APPLIED SCIENCE ASSOCIATES, INC.

POST OFFICE BOX 12422

RESEARCH TRIANGLE PARK NORTH CAROLINA 27709

ENGINEERING STUDIES RELATED TO THE SKYLAB PROGRAM

prepared under

NASA Contract No. NAS6-2307

for

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

WALLOPS STATION

WALLOPS ISLAND, VIRGINIA 23337

APPLIED SCIENCE ASSOCIATES, INC.
POST OFFICE BOX 1042
RESEARCH TRIANGLE PARK, NORTH CAROLINA 27709

ENGINEERING STUDIES RELATED TO THE SKYLAB PROGRAM

Task B Technical Report

July, 1973

by

George S. Hayne

prepared under

NASA Contract No. NAS6-2307

for

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

WALLOPS STATION

WALLOPS ISLAND, VIRGINIA 23337

Table of Contents

Section I

1.0 Introduction and Summary of Results

1.1 Introduction

This report summarizes the work performed under Task B of NASA Contract NAS6-2307. The purpose of the work was to characterize the relationship between the S-193 Automatic Gain Control (AGC) data and the magnitude of received signal power, which in essence is the ratio of the "peak of the mean waveform" to the "mean of the peak (individual) waveform values." In accordance with earlier S-193 program documents we use the abbreviation r-factor to describe this ratio. The r-factor will be less than unity, and will be a function of off-nadir angle, ocean surface roughness, and receiver signal-to-noise ratio (SNR). Of these items, the largest variation of r-factor is expected to be due to change of angle off-nadir. The effect of ocean surface roughness should be much less important and all work reported here is based on a quasi-flat (but diffuse scattering) ocean. The digital computer simulation used in the present work includes provision for additive receiver (white) noise, but all work to date has been for the zero-noise, infinite SNR case. As described in section 3.0, we specialized to the noise-free case because of the nature of calibration data supplied by the S-193 hardware contractor.

We considered the possibility of conducting either a hybrid simulation or an entirely-analog simulation as a way to estimate r-factor, but the difficulty in either of these approaches is approximately the same; it is difficult to adequately shape the range of input process waveforms to be encountered. Largely for this reason, we used an entirely digital simulation with the system model to be described in the following section. The main body of this report discusses the IF and video impulse response functions used, details of the input (expectation value) waveforms, and the results to date. An appendix provides a more specific discussion of the digital computer programs used.

1.2 Summary of Results

During the time in which this r-factor analysis was undertaken, it appeared that the Skylab altimeter contractor would provide all calibration data using deterministic (non-fluctuating) calibration waveforms and that r-factor would explicitly enter into the calibration and data reduction process. In the interim we learned that the contractor was instead attempting to simulate ocean scattered signals through: (1) use of noisy triangular waveshapes for the

on-nadir case, and (2) noisy rectangular approximations to the off-nadir waveforms. As a consequence our effort was redirected to yield a comparison of these waveforms with theoretically derived and digitally simulated waveshapes. Results to be presented indicate that a correction factor should be used with the r-factor calibration data obtained from these approximate waveforms.

There are other complications: Based on presently available data, the S-193 receiver was operating between -7 and 0° C during the SL-2 mission. All published calibration data was taken at other temperatures and the available data indicates system characteristics change drastically between test temperatures. A comparison of calibration data taken using deterministic and random waveforms (which exists only for the on-nadir case) shows r-factor to be about 0.7 at 5° C, and to be near unity at the other two extremes of temperatures.

We therefore conclude, based on our examination of the recent calibration data,* that the entire receiver including the agc loop, the agc attenuator, the boxcar detector, and so forth, appears to be grossly nonlinear outside the approximate ambient temperature range and for both low and high SNR values (although this latter nonlinearity is to be expected to some degree). We had originally expected that the variation of the r-factor with SNR would be an important effect; now these other phenomena are seen to provide more serious changes. For these reasons, the possibility of calibration data interpolation does not appear feasible. Should calibration data not be available near the S-193 receiver operating temperature, the only recourse is to record data on the back-up hardware, as an assessment of the accuracy of interpolated data.

The second difficulty is in the shape of the waveforms extracted from presently available quick-look data. If further experimental data studies (from SL-2, 3, and 4) show the recorded waveforms to seriously depart from the theoretically computed waveshapes discussed herein, then r-factor should be re-computed for these waveshapes using the digital programs documented in the Appendix.

In summary, we recommend that the S-193 calibration data be compensated to reflect r-factor perturbations due to waveshapes used in the calibration

---

*"S-193 Microwave Radiometer/Scatterometer Altimeter," Calibration Data Report, Vol. IB, Revision D, Contract NAS9-11195, GE, 22 March 1973.

process. We specifically recommend that the corrections shown in Fig. 6 of this report be incorporated into S-193 data reduction procedures, subject to (1) further examination of measured and computed waveshapes and r-factor recomputation as required, and (2) acquisition and study of calibration data near the actual operating temperature to better understand the implied nonlinear receiver behavior with temperature.

Section II

2.0 Digital Simulation of r-factor Relationship

2.1 System Model

Figure 1 (which is intended to be largely self-explanatory) shows the overall system model used to estimate r-factors through a digital simulation process. The input signal S(t) is the "input waveform" as discussed in a subsequent section; S(t) is time varying over times relatively long compared to the IF and video response times. For the digital model, the convolutions become finite sums of products with the number of terms in the sums chosen on the basis of the quantized time increment and on the rate of decay of the impulse response function h(t); the choice of number of terms for the present case is discussed in the second section.

The model shows provision for input of both a "signal" S(t) and a noise N, and our computer program includes the noise possibility, but all our work to date has been done for the case of N = 0 (SNR = ∞).

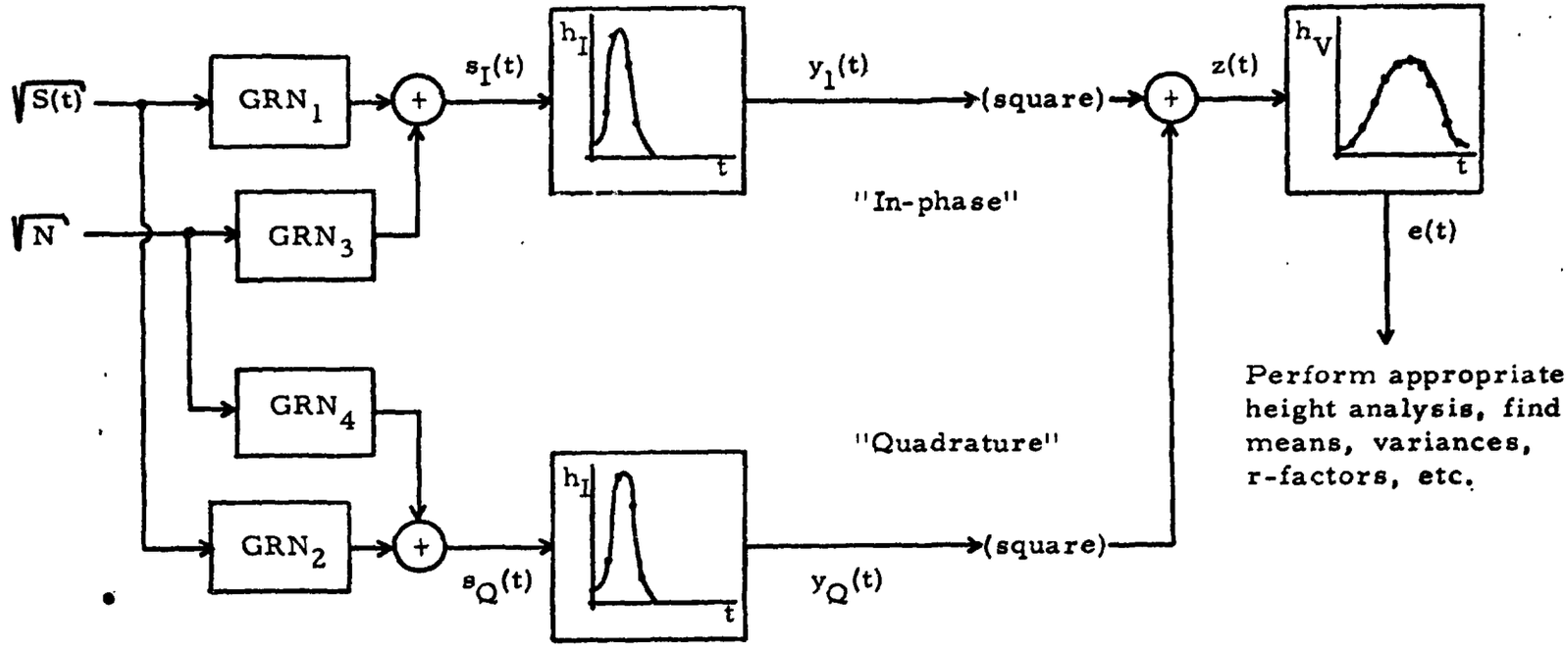
By keeping the outputs GRN_3 and GRN_4 both zero, and by replacing GRN_1 and GRN_2 of Fig. 1 with unit transfer devices (ones for which the output equals the input), the model will produce a "deterministic output." This is important for verifing that the system output expectation value has the desired shape. There are two ways of obtaining this output expectation value of course; either by use of the deterministic model or by running the random model for infinitely long times. Clearly the first is preferable for finite computation budgets. The necessary modification to produce the deterministic model merely involves replacement of the Gaussian random number subroutine GAUSS (see discussion in the Appendix) by a "fake" GAUSS which returns the input as output when called by the rest of the program.

2.2 Impulse Response Functions $h_I(t)$ and $h_V(t)$:

The exact IF and video impulse response functions (h_I and h_V) depend upon details of the S-193 circuitry which are not readily available; however, the central limit theorem assures that if one has a number of individual time functions, $f_1(t)$, $f_2(t)$, ... , $f_n(t)$ which are in general non-negative,

$$f_i(t) \geq 0,$$

then under fairly general conditions, their convolution will approach a normal curve,



$GRN_1, \ldots, GRN_4$ = independant Gaussian Random Number Generators, zero-mean, with output variance equal to square of input signal

h_I, h_V = IF, Video impulse response functions

$y = s * h_I$ (convolution)

$z = y_I^2 + y_Q^2$

$e = z * h_V$

Figure 1. System Model Used for r-factor Simulation Results

$$f(t) = f_1(t) * f_2(t) * \cdots * f_n(t)$$

$$\simeq \frac{A(0)}{\sigma\sqrt{2\pi}} \; e^{-(t-\eta)^2/2\sigma^2}$$

where

$$A(0) = \int_{-\infty}^{\infty} f(t)dt$$

$$\eta = \eta_1 + \eta_2 + \cdots \eta_n$$

$$\sigma^2 = \sigma_1^{\,2} + \sigma_2^{\,2} + \cdots \sigma_n^{\,2}$$

and where η_i, $\sigma_i^{\,2}$ are the individual mean and variance of $f_i(t)$. Thus for a system comprising several stages $f_i(t)$ in cascade, a normal or Gaussian impulse response function becomes a progressively better overall system description as the number of stages becomes greater.

This is discussed briefly by A Papoulis,* with an example showing that for a cascade of four identical one-pole systems each having response

$$h_i(t) = e^{-\alpha t}U(t)$$

$$H_i(\omega) = \frac{1}{\alpha + j\omega} \quad , \quad i = 1, \cdots 4,$$

the overall system response $h(t) = h_1(t) * \cdots * h_4(t)$ is given approximately by

$$h(t) \simeq \frac{1}{2\sqrt{2\pi}} \frac{e^{-(\alpha t-4)^2/8}}{\alpha^3} \quad ,$$

with a system function $H(\omega) = H_1(\omega) \times \cdots \times H_4(\omega)$ given approximately by

$$H(\omega) \simeq \frac{1}{\alpha^4} e^{-j4\omega/\alpha} \; e^{-2\omega^2/\alpha^2} \quad .$$

The $e^{-j4\omega/\alpha}$ is of course the phase shift associated with the h(t) Gaussian being centered at $t_o = 4/\alpha$.

---

*"Systems and Transforms with Applications in Optics", New York; McGraw-Hill, 1968, pp. 78-81.

For our problem, we are not too concerned with where the time origin of the problem lies, and we will assume an impulse response function of the form $h(t) = Ne^{-\alpha^2(t-t_o)^2/8}$, with N an amplitude normalization constant to be determined later. The time shift to will be chosen merely for convenience, and α will be determined from the quoted width of the IF or video filter.

As described in the Appendix discussing the computer program, the functional form used for the impulse response function is the shifted Gaussian form,

$$h(t) = Ke^{-(t-t_o)^2/\tau^2} ,$$

and we need to express this τ in terms of α and to express α in terms of the 3dB width of the video response. The time shifts or phase shifts are of no particular importance to the r-factor simulation, and so setting $t_o = 0$ temporarily,

$$e^{-(\alpha t)^2/8} = e^{-t^2/\tau^2}$$

$$\text{or } \tau = \sqrt{8}/\alpha.$$

It is easy to show (based on tabulated values for the normal distribution function) that this leads to

$$\alpha = \frac{4\pi f_c}{0.831} ,$$

$$\text{or } \tau = \frac{0.831\sqrt{8}}{4\pi} \frac{1}{f_c}$$

where f_c is the system bandwidth to be simulated. For the video bandwidth of 5 MHz, this yields

$$\tau_V = 37.36 \text{ nanoseconds},$$

where the subscript V denotes video (as the subscript I will denote IF). The IF bandwidth is taken as twice the video, so that

$$\tau_I = 18.68 \text{ nanoseconds}.$$

The video response will dominate the system, and as we are going to approximate the continuous video impulse response function $h_V(t)$ by a finite set of weights h_{V_i} spaced a distance δt apart, δt for the video weights (and

the same δt is to be used for the IF h_I as well as the input waveform) must be chosen small enough to provide several time samples within the interval τ_V. For the work reported, we used $\delta t = 10$ nanoseconds.

The number of weights h_I and h_V can be chosen by discarding all weights of amplitude less than 2% of the greatest weight; the time shifts t_{o_V} and t_{o_I} are chosen simply for convenience. Under these various criteria, the following quantities were used (in addition to the δt, τ_I, and τ_V already given):

$$t_{o_V} = 80 \text{ nanoseconds}, \quad n_V = 17$$

$$t_{o_I} = 40 \text{ nanoseconds}, \quad n_I = 8.$$

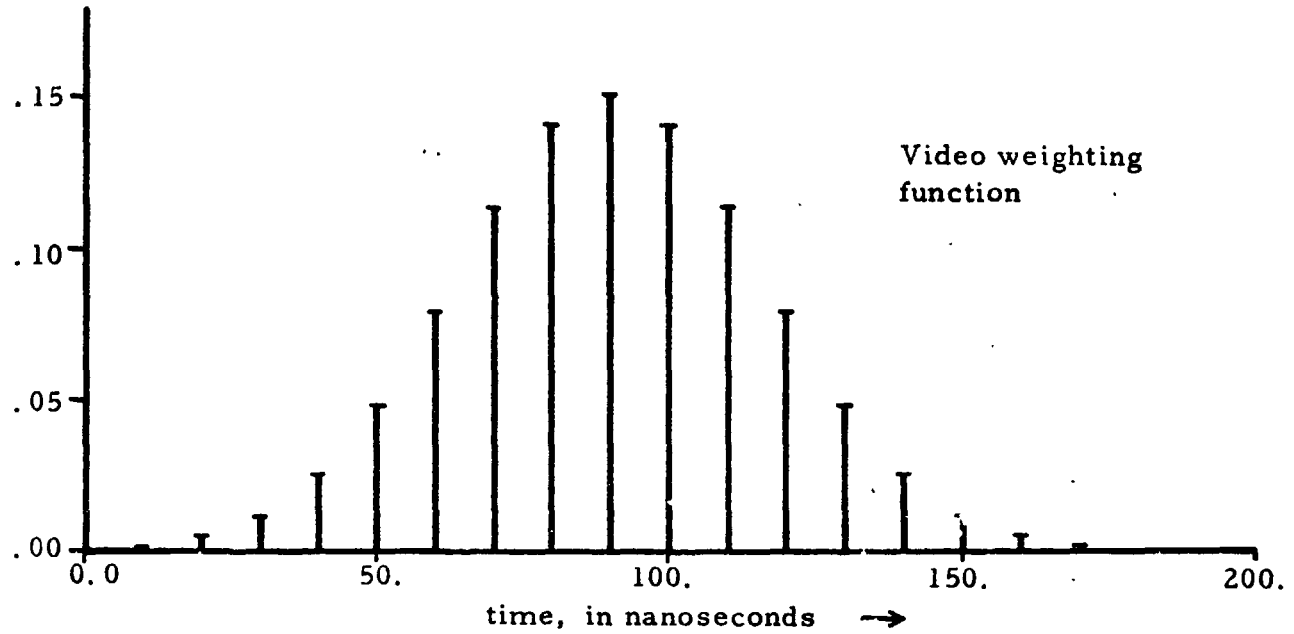
Figure 2 shows the video and IF response functions which result from this choice of input parameters.

2.3 Input Waveforms

For the input waveforms employed, we refer to a February, 1973, Research Triangle Institute (RTI) report by Miller, Brown, and Hayne, "Engineering Studies Related to Geodetic and Oceanographic Remote Sensing Using Short Pulse Techniques," Contract No. NAS6-2135; in particular, see Figures on pages 2-46 to 2-50 together with the discussion of these figures. We have required that the input waveforms in this present work reproduce those waveforms of the RTI February, 1973, report (apart from possible amplitude and time-origin differences) when the input waveforms are run in the "deterministic" program.

Waveforms shown in this report (hereafter referred to simply as "RTI waveforms") include composite IF and video bandwidth effects; therefore, direct use of these waveforms in the present simulation program would have the effect of including the IF and video characteristics twice. This is significant only for the 0° and 0.5° off-nadir waveforms, and those have been recalculated with IF and video bandwidth effects removed. For all other angles, the waveshape effect of the IF and video is a simple time-shift with imperceptible changes in shape. Figure 3 shows the different input waveforms of the present study; all have been scaled to the same peak value and plotted versus a logarithmic time scale. This figure serves primarily to emphasize the different running time requirements of different off-nadir angles.

We chose to include the trailing edge of each waveform out as far as 1/10 maximum value. For $\xi = 0°$ this is achieved by 700 nanoseconds, whereas the



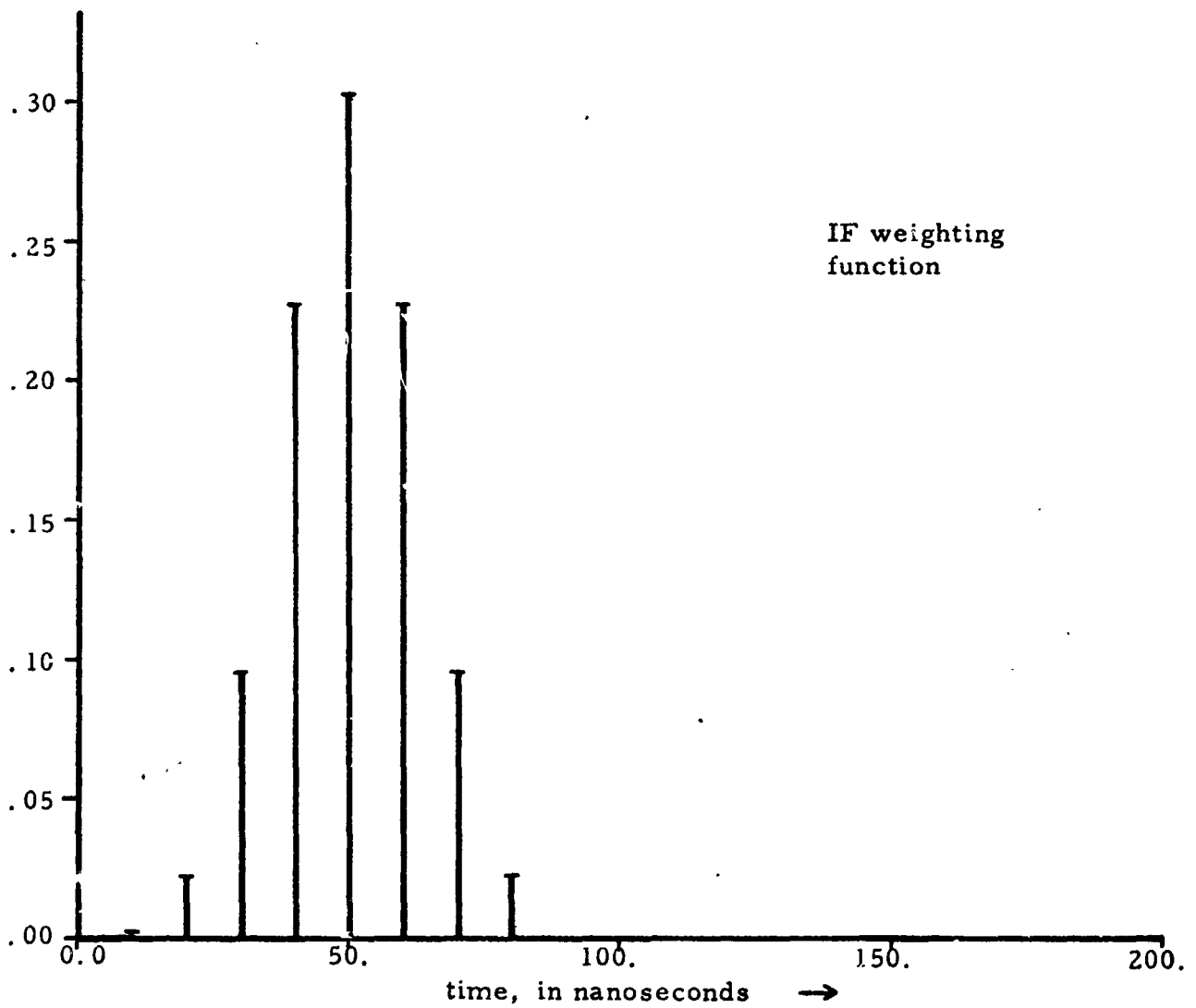


Figure 2. Impulse Response Weighting Functions vs. Time

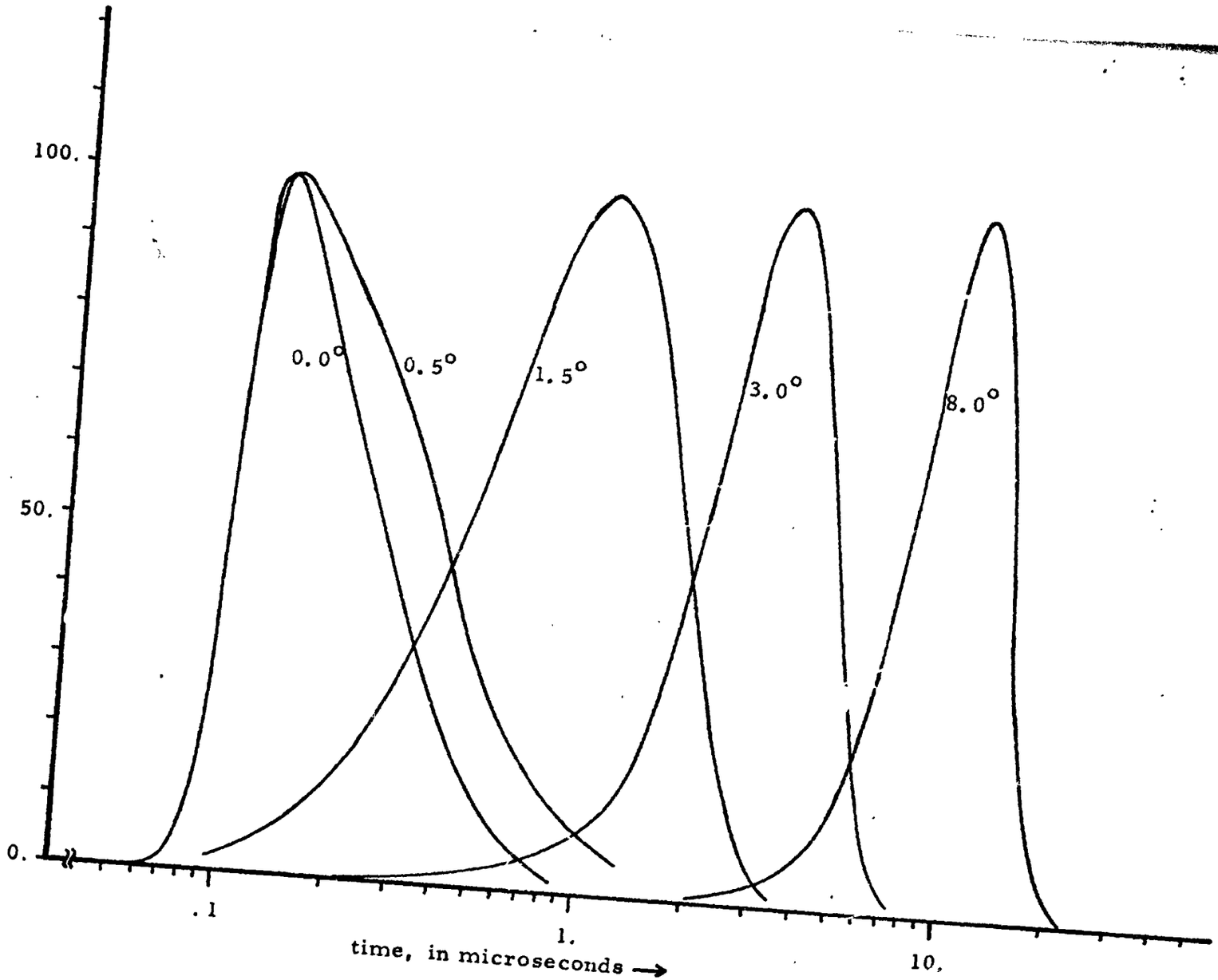


Figure 3. Expected Mean Waveforms vs. Time

$\xi = 8°$ case requires about 20 microseconds to decay enough; thus since DT is fixed by hν considerations (with a value of 10 nanoseconds for our work), the $\xi = 8°$ case will require $20/.7 \approx 29$ times the running time of the $\xi = 0°$ cases.

Results were also obtained for a triangular waveform (100 nanosec rise - 500 nanosec fall) and for rectangular waveforms of width 0.4, 1.0, 4.0, and 14.0 microseconds; these waveforms were used by GE in measuring results for the most recent S-193 calibration data report, and our r-factor results for RTI waveforms compared to GE triangular or rectangular waveforms will provide a set of corrections to the data corrections indicated by the GE report.

Section III

3.0 Simulation Results

3.1 Discussion

The previously discussed r-factor programs have been run for both the RTI waveforms and the GE approximation waveforms (triangular or rectangular) for the off-nadir angles 0, 0.5, 1.3, 3.0, and 8.0 degrees with at least 250 simulated pulses averaged for each case.

Figure 4 shows a typical single pulse result from the simulation for 1.5° off-nadir; the time increment DT = 10 nanoseconds used for these computations is explicitly shown in the figure. Taking a smaller DT would increase computer running times, and the figure suggests that 10 nanoseconds may be a reasonably good choice. Figure 5 shows the average of 50 individual pulses for the 1.5° case. Notice that the vertical scale is different in the single-pulse and the 50 pulse figures. The peak of the mean return occurs at 1110 nanoseconds for this 1.5° case.

The results of our runs are presented by Table I. The table also compares the r-factors from the RTI waveforms and from the GE approximation waveforms (triangular or rectangular) to derive the correction factor to apply to GE's r-factor correction as discussed in the next section.

Figure 6 shows the same results which are tabulated in Table I. The errors indicated (by $\pm$ in the table and by bars in the figure) are estimates of (plus or minus) one standard deviation of the mean r-factor as given. Note that the correction factor shown in Figure 6 is not an easily describable or monotone process. This is because there is no particular relationship between the RTI waveforms and the rectangular approximants used by GE. For example, the 3dB width of the computed waveforms may match the width of the rectangular waveform at one off-nadir angle and not at another.

Figure 7 reproduces Figure 2B4, pg. 2-58 of the RTI February, 1973, report with the present r-factor results added to the figure; this is to make easier the comparison with the earlier r-factor work. While we present our current results as being of an interim nature, and capable of improvement with additional running time, this digital simulation method is flexible and powerful enough that the present results should be better than any previous computed or simulated results.

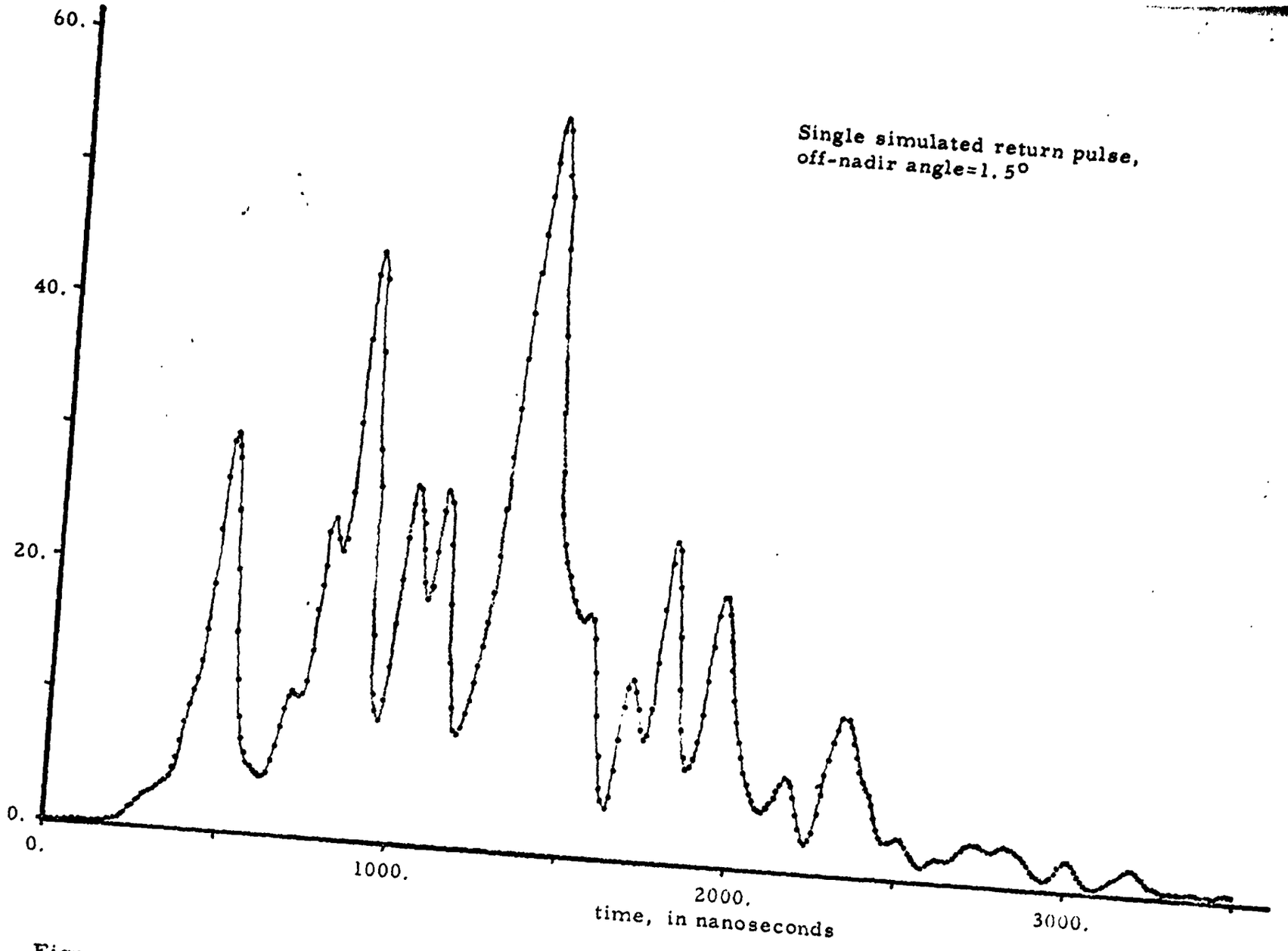


Figure 4. Single Return Pulse vs. Time

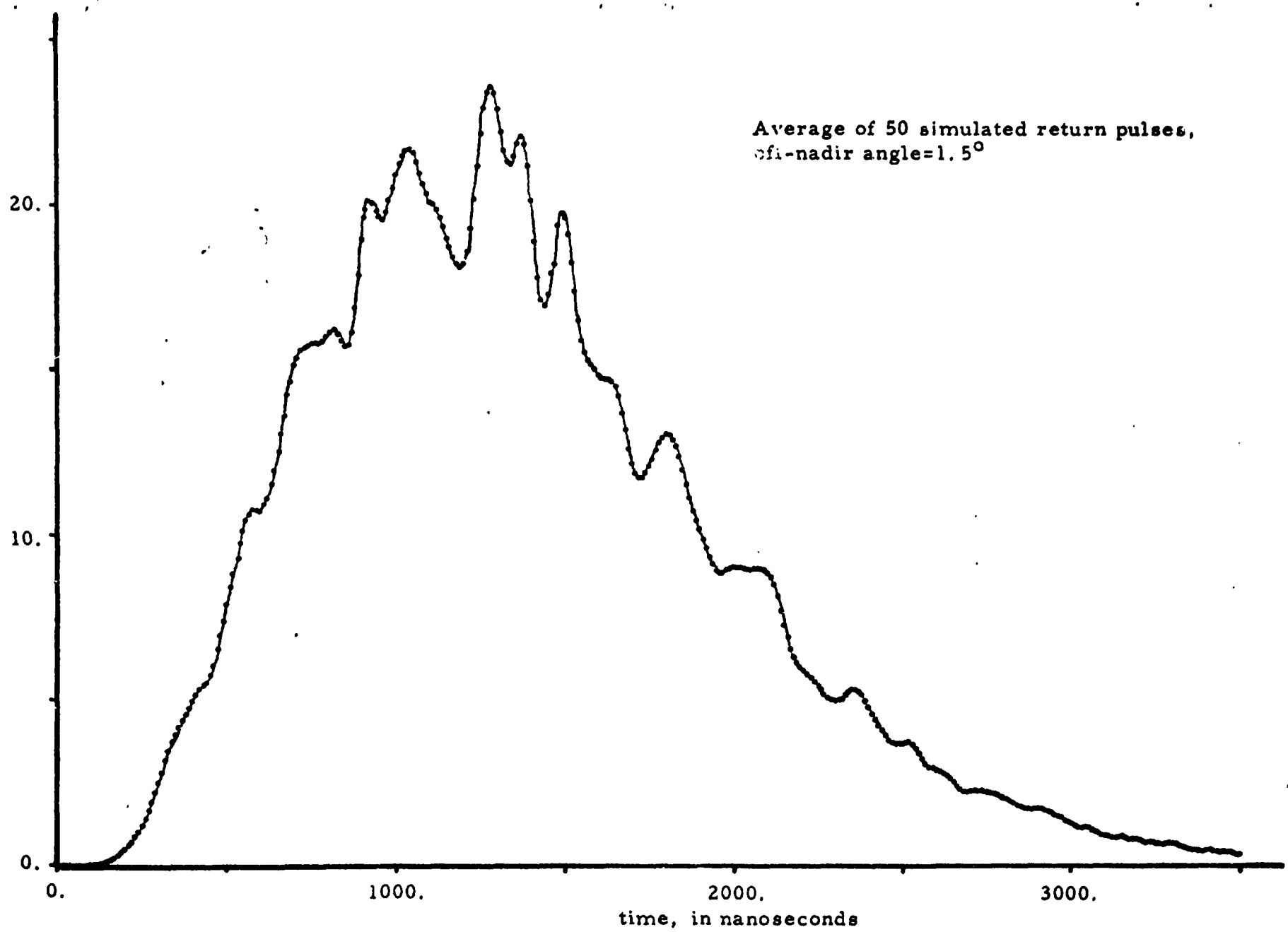


Figure 5. Average of 50 Return Pulses vs. Time

Table I. Summary of Mode II Simulation Results

(Submode) DAS/ξ	Waveform input	r-factor	Ratio of r-factors, expected ÷ GE simplified
(Submode 0)	RTI-calculated	0.748 ± .020	1.128 (+0.52dB)
DAS-1/ξ = 0°	GE triangular	0.664 ± .009	
(Submode 1)	RTI	0.684 ± .011	1.328 (+1.23dB)
DAS-2/ξ = 0.5°	GE 400 nanosec rectangular	0.515 ± .009	
(Submode 2)	RTI		
DAS-3/ξ = 15.6	GE rectangular		
(Submode 3)	RTI	0.329 ± .014	1.306 (+1.16dB)
DAS-4/ξ = 8°	GE 14 μsec rectangular	0.252 ± .010	
(Submode 4)			
DAS-5/ξ = 3°	GE μsec rectangular	0.304 ± .019	1.290 (+1.10dB)
(Submode 5)	RTI	0.457 ± .016	1.124 (+.51dB)
DAS-6/ξ = 1.5°	GE 1 μsec rectangular	0.406 ± .011	

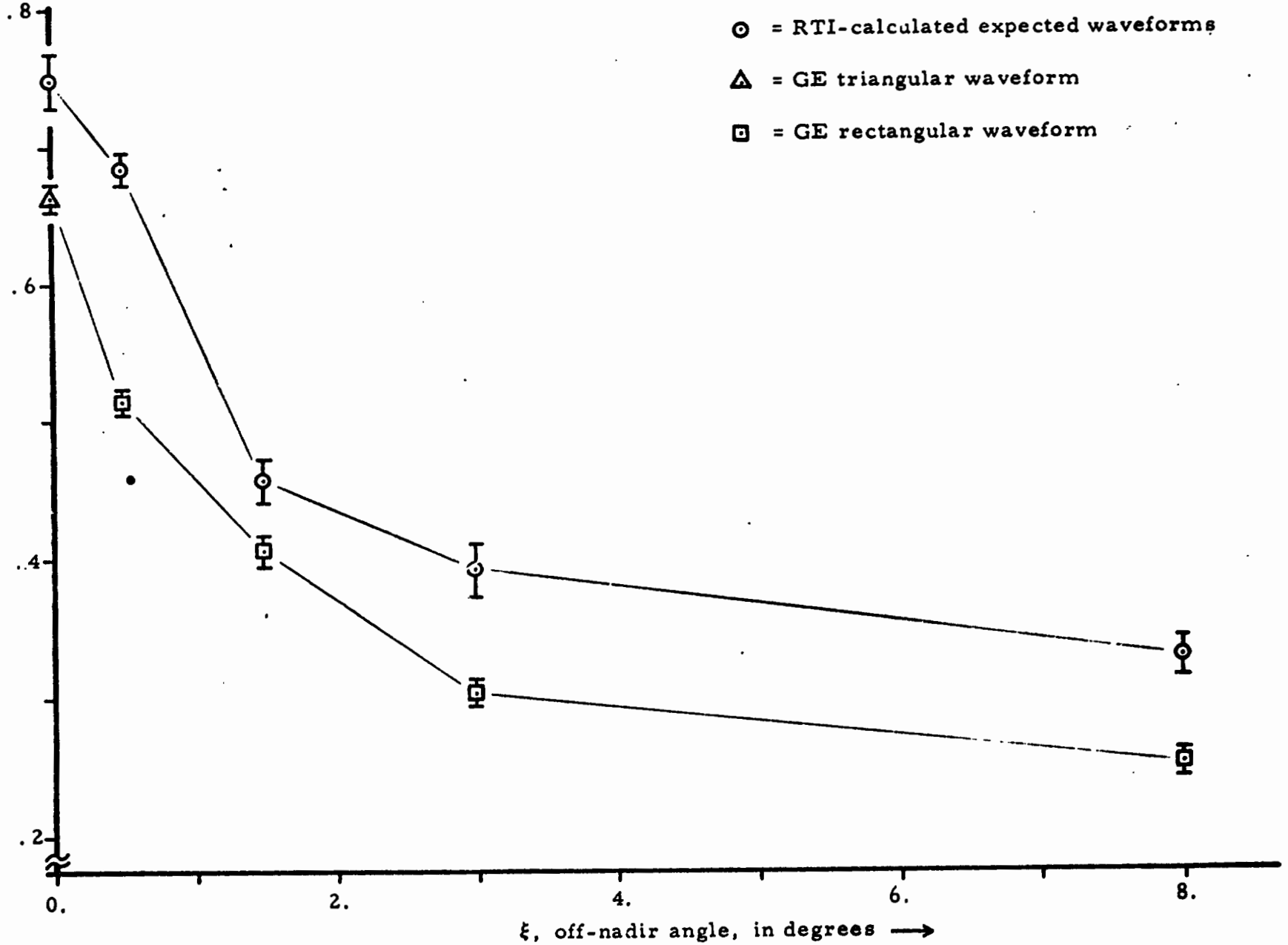


Figure 6. Results of r-factor Computation vs. Off-nadir Angle

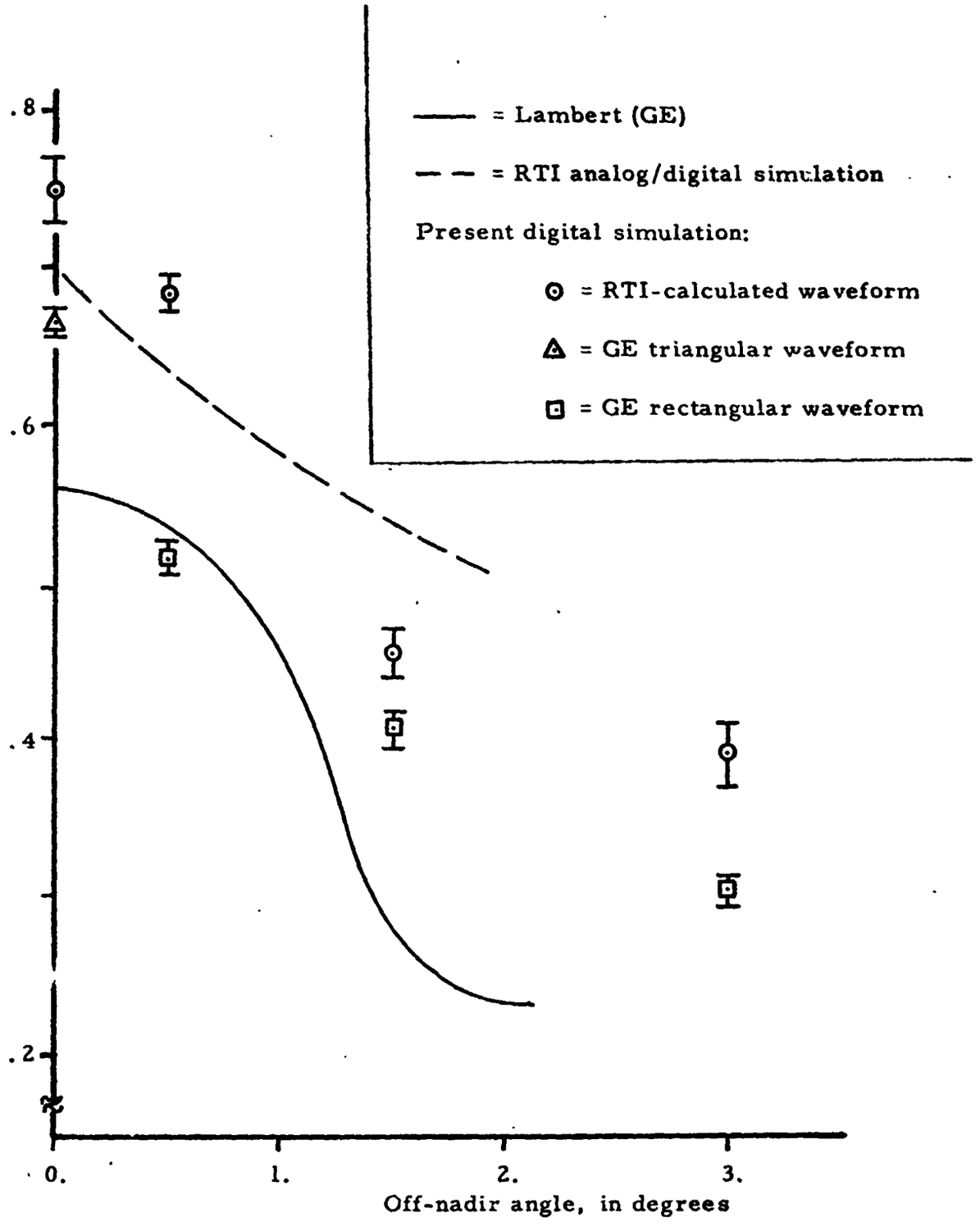


Figure 7. Comparison of Present and Earlier r-factor Results vs. Off-nadir Angle

Appendix

Details of Digital Computer Simulation for r-factor Estimation

General Discussion

A program listing is supplied here and a punched source deck is being separately provided for a Fortran IV computer program which is a straightforward implementation of the r-factor estimation model supplied in the main body of this report. Apart from a subroutine GAUSS, this program should be directly transferable to NASA, Wallops Island. We discuss GAUSS in the following two paragraphs, and then return to the general discussion of the program logic.

At the heart of the process is a subroutine GAUSS (IX,S,AM,V) which computes a normally-distributed random variable V, of mean AM and standard deviation S, at each subroutine call. The integer variable IX must contain an odd integer with nine or fewer digits on the first call to GAUSS; thereafter, each call to GAUSS uses the value of IX returned from the immediately previous call. The variable IX is included in the output from the simulation program so that each additional computer run can resume the random number sequence at the point at which the last previous run terminated; without the ability to start each new run at the last used value of IX, one would in effect be using the same sequence of random numbers over and over rather than moving to later in the sequence (this sequence length is at least $2^{31} \div 12$ for our computation).

Subroutine GAUSS uses another subroutine RANDU(IX,IY,YFL); RANDU provides a floating-point random number YFL which is uniformly distributed on the interval (0,1). Every call to GAUSS actually uses RANDU twelve times, approximating a Gaussian-distributed random variable by the sum of twelve uniformly distributed random variables. Source listings for GAUSS and RANDU are also provided. Both GAUSS and RANDU have been taken from the IBM Scientific Subroutine Package, Version III, and further details and comments are available in the publication, IBM Form H20-0205, "System/360 Scientific Subroutine Package (360A-CM-03X) Version III Programmer's Manual," (White Plains, New York, IBM Technical Publications Department, 1968). Subroutine RANDU is machine-dependent, and is specific to the word length of the IBM 360 series computer. To transfer the r-factor program to NASA, Wallops Island, subroutine GAUSS will have to be replaced by the subroutine RAND(AVG,VAR,N,X) which already exists at Wallops Island. Since RAND computes an array X(N) of N normally-distributed

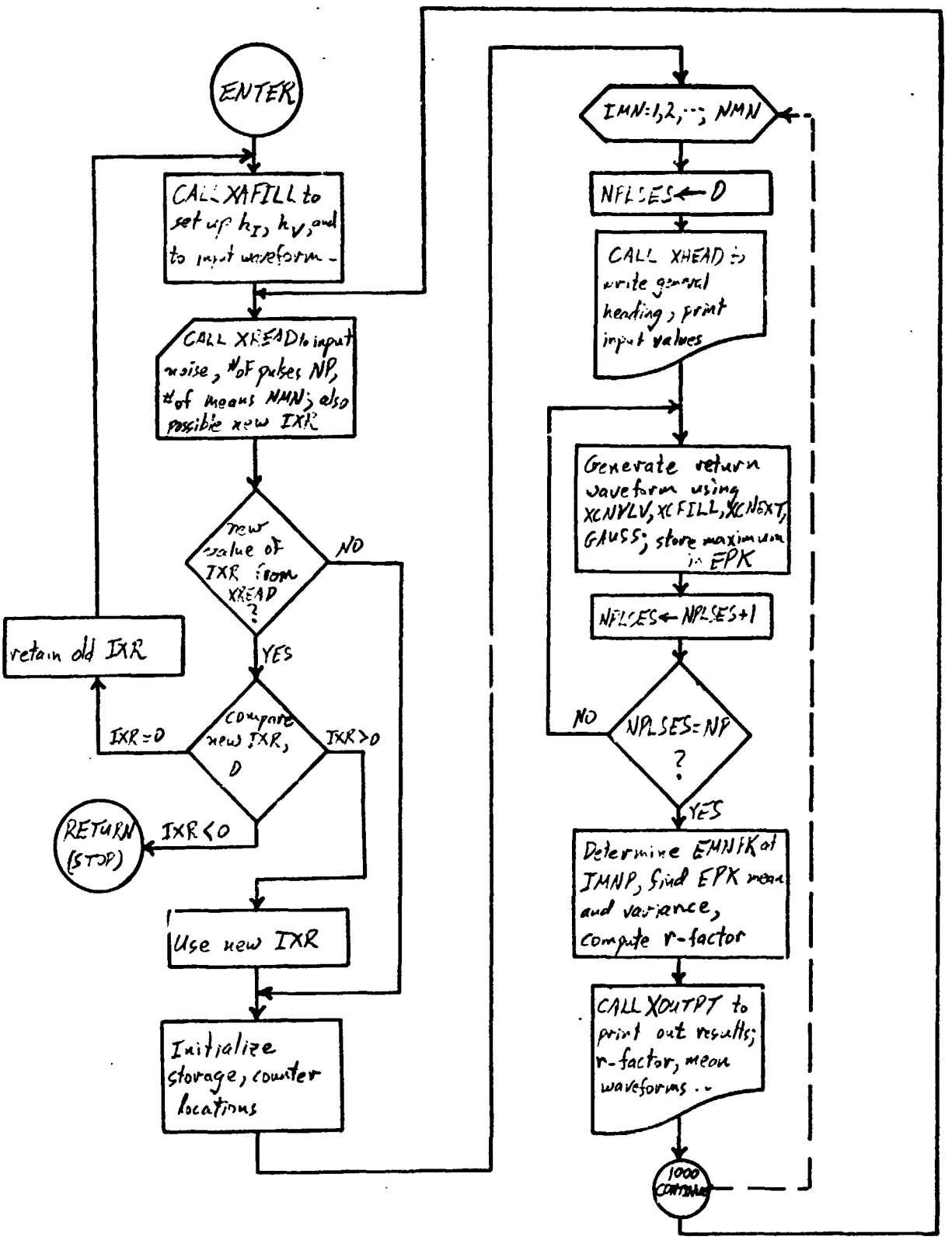


Figure A1. Logic Flow Diagram for Subroutine XSIMUL.

random variables of mean AVG and variance VAR, whereas our program calls only a single number at a time, it would probably be appropriate to rewrite RAND, renaming it GAUSS and using the same argument list as the present GAUSS. This new GAUSS can then use the present Wallops subroutine RANDOM for its uniformly-distributed random numbers.

Returning to discussion of the overall program, the subroutine XSIMUL performs the bulk of the work with a part of XSIMUL being written separately as XCNEXT. The digital convolution required is performed by XCNVLV; a separate entry point XCFILL assists in filling arrays for convolutions. The subroutine XAFILL sets up the IF and video impulse response weights, and also calls subroutine EPTFIL which provides the input waveforms. Subroutine XREAD provides input data for each run, and the output for results is handled by subroutines XHEAD and XOUTPT.

A general logic flow diagram for XSIMUL is given by Figure A-1. The following paragraphs will discuss additional details of subroutines XAFILL and EPTFIL, and then the general form of program data input will be summarized.

Impulse Response Weights Computed by XAFILL

Subroutine XAFILL computes a set of N weights h_i from the expression

$$h_i = K \exp\left[- \frac{(t_i - t_o)^2}{\tau^2} \right], \quad i = 1,2, \cdots, N$$

$$\text{where } t_i = (i-1)\delta t$$

$$\text{and } K = 1 / \sum_{i=1}^{N} h_i .$$

Thus, h_i are derived from a Gaussian function centered at t_o and having a width τ. The normalization factor K is chosen to provide an output equal to the input when the input is held at a constant value for a time at least as long as $N\delta t$. The various relationships in $h_i(t)$ are shown on the following page.

Input quantities clearly are τ, t, δt, and N. These variables in XAFILL are TAU, TO, DT, and N, respectively. Since XAFILL sets up both the IF and the video response functions, these are distinguished by the labels I and V, and the following correspondence list results:

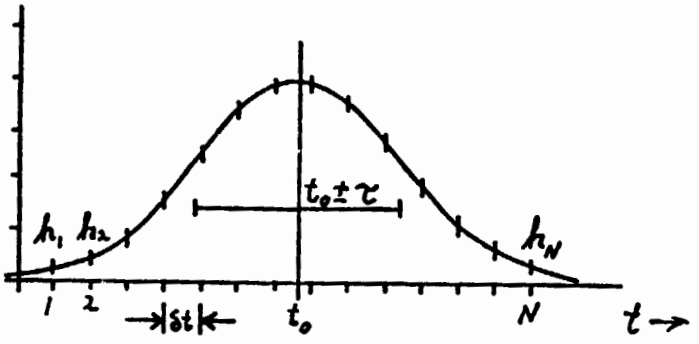


Figure A2. Sketch Showing Relationship Between τ, t_o, and $h_i(t)$, i=1, 2, . . . , N.

Quantity	XAFILL Variables
H_{I_i}, i=1, ··· , N_I	HI(I), I=1, ··· , NHI
t_{o_I}	TOI
τ_I	TAUI
h_{V_i}, i=1, ··· , N_V	HV(I), I=1, ··· , NHV
t_{o_V}	TOV
τ_V	TAUV
δt	DT

The time increment DT is common to both the IF and video characteristics, and is supplied from subroutine EPTFIL called by XAFILL; the remaining quantities TOI, TOV, TAUI, TAUV, NHI, and NHV are ready by XAFILL as LIST1 using the NAMELIST procedure available to some Fortran IV compilers. Much of our computation has been done using a Teletype ASR-33 as a time-sharing terminal, and input via the NAMELIST avoids the awkwardness of formatted data input from a teletype.

Waveform Input From Subroutine EPTFIL

The purpose of EPTFIL is to fill an array EPT(NE) (labelled EXP(NE) in XSIMUL) with discrete values from the input waveform. (Actually because the input signal sequence gets squared by the square-law detection, the square-root of the input waveform is loaded into the array.) The time increment DT was chosen as 10 nanoseconds from consideration of the video response, and the number of points to be put into EPT(I) will depend upon the time extent of the input waveform's region of interest. This varies from about 700 nanoseconds for the on-nadir result to about 30 microseconds for an off-nadir angle of 8°; this latter time corresponds to 3000 points in EPT(I). It is clearly unnecessary to read in 3000 values from a previously computed waveform, and instead we enter 20 sample values from the input waveform and interpolate between 20 input data point pairs by use of a spline-fit performed by subroutine ESPLIN.

The input y vs. t waveform is entered in the arrays TIN(I) and YIN(I), I=1, 20. It is necessary that the TIN(I) values be in ascending order, that

TIN(1) be less than or equal to DT, and that TIN(20) be greater than or equal to (NE x DT) where NE is the total number of points in EPT. YIN and TIN are read into EPTFIL through LIST1A as are NE and IMNP. The integer IMNP is the index at which the "peak of the mean" is to be found for the r-factor computation. IMNP will be larger than the value of J for which EPT(J) has its maximum because of the time-delay of the IF and video characteristics; for the IF and video values of our work this index shift was around 11 corresponding to a time-shift of about 110 nanoseconds.

To simulate the triangular and rectangular waveforms used by GE for the Skylab agc calibration data, a simpler replacement EPTFIL subroutine was written to provide these simple waveforms directly rather than through use of a spline-fit. No listing has been provided for this trivial alternative EPTFIL.

Summary of Data Input to r-factor Program

The discussion of XAFILL has already pointed out the use of the Fortran NAMELIST feature. Data input to the program consisted first of LIST1 providing the parameters to calculate impulse response weights HI(NHI) and HV(NHV), then second LIST1A setting up the input waveform, and then LIST2 to be described here.

LIST2, as read by subroutine XREAD, inputs the program quantities IXR, RN, NN, NP, NMN, and MOS. First, NN and MOS are for program features not used in this study; NN should be set to zero at the first use of LIST2 and need not be entered thereafter, and MOS need not ever be entered. The integer IXR is for the random number generator. If no IXR value is entered, the program will continue with the current value, starting initially with IXR=4321. A positive IXR in will replace the current IXR, a negative IXR in will cause subroutine XSIMUL to return to the main calling program (and there to STOP), and a zero entered for IXR will cause a jump back to XAFILL (to permit changing HI, HV, or the input waveform if desired). Usually, one would wish in the first LIST2 to explicitly enter the value of IXR last appearing in the printout of the last previous program run, and to leave IXR off all subsequent LIST2 data in a given run.

The value RN allows additive receiver noise to be simulated. All our results to date have been for RN=0; once RN is set to zero in LIST2, it need not be entered in subsequent LIST2 calls. The remaining quantities NP and

NMN are the number of pulses to be averaged for one mean waveform, and the number of separate mean return waveforms to be run.

```
C   MAIN ROUTINE CALLING DIGITAL SIMULATION SUBROUTINE
      DIMENSION ET(3000),EXP(3000),TPYI(100),TPYQ(100),HI(100),
     1 TPYV(100),HV(100)
      NE=3000
      CALL XSIMUL(ET,NET,EXP,NE,TPYI,TPYQ,HI,NHI,TPYV,HV,NHV)
      WRITE (3,1000)
 1000 FORMAT( / / '0*****NORMAL END (GSH)*****' )
      STOP
      END
C
C
C
      SUBROUTINE XCNVLV(YI,XI,JXY,TEMP,H,NH)
      DIMENSION TEMP(NH),H(NH)
      YI=0.
      JP=NH+1
      NP=MOD(JXY-1,NH)
      IF (NP) 100,100, 00
  100 TEMP(NH)=XI
      DO 110 I=1,NH
      M=JP-I
  110 YI=YI+TEMP(I)*H(M)
      RETURN
  200 TEMP(NP)=XI
      K=NH-NP
      DO 210 I=1,K
      J=I+NP
      M=JP-I
  210 YI=YI+TEMP(J)*H(M)
      JP=NP+1
      DO 220 I=1,NP
      J=I+K
      M=JP-I
  220 YI=YI+TEMP(I)*H(M)
      RETURN
C
C
      ENTRY XCFILL(XI,JXY,TEMP,NH)
      NP=MOD(JXY-1,NH)
      IF (NP) 300,300,400
  300 TEMP(NH)=XI
      RETURN
  400 TEMP(NP)=XI
      RETURN
      END
C
C
      SUBROUTINE XSIMUL(ET,NET,EXP,NE,TPYI,TPYQ,HI,NHI,TPYV,HV,NHV)
      DIMENSION ET(NET),EXP(NE),TPYI(NHI),TPYQ(NHI),HI(NHI),TPYV(NHV),
     1 HV(NHV)
```

```
      IMNPK=26
      IXR=4321
    3 CALL XAFILL(HI,NHI,HV,NHV,EXP,NE,IMNP)
  100 KXR=IXR
      WRITE (3,5) IXR
    5 FORMAT('AT START,IXR=',I10//)
      CALL XREAD(IXR,RN,NN,NP,NMN,MOS,NE,NET)
      IF (IXR) 105,107,110
  107 IXR=KXR
      GO TO 3
C---------INPUT OF IXR=0 WILL CONTINUE CURRENT IXR AND JUMP BACK TO
C---TO XAFILL;IXR .LT. 0 CAUSES RETURN FROM XSIMUL, AND IXR .GT. 0
C--- ENTERS THE NEW IXR. (DON'T USE IXR=-37)
  105 RETURN
  110 CONTINUE
      DO 1000 IMN=1,NMN
      CALL XHEAD(IXR,RN,NN,NP,NMN,MOS)
      NOISES=0
      XPK=0.
      XPK2=0.
      DO 200 I=1,NET
  200 PT(I)=0.
  250 DO 300 I=1,NHV
  300 TPYV(I)=0.
      IST=1
      EPK=0.
      IF (RN-.00001) 360,360,330
  330 DO 700 I=1,NHI
      CALL GAUSS(IXR,RN,0.,V)
      TPYI(I)=V
      CALL GAUSS(IXR,RN,0.,V)
  700 TPYQ(I)=V
      JXY=0
      DO 800 I=1,NHV
      JXY=JXY+1
      IF (RN-.00001) 797,797,798
  797 SI=0.
      SQ=0.
      GO TO 799
  798 CONTINUE
      CALL GAUSS(IXR,RN,0.,SI)
      CALL GAUSS(IXR,RN,0.,SQ)
  799 CONTINUE
      CALL XCNVLV(XI,SI,JXY,TPYI,HI,NHI)
      CALL XCNVLV(XQ,SQ,JXY,TPYQ,HI,NHI)
      XC=XI*XI+XQ*XQ
  800 CALL XCFILL(XO,JXY,TPYV,NHV)
      IF (NN) 450,450,850
  850 SG=0.
      DO 900 I=1,NN
```

```
      JXY=JXY+1
  900 CALL XCNEXT(ET,NET,IET,EPK, SG,RN,IXR, JXY,TPYI,TPYQ,HI,NHI,
     1 TPYV,HV,NHV, MOS)
      GO TO 450
  360 DO 400 I=1,NHI
      TPYI(I)=0.
  400 TPYQ(I)=0.
      JXY=0
C--------SEE NEXT COMMENT
      IF (NN) 450,450,405
  405 DO 435 I=1,NN
      IF (MOS-1) 410,410,415
  410 KET=IET
      GO TO 430
  415 KET=IET/MOS
      IF (MOD(IET,MOS)) 435,430,435
  430 ET(KET)=0.
  435 IET=IET+1
C--------THIS SET OF STATEMENTS KEEPS INDEXING SAME AS FOR RN .GT. 0
  450 DO 500 I=1,NE
      JXY=JXY+1
      SG=EXP(I)
  500 CALL XCNEXT(ET,NET,IET,EPK, SG,RN,IXR, JXY,TPYI,TPYQ,HI,NHI,
     1 TPYV,HV,NHV, MOS)
      NPLSES=NPLSES+1
      XPK2=XPK2+EPK*EPK
      XPK=XPK+EPK
      IF (NPLSES-NP) 250,550,550
  550 XNP=NPLSES
      EMNPK=0.
      EPK=XPK/XNP
      VEPK=XPK2/XNP-EPK*EPK
      DO 600 I=1,NET
  600 ET(I)=ET(I)/XNP
      EMNPK=ET(IMNP)
C----LATER PUT TAPE RECORD PROVISION HERE OR IN XOUTPT
      CALL XOUTPT(ET,NET,EXP,NE,EMNPK,EPK,VEPK,NPLSES,IXR)
 1000 CONTINUE
      GO TO 100
      END
C
C
      SUBROUTINE XCNEXT(ET,NET,IET,EPK,  SG,RN,IXR,  JXY,
     1 TPYI,TPYQ,  ,NHI,  TPYV,HV,NHV,  MOS)
      DIMENSION ET(NET),TPYI(NHI),TPYQ(NHI),HI(NHI),TPYV(NHV),HV(NHV)
C
C   XCNEXT "STEPS ALONG" TO THE NEXT OUTPUT POINT TO BE COMPUTED
      IF (SG-.00001) 2,2,3
    2 SI=0.
      SQ=0.
```

```
      GO TO 4
    3 CALL GAUSS(IXR,SG,0.,SI)
      CALL GAUSS(IXR,SG,0.,SQ)
    4 IF (RN-.00001) 10,10,5
    5 CALL GAUSS(IXR,RN,0.,V)
      SI=SI+V
      CALL GAUSS(IXR,RN,0.,V)
      SQ=SQ+V
   10 CALL XCNVLV(XI,SI,JXY,TPYI,HI,NHI)
      CALL XCNVLV(XQ,SQ,JXY,TPYQ,HI,NHI)
      XO=XI*XI+XQ*XQ
      IF (MOS-1) 14,14,13
   14 KET=IET
      GO TO 15
   13 KET=IET/MOS
      IF (MOD(IET,MOS)) 30,15,30
   15 CALL XCNVLV(XV,XO,JXY,TPYV,HV,NHV)
      IET=IET+1
      ET(KET)=ET(KET)+XV
      IF (XV-EPK) 25,25,20
   20 EPK=XV
   25 RETURN
   30 CALL XCFILL(XO,JXY,TPYV,NHV)
   33 IET=IET+1
      RETURN
      END
C
C
      SUBROUTINE XHEAD(IXR,RN,NN,NP,NMN,MOS)
C   WRITES GENERAL HEADING FOR CASE TO EE RUN
C
      WRITE (3,10) IXR
   10 FORMAT(1HC/40HC FOLLOWING ARE INPUT PARAMETERS---(IXR= I10,1H) )
      RSQ=RN*RN
      WRITE (3,20) RN,RSQ
   20 FORMAT(17H NOISE FACTOR RN= F12.6,16H,AND ITS SQUARE= F12.6 )
      WRITE (3,30) NN,MOS,NP,NMN
   30 FORMAT(' NN=',I3,', MOS=',I3,', NP=',I4,', AND NMN=',I4//)
      RETURN
      END
C
C
      SUBROUTINE XOUTPT(ET,NET,EPT,NE,EMNPK,EPK,VEPK,NPLSES,IXR)
      DIMENSION ET(NET),EPT(NE)
C
C   XOUTET HANDLES PRINTOUT OF RESULTS
      QA=EMNPK/EPK
      QB=EPK/EMNPK
      WRITE (3,10) IXR
   10 FORMAT(5H IXR= I10)
```

```
      WRITE (3,20) NPLSES,QA
   20 FORMAT(22HOFOLLOWING RESULTS FOR I4, 17H RETURNS YIELD R= F10.6)
      WRITE (3,30) EPK,EMNPK,QB
   30 FORMAT(5H EPK=E12.6,7H,EMNPK=E12.6,9H,AND 1/R= F10.6)
      SEPK=SQRT(ABS(VEPK))*SIGN(1.,VEPK)
      WRITE (3,35) SEPK
   35 FORMAT(' SIGMA(EPK)=',E12.5)
      WRITE (3,40) NET
   40 FORMAT(/' THE ',I4,' OUTPUT POINTS ARE:')
      WRITE (3,950)
  950 FORMAT(//' ***',5('..I.;...OUTPUT...***')/)
      WRITE (3,900) (I,ET(I), I=1,NET)
  900 FORMAT((' ***',5(I4,';',1PE12.5,'***')))
      WRITE (3,65)
   65 FORMAT(1HC/9H */*/*/*  /1H0)
      RETURN
      END
C
C
      SUBROUTINE XAFILL(HI,NHI,HV,NHV,EPT,NE,IMNP)
      DIMENSION HI(NHI),HV(NHV),EPT(NE)
      NAMELIST/LIST1/TOI,TOV,TAUI,TAUV,NI,NV
C
C   XAFILL SETS UP THE IMPULSE RESPONSE FUNCTIONS HI(.) FOR THE IF
C   AND HV(.) FOR THE VIDEO, AND SETS UP EXPECTATION VALUE
C   FOR WAVEFORM BY CALLING EPTFIL...
C
      READ (1,LIST1)
      WRITE (3,LIST1)
      NHI=NI
      NHV=NV
      CALL EPTFIL(DT,EPT,NE,IMNP)
      SUM=0.
      T=0.
      DO 100 I=1,NHI
      TI=(TOI-T)/TAUI
      X=EXP(-(TI*TI))
      SUM=SUM+X
      HI(I)=X
  100 T=T+DT
      DO 200 I=1,NHI
  200 HI(I)=HI(I)/SUM
      SUM=0.
      T=0.
      DO 300 I=1,NHV
      TI=(TOV-T)/TAUV
      X=EXP(-(TI*TI))
      SUM=SUM+X
      HV(I)=X
  300 T=T+DT
```

```
      DO 400 I=1,NHV
  400 HV(I)=HV(I)/SUM
C
C   DELETE THIS AFTER DEBUGGING?
C
      WRITE (3,450) NHI,TOI,TAUI,DT
  450 FORMAT('0FOLLOWING ARE THE',I4,' VALUES I,H(I) FOR TOI=',
     1 F6.3,', TAUI=',F6.3,','/' AND DT=',F6.3//)
      WRITE (3,500) (I,HI(I), I=1,NHI)
  500 FORMAT( (' ',4(I5,',',F9.5)))
      WRITE (3,550) NHV,TOV,TAUV,DT
  550 FORMAT('0FOLLOWING ARE THE',I4,' VALUES I,H(I) FOR TOV=',
     1 F6.3,', TAUV=',F6.3,','/' AND DT=',F6.3//)
      WRITE (3,500) (I,HV(I), I=1,NHV)
C
C   DELETE ABOVE?
C
      RETURN
      END
C
C
      SUBROUTINE XREAD(JXR,XRN,JN,JP,JMN,JOS,NE,NET)
      NAMELIST/LIST2/IXR,RN,NN,NP,NMN,MOS
C
C   SUBROUTINE XREAD HANDLES INPUT OF PARAMETERS FOR NEXT SIMULATION
C   RUN...
C
      IXR=-37
      JOS=1
      MOS=0
      READ (1,LIST2)
      IF (IXR+37) 10,20,10
C------- IXR=-37 SIGNIFIES NO IXR ENTERED ON &LIST2;RETAIN PRESENT
C   (SEE TEST PERFORMED ON IXR IN ROUTINE XSIMUL...)
C
   10 JXR=IXR
   20 XRN=RN
      JN=NN
      JP=NP
      JMN=NMN
      IF (MOS) 40,40,30
   30 JOS=MOS
   40 NET=(JN+NE)/JOS
      RETURN
      END
C
      SUBROUTINE EPTFIL(DT,YOUT,KE,KMNP)
      DIMENSION YOUT(KE),TOUT(3000),YIN(20),TIN(20),X(50),Y(50)
      NAMELIST/LIST1A/IMNP,NE,TIN,YIN
C..FOR 8 DEGREES OFF NADIR, MAIN PROG DIMENSIONS UP TO 3000
```

```
C
C   SUBROUTINE EPTFIL USES A SPLINE FUNCTION FIT TO 20 INPUT WAVEFORM
C   PAIRS TIN(I),YIN(I);THESE SAMPLE POINTS CAN BE CHOSEN TO
C   ADEQUATELY REPRESENT THE ENTIRE INPUT WAVEFORM.
C
      DO 15 I=1,KE
   15 YOUT(I)=0.
      DT=10.
      TT=10.
      READ (1,LIST1A)
      KMAE=IMNP
      DO 99 I=1,20
      X(I)=TIN(I)
   99 Y(I)=YIN(I)
      KE=NE
      IF (NE-3000) 19,19,16
   16 WRITE (3,17) NE
   17 FORMAT(//' ERROR: NE=',I4,' BUT 3000 IS MAXIMUM: QUIT!!')
      STOP
   19 DO 20 I=1,NE
      TOUT(I)=TT
   20 TT=TT+DT
      CALL ESPLIN(X,Y,20,TOUT,YOUT,NE,.00001)
      DO 25 I=1,NE
      XQ=YOUT(I)
      IF (XQ) 30,25,35
   30 YOUT(I)=0.
      GO TO 25
   35 YOUT(I)=SQRT(XQ)
   25 CONTINUE
C.....TOUT NOW A FREE ARRAY, USE TO OUTPUT THE INPUT SEQUENCE
      DO 100 I=1,NE
      XQ=YOUT(I)
  100 TOUT(I)=XQ*XQ
      WRITE (3,150)
  150 FORMAT(/' FOLLOWING IS INPUT SEQUENCE, TIME=I*(10 NANOSEC)'//)
      WRITE (3,200)
  200 FORMAT(' ***',5('..I.;...OUTPUT...***')/)
      WRITE (3,250) (I,TOUT(I), I=1,NE)
  250 FORMAT((' ***',5(I4,';',1PE12.5,'***')))
      RETURN
      END
C
C
      SUBROUTINE ESPLIN(X,Y,N,T,SS,M,EPSLN)
      DIMENSION T(M),SS(M)
      DIMENSION X(50),Y(50),H(50),DELY(50),H2(50),B(50),DELSQY(50),
     1 S2(50),C(50),S3(50)
C
C   THIS IS A SPLINE-FIT ROUTINE ADAPTED FROM "SPLINE FUNCTIONS,
```

```
C   INTERPOLATION, AND NUMERICAL QUADRATURE," T.N.E.GREVILLE, IN
C   "MATHEMATICAL METHODS FOR DIGITAL COMPUTERS, VOL.II," A.RALSTON &
C   H.S.WILF,ED'S.,(NEW YORK;J.WILEY,1967), PP. 156-168
C
      IF (N-50) 2,2,999
  999 WRITE (3,998) N
  998 FORMAT(//' ERROR! HAD',I4,' INPUT POINTS,ESPLIN HANDLES ONLY 50'/)
      RETURN
    2 N1=N-1
    3 DO 51 I=1,N1
      H(I)=X(I+1)-X(I)
   51 DELY(I)=(Y(I+1)-Y(I))/H(I)
    4 DO 52 I=2,N1
      H2(I)=H(I-1)+H(I)
      B(I)=.5*H(I-1)/H2(I)
      DELSQY(I)=(DELY(I)-DELY(I-1))/H2(I)
      S2(I)=2.*DELSQY(I)
   52 C(I)=3.*DELSQY(I)
      S2(1)=0.
      S2(N)=0.
      OMEGA=1.071797
      NITR=1
    5 ETA=0.
    6 DO 10 I=2,N1
    7 W=(C(I)-B(I)*S2(I-1)-(.5-B(I))*S2(I+1)-S2(I))*OMEGA
    8 IF (ABS(W)-ETA) 10,10,9
    9 ETA=ABS(W)
   10 S2(I)=S2(I)+W
   13 IF (ETA-EPSLN) 14,14,997
  997 IF (NITR-10) 996,996,995
  996 NITR=NITR+1
      GO TO 5
  995 WRITE (3,994) NITR,ETA
  994 FORMAT(/' NUMBER OF ITERATIONS=',I3,' AND EPSILON=',E12.5//)
   14 DO 53 I=1,N1
   53 S3(I)=(S2(I+1)-S2(I))/H(I)
C
C   COEFFICIENTS NOW ESTABLISHED, FOLLOWING STEPS PERFORM DESIRED
C   INTERMEDIATE POINT INTERPOLATION.
C
   15 DO 61 J=1,M
   16 I=1
   54 IF (T(J)-X(1)) 58,17,55
   55 IF (T(J)-X(N)) 57,59,58
   56 IF (T(J)-X(I)) 60,17,57
   57 I=I+1
      GO TO 56
   58 WRITE (3,44) J
   44 FORMAT(//' ',I4,'TH ARGUMENT OUT OF RANGE',//)
      SS(J)=0.
```

```
      GO TO 61
   59 I=N
   60 I=I-1
   17 HT1=T(J)-X(I)
      HT2=T(J)-X(I+1)
      PROD=HT1*HT2
      SS2=S2(I)+HT1*S3(I)
      DELSQS=(S2(I)+S2(I+1)+SS2)/6.
      SS(J)=Y(I)+HT1*DELY(I)+PROD*DELSQS
   61 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE GAUSS(IX,S,AM,V)
      A=0.0
      DO 50 I=1,12
      CALL RANDU(IX,IY,Y)
      IX=IY
   50 A=A+Y
      V=(A-6.0)*S+AM
      RETURN
      END
      SUBROUTINE RANDU(IX,IY,YFL)
      IY=IX*65539
      IF(IY)5,6,6
    5 IY=IY+2147483647+1
    6 YFL=IY
      YFL=YFL*.4656613E-9
      RETURN
      END
```