NASA-CR-172,333

NASA Contractor Report 172333

NASA-CR-172333
19850004577

Realistic Localizer Courses For Aircraft Instrument Landing Simulators

Murphy, Timothy A.

Avionics Engineering Center
Department Of Electrical And Computer Engineering
Ohio University
Athens, Ohio 45701

Contract NAS1-17368
May 1984

NASA
National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

FOREWORD

This report, authored by Mr. Timothy A. Murphy, presents results obtained under Contract NAS1-17368. Dr. Richard H. McFarland director of the Avionics Engineering Center, served as Project Director and provided the description of the Ohio University ILS course structure data collection system and course structure repeatability information. Mr. James D. Nickum was Project Engineer for this work.

The work was supported by Ohio University Computer Services, the Department of Electrical and Computer Engineering, and the word-processing section of the Avionics Engineering Center.

N85-12885#

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF TABLES

I. SUMMARY

To this point in time flight simulation used in flight training and research has employed ideal straight-line paths for the glide slope and localizer portions of the ILS. This work reported gives, instead, practical path shapes and structures for simulator use. The difficulty is that real ILS courses are not smooth paths but do possess noise which is presented to the pilot. The work reported here will improve the quality of flight simulation by supplying the pilot with a more realistic indication of ILS paths.

Records of real-world ILS facilities have been obtained or made and these form the basis of the research product, viz, digitized real-world paths prepared for inclusion in the programs which the flight simulator uses for producing the indications on the pilot's course deviation indicator (CDI). In addition, some synthesized paths derived from some FAA supported parallel work [1] are provided to cause the pilot to experience path excursions at Category II tolerance limits.

The computer programs and data presented in this report provide a flight training research capability not previously available. With the emphasis on the workload of a single-pilot IFR (SPIFR) situation, the ability to simulate realistically the aircraft cockpit environment for investigating pilot performance is important. This work will allow more realistic flight simulations of ILS approaches. Typically in the past a flight simulator used perfectly smooth ILS paths as input to the graphic display device usually course deviation indicators.

II. INTRODUCTION

The idea of using practical, real-world ILS beam structures in flight simulators is believed to have considerable merit, in particular with respect to flight training. Investigation has revealed that while many pilots using a simulator believe practical, representative beam structures were used, some who have flown actual ILS Category III approaches are well aware that the simulator does not represent actual conditions. Recognizing this problem, NASA Langley Research Center established contract NAS1-17368 with the Avionics Engineering Center of Ohio University to implement realistic ILS course structures for an aircraft simulator at Langley. The major program steps are:

1. Determine where and to what extent other contractors may have prepared non-ideal structures for use in simulation.

2. Review existing measured data from various ILS facilities and select ten sets of data for implementation in the simulator.

3. Digitize the selected paths with the structure noise included.

4. Investigate a means of implementing the roughness to provide the most efficient method and to produce the most realistic results in terms of the pilot seeing what actually exists in space at the particular facility.

5. Prepare several generic paths to allow a rigorous approach to determine pilot difficulties.

6. Subsequent to selecting a method of inputting the noisy path, prepare ten (10) actual paths in a format compatible to the NASA Cyber computer.

7. Prepare a setup of a cockpit display at the contractor's facility to allow inspection of the results of the implementation of the path with noise.

Actions taken in the completion of these tasks are discussed in this report.

Also, as a part of this work, inquiries were made of several large commercial manufacturers of aircraft simulators regarding ILS course structures produced in simulators. Of the 5 companies contacted, none indicated that actual path structures were being used. The typical approach has been to assume a perfectly straight structure oriented properly with respect to the runway.

In addition to the inquiries to industry, a literature search was conducted. Dialog (an electronic data base operated by Lockhead Information Services) was searched for relevant publications and papers. Nothing relevant was found. Further, the Avionics Engineering Center's library contains no reports of significant work in this area.

III. REVIEW OF EXISTING MEASURED DATA FROM ILS FACILITIES

When precision measurements are made of localizer and glide slope structures, it is necessary to provide a reference against which the path is compared. Two different reference systems are in common use. The FAA uses an inertial platform which is updated by DME information and manual marks when the flight passes over the threshold. Ohio University employs a radio telemetering theodolite. This is an optical instrument which is used by skilled operator to track the airplane from a specially designated point on the ground. The angular position is taken in electrical analog form from the theodolite and telemetered to the aircraft. Here it is subtracted from the course indication produced by the ILS receiver giving a difference which is desirably independent of the aircraft position.

By using the inertial reference or theodolite it is possible to remove for the most part the course information dependence on the aircraft position. Obviously, an aircraft flying above the course will get a fly-down indication. Unless the precise position of the aircraft is known, one cannot ascertain whether the aircraft was high or the path had a dip in it. The references eliminate this problem.

It should be noted that the ideal glide slope on-course is a three-dimensional geometrical shape, which in the case of the common image system is a cone. If the shape of this cone is perturbed, typically the perturbation is not linear vertically and horizontally. Consequently, flight measurements made with the aircraft not on the on-course means that the actual path is predicted based on assumptions of linearity. For example, if the aircraft is off-course on the localizer for one measurement of the glide slope and for another the aircraft is on the localizer, then it is possible that different glide slope structures will be recorded. The reason is that the horizontal structure of the cone is not uniform.

From these circumstances one can observe variations in path structures that are even measured in close time proximity. This does not mean, of course, the structures are not repeatable, but rather that the flight tracks for measurements were not executed with sufficient precision. Precise positioning of the aircraft and use of an accurate reference system will insure repeatability in glide slope structure measurements.

Data used in preparing the tracks for these simulations were scrutinized for repeatability which confirms use of a satisfactory reference and reasonable positioning of the aircraft while making measurements.

Approximately 40 FAA flight check recordings were obtained through a visit to the Flight Inspection Field Office based at Oklahoma City, Oklahoma on July 15, 1983. These localizer course structure recordings were made of many different ILS facilities in the U.S. The localizer course structure recordings were acquired through the use of the FAA's Automated Flight Inspection System or AFIS (see figure 1).[2] The AFIS uses position information supplied by an inertial navigation system in a comparison with the position indicated by the measured ILS RF fields. The result of this comparison is the difference between the actual and

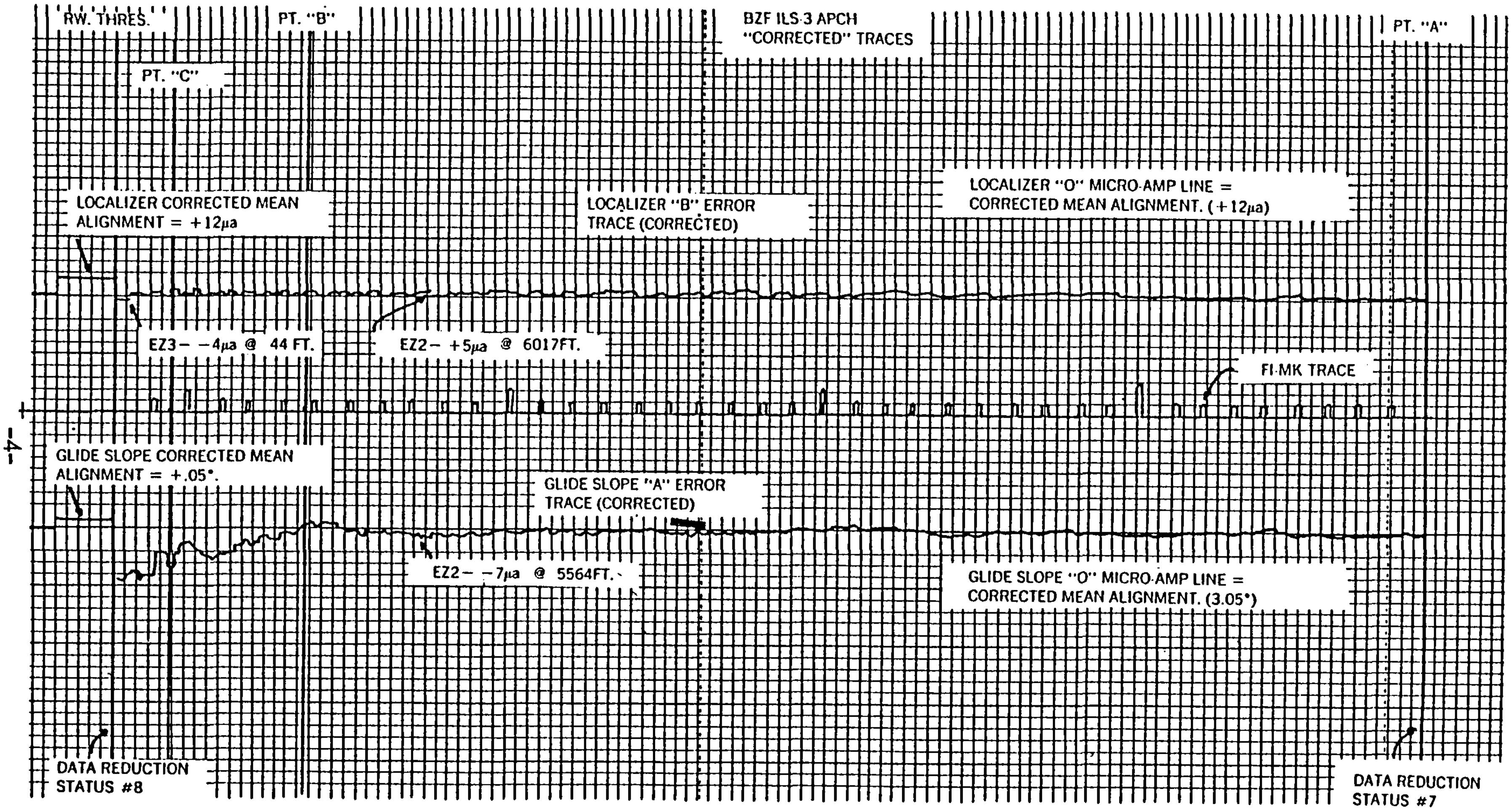

Figure 1. Localizer Course Structure Data Taken by AFIS.

indicated position of the aircraft. This information represents the error in the localizer course structure. The recordings were carefully reviewed and the best ten were selected as a basis for the real localizer path error structures to be prepared for NASA. The paths were selected based on the significance of abnormalities displayed by each path. Those paths with the greatest deviations being selected.

Glide slope courses were selected from ILS flight test measurements made by the Ohio University Avionics Engineering Center staff between 1980 and 1982 (See figure 2). Course structure measurements are accomplished by subtracting the known position of the aircraft from the observed (CDI) indication. The aircraft's position is tracked accurately using a radio telemetering theodolite. This information is then fed, along with the aircraft CDI indication, to a differential amplifier. The output of this amplifier is the difference between the known position and indicated position of the aircraft. This is the glide slope structure error information to be utilized by the simulation. Approximately 100 recordings were reviewed. Fifteen of the most interesting glide slope structures were selected and digitized for further inspection. Ten courses of these fifteen were selected for use.

A listing of the courses selected and the installations from which the data was taken is given in appendix A. Plots of the localizer and glide slope courses are also given in appendix B. Table B-I, located in appendix B, specifies which glide slope and localizer courses were paired together to form the ILS courses.

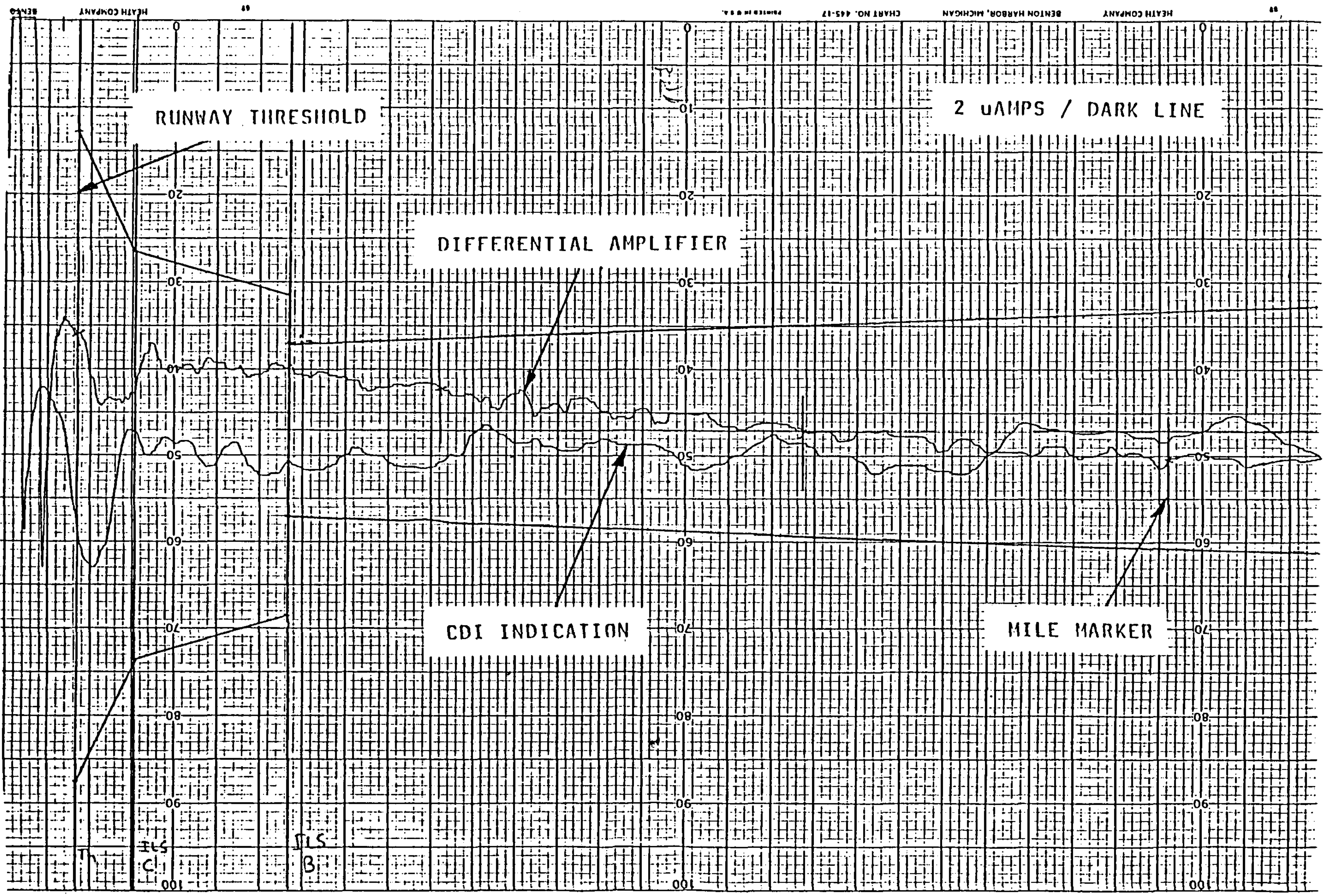

Figure 2. Glide Slope Course Structure Data Taken by the Staff of Avionics Engineering Center.

IV. DIGITIZING THE SELECTED PATHS WITH STRUCTURE NOISE

The courses selected were digitized using an analog strip-chart to digital data translator developed by the Avionics Engineering Center in 1976.[3] The position of a manually-operated pen is sensed by the data translator and is coded into BCD format for recording on a 9-track digital magnetic tape recorder. Sample rates of the data translator are adjustable to run the manual track at a comfortable speed, allowing good resolution and accuracy in the converted data. Sensitivity of the device is also adjustable to allow for the most efficient use of the available quantization. The best settings for both the sample rate and the sensitivity were determined empirically and the same settings were used while digitizing all courses.

The tape with the courses encoded into a BCD format was delivered to Ohio University Computer Services. The data were transferred to an IBM 370 mainframe and stored for further processing. It was necessary to develop software to read the tapes and to unpack and translate the BCD coded data. A listing of the software used can be found in appendix C. The data for each localizer course were scaled to represent degrees deviation from the runway centerline. The software used in the rescaling process took into account the scaling of the original measurements, the sensitivity setting of the data translator and the localizer course width of each particular site. All ten courses were scaled such that the error will have the same relative significance on the four-degree course width (assumed by the NASA simulator) as it did with respect to the course width of the original ILS site. Glide slope course structures were scaled similarly. The NASA simulator assumes a 0.7 degree deviation from the glide slope angle for full scale deflection of the CDI needle. The glide slope data were scaled to account for this as well as the scaling of the original measurements and the data translator used to digitize the data.

An interpolation routine was developed which would readjust the sampling of each course structure such that samples occurred at 25-foot intervals. This normalization was applied to all ten courses so that all ten would be uniform in this respect. Specifying a uniform course format simplifies the implementation of the course stuctures in the existing NASA software, as the routine will be the same regardless of which particular course structure is used. The original measurements were made at various chart recorder speeds. Event marks corresponding to nautical mile marks were recorded during the data translation and this information was used to readjust the courses into the standardized form.

Each course was truncated to 1000 points (hence 25000 feet of localizer and glide slope structure or just over 4 nautical miles). The last 500 feet of course data (farthest from the runway) was altered so that the error data for both localizer and glide slope rises smoothly from zero to its actual value at 24,500 feet from the runway threshold. This was done to avoid a discontinuity in the CDI indication when the simulated aircraft first enters the area within 25,000 feet of the runway threshold for which the course structure data exists. The error beyond 25,000 feet is assumed to be insignificant for this study.

V. IMPLEMENTING THE ROUGHNESS

The noise structure of ILS courses in space, whether localizer or glide slope are repeatable when measured precisely. The structure contains noise which is a result of the multipath effects of objects surrounding the airport such as buildings, terrain, power lines and towers. An aircraft moving along a given path in space will, provided no changes in terrain or building or structure changes have occurred, receive the same structure information. In fact, this characteristic quality of each localizer and glideslope is used to ascertain that if on successive flight evaluations the characteristic structure of the path is the same then the flight measurement is valid. The structures presented in this report are true path structures as would be seen in actual flight. They are not due to noise in the receiver telemetry or recording systems. These noise levels are below the observable thresholds.

Several schemes for implementing the course roughness were discussed. The simplest method was determined to be a direct table lookup. The distance of the aircraft from the runway threshold was used in a formula to calculate the table index. This method is economical in that it is memory-conservative as well as easily implemented. For this table lookup approach it is necessary that all the courses assume a standard form. Data tables of 2,000 elements, each representing the localizer and glide slope structures sampled at 25-foot intervals, were created. The tables are arranged as 1,000 pairs of elements, where each pair corresponds to the localizer and glide slope error in degrees for a particular distance from the runway threshold. Distance is determined by the position of the pair in the table. Figures 3 and 4 illustrate the relationship between the values in the table and the distance from the runway threshold. The model assumes bilateral uniformity. At any given point it is assumed that the error in the CDI indication is a function of the aircraft's distance from the runway threshold and not a function of the aircraft's lateral position with respect to the runway centerline.

The table for a specific site may be loaded prior to the beginning of the simulation. This information is then referenced throughout the flight. For distances beyond 25000 feet beyond the runway threshold, the last element in the table will be selected (which is zero error, as described above). As the distance to the threshold falls below 25,000 feet the table lookup routine will index farther down the table until, as the aircraft crosses the threshold, the first pair of elements in the table will be selected. The values resulting from this table lookup process are simply added to the localizer and glide slope indication which is calculated by existing NASA simulator software. The calculated ILS CDI indication is, of course, a function of the aircraft lateral displacement with respect to the runway centerline and displacement with respect to nominal glide slope angle. However, the ILS course structure error information is only a function of the distance to the runway threshold.

For each of the various ILS simulations, approximately four miles of localizer and glide slope data are contained in a table of 1,000 pairs of elements. Each pair of elements corresponds to the error in the indicated

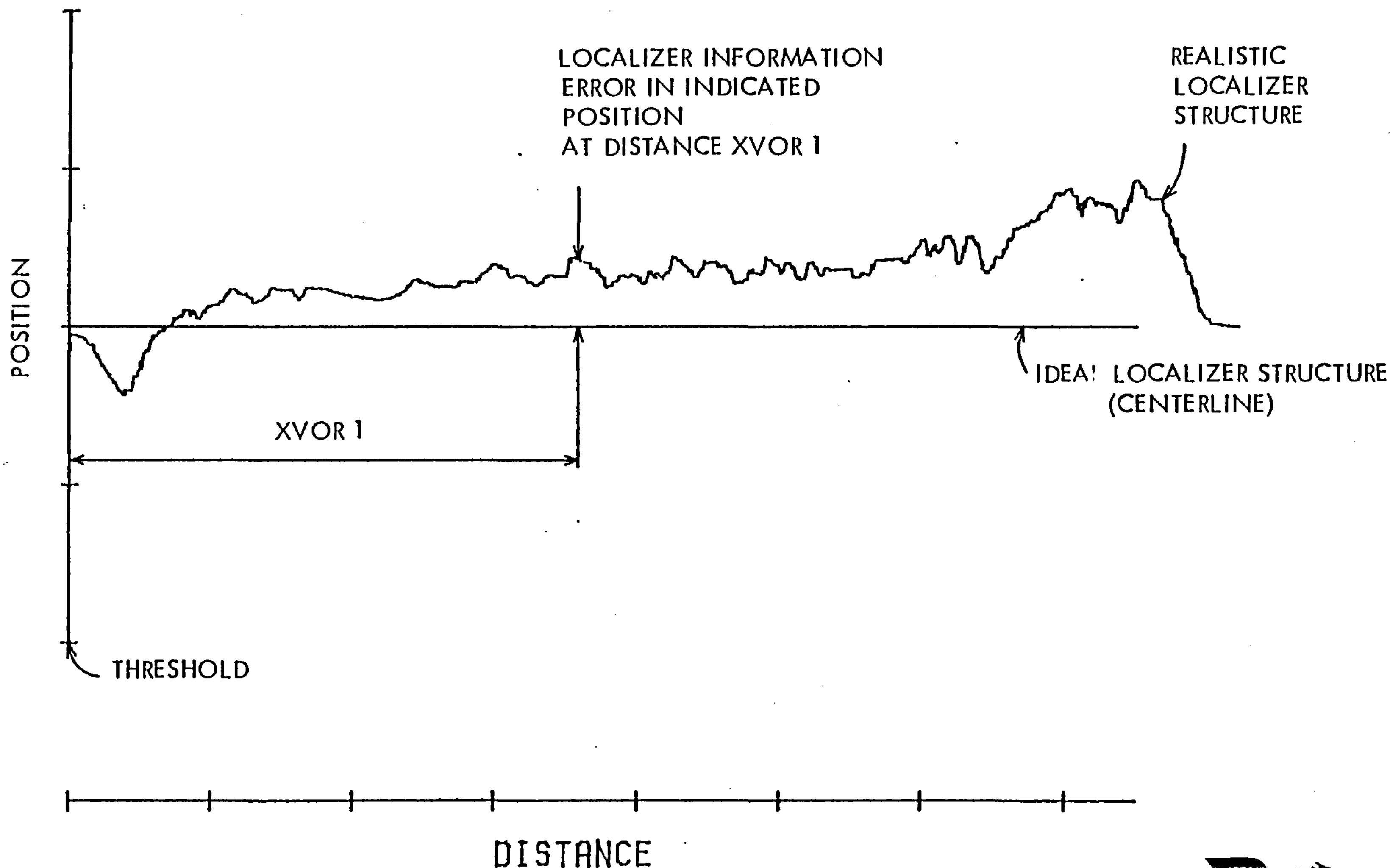

Figure 3. Illustration of Relationship Between the Values in the Table and the Localizer Structure.

Ohio University / Athens

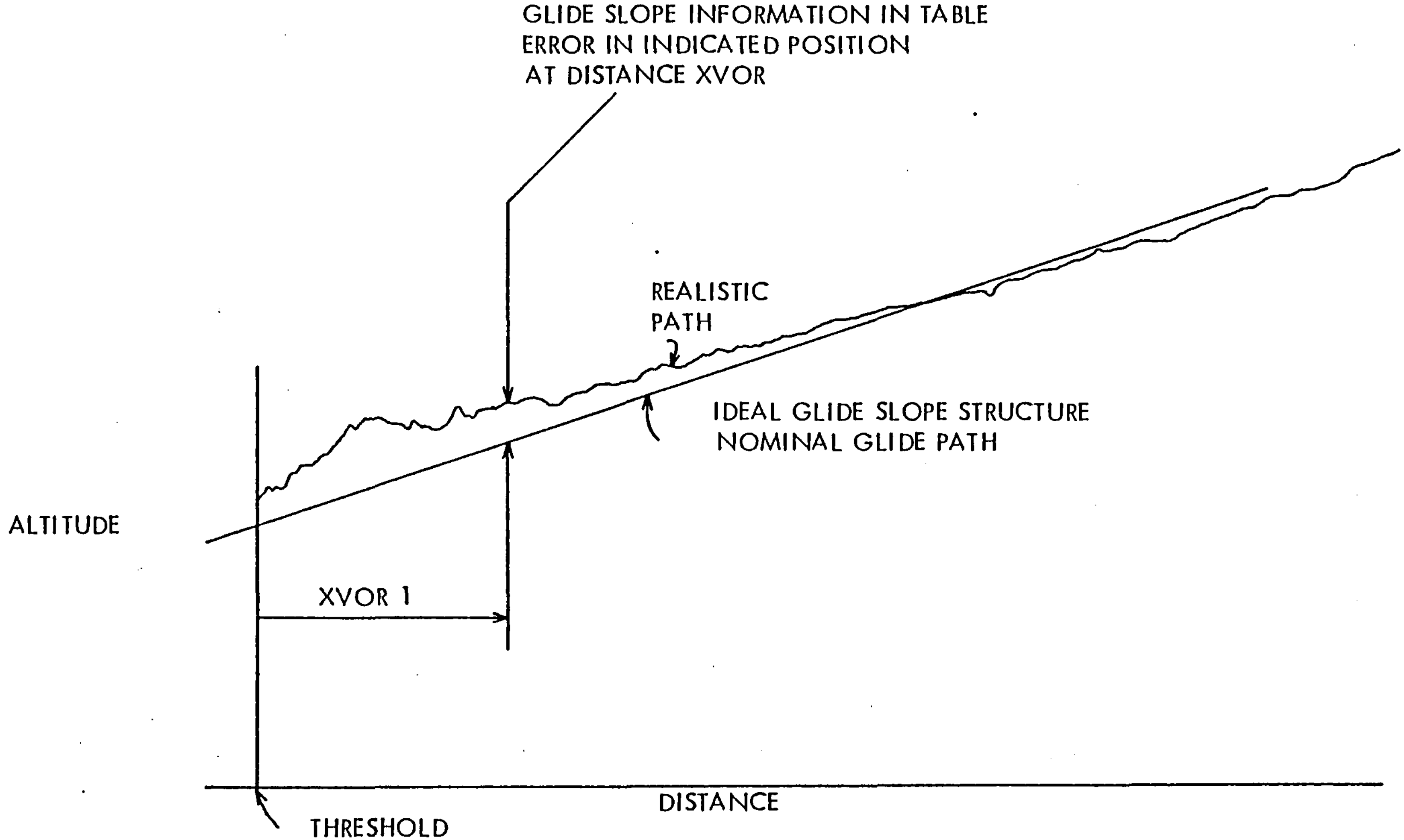

Figure 4. Illustration of Relationship Between the Values in the Table and the Glide Slope Structure.

Ohio University / Athens

position in degrees at intervals of 25 feet. The table is constructed such that localizer data appears first. Hence, for n in the range 1 to 1,000, the data contained in the (2*n)-1 table position corresponds to the error in the indicated position (in degrees from centerline) at a distance of (n-1)*25.0 feet from the runway threshold for the localizer. Similarly, the 2*n element in the array corresponds to the error in the indicated position in degrees from the glide slope reference angle at a distance of (n-1)*25 ft from the runway threshold. The model for the course irregularities assumes lateral uniformity. The localizer values in the table correspond to the indicated position of an observer standing on the runway centerline at a distance x=(n-1)*25.0 from the runway threshold.

The glide slope values in the table correspond to the indicated postion of an observer on the nominal glide path at some distance x=(n-1)*25.0 feet from the runway threshold. Lateral uniformity implies that the relative error in indicated position at a given distance x will be the same for any position along the y axis at that distance (within the relevant range of y values). Since the maximum angular deviations are small this should be a valid approximation. The assumption of lateral uniformity allows simple use of the x distance from the threshold to calculate the table index. (The n referred to above is the table element pair index.)

Figure 5 is a section of a listing of the existing NASA software. Some modification to the subroutine RADNAV will be required. This section of the subroutine is shown below. In the existing software the variable XVOR1 corresponds to the distance along the x-axis from the runway threshold to the current position of the aircraft. This distance along with YVOR1 (the distance along the y-axis from the runway centerline) is used to calculate the CDI indication in degrees for the localizer. XVOR1 is also used along with aircraft height above ground to calculate the glide slope indication. The calculated glide slope deviation is referenced to the nominal glide path.

To implement the non-ideal course structures it would be sufficient to alter the code shown in figure XX as follows:

```
      CURSCL=1.0
      GSSCL=1.0
C     Specification of scale factors. This should be done during the program
C     initialization.
      .
      .


C     In the RADNAV section beginning at line 494:

      .
      .
      LOC1=ATAN2(-YVOR1,ABS(XVOR1-TABLE(105)) )*RADDEG           RADNAV494
```

```
C******  CALCULATE COMPASS MAGNETIC ERROR                                    RADNAV 442
C                                                                           RADNAV 443
   CERR = (8.2103E-07 * X + 3.29945E-06 * Y + 7.96)/57.295                  RADNAV 444
   IF(LDISI(34)) CERR = 0.                                                  RADNAV 445
C                                                                           RADNAV 446
   IF (.NOT. NAVCOM2) GO TO 10                                              RADNAV 447
C                                                                           RADNAV 448
C  COMPUTE VOR1 COORDINATES                                                 RADNAV 449
C                                                                           RADNAV 450
   CALL STATION(ISTATN, NAVFRQ1,FREQ,NMAX)                                  RADNAV 451
   IF (ISTATN, GT. NMAX) GO TO 10                                           RADNAV 452

   LOCIT = ICODE(ISTATN) .EQ. ILL                                           RADNAV 457
   VOR1  =  .NOT. LOCIT                                                     RADNAV 458
     BOBS1  =  ATAN2 (BVOR1S, BVOR1C)                                       JFIX    19
   IF (.NOT.LOCIT)GO TO 30                                                  RADNAV 459
   BVOR1S  =  SIN(RUNWAY(ISTATN)/57.295                                     RADNAV 460
   BVOR1C  =  COS(RUNWAY(ISTATN)/57.295                                     RADNAV 461
     GO TO 31                                                               RADNAV 462
   30 CONTINUE                                                              RADNAV 463
      SR = SIN(ORIENT(ISTATN)/57.295)                                       RADNAV 464
      CR = COS(ORIENT(ISTATN)/57.295)                                       RADNAV 465
      BS = BVOR1S                                                           RADNAV 466
      BC = BVOR1C                                                           RADNAV 467
      BVOR1S  = BS*CR - BC*SR                                               RADNAV 468
      BVOR1C  = BC*CR + BS*SR                                               RADNAV 469
   31 CONTINUE                                                              RADNAV 470
      DELX1 = X - XSTN(ISTATN)                                              RADNAV 472
      IF(DELX1 .EQ 0.) DELX1 = 1.E-6                                        RADNAV 473
      DELY1 = Y - YSTN(ISTATN)                                              RADNAV 474
      B1TOSTN = ATAN2(-DELY1,-DELX1)                                        RADNAV 475
      IF(.NOT.LOCIT)B1TOSTN=B1TOSTN+ORIENT(ISTATN)/57.295                   RADNAV 476
      RMAX = 1.E30                                                          RADNAV 477
      IF (LOCIT) RMAX = 1.E30                                               RADNAV 478
      GSRANGE = SQRT(DELX1*DELX1+DELY1*DELY1) .LT. RMAX                     RADNAV 479
      XVOR1    = DELX1*COSVOR1 + DELY1*SINVOR1                              RADNAV 480
      YVOR1    = -DELX1*SINVOR1 + DELY1*COSVOR1                             RADNAV 481
      TEMP     = XVOR1*XVOR1 + YVOR1*YVOR1                                  RADNAV 482
      DME1   = SQRT(TEMP + (H-ZSTN(ISTATN*(H-ZSTN(ISTATN))) *0.000164468    RADNAV 483

      DMERATE = (DEMOLD - DME1)*3600./H7                                    RADNAV 487
        DMERAT1=ABS(DMERATE)                                                RADNAV 488
      DMEOLD = DME1                                                         RADNAV 489
      RANGE1 = SQRT(TEMP)                                                   RADNAV 490
      RCONE = 1.4281*(H-ZSTN(ISTATN)                                        RADNAV 491
      IF (LOCIT) 1,2                                                        RADNAV 492
   1  IF(GSRANGE)3,4                                                        RADNAV 493
   3  LOCI = ATAN2( -YVOR1,ABS(XVOR1-TABLE(105)) *RADDEG                    RADNAV 494
      GS1 = XVORI .LT. 1000.                                                RADNAV 495
      GSDEV1 = 0.0                                                          RADNAV 496
      IF(GS1) GSDEV1 = ATAN2 (H-ZSTN(ISTATN),-XVOR1)*RADDEG -GLSI           RADNAV 497
      TO1  =  FROM1  =  .F.                                                 RADNAV 498
      GO TO 5                                                               RADNAV 499
```

Figure 5. Section of Listing Received from NASA Programmer. In this section of the program the relative distance to runway threshold and the CDI indication is calculated.

```
C     Calculation of localizer indication in degrees.
      GS1=XVOR1.LT.1000.
      GSDEV1=0.0                                                 RADNAV495
      IF(GS1) GSDEV = ATAN2(H-ZSTN(ISTATN),-XVOR1)*RADDEG-GLSI   RADNAV496
C     Calculation of glide slope.
C     The above lines are unchanged.
      LDEX=INT(ABS(XVOR/25.0))*2-1                               RADNAV497
C     Calculation of the the lookup table index as a function of
C      the x distance from the runway threshold.
      IF (LDEX.GT.1999) THEN LDEX=1999                           RADNAV498
      IF (LDEX.LE.0) THEN LDEX=1                                 RADNAV499
C      If LDEX is greater than 1999 the aircraft is farther than
C      25000 feet from the runway threshold. If this is the case, the
C      last element in the table is selected. In all the tables this
C      last element is zero and therefore no error is added for
C      distances beyond 25000 feet from the threshold.
      LOC1=LOC1+ILSERR(LDEX)*CURSCL                              RADNAV500
C      The localizer indication is simply the sum of the calculated
C      localizer indication and the error data.
C      similarly for the glide slope.
      GSDEV1=GSDEV1+ILSERR(LDEX+1)*GSSCL                         RADNAV501
```

Where: ILSERR is an array of 2000 elements which is loaded with the localizer and glide slope error structure prior to the beginning of the simulation.

CURSCL and GSSCL are optional variables which will allow scaling of the sensitivity of the system to the course width. If CURSCL=1.0 the error added to the calculated indication is exactly the error as measured at the various sites. If for example CURSCL=1.01 then the error added to the calculated localizer reading will be 1% greater than measured and so on. This should allow some study of tolerable course irregularities. These variables should be initialized to a value in some convenient manner earlier in the program execution.

The above approach assumes that the course structures can be loaded into a one-dimensional array of 2,000 elements (from tape or other storage media) prior to the beginning of the simulation run. All the data collected and digitized have been rescaled so that simulated errors have the same relative significance on the 4-degree course width assumed by the NASA simulator for the localizer and the +0.7 degree angle (referenced to the nominal glide path) assumed for the glide slope. Figure 6 is a section of a listing of the existing NASA software. This section of the code outputs the calculated CDI indication to the appropriate D/A converter to drive the instruments in the cockpit. In this section the data are scaled such that full scale deflection occurs for +2 degrees deviation from centerline for the localizer and +0.7 degrees deviation from the nominal glide path for the glide slope. The variable CURSCL would allow the sensitivity of the real localizer signal to be altered if desired. In a similar manner GSSCL would allow alteration of the glide slope sensitivity.

```
C**** GLIDE SLOPE FOR HSI ON                                  DACOUT 419
C                                                             DACOUT 420
      DAC(30)  =  -GSDEV1 * SDAC19                            DACOUT 421
C                                                             DACOUT 422
C**   HSI VOR/LOC INDICATOR                                   DACOUT 423
C                                                             DACOUT 424
      DAC(28) = LOC1*SDAC16                                   DACOUT 425
      IF((LOCIT)) DAC(28)  =  .4*LOC1                         DACOUT 426
C                                                             DACOUT 427
C                                                             DACOUT 428
C*                                                            DACOUT 429
C************************                                     DACOUT 430
C**********************  SIGNAL FOR SOUND SYSTEM              DACOUT 431
```

Figure 6. Section of Listing Received From NASA Programmer.

VI. GENERIC PATHS

In addition to the ten real courses digitized for use in the simulator, nine generic rough paths were developed. Three sets of generic localizer and glide slope paths were designed by Dr. R. H. McFarland, director of Avionics Engineering Center. The nine generic courses were created by taking all possible combinations of the three different idealized localizer courses with the three different glide slope courses. The generic localizer and glide slope courses were designed to provide interesting noise disturbances for the pilot/aircraft control loop. These courses will be useful in a study of specific types of pilot difficulties when flying instrument approaches in non ideal situations. The generic courses were combined in the following manner: (See the plots accompanying this document.)

Generic course 1 = GENLOC1 with GENGS1
Generic course 2 = GENLOC1 with GENGS2
Generic course 3 = GENLOC1 with GENGS3
Generic course 4 = GENLOC2 with GENGS1
Generic course 5 = GENLOC2 with GENGS2
Generic course 6 = GENLOC2 with GENGS3
Generic course 7 = GENLOC3 with GENGS1
Generic course 8 = GENLOC3 with GENGS2
Generic course 9 = GENLOC3 with GENGS3

Plots of the courses GENLOCn and GENGSn are given in appendix B.

VII. COCKPIT DISPLAY

To facilitate assessment of the course structures chosen and designed, a cockpit display was fabricated in the laboratory. A flight simulation program was used in conjuction with a routine which simulates the table lookup technique discussed earlier in this paper to access the course structure data and simulate in-flight conditions. The flight simulator was directed to fly a perfect path (straight down the runway centerline, on the nominal glide slope angle). During the flight, the instrument indication was driven using the table lookup method. This allowed the error data (and only the error data) to be viewed. The results of these simulations were then downloaded to a Heath H-89 microcomputer. A simple routine was written in FORTH to take the data and output it at an appropriate rate to a serial digital-to-analog converter. This serial device has two channels of D/A conversion available.[4] These D/A channels were used to drive a modified CDI instrument. The data were scaled and converted to the appropriate form during the initial simulation run to give the displayed data the same significance it will have on the NASA simulator. All the courses prepared for this study were reviewed on this display.

VIII. DIGITIZATION AND COURSE CONSTRUCTION SOFTWARE

A variety of programs were written in a number of different languages to record, translate and display the data for the simulator's ILS error structures. A description of the programs developed and their uses is given below. Listings of these programs may be found in appendix C.

To facilitate the study of the effect of the structure noise on flight performance a FORTRAN program called USRPOSA was developed, which would simulate arbitrary flight paths. This program uses flight instruction data to generate appropriate position information. This flight position information is then used by another routine which simulates the table lookup to be used in the NASA simulation. A FORTRAN program called FLY was developed to take the position data generated by USRPOSA and apply the table lookup method to generate the CDI indication information in much the same way as it will be done in the NASA simulator. This information was then reviewed on the laboratory cockpit display. Several IBM 370/CMS EXEC routines were developed to manage the file manipulations for each of the flight path programs in this study.

The strip chart data translator used to digitize the measurements made at the various ILS sites produces a magnetic tape with all the samples written in BCD format, with each digit in a different byte. A simple CMS EXEC routine called TAMMOVE was written to allow these data files to be read by the IBM 370. Once the files had been read, it was necessary to translate the files from this packed BCD format to FORTRAN compatible integer format. This was done by using the program MAS5. MAS5 handles the conversion by using two variables which have been declared equivalent. One variable is of the logical type (1 byte) and the other is standard FORTRAN integer type. The program reads the digitized data and unpacks the records one digit at a time. After 4 digits have been unpacked, these four digits are used to calculate the integer number corresponding the BCD (4-digit representation). The most significant digit is used only as an event mark. This event information is preserved at this point as it will be needed later for spatial scaling of the data. A CMS EXEC was developed to define the files and determine the length of the input file (as this changes from one course to another) and pass this information on to the program MAS5.

Once the data were obtained in integer representation, the next step was to scale the data. For this purpose two Pascal programs called SCLVERT and GSSCLV were developed. These programs are identical with the exception that one has the proper scaling constants for the localizer data (SCLVERT) and the other is designed to scale the glide slope data (GSSCLV). This difference in the scaling procedure arises from the fact that two different measurement techniques were used in the original data acquisition. The localizer data taken by the FAA's Automated Flight Inspection System is already expressed in degrees. The data for the glide slope paths taken by the Ohio University is in terms of microamperes. The end result of both programs is a 1,000 point table in a standard format with all values expressed in degrees. The original data were recorded at various chart speeds and hence, during the digitizing, the paths were translated with different sampling intervals. The scaling programs use an interpolation routine to adjust the spatial sampling to a uniform 25 feet per sample for all the error structures. Event marks corresponding to intervals of nautical miles which were recorded during the data translation are used in the interpolation routine for the spatial scaling. Uniform spatial scaling was chosen in order to simplify the implementation of the noisy paths in the NASA simulator, since the method of implementation would be independent of the specific structure used for a particular flight simulation. The scaling of the magnitudes in the tables is also uniform in the sense that each point is scaled to have the same significance with respect to course width assumed by the NASA simulator as it had at the measured site. The program SCLVERT prompts the user for the course width of the original site in order to scale the localizer data accordingly. The glide slope data is already in terms of microamperes and hence this information is not needed. A CMS EXEC routine to make file definitions etc., was developed for both scaling programs.

Once the data was properly scaled and in a uniform format, a magnetic tape was created containing all ten course error structures. This tape was then delivered to NASA Langley with a short description of the recommended implementation method and details pertaining to the tape and data format. This information is repeated below.

IX. TAPE FORMAT

The magnetic tape supplied to NASA as part of this contract is in the following format:

9-Track
1600 BPI
Lrecl 80
Block 800
Recfm FB
EBCDIC

The ten courses were written on the tape one element per record using a FORTRAN format:

```
FORMAT(F10.8)
```

The table is arranged in pairs of elements with the localizer data appearing first:

Localizer error
Glide slope error
Localizer error
Glide slope error
•
•
•
•
etc.

Each pair represents the noise at some distance from the threshold.

The tape contains 10 real ILS course structures and 9 generic courses as described above. There are no tape marks between each course and no tape label at the end of all 19 courses, i.e.:

Course 1
Course 2
Course 3
•
•
•
•
•
etc.
•
•
•

Course 10
Generic Course 1
Generic Course 2
.
.
.
.
Generic Course 9

Each course is 2,000 elements long. Hence there are 38,000 records on the tape.

XI. REFERENCES

[1] McFarland, Richard H., "Initial Assessment of Appropriateness of Quantitative Tolerance Values Used to Qualify Glide Slope Structures," Technical Memorandum G-3, Avionics Engineering Center, February 1984.

[2] "Automated Flight Inspection System Training Manual," Federal Aviation Administration, Catalog No. 20717/20718.

[3] Blasche, Paul R., "Operating Instructions: Analog Strip-Chart to Digital Data Translator," Technical Memorandum S-17R, Avionics Engineering Center, March 1976.

[4] Intelligent Remote Serial I/O Unit (SL-800 Series), Users Manual

XI. APPENDICES

1. Appendix A.

The data for the real localizer courses were taken from the following locations:

LOC1 - CDG Houston, Texas. ILS runway 32, Jan. 4, 1982.
LOC2 - FWH Carswell AFB, Fort Worth, Texas. Runway 35, July 13, 1982.
LOC3 - TIK Tinker AFB Oklahoma City OK. Runway 35, January 19, 1981.
LOC4 - DRT Del Rio, Texas. Runway 13, June 9, 1982.
LOC5 - TIK Tinker AFB Oklahoma City, Oklahoma. Runway 35, Jan. 19, 1981.
LOC6 - VQE Randalf AFB, San Antonio, Texas. ILS runway 32L
LOC7 - ALW Walla Walla, Washington. ILS runway 20
LOC8 - TCB Fort Worth, Texas. Runway 17, July 1, 1982.
LOC9 - SPT Albuquerque, New Mexico. ILS runway 8, July 26, 1982.
LOC10 - SPT Albuquerque, New Mexico. ILS runway 8, July 26, 1982.

The real glide slope course data was obtained from measurements made at the following times and places:

GS1 - SHV Shreveport, Louisiana. Runway 13, May 27, 1981.
GS2 - SHV Shreveport, Louisiana. Runway 13, May 27, 1981.
GS3 - SHV Shreveport, Louisiana. Runway 13, May 28, 1981.
GS4 - SHV Shreveport, Louisiana. Runway 13, May 28, 1981.
GS5 - SHV Shreveport, Louisiana. Runway 13, May 28, 1981.
GS6 - IPT Williamsport, Pennsylvania. Runway 27, Oct. 30, 1980.
GS7 - IPT Williamsport, Pennsylvania. Runway 27, Oct. 21, 1980.
GS8 - SHV Shreveport, Louisiana. Runway 13, May 28, 1981.
GS9 - IPT Williamsport, Pennsylvania. Runway 27, Oct. 30, 1980.
GS10 - SHV Shreveport, Louisiana. Runway 13, may 28, 1981.

2. Appendix B.

Course1	....	LOC1	and	GEN1
Course 2	....	LOC 2	and	GEN 2
Course3	....	LOC3	and	GEN3
Course4	....	LOC4	and	GEN4
Course5	....	LOC5	and	GEN5
Course6	....	LOC6	and	GEN6
Course7	....	LOC7	and	GEN7
Course8	....	LOC8	and	GEN8
Course9	....	LOC9	and	GEN9
Course10	....	LOC10	and	GEN10

Generic course 1	...	GENLOC1	and	GENGS1
Generic course 2	...	GENLOC1	and	GENGS 2
Generic course 3	...	GENLOC1	and	GENGS3
Generic course 4	...	GENLOC 2	and	GENGS1
Generic course 5	...	GENLOC 2	and	GENGS 2
Generic course 6	...	GENLOC 2	and	GENGS3
Generic course 7	...	GENLOC3	and	GENGS1
Generic course 8	...	GENLOC3	and	GENGS 2
Generic course 9	...	GENLOC3	and	GENGS3

Table B-1. Glide Slope Localizer Courses Paired Together to Form ILS Courses.

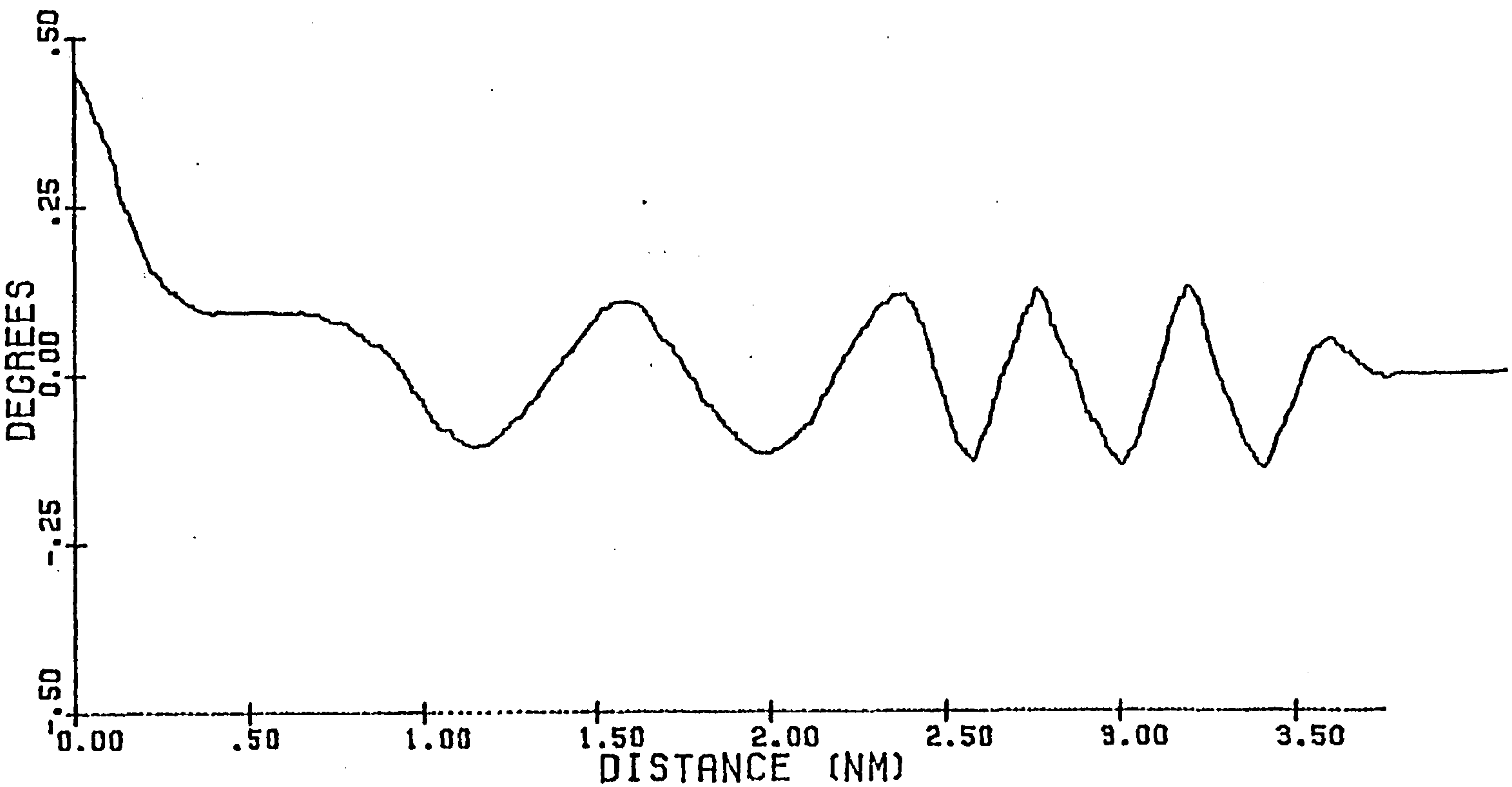

Figure B-1. Generic Glide Slope No. 1 (GENGS1).

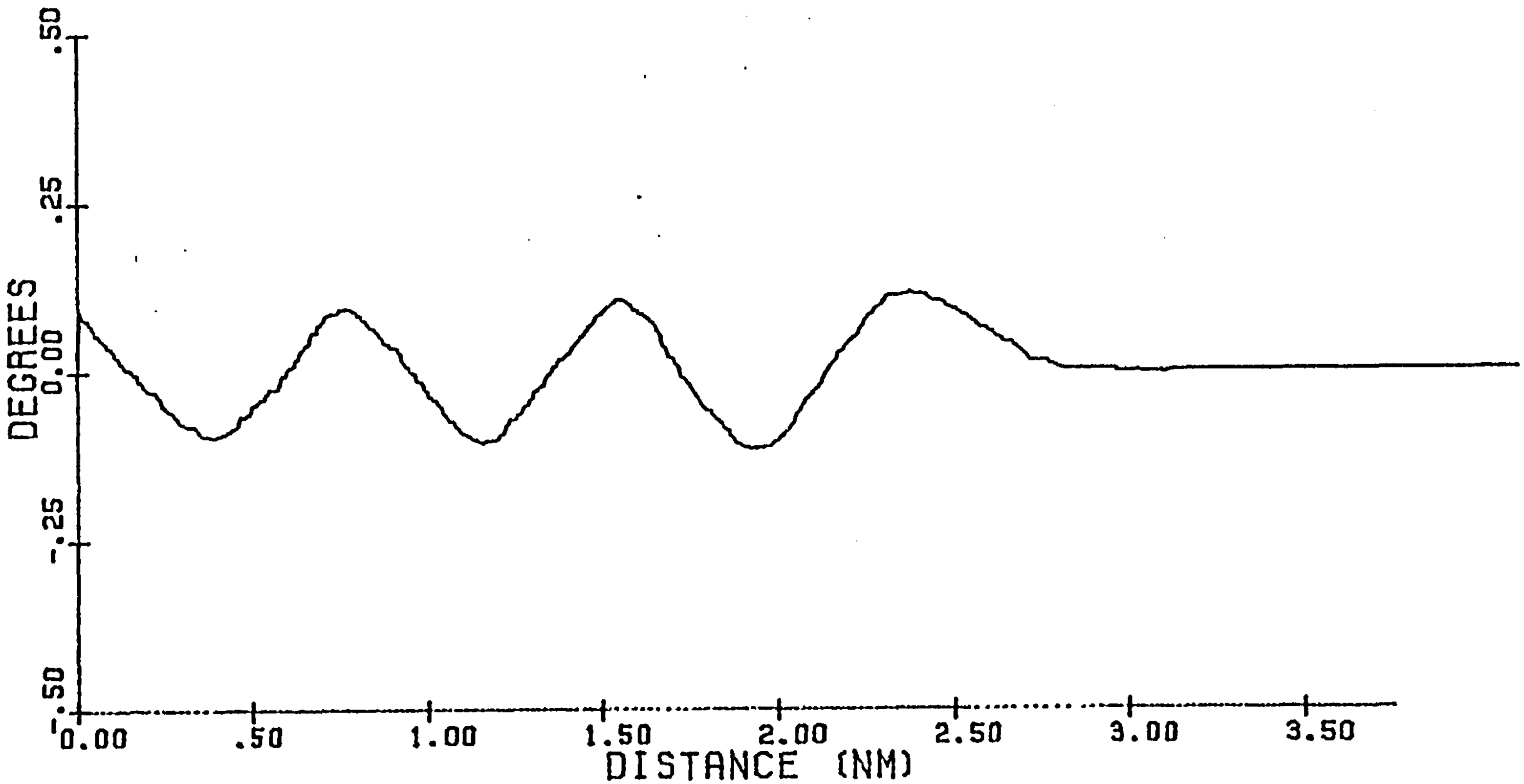

Figure B-2. Generic Glide Slope No. 2 (GENGS2).

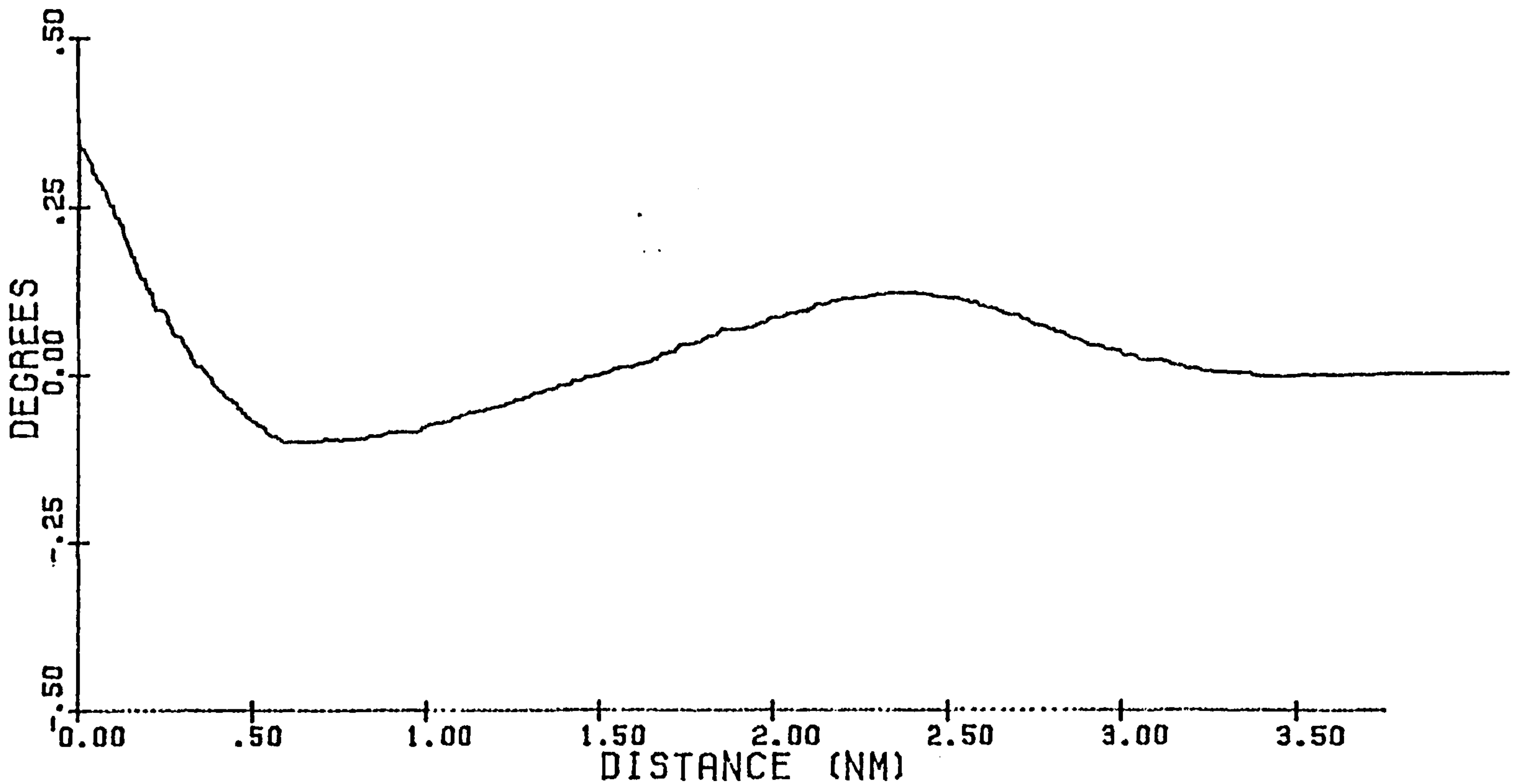

Figure B-3. Generic Glide Slope No. 3 (GENGS3).

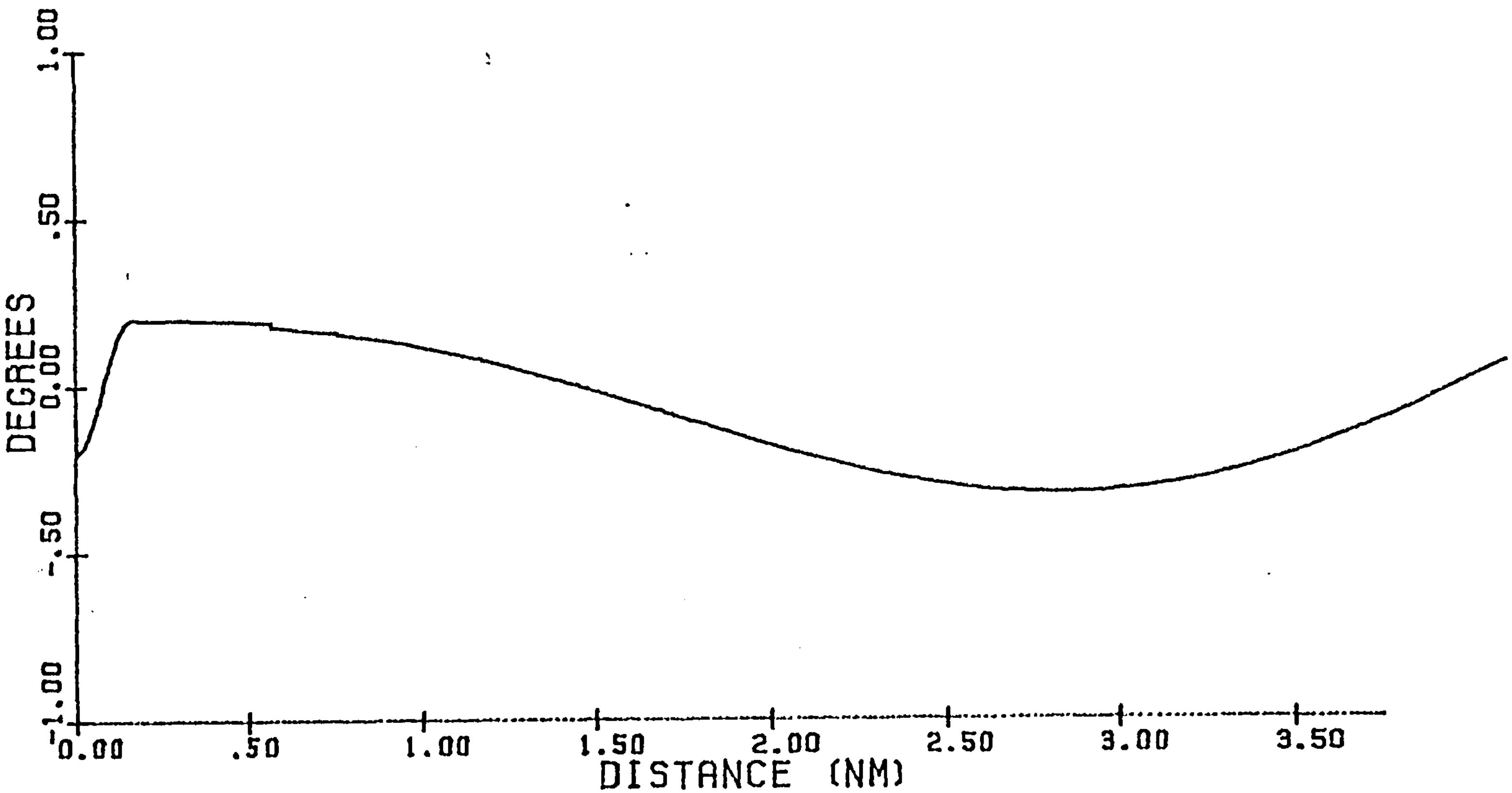

Figure B-4. Generic Localizer No. 1 (GENLOC1).

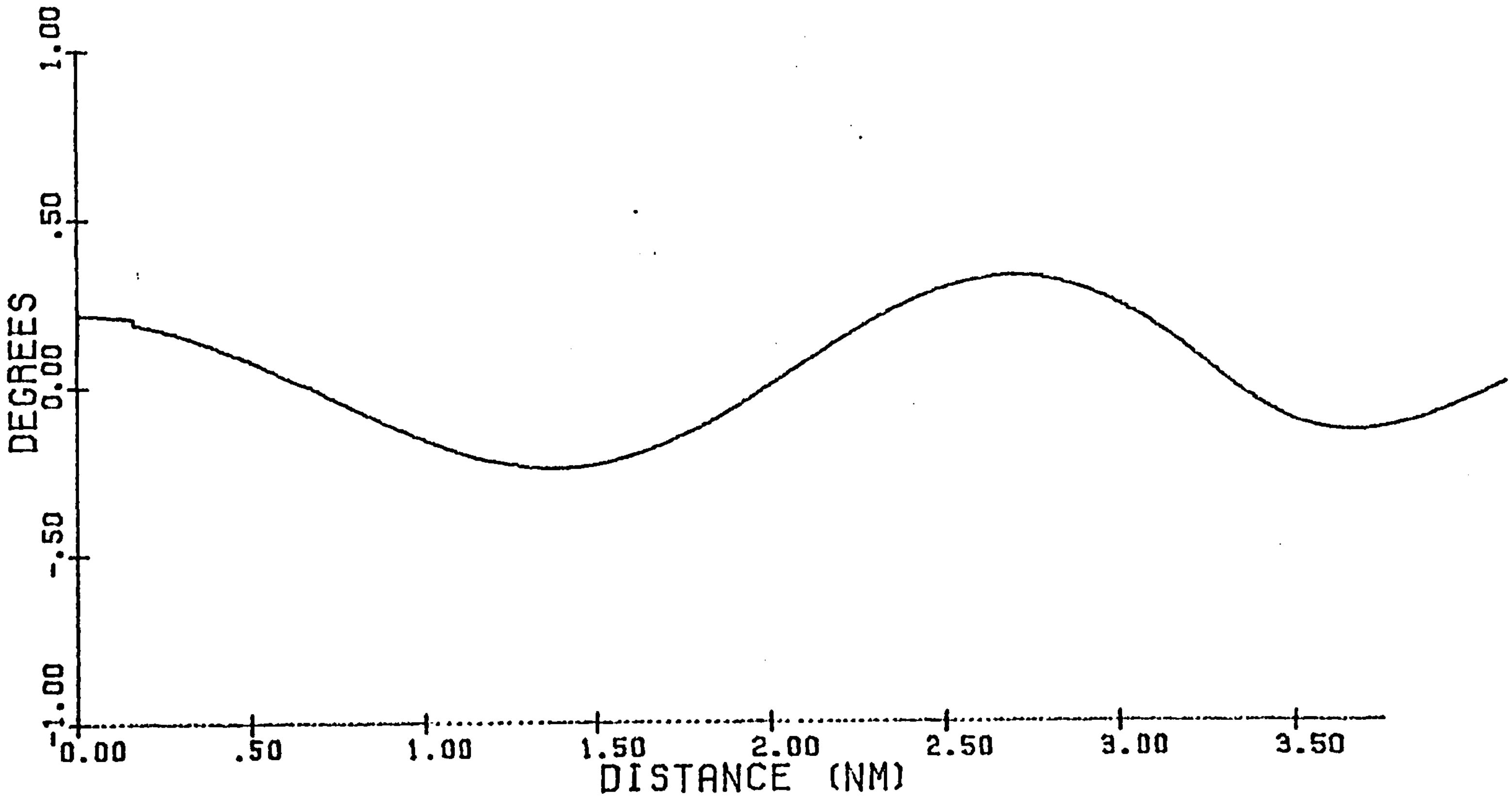

Figure B-5. Generic Localizer No. 3 (GENLOC2).

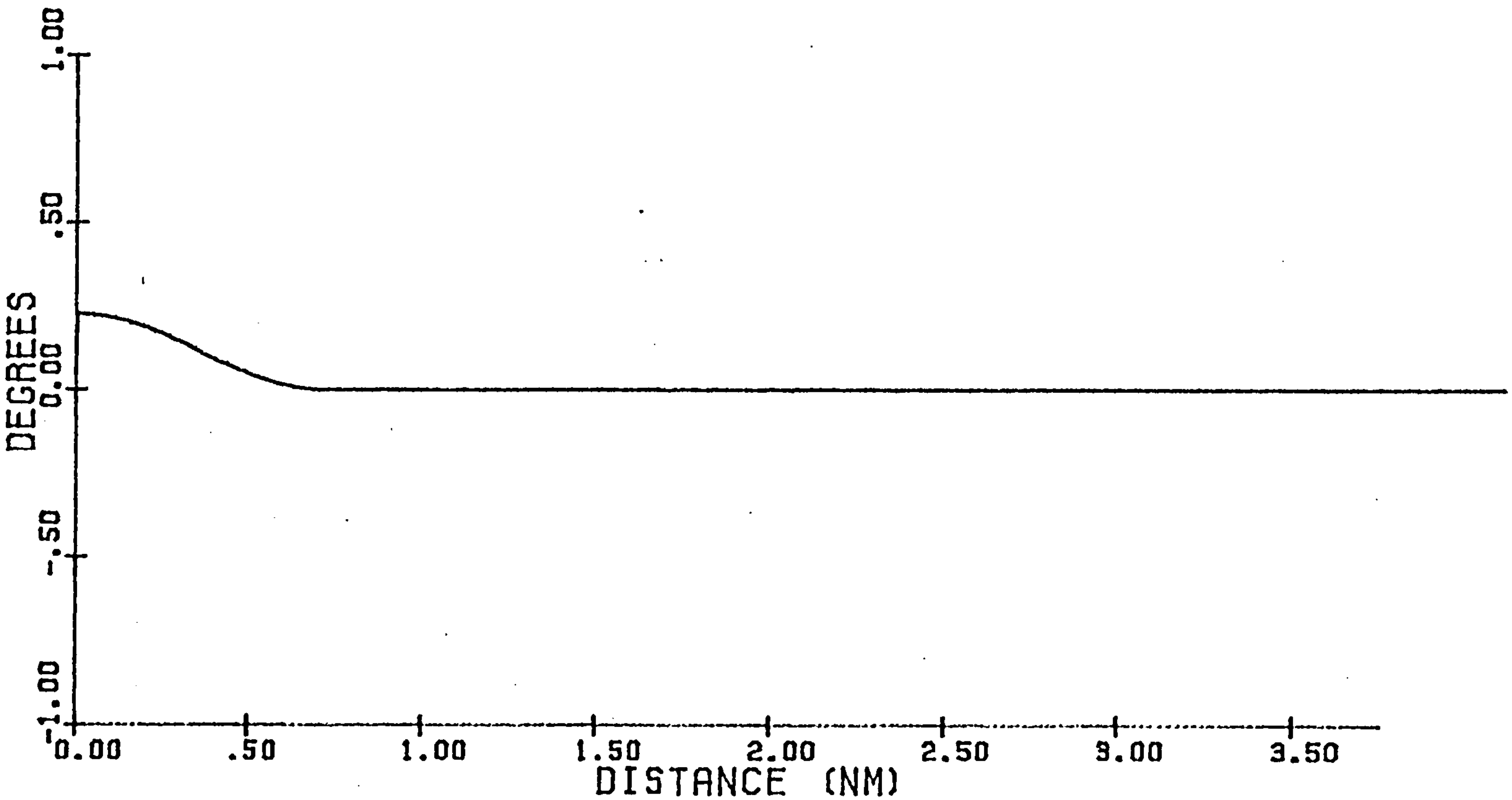

Figure B-6. Generic Localizer No. 3 (GENLOC3).

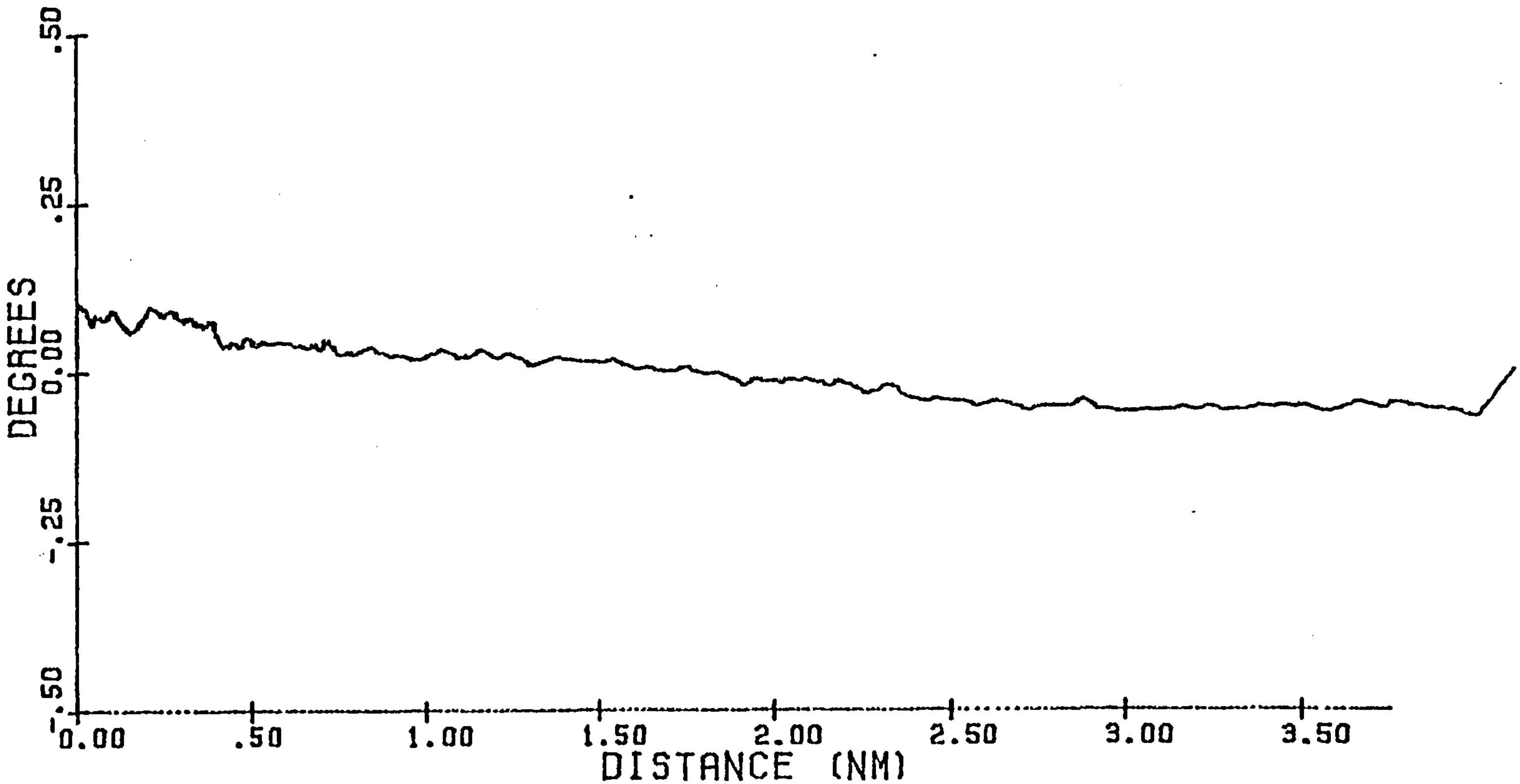

Figure B-7. Glide Slope Course 1 (GS1)

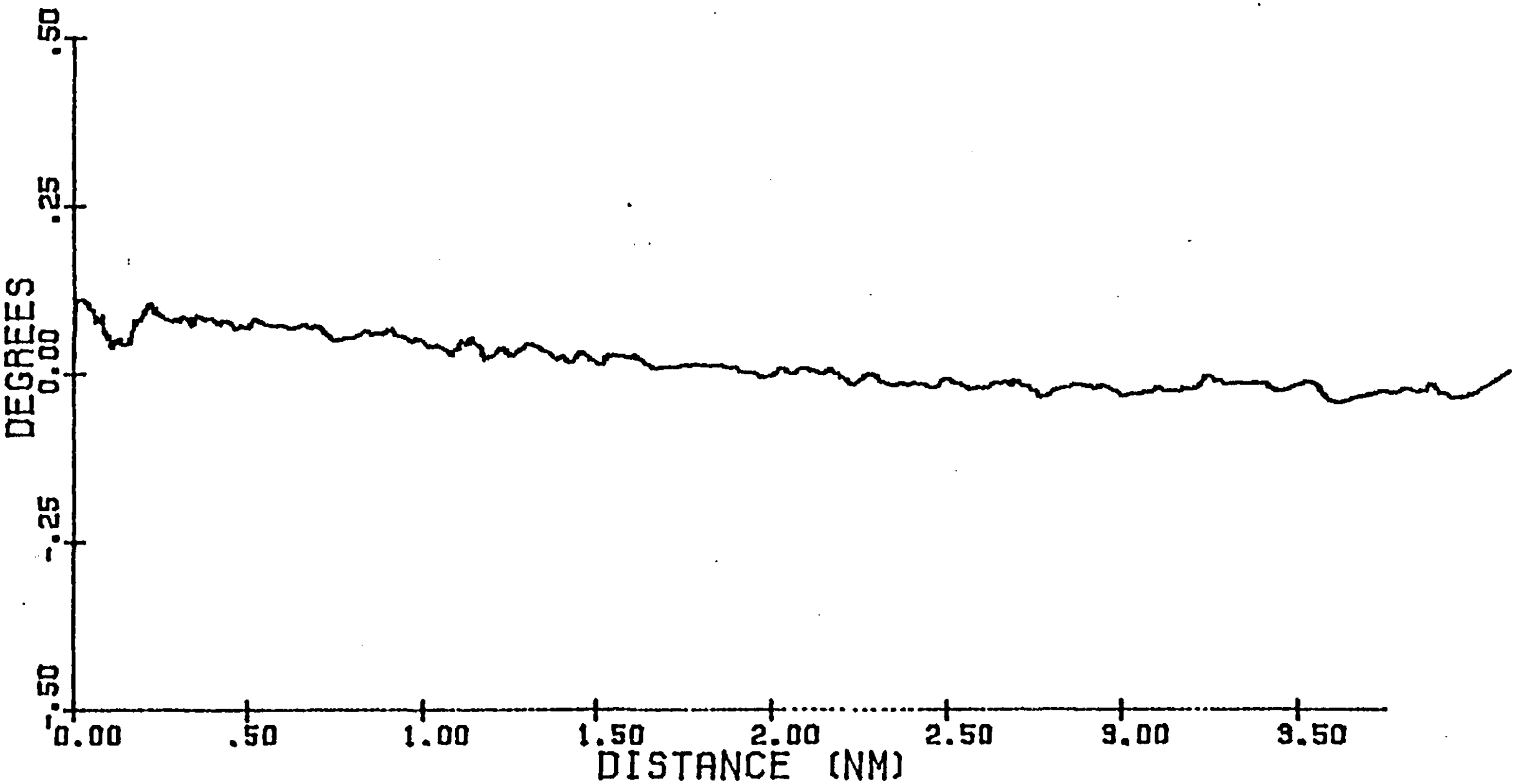

Figure B-8. Glide Slope Course 2 (GS2)

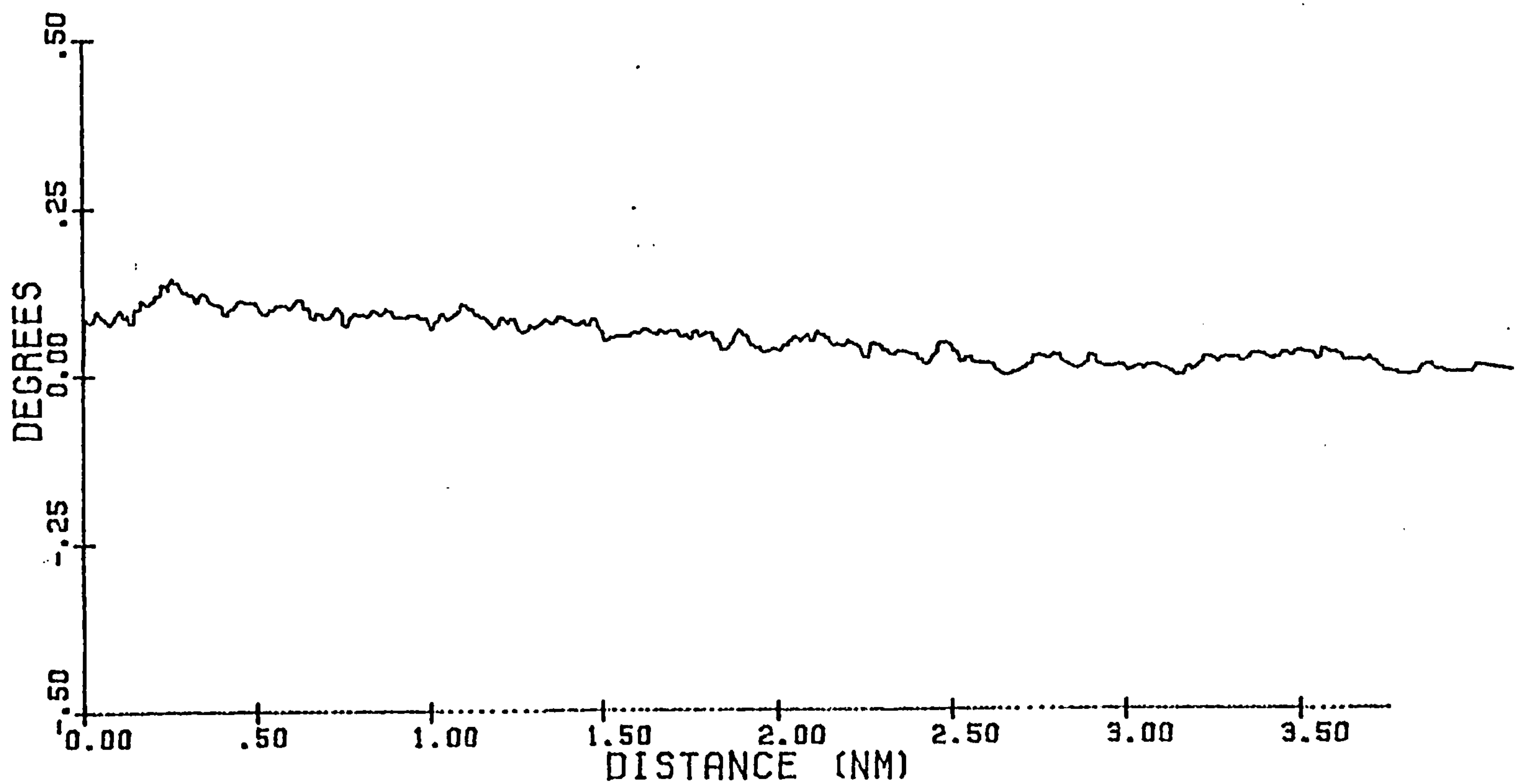

Figure B-9. Glide Slope Course 3 (GS3)

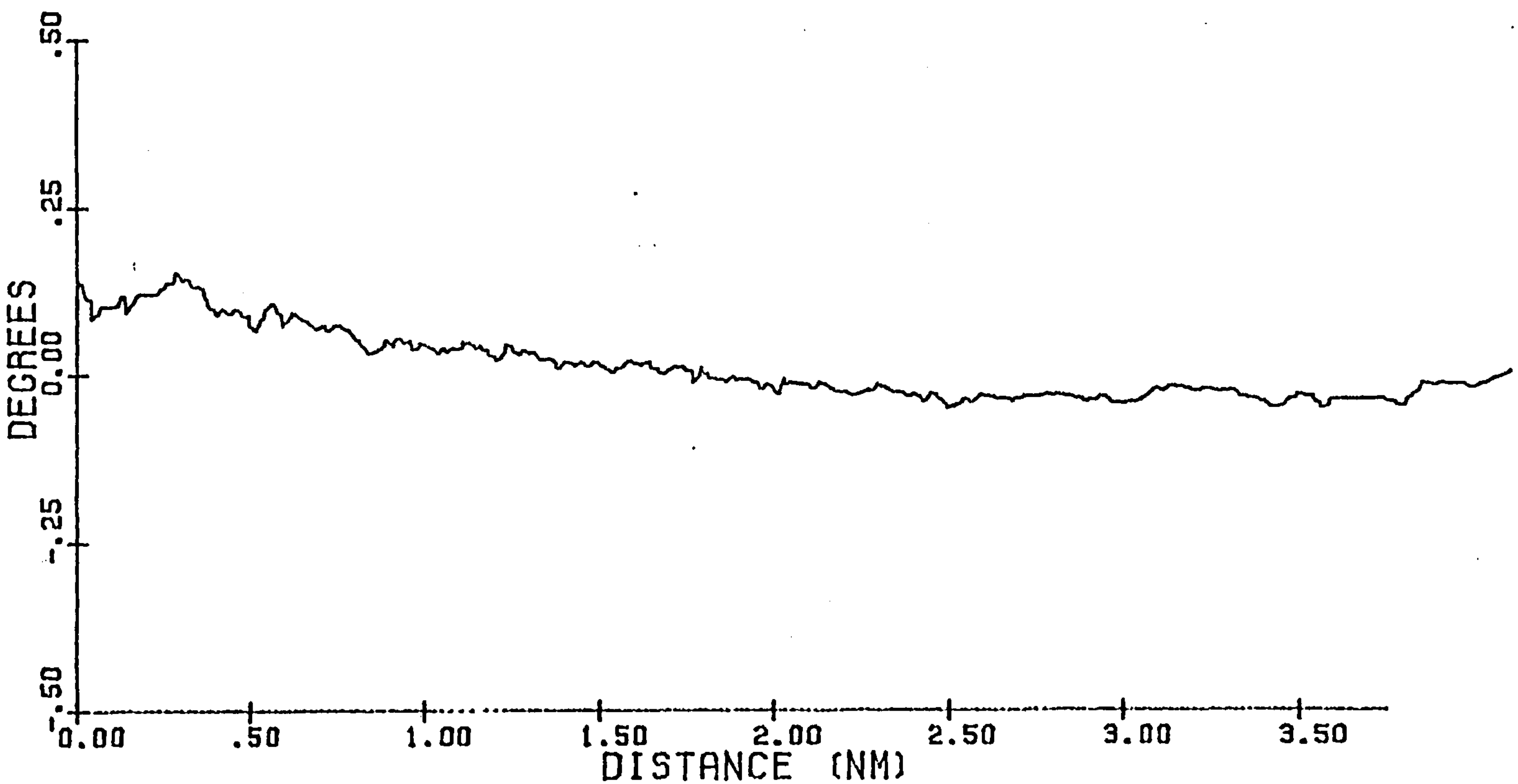

Figure B-10. Glide Slope Course 4 (GS4)

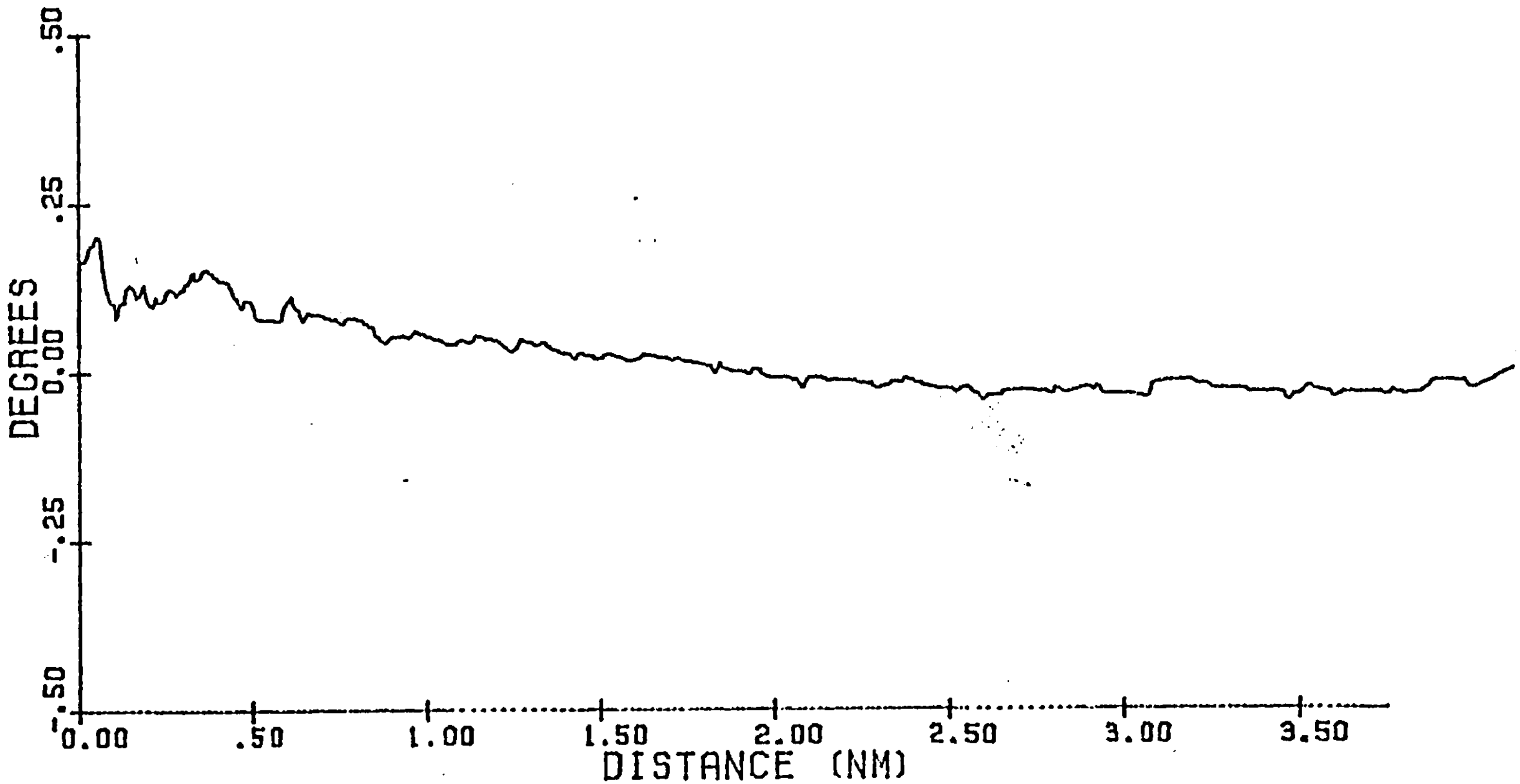

Figure B-11. Glide Slope Course 5 (GS5)

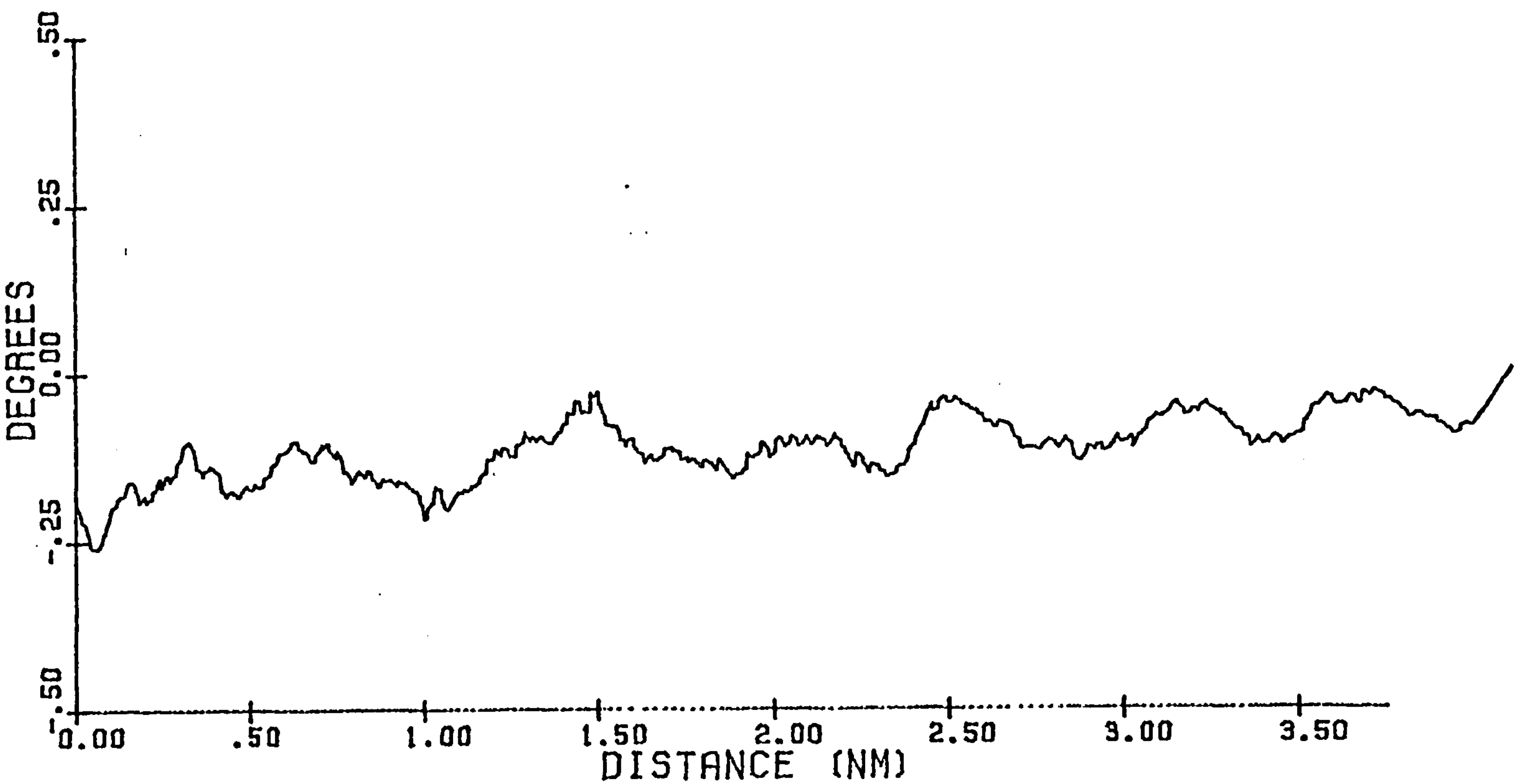

Figure B-12. Glide Slope Course 6 (GS6)

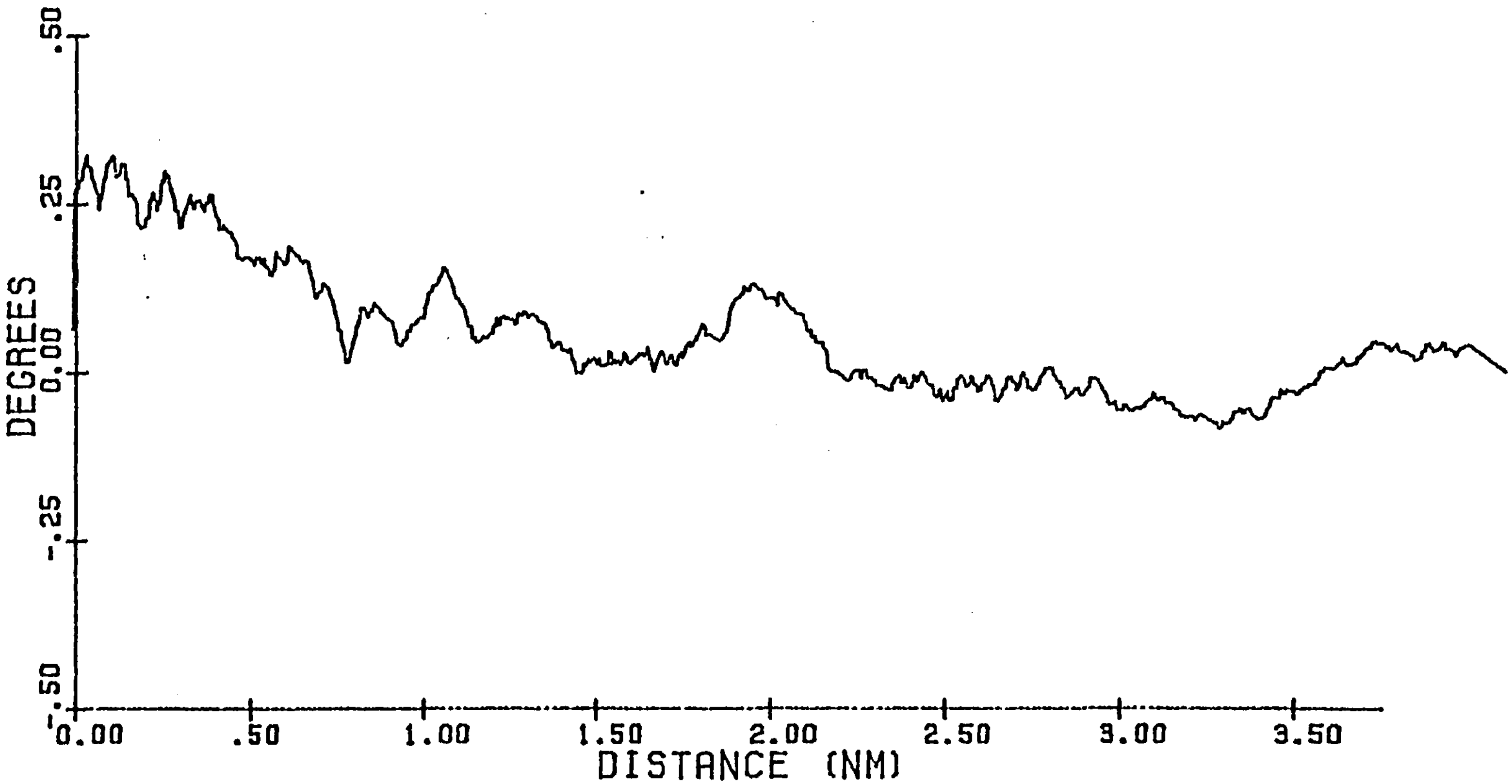

Figure B-13. Glide Slope Course 7 (GS7)

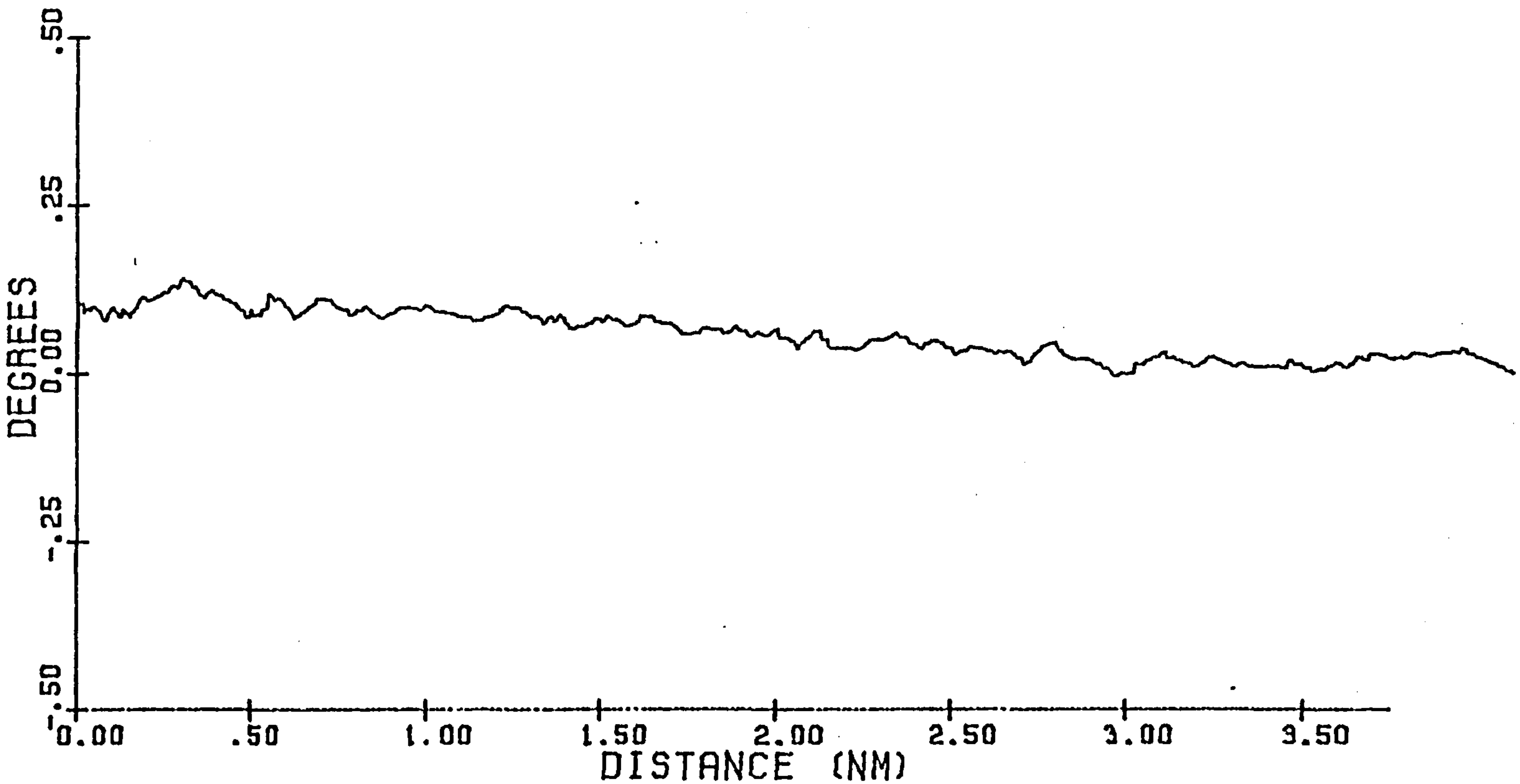

Figure B-14. Glide Slope Course 8 (GS8)

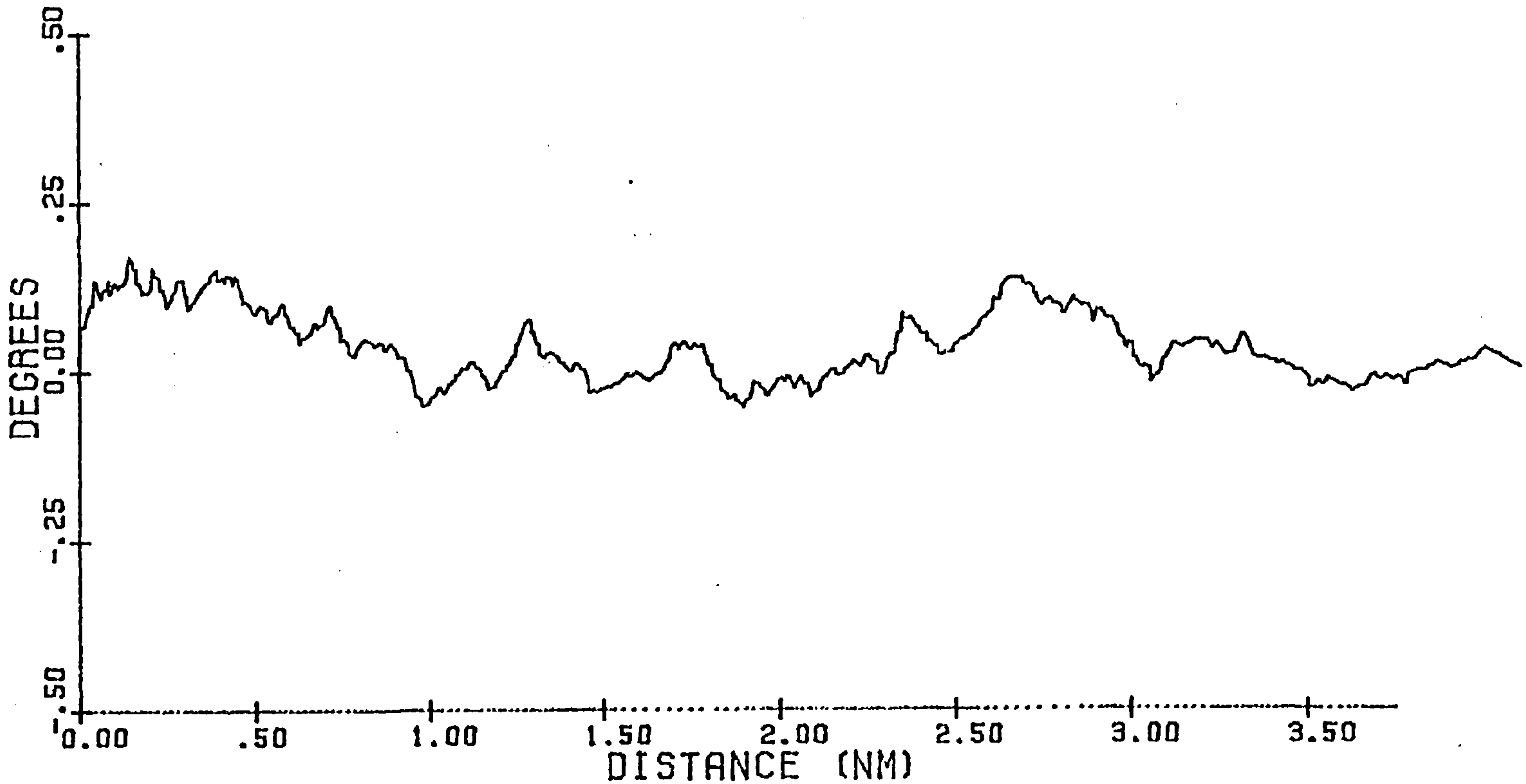

Figure B-15. Glide Slope Course 9 (GS9)

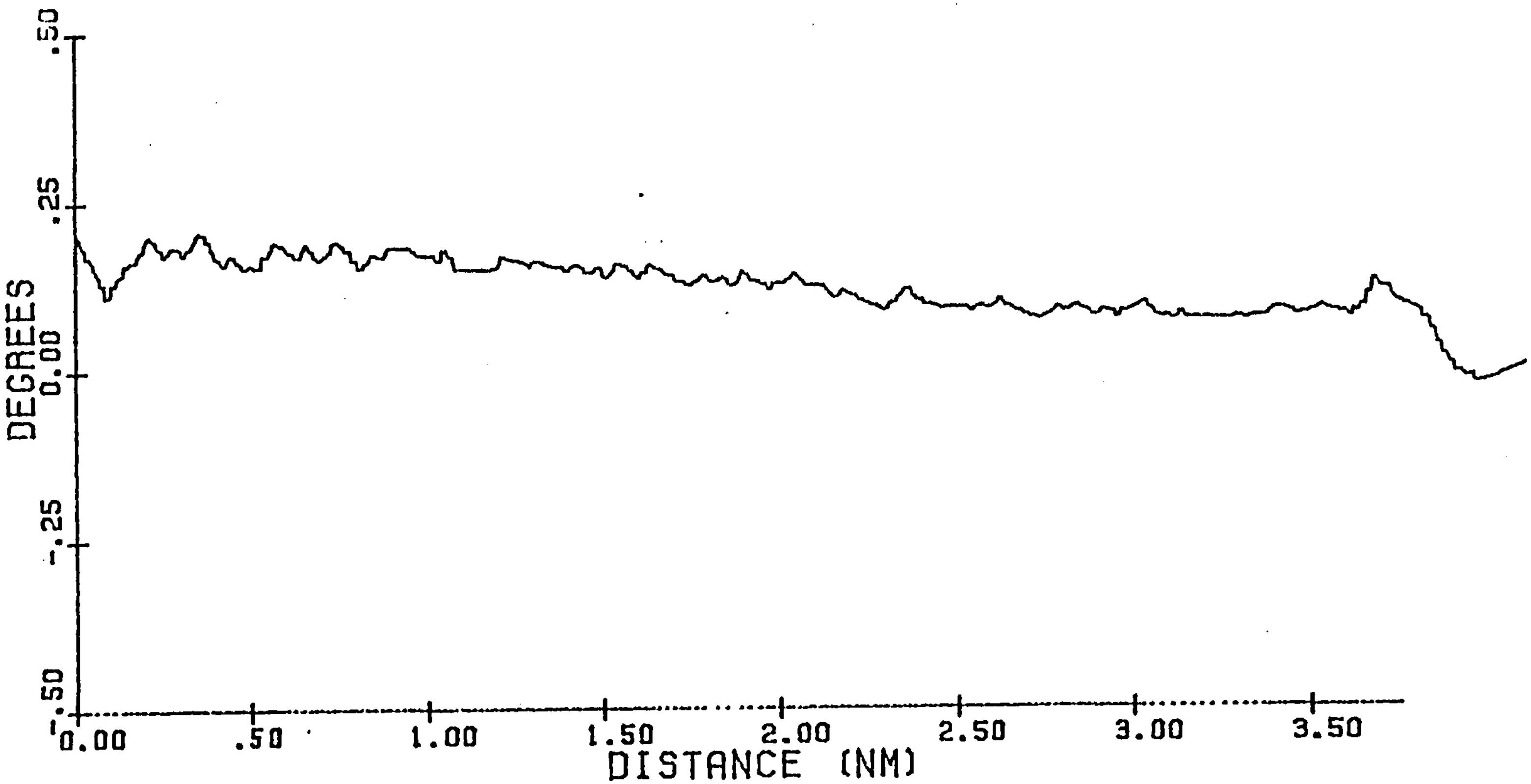

Figure B-16. Glide Slope Course 10 (GS10)

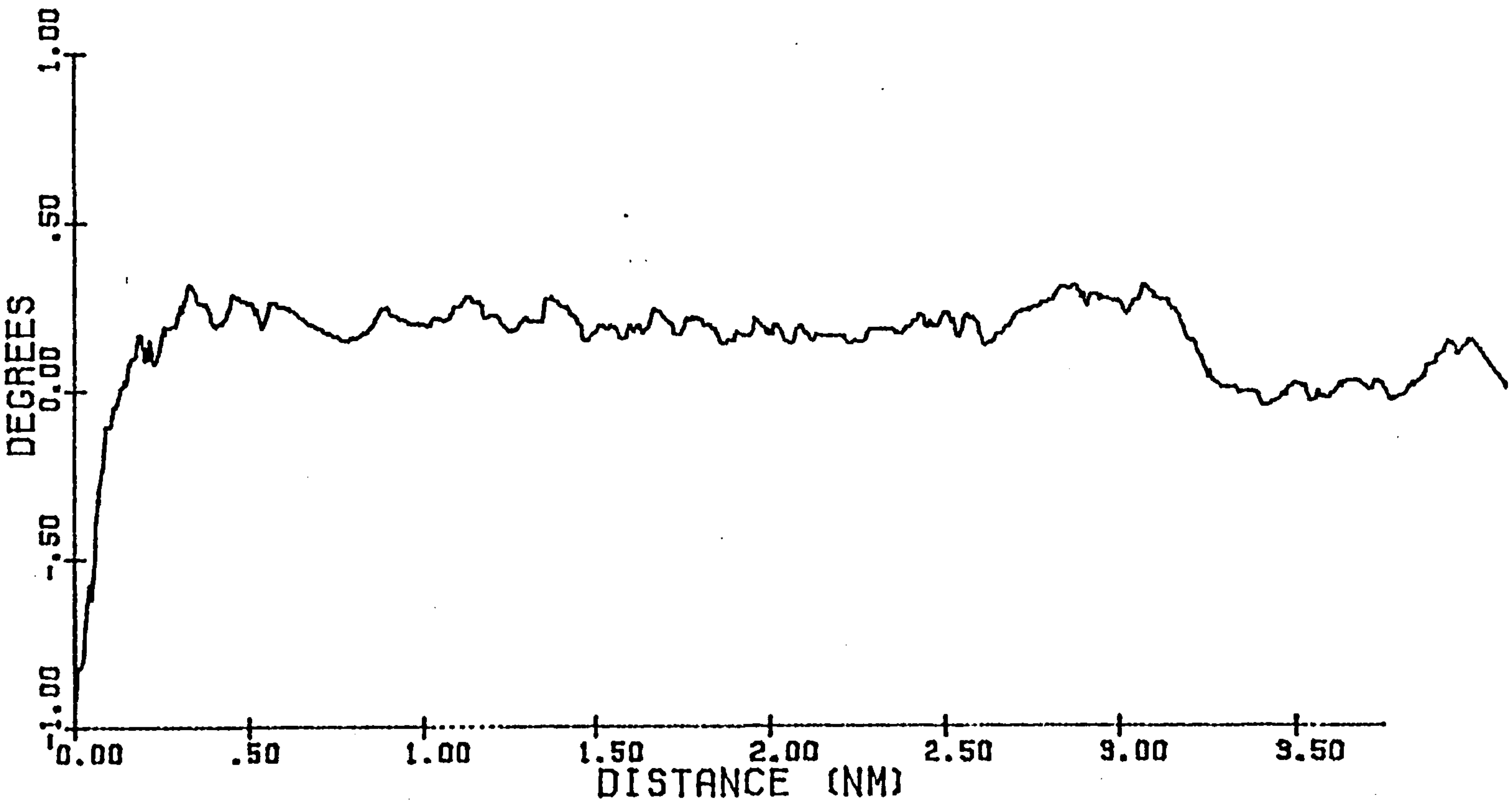

Figure B-17. Localizer Course 1 (LOC1)

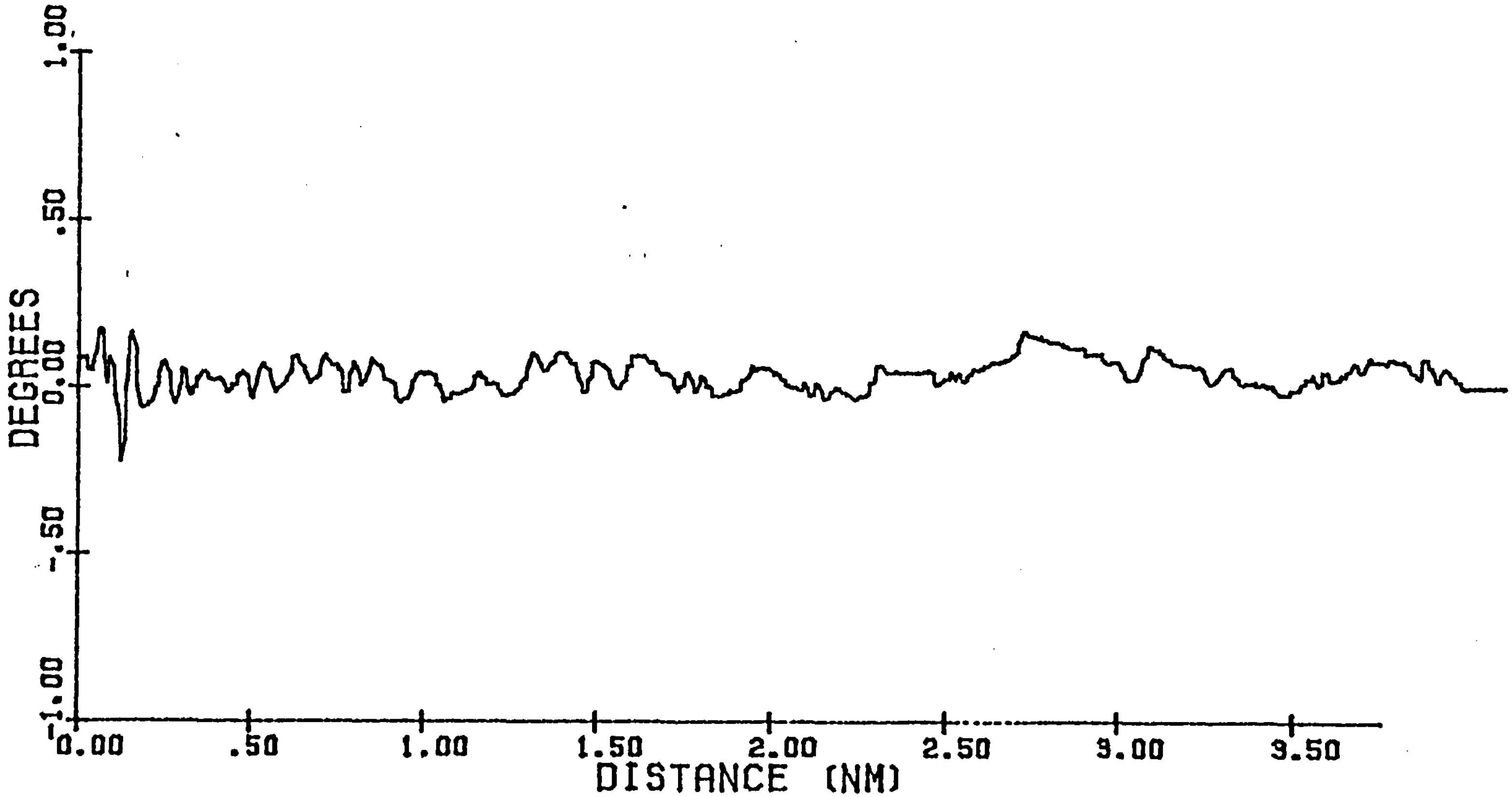

Figure B-18. Localizer Course 2 (LOC2)

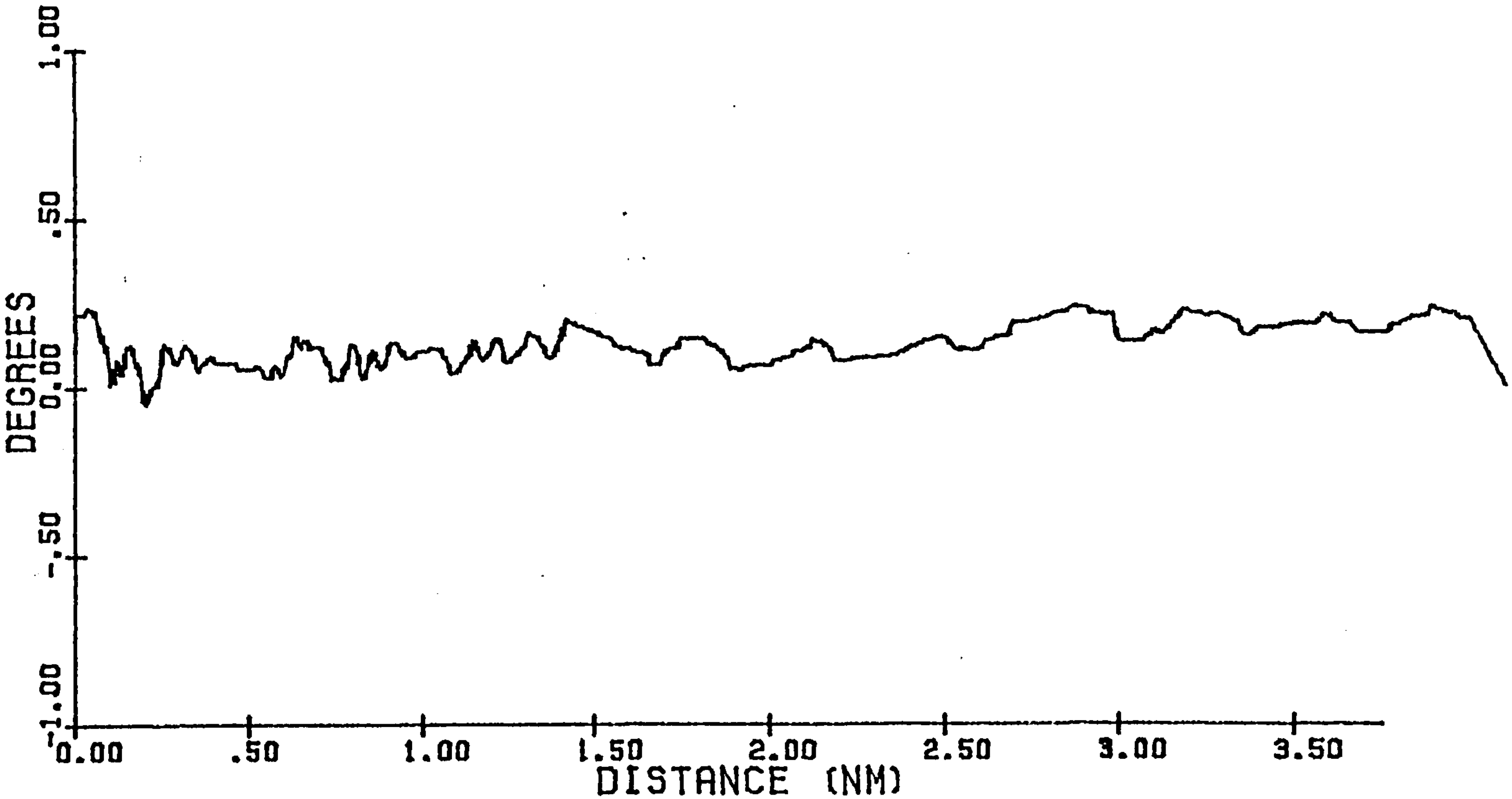

Figure B-19. Localizer Course 3 (LOC3)

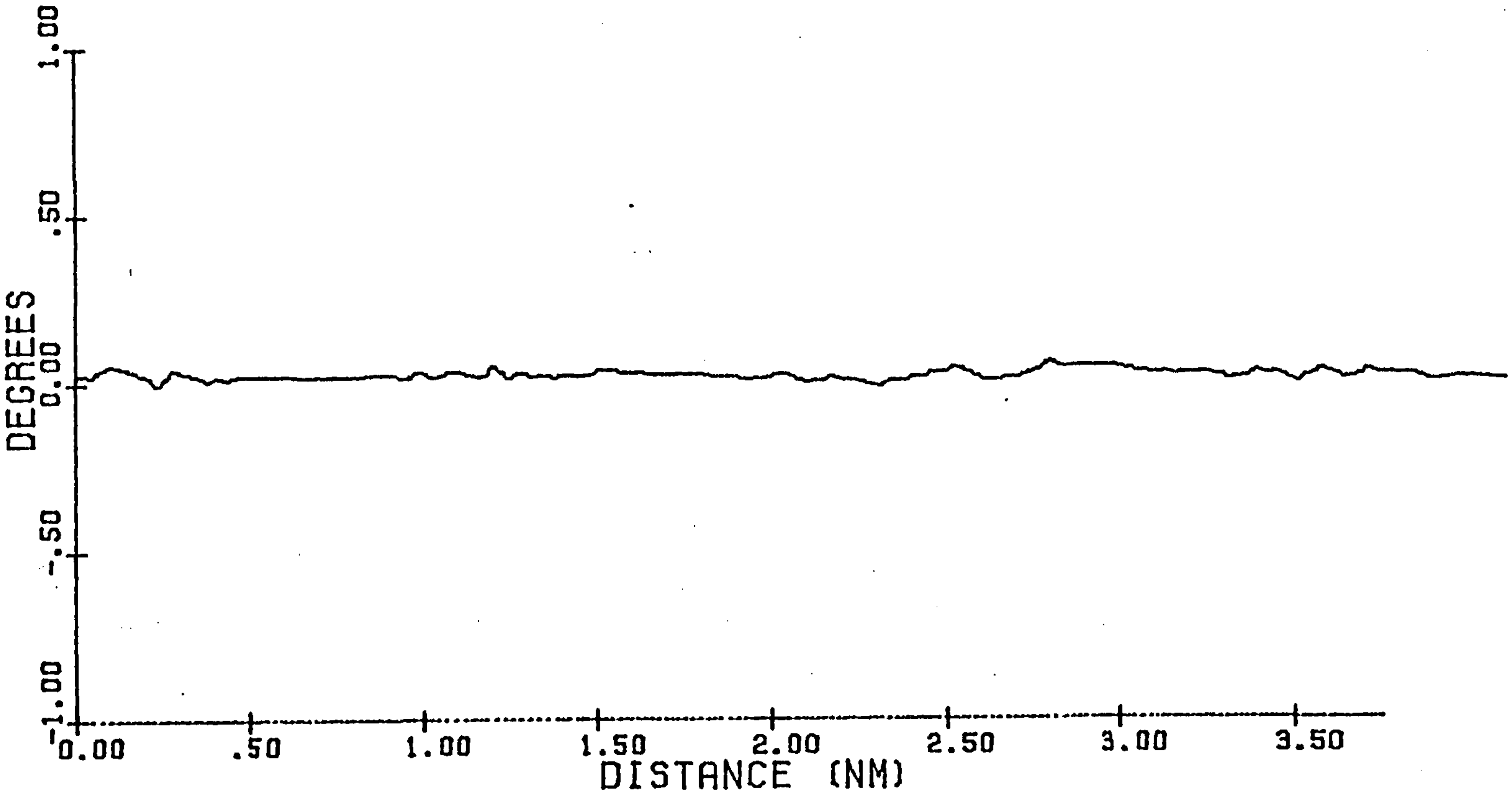

Figure B-20. Localizer Course 4 (LOC4)

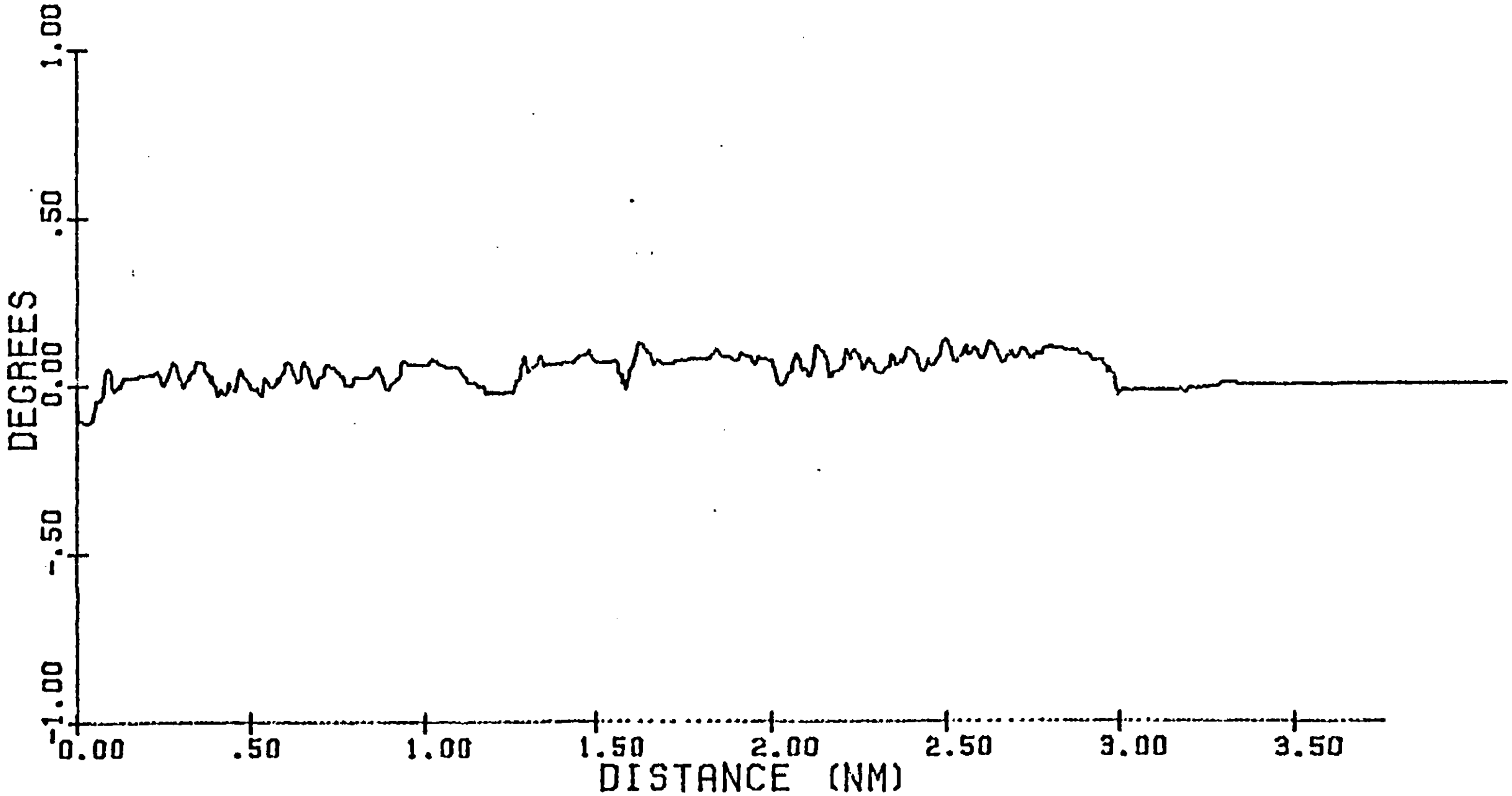

Figure B-21. Localizer Course 5 (LOC5)

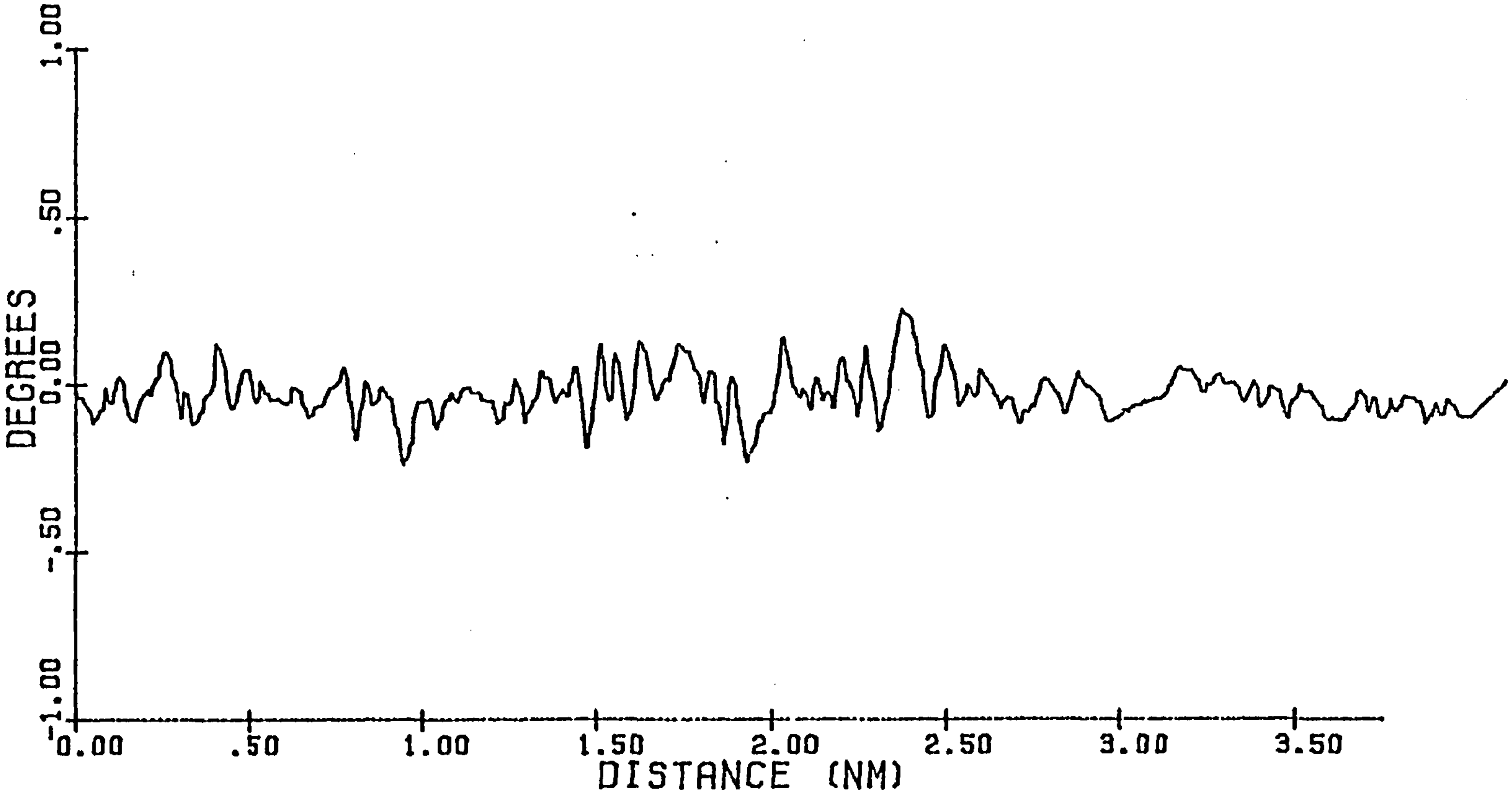

Figure B-22. Localizer Course 6 (LOC6)

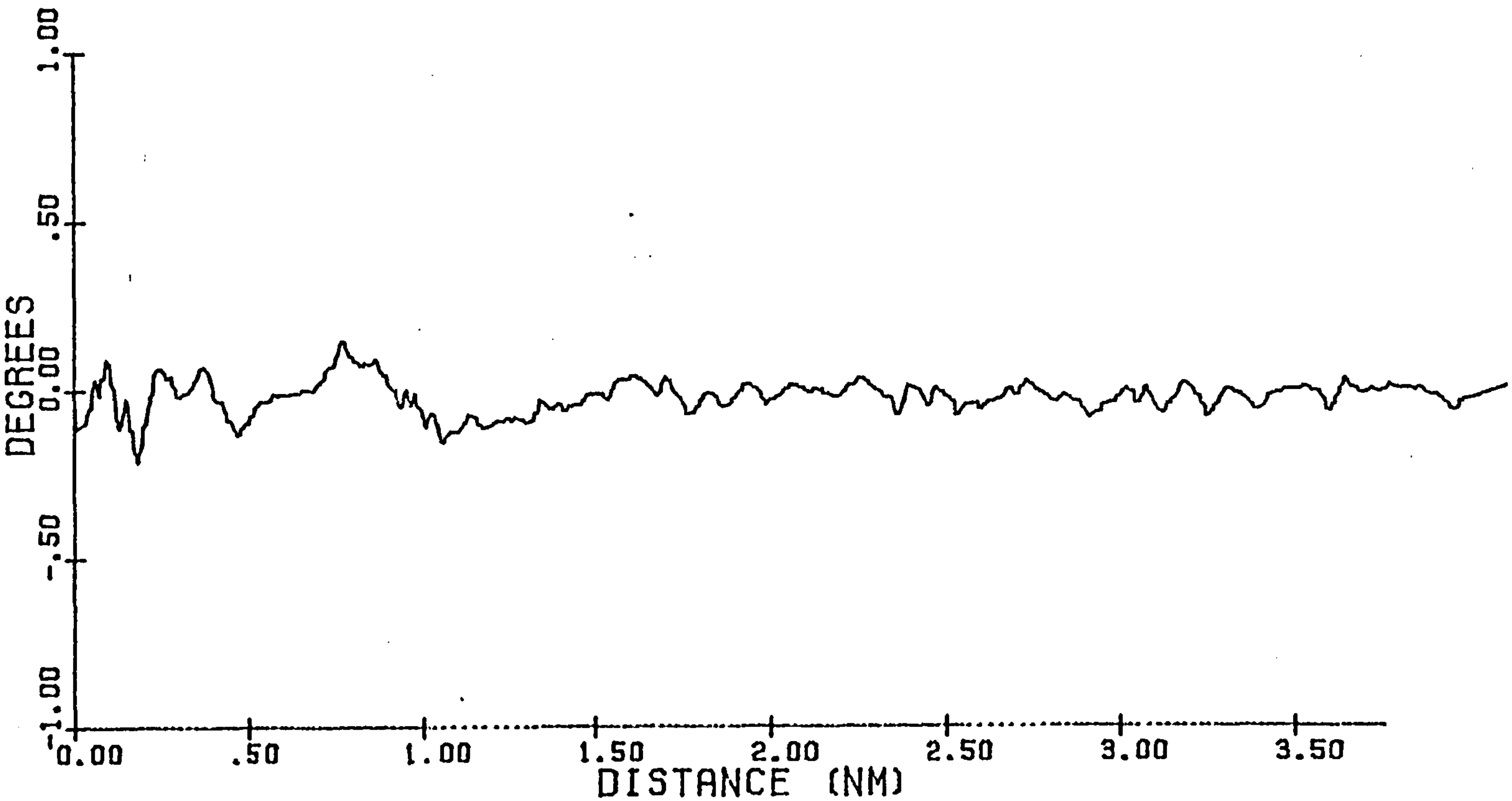

Figure B-23. Localizer Course 7 (LOC7)

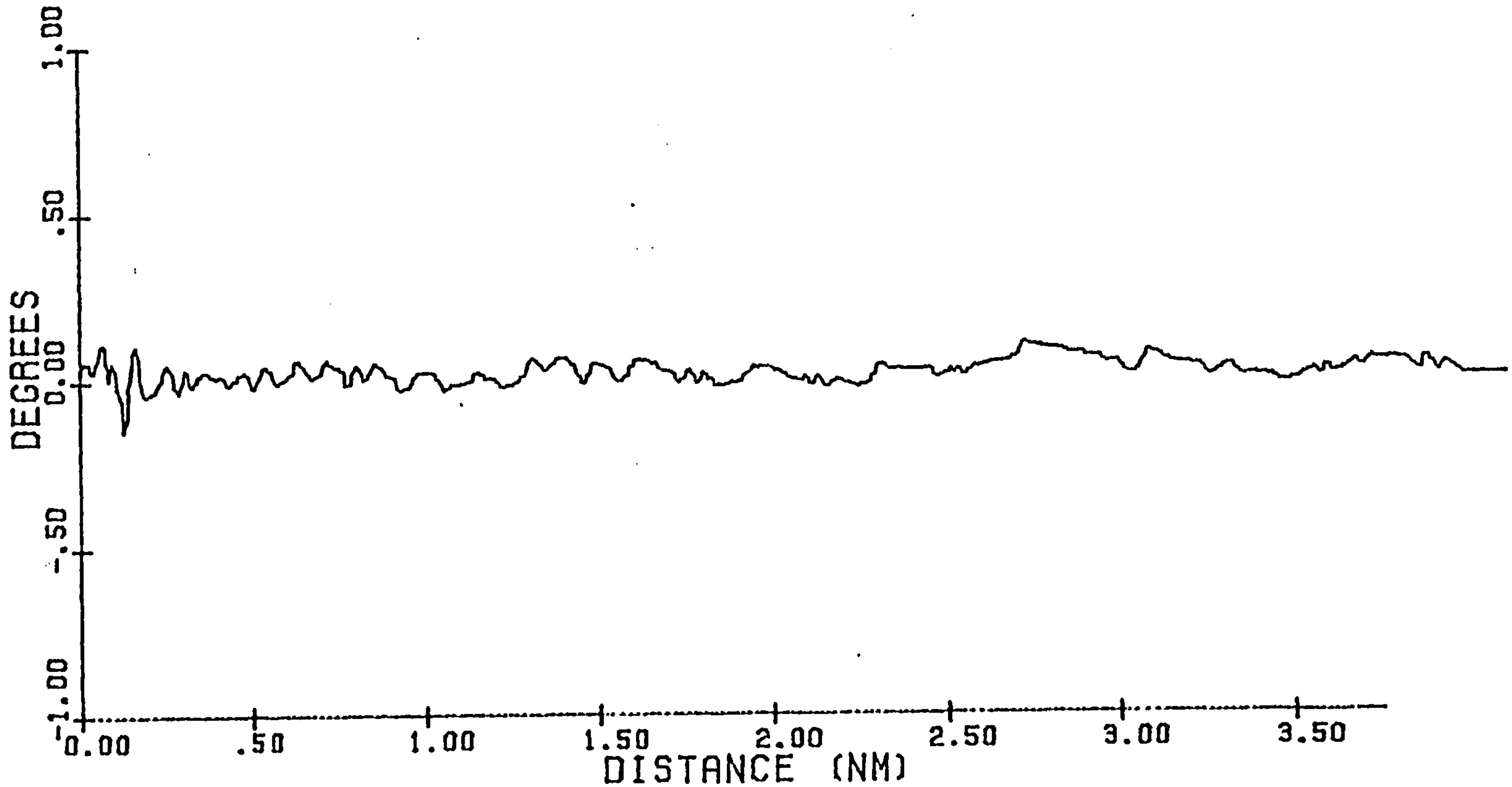

Figure B-24. Localizer Course 8 (LOC8)

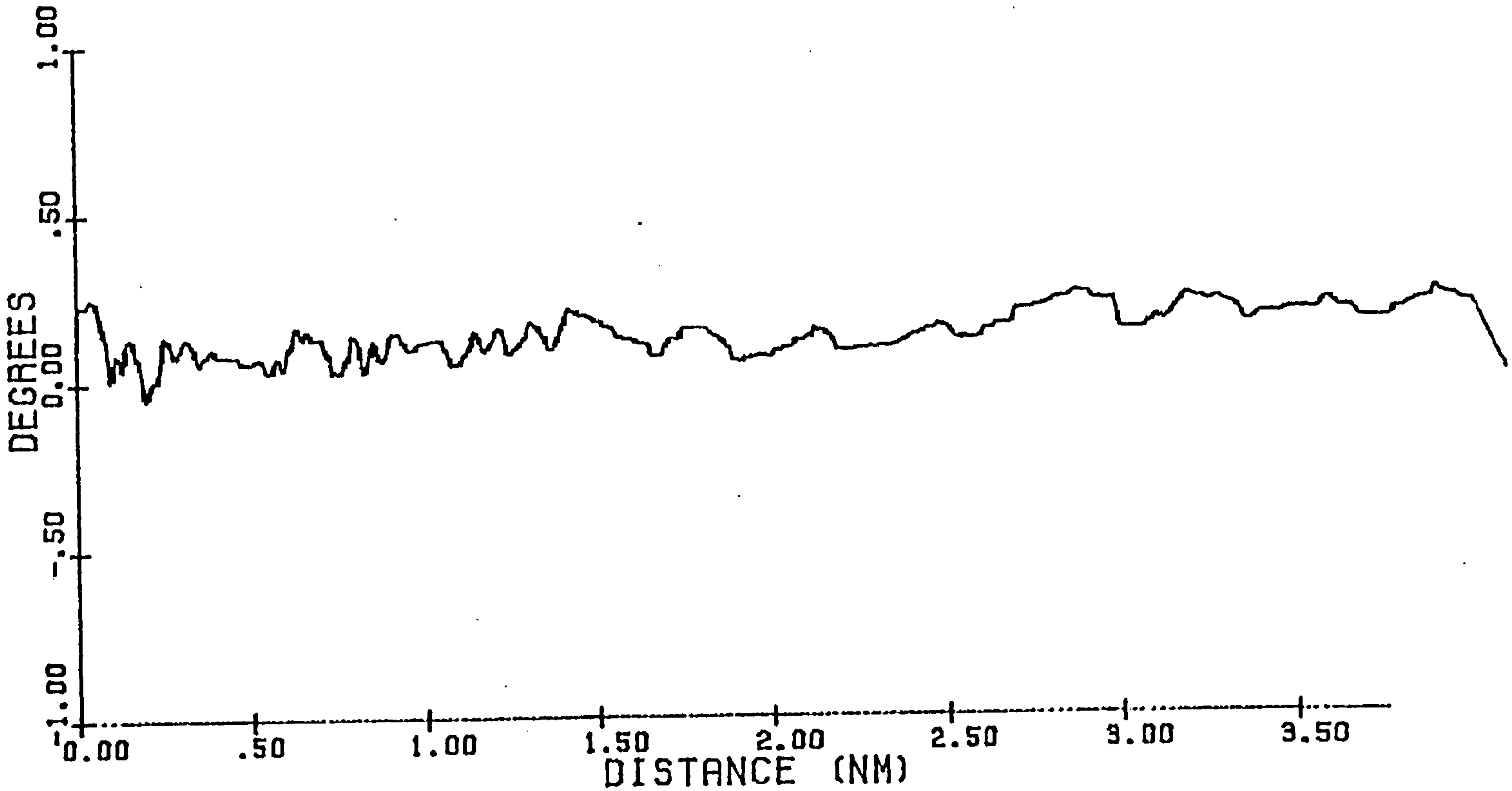

Figure B-25. Localizer Course 9 (LOC9)

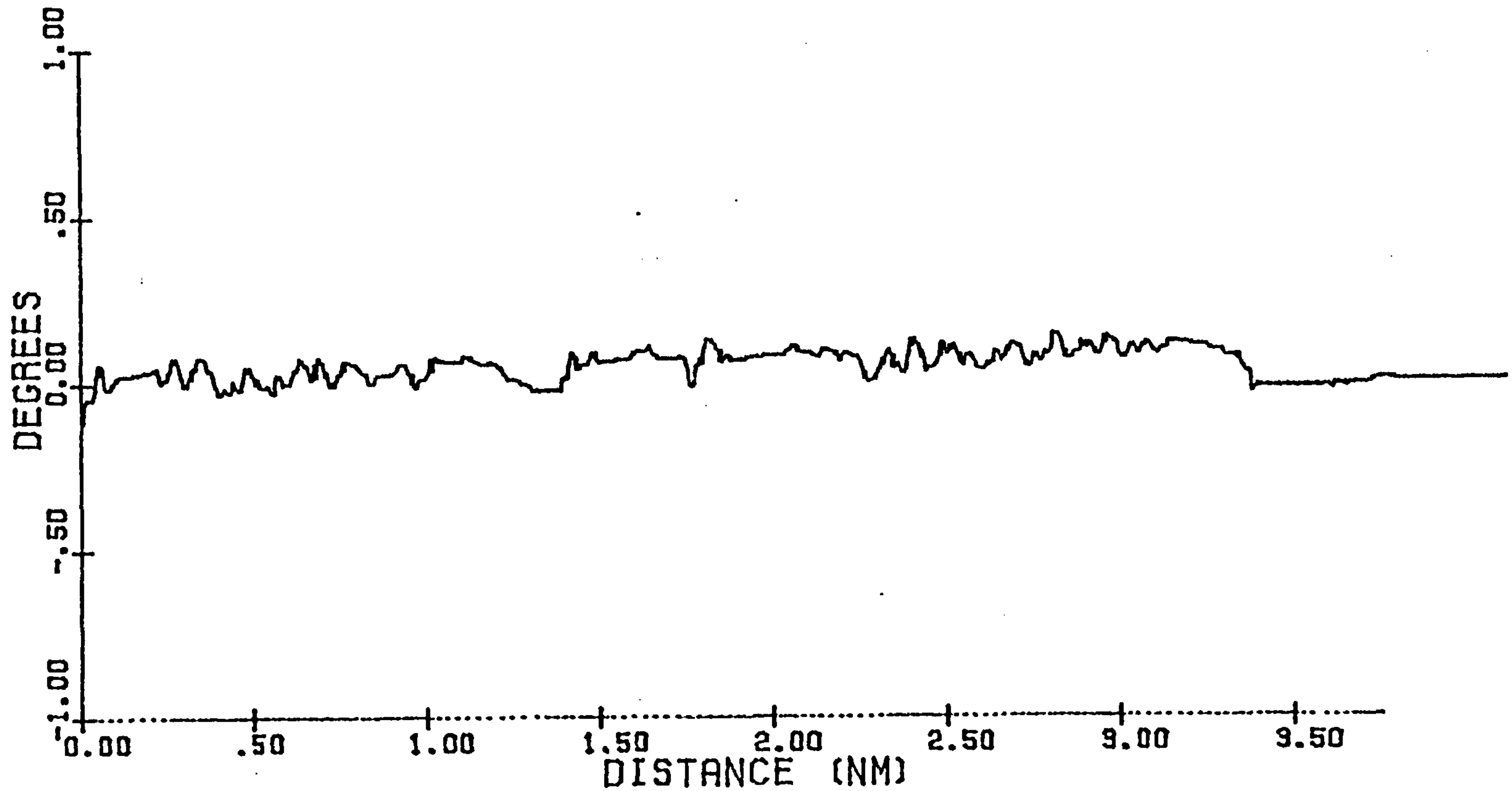

Figure B-26. Localizer Course 10 (LOC10)

3. <u>Appendix C.</u> Program Listings

This exec allows the IBM370 to read tapes created by the strip chart to digital data translator.

```
FI INMOVE TAP1 (RECFM U LRECL 20000 BLOCK 20000
FI OUTMOVE DISK A A C (RECFM U LRECL 20000 BLOCK 20000
MOVEFILE
XEDIT A A C
```

```
&CONTROL OFF
&IF .&1 EQ .? &GOTO -TELL
LISTFILE &1 &2 &3 (L NOHEADER STACK
&READ VARS &A &B &C &D &E &F &G &H &I &J &K
FI 3 DISK &1 &2 &3 (RECFM V LRECL 20000 BLOCK 20000
FI 7 DISK C&1 &2 &3
FI 9 TERM
&TYPE FILE LENGTH &E
&STACK &E
EXEC LODE MAS5
&TYPE CONVERSION COMPLETE...FILE C&1 &2 &3 CREATED.......
&EXIT
-TELL
&BEGTYPE
* ...THIS EXEC TAKES A RECORD AS PRODUCED BY THE ANALOG TO DIGITAL
*    DATA TRANSLATOR AND CONVERTS IT TO A MORE USEFUL FORM.
*    THE EXEC DETERMINES THE LENGTH OF THE FILE, SETS UP THE PROPER
*    FILE DEFINITIONS AND THEN CALLS THE PROGRAM MAS5.  MAS5 CONVERTS
*    THE FILE FROM BCD TO INTEGER REPRESENTATION.   THE EXEC PASSES
*    THE LENGTH OF THE VARIABLE LENGTH RECORD TO THE PROGRAM .
*    USE OF THIS EXEC CREATES A NEW FILE AND DOES NOT DESTROY THE
*    INPUT RECORD.   EXEC ASSUMES 1 RECORD IN FILE.  FILE CREATED HAS
*    SAME FN FT FM EXCEPT A 'C' IS ADDED TO THE BEGINNING OF THE FILE
*    NAME.
&END
&EXIT
```

```
C   THIS PROGRAM CONVERTS FILES WRITTEN IN BCD FORMAT BY THE ANALOG TO
C   DIGITAL DATA TRANSLATOR TO FORTRAN-PASCAL COMPATIBLE INTEGER FORMAT.
C   THE PROGRAM REQUIRES THE LENGTHT OF THE RECORD TO BE PROCESSED TO
C   BE CONTAINED IN I4 FORMAT AT THE BEGINNING OF THE FIRST RECORD.
C   USE OF THE CONVERT EXEC IS RECOMMENDED.
        INTEGER*4 L(4000),EVENT,N3,N2,N1
        LOGICAL*1 K(20000)
        LOGICAL*1 LBUFF(4)
        EQUIVALENCE (LBUFF(1),INTGER)
C   THE PACKED BCD FORMAT IS UNPACKED INTO INTEGER NUMBERS 1 BYTE
C   AT A TIME THROUGH THIS EQUVLENT LOGICAL/INTEGER VARIABLE.
        READ(9,10)NN
C    LENGTH OF VARIBLE LENGTH RECORD IS READ
10      FORMAT(I5)
        READ(3,300,END=999)(K(I),I=1,NN)
C   ENTIRE RECORD IS READ INTO LOGICAL*1 ARRAY.
300     FORMAT(150(10(11A1)))
999   CONTINUE
        N=I/4
C NUMBER OF INTEGER NUMBERS EXPECTED TO RESULT FROM BCD INPUT OF LENGTH
C NN IS CALCULATED.
        WRITE(7,30)N
C NUMBER OF VALUES IN THE OUTPUT FILE IS THE FIRST RECORD IN THE OUTPUT
C FILE.
      DO 20 I=1,N
C DO THE CONVERSTION FOR AS MANY VALUES AS EXPECTED.
       INTGER=0
C CLEAR OUT EQUIVALENT VARIABLE.
       II=I*4-3
C INDEX BY FOUR AND SKIP THE FIRST BYTE.
        LBUFF(4)=K(II+1)
C STEP THROUGH BYTE BY BYTE FOR FOUR BYTES
        EVENT=INTGER
        LBUFF(4)=K(II+2)
        N3=INTGER
        LBUFF(4)=K(II+3)
        N2=INTGER
        LBUFF(4)=K(II+4)
        N1=INTGER
        L(I)=EVENT*1000+N3*100+N2*10+N1
C  CALCULATE THE INTEGER EQUIVALENT OF THE LAST FOUR BYTES OF BCD
C  CODED DATA
20      CONTINUE
        WRITE(7,30)(L(I),I=1,N)
C  PLACE IN OUTPUT FILE
30       FORMAT(' ',10I6,/)
1000   FORMAT(1X,I10)
          END
```

```
(*
***************************************************************************
*                                                                         *
*  THIS PROGRAM IS DESIGNED TO PROCESS COURSE STUCTURES FOR THE NASA      *
*  SIMULATION PROJECT.  THE PROGRAM IS DESIGNED TO READ FILES CREATED     *
*  BY THE 'CONVERT' PROGRAM.  THIS PROGRAM THEN TAKES THE VALUES AND      *
*  SCALES THEM TO DEGREES (SEE DOCUMENTATION ON DIGITIZING PROCESS)       *
*  THEN THIS PROGRAM USES THE EVENT MARKES IN THE CONVERTED FILE TO       *
*  DETERMINE THE DISTANCE OF EACH POINT FROM THE RUNWAY THRESHOLD.        *
*  (IN FEET).   NEXT THE PROGRAM CREATES A FILE OF 1000 POINTS WHICH      *
*  THROUGH AN INTERPOLATION PROCESS HAVE BEEN ADJUSTED TO BE 25 FEET      *
*  APART. THIS PROVIDES A COURSE STUCTURE WHICH IS JUST OVER 4 NM LONG.   *
*  FINALLY THE VAULES IN THIS TABLE ARE AGAIN RESCALED BY: FIRST          *
*  NORMALIZING BY THE WIDTH OF THE LOCALIZER PATH AT THE ACTUAL           *
*  INSTALLATION FROM WHICH THE DIGITIZED DATA WAS TAKEN, AND THEN         *
*  MULTIPLING BY THE LOCALIZER PATH WIDTH WHICH IS ASSUMED BY THE         *
*  NASA SIMULATOR (4 DEGREES).   THIS RESCALING ALLOWS THE DATA POINTS IN*
*  THE TABLE TO HAVE THE SAME (RELATIVE) SIGNIFICANCE AS THEY WOULD AT    *
*  THE INSTALLATION.  THE USER IS PROMPTED TO ENTER THE LOCALIZER PATH    *
*  WIDTH FOR THIS SCALING OPERATION.                                      *
*                                                                         *
***************************************************************************)
PROGRAM SCALVERT(INPUT,OUTPUT);


CONST
     (* DX IS THE INCREMENTAL DISTANCE FOR THE INTERPOLATION *)
     DX=25.0;



VAR
     NUM:INTEGER;            (* NUMBER OF SAMPLES IN INPUT *)
     ECNT:INTEGER;           (* NUMBER OF EVENT MARKS *)
     POSITION:ARRAY(.1..5000,1..2.) OF REAL;
                             (* POSITION AND ANGLE STORAGE FOR FIRST PASS *)
     RAWSTUF:ARRAY(.1..5000.) OF REAL;
                             (* RAW, UNSCALED INPUT DATA FROM CONVERT *)
     I,II,III:INTEGER;         (* GLOBAL INDEXERS *)
     EVENT:ARRAY(.1..30.) OF INTEGER;
                               (* STORAGE FOR EVENT POSITIONS *)
     COR:ARRAY(.1..1000.) OF REAL;
                               (* FINISHED TABLE STORAGE SPACE *)
     DPERSAM:REAL;                (* DISTANCE PER SAMPLE ON INPUT *)

(*********************************************************************
* RAWIN....THIS PROCEDURE READS IN THE DATA FILE CREATED BY CONVERT *
*     AND STORES IT IN RAWSTUFF ARRAY.                              *
*********************************************************************)
PROCEDURE RAWIN;

VAR
     I,II,III:INTEGER;          (* INDEXES *)
     EFLOP:BOOLEAN;             (* EVENT FLIP FLOP, USED TO INSURE
                                   EACH EVENT IS COUNTED ONLY ONCE *)

BEGIN
     EFLOP:=FALSE;                 (* NO ACTIVE EVENT *)
     READ(NUM);                    (* READ NUMBER OF SAMPLES *)
     ECNT:=0;                      (* BEGIN COUNTING EVENTS *)
     FOR I:=1 TO NUM DO
    BEGIN
     IF NOT EOF(INPUT) THEN READ(RAWSTUF(.I.));
      (* READ RAW DATA INTO RAWSTUFF ARRAY *)
     IF (RAWSTUF(.I.)>1000) AND ((RAWSTUF(.I.)-1000)>1000) THEN RAWSTUF(.I.):=
        RAWSTUF(.I-1.);
        (* IF INCOMMING DATA IS OUT OF RANGE AND IT IS NOT AN EVENT MARK,
                THEN IGNORE AND SET CURRENT SAMPLE EQUAL TO LAST SAMPLE *)
     IF RAWSTUF(.I.)>1000 THEN
         (* IF IT MAKES IT HERE IT MUST BE AN EVENT *)
```

```
            BEGIN
            (* PROCESS THE EVENT MARK *)
            RAWSTUF(.I.):=RAWSTUF(.I.)-1000;
            IF EFLOP=FALSE THEN
                 BEGIN
                 ECNT:=ECNT+1;
                 EVENT(.ECNT.):=I;
                 EFLOP:=TRUE;
                 END;
              END
      ELSE  EFLOP:=FALSE;
          (* DONE  IN THE EVENT OF NO EVENT.... *)
      RAWSTUF(.I.):=RAW.TUF(.I.)-500;
          (* SUBTRACT OUT DC OFFSET *)
      END;
END;


(*********************************************************************
* ANGLESCALE.....THIS PROCEDURE SCALES THE VALUES IN POSITION TO *
*        THE CORRESPONDING ANGULAR VALUE AS DETERMINED BY THE     *
*        DIGITIZING METHOD.                                       *
*********************************************************************)
PROCEDURE ANGLESCALE;

VAR  I,II,III:INTEGER;

BEGIN
      FOR I:=1 TO NUM DO
         BEGIN
         POSITION(.I,2.):=RAWSTUF(.I.)*(-0.005);
             (* EACH BIT OF QUANTIZATION CORRESPONDS TO 0.005 DEG. *)
         END;
END;


(***********************************************************************
*  DISTANCE....THIS PROCEDURE USES THE EVENT MARKES (COUNTED AND       *
*         LOCATED IN RAWIN PROCEDURE) TO DETERMINE THE DISTANCE VALUES *
*         FOR POSITION ARRAY                                           *
***********************************************************************)
PROCEDURE DISTANCE;

VAR
     I,II,III:INTEGER;
    AVGSMP:REAL;          (* AVERAGE NUM OF SAMPLES PER NM *)
    SUM,MAXPOS:REAL;       (* SUM USED IN AVG PROCESSING
                              MAXPOS.. POSITION OF FARTHEST SAMPLE *)

BEGIN
     SUM:=0;
     FOR I:=2 TO ECNT DO
          SUM:=EVENT(.I.)-EVENT(.I-1.)+SUM;
     AVGSMP:=SUM/(ECNT-1);          (* AVG SAMPLES PER NM FOUND *)
     DPERSAM:=6076.0/AVGSMP;        (* FEET PER NM FOUND *)
     MAXPOS:=EVENT(.ECNT.)*DPERSAM;
     FOR I:=1 TO NUM DO
           POSITION(.I,1.):=MAXPOS-(I-1)*DPERSAM;
              (* CALCULATE POSITIONS WORKING BACK FROM THE MAXIMUM POS.*)

END;


(********************************************************************
*  STRUCTUREOUT.... THIS PROCEDURE WRITES OUT A FILE OF THE        *
*        FIRST PASS RESULTS...                                      *
********************************************************************)
PROCEDURE STRUCTUREOUT;
```

```
VAR
    I,II:INTEGER;

BEGIN
     WRITELN('      ',NUM:7);
    FOR I:=1 TO NUM DO
    WRITELN('     ',POSITION(.I,1.):10:2,'      ',POSITION(.I,2.):10:3);
END;

(*********************************************************************
*  UNICOR.....THIS PROCEDURE CREATES A 'UNIFORM' TABLE BY WAY OF   *
        AN INTERPOLATION ROUTINE.   ALL TABLES HAVE 1000 ENTRIES   *
*       WHICH ARE 25 FEET (DX) APART.                               *
*********************************************************************)
PROCEDURE UNICOR;

VAR
     SL,XX,DXX,CURPOS:REAL;
          (* SL - SLOPE FOR STRAIGHT LINE INTERPOLATION APPROXIMATION *)
          (* XX - DISTANCE BETWEEN POINT TO BE INTERPOLATED AND THE
                  LOWEST KNOWN BRACKETING VALUES......*)
          (* DXX - INCRENTAL Y VALUE FROM INTERPOLATION *)
          (* CURPOS - INSTANTAENOUS POSITION TO BE INTERPOLATED FOR *)
     BACK,NOPOINT:BOOLEAN;
           (* BACK - FLAG FOR BACK COURSE *)
           (* NOPOINT - FLAG WHICH INDICATES THAT THE CURPOINT IS NOT
                        BETWEEN THE BRACKETING VALUES *)
     I,II,III:INTEGER;

BEGIN
     II:=1;         (* START COUNTING *)
     CURPOS:=25000.0;        (* START FROM FARTHEST POSITION *)
     FOR I:=1 TO 1000 DO
         BEGIN
         NOPOINT:=TRUE;
         WHILE NOPOINT DO
              BEGIN
              IF CURPOS > POSITION(.II,1.)
              THEN NOPOINT:=FALSE       (* CURPOS BETWEEN B   POINTS *)
              ELSE II:=II+1;      (* ELSE MOVE TO NEXT SET OF POINTS *)
              END;
         (* WHEN THIS POINT IS REACHED A PROPER POINT FOR INTERPOLATION
               HAS BEEN FOUND, SO BEGIN INTERPOLATING *)
         SL:=(POSITION(.II+1,2.)-POSITION(.II,2.))/DPERSAM;
         XX:=CURPOS - POSITION(.II,1.);
         DXX:=XX*SL;
         III:=1001-I;
         COR(.III.):=POSITION(.II,2.)+DXX;
         CURPOS:=25000.0-I*25.0;
         END;
END;

(*****************************************************************
*   CORSCALE....THIS IS THE FINAL SCALING PASS.  THE USER IS *
*        PROMPTED FOR THE LOCALIZER PATH WIDTH FROM THE       *
*        ACTUAL SITE.  THE TABLE IS NORMALIZED BY THIS        *
*        ANGLE AND THEN RESCALED BY THE WIDTH ASSUMED BY THE *
*        NASA SIMULATION (4 DEGREES)                          *
*****************************************************************)
PROCEDURE CORSCALE;

VAR
     I,II:INTEGER;
     OWIDTH:REAL;          (* ORIGINAL LOCALIZER WIDTH *)

BEGIN
        (* THESE STATMENTS REDIFINE THE FILES FOR USER INTERACTION *)
     CLOSE(INPUT);
     SYSTEM('FI SYSIN TERM',I);
```

```
    CLOSE(OUTPUT);
    SYSTEM('FI SYSPRINT TERM',I);
    REWRITE(OUTPUT);
    WRITELN(' ENTER LOCALIZER COURSE WIDTH FOR THIS INSTALLATION....');
    RESET(INPUT);
    READLN(OWIDTH);
    FOR I:=1 TO 1000 DO
    COR(.I.):=COR(.I.)/OWIDTH*4.0;
END;


(***********************************************************************
*  UNICOROUT.....CREATES A FILE ON DISK CONTAINING THE FINISHED       *
*       TABLE.                                                         *
***********************************************************************)
PROCEDURE UNICOROUT;

VAR
     I,II:INTEGER;

BEGIN
    (* DEFINE FIDEFS FOR DISK FILE OUTPUT *)
    CLOSE(OUTPUT);
     SYSTEM('FI SYSPRINT DISK UC CHART C',II);
     REWRITE(OUTPUT);
     FOR I:=1 TO 1000 DO
     WRITELN(COR(.I.):10:8);
END;

(*******************************************************************
*  SMOOTHEND.....THIS PROCEDURE 'SMOOTHES' THE END OF THE TABLE *
*       SO THAT THE FIRST SMPLE WILL NOT CAUSE A BIG JUMP ON     *
*       CDI NEEDLE.                                               *
*******************************************************************)
PROCEDURE SMOOTHEND;

VAR I:INTEGER;
     SLOPE:REAL;

BEGIN
     SLOPE:=(-COR(.975.))/625.0;
    FOR I:=976 TO 1000 DO
          (* DRAW A STRAIGHT LINE FROM THE 975TH SAMPLE TO ZERO *)
            BEGIN
            COR(.I.):=COR(.975.)+(I-975)*25*SLOPE;
            END;
END;


(* MAIN PROGRAM BODY......................*)


BEGIN
     RAWIN;
     ANGLESCALE;
     DISTANCE;
     STRUCTUREOUT;
     UNICOR;
     SMOOTHEND;
     CORSCALE;
     UNICOROUT;
END.
```

```
(*
*************************************************************************
*                                                                       *
*   GLIDE SLOPE VERSION................                                 *
*  THIS PROGRAM IS DESIGNED TO PROCESS COURSE STUCTURES FOR THE NASA    *
*  SIMULATION PROJECT.   THE PROGRAM IS DESIGNED TO READ FILES CREATED  *
*  BY THE 'CONVERT' PROGRAM.  THIS PROGRAM THEN TAKES THE VALUES AND    *
*  SCALES THEM TO DEGREES (SEE DOCUMENTATION ON DIGITIZING PROCESS)     *
*  THEN THIS PROGRAM USES THE EVENT MARKES IN THE CONVERTED FILE TO     *
*  DETERMINE THE DISTANCE OF EACH POINT FROM THE RUNWAY THRESHOLD.      *
*  (IN FEET).   NEXT THE PROGRAM CREATES A FILE OF 1000 POINTS WHICH    *
*  THROUGH AN INTERPOLATION PROCESS HAVE BEEN ADJUSTED TO BE 25 FEET    *
*  APART. THIS PROVIDES A COURSE STUCTURE WHICH IS JUST OVER 4 NM LONG. *
*  FINALLY THE VAULES IN THIS TABLE ARE AGAIN RESCALED BY: FIRST        *
*  NORMALIZING BY THE WIDTH OF THE GLIDE SLOPE PATH AT THE ACTUAL       *
*  INSTALLATION FROM WHICH THE DIGITIZED DATA WAS TAKEN.                *
*  THE USER IS NOT  PROMPTED TO ENTER THE GLIDS SLOPE BEAM              *
*  WIDTH FOR THIS SCALING OPERATION.                                    *
*                                                                       *
*************************************************************************)
PROGRAM SCALVERT(INPUT,OUTPUT);


CONST
     (* DX IS THE INCREMENTAL DISTANCE FOR THE INTERPOLATION *)
     DX=25.0;



VAR
     NUM:INTEGER;          (* NUMBER OF SAMPLES IN INPUT *)
     ECNT:INTEGER;         (* NUMBER OF EVENT MARKS *)
     POSITION:ARRAY(.1..5000,1..2.) OF REAL;
                           (* POSITION AND ANGLE STORAGE FOR FIRST PASS *)
     RAWSTUF:ARRAY(.1..5000.) OF REAL;
                             (* RAW, UNSCALED INPUT DATA FROM CONVERT *)
     I,II,III:INTEGER;        (* GLOBAL INDEXERS *)
     EVENT:ARRAY(.1..30.) OF INTEGER;
                             (* STORAGE FOR EVENT POSITIONS *)
     COR:ARRAY(.1..1000.) OF REAL;
                             (* FINISHED TABLE STORAGE SPACE *)
     DPERSAM:REAL;              (* DISTANCE PER SAMPLE ON INPUT *)

(*********************************************************************
* RAWIN....THIS PROCEDURE READS IN THE DATA FILE CREATED BY CONVERT *
*     AND STORES IT IN RAWSTUFF ARRAY.                              *
*********************************************************************)
PROCEDURE RAWIN;

VAR
     I,II,III:INTEGER;          (* INDEXES *)
     EFLOP:BOOLEAN;             (* EVENT FLIP FLOP, USED TO INSURE
                                   EACH EVENT IS COUNTED ONLY ONCE *)

BEGIN
     EFLOP:=FALSE;                (* NO ACTIVE EVENT *)
     READ(NUM);                   (* READ NUMBER OF SAMPLES *)
     ECNT:=1;                     (* BEGIN COUNTING EVENTS *)
     FOR I:=1 TO NUM DO
    BEGIN
     IF NOT EOF(INPUT) THEN READ(RAWSTUF(.I.));
      (* READ RAW DATA INTO RAWSTUFF ARRAY *)
     IF (RAWSTUF(.I.)>1000) AND ((RAWSTUF(.I.)-1000)>1000) THEN RAWSTUF(.I.):=
        RAWSTUF(.I-1.);
        (* IF INCOMMING DATA IS OUT OF RANGE AND IT IS NOT AN EVENT MARK,
              THEN IGNORE AND SET CURRENT SAMPLE EQUAL TO LAST SAMPLE *)
     IF RAWSTUF(.I.)>1000 THEN
        (* IF IT MAKES IT HERE IT MUST BE AN EVENT *)
           BEGIN
           (* PROCESS THE EVENT MARK *)
```

```
            RAWSTUF(.I.):=RAWSTUF(.I.)-1000;
            IF EFLOP=FALSE THEN
                BEGIN
                EVENT(.ECNT.):=I;
                ECNT:=ECNT+1;
                EFLOP:=TRUE;
                END;
              END
      ELSE  EFLOP:=FALSE;
          (* DONE  IN THE EVENT OF NO EVENT.... *)
      RAWSTUF(.I.):=RAWSTUF(.I.)-500;
           (* SUBTRACT OUT DC OFFSET *)
      END;
END;


(**********************************************************************
* ANGLESCALE.....THIS PROCEDURE SCALES THE VALUES IN POSITION TO *
*        THE CORRESPONDING ANGULAR VALUE AS DETERMINED BY THE     *
*        DIGITIZING METHOD.                                        *
**********************************************************************)
PROCEDURE ANGLESCALE;

VAR  I,II,III:INTEGER;

BEGIN
      FOR I:=1 TO NUM DO
         BEGIN
         POSITION(.I,2.):=RAWSTUF(.I.)*(-0.2);   (* 2 MICO AMP QUANTIZATION *)
         END;
END;


(*************************************************************************
*  DISTANCE....THIS PROCEDURE USES THE EVENT MARKES (COUNTED AND         *
*         LOCATED IN RAWIN PROCEDURE) TO DETERMINE THE DISTANCE VALUES  *
*         FOR POSITION ARRAY                                              *
*************************************************************************)
PROCEDURE DISTANCE;

VAR
     I,II,III:INTEGER;
    AVGSMP:REAL;          (* AVERAGE NUM OF SAMPLES PER NM *)
    SUM,MAXPOS:REAL;       (* SUM USED IN AVG PROCESSING
                              MAXPOS.. POSITION OF FARTHEST SAMPLE *)

BEGIN
     SUM:=0;
     FOR I:=2 TO ECNT-1 DO
           SUM:=EVENT(.I.)-EVENT(.I-1.)+SUM;
     AVGSMP:=SUM/(ECNT-2);           (* AVG SAMPLES PER NM FOUND *)
     DPERSAM:=6076.0/AVGSMP;         (* FEET PER NM FOUND *)
     MAXPOS:=EVENT(.ECNT-1.)*DPERSAM;
     FOR I:=1 TO NUM DO
            POSITION(.I,1.):=MAXPOS-(I-1)*DPERSAM;
                (* CALCULATE POSITIONS WORKING BACK FROM THE MAXIMUM POS.*)

END;


(********************************************************************
*  STRUCTUREOUT.... THIS PROCEDURE WRITES OUT A FILE OF THE         *
*        FIRST PASS RESULTS...                                       *
********************************************************************)
PROCEDURE STRUCTUREOUT;


VAR
     I,II:INTEGER;
```

```
BEGIN
     WRITELN('     ',NUM:7);
     FOR I:=1 TO NUM DO
     WRITELN('     ',POSITION(.I,1.):10:2,'      ',POSITION(.I,2.):10:3);
END;

(*******************************************************************
*  UNICOR.....THIS PROCEDURE CREATES A 'UNIFORM' TABLE BY WAY OF  *
*       AN INTERPOLATION ROUTINE.   ALL TABLES HAVE 1000 ENTRIES  *
*       WHICH ARE 25 FEET (DX) APART.                              *
*******************************************************************)
PROCEDURE UNICOR;

VAR
     SL,XX,DXX,CURPOS:REAL;
           (* SL - SLOPE FOR STRAIGHT LINE INTERPOLATION APPROXIMATION *)
           (* XX - DISTANCE BETWEEN POINT TO BE INTERPOLATED AND THE
                   LOWEST KNOWN BRACKETING VALUES......*)
           (* DXX - INCRENTAL Y VALUE FROM INTERPOLATION *)
           (* CURPOS - INSTANTAENOUS POSITION TO BE INTERPOLATED FOR *)
     BACK,NOPOINT:BOOLEAN;
            (* BACK - FLAG FOR BACK COURSE *)
            (* NOPOINT - FLAG WHICH INDICATES THAT THE CURPOINT IS NOT
                         BETWEEN THE BRACKETING VALUES *)
     I,II,III:INTEGER;

BEGIN
     II:=1;        (* START COUNTING *)
     CURPOS:=25000.0;       (* START FROM FARTHEST POSITION *)
     FOR I:=1 TO 1000 DO
         BEGIN
         NOPOINT:=TRUE;
         WHILE NOPOINT DO
              BEGIN
              IF CURPOS > POSITION(.II,1.)
              THEN NOPOINT:=FALSE       (* CURPOS BETWEEN B   POINTS *)
              ELSE II:=II+1;      (* ELSE MOVE TO NEXT SET OF POINTS *)
              END;
         (* WHEN THIS POINT IS REACHED A PROPER POINT FOR INTERPOLATION
                HAS BEEN FOUND, SO BEGIN INTERPOLATING *)
         SL:=(POSITION(.II+1,2.)-POSITION(.II,2.))/DPERSAM;
         XX:=CURPOS.- POSITION(.II,1.);
         DXX:=XX*SL;
         III:=1001-I;
         COR(.III.):=POSITION(.II,2.)+DXX;
         CURPOS:=25000.0-I*25.0;
         END;
END;

(*****************************************************************
*   CORSCALE....THIS IS THE FINAL SCALING PASS.  THE USER IS *
*        PROMPTED FOR THE GS BEAM   PATH WIDTH FROM THE      *
*        ACTUAL SITE.  THE TABLE IS NORMALIZED BY THIS       *
*        VALUE SO THAT IT CAN BE RESCALED BY                 *
*        NASA SIMULATION                                     *
*****************************************************************)
PROCEDURE CORSCALE;

VAR
     I,II:INTEGER;
     OWIDTH:REAL;         (* ORIGINAL LOCALIZER WIDTH *)

BEGIN
    OWIDTH:=150.0;
    FOR I:=1 TO 1000 DO
    COR(.I.):=COR(.I.)/OWIDTH*0.7;
      (*  IN EXISTING NASA SOFTWARE AN ANGULAR DEVIATION OF +-0.7
            DEGREES CAUSES A FULL SCALE DEFLECTION OF THE CDI   *)
END;
```

```
(*************************************************************************
*  UNICOROUT.....CREATES A FILE ON DISK CONTAINING THE FINISHED          *
*       TABLE.                                                           *
*************************************************************************)
PROCEDURE UNICOROUT;

VAR
     I,II:INTEGER;

BEGIN
    (* DEFINE FIDEFS FOR DISK FILE OUTPUT *)
    CLOSE(OUTPUT);
     SYSTEM('FI SYSPRINT DISK UC CHART C',II);
     REWRITE(OUTPUT);
     FOR I:=1 TO 1000 DO
     WRITELN(COR(.I.):10:8);
END;


(*******************************************************************
*  SMOOTHEND.....THIS PROCEDURE 'SMOOTHES' THE END OF THE TABLE *
*       SO THAT THE FIRST SMPLE WILL NOT CAUSE A BIG JUMP ON     *
*       CDI NEEDLE.                                               *
*******************************************************************)
PROCEDURE SMOOTHEND;

VAR I:INTEGER;
     SLOPE:REAL;

BEGIN
     SLOPE:=(-COR(.975.))/625.0;
    FOR I:=976 TO 1000 DO
         (* DRAW A STRAIGHT LINE FROM THE 975TH SAMPLE TO ZERO *)
           BEGIN
           COR(.I.):=COR(.975.)+(I-975)*25*SLOPE;
           END;
END;


(* MAIN PROGRAM BODY.......................*)


BEGIN
     RAWIN;
     ANGLESCALE;
     DISTANCE;
     STRUCTUREOUT;
     UNICOR;
     SMOOTHEND;
     CORSCALE;
     UNICOROUT;
END.
```

```
PROGRAM GENPATH(INPUT,OUTPUT);

CONST
   (*  THESE ARE THE CONSTANTS RELATIVE TO THE CAT I TOLLERANCE STUDY *)
    PI=3.14593;
    NM=6076.0;
    NM2=12152.0;
    NM3=18228.0;
    NM4=24304.0;
    NMH=3038.0;
    ILSPTA=24304.0;
    ILSPTB=3500.0;
    ILSPTC=1000.0;
    ZONE2A=30.0;
    ZONE2B=15.0;
    ZONE3B=15.0;
    ZONE3C=15.0;

TYPE
   LABL=ARRAY(.1..13.)OF CHAR;


(* THESE ARE THE GLOBAL VARIABLES TO BE USED..... *)
VAR
    DUMMY:INTEGER;


(*  THE FOLLOWING PROCEDURE DECLARATIONS ENABLE THIS PROGRAM TO LINK TO
    THE FORTRAN PLOTTING ROUTINES IN THE LIBRARY.  *)
PROCEDURE PLOTS(IBUF:REAL;NLOC,IDEV:INTEGER);FORTRAN;
PROCEDURE PLOT(X,Y:REAL;IPEN:INTEGER);FORTRAN;
PROCEDURE FACTOR(FCTR:REAL);FORTRAN;
PROCEDURE OFFSET(XOFF,XFCTR,UOFF,YFCTR:REAL);FORTRAN;
PROCEDURE WHERE(X,Y,FCTR:REAL);FORTRAN;
PROCEDURE SYMBOL(X,Y,HEIGHT,IBCD,ANGLE:REAL;NCHAR:INTEGER);FORTRAN;
PROCEDURE NUMBER(X,Y,HEIGHT,FNUM,ANGLE:REAL;NPLACE:INTEGER);FORTRAN;
PROCEDURE AXIS(X,Y:REAL;IBCD:LABL;NCHAR:INTEGER;SIZE,ANGLE,XMIN,DX,
                                DV:REAL);FORTRAN;
PROCEDURE LINE(XONE,YONE:REAL;NPTS,IRPT,IALTR,ISYM:INTEGER);FORTRAN;
PROCEDURE  SLANT(X,Y,HEIGHT:REAL;IBCD:INTEGER;ANGLE:REAL;NCHAR:INTEGER);FORTRAN;
PROCEDURE AX;FORTRAN;


(* THIS IS THE PROCEDURE THAT PLOTS THE AXIS AND TOLERANCE LIMITS *)
PROCEDURE PLOTTOL;

VAR
    I,II:INTEGER;
    X:REAL;

BEGIN
     DUMMY:=0;
     PLOTS(DUMMY,4,11);
     AX;
     PLOT(1.0,4.0,-3);
     PLOT(0.0,-2.0,2);
     PLOT(0.0,2.0,2);
     PLOT(0.0,0.0,3);
     PLOT(8.0,0.0,2);
     PLOT(ILSPTC/NMH,ZONE3C/37.5,3);
     PLOT(ILSPTB/NMH,ZONE3B/37.5,2);
     PLOT(ILSPTB/NMH,ZONE2B/37.5,3);
     PLOT(ILSPTA/NMH,ZONE2A/37.5,2);
     PLOT(ILSPTC/NMH,ZONE3C/(-37.5),3);
     PLOT(ILSPTB/NMH,ZONE3B/(-37.5),2);
     PLOT(ILSPTB/NMH,ZONE2B/(-37.5),3);
     PLOT(ILSPTA/NMH,ZONE2A/(-37.5),2);
END;

PROCEDURE CLOSEPLOT;
BEGIN
     PLOT(0.0,0.0,999);
END;
```

```
(* THE FOLLOWING PROCEDURES GENERATE THE COURSES FOR THE THREE GENERIC
   LOCALIZER COURSES.  EACH CASE PROCEDURE GENERATES A DIFFERENT COURSE
   STRUCTURE.                                                            *)


PROCEDURE CASEI;

VAR
    X,Y,YY,A:REAL;
    I:INTEGER;

BEGIN
    PLOT(0.0,0.0,3);
    FOR I:=1 TO 1000 DO
        BEGIN
        X:=(I-1)*25.0;
        IF X<1000.0 THEN Y:=-15.0*COS((X*PI)/1000.0)
        ELSE IF X<3500.0 THEN Y:= 15*COS(((X-1000.0)*PI)/(NM4-1000.0))
        ELSE Y:=(15+(X-3500.0)*15.0/(NM4-3500.0))*
                   COS(((X-1000.0)*1.5*PI)/(NM4-1000.0));
         YY:=2.0*Y/150.0;
         WRITELN(YY:10:8);
         PLOT(X/NMH,Y/37.5,2);
         END;
END;

PROCEDURE CASEII;

VAR
      X,Y,YY,A:REAL;
      I,II:INTEGER;

BEGIN
      PLOT(0.0,0.0,3);
      FOR I:=1 TO 1000 DO
           BEGIN
           X:=(I-1)*25.0;
           IF X < 1000.0 THEN A:=16.203
           ELSE IF X < 3500.0 THEN A:=15.0
           ELSE IF X < 20000.0 THEN A:=(15 + (X-3500.0)*15.0/20804.0)
           ELSE A:=(27.229-(X-20000.0)*0.005446);
           Y:=A*COS(X/2578.734);
           YY:=Y*2.0/150.0;
           WRITELN(YY:10:8);
           PLOT(X/NMH,Y/37.5,2);
           END;
END;


PROCEDURE CASEIII;

VAR
    X,Y,YY,A:REAL;
    I:INTEGER;


BEGIN
     PLOT(0.0,0.0,3);
     FOR I:=1 TO 1000 DO
         BEGIN
         X:=(I-1)*25.0;
         IF X<4500.0 THEN Y:=8.493557+8.493557*COS((X*PI)/4500.0)
         ELSE Y:=0.0;
           YY:=Y*2.0/150.0;
         WRITELN(YY:10:8);
         PLOT(X/NMH,Y/37.5,2);
         END;
END;

BEGIN (*  MAIN PROGRAM BODY *);
     PLOTTOL;
     CASEII;
     CLOSEPLOT;
END.
```

```
C   THIS PROGRAM PRODUCES A PLOT FILE FOR A FILE CREATED BY
C THE PROGRAM GSSCLV.
      CALL PLOTS(IBUF,4,11)
      CALL AX
       CALL PLOT(1.0,4.0,-3)
      READ(3,30)Y
      X=0
      CALL PLOT(X,Y,3)
10    FORMAT(5X,I7)
      DO 20 I=1,999
      READ(3,30)Y
      X=I*50.0/6076.0
      Y=Y*4.0
30    FORMAT(F12.8)
      CALL PLOT(X,Y,2)
 20   CONTINUE
      CALL PLOT(0.0,0.0,999)
      END
       SUBROUTINE AX
      CALL AXIS(1.0,2.0,'DEGREES',7,4.0,90.0,-0.5,0.25,10.0)
      CALL AXIS(1.0,2.0,'DISTANCE (NM)',-13,7.5,-0.0,-0.0,0.5,10.0)
       RETURN
       END
```

```
COPY &1 &2 &3 = = = (REPLACE RECFM F LRECL  80
FI 3 DISK &1 &2 &3
FI 11 DISK PL&1 &2 &3
EXEC LODE GSCORPLT
```

```
C   THIS PROGRAM PRODUCES A PLOT FILE FOR A FILE CREATED BY
C THE PROGRAM SCLVERT.  THIS PROGRAM REQIURES THE RECORD COUNT TO
C BE CONTAINED IN THE FIRST RECORD.
      CALL PLOTS(IBUF,4,11)
      CALL AX
       CALL PLOT(1.0,4.0,-3)
      READ(3,30)Y
      X=0
      CALL PLOT(X,Y,3)
10    FORMAT(5X,I7)
      DO 20 I=1,999
      READ(3,30)Y
      X=I*50.0/6076.0
      Y=Y*2.0
30    FORMAT(F12.8)
      CALL PLOT(X,Y,2)
 20   CONTINUE
      CALL PLOT(0.0,0.0,999)
      END
       SUBROUTINE AX
      CALL AXIS(1.0,2.0,'DEGREES',7,4.0,90.0,-1.0,0.5,10.0)
      CALL AXIS(1.0,2.0,'DISTANCE (NM)',-13,7.5,-0.0,-0.0,0.5,10.0)
       RETURN
       END
```

```
COPY &1 &2 &3 = = = (REPLACE RECFM F LRECL  80
FI 3 DISK &1 &2 &3
FI 11 DISK PL&1 &2 &3
EXEC LODE CORPLT
```

```
C *     THIS PROGRAM READS A GENERIC APPROACH PATH FRO DISK FILE 5 *
C *     AND USES THE TABLE LOOKUP METHOD TO ADD IN THE DIGITIZED   *
C *     LOCALIZER COURSE ERROR (WHICH HAS BEEN READ IN FROM DISK   *
C *     FILE 4.)   THE SUBROUTINE IRSOUT IS THEN USED TO PUT THE   *
C *     OUTPUT INTO A FORM COMPATIBLE WITH THE INTELLIGENT REMOTE  *
C *     SERIAL DEVICE.  VALUES OF -+2048 CORRESPOND TO FULL SCALE  *
C *     DEFLECTION OF THE LOCALIZER NEEDLE.  (WHICH IS 2 DEGREES   *
C *     ACCORDING TO THE SCALING OF THE ERROR DATA ).              *
        DIMENSION X(1000),ER(1000)
        DO 10 I=1,1000
        READ(5,20)X(I)
20      FORMAT(3(1X,F10.4))
        READ(4,30)ER(I)
30      FORMAT(F10.8)
10      CONTINUE
        XLOCER=0.0
        DO 100 I=1,1999
        IF (X(I).GT.25000.0) GOTO 50
        XLOCER=ER(INT(X(I)/25.))
        XLOCER=XLOCER*1024.0
C       WRITE(2,22)XLOCER
22      FORMAT('  ',F10.4)
50      CONTINUE
C       WRITE(2,23)LOCER
23      FORMAT('  ',I5)
        CALL IRSOUT(XLOCER,EFLAG)
100     CONTINUE
        STOP
        END
```

```
      COMMON DELT
      LOGICAL*4 BEGIN
      DELT=1.0
      IBUF=0
      BEGIN=.TRUE.
      X=0
      Y=0
      Z=0
      CALL PLOTS(IBUF,4,11)
      CALL USRPOS(1.0,X,Y,Z,BEGIN)
      CALL CARSC
      CALL RUNWAY
      XX=X/500.0
      YY=Y/500.0
      ZZ=Z/500.0
      CALL CARPL(XX,YY,ZZ,3)
      WRITE(6,15)X,Y,Z
15    FORMAT(3(1X,F10.4))
      BEGIN=.FALSE.
      DO 10 I=1,999
      T=I
      CALL USRPOS(T,X,Y,Z,BEGIN)
      X=X/500.0
      Y=Y/500.0
      Z=Z/500.0
      CALL CARPL(X,Y,Z,2)
      WRITE(6,15)X,Y,Z
10    CONTINUE
      END
C
C
C
C
C
C
C       CARTESIAN PLOT
C
C
C
C
      SUBROUTINE CARPL(X,Y,Z,IPEN)
      XX=Y-0.707167*X+4.15
      YY=Z-0.707167*X+4.15
      CALL PLOT(XX,YY,IPEN)
      RETURN
      END
C
C
C
C
C       DRAW CARTESIAN AXIS.
C
C
        SUBROUTINE CARSC
        CALL AXIS(4.15,4.15,'Y AXIS',6,5.85,0.0,0.0,1.0,10.0)
        CALL AXIS(4.15,4.15,'Z AXIS',6,5.85,90.0,0.0,1.0,10.0)
        CALL AXIS(4.15,4.15,'X AXIS',6,4.5,225.0,0.0,1.0,10.0)
        RETURN
        END
C
C
C
C
C
C              DRAW RUNWAY
C
C
C
      SUBROUTINE    RUNWAY
C
```

```
10     CALL CARPL(0.0,0.2,0.0,3)
       CALL CARPL(-5.0,0.2,0.0,2)
       CALL CARPL(0.0,-0.2,0.0,3)
20     CALL CARPL(-5.0,-0.2,0.0,2)
       CALL CARPL(0.0,-0.2,0.0,3)
       CALL CARPL(0.0,0.2,0.0,2)
       CALL CARPL(0.0,0.0,0.0,3)
30     CALL CARPL(-5.0,0.0,0.0,2)
       CALL CARPL(0.0,0.0,0.0,3)
       RETURN
       END
C
C
C
C
C
C
C       VECTOR INTEGRATION
C
C           THIS SUBROUTINE ACCEPTS AN ARRAY X(3,1000) AND USES
C      THE TRAPAZOID METHOD TO CALCULATE THE INTEGRAL OF THE FUNCTION
C      REPRESENTED IN THE ARRAY.  THE INITIAL CONDITIONS FOR THE
C      INTEGRATION ARE SUPPLIED THROUGH THE ARRAY ICON(3).
C      DELT IS A REAL VARIABLE SPECIFYING THE SIZE OF THE INTEGRATION
C      STEPS.  THE CALLING FORM  IS:
C
        SUBROUTINE VECINT(VECIN,VECOUT,ICON,DELT)
C
C         WHERE:
C                     VECIN - IS THE INPUT VECTOR (3X1000)
C                     VECOUT - IS THE OUPUT VECTOR (3X1000)
C                     ICON - IS THE INITIAL CONDITIONS (3)
C                     DELT - IS THE DIFFERENTIAL SIZE
C
C
C
C
          REAL*4 VECIN(3,1000)
          REAL*4 VECOUT(3,1000)
          REAL*4 ICON(3)
          REAL*4   SUM
          DO 10 J=1,3
                SUM=ICON(J)
                DO 20 I=1,999
                SUM=SUM+(VECIN(J,I)+VECIN(J,I+1))/2*DELT
                VECOUT(J,I)=SUM
 20             CONTINUE
               VECOUT(J,1000)=SUM+(VECIN(J,999)+VECIN(J,1000))/2*DELT
 10         CONTINUE
           RETURN
           END
C
C
C
C                    USER POSTITION CALULATOR
C
C
C
C      THIS ROUTINE WILL CALCULATE AN AIRCRAFTS FLIGHT PATH ACCORDING TO
C      THE USERS INITIAL SPECIFICATIONS AND FLIGHT INSTRUCTIONS.
C      THE ROUTINE THEN STORES (1000 SAMPLES OF) THE FLIGHT PATH IN AN
C      ARRAY.  THIS STEP REQUIRES AN INITIALIZING CALL TO THIS ROUTINE.
C      SUBSEQUENT CALLS TO THE ROUTINE RESULTS IN THE RETURN OF THE
C      POSITION AT THE TIME REQUESTED.  THE CALLING FORM IS:
C
        SUBROUTINE USRPOS(T,X,Y,Z,BEGIN)
C
C           WHERE:
C                T - IS THE TIME (REAL)  FOR WHICH THE POSITION IS TO
C                    BE RETURNED.
```

```
C              X - IS THE X CORRDINATE (REAL*4) AT TIME T.
C              Y - IS THE Y CORRDINATE (REAL*4) AT TIME T.
C              Z - IS THE Z CORRDINATE (REAL*4) AT TIME T.
C              BEGIN - IS A FLAG (LOGICAL*4) INDICATING WHETHER THIS
C                     IS AN INITIALIZING CALL OR NOT.  IF BEGIN=.TRUE.
C                     THE ROUTINE READS THE FLIGHT INSTRUCTIONS AND
C                     CALCULTES THE FIRST 1000 POINTS OF THE FLIGHT
C                     PATH.  IF BEGIN=.FALSE. THEN THE ROUTINE SIMPLY
C                     ATTEMPTS TO LOOK UP IN THE PREVIOUSLY CALCULATED
C                     POSITION ARRAY THE POSITION CORRESPONDING TO
C                     TIME T.
C
      COMMON DELT
      DATA PI/3.141593/,RPD/0.017453/,DPR/57.29578/
      LOGICAL*4 BEGIN
      REAL*4 ICON(3)
      REAL*4 VEL(3,1000)
      REAL*4 POS(3,1000)
      REAL*4 TIME(1000)
      REAL*4 TIMES(1000)
      REAL*4 HEAD(1000)
      REAL*4 ACCELF(1000)
      REAL*4 VELF(1000)
      REAL*4 ACCELZ(1000)
      REAL*4 VELZ(1000)
1     CONTINUE
      IF (.NOT.(BEGIN)) GOTO 1010
C     DO THIS SECTION IF THIS IS AN INITIALIZATION CALL
      COUNT=0
      PSI=0
      DO 12 I=1,3
12     ICON(I)=0.0
      TMAX=0.0
      READ(5,11)
C     SKIP HEADING CARD
11    FORMAT(10X)
      READ(5,20)TINIT,XINIT,YINIT,ZINIT,HEADIN,VELIN
C     READ ANOTHER HEADING CARD
       READ(5,11)
C     READ INITIAL VALUES
20    FORMAT(6(1X,F10.4))
100   CONTINUE
      READ(5,50,END=110)(TIMES(I),HEAD(I),VELF(I),VELZ(I),ACCELF(I),
     *ACCELZ(I),I=1,1000)
C      READ AND STORE FLIGHT INSTRUCTIONS
50       FORMAT(6(1X,F10.4))
110      CONTINUE
C        REMEMBER HOW MANY INSTRUCTIONS THERE WERE
         MCOUNT=I
C        CLEAR OUT ACCELERATION ARRAY
        DO 60 J=1,3
60      VEL(J,1000)=0.
C        CALCULATE INITIAL HEADING ANGLE
        PSI=HEADIN*RPD*DELT
C       LOAD INITIAL VELOCITIES.
        VEL(1,1000)=VELIN*COS(PSI)
        VEL(2,1000)=VELIN*SIN(PSI)
        VEL(3,1000)=0.0
C        LOAD INITIAL POSITIONS
        POS(1,1000)=XINIT
        POS(2,1000)=YINIT
        POS(3,1000)=ZINIT
C        INITIALIZE TIME
        TMAX=TINIT
1000  CONTINUE
C        CLEAR REMAINING FLIGHT COMMAND ARRAYS
      DO 1010 I=MCOUNT,1000
      TIMES(I)=0.0
      HEAD(I)=0.0
      VELF(I)=0.0
```

```
      VELZ(I)=0.0
      ACCELF(I)=0.0
      ACCELZ(I)=0.0
C
C    COME HERE IF THIS IS NOT AN INITIALIZATION CALL.
C
1010  CONTINUE
C       IF REQUSESTED TIME IS IN THE RANGE OF ALREADY CALCULATED VALUES
C       SKIP CALCUALTION PHASE
1100     IF (T.LT.TMAX) GOTO 2000
C        CALCULATE NEW TIME INTERVAL (1000) POINTS
         DO 1500 I=1,1000
                 TIME(I)=TMAX+(I-1)*DELT
C               TEST TO SEE IF STILL IN RANGE OF CURRENT FLIGHT INSTRCTION
                 IF (TIME(I).LT.TIMES(COUNT+1)) GOTO 1400
C                 CHECH IF THERE ARE ANY MORE INSTRUCTIONS.
                 IF (COUNT.GE.MCOUNT) GOTO 1400
C                     GET NEXT FLIGHT INSTRUCTION
                      COUNT=COUNT+1
C                 CALCULATE HEADING ANGLE
1400             PSI=HEAD(COUNT)*DELT*RPD+PSI
                 IF(FVEL.NE.VELF(COUNT)) FVEL=FVEL+ACCELF(COUNT)
                 IF(ZVEL.NE.VELZ(COUNT)) ZVEL=ZVEL+ACCELZ(COUNT)
                 IF(FVEL.GE.VELF(COUNT).AND.(FVEL-ABS(ACCELF(COUNT))).LE.
     *VELF(COUNT)) FVEL=VELF(COUNT)
                 IF(ZVEL.GE.VELZ(COUNT).AND.(ZVEL-ABS(ACCELZ(COUNT))).LE.
     *VELF(COUNT)) ZVEL=VELF(COUNT)
C                     LOAD VEL ARRAY
                 VEL(1,I)=VELF(COUNT)*COS(PSI)
                 VEL(2,I)=VELF(COUNT)*SIN(PSI)
                 VEL(3,I)=VELZ(COUNT)
1500      CONTINUE
C              INCREMENT TIME INTERVAL
          TMAX=TIME(1000)+DELT
C                   LOAD INITIAL POSITION CONDITIONS
               ICON(1)=POS(1,1000)
               ICON(2)=POS(2,1000)
               ICON(3)=POS(3,1000)
C              PERFORM VECTOR INTEGRATION ON VELOCITY TO OBTAIN POS
          CALL VECINT(VEL,POS,ICON,DELT)
2000      CONTINUE
C                COME HERE IF T IS IN PREVIOUSLY CALCULTED RANGE
C                OF POSITIONS
C                LOOK UP X, Y, AND Z IN THE POSITION ARRAY
          N=INT((T-(TMAX-1000.0*DELT))/DELT)
          IF (N.EQ.0) N=1
          X=POS(1,N)
          Y=POS(2,N)
          Z=POS(3,N)
         RETURN
         DEBUG UNIT(6),TRACE
         END
```

```
&CONTROL ALL
&IF .&1 EQ .? &GOTO -TELL
&CONTROL ALL
&X = 0
&LOOP 5 10
&X = &X + 1
FI 9 DISK UCC&X CHART C
FI 8 DISK UCGS&X CHART C
FI 12 TAP1
LOAD TAPETRAN (START
&X = 0
&Y = 0
&LOOP 8 3
&X = &X + 1
&LOOP 5 3
&Y = &Y + 1
FI 9 DISK GENLOC&X CHART C
FI 8 DISK GENGS&Y CHART C
FI 12 TAP1
LOAD TAPETRAN (START
&Y = 0
&EXIT
&BEGTYPE
     THIS EXEC WRITES THE TEN COURSES PREPARED FOR NASASIM TO MAG TAPE.
THE EXEC EXPECTS TO SEE ALL TEN COURSES WITH UCC PREFIX ON THE A DISK.
THE EXEC ALSO EXPECTS TO SEE THREE GENERIC PATH
IT WRITES THESE TO TAPE IN ALL POSSIBLE COMBINATIONS

&END
```

This FORTRAN program writes the Localizer and Glide Slope error information to tape.

```
      DO 50 J=1,1000
      READ(9,20)LOC
      READ(8,20)GLS
      WRITE(12,30)LOC
      WRITE(12,30)GLS
20    FORMAT(F10.8)
30    FORMAT(F10.8)
50     CONTINUE
      STOP
      END
```

1. Report No. NASA CR-172333	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Realistic Localizer Courses for Aircraft Instrument Landing Simulators		5. Report Date May 1984
		6. Performing Organization Code
7. Author(s) Timothy A. Murphy		8. Performing Organization Report No. OU/AEC/EER 66-1
		10. Work Unit No.
9. Performing Organization Name and Address Avionics Engineering Center Dept. of Electrical and Computer Engineering Ohio University Athens, Ohio 45701		11. Contract or Grant No. NAS1-17368
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546		13. Type of Report and Period Covered Contractor Report
		14. Sponsoring Agency Code 505-35-13-10

15. Supplementary Notes

Langley Technical Monitor: James J. Adams
Final Report

16. Abstract

Work done to provide realistic ILS course structures for use in aircraft simulators is described. Software developed for data conversion and translation of ILS course structure measurements is documented together with calcomp plots of the courses provided. A method of implementing the ILS course structure data in existing aircraft simulators at NASA Langley Research Center is described. A cockpit display used in the lab to review the digitized ILS course structures is also given.

17. Key Words (Suggested by Author(s)) glide slope, localizer, course deviation indicator (CDI), realistic ILS course structure aircraft simulator, (realistic) ILS approaches in simulation	18. Distribution Statement Unclassified - Unlimited Subject Category 05

19. Security Classif. (of this report) unclassified	20. Security Classif. (of this page) unclassified	21. No. of Pages 77	22. Price A05