:
     nc = scanf("%d", &num);
     if( nc == EOF ) goto enter;
     if( nc != 1 ) { c = getchar();
                if( c == 'q' || c =='Q' ) qflag = 1;
                        while( c != '\n' ) c = getchar();
                goto end;  }
     if( num > m || num < 0 ) {
printf(" *** Error in input number %d , entry ignored.\n",num);
                        goto enter; }

     n = n + 1;
     if( n <= 15000 ) data[n] = num;
     else { n = n - 1;
          printf("\n *** Input file exceeds limit of 15,000 numbers");
          goto end;    }
     fnum = fnum + num;
     if( max < num ) max = num;
     if( min > num ) min = num;
     goto enter;

file:
     printf(" Type input file name : ");
     fgets( line, 133, stdin );
          if( line[0] == 'q' ) goto quit;
          i = 0; while( i < 132 ) {
                    if( line[i] == '\n' ) { line[i] = '\0'; i = 132; }
                    else i = i + 1;     }
          fp = fopen( line, "r" );
               if( fp == NULL ) {
printf(" *** Unable to open file  %s", line );
printf("\n      Check name and try again, or type q and RETURN to quit\n\n");
                                  goto file;      }

fenter:
          if( flag == 0 ) {
                    nc = fscanf( fp, "%d", &num);
                    if( nc == EOF ) goto close;
                    fscanf( fp, "%c", &c);  }
          else {  num = getw( fp );
                    if( num == EOF ) goto close; }
          if( num < 0 || num > m ) goto fenter;

          n = n + 1;
```

```
    if( n <= 15000 ) data[n] = num;
    else { n = n - 1;
        printf("\n *** Input file exceeds limit of 15,000 numbers");
            goto close; }
    fnum = fnum + num;
    if( max < num ) max = num;
    if( min > num ) min = num;
    goto fenter;
close:
    fclose( fp );
end:
    if( n == 0 ) goto quit;
    fnum = fnum/n;

printf("\n\n List input data ? Type y or n and RETURN: ");
    fgets(line,133,stdin);
    if( line[0] != 'y' ) goto contin;

printf("\n For N = %5d random binary events per trial :",m);
printf("\n Expected trial result = %8.2f with SDEV = %10.4f",avg,sdev);
printf("\n ACTUAL MEAN for %5d TRIALS = %8.2f with MIN = %5d MAX = %5d",
      n, fnum, min, max);
printf("\n\n ROW DATA\n\n");
    i = 0; k = 0;
    while( i < n ) { j = 0; k = k + 1; printf(" %4d ",k);
            while( j < 10 && i < n ) { j = j + 1; i = i + 1;
                    printf(" %6d",data[i]); }
            printf("\n"); }

printf("\n Type RETURN to continue ");
    fgets(line,133,stdin);

/* Store input data in data file for future reference */

contin:
   if( sflag == 0 ) goto skip;
   fp = fopen( name[0], "w" );
   if( fp == NULL ) { printf(" *** Unable to create data file\n");
            goto quit;}
   i = 0; k = 0;
   if( sflag != 2 ) {
     while( i < n ) { if( k < 10 ) { i = i + 1; k = k + 1;
                    fprintf( fp, "%6d ", data[i] ); }
            else { k = 0; fprintf( fp, "\n" ); }
            }
     fprintf(fp, "\n" );
            }
   else {
     while( i < n ) { i = i + 1; putw( data[i], fp ); }
     }
   fclose(fp);
skip:

/* Call Statistical Analysis Program */

   stan( m, n );

printf("\n Type RETURN to continue ");
      fgets(line,133,stdin);
if( qflag == 0 ) goto start;   /* Start over again, or quit. */

quit:
   exit(0);
}
/*
 * Statistical Analysis Program
 *
 * m = number of random events per trial ( N )
 * n = number of trials ( data points in sample, n >= 4 )
 */
stan(m,n) int m, n; {
    FILE *fopen(), *fp;
    extern int data[], sflag;
    extern double PC[], C8[], C16[];
    int a[201], x, me, i, j, k, sum, nhi, nlo, nme, max, min, skp8, skp16;
        char c, cc[2], line[133];
    double p[101], e[101], en8[9], en16[17], an8[9], an16[17], excess;
    double em, esig, avg, s, es, t, z, cves, cvs, eskew, askew, pt, pz;
    double mx, mu2, mu3, ms2, ms3, fm, fx, xn, fsum, fk, fk1, sdev;
    double mu4, ms4, ams4, ecur, acur, aacur;
    double chi8, chi16, tmp, p8, p16;
    double sqrt(), log(), exp(), fabs(), prob();

    if( n < 4 ) goto end;
    if( m < 2 ) goto end;
    xn = n; fm = m; em = 0.5*fm; me = m/2;
    sdev = sqrt(0.25*fm);

    p[0] = (0.7978846/sqrt(fm))*(1.0 - 0.25/fm + 0.032/(fm*fm) );
        if( m <= 10 ) { if( m == 2 ) p[0] = 0.5000;
                if( m == 4 ) p[0] = 0.3750;
                if( m == 6 ) p[0] = 0.3125;
                if( m == 8 ) p[0] = 0.27343750;
                if( m == 10) p[0] = 0.24609375; }
    e[0] = xn*p[0];

    for( j = 1; j <= 100; j++ ) {
       p[j] = 0.0; e[j] = 0.0;}
    k = me; if( k > 100 ) k = 100;
    fk = em; fk1 = fk + 1.0;
    for( j = 1; j <= k; j++ ) {
        p[j] = p[j-1]*fk/fk1;
        e[j] = xn*p[j];
        fk = fk -1.0; fk1 = fk1 + 1.0; }
```

```
    en8[4] = e[0] + 2.0*e[1];
    en8[3] = en8[5] = e[2] + e[3];
    en8[2] = en8[6] = e[4] + e[5];
    en8[1] = en8[7] = e[6] + e[7] + e[8];
    fsum = 0.0;
    for( j = 1; j <= 7; j++ ) { fsum = fsum + en8[j]; }
    en8[0] = en8[8] = 0.5*(xn - fsum);

    en16[8] = e[0];
    en16[7] = en16[9] = e[1];
    en16[6] = en16[10] = e[2];
    en16[5] = en16[11] = e[3];
    en16[4] = en16[12] = e[4];
    en16[3] = en16[13] = e[5] + e[6];
    en16[2] = en16[14] = e[7] + e[8];
    en16[1] = en16[15] = e[9] + e[10] + e[11];
    fsum = 0.0;
    for( j = 1; j <= 15; j++ ) { fsum = fsum + en16[j]; }
    en16[0] = en16[16] = 0.5*(xn - fsum);


    sum = 0; nhi = 0; nlo = 0; nme = 0; max = 0; min = m;
    for( j = 0; j <= 200; j++ ) { a[j] = 0; }

    for( j = 1; j <= n; j++ ) {
        x = data[j];
        if( x > me ) nhi = nhi + 1;
        if( x < me ) nlo = nlo + 1;
        if( x == me ) nme = nme + 1;
        if( max < x ) max = x;
        if( min > x ) min = x;
        k = x - me + 100;
        if( k < 0 ) k = 0; if( k > 200 ) k = 200;
        a[k] = a[k] + 1;
        sum = sum + x - me;      }

    excess = sum;
    avg = em + (excess/xn);

    mu2 = 0.0; mu3 = 0.0; ms2 = 0.0; ms3 = 0.0; mu4 = 0.0; ms4 = 0.0;
    for( j = 1; j <= n; j++ ) {
        fx = data[j];
        mx = fx - em;
        tmp = mx*mx;
        mu2 += tmp;
        mu3 += tmp*mx;
        mu4 += tmp*tmp;

        mx = fx - avg;
        tmp = mx*mx;
        ms2 += tmp;
        ms3 += tmp*mx;
        ms4 += tmp*tmp;    }

    mu2 = mu2/xn;   mu3 = mu3/xn;
    mu4 = mu4/xn;   ams4 = ms4/xn;
    ms2 = ms2/(xn-1.0);  ms3 = ms3*xn/((xn-1.0)*(xn-2.0));
    esig = sdev/sqrt(xn);
    s = sqrt(ms2);    cvs = s/sdev;
    es = sqrt(mu2);   cves = es/sdev;

    z = ( avg - em )/esig;
    if( s == 0.0 ) t = 0.0;
        else t = (avg - em )*sqrt(xn)/s;

    tmp = 3.0*(2.0*xn - 3.0 )*ms2*ms2;
    fx = (xn - 1.0)*(xn - 2.0)*(xn - 3.0);
    ms4 = ((xn*xn - 2.0*xn + 3.0)*ms4 - tmp)/fx;

    ecur = (mu4/(mu2*mu2)) - 3.0;
    acur = (ms4/(ms2*ms2)) - 3.0;
    fx = ms2*(xn-1.0)/xn;
    fx = fx*fx;
    aacur = (ams4/fx) - 3.0;
    fx = fabs(ecur);
    fx = exp( log(fx)/4.0 );
    if( ecur < 0.0 ) fx = -fx;
    ecur = fx;

    fx = fabs(acur);
    fx = exp( log(fx)/4.0 );
    if( acur < 0.0 ) fx = -fx;
    acur = fx;

    fx = fabs(aacur);
    fx = exp( log(fx)/4.0 );
    if( aacur < 0.0 ) fx = -fx;
    aacur = fx;

    if( ms3 == 0.0 ) { askew = 0.0; goto contin1; }
    if( s == 0.0 )   { askew = 0.0; goto contin1; }
    askew = fabs(ms3);
    askew = exp( log(askew)/3.0 )/s;
    if( ms3 < 0.0 ) askew = -askew;

contin1:
    if( mu3 == 0.0 ) { eskew = 0.0; goto contin2; }
    eskew = fabs(mu3);
    eskew = exp( log(eskew)/3.0 )/sdev;
    if( mu3 < 0.0 ) eskew = -eskew;

contin2:
    chi8 = 0.0;  chi16 = 0.0;  skp8 = 0;  skp16 = 0;
    if( en8[0]<1.0 || en8[1]<1.0 ) { skp8 = 1; }
```

```
    an8[0] = an8[8] = 0.0;
    an8[4] = ( a[100] + a[99] + a[101] );
    an8[3] = ( a[97] + a[98] );
    an8[2] = ( a[95] + a[96] );
    an8[1] = ( a[92] + a[93] + a[94] );
    for( j = 0; j <= 91; j++ ) { an8[0] = an8[0] + a[j]; }
    an8[5] = ( a[102] + a[103] );
    an8[6] = ( a[104] + a[105] );
    an8[7] = ( a[106] + a[107] + a[108] );
    for( j = 109; j<= 200; j++ ) {an8[8] = an8[8] + a[j]; }

    for( j = 0; j <=8; j++ ) {
        tmp = an8[j] - en8[j];
        chi8 = chi8 + tmp*tmp/en8[j]; }

    if( en16[0]<1.0 || en16[1]<1.0 ) { skp16 = 1; }
    an16[0] = an16[16] = 0.0;
    an16[8] = a[100];
    an16[7] = a[99];
    an16[6] = a[98];
    an16[5] = a[97];
    an16[4] = a[96];
    an16[3] = ( a[94] + a[95] );
    an16[2] = ( a[92] + a[93] );
    an16[1] = ( a[89] + a[90] + a[91] );
    for( j = 0; j <= 88; j++ ) { an16[0] = an16[0] + a[j]; }
    an16[9] = a[101];
    an16[10] = a[102];
    an16[11] = a[103];
    an16[12] = a[104];
    an16[13] = ( a[105] + a[106] );
    an16[14] = ( a[107] + a[108] );
    an16[15] = ( a[109] + a[110] + a[111] );
    for(j=112; j <= 200; j++ ) { an16[16] = an16[16] + a[j]; }
    for( j = 0; j <= 16; j++ ) {
        tmp = an16[j] - en16[j];
        chi16 = chi16 + tmp*tmp/en16[j]; }

    fk = ( min - me )/sdev;
    fk1 = ( max - me )/sdev;
    fsum = excess/(xn*esig);

/*  Optional print of distribution data      */

printf("\n  List integer distribution data ?  Type y or n and RETURN:  ");
    fgets(line,133,stdin);
    if( line[0] != 'y' ) goto skip1;

printf("\n\n      Expected Distribution  -  Not Rounded\n\n");
printf("ROW/COL=");
i=0; while( i <= 9 ) { printf("  %1d    ",i); i = i + 1; }
printf("\n\n"); i = 0; k = 0;
while( i < 99 ) { j = 0; printf("  %3d  ",k); k = k + 1;
            while( j < 10 ) { printf(" %6.2f",e[i]);
                        i = i + 1; j = j + 1; }
            printf("\n"); }
i = 100; printf("  %3d   %6.2f",k,e[i]);

sleep(5);

printf("\n\n      Actual Distribution  -  Integer\n\n");
i = 0; k = 0;
while( i < 199 ) { j = 0; printf("  %3d  ",k); k = k + 1;
            while( j < 10 ) { printf("%7d",a[i]);
                        i = i + 1; j = j + 1; }
            printf("\n"); }
i = 200; printf("  %3d  %7d",k,a[i]);

sleep(5);

printf("\n\n      Chi-Square Cell Distributions");
printf("\n\nen = Expected Number per Cell, an = Actual Number per Cell");
printf("\n\nen8 = %8.2f%8.2f%8.2f%8.2f%8.2f%8.2f%8.2f%8.2f%8.2f",
    en8[0],en8[1],en8[2],en8[3],en8[4],en8[5],en8[6],en8[7],en8[8]);
printf("\nan8 = %8.2f%8.2f%8.2f%8.2f%8.2f%8.2f%8.2f%8.2f%8.2f",
    an8[0],an8[1],an8[2],an8[3],an8[4],an8[5],an8[6],an8[7],an8[8]);
printf("\n\nen16 = %8.2f%8.2f%8.2f%8.2f%8.2f%8.2f%8.2f%8.2f%8.2f",
 en16[0],en16[1],en16[2],en16[3],en16[4],en16[5],en16[6],en16[7],en16[8]);
printf("\nan16 = %8.2f%8.2f%8.2f%8.2f%8.2f%8.2f%8.2f%8.2f%8.2f",
 an16[0],an16[1],an16[2],an16[3],an16[4],an16[5],an16[6],an16[7],an16[8]);
printf("\n\nen16 = %8.2f%8.2f%8.2f%8.2f%8.2f%8.2f%8.2f%8.2f",
 en16[9],en16[10],en16[11],en16[12],en16[13],en16[14],en16[15],en16[16]);
printf("\nan16 = %8.2f%8.2f%8.2f%8.2f%8.2f%8.2f%8.2f%8.2f",
 an16[9],an16[10],an16[11],an16[12],an16[13],an16[14],an16[15],an16[16]);

skip1:

printf("\n\n  List Per Cent Distribution Data ( 120 Columns ) ?  ");
    fgets(line,133,stdin);
    if( line[0] != 'y' ) goto skip2;

    printf("\n Create distribution data file?");  /* added 9/81 */
    fgets(line,133,stdin);
    if (line[0] == 'y') {
                dflag = 1;
                if((fp = fopen(name[2],"w")) == NULL){
                        printf("\n *** Unable to create dist file\n");
                        goto skip2;
                }
        }

printf("\n\n      Probability Distribution  -  Per Cent\n\n");

if (dflag == 1)                                /* 9/81 */
```

```
        fprintf(fp,"\n\n      Probability Distribution  -  Per Cent\n\n");

if (dflag == 1)
        fprintf(fp,"ROW/COL=");
printf("ROW/COL=");

for(i=0; i <= 9;i++) {
        printf(" %1d       ",i);
        if(dflag == 1) fprintf(fp," %1d       ",i);
}
printf("\n\n");
if (dflag == 1) fprintf(fp,"\n\n");

for( j = 0; j <= 100; j++ ) { p[j] = 100.0*p[j]; }
i = 0; k = 0;
while( i < 99 ) {
        j = 0;
        printf(" %3d ",k);
        if(dflag==1) fprintf(fp," %3d ",k);
        k++;
        while( j < 10 ) {
                printf("%11.3e",p[i]);
                if(dflag==1) fprintf(fp,"%11.3e",p[i]);
                i++; j++;
        }
        printf("\n");
}
i = 100;
printf(" %3d %11.3e",k,p[i]);
if(dflag==1) fprintf(fp," %3d %11.3e",k,p[i]);

printf("\n\n      Actual Distribution  -  Per Cent for n = %5d\n\n",n);
if(dflag==1)
 fprintf(fp,"\n\n      Actual Distribution  -  Per Cent for n = %5d\n\n",n);

i = 0; k = 0; tmp = 100.0/xn;
while( i < 199 ) {
        j = 0;
        printf(" %3d ",k);
        if (dflag == 1) fprintf(fp," %3d ",k);
        k++;
        while( j < 10 ) {
                fx = a[i]; fx = tmp*fx;
                printf("%11.3e",fx);
                if(dflag==1) fprintf(fp,"%11.3e",fx);
                i++;j++;
        }
        printf("\n");
        if(dflag==1) fprintf(fp,"\n");
}
i = 200; fx = a[i]; fx = tmp*fx; printf(" %3d %11.3e",k,fx);

skip2:

/*      Compute z-parameter Probability         */

  if( z == 0.0 ) { pz = 0.50;  goto cont; }
  fx = z;  if( fx < 0.0 ) fx = -fx;

  pz = prob( fx );

cont:

/*      Compute t-parameter Probability         */

  if( t == 0.0 ) { pt = 0.50; goto con0; }
  fx = t; if( fx < 0.0 ) fx = -fx;
  tmp = 0.01817 + 0.01991*fx;
  tmp = 0.4705 + fx*tmp;
  tmp = 1.0 - fx*tmp/(xn - 1.0);
  fx = fx*tmp;

  pt = prob( fx );

con0:

/*      Compute Chi-Sq Probabilities            */

if( chi8 <= C8[0] ) { p8 = PC[0]; goto con2; }
if( chi8 >= C8[16] ) { p8 = PC[16];
                if( chi8 >= 40.0 ) p8 = 0.0;
                goto con2;  }

k = 1;
while( k <= 16 ) { if( chi8 <= C8[k] ) goto con1;
            k = k + 1; }
con1:
    fx = C8[k] - C8[k-1];
    p8 = PC[k-1] - PC[k];
    p8 = PC[k-1] - p8*(chi8 - C8[k-1])/fx;
con2:
    if( chi16 <= C16[0] ) { p16 = PC[0]; goto con4; }
    if( chi16 >= C16[16] ) { p16 = PC[16];
                    if( chi16 >= 60.0 ) p16 = 0.0;
                    goto con4;  }
    k = 1;
    while( k <=16 ) { if( chi16 <= C16[k] ) goto con3;
                k = k + 1;  }
con3:
    fx = C16[k] - C16[k-1];
    p16 = PC[k-1] - PC[k];
    p16 = PC[k-1] - p16*(chi16 - C16[k-1])/fx;

con4:
```

```
    tmp = 0.0;  fx = 1.0;

/*    Display results on the control console    */

printf("\n\n          STATISTICAL ANALYSIS OF DATA \n \n");
k = m/2;
printf(" Summary:   %5d  Trials for  N      = %4d  \n",n,m);
printf("                        Mean    = %4d\n",k);
printf("                        Std Dev = %8.3f\n\n",sdev);
printf(" Expected Mean = %8.3f    Std Dev of Mean of Mean = %8.3f \n",
                em,                          esig);
printf(" Actual Mean   = %8.3f   ( Delta Mean / Std Dev ) = %8.3f \n\n",
                avg,                         z );
printf(" Accumulated Total Relative to Exp Mean = %6d  ( %6.3f ) \n\n",
                                  sum, fsum);
printf(" MINIMUM = %4d ( %6.2f )      MAXIMUM = %4d ( %6.2f ) \n",min,fk,max,fk1);
printf(" Distribution:  n-Low = %3d   n-Mean = %3d   n-High = %3d \n\n",
                    nlo,       nme,       nhi );
printf(" Parameter          Nominal   Rel-Exp-Mean    Rel-Actual-Mean \n");
printf(" ---------          -------  ------------   ---------------- \n");
printf(" Std Dev / Trial   =  %8.3f    %8.3f      %9.3f \n",sdev,es,s);
printf(" Std Dev Ratio     =  %8.3f    %8.3f      %9.3f \n",fx,cves,cvs);
printf(" Skew Coef     3R =  %8.3f    %8.3f      %9.3f \n",tmp,eskew,askew);
printf(" Kurtosis Coef  4R =  %8.3f    %8.3f      %9.3f \n\n",tmp,ecur,acur);

printf(" z-parameter  =  %8.3f ,  Probability = %8.5f \n\n",z,pz);
printf(" t-parameter  =  %8.3f ,  Probability = %8.5f \n\n",t,pt);
     if( skp8 == 1 ) goto skip3;
printf(" Chi-Sq 8     =  %7.2f  ,  Probability = %6.3f \n\n",chi8,p8);
     if( skp16 == 1 ) goto skip3;
printf(" Chi-Sq 16    =  %7.2f  ,  Probability = %6.3f \n\n",chi16,p16);
skip3:

/*    Store results in a stan file for future reference   */

if( sflag == 0 ) goto end;
fp = fopen( name[1], "w" );
    if( fp == NULL ) { printf("\n  Unable to create stan file");
                goto end; }
fprintf(fp, "\n\n          STATISTICAL ANALYSIS OF DATA \n \n");
k = m/2;
fprintf(fp, " Summary: %5d  Trials for  N      = %4d \n",n,m);
fprintf(fp, "                        Mean    = %4d\n",k);
fprintf(fp, "                        Std Dev = %8.3f\n\n",sdev);
fprintf(fp, " Expected Mean = %8.3f    Std Dev of Mean of Mean = %8.3f \n",
                em,                          esig);
fprintf(fp, " Actual Mean   = %8.3f   ( Delta Mean / Std Dev ) = %8.3f \n\n",
                avg,                         z );
fprintf(fp, " Accumulated Total Relative to Exp Mean = %6d  ( %6.3f ) \n\n",
                                  sum, fsum);
fprintf(fp, " MINIMUM = %4d ( %6.2f )      MAXIMUM = %4d ( %6.2f ) \n",min,fk,max,fk1);
fprintf(fp, " Distribution:  n-Low = %3d   n-Mean = %3d   n-High = %3d \n\n",
                    nlo,       nme,       nhi );
fprintf(fp, " Parameter          Nominal   Rel-Exp-Mean    Rel-Actual-Mean \n");
fprintf(fp, " ---------          -------  ------------   ---------------- \n");
fprintf(fp, " Std Dev / Trial   =  %8.3f    %8.3f      %9.3f \n",sdev,es,s);
fprintf(fp, " Std Dev Ratio     =  %8.3f    %8.3f      %9.3f \n",fx,cves,cvs);
fprintf(fp, " Skew Coef     3R =  %8.3f    %8.3f      %9.3f \n",tmp,eskew,askew);
fprintf(fp, " Kurtosis Coef  4R =  %8.3f    %8.3f      %9.3f \n\n",tmp,ecur,
                                  acur);

fprintf(fp, " z-parameter  =  %8.3f ,  Probability = %8.5f \n\n",z,pz);
fprintf(fp, " t-parameter  =  %8.3f ,  Probability = %8.5f \n\n",t,pt);
     if( skp8 == 1 ) goto skip4;
fprintf(fp, " Chi-Sq 8     =  %7.2f  ,  Probability = %6.3f \n\n",chi8,p8);
     if( skp16 == 1 ) goto skip4;
fprintf(fp, " Chi-Sq 16    =  %7.2f  ,  Probability = %6.3f \n\n",chi16,p16);
skip4:

fclose(fp);
printf("\n Results stored in files:  %s  %s",name[0],name[1]);

end:
    return(0);
}
double prob(x)  double x;  {
    double px, tmp, exp();
    tmp = 1.0/(1.0 + 0.2316419*x);
    px = -1.821255978 + 1.330274429*tmp;
    px = tmp*px + 1.781477937;
    px = tmp*px - 0.356563782;
    px = tmp*px + 0.319381530;
    px = 0.39894228*tmp*px;
    px = px*exp( -0.50*x*x );
return( px );
}
```

Appendix B: Distributions of Extreme Scores in REG Calibrations

(Originally printed November 1989 as an addendum to Technical Note PEAR 89001)

Introduction

In experiments consisting of trials, runs, or other grouped data drawn from a population of random variates, large blocks of calibration data clearly should display distribution parameters that are nominal, according to a variety of standard tests. But beyond this, any subgroups of these calibration data, particularly subgroups corresponding in length to experimental trials, runs, or series, must also conform to theoretical or empirically established expectations. This addendum describes a set of procedures that augment the canonical statistical tests detailed in the body of this report, to assure conformance of such grouped data to appropriate standards.

Extreme Score Distributions

Since experimental anomalies data are defined primarily by an accumulation of terminal group scores, it is necessary to determine the distribution of such scores in calibration data. For example, a large number of raw trials can be examined in sequential groups comparable in length to experimental runs, e. g., groups of 50, 100, and 1000 trials. The standard parametric tests of mean, standard deviation, skew, and kurtosis for the grouped data can then be supplemented by comparing against theoretical expectation the actual number of terminal scores above the mean and in the upper and lower 5% and 1% tails.

The assessment of the distribution of extreme scores is a particularly appropriate and sensitive indicator of deviations from randomicity in the time-series structure that may escape detection by the usual autocorrelation and frequency analysis procedures. For example, weak long-wavelength structure in the serial data, on the order of the grouping length, may produce concatenations that are deviant, resulting in spurious suggestions of anomalous effects in experiments.

Procedure

This supplementary analysis was applied to 100,000 REG calibrations trials, each consisting of 200 binary samples. The 50,000 trials employed for the tests outlined in the body of the report were combined with 50,000 calibrations done immediately after the REG was checked and electronically calibrated in 1988. These were again subjected to the original battery of statistical tests, then processed using a set of programs developed to examine the distribution of serial subgroup scores greater than the expected mean and scores in the 5% and 1% tails. Briefly, the raw data in their original order were assembled into groups or 50, 100, and 1000 trials, and the group distribution parameters were computed. A new set of normalized database statistics was calculated from the group results, and the number of Z-scores above the mean, and beyond 1.645 and 2.326 in each tail was counted. The expected numbers were calculated for the ideal distribution, and a Z or T score was computed for the difference between actual and expected values.

To confirm the validity of the procedures, and to provide a standard against which the effect of any order structure in the original data could be compared, the data were randomly re-ordered a large number of times, and the calculations repeated, thus presumably eliminating any time-series or autocorrelation effects on variance, skew, and kurtosis in the grouped data. The averaged distribution of these permuted-group scores showed no significant deviations in extreme values, compared with the empirical expectation embodying the overall mean shift and an appropriate correction for comparing the actual integral histogram against a continuous theoretical function.

Results

The following tables summarize the distribution characteristics of the 100,000 PEAR calibration trials, using serial sub-groupings in the original order of the trials.

Groups of 50

Scores	Ideal	Actual	Z-score		Statistics	
>2.326	20	18	-0.449	Mean	0.02053	p = .348
>1.645	100	106	0.616	Std. Dev.	0.99994	p = .996
>0	1000	1016	0.716	Std. Err.	0.02236	
<-1.645	100	93	-0.718	Skew	-0.05098	p = .352
<-2.326	20	18	-0.449	Kurtosis	0.03361	p = .758

Groups of 100

Scores	Ideal	Actual	Z-score		Statistics	
>2.326	10	9	-0.318	Mean	0.02903	p = .348
>1.645	50	51	0.145	Std. Dev.	0.97874	p = .348
>0	500	506	0.379	Std. Err.	0.03095	
<-1.645	50	34	-2.322	Skew	0.14429	p = .062
<-2.326	10	8	-0.635	Kurtosis	0.13720	p = .376

Groups of 1000

Scores	Ideal	Actual	Z-score		Statistics	
>2.326	1	2	1.005	Mean	0.09181	p = .348
>1.645	5	6	0.459	Std. Dev.	0.97674	p = .770
>0	50	53	0.600	Std. Err.	0.09767	
<-1.645	5	5	0.000	Skew	0.08326	p = .734
<-2.326	1	0	-1.005	Kurtosis	-0.21445	p = .662

Conclusion

The calibration data show no significant deviations in the distribution parameters for any serial group size. One of the 100-group extreme score comparisons is significantly large, but in the context of the total of 15 comparisons made, this cannot be distinguished from chance fluctuation. We conclude that no significant artifacts are imposed by the experimental group sizes.